A FRAMEWORK TO SUPPORT DEVELOPERS IN THE INTEGRATION AND APPLICATION OF LINKED AND OPEN DATA

by

TIMM HEUSS

A thesis submitted to Plymouth University
in partial fulfilment of the degree of

DOCTOR OF PHILOSOPHY

Faculty of Science and Engineering

April 2016

Abstract

Timm Heuss: A Framework to Support Developers in the Integration and Application of Linked and Open Data

In the last years, the number of freely available Linked and Open Data datasets has multiplied into tens of thousands. The numbers of applications taking advantage of these, however, have not. Thus, large portions of potentially valuable data remain unexploited and are inaccessible for lay users. Therefore the upfront investment in releasing data in the first place is hard to justify. The lack of applications needs to be addressed in order not to undermine efforts put into Linked and Open Data.

In existing research, strong indicators can be found that the dearth of applications is due to a lack of pragmatic, working architectures supporting these applications and guiding developers.

In this thesis, a new architecture for the integration and application of Linked and Open Data is presented. Fundamental design decisions are backed up by two studies: firstly, based on real-world Linked and Open Data samples, characteristic properties are identified. A key finding is the fact that large amounts of structured data display tabular structures, do not use clear licensing and involve multiple different file formats. Secondly, following on from that study, a comparison of storage choices in relevant query scenarios is made. It includes the de-facto standard storage choice in this domain, triples stores, as well as relational and NoSQL approaches. Results show significant performance deficiencies of some technologies in certain scenarios. Consequently, when integrating Linked and Open Data in scenarios with application-specific entities, the first choice of storage is relational databases.

Combining these findings and related best practices of existing research, a prototype framework is implemented using Java 8 and Hibernate. As a proof-of-concept it is employed in an existing Linked and Open Data integration project. Thereby, it is shown that a best practice architectural component is introduced successfully, while development effort to implement specific program code can be simplified.

Thus, the present work provides an important foundation for the development of semantic applications based on Linked and Open Data and potentially leads to a broader adoption of such applications.

Table of Contents

List of Figures

List of Tables

List of Listings

Acknowledgements

From the early beginnings, to the very end of my studies, I always loved being a researcher. The following persons contributed to this.

Without Bettina Harriehausen-Mühlbauer, I would not have even considered being a Ph.D. student at all. What I would have missed! Her support gave me the confidence and courage to reach for new goals and challenges. Thanks for the many hours of supervision, hints and recommendations, and also, the chatting.

Bernhard Humm took a significant part in making my thesis useful in practice. We were both working in an exciting research project, through which I learned a lot - last but not least in regard to my research topic. Now, it is the perfect case study for my work in this thesis. Thank you very much for the years of success, technology-hands-on and fun!

Thanks to Shirley Atkinson for the valuable exchanges.

Janina Fengel helped me significantly in directing the outcomes of my flaming passion into something that conforms to the standards of the academic world. Thank you very much for constantly providing one more improvement, a final hint, yet another best practice or must-read reference.

Thanks to Paul Walsh and Kellyn Rein for their time and commitment to produce their incredible proof-readings. I am particularly grateful to Brendan Lawlor, who put tremendous efforts, dedication and patience in the correction of this work.

Thanks to Werner Helm and Adrian Schmitt from the Competence Center Stochastics & Operations Research of the University of Applied Sciences Darmstadt for their essential hints in finding, developing, and interpreting the mathematical and statistical foundation of the field study.

Thanks to Uta Störl for the exchange during the preparation of the benchmarking work. It was very valuable input for me and helped avoiding pitfalls in this tricky field.

Thanks to Uwe Reh and Thomas Striffler from the Library and Information System of Hesse for the interesting exchange and for providing real-world data dumps for my benchmark.

Thanks to Ralf Dotzert for reviewing the benchmark scenario and sharing his ideas from his practical perspectives.

Thanks to Maria Lusky, who kindly reviewed the SPARQL queries used in the database benchmark.

I also want to thank Theresa Schaub and Marco Hamm for assisting me working in the field of the present work.

You have all made this time the unique, exciting and rewarding experience it was to me. Thank you all.

Author's Declaration

At no time during the registration for the degree of Doctor of Philosophy has the author been registered for any other University award without prior agreement of the Graduate Committee.

Work submitted for this research degree at the Plymouth University has not formed part of any other degree either at Plymouth University or at another establishment

Relevant scientific seminars and conferences were regularly attended at which work was often presented; external institutions were visited for consultation purposes and several papers prepared for publication.

Publications (or presentation of other forms of creative and performing work):

Timm Heuss, Janina Fengel, Bernhard Humm, Bettina Harriehausen-Mühlbauer, and Shirley Atkinson. A Field Study on Linked and Open Data at Datahub.io. In Sergej Alekseev, Paul Dowland, Bogdan V. Ghita, and Oliver Schneider, editors, *Eleventh International Network Conference, INC 2016, Frankfurt, Germany, July 19-21, 2016. Proceedings*, pages 79–84. Plymouth University, 2016.

Thomas Hoppe, Bernhard Humm, Ulrich Schade, Timm Heuss, Matthias Hemmje, Tobias Vogel, and Benjamin Gernhardt. Corporate Semantic Web – Applications, Technology, Methodology. *Informatik-Spektrum*, pages 1–7, Nov. 2015.

Tilman Deuschel, Timm Heuss, and Bernhard Humm. Die Medienplattform: Ein System für gerichtete und ungerichtete semantische Suchen *(The Mediaplatform: system for directed and undirected searches). Information - Wissenschaft & Praxis*, 35:201–206, August 2015.

Timm Heuss, Bernhard Humm, Tilman Deuschel, Torsten Fröhlich, Thomas Herth, and Oliver Mitesser. *Linked Data and User Interaction*, chapter Semantically Guided, Situation-Aware Literature Research In Frank H. Cervone and Lars G. Svensson, editors, pages 66–84. DE GRUYTER SAUR, 2015.

Bernhard Humm and Timm Heuss. Schlendern durch digitale Museen und Bibliotheken (*Strolling through digital museums and libraries*). In B. Ege, Bernhard Humm, and A. Reibold, editors, *Corporate Semantic Web*, X.media.press, pages 59–70. Springer Berlin Heidelberg, 2015.

Timm Heuss, Bernhard Humm, Christian Henninger, and Thomas Rippl. A Comparison of NER Tools W.R.T. A Domain-specific Vocabulary. In *Proceedings of the 10th International Conference on Semantic Systems*, SEM '14, pages 100–107, New York, NY, USA, 2014. ACM.

Tilman Deuschel, Timm Heuss, and Christian Broomfield. The Design Scope of Adaptive Storytelling in Virtual Museums. In R. Klein and P. Santos, editors, *Euro-*

graphics Workshop on Graphics and Cultural Heritage, pages 97–106, Darmstadt, Germany, 2014. Eurographics Association.

Tilman Deuschel, Timm Heuss, and Bernhard Humm. The digital online museum: A new approach to experience virtual heritage. In T. Risse, L. Predoiu, A. Nürnberger, and S. Ross, editors, *Proceedings of the 4th International Workshop on Semantic Digital Archives (SDA 2014) co-located with the International Digital Libraries Conference (DL 2014), London, UK, September 12, 2014.*, volume 1306 of *CEUR Workshop Proceedings*, pages 38–48. CEUR-WS.org, 2014.

Tilman Deuschel, Timm Heuss, Bernhard Humm, and Torsten Fröhlich. Finding without searching: A serendipity-based approach for digital cultural heritage. In *To Appear: Proceedings of the Digital Intelligence (DI) conference, Nantes, Frankreich, Sept. 17-19, 2014*, 2014.

Timm Heuss. Faust.rdf - Taking RDF literally. In *2nd Workshop on Linked Data in Linguistics (LDL-2013): Representing and linking lexicons, terminologies and other language data. Collocated with the Conference on Generative Approaches to the Lexicon*, Pisa, Italy, September 2013.

Timm Heuss. Lessons learned (and questions raised) from an interdisciplinary Machine Translation approach. In *W3C Workshop on the Open Data on the Web, 23 - 24 April 2013, Google Campus, Shoreditch, London, United Kingdom*, 2013.

Bettina Harriehausen-Mühlbauer and Timm Heuss. Semantic Web Based Machine Translation. In *Proceedings of the Joint Workshop on Exploiting Synergies Between Information Retrieval and Machine Translation (ESIRMT) and Hybrid Approaches to Machine Translation (HyTra)*, EACL 2012, pages 1–9, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

Presentation and Conference Talks:

Corporate Semantic Web Workshop 2015, Presentation: *Die Suche nach dem Linked and Open Data Warehouse (The search for a Linked and Open Data Warehouse)*, June 06, 2015, Dagstuhl, Germany

Darmstädter Symposium für Informationswissenschaften *(Symposium for Information Science Darmstadt)*, University of Applied Sciences Darmstadt, Germany

Ph.D. seminar, Talk: *Supporting the Application and Integration of Linked and Open Data from a Developer's Viewpoint - Deliverable 1* May 12, 2015, University of Applied Science, Darmstadt, Germany

Science Slam 2014, Tag der Forschung, (co-winning) Talk: *Wo sind die Apps? (Where are the Apps?)*, December 1, 2014, University of Applied Sciences, Darmstadt, Germany

10th International Conference on Semantic Systems
Presentation: *Comparison of NER Tools W.R.T. a Domain-Specific Vocabulary*
Talk: *Horizon 2020 Linked Data Expert Statement*,
September 4-5, 2014, Leipzig, Germany

4th International Workshop on Semantic Digital Archives (SDA 2014), Presentation: *The Digital Online Museum* collocated with the International Digital Libraries Conference (DL 2014), September 12, 2014, London, UK

Corporate Semantic Web Workshop 2014, Presentation: *Schlendern durch digitale Museen und Bibliotheken* (Strolling through digital museums and libraries), June 25-27, 2014, Dagstuhl, Germany

8th Collaborative Research Symposium on Security, E-learning, Internet and Networking (SEIN) Topic: *Quo vadis NLP + Web*, November 15, 2013, University of Applied Sciences Darmstadt, Germany

2nd Workshop on Linked Data in Linguistics (LDL), Presentation: *Faust.rdf - Taking RDF Literally*, collocated with the Conference on Generative Approaches to the Lexicon, September 23, 2013, Pisa, Italy

Guest lecture in Natural Language Processing (winter semester 2013/2014), Presentation: *Semantic Web-based Machine Translation*, December 17, 2013, University of Applied Sciences Darmstadt, Germany

W3C Workshop on the Open Data on the Web, Presentation: *Lessons learned (and questions raised) from an interdisciplinary Machine Translation approach* April 23-24, 2013, London, United Kingdom

Guest lecture in Natural Language Processing (winter semester 2012/2013), Presentation: *Semantic Web-based Machine Translation*, December 12, 2012, University of Applied Sciences, Darmstadt, Germany

Joint Workshop on Exploiting Synergies Between Information Retrieval and Machine Translation (ESIRMT) and Hybrid Approaches to Machine Translation (HyTra), Presentation (Screencast): *Semantic Web-based Machine Translation*, collocated with the 13th Conference of the European Chapter of the Association for computational Linguistics, University of Le Mans, France

Word count of main body of thesis: 40,108

Signed: _____

Timm Heuss

Date: November 29, 2016

Abbreviations

ACID Atomicity, Consistency, Isolation, Durability

AJAX Asynchronous JavaScript and XML

API Application Programming Interface

BBC British Broadcasting Corporation

BSBM Berlin SPARQL Query Language (SPARQL) Benchmark

CEP Complex Event Processing

CERN European Organization for Nuclear Research

CFL Comprehensive, Functional, Layered Architecture

CH Cultural Heritage

CI Continuous Integration

CKAN Comprehensive Knowledge Archive Network

CMS Content Management System

CRUD Create, Read, Update, Delete

CSS Cascading Style Sheet

CSV Comma-separated values

DBMS Database Management System

DBPSB DBpedia SPARQL Benchmark

DSL Domain Specific Language

ETL Extraction, Transform, Load

GLAM Gallery, Library, Archive, and Museum

GND Gemeinsame Normdatei *(Integrated Authority File)*

GTFS General Transit Feed Specification

GUI Graphical User Interface

GZIP Gnu ZIP

HeBIS Hessisches Bibliotheks- und Informationssystem *(Library and Information System of Hesse)*

HTML Hypertext Markup Language

HTTP Hypertext Transfer Protocol

IETF Internet Engineering Task Force

IMS Identity Mapping Service

IR Information Retrieval

IRS Identity Resolution Service

JDBC Java Database Connectivity

JSON JavaScript Object Notation

JVM Java Virtual Machine

KDD Knowledge Discovery in Databases

LarKC Large Knowledge Collider

LDBC SPB Linked Data Benchmark Council Semantic Publishing Benchmark

LLOD Linguistic Linked and Open Data

LMF Linked Media Framework

LoC Lines of Code

LUBM Lehigh University Benchmark

MARC Machine-Readable Catalog

MODS Metadata Object Description Schema

NER Named Entity Recognition

NGO Non Governmental Organisation

NLP Natural Language Processing

NoSQL Not Only SQL

OAI-PMH Open Archives Initiative Protocol for Metadata Harvesting

ODBC Open Database Connectivity

OKFN Open Knowledge Foundation

OLAP Online Analytical Processing

OWL Web Ontology Language

PCDHN Pan-Canadian Documentary Heritage Network

PDF Portable Document Format

PHP PHP Hypertext Preprocessor

RDF Resource Description Framework

RDFS Resource Description Framework (RDF) Schema

REST Representational State Transfer

RSS Really Simple Syndication

SDF Simple Data Format

SP^{2}Bench SPARQL Performance Benchmark

SPARQL SPARQL Query Language

SQL Structured Query Language

SRU Search/Retrieval via URL

SVN Subversion

TSV Tabular-separated Values

ULB University and State Library Darmstadt

URI Uniform Resource Identifier

URL Uniform Resource Locator

VIAF Virtual International Authority File

W3C World Wide Web Consortium

WHATWG Web Hypertext Application Technology Working Group

Web World Wide Web

WSDL Web Services Description Language

XML Extensible Markup Language

Chapter 1

Introduction

This is a golden era of information. In a matter of seconds, data about arbitrary topics can be accessed through computers, smartphones, tablets, and even televisions and refrigerators. This revolution would not have been possible without the invention of Tim Berners-Lee: the World Wide Web. Celebrating its 25^{th} anniversary in 2014, it is today one of the most important applications of the Internet. It lowered technical barriers, so that now everybody can access and contribute, and disseminated human-readable data in an open technology stack.

Inspired by the Web, but so far not realised, efforts have been invested in the last decade in a second data revolution: the endeavour of letting machines benefit from Web-like data structures. The aim of these efforts is to enable applications to retrieve and exchange information about arbitrary topics automatically and to provide meaningful contexts. In the last years, strategic prerequisites for this new era have been fulfilled with the Linked and Open Data movements. However the over-due, broad-scale success - a multiplicity of new, semantic applications - is still pending.

In the following discussion, the topics of the present work, namely the World Wide Web, Semantic Web, Linked Data, and Open Data, are introduced, along with corresponding basic terminology. Moreover, the Mediaplatform research project, set within these domains, is introduced.

Based on that, current and pressing issues in these domains are identified. It is pointed out why solving them is of importance for the domain of Linked and Open Data, for corresponding communities, end users as well as investors.

The chapter concludes with an outline for the structure of the following thesis.

1.1 Topics and Motivation

This section introduces topics on which this research is focused. They are typically associated with the term *Web Science* and are in the realm of the standardisation efforts of the World Wide Web Consortium (W3C).

1.1.1 World Wide Web

The *World Wide Web (WWW, Web)* was invented in 1990 by Tim Berners-Lee to address practical, ordinary issues of his working environment in the European Organization for Nuclear Research (CERN) laboratory. Because it was a "technological melting pot" (Berners-Lee and Fischetti, 1999, p. 19), sharing and exchanging research results across different operating systems and data formats was a major problem within the organisation.

Berners-Lee considered the diversity of computer systems and networks as a potential resource - not as a problem - and had the goal to break out of local barriers to create a cross-computer, free and "living world of hypertext" (Berners-Lee and Fischetti, 1999, p. 27). The early Web was designed with fundamental properties such as world-wide referencing system and incorporated complete decentralisation (Berners-Lee and Fischetti, 1999, p. 15f). It originally consisted of three basic technologies, allowing the Web to be as simple, as decentralised and as interoperable as possible (Berners-Lee and Fischetti, 1999, p. 36f): Resources are identified using Uniform Resource Identifiers (URIs), data is transferred using Hypertext Transfer Protocol (HTTP), and documents are written and annotated with Hypertext Markup Language (HTML).

Rapidly put to use, this simple technological foundation allowed the Web to prosper and supported the emergence of several principles.

One of the basic principles of the Web is the *networking effect* (Breslin et al., 2009, p. 20), sometimes also referred as *Metcalfe's law* (Breslin et al., 2009, p. 20), (Jacobs and Walsh, 2004). It states that the value of a network is proportional to the number of nodes

(Breslin et al., 2009, p. 20). Single nodes can be evaluated by the number and value of resources that link to it (Jacobs and Walsh, 2004). Consequently, the act of linking resources becomes more attractive the more links resources already have (Allemang and Hendler, 2011, p. 7). Thus, prior to reaching a critical mass of contributions, publishing in the Web is difficult to justify. After reaching that point, however, further contributions are accepted because they are requested by an adequately-sized user base.

Another principal idea of the Web is *federation* or *decentrality*; the fact that "there [is] no central computer 'controlling' the Web, no single network on which these protocols worked, not even an organisation anywhere that '[runs]' the Web." (Berners-Lee and Fischetti, 1999, p. 36). This way, the Web allowed everyone to contribute at any time, "without asking for access from anyone else" - but this also was considered to be "good Internet-style engineering", as there was no single point of failure (Berners-Lee and Fischetti, 1999, p. 16).

In the middle of the 2000s, Web development efforts tended to open up to broader, non-technical communities. This observation was called *Social Web* or *Web 2.0* (O'Reilly, 2005).

The Social Web represented "new structure and abstractions [...] on top of the ordinary Web" (Breslin et al., 2009, p. 11), and thus shared the same basic technology stack. Resulting applications, however, like blogs or wikis, lowered technical barriers, so that activities like "collaboration and sharing between users" became common (Breslin et al., 2009, p. 12).

Since 1990, the Web was the subject of constant development and additions, driven by community as well as business needs. Today, with HTML, Cascading Style Sheet (CSS), JavaScript and several developer-friendly JavaScript-frameworks, a powerful ecosystem has matured that reaches a broad range of end-user devices.

1.1.2 Semantic Web

Originally, the *Semantic Web* was designed to be a "a Web of data that can be processed directly or indirectly by machines." (Berners-Lee and Fischetti, 1999, p. 177). Basic concepts and best practises of the Web are reused, such as URIs, the decentralisation which they enable, and Extensible Markup Language (XML) to serialise arbitrary information (Berners-Lee et al., 2001).

Besides reuse of existing technology, the Semantic Web also introduced the Resource Description Framework (RDF), a new syntax that encodes meaning in sets of triples, each being "rather like the subject, verb and object of an elementary sentence" (Berners-Lee et al., 2001). RDF statements assert that "particular things (people, Web pages or whatever) have properties (such as 'is a sister of,' 'is the author of') with certain values (another person, another Web page)" (Berners-Lee et al., 2001).

A commonly envisioned use case of the Semantic Web was the semantic markup of existing Web pages (Hendler, 2001, p. 32), or intelligent agents that can automatically negotiate appointments and tasks (Berners-Lee et al., 2001).

SPARQL Query Language (SPARQL) is the standard query language for accessing RDF (Allemang and Hendler, 2011, p. 61). It can not only be used for retrieval, but also to federate queries within remote endpoints, to merge data from different sources, and to transform the structure of RDF (Allemang and Hendler, 2011, p. 110, 112).

Another principal idea for the Web, federation, was reconsidered for the Semantic Web - resulting in the *Open World Assumption*: because "at any time [...] new information could come to light" (Allemang and Hendler, 2011, p. 10), there is no assured and defined point in time where data can considered to be complete and even basic operations such as counting are "difficult" (Allemang and Hendler, 2011, p. 252).

Just like in the Web, the networking effect was expected to take effect in the Semantic Web, too. But unfortunately, the Semantic Web never really archieved a critical mass of providers and consumers. Various sources describe this as the *chicken-egg-problem*.

The chicken-egg-problem was the subject of the early works on the Web (Berners-Lee and Fischetti, 1999, p. 30), as well as on the Semantic Web (Hendler, 2001), (Breslin et al., 2009, p. 71), (Summers and Salo, 2013). Ever since then this problem is used to explain why the Semantic Web has not yet started off properly: "it is difficult to produce data without interesting applications, and vice versa" (Breslin et al., 2009, p. 71).

Today, the term Semantic Web has almost entirely vanished in research and is considered to be outdated (Edelstein et al., 2013, p. 14).

The idea of semantically annotating Web pages has been reapplied in *schema.org*, an initiative of the common search engine providers Bing, Google, Yandex and Yahoo (*Microdata (HTML)*, 2015). In contrast to RDF, schema.org is centralised and only offers a pre-defined vocabulary. Besides W3C Recommendations, it makes use of the Web Hypertext Application Technology Working Group (WHATWG) Microdata standard (*Microdata (HTML)*, 2015).

1.1.3 Open Data

Open Data is data that is available under open licences, such as Creative Commons or GNU FDL (known from Wikipedia) (Breslin et al., 2009, p. 67f). The term is promoted by organisations such as the Open Knowledge Foundation (OKFN) and has political aspects, as demonstrated by the Open Data Barometer (Open Data Research Project, 2013) or the Open Data Index (Open Knowledge Foundation, 2015).

A number of *data portals* have emerged to make Open Data available and easily accessible. A lot of Open Data is provided by governmentally-funded portals such as data.gov (USA) and data.gov.uk (UK), data.gov.ie (Ireland) or GovData.de (Germany). In addition, there are cross-government, cross-domain data portals, a commonly known example of which is datahub.io. There are also domain-specific portals, such as Archives-Hub.ac.uk (data from archives), LibHub.org (data from libraries), opendata.cern.ch (data from the CERN), or even city-specific sites like data.london.gov.uk. In November 2015, a meta portal for such portals, europeandataportal.eu, was launched by the

European Union (Krempl, 2015).

Commonly, such data portals are driven by the *Comprehensive Knowledge Archive Network (CKAN)*, a software system comparable with a Content Management System (CMS) for data. It allows data providers to register datasets in several formats and licenses, and offers user-friendly capabilities for developers and data scientists to explore and search for data.

1.1.4 Linked Data

The term *Linked Data*, also known as the *Web of Data*, dates back to a publication of Tim Berners-Lee in 2006 (Berners-Lee, 2006), which "distilled the essence of the Semantic Web project into a set of simple rules for Web publishers to use" (Summers and Salo, 2013, p. 5). Linked Data, and the associated community effort *Linking Open Data* started in mid-2007, aims at promoting use and interlinkage of RDF for arbitrary data (Breslin et al., 2009, p. 67f). Nowadays, Linked Data is considered to be current terminology of what previously has been called Semantic Web (Edelstein et al., 2013, p. 14).

According to the popular maturity model of Berners-Lee, Linked Data is Open Data in RDF that ideally links to "other people's data" (Berners-Lee, 2006), though Linked Data is often also referred to as *Linked Open Data (LOD)*, emphasising the fact that is open as well as linked. The worldwide, interconnected construct of Linked Data is called the *LOD Cloud* or the *Giant Global Graph* (Breslin et al., 2009, p. 67f), partly shown in Figure 1.1 on the facing page.

Every bubble in Figure 1.1 on the next page represents a *dataset*, each consisting of hundreds, thousands or even millions of facts. The diagram is generated automatically by using the Application Programming Interface (API) from the popular datahub.io data portal (Schmachtenberg et al., 2014).

Over time, more and more datasets began linking to existing entities of commonly known datasets, such as DBpedia, GeoNames and Freebase (Digital Meets Culture,

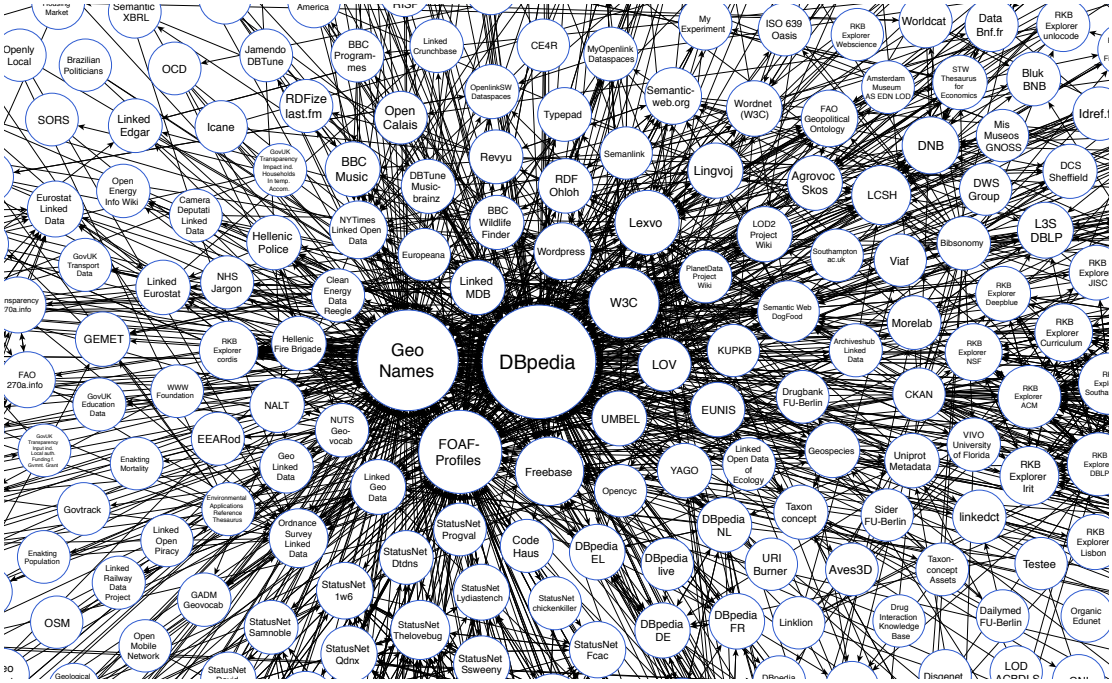


Figure 1.1: Excerpt of the LOD Cloud (Schmachtenberg et al., 2014).

2014). DBpedia in particular is often referenced, which makes it the *nucleus* of the LOD Cloud (Schandl et al., 2012). It is built from extracted data from Wikipedia pages, especially "semi-structured information from the wiki data structure and from info-boxes found on many pages" (Schandl et al., 2012, p. 536). Thus, DBpedia is a general dataset and can be used as reference value vocabulary (Isaac et al., 2005). As an encyclopaedia, DBpedia covers a "broad range of topics" and "provides identifiers for nearly every concept one can imagine" (Schandl et al., 2012, p. 536).

Besides cross-domain datasets like DBpedia, the Linked Open Data Cloud consists of domain-specific datasets, for example for life sciences, social networking, geographic information, government data, and linguistic data (Schmachtenberg et al., 2014). Such domain-specific excerpts are sometimes called *Micro Linked Open Data Clouds* (Blumauer, 2013). Popular samples are Bio2RDF in the life sciences domain or the Linguistic Linked and Open Data (LLOD).

1.1.5 Term Usage in the present Work

Originally, the term Linked Data had a technical context, because it refers to specific technologies and formats, while Open Data is settled in a political domain and affects a broader, less technical range of data. Such a clear and strict disambiguation can, in fact, not be observed in practice (Edelstein et al., 2013, p. 2).

Within this present work, the compound *LOD* will be used as an umbrella term for data that is, in general, published on data portals. This involves both, RDF-based Linked Data and openly-licensed Open Data, as well as data which is neither linked nor open[1].

1.1.6 Research Project Mediaplatform

The state-funded research project *Mediaplatform* was initiated to research new and enhanced ways of searching, displaying and teaching the online collections of galleries, libraries, archives and museums (GLAMs). Domain partners in this project are the Städel Museum and the University and State Library Darmstadt (ULB).

To match the individual needs of the partners, two dedicated Graphical User Interfaces (GUIs) (shown in Figures 1.2 and 1.3 on the next page) have been developed with a common server architecture to support different usage patterns by combining various data sources and their semantics.

On the library side, the primary goal is to support the user in finding the desired literature as fast as possible, in a process that rapidly reduces the number of search results in as few steps as possible. In contrast to this, the museum side aims at providing a continuous and lasting experience, without emphasising a single, final result - a process called digital strolling.

The Mediaplatform represents novel user interfaces for rich media databases as found in GLAMs. The user can query and browse through results and can get inspired by the system's recommendations. Based on modern technologies such as HTML5, CSS3 and

[1] A field study is conducted to clarify this, by assessing actual data of a data portal (Section 4.1 on page 122).

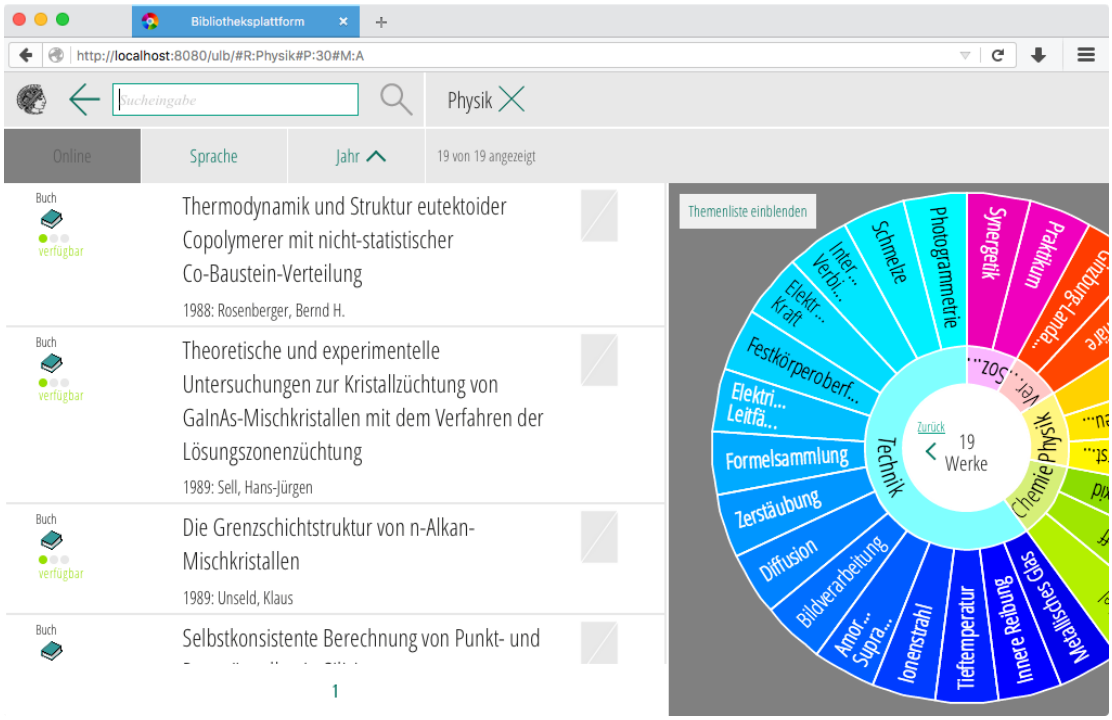


Figure 1.2: The ULB application of the Mediaplatform offers an assisted literature search using the semantic interaction component named topic wheel.

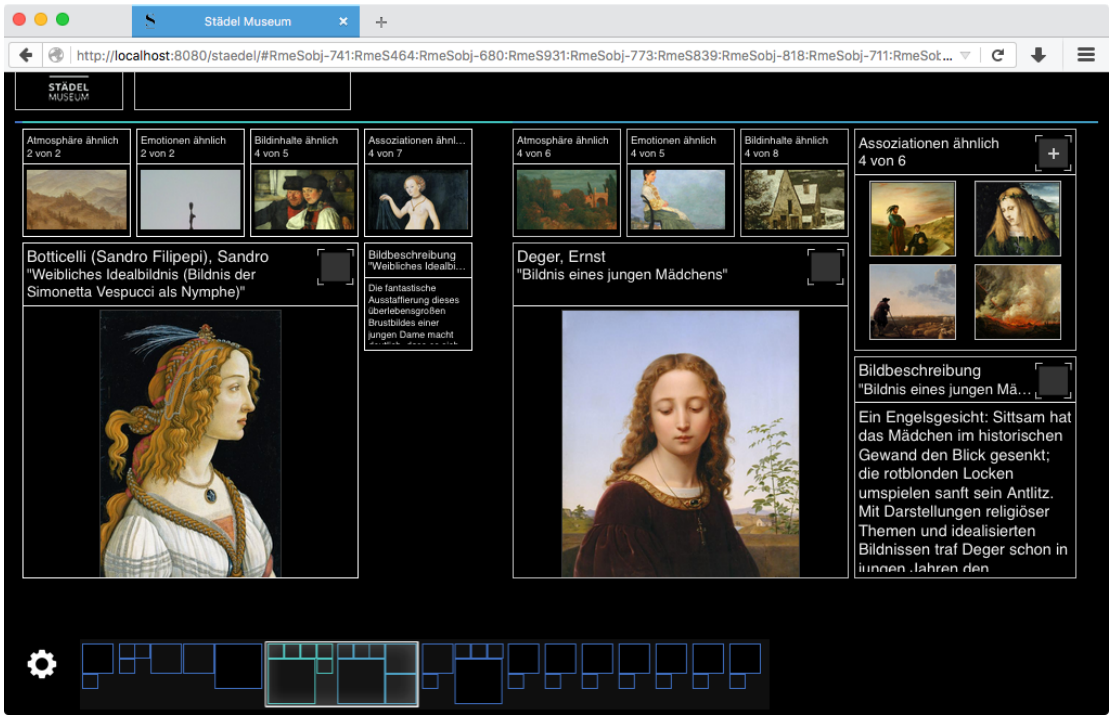


Figure 1.3: The Städel application offers a novel way to explore digital museum collections using semantic connections in a process named digital strolling.

JavaScript, the Web applications run on computers with large screens, as well as on mobile devices. Its responsive design makes it independent of different screen size. For very small screens, like smartphones, a different interaction concept is employed that suits the smartphone's interaction possibilities.

The backend server architecture has to deal with different and sometimes opposing requirements. On one side, the system needs to cope with data from various sources with different formats and licenses. This involves Open Data, Linked Data and proprietary data. The Mediaplatform is designed to be a place where different data can be used symbiotically, where the uses of a single dataset can multiply. On the other side, there are strong constraints regarding the application performance, feasibility and usability aspects.

A process called *Semantic ETL* parses the various input data, transforms the data formats into a unified entity model which is suitable for different kinds of media, adds or applies authority files, and finally streamlines the data, based on the common usage patterns of end user applications. As both the museum and the library scenarios are Information Retrieval (IR) situations, the target of the integration efforts is a highly optimised Apache Lucene search index, capable of answering common information needs with a single query. Once the index, as well as other pre-processed structures, are created in an overnight batch job, these constitute the backend of all online-operations of a server. Besides queries that go directly to the prepared Lucene index, the server contains a recommendation component capable of making semantic suggestions based on authority files or semantic tagging.

The museum site launched with a modified corporate identity design at the 200-year anniversary of the Städel in 2015[2]. It won the Forschungspreis der Hessischen Hochschulen für Angewandte Wissenschaften *(Research Award of the Universities of Applied Science Hesse)* 2014[3] and was awarded with the World Summit Award 2015[4].

[2] `https://digitalesammlung.staedelmuseum.de`, last access on 2015-12-06.

[3] `http://www.forschung-fuer-die-praxis.de/content/view/150/79/`, last access on 2015-12-19.

[4] `http://www.wsis-award.org/winner/digital-collection-141620151012`, last access on 2015-12-19.

The author was the principle backend developer of this project from 2013 to 2015 contributing core components and coordinating the development of new features. Based on that experience, in this thesis, best practices of this project are analysed, challenges are addressed, and some software components are published (Chapter 5 on page 197).

1.2 Problem Statement

The Linked Data and Open Data movements have attracted much attention to the importance of providing data and having it in suitable formats. Governments like the USA and the UK, institutions of the Cultural Heritage (CH)- and others, as well as Non Governmental Organisations (NGOs) support and foster these ideals and have launched several dedicated data portals to collect datasets and applications working with them.

Today, more Linked and Open Data than ever is available in such portals: in 2014, the number of available RDF datasets has tripled since 2011 (Hasso-Plattner-Institut, 2014). In the same period, however, and despite explicit calls (Govdata.de, 2013), the number of applications utilising this Linked and Open Data nearly remained the same; the number of datasets exceed the number of apps by orders of magnitude.

Tables 1.1 and 1.2 show the numbers of datasets and applications taken from common data portals at different points in time.

Data portal	Nov 2013	Feb 2014	Sep 2014	Oct 2014	Jul 2015	Dec 2015
data.gov	91,101	92,124	156,748	130,560	142,715	188,427
data.gov.uk	16,452	17,851	19,516		25,786	25,460
govdata.de	4,409	6,693		7,489	14,136	15,821
publicdata.eu		46,699		48,386	47,863	47,863

Table 1.1: Number of datasets per data portal, as stated on the individual pages.

Data portal	Nov 2013	Feb 2014	Sep 2014	Oct 2014	Jul 2015	Dec 2015
data.gov	342	342	341	341	75	81
data.gov.uk	312	321	349		377	384
govdata.de	15	20		19	75	24
publicdata.eu		79		79	80	85

Table 1.2: Number of applications (sometimes also called "apps" or "ideas") per data portal, as stated or summed up on the individual pages.

For datasets shown in Table 1.1, a clearly positive trend can be seen over time: the number of available data has grown on every portal over the last years. There are typically several thousand different datasets, while, in contrast, the number of applications, only go into the hundreds as shown in Table 1.2. The absolute number of applications

usually stagnates or decreases.

Based on these observations, Figure 1.4 compares the development of the relative application/dataset-ratio for each data portal. It must be measured in one-tenth of a percent and is has a clear declining trend: There are almost no apps, compared to the mass of available data.

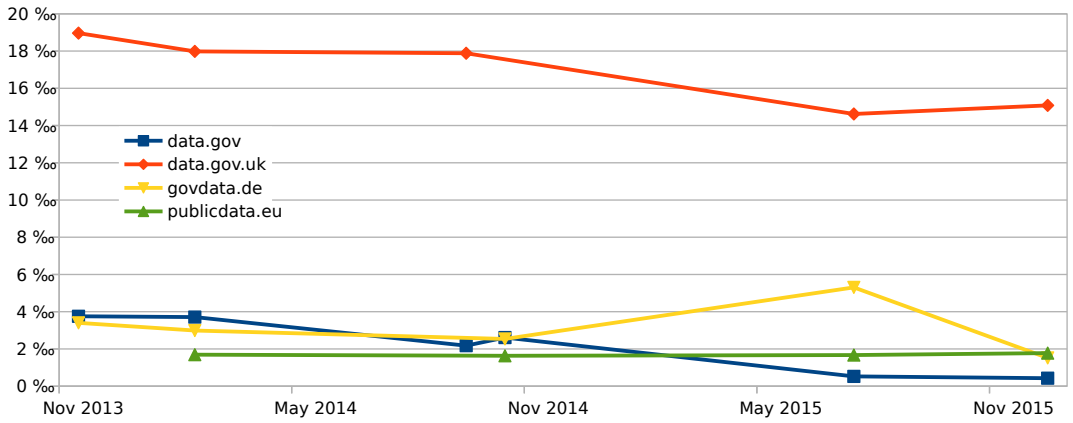


Figure 1.4: Development of the application/dataset-ratio (in one-tenth of a percent), per data portal.

On first inspection, this is surprising, because all the success factors of the Web seem to be in place: plenty of data is available in various formats and there are well-known, easy-to-use data portals publishing and distributing them.

According to the networking effect, the previous catalyst of the Web and Web 2.0, the number of applications should go in the thousands as well. As shown in Figure 1.4, this is not the case: the relative application/dataset-ratio is even decreasing over the last years. Because there definitely is data, unlike in the Semantic Web era, the chicken-egg-problem (Section 1.1.2 on page 32) is no longer an excuse for the lack of applications.

Exciting semantic applications, however, do not simply appear, just because there is data. Advanced engineering and domain specific skills are required to deliver "truly semantically empowered, friendly to use tools" (Vavliakis et al., 2011, p. 3). Even basic operations like "querying [...] and integrating/interlinking [LOD] is [...] difficult." (Janowicz et al., 2014, p. 1). In order to deliver the advantages of Linked and Open

Data to end users, more steps must be taken than current Web paradigms offer.

Commercial products such as Google's Knowledge Graph (Kohs, 2014) or IBM Watson (Ferrucci et al., 2013) demonstrate the value and possibilities of customer-oriented applications utilising Linked and Open Data.

Others, especially those who cannot invest the engineering knowledge and experience, are confronted with a complex technology stack that is considered to be of prototypical nature (Heath and Bizer, 2011, p. 85), (Kurz et al., 2011, p. 17), and, as a consequence, few end user applications which often suffer from bad usability and performance (Edelstein et al., 2013, p. 60) (Dirschl and Eck, 2015, p. 10).

It can be observed that there is a lack of pragmatic architectures (WaSABi Organizing Committee, 2014), (Edelstein et al., 2013), which, in the opinion of the author, need to cover the entire data integration process of Linked and Open Data for application-specific domains. This includes best practices from current Web development, as well as from other disciplines such as Data Warehousing.

The lack of pragmatic Linked and Open Data integration strategies and applications is one of the most pressing issues of current Web science, as it impacts all possible application domains of this technology and might undermine investments in Linked and Open Data.

Because it is open, ready to be used for everyone, and everyone can contribute, Linked and Open Data provides a foundation of a democratisation of data. Just by publishing it, however, insights from the data remain sealed for majority of the citizens, because it is not transported to end users. As long as data remains in technical formats, only people with technical skills can access it. A lesson that can be learned from the Web, and particularly later from the Web 2.0, is the importance of lowering technical barriers to facilitate a broad adoption. As a matter of course, today, nobody needs to understand HTML to benefit from the Web or contribute to it. Applied to Linked and Open Data, this means that by providing usable, fast applications with proper interaction mechanisms, data has the potential to reach more of society. Without this, only those

who can access expert know-how and resources can benefit from Linked and Open Data. The huge advantage of openness in Linked and Open Data is lost.

The lack of applications is also an issue for the many corporations, non-profit-organisations, government agencies and voluntary contributors who actively publish data as Linked and Open Data, as well as academia. Publishing Linked and Open Data requires skills and experience in handling an expert technology and is thus time-consuming and costly. It also requires a long-term commitment. Some institutions, like the British Museum, reinterpret their educational mandate and start experimenting with their role as Linked and Open Data providers. As such, they focus on their core competence, the publication of raw data in a specific domain, and decouple data from their idea of how it should be presented. This high-quality, expert-curated content offers huge potential for new kinds of semantic applications, built with a mix of expert knowledge of several domains and combining it into new offerings. All these efforts, however, usually need to be justified, e.g. towards tax payers, shareholders, or givers.

As long as the result of investment is just plain data without immediate use or application scenarios, a justification of the costs is difficult. Many Linked and Open Data investments might be in upfront, in the trust that Linked and Open Data will end-up with something useful. Thus, the persistent long-term lack of Linked and Open Data applications is a threat to the entire Linked and Open Data movement, as it undermines major Linked and Open Data investments.

Only by successfully producing working applications can the use of Linked and Open Data ultimately be demonstrated. Once applications exist, it is easier to advocate solutions and to measure their impact and acceptance, for example by using the application registries many data portals maintain.

The Mediaplatform is a sample of a successful Linked and Open Data integration project towards working applications, with a proven high utility and user-friendliness (Deuschel, Greppmeier, Humm and Stille, 2014). The project can be seen as a prototype for a whole class of similar approaches, trying to integrate and apply Linked and Open Data in an

application-specific domain. Thus, similar projects are likely to face the same challenges as well.

The system is designed to store all kinds of source data, including Linked and Open Data, into the search engine Apache Lucene, as offering search capabilities is the primary use case for the end user applications. The author considers the streamlining of data towards actual end-user application scenarios to be one of the success factors of this project (Heuss et al., 2015). New and advanced features, however, require the architecture to accept new challenges (Deuschel, Heuss and Broomfield, 2014). This includes novel components for content recommendation, crowdsourcing and storytelling, which need additional means to access integrated data going beyond the search engine capabilities. Furthermore, the operational practice suggests a number of improvements, such as allowing data sources to be loaded differentially.

To support these new features, new approaches must be integrated in the project, preserving existing best practices and sustaining the working end-user applications.

Thanks to the cooperation partners Städel Museum and ULB, the research project is based on real use cases and real business cases. It can be expected that solutions found in the present work have impact on future developments of the Mediaplatform, while respecting existing architectural success factors. Vice versa, research results can be benchmarked by the domain-specific setting and the high usability demands of the Mediaplatform.

1.3 Research Question

To address the lack of Linked and Open Data integration strategies, the research question studied in the present work is: How can developers be supported in the integration of Linked and Open Data towards end-user applications? Four topics to be explored are related to this question, as shown in Figure 1.5.

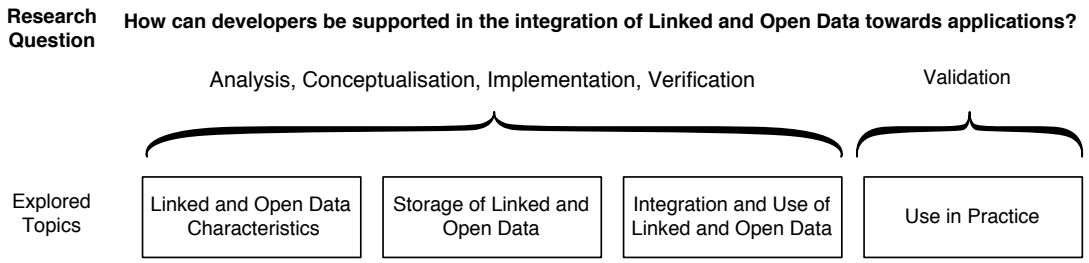


Figure 1.5: Research Question and explored topics with its context. Based on this, Figure 3.2 on page 94 shows how these questions are addressed with scientific methodologies.

The topics explored within the context of that question are:

Linked and Open Data Characteristics What are the basic characteristics of Linked and Open Data? This includes data formats, licences, structured-ness, and availability.

Storage of Linked and Open Data What is a suitable storage for the integration of Linked and Open Data? This must include findings about the basic Linked and Open Data characteristics.

Integration and Use of Linked and Open Data What are logical steps to be taken in order to use Linked and Open Data in practical applications? What are the best practices, known from other domains and operational reality? How can these steps be implemented with existing technology?

Use in Practice How can the mentioned steps be supported in an actual, real-world project?

1.4 Outline of Thesis

The remainder of the present thesis is structured as the follows:

- chapter Background and Literature Review on the facing page conducts a review of relevant sources and a status quo in several domains of Web Science, in related data handling disciplines and current Web developer paradigms. In addition, a research gap analysis is made (Section 2.9 on page 85).
- chapter Research Methodology on page 94 describes how the previously identified research question (Section 1.3 on page 45) in research and practice can be tackled with scientific methodologies.
- chapter Research Design on page 96 defines how the previously selected methodology is applied in the specific context of the present thesis.
- chapter Framework for the Integration of Linked and Open Data on page 121 explains the application of selected methods. Individual objectives are:
  - A Field Study On Linked and Open Data at Datahub.io on page 122 to clarify Linked and Open Data characteristics based on the real and representative data portal datahub.io.
  - A Storage Evaluation for a Linked and Open Data Warehouse on page 134 to identify a suitable database technology based on integration patterns and today's storage choices.
  - Development of a Linked and Open Data Integration Framework on page 159 to support developers in the integration and use of Linked and Open Data.
- chapter Proof of Concept Framework Application on page 197 applies the developed framework in the existing code base of Mediaplatform and analyses conceptual and software changes.
- chapter Conclusion on page 211 gives a short summary and outlook.

Chapter 2

Background and Literature Review

In the introduction of this thesis, it is pointed out that while there are many datasets freely available today, there are not even close to that many applications exploiting this data. Also, the theory is described that this is the result of the lack of integration strategies in application-specific domains. This not only hinders developers in implementing new semantic applications on a broad scale, but it is also a threat for the entire Linked and Open Data movement, as investments in providing such data do not seem to pay off with end-user-ready benefits.

In this chapter, relevant literature in the Web Science domain is reviewed. The subjects of this analysis are existing reference architectures for building semantic applications, statements about fundamental characteristics of Linked and Open Data, and suggested storage solutions.

In addition, existing conceptual approaches in data integration are described, as well as reviews of existing data integration frameworks and a selection of existing semantic applications. Moreover, current development trends and approaches are put into context and are described shortly.

This chapter concludes with a Summary of Findings on page 85, which collects the key findings of this literature review, extracts patterns and best practices and identifies the research gaps.

2.1 Reference Architectures

Reference architectures capture the essence of an architecture of a certain system in a certain domain (Nakagawa, 2012, p. 159). They have several purposes: Firstly, they guide developers in their goal of building an application within the given domain (Nakagawa, 2012, p. 159). Secondly, they offer "an agreed-upon functional description of the system structure, components and component interactions" (Gerber et al., 2008, p. 1). Thirdly, they express the goals and visions of a certain technology stack. Reference architectures sum up experiences and best practices, with the "aim of disseminating and reusing this knowledge and standardising the systems as well" (Nakagawa, 2012, p. 159).

In the following, the status quo of reference architectures in Web Science is presented.

2.1.1 Semantic Web Layered Cake, 2000-2006

From 2000 to 2006, Tim Berners-Lee published several versions of the *Semantic Web architecture*, also known as *Semantic Web Layered Cake*. They have not been the subject of a W3C recommendation or any other formal publication, but they were presented at conferences such as the World Wide Web Conference 2005. (Gerber et al., 2008, p. 2f). Figure 2.1 on the facing page shows all four known versions of the Semantic Web architecture.

All versions describe a stack of standards that are related to the Semantic Web. Lower layers contain the basic building blocks already known from the Web: URI and XML, as well as the Unicode character encoding. In the middle, Semantic Web essentials technology, like SPARQL, RDF, RDF Schema (RDFS) and Web Ontology Language (OWL). On top of that, concepts like trust, proof or logic are mentioned. It is remarkable that the fourth version was the first to introduce positions for applications and user interfaces, as a new topmost layer.

Because none of these versions depicts functionality at conceptual levels, but instead W3C languages and technologies, they have been subject to controversial discussions

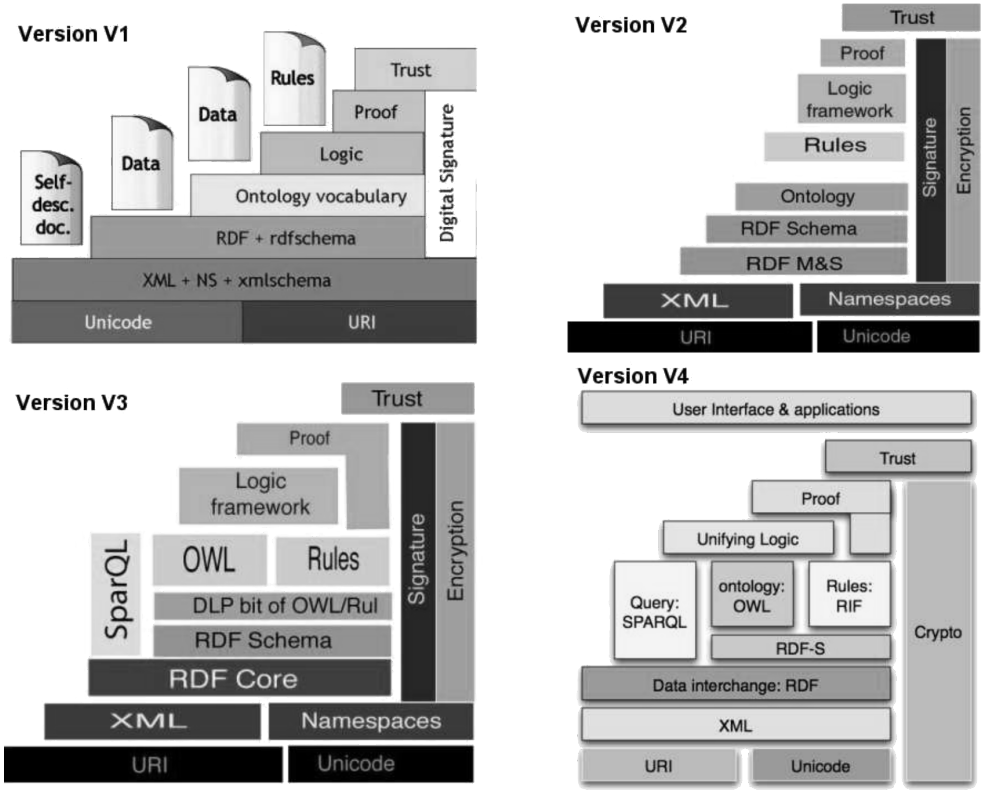


Figure 2.1: Four different versions of the Semantic Web Layered Cake, presented from 2000 to 2006 (Gerber et al., 2008, p. 3).

(Gerber et al., 2008, p. 3). Besides the lack of functional descriptions, Gerber and Barnard have identified further inconsistencies, regarding the layer design, the principle triangular structure and mixed naming (Gerber et al., 2008). Nevertheless, even some current works, e.g. Anibaldi et al., still refer to this 15 year old architecture.

2.1.2 Comprehensive, Functional, Layered Architecture (CFL), 2008

Because of the consistency issues with Tim Berners-Lee's Semantic Web Architecture, Gerber and Barnard introduced a refined version of the Semantic Web Architecture diagram in 2008: the *Comprehensive, Functional, Layered Architecture (CFL)*, shown in Figure 2.2 on the next page.

The main difference between the CFL compared to Berners-Lees architectures is the fact that the CFL "abstracts and depicts related functionalities rather than the W3C

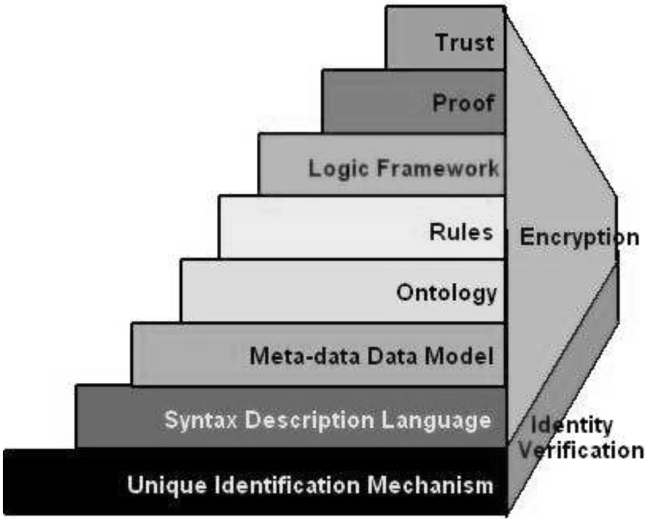


Figure 2.2: Semantic Web CFL Architecture, (Gerber et al., 2008, p. 7).

technologies used to instantiate these functionalities" (Gerber et al., 2008, p. 6). This abstraction can also be found in the choice of data models for meta-data; where, in contrast, "the original reference architecture would have been limited [...] to RDF " (Gerber et al., 2008, p. 11).

2.1.3 5-Star LOD Maturity Model, 2006

In 2006, Berners-Lee introduced the term Linked Data with "simple rules for Web publishers" for releasing data in the Web (Summers and Salo, 2013, p. 5). Just like the previously introduced Semantic Web architecture, these principles contain W3C standards - a fact that is subject of criticism (Gerber et al., 2008) - but they also include functional descriptions and expectations that "apply to make the web grow" (Berners-Lee, 2006):

1. Use URIs as names for things
2. Use HTTP URIs so that people can look up those names.
3. When someone looks up a URI, provide useful information, using the standards (RDF*, SPARQL)

4. Include links to other URIs [...] so that they can discover more things.

Later, Berners-Lee added the star ranking system as shown in Table 2.1, to assess data (Berners-Lee, 2006):

Rating	Requirement
1 Star	Available on the web (whatever format) but with an open licence, to be Open Data
2 Stars	Available as machine-readable structured data (e.g. excel instead of image scan of a table)
3 Stars	as (2) plus non-proprietary format (e.g. CSV instead of excel)
4 Stars	All the above plus [...] use [of] open standards from W3C (RDF and SPARQL) to identify things, so that people can point at [data].
5 Stars	All the above, plus: Link your data to other people's data to provide context.

Table 2.1: 5-Star-Maturity-Model for Open Data, (Berners-Lee, 2006).

Use of the W3C recommendations RDF and SPARQL is mandatory to achieve four stars or more. Neither the four rules nor the 5-star ranking system give architectural hints for developers, even though comprehensive works such as (Edelstein et al., 2013) indicate they are refereed with this expectation.

2.1.4 Linked Data Application Architectural Patterns, 2011

In 2011, Heath and Bizer described three basic architectural patterns for Linked Data applications (Heath and Bizer, 2011, p. 97f). They can be classified by the extent of their use of external processing power and external storage resources.

According to the *Query-Federation-Pattern*, applications send SPARQL queries or parts of queries to external endpoints and receive the results. Data processing and storage happens entirely on remote resources. With the *On-The-Fly-Dereferencing-Pattern*, a Linked and Open Data application resolves links to external data sources in real-time, when the application needs it. Data processing is local, while the primary data storage is remote. The *Crawler-Pattern*, shown in Figure 2.3, maintains a local RDF storage, that is loaded constantly or during an offline operation, and that is accessed via SPARQL or a RDF API to process queries. Thus, data processing and storage are both local.

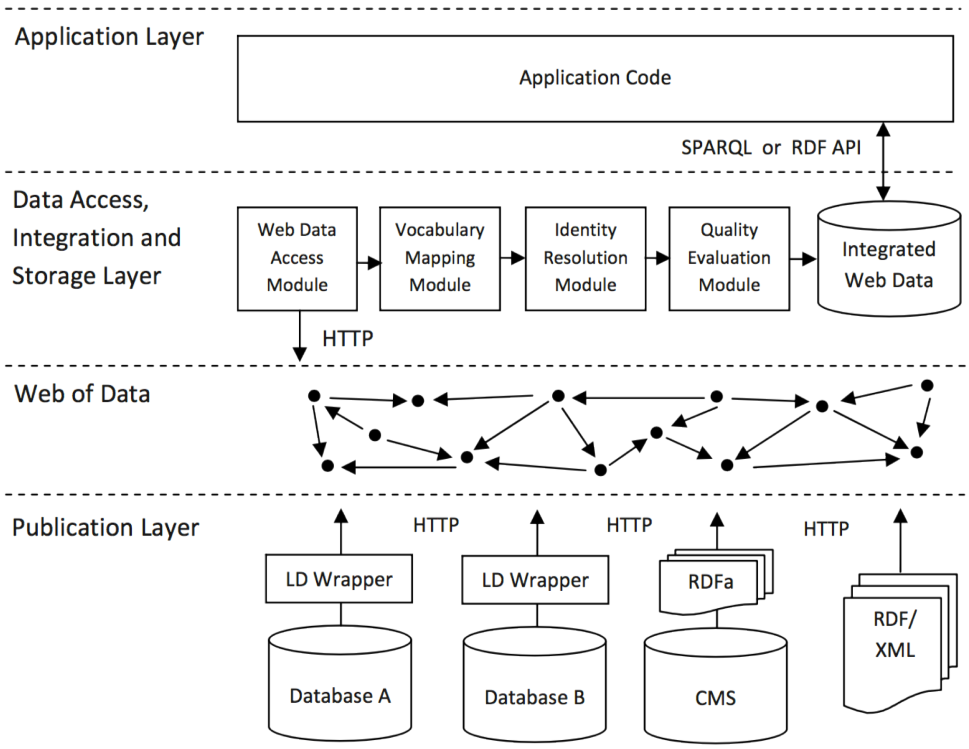


Figure 2.3: Semantic Web Crawling Pattern, (Heath and Bizer, 2011, p. 99).

There are some transition steps between the public Web of Data layer and the application layer. These involve activities to acquire, interlink, and quality assure RDF (Heath and Bizer, 2011, p. 99f).

Each pattern complies to Berners-Lee's third rule of releasing data on the Web (Section 2.1.3 on page 50), thus the W3C recommendations SPARQL and RDF are used.

2.2 Characteristics of Linked Data and RDF

In existing literature, some statements about fundamental characteristics of Linked Data and RDF can be found.

2.2.1 Uniformity, Referencibility, Timeliness

Auer et al. mention a number of "significant benefits" (Auer et al., 2011, p. 3) associated with the use of 5-star Linked Data: By expressing every kind of information in triples with subject, predicate and object, RDF is a *uniform* format (Auer et al., 2011, p. 3).

Because identifiers in RDF are URIs, they offer *de-referencibility* as further information can be given by following a link. RDF triples can also be *coherent* when using "different namespaces in subject and object position" (Auer et al., 2011, p. 4). As RDF is a shared "single mechanism for representing information", it is "very easy to attain a syntactic and simple semantic integration of different Linked Data sets" (Auer et al., 2011, p. 4). Finally, RDF is considered to confer *timeliness*, because "is straightforward to access and use the updated data source, since time consuming and error prone extraction, transformation and loading is not required." (Auer et al., 2011, p. 4)

2.2.2 RDF is Entity-Agnostic

When using RDF, it is important to note that datasets "do not necessarily consist of clearly identifiable 'records'" (Isaac et al., 2005). Instead, the format is constructed to allow information even about one single topic to be distributed across the Web (Allemang and Hendler, 2011, p. 28). As a result, one single logical entity with n attributes is composed of at least n triples - regardless of whether information is retrieved from the Web or consolidated in one local file.

This flexibility allows arbitrary information to be stated about an entity, without violating any predefined schema, and existing data can, in theory, be extended easily (Callimachus - 3 Round Stone, 2015). The downside however is the fact that this introduces redundancy and a lack of compactness (Allemang and Hendler, 2011, p. 49).

2.2.3 Not Consistently Open, Not Consistently Linked

In theory, the 5-Star maturity model (Section 2.1.3 on page 50) provides a clear definition and differentiation of what Open Data and Linked Data means. Such a strong differentiation, however, can not usually be observed in practice. In reality, "[d]ata may be linked but not open, or open but not thoroughly linked" (Edelstein et al., 2013, p. 2). Moreover, Linked Data is often not based on Open Data: 57.4% is not openly licensed for reuse (Digital Meets Culture, 2014).

2.2.4 Quality Issues

Today, many organisations, e.g. publishers, institutions, companies, research projects, press, governments and individuals contribute to the Web of Data (Vatant, 2012). Quality is essential for working with world-wide-interconnected knowledge and maintenance is a major challenge (Auer, 2011, p. 6).

Current statistics for 1048 datasets of the Linked and Open Data cloud by LODStats (Demter et al., 2012) show that nearly 2/3 of the available datasets have "problems" (Webpage, 2014), e.g. the dumps contain errors or SPARQL endpoints are not available.

This is especially an issue for application scenarios which are sensitive to data quality, e.g. in newsrooms (Pellegrini, 2012) or in the publishing business (Dirschl and Eck, 2015).

Furthermore, DBpedia has known consistency problems (Houle, 2014), which can be explained by the exact way in which DBpedia entries are extracted from Wikipedia.

Some of the Linked and Open Data quality issues are addressed by tools like Jena Eyeball (Apache Jena, 2014) or by the W3C working group RDF Data Shapes (Prud'hommeaux et al., 2014). Both approaches allow developers to define constraints on the exact structure of expected RDF data, so that they conform to respective information needs.

2.2.5 Completeness

Within the Linked Open Data Cloud, DBpedia is usually referred as the nucleus (Schandl et al., 2012). The majority of tools centre their operations around DBpedia (Digital Meets Culture, 2014), e.g. using DBpedia as controlled vocabulary when detecting named entities (Kobilarov et al., 2009). Especially in the domain of Natural Language Processing (NLP), the structured nature of information from DBpedia is considered to have benefits (Gunes et al., 2012).

Some works, however, indicate issues with the normalisation and granularity of the

represented information (Dirschl et al., 2014). The maintenance of large datasets is expensive and "difficult to archive by academic projects" (Neubert and Tochtermann, 2012) like DBpedia. In contrast, alternative datasets such as Virtual International Authority File (VIAF), Gemeinsame Normdatei *(Integrated Authority File)* (GND), or the Library of Congress Authorities are collected and maintained by trained professionals (Neubert and Tochtermann, 2012). Table 2.2 compares the number of entities.

	Persons	Organisations
DBpedia	364,000	148,000
Library of Congress Authorities	3,800,000	900,000
GND	1,797,911	1,262,404
VIAF	10,000,000	3,250,000

Table 2.2: Comparison of the Number of Entities in DBpedia, VIAF, GND and Library of Congress Authorities (Neubert and Tochtermann, 2012)

2.2.6 Simple Constructs

RDF is a conceptually simple format. In combination with the OWL and RDFS specifications, however, more sophisticated modelling and reasoning is possible, like the definition of "what inferences are valid, given certain patterns of triples" (Allemang and Hendler, 2011, p. 121). With OWL 2.0 in the late 2012 (W3C OWL Working Group, 2012), the possibilities have been extended further. Use of these modelling techniques requires advanced tool support.

Despite the standardisation efforts that have been invested in extending the feature richness of RDF, in practice, simple RDF constructs are preferred by most datasets (Glimm et al., 2012).

2.2.7 JSON-LD and SDF

The W3C-promoted technology stack for Linked Data is not entirely accepted by the whole Web Science community. A "large community" refuses to use RDF and SPARQL because it is "unnecessarily complicated" (Zaino, 2013). This is especially true for Web developers who "had not traditionally been able to keep up with the steep learning curve associated with the Semantic Web technology stack" (Franzon, 2014*a*).

In response to this criticism, in the last years, two alternatives to RDF have developed. Both use JavaScript Object Notation (JSON), a format that is commonly regarded to be Web developer friendly (Franzon, 2014*a*).

In 2013, the OKFN introduced the *Simple Data Format (SDF)* in competition to RDF. SDF is built on open formats specified by the Internet Engineering Task Force (IETF), such as CSV and JSON, and aims at simplicity and practicality (Sporny et al., 2014).

In addition to that, in 2014, the W3C introduced RDF 1.1 (Wood, 2014), which highlights a "clear separation between RDF, the data model, and its serialisation formats" (Franzon, 2014*b*). It introduces the new serialisation format JSON-LD, which allows "existing JSON to be interpreted as Linked Data with minimal changes" (Sporny et al., 2014). Just like SDF, JSON-LD aims at using "Linked Data in Web-based programming environments", to build "interoperable Web services" and to store "Linked Data in JSON-based storage engines" (Sporny et al., 2014).

2.2.8 Pay-as-you-go Data Integration

Traditionally, data integration requires a complex integration infrastructure that processes data from different sources and that unifies different data models and schemata (Heath and Bizer, 2011, p. 106). In contrast, using Linked Data, data publishers can contribute to this process, by "reusing terms from widely used vocabularies, publishing mappings between terms from different vocabularies, and by setting RDF links pointing at related resources as well as at identifiers used by other data sources to refer to the same real-world entity." (Heath and Bizer, 2011, p. 106).

Thus, Linked Data gives suggestions on integrating it, even though the quality of these suggestions is uncertain (Heath and Bizer, 2011, p. 106). As a result, it is perceived that less upfront investment in Linked Data integration is required, referred to *pay-as-you-go data integration* (Heath and Bizer, 2011, p. 106).

2.2.9 Data Portal Analysis

With Roomba (Assaf et al., 2015), recent work provides means for the analysis of data that is stored on data portals, including CKAN-based repositories. In order to work with "bad quality metadata" of "most of the datasets", the tool provides capabilities to identify "missing information and [...] to automatically correct them [...] when possible" (Assaf et al., 2015, p. 2).

2.3 Storage

A *Triple Store*, also known as a *RDF store*, is a database optimised for the storage and processing of RDFs (Allemang and Hendler, 2011, p. 52). As the underlying technology of triple stores solves graph-problems, triple stores can be considered as part of the Not Only SQL (NoSQL) class of graph stores (Edlich et al., 2011, p. 8). Vice versa, Neo4J, a common NoSQL graph store, can be used as a "full triple store" (Neo Technology, Inc., 2014)

The majority of triple stores can also be considered to be *Linked Data Server*s (Kurz et al., 2011), tools that strictly implement the W3C paradigms and best practices and that offer access to information in standardised ways (Bizer et al., 2009, p. 8), like the Linked Data Platform (Speicher et al., 2014).

Triple Stores usually form the backend in Semantic Web and Linked Data architectures, because they offer RDF and SPARQL capabilities that are mandatory according to existing reference architectures (Section 2.1 on page 48). In the following, common approaches are briefly introduced.

2.3.1 Typical Triple Stores

The majority of available triple stores offer database features like storage, a query engine and management interfaces for RDF. Common samples are *Virtuoso*, *4store*, *BigData*, *BigOwlim*, *Sesame*, and *Redland*. Recent stores focus on application developers by providing Create, Read, Update, Delete (CRUD) operations on RDF via HTTP, like *RWW.io* (rww.io, 2015) or *Apache Clerezza* (Apache Clerezza, 2015). Besides Open

Source or community editions, there also exist commercial triple store such as *Stardog* (Stardog, 2014).

Virtuoso is the most popular triple store, known from many applications such as backend of DBpedia (DBpedia, 2015), as being part of commercial integration suites PoolParty (Semantic Web Company, 2015), or as storage engine for the Semantic Desktop KDE Nepomuk. The triple store has a long history, starting at the end of the 90s, and is known for continuous improvements.

2.3.2 NoSQL and Big Data

To address scalability issues in triple stores (Khadilkar et al., 2012), solution strategies can be found in the domain of NoSQL databases (Auer et al., 2012).

With *Jena-HBase* (Khadilkar et al., 2012) and the approach by Sun & Jin (Sun and Jin, 2010), there are two scalable triple stores implemented with the NoSQL database HBase. Sun and Jin also offer the possibility to scale-out a SPARQL query via MapReduce.

Large Knowledge Collider (LarKC) focusses on providing reasoning capabilities on Web-scale over billions of data sets (Assel et al., 2011). Similar to this, CLODA, a crowd-sourcing plattform for Linked Data, employs HBase and CouchDB (Larkou et al., 2013).

The dataset analysis of Stadtmüller et al. is based on an Hadoop infrastructure (Stadtmüller et al., 2013).

The OCLC, publisher of the 197-billion-entry WordCat RDF dump, created the dataset entirely without any triple store, but with Hadoop (Wallis, 2014).

2.3.3 Translation Layers

Storage solutions often have to be integrated into existing storage landscapes (Nimis et al., 2014, p. 19). A number of approaches thus focus on providing W3C standardised access to information that is stored in other storages forms.

Triplify, for example, transforms data from relational Database Management Systems (DBMSs) into Linked Data. *D2R Server* (Kurz et al., 2011) and *R2O* (Rodriguez

and Gómez-Pérez, 2006) allow mappings from data stored in relational DBMS to the Semantic Web. Vice versa, there are approaches such as *MAESTRO* that provide an access layer for Structured Query Language (SQL) over ontology-based mappings (Calvanese et al., 2011).

Apache *Any23* offers a web service and command line to transform structured data in various formats into RDF (Apache Any23, 2013). Related to that, *CSV2RDF* offers a conversion from CSV to RDF (PublicData.eu, 2014).

2.4 Storage Benchmarks

A number of existing benchmarks evaluate the performance and reliability of Linked and Open Data storage solutions. All latest benchmark results are limited to triple store implementations only and some comparisons are over five years old - a fact that is addressed later on. The exact application scenario simulated in individual benchmarks is subject of controversial discussions within the community.

2.4.1 Lehigh University Benchmark (LUBM)

Lehigh University Benchmark (LUBM) is a benchmark based on the Univ-Bench, an ontology that "describes universities and departments and the activities that occur at them." (Guo et al., 2005, p. 160). For this ontology, LUBM creates random data of arbitrary sizes and measures reasoning and storage capabilities (Guo et al., 2005, p. 158).

The evaluation conducted in 2005 involved memory-based systems and persistent storage systems. It contained the implementations DLDB-OWL, Sesame and OWLJessKB. The results of both the loading times and the query performances "lead to some scalability and efficiency concerns." (Guo et al., 2005, p. 167). Because the implementations performed very differently, recommendations for certain use case scenarios depend on individual tool properties (Guo et al., 2005, p. 174).

2.4.2 SPARQL Performance Benchmark (SP2Bench)

SPARQL Performance Benchmark (SP²Bench) simulates a scenario based on DBLP, a "database contains bibliographic information about the field of Computer Science and, particularly, databases." (Schmidt et al., 2009, p. 224). The benchmark includes a data generator for DBLP-documents as well as "a set of carefully designed benchmark queries. " (Schmidt et al., 2009, p. 222). In contrast to LUBM, the queries of SP²Bench contain more advanced SPARQL features like `OPTIONAL` and `FILTER` (Morsey et al., 2011, p. 467).

The 2008 SP²Bench evaluation included the triple stores Redland (native), the abstraction layer SDB, in-memory and persistent Sesame, as well as Virtuoso (Schmidt et al., 2008).

The results show that almost all implementations took multiple seconds to perform even simple queries. In addition, all implementations showed tendencies to time-out, especially with growing dataset sizes (Schmidt et al., 2008). Besides that, SDB and Redland further "returned wrong results for a couple of queries" (Schmidt et al., 2008, p. 11).

2.4.3 Berlin SPARQL Benchmark (BSBM)

The *Berlin SPARQL Benchmark (BSBM)* simulates an actual application scenario of a database in the domain of e-commerce. Thereby, "a set of products is offered by different vendors and consumers have posted reviews about products" and the "benchmark query mix illustrates the search and navigation pattern of a consumer looking for a product." (Bizer and Schultz, 2015).

Early versions of BSBM covered the triple store implementations Sesame, Jena, Virtuoso, and D2R in different configurations and underlying engines (native RDF and SPARQL-to-SQL engines), as well as SQL-based engines MySQL and Virtuoso relational DBMS (Bizer and Schultz, 2009, p. 15).

These early comparisons indicated significant performance issues: native SQL engines

outperformed native RDF engines by a factor 8.5 (Bizer and Schultz, 2009, p. 26). Similar observations were made for loading times, and growing dataset sizes amplified this effect even further (Bizer and Schultz, 2009, p. 26).

Bizer and Schultz have two explanations for the observed performance discrepancies: The fact that (1) SPARQL was relatively new in 2009, especially compared to SQL as well as the fact that (2) the use case used a relatively homogeneous data model, which does not exploit RDFs flexibility.

Since 2011, new versions of the BSBM (V3 and above) no longer contain non-native RDF engines and different storage setups. The evaluation includes the Triple Stores 4store, BigData, BigOwlim, TBD, and Virtuoso. Evaluations revealed technical problems with Virtuoso, TDB, and BigOwlim, as well as incomplete support for SPARQL in BigData and 4store. (Bizer and Schultz, 2015)

2.4.4 DBpedia SPARQL Benchmark (DBPSB)

The DBpedia SPARQL Benchmark (DBPSB) has been developed in response to LUBM, SP^2Bench and BSBM. All three benchmarks are manually constructed based on defined use case scenarios, which causes them, in the opinion of Morsey et al., to have “artifical” and “synthetic” datasets and queries (Morsey et al., 2011, p. 466f).

In contrast, DBPSB claims to work with “real” data and “real” queries (Morsey et al., 2011). Thereby, “dataset-agnostic” data is processed by queries that have actually been fired against a DBpedia endpoint (in a three month period in 2010) (Morsey et al., 2011, p. 456ff).

Some of the results confirm evaluation results of other benchmarks, e.g. the fact that Virtuoso is the best performer (Morsey et al., 2011, p. 466). Generally, it is stated that the “observed differences in performance between different triple stores amplify when they are confronted with actually asked SPARQL queries, i.e. there is now a wider gap in performance compared to essentially relational benchmarks” (Morsey et al., 2011, p. 466).

2.4.5 Linked Data Benchmark Council Semantic Publishing Benchmark (LDBC SPB)

The *Linked Data Benchmark Council Semantic Publishing Benchmark (LDBC SPB)* is a benchmark for "RDF database engines", developed with and for the Semantic Web business case of the British Broadcasting Corporation (BBC) (Kotsev et al., 2015, p. 2). It is one of the most recent benchmarks in the domain, published in draft since June 2014 (Kotsev et al., 2015, p. 2).

The subject of analysis is the application of a media or publishing organisation, dealing with "large volume of streaming content, namely articles and other "creative works" and "media assets" [,] metadata that describes it and links it to [...] taxonomies and databases that include relevant concepts, entities and factual information" (Kotsev et al., 2015, p. 2). Thus, the queries include insert, update, delete, aggregation operations that are simultaneously executed (Kotsev et al., 2015, p. 2, p. 22). As well as using existing reference datasets, a data generation tool is provided (Kotsev et al., 2015, p. 17).

First results contain performances measures for Virtuoso and GraphDB on different hardware, with Virtuoso achieving the most queries per second (Linked Data Benchmark Council, 2015).

2.5 Conceptual Data Integration

In contrast to the frameworks introduced in the previous section, which represent what has been implemented for the data integration of Linked and Open Data, this section focuses on the conceptual, logical processing steps that are passed through when integrating data. Thereby, the requirements of several domains are considered.

2.5.1 Knowledge Discovery in Databases Process

The primary goal of the *Knowledge Discovery in Databases (KDD)* process (Fayyad et al., 1996) is "the development of methods and techniques for making sense of data" and "extracting useful information (knowledge) from the rapidly growing volumes of

digital data" (Fayyad et al., 1996, p. 37). It consists of following nine iterative steps, containing "loops between any two steps" (Fayyad et al., 1996, p. 42):

Goal definition In the first step, the goal of the KDD application from the viewpoint of the customer is defined.

Data Selection In the second step, source data is selected (Fayyad et al., 1996, p. 42)

Preprocessing Preprocessing involves basic (cleanup) operations like "removing noise [...], collecting [...] information to model or account for noise, deciding on strategies for handling missing data fields, and accounting for time-sequence information and known changes" (Fayyad et al., 1996, p. 42)

Transformation In the transformation step data is reduced and projected by "finding useful features to represent the data depending on the goal of the task. " (Fayyad et al., 1996, p. 42).

Selection of Data Mining Method In step five, the proper data mining method is selected, according to the respective goals in step 1. (Fayyad et al., 1996, p. 42).

Evaluation Step six is about selecting the proper data-mining and data pattern search-methods and parameters. (Fayyad et al., 1996, p. 42).

Data Mining Step seven involves the actual act of data mining as defined in previous steps: "searching for patterns of interest in a particular representational form or a set of such representations, including classification rules or trees, regression, and clustering" (Fayyad et al., 1996, p. 42).

Interpretation In the interpretation step, results of the data mining are interpreted, which might result in changes to any previous step. (Fayyad et al., 1996, p. 42).

Acting on Discoveries The last step of an iteration of the KDD process is using the gathered knowledge, e.g. by "incorporating [it] into another system" (Fayyad et al., 1996, p. 42).

2.5.2 Data Warehouse Architecture

Figure 2.4 shows a typical *Data Warehouse* with five layers (Vaisman and Zimányi, 2014).

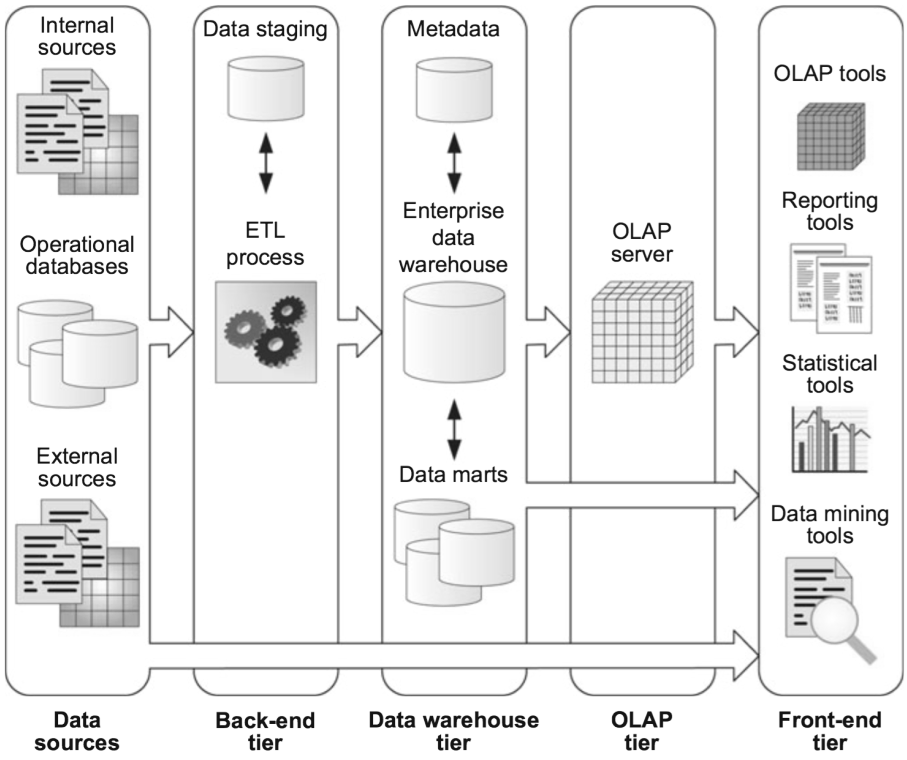


Figure 2.4: A typical data warehouse architecture (Vaisman and Zimányi, 2014, p. 77).

Data source

Data source represents data from "operational databases and other external or internal" sources (Vaisman and Zimányi, 2014, p. 76).

Back-end tier

The *back-end* tier provides a *staging area*, and an intermediate area where data is processed with an Extraction, Transform, Load (ETL) process (Vaisman and Zimányi, 2014, p. 76). *Extraction* describes the gathering of multiple, heterogeneous data sources, *Transformation* unifies them to a common data model, and *Load* feeds the transformed data to the next tier, the data warehouse tier (Vaisman and Zimányi, 2014, p. 76f).

Data warehouse tier

The *Data Warehouse* tier contains the transformed, unified data as well as metadata about the structure of the warehouse, the data sources and the applied ETL process (Vaisman and Zimányi, 2014, p. 76f).

OLAP tier

The *Online Analytical Processing (OLAP)* tier provides "business users" with an additional layer to navigate, query, analyse and report the integrated data (Vaisman and Zimányi, 2014, p. 78f).

Front-end tier

The *front-end* tier provides means for "client tools that allow users to exploit the contents of the data warehouse" (Vaisman and Zimányi, 2014, p. 79).

2.5.3 News Production Content Value Chain

Pellegrini conceptualises the process of news production in the five steps of the *News Production Content Value Chain*:

1. Content Acquisition The first step is about the "collection, storage and integration of relevant information necessary to produce a news item.", (Pellegrini, 2012, p. 96). Thereby, data from various internal and external sources are "pooled [...] for further processing" (Pellegrini, 2012, p. 97).

2. Content Editing In the second step, several transformations are executed on the pooled data. Data is *adapted* "in a way that it can be used in the editorial process" (Pellegrini, 2012, p. 97). Interlinking and enrichment aims at "disambiguating existing concepts or [at] providing background knowledge for deeper insights" (Pellegrini, 2012, p. 97).

3. Content Bundling The third step is about bringing information into specific contexts and personalising it. By customising information access via "device-senstive de-

livery", landing pages or dossiers, an improved experience is possible (Pellegrini, 2012, p. 97).

4. Content Distribution The fourth step is about dealing with "the provision of machine-readable and semantically interoperable (meta)data via Application Programming Interfaces (APIs) or SPARQL Endpoints" (Pellegrini, 2012, p. 97).

5. Content Consumption The final step is about "end user applications" aiming at "enabl[ing] a human user to search for and interact with content items in a pleasant [and] purposeful way." (Pellegrini, 2012, p. 98).

2.5.4 Multimedia / Metadata Life-Cycles

Smith and Schirling associates the *Multimedia Life Cycle* with a central role of metadata, as shown in Figure 2.5.

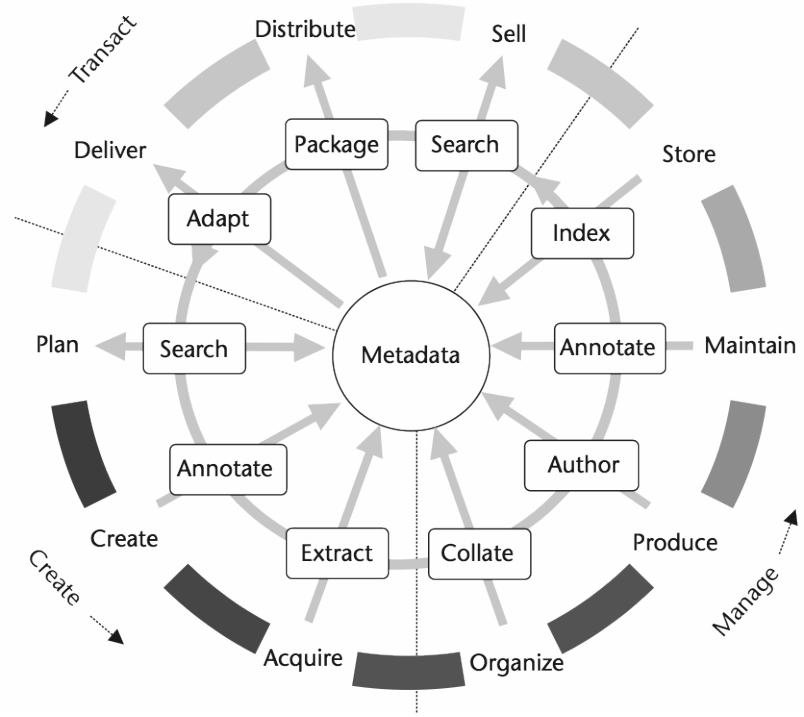


Figure 2.5: Multimedia Lifecycle, (Smith and Schirling, 2006, p. 85).

Thereby, content passes through several lifetime phases: planning, creation, acquisition, organisation, production, maintenance, storage, selling, distribution, delivery. Within

each step, the content lifetime phase interacts with metadata, producing or consuming metadata in several different operations. (Smith and Schirling, 2006, p. 84)

According to Kosch et al., metadata spans *content space*, *metadata space* and *user space*.

The content space includes the steps of content production/creation, postproduction processing, delivery, and consumption. These are augmented by the metadata space, consisting of two parts: the (1) *metadata production* (during or after content production) creates metadata from the content production (like author, actor, date of production) or from the content postproduction (like colour histograms). The (2) *metadata consumption*, where metadata is used in delivery or consumption, e.g. for content filtering. In the User Space, there are three user classes that produce, process or consume content: (1) *Content providers and producers* enrich content by generating globally valid metadata, (2) *processing users* involved in postproduction processing and (3) *end users*, who consume metadata and content. (Kosch et al., 2005, p. 81)

2.5.5 Linked Media Workflow

(Kurz et al., 2011) describes "the typical process in media content pools" as being composed of the following five steps, "including (semantic) annotation, metadata storage, indexing and search" (Kurz et al., 2011, p. 16f).

(1) *Enrichment*, where content is interlinked with "data from the Linked and Open Data cloud" (Kurz et al., 2011, p. 17), (2) *Integration*, where additional data with existing metadata is integrated, (3) *Indexing* the complete or updated parts of the integrated metadata and underlying content, (4) *Publishing* metadata and content, (5) *Interfaces*, where "search and query interfaces" are provided (Kurz et al., 2011, p. 17).

2.5.6 Linked Data Value Chain

The *Linked Data Value Chain* (Latif et al., 2009) applies the concept of a Value Chain (Porter, 1985) to Linked Data business cases, having "[h]uman-[r]eadable data [as] the most valuable output for the targeted [end-user]." (Latif et al., 2009, p. 3). Figure 2.6 on the next page depicts "the process of assigning Linked Data Roles to Entities, modelling

interactions and responsibilities of Linked Data Roles".

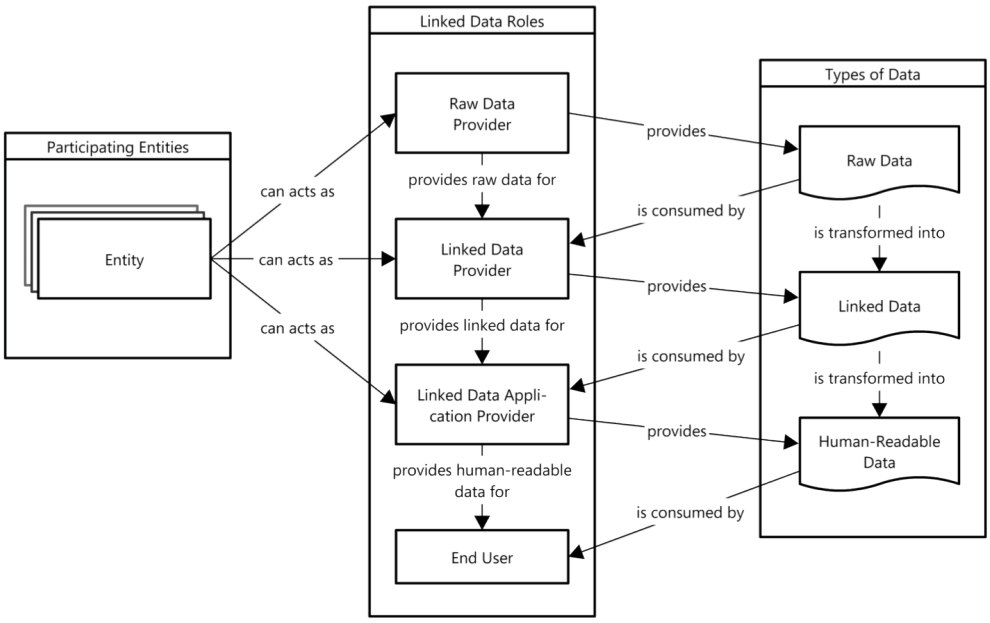


Figure 2.6: Linked Data Value Chain, (Latif et al., 2009, p. 2).

Linked Data participants can thus be assigned to several *Linked Data Roles*, depending on the exact kind of data they handle: *raw data provider*s provide non-RDF data, *Linked Data provider*s provide RDF-data or SPARQL endpoints, the *Linked Data Application Provider*s generates human-readable output for human end users, and an *End User*, who consumes the human-readable presentation of Linked Data, "does not directly get in touch with Linked Data, and typically does not event want to" (Latif et al., 2009, p. 4).

2.6 Data Integration Frameworks

A primary use case of Linked Data is to use data of different sources and domains. This allows information reuse, and can enable a faster application development and the manifestation of open and committed standards (Gray et al., 2014, p. 104).

Complex frameworks have emerged in the last years to support Linked Data integration and application. A selection of these projects is introduced in this section. Thereby, approaches are data centric, having the entity integration (Freytag, 2014, p. 101) of

data as a primary goal.

Integration was considered to be an issue in the early days of the Semantic Web (Goth, 2006, p. 3). More recent sources suggest that Linked Data integration is still immature, e.g. by just focussing on "technical feasibility" (Pellegrini, 2012, p. 97) while ignoring additional aspects that are relevant in practice, like "provenance, reliability or trustworthiness" (Pellegrini, 2012, p. 97). Furthermore, basic technology features are considered "mostly prototyped and thus not usable for industry" (Kurz et al., 2011, p. 17). Many tools come from an academic context, so there is usually no professional support, or steady and secure development; as a result use cases are not adoptable. This leads to poor performance of the tool stacks when using realistic data and to bad usability. (Dirschl and Eck, 2015)

2.6.1 Open PHACTS

Open PHACTS is a Linked Data architecture that integrates multiple pharmacology datasets for drug discovery applications (Gray et al., 2014). Figure 2.7 shows the architecture, consisting of seven layers (Gray et al., 2014, p. 104), which are the result of several subject-specific considerations.

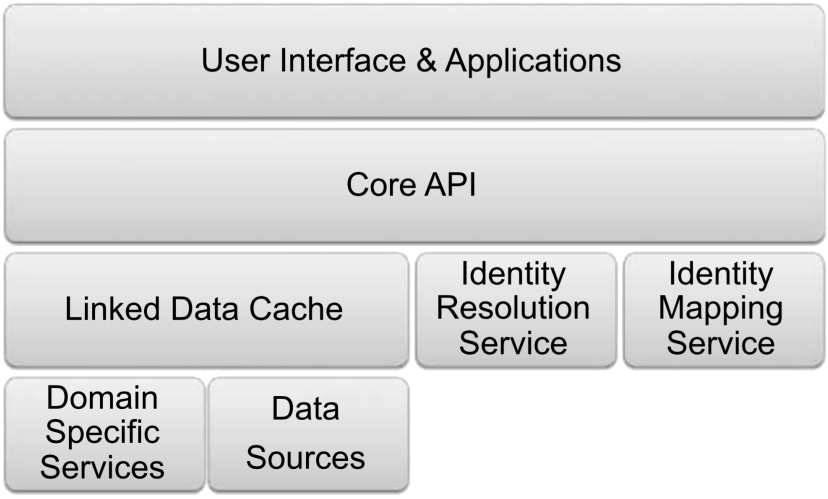


Figure 2.7: Open PHACTS Architecture (Gray et al., 2014, p. 104).

The project is based on Linked Data, which helped it to "quickly develop a working

system" (Gray et al., 2014, p. 104) as well as supplying the Linked Data Providers with a real use case for their data. This data is stored in a warehouse-like environment (Linked Data Cache), a choice explicitly made in order to ensure a stable service operation without third-party dependencies (Gray et al., 2014, p. 105).

Regarding storage, common triple store implementations like BigData, Virtuoso and 4store showed "a number of instability issues" (Gray et al., 2014, p. 108). As a result, Open PHACTS uses the in-memory solution Sesame, augmented by the Identity Resolution Service (IRS) and Identity Mapping Service (IMS) components to address further deficiencies: IRS was introduced to handle free text searches, which are very common in the pharmacology domain (Gray et al., 2014, p. 105). IMS handles URIs that do not exist, for which the triple store would otherwise provide the "worst-case performance when queried" (Gray et al., 2014, p. 108).

Another lesson learned is about the API that Open PHACTS exposes to applications: Early prototypes providing a SPARQL endpoint failed because it required "intimate knowledge of the data exposed" (Gray et al., 2014, p. 106) from application developers and exposed the platform to "poorly formed queries" (Gray et al., 2014, p. 106). This was solved with the introduction of the component *Core API*, which "provides a set of common methods that applications can call" (Gray et al., 2014, p. 106).

2.6.2 WissKI

WissKI is an ontology-based CMS application supporting cultural heritage experts in the storing of their data (Scholz and Goerz, 2012). Thereby, it "encourages the use of local and global controlled vocabularies or thesauri for disambiguation and linkage of data sets." (Scholz and Goerz, 2012, p. 1017).

WissKI is based on the CMS Drupal and can thus "be easily deployed and maintained on a standard web stack configuration" - PHP Hypertext Preprocessor (PHP) and MySQL - because "many CH experts are no computer experts." (Scholz and Goerz, 2012, p. 1017).

The system core uses the "Erlangen CRM, an OWL-DL implementation of the CIDOC CRM (ISO 21127)", which however, "requires a lot of expertise that cannot be assumed for most practitioners" (Scholz and Goerz, 2012, p. 1017). To address this issue, WissKI introduces several abstraction layers, shown in Figure 2.8.

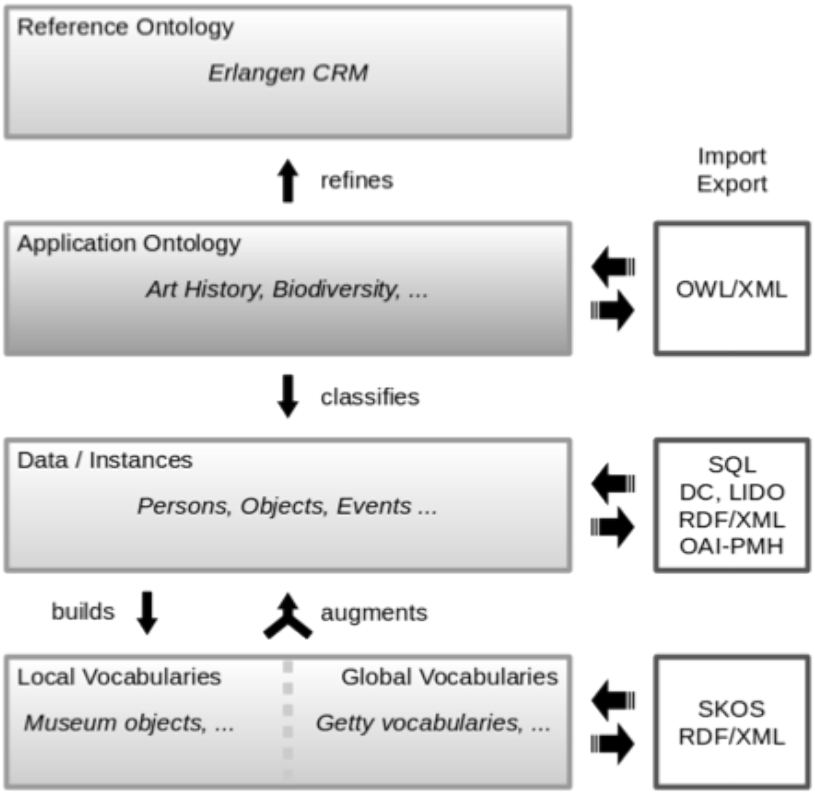


Figure 2.8: Architecture of WissKI (Scholz and Goerz, 2012, p. 1018).

In use, WissKI operates on an optimised subset of Erlangen CRM, the Application Ontology, based on "specific needs" of the application (Scholz and Goerz, 2012, p. 1017). Between the layers, modelling patterns and related transformations are described with Ontology Paths (Scholz and Goerz, 2012, p. 1018). In addition, every layer can be imported and exported in usual formats like Dublin Core and LIDO. This way, information in WissKI can be used universally "for data input, presentation and querying" (Scholz and Goerz, 2012, p. 1018).

2.6.3 OntoBroker

OntoBroker is a Semantic Web middleware, offering capacities for various W3C recommendations such as SPARQL and OWL. It focuses on rule processing and data integration capabilities: Through its connectors, it operates on structured as well as on unstructured data, like relational DBMS, Excel, Search Engines, and more. OntoBroker stores data in quads within the custom storage named extensional database (EDB). (Angele, 2011)

The system can be used for the ontology-based data integration, so "sources such as databases, Web services, Linked and Open Data sources, search engines, and so on, are attached to an ontology" (Angele, 2011, p. 10). Thereby, OntoBroker has several layers and mappings of ontologies, as depicted in Figure 2.9.

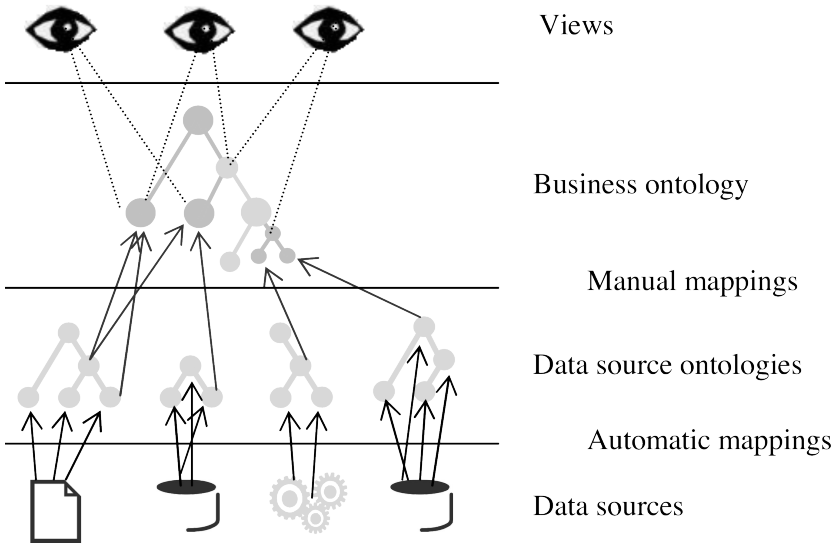


Figure 2.9: OntoBroker Ontology Layers (Angele, 2011, p. 11).

In the lower layers, data from different sources is automatically mapped into Data source ontologies that represent database or Web Services Description Language (WSDL) schemas. Hence, it does "not represent a shared conceptualization of a domain" (Angele, 2011, p. 10). This conceptualisation is first created with the manual mapping in the next layer, resulting in a shared ontology named business ontology that offers an

application specific terminology "relevant to business users" (Angele, 2011, p. 10). In the fourth layer, this business ontology is used through views.

According to (Angele, 2011), OntoBroker customers prefer to use the described ontology-based integration in real-time, corresponding to the previously introduced On-The-Fly-Dereferencing architectural pattern (Section 2.1.4 on page 51).

2.6.4 LOD2 Stack

The *LOD2 Stack* is a set of tools supporting the "whole life cycle of Linked Data from extraction, authoring/creation via enrichment, interlinking, fusing to maintenance." (Auer et al., 2012, p. 1). The project aims at simplifying the "distribution and installation" of components, as well as to "smoothen the information flow" between those components (Auer et al., 2012, p. 4). Together, these components "resemble traditional workflow steps in a publishing house" (Auer et al., 2012, p. 12). Figure 2.10 shows the steps within the life cycle.

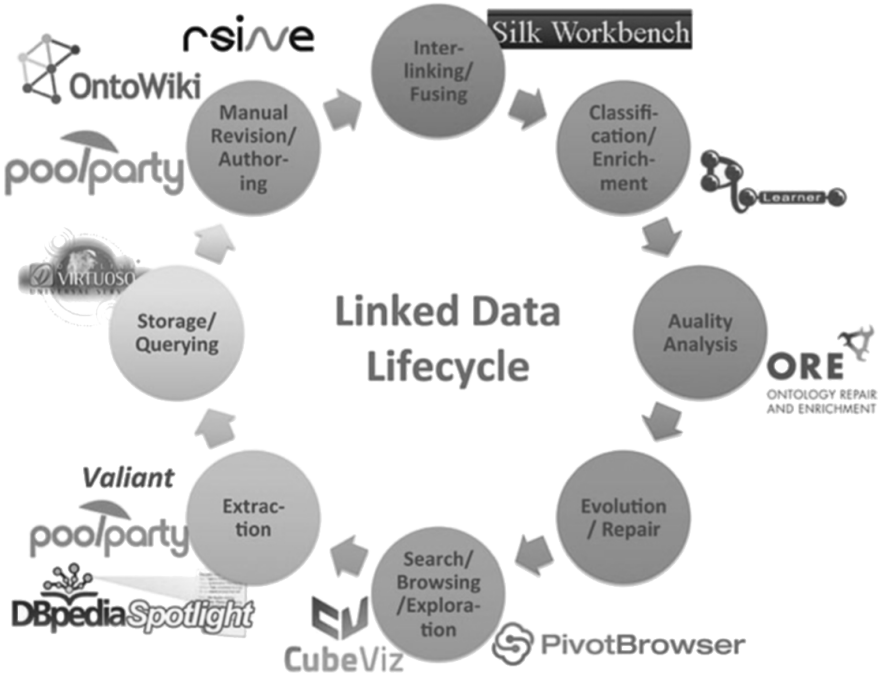


Figure 2.10: LOD2 Lifecycle (Dirschl and Eck, 2015, p. 133).

The LOD2 stack claims to bridge the gap between "between technical partners mainly

coming from an academic world and the requirements of an industrial partner" (Auer et al., 2012, p. 15). For each displayed life cycle phase, the LOD2 stack provides an existing tool or framework, e.g. Virtuoso as triple store (Auer et al., 2012, p. 9). Thereby, some strategies are introduced to address the performance issues (Auer et al., 2012, p. 2).

In particular, the LOD2 Stack comprises of the stages (Thurner, 2014):

- Extraction of RDF from text, XML and SQL
- Querying and Exploration using SPARQL
- Authoring of Linked Data using a Semantic Wiki
- Semi-automatic link discovery between Linked Data sources
- Knowledge-base Enrichment and Repair

2.6.5 Catmandu

Catmandu is a Perl-based backend ETL framework to "import data from various sources, map the fields to a common data model and put it all into a database or search engine" (LibreCat, 2015). Catmandu originates from a library context and thus supports a number of common library standards such as Machine-Readable Catalog (MARC), Metadata Object Description Schema (MODS) and Dublin Core, Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH), Search/Retrieval via URL (SRU) and open repositories such as DSpace and Fedora, and typical Semantic Web formats (LibreCat, 2015).

The framework integrates data in schema-less data stores such as MongoDB and ElasticSearch, each offering "developer friendly APIs" (LibreCat, 2015). To transform imported data, Catmandu provides its own Domain Specific Language (DSL) named *Fix*, which basically consists of Perl-based scripts with convenience methods for data manipulation (Catmandu Fix Language, 2015).

2.6.6 Apache Marmotta

Marmotta is a Linked Data platform from Apache to integrate "content as well as metadata" in storage and retrieval (Kurz et al., 2011, p. 17), aiming at organisations "who want to publish Linked Data or build custom applications on Linked Data." (Marmotta, 2015*a*). It was originally named Linked Media Framework (LMF) and developed by the Salzburg NewMediaLab aiming at bridging the gap between "the common document web and the Web of Data" (Kurz et al., 2011, p. 16).

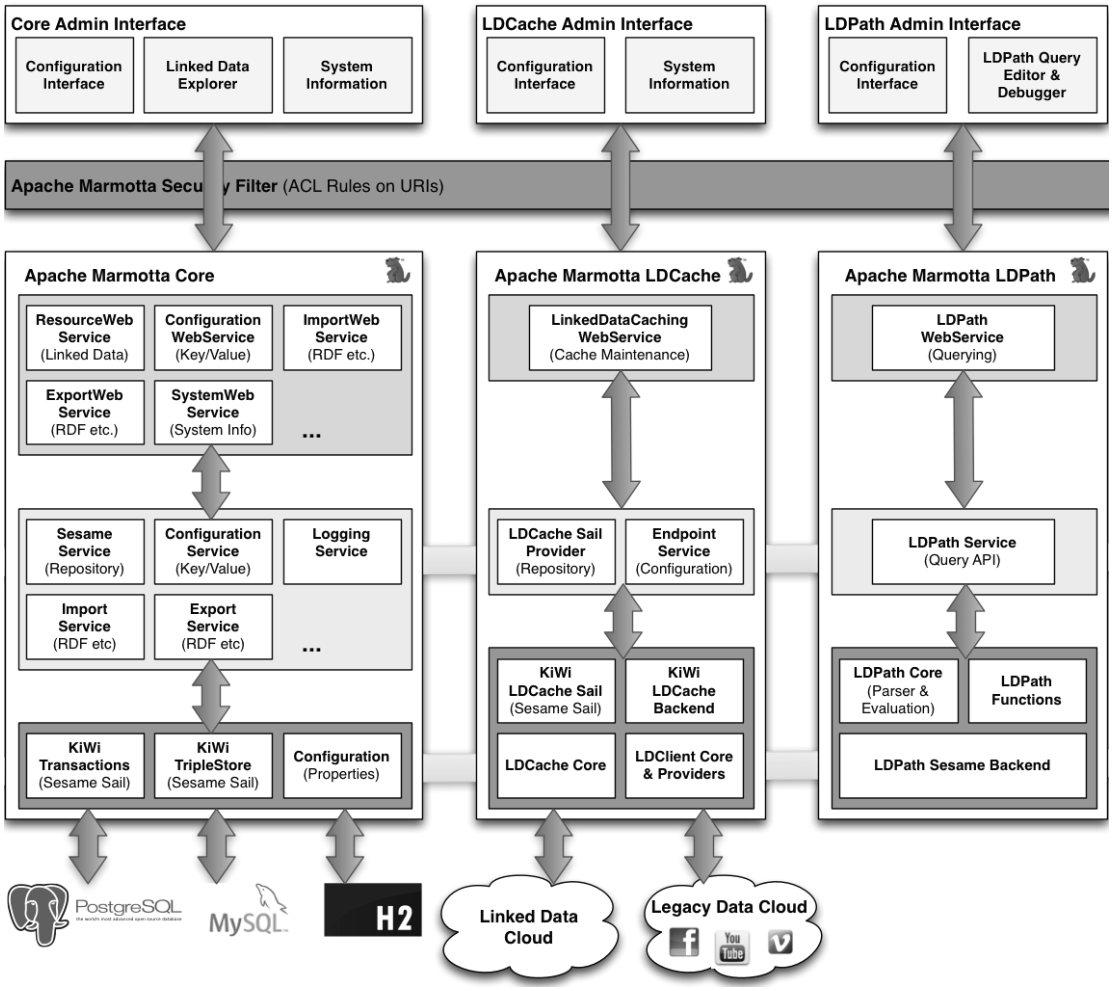


Figure 2.11: Architectural overview of three components of Apache Marmotta (Marmotta, 2015*b*).

Marmotta consists of a layered architecture, shown in Figure 2.11. It offers capabilities to store and model RDF, as well as the option to use it as a Web service in a modern

HTML 5 and JavaScript user interface.

Besides its core components, Marmotta provides an ecosystem for the development of a Linked Data-enabled application. This includes *LDCache*, a transparent caching mechanism for "Linked Data resources" (Marmotta LDCache, 2015), and *LDPath*, a simple path-based query language (Marmotta LDPath, 2015).

2.7 Linked Data Applications

Linked Data applications target end users with certain information needs. Heath and Bizer classify Linked Data applications into "two categories: generic applications and domain-specific applications" (Heath and Bizer, 2011, p. 85).

Generic Linked Data applications can process arbitrary data from a topical domain, like "library as well as life science data" (Heath and Bizer, 2011, p. 86). Examples are usually Linked Data browsers (like LinkSailor) or Linked Data search engines (like SWSE).

In contrast, domain-specific Linked Data applications "cover the needs of specific user communities" (Heath and Bizer, 2011, p. 90). Examples are applications listed on common data portals, such as data.gov or data.gov.uk.

2.7.1 Building Linked Data Applications

To create such applications, Hausenblas describes the steps "needed to exploit linked data sets in an exemplary Web application" as the follows:

Data Preparation

As "the data you're about to use is [typically] available in a non-RDF format", own data must be minted - it must be made "Web-of-Data-compliant" by giving them URIs. Then, existing Linked and Open Data vocabularies should be used and extended, as needed. The last step of data preparation includes the decision how to expose the data, either as stand-alone document (RDF/XML), as metadata (HTML + RDFa) or as SPARQL endpoint. (Hausenblas, 2009, p. 70f)

Linked Data Discovery and Usage

This step involves actions to "find and select target data sets for interlinking [...] to enrich your content". The source suggests to use the follow-your-nose-principle to find additional datasets, a manual and time consuming process. (Hausenblas, 2009, p. 71f)

Reusing data from the Linked and Open Data has two beneficial sides: First, an isolated dataset is enhanced by other data. Second, the service is linked into the Linked and Open Data cloud, which can considered to be a kind of advertisement. He mentions the typical method for doing this is SPARQL.

2.7.2 Search Engines

Semantic Web Search Engines are considered to be the prime sample of a semantic application that "nicely demonstrate the advantages of the open, standards-based Linked Data architecture, compared to Web 2.0 mashups which rely on a fixed set of data sources exposing proprietary interfaces" (Heath and Bizer, 2011, p. 87).

There are two kinds of Semantic Web Search Engines: Sig.ma, Falcons, SWSE, and VisiNav, for example, which are designed to be used directly by humans, as they "follow a similar interaction paradigm as [...] Google and Yahoo" (Heath and Bizer, 2011, p. 87). Today, only Falcons is still operative.

The second kind of Semantic Web Search Engine aims at being used by machines, to "discover RDF document on the Web that reference a certain URI or contain certain keywords." (Heath and Bizer, 2011, p. 89). Examples are Sindice, Swoogle, and Watson. While Sindice has been discontinued (Tummarello, 2014), Swoogle and Watson still work.

2.7.3 BBC

The BBC is considered to be a pioneer of applying Linked and Open Data towards certain business cases (Latif et al., 2009, p. 5), e.g. by aiming at a coherent experience of the webpage (Kobilarov et al., 2009, p. 733) by improving discoverability and brows-

ability of semantic topics across different BBC offerings (Kobilarov et al., 2009, p. 724).

One of the results is the BBC CIS, a catalogisation system that interlinks data items in different services with corresponding DBpedia identifiers (Kobilarov et al., 2009, p. 736). It bases on a Named Entity Recognition (NER) approach for brands, locations, persons and subjects, exploiting DBpedia article redirects in a PageRank-like algorithm (Kobilarov et al., 2009, p. 730).

In addition to that, other sources, such as the unstructured introductions of Wikipedia articles as well as excerpts of the MusicBrainz database, are used to enhance existing resources (Latif et al., 2009, p. 7).

2.7.4 Wolters Kluwer's LOD2 Stack Application

Wolters Kluwer is one of the industry partners of the previously introduced LOD2 Stack project (Section 2.6.4 on page 73). It is one of the first implementors of components of the stack to show "the relevance and usefulness of this technology." (Dirschl et al., 2014, p. 133). Thereby, however, not all LOD2 components are implemented, but a small subset of it (Dirschl et al., 2014, p. 152).

Dirschl et al. centres around the two use cases *converting data into Linked Data* and *enriching this Linked Data*, both implemented with LOD2 stack components. These two application scenarios are augmented by search and retrieval steps, as Figure 2.12 on the next page.

Many tools come from an academic context. There is usually no professional support, or a steady and secure development roadmap. Moreover, use cases are not adoptable to other domains. This leads, for example, to bad performance of the tool stack when using realistic data and to a bad usability. (Dirschl and Eck, 2015)

Figure 2.12 on the facing page also shows that several non-LOD2 stack components are involved before ending up with a working applications. This includes components for IR (Solr), user profiling, and several linguistic processors (Dirschl and Eck, 2015, p. 10).

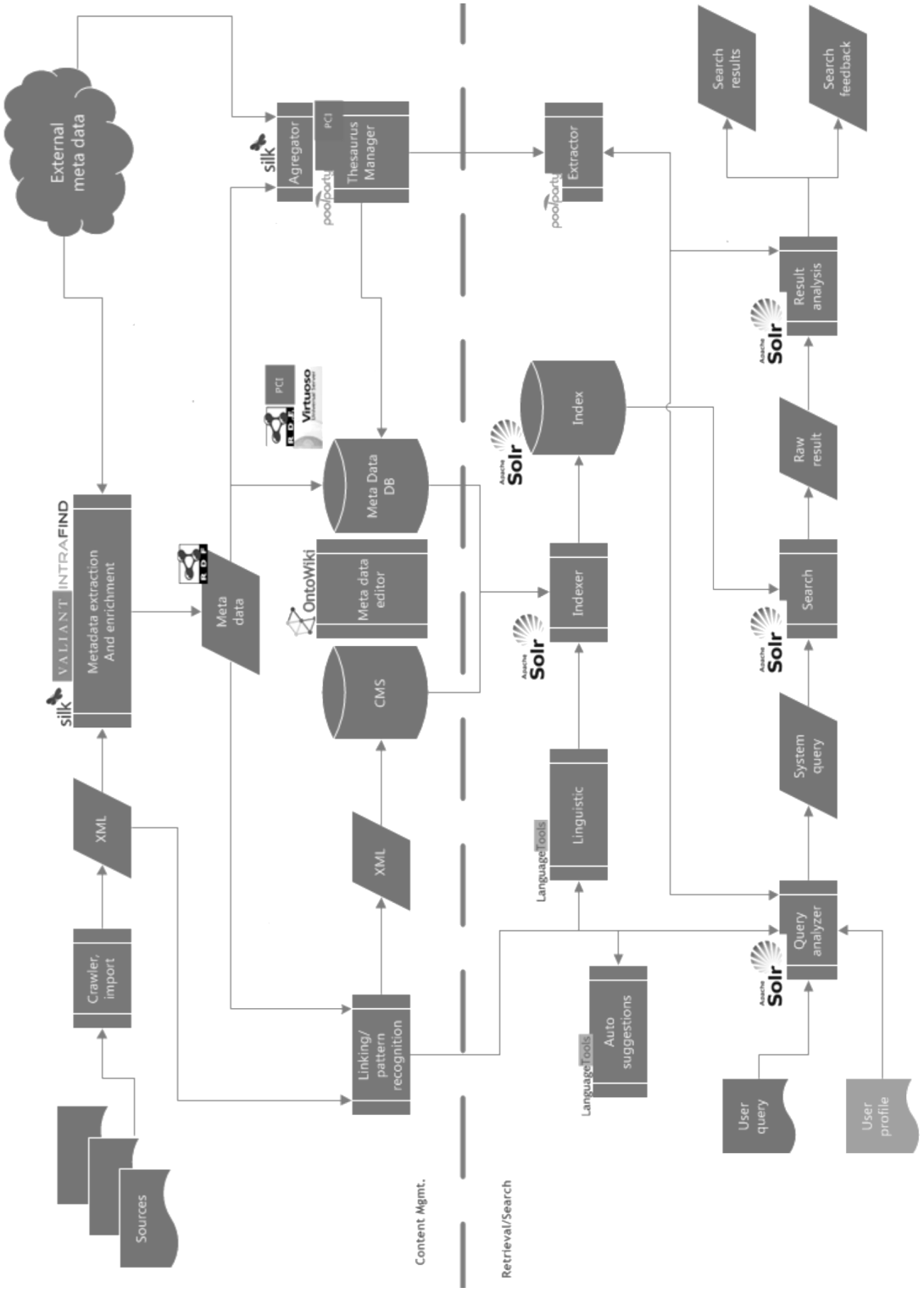


Figure 2.12: Implementation of the LOD2 Stack, (Dirschl and Eck, 2015, p. 9).

2.7.5 KDE Nepomuk

Nepomuk is a semantic desktop environment that was planned to be integrated in the Unix/Linux Desktop Environment KDE. Nepomuk is based on the triple store Virtuoso, and a IR Framework named Strigi (Wolters, 2007).

While the majority of the sources focus on the search use case (Maurhart, 2010), (Ubuntuusers Wiki, 2014), Nepomuk is designed for advanced semantic features, such as task management, semantic mail, semantic file dialog, text recognition in images and a SPARQL endpoint (Mandriva Linux Community, 2012). Thereby, Nepomuk exploits relationships (e.g. "pictures sent by a friend") and contexts (when having separate working environments) (KDE UserBase Wiki, 2015). In addition, it introduces system-wide tags and search capabilities (KDE Nepomuk Manual, 2014).

On December 2012, the desktop integration of Nepomuk was officially announced to be discontinued (Handa, 2013). The announcement also contains experiences that the KDE developers made with the Semantic Web technology stack. This includes statements about RDF ("hard to understand and to optimise") and ontologies ("sub-optimal performance", "not easy to use", "too vague in certain areas") as well as about storage ("not suited for desktop use", "does not allow optimisation for a specific data type") (KDE Community Wiki, 2014).

On the mailing list, further explanations are made about the principle information design architecture problem that RDF causes (Handa, 2013):

> The biggest problem with RDF is that it raises the knowledge needed to contribute to a point where most developers decide to to skip it. After all these years only a handful of brave developers have worked with it and the experience hasn't been good.

In addition, RDF's flexibility hinders efficiency, because data is "generally completely normalised" even though it is not needed in the practice of this semantic desktop application (Handa, 2013).

2.7.6 Cultural Heritage

In 2013, Edelstein et al. conducted a study on 15 Linked and Open Data projects in the cultural heritage domain. Thereby, projects were evaluated with a 6-stage-maturity model, a finer-grained version of Berners-Lee's 5-star model (Section 2.1.3 on page 50) that is more suitable for the cultural heritage practice (Edelstein et al., 2013, p. 2)

Stage 4 in this maturity model analyses projects that develop user interfaces (Edelstein et al., 2013, p. 37) and included the Pan-Canadian Documentary Heritage Network (PCDHN) Video Visualization, the Agora Project, Amsterdam Mobile City App and the Australian War Memorial (AWM). These projects allow interesting insights in Linked and Open Data integration projects.

In at least one project, focus was on producing "papers and conference presentations", while development "did not increase" and the produced prototype application was not considered useful (Edelstein et al., 2013, p. 41). Others had the vision to make Linked and Open Data explorable with user friendly user interfaces (Edelstein et al., 2013, p. 44), but they did not solve the data integration issues: the "retrieval process needed to grab data from multiple sources, parse it, and display it on a users smartphone is cumbersome and slow." (Edelstein et al., 2013, p. 42).

Edelstein et al. conclude that many Cultural Heritage-LOD projects "seem[] to be highly curated and experimental", "remain at the proof-of-concept stage" and that end users "cannot actually access the datasets or interfaces" (Edelstein et al., 2013, p. 60). In addition, not a single project "embodied both the philosophical aims of Linked and Open Data with the technological expertise" (Edelstein et al., 2013, p. 60): Data in cultural heritage domain tends to be more open than linked.

2.7.7 Natural Language Processing

Gangemi describes is a symbiotic relationship between linguistic knowledge and formal knowledge: "linguistic knowledge uses formal background knowledge, but can enable access to formal knowledge (and enrich it) as well" (Gangemi, 2013, p. 352f).

In the last years, NLP research started using Linked and Open Data resources as background knowledge, which lead to application-ready frameworks (Gangemi, 2013, p. 351), e.g. for *Knowledge Extraction* or *Semantic Enhancement*: the approach of automatically creating "semantic annotations via linking to existing resources" into the Linked and Open Data Cloud (Kurz et al., 2011, p. 18). Recent systems are "hybridizing statistical (trained models) and rule-based methods, and taking advantage of existing knowledge from Linked Open Data as well as smart heuristics that cling to all sorts of features and structures that become incrementally available on the Web." (Gangemi, 2013, p. 352).

Many existing frameworks have adopted this approach, including as-a-service-tools like AlchemyAPI, NERD (Rizzo et al., 2012), or DBpedia Spotlight (Mendes et al., 2011), and locally installable platforms like Apache Stanbol (Grisel, 2012) and AIDA (Yosef et al., 2011).

2.8 Development Paradigms

> Only a few years ago a large application had tens of servers, seconds of response time, hours of offline maintenance and gigabytes of data. Today applications are deployed on everything from mobile devices to cloud-based clusters running thousands of multi-core processors. Users expect millisecond response times and 100% uptime. Data is measured in Petabytes. Today's demands are simply not met by yesterday's software architectures. (The Reactive Manifesto, 2014)

In a rapidly changing environment like the Web, development paradigms change as well. They must be respected in applied research in order to provide practical solutions. In the following, an excerpt of current Web development paradigms, best practices and user studies is given.

2.8.1 Speed

Several studies and best practises clearly indicate the importance of speed for Web applications and search interfaces (Mayer, 2009), (Schurman and Brutlag, 2009), (Linden, 2006), (Forrester Consulting, 2009), (Nielsen, 2010).

Speed is one key factor of the search experience (Mayer, 2009). At Amazon, for example, a 100 ms delay results in 1% of revenue drop (Linden, 2006). A study at Microsoft's search engine Bing showed that delays even under half a second have negative business impacts (Schurman and Brutlag, 2009). After 1 second of waiting time, the user's flow of thoughts is interrupted (Nielsen, 2010). The majority of users would leave a webpage, if it is not loaded within 2 seconds (Forrester Consulting, 2009).

2.8.2 Data Binding

The dominating paradigm in which data is visualised or inserted in Web applications changed significantly over the last two decades.

At the end of the 90s, server-side CGI, like Perl-scripts, generated fragments of HTML which just needed to be displayed on the client side with Web browsers. Then, in the middle of the millennium, Asynchronous JavaScript and XML (AJAX) became common. Now, server responses typically only contained data, which was processed on the client side using JavaScript. By this time, plain JavaScript had been extended by abstractions like JQuery, which reduced the manual programming effort. (Möller, 2014, p. 104)

The most recent paradigm is Reactive Programming (Schürmann, 2014), supported by large Web companies with JavaScript frameworks like *Angular* (Google) and *React* (Facebook). Using this paradigm, view components are bound to certain entities in the response, which significantly reduces manual programming effort even further, as all view components are updated automatically as soon as data is updated. (Möller, 2014, p. 104)

2.8.3 Polyglot Persistence

To meet today's demands of high usability and low maintenance outages, services employ the *Polyglot Persistence* paradigm: the proper storage technology is selected for each use case individually (Edlich et al., 2011, p. 381). This is amplified by the fact that today's data management scenarios do not match to the capabilities conventional relational DBMS offer (Halevy et al., 2006, p. 27).

In the context of *NoSQL* database management systems, quite a few new possibilities emerged to store data in such specific scenarios, each having certain strengths and weaknesses (Edlich et al., 2011). These are usually classified into the four groups *wide column stores*, *document store*s, *key/value stores* and *graph stores* (Edlich et al., 2011), (McCreary and Kelly, 2013).

In contrast to relational DBMS, there is no standardised query language for all NoSQL databases (Ong et al., 2014, p. 1). Interfaces to the stored data are usually designed to be used directly by applications developers, through dedicated APIs or Representational State Transfer (REST) Web services. Some NoSQL databases are considered to be specifically Web developer friendly, such as document stores like MongoDB or CouchDB (McCreary and Kelly, 2013, p. 6). Particularly being able to work with MongoDB was an important design goal in the specification of JSON-LD (Franzon, 2014*a*).

In addition, NoSQL databases can be used complementarily to existing storage forms (Nimis et al., 2014), (Müller, 2014).

2.8.4 New Web Development Approaches

Web development paradigms are constantly developed and adapted to current challenges.

Offline First and *noBackend*, for example, are two initiatives that try to address the issue that current "technology and best practices [in Web development] are a leftover from the always connected & steadily powered past" (offlinefirst.org, 2014). As a result, they do not respect the limitations of a "disconnected & battery powered world" (offlinefirst.org, 2014).

Thus, Offline First tries to establish GUI conventions and architectural patterns to handle offline-ness. In contrast, noBackend tries to "decouple apps from backends, by abstracting backend tasks with front-end code" (noBackend.org, 2014). The JavaScript-like result is called *Dreamcode* and sums up the functionality Web developers need, in a manner that matches to their workflow and design goals.

Today, not only storage might be polyglot, but also the programming environment: Web services can be programmed in quite a number of different languages, like Ruby, Go, Python, Clojure etc. Thereby, *Microservices* become important. Microservices are use-case specific but open REST APIs, that allow complex services to be constituted by a multiplicity of different implementations (Borsje, 2014).

2.9 Summary of Findings

Based on the conducted literature research, in the following, the essence of the references is grouped by topics, compared and classified.

2.9.1 Entity-Agnostic RDF versus Developer APIs

In application-specific scenarios, data forms logical entities that can be found as homogeneous data models (Morsey et al., 2011, p. 466f). On the client side, these entities are subject of today's Web development paradigms, such as data-binding. An ecosystem of convenient tools has emerged (Möller, 2014), (Schürmann, 2014), allowing the developer to specify which kind of information to be displayed in which component (Section 2.8.2 on page 83). This requires a certain perception of the data and the logical entities they form. APIs as well as database queries will usually have to reply with data conforming to this perception.

RDF, however, is designed to be entity-agnostic (Isaac et al., 2005). As such, entities must first be assembled from several atomic RDF statements in order to create an entity-aware view on the data. In case of application-specific scenarios, the flexibility provided by RDF is not only not needed, it is in fact a complication and undermines optimisation efforts as well as query capabilities (Handa, 2013), (Zaino, 2013).

This dilemma is not solved by new serialisation formats of RDF 1.1 (Wood, 2014), allowing Linked and Open Data to be serialised in "developer friendly" formats like JSON-LD (Sporny et al., 2014) - the fundamental data model of RDF remains the same.

Similar experiences were had by a number of projects, with Open PHACTS (Gray et al.,

2014) and Nepomuk (Handa, 2013) leading the way: application developers were not able to use the originally exposed SPARQL endpoint properly, e.g. because it required "intimate knowledge of the data exposed" (Gray et al., 2014, p. 106).

Across different approaches in different domains, it can be observed that the Linked and Open Data integration projects provide additional capabilities to work with application-specific entities or terminology: Open PHACTS (Gray et al., 2014) introduced a dedicated API ("Core API") to offer developers a application-specific view on the data. WissKI (Scholz and Goerz, 2012) and OntoBroker (Angele, 2011) map external ontologies into internal representations ("Application Ontology", "Business Ontology"), which represent data in application-specific needs (Angele, 2011, p. 10), (Scholz and Goerz, 2012, p. 1017).

2.9.2 Storage in Triple Stores

Previously, benchmarks for triple stores are introduced: LUBM, SP^{2}Bench, and BSBM simulate an application-specific scenario and evaluate usual queries associated to that scenario. The fact that this results in homogeneous data structures was subject of criticism (Morsey et al., 2011), which led to the the design and implementation of DBPSB, claiming to be more realistic by applying heterogeneous data structures. Furthermore, LDBC SPB was introduced.

According to the definition of Heath and Bizer however, both domain-specific as well as generic approaches are valid application scenarios of Linked and Open Data (Heath and Bizer, 2011, p. 86), p. 90, thus all four benchmark scenarios are valid as well.

A principle problem is the fact that a comprehensive comparison involving all current storage technologies is missing.

Putting all results together, it seems that there is not a single flawless triple store that could be recommended for any scenario. Discounting LUBM from the year 2005, serious issues like incomplete SPARQL support (Bizer and Schultz, 2011), timeouts (Schmidt et al., 2009, p. 230), wrong results (Schmidt et al., 2008, p. 11), and other

technical problems (Bizer and Schultz, 2011) could be observed in all popular triple store implementations: Sesame, Virtuoso, SDB, Redland, TDB, BigOwlim, BigData and 4store.

Because the benchmarks do not define a minimum performance requirement for their respective scenarios, it is not possible to judge if certain triple store performances are suitable or not. Some findings, however, can be concluded from these plain numbers, too: the loading and execution times vary significantly between individual triple store implementations, from a few seconds to minutes loading time in the simplest case (Bizer and Schultz, 2009), p .19. Even in the smallest datasets, query executions usually took more than one second (Schmidt et al., 2009, p. 232).

Early versions of the BSBM contained relational database approaches, too, which revealed significant performance discrepancies: native SQL engines outperformed native RDF engines by a factor of 8.5 (Bizer and Schultz, 2009, p. 26). Unfortunately, newer benchmarks of BSBM no longer include non-RDF-stores.

Practical experiences with the technology seem to confirm these observations: in Open PHACTS (Gray et al., 2014) BigData, Virtuoso and 4store showed "a number of instability issues" (Gray et al., 2014, p. 10) and, as a result, could not be used for storage. The solution was Sesame-based, but additional components had to be implemented manually to by-pass some of its shortcomings (Gray et al., 2014, p. 105ff).

In the creation of RDF datasets, it is not uncommon that the triple store technology is not used at all, as demonstrated by one of the largest Linked and Open Data contributors, the OCLC, created their dataset entirely without the triple store technology, but with Hadoop (Wallis, 2014).

It can be observed that recent approaches try to address triple store performance issues by scaling them up using NoSQL or cluster computing technology (Auer et al., 2012), (Khadilkar et al., 2012), (Sun and Jin, 2010), (Larkou et al., 2013), (Stadtmüller et al., 2013).

Promising solutions exist, but it is uncertain if they address existing performance gaps towards relational approaches. Some, in addition, might complicate the technology stack even further. Based on available data, however, more conclusions regarding triple stores cannot be made. Recent comparisons are missing.

2.9.3 RDF-centred Architectures versus Application-specific Architectures

Taking the previous two findings into consideration, the fact that (1) RDF is not a suitable data model for application developers and (2) there are unanswered questions in the storage of Linked and Open Data in application-specific scenarios, existing reference architecture do not seem to represent the necessities of application-specific scenarios. This includes the technological choices and constraints they define, e.g. in the employment of triple stores and the perception that the use of RDF and SPARQL is mandatory (Heath and Bizer, 2011, p. 97ff) or, at least, something to aim for (Berners-Lee, 2006).

Obviously, the differentiation between "generic [LOD] applications and domain-specific [LOD] applications" (Heath and Bizer, 2011, p. 85) results in fundamentally different architectures - a fact that is not respected in current research and that would explain why existing architectures do not guide application developers (Edelstein et al., 2013, p. 4).

According to Berners-Lee, Linked and Open Data does not necessarily involve using the RDF format. In reality, the situation is more diverse: "[d]ata may be linked but not open, or open but not thoroughly linked" (Edelstein et al., 2013, p. 2). Exact numbers are not available and there are no current works that compare the quantity of available Linked and Open Data in regard of their format. However, recent work (Assaf et al., 2015) might provide means to do so. According to existing paradigms, Linked and Open Data in non-RDF-formats would first need to be artificially upgraded to RDF, "minted", (Hausenblas, 2009, p. 70f) - a process that must be considered to be error-prone (Latif et al., 2009, p. 7).

By selecting the architecture components based on necessities of the application-specific scenario, thus allowing Linked and Open Data applications to be RDF-free in their core

functionality, optimisation steps inherent in these individual scenarios become possible. These optimisation steps are in fact mandatory in the reality of today's developers, who have to deal with mobile devices, limited connectivity and low bandwidths (The Reactive Manifesto, 2014).

2.9.4 Essential Steps Not Covered by Existing Architectures

Conceptually, data integration processes form a linear pipeline of similar steps, where given input data is stepwise consolidated, enriched, evaluated, reprocessed, prepared and delivered (Fayyad et al., 1996), (Pellegrini, 2012), (Smith and Schirling, 2006), and in addition they contain steps for the user space (Kosch et al., 2005).

By comparing these logical integration activities with actual Linked and Open Data integration approaches, such as the LOD2 Stack (Section 2.6.4 on page 73), or architectures, such as the Crawler-Pattern (Section 2.1.4 on page 51), significant differences become obvious.

In Wolters Kluver's implementation scenario of the LOD2 Stack, the architecture of the semantic search is a linear pipeline (Dirschl and Eck, 2015, p. 9). Only half of the components are selected LOD2 Stack components, the other half are auxiliary components, like Solr, that are not part of the LOD2 Stack at all (Section 2.7.4 on page 78). This second part however is key to the main end-user application.

Obviously, in practice, architectures include components to cover steps for application-specific preparation and delivery of data. These steps do not seem to be part of common models or reference architectures in Web Science. Similar observations can be made in BBC's Linked and Open Data project, for example, where the classical Linked and Open Data technology stack is augmented with information retrieval technology (Kobilarov et al., 2009, p. 730), or in Open PHACTS (Gray et al., 2014), where non-Semantic-Web components are introduced to streamline the API usage.

Just by following Linked and Open Data principles, reference architectures or life-cycle models, projects do not necessarily end-up with working Linked and Open Data appli-

cations. As a result, this finding supports the claim of the previous finding, that the lack of architectures needs to be addressed. New conclusion is that this conceptual framework for application-specific scenarios needs to cover the conceptual data integration steps that are needed in practice.

2.9.5 Warehousing Data

When humans browse the Web, information is dereferenced and retrieved on the fly. Among others, this paradigm for information retrieval has been transferred from the Web for humans to Linked and Open Data for machines and is called the On-The-Fly Dereferencing Pattern (Heath and Bizer, 2011, p. 97). A similar pattern exists with the Query Federation Pattern (Heath and Bizer, 2011, p. 98).

Following these patterns, architectures rely on external resources during their online operation. OntoBroker supports this with a feature called "run-time ontology-based information integration" and claims that "customers usually prefer" this feature (Angele, 2011, p. 10).

According to available studies, relying on Linked and Open Data Cloud resources on the fly must be considered to be at least risky: In December 2014, the Linked and Open Data Stats project (Demter et al., 2012) indicated issues with over 2/3 of the analysed datasets (Webpage, 2014). This does not include data qualitiy issues, like DBpedia's known problems with ambiguities (Houle, 2014) and incompleteness (Neubert and Tochtermann, 2012). Moreover, about 3% of URIs become invalid per year (Nelson and Allen, 2002). This link rot phenomenon can also be observed for links in publications, whereas up to 80% percent of the referenced resources do not exist anymore (Klein et al., 2014, p. 25).

Using the On-the-Fly-Dereferencing pattern also means that all data must be 5-Star Linked and Open Data (Berners-Lee, 2006) - RDF that references to external resources. In practice however, "[d]ata may be linked but not open, or open but not thoroughly linked" (Edelstein et al., 2013, p. 2). This data would first needed to be upgraded to 5-Star Linked and Open Data. While there exists a number of approaches (Kurz et al.,

2011), (Rodriguez and Gómez-Pérez, 2006), (Calvanese et al., 2011), (Apache Any23, 2013), (PublicData.eu, 2014), they are error-prone (Latif et al., 2009, p. 7).

In contrast to this, the Open PHACTS project (Gray et al., 2014, p. 105) provides a pragmatic solution for these issues, as "third party services [...] cannot and should not be relied upon to provide consistent access". Thus, the project explicitly decided to pursue a data warehouse approach for handling Linked and Open Data.

Another aspect of warehousing data is the fact that the Open World Assumption (Allemang and Hendler, 2011, p. 252) is no longer a constraint. When working with online data, new information might always come up and thus invalidate queries in the moment they are fired. When storing data locally, however, the exact moment of data import can be chosen intentionally and can be part of the system architecture. Applications working with this local data then can assume that their view on the data is complete, even if there might be new data online. The *Closed World Assumption* ultimately allows mathematical operations on data again, which would otherwise not have been possible with the Open World Assumption. For example, when using Linked and Open Data, NLP approaches are "hybridizing statistical [...] and rule-based methods, and taking advantage [...] of smart heuristics" (Gangemi, 2013, p. 352)

2.10 Conclusions

A number of open questions remain when trying to implement Linked and Open Data-based, domain-specific applications.

The Semantic Web technology stack is considered mandatory according to existing reference architectures, but is conceptually less suitable for the entity-aware scenarios of specific applications. This results in complex infrastructures and suboptimal abstraction layers for developers. Given the fact that it is unclear how much Linked and Open Data is actually available in RDF, and how much of it uses other data formats, it is unclear if the effort of using a RDF-centred tooling is actually justified.

Existing benchmarks cover a whole spectrum of application scenarios, but are artificially

limited to triple stores, even though the polyglot persistence paradigm (Section 2.8.3 on page 83) is common. Possible pitfalls such as indicated performance gaps have not been investigated any further by current research.

Hence, the rules of existing reference architectures do not only not guide developers - they can even be considered to be misleading for them. Obviously, specific scenarios require a fundamentally different architecture, that supports the entire data integration process and ends up with working end-user applications.

Based on the available state of research, a need for pragmatic architectures for the integration of Linked and Open Data in application-specific scenarios can clearly be identified.

Chapter 3

Research Approach

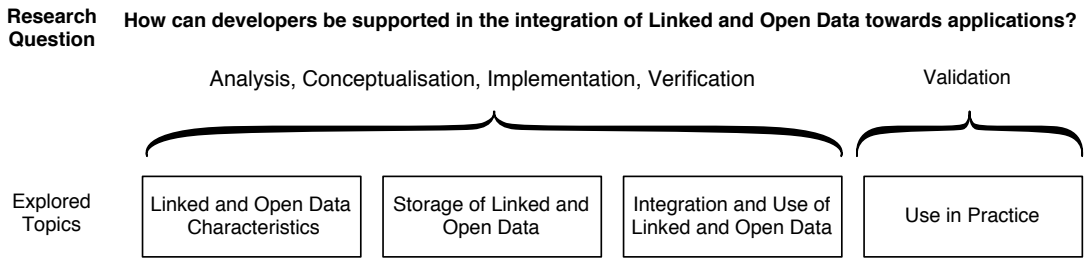


Figure 3.1: Research Question and explored topics with its context.

In the preceding chapters, it is described that the lack of Linked and Open Data applications can be derived from the lack of suitable Linked and Open Data reference architectures and frameworks for specific scenarios. To address this deficit, a pragmatic framework that supports the logical steps of data integration is designed. In contrast to existing work that might present architectures for generic Linked and Open Data applications (Heath and Bizer, 2011), the framework described here should guide developers to create application-specific semantic solutions based on Linked and Open Data. To do so, four explored topics have been identified, shown in Figure 3.1.

The subject of this endeavour is an information system. Hence, the conceptual framework for understanding, executing and evaluating information systems of Hevner et al. can be applied. In the following, a set of general, scientific methods of Hevner et al. is chosen to match the topics of the research question (Section 1.3 on page 45). Based on that, the exact application of these methods is defined. This involves the development of analysis algorithms and tools.

3.1 Research Methodology

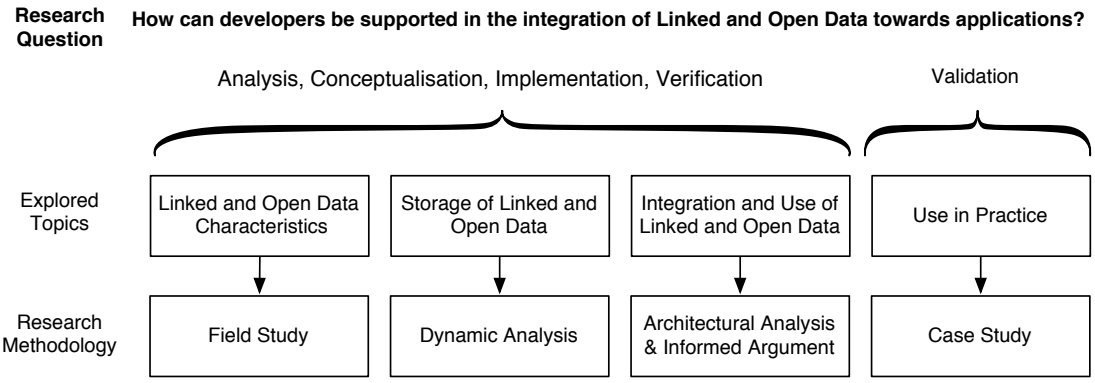


Figure 3.2: Research Question, explored topics and selection of appropriate scientific methods. Based on that, Figure 3.3 on page 96 shows in detail how the methods are applied.

The methodology of the present work is based on a subset of the available methodologies from design science (Hevner et al., 2004, p. 86). In the following, these are associated to the corresponding topics of the research question (Section 1.3 on page 45) to design a framework for the integration and use of Linked and Open Data in specific scenarios. As shown in Figure 3.2, this involves five methods.

3.1.1 Field Study

To design the system realistically, it is crucial to understand basic characteristics of Linked and Open Data, such as data formats, licences, the presence or absence of structures, and availability.

To observe and analyse multiple Linked and Open Data publication projects systematically, a field study is an appropriate method (Hevner et al., 2004, p. 86). It identifies the nature of Linked and Open Data on a wide scale independent from individual projects or domains, and thus forms a new foundation for proposing a real-world solution in subsequent research.

3.1.2 Dynamic Analysis

Storing and querying data is essential when building applications. To find the most suitable storage solution for Linked and Open Data, findings of the field study as well as a broad spectrum of current storage choices, including relational DBMSs, triple stores, and NoSQL systems must be considered.

When comparing different approaches, performance is a primary criterion. A suitable analytical method is a dynamic analysis (Hevner et al., 2004, p. 86). A broad-scale evaluation provides critical information to developers and closes gaps that are not answered in current research (Section 2.4 on page 59).

3.1.3 Architecture Analysis and Informed Argument

In the design and implementation of a new framework, the logical steps to integrate data into applications need to be identified. This is based on existing research and best practices. A suitable method to do so is an informed argument (Hevner et al., 2004, p. 86). In addition, the findings of the field study and dynamic analysis might set technological constraints that need to respected. This will be incorporated into the research using an architectural analysis (Hevner et al., 2004, p. 86).

Using both methods, a prototypical framework is developed that incorporates current research, operational practice and technical necessities.

3.1.4 Case Study

The developed prototype is evaluated in the context of a realistic setup. The utility is shown in a proof-of-concept application.

An observational method to study the in depth-behaviour of a concrete business environment is a case study (Hevner et al., 2004, p. 86). Based on measurable metrics, a case study shows how useful the previously developed framework is.

3.2 Research Design

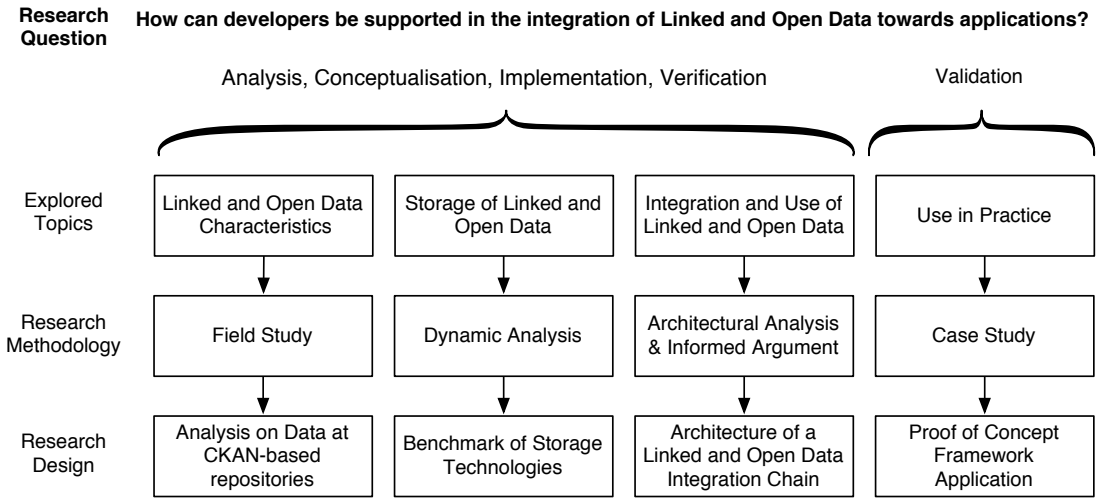


Figure 3.3: Research Question, explored topics, selected methods and the detailed research design of the present thesis. Based on this, Figure 4.1 on page 121 shows the developed artefacts.

The previous section 3.1 on page 94 describes the scientific methods that were selected to answer individual topics of the research question. In this chapter, it is described how these selected methods are applied to the specific endeavour of this thesis: the development of a Linked and Open Data integration framework. Contributions are:

- Definition of a Field Study on Data at CKAN-based repositories
- Specification of a Benchmark of Storage Technologies, on page 106
- Architecture of a Linked and Open Data Integration Chain, on page 114
- Definition of a Proof of concept Framework Application, on page 118

3.2.1 Field Study on Data at CKAN-based repositories

In the following, a new method for creating a field study on data at CKAN-based repositories is developed. It is designed to consist of three parts. Firstly, the meta-information about the data available at datahub.io is extracted and stored in a CSV file. Secondly, the CSV file is loaded, interpreted and analysed. Based on this metadata, information about formats, popularity, licences, and ages of the datasets can be acquired. Thirdly, based on the metadata, the highest rated RDF-based resources are attempted to be downloaded, imported, and analysed.

The exact analytical process is described in the following. For demonstration purposes, the third part uses excerpts of the New York Times Linked Open Data dataset "People" (Datahub.io, 2010) as an example for the analysis conducted. All scripts, queries, program source code and results are attached in Appendix A on page 249, as well as being published in the GitHub repositories CKANstats[1] and LODprobe[2].

Extraction

Like many other data portals, datahub.io is based on CKAN, offering an open, REST-based, JSON-based API (Datahub.io, 2015). In CKAN-terminology, the actual data is published as a resource, and one or more resources are provided in units named datasets (Open Knowledge Foundation, 2013*c*). A Python script named *CKANstats.py* uses the CKAN-API (Version 3) (Open Knowledge Foundation, 2013*a*) and extracts the following meta-information about the datasets registered at datahub.io (Open Knowledge Foundation, 2013*b*):

- Dataset name + id
- Dataset licence name + id
- Boolean flag if the dataset is openly licensed
- The dataset's page views (total + last 14 days)

[1] `https://github.com/heussd/CKANstats`, last access on 2015-06-09.
[2] `https://github.com/heussd/LODprobe`, last access on 2015-06-09.

- Resource name + id
- Resource creation + revision timestamp
- Resource's format
- The resource's download Uniform Resource Locator (URL)
- The resource's page views (total + last 14 days)

The script stores the retrieved meta-information in a CSV file[3].

Metadata Analysis

In order to conduct further analysis with SQL, the first step is to load the CSV file into PostgreSQL (Version 9.4.1) using `COPY datahubio FROM 'datahubio.csv' DELIMITER ',' CSV;`. Unfortunately, values found in format column are not unified. For example, there are at least 29 different notations given for specifying a Microsoft Excel resource. Another issue with this column is the fact that in about 20% of the cases, there is no format specification at all.

To address these issues, a generic mapping table has been manually created. It assigns the various source values stated in resource to a unified format definitions, as shown in Table 3.1.

Expression	**Unification**
`application/zip+vnd.ms-excel`	Spreadsheet
`application/zip+application/vnd.ms-excel`	Spreadsheet
`data file in excel`	Spreadsheet
`microsoft excel`	Spreadsheet
`excel`	Spreadsheet
`application/x-excel`	Spreadsheet
`format-xls`	Spreadsheet

Table 3.1: Sample for the format unification.

[3] `https://github.com/heussd/CKANstats/blob/master/datahub.io/datahub.io.csv`, last access on 2015-06-10.

Based on the table holding the imported data, a database view is created using this mapping table twice:

- Firstly, the format definition of datahub.io is translated via SQL-like-patterns `left outer join mptbl as a on lower(trim(resource_format)) like lower(a.expr)`.
- Secondly, for every remaining format unknown, it is attempted to join the mapping table an additional time based on the last characters of the resource URL - `left outer join mptbl as b on (a.format = 'n/a' and lower(substring(trim(resource_url) from '...$' )) like b.expr)`. So if, for example, a resource has a URL pointing at "example.com/filename.pdf", this is an indication for the file format Portable Document Format (PDF).

Both joins produce a best-effort corrected view on the metadata extracted. Thus, further analysis is enabled based on this view, and the SQL scripts developed are documented online[4].

In-Depth Analysis of RDF-based Resources

The in-depth analysis is conduced for every RDF resource which has ever been visited, ergo having a `resource tracking summary total` larger than 0. At the time of the described metadata extraction this included 606 resources.

In a semi-automated process, each single resource is downloaded using GNU Wget (Version 1.15) and loaded into an empty Apache Fuseki (Version 2.0.0 2015-03-08T09:49:20, Xmx set to 14.240M) using s-put and, if that fails, using Fuseki's Web front end. Errors during this process are logged as follows:

Not Found Wget could not download the resource, either because it was no longer available or the connection timed out.

[4] `https://github.com/heussd/CKANstats`, last access on 2015-06-02.

Parse Error Fuseki failed to load the downloaded resource using s-put and Fuseki Web.

Partly Some files of the resource were loaded, others not. Evaluation is done with the loaded files only.

Out of Memory Fuseki failed to load the resource and reported an out of memory exception or a garbage collection overhead exception.

This process is documented for all 606 resources[5]. Of these, the resources that could successfully be loaded into Apache Fuseki are the foundation for the subsequent analysis[6].

The Java tool called *LODprobe*[7] has been specifically developed for this field study to analyse the inner structure of the RDF-based datasets. Once a dataset is entirely loaded in an empty local Fuseki dataset, LODprobe fires a number of SPARQL queries against the default graph.

As a result, quantities about several basic characteristics of the resources are extracted:

- The number of unique RDF subject identifiers.
- The number of occurrences of each RDF property the default graph contains.
- The number of co-occurrences of two RDF properties, considering every property with each other.

The result is a symmetrical matrix of co-occurrences, accompanied with individual property counts, as excerpted in Table 3.2 on the facing page. For example, the value 4979 in the second row (starting with [0]) and column [1] shows that the RDF property `http://creativecommons.org/ns#attributionName` ([0]) co-occurs with the RDF property `http://creativecommons.org/ns#attributionURL` ([1]) in 4979 subjects.

[5] `https://github.com/heussd/CKANstats/blob/master/datahub.io/all_ld_rdf_res_tracking_sum.csv`, last access on 2015-06-02.

[6] The author notices that in some cases, RDF dumps could be loaded using s-put, but not via Web front end, and in others vice versa.

[7] `https://github.com/heussd/lodprobe`, last access on 2015-06-02.

Number of unique subjects: 9958	**Count**	**[0]**	**[1]**	**[2]**	**[3]**	**[4]**	**[5]**
[0] cc:attributionName	4979	-	4979	4979	0	0	0
[1] cc:attributionURL	4979	-	-	4979	0	0	0
[2] cc:license	4979	-	-	-	0	0	0
[3] nyt:associated_article_count	4979	-	-	-	-	4281	4281
[4] nyt:first_use	4281	-	-	-	-	-	4281
[5] nyt:latest_use	4281	-	-	-	-	-	-

Table 3.2: Excerpt of LODprobe analysis result for the New York Times Linked Open Data dataset "People".

Number of unique subjects: 9958	**[0]**	**[1]**	**[2]**	**[3]**	**[4]**	**[5]**
[0] cc:attributionName	4979	4979	4979	0	0	0
[1] cc:attributionURL	4979	4979	4979	0	0	0
[2] cc:license	4979	4979	4979	0	0	0
[3] nytdata:associated_article_count	0	0	0	4979	4281	4281
[4] nytdata:first_use	0	0	0	4281	4281	4281
[5] nytdata:latest_use	0	0	0	4281	4281	4281

Table 3.3: Excerpt of the previous Table 3.2, loaded, mirrored, and converted as an R matrix object.

By considering the co-occurring values of two RDF properties row- or column-wise, further insights into the structure of the resource can be gained. For the example above, the properties [0], [1], and [2] always co-occur. This also holds true for the properties [4] and [5], which all co-occur as well. Both property groups seem to be part of distinct entities, as [0], [1], and [2] never co-occur with [4] and [5], as indicated by the zero values in the matrix. Thus it may be concluded that this sample data contains two entities.

In such obvious cases, the result of a LODprobe analysis contains about 20x20 RDF properties with few, clearly identifiable entities. Usually, however, there are more entities, less clear co-occurrences, and/or many more properties.

In a next step, a large-scale analysis of the individual LODprobe outputs is conducted using the scriptable statistics software R. A co-occurrence diagonal matrix from the CSV-files is loaded, converted into a numerical matrix and mirrored in a symmetric one. Individual property-counts are moved into the diagonal. Table 3.3 shows this for

the chosen example.

Having the LODprobe results available as R objects allows for further advanced analysis in consecutive order: calculation of the dissimilarity matrix of the LODprobe matrix, followed by a cluster analysis of the dissimilarity matrix. From this cluster analysis, a number of metrics for the structured-ness of the individual resources is extracted. This metrics-based analysis is augmented by generating visualisations of the clusters detected.

Follow-up analytical steps are:

Dissimilarity Calculation In this step, based on the counted co-occurrences, the dissimilarity of the property-pairs in the matrix is calculated. The work showed that LODprobe results usually contain more zero values than non-zero. Therefore, a non-euclidean distance metric has been applied. This field study uses the Gower Distance (Gower, 1971) by utilising the R function `daisy` from the `cluster` package, using `metric = "gower"`.

Cluster Analysis In this step, based on their mutual (dis-)similarity calculated previously, groups or clusters of the properties are searched for. Thereby, the complete linkage method (Wikipedia, 2014) is used (via R's `hclust` using `method ="complete"`), so resulting cluster analyses are usually scaled from a minimum height of 0 to a maximum height of 1.

Table 3.4 on the facing page shows the metrics that are calculated for all LODprobe results for comparison purposes.

They characterise a RDF resource: in the case of New York Times Linked and Open Data "People" dataset, judging by the minimum and maximum height of the analysis, the dissimilarity of groups of properties in the dataset is very high - it is obvious that the RDF resource contains different entities.

Upon considering a number of cluster analyses of different RDF resources, the grouping

Metric	Sample
LODprobe analysis name	people.csv
Number of unique subjects	9958
Number of properties	20
Minimum height cluster analysis	0
Maximum height cluster analysis	1
Number of cluster groups at h=0.1	3
Number of cluster groups at h=0.2	3
Number of cluster groups at h=0.3	3
Number of cluster groups at h=0.4	3

Table 3.4: Calculated Metrics for LODprobe results, sample measurement results for the New York Times Linked and Open Data "People" resource.

behaviour of the clusters between the heights 0.1 and 0.4 seems to be most informative. Especially at lower heights of 0.1, 0.2 or sometimes even 0.3, properties usually seem to be clustered based on the logical entities found in the data, just before those clusters are again grouped together with other clusters. In the example above, even at lower heights, the 20 involved RDF properties constitute three groups. This is an indicator that the properties within the groups are very similar, but the groups themselves are very distinctive.

Finally, the collected metrics are compared to metrics computed for synthetically generated resources, simulating the case in which RDF data is truly heterogeneous. Thereby, property occurrence and co-occurrence counts are randomised and then normalised by the amount of actual unique subjects. This is repeated in a Monte-Carlo-like process based on two real examples, a small RDF resource with 9,958 subjects, 20x20 properties, and 1,000,000 simulations and a large one with 694,400 subjects, 222x222 properties, and 15,127 simulations.

In addition to the cluster analysis metrics, dendrograms are generated for each of the 251 LODprobe results to support the interpretation. They are generated using R's 'plot' function with the generated cluster analysis from above, without any further parameters.

Figure 3.4 on page 105 shows such a dendrogram for the New York Times Linked and

Open Data “People”. Just with the plain numbers, two entities can clearly be identified: Properties on the left side are grouped together at a height of 0, on the right side, all other entities except one are grouped together at a height of < 0.2. A single property then joins this group at approximately 0.4. Both groups then are joined at a height of 1.0, indicating a high dissimilarity of both.

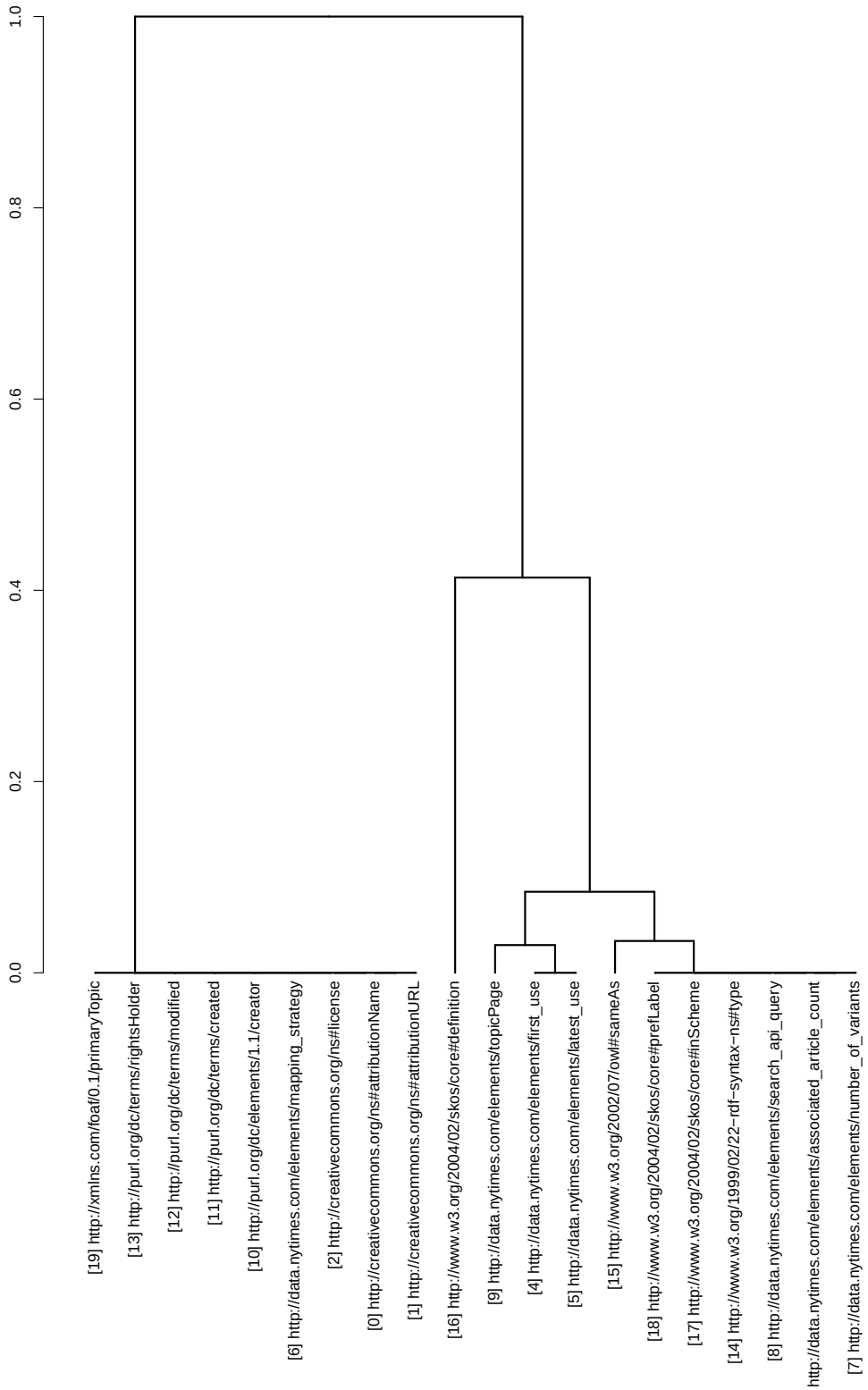


Figure 3.4: Dendrogram of the LODprobe analysis for the New York Times Linked and Open Data "People" resource.

3.2.2 Benchmark of Storage Technologies

In this section, a benchmark for the warehouse storage of Linked and Open Data is developed. Thereby, the following existing work is compared (Section 2.4 on page 59):

- Berlin SPARQL Benchmark (BSBM) (Section 2.4.3 on page 60)
- Lehigh University Benchmark (LUBM) (Section 2.4.1 on page 59)
- SPARQL Performance Benchmark (SP2Bench) (Section 2.4.2 on page 60)
- DBpedia SPARQL Benchmark (DBPSB) (Section 2.4.4 on page 61)
- Linked Data Benchmark Council Semantic Publishing Benchmark (LDBC SPB) (Section 2.4.5 on page 62)

Besides this existing work, another foundation for this section is the work of Gray, who underlined the necessity of domain-specific benchmarks (Gray, 1993, p. 2), as long as they meet the following four criteria (Gray, 1993, p. 3f):

Relevant - measures must be settled within the specific domain of interest.
Portable - implementation should be possible on different systems and architectures.
Scaleable - scaling the benchmark should be possible.
Simple - the benchmark should be easily understandable.

In regard of these properties, the benchmark endeavour will be defined in the following. Fundamental design decisions are made for the queried data (Section 3.2.2 on the facing page), performed database operations (Section 3.2.2 on page 109) in defined query scenarios (Section 3.2.2 on page 111), and measured metrics (Section 3.2.2 on page 111). Each decision will be compared to corresponding existing benchmarks. The final execution will be defined in several phases (Section 3.2.2 on page 113).

All scripts, queries and program source code are attached in appendix B on page 293, as well as being published in the GitHub repository loddwhbench[8].

[8] `https://github.com/heussd/loddwhbench`, last access on 2016-02-15.

Queried Data

The exact choice for a certain dataset is essential, as it focuses the application scenario, and queries must be designed to work with it. So in order to make the benchmark relevant for a specific scenario (Gray, 1993, p. 3f), the selected dataset must be application specific.

In this benchmark, the bibliographic resources of the library union catalogues of Hesse, *HeBIS catalog*, will be used as the foundation of the dataset and query scenarios. This exact dataset is suitable for multiple reasons: firstly, it is relevant, as it is true, real-world Linked and Open Data, available at the data portal datahub.io[9]. Secondly, it comprises about 14 GB of data - about 11 million books - so the benchmark can be scaled at arbitrary sizes from small query scenarios up to large ones. Thirdly, *Hessisches Bibliotheks- und Informationssystem (Library and Information System of Hesse) (HeBIS)* provided the author with three nearly identical dumps in the different data formats RDF, MARC, and PICA. This allows the comparison of integration efforts of specific formats too, e.g. when RDF can be loaded natively (in triple stores), whereas, in contrast, MARC must first be parsed. Table 3.5 on the following page shows the fields of the HeBIS catalogue in detail, including a sample entry.

Different sizes of the HeBIS catalogue are compared in different *test series*, as required by the scalability criteria (Gray, 1993, p. 3f):

TINY A random selection of 1,000 records.

SMALL A random selection of 100,000 records.

MEDIUM About 2,2 million records[10], loaded from `hebis_10147116_13050073_rdf_gz`.

LARGE About 5,8 million records[9], loaded from `hebis_10147116_13050073_rdf_gz` and `hebis_29873806_36057474_rdf_gz`.

[9] `http://datahub.io/dataset/hebis-bibliographic-resources`, last access on 2015-09-14.

[10] The HeBIS catalogue is delivered in seven GZIP-compressed RDF/XML parts. As each part can be loaded individually, the sizes of the MEDIUM and LARGE test series represent exactly one (MEDIUM) or two (LARGE) specific original parts of the catalog.

Property / Column	Samples from the dataset
DCTERMS_IDENTIFIER	334023971
BIBO_OCLCNUM	3252176
RDF_ABOUT	http://lod.hebis.de/resource/33402397
DCTERMS_TITLE	Where the gods are mountains
DCTERMS_PUBLISHER	Weidenfeld and Nicolson
ISBD_P1017	Weidenfeld and Nicolson
ISBD_P1016	London
ISBD_P1008	1. publ. in Great Britain
ISBD_P1006	three years among the people of the Himalayas
ISBD_P1004	Where the gods are mountains
ISBD_P1018	1956
DCTERMS_ISSUED	1956
OWL_SAMEAS	http://d-nb.info/1034554514
RDF_TYPE	{http://purl.org/vocab/frbr/core#Manifestation http://purl.org/dc/terms/BibliographicResource, http://purl.org/ontology/bibo/Book}
DCTERMS_MEDIUM	paper
DCTERMS_FORMAT	print
BIBO_EDITION	1. publ. in Great Britain
WDRS_DESCRIBEDBY	http://lod.hebis.de/catalog/html/33402397
DCTERMS_SUBJECT	http://d-nb.info/gnd/4024923-2

Table 3.5: Per-entry properties of the HeBIS catalogue, including a sample entry, following the DINI-AG KIM recommendation (DINI, 2014). Usually, not all properties are set for every record. Also some fields may contain multiple values, e.g. RDF_TYPE.

Table 3.6 shows which datasets are used by existing benchmarks. Artificial datasets are usually generated with available tools to generate arbitrary amount of data, following a pre-defined, application-specific schema.

BSBM	LUBM	SP^2Bench	DBPSB	LDBC SPB
Synthetically generated based on e-commerce scenario	Synthetically generated based on real ontology (university & department domain) (Guo et al., 2005, p. 160)	Synthetically generated based on DBLP, based on real data distributions (Schmidt et al., 2009, p. 224)	DBpedia	Named Entities from the BBC and DBpedia, also synthetically generated based on media assets (Kotsev et al., 2015, p. 12)

Table 3.6: Comparison of Datasets used in existing Benchmarks.

Database Operations

In a data warehouse scenario, all basic data manipulation operations - *CRUD* - as well as fundamental schema changes - are likely and relevant:

Schema Change describes operations that change the structure of stored entities, including introduction or removal of attributes. In a data warehouse, this happens when existing data is migrated. In triples stores, this operation is conceptually impossible because there is no predefined entity structure (Section 2.2.2 on page 53).

Create includes operations that create or load entities initially. Where the DBMS is schema-aware, this usually involves steps for the creation of structures (e.g. tables in relational DBMS) and entity validation. In a warehouse scenario, this happens when data is imported, e.g. within an ETL step.

Read describes operations to access stored information. This usually includes more advanced queries, containing features such as aggregations, groups or joins. In a data warehouse, reading is the fundamental usage scenario to answer several kind of questions.

Update specifies operations to change stored information. Again, in the case the DBMS is schema-aware, this usually involves validation of inserted information. In a data

warehouse, this also happens as part of an ETL-process.

Delete includes the deletion of stored information. In a warehouse scenario, deletion happens for example when outdated data needs to be removed.

It is important to note that use of these database operations is not expected to be evenly distributed. In practice, read operations in a data warehouse scenario will be more frequent than changing a schema.

Naturally, the performance of many operations is dependent on their *selectivity* - the amount of information that is affected on the exact operation. In a warehouse scenario, this selectivity might occur in two principle variants:

High Selectivity The pre-selection of data is specific, little information is affected by the testified operation.

Low Selectivity The pre-selection of data is unspecific, the operation affects large parts of stored information.

Table 3.7 shows which fundamental operations are analysed in existing benchmarks.

	Existing Benchmarks					
Database Operation	BSBM	LUBM	SP^2Bench	DBPSB	Kämpgen and Harth	LDBC SPB
Schema Change	-	-	-	-	-	-
Create	-	✓	✓	-	✓	✓
Read	✓	✓	✓	✓	✓	✓
Update	✓	-	-	-	-	✓
Delete	-	-	-	-	-	✓

Table 3.7: Comparison of database operations subject of existing benchmarks.

Usually, read-based operations are focussed. In contrast, LDBC SPB not only covers all basic CRUD operations, but it also features many sophisticated (analytical) read operations, e.g. for drilling in the data.

Measured Metrics

Besides the selection of a certain dataset (Section 3.2.2 on page 107) and the scrutinised database operations (Section 3.2.2 on page 109), the exact measurement method is another principle design decision of a benchmark.

In this benchmark, read-only operations are measured three times, and all timings are logged individually. This way, the impact of caching mechanisms can be observed. Write-based operations, including initial dataset loading times, are executed and measured once. Measuring plain query performance, however, does not represent real application scenarios. In real scenarios, the performance of intermediate layers such as Java Database Connectivity (JDBC) is just as important, as these layers make the content actually actionable for the application's program logic. Time measures in this benchmark thus not only contain the query times a DBMS takes, but the time required to make the results of these queries available in a dedicated object.

Table 3.8 shows which existing benchmark uses the same metrics as this benchmark.

	Existing Benchmarks					
Metric	BSBM	LUBM	SP^2Bench	DBPSB	Kämpgen and Harth	LDBC SPB
Load Time	-	✓	✓	-	✓	-
Single Q. Time	✓	✓	✓	✓	✓	-
Min. Q. Time	-	-	-	-	-	✓
Max. Q. Time	-	-	-	-	-	✓
Aggregated Q. Times	✓	-	-	-	-	✓

Table 3.8: Comparison of (selected) metrics in existing benchmarks.

Query Scenarios

Based on the selected HeBIS catalogue (Section 3.2.2 on page 107) and the chosen operations (Section 3.2.2 on page 109), Table 3.9 on the next page shows the *query scenarios* used to evaluate the DBMS performances. They cover schema change, read, update and delete operations. Create operations are covered with the load of datasets.

	Query scenario	Description
Schema Change	SCHEMA_CHANGE_MIGRATE_RDF_TYPE	Create a boolean field for every single RDF_TYPE, each indicating if a record is of this type. After that, remove RDF_TYPE from every record.
	SCHEMA_CHANGE_REMOVE_RDF_TYPE	Remove RDF_TYPE from every record.
	SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	Create a new property for every record, defaulting to "cheesecake".
	SCHEMA_CHANGE_INTRODUCE_STRING_OP	Take substring of ID (RDF_ABOUT) and store it in a new property.
Read	ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	Retrieve the first, ten or 100 entities of a list of all complete entities, ordered by dcterms_medium, isbd_p1008, dcterm_contributor, and dcterms_subject.
	AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10 AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100 AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL	Count the publications per publisher and order by this count descending. Return two columns (publisher + the respective count). Show 10, 100 highest or all highest counts.
	AGGREGATE_ISSUES_PER_DECADE_TOP10 AGGREGATE_ISSUES_PER_DECADE_TOP100 AGGREGATE_ISSUES_PER_DECADE_ALL	Count the publications per issued century and order by this count descending. Return two columns (century + the respective count). Show the 10, 100 highest or all counts.
	CONDITIONAL_TABLE_SCAN_ALL_STUDIES CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES	Returns all complete entities matching: 1) "Studie" or "Study" (case insensitive) in their title (about 2% of the records). 2) "Bibliographic Resources"-type (about 92% of all entities. 3) both 1) and-chained with 2)
	GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_2HOPS	For each record, find records that share a DCTERMS_SUBJECT and return record identifiers and shared subjects.
Update	UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	Set DCTERMS_MEDIUM to "recycled trees" on records having DCTERMS_MEDIUM == "paper", affects about 90% of the records.
	UPDATE_HIGH_SELECTIVITY_NON_ISSUED	Set DCTERMS_ISSUED to 0 on records that have no value for this property, affects about 2% of the records.
Delete	DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	Remove records having DCTERMS_MEDIUM == "paper" (about 90% of the records).
	DELETE_HIGH_SELECTIVIY_NON_ISSUED	Remove records that have no value for DCTERMS_ISSUED (about 2% of the records).

Table 3.9: Query Scenarios of this evaluation. They cover Schema Change, Read, Update and Delete operations. Create operations are covered in the load of datasets.

Execution Phases

Putting it together, for a given database implementation, the benchmark is executed in the following phases:

Set Up Initialises the environment of the given database, makes sure it is available, prepares and clears internal structures. It is only executed once for each database.

Load Imports a given test series (Section 3.2.2 on page 109), consisting of one or more files. The required time is measured. Errors in this step invalidate all follow-up timings.

Prepare Instructs the database to prepare for a specific query scenario (Table 3.9 on page 112). This preparation might include creation of additional data structures, such as index views, etcetera. Timings are measured for this phase. It is executed once for each query scenario.

Query Executes one or more queries, required to fulfil a certain query scenario (Table 3.9 on page 112). If the query scenario only contains read operations, this phase is executed three times to observe caching behaviour. All timings are recorded individually.

3.2.3 Architecture of a Linked and Open Data Integration Chain

Integrating data is the essence of various findings of the literature review, such as the preference for application specific entities (Section 2.9.1 on page 85), APIs, and architectures (Section 2.9.3 on page 88). In addition to that, warehousing this integrated data ensures data quality and reliability of the service, and thus must be considered good practice (Section 2.9.5 on page 90).

In the following, the architecture of a framework for the integration of application-specific Linked and Open Data is developed. Thereby, required steps are identified and existing philosophies (Section 2.1 on page 48) and approaches (Section 2.6 on page 68) are compared.

Data integration endeavours often involve similar, transferable steps. In many specific application scenarios, data first needs to be integrated in application-specific entities, which then need to be stored, customised, delivered or distributed, and finally consumed. Consequently, equivalent steps or life cycle phases can be found in existing, conceptual models such as (Vaisman and Zimányi, 2014), (Fayyad et al., 1996), (Pellegrini, 2012), (Smith and Schirling, 2006), (Kosch et al., 2005).

Table 3.10 on the facing page identifies and consolidates the basic steps data integration steps, describes corresponding tasks associated with each, and compares them with steps or life cycle phases of the mentioned conceptual work.

	ETL	**Storage**	**Customisation**	**Delivery**	**Use**
Tasks	Loading multiple formats, filtering, duplicate & quality checks, enrichment, applying application specific schemata	Storing data in application specific entities, source for differential data loads, linking and enrichment of data	Implementing application semantics with the stored entities, data transformation in use-case-oriented formats	Storing data in use-case oriented formats, providing query interfaces / APIs to the data	Using the prepared data in pre-defined application scenarios, e.g. in user-friendly GUIs
	Corresponding phase names in related models (Section 2.5 on page 62)				
Vaisman and Zimányi	Data Sources / Back-end Tier	Data Warehouse Tier	OLAP Tier	OLAP Tier	Front-End Tier
Fayyad et al.	Data Selection / Preprocessing	Evaluation / Data Mining	Data Mining / Interpretation	Acting	-
Pellegrini	Content Acquisition	Content Editing	Content Bundling	Content Distribution	Content Consumption
Smith and Schirling	Acquire	Maintain / Store	Index	Distribute	Delivery
Kurz et al.	Integration	Enrichment	Index	Publishing / Search	-

Table 3.10: Description of the five basic, subsequent data integration steps, conceptualised and consolidated from existing work of (Vaisman and Zimányi, 2014), (Fayyad et al., 1996), (Pellegrini, 2012), (Smith and Schirling, 2006), and (Kosch et al., 2005).

ETL

In the ETL step, data is initially extracted from external data sources or from local data dumps (Extraction). This might involve different data formats. Depending on the source, further filtering, de-duplicating, quality ensuring, and enrichment tasks are required (Transformation). Finally, an application-specific data model is introduced and entities are loaded (Load).

In related work, this step is also called *data sources* (Vaisman and Zimányi, 2014, pp. 76), *data selection* and *preprocessing* (Fayyad et al., 1996, p. 42), *acquire* (Smith and Schirling, 2006, p. 85) or *content acquisition* (Pellegrini, 2012, p. 97) as well as *integration* (Kurz et al., 2011, pp. 16).

Storage

The storage step is about storing required data in entities that are specific to the application's semantic endeavour. Because of its persistency, storage is the foundation for building a differential load setup. Once the data is pooled, further interlinking and enrichment in the specific context of an application is possible. Therefore, developer-friendly interfaces to the integrated data are provided.

The storage step and described actions here can be compared to the *backend tier* (Vaisman and Zimányi, 2014, pp. 76), *content editing* (Pellegrini, 2012, p. 97), *enrichment* (Kurz et al., 2011, pp. 16), *evaluation* and *data mining* (Fayyad et al., 1996, p. 42) as well as *store* and *maintain* (Smith and Schirling, 2006, p. 85).

Customisation

The customisation step involves implementing an application's specific semantics with the previously prepared and stored entities. Depending on the selected application scenario, the polyglot persistence paradigm (Section 2.8.3 on page 83) is realised in this step: entities are transformed in use-case-oriented formats, suitable for an efficient delivery.

In related work, these activities are associated to *index* (Smith and Schirling, 2006, p. 85), (Kurz et al., 2011, pp. 16), the *data warehouse tier* (Vaisman and Zimányi, 2014, pp. 77) *content bundling* (Pellegrini, 2012, p. 97), *data mining* and *interpretation* (Fayyad et al., 1996, p. 42).

Delivery

In the delivery step, data is served in use-case oriented formats, in order to make use of it as easy and as streamlined as possible. Accordingly, query interfaces and APIs to the data are provided. Following the polyglot persistence paradigm (Section 2.8.3 on page 83), this might involve additional storage technologies, such as IR frameworks or NoSQL databases.

This step can be compared to *content distribution* (Pellegrini, 2012, p. 98), the *OLAP tier* (Vaisman and Zimányi, 2014, pp. 78), *distribute* (Smith and Schirling, 2006, p. 85), *publishing* and *search* (Kurz et al., 2011, pp. 16) and *acting* (Fayyad et al., 1996, p. 42).

Use

Using the integrated data is sometimes seen as data integration step, too. It involves an ecosphere for working with the previously provided query interface or API, usually in very specific realms.

Corresponding steps are *content consumption* (Pellegrini, 2012, p. 98), the *front-end tier* (Vaisman and Zimányi, 2014, pp. 79) and *use* (Smith and Schirling, 2006, p. 85).

3.2.4 Proof of concept Framework Application

> To be relevant to [a] community, research must address the problems faced and the opportunities afforded by the interaction of people, organisations, and information technology. (Hevner et al., 2004, p. 85)

As a proof of concept, the developed framework is utilised in the code base of the Mediaplatform research project. With its use-case-oriented partners Städel Museum and the ULB, it represents real use and business cases and can, moreover, be seen as a prototype for a number of comparable Linked and Open Data integration endeavours in the GLAM domain as well as in other domains.

Thereby, it will address existing data integration challenges in the existing Java code base and will replace manually implemented functionality with framework capabilities.

It is of interest to analyse what changes through the use of this framework. The following criteria are relevant:

Clear Separation of Data Integration Steps

The developed prototype framework introduces a clear separation of components that support the different steps of data integration (Section 3.2.3 on page 114). This is an important architectural foundation for a semantic application and has a number of benefits:

- It allows the dedicated use of one or more specialised persistence technologies, depending on the use case of front-end applications, and thus, following the polyglot persistence paradigm (Section 2.8.3 on page 83).

- Development processes can be supported, as code is written for and tested against clearly distinguished deployment and execution phases of the software.

- Introducing an intermediate storage layer is essential when implementing advanced load techniques, such as incremental load.

This criteria can be demonstrated with source code excerpts and with architectural descriptions.

Execution Time

A clear separation of the several data integration steps can be ultimately demonstrated by the ability to re-execute, skip or repeat certain integration steps. Thus, it is the foundation to optimising the execution process. For example, certain data source loaders and dependent processes can be skipped if a source has not been changed since the last load.

To demonstrate this, it is shown that execution times can be reduced significantly if the data sources are not changed.

Lines of Code

Even though it is a controversial metric, the *Lines of Code (LoC)* metric allows a rough estimation of the code complexity and development efforts. After integrating the developed prototype framework, the following two aspects can be expected:

- In components which provide functionality that can be replaced by the prototype's core functionality, the LoC is decreased significantly. This demonstrates that the prototype reduces manual development effort.

- For other components, the LoC is not increased significantly. This demonstrates that even if the framework introduces new components, it does not increase the overall development efforts compared to straightforward scenarios without these components.

The LoC change in the code base is measured with the common open-source tool cloc (AIDanial, 2015). It supports various programming languages and distinguishes between blank-, code- and comment-lines.

Functionally Equal

The existing and successful JUnit test cases should still work after the prototype is integrated in the project and tested data is newly created with it.

This will be demonstrated by successful maven test phase execution.

Chapter 4

Framework for the Integration of Linked and Open Data

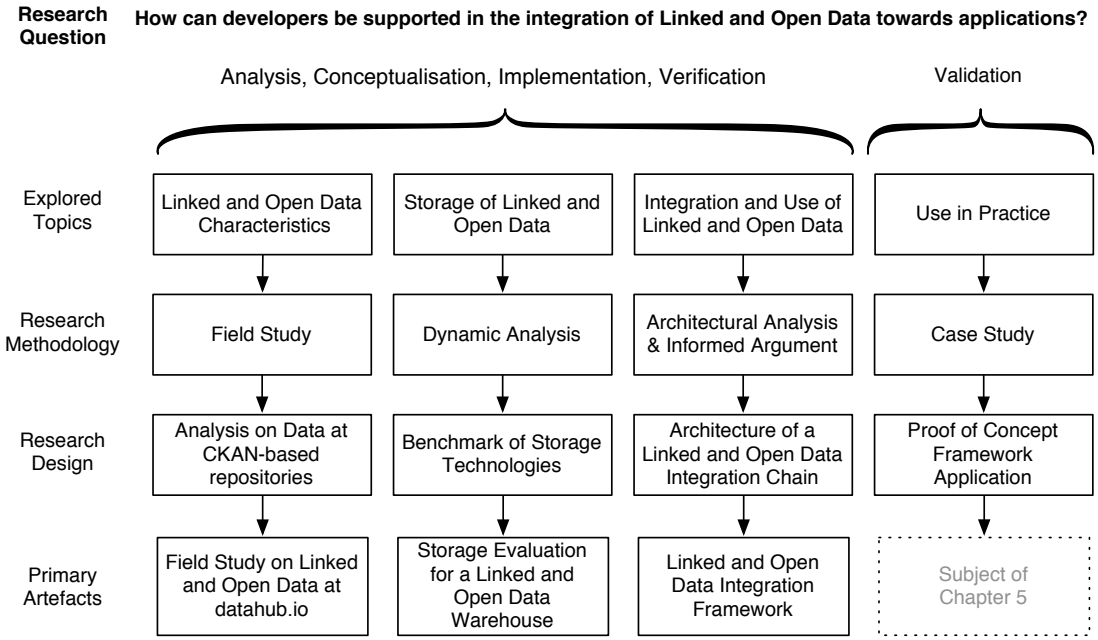


Figure 4.1: Overview of the Research Question, explored topics, selected methods, research design and the developed, novel artefacts of the present thesis.

In this chapter, the methods provided are used to construct novel insights and artefacts. The individual contributions are:

- Field Study On Linked and Open Data at Datahub.io on the following page
- Storage Evaluation for a Linked and Open Data Warehouse on page 134
- Development of a Linked and Open Data Integration Framework on page 159

4.1 Field Study On Linked and Open Data at Datahub.io

The 5-star model of Tim Berners-Lee was previously introduced as a maturity model for "the openness and linking potential of the data" (Edelstein et al., 2013, p. 4) (Section 2.1.3 on page 50). As there is only a licence-based constraint for data to achieve the first star, many different file formats might come into question when assessing Open Data. Even the term Linked Data might include Linked *Open* Data and just plain Linked Data. In addition, in practice, both worlds are more convergent than they might suggest (Edelstein et al., 2013, p. 2).

When building applications based on Linked and Open Data, the question of format becomes important: it requires an entirely different technology stack to integrate, assure quality, and store data from, for example, RDF, compared to the technology needed for data in Microsoft Excel format. Moreover while building apps based on Office formats is straightforward (thanks to frameworks like Apache POI (Apache POI, 2014)), years of experience and complex infrastructure might be required to master the advanced possibilities of RDF, such as OWL-reasoning. Furthermore, integrating different formats is challenging, e.g., if the internal structure of the data differs fundamentally. While data stored in spreadsheets might usually be of tabular nature, RDF-based data, by design, does "not necessarily consist of clearly identifiable 'records'" (Isaac et al., 2005). RDF knowledge bases are considered to be heterogeneous, non-tabular structures, which do not resemble relational structures (Morsey et al., 2011, p. 455).

Therefore, the question that needs to be answered before putting the data to use is: what are the relevant formats when dealing with Linked and Open Data in practice? Under what terms of use may the data be reused and processed? If the format is RDF, can it be flawlessly loaded and does it require sophisticated tool support (in form of OWL reasoners)? What is the internal structure of the real-world RDF? Does it constitute homogeneous, tabular structures or is it heterogeneous as it is often the case in large knowledge bases?

This field study is conducted using the well-known data portal datahub.io, as it is

used as an indicator for the progress of Linked Data upon creating the LOD Cloud (Section 1.1.4 on page 34) (Richard Cyganiak, 2014). The data has been extracted in March 2015 using CKANstats.py (Section 3.2.1 on page 97).

4.1.1 Common Formats

Considering unified[1] format values, the most frequently used formats for data are full-featured spreadsheets such as Microsoft Excel or LibreOffice Calc documents, and CSV, including its variations like Tabular-separated Values (TSV). Together, both tabular formats add up to almost one third of all data formats (27%), followed by RDF (11%), PDF (8%), and images (7%). The total format-distribution is shown in Table 4.1.

Unified Format Statement	Count	Percent
n/a	2983	14.78%
Spreadsheet	2983	14.78%
CSV, TSV, and Variations	2533	12.55%
RDF	2279	11.29%
PDF	1643	8.14%
Image	1371	6.79%
RDF example record	1079	5.35%
HTML	985	4.88%
URL to SPARQL Endpoint	682	3.38%
URL to Web Service	627	3.11%
Compressed Archive File	534	2.65%
Geo	511	2.53%
Database Dump	481	2.38%
XML	463	2.29%
JSON	364	1.80%
Binary	137	0.68%
Text	132	0.65%
Non-Spreadsheet Office Document	69	0.34%
API	63	0.31%
Beacon	59	0.29%
MARC	51	0.25%
Really Simple Syndication (RSS)	47	0.23%
Script	44	0.22%
Source Code Repository	28	0.14%
Map	25	0.12%
General Transit Feed Specification (GTFS)	3	0.01%
Sound	2	0.01%

Table 4.1: Frequency of different formats of data online at datahub.io.

[1]The complete mapping table used for the unification of the format statements can be found in Appendix A.2 on page 251.

With regard of the openness of data[2], the frequently used data formats are usually not openly licensed: only 21% of the spreadsheets are open, about 59% of the CSVs, and roughly 14% of the PDFs. RDF, in contrast, is openly licensed in more than three of four cases (76%). The highest openness-percentages can be archived for the formats MARC (100%), GTFS (100%), and Beacon (98%).

By using the 5-star rating model (Section 2.1.3 on page 50) to classify the data[3], over a quarter (25%) of the data is 1-star, 6% 2-stars, 24% 3-stars, and 21% is 4 stars (or more), as shown in Figure 4.2.

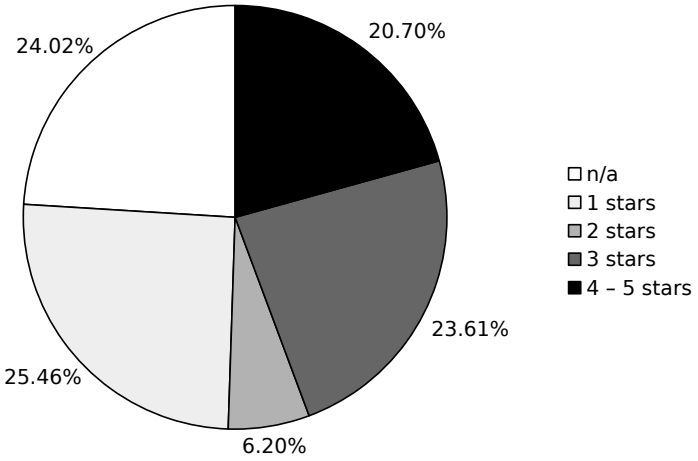


Figure 4.2: Distribution of real data from datahub.io in Berners-Lees' 5-Star Model.

Excluded from this is data that is not explicitly openly licensed (about 48%). One fourth of open data formats could not be classified ("n/a"). The low frequencies of 2-star data can be explained with the fact that the typical format in this category is Microsoft Excel. Data in Microsoft Excel, however, is usually not openly licensed.

4.1.2 Popular Formats

Based on existing data, a popularity measure can only be approximated using the tracked visits from the CKAN-API (Open Knowledge Foundation, 2013*b*).

[2]See "About openness" on page 267 for further information and complete queries.

[3]See "Data Classified in TBL's 5-Star Maturity Model" on page 269 for further information and complete queries.

According to the resource tracking[4] , in the last two weeks before the extraction, 33% of the clicks at datahub.io were on resources with an unknown format, followed by RDF (13%), RDF sample record (10%), CSV (8%), spreadsheet (7%), SPARQL (6%).

Despite the fact that PDF is quite a quite common format, resources with that format only received 2% of the tracked clicks. The most unpopular formats are URLs, RSS and maps with 0,3%, 0,2% and 0,05% of the visits.

4.1.3 Licences

A clear license statement for data is important as it defines a terms of use of the referred resource(s).

However, similar to the case of the non-unified format information, the licence-field does not contain standard and unified values[5] either. Even worse, in more than 40% of the cases, there is no specification of a licence at all. In the remaining cases, properly defined licences (like Creative Commons licences) are mixed with country- or language-specific licences, and insufficiently named licences like “None”, “Other (Not Open)”, or “apache”.

Moreover, the boolean openness flag is set in 10,612 of the total 20,178 cases, which corresponds to 52.59%.

4.1.4 Ages

The extract contains meta-information about data up to four years old. Judging by the created and revision timestamps[6], in most cases (80%) this meta-information is never updated after the dataset had been put online. Of the remaining cases, more than 10% are updated within less than 50 days.

[4] See “Recent & total visits per format” on page 275 for further information and complete queries.
[5] See “Licences” on page 276 for further information and complete queries.
[6] See “Created vs. Timestamp, Ages in days” on page 270 for further information and complete queries.

4.1.5 In-Depth RDF Analysis Results

In addition to the previous analyses, based on metadata of all data on datahub.io, the following analyses are limited to specific datasets of the type RDF and that have a popularity ranking larger than zero. At the time of extraction, this included 606 RDF resources.

Download and Process Results

Table 4.2 shows the download and processing result, classified with the predefined status codes section 3.2.1 on page 99.

Download	Processing	Count	Percent
Not Found	-	202	33,33%
Successful	Parse error	114	18,81%
Successful	Out of Memory	5	0,83%
Successful	Partly	4	0,66%
Successful	Successful	281	46,37%
Total		**606**	**100%**

Table 4.2: Download and processing result for observed RDF datasets.

In two-thirds of the cases, the download-URL worked and returned a server response (33% of the cases are not found). The downloaded resources, however, could only be flawlessly loaded in about 46,37% of the cases - the rest resulted primarily in parser errors (18,81%).

Only a few datasets (0,83%) could not be loaded due to insufficient memory of the test machine[7] - this includes the knowledge base DBpedia. An additional four resources[8] (0.66%) could only partly be loaded, e.g., because they are split-up into several parts.

[7] "dbpedia", "library-of-congress-name-authority-file", "semantic-xbrl", "europeana-lod-v1" and "allie-abbreviation-and-long-form-database-in-life-science"

[8] "jiscopenbib-bl_bnb-1", "geowordnet", "datos-bcn-cl", and "rkb-explorer-citeseer"

Homogeneous Structures in RDF

Table 4.3 shows the aggregated results of 253 LODprobe and cluster analyses for distinct resources[9]. On average, the RDF datasets contained 28 (standard deviation $\sigma = 33$) different RDF properties, and thus display a high statistical variance. However, these different counts of properties can be grouped in only 7 ($\sigma = 4$) clusters at a height of 0.1 and in only 5 ($\sigma = 2$) clusters at a height of 0.2. Table 4.3 compares the measured values with two synthetically generated and simulated heterogeneous RDF datasets of different sizes. In these, there are significantly more cluster groups at lower heights, such as for h=0.1, the number of groups is (almost) identical to the total number of groups.

	Average at datahub.io	**Synthetically Simulated**	
		small heterogeneous	**large heterogeneous**
number of resources	253	1,000,0000	15,127
number of unique subjects	217,659.05 (σ = 994,267.50)	9,958	694,400
min. height	0.00 (σ = 0.05)	0.13 (σ = 0.03)	0.04 (σ = 0.01)
max. height	8.48 (σ = 79.17)	0.58 (σ = 0.05)	0.51 (σ = 0.02)
Total Groups (h=0.0)	28.07 (σ = 33.03)	20	222
Groups at h=0.1	7.66 (σ = 3.96)	20 (σ = 0.5)	210 (σ = 3.91)
Groups at h=0.2	5.05 (σ = 2.08)	16 (σ = 2)	106 (σ = 13.1)
Groups at h=0.3	3.76 (σ = 1.35)	8.6 (σ = 1.4)	20 (σ = 2.6)
Groups at h=0.4	2.96 (σ = 1.00)	4 (σ = -1)	3 (σ = 1)

Table 4.3: Average cluster analysis results of observed data at datahub.io and of artificially simulated heterogeneous data.

Accordingly, Figure 4.3 on the next page shows a dendrogram of the real RDF resource `southampton-ac-uk-org` that has the exact characteristics of an average resource, thus having 28 different RDF properties that form 7 groups at a height of 0.1, 5 groups at a height of 0.2, and 4 groups at a height of 0.3. The diagram gives the impression of a clearly structured resource, as many properties are already grouped together at a height of 0 and the properties seem to have quite similar co-occurrence counts.

In contrast, Figure 4.4 on page 129 depicts a dendrogram of synthetically generated

[9] See "Cluster Analysis Results" on page 282 for further information and complete queries.

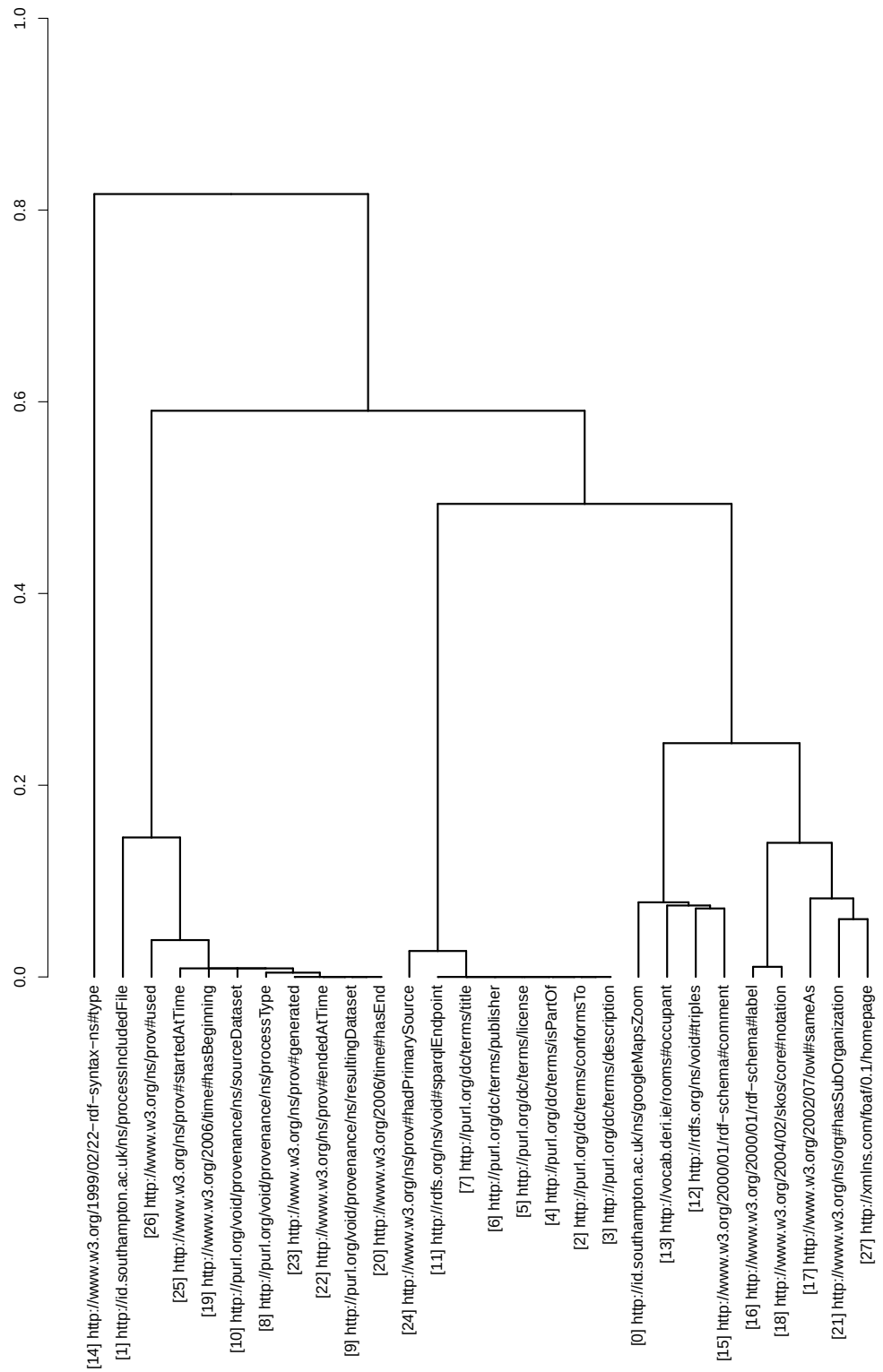


Figure 4.3: A typical dendrogram of a real-world RDF resource: Many joins at lower (< 0.2) heights, indicating a high number of co-occurrences for these properties. Few joins above heights > 0.3, an high maximum height of 0.8 or above.

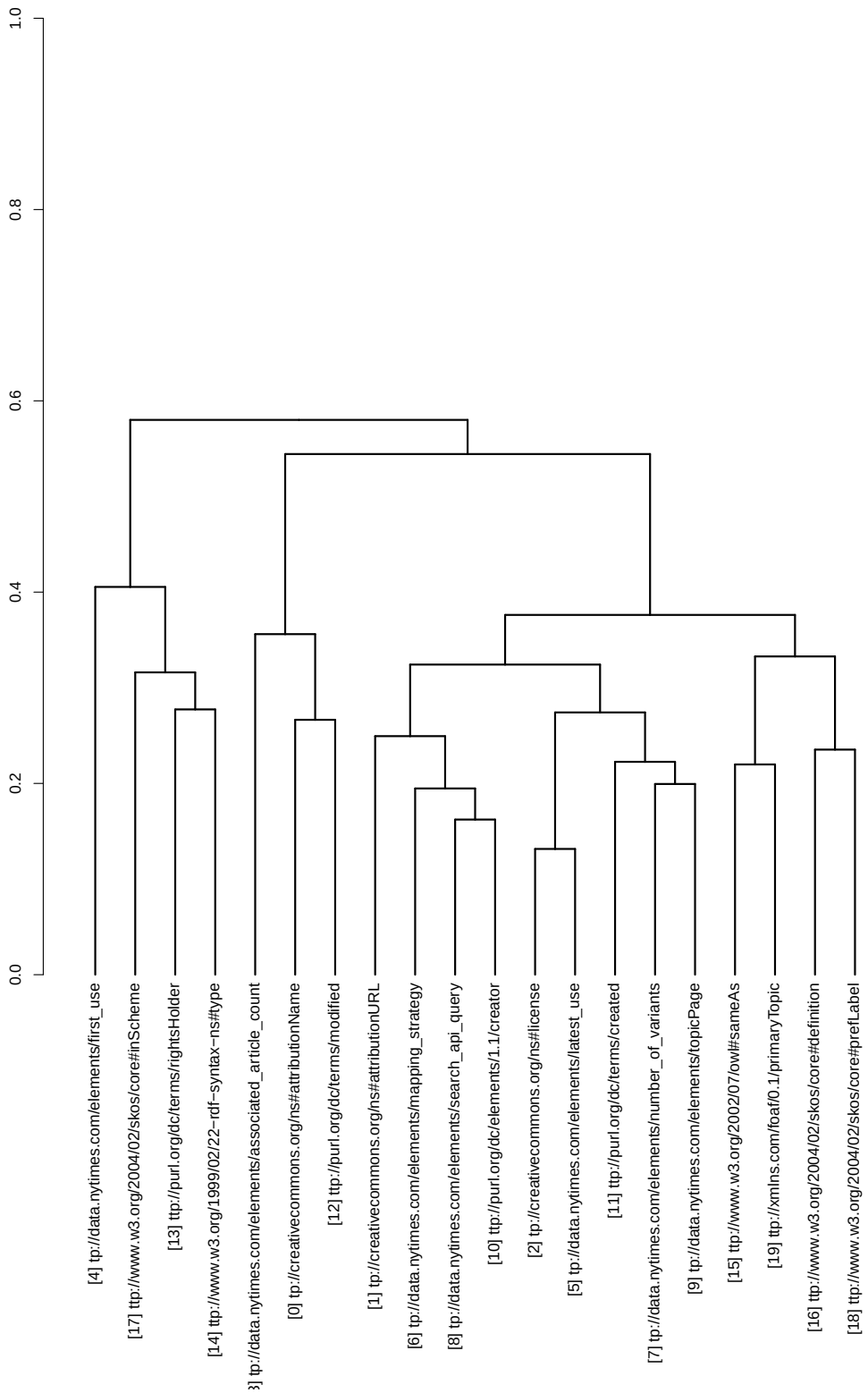


Figure 4.4: A typical dendrogram of a synthetically generated heterogeneous RDF resource: Joins are evenly distributed between heights of 0.1 and 0.5, almost no joins at lower (< 0.2) heights, a low maximum height of < 0.6.

resources (simulation number 473,494 of 1,000,000) that shows the average characteristics of simulated, small heterogeneous resources - 20 properties, 20 groups at h=0.1, 16 at h=0.2, 9 at h=0.3. The lack of structure can be deduced from the high number of groups at lower heights, individually consisting of less properties (usually maximum 2 properties per group at h=0.2). In addition, the lower maximum dendrogram height indicates that the properties are less mutually differentiable.

As expected, the number of groups found in a cluster analysis is proportional to the number of unique properties a resource has, as shown in Figures 4.5 and 4.6.

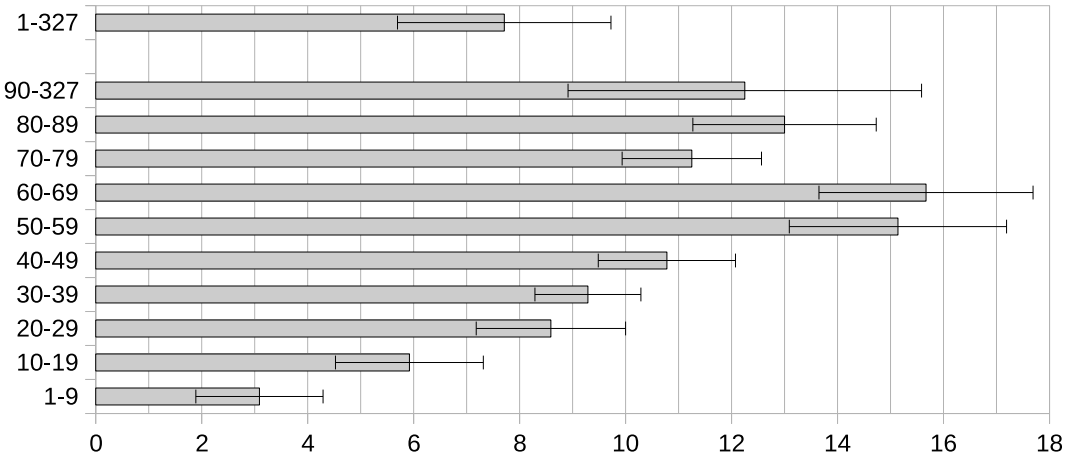


Figure 4.5: Average cluster groups at a heights of h=0.1, grouped by the number of properties.

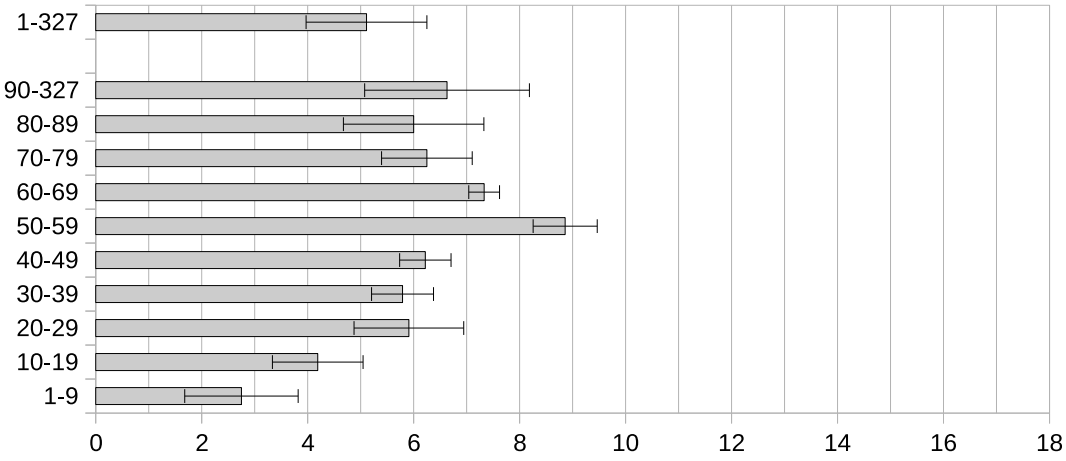


Figure 4.6: Average cluster groups at a heights of h=0.2, grouped by the number of properties.

Resources with an RDF property count up to nine show on average 3 groups (standard deviation $\sigma = 2.4$) at a height of 0.1. Resources with more properties, e.g. 70-79 distinct RDF properties, have 11 groups ($\sigma = 2.63$) at this height. This observation can be approximated by the rule of thumb "the more properties, the more groups". However, resources with 60 to 69 properties contain the most groups of all: 15.67 groups ($\sigma = 4$) at a height of 0.1.

Figure 4.7 visualises the observed proportionality between the number of properties and the groups they form: it compares the ratio between the number of groups at a height of 0.1 with the number of unique RDF properties a resource has in total for all involved analyses.

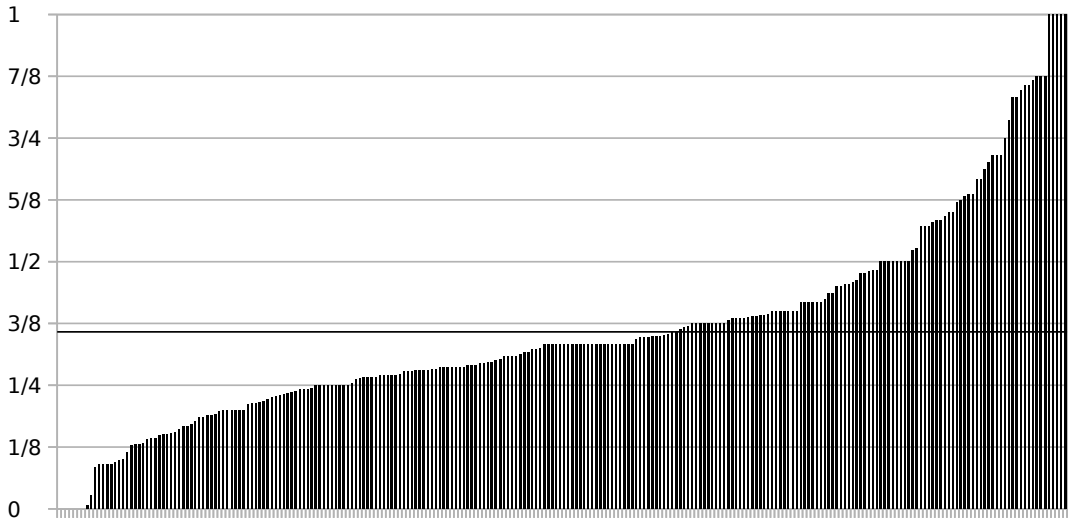


Figure 4.7: Ratio of the number of properties and the number of cluster groups at h=0.1 for all analysed RDF datasets.

Almost all ratios are below 4/5 and the average ratio is about 3/8, while, in contrast, the simulated heterogeneous resources both have ratios of nearly 1/1 at this height.

Frequently Used Properties

By summing up all individual LODprobe property counts, a big picture of the most frequently used RDF properties and vocabularies can be calculated. In total, over 373 million triples have been analysed, containing nearly 3000 different RDF properties.

Judging by the quantities, the most frequently found properties are shown in Table 4.4.

Property	Occurrences
`http://www.w3.org/1999/02/22-rdf-syntax-ns#type`	54,258,880
`http://www.w3.org/2000/01/rdf-schema#label`	19,172,810
`http://www.w3.org/2004/02/skos/core#subject`	10,922,966
`http://www.aktors.org/ontology/portal#has-author`	8,060,091
`http://purl.org/vocab/relationship/value`	6,885,590
`http://iflastandards.info/ns/isbd/elements/P1004`	6,646,736
`http://www.aktors.org/ontology/portal#has-author`	8,060,091
`http://www.w3.org/1999/02/22-rdf-syntax-ns#value`	6,885,590
`http://iflastandards.info/ns/isbd/elements/P1004`	6,646,736
`http://iflastandards.info/ns/isbd/elements/P1018`	6,564,523
`http://iflastandards.info/ns/isbd/elements/P1016`	6,270,126
`http://transport.data.gov.uk/0/ontology/traffic/count`	6,097,741
`http://transport.data.gov.uk/0/ontology/traffic/point`	6,097,741

Table 4.4: Top ten RDF properties, prefix.cc-based prefixes.

Unsurprisingly, basic properties are very frequent: one out of every six observed properties is an `rdf:type` assignment, one out of 20 is an `rdfs:label`.

Regarding OWL, the most frequently used properties with the default namespace `http://www.w3.org/2002/07/owl#` is `sameAs` - position 19 of the most frequently used properties with more than 4.5 million occurrences, followed by `onProperty` (Position 948 - 3,213 occurrences) and `intersectionOf` (position 1,016 - 2,038 occurrences).

4.1.6 Summary of the Findings

To explore characteristics of real-world Linked and Open Data, in this section, the results of an analysis about data on datahub.io is described.

Only about half of the data on datahub.io is clearly open, the majority of the rest holds legal uncertainty for application developers wishing to use it. Openness varies with the data formats: Excel is a very common format, but is usually not open. RDF is the third most common format, and is usually open. Accordingly, data on datahub.io is 53% Open Data, 15% Linked Data and about 10% Linked *and* Open Data. Moreover, after evaluating the 606 most popular RDF-based resources, 33% do not exist anymore and nearly 20% are inaccessible due to parser errors.

As a consequence of this distribution, RDF-only data integration approaches would not reach at least 85% of the public resources on datahub.io. This fact underlines the necessity that data integration solutions support a whole spectrum of different formats.

Real data format quantities are almost evenly distributed across the scale of the 5-star rating model, with the exception of 2-star data. 2-star data is usually data in the Microsoft Excel format, which is often not openly licensed. However, the star rating in the 5-star model is proportional to the number of page views a resource gets: The more stars, the more page views. It seems that RDF is data-consumer friendly.

For a portal that conceptually only distributes URLs to resources and not the resources themselves, having up-to-date metadata is essential - especially when considering the link rot phenomenon (Ceglowski, 2011). Unfortunately, the vast majority of metadata is never updated after it has been published. This might contribute to the 33% missing downloads mentioned above.

The cluster analysis revealed that RDF resources show clearly more characteristics of homogeneous data than of heterogeneous data. Almost all of the 251 dendrograms[10] contain structures in the form of properties that more or less exclusively co-exist with certain other properties.

So, even if it is not a constraint of the format, RDF triples, in the wild, constitute entities that are differentiable to some degree. These entities might fit in tabular structures as well as in graph-like ones.

Moreover, the cluster groups might give an approximate hint regarding the number of entities and the number of attributes each entity has: taking the average 28 properties and the 8 groups at h=0.1, and the 5 groups at h=0.2 into consideration, RDF data on average contains roughly 4-7 entities, each consisting of 4-6 properties.

[10] `https://github.com/heussd/CKANstats/tree/master/datahub.io/png`, last access on 2015-06-09.

4.2 Storage Evaluation for a Linked and Open Data Warehouse

Previously, the data warehouse pattern was pointed out as a solid, battle-tested architectural pattern for data integration scenarios (Section 2.9.5 on page 90). It allows quality assurance as well as pooling and streamlining of the data for the required use cases and thus, maximum performance and reliability during the online operation.

However, even though many benchmarks emerged within the last few years, existing work in the Web Science community (Section 2.4 on page 59) is only of limited use when talking about a specific data warehouse scenario. So, to the knowledge of the author, there is currently no single benchmark that combines all the following properties:

- tests all fundamental data manipulation operations that are of interest for the given data warehouse scenario.
- considers the whole spectrum of storage choices available today, including relational DBMS and NoSQL approaches, that are relevant in practise, but also recent research.
- works with realistic, domain- and application-specific Linked and Open Data datasets.

In addition, according to (Gray, 1993), it is always preferable to construct a custom benchmark that reflects the exact application scenario, instead of trying to transcribe general purpose results.

Motivated by this lack of application specific comparisons, in this chapter, the benchmark for the integration of Linked and Open Data in the data warehouse approach specified earlier (Section 3.2.2 on page 106) is implemented and results are described.

4.2.1 Selection of Databases

The choice for the considered databases in the following is driven by engineering: which candidate would be used in practice? Besides this technical consideration, licensing costs play a role, too. As the present thesis seeks to help a wide range of developers to benefit from Linked and Open Data, the required infrastructure should be open and available cross-platform. This leaves the ultimate freedom of choice to the developer. It should also foster experiments out of curiosity and help to reduce costly upfront investment in the technology.

Previously, it was noted that Web Science benchmarks are usually limited to triple stores, RDF and SPARQL (Section 2.4 on page 59). Conceptually, a fundamental advantage of triple stores is the fact that loading existing Linked Data is easy and effortless, as the database can parse RDF natively. Thus, in this evaluation, two triple stores are considered: Virtuoso, usually one of the best performers in existing benchmarks (Section 2.4 on page 59), as well as Apache Fuseki, which is already successfully employed in the LODprobe analysis (Section 3.2.1 on page 99).

The previously conducted field study pointed out that the usual office formats like spreadsheets or CSV files are just as common as RDF (Section 4.1 on page 122). Furthermore, existing RDF data dumps often expose tabular-like structures (Section 4.1.5 on page 127), so large quantities of structured Linked and Open Data might consist of tables, independent from their actual data format. Storing these data structures in a relational DBMS is a natural engineering choice, so this evaluation contains common relational approaches, PostgreSQL and the SQLite-based sqlite4java and SQLite-Xerial. In contrast to the latter one, sqlite4java has no JDBC abstraction layer, and thus claims to be faster (ALM Works, 2015).

In the context of NoSQL (Section 2.8.3 on page 83), alternative storage forms became popular, battle-tested and developer-friendly as well. Thereby, conceptually, a promising class to handle (large) collections of homogeneous data is a document store (Section 2.8.3 on page 84). In this evaluation, a popular implementation, MongoDB, is

included in the comparison.

Recently, databases started combining multiple NoSQL approaches (Edlich, 2015). One sample for such a database is ArangoDB, which is a document store, graph store as well as a key/value store (Section 2.8.3 on page 84). To see how this database performs compared to others, ArangoDB is part of the evaluation as well.

In summary, this comparison contains seven current database choices: sqlite4java, SQLite-Xerial, PostgreSQL, Apache Jena Fuseki, Virtuoso, MongoDB, and ArangoDB. Figures 4.8 and 4.9 on the next page and on page 138 provide an overview of characteristic properties these databases, such as options for atomicity, consistency, isolation, durability (ACID), backup or scale-out.

4.2.2 Installation

The evaluation is executed on an Apple MacBook Pro Retina, 15-inch, Late 2013, 2.6 GHz Intel Core i7 with 16 GB memory and 1 TB flash storage.

For the target platform OS X, most DBMS involved in this evaluation could be installed using the package manager Homebrew. This included PostgreSQL, MongoDB, and ArangoDB. There is also a current version of Virtuoso available via Homebrew, however, after installation, it cannot be launched due to segmentation faults. This could only be solved by manually downloading and compiling a previous version.

Both SQLite-approaches are retrieved and managed by Maven. Fuseki was manually downloaded[11] and extracted.

[11] `https://jena.apache.org/download/index.cgi`, last access on 2015-11-22.

Database Name	Open Virtuoso	PostgreSQL	sqlite4java
Version	07.20.3214	9.4.4.1 (Postgres.app)	392 (SQLite 3.8.7)
Licence	GPL v2	PostgreSQL Licence (similar to BSD or MIT)	Apache License 2.0
ACID Support	Full	Full	Full
Backup capabilities	Offline backup, Online backup, full databases or single tables / indices	SQL dump, File system level backup, Continuous archiving	Offline Backup, Online Backup API
Scale-Out / Replication	Clustering for dividing large databases (still fully transactional); Master-Master and Master-Slave replication	Via Extensions, e.g. https://github.com/citusdata/pg_shard	None
Import Formats	Linked Data, CSV, binary	PostgreSQL dumps, CSV, text, binary	PostgreSQL dumps, CSV, text, binary
REST Support?	Yes	No	No
Installation efforts (1 simple - 3 hard)	OS X: 3 - A specific version had to be compiled manually Windows: 2 - Installer	OS X: 1 - Homebrew Windows: 2 - Installer	1 - None - packaged as library
Operating Systems	Linux, OS X, Windows	BSD, Linux, OS X, Solaris, Windows	Android, Linnux, OS X, Windows
Can be used in Java via	JDBC, Open Database Connectivity (ODBC)	JDBC	Native API
Query Languages	SPARQL (Meta Schema Language for mapping SQL Data to RDF Ontologies),SQL	SQL	SQL

Figure 4.8: Characteristics of Evaluated DBMS, part 1

Database Name	SQLite-Xerial	MongoDB	ArangoDB	Fuseki
Version	3.7.2	3.0.6	2.6.9	2.3.0
Licence	Apache Licence 2.0	GNU AGPL v3.0	Apache License 2.0	Apache Licence 2.0
ACID Support	Full	Limited	Full	Full
Backup capabilities	Offline Backup, Online Backup API	Offline Backup: Tool mongodump for export of data	Offline Backup: Tool arangodump for dumping and arangorestore for restoring dumps	Online Backup using TBDump
Scale-Out / Replication	None	Replication possible; Sharding possible	Clustering via Sharding, Master-Slave Replication	None
Import Formats	PostgreSQL dumps, CSV, text, binary	JSON, CSV, TSV	JSON, CSV, TSV, Edge-Definitions as JSON	Linked Data
REST Support?	No	With Plugins such as RESTHeart	Yes	Yes
Installation efforts (1 simple - 3 hard)	1 - None - packaged as library	OS X: 1 - Homebrew Windows: 2 - Installer	OS X: 1 - Homebrew Windows: 2 - Installer	2 - Manual download and extract
Operating Systems	Linux, OS X, Windows	Linux, OS X, Solaris, Windows	Cloud (Azure or AWS), Linux, OS X, Windows	Linux / Unix, OS X, Windows
Can be used in Java via	JDBC	MongoDB Driver, Third-party POJO-Mappers	ArangoDB Driver, ArangoDB Object Mapper	Apache Jena
Query Languages	SQL	MongoDB Language (JSON-based)	ArangoDB Query Language	SPARQL

Figure 4.9: Characteristics of Evaluated DBMS, part 2

4.2.3 Setup

All databases are configured for an environment with 16 GB memory. Java-based databases (SQLite-Xerial, sqlite4java, and Fuseki) are given 16 GB heap space[12].

Virtuoso requires the configuration of each individual data import folder using the `DirsAllowed` setting in the main configuration file Virtuoso.ini. This file was also used to configure a 16 GB memory environment[13]. It is important that the current working directory is `/usr/local/virtuoso-opensource/var/lib/virtuoso/db` when launching Virtuoso. Otherwise, the database does not start, and prints no error message, even though its start-command is available system-wide. Parameters must be passed using `+`, e.g. `virtuoso-t +foreground`, and not using the standard Unix convention of minus (-)(Kernighan and Pike, 1984, p. 13).

MongoDB requires the one-time, manual creation of the folder `/data/db`. To automate the database creation on the fly, Fuseki is started with parameters to automatically create a plain database. Analogously, ArangoDB is instructed to create an empty database after launch. PostgreSQL is configured using the tool pgtune to specify 16 GB system memory.

The detailed startup and initialisation scripts can be found in appendix B.11 on page 369. Besides heap space and a database file name, neither SQLite-based approaches are configured at all.

4.2.4 Storage Decisions

Some fields, such as `RDF_TYPE` or `DCTERM_SUBJECTS`, may contain multiple values for certain entities (Table 3.5 on page 108). In relational DBMS approaches, multiple values are embedded within the tables, using the individual capabilities that the database offers.

In PostgreSQL, multiple values are stored as an array type (PostgreSQL Documentation,

[12] Using the usual `-xmx16GB` Java Virtual Machine (JVM) parameter.

[13] Virtuoso.ini already contains configurations for different memory setups, the suggested 16 GB includes `NumberOfBuffers = 1360000` and `MaxDirtyBuffers = 1000000`

2015). To implement the `GRAPH_LIKE`-query scenarios (Table 3.9 on page 112), there are a number of options. Using native array operators (*PostgreSQL 9.1.18 Documentation*, 2015, Section 9.17. Array Functions and Operators) such as contains (`<>`) or `unnest` turned out to be slower than using dedicated materialised views. So, additional views are created exclusively for `GRAPH_LIKE`-query scenarios. The time required to create and index this view is measured in the prepare phase (Section 3.2.2 on page 113) of the first `GRAPH_LIKE`-query scenario.

Neither of the two SQLite implementations provide support for non-trivial, embedded data types. For both, the Java tooling creates JSON structures to store multiple field values. For the `GRAPH_LIKE`-query scenarios, dedicated tables is are produced during the prepare phase (Section 3.2.2 on page 113).

4.2.5 MARC versus RDF

The original intension was to compare integration and parsing efforts associated to the different serialisation formats of the HeBIS catalogue, RDF and MARC. Thereby, it is important that both dumps produce the same logical entities.

Reading and semantically interpreting data from RDF is self-explanatory. MARC, in contrast, requires at least reading the online handbook (Library of Congress, 2015) to understand how to treat data strings such as `008 110719s1984 xx u00 u und c`. Using this handbook, a MARC parser is developed, containing nested conditions and cross-dependencies between various fields[14]. However, after a reasonable time of work, the MARC parser still did not produce entities that were completely identical to entities coming from the RDF dump. As both source datasets must be considered identical according to HeBIS, this is most likely due to remaining issues in the conditions or additional, as-yet-unexplored cross-dependencies between MARC fields.

As a consequence, reading and parsing of dumps was limited to the RDF representations of the HeBIS catalogue. Thus, native RDF stores, Virtuoso and Fuseki, have a theoretical advantage over the other, non-native RDF stores.

[14] See "Entity Representation" on page 365 for further information and complete queries.

The HeBIS catalogue is available as XML-serialised RDF, compressed using Gnu ZIP (GZIP). The delivery from HeBIS contained several dumps, each containing information about the encoding in its first 50 bytes:

```
Content-Type:  application/rdf+xml; charset=UTF-8
```

This line caused problems with all parsers involved in this evaluation (Jena, Fuseki, and Virtuoso), so it was removed from all dumps before the evaluation.

Fuseki and Virtuoso are both able to load RDF dumps directly. Virtuoso is in addition able to load GZIP compressed dumps.

For all other databases, a common, streaming RDF parser [15] was implemented and used. Existing Semantic Web parser libraries such as Jena are of only limited help in doing so, as they first build up the entire graph into memory before they allow working with data. When handling a large RDF dump, aiming at the transformation of RDF data to other formats, this behaviour is inefficient, error-prone, and not developer friendly. The parser developed here thus focusses on the assignment of field names and values and the production of records, and does not build up the mentioned graph model.

4.2.6 Overview of Measurement Results

Tables 4.7 to 4.10 on the following pages depict the measurement results of all databases. Full results are attached in the Appendices B.12 to B.15 on page 370, on page 384, on page 398 and on page 412, as well as published in the GitHub repository lodwhbench[16].

In some cases, errors occurred during load or query execution. In the Tables 4.7 to 4.10, these are indicated with the field value `Error`. If the error also effects subsequent query executions, these are indicated as `n/a`. Full error logs are attached in the appendices B.16.1 to B.16.6.

[15] See "Helper Methods" on page 358 for further information and complete queries.
[16] `https://github.com/heussd/loddwhbench/tree/master/results`, last access on 2016-02-15.

QueryScenario	sqlite4java	SQLite-Xerial	PostgreSQL	Virtuoso	Fuseki	MongoDB	ArangoDB
Load	329.39 ms	266.72 ms	500.93 ms	**121.22 ms**	1061.05 ms	580.58 ms	726.13 ms
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	1.38 ms	**0.23 ms**	1.02 ms	63.13 ms	34.00 ms	0.82 ms	2.80 ms
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	1.10 ms	**0.47 ms**	2.53 ms	167.30 ms	38.52 ms	6.15 ms	22.13 ms
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	2.68 ms	**2.05 ms**	5.19 ms	358.24 ms	133.77 ms	55.90 ms	173.48 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10	0.68 ms	**0.28 ms**	0.99 ms	0.46 ms	28.55 ms	4.75 ms	2.35 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100	0.62 ms	**0.31 ms**	1.39 ms	0.59 ms	12.56 ms	1.95 ms	2.47 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL	0.62 ms	**0.28 ms**	0.93 ms	0.62 ms	13.50 ms	1.86 ms	2.29 ms
AGGREGATE_ISSUES_PER_DECADE_TOP10	**0.89 ms**	1.70 ms	1.17 ms	2.87 ms	19.63 ms	1.85 ms	21.49 ms
AGGREGATE_ISSUES_PER_DECADE_TOP100	**0.91 ms**	1.55 ms	1.22 ms	2.88 ms	14.80 ms	1.84 ms	12.92 ms
AGGREGATE_ISSUES_PER_DECADE_ALL	**0.96 ms**	1.55 ms	1.34 ms	2.84 ms	11.95 ms	1.84 ms	13.95 ms
CONDITIONAL_TABLE_SCAN_ALL_STUDIES	**2.29 ms**	2.89 ms	2.45 ms	231.78 ms	133.41 ms	2.97 ms	3.39 ms
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES	15.26 ms	16.23 ms	14.46 ms	1962.31 ms	757.57 ms	**8.54 ms**	47.13 ms
CONDITIONAL_TABLE_SCAN_ALL_BIB._RES._AND_STUDIES	**2.25 ms**	3.36 ms	2.45 ms	231.65 ms	288.08 ms	3.15 ms	13.36 ms
GRAPH_LIKE_RELATED_BY_DCTERMS_SUB._1HOP_ONE_ENTITY	0.10 ms	**0.09 ms**	0.80 ms	0.72 ms	7.91 ms	2.78 ms	4.97 ms
GRAPH_LIKE_RELATED_BY_DCTERMS_SUB._1HOP_10_ENTITIES	**0.10 ms**	0.12 ms	0.72 ms	1.13 ms	8.35 ms	19.65 ms	31.05 ms
GRAPH_LIKE_RELATED_BY_DCTERMS_SUB._1HOP_100_ENTITIES	**0.10 ms**	0.12 ms	0.96 ms	1.12 ms	6.82 ms	17.61 ms	212.67 ms
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	0.95 ms	**0.83 ms**	32.83 ms	158.15 ms	236.66 ms	10.27 ms	53.24 ms
SCHEMA_CHANGE_INTRODUCE_STRING_OP	7.52 ms	8.60 ms	27.75 ms	**2.17 ms**	7.66 ms	187.42 ms	66.43 ms
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	**12.17 ms**	16.79 ms	135.54 ms	1010.09 ms	368.64 ms	17.36 ms	234.75 ms
SCHEMA_CHANGE_REMOVE_RDF_TYPE	4.43 ms	7.41 ms	16.27 ms	**0.21 ms**	5.33 ms	3.38 ms	58.49 ms
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	5.32 ms	9.39 ms	40.54 ms	20.29 ms	33.38 ms	**3.30 ms**	57.54 ms
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	1.00 ms	0.78 ms	2.04 ms	2.08 ms	73.33 ms	**0.58 ms**	9.66 ms
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	**2.56 ms**	5.32 ms	2.73 ms	512.00 ms	5.31 ms	5.45 ms	4.21 ms
DELETE_HIGH_SELECTIVIY_NON_ISSUED	0.06 ms	**0.04 ms**	1.28 ms	2.14 ms	3.67 ms	0.26 ms	0.69 ms

Table 4.7: Measurement results of the test series TINY, containing 1,000 records. Best performances are highlighted bold.

QueryScenario	sqlite4java	SQLite-Xerial	PostgreSQL	Virtuoso	Fuseki	MongoDB	ArangoDB
Load	21536.39 ms	21137.56 ms	28130.98 ms	**7939.37 ms**	30072.75 ms	31006.20 ms	37685.51 ms
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	1.38 ms	**0.31ms**	1.14 ms	566.57 ms	41.63 ms	61.52 ms	127.41ms
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	0.94 ms	**0.66 ms**	2.84 ms	6164.64 ms	48.79 ms	451.85 ms	1283.15ms
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	2.38 ms	**2.06 ms**	4.37 ms	54452.35 ms	153.49 ms	4643.60 ms	12976.55 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10	134.64 ms	32.45 ms	41.04 ms	**21.84 ms**	174.15 ms	83.63 ms	158.19ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100	143.94 ms	32.09 ms	41.83 ms	**27.45 ms**	81.54 ms	83.09 ms	145.58ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL	154.91 ms	**40.26 ms**	50.25 ms	116.82 ms	458.24 ms	110.77 ms	301.44 ms
AGGREGATE_ISSUES_PER_DECADE_TOP10	174.19 ms	225.04 ms	**46.10 ms**	136.46 ms	166.90 ms	127.21 ms	399.98ms
AGGREGATE_ISSUES_PER_DECADE_TOP100	164.77 ms	228.12 ms	**44.40 ms**	133.34 ms	144.22 ms	114.53 ms	383.72ms
AGGREGATE_ISSUES_PER_DECADE_ALL	161.50 ms	226.51 ms	**46.88 ms**	143.11 ms	112.12 ms	114.41 ms	375.78ms
CONDITIONAL_TABLE_SCAN_ALL_STUDIES	207.82 ms	299.94 ms	**130.40 ms**	16675.65 ms	7521.54 ms	217.29 ms	152.86ms
CONDITIONAL_TABLE_SCAN_ALL_BIB._RESOURCES	980.37ms	1189.01 ms	1031.39 ms	2.99E5 ms	65087.94 ms	**515.95ms**	3203.18 ms
CONDITIONAL_TABLE_SCAN_ALL_BIB._RES._AND_STUDIES	250.47 ms	339.87 ms	**128.86 ms**	13504.67 ms	23235.87 ms	267.11 ms	747.21 ms
GRAPH_LIKE_REL._BY_DCTERMS_SUB._1HOP_ONE_ENTITY	0.31 ms	**0.19 ms**	0.82 ms	0.78 ms	136.78 ms	269.81 ms	73.29 ms
GRAPH_LIKE_REL._BY_DCTERMS_SUB._1HOP_10_ENTITIES	0.62 ms	**0.54 ms**	7.73 ms	1.43 ms	7.61 ms	3042.83 ms	679.41 ms
GRAPH_LIKE_REL._BY_DCTERMS_SUB._1HOP_100_ENTITIES	15.95 ms	13.22 ms	**8.55 ms**	13.61 ms	122.48 ms	41676.83 ms	6267.49 ms
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	1.67 ms	**0.82 ms**	680.37 ms	16271.80 ms	12291.21 ms	730.76 ms	7259.30 ms
SCHEMA_CHANGE_INTRODUCE_STRING_OP	776.19 ms	899.90 ms	4984.46 ms	**489.95 ms**	524.28 ms	10512.26 ms	6907.28ms
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	**1619.26 ms**	2103.30 ms	14700.13 ms	91227.25 ms	29899.91 ms	2156.42 ms	28700.81 ms
SCHEMA_CHANGE_REMOVE_RDF_TYPE	442.17 ms	576.15 ms	20.84 ms	**0.19 ms**	8.06 ms	404.88 ms	5876.80ms
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	529.73 ms	908.66 ms	2348.41 ms	3837.52 ms	1964.35 ms	**346.10 ms**	6928.84ms
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	82.10 ms	**28.92 ms**	105.60 ms	328.60 ms	4643.03 ms	60.63 ms	1073.74 ms
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	305.24 ms	617.56 ms	423.15 ms	Error[a]	**6.33 ms**	442.88 ms	1213.62ms
DELETE_HIGH_SELECTIVIY_NON_ISSUED	8.94 ms	4.64 ms	**2.14 ms**	11342.42 ms	3.56 ms	5.95 ms	18.37ms

[a]Error occurred during query execution, see appendix B.16.4 on page 427 for full error log.

Table 4.8: Measurement results of the test series SMALL, containing 100,000 records. Best performances are highlighted bold.

QueryScenario	sqlite4java	SQLite-Xerial	PostgreSQL	Virtuoso	Fuseki	MongoDB	ArangoDB
Load	4.87E5 ms	4.47E5 ms	6.13E5 ms	**2.12E5 ms**	Error[a]	6.88E5 ms	8.61E5 ms
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	2.73 ms	**0.52 ms**	1.21 ms	9382.19 ms	n/a[a]	999.25 ms	3235.56 ms
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	2.23 ms	**1.68 ms**	2.17 ms	1.89E5 ms	n/a[a]	9527.43 ms	32524.25 ms
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	10.85 ms	**2.59 ms**	3.91 ms	1.37E6 ms	n/a[a]	99333.86 ms	3.23E5 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10	3958.27 ms	795.35 ms	1138.18 ms	**737.28 ms**	n/a[a]	4043.91 ms	5209.33 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100	3992.14 ms	838.87 ms	1174.99 ms	**724.22 ms**	n/a[a]	3870.72 ms	5229.35 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL	4273.47 ms	**1112.13 ms**	1592.18 ms	3018.93 ms	n/a[a]	5229.01 ms	8573.41 ms
AGGREGATE_ISSUES_PER_DECADE_TOP10	3992.82 ms	5587.86 ms	**975.43 ms**	2456.79 ms	n/a[a]	3063.77 ms	8667.55 ms
AGGREGATE_ISSUES_PER_DECADE_TOP100	4269.83 ms	5626.93 ms	**1034.89 ms**	2489.62 ms	n/a[a]	2758.35 ms	8746.80 ms
AGGREGATE_ISSUES_PER_DECADE_ALL	4025.16 ms	5580.76 ms	**1010.88 ms**	2562.25 ms	n/a[a]	2920.50 ms	8668.04 ms
CONDITIONAL_TABLE_SCAN_ALL_STUDIES	3842.06 ms	4965.95 ms	1989.96 ms	3.42E5 ms	n/a[a]	3676.14 ms	**1140.15 ms**
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES	25422.12 ms	27719.41 ms	29016.58 ms	9.01E6 ms	n/a[a]	**15985.97 ms**	79574.47 ms
CONDITIONAL_TABLE_SCAN_ALL_BIB._RES._AND_STUDIES	4796.04 ms	5815.28 ms	**2164.36 ms**	3.52E5 ms	n/a[a]	4907.33 ms	14780.01 ms
GRAPH_LIKE_RELATED_BY_DCTERMS_SUB._1HOP_ONE_ENTITY	69.35 ms	32.01 ms	**0.82 ms**	2.30 ms	n/a[a]	7374.98 ms	1993.19 ms
GRAPH_LIKE_RELATED_BY_DCTERMS_SUB._1HOP_10_ENTITIES	119.51 ms	81.21 ms	**0.90 ms**	27.33 ms	n/a[a]	87055.70 ms	16144.05 ms
GRAPH_LIKE_RELATED_BY_DCTERMS_SUB._1HOP_100_ENTITIES	1702.85 ms	1436.59 ms	2412.83 ms	**336.56 ms**	n/a[a]	7.24E5 ms	1.49E5 ms
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	1.14 ms	**0.99 ms**	25215.62 ms	Error[b]	n/a[a]	17724.65 ms	3.44E5 ms
SCHEMA_CHANGE_INTRODUCE_STRING_OP	**21181.29 ms**	23606.17 ms	1.69E5 ms	Error[b]	n/a[a]	2.33E5 ms	3.35E5 ms
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	**66071.02 ms**	77057.79 ms	5.04E5 ms	Error[b]	n/a[a]	66910.28 ms	Error[c]
SCHEMA_CHANGE_REMOVE_RDF_TYPE	13795.24 ms	17329.30 ms	**57.96 ms**	Error[b]	n/a[a]	15041.54 ms	n/a[c]
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	17342.08 ms	25041.33 ms	1.57E5 ms	Error[b]	n/a[a]	**15549.93 ms**	n/a[c]
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	3341.25 ms	**2984.77 ms**	12806.84 ms	50413.15 ms	n/a[a]	3800.26 ms	n/a[c]
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	7494.29 ms	15184.78 ms	15042.79 ms	**2.57 ms**	n/a[a]	16511.32 ms	n/a[c]
DELETE_HIGH_SELECTIVIY_NON_ISSUED	281.05 ms	217.26 ms	**97.68 ms**	Error[b]	n/a[a]	659.22 ms	n/a[c]

[a] Timeout error occurred during load, see appendix B.16.6 on page 429 for complete error log. Subsequent query executions invalid due to incomplete load.
[b] Query execution errors occured during runtime, see appendix B.16.5 on page 428 for full error log.
[c] An internal error occurred during execution, see appendix B.16.1 on page 426 for full database log. Database was unusable for subsequent queries.

Table 4.9: Measurement results of the test series MEDIUM, containing 1,7 million records. Best performances are highlighted bold.

QueryScenario	sqlite4java	SQLite-Xerial	PostgreSQL	Virtuoso	Fuseki	MongoDB	ArangoDB
Load	1.27E6 ms	**1.19E6 ms**	1.64E6 ms	Error[a]	Error[b]	1.93E6 ms	2.32E6 ms
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	9.44 ms	**1.12ms**	2.93 ms	n/a[a]	n/a[b]	2604.03 ms	8014.28 ms
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	6.26 ms	**2.50 ms**	2.80 ms	n/a[a]	n/a[b]	24540.70 ms	81197.58ms
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	6.75 ms	**4.33 ms**	10.19 ms	n/a[a]	n/a[b]	2.46E5 ms	8.58E5ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10	10823.61 ms	**2151.83 ms**	2919.83 ms	n/a[a]	n/a[b]	9599.36 ms	14408.34ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100	11486.73 ms	**2283.26 ms**	3088.30 ms	n/a[a]	n/a[b]	9597.52 ms	14173.33ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL	11590.62 ms	**3124.37 ms**	3912.75 ms	n/a[a]	n/a[b]	Error[c]	22478.96ms
AGGREGATE_ISSUES_PER_DECADE_TOP10	11872.51 ms	14790.62ms	**2714.04 ms**	n/a[a]	n/a[b]	7096.29 ms	20785.49ms
AGGREGATE_ISSUES_PER_DECADE_TOP100	12492.21 ms	14176.74ms	**2675.42 ms**	n/a[a]	n/a[b]	7056.18 ms	20471.41ms
AGGREGATE_ISSUES_PER_DECADE_ALL	12842.47 ms	14162.48 ms	**2685.78 ms**	n/a[a]	n/a[b]	7208.70 ms	20592.58ms
CONDITIONAL_TABLE_SCAN_ALL_STUDIES	14250.52 ms	13250.13ms	7171.27 ms	n/a[a]	n/a[b]	10205.19 ms	**3115.19ms**
CONDITIONAL_TABLE_SCAN_ALL_BIB._RES.	64200.50 ms	55987.28 ms	70342.26 ms	n/a[a]	n/a[b]	**29403.51 ms**	1.76E5 ms
CONDITIONAL_TABLE_SCAN_ALL_BIB._RES._AND_STUDIES	12945.22 ms	13167.53 ms	**6119.93 ms**	n/a[a]	n/a[b]	10084.06 ms	39115.80 ms
GRAPH_LIKE_RELATED_BY_DCTERMS_SUB._1HOP_ONE_ENTITY	206.78 ms	48.64 ms	**1.20 ms**	n/a[a]	n/a[b]	17971.03 ms	4869.37 ms
GRAPH_LIKE_RELATED_BY_DCTERMS_SUB._1HOP_10_ENTITIES	376.90 ms	**135.84 ms**	1620.94 ms	n/a[a]	n/a[b]	2.07E5ms	44759.81 ms
GRAPH_LIKE_RELATED_BY_DCTERMS_SUB._1HOP_100_ENTITIES	4006.99 ms	**3084.81 ms**	7341.36 ms	n/a[a]	n/a[b]	1.81E6ms	4.38E5 ms
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	3.51 ms	**1.02 ms**	64333.54 ms	n/a[a]	n/a[b]	45648.67 ms	1.39E6ms
SCHEMA_CHANGE_INTRODUCE_STRING_OP	75640.81 ms	**66496.15 ms**	5.24E5 ms	n/a[a]	n/a[b]	6.18E5 ms	1.45E6ms
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	2.16E5 ms	1.94E5 ms	1.43E6 ms	n/a[a]	n/a[b]	**1.69E5 ms**	1.86E7ms
SCHEMA_CHANGE_REMOVE_RDF_TYPE	1.12E5 ms	39956.90 ms	**156.75 ms**	n/a[a]	n/a[b]	43014.13 ms	8.39E6ms
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	53093.87 ms	55434.40ms	3.48E5 ms	n/a[a]	n/a[b]	**30071.94 ms**	8.00E6ms
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	9160.28 ms	**5676.63 ms**	46110.89 ms	n/a[a]	n/a[b]	9358.86 ms	5.41E5ms
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	**17561.26 ms**	34682.37 ms	29052.12 ms	n/a[a]	n/a[b]	35961.72 ms	4.40E6ms
DELETE_HIGH_SELECTIVIY_NON_ISSUED	1950.12 ms	775.04 ms	**666.58 ms**	n/a[a]	n/a[b]	2821.43 ms	34492.86ms

[a]Load failed due to RDF Syntax errors, see appendix B.16.3 on page 427 for full error log. Subsequent queries invalid due to incomplete load.
[b]Timeout error occurred during load, see appendix B.16.6 on page 429 for complete error log. Subsequent query executions invalid due to incomplete load.
[c]Sorting failed in this query, see appendix B.16.2 on page 426 for full error log.

Table 4.10: Measurement results of the test series LARGE, containing 5.7 million records. Best performances are highlighted bold.

4.2.7 Stability of Operation

A first observation is the fact that errors occurred on all DBMSs except the relational approaches, reproducible for all kinds of dataset sizes. Even at very low data sizes such as 100,000 entries, one storage candidate, Virtuoso, ran into an error during query execution (see appendix B.16.4 on page 427).

With growing data sizes, the number of execution problems increases as well. With 1,7 million records, Virtuoso and ArangoDB show occasional but repeatable query execution issues (see appendices B.16.1 and B.16.5 on page 426 and on page 428). Fuseki cannot even load the dump with the selected loading method (see appendix B.16.6 on page 429).

Despite the issue in MEDIUM, ArangoDB loads the largest test series LARGE, consisting of 5.7 million records, flawlessly. A single issue can be observed for MongoDB (see appendix B.16.2 on page 426). Again, Fuseki times out during load (see appendix B.16.6 on page 429). Syntax errors in the LARGE data source prevent Virtuoso from loading it entirely (see appendix B.16.3 on page 427).

4.2.8 Loading Times

Table 4.11 shows the different loading times per database implementation.

Records	sqlite4java	SQLite-Xerial	PostgreSQL	ArangoDB	MongoDB	Virtuoso	Fuseki
LARGE	00:19:18	**00:19:02**	00:26:01	00:38:40	00:32:12	Error[a]	Error[b]
MEDIUM	00:07:22	00:06:43	00:09:56	00:14:21	00:11:28	**00:03:28**	Error[b]
SMALL	00:00:19	00:00:18	00:00:27	00:00:38	00:00:31	**00:00:08**	00:00:31
TINY	00:00:00	00:00:00	00:00:00	00:00:01	00:00:01	**00:00:00**	00:00:01

[a] Virtuoso failed to parse one of the two involved Datasets, see appendix B.16.3 on page 427.
[b] Fuseki Server reported a time-out, see appendix B.16.6 on page 429.

Table 4.11: Comparison of the loading times.

With all approaches, loading of 1,000 (TINY) or 100,000 entries (SMALL) only requires a few milliseconds or seconds. With larger dataset sizes, Virtuoso in particular can make use of its inherent advantage of loading data dumps directly: it is twice as fast as the fastest non-native RDF store. However, there is a critical weakness of this loading

method: even small syntax errors can invalidate the entire loading process. This happens in case of the LARGE test series: One of the two datasets involved has a syntax error, which causes Virtuoso to fail when loading it (see appendix B.16.3 on page 427 for the entire loading log). The other triple store, Fuseki, starts failing with MEDIUM data sizes because of timeouts (see appendix B.16.6 on page 429 for the loading log). Even though very large timeout thresholds are configured (Section 4.2.3 on page 139), they don't seem to be sufficient.

Given the fact that MongoDB does not provide transactional security, it seems surprising that it is not one of the fastest data loaders. ArangoDB is the slowest loader, requiring about 19% more time in loading the LARGE test series than the second slowest loader MongoDB.

Figure 4.10 compares the ratio of loading time and dataset sizes for each database. The relational approaches, with SQLite-implementations leading the way, seem to provide the best ratio between load throughput and load stability.

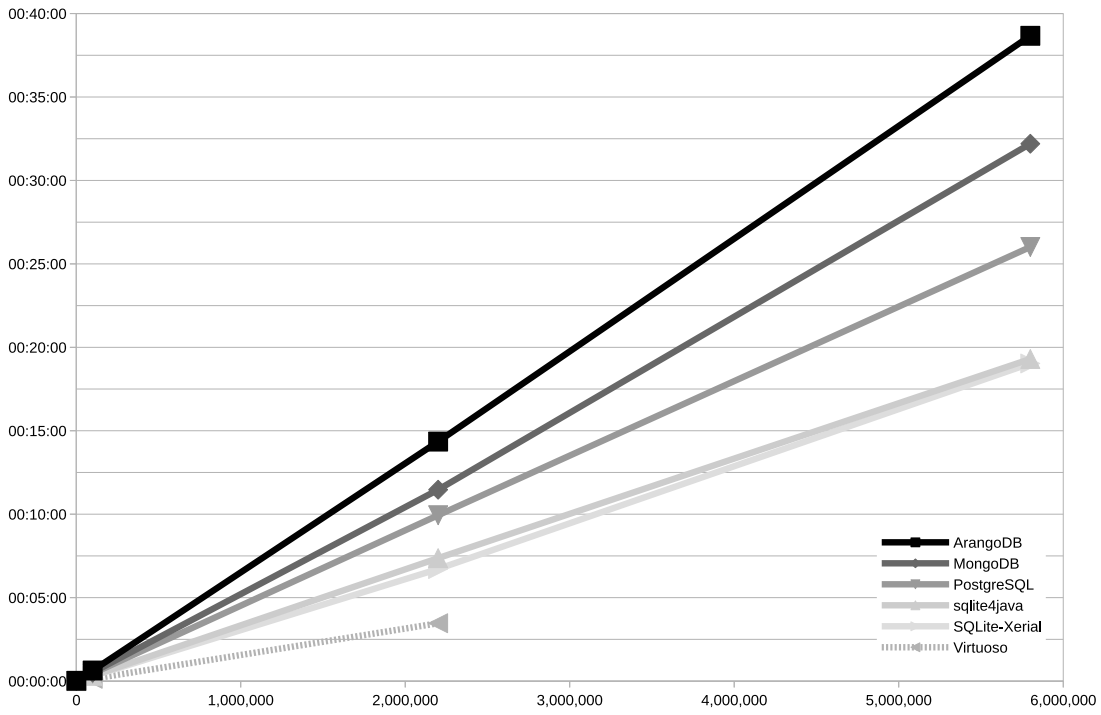


Figure 4.10: Loading Throughput versus Loading Time Consumption

4.2.9 Entity Retrieval

Entity retrieval scenarios evaluate how fast a storage system can retrieve complete logical entities that are relevant to the given specific application scenario. In a data warehouse scenario, entity retrieval might be one of the most common data operations. In case of the HeBIS catalogue, such an entity retrieval involves all data associated with a given bibliographic resource.

In this evaluation, the query scenarios prefixed with `ENTITY_RETRIEVAL` and `CONDITIONAL_TABLE_SCAN` require the creation of complete, logical entities (Table 3.9 on page 112). Different storage approaches provide different means for entity retrieval, depending on their storage philosophy.

Relational approaches already store requested entities. The construction of entities is part of the integration of data. Triple stores, in contrast, have to constitute entities on the fly, as they just store atomic triples. Thereby, usually, the SPARQL command `DESCRIBE` can be used to retrieve all triples associated to a given set of RDF subjects (which then constitute entities) (Prud'hommeaux and Seaborne, 2008).

This works flawlessly with Fuseki, allowing the Java parser to sequentially read and create entities. With Virtuoso, however, `DESCRIBE`-queries returned all triples of all involved subjects in an arbitrary order. This might be due to a bug in Virtuoso (GitHub Virtuoso Issue Tracker, 2015), known since January 2013. Due to this, it is impossible to read records sequentially, and thus efficiently from a seemingly-random ordered set of triples of multiple records.

To overcome this issue, a `SELECT` statement with a pre-defined column format was used (Listing 4.1). In cases where the fields might have multiple values, a comma-separated embedded serialisation format is produced, using `group_concat`. This serialisation format then needs to be merged with the Java logic.

```
select
 ?s
 (group_concat(distinct ?edition ; separator = ",") as ?edition)
 ?oclcum
 ?format
 ?identifier
```

```
?issued
?medium
?publisher
(group_concat(distinct ?subject ; separator = ",") as ?subject)
(group_concat(distinct ?title ; separator = ",") as ?title)
(group_concat(distinct ?contributor ; separator = ",") as ?contributor)
(group_concat(distinct ?P1004 ; separator = ",") as ?P1004)
?P1006
(group_concat(distinct ?P1008 ; separator = ",") as ?P1008)
(group_concat(distinct ?P1016 ; separator = ",") as ?P1016)
?P1017
?P1018
?sameAs
(group_concat(distinct ?type ; separator = ",") as ?type)
?describedby
```

Listing 4.1: Select clause of a SPARQL statement to workaround a possible `DESCRIBE` bug in Virtuoso with a `SELECT` statement.

Table 4.12 shows the detailed entity retrieval behaviour for the TINY and MEDIUM test series[17]. It can be observed that relational approaches outperform all other approaches by at least a factor of two.

Apparently, both the on-the-fly-constitution of entities (required to be done by triple stores) as well as the `DESCRIBE`-query-workaround (specific to Virtuoso) seems to impact entity retrieval efforts significantly: compared to the relational results, the query times differ by one order of magnitude for Fuseki, and Virtuoso even doubles that. With growing dataset sizes, this effect seems to be amplified on all non-relational storage technologies.

Both NoSQL approaches perform well on TINY, but perform significantly worse on MEDIUM datasets.

Database	**Approach**	**Retrieval of Ten of 1,000**		**Retrieval of Ten of 2,2 million**	
		Retrieval Time	σ	Retrieval Time	σ
sqlite4java	Relational	1.37 ms	2.08	2.74 ms	4.33
SQLite-Xerial	Relational	**0.23 ms**	**0.15**	**0.53 ms**	**0.59**
PostgreSQL	Relational	1.02 ms	0.81	1.21 ms	1.06
Virtuoso	Triple store	63.14 ms	102.30	9,382.19 ms	3890.74
Fuseki	Triple store	34.01 ms	27.32	-	-
MongoDB	Document store	0.82 ms	0.33	9,99.25 ms	62.21
ArangoDB	Multi model	2.80 ms	0.50	3,235.56 ms	14.03

Table 4.12: Average entity retrieval times (including the standard deviations σ) of ten predefined entities out of 1,000 and out of 2,2 million entities. Full measurement results are available in section appendix B.12 on page 370 and appendix B.14 on page 398.

[17] See Appendix B.12 on page 370 for all measurement results for test series TINY.

4.2.10 Caching

Ideally, caching allows a database to respond to repeated queries faster. In this evaluation, however, this behaviour is most varied across the different approaches. Table 4.13 compares the fasted query execution of each database with its slowest, for TINY and MEDIUM.

Database	Retrieval of Ten of 1,000			Retrieval of Ten of 2,2 million		
	Min. Time	Max. Time	Change	Min. Time	Max. Time	Change
sqlite4java	0.16 ms	3.77 ms	96%	0.2 ms	7.74 ms	**97%**
SQLite-Xerial	**0.14 ms**	**0.41 ms**	66%	**0.17 ms**	**1.21 ms**	86%
PostgreSQL	0.51 ms	1.95 ms	74%	0.57 ms	2.43 ms	77%
Virtuoso	3.82 ms	181.26 ms	**98%**	7069.5 ms	13874.17 ms	49%
Fuseki	18.2 ms	65.55 ms	72%	-	-	
MongoDB	0.62 ms	1.2 ms	48%	940.48 ms	1064.41 ms	12%
ArangoDB	2.39 ms	3.36 ms	29%	3226.04 ms	3251.67 ms	1%

Table 4.13: Three times repeated execution times for entity retrieval of ten predefined entities out of 1,000 and 2,2 million entities. Full measurement results are available in section appendix B.12 on page 370 and appendix B.14 on page 398.

Relational DBMS approaches can usually speed up queries at least by two thirds. sqlite4java displays best caching scores, with 96% and 97% query time reduction compared to their respective first query executions. A comparable caching performance can be observed for Virtuoso for TINY test series, though the caching falls to about 50% for MEDIUM. Fuseki showed a medium change in TINY. ArangoDB and MongoDB both show significantly lower performance changes in repeated query scenarios.

4.2.11 Conditional Table Scan

Just like the entity retrieval scenarios (Section 4.2.9 on page 148), conditional table scan scenarios retrieve complete entities relevant to the given application context. In addition, instead of the very specific selection criteria (such as an entity ID), the scenarios in the following match for about 2% in `CONDITIONAL_TABLE_SCAN_ALL_STUDIES` and about 92% `CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES` of the entities.

Figure 4.11 on the next page compares the measurement results for the MEDIUM and LARGE test series.

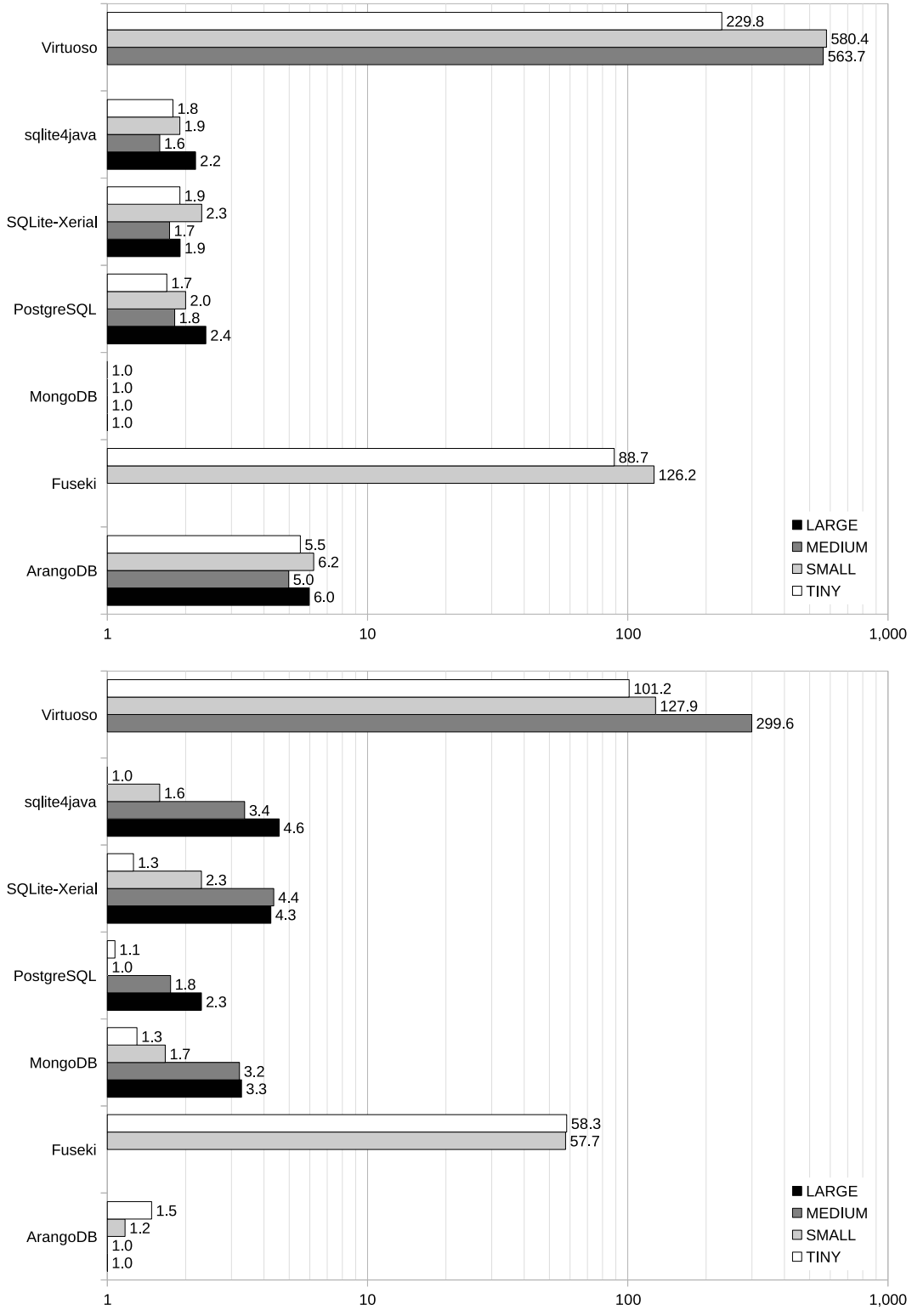


Figure 4.11: Comparison of average query times for the conditional table scan scenarios `CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES` (above) and `CONDITIONAL_TABLE_SCAN_ALL_STUDIES` (below), each normed by the best performer per dataset size.

All relational approaches perform similarly, differing only by a few milliseconds. Like before, they display stable operation at good performances.

MongoDB performs close to relational approaches. It can even outperform the best relational result in case of `CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES`. It seems that the more entities are involved, the relatively faster MongoDB can retrieve them, though this observation could not be made in the entity retrieval scenarios earlier (Section 4.2.9 on page 148).

Regarding `CONDITIONAL_TABLE_SCAN_ALL_STUDIES`, ArangoDB improved also, compared to its previous entity retrieval performance.

Triple store approaches show bad performances, just as they do in entity retrieval scenarios. Fuseki fails to process the requested query in three out of four cases (see Fuseki Error Log - Load of MEDIUM and LARGE Test Series on page 429 for full logs), and for the remaining query, the query time increases ten-fold compared to other approaches. Virtuoso completes all requested queries, though with worse performances. The performance differences with respect to the best competitor vary from three to five orders of magnitudes. Again, this might be due to the query workaround (Listing 4.1 on page 148).

Roughly said, previous findings regarding entity retrieval performances can be confirmed for conditional table scan scenarios: Relational approaches perform just as well, NoSQL approaches perform better, triple stores significantly worse.

4.2.12 Aggregation

In contrast to the complete record generation required in the entity retrieval scenarios, aggregation represents yet another class of scenarios where only certain fields of the stored data are relevant and processed using a group or sum operation.

Figure 4.12 on the next page visualises the measurement results of two aggregation scenarios. On the left, the aggregation is based on a calculated value, on the right, already stored record values are aggregated.

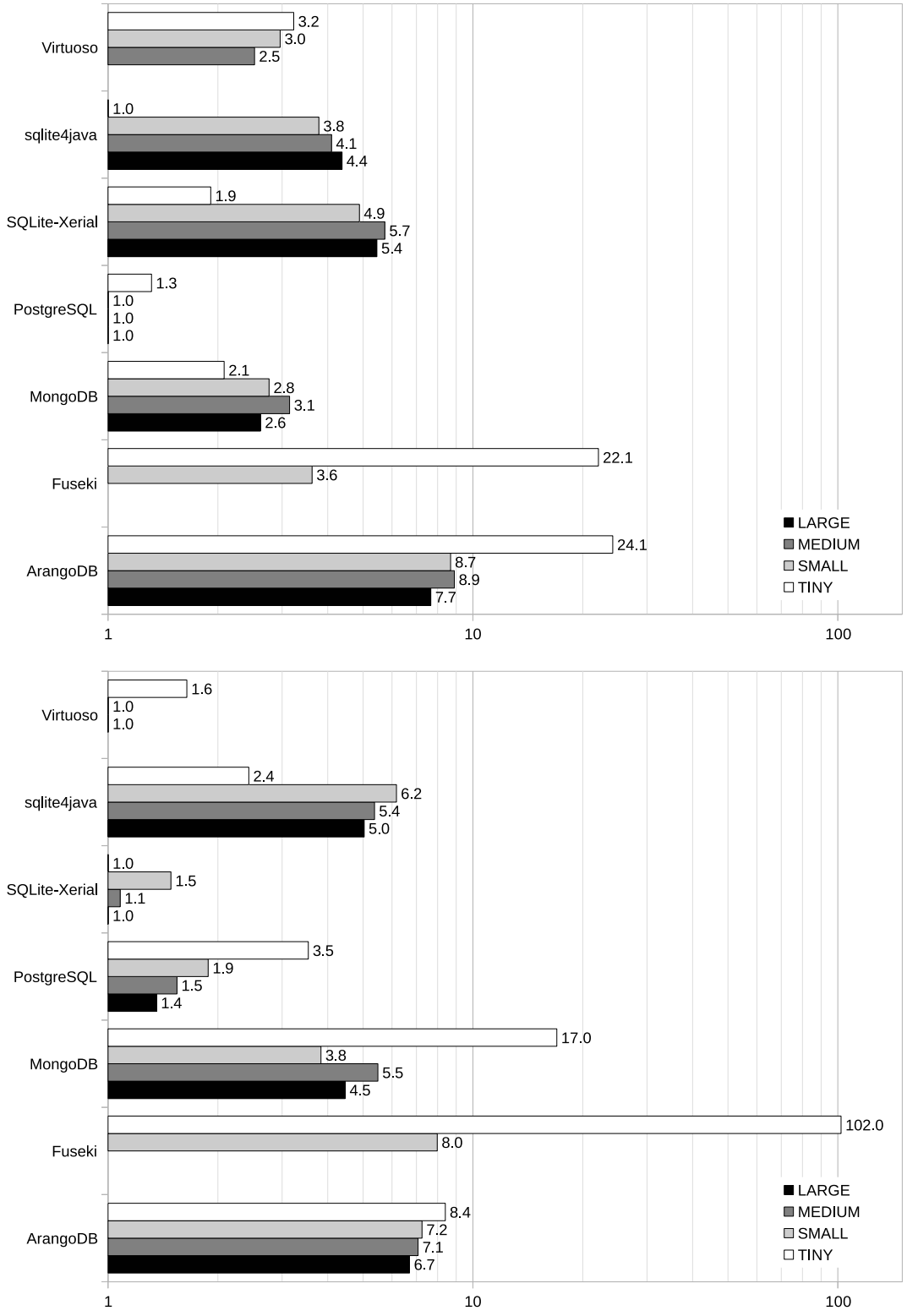


Figure 4.12: Comparison of average query times for the aggregation scenarios `AGGREGATE_ISSUES_PER_DECADE_TOP10` (above) and `AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10` (below), each normed by the best performer per dataset size.

Apparently, the fact that Virtuoso already stores atomic field information instead of complete records has benefits in both aggregation scenarios: the triple store is among the best aggregation performers.

Moreover, PostgreSQL displays a solid aggregation performance, independent from the fundamental test series and scenario. In contrast, query times in SQLite-based approaches vary: aggregation seems to be a principal weak point especially in sqlite4java, as well as for SQLite-Xerial when talking about the aggregation of immediately calculated values. When aggregating stored field values, SQLite-Xerial performs nearly as well as PostgreSQL.

MongoDB seems to be an average aggregation performer, displaying query times between the PostgreSQL and sqlite4java. The poor performance in the aggregation of existing field values in particular can be explained by the fact that MongoDB is a document store (Section 2.8.3 on page 84), and thus, the formation of single field values can be assumed to be more expensive than in schema-aware systems.

ArangoDB performs worst in all aggregation scenarios, taking about 4 to 5 times longer compared to the best performers. Just like MongoDB, ArangoDB is (also) a document store, which might be the reason for the worse performances. Using ArangoDB's graph store capabilities might have sped up the aggregation, however, this would have required the explicit upfront modelling of the data.

4.2.13 Graph Scenarios

Graph scenarios represent a further class of queries which also does not require formation of complete records. Instead, they operate on an n:m relation between `DCTERMS_IDENTIFIER` and `DCTERM_SUBJECT` (Table 3.5 on page 108).

Figure 4.13 on the facing page compares the query times for each graph scenario, each on different sizes of base data.

A first observation is the fact that Virtuoso and Fuseki perform naturally well in this query discipline. Moreover, all relational approaches (PostgreSQL, SQLite-Xerial and

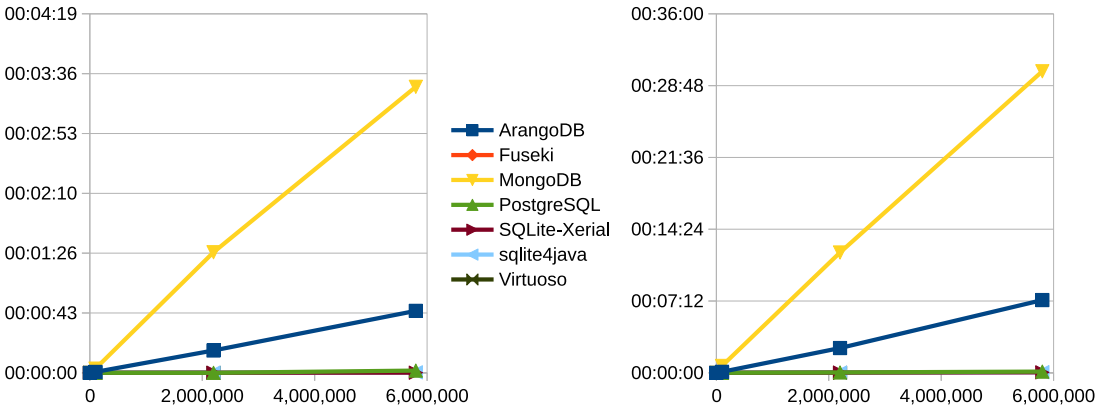


Figure 4.13: Average query times for the graph query scenarios `GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES` (left) and `GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES` (right), for all 1,000, 100,000, 2,2 million and 5,8 million records.

sqlite4java) can compete well, even though they are not dedicated graph stores. All mentioned databases process the graph scenarios within milliseconds, almost independent from the size of the base dataset and the number of requested graph relations.

MongoDB is not a natural graph store, and the fact that its a document store actually seems to hinder the structured, relation-based querying of data significantly. It can be observed that query times grow almost linearly, based on the number of base test series.

ArangoDB, in contrast, is a graph store. The preparation phase explicitly structures the data in ArangoDB so that graph query processing is possible. Therefore, the displayed performance is surprisingly bad, compared to the good performances of triple stores and relational DBMSs on the one hand and the non-performance of MongoDB on the other.

4.2.14 Change Operations

Besides read-only operations, certain write patterns are relevant to data warehouse scenarios, too. In the following, delete and update operations are compared. As mentioned earlier (Section 3.2.2 on page 110), these operations can be distinguished depending on their selectivity: Highly selective operations provide a very specific pre-selection of affected data, so very few records are affected by the operation. Low selective operations accordingly provide a very unspecific data selection criteria and the operation affects

large parts of stored information.

Table 4.14 shows the data manipulation execution times of each database, differences between the kind of operation and its selectivity, for the SMALL test series. In this evaluation, based on the HeBIS catalogue, high selectivity operations affect about 2% of the records, low selectivity operations about 90%.

Operation	Database	High Selectivity	Low Selectivity
Delete	ArangoDB	18,37 ms	1.213,62 ms
	Fuseki	3,61 ms	**5,25 ms**
	MongoDB	5,95 ms	442,88 ms
	PostgreSQL	**2,14 ms**	423,15 ms
	SQLite-Xerial	4,64 ms	617,56 ms
	sqlite4java	8,94 ms	305,24 ms
	Virtuoso	11.342,42 ms	Error
Update	ArangoDB	1.073,74 ms	6.928,84 ms
	Fuseki	170.125,08 ms	Error
	MongoDB	60,63 ms	**346,10 ms**
	PostgreSQL	105,60 ms	2.348,41 ms
	SQLite-Xerial	**28,92 ms**	908,66 ms
	sqlite4java	82,10 ms	529,73 ms
	Virtuoso	328,60 ms	3.837,52 ms

Table 4.14: Data manipulation execution times, update versus delete, high versus low selectivity, SMALL test series (100,000 records).

At first glance, it is noticeable that update operations are always more expensive than delete operations, and that low selectivity is always more expensive than high selectivity. These two rules of thumb can be observed across all different database approaches.

Even though most databases perform quite differently, MongoDB is one of the best in all these operations. This might be due to the fact that MongoDB is the only database in this evaluation that does not provide full support of ACID, but rather eventual consistency.

Relational approaches (PostgreSQL, sqlite4java, and SQLite-Xerial) form a solid mid-range of data manipulation performances. The performances of remaining approaches

strongly vary, e.g. in the case of Fuseki, from the by-far-best performance in low selectivity delete operations, to errors in low selectivity update operations.

4.2.15 Summary of the Findings

Because existing benchmarks in the Web Science domain leave open questions, a novel comparison of application-specific query scenarios are executed on a number of current storage choices, involving relational DBMS, triple stores and two NoSQL approaches. Results are critical information for the design and the technological choice of a data warehouse.

Generally speaking, relational DBMS (PostgreSQL, sqlite4java and SQLite-Xerial) perform well or best on almost all scenarios, provide the most stable operation, and can load data of all compared sizes with adequate speeds and offer the best ratio of throughput and load-stability. They are the first choice whenever application-specific entities are involved in the queries, yet they can compete in scenarios where this is not the case - even in simple graph scenarios. They show best caching behaviours and display suitable performances for data manipulation. In detail, PostgreSQL, sqlite4java and SQLite-Xerial roughly perform analogously in TINY, SMALL and MEDIUM sizes, though there are significant differences in certain scenarios such as manipulation (Section 4.2.14 on page 155) and aggregation (Section 4.2.12 on page 152).

For non-relational approaches, the performance strongly depends on details of the query scenario in specific technological architectures.

Triple stores can load native RDF most rapidly, as long as the data is 100% free of syntax errors. As pointed out earlier, this assumption does not hold true for findings in existing research (Section 2.2.4 on page 54) as well as for the actual data analysed at datahub.io (Section 4.1.5 on page 126). Once loaded, triple stores perform well on certain aggregation scenarios (Section 4.2.12 on page 152), and in graph scenarios, though the performance is roughly comparable to the performance of relational DBMS. In scenarios with application-specific entities, triple stores are outperformed by one up to two orders of magnitudes, making them basically unsuitable for these scenarios.

With its document store approach, MongoDB performs well and partially best for the mass-retrieval of application-specific entities (Section 4.2.11 on page 150) and in some data manipulation scenarios. In other scenarios, MongoDB lacks the flexibility to optimise data storage towards a scenario, which results in comparable bad performances (Section 4.2.13 on page 154).

ArangoDB performs even better in a certain mass-retrieval scenario, but worse in the others (Section 4.2.11 on page 150). Even though data had been prepared for the graph scenarios, results are not as good as for triple stores and relational DBMSs, though not as bad as MongoDB (Section 4.2.13 on page 154).

Based in these findings, a relational DBMS seems to be the most suitable technological foundation for a data warehouse in the given scenario. Results also indicate that the polyglot persistence paradigm seems to be justified, as certain database technologies show, in detail, most different performance. Thereby, the range is from unsuitable performance up to best performance, depending on query scenarios, data preparation, and the capabilities of the database. The niches in which a specialised storage technology is superior might, however, be quite small in practice.

4.3 Development of a Linked and Open Data Integration Framework

Previously, in a field study about data on the representative data portal datahub.io, it was pointed out that structured Linked and Open Data mainly consists of tabular data and is serialised in ordinary formats such as spreadsheets and CSVs (Section 4.1 on page 122).

Existing research and reference architectures suggests to integrate this data in triple stores, and existing integration frameworks are based on this assumption. In the conducted storage evaluation (Section 4.2 on page 134), in contrast, it could be shown that relational DBMSs display better performances in almost all relevant data integration and usage scenarios.

In regard of this previous work, in this chapter, a novel framework for the integration of Linked and Open Data specific applications is developed. It is a pragmatic, developer-oriented approach, providing features that to handle the lifecycle of Linked and Open Data in application-specific scenarios:

- Support common steps of data integration towards end-user applications, recommended in research and practice.
- Support the developer in their development workflow, so that they can focus on solving application-specific challenges rather than building up baseline functionality for routine tasks.
- Support a multiplicity of Linked and Open Data formats and today's distribution channels for Linked and Open Data, such as CKAN-based repositories or SPARQL endpoints.

After comparing existing approaches, based on identified gaps, use cases and requirements for the framework are defined. In the remainder of this chapter, a framework matching these requirements is designed.

4.3.1 Comparison of Existing Approaches

As previously found, data often needs to be integrated, stored, and customised, before it can delivered to - and actually be used - in an end-user ready (Web) application. Thus, in the practice of application specific scenarios, technical data integration solutions need to cover (at least) these five steps.

With the background of the previous findings, the following existing data integration solutions and approaches are analysed:

Mediaplatform , the research project, is an example of a real-world integration of Linked Data and Open Data towards actually working, high-usability (Deuschel, Greppmeier, Humm and Stille, 2014) and launched[18] applications.

Catmandu is a Perl-based Data integration tool, originating from the bibliographic context (Section 2.6.5 on page 74).

Apache Marmotta is an Java-based Linked Data integration framework (Section 2.6.6 on page 75), aiming at building semantic search applications.

Plain Semantic Web represents the mindset of how to build apps according to existing research (Section 2.7 on page 76).

Table 4.15 on the next page compares these approaches, in regard of how they cover the identified integration steps (Table 3.10 on page 115) in their architecture.

[18] Städel application was launched in the beginning of 2015, `https://digitalesammlung.staedelmuseum.de/index.html`, last access on 2015-09-29

	ETL	**Storage**	**Customisation**	**Delivery**	**Use**
Tasks	Loading multiple formats, filtering, duplicate & quality checks, enrichment, applying application specific schemata	Storing data in application specific entities, source for differential data loads, linking and enrichment of data	Implementing application semantics with the stored entities, data transformation in use-case-oriented formats	Storing data in use-case oriented formats, providing query interfaces / APIs to the data	Using the prepared data in pre-defined application scenarios, e.g. in user-friendly GUIs
	Existing Integration Frameworks and their architectural solution for each of the previously identified integration steps				
Mediaplatform	Load of local file dumps, custom Java code	custom Java code, re-use of Delivery-storage	custom Java code	Application specific API, protocol (JSON) and storage (Lucene)	JavaScript-Entity-Binding frameworks
Catmandu	Load of local file dumps, Fix (Perl scripts)	Fix (Perl scripts)	Fix (Perl scripts)	MongoDB / ElasticSearch	JavaScript
Apache Marmotta	Linked Open Data Cloud, transparently cached using LDCache	triple store (abstraction layer on relational storage)	LDPath	Solr	JavaScript
Plain Semantic Web	Linked Open Data Cloud – "follow your nose" (Hausenblas, 2009, p. 71f)	triple store	-	SPARQL endpoints	(JavaScript) SPARQL-Client

Table 4.15: Architecture analysis of common, Linked and Open Data integration approaches (Section 2.6 on page 68).

ETL

In the ETL step, external or local data is extracted, transformed and loaded into application-specific entities.

Plain Semantic Web as well as Marmotta integrate RDF-based data only. Both use SPARQL or - in case of Marmotta - LDPath to do the extraction and transformation of required data. This represents the pay-as-you-go approach (Section 2.2.8 on page 56), in which existing Linked Data sources can "natively and simply" be reused across the Web (Heath and Bizer, 2011, p. 106).

As the previous field study shows, by just relying on RDF, both approaches are significantly limited. In the case of datahub.io, only about 11% of the data is actually in the RDF format (Section 4.1.1 on page 123). In order to load non-RDF-data with these tools, an additional RDF conversion, "minting" (Hausenblas, 2009, p. 70f), is required - a process that must be considered to be error-prone (Latif et al., 2009, p. 7).

Moreover, even if only RDF is about to be loaded, there are many factors preventing that data from being directly used in applications, such as quality issues (Section 2.2.4 on page 54) or transfer and parser problems (Section 4.1.5 on page 126). The transparent caching mechanism of Marmotta seems to partly address this second issue, with the drawback of losing the control over the import process. With both approaches, integrated entities must be stored in the atomic, general purpose RDF format.

Both Mediaplatform and Catmandu, in contrast, put special emphasis on transforming data. This allows developers to introduce their application-specific entities using custom Java code or a Perl-based DSL to transform data. With Mediaplatform and also with Catmandu, handling data formats is up to developers, which also includes implementing even basic and routine tasks for retrieval, reading, storing, and updating. This is error-prone and must be repeated in every new application. This is not optimal, as end users do not directly benefit from it and it takes developers longer to produce end-user features.

Storage

Storing is about persisting data in their integrated, application-specific entities. As such, the create store is the source for differential data loads as well as the reprocessing and enrichment of data.

Once integrated, Marmotta and the Plain Semantic Web approach store data as RDF in triple stores. Because of the entity-agnostic nature of the technology stack (Section 2.2.2 on page 53), both approaches conceptually do not provide capabilities for the storage of application-specific entities (Section 2.2.2 on page 53). In application-specific scenarios, however, there are application-specific entities. Consequently, triple stores have to create entities on-the-fly. In the previous evaluation, it is shown that this is unfeasible with current triple stores implementations, as they are outperformed by orders of magnitude in respective scenarios such as entity retrieval (Section 4.2.9 on page 148) and conditional table scans (Section 4.2.11 on page 150).

Catmandu and Mediaplatform address the further entity integration with custom code, either with Perl or with Java. Both lack a data warehouse-like storage to offer a delivery-independent access to integrated entities. Warehousing data is a common best practice (Section 2.9.5 on page 90), and is especially important for scenarios where data needs to be delivered in different formats, when implementing a polyglot persistence architecture (Section 2.8.3 on page 83), when the enrichment is more complicated or when the loads should operate differentially.

Besides that, it is important to note that the plain storage of data still does not meet expectations that customers today have from semantic applications. Storing is a necessary, but usually unpaid job, so efforts should be minimal. This involves a technological footprint, too. Storing should not introduce technological constraints, expensive or complex infrastructure, and should ideally not introduce run-time dependencies to other local or external services.

Customisation

In the next step, customisation, the data is streamlined towards the use cases of the application, such as indexing of entities using a text retrieval framework like Lucene or Solr to provide the application with a search feature. This is where developers actually do things they are paid for: they implement the semantic logic of a semantic application.

In the plain Semantic Web mentality, data conceptually does not need to be customised. This lack of preprocessing impacts subsequent phases, delivery (Section 4.3.1) and use (Section 4.3.1 on the facing page).

Marmotta, in contrast, provides a custom DSL, LDPath, to offer the developer a conformable way to customise the data. As previous steps are limited to RDF, the customisation in Marmotta is limited to RDF as well.

Catmandu, again, makes use of its DSL to offer the developer a customisation. The Mediaplatform uses custom Java code. Both approaches have introduced application-specific entities earlier, so developers can work smoothly in their domain with domain specific vocabulary.

Delivery

Delivery is about providing use-case oriented formats, query interfaces and APIs to data.

As a result of the customisations, the delivery phase of Mediaplatform, Catmandu as well as Marmotta is built in an application-specific way. All three approaches focus on building semantic search applications, thus, all three approaches use an information retrieval framework (Lucene, Solr, ElasticSearch) to delivery their data towards end users. This use-case oriented delivery is a central idea of polyglot persistence (Section 2.8.3 on page 83) and is considered to be best practice in modern development.

In contrast, the Plain Semantic Web approach offers a generic SPARQL endpoint. Application-specific entities must be constituted on-the-fly. In the previous evaluation,

it is shown that this is unfeasible with current triple stores implementations, as they are outperformed by order of magnitudes in respective scenarios such as entity retrieval (Section 4.2.9 on page 148) and conditional table scans (Section 4.2.11 on page 150).

Use

In the use step, data is used pre-defined application scenarios.

In modern Web development, this usually happens with additional scripting languages such as JavaScript. In the last years, an rich ecosystem for frameworks based on JavaScript has developed, for example entity binding frameworks like AngularJS (Section 2.8.2 on page 83). All solutions are streamlined towards using simple, application-specific interfaces (usually based on REST) and simple entity serialisation formats (usually JSON).

4.3.2 Gap Analysis

Based on existing framework, the phases ETL, Storage and Customisation are not covered ideally by existing approaches.

An ideal solution would assist developers in routine tasks, and allow them to introduce custom, application-specific entity-formats. This rapidly enables them to solve application-specific problems that are demanded by the end user, without having to solve baseline technological challenges first. Instead, Mediaplatform and Catmandu requires upfront investments, and Marmotta and Plain Semantic Web are substantially limited.

The storage layer should be nearly invisible, even for developers. The storage should be manageable by a given framework, in best case in-app, with working default settings to ensure maintainability and testability, without requiring the developer to learn new technologies.

An ideal solution would provide an ecosystem for developers to work with application-specific entities, e.g. like using a DSL similar to LDPath, without being limited to RDF, or a popular programming language such as Java.

4.3.3 Requirements

Previously, it is pointed out that data integration solutions are only partly useful for application-specific scenarios. Especially the steps ETL, Storage and Customisation should be supported by a generic framework, offering the features:

ETL provide capabilities for routine tasks to extract, transform, filter, enrich, and load data, allow easily customisation by developers, introduce application-specific entities

Storage provide a streamlined, developer-friendly storage layer for warehousing data.

Customisation provide an ecosystem for developers to work with application-specific entities, e.g. to convert them into other storage forms.

It can be expected that for the other steps, delivery and use, there is already-existing and industry-ready software support so that they do not need to be covered by a new framework. Besides supporting these basic integration steps, the following properties are required for data integration frameworks for Linked and Open Data:

Small technological footprint To simplify usage and to leave the freedom of choice to developers, the framework should set as few technological constraints as possible. This includes run-time dependencies to the operating system, such as application or database servers.

Developer friendly The developer should be able to focus on their goal, the application-specific development of a semantic application. They should not need to spend time in solving repeated, default functionality.

Schema-aware entities Applications for specific contexts usually have a common perception of what certain entities are. The framework should support creation, serialisation, and validation of these entities.

4.3.4 The LODicity Framework

LODicity (artificial compound of Linked and Open Data and velocity) is a novel integration framework for Linked and Open Data. It addresses the the lack of pragmatic, developer-oriented approaches in the development of application-specific solutions based on Linked and Open Data.

As shown in Table 4.16, the framework helps developers implementing the first three steps of a Linked and Open Data integration chain (Section 3.2.3 on page 114), by providing capabilities for the steps ETL, storage, and customisation.

<table>
<tr><th>ETL</th><th>Storage</th><th>Customisation</th><th>Delivery</th><th>Use</th></tr>
<tr><td>Loading multiple formats, filtering, duplicate & quality checks, enrichment, applying application specific schemata</td><td>Storing data in application specific entities, source for differential data loads, linking and enrichment of data</td><td>Implementing application semantics with the stored entities, data transformation in use-case-oriented formats</td><td>Storing data in use-case oriented formats, providing query interfaces / APIs to the data</td><td>Using the prepared data in pre-defined application scenarios, e.g. in user-friendly GUIs</td></tr>
<tr><td colspan="3">LODicity
Linked and Open Data Integration Framework</td><td>Existing specialised databases or frameworks</td><td>Existing frameworks</td></tr>
</table>

Table 4.16: LODicity supports developers implementing the ETL, Storage, and Customisation steps of the Linked and Open Data integration process.

LODicity provides an application-specific entity ecosystem, having the storage of entities in its core. Therefore, a data warehouse approach is pursued. For the remaining steps in the Linked and Open Data integration process towards applications, delivery and use, it can be assumed that there already exists a wide range of industry-standard, developer-friendly software (Section 2.8 on page 82).

The following features are eminent for a developer-oriented, pragmatic integration frame-

work for Linked and Open Data. They are assigned to the individual sub-steps of the ETL step, as well as to the storage and customisation steps.

Features to Support Extraction

In the extraction phase of the ETL step, data sources are initially retrieved from external sources. The following features support the developer in this endeavour.

Retrieval from CKAN-based Repositories[19] A number of popular data portals are based on the CMS CKAN, including datahub.io, data.gov or data.gov.uk. It provides an open, standardised API to access the metadata and to retrieve Linked and Open Data (Section 4.1 on page 122) stored on the portal. When extracting data, LODicity can make use of this API, e.g. to download and persist data sources as well as their metadata automatically.

Data Source Metadata Management[19] When extracting entities from external data sources, further information about the extraction are of interest. This includes the last successful and unsuccessful extraction date, the creation and update timestamps of the source data, as well as licence definitions and author names. LODicity not only provides an ecosystem for data, but also for its metadata. As such, this information can be accessed in a unified and developer-friendly way, and thus allows other components to implement advanced features, such as respecting the Creative Commons Attribution license[20] or to prevent unnecessary loads if the data has not changed.

Pragmatic RDF Parser RDF is a flexible data format which can be used in different ways. By design, RDF might be very heterogeneous and require support for advanced features such as OWL reasoning to be completely parsed. As shown in the field study (Section 4.1 on page 122), in practice, it is not uncommon that RDF might be more homogenous (Section 4.1.5 on page 127) and might often

[19] This feature is prototypically implemented in the application described in section 4.3.8 on page 184 and is also subject of the proof-of-concept chapter 5 on page 197.

[20] The Creative Commons Attribution licence requires to "give appropriate credit" when using or changing licensed material (Creative Commons, 2016).

only use simple RDF features - possibly because the dumps were not created with Semantic Web tooling in the first place (Section 2.3.2 on page 58). To handle the latter kind of RDF, LODicity can provide a streaming RDF parser that does not require building up the entire graph model when loading data.

Staging Area / Cache Working with remote resources always introduces an uncontrollable dependency. Without warning, a resource might disappear, change its format and semantic or suddenly contain syntax errors. existing work addresses this with staging areas (Section 2.5.2 on page 64) or transparent caching (Section 2.6.6 on page 76). LODicity can support developers in handling these cases by keeping an unmodified copy of data sources which is used if an error occurs with the online resource.

Features to Support Transformation

In the transformation phase of the ETL step, data from the sources is changed to match the application-specific entities and their quality constraints. The following features support the developer in this endeavour.

Definition of Application-specific Entities[19] In specific semantic applications, data usually possess a fixed structure that is established, maintained, enriched and evaluated across the entire application and all components. In a developer-defined, tabular *Schema*, all relevant application-specific entities are defined, as well as their respective data fields and field value types. By providing easy means to do so, LODicity can support developers in this task, as well as it can provide an ecosystem for related tasks, such as APIs to use the defined entities, automatic validation of entities or automatic serialisation in common formats such as JSON (Section 4.3.3 on page 166).

Custom Transformation Logic[19] Some Linked and Open Data formats, such as the most common Linked and Open Data formats Microsoft Excel or CSV (Section 4.1.1 on page 123), are less normative formats. The semantics of rows and columns

are different for each individual file. Depending on data sources like this and the developed application, large portions of the Linked and Open Data integration logic are likely to be specific and dedicated to the given data source, application and context. Developers will usually address these challenges with program code that is just as specific and dedicated. LODicity can support this by providing an environment in which custom code can be hooked into standard routines as well as by providing suitable APIs, e.g. to entity or storage capabilities.

Generic Transformation Logic Other Linked and Open Data formats such as XML and especially RDF are more normative formats. Processing these data formats is likely to contain transferable capabilities for filtering, selecting sub-structures, transformation, conditional selecting or renaming. Existing work already addressed this with format-specific DSLs such as XPath for XML or RDF Data Shapes (Section 2.2.4 on page 54) and LDPath (Section 2.6.6 on page 76) for RDF. LODicity can support these DSLs in the definition and construction of application-specific entities. For example, besides defining valid data fields and values for each entity, the schema (Section 4.3.4 on page 169) can contain additional rows to define the data source and a LDPath expression to automatically extract and transform the defined values from the source data. This can be combined with custom transformation capabilities.

SPARQL Support for Public Endpoints SPARQL is a common query language in the Web of Data (Section 1.1.2 on page 32). It allows the selection of defined data from one or more SPARQL endpoints, the definition of constraints and conditions as well as transformations for specific field values. By natively supporting SPARQL, LODicity can integrate this mature and common language in its Linked and Open Data integration ecosystem and thus allow the creation of application-specific entities directly from external sources.

License Compatibility Management Linked and Open Data, in practice, is published under various kinds of licences (Section 4.1.3 on page 125). Each license might

introduce specific constraints of how to use and how to combine licensed data. LODicity can support developers in this delicate field by strictly tracking licenses of data sources in their corresponding metadata and maintaining a license compatibility matrix. This way, when mixing data with incompatible licenses, the developer can be directly informed.

Ontology-Merging Capabilities Specific data integration scenarios, for example the merge of complex ontologies, might require additional tooling besides providing DSL expressions. For example, the existing work in context of the LOD2Stack provides an number of useful integration tools (Section 2.6.4 on page 73). With tools like SILK, the domain specific knowledge of human experts can be incorporated into the process. LODicity can support this by providing adaptors for such third-party tools.

Features to Support Load

In the final load phase of the ETL step, previously created application-specific entities are persisted. The following features support the developer in this endeavour.

Abstracted Persistence Layer[19] During ETL, defined application-specific entities are created, persisted, and stepwise enriched. LODicity can support this by providing developer-friendly persistence and retrieval capabilities with different commit strategies.

Definition of Loaders, Data Sources and Dependencies[19] Data sources, specific loaders and related transformation or enrichment steps form a dependent integration chain. By letting developers define these dependencies in *Loader Chains*, LODicity can automatically maintain pre-conditions, licenses and the execution order of one or more dependent loaders. This also involves advanced capabilities such as error-handling, incremental or unnecessary loads.

Features to Support Storage

In the storage step, application-specific entities produced in the ETL process are persisted and maintained. The following features support the developer in this endeavour.

Entity Aware Storage[19] Entities in LODicity follow a well described, pre-defined schema definition. This allows developers to implement their custom program code based on attributes and field values of this schema, independently of any specific storage technology. LODicity can support this by providing a developer-friendly abstraction of the schema-defined application-specific entities when accessing, persisting or retrieving entities. This also includes transparent optimisation for certain field values, indexing and duplication checking.

Exchangeable Storage Technology[19] The best choice for a given storage technology might, in practice, depend on a number of different factors, as shown in the previous study (Section 4.2 on page 134). LODicity can support this by abstracting the exact storage technology and by providing the developer with a functional API for their storage endeavours. Therefore, code developed to work with LODicity is generally storage-agnostic.

Zero Configuration by Default[19] A broad range of different kinds of Linked and Open Data about arbitrary topics is available on the different portals. Especially in the beginning of the development of a new application exploiting this data, developers should focus on domain-specific challenges and start creating value for their customers rapidly. LODicity can support this by providing a solid architecture that, besides the application-specific entities, requires no specific configuration and has no runtime dependency to other services. This allows the architecture to remain easy in the beginning, but to be still ready for future challenges, thanks to the previous feature.

Features to Support Customisation

In the Customisation step, integrated entities are prepared to be delivered in specialised storage technologies, especially towards end user applications. The following features support the developer in this endeavour.

Native Query Capabilities[19] The entire Customisation step is highly specific to the exact selected delivery technology and the optimisation towards specific problems that corresponding end-user applications have. In practice, this will likely lead to highly specific custom code. LODicity can support this by providing elegant and seamless access to the integrated entities. This involves native query, filtering and iteration capabilities.

Generic Search Engine A full text search for content of an application is a common, often expected feature for an application. Furthermore, indexing data with an IR framework such as Lucene or Solr is a straightforward job. LODicity can support this by automatically creating the index structures for defined or all entities. Required configuration for the indexing is either already present in the defined schema, such as field name and value type, or could be added easily, such as language[21] or the exact indexing method.

LOD Export Well integrated and enriched Linked and Open Data is valuable. It is thus not only a good foundation to build rich and exciting applications with, it is also a chance to contribute to the idea of Linked and Open Data by releasing it. However, this requires additional and possibly unpaid efforts. LODicity can support the release of Linked and Open Data by providing a basic export of integrated entities in simple RDF structures. Thereby, this could again make use of information already defined in the schema or that can be simply added.

Figure 4.14 on the next page shows these features in a theoretical application.

[21] A definition of a language is usually required for advanced indexing features, such as stemming or lemmatisation, because these methods are language specific.

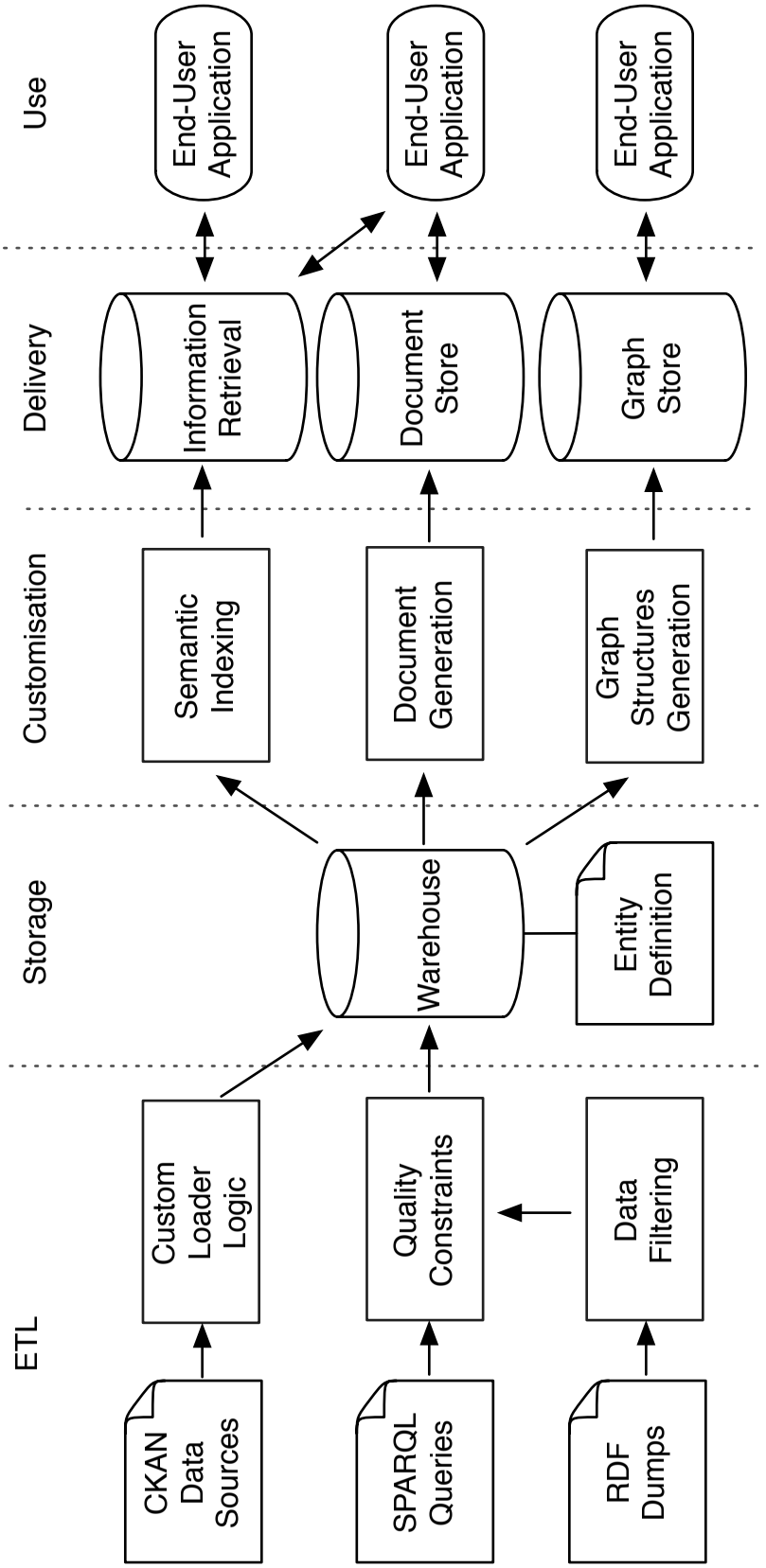


Figure 4.14: A full-featured, theoretical scenario for one or more LODicity-enabled applications.

4.3.5 Design Decisions

Based on existing research and previous work, a number of principle design decisions were made that influence the architecture of the prototype of LODicity fundamentally.

Selection of Programming Language

A primary goal of the present thesis is that developers are supported in their Linked and Open Data integration endeavour. Thereby, the choice of the programming language is a crucial decision.

Based on plain numbers, Java is one of the most favoured programming languages to date. In the commonly referred Tiobe index, Java is consistently either the most popular or the second most popular programming language since 2002 (Tiobe Index, 2015). At lower rankings, the index in addition contains newer programming languages such as Scala or Groovy that are based on the JVM. They all compile to Java Byte Code, can be mixed depending on the needs, and can be executed cross-platform.

In addition, Java is openly licensed and has as a rich ecosystem of third-party software (GitHut, 2015), including many high-quality contributions, e.g. from the (Apache Software Foundation, 2015).

In order to make a developer-friendly (Section 4.3.3 on page 166) choice, Java seems to be a good programming language to archive a high adoption of a Linked and Open Data Integration Framework.

Selection of Storage Technology

Because the storage of application-specific entities is the core of a data integration framework, a detailed evaluation for a current selection of storage technologies was conducted (Section 4.2 on page 134). It was streamlined towards a warehouse-like scenario, so the retrieval, aggregation and change queries can be used to assess suitable data integration solutions.

Besides plain performance, the technological footprint (Section 4.3.3 on page 166) is,

in practice, an important factor for the adaption of a storage technology. Start small, with an option to grow big.

Based on the evaluation results (Section 4.2.15 on page 157), a reasonable storage choice to start with is SQLite, with the option to scale up using PostgreSQL when needed. This way, thanks to SQLite, by default, no additional configuration of the executing system is required - the database is just created on the fly. The developer can literally instantly start working. PostgreSQL would then come into play if this single file database approach is no longer suitable, e.g. in production use, or when the data sizes grow significantly.

For the development of the Linked and Open Data Integration Framework, this demand to choose the actual database implementation suggests the use of JDBC as an abstraction layer. This way, it is ensured that the actual database implementation is not hard-wired into the framework and thus can, in theory, be exchanged later on. Selecting a JDBC-based approach, however, technically excludes sqlite4java, as it was explicitly designed to work without JDBC. SQLite-Xerial is thus the default storage choice.

A SQLite-based approach is an important contribution to keep simple scenarios simple. It requires zero maintenance, the database file is created when needed, and can be distributed and used easily for test cases or even simple production scenarios. Moreover, it has no runtime dependencies on running services.

Entity-Inherence

A primary characteristic of LODicity is the focus on application-specific entities. Providing, maintaining, storing and validating these entities is fundamental. Sometimes, as pointed out in the literature review, being able to access data in specific, application-specific can even be considered to be mission critical and good practice.

An essential finding of the previously conducted field study (Section 4.1 on page 122) is the fact that large amounts of data expose a tabular structure. This observation is independent from the individual serialisation format and is also true for formats like

RDF which do not specify any structure at all.

A clear definition of entities is important, even though it is no longer a requirement from the storage perspective of some modern NoSQL approaches such as MongoDB. When using these Schema-less storage solutions, they usually tend to introduce an implicit schema. Instead of being pre-defined and well designed, implicit schemata emerge in the way the storage layer is used within the application (Peterse, 2015). Whenever an application uses a certain data field for a certain functionality in the application semantic or GUI, another property is introduced to the implicit schema. But in contrast to explicit schemata, implicit schemata can not be used to validate, integrate, or consolidate entities, as it is just defined by behaviour. This also undermines optimisation efforts, and might, as shown in the benchmark, have impacts on the performance, especially in comparison to triple stores which might be seen as most consequent implementations of schema-less storages.

A clear definition of entities in a predefined schema plays a role for the integration, quality assurance and migration of Linked and Open Data, even if certain steps of the integration chain, such as the delivery, might be schema-less.

For the development of the Linked and Open Data Integration Framework, this implies that it should provide a means for developers to define these entities, but also for associated features like validation, serialisation into and deserialisation from common formats like JSON, and simple, schema-aware access methods.

4.3.6 Main Components

Figure 4.15 on the next page shows which components are developed to cover capabilities for the ETL, storage and customisation of the LODicity framework.

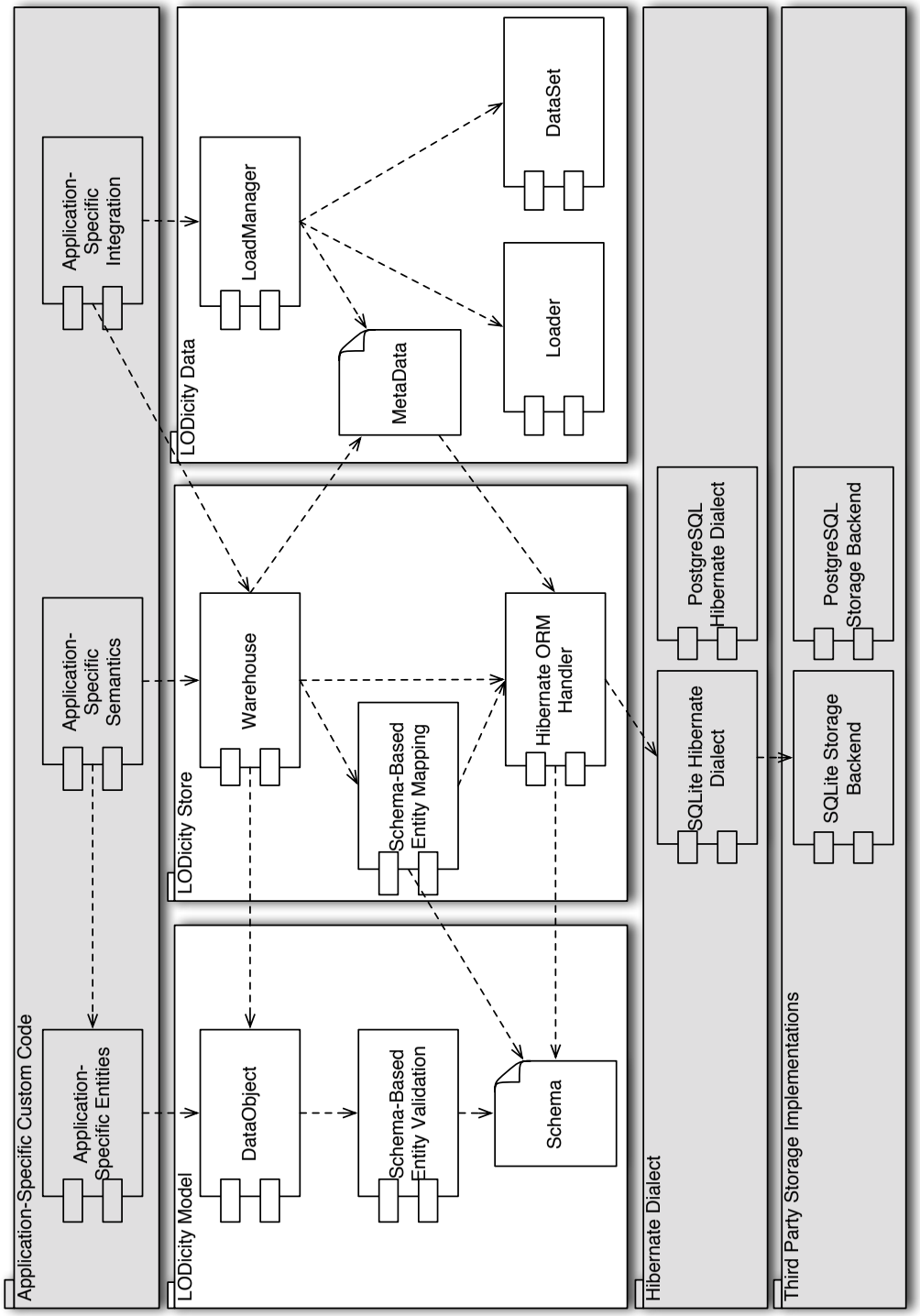


Figure 4.15: Components of the Linked and Open Data Integration Framework developed in the present thesis (white), together with associated third-party components and custom developer code (grey).

Application-Specific Custom Code

This entire package contains components that are developed by the users the Linked and Open Data Integration Framework. In this context, three major areas of activities can be identified (Section 4.3.3 on page 166):

Application-Specific Entities are the specific entities of a given semantic application. This includes definition, integration and verification code writing by the developer.

Application-Specific Semantics is the specific semantic logic of an application. According to the polyglot persistence paradigm, this might involve one or more storage technologies, depending on the usage scenarios.

Application-Specific Integration represents the exact actions required to extract, combine, unify, quality assure and load specific entities from one or more data sources.

A design goal of the LODicity is to support these actions so that developers find an ecosystem of tools to handle routine tasks, so they are able to start solving the application-specific challenges. To do so, the framework provides designated extension points for custom code: DataObject, Warehouse, and LoadManager.

DataObject

DataObject is the superclass for all application-specific entities defined by a user of the framework. Alternatively, it can be used directly if one single entity is sufficient. It provides a set of field-access, validation, and serialisation methods. In addition to that, it implements the Java Map Interface, which allows JVM-based languages convenient access capabilities.

Schema-Based Entity Validation

Schema-Based Entity Validation automatically verifies DataObjects or custom instances of it during runtime. This is performed when setting or retrieving field values and includes field and type checks.

Schema

The Schema defines valid entity names, associated fields, types and cardinalities. After instantiation, these fundamental properties are parsed, fixed and made available programmatically, so that other components like the schema-based entity validation or the schema-based entity mapping can use them during runtime.

Warehouse

The Warehouse is the central component of the LODicity. It provides developers with a means to integrate, query and use Linked and Open Data in an application-specific context. Routine tasks are abstracted, so that developers can focus on their specific application semantics.

Schema-Based Entity Mapping

This component translates the properties about the application-specific entities in the Schema into a Hibernate-compatible entity mapping. This allows Schema-defined entities to be processed with the regular Hibernate ORM tooling, e.g. to persist, query, and search them, in regard of their special field- and type-definitions in Schema.

Hibernate ORM Handler

The Hibernate ORM Handler is an internal component that is called before an entity is persisted, allowing to influence the exact details of how whole entities and certain fields are stored. As such, this component ensures that all DataObject implementations are recognised as such by Hibernate. This component also allows to define specific behaviour in the case of non-relational types such as lists. As pointed out earlier, different database implementations offer different capabilities to handle certain types such as lists.

LoadManager

LoadManager provides developers with an API to define and register loader chains, a defined sequence of individual loaders, and to trigger the integration process. In the background, the component calculates Dataset dependencies of loaders, collects

metadata and detects if a certain loader chain needs to be executed at all.

MetaData

Metadata contains information about the load process of data sources, such as involved, local files, and their last modified timestamp. It is maintained automatically by the LoadManager and stored in the storage backend alongside entities.

Loader

A loader is responsible for the loading, integration, quality assurance and enrichment of data sources. In a defined loader sequence, one or more loaders constitute a loader chain, which is executed by the LoadManager component. A loader produces instances of DataObject or a custom specialisation of it.

Dataset

A dataset represents a source data and is based on the CKAN definition (Section 3.2.1 on page 97). Thus, a dataset might refer to more than one file, as well as local or remote resources.

Hibernate Dialect

A Hibernate dialect is a basic configuration property in Hibernate (Konda, 2014, p. 22). It specifies implementation details of the underlying storage technology. Out of the box, Hibernate provides a number of dialects, such as a PostgreSQL dialect. SQLite support is provided by third-party dialects, such as SQLite-Dialect (GitHub, 2015).

Third party Storage Implementations

Third party storage implementation represent the concrete storage backend. Based on the findings of the previous benchmark section, SQLite as well as PostgreSQL are the most suitable technologies to handle warehouse-like scenarios.

4.3.7 Execution Process

LODicity is designed to support a developer's workflow and to handle routine tasks when building applications based on Linked and Open Data. This fosters the creation of applications based on Linked and Open Data, as developers can can start solving application-specific challenges faster.

Figure 4.16 on the facing page shows a full featured Linked and Open Data integration process and how custom developer code interacts with LODicity. It is shown that typical, specific application-based tasks remain in the realm of the developer, such as the definition of entities, the specification of one or more loaders and the core application semantics, based on integrated entities. The process is two-fold and starts with the integration of data sources into application-specific entities (green start event to intermediate yellow event) and ends with the customisation of entities towards their delivery (intermediate event to red close event).

The first phase aims at the definition and integration of application-specific entities. In the beginning, a developer defines the exact entities for this specific application scenario using the Schema component. Details such as the type, format and cardinality of fields and their values depend on the information needs end users or other components have. LODicity supports the developer in this endeavour by providing the baseline entity functionality and specifying the backend storage structures, based on the defined Schema. In the next steps, the developer specifies the execution order of the loaders in one or more loader chains (Section 4.3.4 on page 171) and the required datasets. Thereby, existing framework functionality can be used, e.g. when defining a CKAN-based dataset.

After the developer triggered the load, LODicity first tries to retrieve metadata about the referred datasets from the internal storage. Retrieved metadata is then used to check if loading the source is actually necessary or if the dataset is already loaded and up to date. If all datasets of a defined loader chain are up to date, the LODicity skips the execution of all involved loaders.

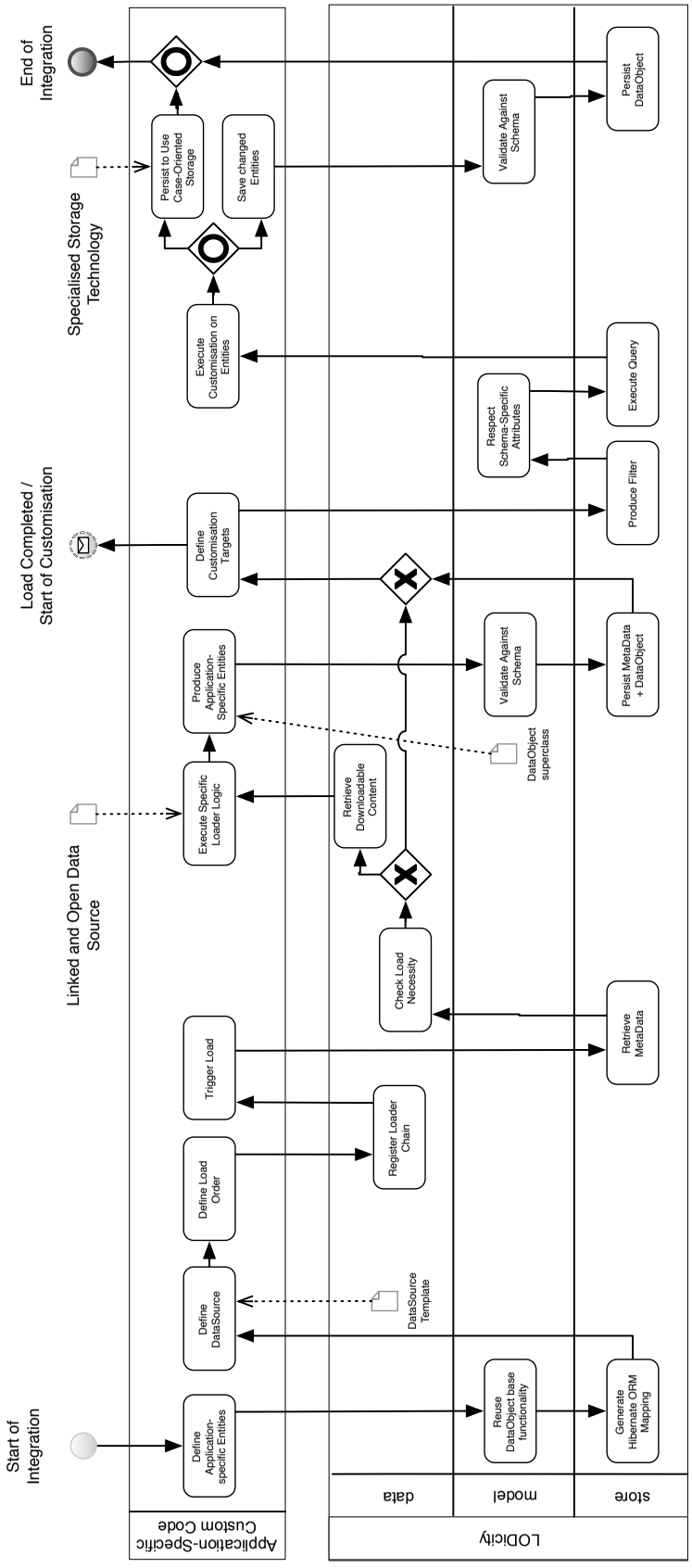


Figure 4.16: The interaction of custom developer code (upper swimlane) with LODicity (three grouped lanes) in a full-featured Linked and Open Data integration process. The intermedia event marks the end of the integration process and the transition towards the customisation process.

If at least one dataset per loader chain indicates that a load is necessary, the referenced online resources are downloaded (depending on its type), locally stored, and the developer's custom loaders are triggered in the predefined order.

During the process, the loaders produce, enrich, update, or delete application-specific entities, defined by the Schema. LODicity, in the background, handles validation and persistence and the production of metadata. The developer is provided comfortable methods so they can focus on their application-specific tasks.

After all registered loader chains completed, the integration phase is finished (marked as yellow intermediate event in Figure 4.16 on page 183).

The next phase, the customisation of entities, aims at producing streamlined delivery stores, optimised for actual use cases of the application.

The customisation begins with selecting the subset of the integrated entities that are the subject of the customisation. Thereby, the warehouse component provides the developer comfortable method, so that they can implement their custom application-specific, problem-centred customisation with the exact entities that are required.

In the background, based on the schema, LODicity automatically translates queries into the structure of the storage backend and executes them on the internal Hibernate abstraction layer.

Next, according to the polyglot persistence paradigm, a developer might want to choose a dedicated third-party storage technology, such as Lucene or MongoDB, to deliver the semantics to end user applications. They can also hand back customised entities to the warehouse, as long as entities still comply to the Schema.

4.3.8 Prototype Implementation

In this section, the prototypical implementation of the main components of LODicity in Java is described. The source code of core classes is attached in appendix C on page 431.

DataSource

With a `DataSource`, the developer assigns external data sources to a defined loader. In the most basic case, `DataSource` just defines one or more local files, an identifier, and a so called `CurrentnessToken` which represents the `DataSource`'s up-to-dateness. As shown in Figure 4.17, depending on the exact kind, LODicity provides more specialised `DataSources` which handle case-specific routine tasks, such as a `PackagedResource` to load local files using Java's class loader.

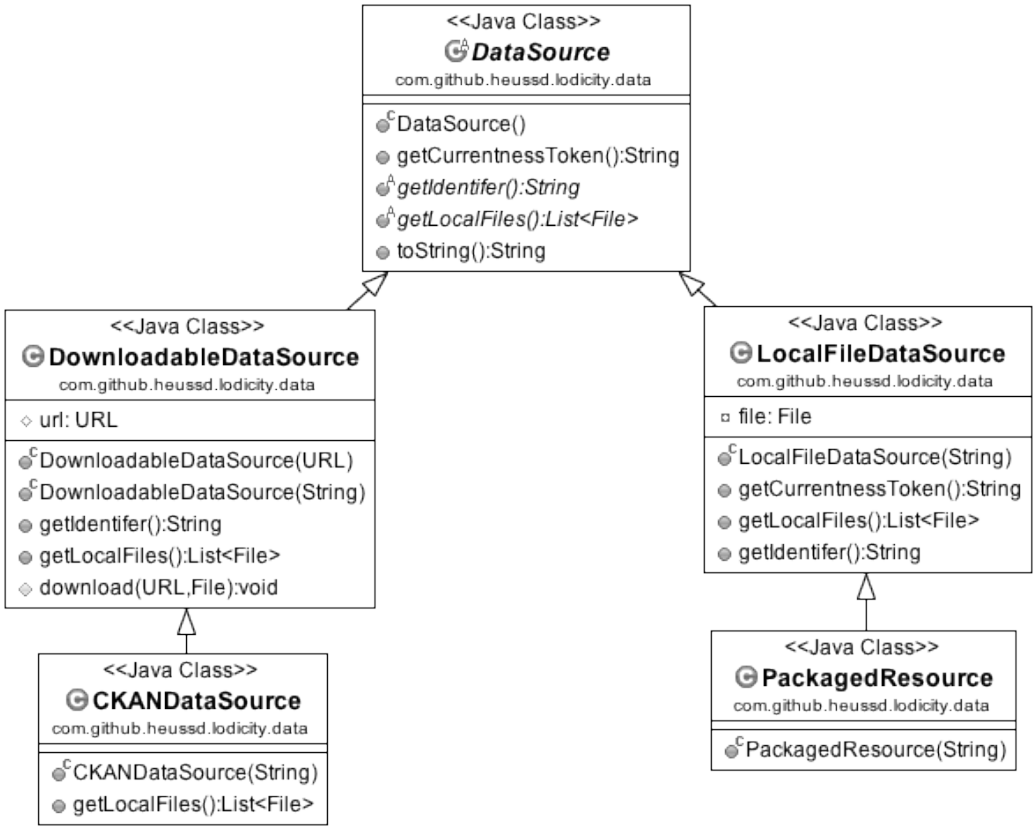


Figure 4.17: UML Class Diagram of `DataSource` and it's specialisations.

To foster the development of applications based on Linked and Open Data from data portals such as datahub.io or data.gov.uk, LODicity provides a specific `CKANDataSource`. Using it, developers can easily refer to datasets on CKAN-based data portals, with a simple constructor call, such as:

```
DataSource ds = new CKANDataSource("https://datahub.io/dataset/hebis-bibliographic-resources");
```

Listing 4.2: Definition of a CKAN-based data source in LODicity.

This will query the CKAN-JSON-API, download referred datasets and store them locally.

Loader

`Loader` (Figure 4.18) is the superclass that custom code has to specialise in order to be executed by the `LoadManager`. `Loader` implement specific integration semantics for entities of one or more `DataSources`.

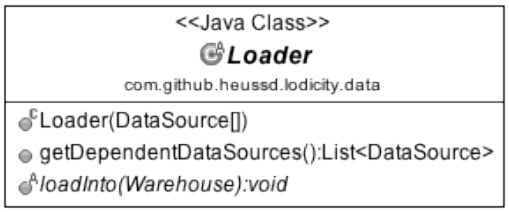


Figure 4.18: UML Class Diagram of Loader.

The program logic of a loader is called with the method `loadInto()`, which provides a working instance of the `Warehouse` as target for the integrated entities. Using the method `getDependentDataSources()`, a loader specifies `DataSources` that are processed by it.

LoadManager

Purpose of `LoadManager` is to coordinate all loading activities and to maintain metadata about `DataSources`. It is used to define one or more loader chains, each consisting of one or more `Loaders`, which then integrate application-specific entities into a `Warehouse` instance.

The typical call-order is shown in Listing 4.3.

```
loadManager.register(new TestFileLoader());
loadManager.register(new DBLoader(),new DBDependencyResolution());
loadManager.loadAll();
```

Listing 4.3: Registration of Loaders in LoadManager.

In the code sample, two loader chains are registered. The first one consists of a single `Loader` (`TestFileLoader`), the second one of two consecutive `Loaders`, `DBLoader` and `DBDependencyResolution`.

`loadAll()` triggers the load of all registered loader chains.

Schema

In this component, the developer defines the specific entities for their application. For this task, he uses a spreadsheet to configure entity types, field names, data types and cardinality, as shown in Figure 4.19.

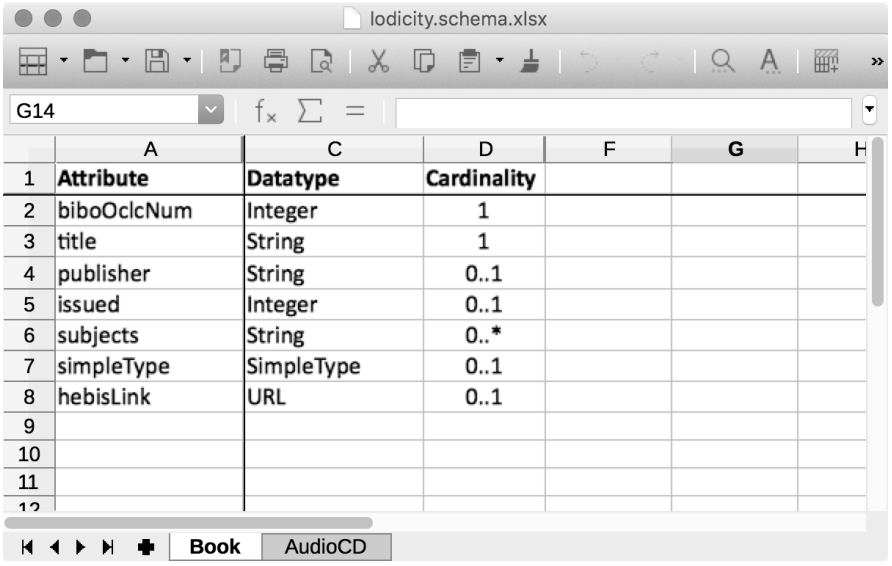


Figure 4.19: Schema is configured using a spreadsheet.

The `Schema` class represents this configuration in Java code. As shown in Figure 4.20 on the following page, it provides developers means to access specific properties of the entity definition, such as convenience methods to check if a certain attribute of a given type is mandatory (`isMandatory()`) or a list type (`isListType()`). In addition, it provides entity validation mechanisms using `validate()`.

Moreover, based on the spreadsheet configuration, `Schema` produces a so called Hibernate Mapping to define database structures in Hibernate (Konda, 2014, p. 8). It makes use from the fact that besides using Java class annotations, mapping instructions in Hibernate can be defined in an XML document, which is then passed to the Hibernate abstraction layer during initialisation.

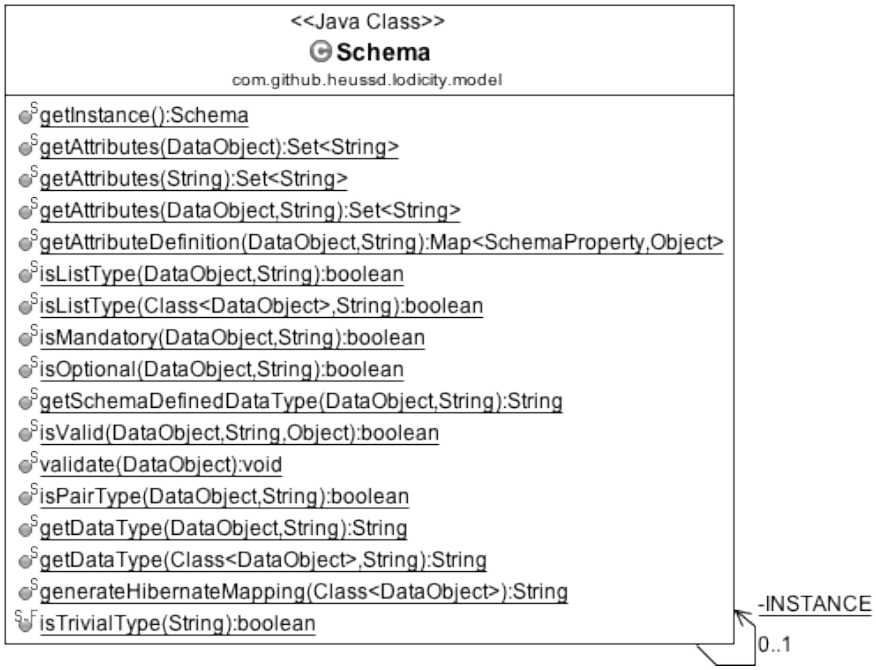


Figure 4.20: UML Class Diagram of Schema.

DataObject

`DataObject` is the base class for all application-specific entities. It can be used directly by developers or it can be extended by custom code. As shown in Figure 4.21 on the next page, `DataObject` contains getter, setter, serialisation and validation methods.

When `DataObject` is extended in custom code to create more application-specific entities, a minimum code is required, as shown in Listing 4.5 on the facing page.

```
class AudioCD extends DataObject {

 public AudioCD(Map<String, Object> dataObject) {
  super(dataObject);
 }
}
```

Listing 4.4: Minimal Sample for a custom entity.

Warehouse

`Warehouse` is the main class of LODicity. As shown in Figure 4.22 on page 190, it provides developers comfortable means to persist, change, query and iterate over application-specific entities.

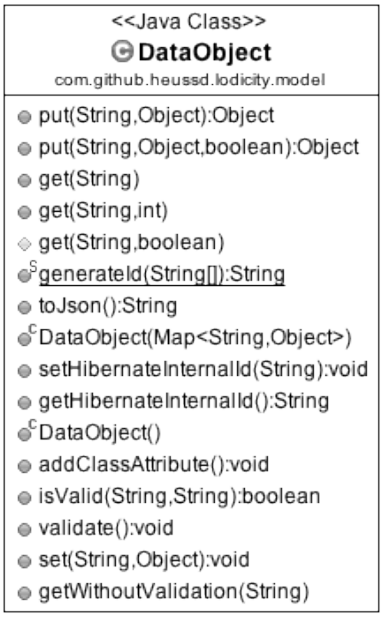


Figure 4.21: UML Class Diagram of DataObject.

In the initialisation, `Warehouse` is given all application-specific entities that should be supported in the newly create instance. In the background, based on these entities and their definitions in the `Schema`, the Hibernate abstraction layer is set up.

`Warehouse` supports Java 8 features, such as lambda expressions. So, for example, because the `Iterable` Interface is used, actions such as updating a certain field in every available `DataObject`, only requires very few LoC:

```
warehouse.forEach(DataObject.class, dataObject -> {
 dataObject.set("title", "Changed value");
 warehouse.update(d);
});
```

Listing 4.5: Java 8 Lambda expression to update and persist a single field of all `DataObjects`.

Filter

`Filter` offers a developer simple means to specify their information need in the `Warehouse`, based on `Schema`-defined terminology. As shown in Figure 4.23 on the following page, it supports basic, SQL-like filter constraints such as equal (`eq`), contains (`like` respectively `ilike` for case insensitive comparison) and or.

Internally, each method generates a new Hibernate criterion with new constraints.

Figure 4.22: UML Class Diagram of Warehouse.

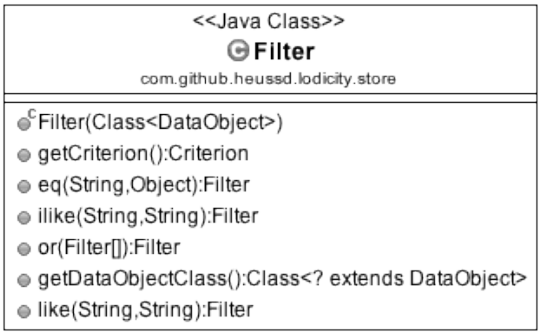


Figure 4.23: UML Class Diagram of Filter.

`Schema` properties are respected, so that, for example, lists can just be filtered as if they are atomic values. `getCriterion()` is used to retrieve all aggregated criterions.

Multiple filters can be supplied arbitrarily and are by default and-chained. In combination with the previously introduced `Warehouse` capabilities, this allows powerful but low-effort working with entities:

```
Filter f = new Filter(DataObject.class);

warehouse.query(f.or(f.ilike("title", "study"), f.ilike("title", "studie"))).forEach(d -> {
 System.out.println("Found Study / Studie: " + d);
});
```

Listing 4.6: Application of multiple filters in a Warehouse query.

DataObjectInterceptor

Prior to persisting application-specific entities in the storage layer, non-trivial field types such as lists or other `DataObject`s must be handled. For this purpose, Hibernate allows to define an `Intercepter` that allows to manipulate specific lower-level database calls.

Listing 4.7 shows the core of LODicity's Hibernate interceptor, which basically embeds all non-trivial and non-scalar field types as JSON object.

```
public class DataObjectInterceptor extends EmptyInterceptor {
 @Override
 public String getEntityName(Object object) {
  if (object != null && object instanceof DataObject) {
   return object.getClass().getName();
  } else {
   return super.getEntityName(object);
  }
 }

 @Override
 public boolean onSave(Object entity, Serializable id, Object[] state, String[] propertyNames,
      Type[] types) {
  state = embedListsAsJson(entity, state, propertyNames, types);
  state = embedDataObjectsAsJson(entity, state, propertyNames, types);
  return super.onSave(entity, id, state, propertyNames, types);
 }
 @Override
 public boolean onFlushDirty(Object entity, Serializable id, Object[] currentState, Object[]
      previousState, String[] propertyNames, Type[] types) {
  currentState = embedListsAsJson(entity, currentState, propertyNames, types);
  currentState = embedDataObjectsAsJson(entity, currentState, propertyNames, types);
  return super.onFlushDirty(entity, id, currentState, previousState, propertyNames, types);
 }

 private Object[] embedListsAsJson(Object entity, Object[] states, String[] propertyNames, Type[]
       types) {
  if (entity instanceof DataObject) {
   for (int i = 0; i < propertyNames.length; i++) {
    String propertyName = propertyNames[i];

    if (Schema.isListType((DataObject) entity, propertyName)) {
     Object value = states[i];
     value = (String) new JSONArray((List<String>) value).toString();
     states[i] = value;
    }
   }
  }
  return states;
 }

 private Object[] embedDataObjectsAsJson(Object entity, Object[] states, String[] propertyNames,
      Type[] types) {
  if (entity instanceof DataObject) {
   for (int i = 0; i < propertyNames.length; i++) {
    Object value = states[i];

    if (value != null && value instanceof DataObject) {
     value = (String) ((DataObject) value).toJson();
     states[i] = value;
    }
   }
  }
  return states;
 }
}
```

Listing 4.7: Core Logic of LODicity's Hibernate Interceptor.

Complete Code Sample

Based on the previously introduced components, Listing 4.8 shows a compilable and executable sample of the interplay of all LODicity components.

```
import org.junit.Test;

import com.github.heussd.lodicity.data.LoadManager;
import com.github.heussd.lodicity.data.Loader;
import com.github.heussd.lodicity.model.DataObject;
import com.github.heussd.lodicity.store.Filter;
import com.github.heussd.lodicity.store.Warehouse;

public class CompleteTestCase {

 @Test
 public void testCompleteRun() {
  final String field = "string";
  Warehouse warehouse = new Warehouse(true, DataObject.class);

  LoadManager loadManager = new LoadManager(warehouse);

  loadManager.register(new Loader() {
   @Override
   public void loadInto(Warehouse warehouse) {

    for (int i = 0; i <= 10; i++) {
     DataObject dataObject = new DataObject();

     // Randomly insert a selection criteria
     if (Math.round(Math.random() * 10) % 2 == 0) {
      dataObject.set(field, "please find me");
     } else {
      dataObject.set(field, "meh, dont find me");
     }
     warehouse.persist(dataObject);
    }
   }
  });

  loadManager.loadAll();

  Filter filter = new Filter(DataObject.class);
  warehouse.query(filter.eq(field, "please find me")).forEach(dataObject -> {
   System.out.println("Oh look, i found it: "+ dataObject);
  });
 }

}
```

Listing 4.8: A complete, minimal code sample for the load, storage and use of custom entities in LODicity.

In the code, a trivial loader chain is defined with a single, anonymous `Loader`. This `Loader` generated 11 simple `DataObjects`, about half of them contain a certain selection criteria. After the `loadAll` is triggered in line 36, the generated `DataObjects` are persisted in the `Warehouse`.

As from line 38, the selection criteria is used in a `Filter` to query the `Warehouse` for matching `DataObjects`. These are then printed to the console.

Hibernate Configuration File

Besides the Hibernate mapping file that is generated on the fly by `Schema`, another basic Hibernate configuration file specifies the database configuration (such as driver and connection string) and simple, annotated entity classes like `MetaData`.

By default, the configuration file shipped with LODicity defines a SQLite storage backend, as it provides a adequate performance for most scenarios (Section 4.2.15 on page 157).

The configuration file also allows fine tuning specific database behaviour. SQLite, for example, requires a special session management, it is good practice to open only one single session at a time (Stack Overflow, 2015), which in addition must be held during the entire application life time. This behaviour can be configured using Hibernate capabilities, as shown in Listing D.2 on page 449.

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-configuration SYSTEM
"http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>
 <session-factory>
  <property name="hibernate.connection.driver_class">org.sqlite.JDBC</property>
  <property name="hibernate.connection.url">jdbc:sqlite:warehouse.db</property>
  <property name="hibernate.dialect">org.hibernate.dialect.SQLiteDialect</property>

  <property name="hibernate.connection.pool_size">1</property>
  <property name="hibernate.c3p0.min_size">1</property>
  <property name="hibernate.c3p0.max_size">1</property>
  <property name="hibernate.connection.release_mode">on_close</property>

  <mapping class="com.github.heussd.lodicity.data.MetaData" />
 </session-factory>
</hibernate-configuration>
```

Listing 4.9: XML-based configuration file of Hibernate.

The XML statements also specify a certain Hibernate dialect, which translates general Hibernate calls into database-specific queries. It is noteworthy that `org.hibernate.dialect.SQLiteDialect` is not built, but is third-party open source software (GitHub, 2015).

Thanks to Hibernate, the storage backend can simply be switched by specifying another database driver this file.

SQLite Hibernate Dialect

A Hibernate Dialect translates general-purpose Hibernate calls into specific database implementations. As mentioned, the previously configured SQLite Hibernate dialect is third-party, open source community software (GitHub, 2015).

During development, however, it turned out that selected SQLite dialect had a bug in certain scenarios in which existing structures should be dropped.

In the realm of the development of LODicity, this bug in the third-party software was fixed, published and successfully merged into the community project (see Appendix C.5 on page 445 for details).

Third Party Dependencies

One goal in the design of LODicity is to require as few framework and third party libraries as possible, as too complex dependencies might hinder possible application scenarios.

Thus, LODicity only requires the following third-party dependencies:

Apache POI (Apache License 2.0), to parse the `Schema` configuration.

Hibernate ORM (GNU Lesser General Public License), to implement the warehouse storage capability.

SQLite-Xerial (Apache License 2.0) JDBC-capable SQLite implementation.

SLF4J Logging (MIT) for basic logging.

JSON library (JSON License) for parsing and generation of JSON.

Wget Download Library (GNU Lesser General Public License) to download online-resources.

Apache Commons IO (Apache License 2.0) provide local file operation utilities.

Test Coverage

Tests cases exist for the storage, data and the model core functionality of LODicity, shown in Table 4.17. As it is distributed as open source GitHub project, new commits are automatically tested using the Cloud-based Continuous Integration (CI) suite Travis-CI[22].

Package	Tests	Errors	Failures	Skipped	Success	Time
	1	0	0	0	100%	0.304
store	21	0	0	0	100%	6.315
data	5	0	0	0	100%	3.843
model	14	0	0	0	100%	0
Total	41	0	0	0	100%	10.462

Table 4.17: Overview of Unit Tests in LODicity.

Existing tests cover 83% of LODicity's Java logic, as shown in Table 4.18, measured using EclEmma Eclipse plugin.

Package	Class	Instructions		
		Missed	Covered	Coverage
data	CKANDataSource	21	131	86.18%
data	DataSource	5	7	58.33%
data	DownloadableDataSource	4	62	93.94%
data	Loader	0	10	100.00%
data	LoadManager	0	186	100.00%
data	LocalFileDataSource	25	53	67.95%
data	MetaData	0	9	100.00%
data	PackagedResource	0	12	100.00%
model	DataObject	42	199	82.57%
model	DataObjectIterable	15	42	73.68%
model	DataObjectIterable.new	26	41	61.19%
model	Schema	351	981	73.65%
model	SchemaProperty	5	94	94.95%
store	DataObjectInterceptor	0	128	100.00%
store	Filter	0	130	100.00%
store	Warehouse	27	396	93.62%
Total		521	2481	82.64%

Table 4.18: Test coverage of unit tests in LODicity, measured using the EclEmma Eclipse plugin.

[22] `https://travis-ci.org/heussd/lodicity`, last access on 2015-11-07.

4.3.9 Summary of the Findings

Data integration endeavours often involve similar, transferable steps. To address the needs of developers in the integration of Linked and Open Data in specific application scenarios, a pragmatic framework LODicity is developed. It covers the capabilities for ETL, Storage and Customisation, and provides developers means to focus on the specific Delivery and Use using existing technology.

LODicity implements a common data warehouse pattern for Linked and Open Data. Thereby, by default, SQLite is used as storage backend, which displayed reasonable performance for most scenarios, and which requires no additional configuration and runtime dependencies to other services. Just by changing the configuration, the storage backend can be changed.

The framework also provides current Java 8 language features, such as lambda expressions, which allows developers to write operations on integrated entities with few, readable lines of code. It allows the definition and validation of custom entities in a spreadsheet and has a small technological footprint. This allows a large spectrum of different appliances.

Once the data is integrated, the developer can almost instantly start solving application-specific challenges and thus can produce actual value of Linked and Open Data in a semantic application, while the Linked and Open Data Integration Framework handles routine and error-prone tasks.

Chapter 5

Proof of Concept Framework Application

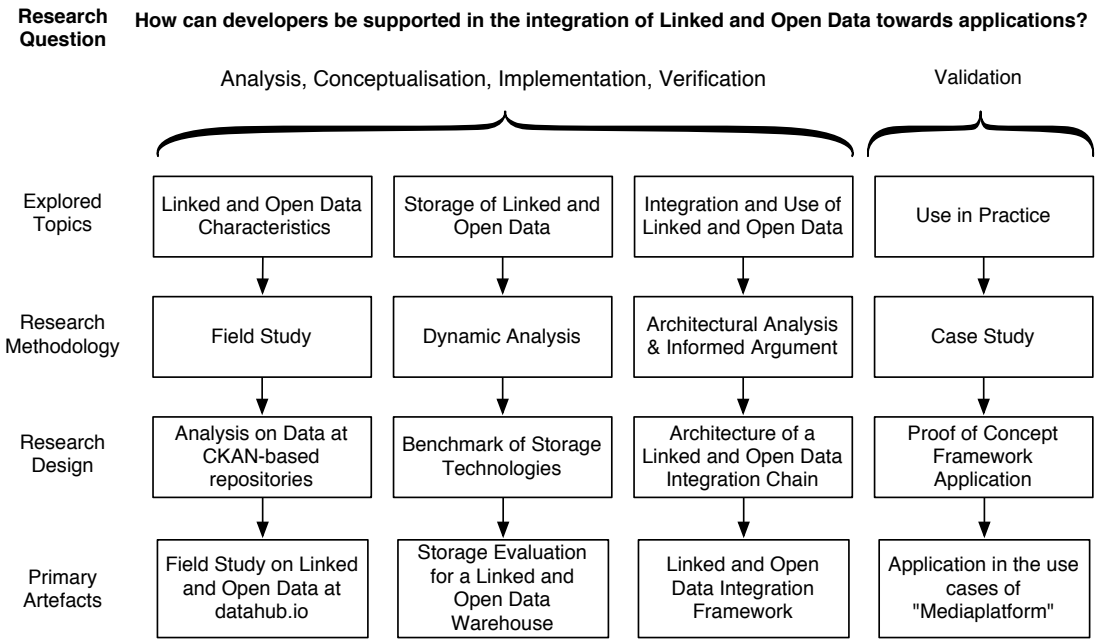


Figure 5.1: Overview of the Research Question, explored topics, selected methods, research design and the developed, novel artefacts of the present thesis.

In the previous chapters, the development of a Linked and Open Data integration framework, LODicity, is described. It combines insights gained in previous studies (Sections 4.1 and 4.2 on page 122 and on page 134) as well as best practices of existing research (Section 2.9 on page 85).

To validate the LODicity, a proof of concept application is documented. Thereby, the framework is integrated into the existing code base of the Mediaplatform (Section 1.1.6 on page 36).

5.1 Challenges

The Mediaplatform is a semantic application based on Linked and Open Data, as well as on proprietary data of the Städel Museum. It provides two semantic search Web clients, one for museum and one for library use cases, which allows end users to explore the respective inventories in new and innovative ways. Figure 5.2 shows a conceptual overview on its data integration process Semantic ETL, based on the previously identified phases (Section 3.2.3 on page 114).

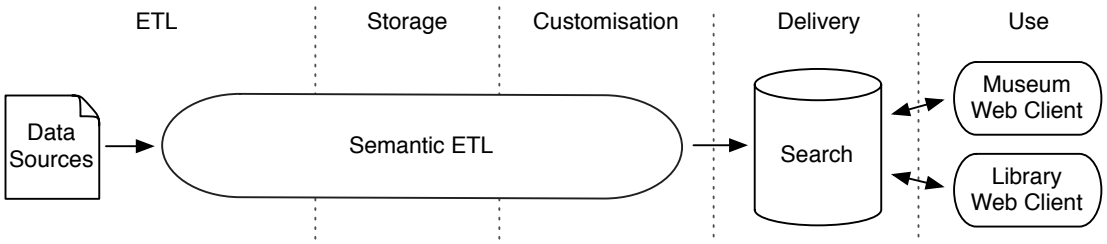


Figure 5.2: Conceptual Data Integration Process of the original Mediaplatform project. A common Semantic ETL process parses data sources, transforms data into entities, and then customises these entities for the search use case a single execution.

The entire integration chain is designed to deliver streamlined data to the clients, matching to the representative use cases. Because both involve information retrieval problems, the goal of the Semantic ETL process is a specialised Lucene search index. It consists of three different kinds of application-specific entities:

Resource a bibliographic resource or a museum exhibit.

Term a thesaurus item, referring to broader, narrower or synonym concepts.

Person an editor, author or artist.

These entities are generated based on several data sources, which are loaded, integrated, combined and enriched. Table 5.1 on the facing page shows which data sources constitute which entities.

In the context of this proof of concept, the existing Semantic ETL process is modified to

Data Source	Format	Produced Entities
OpenThesaurus	Plain Text	Terms
HeBIS catalogue	MARC	Resources
GND	RDF	Terms
Städel exhibits, thesaurus and persons	XML	Resources, Terms, Persons

Table 5.1: Entities of the Mediaplatform project.

include the data warehouse capabilities by the previously developed framework LODicity (Section 4.3.4 on page 167). Therefore, the primary functionality should be preserved, and improvements through the capabilities of the framework should be demonstrated.

5.2 Analysed Code State

Subject of this proof of concept is the non-public source code of the Mediaplatform, Subversion (SVN) revision 2273. It contains 87 JUnit tests, that pass successfully in less than 5 seconds. Tests cover core functionality of the platform and data integration details, such as if entities are properly integrated and can be found using certain search patterns. In addition, there are two working clients.

5.3 Application of the Linked and Open Data Integration Framework

In the following, it is documented how LODicity is successfully integrated into the project.

5.3.1 Separation of Data Integration and Customisation

The separation of the data integration steps and the data customisation steps is an important foundation for a semantic application. It allows the dedicated use of database technologies (Section 2.8.3 on page 83) and advanced features such as incremental loading of data.

Figure 5.3 on the next page gives an overview of how the previously introduced integration process (Figure 5.2 on page 198) has conceptually changed through use of LODicity.

Just as before, data sources are used to integrate data of different kinds and formats into

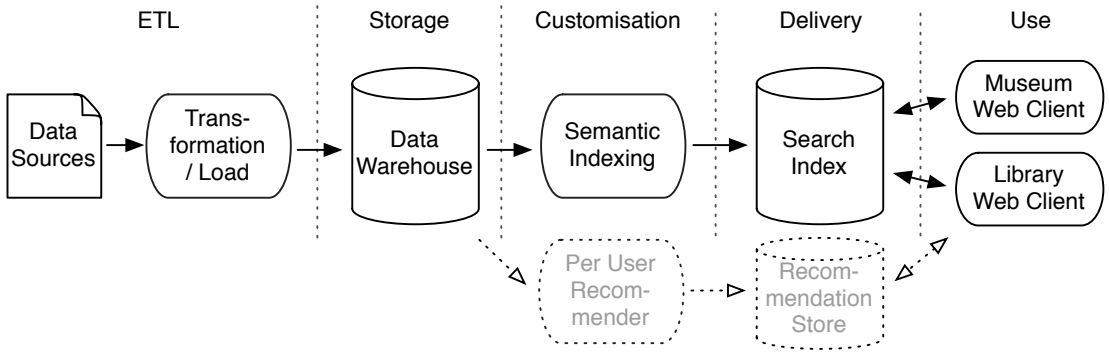


Figure 5.3: Data Integration Process of the Mediaplatform after integration of LODicity. ETL and Customisation are isolated processes, so is the persistence in Storage and Delivery. This is the foundation development of new use cases besides the existing search use case, such as recommendations.

application-specific entities. The core contribution of LODicity is a new data warehouse component to store entities after they have been processed by the ETL process, but before they are customised to meet certain end-user application use cases. Existing data source-specific loaders are retained, but updated to implement LODicity's loader concept[1].

In the Customisation phase, the developer uses the `Warehouse`-instance arbitrarily to produce the application-specific semantics. In this case study, this involves the creation and the stepwise enrichment of a Lucene search index, using adapted, existing classes:

```
130  LuceneWriteEngine<Resource> resourceStoreWriteEngine = LuceneWriteEngine.
         createResourceStoreWriteEngine();
131
132  warehouse.all(Resource.class).forEach(resource -> resourceStoreWriteEngine.add((Resource)
         resource));
133  resourceStoreWriteEngine.commit();
134
135  LibraryCacheBuilder libCacheBuilder = new LibraryCacheBuilder(resourceStoreWriteEngine,
         warehouse);
136
137  ResourceStoreTermStoreEnrichment resourceStoreTermStoreEnrichment = new
         ResourceStoreTermStoreEnrichment(warehouse, resourceStoreWriteEngine,
138    libCacheBuilder);
139  resourceStoreTermStoreEnrichment.enrich();
```

Listing 5.1: LODicity integrated in the customisation-logic of Mediaplatform.

The `Warehouse`-instance provides developers with an easy API to work with integrated entities. For example, on line 132, a Java 8 lambda expression is used to export all

[1]Full source codes for the original and new ETL processes are show in appendices D.1 and D.2 on page 447 and on page 449.

previously loaded Resource entities in a Lucene index. Other components make use of the query and filter capabilities of LODicity (Listing 4.6 on page 190).

Using the components mentioned, LODicity is successfully integrated into the Mediaplatform. This has a number of benefits for the application:

- In addition to the existing Lucene index, addressing the search use case of end-user applications, the integration of LODicity now allows the architecture to incorporate further stores to address other use cases of end-user applications, such as the production and storage of recommendations. Moreover, because the creation processes are no longer coupled, stores can be created, updated and enriched independently and scheduling can be aligned to the respective use case. This allows, for example, updating of the main search index on a daily basis, while the recommendation store is updated with new per-user-recommendations every 5 minutes.

- Thanks to LODicity, development of new semantic features in the customisation step can happen on already integrated entities. This allows faster cycles for implementation and testing, a simplified automatic unit test setup and continuous deployment. For example, detailed changes in the boosting of certain search index fields can be instantly demonstrated and discussed with the customer, without having to wait for the ETL steps to complete.

- The intermediate storage layer for integrated entities is crucial when targeting a differential load scenario, which is especially important when the ETL step takes a lot of time to complete. In these scenarios, it is desirable to have a use-case independent, consistent storage, which is transactionally added with new input from the data sources.

5.3.2 Execution Time

A clear separation of the several data integration steps can be ultimately demonstrated by the ability to re-execute, skip or repeat certain integration steps. Moreover, sepa-

rating the ETL logic from the customisation logic allows a finer decision on what to do on repeated executions, such as differential load scenarios.

To demonstrate the isolation of steps of the new architecture for this proof of concept, the existing loader code is modified to implement a trivial differential load scenario with LODicity.

In LODicity, when registered in a loader chain, loaders specify which data sources they use. Based on this, LoadManager determines during execution time if a loader chain must be executed or not, by resolving all associated data source. In this case study, LODicity's basic determination algorithm is used based on the data source's file change date. So if a data source has not changed since the last execution, the execution of any loader associated to this data source is skipped in the ETL phase.

In the code sample below, four loader chains are registered at the LoadManager (Section 4.3.6 on page 180):

```
122  LoadManager loadManager = new LoadManager(warehouse);
123  loadManager.register(new OpenThesaurusLoader(Places.DATA_SOURCES_OPENTHESAURUS.file("
         openthesaurus.txt")));
124  loadManager.register(new HebisLoader());
125  loadManager.register(new GndLoader(), new TermRanker());
126  loadManager.register(new StaedelLoader());
127  loadManager.loadAll();
```

Listing 5.2: LODicity integrated in the load-logic of Mediaplatform.

Each loader specifies on which data source it depends. Based on that, LoadManager decides whether to execute a certain loader chain or to skip it, e.g. because none of the associated data sources have changed. After the `loadAll()` call is issued by the developer, the previously defined `Warehouse`-instance contains integrated, application-specific entities.

Using this basic method, Figure 5.4 on the next page shows a comparison of repeated executions of the original Mediaplatform ETL process and this case study's LODicity-based ETL process.

Because it is a single, atomic process, the original Mediaplatform ETL process always takes about 46 seconds to complete with the selected test data. It is designed to clear

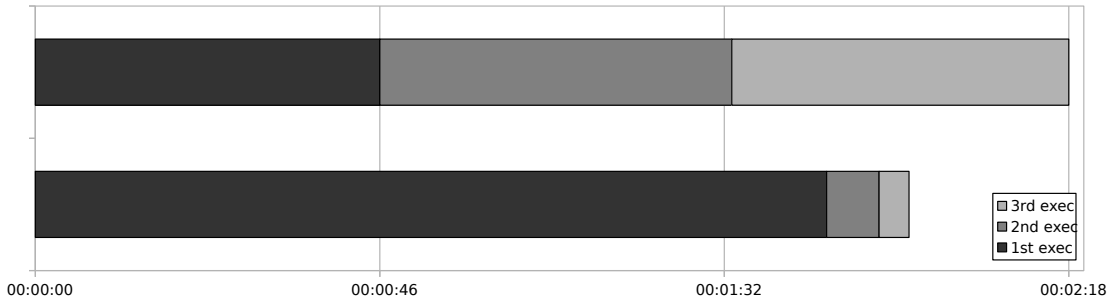


Figure 5.4: Comparison of subsequent execution times of the original Mediaplatform ETL process (above) and the LODicity-enabled ETL process (below).

and rebuild all required structures at the beginning of every launch, and must complete successfully in order to re-create them.

In contrast, because the new data warehouse component is created and filled initially, the first execution of a LODicity-enabled process adds about 1 minute execution time compared to the original process. After this, however, LODicity allows the integrated entities across different runs to be kept, so subsequent executions only take about 9 seconds by skipping unnecessary loader runs.

By implementing a simple loader modification that skips already integrated files, this demonstrates successfully that the introduced components help isolating conceptual steps in the architecture.

5.3.3 Lines of Code

LoC can be seen as an indicator for development effort. As LODicity is integrated in an existing code base which is expected to keep full functional, it is of interest what changes through introduction of the framework.

Reduced Development Effort

TermRanker is a class that enriches subjects from the GND with certain counts of library resources. These counts are then used in the topic wheel component (Figure 1.2 on page 37) in the recommendation of relevant subjects.

As such, TermRanker operates on integrated Resource and Term objects. In the original

Mediaplatform code, this is implemented by reading and updating the Lucene store (ResourceStore and TermStore). Thanks to LODicity, the interfaces of TermRanker can be replaced to work solely with entities from the data warehouse.

While keeping the intended functionality, this reduces the LoC of TermRanker by 32%, leading to smaller, more precise code and less involvement of helper methods. This is demonstrated by the Listings 5.3 and 5.4 on the next page, which examines the changes to the core program logic of TermRanker in detail.

A major advantage is that with LODicity, code can be more strictly based on the data model defined in the schema, and independent of the storage technology. As compared in line 55 of Listing 5.3 on the facing page, instead of developing and involving a helper method to create a specific Lucene-query (`subjectOrLocationQuery`), the information need in Listing 5.4 on the next page can directly be formulated for the warehouse, using application-specific vocabulary defined in the schema.

Moreover, another component can be removed entirely: TermStore is a class to provide access to Term objects in the search index. In the Mediaplatform, this access is only required during the ETL process to work with integrated entities. With LODicity, this custom development is no longer necessary and can be replaced entirely with generic LODicity capabilities.

```
44 public void rank() {
45  LOGGER.info("Ranking Terms...");
46  Map<Term, Integer> hitsForTerms = findNonZeroHitsForAllTerms();
47  int maxCount = findTermWithMostHits(hitsForTerms);
48  updateTermsWithNonZeroHits(hitsForTerms, maxCount);
49 }
50
51 private Map<Term, Integer> findNonZeroHitsForAllTerms() {
52  Map<Term, Integer> counts = new HashMap<>();
53  try {
54
55   Iterator<Term> it = termStoreWriter.iterator(TermStore.subjectOrLocationQuery);
56   while(it.hasNext()){
57    Term term = it.next();
58    String label = term.get("label");
59    int count = resourceStoreWriter.count(andQuery(exactMatch("subject", label),exactMatch("
          application", "library")));
60    if (count > 0)
61     counts.put(term, count);
62   }
63
64  } catch (Exception e) {
65   throw new RuntimeException(e);
66  }
67  return counts;
68 }
69
70 private Term updateTermWithRankInformation(Term term, int hits,
71   int maxCount) {
72  float normalizedRank = hits / maxCount;
73  term.set("rank", normalizedRank);
74  term.set("count", hits);
75
76  return term;
77 }
78
79 private void updateTermsWithNonZeroHits(Map<Term, Integer> hitsForTerms,
80   int maxCount) {
81  for (Map.Entry<Term, Integer> termWithHits : hitsForTerms
82     .entrySet()) {
83    Term term = termWithHits.getKey();
84
85    term = updateTermWithRankInformation(term,
86      termWithHits.getValue(), maxCount);
87
88    termStoreWriter.update(term);
89   }
90
91  termStoreWriter.commit();
92 }
93
94
95 private int findTermWithMostHits(Map<Term, Integer> hitsForTerms) {
96  int maxHits = 0;
97
98  for (Integer hits : hitsForTerms.values())
99   if (hits > maxHits)
100   maxHits = hits;
101
102  return maxHits;
103 }
```

Listing 5.3: Core of TermRanker without LODicity.

```
public void loadInto(Warehouse warehouse) {
 LOGGER.info("Ranking Terms...");

 Map<Term, Integer> counts = new HashMap<>();
 Filter termFilter = new Filter(Term.class);
 Filter resourceFilter = new Filter(Resource.class);




 warehouse.query(termFilter.or(termFilter.eq("type", "subject"), termFilter.eq("type", "location")))
       .forEach(term -> {


  Integer count = (int) (long) warehouse.count(resourceFilter.eq("subject", term.<String> get("label
        ")),resourceFilter.eq("application","library"));
  if (count > 0) {
   counts.put((Term) term, count);
  }
 });

 final int max = findTermWithMostHits(counts);


 warehouse.openTransaction();

 counts.forEach((term, count) -> {
  float normalizedRank = count / max;
  term.set("rank", normalizedRank);
  term.set("count", count);









  warehouse.massUpdate(term);
 });

 warehouse.commit();
}


private int findTermWithMostHits(Map<Term, Integer> hitsForTerms) {
 int maxHits = 0;

 for (Integer hits : hitsForTerms.values())
  if (hits > maxHits)
   maxHits = hits;

 return maxHits;
}
```

Listing 5.4: Core of TermRanker with LODicity, lines are adjusted to match corresponding functionality in Listing 5.3.

Not Increased Overall Effort

The Mediaplatform is designed to work with an optimised, use-case specific Lucene search store, specifically created by the dedicated Semantic ETL process. With LODicity, this process is split up into a general ETL step for entities and the specific creation of a Lucene store. In comparison, it is of interest how the overall development effort is changed by using LODicity. LODicity introduces a new architectural component, that moreover requires changing the existing ETL process as well.

Table 5.2 compares the LoC changes of classes modified for this proof of concept. Full cloc reports can be found in the Appendices D.3 and D.4 on page 452 and on page 453.

Class	Lines of Code		Difference	
	Original	PoC	LoC	Percent
dataobject.common.Schema.java	222	0	-222	-100%
datastore.common.TermStore.java	62	0	-62	-100%
datastore.library.gnd.TermRanker.java	68	46	-22	-32%
dataobject.common.DataObject.java	294	206	-88	-30%
service.library.LibraryService.java	90	86	-4	-4%
guide.library.LibraryCacheBuilder.java	201	195	-6	-3%
guide.museum.MuseumGuide.java	102	99	-3	-3%
service.LoadCoordinator.java	100	98	-2	-2%
service.museum.MuseumService.java	181	179	-2	-1%
datastore.library.hebis.HebisLoader.java	769	772	3	0%
datastore.museum.StaedelLoader.java	540	544	4	1%
datastore.library.ResourceStoreTermStoreEnrichment.java	86	87	1	1%
datastore.common.searchengine.LuceneWriteEngine.java	136	138	2	1%
datastore.common.file.FileService.java	54	55	1	2%
dataobject.common.Resource.java	222	227	5	2%
dataobject.common.Result.java	30	31	1	3%
dataobject.common.Topic.java	30	31	1	3%
datastore.library.gnd.GndLoader.java	100	104	4	4%
datastore.common.openthesaurus.OpenThesaurusLoader.java	56	60	4	7%
dataobject.common.Term.java	45	50	5	11%
dataobject.common.Person.java	32	37	5	16%
dataobject.common.Request.java	3	4	1	33%
Total changes	3,423	3,049	-374	-11%

Table 5.2: Comparison of the overall LoC changes in all modified classes.

Even though significant architectural features are added through LODicity, the LoC can be lowered by 11% in modified classes, compared to the previous straightforward scenario. These reductions are neither an intended goal of LODicity, nor they can

be expected in other applications. Accordingly, major reductions could be achieved by allowing more optimal code (classes TermStore and TermRanker) as well as the adoption and generalisation of best practice classes in LODicity (Schema and DataObject). In contrast, the largest increase of LoC through the introduction of LODicity is only 5 LoC per class, due to refactoring measurements.

Taking LoC as a rough estimation for the overall development effort, this demonstrates that use of LODicity does not increase development efforts compared to straightforward scenarios, but introduces the architectural benefits described earlier.

5.3.4 Functionally Equal

Besides conceptual and technical improvements, a data integration framework must conserve the existing functionality of the code base. After the integration of LODicity, loaders no longer create a Lucene search index directly, but store entities in the data warehouse, which is then used in a second, isolated step to create the search index. Thus, the data flow has changed entirely.

Executing on a search index created this way, all 87 tests pass just as before the integration of LODicity, as shown in Table 5.3. More details can be found in Appendices D.5 and D.6 on page 454 and on page 455.

Tests	Errors	Failures	Skipped	Success Rate	Time
87	0	0	2	97.701%	2.848

Table 5.3: Surefire test summary after the executing the test cases on LODicity-integrated data.

This demonstrates that while the ETL process was changed conceptionally and a new component was introduced, the outcome is still the same and not manipulated through the integration, write or read through LODicity.

Besides that, the existing Web applications for the ULB and the Städel Museum are still compatible with the newly created index.

5.4 Summary of Findings

To demonstrate the utility of the previously developed Linked and Open Data integration framework LODicity, this proof of concept integrates the framework into the existing source code of the Mediaplatform. Because it processes a number of Linked and Open Data and related data sources, the projects is a suitable case study. Thereby, the existing code base of the Mediaplatform is analysed and ported to use LODicity, as designed.

It is shown that the core functionality was preserved, as previously successful Mediaplatform test cases still executed successfully. In addition, the two existing Web applications remained fully functional.

Compared to a straightforward scenario, the development effort of using LODicity is minimal, even if it introduces a new architectural component that requires changing existing ETL code. Moreover, it can simplify program logic considerably. This allows the assumption that the framework matches a developer's workflow so that they can focus on the application-specific challenges and semantics.

The added data warehouse functionality allows for the distinction between the ETL and customisation of data, which introduces a number of advantages considering the use case specific delivery of data, the development processes and operating modes. A clean separation of these steps is demonstrated in this proof of concept by a trivial differential load scenario, allowing to successfully skip certain ETL steps.

Integrated entities are pooled application-wide, so they are available to additional components. Through this, the developer gains flexibility in their application design. Following the polyglot persistence paradigm (Section 2.8.3 on page 83), one or more delivering storage technologies can be selected independently, based on the application's use cases.

Considering these observations, it can be assumed that LODicity is useful for a real-world Linked and Open Data integration projects and helps developers in solving existing and future challenges.

5.5 Reflections on Findings

In the project Mediaplatform, over the years, the lack of a warehouse storage layer has had implications on the development of new features and components: The only way to interact with integrated entities, e.g. to enrich Term objects with the number of associated books, was to use the Lucene-based search index. Even though the Mediaplatform is a read-only application, this lead to the development of additional components with the sole purpose of iterating and updating Lucene index structures during the Semantic ETL process.

As there was no business case for doing that, these additional components are wasted development time and increased the complexity and fragility of the code base.

But this was not the only problem: Following good practices, the code of the Mediaplatform was designed to be technologically independent. With the Lucene-specific iterators and updaters, specific components were suddenly tightly wired to components that assumed a local Lucene index. This assumption turned out to be fatal when the use of a Solr cluster was considered for use in production. In a distributed system like Solr, iterating and updating individual entities can be very costly and introduce a number of further implications in regard of the provided consistency level. A migration was not directly possible without introducing further issues.

With LODicity, both problems would have been avoided: it already provides generic capabilities for the update and iteration of entities, so developers do not need to implement these features on their own. LODicity furthermore delegates to Hibernate, thus the iteration and update is handled by the underlying databases. Also, the steps for ETL, customisation and delivery can be strictly separated. A fatal mix of several storage zones could have been avoided in the first place.

In another scenario, a new recommender component required knowledge about the (integrated) Resource items, such as the top five epochs of the collection with the exact number of items associated. With the existing Lucene index, grouping and summing

operations are not ideally supported as these operations are not IR problems. Again, this is a case where additional efforts had to be invested, because a search engine was used in the delivery-phase of the system and it was the only way of accessing integrated data. In contrast, with its relational backend, grouping or summing operations, among many others, would have been provided easily with LODicity.

Chapter 6

Conclusion

After the huge success the Web has had, in the last decade, a subsequent revolution in data, the Linked and Open Data movements, becomes more and more popular. In this realm, customer-friendly data portals such as datahub.io or data.gov.uk have emerged and the number of freely available, ready to use Linked and Open Data sources have multiplied into thousands. Applications, however, have not. In fact, within the same period, the number of applications known to the data portals has stagnated. But without applications, plain data is useless to majority of the Web users and the upfront investments of governments and organisations into the complex technology stack of Linked and Open Data might be difficult to justify. The lack of applications needs to be addressed in order to avoid undermining the success and efforts put into Linked and Open Data. Existing research provides reference architectures for the integration and application of Linked and Open Data, which do not distinguish between generic and domain-specific applications. There are however strong indications that the required technical architectures in generic scenarios differ significantly from domain-specific scenarios. This leads to missing developer support and inadequate technological choices.

To address these issues, in the present work, an architecture is designed and implemented to support developers of specific semantic applications in the integration and use of Linked and Open Data. In order to do so, characteristics about real-world Linked and Open Data are measured, current storage technologies are compared and common data integration steps are identified. The resulting framework combines novel insights and best engineering practices.

6.1 Summary

This thesis studies how developers can be supported in the integration of Linked and Open Data towards applications (Section 1.3 on page 45). Herein, the four topics LOD characteristics, storage of Linked and Open Data, integration and use of Linked and Open Data as well as Use in practice are explored using design science methodology (Hevner et al., 2004).

To identify common Linked and Open Data characteristics such as format and license distribution, availability and structure of RDF, a field study is designed for CKAN repositories (Section 3.2.1 on page 97). This involves several SQL and detailed cluster analysis. The designed study is successfully executed on the representative data portal datahub.io.

As storing Linked and Open Data is essential when integrating it, a benchmark for storage technologies is designed for data warehouse scenarios based on the real Linked and Open Data of the HeBIS catalogue (Section 3.2.2 on page 106). It is successfully conducted on a number of current persistence choices in the domains relational, NoSQL and triple store technologies, namely sqlite4java, SQLite SQLite-Xerial, PostgreSQL, Apache Fuseki, Virtuoso, MongoDB, and ArangoDB (Section 4.2 on page 134).

To identify the logical and practical steps for the integration of Linked and Open Data, an architectural analysis and informed argument is made with the existing concepts of Vaisman and Zimányi, Fayyad et al., Pellegrini, Smith and Schirling, and Kosch et al. (Section 3.2.3 on page 114). Based on these identified logical steps, the approaches Media-platform, Catmandu, Apache Marmotta and Plain Semantic Web (Hausenblas, 2009) are compared, best practices are extracted and the successful development of prototypical Linked and Open Data integration framework, LODicity, is described (Section 4.3 on page 159).

To show the practical use of the framework, a proof-of-concept in the existing Linked and Open Data integration project Mediaplatform it is used to study the separation

of integration steps, execution time, lines of code, and preserved functionality (Section 3.2.4 on page 118). A detailed analysis documents the successful demonstration (Chapter 5 on page 197).

The present work thus provides a promising approach to address the lack of pragmatic architectures and, due to this, the lack of applications making use of Linked and Open Data. Developed software and the scientific foundation is hereby released as Open Source and Open Science.

6.2 Key Findings

The field study reveals that for Linked and Open Data on the representative data portal datahub.io, only about half of the data on datahub.io is clearly open, the majority bears legal uncertainty for application developers using it. Openness varies with the data formats: Excel is a very common format, but is usually not open. RDF is the third most common format, and is usually open. Accordingly, data on datahub.io is 53% Open Data, 15% Linked Data and about 10% Linked *and* Open Data. Moreover, after evaluating the 606 most popular RDF-based resources, nearly 20% are inaccessible due to parser errors and 33% do not exist anymore. The latter observation might be due to the fact that the vast majority of metadata is never updated after it has been published.

The broad scale cluster analysis of the most visited RDF datasets reveals that the resources show clearly more characteristics of homogeneous data than of heterogeneous data. Almost all of the 251 dendrograms[1] contain structures in form of properties that more or less exclusively co-exist with certain other properties. So, even if it is not a constraint of the format, RDF triples, in the wild, constitute somewhat differentiable entities that might fit in tabular structures as well as in graph-like ones.

The comparison of several current storage choices, namely sqlite4java, SQLite SQLite-Xerial, PostgreSQL, Apache Jena Fuseki, Virtuoso, MongoDB, and ArangoDB reveals quite different results for the different conceptual approaches. It is based on realistic

[1] `https://github.com/heussd/CKANstats/tree/master/datahub.io/png`, last access on 2015-06-09.

Linked and Open Data from the HeBIS catalogue.

Generally, relational DBMS (PostgreSQL, sqlite4java and SQLite-Xerial) perform well or best in almost all scenarios, provide the most stable operation, and can load data of all compared sizes with adequate speeds and offer the best ratio of throughput and load-stability. They are the first choice whenever application-specific entities are involved in the query results, yet they can compete in scenarios where this is not the case - even in simple graph scenarios. They show best caching behaviours and display suitable performances for data manipulation. In detail, PostgreSQL, sqlite4java and SQLite-Xerial perform roughly comparably - especially in low to medium sizes - though there are significant differences in certain scenarios, such as manipulation (Section 4.2.14 on page 155) and aggregation (Section 4.2.12 on page 152).

For non-relational approaches, the performance strongly depends on details of the query scenario in specific technological architectures.

Triple stores can load native RDF most rapidly, as long as the data is 100% free of errors. This is usually not a realistic assumption, as the previously conducted field study shows, as well as the utilised test data from HeBIS. Once loaded, triple stores perform well on certain aggregation scenarios (Section 4.2.12 on page 152), and in graph scenarios, though the performance is roughly comparable to the performance of relational DBMS. In scenarios containing application-specific entities, triple stores are outperformed by up to two orders of magnitude, making them basically unsuitable for these scenarios.

With its document store approach, MongoDB performs well and partially best for the mass-retrieval of application-specific entities (Section 4.2.11 on page 150) and in some data manipulation scenarios. In others, MongoDB lacks the flexibility to optimise data storage towards a scenario, which results in comparable bad performances (Section 4.2.13 on page 154).

ArangoDB performs even better in a certain mass-retrieval scenario, but worse in another (Section 4.2.11 on page 150). Even though data had been prepared for the graph

scenarios, results are not as good as for triple stores and relational DBMSs, but not as bad as MongoDB (Section 4.2.13 on page 154).

Based in these findings, a relational DBMS seems to be the most suitable technological foundation for a data warehouse. Results also indicate that the polyglot persistence paradigm (Section 2.8.3 on page 83) seems to be justified, as certain database technologies show, in detail, most different performance. Thereby, the range is from unsuitable performance up to best performance, depending on query scenarios, data preparation, and the capabilities of the database. However, the niches in which a specialised storage technology is superior might be quite small in practice.

The architectural analysis and informed argument of data integration approaches reveal that integration endeavours often involve similar, transferable steps. To address the needs of developers in the integration of Linked and Open Data in specific application scenarios, a pragmatic framework LODicity is developed. It covers architectural capabilities for ETL, storage and customisation, and provides developers with a means to focus on application-specific challenges.

LODicity implements a common data warehouse pattern for Linked and Open Data. By default, SQLite is used as storage backend, which displays reasonable performance for most scenarios, and which requires no additional configuration and runtime dependencies to other services. Just by changing the configuration, the storage backend can be changed.

The framework also supports current Java 8 language features, such as lambda expressions, which allow developers to write operations on integrated entities with few lines of code. It allows the definition and validation of custom entities in a spreadsheet and has a small technological footprint. This allows a large spectrum of different applications.

Once the data is integrated, the developer can almost instantly start solving application-specific challenges and thus can produce actual value from Linked and Open Data in a semantic application, while the framework handles routine and error-prone tasks.

The proof of concept application of LODicity integrates the framework into the existing source code of the Mediaplatform. It shows that the core functionality is preserved, as previously successful Mediaplatform test cases still execute successfully. In addition, the two existing Web applications remained fully functional.

Compared to the previous straightforward scenario, the development effort of using LODicity is minimal, even if it introduces a new architectural component that requires changing existing ETL code. Moreover, it can simplify program logic considerably. This allows the assumption that the framework matches into a developers workflow so that they can focus on the application-specific challenges and semantics.

The added data warehouse functionality allows distinguishing between the ETL and customisation of data, which introduces a number of advantages concerning the use case specific delivery of data, the development processes and operating modes. A clean separation of these steps is demonstrated in this proof of concept by a trivial differential load scenario, allowing to successfully skip certain ETL steps.

Integrated entities are pooled application-wide, so they are available to additional components. Through this, the developer gains flexibility in their application design. Following the polyglot persistence paradigm, one or more delivering storage technologies can be selected independently, based on the application's use cases.

Considering these observations, it can be assumed that LODicity is useful for a real-world Linked and Open Data integration projects and helps developers in solving existing and future challenges.

6.3 Contributions

In the creation of the framework for the integration and application of Linked and Open Data is based on substantial scientific contributions and contributions to practice.

6.3.1 Scientific Contribution

In the present work, a method is developed and implemented to analyse metadata of Linked and Open Data on CKAN-based data portals. Because it uses the CKAN-3-REST-API as well as SQL and R scripts, the analysis can be conducted on typical, common Linked and Open Data portals, such as data.gov, data.gov.uk, and govdata.de. In contrast to recent, related work in this field such as (Assaf et al., 2015), the developed approach allows a broad scale analysis to create a big picture of all data, based on their metadata.

To gather insights about the exact data formats used in published Linked and Open Data, a mapping table is created to assign the free-text-format statements on CKAN to unified, consistent type definitions. This table unifies over 400 different free-text-format statements. It is serialised as CSV, is public domain and can be reused, corrected and improved in the analysis of any CKAN-based data portal.

The analysis allows a fact-based quantification of Berners-Lee's popular 5-star model (Berners-Lee, 2006) through analysing a real and representative data portal datahub.io. It can be shown that to date, data is almost evenly distributed across the several star-classifications.

As a consequence of this distribution, RDF-only data integration approaches are substantially limited and do not reach a great majority of 85% of the public resources on datahub.io. This fact underlines the necessity that data integration solutions support a whole spectrum of different formats.

In addition, about half of the data does not fit into the 5-star model at all, as it does not clearly posess an open license. This quantifies and confirms the assumption in current research that "[d]ata may be linked but not open, or open but not thoroughly linked" (Edelstein et al., 2013, p. 2).

However, the rating in the 5-star model is proportional to the number of page views a resource gets: The more stars, the more page views. It seems that RDF is data-consumer

friendly.

LODprobe is a open source contribution to systematically scrutinise the structure that RDF triples form within datasets. It is accompanied by a number of specialised and optimised R scripts. This tooling allows to conduct a cluster analysis of the RDF property co-occurrences, and thus, provides novel means to assess the homogeneity or heterogeneity of RDF data.

The broad-scale analysis of RDF revealed that real datasets are clearly more homogeneous than they are heterogeneous. The assumption that "RDF knowledge bases are increasingly heterogeneous" (Morsey et al., 2011, p. 455) only seems to be true for a rather limited number of RDF datasets, especially the few large dumps on datahub.io. This is an essential contribution in selecting the most suitable storage solution for Linked and Open Data. Only triple stores are considered capable of storing heterogenous data efficiently, and thus, are commonly the only storage choice for Linked and Open Data in many architectures. This can not be observed for real RDF on datahub.io. Thus, unlike as is commonly assumed in existing architectures (Heath and Bizer, 2011, p. 99), (Auer et al., 2012, p. 9), (Kurz et al., 2011, p. 17), the role of triple stores as a ultimate storage choice for semantic applications can be questioned.

All latest storage comparisons in Web Science are limited to triple stores. Considering previous findings, this must be considered a gap. It does not comply to the finding of the previous field study of the format's multiplicity and to the homogeneity of the data, as well as not reflecting the multiplicity of storage choices that are available in practical engineering today. The conducted benchmark therefore is a contribution to address questions left open by existing benchmarks BSBM, LUBM, SP^{2}Bench, DBPSB and LDBC SPB. It is designed to be storage-agnostic and is not limited to a certain storage technology. The evaluated query scenarios (Table 3.9 on page 112) are selected based on the technical necessities in a data warehouse scenario, which is considered to be best practice in existing works (Gray et al., 2014). The benchmark definition and framework is publicly available and can be easily reconstructed and completed with new

database implementations.

Furthermore, the defined benchmark is implemented for a realistic selection of current storage choices, namely sqlite4java, SQLite SQLite-Xerial, PostgreSQL, Apache Fuseki, Virtuoso, MongoDB, and ArangoDB. Supported by the previous field study, entity-aware scenarios are a subject of the evaluation as well. Because of their fundamental storage concept, triple stores have to assemble entities out of atomic RDF triples to answer these queries. In this comparison, it is shown how much this affects performance, compared to relational DBMS and document stores. In summary, in scenarios where entities are involved, it is shown that triple store are outperformed by up to two orders of magnitude, compared to other approaches. Besides best practice reports such as (Gray et al., 2014) and (Handa, 2013), this confirms triple stores clearly as an architectural anti-pattern for entity-aware, application-specific scenarios. Again, this in contrast to the existing doctrine in common architectures, where triple stores are considered to be the fundamental storage choice, such as Apache Marmotta (Kurz et al., 2011) and the LOD2 Stack (Auer et al., 2012). The possible huge performance deficiencies further underline the need for current comparisons such as BSBM to not only include triple store technologies, but to expand their comparison on a more polyglot selection. According to the presented work, data warehouse scenarios are best implemented using relational DBMS.

Current Web Science research differentiates between generic applications and specific applications (Heath and Bizer, 2011, p. 85-90). This difference is not reflected in existing reference architectures. The present thesis shows first steps towards a reference architecture for specific applications utilising Linked and Open Data are taken. Based on the principal steps of data integration, it respects the necessities of the previous findings, such as the format multiplicity, the fact that application-specific entities exist and that they cannot be queried efficiently on promoted de-facto standard storage technology, triple stores. Moreover, it combines a number of best practices in research, such as data warehousing.

6.3.2 Contributions to Practice

The Semantic Web tool stack is often named with a number of advantages (Auer et al., 2011): there is no predefined data model, data is stored in universal formats, so it can be read and used everywhere, existing parsers can be reused and loading into databases is trivial.

This work shows that this not only has benefits, but also introduces a number of architectural implications, including technical constraints. Despite the fact that RDF has no predefined data model, it is not uncommon for the data to form homogenous rather than heterogeneous structures, which are suboptimal to be implemented and used with suggested RDF technologies. Only basic capabilities of the RDF are actually used in practice, but the parsers are usually complex and require building up the entire RDF graph model every time the data is handled. This complicates processing large but simple-featured datasets. Moreover, datasets might not actually have been created with Semantic Web tooling in the first place, but with Big Data tooling (Section 2.9.2 on page 87) or just exported from existing operational, relational databases. The high error rates of RDF known in current research (Section 2.2.4 on page 54) could be confirmed: Datasets contain errors in about half of the cases, which prevents subsequent tools from utilising their primary advantage: the ability to load data directly. Thus, proposed architectural patterns such as on-the-fly dereferencing (Heath and Bizer, 2011, p. 97) or the query federation (Heath and Bizer, 2011, p. 98) must be considered unfeasible in practice. Moreover, with general purpose RDF, case-specific error handling is more complex.

After all, RDF still has practical advantages. Properties like the simple and self-explanatory nature of RDF could explicitly be confirmed in this work and make RDF superior compared to other formats such as MARC. This also holds true even if the rest of the infrastructure is not Semantic Web-based, as RDF's fundamental design allows developers to easily implement custom parsers, as demonstrated in section 4.2.5 on page 140. This would explain why on data portals such as datahub.io, RDF is the

most popular Linked and Open Data format (Section 4.1.2 on page 124).

In existing RDF-based architectures, when handling non-RDF data, it is usually suggested to do "minting" (Hausenblas, 2009, p. 70f), the artificial conversion of arbitrary data into RDF. Based on the findings of this thesis, this can not be recommended if the data is queried in entity-aware scenarios. Instead of creating entity-aware RDF that can not efficiently be queried in triple stores, this work has shown that choosing a storage technology that is as entity-aware as the usage scenario is crucial for the application performance.

The present work implements a pragmatic approach to the Semantic Web tooling. A simple, streaming RDF parser is implemented in the context of the benchmark for data of the HeBIS catalogue. It does not create a graph model, and is error-tolerant in the case of minor syntax issues. This allows the efficient, low-memory load of even large RDF datasets, where plain Semantic Web tools such as Jena fail.

The evaluation of storage technologies shows that modern NoSQL storage technologies might only excel in certain niches, while traditional relational DBMS often display good general-purpose qualities. It is a contribution which allows developers in practice to assess storage technologies, depending on the scenario.

This work contributes LODicity, a novel framework and architecture for the integration of real-world Linked and Open Data, combining best practices of research and real-world projects such as the Mediaplatform. It provides developers with a means to easily define application-specific entities and provides persistence and query capabilities for these using evaluated storage technologies. With very few LoC, it handles default tasks for application developers and enables them to implement custom application semantics with fewer LoC.

In the realm of LODicity, a third-party software to employ SQLite with Hibernate was used. During development, however, a bug in that component was identified, fixed, and contributed to the public version (Appendix C.5 on page 445).

6.4 Limitations

The present work is an important contribution to the broader adoption of Linked and Open Data in practical, end-user-ready applications. It does, however, only address technical issues, such as the storage or a developer-friendly tooling. In practice, more factors might be involved than they can be covered in the present or in a technical work at all. Attractiveness and practical usefulness of the provided Linked and Open Data, for example, are non-technical factors that affect the adoption of Linked and Open Data significantly.

The present work proposes an architecture and implementation for the creation of application-specific semantic solutions. It involves the creation of application-specific entities and the streamlining towards specific use cases, and, consequently, purposes a non-Semantic Web technology stack. This, however, does not cover necessities of "generic applications" (Heath and Bizer, 2011, p. 85), such as general purpose knowledge systems. Those systems will still have common, basic (instead of specific) entities, whereas the Semantic Web technology stack might still be suitable.

The identified data quality problems on datahub.io can only partly be solved with technical capabilities. The created mapping table must considered to be just a temporary solution. In fact, there should rather be a more intelligent metadata acquisition in the first place, for example drop-down-menus and reasonable default settings for important fields like the license.

The in-depth analysis of RDF data is limited to the 606 most popular RDF resources.

Some databases such as Fuseki and ArangoDB are constantly evolving. It is possible that future releases address potentially identified weaknesses in this evaluation.

The individual claims of databases about scale-out, replication and backup capabilities have not been verified in query scenarios.

This comparison emphasises performance aspects, especially when evaluating databases. Especially during development, however, providing top-notch performance might not

always be of primary interest, but rather maintainability and the investment in the learning curve. Some databases such as Fuseki and ArangoDB provide very comfortable and sophisticated GUIs, which are not the subject of this evaluation.

The query scenarios have been selected to evaluate the specific application of databases in data warehouse scenarios. Thus, the present results do not necessarily allow performance statements for application scenarios unlike a data warehouse.

Not only query times are measured, but also the time it takes to produce Java representations of query results (Section 3.2.2 on page 111). This setup is valid as long as further semantics are intended to be implemented in Java, as in the present case. Some involved databases such as MongoDB and ArangoDB, however, might be designed to be employed in different environments, such as REST and JavaScript. This architectural intention is not considered in this benchmark, and plays, in practice, an important role for the final assessment of a certain technology.

LODicity does not currently contain a typical data warehouse staging zone (Section 2.5.2 on page 64), which, in a first step, stores to-be-imported data as-is without changing it. Because many different kinds of data come into question, adding this feature requires new, conceptual and implementation work to the existing approach.

LODicity also does not provide primary support for real-time data analysis, such as Complex Event Processing (CEP). In addition, a fundamental design assumption is that data can be handled on the local machine, unlike in Big Data scenarios. This is suitable for the observed properties of most Linked and Open Data (Section 4.1.5 on page 126). Again, extending the existing framework for these features involves additional engineering efforts. Possibly, the storage abstraction in Hibernate or the underlying storage technology such as PostgreSQL can be of use when scaling out due to data quantities.

6.5 Recommendation on Future Work

The present work is a promising foundation to support developers in the creation of semantic application-specific solutions based on Linked and Open Data. This is an important contribution to foster development of new and exiting applications. For a large scale adoption, a number of further steps seem useful.

From a technical point of view, the remaining planned features for LODicity (Section 4.3.4 on page 167) should be implemented. The pragmatic RDF parser (Section 4.3.4 on page 168), the generic transformation logic, support for SPARQL endpoints as well as license compatibility management (Section 4.3.4 on page 169) seem to address the next, most pressing issues.

Furthermore, promoting the developed solution to developers seems to be reasonable. A two-fold approach is preferable. Firstly, to make LODicity match completely in a developer's workflow, its dependencies should be retrievable through the Maven central repository. By simply defining a few lines of XML, LODicity is automatically retrieved and integrated into Java projects without additional efforts. Secondly, the framework itself should be marketed, for example, at practice-oriented events such as Open Data-themed Hackathrons[2]. In these events, working prototypes of applications based on Open Data have to be built within defined time frames. As LODicity supports developers handling Linked and Open Data, this kind of event seems suitable to demonstrate the capabilities of the framework in practice.

The present work shows that Linked and Open Data is distributed across various data formats and licenses. Nearly half of the data online at datahub.io does not possess a clearly open license. This data is not represented in Berners-Lee's 5-star model (Section 2.1.3 on page 50). In the opinion of the author, this data is a valuable resource too, and integration projects should provide capabilities to integrate such data as well. Giving this data a name might be an important first step. Thus, the author proposes to

[2] An example for such a Hackathron is Coding Da Vinci, `http://codingdavinci.de/`, last access on 2016-02-25.

introduce a 0-star ranking in extension of Berners-Lees 5-star model, aiming at data that unfortunately does not clearly indicate an open license, but is published online. Frameworks such as LODicity could then provide special support for the licensing demands and limitations of this data.

Last but not least, to make use of today's multicore CPUs, multiple loader chains could be executed in parallel, as long as they do not influence each other.

It could be interesting to find out if the cluster analysis from LODprobe can be used to conclude complete entities in arbitrary datasets. This could be used to help developers even further by suggesting custom entities.

Literature

AIDanial (2015), 'AlDanial/cloc'. Last access 2015-09-21.
https://github.com/AlDanial/cloc

Allemang, D. and Hendler, J. A. (2011), *Semantic Web for the Working Ontologist - Effective Modeling in RDFS and OWL, Second Edition*, Morgan Kaufmann.

ALM Works (2015), 'almworks / sqlite4java — Bitbucket'. Last access 2015-12-07.
https://bitbucket.org/almworks/sqlite4java

Angele, J. (2011), Ontobroker - mature and approved semantic middleware, *in* 'Semantic Web Journal'.

Anibaldi, S., Jaques, Y., Celli, F., Stellato, A. and Keizer, J. (2013), 'Migrating bibliographic datasets to the Semantic Web: The AGRIS case', *Semantic Web* . doi: 10.3233/SW-130128.
http://dx.doi.org/10.3233/SW-130128

Apache Any23 (2013), 'Introduction to Apache Any23'. Last access 2015-01-13.
https://any23.apache.org/index.html

Apache Clerezza (2015), 'Welcome to Apache Clerezza'. Last access 2015-12-20.
https://clerezza.apache.org/

Apache Jena (2014), 'Eyeball - checking RDF/OWL for common problems'. Last access 2014-08-06.
https://jena.apache.org/documentation/tools/eyeball-getting-started.html

Apache POI (2014), 'Apache POI - the Java API for Microsoft Documents'. Last access 2015-06-08.
https://poi.apache.org/index.html

Apache Software Foundation (2015), 'Welcome to The Apache Software Foundation!'. Last access 2015-11-03.
https://www.apache.org/

Assaf, A., Sénart, A. and Troncy, R. (2015), 'Roomba: automatic validation, correction and generation of dataset metadata'. doi: 10.1145/2740908.2742827.
http://www.eurecom.fr/publication/4525

Assel, M., Cheptsov, A., Gallizo, G., Celino, I., Dell'Aglio, D., Bradeško, L., Witbrock, M. and Valle, E. D. (2011), Large Knowledge Collider: A Service-oriented Platform for Large-scale Semantic Reasoning, *in* 'Proceedings of the International Conference on Web Intelligence, Mining and Semantics', WIMS '11, ACM, New York, NY, USA, pp. 41:1–41:9. doi: 10.1145/1988688.1988737.
http://doi.acm.org/10.1145/1988688.1988737

Auer, S. (2011), Creating Knowledge out of Interlinked Data: Making the Web a Data Washing Machine, *in* 'Proceedings of the International Conference on Web Intelligence, Mining and Semantics', WIMS '11, ACM, New York, NY, USA, pp. 4:1–4:8. doi: 10.1145/1988688.1988693.
http://doi.acm.org/10.1145/1988688.1988693

Auer, S., Bühmann, L., Dirschl, C., Erling, O., Hausenblas, M., Isele, R., Lehmann, J., Martin, M., Mendes, P. N., Nuffelen, B. v., Stadler, C., Tramp, S. and Williams, H. (2012), Managing the Life-Cycle of Linked Data with the LOD2 Stack, *in* P. Cudré-Mauroux, J. Heflin, E. Sirin, T. Tudorache, J. Euzenat, M. Hauswirth, J. X. Parreira, J. Hendler, G. Schreiber, A. Bernstein and E. Blomqvist, eds, 'The Semantic Web – ISWC 2012', number 7650 *in* 'Lecture Notes in Computer Science', Springer Berlin Heidelberg, pp. 1–16. doi: 10.1007/978-3-642-35173-0_1.
http://link.springer.com/chapter/10.1007/978-3-642-35173-0_1

Auer, S., Lehmann, J. and Ngomo, A.-C. N. (2011), Introduction to Linked Data and Its Lifecycle on the Web, *in* A. Polleres, C. d'Amato, M. Arenas, S. Handschuh, P. Kroner, S. Ossowski and P. Patel-Schneider, eds, 'Reasoning Web. Semantic Technologies for the Web of Data', number 6848 *in* 'Lecture Notes in Computer Science', Springer Berlin Heidelberg, pp. 1–75. doi: 10.1007/978-3-642-23032-5_1.
http://link.springer.com/chapter/10.1007/978-3-642-23032-5_1

Berners-Lee, T. (2006), 'Linked Data', Webpage. Last access 2012-02-11. doi: 10.4018/978-1-60960-593-3.ch008.
http://www.w3.org/DesignIssues/LinkedData.html

Berners-Lee, T. and Fischetti, M. (1999), *Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by Its Inventor*, Vol. 37, 1st edn, Harper San Francisco. doi: 10.5860/choice.37-3934.
http://dx.doi.org/10.5860/choice.37-3934

Berners-Lee, T., Hendler, J. and Lassila, O. (2001), 'The Semantic Web', *Scientific American* **284**(5), 34–43. Last access 2011-08-15. doi: 10.1038/scientificamerican0501-34.
http://www.scientificamerican.com/article.cfm?id=the-semantic-web

Bizer, C., Heath, T. and Berners-Lee, T. (2009), 'Linked Data - The Story So Far', *International Journal on Semantic Web and Information Systems (IJSWIS)* **5**(3), 1–22. doi: 10.4018/jswis.2009081901.
`http://dx.doi.org/10.4018/jswis.2009081901`

Bizer, C. and Schultz, A. (2009), 'The Berlin SPARQL Benchmark', *International Journal on Semantic Web and Information Systems (IJSWIS)* **5**(2), 1–24. doi: 10.4018/jswis.2009040101.
`http://dx.doi.org/10.4018/jswis.2009040101`

Bizer, C. and Schultz, A. (2011), 'BSBM V3 Results (February 2011)'. Last access 2015-01-13.
`http://wifo5-03.informatik.uni-mannheim.de/bizer/berlinsparqlbenchmark/results/V6/index.html#expBI`

Bizer, C. and Schultz, A. (2015), 'Berlin SPARQL Benchmark (BSBM)'. Last access 2015-01-10.
`http://wifo5-03.informatik.uni-mannheim.de/bizer/berlinsparqlbenchmark/`

Blumauer, A. (2013), 'The LOD cloud is dead, long live the trusted LOD cloud'. Last access 2014-08-04.
`http://blog.semantic-web.at/2013/06/07/the-lod-cloud-is-dead-long-live-the-trusted-lod-cloud/`

Borsje, S. (2014), 'How we build microservices at Karma'. Last access 2014-09-24.
`https://blog.yourkarma.com/building-microservices-at-karma`

Breslin, J. G., Passant, A. and Decker, S. (2009), *The Social Semantic Web*, 2010 edn, Springer Berlin Heidelberg, Berlin. doi: 10.1007/978-3-642-01172-6.
`http://link.springer.com/chapter/10.1007/978-3-642-01172-6_1`

Callimachus - 3 Round Stone (2015), 'Advantages of RDF over relational databases'. Last access 2015-01-10.
`http://callimachusproject.org/docs/1.0/articles/advantages-of-rdf-over-relational-databases.docbook?view`

Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Poggi, A., Rodriguez-Muro, M., Rosati, R., Ruzzi, M. and Savo, D. F. (2011), 'The MASTRO system for ontology-based data access', *Semantic Web* **2**(1), 43–53. doi: 10.3233/SW-2011-0029.
`http://dx.doi.org/10.3233/SW-2011-0029`

Catmandu Fix Language (2015), 'Catmandu'. Last access 2015-09-29.
`http://librecat.org/Catmandu/#fix-language`

Ceglowski, M. (2011), 'Remembrance of Links Past (Pinboard Blog)'. Last access 2014-04-22.
`https://blog.pinboard.in/2011/05/remembrance_of_links_past/`

Creative Commons (2016), 'Creative Commons — Attribution 3.0 Ireland — CC BY 3.0 IE'. Last access 2016-02-16.
`https://creativecommons.org/licenses/by/3.0/ie/`

Datahub.io (2010), 'New York Times - Linked Open Data - People (SKOS) - the Datahub'. Last access 2015-06-02.
`http://datahub.io/dataset/nytimes-linked-open-data/resource/1610e7b1-f8d2-4089-8d4a-0180ec8730f2`

Datahub.io (2015), 'About - the Datahub'. Last access 2015-06-11.
`http://datahub.io/about`

DBpedia (2015), 'Accessing the DBpedia Data Set over the Web'. Last access 2015-01-13.
`http://wiki.dbpedia.org/OnlineAccess`

Demter, J., Auer, S., Martin, M. and Lehmann, J. (2012), Lodstats—an extensible framework for high-performance dataset analytics, *in* 'Proceedings of the EKAW 2012', Lecture Notes in Computer Science (LNCS) 7603, Springer, pp. 353–362. doi: 10.1007/978-3-642-33876-2_31.
`http://svn.aksw.org/papers/2011/RDFStats/public.pdf`

Deuschel, T., Greppmeier, C., Humm, B. G. and Stille, W. (2014), Semantically Faceted Navigation with Topic Pies, *in* 'Proceedings of the 10th International Conference on Semantic Systems', SEM '14, ACM, New York, NY, USA, pp. 132–139. doi: 10.1145/2660517.2660522.
`http://doi.acm.org/10.1145/2660517.2660522`

Deuschel, T., Heuss, T. and Broomfield, C. (2014), The Design Scope of Adaptive Storytelling in Virtual Museums, *in* R. Klein and P. Santos, eds, 'Eurographics Workshop on Graphics and Cultural Heritage', The Eurographics Association, Darmstadt, Germany, pp. 97–106. doi: 10.2312/gch.20141308.
`http://diglib.eg.org/EG/DL/WS/GCH/GCH2014/097-106.pdf`

Digital Meets Culture (2014), 'Linking Cultural Heritage Information'. Last access 2014-09-22.

http://www.digitalmeetsculture.net/article/linking-cultural-heritage-information/

DINI, A. K. G. T. (2014), Empfehlungen zur rdf-repräsentation bibliografischer daten, number 14 *in* 'DINI-Schriften', Deutsche Initiative für Netzwerkinformation (DINI).
http://edoc.hu-berlin.de/docviews/abstract.php?id=40697

Dirschl, C. and Eck, K. (2015), Verlage müssen sich neu erfinden, *in* B. Ege, B. Humm and A. Reibold, eds, 'Corporate Semantic Web', X.media.press, Springer Berlin Heidelberg, pp. 129–143. doi: 10.1007/978-3-642-54886-4_10.
http://dx.doi.org/10.1007/978-3-642-54886-4_10

Dirschl, C., Pellegrini, T., Nagy, H., Eck, K., Nuffelen, B. V. and Ermilov, I. (2014), LOD2 for Media and Publishing, *in* S. Auer, V. Bryl and S. Tramp, eds, 'Linked Open Data – Creating Knowledge Out of Interlinked Data', Lecture Notes in Computer Science, Springer International Publishing, pp. 133–154. doi: 10.1007/978-3-319-09846-3_7.
http://link.springer.com/chapter/10.1007/978-3-319-09846-3_7

Edelstein, J., Galla, L., Li-Madeo, C., Marden, J., Rhonemus, A. and Whysel, N. (2013), 'Linked Open Data for Cultural Heritage: Evolution of an Information Technology'. doi: 10.1145/2507065.2507103.
http://dx.doi.org/10.1145/2507065.2507103

Edlich, S. (2015), 'NOSQL Databases'. Last access 2015-12-07.
http://nosql-database.org/

Edlich, S., Friedland, A., Hampe, J. and Brauer, B. (2011), *NoSQL: Einstieg in die Welt nichtrelationaler Web-2.0-Datenbanken*, 2., aktualisierte und erw. aufl. edn, Hanser, München. doi: 10.3139/9783446428553.
http://dx.doi.org/10.3139/9783446428553

Fayyad, U., Piatetsky-shapiro, G. and Smyth, P. (1996), 'From Data Mining to Knowledge Discovery in Databases', *AI Magazine* **17**, 37–54.

Ferrucci, D., Brown, E., Chu-Carroll, J., Fan, J., Gondek, D., Kalyanpur, A. A., Lally, A., Murdock, J. W., Nyberg, E., Prager, J., Schlaefer, N. and Welty, C. (2013), 'Building watson: An overview of the deepqa project'.

Forrester Consulting (2009), 'eCommerce Web Site Performance Today'. Last access: 2015-01-13.
https://www.scribd.com/document/50109180/Forrester-Consulting-eCommerce-Web-Site-Performance-Today

Franzon, E. (2014*a*), 'JSON-LD is an official Web Standard - Semanticweb.com'. Last access 2014-02-13.
http://semanticweb.com/j-son-ld-official-web-standard_b41441

Franzon, E. (2014*b*), 'RDF 1.1 is a W3C Recommendation'. Last access 2015-01-13.
http://www.dataversity.net/rdf-1-1-w3c-recommendation/

Freytag, J.-C. (2014), 'Grundlagen und Visionen großer Forschungsfragen im Bereich Big Data', *Informatik-Spektrum* **37**(2), 97–104. doi: 10.1007/s00287-014-0771-y.
http://link.springer.com/article/10.1007/s00287-014-0771-y

Gangemi, A. (2013), A Comparison of Knowledge Extraction Tools for the Semantic Web, *in* P. Cimiano, O. Corcho, V. Presutti, L. Hollink and S. Rudolph, eds, 'The Semantic Web: Semantics and Big Data', number 7882 *in* 'Lecture Notes in Computer Science', Springer Berlin Heidelberg, pp. 351–366. doi: 10.1007/978-3-642-38288-8_24.
http://link.springer.com/chapter/10.1007/978-3-642-38288-8_24

Gerber, A., van der Merwe, A. and Barnard, A. (2008), A functional semantic web architecture, *in* 'Proceedings of the 5th European semantic web conference on The semantic web: research and applications', ESWC'08, Springer-Verlag, Berlin, Heidelberg, pp. 273–287. doi: 10.1007/978-3-540-68234-9_22.
http://dl.acm.org/citation.cfm?id=1789394.1789422

GitHub (2015), 'gwenn/sqlite-dialect'. Last access 2015-12-01.
https://github.com/gwenn/sqlite-dialect

GitHub Virtuoso Issue Tracker (2015), 'SPARQL: ORDER BY not working with DESCRIBE queries · Issue #23 · openlink/virtuoso-opensource'. Last access 2015-12-21.
https://github.com/openlink/virtuoso-opensource/issues/23

GitHut (2015), 'GitHut - Programming Languages and GitHub'. Last access 2015-11-03.
http://githut.info/

Glimm, B., Hogan, A., Krötzsch, M. and Polleres, A. (2012), Owl: Yet to arrive on the web of data?, *in* C. Bizer, T. Heath, T. Berners-Lee and M. Hausenblas, eds, 'LDOW', Vol. 937 of *CEUR Workshop Proceedings*, CEUR-WS.org.
http://dblp.uni-trier.de/db/conf/www/ldow2012.html#GlimmHKP12

Goth, G. (2006), 'News: data-driven enterprise - slouching toward the semantic web', *Distributed Systems Online, IEEE* **7**(3). doi: 10.1109/mdso.2006.21.
http://dx.doi.org/10.1109/mdso.2006.21

Govdata.de (2013), 'Wo sind all die Open-Data-Apps?!?'. Last access 2015-01-13.
https:/ / www.govdata.de/ neues/ -/ blogs/ wo-sind-all-die-open-data-apps-

Gower, J. C. (1971), 'A General Coefficient of Similarity and Some of Its Properties', *Biometrics* **27**(4), 857–871. doi: 10.2307/2528823.
http://www.jstor.org/stable/2528823

Gray, A. J., Groth, P., Loizou, A., Askjaer, S., Brenninkmeijer, C., Burger, K., Chichester, C., Evelo, C. T., Goble, C., Harland, L., Pettifer, S., Thompson, M., Waagmeester, A. and Williams, A. J. (2014), 'Applying linked data approaches to pharmacology: Architectural decisions and implementation', *Semantic Web* **5**(2), 101–113. doi: 10.3233/SW-2012-0088.
http://dx.doi.org/10.3233/SW-2012-0088

Gray, J. (1993), Database and transaction processing performance handbook, *in* J. Gray, ed., 'The Benchmark Handbook for Database and Transaction Systems (2nd Edition).', Morgan Kaufmann.

Grisel, O. (2012), Automated linking data with apache stanbol, *in* '21st International World Wide Web Conference (WWW2012)'.
http://data.semanticweb.org/conference/www/2012/dev/42

Gunes, O. F., Furche, T., Schallhart, C., Lehman, J. and NGonga, A. (2012), EAGER: Extending automatically gazetteers for entity recognition, *in* 'Proc. of the 3rd Intl Work. on the People's Web meets NLP'.

Guo, Y., Pan, Z. and Heflin, J. (2005), 'LUBM: A Benchmark for OWL Knowledge Base Systems', *Web Semant.* **3**(2-3), 158–182. doi: 10.1016/j.websem.2005.06.005.
http://dx.doi.org/10.1016/j.websem.2005.06.005

Halevy, A., Franklin, M. and Maier, D. (2006), Principles of Dataspace Systems, *in* 'Proceedings of the Twenty-fifth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems', PODS '06, ACM, New York, NY, USA, pp. 1–9. doi: 10.1145/1142351.1142352.
http://doi.acm.org/10.1145/1142351.1142352

Handa, V. (2013), '[Nepomuk] Nepomuk in 4.13 and beyond'. Last access 2014-02-18.
https://mail.kde.org/pipermail/nepomuk/2013-December/004858.html

Hasso-Plattner-Institut (2014), 'Internet-Wachstum: Datenweb seit 2011 mehr als verdreifacht'. Last access 2014-09-26.

http://hpi.de/news/forschung/internet-wachstum-datenweb-seit-2011-mehr-als-verdreifacht.html

Hausenblas, M. (2009), 'Exploiting Linked Data to Build Web Applications', *IEEE Internet Computing* **13**(4), 68–73. doi: 10.1109/MIC.2009.79.
http://dx.doi.org/10.1109/MIC.2009.79

Heath, T. and Bizer, C. (2011), 'Linked Data: Evolving the Web into a Global Data Space', *Synthesis Lectures on the Semantic Web: Theory and Technology* **1**(1), 1–136. doi: 10.2200/S00334ED1V01Y201102WBE001.
http://www.morganclaypool.com/doi/abs/10.2200/S00334ED1V01Y201102WBE001

Hendler, J. (2001), 'Agents and the semantic web', *IEEE INTELLIGENT SYSTEMS* **16**(2), 30–37. doi: 10.1109/5254.920597.
http://dx.doi.org/10.1109/5254.920597

Heuss, T., Humm, B., Deuschel, T., Fröhlich, T., Herth, T. and Mitesser, O. (2015), *Linked Data and User Interaction*, DE GRUYTER SAUR, chapter Semantically Guided, Situation-Aware Literature Research, pp. 66–84. doi: http://dx.doi.org/10.1515/9783110317008-007.
http://dx.doi.org/10.1515/9783110317008-007

Hevner, A. R., March, S. T., Park, J. and Ram, S. (2004), 'Design Science in Information Systems Research', *MIS Q.* **28**(1), 75–105.
http://dl.acm.org/citation.cfm?id=2017212.2017217

Houle, P. (2014), 'The trouble with DBpedia'. Last access 2015-01-10.
http://blog.databaseanimals.com/the-trouble-with-dbpedia

Isaac, A., Waites, W., Young, J. and Zeng, M. (2005), Library linked data incubator group: Datasets, value vocabularies, and metadata element sets, Technical report, W3C Incubator Group Report.
http://www.w3.org/2005/Incubator/lld/XGR-lld-vocabdataset-20111025/

Jacobs, I. and Walsh, N. (2004), Architecture of the World Wide Web, Volume One, Url, W3C Recommendation. Last access 2014-10-11.
http://www.w3.org/TR/webarch/

Janowicz, K., Hitzler, P., Adams, B., Kolas, D. and Vardeman II, C. (2014), 'Five stars of linked data vocabulary use', *Semantic Web – Interoperability, Usability, Applicability an IOS Press Journal* .

Kämpgen, B. and Harth, A. (2013), No size fits all - running the star schema benchmark with SPARQL and RDF aggregate views, *in* P. Cimiano, Ó. Corcho, V. Presutti, L. Hollink and S. Rudolph, eds, 'The Semantic Web: Semantics and Big Data, 10th International Conference, ESWC 2013, Montpellier, France, May 26-30, 2013. Proceedings', Vol. 7882 of *Lecture Notes in Computer Science*, Springer Science + Business Media, pp. 290–304. doi: 10.1007/978-3-642-38288-8_20.
http://dx.doi.org/10.1007/978-3-642-38288-8_20

KDE Community Wiki (2014), 'Baloo'. Last access 2014-02-10.
http://community.kde.org/Baloo

KDE Nepomuk Manual (2014), 'Nepomuk switches to Baloo in KDE SC 4.13'. Last access 2015-01-13.
https://kdenepomukmanual.wordpress.com/

KDE UserBase Wiki (2015), 'Nepomuk'. Last access 2015-01-13.
https://userbase.kde.org/Nepomuk

Kernighan, B. W. and Pike, R. (1984), *The UNIX Programming Environment*, Vol. 9, Prentice-Hall. doi: 10.1002/spe.4380090102.
http://dx.doi.org/10.1002/spe.4380090102

Khadilkar, V., Kantarcioglu, M., Thuraisingham, B. M. and Castagna, P. (2012), Jena-hbase: A distributed, scalable and effcient rdf triple store., *in* B. Glimm and D. Huynh, eds, 'International Semantic Web Conference (Posters & Demos)', Vol. 914 of *CEUR Workshop Proceedings*, CEUR-WS.org.
http://dblp.uni-trier.de/db/conf/semweb/iswc2012p.html#KhadilkarKTC12

Klein, M., de Sompel, H. V., Sanderson, R., Shankar, H., Balakireva, L., Zhou, K. and Tobin, R. (2014), 'Scholarly context not found: One in five articles suffers from reference rot', *PLoS ONE* **9**(12), e115253. doi: http://dx.doi.org/10.1371/journal.pone.0115253.
http://dx.doi.org/10.1371/journal.pone.0115253

Kobilarov, G., Scott, T., Raimond, Y., Oliver, S., Sizemore, C., Smethurst, M., Bizer, C. and Lee, R. (2009), Media Meets Semantic Web – How the BBC Uses DBpedia and Linked Data to Make Connections, *in* L. Aroyo, P. Traverso, F. Ciravegna, P. Cimiano, T. Heath, E. Hyvönen, R. Mizoguchi, E. Oren, M. Sabou and E. Simperl, eds, 'The Semantic Web: Research and Applications', number 5554 *in* 'Lecture Notes in Computer Science', Springer Berlin Heidelberg, pp. 723–737. doi: 10.1007/978-3-642-

02121-3_53.
http://link.springer.com/chapter/10.1007/978-3-642-02121-3_53

Kohs, G. (2014), 'Google's Knowledge Graph Boxes: killing Wikipedia? | Wikipediocracy'.
http://wikipediocracy.com/2014/01/06/googles-knowledge-graph-killing-wikipedia/

Konda, M. (2014), *Just Hibernate*, 1 edition edn, O'Reilly Media.

Kosch, H., Boszormenyi, L., Doller, M., Libsie, M., Schojer, P. and Kofler, A. (2005), 'The life cycle of multimedia metadata', *IEEE MultiMedia* **12**(1), 80–86. doi: 10.1109/mmul.2005.13.
http://dx.doi.org/10.1109/mmul.2005.13

Kotsev, V., Kiryakov, A., Fundulaki, I. and Alexiev, V. (2015), LDBC Semantic Publishing Benchmark (SPB) - v2.0 First Public Draft Release, Technical report, Linked Data Benchmark Council.
https://github.com/ldbc/ldbc_spb_bm_2.0/

Krempl, S. (2015), 'Europäisches Open-Data-Portal geht mit 250.000 Datensätzen online | heise online'. Last access 2015-12-06.
http://www.heise.de/newsticker/meldung/Europaeisches-Open-Data-Portal-geht-mit-250-000-Datensaetzen-online-3010257.html

Kurz, T., Schaffert, S. and Burger, T. (2011), LMF: A Framework for Linked Media, *in* '2011 Workshop on Multimedia on the Web (MMWeb)', Institute of Electrical & Electronics Engineers (IEEE), pp. 16–20. doi: 10.1109/mmweb.2011.22.
http://dx.doi.org/10.1109/mmweb.2011.22

Larkou, G., Metochi, J., Chatzimilioudis, G. and Zeinalipour-Yazti, D. (2013), CLODA: A Crowdsourced Linked Open Data Architecture, *in* '2013 IEEE 14th International Conference on Mobile Data Management (MDM)', Vol. 2, Institute of Electrical & Electronics Engineers (IEEE), pp. 104–109. doi: 10.1109/MDM.2013.76.
http://dx.doi.org/10.1109/MDM.2013.76

Latif, A., Saeed, A. U., Höfler, P., Stocker, A. and Wagner, C. (2009), The linked data value chain: A lightweight model for business engineers, *in* 'I-SEMANTICS'09', pp. 568–575.

Library of Congress (2015), 'MARC 21 Format for Bibliographic Data: Summary Statement of Content Designators (Network Development and MARC Standards Office,

Library of Congress)'. Last access 2015-12-07.
http://www.loc.gov/marc/bibliographic/bdsummary.html

LibreCat (2015), 'LibreCat — Open Tools for Library and Research Services'. Last access 2015-09-29.
http://librecat.org/

Linden, G. (2006), 'Make Data Useful'.

Linked Data Benchmark Council (2015), 'Semantic Publishing Benchmark | LDBCouncil'. Last access 2015-12-10.
http://ldbcouncil.org/benchmarks/spb

Mandriva Linux Community (2012), 'Nepomuk how-to'. Last access 2015-01-13.
https://web.archive.org/web/20120707015053/http://wiki.mandriva.com/en/Nepomuk_how-to

Marmotta (2015*a*), 'Apache Marmotta - Home'. Last access 2015-09-29.
https://marmotta.apache.org/

Marmotta (2015*b*), 'Apache Marmotta - Platform'. Last access 2015-09-26.
http://marmotta.apache.org/platform/index.html

Marmotta LDCache (2015), 'Apache Marmotta - Linked Data Caching'. Last access 2015-09-29.
https://marmotta.apache.org/ldcache/

Marmotta LDPath (2015), 'Apache Marmotta - LDPath Query Language'. Last access 2015-09-29.
https://marmotta.apache.org/ldpath/

Maurhart, O. (2010), 'Wer braucht Nepomuk?'. Last access 2014-12-10.
http://www.gentooforum.de/artikel/18437/wer-braucht-nepomuk.html

Mayer, M. (2009), 'In search of ... a better, faster stronger web'.

McCreary, D. and Kelly, A. (2013), *Making Sense of NoSQL - A Guide for Managers and the Rest of Us*, pap/psc edn, Manning Publications Company, Birmingham.

Mendes, P. N., Jakob, M., García-Silva, A. and Bizer, C. (2011), Dbpedia spotlight: Shedding light on the web of documents, *in* 'Proceedings of the 7th International Conference on Semantic Systems', I-Semantics '11, ACM, New York, NY, USA, pp. 1–8. doi: 10.1145/2063518.2063519.
http://doi.acm.org/10.1145/2063518.2063519

Microdata (HTML) (2015). Page Version ID: 641781027.
https://en.wikipedia.org/w/index.php?title=Microdata_(HTML)&oldid=641781027

Möller, A. (2014), 'Gefällt mir - Webinterfaces mit React', *Linux-Magazin* pp. 104–108.

Morsey, M., Lehmann, J., Auer, S. and Ngomo, A.-C. N. (2011), DBpedia SPARQL Benchmark – Performance Assessment with Real Queries on Real Data, *in* L. Aroyo, C. Welty, H. Alani, J. Taylor, A. Bernstein, L. Kagal, N. Noy and E. Blomqvist, eds, 'The Semantic Web – ISWC 2011', number 7031 *in* 'Lecture Notes in Computer Science', Springer Berlin Heidelberg, pp. 454–469. doi: 10.1007/978-3-642-25073-6_29.
http://link.springer.com/chapter/10.1007/978-3-642-25073-6_29

Müller, S. (2014), 'Die neue Realität: Erweiterung des Data Warehouse um Hadoop, NoSQL & Co', *HMD Praxis der Wirtschaftsinformatik* **51**(4), 447–457. doi: 10.1365/s40702-014-0053-9.
http://link.springer.com/article/10.1365/s40702-014-0053-9

Nakagawa, E. Y. (2012), Reference Architectures and Variability: Current Status and Future Perspectives, *in* 'Proceedings of the WICSA/ECSA 2012 Companion Volume', WICSA/ECSA '12, ACM, New York, NY, USA, pp. 159–162. doi: 10.1145/2361999.2362032.
http://doi.acm.org/10.1145/2361999.2362032

Nelson, M. L. and Allen, B. D. (2002), 'Object Persistence and Availability in Digital Libraries', *D-Lib Magazine* **8**(1). doi: 10.1045/january2002-nelson.
http://www.dlib.org/dlib/january02/nelson/01nelson.html

Neo Technology, Inc. (2014), 'Neo4j Developer: Get Started'. Last access 2014-07-22.
http://www.neo4j.org/develop/linked_data

Neubert, J. and Tochtermann, K. (2012), Linked Library Data: Offering a Backbone for the Semantic Web, *in* D. Lukose, A. R. Ahmad and A. Suliman, eds, 'Knowledge Technology', number 295 *in* 'Communications in Computer and Information Science', Springer Berlin Heidelberg, pp. 37–45. doi: 10.1007/978-3-642-32826-8_4.
http://link.springer.com/chapter/10.1007/978-3-642-32826-8_4

Nielsen, J. (2010), 'Die Reaktionszeiten von Websites'. Last access 2012-09-26.
http://www.usability.ch/bn/news/alertbox/detail/die-reaktionszeiten-von-websites.html

Nimis, P. D. J., Armbruster, M. and Kammerer, M. (2014), Zukunftsfähiges Datenmanagement durch hybride Lösungen – Ein Entwurfsmusterkatalog zur Integration von SQL- und NoSQL-Datenbanken, *in* J. Jähnert and C. Förster, eds, 'Technologien für digitale Innovationen', Springer Fachmedien Wiesbaden, pp. 19–42. doi: 10.1007/978-3-658-04745-0_2.
`http://link.springer.com/chapter/10.1007/978-3-658-04745-0_2`

noBackend.org (2014), 'Build apps like there is noBackend!'. Last access 2014-10-17.
`http://nobackend.org/`

offlinefirst.org (2014), 'Offline First!'. Last access 2014-10-17.
`http://offlinefirst.org/`

Ong, K. W., Papakonstantinou, Y. and Vernoux, R. (2014), 'The SQL++ Unifying Semi-structured Query Language, and an Expressiveness Benchmark of SQL-on-Hadoop, NoSQL and NewSQL Databases', *arXiv:1405.3631 [cs]* . arXiv: 1405.3631.
`http://arxiv.org/abs/1405.3631`

Open Data Research Project (2013), 'Research project: Open Data Barometer'. Last access 2015-01-10.
`http://www.opendataresearch.org/project/2013/odb`

Open Knowledge Foundation (2013*a*), 'API guide — CKAN 2.4a documentation'. Last access 2015-05-24.
`http://docs.ckan.org/en/latest/api/index.html`

Open Knowledge Foundation (2013*b*), 'Page View Tracking — CKAN Data Management System Documentation 2.1a documentation'. Last access 2015-05-24.
`http://docs.ckan.org/en/tracking-fixes/tracking.html`

Open Knowledge Foundation (2013*c*), 'User guide — CKAN 2.4a documentation'. Last access 2015-05-24.
`http://docs.ckan.org/en/latest/user-guide.html#datasets-and-resources`

Open Knowledge Foundation (2015), 'Tracking the state of government open data.'. Last access 2015-01-10.
`https://index.okfn.org/`

O'Reilly, T. (2005), 'What Is Web 2.0?'. Last access 2016-02-29.
`http://www.oreilly.de/artikel/web20.html`

Pellegrini, T. (2012), Integrating Linked Data into the Content Value Chain: A Review of News-related Standards, Methodologies and Licensing Requirements, *in* 'Proceedings of the 8th International Conference on Semantic Systems', I-SEMANTICS '12, ACM, New York, NY, USA, pp. 94–102. doi: 10.1145/2362499.2362513.
http://doi.acm.org/10.1145/2362499.2362513

Peterse, Y. (2015), 'Goodbye MongoDB, Hello PostgreSQL'. Last access 2015-11-04.
http://developer.olery.com/blog/goodbye-mongodb-hello-postgresql/

Porter, M. E. (1985), *Competitive Advantage: Creating and Sustaining Superior Performance*, Free Press ; Collier Macmillan, New York; London.

PostgreSQL Documentation (2015), 'PostgreSQL: Documentation: 9.1: Arrays'. Last access 2015-10-05.
http://www.postgresql.org/docs/9.1/static/arrays.html

PostgreSQL 9.1.18 Documentation (2015). Last access 2016-09-12.
http://www.postgresql.org/docs/9.1

Prud'hommeaux, E., Archer, P. and Hawke, S. (2014), RDF Data Shapes Working Group Charter, Url, W3C Working Group. Last access 2014-09-23.
http://www.w3.org/2014/data-shapes/charter

Prud'hommeaux, E. and Seaborne, A. (2008), SPARQL Query Language for RDF, url, W3C. Last access 2015-10-02.
http://www.w3.org/TR/rdf-sparql-query/

PublicData.eu (2014), 'CSV2RDF Application'. Last access 2015-01-13.
http://wiki.publicdata.eu/wiki/CSV2RDF_Application

Richard Cyganiak (2014), 'lod-cloud/datahub2void - GitHub'. Last access 2014-09-21.
https://github.com/lod-cloud/datahub2void

Rizzo, G., Troncy, R., Hellmann, S. and Bruemmer, M. (2012), NERD meets NIF: Lifting NLP extraction results to the linked data cloud, *in* 'LDOW 2012, 5th Workshop on Linked Data on the Web, Volume 937, April 16, 2012, Lyon, France', Lyon, FRANCE.
https://www.eurecom.fr/publication/3675

Rodriguez, J. B. and Gómez-Pérez, A. (2006), Upgrading Relational Legacy Data to the Semantic Web, *in* 'Proceedings of the 15th International Conference on World Wide Web', WWW '06, ACM, New York, NY, USA, pp. 1069–1070. doi: 10.1145/1135777.1136019.
http://doi.acm.org/10.1145/1135777.1136019

rww.io (2015), 'rww.io: data cloud'. Last access 2015-12-20.
http://rww.io/

Schandl, B., Haslhofer, B., Bürger, T., Langegger, A. and Halb, W. (2012), 'Linked Data and multimedia: the state of affairs', *Multimedia Tools and Applications* **59**(2), 523–556. doi: 10.1007/s11042-011-0762-9.
http://link.springer.com/article/10.1007/s11042-011-0762-9

Schmachtenberg, M., Bizer, C., Jentzsch, A. and Cyganiak, R. (2014), 'The Linking Open Data cloud diagram'. Last access 2014-09-22.
http://lod-cloud.net/

Schmidt, M., Hornung, T., Lausen, G. and Pinkel, C. (2008), 'SP2Bench: A SPARQL Performance Benchmark', *arXiv:0806.4627 [cs]* . arXiv: 0806.4627. doi: 10.1109/icde.2009.28.
http://arxiv.org/abs/0806.4627

Schmidt, M., Hornung, T., Lausen, G. and Pinkel, C. (2009), SP^2Bench: A SPARQL Performance Benchmark, *in* 'IEEE 25th International Conference on Data Engineering, 2009. ICDE '09', Institute of Electrical & Electronics Engineers (IEEE), pp. 222–233. doi: 10.1109/ICDE.2009.28.
http://dx.doi.org/10.1109/ICDE.2009.28

Scholz, M. and Goerz, G. (2012), WissKI: A Virtual Research Environment for Cultural Heritage., *in* L. D. Raedt, C. Bessière, D. Dubois, P. Doherty, P. Frasconi, F. Heintz and P. J. F. Lucas, eds, 'ECAI', Vol. 242 of *Frontiers in Artificial Intelligence and Applications*, IOS Press, pp. 1017–1018.
http://dblp.uni-trier.de/db/conf/ecai/ecai2012.html#ScholzG12

Schurman, E. and Brutlag, J. (2009), 'The User and Business Impact of Server Delays, Additional Bytes, and HTTP Chunking in Web Search Presentation'.

Schürmann, T. (2014), 'Reaktionsfreudig - Reactive Programming', *Linux-Magazin* pp. 28–32.

Semantic Web Company (2015), 'PoolParty Semantic Integrator'. Last access 2015-01-10.
http://www.poolparty.biz/portfolio-item/semantic-integrator/

Smith, J. and Schirling, P. (2006), 'Metadata standards roundup', *IEEE MultiMedia* **13**(2), 84–88. doi: 10.1109/mmul.2006.34.
http://dx.doi.org/10.1109/mmul.2006.34

Speicher, S., Arwe, J. and Malhotra, A. (2014), Linked Data Platform 1.0, Url, W3C Technical Report. Last access 2014-09-22.
`http://www.w3.org/TR/2014/WD-ldp-20140311/`

Sporny, M., Longley, D., Kellogg, G., Lanthaler, M. and Lindström, N. (2014), JSON-LD 1.0, W3C Recommendation, W3C.
`http://www.w3.org/TR/json-ld/`

Stack Overflow (2015), 'java - Getting [SQLITE_BUSY] database file is locked with select statements - Stack Overflow'. Last access 2015-12-02.
`https://stackoverflow.com/questions/13891006/getting-sqlite-busy-database-file-is-locked-with-select-statements`

Stadtmüller, S., Harth, A. and Grobelnik, M. (2013), Accessing Information About Linked Data Vocabularies with vocab.cc, *in* J. Li, G. Qi, D. Zhao, W. Nejdl and H.-T. Zheng, eds, 'Semantic Web and Web Science', Springer Proceedings in Complexity, Springer New York, pp. 391–396. doi: 10.1007/978-1-4614-6880-6_34.
`http://link.springer.com/chapter/10.1007/978-1-4614-6880-6_34`

Stardog (2014), 'http://stardog.com'. Last access 2014-09-20.
`http://stardog.com`

Summers, E. and Salo, D. (2013), 'Linking Things on the Web: A Pragmatic Examination of Linked Data for Libraries, Archives and Museums', *arXiv:1302.4591 [cs]* . doi: 1302.4591.
`http://arxiv.org/abs/1302.4591`

Sun, J. and Jin, Q. (2010), Scalable RDF store based on HBase and MapReduce, *in* '2010 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE)', Vol. 1, Institute of Electrical & Electronics Engineers (IEEE), pp. V1–633–V1–636. doi: 10.1109/icacte.2010.5578937.
`http://dx.doi.org/10.1109/icacte.2010.5578937`

The Reactive Manifesto (2014), 'The Reactive Manifesto'. Last access 2014-09-24.
`http://www.reactivemanifesto.org/`

Thurner, T. (2014), 'Linked Data Stack | LOD2 Technology Stack – Linked Open Data Tools and Services'. Last access 2014-08-18.
`http://stack.lod2.eu/blog/`

Tiobe Index (2015), 'TIOBE Software: Tiobe Index'. Last access 2015-11-03.
`http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html`

Tummarello, G. (2014), 'LOD publishing question'. Last access 2015-01-13.
`http://lists.w3.org/Archives/Public/public-lod/2014Jan/0070.html`

Ubuntuusers Wiki (2014), 'Nepomuk'. Last access 2014-12-10.
`http://wiki.ubuntuusers.de/Nepomuk`

Vaisman, A. and Zimányi, E. (2014), Data Warehouse Concepts, *in* 'Data Warehouse Systems', Data-Centric Systems and Applications, Springer Berlin Heidelberg, pp. 53–87. doi: 10.1007/978-3-642-54655-6_3.
`http://link.springer.com/chapter/10.1007/978-3-642-54655-6_3`

Vatant, B. (2012), 'LOV stories, Part 2 : Gardeners and Gatekeepers'. Last access 2014-09-22.
`http://bvatant.blogspot.de/2012/03/lov-stories-part-2-gardeners-and.html`

Vavliakis, K. N., Symeonidis, A. L., Karagiannis, G. T. and Mitkas, P. A. (2011), 'An Integrated Framework for Enhancing the Semantic Transformation, Editing and Querying of Relational Databases', *Expert Syst. Appl.* **38**(4), 3844–3856. doi: 10.1016/j.eswa.2010.09.045.
`http://dx.doi.org/10.1016/j.eswa.2010.09.045`

W3C OWL Working Group (2012), http://www.w3.org/TR/owl2-overview/, Url, W3C. Last access 2014-12-08.
`http://www.w3.org/TR/owl2-overview/`

Wallis, R. (2014), 'WorldCat Works Linked Data – Some Answers To Early Questions'. Last access 2015-01-13.
`http://dataliberate.com/2014/03/worldcat-works-linked-data-some-answers-to-early-questions/`

WaSABi Organizing Committee (2014), 'Introduction - WaSABi SI 2014'. Last access 2014-09-30.
`http://www.wasabi-ws.org/`

Webpage (2014), 'LODStats - 1048 datasets'. Last access 2014-12-08.
`http://stats.lod2.eu/`

Wikipedia (2014), 'Complete-linkage clustering'. Page Version ID: 625941679.
`http://en.wikipedia.org/w/index.php?title=Complete-linkage_clustering&oldid=625941679`

Wolters, R. (2007), 'Semantic Desktop and KDE 4 – State and Plans of Nepomuk-KDE [Update]'. Last access 2015-01-13.

https://liquidat.wordpress.com/2007/05/31/semantic-desktop-and-kde-4-state-and-plans-of-nepomuk-kde/

Wood, D. (2014), 'What's New in RDF 1.1'. Last access 2015-01-13.
http://www.w3.org/TR/2014/NOTE-rdf11-new-20140225/

Yosef, M. A., Hoffart, J., Bordino, I., Spaniol, M. and Weikum, G. (2011), 'Aida: An online tool for accurate disambiguation of named entities in text and tables', *PVLDB* **4**(12), 1450–1453.

Zaino, J. (2013), 'W3C's Semantic Web Activity Folds Into New Data Activity'. Last access 2013-12-13.
http://semanticweb.com/w3cs-semantic-web-activity-folds-new-data-activity/

List of Publications

2016 Timm Heuss, Janina Fengel, Bernhard Humm, Bettina Harriehausen-Mühlbauer, and Shirley Atkinson. A Field Study on Linked and Open Data at Datahub.io. In Sergej Alekseev, Paul Dowland, Bogdan V. Ghita, and Oliver Schneider, editors, *Eleventh International Network Conference, INC 2016, Frankfurt, Germany, July 19-21, 2016. Proceedings*, pages 79–84. Plymouth University, 2016.

2015 Thomas Hoppe, Bernhard Humm, Ulrich Schade, Timm Heuss, Matthias Hemmje, Tobias Vogel, and Benjamin Gernhardt. Corporate Semantic Web – Applications, Technology, Methodology. *Informatik-Spektrum*, pages 1–7, Nov. 2015.

2015 Tilman Deuschel, Timm Heuss, and Bernhard Humm. Die Medienplattform: Ein System für gerichtete und ungerichtete semantische Suchen. *Information - Wissenschaft & Praxis*, 35:201–206, August 2015.

2015 Timm Heuss, Bernhard Humm, Tilman Deuschel, Torsten Fröhlich, Thomas Herth, and Oliver Mitesser. *Linked Data and User Interaction*, chapter Semantically Guided, Situation-Aware Literature Research In Frank H. Cervone and Lars G. Svensson, editors, pages 66–84. DE GRUYTER SAUR, 2015.

2015 Bernhard Humm and Timm Heuss. Schlendern durch digitale Museen und Bibliotheken. In B. Ege, Bernhard Humm, and A. Reibold, editors, *Corporate Semantic Web*, X.media.press, pages 59–70. Springer Berlin Heidelberg, 2015.

2014 Timm Heuss, Bernhard Humm, Christian Henninger, and Thomas Rippl. A Comparison of NER Tools W.R.T. A Domain-specific Vocabulary. In *Proceedings of the 10th International Conference on Semantic Systems*, SEM '14, pages 100–107, New York, NY, USA, 2014. ACM.

2014 Tilman Deuschel, Timm Heuss, and Christian Broomfield. The Design Scope of Adaptive Storytelling in Virtual Museums. In R. Klein and P. Santos, editors, *Eurographics Workshop on Graphics and Cultural Heritage*, pages 97–106, Darmstadt, Germany, 2014. Eurographics Association.

2014 Tilman Deuschel, Timm Heuss, and Bernhard Humm. The digital online museum: A new approach to experience virtual heritage. In T. Risse, L. Predoiu, A. Nürnberger, and S. Ross, editors, *Proceedings of the 4th International Workshop on Semantic Digital Archives (SDA 2014) co-located with the International Digital*

Libraries Conference (DL 2014), London, UK, September 12, 2014., volume 1306 of *CEUR Workshop Proceedings*, pages 38–48. CEUR-WS.org, 2014.

2014 Tilman Deuschel, Timm Heuss, Bernhard Humm, and Torsten Fröhlich. Finding without searching: A serendipity-based approach for digital cultural heritage. In *To Appear: Proceedings of the Digital Intelligence (DI) conference, Nantes, Frankreich, Sept. 17-19, 2014*, 2014.

2013 Timm Heuss. Faust.rdf - Taking RDF literally. In *2nd Workshop on Linked Data in Linguistics (LDL-2013): Representing and linking lexicons, terminologies and other language data. Collocated with the Conference on Generative Approaches to the Lexicon*, Pisa, Italy, September 2013.

2013 Timm Heuss. Lessons learned (and questions raised) from an interdisciplinary Machine Translation approach. In *W3C Workshop on the Open Data on the Web, 23 - 24 April 2013, Google Campus, Shoreditch, London, United Kingdom*, 2013.

2012 Bettina Harriehausen-Mühlbauer and Timm Heuss. Semantic Web Based Machine Translation. In *Proceedings of the Joint Workshop on Exploiting Synergies Between Information Retrieval and Machine Translation (ESIRMT) and Hybrid Approaches to Machine Translation (HyTra)*, EACL 2012, pages 1–9, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

2011 Timm Heuss. Social Resource Promotion. In *Federated Social Web - W3C Workshop on the Federated Social Web, June 3rd - 5th 2011, Berlin, Germany*, 2011.

List of Open Source Contributions

During research of the present work, the following open source software and Open Data was contributed by the author:

CKANstats.py , a Python script to extract all metadata about datasets on CKAN data repositories.
`https://github.com/heussd/CKANstats/blob/dd811215c677073e072aae9a1bac8ba57358b0e1/CKANstats.py`, last access on 2015-11-29

mptbl , a mapping table to assign free text file format definitions in CKAN repositories towards defined file extensions.
`https://github.com/heussd/CKANstats/blob/master/mptbl.csv`, last access on 2015-11-29

LODprobe , a Java tool to analyse the homogeneity of RDF datasets.
`https://github.com/heussd/LODprobe`, last access on 2015-11-29

A number of R scripts to parse, analyse and visualise the output of LODprobe.
`https://github.com/heussd/LODprobe`, last access on 2015-11-29

Benchmark specification and implementation for Linked and Open Data integration scenarios, with partial support of Marco Hamm.
`https://github.com/heussd/loddwhbench`, last access on 2015-11-29

Detailed benchmark evaluation results for the databases SQLite-Xerial, sqlite4java, PostgreSQL, Virtuoso, Fuseki, MongoDB, and ArangoDB, with partial support of Marco Hamm.
`https://github.com/heussd/loddwhbench/tree/master/results`, last access on 2015-11-29

A patch for "Glenn's" SQLite Hibernate Dialect
`https://github.com/heussd/sqlite-dialect`, last access on 2015-11-07

LODicity , a Linked and Open Data data integration framework for Java-based applications.
`https://github.com/heussd/lodicity`, last access on 2015-11-29

Appendix A

Field Study Appendix

A.1 CKANstats.py

```
#!/usr/bin/env python

# Retrieves meta data from CKAN repositories and writes it into a CSV file
# Some snippets taken from http://docs.ckan.org/en/ckan-2.0/api.html


import urllib2
import urllib
import json
import pprint

import csv

repository_url = "datahub.io" # <--- Change this
delimiter = ','


package_list_url = 'http://' + repository_url + '/api/3/action/package_list'

# http://demo.ckan.org/api/3/action/package_search?q=name:ckandown
package_search_url = 'http://' + repository_url + '/api/3/action/package_search'



# Removes non-ASCII / Delimiter characters
def cleanFieldValue(fieldValue):
 if not (isinstance(fieldValue, bool) or isinstance(fieldValue, int)):
  fieldValue = fieldValue.encode("ascii", 'ignore')
  fieldValue = fieldValue.replace(delimiter, "")
 return fieldValue


# Use the json module to dump a dictionary to a string for posting.

data_string = ""

# Make the HTTP request.
list_response = urllib2.urlopen(package_list_url)
assert list_response.code == 200

# Use the json module to load CKAN's response into a dictionary.
package_list = json.loads(list_response.read())

# Check the contents of the response.
assert package_list['success'] is True


with open(repository_url + '.csv', mode='wb') as csvfile:
 csvwriter = csv.writer(csvfile, delimiter=delimiter,quotechar='|', quoting=csv.QUOTE_MINIMAL)

 # dataset.id
 # dataset.name
 # dataset.license_title
 # dataset.license_id
 # resource.id
 # resource.name
 # resource.created
 # resource.revision_timestamp
 # resource.format
```

```
# resource.url
csvwriter.writerow(['dataset.id','dataset.name','dataset.license_title','dataset.license_id','
    dataset.isopen','dataset.tracking_summary.total','dataset.tracking_summary.recent','resource
    .id','resource.name','resource.created','resource.revision_timestamp','resource.format','
    resource.url','resource_tracking_summary_total','resource_tracking_summary_recent'])

for package_name in package_list['result']:
 data_string = urllib.quote(json.dumps({'q': ('name:' + package_name)}))

 response = urllib2.urlopen(package_search_url, data_string)
 assert response.code == 200

 search_result = json.loads(response.read())

 assert search_result['success'] is True

 # Filter out non-existent datasets
 if search_result['result']['count'] is 0:
  continue

 assert search_result['result']['count'] is 1

 for inner_result in search_result['result']['results']:
  dataset_id = cleanFieldValue(inner_result['id'] or "n/a")
  dataset_name = cleanFieldValue(inner_result['name'] or "n/a")
  dataset_license_title = cleanFieldValue(inner_result['license_title'] or "n/a")
  dataset_license_id = cleanFieldValue(inner_result['license_id'] or "n/a")
  dataset_is_open = cleanFieldValue(inner_result['isopen'] or "False")
  # http://docs.ckan.org/en/tracking-fixes/tracking.html#retrieving-tracking-data
  dataset_tracking_summary_total = cleanFieldValue(inner_result['tracking_summary']['total'] or "
      0")
  dataset_tracking_summary_recent = cleanFieldValue(inner_result['tracking_summary']['recent'] or
       "0")

  print inner_result['name']

  for resource in inner_result['resources']:
   resource_id = cleanFieldValue(resource['id'] or "n/a")
   resource_name = cleanFieldValue(resource['name'] or "n/a")
   resource_created = cleanFieldValue(resource['created'] or "n/a")
   resource_revision_timestamp = cleanFieldValue(resource['revision_timestamp'] or "n/a")
   resource_format = cleanFieldValue(resource['format'] or "n/a")
   resource_url = cleanFieldValue(resource['url'] or "n/a")

   resource_tracking_summary_total = cleanFieldValue(resource['tracking_summary']['total'] or "0"
       )
   resource_tracking_summary_recent = cleanFieldValue(resource['tracking_summary']['recent'] or "
       0")


   # Python 2's CSVWriter does not seem to support UTF-8 :/
   # https://github.com/jdunck/python-unicodecsv
   csvwriter.writerow([dataset_id,dataset_name,dataset_license_title,dataset_license_id,
       dataset_is_open,dataset_tracking_summary_total,dataset_tracking_summary_recent,
       resource_id,resource_name,resource_created,resource_revision_timestamp,resource_format,
       resource_url,resource_tracking_summary_total,resource_tracking_summary_recent])
```

Listing A.1: Source code of the Python script CKANstats.py.

A.2 Mapping Table

expr	format	star
rest_api	API	1
api/search	API	1
rest service	API	1
api/query	API	1
|api/xml||json|	API	1
api	API	1
solr	API	1
web_services	API	1
zip:bib	Archive	1
zip:8c	Archive	1
rar:shp	Archive	1
tar.bz2	Archive	1
bzip2:text/sql	Archive	1
application/x-tgz	Archive	1
application/gzip	Archive	1
tgz	Archive	1
gzip	Archive	1
application/x-zip-compressed	Archive	1
zip archive	Archive	1
.tar.gz	Archive	1
rar	Archive	1
iso	Archive	1
gz	Archive	1
tar.gz	Archive	1
bz_	Archive	1
x-zip-compressed	Archive	1
zip	Archive	1
tar	Archive	1
application/x-gzip	Archive	1
application/x-bzip	Archive	1
beacon	Beacon	3
application/octet-stream	Binary	1
jar	Binary	1
binary	Binary	1
text/x-csv	CSV	3
|tsv|	CSV	3
tsv	CSV	3
csv	CSV	3
zip:csv	CSV	3
tab-separated-values	CSV	3

expr	format	star
comma-separated-values	CSV	3
csv:1	CSV	3
csvxml	CSV	3
csvjson	CSV	3
csv stata excel	CSV	3
text/x-osdata-csv	CSV	3
fixed width	CSV	3
text/tab-separated-values	CSV	3
csv (zip)	CSV	3
csv xml	CSV	3
tsv.gz	CSV	3
bz2:csv	CSV	3
csv (zipped)	CSV	3
|tgz||tsv|	CSV	3
zip csv shp	CSV	3
|bittorrent||zip||tsv|	CSV	3
mysqlcsv	CSV	3
csv (zipado)	CSV	3
text/x-osmapping-csv	CSV	3
text/comma-separated-values	CSV	3
application/vnd.openxmlformats-officedocument.wordprocessingml.document	DOC	1
epub	DOC	1
application/vnd.oasis.opendocument.text	DOC	1
doc:04	DOC	1
google doc	DOC	1
msword	DOC	1
sdf	DOC	1
wordprocessingml.document	DOC	1
doc	DOC	1
opendocument_text	DOC	1
rtf	DOC	1
application/msword	DOC	1
application/dbase	Database	2
dbase	Database	2
application/dbase+dbc	Database	2
spss	Database	2
access database	Database	2
ms_access	Database	2
data file in stata	Database	2
mdb	Database	2
dbf	Database	2
dta	Database	2

expr	format	star
data file in spss	Database	2
data file in spss and stata	Database	2
data file in access	Database	2
ms access mdb	Database	2
sisis export format	Database	2
microsoft access database	Database	2
data file in access and spss	Database	2
ms acess mdb	Database	2
stata	Database	2
|csv || mdb|	Database	2
gz:sql	Database	2
csv/sqlite	Database	3
sqlite	Database	3
application/zip+x-sqlite3	Database	3
sqlite 3	Database	3
gzip:sql	Database	3
example/rdf+xml	Example	
example	Example	
example/x-turtle	Example	
example/rdf+json	Example	
example/application/rdf+xml	Example	
example/rdf+n3	Example	
example/rdf+ttl	Example	
example/*	Example	
example/n3	Example	
example/ntriples	Example	
example/warc	Example	
example/rdfa	Example	
example/rdf xml	Example	
example/turtle	Example	
gtfs	GTFS	3
geotiff	Geo	2
esri shapefiles	Geo	2
.gml	Geo	2
esri shape	Geo	2
google-earth.kml	Geo	2
application/x-esri%	Geo	2
esri grid	Geo	2
kml/shp	Geo	2
esri shape files	Geo	2
ppt kmz doc	Geo	2
esri	Geo	2

expr	format	star
shp	Geo	2
esri arc export	Geo	2
esri shape file	Geo	2
.geojson	Geo	2
wms	Geo	2
geotif	Geo	2
geopdf	Geo	2
geojson	Geo	2
esri shapefile	Geo	2
yml	Geo	2
gtfs-realtime	Geo	2
application/vnd.google-earth.kml+xml	Geo	2
shapefile	Geo	2
kmz	Geo	2
gml	Geo	3
w___/gml	Geo	3
wfs - gml	Geo	3
kml	Geo	3
.kml	Geo	3
kml/kmz	Geo	3
application/xhtml+xml	HTML	1
xhtml	HTML	1
html	HTML	1
htm	HTML	1
xhtml+rdfa	HTML	1
html/doc	HTML	1
html page	HTML	1
example/html+rdfa	HTML	1
html5	HTML	1
xhtml rdf/xml turtle	HTML	1
html+rdfa	HTML	1
tml	HTML	1
image (nifti)	Image	1
ai	Image	1
png jpg	Image	1
png	Image	1
.png	Image	1
image/png	Image	1
tiff	Image	1
jpeg	Image	1
gif	Image	1
image/jpeg	Image	1

expr	format	star
image/tiff	Image	1
png jpg txt	Image	1
shape jpg pdf	Image	1
SHAPE	Image	1
jpg	Image	1
dxf	Image	1
json in zip	JSON	3
jsonp	JSON	3
zip:json	JSON	3
json xml	JSON	3
json	JSON	3
|application/tgz||text/jsondir|	JSON	3
xml json	JSON	3
topojson	JSON	3
marc21	MARC	3
application/marc	MARC	3
marc	MARC	3
marc 21	MARC	3
marcxml	MARC	3
tomtom	Map	1
garmin	Map	1
map	Map	1
zip:pdf	PDF	1
pdf:0_documentation	PDF	1
pdf	PDF	1
|html || pdf|	PDF	1
html pdf	PDF	1
bz2:nt	RDF	4
text/ntriples	RDF	4
compressed tarfile containing n-triples	RDF	4
rdf nt	RDF	4
gz:ttl	RDF	4
ttl:e1:csv	RDF	4
7z:ttl	RDF	4
mapping/rdfs	RDF	4
application/rdf+json	RDF	4
owl ontology meta/owl	RDF	4
text/turtle	RDF	4
application/ld+json	RDF	4
rdf+xml	RDF	4
json-ld	RDF	4
meta/rdf+schema	RDF	4

expr	format	star
gz:nq	RDF	4
appliation/x-trig	RDF	4
html rdf dcif	RDF	4
xml rdf+xml turtle and json	RDF	4
application/ntriples	RDF	4
ttl.bzip2	RDF	4
rdf/license	RDF	4
:owl	RDF	4
application/x-nquads	RDF	4
application/x-ntriples	RDF	4
rdf/turtle	RDF	4
text/n3	RDF	4
gzip::nquads	RDF	4
application/n-triples	RDF	4
n-triples	RDF	4
owl	RDF	4
n3	RDF	4
rdf-turtle	RDF	4
linked data	RDF	4
rdf	RDF	4
meta/owl	RDF	4
application/x-trig	RDF	4
mapping/twc-conversion	RDF	4
mapping/owl	RDF	4
ontology	RDF	4
meta/void	RDF	4
ttl	RDF	4
turtle	RDF	4
nt	RDF	4
xml json rdf	RDF	4
rdf sparql+xml	RDF	4
rdn-n3	RDF	4
/owl	RDF	4
:nt	RDF	4
rdf-xml	RDF	4
rdf/dcat	RDF	4
rdf-n3	RDF	4
rdf/nt	RDF	4
rdf/n3	RDF	4
meta/rdf-schema	RDF	4
api/linked-data	RDF	4
rdf:products:org:openfoodfacts	RDF	4

expr	format	star
application/%trig	RDF	4
application/rdf_xml	RDF	4
rdfa	RDF	4
rdf xml	RDF	4
rdf/void	RDF	4
text/rdf+n3	RDF	4
rdfs	RDF	4
gzip:ntriples	RDF	4
application/xml+rdf	RDF	4
ntriples	RDF	4
nquads	RDF	4
application/x-turtle	RDF	4
rdf/xml turtle html	RDF	4
application/trix	RDF	4
rdf/xml html json	RDF	4
gz:nt	RDF	4
data file in excel and rdf	RDF	4
text/rdf+ttl	RDF	4
trig	RDF	4
gzip:text/turtle	RDF	4
rdf csv xml	RDF	4
application/turtle	RDF	4
rdfxml	RDF	4
application/rdfs	RDF	4
gz:ttl:owl	RDF	4
html rdf	RDF	4
rdf (gzipped)	RDF	4
application/x-pdf	RDF	4
json ld	RDF	4
rdf/provenance	RDF	4
rdf owl	RDF	4
xml/atom/rss	RSS	3
application/atom+xml	RSS	3
application/xml+atom	RSS	3
application/rss+xml	RSS	3
rss1.0	RSS	3
georss	RSS	3
sesame repository	Repository	
datapkg/%	Repository	
git	Repository	
api/git	Repository	
cvs	Repository	

expr	format	star
torrent	Repository	
oai-pmh	Repository	
index/ftp	Repository	
rdf endpoint	SPARQL	4
sparql-xml	SPARQL	4
sparql-json	SPARQL	4
sparql	SPARQL	4
api/sparql	SPARQL	4
sparql endpoint	SPARQL	4
text/sql	Script	1
python source code	Script	1
javascript	Script	1
python	Script	1
python shell script	Script	1
python script on wiki	Script	1
sql	Script	1
src-hg/python	Script	1
text/javascript	Script	1
application/x-javascript	Script	1
js	Script	1
py	Script	1
python script	Script	1
.mif .mid	Sound	1
format-xls	Spreadsheet	2
data file in excel and stata	Spreadsheet	2
application/x-excel	Spreadsheet	2
ms excel csv	Spreadsheet	2
application/vnd.ms-excel.sheet.binary.macroenabled.12	Spreadsheet	2
application/zip+vnd.ms-excel	Spreadsheet	2
application/zip+application/vnd.ms-excel	Spreadsheet	2
gdocs	Spreadsheet	2
googledoc	Spreadsheet	2
services/gdocs/spreadsheet	Spreadsheet	2
googlespreadsheet	Spreadsheet	2
gdocs/table	Spreadsheet	2
gdocs/ccc	Spreadsheet	2
gdocs/spreadsheet	Spreadsheet	2
google spreadsheet	Spreadsheet	2
spreadsheet	Spreadsheet	2
zip:xls	Spreadsheet	2
.xlxs	Spreadsheet	2
xlsx	Spreadsheet	2

expr	format	star
xls	Spreadsheet	2
microsoft excel	Spreadsheet	2
xslx	Spreadsheet	2
csv xls openoffice pdf mm	Spreadsheet	2
data file in excel	Spreadsheet	2
application/vnd.ms-excel	Spreadsheet	2
excel	Spreadsheet	2
xls csv	Spreadsheet	2
csv xls prn dbase med flere	Spreadsheet	2
xls html ascii	Spreadsheet	2
html xls pdf	Spreadsheet	2
html xls	Spreadsheet	2
xls html pdf	Spreadsheet	2
xls (zip)	Spreadsheet	2
csv and xls	Spreadsheet	2
csv xls m.fl.	Spreadsheet	2
pdf / xls	Spreadsheet	2
data file in stata and excel	Spreadsheet	2
application/ods	Spreadsheet	3
application/x-vnd.oasis.opendocument.spreadsheet	Spreadsheet	3
openoffice_calc	Spreadsheet	3
ods	Spreadsheet	3
application/vnd.oasis.opendocument.spreadsheet	Spreadsheet	3
application/vnd.openxmlformats-officedocument.spreadsheetml.sheet	Spreadsheet	3
openxmlformats_spreadsheetml	Spreadsheet	3
text/ascii	TXT	1
gz:txt	TXT	1
TEXT	TXT	1
gzip+txt	TXT	1
txt	TXT	1
plain text	TXT	1
zip:txt	TXT	1
text/plain	TXT	1
txt:r	TXT	1
ascii	TXT	1
txt (zipped)	TXT	1
jsp	URL	1
url	URL	1
uri	URL	1
asp	URL	
aspx	URL	
web page	URL	

expr	format	star
php	URL	
website	URL	
web site	URL	
xml (current country)	XML	3
meta/sitemap	XML	3
xml (tomorrow metro)	XML	3
atom	XML	3
rss+xml	XML	3
xml (zipado)	XML	3
example/xhtml+xml	XML	3
rss	XML	3
atom_xml	XML	3
:xml	XML	3
xml+gzip	XML	3
application/unixref+xml	XML	3
gz:xml	XML	3
tgz:xml	XML	3
html xml xml+xslt	XML	3
xml:txt	XML	3
api/opensearch	XML	3
xml (tomorrow country)	XML	3
xml (current)	XML	3
xml (tomorrow)	XML	3
bz2:xml	XML	3
xml / jpg	XML	3
wsdl	XML	3
xml (zipped)	XML	3
xml_atom	XML	3
yaml	XML	3
application/wsdl+xml	XML	3
json or xml	XML	3
XSD	XML	3
xml	XML	3
xsl	XML	3
xsd	XML	3
xml (current metro)	XML	3
various	n/a	
-	n/a	
other	n/a	
n/a	n/a	

A.3 CKANstats Metadata Analysis Results

CKANstats is a set of scripts to analyse CKAN-based data repositories. These are augmented by LODprobe, a Java tool that does a further in-depth analysis of individual RDF-based datasets. In the following, this collection of tools is applied to datahub.io.

Setup

1. Retrieve metadata from CKAN-based data repositories

You'll need to edit the source to change the data portal `CKANstats.py`

2. Load retrieved CSV into SQL (here: PostgreSQL)

```
CREATE TABLE datahubio
(
 dataset_id character(36),
 dataset_name character(1024),
 dataset_license_title character(2014),
 dataset_license_id character(255),
 dataset_is_open boolean,
 dataset_tracking_summary_total integer,
 dataset_tracking_summary_recent integer,
 resource_id character(36),
 resource_name character(1024),
 resource_created character(26),
 resource_revision_timestamp character(26),
 resource_format character(1024),
 resource_url character(1024),
 resource_tracking_summary_total integer,
 resource_tracking_summary_recent integer
)

COPY datahubio FROM 'datahubio.csv' DELIMITER ',' CSV;
```

Listing A.2: SQL Code to create a table for extracted CKAN-results.

3. Start querying :)

First impressions of datahub.io

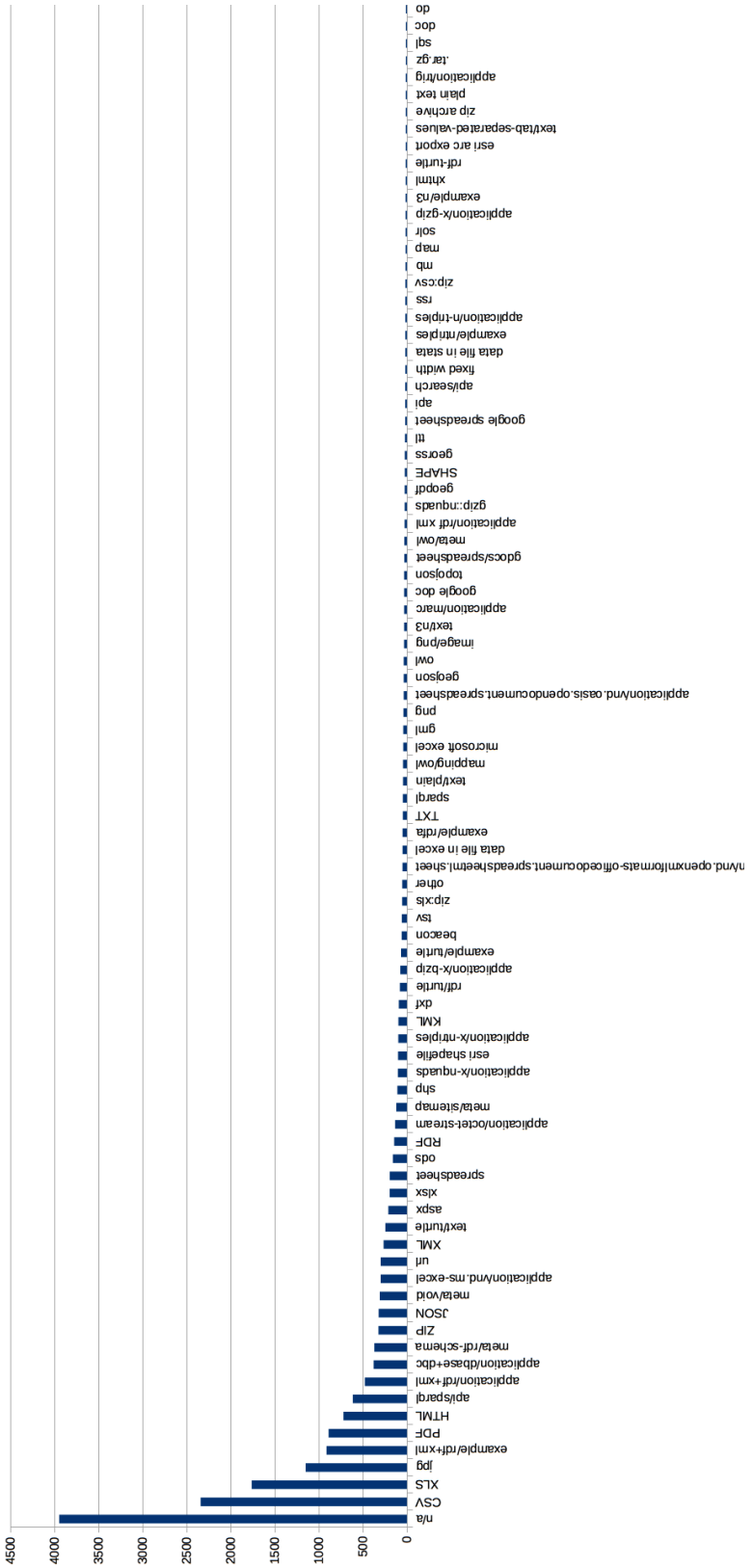
“format” row is not unified, about 20% w/o definition

```
select trim(resource_format), (COUNT(resource_format)::double precision / (select COUNT(resource_format) from datahubio)::double precision * 100) as c
      from datahubio
group by resource_format order by c desc
```

Listing A.3: SQL Query to show the different of the format field.

```
select trim(resource_format), COUNT(resource_format) as count from datahubio
group by resource_format order by count desc
```

Listing A.4: SQL Query to aggregate data formats.



There are at least 29 variants for the format "Excel"

```
select trim(resource_format), COUNT(resource_format) as count from datahubio
where resource_format like '%xls%' or resource_format like '%excel%' or resource_format like '%
    openxml%.spreadsheet%'
group by resource_format order by count desc
```

Listing A.5: SQL Query to show the different Variations of Excel Formats.

resource_format	count
application/vnd.ms-excel	298
xlsx	196
zip:xls	54
application/vnd.openxmlformats-officedocument.spreadsheetml.sheet	50
data file in excel	49
microsoft excel	41
application/zip+application/vnd.ms-excel	10
excel	10
application/zip+vnd.ms-excel	6
format-xls	6
xls (zip)	5
application/vnd.ms-excel.sheet.binary.macroenabled.12	4
ms excel csv	2
application/x-excel	2
csv stata excel	2
xls html pdf	2
data file in stata and excel	1
html xls	1
csv xls ods pdf mm	1
data file in excel and rdf	1
data file in excel and stata	1
csv and xls	1
csv xls m.fl.	1
html xls pdf	1
csv xls prn dbase med flere	1
xls csv	1

resource_format	count
pdf / xls	1
xls html ascii	1
csv xls openoffice pdf mm	1

The number of resources per datasets strongly varies

```
select AVG(count), STDDEV(count), variance(count) from (
  select trim(dataset_name), COUNT(resource_id) as count from datahubio
  group by dataset_name order by count desc
) as l
```

Listing A.6: SQL Query for the Average Resources per Dataset.

avg	stddev	variance
3.2128405289150868	13.2269114412110874	174.9511862736407655

Majority of the datasets have exactly one resource file associated

```
select count, count(count) from (
  select trim(dataset_name), COUNT(resource_id) as    count from datahubio
  group by dataset_name order by count desc
) as l group by count order by l.count
```

Listing A.7: SQL Query to Count by the Number Resources.

Some of the 'n/a' formats might be retrieved from the resource_url

```
select format, count(format) from (
  select UPPER(substring(trim(resource_url) from '...$')) as format    from datahubio where
     resource_format = 'n/a'
) as i
group by format order by count(format) desc
```

Listing A.8: SQL query to extract the last three characters from a resource URL.

format	count
PDF	747
RG/	295
CES	250
KEY	247
TML	158
PHP	53
CSV	51

format	count
ZIP	49
HTM	47
COM	34
OM/	30
SPX	30
TA/	26
D=0	23
OV/	21
JSP	19
ATA	18
.GZ	15

302 of the 'n/a'-format resources are accompanied by non-'n/a'-format resources

```
select count(id) from (
  select a.id
  from
  (select distinct(trim(dataset_id)) as id from datahubio where resource_format = 'n/a' ) as a
     inner join
  (select distinct(trim(dataset_id)) as id from datahubio where resource_format <> 'n/a') as b on
     a.id = b.id
) as i
```

Listing A.9: SQL query to compare the resource formats of datasets.

count: 302

```
drop view datahubio2;;
create view datahubio2 as
select dataset_id, dataset_name, dataset_license_title, dataset_license_id,
  dataset_is_open, dataset_tracking_summary_total, dataset_tracking_summary_recent,
  resource_id, resource_name, resource_created, resource_revision_timestamp,
  resource_format, resource_url, resource_tracking_summary_total,
  resource_tracking_summary_recent, trim(coalesce(b.format, a.format, 'n/a')) as unified_format,
     trim(coalesce(b.star, a.star, 'n/a')) as star from datahubio
left outer join mptbl as a on lower(trim(resource_format)) like lower(a.expr)
-- desperate mode: try to map unknown formats via possible resource_url-extensions
left outer join mptbl as b on (a.format = 'n/a' and lower(substring(trim(resource_url) from '...$'
     )) like b.expr)
```

Listing A.10: SQL query to map format definitions with manually created format mapping table.

Format distribution

```
select unified_format, count(unified_format) as count, count(unified_format)::double precision / (
     select count(unified_format) from datahubio2)::double precision from datahubio2
group by unified_format order by count(unified_format) desc
```

Listing A.11: SQL query to count the number of resources per format.

unified_format	count	%
n/a	2983	14.78%
Spreadsheet	2983	14.78%
CSV	2533	12.55%
RDF	2279	11.29%
PDF	1643	8.14%
Image	1371	6.79%
Example	1079	5.35%
HTML	985	4.88%
SPARQL	682	3.38%
URL	627	3.11%
Archive	534	2.65%
Geo	511	2.53%
Database	481	2.38%
XML	463	2.29%
JSON	364	1.80%
Binary	137	0.68%
TXT	132	0.65%
DOC	69	0.34%
API	63	0.31%
Beacon	59	0.29%
MARC	51	0.25%
RSS	47	0.23%
Script	44	0.22%
Repository	28	0.14%
Map	25	0.12%
GTFS	3	0.01%
Sound	2	0.01%

About openness

```
select unified_format, open, nonopen, round((open::double precision / (open::double precision +
    nonopen::double precision) * 100)::numeric, 2) from (
select t.unified_format, coalesce(t.count, 0) as open, coalesce(f.count, 0) as nonopen from
```

```
(select unified_format, count(unified_format) as count from datahubio2
where dataset_is_open = true group by unified_format ) as t
left outer join
(select unified_format, count(unified_format) as count from datahubio2
where dataset_is_open = false group by unified_format ) as f on (f.unified_format = t.
    unified_format)
) as i order by (open::double precision / (open::double precision + nonopen::double precision))
    desc
```

Listing A.12: SQL query to calculate the number of openly-licensed resources per format.

unified_format	open	non-open	% openness
MARC	51	0	100.00
GTFS	3	0	100.00
Beacon	58	1	98.31
Image	1227	144	89.50
URL	535	92	85.33
RDF	1725	554	75.69
RSS	34	13	72.34
JSON	259	105	71.15
SPARQL	472	210	69.21
API	43	20	68.25
Map	17	8	68.00
Binary	87	50	63.50
Geo	313	198	61.25
XML	273	190	58.96
n/a	1758	1225	58.93
CSV	1490	1043	58.82
Repository	16	12	57.14
Archive	292	242	54.68
Example	541	538	50.14
Script	22	22	50.00
DOC	34	35	49.28
TXT	57	75	43.18
HTML	398	587	40.41
Spreadsheet	640	2343	21.45
PDF	224	1419	13.63

unified_format	open	non-open	% openness
Database	43	438	8.94

Data Classified in TBL's 5-Star Maturity Model

```
select star, count(star) as count, count(star)::double precision / (select count(star)
from datahubio2 where dataset_is_open = true)::double precision from datahubio2 where
    dataset_is_open = true
group by star order by star
```

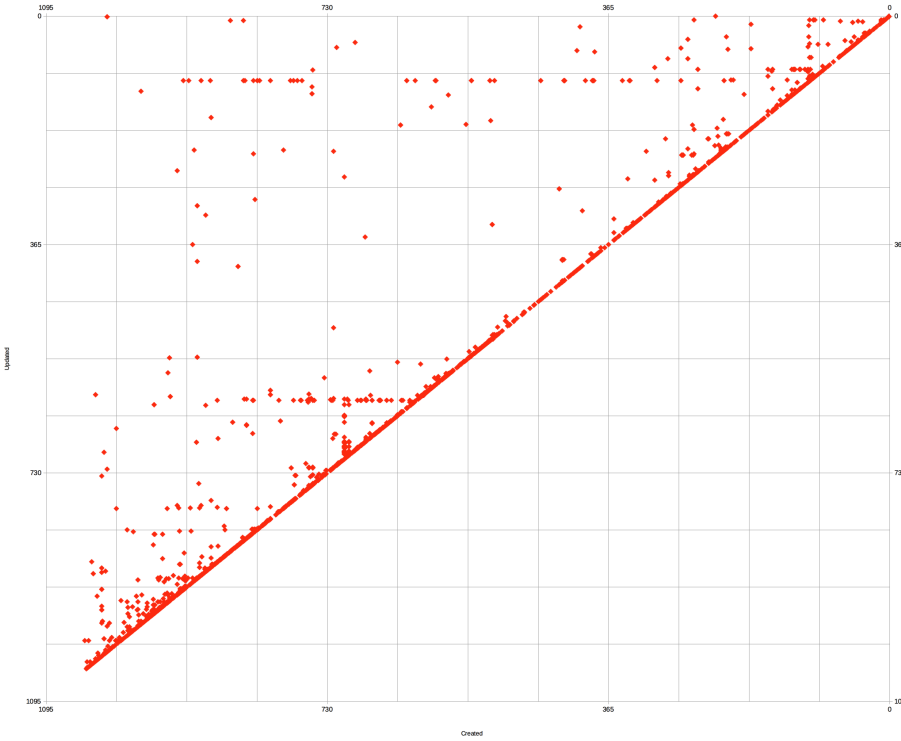
Listing A.13: SQL query to classify the data using Tim Berners-Lee's 5-Star model (Section 2.1.3 on page 50).

5-star rating	count	percent
1	2702	25,46%
2	658	6,20%
3	2506	23,61%
4	2197	20,70%
n/a	2549	24,02%

Created vs. Timestamp, Ages in days

```
select resource_id, round(EXTRACT('epoch' FROM created)/86400) created, round(EXTRACT('epoch' from a.updated)/86400) as updated from (
select datahubio2.resource_id,
  age(to_timestamp('2015-03-10 13:46:00.000000', 'YYYY-MM-DD HH24:MI:SS.US'),
    to_timestamp(resource_created, 'YYYY-MM-DD HH24:MI:SS.US')) as created,
  age(to_timestamp('2015-03-10 13:46:00.000000', 'YYYY-MM-DD HH24:MI:SS.US'),
    to_timestamp(resource_revision_timestamp, 'YYYY-MM-DD HH24:MI:SS.US')) as updated
  from datahubio2
where datahubio2.resource_created <> 'n/a'
) as a order by created
```

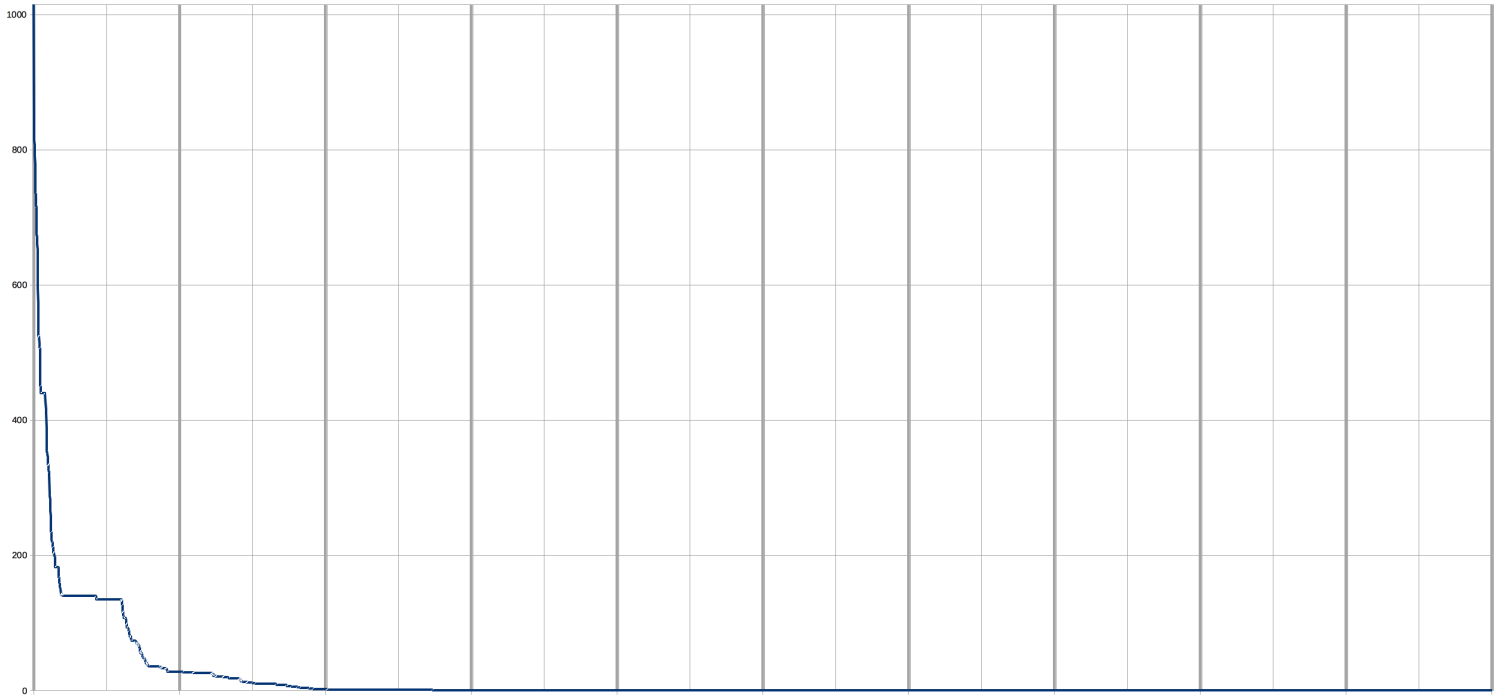
Listing A.14: SQL query to compare created and updated timestamps.


1095
730
365
0
Updated
Created

Last update after release: 10% within ~50 days, ~7% after ~150 days or more, 80% is never updated

```
select resource_id, (round(EXTRACT('epoch' FROM created)/86400) - round(EXTRACT('epoch' from a.updated)/86400)) diff from (
select datahubio2.resource_id,
  age(to_timestamp('2015-03-10 13:46:00.000000', 'YYYY-MM-DD HH24:MI:SS.US'),
    to_timestamp(resource_created, 'YYYY-MM-DD HH24:MI:SS.US')) as created,
  age(to_timestamp('2015-03-10 13:46:00.000000', 'YYYY-MM-DD HH24:MI:SS.US'),
    to_timestamp(resource_revision_timestamp, 'YYYY-MM-DD HH24:MI:SS.US')) as updated
  from datahubio2
where datahubio2.resource_created <> 'n/a'
) as a order by (round(EXTRACT('epoch' FROM created)/86400) - round(EXTRACT('epoch' from a.updated)/86400)) desc
```

Listing A.15: SQL query to compare the number of days after which resource metadata was last updated.



Average Days between Created and Last Update, per format

```
select unified_format,
  round(avg(round(EXTRACT('epoch' FROM created)/86400) - round(EXTRACT('epoch' from a.updated)
     /86400))::numeric, 2) average_created_updated_diff,
  round(stddev(round(EXTRACT('epoch' FROM created)/86400) - round(EXTRACT('epoch' from a.updated)
     /86400))::numeric, 2) stdev_created_updated_diff
from (
select datahubio2.unified_format,
  age(to_timestamp('2015-03-10 13:46:00.000000', 'YYYY-MM-DD HH24:MI:SS.US'),
    to_timestamp(resource_created, 'YYYY-MM-DD HH24:MI:SS.US')) as created,
  age(to_timestamp('2015-03-10 13:46:00.000000', 'YYYY-MM-DD HH24:MI:SS.US'),
    to_timestamp(resource_revision_timestamp, 'YYYY-MM-DD HH24:MI:SS.US')) as updated
  from datahubio2
where datahubio2.resource_created <> 'n/a'
) as a group by unified_format order by average_created_updated_diff desc
```

Listing A.16: SQL query to compare the number of days after which resource metadata was last updated, per format.

unified_format	average_created_updated_diff	stdev_created_updated_diff
Image	81.89	65.72
Archive	56.63	123.91
API	52.76	154.96
SPARQL	40.38	118.51
RSS	28.80	14.50
TXT	28.37	82.42
Example	21.97	98.39
HTML	18.33	105.33
n/a	18.29	87.06
RDF	16.78	78.40
XML	16.32	86.99
PDF	13.16	31.03
Beacon	12.00	7.07
JSON	10.18	32.91
Binary	9.98	61.66
Map	9.82	32.56
CSV	4.58	36.22
Script	2.67	2.58
Spreadsheet	1.97	25.03
DOC	1.58	2.95
Geo	1.09	9.38
URL	0.76	13.60

unified_format	average_created_updated_diff	stdev_created_updated_diff
SQL	0.43	0.53
Database	0.30	2.59
Google Doc	0.00	
MARC	0.00	0.00
Repository	0.00	0.00
GTFS	0.00	0.00

Recent & total visits per format

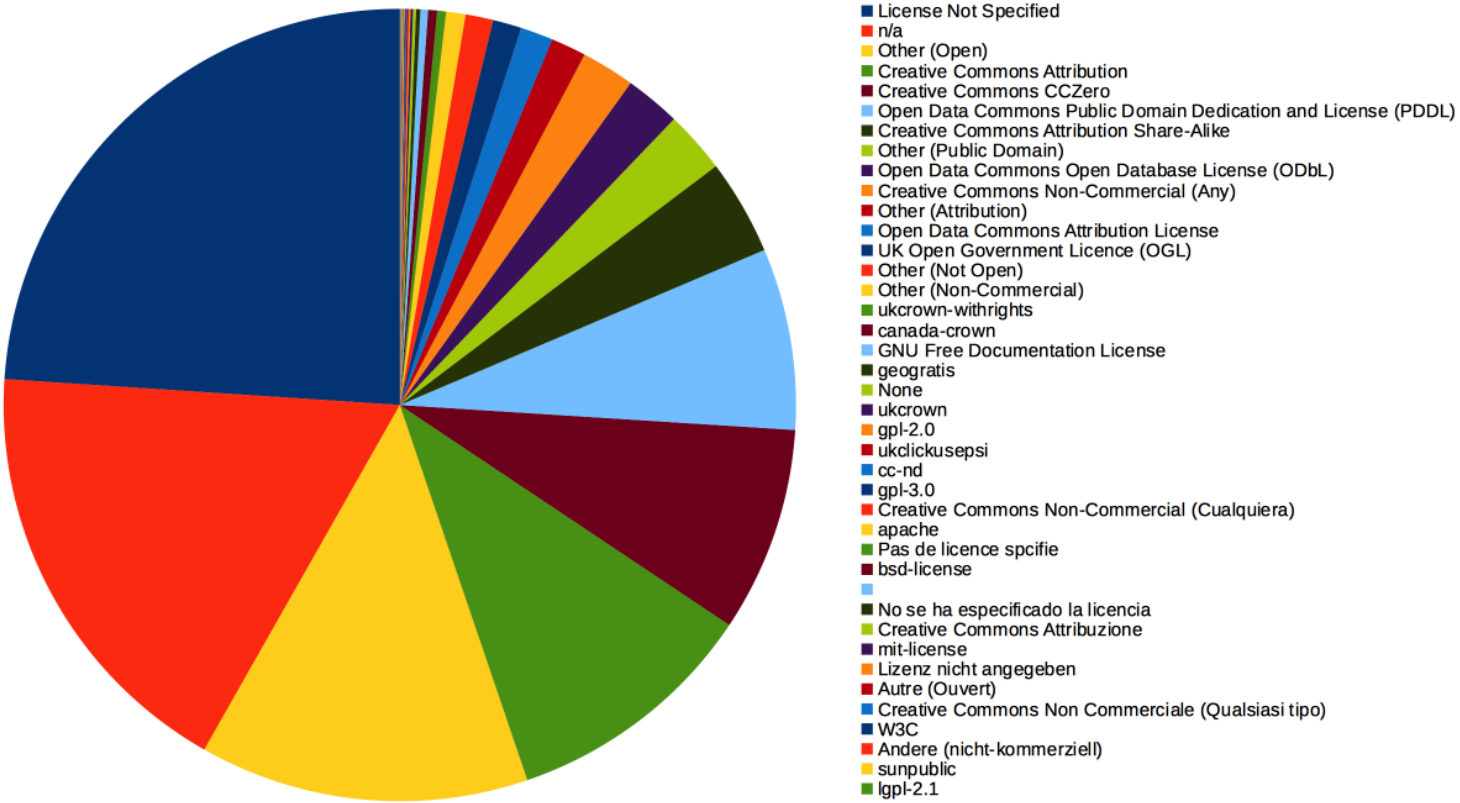
```
select a.unified_format, a.count as recent, b.count as total from (
select unified_format, sum(resource_tracking_summary_recent) as count from datahubio2
where resource_tracking_summary_recent > 0
group by unified_format order by sum(resource_tracking_summary_recent) desc
) as a left outer join (
select unified_format, sum(resource_tracking_summary_total) as count from datahubio2
where resource_tracking_summary_total > 0
group by unified_format order by sum(resource_tracking_summary_total) desc ) as b
on a.unified_format = b.unified_format
order by total desc
```

Listing A.17: SQL Query to compare popularity per format.

Licences

```
select trim(dataset_license_title), count(dataset_license_title) from datahubio2
group by dataset_license_title order by count(dataset_license_title) desc
```

Listing A.18: SQL query to aggregate the various license statements.



Per-Resource Evaluations

Top 50 Linked Data resources

```
select distinct trim(dataset_name), trim(resource_name),trim(resource_url),
     resource_tracking_summary_total from datahubio2
where unified_format = 'Linked Data' order by resource_tracking_summary_total desc
```

Listing A.19: SQL query to list the most popular Linked Data resources.

A.4 LODprobe

```
package com.github.heussd.lodprobe;

import java.io.File;
import java.util.logging.Logger;

import org.apache.commons.io.FileUtils;

import com.hp.hpl.jena.query.Query;
import com.hp.hpl.jena.query.QueryExecution;
import com.hp.hpl.jena.query.QueryExecutionFactory;
import com.hp.hpl.jena.query.QueryFactory;
import com.hp.hpl.jena.query.QuerySolution;
import com.hp.hpl.jena.query.ResultSet;
import com.hp.hpl.jena.query.Syntax;
import com.hp.hpl.jena.rdf.model.Resource;

public class Main {
 private final static Logger LOGGER = Logger.getLogger("lodprobe");

 private final static Query QUERY_COUNT_UNIQUE_SUBJECTS = QueryFactory.create("SELECT COUNT(
      distinct ?s) { ?s ?p ?o .}", Syntax.syntaxARQ);
 private final static Query QUERY_COUNT_PROPERTIES = QueryFactory.create("SELECT ?p (COUNT(?p) as
      ?pCount) { ?s ?p ?o . } GROUP BY ?p ORDER BY ?p", Syntax.syntaxARQ);

 public static void main(String[] args) throws Exception {
  LOGGER.info("T.H. LODprobe");

  System.out.print("Enter target file name (.csv will be suffixed):");
//  String csvfile = System.console().readLine();
  String csvfile = null;

  System.out.print("Enter dataset name on local fuseki [DB]:");
//  String datasetId = System.console().readLine();
  String datasetId = "http://lod2.openlinksw.com/sparql/";

  if (datasetId.trim().isEmpty())
   datasetId = "DB";

  // String datasetId = "people";

  LOGGER.info("Probing \"" + datasetId + "\"...");
  int subjects = getNumberOfUniqueSubjects(datasetId);

  LOGGER.info(" Number of unique subjects: " + subjects);

  // / ########################################
  PropertyMatrix propertyMatrix = new PropertyMatrix();

  QueryExecution queryExecution = QueryExecutionFactory.sparqlService(getServiceUrl(datasetId),
       QUERY_COUNT_PROPERTIES);
  ResultSet results = queryExecution.execSelect();

  // System.out.println(ResultSetFormatter.asText(results));
  while (results.hasNext()) {
   QuerySolution row = results.next();
   Resource resource = row.getResource("p");
   String count = row.getLiteral("pCount").getString();
   LOGGER.info(" found property " + resource.toString());
   propertyMatrix.add(resource, count);

  }
  LOGGER.info(propertyMatrix.size() + " properties in total");
  String prefixes = getPrefixes(propertyMatrix);

  // System.out.print("Enter dataset name on local fuseki:");
  // datasetId = System.console().readLine();
  for (int i = 0; i < propertyMatrix.size(); i++) {
   Resource x = propertyMatrix.get(i);

   for (int j = 0; j < propertyMatrix.size(); j++) {
    if (j > i) {
     Resource y = propertyMatrix.get(j);

     LOGGER.info(x.getLocalName() + " vs. " + y.getLocalName());
```

```
    String result = "";

    try {
     QueryExecution exec = QueryExecutionFactory.sparqlService(getServiceUrl(datasetId),
          createVsQuery(x, y, prefixes));
     ResultSet r = exec.execSelect();

     QuerySolution row = r.next();
     result = row.getLiteral(".1").getString();
    } catch (Exception e) {
     e.printStackTrace();
     result = "E";
    } finally {
     queryExecution.close();
    }
    propertyMatrix.put(i, j, result);

   } else {
    propertyMatrix.put(i, j, "-");
   }

  }
 }

 System.out.println("\n\n");
 String output = "Number of unique subjects: " + subjects + propertyMatrix;
 System.out.println(output);

 FileUtils.writeStringToFile(new File("lodprobe/" + csvfile + ".csv"), output);
}

private static String getServiceUrl(String datasetId) {
 if(datasetId.startsWith("http")) {
  return datasetId;
 } else {
  // This is just a dataset Id for the local fuseki
  return "http://localhost:3030/" + datasetId + "/query";
 }
}

private static int getNumberOfUniqueSubjects(String datasetId) {
 QueryExecution queryExecution = QueryExecutionFactory.sparqlService(getServiceUrl(datasetId),
      QUERY_COUNT_UNIQUE_SUBJECTS);

 ResultSet results = queryExecution.execSelect();

 QuerySolution row = results.next();
 int numberOfUniqueSubjects = 0;
 try {
  numberOfUniqueSubjects = new Integer(row.getLiteral(".1").getString());
 } catch (Exception e) {
  throw new RuntimeException("Cannot retrieve number of unique subjects", e);
 } finally {
  queryExecution.close();
 }
 // System.out.println(label.getString());

 // System.out.println(ResultSetFormatter.asText(results));
 //
 // ResultSetFormatter.out(System.out, results);
 return numberOfUniqueSubjects;
}

private static Query createVsQuery(Resource x, Resource y, String prefixes) {
 String query = prefixes;
 query += "select COUNT(distinct ?s) { ?s " + getNamespace(x) + ":" + escapeProperty(x.
      getLocalName()) + " ?o1 . ?s " + getNamespace(y) + ":" + escapeProperty(y.getLocalName()) +
       " ?o2 . }";

 return QueryFactory.create(query, Syntax.syntaxARQ);
}

private static String getNamespace(Resource resource) {
 return "NS" + Math.abs(resource.getNameSpace().toString().hashCode());
}
```

```
 private static String getPrefixes(PropertyMatrix propertyMatrix) {
  String prefixes = "";
  for (int i = 0; i < propertyMatrix.size(); i++) {
   Resource resource = propertyMatrix.get(i);
   prefixes += "\nPREFIX " + getNamespace(resource) + ": <" + resource.getNameSpace() + ">";
  }
  return prefixes;
 }

 private static String escapeProperty(final String property) {
  String escapedProperty = property.replaceAll("\\.", "\\\\.");
  return escapedProperty;
 }
}
```

Listing A.20: Source Code of LODprobe.

A.5 fromlodproberes.R

```
fromlodproberes <- function(x) {
 m <- read.csv(sep=",",x)
 # Interpret LODprobe output
 row.names(m) <- m[,1]
 # Remember count col
 counts <- m[2:2]
 counts <- as.matrix(counts)
 # Cut off meta
 m <- m[3:length(m)]
 # names(m) <- row.names(m)
 names(m) <- 0:(ncol(m)-1)
 #m[,1] <- 0
 #m[as.matrix(m)=='-'] <- 0
 #m[as.matrix(m)=='E'] <- 0
 # Convert to numeric matrix
 m <- as.matrix(m)
 # Write remembered counts into diagonale
 m[row(m) == col(m)] <- counts[row(m)]

 mode(m) <- "numeric"
 require("gdata")
 lowerTriangle(m) <- lowerTriangle(t(as.matrix(m)))

 #m[is.na(m)] <- 0

 return (as.matrix(m))
}
```

Listing A.21: Source Code of the R script fromlodproberes.R.

A.6 writeclustprops.R

```
writeclustprops <- function(x) {
 m <- read.csv(sep=",",x)
 totalsubjects <- sub(".*subjects..","",names(m[1:1]))

 d <- daisy(fromlodproberes(x), metric="gower")
 maxh <- max(d)
 minh <- min(d)
 totalproperties <- attr(d, "Size")
 if (totalproperties >= 2) {
  # cluster analysis does only work with n >= 2
 clustersh01 <- max(cutree(hclust(d, method="complete"), h=0.1))
 clustersh02 <- max(cutree(hclust(d, method="complete"), h=0.2))
 clustersh03 <- max(cutree(hclust(d, method="complete"), h=0.3))
 clustersh04 <- max(cutree(hclust(d, method="complete"), h=0.4))
 } else {
  clustersh01 <- clustersh02 <- clustersh03 <- clustersh04 <- 0
 }
 write.table(col.names=FALSE, matrix(c(x, totalsubjects, minh, maxh, totalproperties,clustersh01,
      clustersh02,clustersh03,clustersh04),ncol=9),append=TRUE, "../clusteranalysis.csv", sep=",",
       row.names=F)
}
```

Listing A.22: Source Code of the R script writeclustprops.R.

A.7 writepropertycounts.R

```
writepropertycounts <- function(x) {
 m <- read.csv(sep=",",x)
 counts <- m[1:2]
 counts <- as.matrix(counts)
 row.names(counts) <- sub(".* ", "", m[,1])
 countsdf <- as.data.frame(counts)[2]
 countsdf[2] <- x
 write.table(sep=",",col.names=FALSE, countsdf,append=TRUE, "../counts.csv")
}
```

Listing A.23: Source Code of the R script writepropertycounts.R.

A.8 writepng.R

```
library("cluster")

fromlodproberes <- function(x) {
  m <- read.csv(sep=",",x)
  # Interpret LODprobe output
  row.names(m) <- m[,1]
  # Remember count col
  counts <- m[2:2]
  counts <- as.matrix(counts)
  # Cut off meta
  m <- m[3:length(m)]
  # names(m) <- row.names(m)
  names(m) <- 0:(ncol(m)-1)
  #m[,1] <- 0
  #m[as.matrix(m)=='-'] <- 0
  #m[as.matrix(m)=='E'] <- 0
  # Convert to numeric matrix
  m <- as.matrix(m)
  # Write remembered counts into diagonale
  m[row(m) == col(m)] <- counts[row(m)]

  mode(m) <- "numeric"
  require("gdata")
  lowerTriangle(m) <- lowerTriangle(t(as.matrix(m)))

  #m[is.na(m)] <- 0

  return (as.matrix(m))
}

writepng <- function(x) {
  require(graphics)
 png(width = 15, height = 10, units = 'in', res = 300, paste("../png/", x, "_dendrogram",".png",
      sep = ""))
 plot(hclust(daisy(fromlodproberes(x), metric="gower"), method="complete"), cex=0.5)
 dev.off()

  png(width = 15, height = 10, units = 'in', res = 300, paste("../png/", x, "_dendrogram_h02",".
     png", sep = ""))
  hc <- hclust(daisy(fromlodproberes(x), metric="gower"), method="complete")
  plot(hclust(daisy(fromlodproberes(x), metric="gower"), method="complete"), cex=0.5)
  rect.hclust(hc, h=0.2)
  dev.off()

  png(width = 15, height = 10, units = 'in', res = 300, paste("../png/", x, "_dendrogram_h02",".
     png", sep = ""))
  hc <- hclust(daisy(fromlodproberes(x), metric="gower"), method="complete")
  plot(hclust(daisy(fromlodproberes(x), metric="gower"), method="complete"), cex=0.5)
  rect.hclust(hc, h=0.2)
  dev.off()
}
```

Listing A.24: Source Code of the R script writepng.R.

A.9 Cluster Analysis Results

Resource	Number of subjects	Number of properties	Groups at heights			
			h=0.1	h=0.2	h=0.3	h=0.4
lod-it_scuole-italiane.csv	3	14	4	3	3	2
92039_Ag_EU_TEL_a0005_Portugal.nt.csv	56986	38	11	7	6	4
DBpedia_3.5.1 (FBench).csv	9492364	26	12	6	5	4
DumpEarthRDF.csv	14362	16	8	4	3	2
Global-Hunger-Index-2011.rdf.csv	2102	32	9	5	5	4
Gnoss-Ontology.csv	9	7	5	5	5	4
NOMGEO_provincias.csv	167	9	6	5	5	3
Rameau.csv	154981	19	12	6	4	4
... human-imprintome-genes_preditos.csv	476	35	12	6	5	4
a-draft-version-of-the-linked-human-imprintome.csv	460	36	10	6	4	3
aemet.csv	394289	23	5	4	4	4
agris-void.csv	8	21	6	4	4	4
agrovoc-skos.csv	15005	4	4	4	4	4
alexandre_dumas_la_reine_margot-rdf.nt.csv	86	20	4	4	4	2
... database-in-life-science-void.csv	1	11	1	1	1	1
b3kat.csv	9	14	5	3	3	3
bibsonomy-swrc.csv	35	26	9	6	4	3
bibsonomy.csv	31	33	10	6	5	3
bioimages.csv	4	19	4	3	3	3
biolit-void.csv	8	24	8	7	5	3
bluk-bnb.csv	15	20	3	3	2	2
business.data.gov.uk.csv	597	24	9	5	4	2
caloi-caloifinal.csv	89	32	12	6	5	3
cap-italy-rdf.csv	1	12	1	1	1	1

Resource	Number of subjects	Number of properties	Groups at heights			
			h=0.1	h=0.2	h=0.3	h=0.4
capgrids.csv	7532	7	4	3	3	3
data.nytimes.com-organizations.csv	6088	20	4	4	3	3
datos-bcn-cl.csv	186	24	11	7	5	2
datos-bne-es-links-viaf.csv	453414	1	0	0	0	0
datos-bne-es.csv	4220712	65	20	8	5	4
dbpedia-berlin.csv	4699	327	2	2	2	2
dbpedia-fr-rdf-schema.csv	1608	8	7	6	5	4
dbpedia-it-rdf-schema.csv	1608	8	7	6	5	4
dbpedia-owl.csv	1534	7	6	5	4	4
dbpedia_3.6.owl.bz2.csv	1608	8	7	6	5	4
dbtune-john-peel-sessions copy.csv	76229	25	12	7	3	2
dbtune-john-peel-sessions.csv	2087	1	0	0	0	0
dcs-sheffield.csv	1101	21	10	6	4	3
descriptors.rdf.csv	996	23	6	4	3	3
deutsche-nationalbibliografie-dnb-example.csv	2	12	3	3	2	2
deutsche-nationalbibliografie-dnb.csv	2	14	3	3	2	2
dutch-national-regulations-metalex.csv	1	10	1	1	1	1
ecs.csv	232	20	14	6	4	3
edubase-rdf-r2rc3.csv	627259	193	19	8	5	4
education-data-gov-uk.csv	11538	11	7	6	5	4
. . . sisvu-educationalProgram-dump.csv	352742	45	12	5	4	2
educationalprograms_sisvu-sisvu.csv	123	27	13	7	5	3
eea-rod.csv	544	37	12	5	4	3
english-index-of-multiple-deprivation-score-2010.csv	32482	6	1	1	1	1
environment-agency-bathing-water-quality.csv	8	11	5	3	3	3
environment-data-gov-uk.csv	26370	11	7	6	5	4

Resource	Number of subjects	Number of properties	Groups at heights			
			h=0.1	h=0.2	h=0.3	h=0.4
estat-legis.csv	1035	14	10	5	3	2
eunis-Habitat-types.csv	5560	23	8	5	3	2
eunis-void.csv	20	22	13	8	6	5
eunis.eea.europa.eu-sites.csv	124117	42	14	7	5	5
...Ag_EU_EFG_DFI.nt.bz2.csv	314292	30	7	7	5	4
...Ag_EU_EFG_FilmArchiv.csv	2100	37	8	5	5	4
...Ag_UK_ELocal_TWAM-CS.nt.csv	3605	31	9	6	6	4
...Ag_UK_ELocal_TWAM-MN.csv	6139	34	10	6	5	4
...Ag_UK_ELocal_TWAM-JL.csv	1127	31	10	7	6	4
...Ag_UK_ELocal_TWAM-DB.csv	1281	32	9	6	6	4
...Ag_UK_ELocal_TWAM-AOL.csv	8099	32	10	7	6	4
...Ag_UK_ELocal_TyneWearImagine.csv	29666	31	7	5	5	4
...Ag_ES_ELocal_esegen.nt.csv	3441998	57	21	9	7	4
...Ag_BE_Elocal_Bokrijk.csv	3375	34	8	6	5	4
...Ag_EU_TEL_a0078_TEL_NKP_kramerius.nt.csv	75943	34	11	6	6	4
...Ag_EU_TEL_a0444_BritishLibrary.csv	217623	38	9	6	5	4
...Ag_EU_TEL_a0327_Luxemburg.nt.csv	174230	33	8	7	5	4
...Ag_EU_TEL_a0478_Austria.nt.csv	106506	40	8	6	6	4
...SwedishNationalHeritage_fornvannen.nt.csv	22225	35	8	6	5	4
europeana-lod-v1-void.csv	31	15	5	4	3	3
europeana-lod-v1.csv	7161	33	8	5	5	4
europeana-sparql.csv	31	15	5	4	3	3
eurostat-countries.csv	47	12	4	4	4	4
eurostat-rdf-void.csv	8	16	8	5	4	4
eurostat-rdf.csv	996	4	4	4	4	4
euskadi-farmacias.csv	5	15	6	3	3	3

Resource	Number of subjects	Number of properties	Groups at heights			
			h=0.1	h=0.2	h=0.3	h=0.4
fao-geopolitical-ontology-owl.csv	502	99	9	4	3	3
fao-geopolitical-ontology.csv	396	70	10	4	3	3
geological-survey-of-austria-thesaurus.csv	2	17	3	3	2	2
geonames-ontology.csv	716	12	7	6	5	4
geospecies-void.csv	18	26	7	4	4	4
geospecies.csv	104757	165	22	11	6	5
gesis-thesoz-void.csv	12	26	7	3	3	2
glastonbury2011.csv	23097	23	8	4	4	2
global-hunger-index.csv	1188	10	6	6	4	4
gnoss-com-a-social-and-semantic-platform-void.csv	9	30	5	3	2	2
googleart-wrapper.csv	1570	20	7	4	3	3
government-web-integration-for-linked-data copy.csv	96	7	6	6	4	2
government-web-integration-for-linked-data.csv	5765635	79	15	7	4	3
grrp.csv	3	15	4	3	3	2
grrt-void.csv	1	11	1	1	1	1
grrt.csv	117845	19	11	7	4	3
... hebis-00000001-05051126.csv	2654724	27	9	7	5	4
... hebis-26887668-29873805.csv	1895168	30	11	5	4	3
intervals.csv	323	28	11	5	4	3
istat-immigration-void.csv	8	14	8	6	4	4
italian-public-schools-linkedopendata-it copy.csv	3	14	4	3	3	2
italian-public-schools-linkedopendata-it.csv	103310	19	10	7	6	4
iucr-acta-cryst-e.csv	777	31	5	3	2	2
jamendo-rdf.csv	335925	26	10	5	4	4
japan-radioactivity-stat-void.csv	3	16	4	3	2	2
japan-radioactivity-stat.csv	413826	7	2	2	2	2

Resource	Number of subjects	Number of properties	Groups at heights			
			h=0.1	h=0.2	h=0.3	h=0.4
jiscopenbib-bl_bnb-1.csv	3995582	38	11	6	3	2
jita-skos-core.csv	36	21	12	7	4	3
jita.csv	160	14	10	4	3	2
knoesis_linkedsensordata.csv	29271	12	4	4	4	4
latest_reegle_dump.nt.csv	4907	57	8	7	5	4
legislation.gov.uk-ukpga1985-67section6.csv	238	44	13	8	4	3
lexvo-void.csv	1	13	2	1	1	1
lexvo.csv	47	21	13	6	4	3
lingvoj.csv	510	3	1	1	1	1
linked-open-camera copy.csv	3	14	4	3	3	2
linked-open-camera.csv	910	25	10	7	5	4
linked-open-senate copy.csv	3	14	4	3	3	2
linked-open-senate.csv	329	11	4	3	3	2
linkedevents.org-ontology.csv	20	29	7	4	4	2
linkedmarkmail.csv	1	11	3	1	1	1
linkedmdb.csv	694400	222	6	4	3	2
lobi-E-MTAB-327.csv	1278	40	10	6	5	3
lobid-organisations.csv	6	15	4	4	3	3
lobid-resources-void.csv	9	20	4	4	3	2
loc-void.csv	1	13	4	1	1	1
lsoa.csv	34378	8	2	2	2	2
mesh_ipsv_skos_rdf.csv	24628	22	11	5	4	3
mincats.csv	32	16	10	7	3	2
mondial.csv	19236	55	14	8	4	3
movies-argentina-movies.csv	1055	8	6	5	4	4
movies-argentina-persons.csv	1135	14	11	6	3	3

Resource	Number of subjects	Number of properties	Groups at heights			
			h=0.1	h=0.2	h=0.3	h=0.4
museums-in-italy.csv	5	14	4	3	3	3
musicontology.csv	394	27	13	8	5	4
my-experiment-void.csv	8	18	6	4	4	3
nace.csv	996	10	3	2	2	2
national-diet-library-authorities.csv	71644	26	9	5	3	3
national-diet-library-subject-headings.csv	71347	26	9	5	3	3
newsweek.csv	1	3	3	3	3	2
nuts.csv	2009	15	4	4	4	3
ocd.csv	12	28	6	4	3	3
odc-lsoa.csv	34378	8	2	2	2	2
odc-population-2005.csv	34378	7	1	1	1	1
ogolod-void.csv	1	8	2	1	1	1
ogolod.csv	254	13	11	6	4	4
ontologi.es-place.csv	3993	15	6	5	4	3
open-data-euskadi copy.csv	5	15	6	3	3	3
open-data-euskadi-restaurantes.csv	5	15	6	3	3	3
open-data-euskadi-zonas-de-compras.csv	5	15	6	3	3	3
open-data-euskadi.csv	5	15	6	3	3	3
opencyc.csv	137843	18	12	8	4	3
passim.csv	3776	18	9	5	3	3
patents.data.gov.uk.csv	2187	16	6	5	4	3
people.csv	9958	20	3	3	3	3
pokepedia-fr.csv	35531	85	11	5	4	3
productontology.csv	4016	18	5	4	4	4
psh-subject-headings.csv	14032	27	5	4	3	2
purl.org-meducator-ns.csv	51	15	9	7	4	4

Resource	Number of subjects	Number of properties	Groups at heights			
			h=0.1	h=0.2	h=0.3	h=0.4
reference-data-gov-uk.csv	119	24	14	7	5	3
rkb-explorer-acm-void.csv	93	24	10	7	5	3
rkb-explorer-acm.csv	1579891	18	7	5	5	4
rkb-explorer-citeseer copy.csv	64	24	9	6	5	3
rkb-explorer-citeseer.csv	721329	19	6	5	4	4
rkb-explorer-cordis.csv	50	24	9	6	5	3
rkb-explorer-crm.csv	342	6	5	5	4	4
rkb-explorer-darmstadt-void.csv	8	23	9	7	5	3
rkb-explorer-dblp.csv	6013816	42	14	6	4	3
rkb-explorer-deepblue.csv	18	24	8	7	4	3
rkb-explorer-dotac.csv	56	23	8	8	5	3
rkb-explorer-ecs.csv	5	18	9	7	4	3
rkb-explorer-eprints.csv	84	24	10	7	5	3
rkb-explorer-epsrc.csv	4	18	5	4	2	2
rkb-explorer-era.csv	11	24	8	6	5	3
rkb-explorer-eurecom.csv	34	24	8	7	4	3
rkb-explorer-ft.csv	28	24	8	7	4	3
rkb-explorer-fun.csv	14	18	9	7	6	4
rkb-explorer-ibm.csv	38	24	8	7	4	3
rkb-explorer-ieee copy.csv	46	24	8	7	4	3
rkb-explorer-ieee.csv	7884	18	9	6	3	3
rkb-explorer-irit.csv	52	24	9	6	5	3
rkb-explorer-nsf.csv	30	24	8	7	4	3
rkb-explorer-os.csv	22	24	10	7	5	3
rkb-explorer-pisa.csv	38	24	8	7	4	3
rkb-explorer-rae2001.csv	67	24	10	7	5	3

Resource	Number of subjects	Number of properties	Groups at heights			
			h=0.1	h=0.2	h=0.3	h=0.4
rkb-explorer-risks.csv	31204	21	8	5	3	3
rkb-explorer-roma.csv	45	24	8	6	5	3
rkb-explorer-southampton-void.csv	110	24	10	7	5	3
rkb-explorer-southampton.csv	551	24	9	7	4	3
rkb-explorer-ulm.csv	36	23	9	7	5	4
rkb-explorer-webconf.csv	1469	21	11	6	5	4
rkb-explorer-webscience.csv	29	22	10	8	5	3
rkb-explorer-wordnet-void.csv	19	23	10	8	5	3
sec-rdfabout.csv	460446	12	6	5	5	4
semantic-xbrl-schema.csv	401	3	3	2	2	2
semantic_bible.csv	127	16	11	6	6	3
southampton-ac-uk-apps.csv	54	33	11	8	5	4
southampton-ac-uk-bus-stops.csv	1947	18	7	6	4	4
southampton-ac-uk-catering-points-of-service.csv	427	26	6	6	5	4
southampton-ac-uk-cls-extras.csv	152	26	7	6	4	4
southampton-ac-uk-disabledgo.csv	96	22	5	5	4	4
southampton-ac-uk-edshare-video.csv	4160	70	9	6	5	3
southampton-ac-uk-eprints-ecs.csv	167395	138	13	7	5	3
southampton-ac-uk-jacs copy.csv	360	25	7	6	4	4
southampton-ac-uk-jacs.csv	360	25	7	6	4	4
southampton-ac-uk-jargon.csv	154	26	7	6	5	4
southampton-ac-uk-org.csv	612	28	7	5	4	4
southampton-ac-uk-stats.csv	23915	72	11	8	3	2
southampton-ac-uk-vending-machines.csv	306	27	8	6	5	4
southampton-ecs-eprints copy.csv	2	18	4	3	3	3
southampton-ecs-eprints.csv	167390	121	11	7	4	3

Resource	Number of subjects	Number of properties	Groups at heights			
			h=0.1	h=0.2	h=0.3	h=0.4
species.csv	279515	57	18	10	5	4
statistics-data-gov-uk.csv	34959	60	15	7	4	3
stitch-rameau-LCSH.csv	55963	1	0	0	0	0
stitch-rameau-SWD.csv	20538	1	0	0	0	0
sztaki-lod.csv	1	8	3	2	1	1
tags2con-delicious copy.csv	6153	21	4	4	2	2
tags2con-delicious-bookmarks.csv	1474	1	0	0	0	0
tags2con-delicious-concepts.csv	651	1	0	0	0	0
tags2con-delicious.csv	2832	1	0	0	0	0
taxonconcept copy.csv	39781	4	4	4	4	4
taxonconcept-misc.csv	16569	55	17	8	5	4
taxonconcept-void.csv	44	26	7	5	3	3
tcga.csv	10741	85	11	4	3	3
telegraphis-continents.csv	21	12	5	4	3	3
telegraphis-countries.csv	1476	19	5	4	3	2
telegraphis-currencies.csv	533	14	5	4	3	2
telegraphis-void.csv	41	28	7	4	3	3
telegraphis.csv	467	9	3	3	3	2
temple-ov-thee-lemur-datasets copy.csv	15	12	4	4	3	3
temple-ov-thee-lemur-datasets-4inarow.csv	48	13	5	5	4	3
temple-ov-thee-lemur-datasets-cancer-causes.csv	1214	6	5	4	3	2
temple-ov-thee-lemur-datasets-planet.csv	23	13	4	4	4	3
temple-ov-thee-lemur-datasets-zodiac.csv	180	26	11	7	4	3
temple-ov-thee-lemur-datasets.csv	80	13	5	5	3	3
the-view-from.csv	5	12	4	4	4	3
totl-xkcd.csv	1519	17	5	4	3	3

Resource	Number of subjects	Number of properties	Groups at heights			
			h=0.1	h=0.2	h=0.3	h=0.4
traditional-korean-medicine.csv	11157	45	9	5	3	3
transport-data-gov-uk.csv	6134716	41	10	7	5	2
twc-healthdata.csv	55	20	9	5	4	4
twc-ieeevis.csv	55	20	9	5	4	4
txn.owl.csv	375	26	10	6	4	3
umbel-reference-concepts.csv	28627	23	10	6	5	4
umbel.csv	745	21	10	6	4	3
umthes.csv	4	17	6	3	3	2
uniref.csv	4	16	6	6	6	5
vivo-core-1.1.owl.csv	830	31	12	7	5	3
vivo-cornell-university copy.csv	1660	50	14	10	5	3
vivo-cornell-university-PamelaGraham.csv	15	20	4	4	3	2
vivo-cornell-university.csv	15	27	4	3	2	2
vivo-cu-boulder.csv	1660	50	14	10	5	3
vulnerapedia.csv	1787	86	17	9	4	3
w3c-wordnet rdfs.csv	97	15	13	7	4	4
webenemasuno.linkeddata.es:source:rdf:data.csv	1010681	65	12	7	4	3
wn20full.csv	464880	41	7	6	4	3
world-bank-linked-data copy.csv	97809	37	7	4	4	3
world-bank-linked-data.csv	269220	141	16	10	6	5
Average	217659,06	28,075	7,66	5,06	3,76	2,96
Stdev	994267,50	33,03	3,96	2,087	1,35	1,01

Appendix B

Benchmark Appendix

B.1 sqlite4java Benchmark Implementation

```
package database;

import java.io.File;
import java.util.ArrayList;

import org.apache.commons.io.FileUtils;
import org.json.JSONArray;
import org.sqlite.SQLiteConfig.SynchronousMode;

import com.almworks.sqlite4java.SQLiteConnection;
import com.almworks.sqlite4java.SQLiteJob;
import com.almworks.sqlite4java.SQLiteQueue;
import com.almworks.sqlite4java.SQLiteStatement;
import com.google.common.base.Joiner;

import util.Codes;
import util.Config;
import util.DataObject;
import util.Dataset;
import util.QueryResult;
import util.QueryResult.Type;
import util.QueryScenario;
import util.Templates;
import util.dumper.Helpers;

public class SQLite4Java extends Helpers implements Database {

 private static final int COMMIT_EVERY_N_RECORDS = Config.COMMIT_EVERY_N_RECORDS;
 private static final File DATABASE_FILE = new File("sqlite4java.db");
 private SQLiteConnection connection;
 private String createQuery;
 private String genericInsertStatement;
 private ArrayList<SQLiteStatement> scenarioStatements;
 private QueryScenario queryScenario;
 private Templates templates;
 private SQLiteQueue queue;
 private ArrayList<Dataset> lastLoadedDatasets = new ArrayList<>();
 private boolean graphStructurePrepared = false;
 private ArrayList<String> scenarioQueries = new ArrayList<>();
 private ArrayList<String> ids;
 private String queryString;

 public SQLite4Java() {
  // Disable logging of sqlite4java
  java.util.logging.Logger.getLogger("com.almworks.sqlite4java").setLevel(java.util.logging.Level
      .OFF);

  // Produce some queries based on Config / Codes enums - do not prepare
  // statements as PreparedStatements is part of the load() or prepare().
  createQuery = "CREATE TABLE " + Config.TABLE + " ( \n";
  genericInsertStatement = "insert into " + Config.TABLE + " (";
  for (int i = 0; i < Codes.values().length; i++) {
   Codes code = Codes.values()[i];
   createQuery += " " + code.toString() + " text";
   genericInsertStatement += code.toString();
   // Last one?
   createQuery += (Codes.values().length - 1 == i ? "\n)" : ",\n");
   genericInsertStatement += (Codes.values().length - 1 == i ? ")" : ",");
  }
```

```
 genericInsertStatement += "\nVALUES(";
 for (int i = 0; i < Codes.values().length; i++) {
  genericInsertStatement += "?" + (Codes.values().length - 1 == i ? ")" : ",");
 }
 templates = new Templates("sqlite", ".sql");
}

private SQLiteStatement makeInsertStatement(DataObject dataObject) throws Exception {
 SQLiteStatement insertStatement = connection.prepare(genericInsertStatement);

 for (int i = 0; i < Codes.values().length; i++) {
  Codes code = Codes.values()[i];
  if (dataObject.get(code) != null) {
   Object value = dataObject.get(code);

   if (code.IS_MULTIPLE) {
    value = new JSONArray((ArrayList) value).toString();
   }
   insertStatement.bind(i + 1, (String) value);
  }
 }
 return insertStatement;
}

@Override
public String getName() {
 return "sqlite4java";
}

@Override
public String getVersion() {
 return "392 with SQLite 3.8.7";
}

@Override
public void setUp() throws Exception {
 clean();
 reopenConnection(false);

 connection.exec(createQuery);
 lastLoadedDatasets.clear();
}

@Override
public void load(Dataset dataset) throws Exception {
 // Auto commit setting:
 // https://stackoverflow.com/questions/4998630/how-to-disable-autocommit-in-sqlite4java#5005785
 connection.exec("BEGIN");

 readRdf(dataset, dataObject -> {
  try {
   SQLiteStatement insertStatement = makeInsertStatement(dataObject);
   insertStatement.step();
   insertStatement.clearBindings();
  } catch (Exception e) {
   throw new RuntimeException("Cannot insert DataObject: " + dataObject, e);
  }
 } , counter -> {
  try {
   if (counter % COMMIT_EVERY_N_RECORDS == 0) {
    connection.exec("COMMIT");
    connection.exec("BEGIN");
   }
  } catch (Exception e) {
   throw new RuntimeException("Cannot commit at " + counter + " entities", e);
  }
 });

 connection.exec("COMMIT");
 this.lastLoadedDatasets.add(dataset);
}

private void reopenConnection(boolean readonly) {
 try {
  // if (connection != null && connection.isOpen())
  //// connection.c;
  //
```

```
  // // connection = DriverManager.getConnection("jdbc:sqlite:" +
  // // Config.DATABASE + ".db");
  if (connection == null) {
   connection = new SQLiteConnection(DATABASE_FILE);

   // if (readonly)
   // connection.openReadonly();
   // else
   connection.open(true);
  }
  queue = new SQLiteQueue(DATABASE_FILE);

  // connection.is
  // connection = sqLiteConfig.createConnection("jdbc:sqlite:" +
  // Config.DATABASE + ".db");
 } catch (Exception e) {
  throw new RuntimeException("Cannot open connection", e);
 }
}

@Override
public void prepare(QueryScenario queryScenario) throws Exception {
 // reopenConnection(queryScenario.isReadOnly);

 if (scenarioStatements == null)
  scenarioStatements = new ArrayList<>();

 scenarioStatements.clear();

 queue.start();
 queue.execute(new SQLiteJob<Object>() {

  @Override
  protected Object job(SQLiteConnection connection) throws Throwable {
   connection.exec("BEGIN");
   return null;
  }
 });
 // SQL queries / prepared statements to be executed before the actual
 // QueryScenario statement
 switch (queryScenario) {
 case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY:
 case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES:
 case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES:
  // case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_2HOPS_ONE_ENTITY:
  // case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_2HOPS_10_ENTITIES:
  // case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_2HOPS_100_ENTITIES:
  if (!this.graphStructurePrepared) {
   connection.exec("drop table if exists subjects");
   connection.exec("create table subjects (id text, subject text)");
   connection.exec("create index if not exists idxid on subjects (id)");
   connection.exec("create index if not exists idxsubjects on subjects (subject)");

   connection.exec("BEGIN");

   // Resolve multiple dcterms_subject in a new table
   for (Dataset lastLoadedDataset : lastLoadedDatasets) {
    readRdf(lastLoadedDataset, dataObject -> {
     try {
      if (dataObject.get(Codes.DCTERMS_SUBJECT) != null) {
       for (String oneSubject : (ArrayList<String>) dataObject.get(Codes.DCTERMS_SUBJECT)) {
        connection.exec("insert into subjects values ('" + dataObject.getId() + "','" +
             oneSubject + "')");
       }
      }
     } catch (Exception e) {
      throw new RuntimeException("Cannot insert", e);
     }

    } , counter -> {
     if (counter % COMMIT_EVERY_N_RECORDS == 0) {
      try {
       connection.exec("COMMIT");
       connection.exec("BEGIN");
      } catch (Exception e) {
      }
     }
```

```
    });
   }
   connection.exec("COMMIT");
   this.graphStructurePrepared = true;
  }
  break;
 case SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY:
 case SCHEMA_CHANGE_REMOVE_RDF_TYPE:
  break;
 default:
  queue.execute(new SQLiteJob<Object>() {

   @Override
   protected Object job(SQLiteConnection connection) throws Throwable {
    connection.exec(templates.resolve(queryScenario + "_prepare"));
    return null;
   }
  });
 }

 queue.execute(new SQLiteJob<Object>() {

  @Override
  protected Object job(SQLiteConnection connection) throws Throwable {
   connection.exec("COMMIT");
   return null;
  }
 });

 queue.stop(true).join();

 SQLiteStatement preparedStatement = connection.prepare(templates.resolve(queryScenario));

 // Prepare IDs for ENTITY_RETRIEVAL scenarios
 if (queryScenario.equals(QueryScenario.ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY) || queryScenario.
      equals(QueryScenario.ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES)
   || queryScenario.equals(QueryScenario.ENTITY_RETRIEVAL_BY_ID_100_ENTITIES)) {
  String query = "select DCTERMS_IDENTIFIER from justatable order by dcterms_medium, isbd_p1008,
      dcterm_contributor, dcterms_issued, dcterms_identifier limit";
  switch (queryScenario) {
  case ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY:
   query += " 1;";
   break;
  case ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES:
   query += " 10;";
   break;
  case ENTITY_RETRIEVAL_BY_ID_100_ENTITIES:
   query += " 100;";
   break;
  default:
  }

  SQLiteStatement statement = connection.prepare(query);
  ArrayList<String> ids = new ArrayList<>();
  while (statement.step()) {
   ids.add("'" + statement.columnString(0) + "'");
  }

  // I have no idea why this does not work...
  // preparedStatement.bind(1, Joiner.on(",").join(ids));
  this.ids = ids;
  preparedStatement = connection.prepare("select * from justatable where dcterms_identifier in
      ((?))");
 } else if (queryScenario.queryResultType == Type.GRAPH) {
  // Prepare IDs for ENTITY_RETRIEVAL scenarios

  String query = "select DCTERMS_IDENTIFIER from justatable where dcterms_subject not null order
      by dcterms_medium, isbd_p1008, dcterm_contributor, dcterms_issued, dcterms_identifier
      limit";
  switch (queryScenario) {
  // case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_2HOPS_100_ENTITIES:
  case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES:
   query += " 100;";
   break;
  case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES:
   // case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_2HOPS_10_ENTITIES:
   query += " 10;";
```

```
   break;
  case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY:
   // case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_2HOPS_ONE_ENTITY:
   query += " 1;";
   break;
  default:
   throw new RuntimeException("Dont know how to limit " + queryScenario);
  }

  SQLiteStatement statement = connection.prepare(query);
  ArrayList<String> ids = new ArrayList<>();
  while (statement.step()) {
   ids.add("'" + statement.columnString(0) + "'");
  }

  // I have no idea why this does not work...
  // preparedStatement.setString(1, Joiner.on(",").join(ids));

  String queryString = templates.resolve(queryScenario);
  // queryString += " where level0.id in (?)";
  this.queryString = queryString;
  this.ids = ids;
  preparedStatement = connection.prepare(queryString);
 } else if (queryScenario.toString().startsWith("SCHEMA_")) {
  // System.out.println(queryScenario);
  scenarioQueries.clear();
  String queries = templates.resolve(queryScenario);
  for (String query : queries.split(";")) {
   query = query.trim();
   if (query != null)
    scenarioQueries.add(query);
  }

 }

 this.queryScenario = queryScenario;
 this.scenarioStatements.add(preparedStatement);
}

@Override
public QueryResult query(QueryScenario queryScenario) throws Exception {
 if (scenarioStatements == null || this.queryScenario != queryScenario)
  throw new RuntimeException("There is no preparedStatement for QueryScenario " + queryScenario);

 QueryResult queryResult = new QueryResult(queryScenario.queryResultType);

 // Auto commit setting:
 // https://stackoverflow.com/questions/4998630/how-to-disable-autocommit-in-sqlite4java#5005785
 connection.exec("BEGIN");

 if (queryScenario.toString().startsWith("SCHEMA_")) {
  for (String query : scenarioQueries) {
   // System.out.println(query);
   connection.exec(query);
  }
  // connection.exec("COMMIT");
  // System.exit(1);
 } else {

  for (SQLiteStatement preparedStatement : scenarioStatements) {
   switch (queryScenario.queryResultType) {
   case GRAPH:
    // Preparing does not work :/
    preparedStatement = connection.prepare(queryString + " where level0.id in (" + Joiner.on(",")
        .join(ids) + ")");

    while (preparedStatement.step()) {
     switch (queryScenario) {
     case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY:
     case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES:
     case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES:
      queryResult.push(preparedStatement.columnString(0), preparedStatement.columnString(1),
          preparedStatement.columnString(2));
      break;
     // case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_2HOPS_ONE_ENTITY:
     // case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_2HOPS_10_ENTITIES:
     // case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_2HOPS_100_ENTITIES:
```

```
    // queryResult.push(preparedStatement.columnString(0), preparedStatement.columnString(1),
        preparedStatement.columnString(2),
    // preparedStatement.columnString(3), preparedStatement.columnString(4));
    // break;
    default:
     throw new RuntimeException("Unknown case");
    }
   }
   break;
  case TWO_COLUMNS:
   while (preparedStatement.step()) {
    queryResult.push(preparedStatement.columnString(0), preparedStatement.columnString(1));
   }
   break;
  case COMPLETE_ENTITIES:
   if (queryScenario.toString().startsWith("ENTITY_RETRIEVAL_")) {
    // Preparing does not work :/
    preparedStatement = connection.prepare("select * from justatable where dcterms_identifier in
        (" + Joiner.on(",").join(ids) + ")");
   }

   while (preparedStatement.step()) {
    DataObject dataObject = new DataObject();
    for (Codes code : Codes.values()) {
     if (preparedStatement.columnNull(code.ordinal())) {
      dataObject.set(code, null);
      continue;
     }
     if (code.IS_MULTIPLE) {
      // https://stackoverflow.com/questions/3395729/convert-json-array-to-normal-java-array
      JSONArray jsonArray = new JSONArray(preparedStatement.columnString(code.ordinal()));
      if (jsonArray != null) {
       int len = jsonArray.length();
       for (int i = 0; i < len; i++) {
        dataObject.putMultiple(code, jsonArray.get(i).toString());
       }
      }
     } else {
      dataObject.set(code, preparedStatement.columnString(code.ordinal()));
     }
    }
    queryResult.push(dataObject);
   }
   break;

  case NONE:
  default:

   preparedStatement.stepThrough();
   // while (preparedStatement.step()) {
   // }
   break;
  }

 }
}
connection.exec("COMMIT");
return queryResult;
}

public static void main(String[] args) throws Exception {

 Database database = new SQLite4Java();
 database.start();
 database.clean();
 database.setUp();
 database.load(Dataset.hebis_100000_records);

 QueryScenario queryScenario = QueryScenario.ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY;
 queryScenario = QueryScenario.GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY;
 database.prepare(queryScenario);
 QueryResult queryResult = database.query(queryScenario);
 System.out.println(queryResult);

 queryScenario = QueryScenario.ENTITY_RETRIEVAL_BY_ID_100_ENTITIES;
 queryScenario = QueryScenario.GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES;
 database.prepare(queryScenario);
```

```
  queryResult = database.query(queryScenario);
  System.out.println(queryResult);

//  queryScenario = QueryScenario.GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY;
//  database.prepare(queryScenario);
//  queryResult = database.query(queryScenario);
//  System.out.println(queryResult);


  database.stop();
 }

 @Override
 public void clean() throws Exception {
  // reopenConnection(false);
  // connection.exec("drop table if exists " + Config.TABLE);
  FileUtils.deleteQuietly(DATABASE_FILE);
 }

 @Override
 public void start() {
  // TODO Auto-generated method stub

 }

 @Override
 public void stop() {
  // TODO Auto-generated method stub

 }

 @Override
 public String toString() {
  return "SQLite4Java [getName()=" + getName() + ", getVersion()=" + getVersion() + "]";
 }
}
```

Listing B.1: Java Source Code of the sqlite4java Benchmark Implementation.

B.2 SQLite-Xerial Benchmark Implementation

```
package database;

import java.io.File;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;

import org.apache.commons.io.FileUtils;
import org.json.JSONArray;
import org.sqlite.SQLiteConfig;

import com.google.common.base.Joiner;

import util.Codes;
import util.Config;
import util.DataObject;
import util.Dataset;
import util.QueryResult;
import util.QueryResult.Type;
import util.QueryScenario;
import util.Templates;
import util.dumper.Helpers;

public class SQLiteXerial extends Helpers implements Database {

 private static final Integer COMMIT_EVERY_N_RECORDS = Config.COMMIT_EVERY_N_RECORDS;
 private Connection connection;
 private String createQuery;
 private String genericInsertStatement;
 private ArrayList<PreparedStatement> scenarioStatements;
 private QueryScenario queryScenario;
```

```
private Templates templates;
private ArrayList<Dataset> lastLoadedDatasets = new ArrayList<>();
private boolean graphStructurePrepared = false;
private ArrayList<String> scenarioQueries = new ArrayList<>();
private ArrayList<String> ids;
private String queryString;

public SQLiteXerial() {
 // Produce some queries based on Config / Codes enums - do not prepare
 // statements as PreparedStatements is part of the load() or prepare().
 createQuery = "CREATE TABLE " + Config.TABLE + " ( \n";
 genericInsertStatement = "insert into " + Config.TABLE + " (";
 for (int i = 0; i < Codes.values().length; i++) {
  Codes code = Codes.values()[i];
  createQuery += " " + code.toString() + " " + (code.IS_MULTIPLE ? "text ARRAY" : "text");
  genericInsertStatement += code.toString();
  // Last one?
  createQuery += (Codes.values().length - 1 == i ? "\n)" : ",\n");
  genericInsertStatement += (Codes.values().length - 1 == i ? ")" : ",");
 }
 genericInsertStatement += "\nVALUES(";
 for (int i = 0; i < Codes.values().length; i++) {
  genericInsertStatement += "?" + (Codes.values().length - 1 == i ? ")" : ",");
 }

 templates = new Templates("sqlite", ".sql");
}

@Override
public String getName() {
 return "SQLite-Xerial";
}

@Override
public String getVersion() {
 return "3.8.11";
}

@Override
public void setUp() throws Exception {
 clean();
 reopenConnection(false);

 PreparedStatement createTable = connection.prepareStatement(createQuery);
 createTable.execute();
 lastLoadedDatasets.clear();
}

@Override
public void load(Dataset dataset) throws Exception {
 connection.setAutoCommit(false);

 PreparedStatement insertStatement = connection.prepareStatement(genericInsertStatement);

 readRdf(dataset, dataObject -> {
  try {
   for (int i = 0; i < Codes.values().length; i++) {
    Codes code = Codes.values()[i];
    if (dataObject.get(code) == null) {
     // if (code.IS_MULTIPLE) {
     //// insertStatement.setArray(i + 1,
     // connection.createArrayOf("text", new String[] {}));
     // } else {
     insertStatement.setNull(i + 1, java.sql.Types.VARCHAR);
     // }
    } else {
     Object value = dataObject.get(code);

     if (code.IS_MULTIPLE) {
      value = new JSONArray((ArrayList) value).toString();
     }
     // else {
     insertStatement.setString(i + 1, (String) value);
     // }
    }
   }
   insertStatement.execute();
```

```
   insertStatement.clearParameters();
  } catch (Exception e) {
   throw new RuntimeException("Cannot insert DataObject: " + dataObject, e);
  }
 } , counter -> {
  if (counter % COMMIT_EVERY_N_RECORDS == 0)
   try {
    connection.commit();
   } catch (Exception e) {
    throw new RuntimeException(e);
   }
 });
 connection.commit();
 this.lastLoadedDatasets.add(dataset);
}

private void reopenConnection(boolean readonly) {
 try {
  if (connection != null && !connection.isClosed()) {
   connection.close();
  }

  // connection = DriverManager.getConnection("jdbc:sqlite:" +
  // Config.DATABASE + ".db");

  SQLiteConfig sqLiteConfig = new SQLiteConfig();
  sqLiteConfig.setReadOnly(readonly);
  connection = sqLiteConfig.createConnection("jdbc:sqlite:sqlitexerial.db");
 } catch (SQLException e) {
  throw new RuntimeException("Cannot open connection", e);
 }
}

@Override
public void prepare(QueryScenario queryScenario) throws Exception {
 reopenConnection(false);

 if (scenarioStatements == null)
  scenarioStatements = new ArrayList<>();

 scenarioStatements.clear();

 // SQL queries / prepared statements to be executed before the actual
 // QueryScenario statement
 connection.setAutoCommit(false);
 Statement statement = connection.createStatement();

 switch (queryScenario) {
 case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY:
 case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES:
 case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES:
  // case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_2HOPS_ONE_ENTITY:
  // case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_2HOPS_10_ENTITIES:
  // case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_2HOPS_100_ENTITIES:
  if (!this.graphStructurePrepared) {
   connection.createStatement().executeUpdate("drop table if exists subjects");
   connection.createStatement().executeUpdate("create table subjects (id text, subject text)");

   for (Dataset lastLoadedDataset : lastLoadedDatasets) {
    // Resolve multiple dcterms_subject in a new table
    readRdf(lastLoadedDataset, dataObject -> {
     try {
      if (dataObject.get(Codes.DCTERMS_SUBJECT) != null) {
       for (String oneSubject : (ArrayList<String>) dataObject.get(Codes.DCTERMS_SUBJECT)) {
        connection.createStatement()
          .executeUpdate("insert into subjects values ('" + dataObject.getId() + "','" +
               oneSubject + "')");
       }
      }
     } catch (Exception e) {
      throw new RuntimeException("Cannot insert", e);
     }

    } , counter -> {
     if (counter % COMMIT_EVERY_N_RECORDS == 0)
      try {
       connection.commit();
```

```
    } catch (Exception e) {
    }
   });
  }

  connection.createStatement().executeUpdate("create index if not exists idxid on subjects (id)"
       );
  connection.createStatement().executeUpdate("create index if not exists idxsubjects on subjects
        (subject)");
  this.graphStructurePrepared = true;
 }
 break;
case SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY:
case SCHEMA_CHANGE_REMOVE_RDF_TYPE:
 break;
default:
 statement.executeUpdate(templates.resolve(queryScenario + "_prepare"));
}
connection.commit();

reopenConnection(queryScenario.isReadOnly);
connection.setAutoCommit(false);

// Resolves the template associated with this queryScenario
PreparedStatement preparedStatement = connection.prepareStatement(templates.resolve(
     queryScenario));

// Prepare IDs for ENTITY_RETRIEVAL scenarios
if (queryScenario.equals(QueryScenario.ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY) || queryScenario.
     equals(QueryScenario.ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES)
  || queryScenario.equals(QueryScenario.ENTITY_RETRIEVAL_BY_ID_100_ENTITIES)) {
 String query = "select DCTERMS_IDENTIFIER from justatable order by dcterms_medium, isbd_p1008,
      dcterm_contributor, dcterms_issued, dcterms_identifier limit";
 switch (queryScenario) {
 case ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY:
  query += " 1;";
  break;
 case ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES:
  query += " 10;";
  break;
 case ENTITY_RETRIEVAL_BY_ID_100_ENTITIES:
  query += " 100;";
  break;
 default:
 }

 ResultSet resultSet = connection.createStatement().executeQuery(query);
 ArrayList<String> ids = new ArrayList<>();
 while (resultSet.next()) {
  ids.add("'" + resultSet.getString(1) + "'");
 }

 // I have no idea why this does not work...
 // preparedStatement.setString(1, Joiner.on(",").join(ids));
 this.ids = ids;
 preparedStatement = connection.prepareStatement("select * from justatable where
      dcterms_identifier in (?)");
} else if (queryScenario.queryResultType == Type.GRAPH) {
 // Prepare IDs for ENTITY_RETRIEVAL scenarios

 String query = "select DCTERMS_IDENTIFIER from justatable where dcterms_subject not null order
      by dcterms_medium, isbd_p1008, dcterm_contributor, dcterms_issued, dcterms_identifier
      limit";
 switch (queryScenario) {
 // case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_2HOPS_100_ENTITIES:
 case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES:
  query += " 100;";
  break;
 case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES:
  // case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_2HOPS_10_ENTITIES:
  query += " 10;";
  break;
 case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY:
  // case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_2HOPS_ONE_ENTITY:
  query += " 1;";
  break;
 default:
```

```
    throw new RuntimeException("Dont know how to limit " + queryScenario);
   }

   ResultSet resultSet = connection.createStatement().executeQuery(query);
   ArrayList<String> ids = new ArrayList<>();
   while (resultSet.next()) {
    ids.add("'" + resultSet.getString(1) + "'");
   }

   // I have no idea why this does not work...
   // preparedStatement.setString(1, Joiner.on(",").join(ids));

   String queryString = templates.resolve(queryScenario);
//   queryString += " where level0.id in (?)";
   preparedStatement = connection.prepareStatement(queryString);
   this.ids = ids;
   this.queryString = queryString;
  } else if (queryScenario.toString().startsWith("SCHEMA_")) {
   // System.out.println(queryScenario);
   scenarioQueries.clear();
   String queries = templates.resolve(queryScenario);
   for (String query : queries.split(";")) {
    if (query != null)
     scenarioQueries.add(query);
   }

  }

  this.queryScenario = queryScenario;
  this.scenarioStatements.add(preparedStatement);
 }

 @Override
 public QueryResult query(QueryScenario queryScenario) throws Exception {
  if (scenarioStatements == null || this.queryScenario != queryScenario)
   throw new RuntimeException("There is no preparedStatement for QueryScenario " + queryScenario);

  QueryResult queryResult = new QueryResult(queryScenario.queryResultType);

  for (PreparedStatement preparedStatement : scenarioStatements) {
   ResultSet resultSet;
   switch (queryScenario.queryResultType) {
   case GRAPH:
//    preparedStatement.setString(1, Joiner.on(",").join(ids));
    preparedStatement = connection.prepareStatement(queryString + " where level0.id in (" + Joiner
        .on(",").join(ids) + ")");

    resultSet = preparedStatement.executeQuery();
    while (resultSet.next()) {
     switch (resultSet.getMetaData().getColumnCount()) {
     case 3:
      queryResult.push(resultSet.getString(1), resultSet.getString(2), resultSet.getString(3));
      break;
     case 5:
      queryResult.push(resultSet.getString(1), resultSet.getString(2), resultSet.getString(3),
          resultSet.getString(4),
        resultSet.getString(5));
      break;
     default:
      throw new RuntimeException("Cannot parse " + resultSet.getMetaData().getColumnCount() + "
          columns in result set");
     }
    }
    break;
   case TWO_COLUMNS:
    resultSet = preparedStatement.executeQuery();
    while (resultSet.next()) {
     queryResult.push(resultSet.getString(1), resultSet.getString(2));
    }
    break;
   case COMPLETE_ENTITIES:
    if (queryScenario.toString().startsWith("ENTITY_RETRIEVAL")) {
     // preparing does not work :/
     preparedStatement = connection.prepareStatement("select * from justatable where
         dcterms_identifier in (" + Joiner.on(",").join(ids) + ")");
    }
    resultSet = preparedStatement.executeQuery();
```

```
   while (resultSet.next()) {
    DataObject dataObject = new DataObject();
    for (Codes code : Codes.values()) {
     if (resultSet.getObject(code.ordinal() + 1) == null) {
      dataObject.set(code, null);
      continue;
     }
     if (code.IS_MULTIPLE) {
      // https://stackoverflow.com/questions/3395729/convert-json-array-to-normal-java-array
      JSONArray jsonArray = new JSONArray(resultSet.getString(code.ordinal() + 1));
      if (jsonArray != null) {
       int len = jsonArray.length();
       for (int i = 0; i < len; i++) {
        dataObject.putMultiple(code, jsonArray.get(i).toString());
       }
      }
     } else {
      dataObject.set(code, resultSet.getString(code.ordinal() + 1));
     }
    }
    queryResult.push(dataObject);
   }
   break;

  case NONE:
  default:
   if (queryScenario.toString().startsWith("SCHEMA_")) {
    for (String query : scenarioQueries) {
     // System.out.println("EXEC : " + query);
     connection.createStatement().executeUpdate(query);
    }

   } else {
    preparedStatement.executeUpdate();
   }
   break;
  }
 }
 connection.commit();
 return queryResult;
}

public static void main(String[] args) throws Exception {
 Database database = new SQLiteXerial();
 database.start();
 database.clean();
 database.setUp();
 database.load(Dataset.hebis_100000_records);

 QueryScenario queryScenario = QueryScenario.ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY;
//   queryScenario = QueryScenario.GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY;
 database.prepare(queryScenario);
 QueryResult queryResult = database.query(queryScenario);
 System.out.println(queryResult);

 queryScenario = QueryScenario.ENTITY_RETRIEVAL_BY_ID_100_ENTITIES;
//   queryScenario = QueryScenario.GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES;
 database.prepare(queryScenario);
 queryResult = database.query(queryScenario);
 System.out.println(queryResult);

 // queryScenario = QueryScenario.GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY;
 // database.prepare(queryScenario);
 // queryResult = database.query(queryScenario);
 // System.out.println(queryResult);

 database.stop();
}

@Override
public void clean() throws Exception {
 FileUtils.deleteQuietly(new File("sqlitexerial.db"));
 // reopenConnection(false);
 // connection.prepareStatement("drop table if exists " +
 // Config.TABLE).executeUpdate();
}
```

```
 @Override
 public void start() {
  // TODO Auto-generated method stub

 }

 @Override
 public void stop() {
  // TODO Auto-generated method stub

 }

 @Override
 public String toString() {
  return "SQLiteXerial [getName()=" + getName() + ", getVersion()=" + getVersion() + "]";
 }
}
```

Listing B.2: Java Source Code of the SQLite-Xerial Benchmark Implementation.

B.3 SQLite Queries (used for sqlite4java and SQLite-Xerial)

```
select SUBSTR(DCTERMS_ISSUED, 1, 3), count(DCTERMS_IDENTIFIER)
from justatable
group by SUBSTR(DCTERMS_ISSUED, 1, 3)
order by count(DCTERMS_IDENTIFIER) desc, SUBSTR(DCTERMS_ISSUED, 1, 3) asc
```

Listing B.3: SQL Query (SQLite dialect) for Query Scenario AGGREGATE_ISSUES_PER_DECADE_ALL.

```
drop index if exists DCTERMS_ISSUED_idx;
CREATE INDEX DCTERMS_ISSUED_idx ON justatable (DCTERMS_ISSUED);

drop index if exists DCTERMS_IDENTIFIER_idx;
CREATE INDEX DCTERMS_IDENTIFIER_idx ON justatable (DCTERMS_IDENTIFIER);
```

Listing B.4: SQL Query (SQLite dialect) for Query Scenario AGGREGATE_ISSUES_PER_DECADE_ALL_prepare.

```
select SUBSTR(DCTERMS_ISSUED, 1, 3), count(DCTERMS_IDENTIFIER) from justatable
group by SUBSTR(DCTERMS_ISSUED, 1, 3)
order by count(DCTERMS_IDENTIFIER) desc, SUBSTR(DCTERMS_ISSUED, 1, 3) asc
limit 10
```

Listing B.5: SQL Query (SQLite dialect) for Query Scenario AGGREGATE_ISSUES_PER_DECADE_TOP10.

```
select SUBSTR(DCTERMS_ISSUED, 1, 3), count(DCTERMS_IDENTIFIER) from justatable
group by SUBSTR(DCTERMS_ISSUED, 1, 3)
order by count(DCTERMS_IDENTIFIER) desc, SUBSTR(DCTERMS_ISSUED, 1, 3) asc
limit 100
```

Listing B.6: SQL Query (SQLite dialect) for Query Scenario AGGREGATE_ISSUES_PER_DECADE_TOP100.

```
drop index if exists DCTERMS_ISSUED_idx;
CREATE INDEX DCTERMS_ISSUED_idx ON justatable (DCTERMS_ISSUED);

drop index if exists DCTERMS_IDENTIFIER_idx;
CREATE INDEX DCTERMS_IDENTIFIER_idx ON justatable (DCTERMS_IDENTIFIER);
```

Listing B.7: SQL Query (SQLite dialect) for Query Scenario AGGREGATE_ISSUES_PER_DECADE_TOP100_prepare.

```
drop index if exists DCTERMS_ISSUED_idx;
CREATE INDEX DCTERMS_ISSUED_idx ON justatable (DCTERMS_ISSUED);

drop index if exists DCTERMS_IDENTIFIER_idx;
CREATE INDEX DCTERMS_IDENTIFIER_idx ON justatable (DCTERMS_IDENTIFIER);
```

Listing B.8: SQL Query (SQLite dialect) for Query Scenario AGGREGATE_ISSUES_PER_DECADE_TOP10_prepare.

```
select DCTERMS_PUBLISHER, count(DCTERMS_PUBLISHER) from justatable
group by DCTERMS_PUBLISHER
order by count(DCTERMS_PUBLISHER) desc, DCTERMS_PUBLISHER asc
```

Listing B.9: SQL Query (SQLite dialect) for Query Scenario AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL.

```
drop index if exists DCTERMS_PUBLISHER_idx;
CREATE INDEX DCTERMS_PUBLISHER_idx ON justatable (DCTERMS_PUBLISHER);
```

Listing B.10: SQL Query (SQLite dialect) for Query Scenario AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL_-prepare.

```
select DCTERMS_PUBLISHER, count(DCTERMS_PUBLISHER) from justatable
group by DCTERMS_PUBLISHER
order by count(DCTERMS_PUBLISHER) desc, DCTERMS_PUBLISHER asc
limit 10;
```

Listing B.11: SQL Query (SQLite dialect) for Query Scenario AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10.

```
select DCTERMS_PUBLISHER, count(DCTERMS_PUBLISHER)
from justatable
group by DCTERMS_PUBLISHER
order by count(DCTERMS_PUBLISHER) desc, DCTERMS_PUBLISHER asc
limit 100;
```

Listing B.12: SQL Query (SQLite dialect) for Query Scenario AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100.

```
drop index if exists DCTERMS_PUBLISHER_idx;
CREATE INDEX DCTERMS_PUBLISHER_idx ON justatable (DCTERMS_PUBLISHER);
```

Listing B.13: SQL Query (SQLite dialect) for Query Scenario AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100_prepare.

```
drop index if exists DCTERMS_PUBLISHER_idx;
CREATE INDEX DCTERMS_PUBLISHER_idx ON justatable (DCTERMS_PUBLISHER);
```

Listing B.14: SQL Query (SQLite dialect) for Query Scenario AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10_-prepare.

```
select * from justatable where RDF_TYPE LIKE '%http://purl.org/dc/terms/BibliographicResource%'
```

Listing B.15: SQL Query (SQLite dialect) for Query Scenario CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES.

```
select * from justatable where
RDF_TYPE LIKE '%http://purl.org/dc/terms/BibliographicResource%' and
(LOWER(DCTERMS_TITLE) LIKE '%studie%' OR LOWER(DCTERMS_TITLE) LIKE '%study%')
```

Listing B.16: SQL Query (SQLite dialect) for Query Scenario CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES.

```
--DROP INDEX IF EXISTS RDF_TYPE_idx;
--CREATE INDEX RDF_TYPE_idx on justatable USING GIN ("rdf_type");

--drop index if exists DCTERMS_TITLE_idx;
--CREATE INDEX DCTERMS_TITLE_idx ON justatable (DCTERMS_TITLE);
```

Listing B.17: SQL Query (SQLite dialect) for Query Scenario CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES_prepare.

```
DROP INDEX IF EXISTS RDF_TYPE_idx;
--CREATE INDEX RDF_TYPE_idx on justatable USING GIN ("rdf_type");
```

Listing B.18: SQL Query (SQLite dialect) for Query Scenario CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_prepare.

```
select * from justatable where (LOWER(DCTERMS_TITLE) LIKE '%studie%' OR LOWER(DCTERMS_TITLE) LIKE
    '%study%');
```

Listing B.19: SQL Query (SQLite dialect) for Query Scenario CONDITIONAL_TABLE_SCAN_ALL_STUDIES.

```
drop index if exists DCTERMS_TITLE_idx;
CREATE INDEX DCTERMS_TITLE_idx ON justatable (DCTERMS_TITLE);
```

Listing B.20: SQL Query (SQLite dialect) for Query Scenario CONDITIONAL_TABLE_SCAN_ALL_STUDIES_prepare.

```
delete from justatable where DCTERMS_ISSUED == '0'
```

Listing B.21: SQL Query (SQLite dialect) for Query Scenario DELETE_HIGH_SELECTIVIY_NON_ISSUED.

```
drop index if exists DCTERMS_ISSUED_idx;
CREATE INDEX DCTERMS_ISSUED_idx ON justatable (DCTERMS_ISSUED);
```

Listing B.22: SQL Query (SQLite dialect) for Query Scenario DELETE_HIGH_SELECTIVIY_NON_ISSUED_prepare.

```
delete from justatable where DCTERMS_MEDIUM = 'recycled trees'
```

Listing B.23: SQL Query (SQLite dialect) for Query Scenario DELETE_LOW_SELECTIVITY_PAPER_MEDIUM.

```
drop index if exists DCTERMS_MEDIUM_idx;
CREATE INDEX DCTERMS_MEDIUM_idx ON justatable (DCTERMS_MEDIUM);
```

Listing B.24: SQL Query (SQLite dialect) for Query Scenario DELETE_LOW_SELECTIVITY_PAPER_MEDIUM_prepare.

```
select * from justatable where dcterms_identifier in (?);
```

Listing B.25: SQL Query (SQLite dialect) for Query Scenario ENTITY_RETRIEVAL_BY_ID_100_ENTITIES.

```
drop index if exists DCTERMS_IDENTIFIER_idx;
CREATE INDEX DCTERMS_IDENTIFIER_idx ON justatable (DCTERMS_IDENTIFIER);
```

Listing B.26: SQL Query (SQLite dialect) for Query Scenario ENTITY_RETRIEVAL_BY_ID_100_ENTITIES_prepare.

```
select * from justatable where dcterms_identifier in (?);
```

Listing B.27: SQL Query (SQLite dialect) for Query Scenario ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY.

```
drop index if exists DCTERMS_IDENTIFIER_idx;
CREATE INDEX DCTERMS_IDENTIFIER_idx ON justatable (DCTERMS_IDENTIFIER);
```

Listing B.28: SQL Query (SQLite dialect) for Query Scenario ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY_prepare.

```
select * from justatable where dcterms_identifier in (?);
```

Listing B.29: SQL Query (SQLite dialect) for Query Scenario ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES.

```
drop index if exists DCTERMS_IDENTIFIER_idx;
CREATE INDEX DCTERMS_IDENTIFIER_idx ON justatable (DCTERMS_IDENTIFIER);
```

Listing B.30: SQL Query (SQLite dialect) for Query Scenario ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES_prepare.

```
select level0.id, level0.subject, level1.id
from subjects level0
inner join subjects level1 on (level1.subject = level0.subject and level1.id <> level0.id)

-- Using unnest operator to "flaten" arrays: about twice as slower than using an indexed
     materialized view
--select * from (SELECT dcterms_identifier, unnest(dcterms_subject) subject FROM justatable)
     level0
--inner join (SELECT dcterms_identifier, unnest(dcterms_subject) subject FROM justatable) level1
-- on level0.subject = level1.subject and level0.dcterms_identifier != level1.dcterms_identifier

-- Using native array operations - about 1/3 SLOWER than unnesting
-- select level0.dcterms_identifier, level0.dcterms_subject, level1.dcterms_identifier from
     justatable level0
-- inner join justatable level1 on level0.dcterms_subject && level1.dcterms_subject
-- where level0.dcterms_subject <> '{}' and level0.dcterms_subject <> level1.dcterms_subject
```

Listing B.31: SQL Query (SQLite dialect) for Query Scenario GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES.

```
select level0.id, level0.subject, level1.id
from subjects level0
inner join subjects level1 on (level1.subject = level0.subject and level1.id <> level0.id)

-- Using unnest operator to "flaten" arrays: about twice as slower than using an indexed
     materialized view
--select * from (SELECT dcterms_identifier, unnest(dcterms_subject) subject FROM justatable)
     level0
--inner join (SELECT dcterms_identifier, unnest(dcterms_subject) subject FROM justatable) level1
-- on level0.subject = level1.subject and level0.dcterms_identifier != level1.dcterms_identifier

-- Using native array operations - about 1/3 SLOWER than unnesting
-- select level0.dcterms_identifier, level0.dcterms_subject, level1.dcterms_identifier from
     justatable level0
-- inner join justatable level1 on level0.dcterms_subject && level1.dcterms_subject
-- where level0.dcterms_subject <> '{}' and level0.dcterms_subject <> level1.dcterms_subject
```

Listing B.32: SQL Query (SQLite dialect) for Query Scenario GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES.

```
select level0.id, level0.subject, level1.id
from subjects level0
inner join subjects level1 on (level1.subject = level0.subject and level1.id <> level0.id)

-- Using unnest operator to "flaten" arrays: about twice as slower than using an indexed
     materialized view
--select * from (SELECT dcterms_identifier, unnest(dcterms_subject) subject FROM justatable)
     level0
--inner join (SELECT dcterms_identifier, unnest(dcterms_subject) subject FROM justatable) level1
-- on level0.subject = level1.subject and level0.dcterms_identifier != level1.dcterms_identifier

-- Using native array operations - about 1/3 SLOWER than unnesting
-- select level0.dcterms_identifier, level0.dcterms_subject, level1.dcterms_identifier from
     justatable level0
-- inner join justatable level1 on level0.dcterms_subject && level1.dcterms_subject
-- where level0.dcterms_subject <> '{}' and level0.dcterms_subject <> level1.dcterms_subject
```

Listing B.33: SQL Query (SQLite dialect) for Query Scenario GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY.

```
select level0.id, level0.subject, level1.id, level1.subject, level2.id
from subjects level0
inner join subjects level1 on (level1.subject = level0.subject and level1.id <> level0.id)
```

```
inner join subjects level2 on (level2.subject = level1.subject and level2.id <> level1.id and
    level1.id <> level0.id)
```

Listing B.34: SQL Query (SQLite dialect) for Query Scenario GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_2HOPS_100_ENTITIES.

```
select level0.id, level0.subject, level1.id, level1.subject, level2.id
from subjects level0
inner join subjects level1 on (level1.subject = level0.subject and level1.id <> level0.id)
inner join subjects level2 on (level2.subject = level1.subject and level2.id <> level1.id and
    level1.id <> level0.id)
```

Listing B.35: SQL Query (SQLite dialect) for Query Scenario GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_2HOPS_10_ENTITIES.

```
select level0.id, level0.subject, level1.id, level1.subject, level2.id
from subjects level0
inner join subjects level1 on (level1.subject = level0.subject and level1.id <> level0.id)
inner join subjects level2 on (level2.subject = level1.subject and level2.id <> level1.id and
    level1.id <> level0.id)
```

Listing B.36: SQL Query (SQLite dialect) for Query Scenario GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_2HOPS_ONE_ENTITY.

```
alter table justatable add column ${Config.NEWCOLUMN} text default 'cheesecake'
```

Listing B.37: SQL Query (SQLite dialect) for Query Scenario SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY.

```
alter table justatable
add column istrop text;

update justatable set istrop=substr(RDF_ABOUT, 30);
```

Listing B.38: SQL Query (SQLite dialect) for Query Scenario SCHEMA_CHANGE_INTRODUCE_STRING_OP.

```
drop index if exists RDF_ABOUT_idx;
CREATE INDEX RDF_ABOUT_idx ON justatable (RDF_ABOUT);
```

Listing B.39: SQL Query (SQLite dialect) for Query Scenario SCHEMA_CHANGE_INTRODUCE_STRING_OP_prepare.

```
alter table justatable add column manifestation boolean default false;
update justatable set manifestation = "true" where RDF_TYPE like "%http://purl.org/vocab/frbr/
    core#Manifestation%";

alter table justatable add column bibliographicresource boolean default false;
update justatable set bibliographicresource = "true" where RDF_TYPE like "%http://purl.org/dc/
    terms/BibliographicResource%";

alter table justatable add column book boolean default false;
update justatable set book = "true" where RDF_TYPE like "%http://purl.org/ontology/bibo/Book%"
```

Listing B.40: SQL Query (SQLite dialect) for Query Scenario SCHEMA_CHANGE_MIGRATE_RDF_TYPE.

```
--DROP INDEX IF EXISTS RDF_TYPE_idx;
--CREATE INDEX RDF_TYPE_idx on justatable USING GIN ("rdf_type");
```

Listing B.41: SQL Query (SQLite dialect) for Query Scenario SCHEMA_CHANGE_MIGRATE_RDF_TYPE_prepare.

```
DROP TABLE IF EXISTS justatable2;

CREATE TABLE justatable2 (
 DCTERMS_IDENTIFIER text,
 BIBO_OCLCNUM text,
 RDF_ABOUT text,
 DCTERMS_TITLE text,
 DCTERMS_PUBLISHER text,
 ISBD_P1017 text,
 ISBD_P1016 text,
 ISBD_P1008 text,
 ISBD_P1006 text,
 ISBD_P1004 text,
 ISBD_P1018 text,
 DCTERMS_ISSUED text,
 OWL_SAMEAS text,
 DCTERMS_MEDIUM text,
 DCTERMS_FORMAT text,
 BIBO_EDITION text,
 WDRS_DESCRIBEDBY text,
 DCTERMS_SUBJECT text ARRAY,
 DCTERM_CONTRIBUTOR text
);

insert into justatable2 (DCTERMS_IDENTIFIER,
 BIBO_OCLCNUM,
 RDF_ABOUT,
 DCTERMS_TITLE,
 DCTERMS_PUBLISHER,
 ISBD_P1017,
 ISBD_P1016,
 ISBD_P1008,
 ISBD_P1006,
 ISBD_P1004,
 ISBD_P1018,
 DCTERMS_ISSUED,
 OWL_SAMEAS,
 DCTERMS_MEDIUM,
 DCTERMS_FORMAT,
 BIBO_EDITION,
 WDRS_DESCRIBEDBY,
 DCTERMS_SUBJECT,
 DCTERM_CONTRIBUTOR)
select DCTERMS_IDENTIFIER,
 BIBO_OCLCNUM,
 RDF_ABOUT,
 DCTERMS_TITLE,
 DCTERMS_PUBLISHER,
 ISBD_P1017,
 ISBD_P1016,
 ISBD_P1008,
 ISBD_P1006,
 ISBD_P1004,
 ISBD_P1018,
 DCTERMS_ISSUED,
 OWL_SAMEAS,
 DCTERMS_MEDIUM,
 DCTERMS_FORMAT,
 BIBO_EDITION,
 WDRS_DESCRIBEDBY,
 DCTERMS_SUBJECT,
 DCTERM_CONTRIBUTOR from justatable;

drop table justatable;

ALTER TABLE justatable2 rename to justatable
```

Listing B.42: SQL Query (SQLite dialect) for Query Scenario SCHEMA_CHANGE_REMOVE_RDF_TYPE.

```
update justatable set dcterms_issued = '0' where dcterms_issued is null
```

Listing B.43: SQL Query (SQLite dialect) for Query Scenario UPDATE_HIGH_SELECTIVITY_NON_ISSUED.

```
drop index if exists issued_idx;
CREATE INDEX issued_idx ON justatable (dcterms_issued);
```

Listing B.44: SQL Query (SQLite dialect) for Query Scenario UPDATE_HIGH_SELECTIVITY_NON_ISSUED_prepare.

```
update justatable set DCTERMS_MEDIUM = 'recycled trees' where DCTERMS_MEDIUM = 'paper'
```

Listing B.45: SQL Query (SQLite dialect) for Query Scenario UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM.

```
drop index if exists DCTERMS_MEDIUM_idx;
CREATE INDEX DCTERMS_MEDIUM_idx ON justatable (DCTERMS_MEDIUM);
```

Listing B.46: SQL Query (SQLite dialect) for Query Scenario UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM_prepare.

B.4 PostgreSQL Benchmark Implementation

```
package database;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.Properties;

import com.google.common.base.Joiner;

import util.Codes;
import util.Config;
import util.DataObject;
import util.Dataset;
import util.QueryResult;
import util.QueryScenario;
import util.Templates;
import util.QueryResult.Type;
import util.dumper.Helpers;

public class PostgreSQL extends Helpers implements Database {

 private Connection connection;
 private Properties props;
 private String createQuery;
 private String genericInsertStatement;
 private ArrayList<PreparedStatement> scenarioStatements;
 private QueryScenario queryScenario;
 private Templates templates;
 private boolean graphStructurePrepared = false;
 private ArrayList<String> ids;
 private String queryString;

 @Override
 public String getName() {
  return "PostgreSQL";
 }

 @Override
 public String getVersion() {
  return "PostgreSQL 9.4.4 on x86_64-apple-darwin14.3.0, compiled by Apple LLVM version 6.1.0 (
       clang-602.0.53) (based on LLVM 3.6.0svn), 64-bit / 9.4-1201-jdbc41";
 }

 public static void main(String[] args) throws Exception {

  Database database = new PostgreSQL();
  // database.start();
  database.clean();
  database.setUp();
  database.load(Dataset.hebis_100000_records);
```

```
 QueryScenario queryScenario = QueryScenario.ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY;
 queryScenario = QueryScenario.GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY;
 database.prepare(queryScenario);
 QueryResult queryResult = database.query(queryScenario);
 System.out.println(queryResult);

 // queryScenario = QueryScenario.ENTITY_RETRIEVAL_BY_ID_100_ENTITIES;
 // queryScenario = QueryScenario.GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES;
 // database.prepare(queryScenario);
 // queryResult = database.query(queryScenario);
 // System.out.println(queryResult);

 // queryScenario = QueryScenario.GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY;
 // database.prepare(queryScenario);
 // queryResult = database.query(queryScenario);
 // System.out.println(queryResult);

 // database.stop();
}

public PostgreSQL() {
 props = new Properties();
 props.setProperty("user", Config.USER);
 props.setProperty("password", Config.PASSWORD);

 // Produce some queries based on Config / Codes enums - do not prepare
 // statements as PreparedStatements is part of the load() or prepare().
 createQuery = "CREATE TABLE " + Config.TABLE + " ( \n";
 genericInsertStatement = "insert into " + Config.TABLE + " (";
 for (int i = 0; i < Codes.values().length; i++) {
  Codes code = Codes.values()[i];
  createQuery += " " + code.toString() + " " + (code.IS_MULTIPLE ? "text ARRAY" : "text");
  genericInsertStatement += code.toString();
  // Last one?
  createQuery += (Codes.values().length - 1 == i ? "\n)" : ",\n");
  genericInsertStatement += (Codes.values().length - 1 == i ? ")" : ",");
 }
 genericInsertStatement += "\nVALUES(";
 for (int i = 0; i < Codes.values().length; i++) {
  genericInsertStatement += "?" + (Codes.values().length - 1 == i ? ")" : ",");
 }
}

/**
 * Create the database in local PostgreSQL
 */
@Override
public void setUp() throws Exception {
 // try {
 // reopenConnection(false);
 // dropConnections();
 // } catch (Exception e) {
 // // Will drop its own connection - ignore
 // }
 // clean();
 Connection maintenanceConnection = DriverManager.getConnection("jdbc:postgresql://" + Config.
     HOST_POSTGRES + "/postgres", props);
 // connection.setAutoCommit(false);

 // Aggressively drop possibly open connections
 // https://stackoverflow.com/questions/7073773/drop-postgresql-database-through-command-line
 PreparedStatement preparedStatement = maintenanceConnection
   .prepareStatement("select pg_terminate_backend(pid) from pg_stat_activity where datname='" +
       Config.DATABASE + "';");
 preparedStatement.execute();

 preparedStatement = maintenanceConnection.prepareStatement("CREATE DATABASE " + Config.DATABASE
     );
 preparedStatement.execute();

 templates = new Templates("postgres", ".sql");

 preparedStatement.close();
 // connection.commit();
 maintenanceConnection.close();

 reopenConnection(false);
```

```
 // this.connection.createStatement().executeUpdate("DROP TABLE IF EXISTS
 // " + Config.TABLE);
 this.connection.createStatement().executeUpdate(createQuery);
}

@Override
public void load(Dataset dataset) throws Exception {
 reopenConnection(false);
 connection.setAutoCommit(false);

 PreparedStatement insertStatement = connection.prepareStatement(genericInsertStatement);

 readRdf(dataset, dataObject -> {
  try {
   for (int i = 0; i < Codes.values().length; i++) {
    Codes code = Codes.values()[i];
    if (dataObject.get(code) == null) {
     if (code.IS_MULTIPLE) {
      insertStatement.setArray(i + 1, connection.createArrayOf("text", new String[] {}));
     } else {
      insertStatement.setNull(i + 1, java.sql.Types.VARCHAR);
     }
    } else {
     if (code.IS_MULTIPLE) {
      insertStatement.setArray(i + 1, connection.createArrayOf("text", ((ArrayList<String>)
           dataObject.get(code)).toArray()));
     } else {
      insertStatement.setString(i + 1, (String) dataObject.get(code));
     }
    }
   }
   insertStatement.execute();
   insertStatement.clearParameters();
  } catch (Exception e) {
   throw new RuntimeException("Cannot insert DataObject: " + dataObject, e);
  }
 } , counter -> {
  if (counter % Config.COMMIT_EVERY_N_RECORDS == 0)
   try {
    connection.commit();
   } catch (Exception e) {
   }
 });

 connection.commit();
}

@Override
public void prepare(QueryScenario queryScenario) throws Exception {
 reopenConnection(queryScenario.isReadOnly);
 connection.setAutoCommit(false);

 if (scenarioStatements == null)
  scenarioStatements = new ArrayList<>();

 scenarioStatements.clear();

 // SQL queries / prepared statements to be executed before the actual
 // QueryScenario statement
 Statement statement = connection.createStatement();
 switch (queryScenario) {
 case SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY:
 case SCHEMA_CHANGE_REMOVE_RDF_TYPE:
  break;
 case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY:
 case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES:
 case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES:
  // case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_2HOPS_ONE_ENTITY:
  // case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_2HOPS_10_ENTITIES:
  // case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_2HOPS_100_ENTITIES:
  if (!this.graphStructurePrepared) {
   this.graphStructurePrepared = true;
   // Fall through
  } else {
   break;
  }
 default:
```

```
   statement.executeUpdate(templates.resolve(queryScenario + "_prepare"));
  }

  connection.commit();

  reopenConnection(queryScenario.isReadOnly);
  connection.setAutoCommit(false);

  // Resolves the template associated with this queryScenario
  PreparedStatement preparedStatement = connection.prepareStatement(templates.resolve(
      queryScenario));

  // Prepare IDs for ENTITY_RETRIEVAL scenarios
  if (queryScenario.equals(QueryScenario.ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY) || queryScenario.
      equals(QueryScenario.ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES)
    || queryScenario.equals(QueryScenario.ENTITY_RETRIEVAL_BY_ID_100_ENTITIES)) {
   String query = "select DCTERMS_IDENTIFIER from justatable order by dcterms_medium, isbd_p1008,
       dcterm_contributor, dcterms_issued, dcterms_identifier limit";
   switch (queryScenario) {
   case ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY:
    query += " 1;";
    break;
   case ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES:
    query += " 10;";
    break;
   case ENTITY_RETRIEVAL_BY_ID_100_ENTITIES:
    query += " 100;";
    break;
   default:
   }

   ResultSet resultSet = connection.createStatement().executeQuery(query);
   ArrayList<String> ids = new ArrayList<>();
   while (resultSet.next()) {
    ids.add("'" + resultSet.getString(1) + "'");
   }
   // preparedStatement.setString(1, Joiner.on(",").join(ids));

   this.ids = ids;
   // System.out.println("IDS: " + ids.size());
   preparedStatement = connection.prepareStatement("select * from justatable where
       dcterms_identifier in ($1)");

   // System.out.println(preparedStatement);
  } else if (queryScenario.queryResultType == Type.GRAPH) {
   // Prepare IDs for ENTITY_RETRIEVAL scenarios

   String query = "select DCTERMS_IDENTIFIER from justatable where dcterms_subject IS NOT null
       order by dcterms_medium, isbd_p1008, dcterm_contributor, dcterms_issued,
       dcterms_identifier limit";
   switch (queryScenario) {
   // case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_2HOPS_100_ENTITIES:
   case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES:
    query += " 100;";
    break;
   case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES:
    // case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_2HOPS_10_ENTITIES:
    query += " 10;";
    break;
   case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY:
    // case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_2HOPS_ONE_ENTITY:
    query += " 1;";
    break;
   default:
    throw new RuntimeException("Dont know how to limit " + queryScenario);
   }
//   System.out.println(query);
   ResultSet resultSet = connection.createStatement().executeQuery(query);
   ArrayList<String> ids = new ArrayList<>();
   while (resultSet.next()) {
    ids.add("'" + resultSet.getString(1) + "'");
   }

   String queryString = templates.resolve(queryScenario);
   queryString += " where level0.dcterms_identifier in ";

   this.queryString = queryString;
```

```
  this.ids = ids;

  // System.out.println(queryString);
  preparedStatement = connection.prepareStatement(queryString);
 }

 this.scenarioStatements.add(preparedStatement);
 this.queryScenario = queryScenario;

}

@Override
public QueryResult query(QueryScenario queryScenario) throws Exception {
 if (scenarioStatements == null || this.queryScenario != queryScenario)
  throw new RuntimeException("There is no preparedStatement for QueryScenario " + queryScenario);

 QueryResult queryResult = new QueryResult(queryScenario.queryResultType);

 for (PreparedStatement preparedStatement : scenarioStatements) {
  ResultSet resultSet;
  switch (queryScenario.queryResultType) {
  case GRAPH:
   preparedStatement = connection.prepareStatement(queryString + "(" + (ids.size() > 0 ? Joiner.
       on(",").join(ids) : "null") + ")");

   // System.out.println(queryString + "(" + Joiner.on(",").join(ids) + ")");
   resultSet = preparedStatement.executeQuery();
   while (resultSet.next()) {
    switch (resultSet.getMetaData().getColumnCount()) {
    case 3:
     queryResult.push(resultSet.getString(1), resultSet.getString(2), resultSet.getString(3));
     break;
    case 5:
     queryResult.push(resultSet.getString(1), resultSet.getString(2), resultSet.getString(3),
         resultSet.getString(4),
       resultSet.getString(5));
     break;
    default:
     throw new RuntimeException("Cannot parse " + resultSet.getMetaData().getColumnCount() + "
         columns in result set");
    }
   }
   break;
  case TWO_COLUMNS:
   resultSet = preparedStatement.executeQuery();
   while (resultSet.next()) {
    queryResult.push(resultSet.getString(1), resultSet.getString(2));
   }
   break;
  case COMPLETE_ENTITIES:
   if (queryScenario.toString().contains("ENTITY_RETRIEVAL_")) {
    preparedStatement = connection.prepareStatement("select * from justatable where
        dcterms_identifier in (" + Joiner.on(",").join(ids) + ")");
   }
   resultSet = preparedStatement.executeQuery();
   while (resultSet.next()) {
    DataObject dataObject = new DataObject();
    for (Codes code : Codes.values()) {
     if (code.IS_MULTIPLE) {
      for (String value : (String[]) resultSet.getArray(code.ordinal() + 1).getArray()) {
       dataObject.putMultiple(code, value);
      }

     } else {
      dataObject.set(code, resultSet.getString(code.ordinal() + 1));
     }
    }
    queryResult.push(dataObject);
   }
   break;
  case NONE:
  default:
   // This will only work for non-readOnly-scenarios!
   preparedStatement.executeUpdate();
   break;
  }
 }
```

```
  connection.commit();
  return queryResult;

 }

 private void reopenConnection(boolean readonly) throws Exception {
  props.setProperty("ReadOnly", readonly ? "true" : "false");

  if (connection != null && !connection.isClosed())
   connection.close();

  if (connection == null || connection.isClosed())
   connection = DriverManager.getConnection("jdbc:postgresql://" + Config.HOST_POSTGRES + "/" +
        Config.DATABASE, props);
 }

 private void dropConnections() throws Exception {
  connection.createStatement().executeQuery("select pg_terminate_backend(pid) from
       pg_stat_activity where datname='loddwhbench';");
 }

 @Override
 public String toString() {
  return "PostgreSQL [getName()=" + getName() + ", getVersion()=" + getVersion() + "]";
 }

 @Override
 public void clean() throws Exception {
  // try {
  // reopenConnection(false);
  // dropConnections();
  // } catch (Exception e) {
  // // Will drop its own connection - ignore
  // }
  Connection maintenanceConnection = DriverManager.getConnection("jdbc:postgresql://" + Config.
       HOST_POSTGRES + "/postgres", props);
  // connection.setAutoCommit(false);

  // Aggressively drop possibly open connections
  // https://stackoverflow.com/questions/7073773/drop-postgresql-database-through-command-line
  PreparedStatement preparedStatement = maintenanceConnection
    .prepareStatement("select pg_terminate_backend(pid) from pg_stat_activity where datname='" +
         Config.DATABASE + "';");
  preparedStatement.execute();

  preparedStatement = maintenanceConnection.prepareStatement("DROP DATABASE IF EXISTS " + Config.
       DATABASE);
  try {
   preparedStatement.execute();
  } catch (Exception e) {
   e.printStackTrace();
  }
 }

 @Override
 public void start() {
  terminalLaunchScript("postgresql.sh", 5);
 }

 @Override
 public void stop() {
  terminalLaunchApp(new String[] { "/usr/local/bin/pg_ctl", "-D", "/usr/local/var/loddwhbench", "
       stop", "-m", "immediate" });
 }

}
```

Listing B.47: Java Source Code of the PostgreSQL Benchmark Implementation.

```
select SUBSTR(DCTERMS_ISSUED, 1, 3), count(DCTERMS_IDENTIFIER)
from justatable
group by SUBSTR(DCTERMS_ISSUED, 1, 3)
order by count(DCTERMS_IDENTIFIER) desc, SUBSTR(DCTERMS_ISSUED, 1, 3) asc
```

Listing B.48: SQL Query (PostgreSQL dialect) for Query Scenario AGGREGATE_ISSUES_PER_DECADE_ALL.

```
drop index if exists DCTERMS_ISSUED_idx;
CREATE INDEX DCTERMS_ISSUED_idx ON justatable (DCTERMS_ISSUED);

drop index if exists DCTERMS_IDENTIFIER_idx;
CREATE INDEX DCTERMS_IDENTIFIER_idx ON justatable (DCTERMS_IDENTIFIER);
```

Listing B.49: SQL Query (PostgreSQL dialect) for Query Scenario AGGREGATE_ISSUES_PER_DECADE_ALL_prepare.

```
select SUBSTR(DCTERMS_ISSUED, 1, 3), count(DCTERMS_IDENTIFIER)
from justatable
group by SUBSTR(DCTERMS_ISSUED, 1, 3)
order by count(DCTERMS_IDENTIFIER) desc, SUBSTR(DCTERMS_ISSUED, 1, 3) asc
limit 10
```

Listing B.50: SQL Query (PostgreSQL dialect) for Query Scenario AGGREGATE_ISSUES_PER_DECADE_TOP10.

```
select SUBSTR(DCTERMS_ISSUED, 1, 3), count(DCTERMS_IDENTIFIER)
from justatable
group by SUBSTR(DCTERMS_ISSUED, 1, 3)
order by count(DCTERMS_IDENTIFIER) desc, SUBSTR(DCTERMS_ISSUED, 1, 3) asc
limit 100
```

Listing B.51: SQL Query (PostgreSQL dialect) for Query Scenario AGGREGATE_ISSUES_PER_DECADE_TOP100.

```
drop index if exists DCTERMS_ISSUED_idx;
CREATE INDEX DCTERMS_ISSUED_idx ON justatable (DCTERMS_ISSUED);

drop index if exists DCTERMS_IDENTIFIER_idx;
CREATE INDEX DCTERMS_IDENTIFIER_idx ON justatable (DCTERMS_IDENTIFIER);
```

Listing B.52: SQL Query (PostgreSQL dialect) for Query Scenario AGGREGATE_ISSUES_PER_DECADE_TOP100-prepare.

```
drop index if exists DCTERMS_ISSUED_idx;
CREATE INDEX DCTERMS_ISSUED_idx ON justatable (DCTERMS_ISSUED);

drop index if exists DCTERMS_IDENTIFIER_idx;
CREATE INDEX DCTERMS_IDENTIFIER_idx ON justatable (DCTERMS_IDENTIFIER);
```

Listing B.53: SQL Query (PostgreSQL dialect) for Query Scenario AGGREGATE_ISSUES_PER_DECADE_TOP10_prepare.

```
select DCTERMS_PUBLISHER, count(DCTERMS_PUBLISHER) from justatable
group by DCTERMS_PUBLISHER
order by count(DCTERMS_PUBLISHER), DCTERMS_PUBLISHER
desc
```

Listing B.54: SQL Query (PostgreSQL dialect) for Query Scenario AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL.

```
drop index if exists DCTERMS_PUBLISHER_idx;
CREATE INDEX DCTERMS_PUBLISHER_idx ON justatable (DCTERMS_PUBLISHER);
```

Listing B.55: SQL Query (PostgreSQL dialect) for Query Scenario AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL_-prepare.

```
select DCTERMS_PUBLISHER, count(DCTERMS_PUBLISHER) from justatable
group by DCTERMS_PUBLISHER
order by count(DCTERMS_PUBLISHER), DCTERMS_PUBLISHER
desc limit 10;
```

Listing B.56: SQL Query (PostgreSQL dialect) for Query Scenario AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10.

```
select DCTERMS_PUBLISHER, count(DCTERMS_PUBLISHER) from justatable
group by DCTERMS_PUBLISHER
order by count(DCTERMS_PUBLISHER), DCTERMS_PUBLISHER
desc limit 100;
```

Listing B.57: SQL Query (PostgreSQL dialect) for Query Scenario AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100.

```
drop index if exists DCTERMS_PUBLISHER_idx;
CREATE INDEX DCTERMS_PUBLISHER_idx ON justatable (DCTERMS_PUBLISHER);
```

Listing B.58: SQL Query (PostgreSQL dialect) for Query Scenario AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100_prepare.

```
drop index if exists DCTERMS_PUBLISHER_idx;
CREATE INDEX DCTERMS_PUBLISHER_idx ON justatable (DCTERMS_PUBLISHER);
```

Listing B.59: SQL Query (PostgreSQL dialect) for Query Scenario AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10_prepare.

```
select * from justatable where 'http://purl.org/dc/terms/BibliographicResource' = ANY(RDF_TYPE)
```

Listing B.60: SQL Query (PostgreSQL dialect) for Query Scenario CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES.

```
select * from justatable where
 'http://purl.org/dc/terms/BibliographicResource' = ANY(RDF_TYPE)
 and (DCTERMS_TITLE ILIKE '%studie%' OR DCTERMS_TITLE ILIKE '%study%')
```

Listing B.61: SQL Query (PostgreSQL dialect) for Query Scenario CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES.

```
DROP INDEX IF EXISTS RDF_TYPE_idx;
CREATE INDEX RDF_TYPE_idx on justatable USING GIN ("rdf_type");

drop index if exists DCTERMS_TITLE_idx;
CREATE INDEX DCTERMS_TITLE_idx ON justatable (DCTERMS_TITLE);
```

Listing B.62: SQL Query (PostgreSQL dialect) for Query Scenario CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES_prepare.

```
DROP INDEX IF EXISTS RDF_TYPE_idx;
CREATE INDEX RDF_TYPE_idx on justatable USING GIN ("rdf_type");
```

Listing B.63: SQL Query (PostgreSQL dialect) for Query Scenario CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_prepare.

```
select * from justatable where (LOWER(DCTERMS_TITLE) LIKE '%studie%' OR LOWER(DCTERMS_TITLE) LIKE
    '%study%');
```

Listing B.64: SQL Query (PostgreSQL dialect) for Query Scenario CONDITIONAL_TABLE_SCAN_ALL_STUDIES.

```
drop index if exists DCTERMS_TITLE_idx;
CREATE INDEX DCTERMS_TITLE_idx ON justatable (DCTERMS_TITLE);
```

Listing B.65: SQL Query (PostgreSQL dialect) for Query Scenario CONDITIONAL_TABLE_SCAN_ALL_STUDIES_prepare.

```
delete from justatable where DCTERMS_ISSUED = '0'
```

Listing B.66: SQL Query (PostgreSQL dialect) for Query Scenario DELETE_HIGH_SELECTIVIY_NON_ISSUED.

```
drop index if exists DCTERMS_ISSUED_idx;
CREATE INDEX DCTERMS_ISSUED_idx ON justatable (DCTERMS_ISSUED);
```

Listing B.67: SQL Query (PostgreSQL dialect) for Query Scenario DELETE_HIGH_SELECTIVIY_NON_ISSUED_prepare.

```
delete from justatable where DCTERMS_MEDIUM = 'recycled trees'
```

Listing B.68: SQL Query (PostgreSQL dialect) for Query Scenario DELETE_LOW_SELECTIVITY_PAPER_MEDIUM.

```
drop index if exists DCTERMS_MEDIUM_idx;
CREATE INDEX DCTERMS_MEDIUM_idx ON justatable (DCTERMS_MEDIUM);
```

Listing B.69: SQL Query (PostgreSQL dialect) for Query Scenario DELETE_LOW_SELECTIVITY_PAPER_MEDIUM_prepare.

```
select * from justatable where dcterms_identifier in (?);
```

Listing B.70: SQL Query (PostgreSQL dialect) for Query Scenario ENTITY_RETRIEVAL_BY_ID_100_ENTITIES.

```
drop index if exists DCTERMS_IDENTIFIER_idx;
CREATE INDEX DCTERMS_IDENTIFIER_idx ON justatable (DCTERMS_IDENTIFIER);
```

Listing B.71: SQL Query (PostgreSQL dialect) for Query Scenario ENTITY_RETRIEVAL_BY_ID_100_ENTITIES_prepare.

```
select * from justatable where dcterms_identifier in (?);
```

Listing B.72: SQL Query (PostgreSQL dialect) for Query Scenario ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY.

```
drop index if exists DCTERMS_IDENTIFIER_idx;
CREATE INDEX DCTERMS_IDENTIFIER_idx ON justatable (DCTERMS_IDENTIFIER);
```

Listing B.73: SQL Query (PostgreSQL dialect) for Query Scenario ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY_prepare.

```
select * from justatable where dcterms_identifier in (?);
```

Listing B.74: SQL Query (PostgreSQL dialect) for Query Scenario ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES.

```
drop index if exists DCTERMS_IDENTIFIER_idx;
CREATE INDEX DCTERMS_IDENTIFIER_idx ON justatable (DCTERMS_IDENTIFIER);
```

Listing B.75: SQL Query (PostgreSQL dialect) for Query Scenario ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES_prepare.

```
-- Using unnest operator to "flaten" arrays: about twice as slower than using an indexed
     materialized view
--select * from (SELECT dcterms_identifier, unnest(dcterms_subject) subject FROM justatable)
     level0
--inner join (SELECT dcterms_identifier, unnest(dcterms_subject) subject FROM justatable) level1
-- on level0.subject = level1.subject and level0.dcterms_identifier != level1.dcterms_identifier

-- Using native array operations - about 1/3 SLOWER than unnesting
-- select level0.dcterms_identifier, level0.dcterms_subject, level1.dcterms_identifier from
     justatable level0
-- inner join justatable level1 on level0.dcterms_subject && level1.dcterms_subject
-- where level0.dcterms_subject <> '{}' and level0.dcterms_subject <> level1.dcterms_subject
```

```
select level0.dcterms_identifier, level0.subject, level1.dcterms_identifier
from unnestedsubjects level0
inner join unnestedsubjects level1 on (level1.subject = level0.subject and level1.
    dcterms_identifier <> level0.dcterms_identifier)
```

Listing B.76: SQL Query (PostgreSQL dialect) for Query Scenario GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES.

```
drop MATERIALIZED view if exists unnestedsubjects;

create MATERIALIZED view unnestedsubjects as
(SELECT dcterms_identifier, unnest(dcterms_subject) subject FROM justatable where dcterms_subject
     <> '{}');

CREATE INDEX unnestedsubjects_identifer ON unnestedsubjects (dcterms_identifier);
CREATE INDEX unnestedsubjects_subject ON unnestedsubjects (subject);


-- Old gin index - was never really used.
-- DROP INDEX IF EXISTS idx_identifier;
-- CREATE INDEX idx_identifier on "${Config.TABLE}" ("dcterms_identifier");
-- DROP INDEX IF EXISTS idx_subjects;
-- CREATE INDEX idx_subjects on "${Config.TABLE}" USING GIN ("dcterms_subject");
```

Listing B.77: SQL Query (PostgreSQL dialect) for Query Scenario GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES_prepare.

```
-- Using unnest operator to "flaten" arrays: about twice as slower than using an indexed
    materialized view
--select * from (SELECT dcterms_identifier, unnest(dcterms_subject) subject FROM justatable)
    level0
--inner join (SELECT dcterms_identifier, unnest(dcterms_subject) subject FROM justatable) level1
-- on level0.subject = level1.subject and level0.dcterms_identifier != level1.dcterms_identifier

-- Using native array operations - about 1/3 SLOWER than unnesting
-- select level0.dcterms_identifier, level0.dcterms_subject, level1.dcterms_identifier from
    justatable level0
-- inner join justatable level1 on level0.dcterms_subject && level1.dcterms_subject
-- where level0.dcterms_subject <> '{}' and level0.dcterms_subject <> level1.dcterms_subject

select level0.dcterms_identifier, level0.subject, level1.dcterms_identifier
from unnestedsubjects level0
inner join unnestedsubjects level1 on (level1.subject = level0.subject and level1.
    dcterms_identifier <> level0.dcterms_identifier)
```

Listing B.78: SQL Query (PostgreSQL dialect) for Query Scenario GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES.

```
drop MATERIALIZED view if exists unnestedsubjects;

create MATERIALIZED view unnestedsubjects as
(SELECT dcterms_identifier, unnest(dcterms_subject) subject FROM justatable where dcterms_subject
     <> '{}');

CREATE INDEX unnestedsubjects_identifer ON unnestedsubjects (dcterms_identifier);
CREATE INDEX unnestedsubjects_subject ON unnestedsubjects (subject);


-- Old gin index - was never really used.
-- DROP INDEX IF EXISTS idx_identifier;
-- CREATE INDEX idx_identifier on "${Config.TABLE}" ("dcterms_identifier");
-- DROP INDEX IF EXISTS idx_subjects;
-- CREATE INDEX idx_subjects on "${Config.TABLE}" USING GIN ("dcterms_subject");
```

Listing B.79: SQL Query (PostgreSQL dialect) for Query Scenario GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES_prepare.

```
-- Using unnest operator to "flaten" arrays: about twice as slower than using an indexed
     materialized view
--select * from (SELECT dcterms_identifier, unnest(dcterms_subject) subject FROM justatable)
     level0
--inner join (SELECT dcterms_identifier, unnest(dcterms_subject) subject FROM justatable) level1
-- on level0.subject = level1.subject and level0.dcterms_identifier != level1.dcterms_identifier

-- Using native array operations - about 1/3 SLOWER than unnesting
-- select level0.dcterms_identifier, level0.dcterms_subject, level1.dcterms_identifier from
     justatable level0
-- inner join justatable level1 on level0.dcterms_subject && level1.dcterms_subject
-- where level0.dcterms_subject <> '{}' and level0.dcterms_subject <> level1.dcterms_subject

select level0.dcterms_identifier, level0.subject, level1.dcterms_identifier
from unnestedsubjects level0
inner join unnestedsubjects level1 on (level1.subject = level0.subject and level1.
     dcterms_identifier <> level0.dcterms_identifier)
```

Listing B.80: SQL Query (PostgreSQL dialect) for Query Scenario GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY.

```
drop MATERIALIZED view if exists unnestedsubjects;

create MATERIALIZED view unnestedsubjects as
(SELECT dcterms_identifier, unnest(dcterms_subject) subject FROM justatable where dcterms_subject
      <> '{}');

CREATE INDEX unnestedsubjects_identifer ON unnestedsubjects (dcterms_identifier);
CREATE INDEX unnestedsubjects_subject ON unnestedsubjects (subject);


-- Old gin index - was never really used.
-- DROP INDEX IF EXISTS idx_identifier;
-- CREATE INDEX idx_identifier on "${Config.TABLE}" ("dcterms_identifier");
-- DROP INDEX IF EXISTS idx_subjects;
-- CREATE INDEX idx_subjects on "${Config.TABLE}" USING GIN ("dcterms_subject");
```

Listing B.81: SQL Query (PostgreSQL dialect) for Query Scenario GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY_prepare.

```
select level0.dcterms_identifier, level0.subject, level1.dcterms_identifier, level1.subject,
     level2.dcterms_identifier
from unnestedsubjects level0
inner join unnestedsubjects level1 on (level1.subject = level0.subject and level1.
     dcterms_identifier <> level0.dcterms_identifier)
inner join unnestedsubjects level2 on (level2.subject = level1.subject and level2.
     dcterms_identifier <> level1.dcterms_identifier and level1.dcterms_identifier <> level0.
     dcterms_identifier)
```

Listing B.82: SQL Query (PostgreSQL dialect) for Query Scenario GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_2HOPS_100_ENTITIES.

```
drop MATERIALIZED view if exists unnestedsubjects;

create MATERIALIZED view unnestedsubjects as
(SELECT dcterms_identifier, unnest(dcterms_subject) subject FROM justatable where dcterms_subject
      <> '{}');

CREATE INDEX unnestedsubjects_identifer ON unnestedsubjects (dcterms_identifier);
CREATE INDEX unnestedsubjects_subject ON unnestedsubjects (subject);


-- Old gin index - was never really used.
-- DROP INDEX IF EXISTS idx_identifier;
-- CREATE INDEX idx_identifier on "${Config.TABLE}" ("dcterms_identifier");
-- DROP INDEX IF EXISTS idx_subjects;
-- CREATE INDEX idx_subjects on "${Config.TABLE}" USING GIN ("dcterms_subject");
```

Listing B.83: SQL Query (PostgreSQL dialect) for Query Scenario GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_2HOPS_100_ENTITIES_prepare.

```
select level0.dcterms_identifier, level0.subject, level1.dcterms_identifier, level1.subject,
    level2.dcterms_identifier
from unnestedsubjects level0
inner join unnestedsubjects level1 on (level1.subject = level0.subject and level1.
    dcterms_identifier <> level0.dcterms_identifier)
inner join unnestedsubjects level2 on (level2.subject = level1.subject and level2.
    dcterms_identifier <> level1.dcterms_identifier and level1.dcterms_identifier <> level0.
    dcterms_identifier)
```

Listing B.84: SQL Query (PostgreSQL dialect) for Query Scenario GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_2HOPS_10_ENTITIES.

```
drop MATERIALIZED view if exists unnestedsubjects;

create MATERIALIZED view unnestedsubjects as
(SELECT dcterms_identifier, unnest(dcterms_subject) subject FROM justatable where dcterms_subject
     <> '{}');

CREATE INDEX unnestedsubjects_identifer ON unnestedsubjects (dcterms_identifier);
CREATE INDEX unnestedsubjects_subject ON unnestedsubjects (subject);


-- Old gin index - was never really used.
-- DROP INDEX IF EXISTS idx_identifier;
-- CREATE INDEX idx_identifier on "${Config.TABLE}" ("dcterms_identifier");
-- DROP INDEX IF EXISTS idx_subjects;
-- CREATE INDEX idx_subjects on "${Config.TABLE}" USING GIN ("dcterms_subject");
```

Listing B.85: SQL Query (PostgreSQL dialect) for Query Scenario GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_2HOPS_10_ENTITIES_prepare.

```
select level0.dcterms_identifier, level0.subject, level1.dcterms_identifier, level1.subject,
    level2.dcterms_identifier
from unnestedsubjects level0
inner join unnestedsubjects level1 on (level1.subject = level0.subject and level1.
    dcterms_identifier <> level0.dcterms_identifier)
inner join unnestedsubjects level2 on (level2.subject = level1.subject and level2.
    dcterms_identifier <> level1.dcterms_identifier and level1.dcterms_identifier <> level0.
    dcterms_identifier)
```

Listing B.86: SQL Query (PostgreSQL dialect) for Query Scenario GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_2HOPS_ONE_ENTITY.

```
drop MATERIALIZED view if exists unnestedsubjects;

create MATERIALIZED view unnestedsubjects as
(SELECT dcterms_identifier, unnest(dcterms_subject) subject FROM justatable where dcterms_subject
     <> '{}');

CREATE INDEX unnestedsubjects_identifer ON unnestedsubjects (dcterms_identifier);
CREATE INDEX unnestedsubjects_subject ON unnestedsubjects (subject);


-- Old gin index - was never really used.
-- DROP INDEX IF EXISTS idx_identifier;
-- CREATE INDEX idx_identifier on "${Config.TABLE}" ("dcterms_identifier");
-- DROP INDEX IF EXISTS idx_subjects;
-- CREATE INDEX idx_subjects on "${Config.TABLE}" USING GIN ("dcterms_subject");
```

Listing B.87: SQL Query (PostgreSQL dialect) for Query Scenario GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_2HOPS_ONE_ENTITY_prepare.

```
alter table justatable add column inewpop text default 'cheesecake'
```

Listing B.88: SQL Query (PostgreSQL dialect) for Query Scenario SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY.

```
alter table justatable
add column istrop text;

update justatable set istrop=substring(RDF_ABOUT from '........$');
```

Listing B.89: SQL Query (PostgreSQL dialect) for Query Scenario SCHEMA_CHANGE_INTRODUCE_STRING_OP.

```
drop index if exists RDF_ABOUT_idx;
CREATE INDEX RDF_ABOUT_idx ON justatable (RDF_ABOUT);
```

Listing B.90: SQL Query (PostgreSQL dialect) for Query Scenario SCHEMA_CHANGE_INTRODUCE_STRING_OP_prepare.

```
alter table justatable add column manifestation boolean default false;
update justatable set manifestation = true where 'http://purl.org/vocab/frbr/core#Manifestation'
    = ANY(RDF_TYPE);

alter table justatable add column bibliographicresource boolean default false;
update justatable set bibliographicresource = true where 'http://purl.org/dc/terms/
    BibliographicResource' = ANY(RDF_TYPE);

alter table justatable add column book boolean default false;
update justatable set book = true where 'http://purl.org/ontology/bibo/Book' = ANY(RDF_TYPE);
```

Listing B.91: SQL Query (PostgreSQL dialect) for Query Scenario SCHEMA_CHANGE_MIGRATE_RDF_TYPE.

```
DROP INDEX IF EXISTS RDF_TYPE_idx;
CREATE INDEX RDF_TYPE_idx on justatable USING GIN ("rdf_type");
```

Listing B.92: SQL Query (PostgreSQL dialect) for Query Scenario SCHEMA_CHANGE_MIGRATE_RDF_TYPE_prepare.

```
alter table justatable drop column RDF_TYPE;
```

Listing B.93: SQL Query (PostgreSQL dialect) for Query Scenario SCHEMA_CHANGE_REMOVE_RDF_TYPE.

```
update justatable set dcterms_issued = '0' where dcterms_issued is null
```

Listing B.94: SQL Query (PostgreSQL dialect) for Query Scenario UPDATE_HIGH_SELECTIVITY_NON_ISSUED.

```
DROP INDEX IF EXISTS issued_idx;
CREATE INDEX issued_idx on justatable ("dcterms_issued");
```

Listing B.95: SQL Query (PostgreSQL dialect) for Query Scenario UPDATE_HIGH_SELECTIVITY_NON_ISSUED_prepare.

```
update justatable set DCTERMS_MEDIUM = 'recycled trees' where DCTERMS_MEDIUM = 'paper'
```

Listing B.96: SQL Query (PostgreSQL dialect) for Query Scenario UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM.

```
drop index if exists DCTERMS_MEDIUM_idx;
CREATE INDEX DCTERMS_MEDIUM_idx ON justatable (DCTERMS_MEDIUM);
```

Listing B.97: SQL Query (PostgreSQL dialect) for Query Scenario UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM_prepare.

B.5 Virtuoso Benchmark Implementation

```
package database;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;
import java.util.zip.GZIPInputStream;

import com.google.common.base.Joiner;

import util.Codes;
import util.Config;
import util.DataObject;
import util.Dataset;
import util.QueryResult;
import util.QueryScenario;
import util.Templates;
import util.dumper.Helpers;

public class Virtuoso implements Database {

 public static void main(String[] args) throws Throwable {
  Database database = new Virtuoso();
  database.start();
  database.clean();
  database.setUp();
  database.load(Dataset.hebis_100000_records);

  QueryScenario queryScenario = QueryScenario.ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY;
  queryScenario = QueryScenario.GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY;
  database.prepare(queryScenario);
  QueryResult queryResult = database.query(queryScenario);
  System.out.println(queryResult);

  queryScenario = QueryScenario.ENTITY_RETRIEVAL_BY_ID_100_ENTITIES;
  queryScenario = QueryScenario.GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES;
  database.prepare(queryScenario);
  queryResult = database.query(queryScenario);
  System.out.println(queryResult);

 }

 private String graphId, vAD, vADRtVE;
 private Connection connection;
 private Statement stmt;
 private Templates templates;
 private ArrayList<PreparedStatement> scenarioStatements;
 private QueryScenario queryScenario;
 private String queryString;

 public Virtuoso() {
  graphId = Config.DATABASE;
  vAD = Config.VIRTUOSO_ACCESSIBLE_DIRECTORY;
  vADRtVE = Config.VIRTUOSO_ACCESSIBLE_DIRECTORY_RELATIVE_TO_VIRTUOSO_T_EXE;

  templates = new Templates("virtuoso", ".sql");
 }

 @Override
 public String getName() {
  return "Virtuoso";
 }

 @Override
 public String getVersion() {
  return "07.20.3214 / Virtuoso JDBC 4.1";
 }

 @Override
 public void setUp() throws Exception {
  connection = DriverManager.getConnection("jdbc:virtuoso://127.0.0.1/CHARSET=UTF-8", "dba", "dba
```

```
      ");
 stmt = connection.createStatement();
 // Load RDF Loader - this might fail
 try {
  stmt.executeQuery("LOAD " + new File(this.getClass().getResource("/queries/virtuoso/rdfloader.
       sql").getFile()).getAbsolutePath() + ";");
 } catch (Exception e) {

 }
}

private String GetInsertStatement() {
 String insertStatement = "sparql\n";
 insertStatement += "PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>\n";
 insertStatement += "PREFIX picaplus:<http://lod.hebis.uni-frankfurt.de/daten/terms/>\n";
 insertStatement += "PREFIX dbpedia:<http://localhost:8080/resource/>\n";
 insertStatement += "PREFIX skos:<http://www.w3.org/2004/02/skos/core#>\n";
 insertStatement += "PREFIX rel:<http://opendata.hbz-nrw.de/rel/1.0/>\n";
 insertStatement += "PREFIX geonames:<http://www.geonames.org/ontology#>\n";
 insertStatement += "PREFIX yago:<http://localhost:8080/class/yago/>\n";
 insertStatement += "PREFIX units:<http://dbpedia.org/units/>\n";
 insertStatement += "PREFIX p:<http://localhost:8080/property/>\n";
 insertStatement += "PREFIX auth:<http://opendata.hbz-nrw.de/auth/1.0/>\n";
 insertStatement += "PREFIX prvTypes:<http://purl.org/net/provenance/types#>\n";
 insertStatement += "PREFIX bibo:<http://purl.org/ontology/bibo/>\n";
 insertStatement += "PREFIX foaf:<http://xmlns.com/foaf/0.1/>\n";
 insertStatement += "PREFIX j.0:<http://spinrdf.org/sp#>\n";
 insertStatement += "PREFIX frbr:<http://purl.org/vocab/frbr/core#>\n";
 insertStatement += "PREFIX owl:<http://www.w3.org/2002/07/owl#>\n";
 insertStatement += "PREFIX dcterms:<http://purl.org/dc/terms/>\n";
 insertStatement += "PREFIX meta:<http://example.org/metadata#>\n";
 insertStatement += "PREFIX xsd:<http://www.w3.org/2001/XMLSchema#>\n";
 insertStatement += "PREFIX j.1:<http://www.w3.org/2004/03/trix/rdfg-1/>\n";
 insertStatement += "PREFIX wdrs:<http://www.w3.org/2007/05/powder-s#>\n";
 insertStatement += "PREFIX prv:<http://purl.org/net/provenance/ns#>\n";
 insertStatement += "PREFIX isbd:<http://iflastandards.info/ns/isbd/elements/>\n";
 insertStatement += "PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#>\n";
 insertStatement += "PREFIX geo:<http://www.w3.org/2003/01/geo/wgs84_pos#>\n";
 insertStatement += "PREFIX rda:<http://RDVocab.info/Elements/>\n";
 // insertStatement += String.format("INSERT DATA { GRAPH <%s> {%s %s %s}
 // }", /*graphId*/"MYINSERTTESTS");
 insertStatement += "INSERT DATA { GRAPH <%s> {<%s> %s %s} }";
 return insertStatement;
}

@Override
public void load(Dataset dataset) throws Exception {
 PreparedStatement rdfDumpLoadInstruction = connection.prepareStatement("ld_dir(?, ?, ?)");
 rdfDumpLoadInstruction.setString(1, dataset.file.getParentFile().getAbsolutePath());
 rdfDumpLoadInstruction.setString(2, dataset.file.getName());
 rdfDumpLoadInstruction.setString(3, Config.DATABASE);

 rdfDumpLoadInstruction.executeQuery();
 connection.createStatement().executeQuery("rdf_loader_run()");
}

public void theoldload(Dataset dataset) throws Exception {

 // DIE KLEINEN DATEIEN
 if (dataset == Dataset.hebis_10147116_13050073_rdf_gz || dataset == Dataset.hebis_1000_records
      || dataset == Dataset.hebis_10000_records
   || dataset == Dataset.hebis_100000_records) {
  // 1. UnGzip to virtuosoAccessibleDir
  // 2. Load Data into Graph
  // 1.
  byte[] buffer = new byte[1024];
  GZIPInputStream gzis = new GZIPInputStream(new FileInputStream(dataset.string));
  FileOutputStream out = new FileOutputStream(String.format("%s//%s", vAD, graphId));
  int len;
  while ((len = gzis.read(buffer)) > 0) {
   out.write(buffer, 0, len);
  }
  gzis.close();
  out.close();
  // 2.
  stmt.executeQuery(String.format("DB.DBA.RDF_LOAD_RDFXML_MT (file_to_string_output ('%s/%s'),
       '', '%s')", vADRtVE, graphId, graphId));
```

```
 return;
}


// PreparedStatement pstmt =
// connection.prepareStatement(insertStatement);
Statement statement = connection.createStatement();

// Load DataObjects
boolean autoCommitBackup = connection.getAutoCommit();
connection.setAutoCommit(false);
Helpers.readRdf(dataset, dataObject -> {
 try {
  for (int i = 0; i < Codes.values().length; i++) {

   Codes code = Codes.values()[i];
   if (code != Codes.RDF_ABOUT && dataObject.get(code) != null) {
    String subject = dataObject.get(Codes.RDF_ABOUT).toString(), predicate, object;

    if (code.rdfProperty.contains(" "))
     predicate = code.rdfProperty.substring(0, code.rdfProperty.indexOf(" ")); // "wdrs:
          describedby
                     // rdf:resource"
                     // =>
                     // "wdrs:describedby"
    else
     predicate = code.rdfProperty;

    if (code.IS_MULTIPLE) {
     List<String> entries = (ArrayList<String>) dataObject.get(code);
     for (String string : entries) {
      String achehrlich = string;

      // Ach, ehrlich ...
      achehrlich = achehrlich.replace('"', '\'');
      achehrlich = achehrlich.replace('\\', '-');

      String objectVal = code.attributeValue ? "<%s>" : "\"%s\"";
      object = String.format(objectVal, achehrlich);
      String insertStatement = GetInsertStatement();
      String update = String.format(insertStatement, graphId, subject, predicate, object);
      statement.executeUpdate(update);
     }
    } else {
     String achehrlich = dataObject.get(code).toString();

     // Ach, ehrlich ...
     achehrlich = achehrlich.replace('"', '\'');
     achehrlich = achehrlich.replace('\\', '-');

     String objectVal = code.attributeValue ? "<%s>" : "\"%s\"";
     object = String.format(objectVal, achehrlich);
     String insertStatement = GetInsertStatement();
     String update = String.format(insertStatement, graphId, subject, predicate, object);
     statement.executeUpdate(update);
    }

    // pstmt.setString(1, subject);
    // pstmt.setString(2, predicate);
    // pstmt.setString(3, object);
    // //System.out.println(dataObject);
    //
    // pstmt.execute();
    // pstmt.clearParameters();
   }
  }
 } catch (Exception e) {
  throw new RuntimeException("Cannot insert DataObject:\n" + dataObject, e);
 }
} , counter -> {
 if (counter % 50000 == 0) {
  System.out.println(counter + " records so far... (Virtuoso Load())");
  try {
   connection.commit();
  } catch (Exception e) {
   throw new RuntimeException("Commit gone wrong");
```

```
   }
  }
 });
 connection.commit();
 connection.setAutoCommit(autoCommitBackup);
}

@Override
public void prepare(QueryScenario queryScenario) throws Exception {
 if (scenarioStatements == null)
  scenarioStatements = new ArrayList<>();

 scenarioStatements.clear();

 switch (queryScenario) {
 case ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY:
 case ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES:
 case ENTITY_RETRIEVAL_BY_ID_100_ENTITIES: {
  String query = templates.resolve("ENTITY_RETRIEVAL_prepare");
  switch (queryScenario) {
  case ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY:
   query += " limit 1";
   break;
  case ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES:
   query += " limit 10";
   break;
  case ENTITY_RETRIEVAL_BY_ID_100_ENTITIES:
   query += " limit 100";
   break;
  default:
  }

  ResultSet resultSet = connection.createStatement().executeQuery("sparql " + query);
  ArrayList<String> ids = new ArrayList<>();
  while (resultSet.next()) {
   ids.add("\"" + resultSet.getString(1) + "\"");
  }


  scenarioStatements.add(connection.prepareStatement("sparql select ?a where {?a ?b ?c } limit 1"
      ));
  this.queryString = "sparql " + templates.resolve("ENTITY_RETRIEVAL").replaceAll("##ids##",
      Joiner.on(",").join(ids));
  break;
 }
 case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY:
 case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES:
 case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES:
 // case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_2HOPS_ONE_ENTITY:
 // case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_2HOPS_10_ENTITIES:
 // case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_2HOPS_100_ENTITIES:
 {
  String query = templates.resolve("GRAPH_LIKE_prepare");
  switch (queryScenario) {
  // case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECCTS_2HOPS_100_ENTITIES:
  case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES:
   query += " limit 100";
   break;
  case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES:
   // case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_2HOPS_10_ENTITIES:
   query += " limit 10";
   break;
  case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY:
   // case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECCTS_2HOPS_ONE_ENTITY:
   query += " limit 1";
   break;
  default:
   throw new RuntimeException("Dont know how to limit " + queryScenario);
  }
  ResultSet resultSet = connection.createStatement().executeQuery("sparql " + query);
  ArrayList<String> ids = new ArrayList<>();
  while (resultSet.next()) {
   ids.add("\"" + resultSet.getString(1) + "\"");
  }

  String templateName = "GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_2HOPS";
  switch (queryScenario) {
```

```
  case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES:
  case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES:
  case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY:
   templateName = "GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP";
   break;
  }

  scenarioStatements.add(connection.prepareStatement("sparql select ?a where {?a ?b ?c } limit 1"
       ));
  this.queryString = "sparql " + templates.resolve(templateName).replaceAll("##ids##", Joiner.on(
       ",").join(ids));
  break;
 }
 case SCHEMA_CHANGE_MIGRATE_RDF_TYPE:
  for (int part = 1; part <= 4; part++)
   scenarioStatements.add(connection.prepareStatement("sparql " + String.format(templates.resolve
        (queryScenario + "_part" + part), graphId)));
  break;
 default:
  scenarioStatements.add(connection.prepareStatement("sparql " + String.format(templates.resolve(
       queryScenario), graphId)));
 }

 this.queryScenario = queryScenario;
}

@Override
public QueryResult query(QueryScenario queryScenario) throws Exception {
 // scenarioStatements.add(connection.prepareStatement(templates.resolve(queryScenario)));

 // stmt.executeQuery(stmt.execute(String.format(templates.resolve(queryScenario),
 // graphId))
 QueryResult queryResult = new QueryResult(queryScenario.queryResultType);
 for (PreparedStatement preparedStatement : scenarioStatements) {
  ResultSet resultSet;
  // System.out.println(preparedStatement);
  switch (queryScenario.queryResultType) {
  case GRAPH:
   preparedStatement = connection.prepareStatement(queryString);
   resultSet = preparedStatement.executeQuery();
   while (resultSet.next()) {
    switch (resultSet.getMetaData().getColumnCount()) {
    case 3:
     queryResult.push(resultSet.getString(1), resultSet.getString(2), resultSet.getString(3));
     break;
    case 5:
     queryResult.push(resultSet.getString(1), resultSet.getString(2), resultSet.getString(3),
          resultSet.getString(4),
       resultSet.getString(5));
     break;
    default:
     throw new RuntimeException("Cannot parse " + resultSet.getMetaData().getColumnCount() + "
          columns in result set");
    }
   }
   break;
  case TWO_COLUMNS:
   resultSet = preparedStatement.executeQuery();
   while (resultSet.next()) {
    queryResult.push(resultSet.getString(1), resultSet.getString(2));
   }
   break;
  case COMPLETE_ENTITIES:
   preparedStatement = connection.prepareStatement(queryString);
   resultSet = preparedStatement.executeQuery();

   while (resultSet.next()) {
    DataObject dataObject = new DataObject();
    for (int i = 0; i < Codes.values().length; i++) {
     Codes code = Codes.values()[i];
     String fieldValue = resultSet.getString(i + 1);
     if (fieldValue == null) {
      dataObject.set(code, null);
      continue;
     }

     if (code.IS_MULTIPLE) {
```

```
      if (!fieldValue.contains(",")) {
       // Code is defined to be multiple put field only
       // contains a single value
       dataObject.putMultiple(code, fieldValue);
      } else {
       // Assemble the SPARQL-group-concat expressions
       // - workaround for missing efficient describe
       for (String fieldValueIteration : fieldValue.split(",")) {
        dataObject.putMultiple(code, fieldValueIteration);
       }
      }
     } else {
      // Straight forward
      dataObject.set(code, fieldValue);
     }
    }
    queryResult.push(dataObject);
   }
   break;
  case NONE:
  default:
   // This will only work for non-readOnly-scenarios!
   preparedStatement.executeUpdate();
   break;
  }
 }
 return queryResult;
}

@Override
public String toString() {
 return "Virtuoso [getName()=" + getName() + ", getVersion()=" + getVersion() + "]";
}

@Override
public void clean() throws Exception {
 connection = DriverManager.getConnection("jdbc:virtuoso://127.0.0.1/CHARSET=UTF-8", "dba", "dba
      ");
 stmt = connection.createStatement();
 // Drop auf evtl. alten Identifier
 stmt.execute(String.format("SPARQL CLEAR GRAPH <%s>", graphId));
}

@Override
public void start() {
 // Helpers.terminalLaunchScript("virtuoso.sh", 90);
}

@Override
public void stop() {
 // if (Config.THIS_IS_OSX)
 // Helpers.terminalLaunchScript("virtuoso_stop.sh", 10);
}

}
```

Listing B.98: Java Source Code of the Virtuoso Benchmark Implementation.

```
select ?decade (count(*) as ?count) from <loddwhbench>
where
{
 ?a <http://purl.org/dc/terms/issued> ?century
}
group by (substr(?century,1,3) as ?decade)
order by desc(?count) ?decade
```

Listing B.99: SPARQL Query (Virtuoso) for Query Scenario AGGREGATE_ISSUES_PER_DECADE_ALL.

```
select ?decade (count(*) as ?count) from <loddwhbench>
where
{
 ?a <http://purl.org/dc/terms/issued> ?century
}
group by (substr(?century,1,3) as ?decade)
```

```
order by desc(?count) ?decade
limit 10
```

Listing B.100: SPARQL Query (Virtuoso) for Query Scenario AGGREGATE_ISSUES_PER_DECADE_TOP10.

```
select ?decade (count(*) as ?count) from <loddwhbench>
where
{
 ?a <http://purl.org/dc/terms/issued> ?century
}
group by (substr(?century,1,3) as ?decade)
order by desc(?count) ?decade
limit 100
```

Listing B.101: SPARQL Query (Virtuoso) for Query Scenario AGGREGATE_ISSUES_PER_DECADE_TOP100.

```
select ?publisher (count(?publisher) as ?count) from <loddwhbench>
where
{
 ?a <http://purl.org/dc/terms/publisher> ?publisher .
}
group by ?publisher
order by desc(?count) ?publisher
```

Listing B.102: SPARQL Query (Virtuoso) for Query Scenario AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL.

```
select ?publisher (count(?publisher) as ?count) from <loddwhbench>
where
{
 ?a <http://purl.org/dc/terms/publisher> ?publisher .
}
group by ?publisher
order by desc(?count) ?publisher
limit 10
```

Listing B.103: SPARQL Query (Virtuoso) for Query Scenario AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10.

```
select ?publisher (count(?publisher) as ?count) from <loddwhbench>
where
{
 ?a <http://purl.org/dc/terms/publisher> ?publisher .
}
group by ?publisher
order by desc(?count) ?publisher
limit 100
```

Listing B.104: SPARQL Query (Virtuoso) for Query Scenario AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100.

```
select
 ?s
 (group_concat(distinct ?edition ; separator = ",") as ?edition)
 ?oclcum
 ?format
 ?identifier
 ?issued
 ?medium
 ?publisher
 (group_concat(distinct ?subject ; separator = ",") as ?subject)
 (group_concat(distinct ?title ; separator = ",") as ?title)
 (group_concat(distinct ?contributor ; separator = ",") as ?contributor)
 (group_concat(distinct ?P1004 ; separator = ",") as ?P1004)
 ?P1006
 (group_concat(distinct ?P1008 ; separator = ",") as ?P1008)
 (group_concat(distinct ?P1016 ; separator = ",") as ?P1016)
 ?P1017
 ?P1018
 ?sameAs
```

```
 (group_concat(distinct ?type ; separator = ",") as ?type)
 ?describedby
from <loddwhbench>
where
{
 ?s ?o ?p .
 optional { ?s <http://purl.org/ontology/bibo/edition> ?edition . }
 optional { ?s <http://purl.org/ontology/bibo/oclcnum> ?oclcnum . }
 optional { ?s <http://purl.org/dc/terms/format> ?format . }
 optional { ?s <http://purl.org/dc/terms/identifier> ?identifier . }
 optional { ?s <http://purl.org/dc/terms/issued> ?issued . }
 optional { ?s <http://purl.org/dc/terms/medium> ?medium . }
 optional { ?s <http://purl.org/dc/terms/publisher> ?publisher . }
 optional { ?s <http://purl.org/dc/terms/subject> ?subject . }
 optional { ?s <http://purl.org/dc/terms/title> ?title . }
 optional { ?s <http://purl.org/dc/terms/contributor> ?contributor . }
 optional { ?s <http://iflastandards.info/ns/isbd/elements/P1004> ?P1004 . }
 optional { ?s <http://iflastandards.info/ns/isbd/elements/P1006> ?P1006 . }
 optional { ?s <http://iflastandards.info/ns/isbd/elements/P1008> ?P1008 . }
 optional { ?s <http://iflastandards.info/ns/isbd/elements/P1016> ?P1016 . }
 optional { ?s <http://iflastandards.info/ns/isbd/elements/P1017> ?P1017 . }
 optional { ?s <http://iflastandards.info/ns/isbd/elements/P1018> ?P1018 . }
 optional { ?s <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> ?type . }
 optional { ?s <http://www.w3.org/2002/07/owl#sameAs> ?sameAs . }
 optional { ?s <http://www.w3.org/2007/05/powder-s#describedby> ?describedby . }

 ?s <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://purl.org/dc/terms/
     BibliographicResource> .
}
```

Listing B.105: SPARQL Query (Virtuoso) for Query Scenario CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES.

```
select
 ?s
 (group_concat(distinct ?edition ; separator = ",") as ?edition)
 ?oclcum
 ?format
 ?identifier
 ?issued
 ?medium
 ?publisher
 (group_concat(distinct ?subject ; separator = ",") as ?subject)
 (group_concat(distinct ?title ; separator = ",") as ?title)
 (group_concat(distinct ?contributor ; separator = ",") as ?contributor)
 (group_concat(distinct ?P1004 ; separator = ",") as ?P1004)
 ?P1006
 (group_concat(distinct ?P1008 ; separator = ",") as ?P1008)
 (group_concat(distinct ?P1016 ; separator = ",") as ?P1016)
 ?P1017
 ?P1018
 ?sameAs
 (group_concat(distinct ?type ; separator = ",") as ?type)
 ?describedby
from <loddwhbench>
where
{
 ?s ?o ?p .
 optional { ?s <http://purl.org/ontology/bibo/edition> ?edition . }
 optional { ?s <http://purl.org/ontology/bibo/oclcnum> ?oclcnum . }
 optional { ?s <http://purl.org/dc/terms/format> ?format . }
 optional { ?s <http://purl.org/dc/terms/identifier> ?identifier . }
 optional { ?s <http://purl.org/dc/terms/issued> ?issued . }
 optional { ?s <http://purl.org/dc/terms/medium> ?medium . }
 optional { ?s <http://purl.org/dc/terms/publisher> ?publisher . }
 optional { ?s <http://purl.org/dc/terms/subject> ?subject . }
 optional { ?s <http://purl.org/dc/terms/title> ?title . }
 optional { ?s <http://purl.org/dc/terms/contributor> ?contributor . }
 optional { ?s <http://iflastandards.info/ns/isbd/elements/P1004> ?P1004 . }
 optional { ?s <http://iflastandards.info/ns/isbd/elements/P1006> ?P1006 . }
 optional { ?s <http://iflastandards.info/ns/isbd/elements/P1008> ?P1008 . }
 optional { ?s <http://iflastandards.info/ns/isbd/elements/P1016> ?P1016 . }
 optional { ?s <http://iflastandards.info/ns/isbd/elements/P1017> ?P1017 . }
 optional { ?s <http://iflastandards.info/ns/isbd/elements/P1018> ?P1018 . }
 optional { ?s <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> ?type . }
```

```
 optional { ?s <http://www.w3.org/2002/07/owl#sameAs> ?sameAs . }
 optional { ?s <http://www.w3.org/2007/05/powder-s#describedby> ?describedby . }

 filter regex(?title, 'stud(ie|y)', 'i') .
 ?s <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://purl.org/dc/terms/
     BibliographicResource> .
}
```

Listing B.106: SPARQL Query (Virtuoso) for Query Scenario CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES.

```
select
 ?s
 (group_concat(distinct ?edition ; separator = ",") as ?edition)
 ?oclcum
 ?format
 ?identifier
 ?issued
 ?medium
 ?publisher
 (group_concat(distinct ?subject ; separator = ",") as ?subject)
 (group_concat(distinct ?title ; separator = ",") as ?title)
 (group_concat(distinct ?contributor ; separator = ",") as ?contributor)
 (group_concat(distinct ?P1004 ; separator = ",") as ?P1004)
 ?P1006
 (group_concat(distinct ?P1008 ; separator = ",") as ?P1008)
 (group_concat(distinct ?P1016 ; separator = ",") as ?P1016)
 ?P1017
 ?P1018
 ?sameAs
 (group_concat(distinct ?type ; separator = ",") as ?type)
 ?describedby
from <loddwhbench>
where
{
 ?s ?o ?p .
 optional { ?s <http://purl.org/ontology/bibo/edition> ?edition . }
 optional { ?s <http://purl.org/ontology/bibo/oclcnum> ?oclcnum . }
 optional { ?s <http://purl.org/dc/terms/format> ?format . }
 optional { ?s <http://purl.org/dc/terms/identifier> ?identifier . }
 optional { ?s <http://purl.org/dc/terms/issued> ?issued . }
 optional { ?s <http://purl.org/dc/terms/medium> ?medium . }
 optional { ?s <http://purl.org/dc/terms/publisher> ?publisher . }
 optional { ?s <http://purl.org/dc/terms/subject> ?subject . }
 optional { ?s <http://purl.org/dc/terms/title> ?title . }
 optional { ?s <http://purl.org/dc/terms/contributor> ?contributor . }
 optional { ?s <http://iflastandards.info/ns/isbd/elements/P1004> ?P1004 . }
 optional { ?s <http://iflastandards.info/ns/isbd/elements/P1006> ?P1006 . }
 optional { ?s <http://iflastandards.info/ns/isbd/elements/P1008> ?P1008 . }
 optional { ?s <http://iflastandards.info/ns/isbd/elements/P1016> ?P1016 . }
 optional { ?s <http://iflastandards.info/ns/isbd/elements/P1017> ?P1017 . }
 optional { ?s <http://iflastandards.info/ns/isbd/elements/P1018> ?P1018 . }
 optional { ?s <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> ?type . }
 optional { ?s <http://www.w3.org/2002/07/owl#sameAs> ?sameAs . }
 optional { ?s <http://www.w3.org/2007/05/powder-s#describedby> ?describedby . }

 filter regex(?title, 'stud(ie|y)', 'i') .
}
```

Listing B.107: SPARQL Query (Virtuoso) for Query Scenario CONDITIONAL_TABLE_SCAN_ALL_STUDIES.

```
WITH <loddwhbench>
DELETE
 { ?s ?p ?o }
WHERE {
  ?s ?p ?o .
  ?s <http://purl.org/dc/terms/issued> '0' .
}
```

Listing B.108: SPARQL Query (Virtuoso) for Query Scenario DELETE_HIGH_SELECTIVIY_NON_ISSUED.

```
WITH <loddwhbench>
DELETE
 {
  ?s ?p ?o
 }
WHERE
 {
  ?s ?p ?o ;
   <http://purl.org/dc/terms/medium> 'recycled trees'
 }
```

Listing B.109: SPARQL Query (Virtuoso) for Query Scenario DELETE_LOW_SELECTIVITY_PAPER_MEDIUM.

```
select
 ?s
 (group_concat(distinct ?edition ; separator = ",") as ?edition)
 ?oclcum
 ?format
 ?identifier
 ?issued
 ?medium
 ?publisher
 (group_concat(distinct ?subject ; separator = ",") as ?subject)
 (group_concat(distinct ?title ; separator = ",") as ?title)
 (group_concat(distinct ?contributor ; separator = ",") as ?contributor)
 (group_concat(distinct ?P1004 ; separator = ",") as ?P1004)
 ?P1006
 (group_concat(distinct ?P1008 ; separator = ",") as ?P1008)
 (group_concat(distinct ?P1016 ; separator = ",") as ?P1016)
 ?P1017
 ?P1018
 ?sameAs
 (group_concat(distinct ?type ; separator = ",") as ?type)
 ?describedby
from <loddwhbench>
where
{
 ?s ?o ?p .
 optional { ?s <http://purl.org/ontology/bibo/edition> ?edition . }
 optional { ?s <http://purl.org/ontology/bibo/oclcnum> ?oclcnum . }
 optional { ?s <http://purl.org/dc/terms/format> ?format . }
 optional { ?s <http://purl.org/dc/terms/issued> ?issued . }
 optional { ?s <http://purl.org/dc/terms/medium> ?medium . }
 optional { ?s <http://purl.org/dc/terms/publisher> ?publisher . }
 optional { ?s <http://purl.org/dc/terms/subject> ?subject . }
 optional { ?s <http://purl.org/dc/terms/title> ?title . }
 optional { ?s <http://purl.org/dc/terms/contributor> ?contributor . }
 optional { ?s <http://iflastandards.info/ns/isbd/elements/P1004> ?P1004 . }
 optional { ?s <http://iflastandards.info/ns/isbd/elements/P1006> ?P1006 . }
 optional { ?s <http://iflastandards.info/ns/isbd/elements/P1008> ?P1008 . }
 optional { ?s <http://iflastandards.info/ns/isbd/elements/P1016> ?P1016 . }
 optional { ?s <http://iflastandards.info/ns/isbd/elements/P1017> ?P1017 . }
 optional { ?s <http://iflastandards.info/ns/isbd/elements/P1018> ?P1018 . }
 optional { ?s <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> ?type . }
 optional { ?s <http://www.w3.org/2002/07/owl#sameAs> ?sameAs . }
 optional { ?s <http://www.w3.org/2007/05/powder-s#describedby> ?describedby . }

 ?s <http://purl.org/dc/terms/identifier> ?identifier .
 FILTER( ?identifier IN ( ##ids## ))
}
```

Listing B.110: SPARQL Query (Virtuoso) for Query Scenario ENTITY_RETRIEVAL.

```
select ?identifier
where
{
 ?s <http://purl.org/dc/terms/identifier> ?identifier .
 optional { ?s <http://purl.org/dc/terms/medium> ?medium . }
 optional { ?s <http://iflastandards.info/ns/isbd/elements/P1008> ?P1008 . }
 optional { ?s <http://purl.org/dc/terms/contributor> ?contributor . }
 optional { ?s <http://purl.org/dc/terms/issued> ?issued . }
}
```

```
order by ?medium ?P1008 ?contributor ?issued ?identifier
```

Listing B.111: SPARQL Query (Virtuoso) for Query Scenario ENTITY_RETRIEVAL_prepare.

```
select ?identifier
where
{
 ?s <http://purl.org/dc/terms/identifier> ?identifier .
 ?s <http://purl.org/dc/terms/subject> ?subject .
 optional { ?s <http://purl.org/dc/terms/medium> ?medium . }
 optional { ?s <http://iflastandards.info/ns/isbd/elements/P1008> ?P1008 . }
 optional { ?s <http://purl.org/dc/terms/contributor> ?contributor . }
 optional { ?s <http://purl.org/dc/terms/issued> ?issued . }
}
order by ?medium ?P1008 ?contributor ?issued ?identifier
```

Listing B.112: SPARQL Query (Virtuoso) for Query Scenario GRAPH_LIKE_prepare.

```
select ?orig_id ?orig_subj ?related_id from <loddwhbench>
where
{
 ?s1 <http://purl.org/dc/terms/identifier> ?orig_id .
 ?s1 <http://purl.org/dc/terms/subject> ?orig_subj .
 ?s2 <http://purl.org/dc/terms/identifier> ?related_id .
 ?s2 <http://purl.org/dc/terms/subject> ?orig_subj .
 filter( ?orig_id != ?related_id )
 filter( ?orig_id in (##ids##) )
}
```

Listing B.113: SPARQL Query (Virtuoso) for Query Scenario GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP.

```
select ?orig_id ?orig_subj ?related_id ?secondRelated_id from <loddwhbench>
where
{
 ?s1 <http://purl.org/dc/terms/identifier> ?orig_id .
 ?s1 <http://purl.org/dc/terms/subject> ?orig_subj .

 ?s2 <http://purl.org/dc/terms/identifier> ?related_id .
 ?s2 <http://purl.org/dc/terms/subject> ?orig_subj .
 ?s2 <http://purl.org/dc/terms/subject> ?related_subj .

 ?s3 <http://purl.org/dc/terms/identifier> ?secondRelated_id .
 ?s3 <http://purl.org/dc/terms/subject> ?related_subj .
 filter( ?related_subj != ?orig_subj && ?orig_id != ?related_id && ?related_id != ?
     secondRelated_id)
 filter( ?orig_id in (##ids##) )
}
```

Listing B.114: SPARQL Query (Virtuoso) for Query Scenario GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_2HOPS.

```
with <loddwhbench>
delete { ?s <http://my.schema/change> 'something' } #the delete clause will not have any impact
     while the given triples do not exist
insert { ?s <http://my.schema/change> 'cheesecake' } #used delete-insert statement since simple
     insert statement does not work with variables
WHERE
{
 ?s ?p ?o
}
```

Listing B.115: SPARQL Query (Virtuoso) for Query Scenario SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY.

```
with <loddwhbench>                  #created new update statement
delete { ?s <http://my.schema/change> 'something' }
insert { ?s <http://my.schema/subjectid> ?subjectid }
WHERE
{
```

```
 ?s <http://purl.org/dc/terms/subject> ?subject .
 BIND(substr(?subject,22) as ?subjectid)
}
```

Listing B.116: SPARQL Query (Virtuoso) for Query Scenario SCHEMA_CHANGE_INTRODUCE_STRING_OP.

```
with <loddwhbench>
delete { }
insert { ?s <http://my.schema/BibliographicResource> "true"^^xsd:boolean}
where {
 ?s ?p ?o.
 ?s <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://purl.org/dc/terms/
      BibliographicResource> .
}
```

Listing B.117: SPARQL Query (Virtuoso) for Query Scenario SCHEMA_CHANGE_MIGRATE_RDF_TYPE_part1.

```
with <loddwhbench>
delete { }
insert { ?s <http://my.schema/Manifestation> "true"^^xsd:boolean}
where {
 ?s ?p ?o.
 ?s <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://purl.org/vocab/frbr/core#
      Manifestation> .
}
```

Listing B.118: SPARQL Query (Virtuoso) for Query Scenario SCHEMA_CHANGE_MIGRATE_RDF_TYPE_part2.

```
with <loddwhbench>
delete { }
insert { ?s <http://my.schema/Book> "true"^^xsd:boolean}
where {
 ?s ?p ?o.
 ?s <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://purl.org/ontology/bibo/Book> .
}
```

Listing B.119: SPARQL Query (Virtuoso) for Query Scenario SCHEMA_CHANGE_MIGRATE_RDF_TYPE_part3.

```
WITH <loddwhbench> DELETE {
?s <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> ?o
} WHERE {
 ?s <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> ?o.
}
```

Listing B.120: SPARQL Query (Virtuoso) for Query Scenario SCHEMA_CHANGE_MIGRATE_RDF_TYPE_part4.

```
WITH <loddwhbench> DELETE {
?s <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> ?o
} WHERE {
 ?s <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> ?o
}
```

Listing B.121: SPARQL Query (Virtuoso) for Query Scenario SCHEMA_CHANGE_REMOVE_RDF_TYPE.

```
with <loddwhbench>
delete { ?s <http://my.schema/change> 'something' }
insert { ?s <http://purl.org/dc/terms/issued> '0'}
WHERE
{
 ?s ?p ?o .
 filter not exists {
  ?s <http://purl.org/dc/terms/issued> ?a  .
 }
}
```

Listing B.122: SPARQL Query (Virtuoso) for Query Scenario UPDATE_HIGH_SELECTIVITY_NON_ISSUED.

```
with <loddwhbench>
delete { ?s <http://purl.org/dc/terms/medium> 'paper' }
insert { ?s <http://purl.org/dc/terms/medium> 'recycled trees' }
WHERE
{
 ?s <http://purl.org/dc/terms/medium> 'paper'
}
```

Listing B.123: SPARQL Query (Virtuoso) for Query Scenario UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM.

B.6 Fuseki Benchmark Implementation

```
package database;

import java.io.ByteArrayInputStream;
import java.io.File;
import java.io.InputStream;
import java.io.StringWriter;
import java.nio.charset.StandardCharsets;
import java.util.ArrayList;
import java.util.List;

import com.google.common.base.Joiner;
import com.hp.hpl.jena.query.QueryExecutionFactory;
import com.hp.hpl.jena.query.QuerySolution;
import com.hp.hpl.jena.query.ResultSet;
import com.hp.hpl.jena.rdf.model.Model;
import com.hp.hpl.jena.update.UpdateExecutionFactory;
import com.hp.hpl.jena.update.UpdateFactory;
import com.hp.hpl.jena.update.UpdateProcessor;
import com.hp.hpl.jena.update.UpdateRequest;

import util.Dataset;
import util.QueryResult;
import util.QueryScenario;
import util.Templates;
import util.dumper.Helpers;

public class Fuseki extends Helpers implements Database {

 public final String SPARQL_SERVICE_URL = "http://localhost:3030/DB/sparql";
 public final String UPDATE_SERVICE_URL = "http://localhost:3030/DB/update";

 private String queryExecutionString;
 private Templates templates;
 private List<UpdateProcessor> updateProcessors;

 public static void main(String[] args) throws Exception {
  Database database = new Fuseki();

  database.setUp();
  // database.load(Dataset.hebis_21257740_26887667_rdf_gz);
  database.load(Dataset.hebis_10000_records);

  QueryScenario queryScenario = QueryScenario.ENTITY_RETRIEVAL_BY_ID_100_ENTITIES;

  database.prepare(queryScenario);
  QueryResult queryResult = database.query(queryScenario);
  database.query(queryScenario);
  database.query(queryScenario);
  System.out.println(queryResult);

  queryScenario = QueryScenario.CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES;

  database.prepare(queryScenario);
  queryResult = database.query(queryScenario);
  System.out.println(queryResult);
 }

 public Fuseki() {
  this.templates = new Templates("fuseki", ".sql");
 }

 @Override
```

```
public String getName() {
 return "Fuseki";
}

@Override
public String getVersion() {
 return "2.3.0 2015-07-25T17:11:28+0000 / jena-libs 2.13.0";
}

@Override
public void setUp() throws Exception {

}

@Override
public void load(Dataset dataset) throws Exception {
 // Sadly, Fuseki does seem to support loading gz-compressed files when
 // loading via command line - strangely, in contrast to the HTTP-GUI...

 ProcessBuilder gunzipProcess = new ProcessBuilder("gunzip", "-k", dataset.file.getAbsolutePath
      ());

 String extractedFileString = dataset.file.getName();
 extractedFileString = extractedFileString.substring(0, extractedFileString.length() - 3);

 File extractedFile = new File(dataset.file.getParentFile() + File.separator +
      extractedFileString);
 if (extractedFile.exists()) {
  extractedFile.delete();
 }

 Process p = gunzipProcess.start();
 int errorlevel = p.waitFor();
 if (errorlevel != 0)
  throw new RuntimeException("gunzip returned " + errorlevel);

 // Upload extracted dump via curl
 ProcessBuilder curlProcess = new ProcessBuilder("curl", "-v", "-XPOST", "--data-binary", "@" +
      extractedFile.getAbsolutePath(), "--header",
   "Content-type: application/rdf+xml", "http://localhost:3030/DB/data?default");

 p = curlProcess.start();
 errorlevel = p.waitFor();

 if (errorlevel != 0)
  throw new RuntimeException("curl returned " + errorlevel);
}

@Override
public void prepare(QueryScenario queryScenario) throws Exception {
 if (queryScenario.isReadOnly) {

  // Prepare IDs for ENTITY_RETRIEVAL scenarios
  switch (queryScenario) {
  case ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY:
  case ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES:
  case ENTITY_RETRIEVAL_BY_ID_100_ENTITIES: {
   String query = templates.resolve("ENTITY_RETRIEVAL_prepare");
   switch (queryScenario) {
   case ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY:
    query += " limit 1";
    break;
   case ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES:
    query += " limit 10";
    break;
   case ENTITY_RETRIEVAL_BY_ID_100_ENTITIES:
    query += " limit 100";
    break;
   default:
   }

   ResultSet resultSet = QueryExecutionFactory.sparqlService(SPARQL_SERVICE_URL, query).
        execSelect();
   ArrayList<String> ids = new ArrayList<>();
   while (resultSet.hasNext()) {
    QuerySolution querySolution = resultSet.nextSolution();
    ids.add("\"" + querySolution.getLiteral(resultSet.getResultVars().get(0)).getString() + "\"")
```

```
        ;
   }

   this.queryExecutionString = templates.resolve("ENTITY_RETRIEVAL").replaceAll("##ids##", Joiner
        .on(",").join(ids));
   break;
  }
  case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY:
  case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES:
  case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES:
  // case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_2HOPS_ONE_ENTITY:
  // case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_2HOPS_10_ENTITIES:
  // case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_2HOPS_100_ENTITIES:
  {
   String query = templates.resolve("GRAPH_LIKE_prepare");
   switch (queryScenario) {
   // case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_2HOPS_100_ENTITIES:
   case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES:
    query += " limit 100";
    break;
   case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES:
    // case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_2HOPS_10_ENTITIES:
    query += " limit 10";
    break;
   case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY:
    // case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_2HOPS_ONE_ENTITY:
    query += " limit 100";
    break;
   default:
    throw new RuntimeException("Dont know how to limit " + queryScenario);
   }
   ResultSet resultSet = QueryExecutionFactory.sparqlService(SPARQL_SERVICE_URL, query).
        execSelect();
   ArrayList<String> ids = new ArrayList<>();
   while (resultSet.hasNext()) {
    QuerySolution querySolution = resultSet.nextSolution();
    ids.add("\"" + querySolution.getLiteral(resultSet.getResultVars().get(0)).getString() + "\"")
        ;
   }

   String templateName = "GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_2HOPS";
   switch (queryScenario) {
   case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES:
   case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES:
   case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY:
    templateName = "GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP";
    break;
   }

   this.queryExecutionString = templates.resolve(templateName).replaceAll("##ids##", Joiner.on(",
        ").join(ids));
   break;
  }
  default: {
   this.queryExecutionString = templates.resolve(queryScenario);
  }
  }

 } else {
  if (updateProcessors == null) {
   this.updateProcessors = new ArrayList<UpdateProcessor>();
  }
  updateProcessors.clear();

  if (queryScenario == QueryScenario.SCHEMA_CHANGE_MIGRATE_RDF_TYPE) {
   for (int part = 1; part <= 4; part++) {
    UpdateRequest request = UpdateFactory.create(templates.resolve(queryScenario + "_part" + part
        ));
    updateProcessors.add(UpdateExecutionFactory.createRemote(request, UPDATE_SERVICE_URL));
   }
  } else {
   UpdateRequest request = UpdateFactory.create(templates.resolve(queryScenario));
   updateProcessors.add(UpdateExecutionFactory.createRemote(request, UPDATE_SERVICE_URL));
  }

 }
}
```

```
@Override
public QueryResult query(QueryScenario queryScenario) throws Exception {
 QueryResult queryResult = new QueryResult(queryScenario.queryResultType);
 ResultSet resultSet;
 switch (queryScenario.queryResultType) {
 case GRAPH:
  resultSet = QueryExecutionFactory.sparqlService(SPARQL_SERVICE_URL, queryExecutionString).
      execSelect();
  while (resultSet.hasNext()) {
   QuerySolution querySolution = resultSet.nextSolution();
   switch (resultSet.getResultVars().size()) {
   case 3:
    queryResult.push(querySolution.getLiteral(resultSet.getResultVars().get(0)).getString(),
      querySolution.getLiteral(resultSet.getResultVars().get(1)).getString(),
      querySolution.getLiteral(resultSet.getResultVars().get(2)).getString());
    break;
   case 5:
    queryResult.push(querySolution.getLiteral(resultSet.getResultVars().get(0)).getString(),
      querySolution.getLiteral(resultSet.getResultVars().get(1)).getString(),
      querySolution.getLiteral(resultSet.getResultVars().get(2)).getString(),
      querySolution.getLiteral(resultSet.getResultVars().get(3)).getString(),
      querySolution.getLiteral(resultSet.getResultVars().get(4)).getString());
    break;
   default:
    throw new RuntimeException("Cannot parse " + resultSet.getResultVars().size() + " columns in
        result set");
   }
  }
  break;
 case TWO_COLUMNS:
  resultSet = QueryExecutionFactory.sparqlService(SPARQL_SERVICE_URL, queryExecutionString).
      execSelect();
  while (resultSet.hasNext()) {
   QuerySolution querySolution = resultSet.nextSolution();
   queryResult.push(querySolution.getLiteral(resultSet.getResultVars().get(0)).getString(),
     querySolution.getLiteral(resultSet.getResultVars().get(1)).getString());
  }
  break;
 case COMPLETE_ENTITIES:
  Model model = QueryExecutionFactory.sparqlService(SPARQL_SERVICE_URL, queryExecutionString).
      execDescribe();
  StringWriter stringWriter = new StringWriter();
  model.write(stringWriter);

  InputStream stream = new ByteArrayInputStream(stringWriter.toString().getBytes(StandardCharsets
      .UTF_8));

  readRdf(stream, dataObject -> {
   queryResult.push(dataObject);
  } , null);
  break;
 case NONE:
 default:
  for (UpdateProcessor updateProcessor : updateProcessors)
   updateProcessor.execute();
  break;
 }

 return queryResult;
}

@Override
public void clean() throws Exception {
 UpdateRequest request = UpdateFactory.create("drop all");
 UpdateExecutionFactory.createRemote(request, UPDATE_SERVICE_URL).execute();
}

@Override
public void start() {
 Helpers.terminalLaunchScript("fuseki.sh", 7);
}

@Override
public void stop() {
 // TODO Auto-generated method stub
```

```
 }

 @Override
 public String toString() {
  return "Fuseki [getName()=" + getName() + ", getVersion()=" + getVersion() + "]";
 }
}
```

Listing B.124: Java Source Code of the Fuseki Benchmark Implementation.

```
select ?decade (count(*) as ?count)
where
{
 ?a <http://purl.org/dc/terms/issued> ?century
}
group by (substr(?century,1,3) as ?decade)
order by desc(?count) ?decade
```

Listing B.125: SPARQL Query (Fuseki) for Query Scenario AGGREGATE_ISSUES_PER_DECADE_ALL.

```
select ?decade (count(*) as ?count)
where
{
 ?a <http://purl.org/dc/terms/issued> ?century
}
group by (substr(?century,1,3) as ?decade)
order by desc(?count) ?decade
limit 10
```

Listing B.126: SPARQL Query (Fuseki) for Query Scenario AGGREGATE_ISSUES_PER_DECADE_TOP10.

```
select ?decade (count(*) as ?count)
where
{
 ?a <http://purl.org/dc/terms/issued> ?century
}
group by (substr(?century,1,3) as ?decade)
order by desc(?count) ?decade
limit 100
```

Listing B.127: SPARQL Query (Fuseki) for Query Scenario AGGREGATE_ISSUES_PER_DECADE_TOP100.

```
select ?publisher (count(?publisher) as ?count)
where
{
 ?a <http://purl.org/dc/terms/publisher> ?publisher .
}
group by ?publisher
order by desc(?count) ?publisher
```

Listing B.128: SPARQL Query (Fuseki) for Query Scenario AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL.

```
select ?publisher (count(?publisher) as ?count)
where
{
 ?a <http://purl.org/dc/terms/publisher> ?publisher .
}
group by ?publisher
order by desc(?count) ?publisher
limit 10
```

Listing B.129: SPARQL Query (Fuseki) for Query Scenario AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10.

```
select ?publisher (count(?publisher) as ?count)
where
{
 ?a <http://purl.org/dc/terms/publisher> ?publisher .
```

```
}
group by ?publisher
order by desc(?count) ?publisher
limit 100
```

Listing B.130: SPARQL Query (Fuseki) for Query Scenario AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100.

```
describe *
where
{
 ?s ?o ?p .
 optional { ?s <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> ?type . }

 ?s <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://purl.org/dc/terms/
      BibliographicResource> .
}
```

Listing B.131: SPARQL Query (Fuseki) for Query Scenario CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES.

```
describe *
where
{
 ?s ?o ?p .
 optional { ?s <http://purl.org/dc/terms/title> ?title . }
 optional { ?s <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> ?type . }

 filter regex(?title, 'stud(ie|y)', 'i') .
 ?s <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://purl.org/dc/terms/
      BibliographicResource> .
}
```

Listing B.132: SPARQL Query (Fuseki) for Query Scenario CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES.

```
describe *
where
{
 ?s ?o ?p .
 optional { ?s <http://purl.org/dc/terms/title> ?title . }
 filter regex(?title, 'stud(ie|y)', 'i') .
}
```

Listing B.133: SPARQL Query (Fuseki) for Query Scenario CONDITIONAL_TABLE_SCAN_ALL_STUDIES.

```
WITH <loddwhbench>
DELETE
 { ?s ?p ?o }
WHERE {
  ?s ?p ?o .
  ?s <http://purl.org/dc/terms/issued> '0' .
}
```

Listing B.134: SPARQL Query (Fuseki) for Query Scenario DELETE_HIGH_SELECTIVIY_NON_ISSUED.

```
WITH <loddwhbench>
DELETE
 {
  ?s ?p ?o
 }
WHERE
 {
  ?s ?p ?o ;
   <http://purl.org/dc/terms/medium> 'recycled trees'
 }
```

Listing B.135: SPARQL Query (Fuseki) for Query Scenario DELETE_LOW_SELECTIVITY_PAPER_MEDIUM.

```
describe * where {
 ?s ?p ?o .
 ?s <http://purl.org/dc/terms/identifier> ?identifier .
 FILTER( ?identifier IN ( ##ids## ))
}
```

Listing B.136: SPARQL Query (Fuseki) for Query Scenario ENTITY_RETRIEVAL.

```
select ?identifier
where
{
 ?s <http://purl.org/dc/terms/identifier> ?identifier .
 optional { ?s <http://purl.org/dc/terms/medium> ?medium . }
 optional { ?s <http://iflastandards.info/ns/isbd/elements/P1008> ?P1008 . }
 optional { ?s <http://purl.org/dc/terms/contributor> ?contributor . }
 optional { ?s <http://purl.org/dc/terms/issued> ?issued . }
}
order by ?medium ?P1008 ?contributor ?issued ?identifier
```

Listing B.137: SPARQL Query (Fuseki) for Query Scenario ENTITY_RETRIEVAL_prepare.

```
select ?identifier
where
{
 ?s <http://purl.org/dc/terms/identifier> ?identifier .
 ?s <http://purl.org/dc/terms/subject> ?subject .
 optional { ?s <http://purl.org/dc/terms/medium> ?medium . }
 optional { ?s <http://iflastandards.info/ns/isbd/elements/P1008> ?P1008 . }
 optional { ?s <http://purl.org/dc/terms/contributor> ?contributor . }
 optional { ?s <http://purl.org/dc/terms/issued> ?issued . }
}
order by ?medium ?P1008 ?contributor ?issued ?identifier
```

Listing B.138: SPARQL Query (Fuseki) for Query Scenario GRAPH_LIKE_prepare.

```
select ?orig_id ?orig_subj ?related_id
where
{
 ?s1 <http://purl.org/dc/terms/identifier> ?orig_id .
 ?s1 <http://purl.org/dc/terms/subject> ?orig_subj .
 ?s2 <http://purl.org/dc/terms/identifier> ?related_id .
 ?s2 <http://purl.org/dc/terms/subject> ?orig_subj .
 filter( ?orig_id != ?related_id )
 filter( ?orig_id in (##ids##) )
}
```

Listing B.139: SPARQL Query (Fuseki) for Query Scenario GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP.

```
select ?orig_id ?orig_subj ?related_id ?secondRelated_id from <loddwhbench>
where
{
 ?s1 <http://purl.org/dc/terms/identifier> ?orig_id .
 ?s1 <http://purl.org/dc/terms/subject> ?orig_subj .

 ?s2 <http://purl.org/dc/terms/identifier> ?related_id .
 ?s2 <http://purl.org/dc/terms/subject> ?orig_subj .
 ?s2 <http://purl.org/dc/terms/subject> ?related_subj .

 ?s3 <http://purl.org/dc/terms/identifier> ?secondRelated_id .
 ?s3 <http://purl.org/dc/terms/subject> ?related_subj .
 filter( ?related_subj != ?orig_subj && ?orig_id != ?related_id && ?related_id != ?
      secondRelated_id)
 filter( ?orig_id in (##ids##) )
}
```

Listing B.140: SPARQL Query (Fuseki) for Query Scenario GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_2HOPS.

```
delete { ?s <http://my.schema/change> 'something' } #the delete clause will not have any impact
     while the given triples do not exist
insert { ?s <http://my.schema/change> 'cheesecake' } #used delete-insert statement since simple
     insert statement does not work with variables
WHERE
{
 ?s ?p ?o
}
```

Listing B.141: SPARQL Query (Fuseki) for Query Scenario SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY.

```
delete { ?s <http://my.schema/change> 'something' }
insert { ?s <http://my.schema/subjectid> ?subjectid }
WHERE
{
 ?s <http://purl.org/dc/terms/subject> ?subject .
 BIND(substr(?subject,22) as ?subjectid)
}
```

Listing B.142: SPARQL Query (Fuseki) for Query Scenario SCHEMA_CHANGE_INTRODUCE_STRING_OP.

```
delete { }
insert { ?s <http://my.schema/BibliographicResource> "true"^^<http://www.w3.org/2001/XMLSchema#
     boolean>}
where {
 ?s ?p ?o.
 ?s <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://purl.org/dc/terms/
      BibliographicResource> .
}
```

Listing B.143: SPARQL Query (Fuseki) for Query Scenario SCHEMA_CHANGE_MIGRATE_RDF_TYPE_part1.

```
delete { }
insert { ?s <http://my.schema/Manifestation> "true"^^<http://www.w3.org/2001/XMLSchema#boolean>}
where {
 ?s ?p ?o.
 ?s <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://purl.org/vocab/frbr/core#
      Manifestation> .
}
```

Listing B.144: SPARQL Query (Fuseki) for Query Scenario SCHEMA_CHANGE_MIGRATE_RDF_TYPE_part2.

```
delete { }
insert { ?s <http://my.schema/Book> "true"^^<http://www.w3.org/2001/XMLSchema#boolean>}
where {
 ?s ?p ?o.
 ?s <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://purl.org/ontology/bibo/Book> .
}
```

Listing B.145: SPARQL Query (Fuseki) for Query Scenario SCHEMA_CHANGE_MIGRATE_RDF_TYPE_part3.

```
DELETE {
?s <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> ?o
} WHERE {
 ?s <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> ?o.
}
```

Listing B.146: SPARQL Query (Fuseki) for Query Scenario SCHEMA_CHANGE_MIGRATE_RDF_TYPE_part4.

```
DELETE {
?s <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> ?o
} WHERE {
 ?s <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> ?o
}
```

Listing B.147: SPARQL Query (Fuseki) for Query Scenario SCHEMA_CHANGE_REMOVE_RDF_TYPE.

```
delete { ?s <http://my.schema/change> 'something' }
insert { ?s <http://purl.org/dc/terms/issued> '0'}
WHERE
{
 ?s ?p ?o .
 filter not exists {
  ?s <http://purl.org/dc/terms/issued> ?a  .
 }
}
```

Listing B.148: SPARQL Query (Fuseki) for Query Scenario UPDATE_HIGH_SELECTIVITY_NON_ISSUED.

```
delete { ?s <http://purl.org/dc/terms/medium> 'paper' }
insert { ?s <http://purl.org/dc/terms/medium> 'recycled trees' }
WHERE
{
 ?s <http://purl.org/dc/terms/medium> 'paper'
}
```

Listing B.149: SPARQL Query (Fuseki) for Query Scenario UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM.

B.7 MongoDB Benchmark Implementation

```
package database;

import java.util.ArrayList;
import java.util.List;

import org.bson.BSON;
import org.bson.Document;

import com.mongodb.Block;
import com.mongodb.DBObject;
import com.mongodb.MongoClient;
import com.mongodb.client.AggregateIterable;
import com.mongodb.client.FindIterable;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;
import com.mongodb.util.JSON;

import util.Codes;
import util.Config;
import util.DataObject;
import util.Dataset;
import util.QueryResult;
import util.QueryResult.Type;
import util.QueryScenario;
import util.dumper.Helpers;
import static java.util.Arrays.asList;

public class MongoDB implements Database {

 public static void main(String[] args) throws Throwable {
  MongoDB mongoDb = new MongoDB();
  mongoDb.start();
  mongoDb.clean();
  mongoDb.setUp();
  mongoDb.load(Dataset.hebis_10000_records);

  QueryScenario testScenario = QueryScenario.
       GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES;

  mongoDb.prepare(testScenario);
  mongoDb.query(testScenario);
  mongoDb.stop();
 }

 MongoClient mongoClient;
 MongoDatabase database;
 MongoCollection<Document> collection;

 @Override
```

```
public String getName() {
 return "MongoDB";
}

@Override
public String getVersion() {
 return "3.0.6";
}

@Override
public void clean() throws Exception {
 mongoClient = new MongoClient("localhost", 27017);
 mongoClient.dropDatabase(Config.DATABASE);
}

@Override
public void setUp() throws Exception {
 database = mongoClient.getDatabase(Config.DATABASE);
 collection = database.getCollection(Config.DATABASE);
}

@Override
public void load(Dataset dataset) throws Exception {

 Helpers.readRdf(dataset, dataObject -> {
  Document doc = new Document();

  for (int i = 0; i < Codes.values().length; i++) {
   Codes code = Codes.values()[i];
   if (dataObject.get(code) != null) {
    String key = code.toString();
    if (!code.IS_MULTIPLE) {
     String value = dataObject.get(code).toString();
     doc.append(key, value);
    } else {
     List<String> values = (ArrayList<String>) dataObject.get(code);
     doc.append(key, values);
    }
   }
  }

  collection.insertOne(doc);
 } , counter -> {
  if (counter % 50000 == 0)
   System.out.println(counter + " records so far... (MongoDB load())");
 });

}

private List<String> EntityRetrievalAndGraphScenariosDcTermsIdentifiers = new ArrayList<>();

private void FillEntityRetrievalAndGraphScenariosScenariosDcTermsIdentifiers(Document doc) {
 if (!doc.containsKey("DCTERMS_IDENTIFIER")) {
  System.err.println(
    "MongoDB, prepare ENTITY_RETRIEVAL/GRAPH: One element has no DCTERMS_IDENTIFIER. Element is
         ignored. Query-Size will not be 1, 10 or 100.");
  return;
 }

 EntityRetrievalAndGraphScenariosDcTermsIdentifiers.add(doc.getString("DCTERMS_IDENTIFIER"));
}

@Override
public void prepare(QueryScenario queryScenario) throws Exception {

 Document findFilter = new Document();
 if (queryScenario == QueryScenario.GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY
   || queryScenario == QueryScenario.GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES
   || queryScenario == QueryScenario.GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES)
//    || queryScenario == QueryScenario.GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_2HOPS_ONE_ENTITY
//    || queryScenario == QueryScenario.GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_2HOPS_10_ENTITIES
//    || queryScenario == QueryScenario.GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_2HOPS_100_ENTITIES
    )
  findFilter = new Document("DCTERMS_SUBJECT", new Document("$exists", true));

 switch (queryScenario) {
 case ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY:
```

```
  case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY:
//  case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_2HOPS_ONE_ENTITY:
   EntityRetrievalAndGraphScenariosDcTermsIdentifiers.clear();
   FindIterable<Document> prepEntOne = collection.find(findFilter).sort(new Document("
        DCTERMS_MEDIUM", 1).append("ISBD_P1008", 1)
     .append("DCTERM_CONTRIBUTOR", 1).append("DCTERMS_ISSUED", 1).append("DCTERMS_IDENTIFIER", 1))
          .limit(1);
   prepEntOne.forEach(new Block<Document>() {
    @Override
    public void apply(Document arg0) {
     FillEntityRetrievalAndGraphScenariosScenariosDcTermsIdentifiers(arg0);
    }
   });
   break;

  case ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES:
  case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES:
//  case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_2HOPS_10_ENTITIES:
   EntityRetrievalAndGraphScenariosDcTermsIdentifiers.clear();
   FindIterable<Document> prepEntTen = collection.find(findFilter).sort(new Document("
        DCTERMS_MEDIUM", 1).append("ISBD_P1008", 1)
     .append("DCTERM_CONTRIBUTOR", 1).append("DCTERMS_ISSUED", 1).append("DCTERMS_IDENTIFIER", 1))
          .limit(10);
   prepEntTen.forEach(new Block<Document>() {
    @Override
    public void apply(Document arg0) {
     FillEntityRetrievalAndGraphScenariosScenariosDcTermsIdentifiers(arg0);
    }
   });
   break;

  case ENTITY_RETRIEVAL_BY_ID_100_ENTITIES:
  case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES:
//  case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_2HOPS_100_ENTITIES:
   EntityRetrievalAndGraphScenariosDcTermsIdentifiers.clear();
   FindIterable<Document> prepEntHun = collection.find(findFilter).sort(new Document("
        DCTERMS_MEDIUM", 1).append("ISBD_P1008", 1)
     .append("DCTERM_CONTRIBUTOR", 1).append("DCTERMS_ISSUED", 1).append("DCTERMS_IDENTIFIER", 1))
          .limit(100);
   prepEntHun.forEach(new Block<Document>() {
    @Override
    public void apply(Document arg0) {
     FillEntityRetrievalAndGraphScenariosScenariosDcTermsIdentifiers(arg0);
    }
   });
   break;
  }

 }

 private DataObject BuildDataObjectFromDocument(Document document) {
  DataObject dataObject = new DataObject();

  for (Codes code : Codes.values()) {
   if (!document.containsKey(code.toString())) {
    dataObject.set(code, null);
    continue;
   }

   if (code.IS_MULTIPLE) {
    for (String value : (ArrayList<String>) document.get(code.toString())) {
     dataObject.putMultiple(code, value);
    }
   } else {
    dataObject.set(code, document.getString(code.toString()));
   }
  }

  return dataObject;
 }

 @Override
 public QueryResult query(QueryScenario queryScenario) throws Exception {

  QueryResult queryResult = new QueryResult(queryScenario.queryResultType);

  switch (queryScenario.queryResultType) {
```

```
case TWO_COLUMNS:
 switch (queryScenario) {
 case AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL:
  AggregateIterable<Document> results = collection
    .aggregate(asList(new Document("$match", new Document("DCTERMS_PUBLISHER", new Document("
        $exists", true))),
      new Document("$group", new Document("_id", "$DCTERMS_PUBLISHER").append("count", new
          Document("$sum", 1))),
      new Document("$sort", new Document("count", -1).append("_id", 1))));
  results.forEach(new Block<Document>() {
   @Override
   public void apply(Document arg0) {
    queryResult.push(arg0.getString("_id"), arg0.getInteger("count").toString());
   }
  });
  return queryResult;
 case AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10:
  AggregateIterable<Document> results2 = collection
    .aggregate(asList(new Document("$match", new Document("DCTERMS_PUBLISHER", new Document("
        $exists", true))),
      new Document("$group", new Document("_id", "$DCTERMS_PUBLISHER").append("count", new
          Document("$sum", 1))),
      new Document("$sort", new Document("count", -1).append("_id", 1)), new Document("$limit",
          10)));
  results2.forEach(new Block<Document>() {
   @Override
   public void apply(Document arg0) {
    queryResult.push(arg0.getString("_id"), arg0.getInteger("count").toString());
   }
  });
  return queryResult;
 case AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100:
  AggregateIterable<Document> results3 = collection
    .aggregate(asList(new Document("$match", new Document("DCTERMS_PUBLISHER", new Document("
        $exists", true))),
      new Document("$group", new Document("_id", "$DCTERMS_PUBLISHER").append("count", new
          Document("$sum", 1))),
      new Document("$sort", new Document("count", -1).append("_id", 1)), new Document("$limit",
          100)));
  results3.forEach(new Block<Document>() {
   @Override
   public void apply(Document arg0) {
    queryResult.push(arg0.getString("_id"), arg0.getInteger("count").toString());
   }
  });
  return queryResult;

 case AGGREGATE_ISSUES_PER_DECADE_ALL:
  AggregateIterable<Document> ispdec1 = collection
    .aggregate(asList(new Document("$match", new Document("DCTERMS_ISSUED", new Document("
        $exists", true))),
      new Document("$group",
        new Document("_id", new Document("$substr", asList("$DCTERMS_ISSUED", 0, 3))).append("
            count", new Document("$sum", 1))),
    new Document("$sort", new Document("count", -1).append("_id", 1))));
  ispdec1.forEach(new Block<Document>() {
   @Override
   public void apply(Document arg0) {
    queryResult.push(arg0.getString("_id"), arg0.getInteger("count").toString());
   }
  });
  return queryResult;
 case AGGREGATE_ISSUES_PER_DECADE_TOP10:
  AggregateIterable<Document> ispdec2 = collection
    .aggregate(asList(new Document("$match", new Document("DCTERMS_ISSUED", new Document("
        $exists", true))),
      new Document("$group",
        new Document("_id", new Document("$substr", asList("$DCTERMS_ISSUED", 0, 3))).append("
            count", new Document("$sum", 1))),
    new Document("$sort", new Document("count", -1).append("_id", 1)), new Document("$limit",
        10)));
  ispdec2.forEach(new Block<Document>() {
   @Override
   public void apply(Document arg0) {
    queryResult.push(arg0.getString("_id"), arg0.getInteger("count").toString());
   }
  });
```

```
  return queryResult;
 case AGGREGATE_ISSUES_PER_DECADE_TOP100:
  AggregateIterable<Document> ispdec3 = collection
    .aggregate(asList(new Document("$match", new Document("DCTERMS_ISSUED", new Document("
        $exists", true))),
      new Document("$group",
        new Document("_id", new Document("$substr", asList("$DCTERMS_ISSUED", 0, 3))).append("
            count", new Document("$sum", 1))),
    new Document("$sort", new Document("count", -1).append("_id", 1)), new Document("$limit",
        100)));
  ispdec3.forEach(new Block<Document>() {
   @Override
   public void apply(Document arg0) {
    queryResult.push(arg0.getString("_id"), arg0.getInteger("count").toString());
   }
  });
  return queryResult;
 }
 break;

case COMPLETE_ENTITIES:
 switch (queryScenario) {
 case CONDITIONAL_TABLE_SCAN_ALL_STUDIES:
  FindIterable<Document> results4 = collection
    .find(new Document("DCTERMS_TITLE", new Document("$regex", ".*stud(ie|y).*").append("
        $options", "i")));
  results4.forEach(new Block<Document>() {
   @Override
   public void apply(Document arg0) {
    queryResult.push(BuildDataObjectFromDocument(arg0));
   }
  });
  return queryResult;
 case CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES:
  FindIterable<Document> results5 = collection.find(new Document("RDF_TYPE", "http://purl.org/dc
      /terms/BibliographicResource"));
  results5.forEach(new Block<Document>() {
   @Override
   public void apply(Document arg0) {
    queryResult.push(BuildDataObjectFromDocument(arg0));
   }
  });
  return queryResult;
 case CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES:
  FindIterable<Document> results6 = collection.find(new Document("RDF_TYPE", "http://purl.org/dc
      /terms/BibliographicResource")
    .append("DCTERMS_TITLE", new Document("$regex", ".*stud(ie|y).*").append("$options", "i")));
  results6.forEach(new Block<Document>() {
   @Override
   public void apply(Document arg0) {
    queryResult.push(BuildDataObjectFromDocument(arg0));
   }
  });
  return queryResult;

 case ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY:
 case ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES:
 case ENTITY_RETRIEVAL_BY_ID_100_ENTITIES:
  for (String dc_ident : EntityRetrievalAndGraphScenariosDcTermsIdentifiers) {
   FindIterable<Document> resultById = collection.find(new Document("DCTERMS_IDENTIFIER",
       dc_ident));
   resultById.forEach(new Block<Document>() {
    @Override
    public void apply(Document arg0) {
     queryResult.push(BuildDataObjectFromDocument(arg0));
    }
   });
  }
  return queryResult;
 }
 break;

case GRAPH:
 switch (queryScenario) {
 case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY:
 case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES:
 case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES:
```

```
    for (String dc_ident : EntityRetrievalAndGraphScenariosDcTermsIdentifiers) {
     FindIterable<Document> findIter1 = collection
       .find(new Document("DCTERMS_SUBJECT", new Document("$exists", true)).append("
            DCTERMS_IDENTIFIER", dc_ident));
     findIter1.forEach(new Block<Document>() {
      @Override
      public void apply(Document arg0) {
       for (String subject : (ArrayList<String>) arg0.get("DCTERMS_SUBJECT")) {
        FindIterable<Document> findIter2 = collection
          .find(new Document("DCTERMS_SUBJECT", subject).append("_id", new Document("$ne", arg0.
               getObjectId("_id"))));
        findIter2.forEach(new Block<Document>() {
         @Override
         public void apply(Document arg1) {
          queryResult.push(arg0.getString("DCTERMS_IDENTIFIER"), subject, arg1.getString("
               DCTERMS_IDENTIFIER"));
         }
        });
       }
      }
     });
    }
    return queryResult;
//    case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_2HOPS_ONE_ENTITY:
//    case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_2HOPS_10_ENTITIES:
//    case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_2HOPS_100_ENTITIES:
//     for (String dc_ident : EntityRetrievalAndGraphScenariosDcTermsIdentifiers) {
//      FindIterable<Document> findIter2Hops = collection
//        .find(new Document("DCTERMS_SUBJECT", new Document("$exists", true)).append("
     DCTERMS_IDENTIFIER", dc_ident));
//      findIter2Hops.forEach(new Block<Document>() {
//       @Override
//       public void apply(Document arg0) {
//        for (String subject : (ArrayList<String>) arg0.get("DCTERMS_SUBJECT")) {
//         FindIterable<Document> findIter2 = collection
//           .find(new Document("DCTERMS_SUBJECT", subject).append("_id", new Document("$ne", arg0
     .getObjectId("_id"))));
//         findIter2.forEach(new Block<Document>() {
//          @Override
//          public void apply(Document arg1) {
//           for (String subject2 : (ArrayList<String>) arg1.get("DCTERMS_SUBJECT")) {
//            FindIterable<Document> findIter3 = collection.find(
//              new Document("DCTERMS_SUBJECT", subject2).append("_id", new Document("$ne", arg1.
     getObjectId("_id"))));
//            findIter3.forEach(new Block<Document>() {
//             @Override
//             public void apply(Document arg2) {
//              queryResult.push(arg0.getString("DCTERMS_IDENTIFIER"), subject, arg1.getString("
     DCTERMS_IDENTIFIER"),
//                 subject2, arg2.getString("DCTERMS_IDENTIFIER"));
//             }
//            });
//           }
//          }
//         });
//        }
//       }
//      });
//     }
//     return queryResult;
   }
   break;

  case NONE:
   switch (queryScenario) {
   case SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY:
    collection.updateMany(new Document(), new Document("$set", new Document("newfield", "
         cheesecake")));
    return queryResult;

   case SCHEMA_CHANGE_INTRODUCE_STRING_OP:
    FindIterable<Document> stringOps = collection.find();
    stringOps.forEach(new Block<Document>() {
     @Override
     public void apply(Document arg0) {
      String suffix = arg0.getString("RDF_ABOUT").substring(29);
      collection.updateOne(new Document("_id", arg0.getObjectId("_id")), new Document("$set", new
```

```
          Document("idSuffix", suffix)));
     }
    });
    return queryResult;

   case SCHEMA_CHANGE_MIGRATE_RDF_TYPE:
    collection.updateMany(new Document(),
      new Document("$set", new Document("manifestation", false).append("bibresource", false).
          append("book", false)));
    collection.updateMany(new Document("RDF_TYPE", "http://purl.org/vocab/frbr/core#Manifestation"
        ),
      new Document("$set", new Document("manifestation", true)));
    collection.updateMany(new Document("RDF_TYPE", "http://purl.org/dc/terms/BibliographicResource
        "),
      new Document("$set", new Document("bibresource", true)));
    collection.updateMany(new Document("RDF_TYPE", "http://purl.org/ontology/bibo/Book"), new
        Document("$set", new Document("book", true)));
    return queryResult;

   case SCHEMA_CHANGE_REMOVE_RDF_TYPE:
    collection.updateMany(new Document(), new Document("$unset", new Document("RDF_TYPE", "")));
    return queryResult;

   case UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM:
    collection.updateMany(new Document("DCTERMS_MEDIUM", "paper"), new Document("$set", new
        Document("DCTERMS_MEDIUM", "recycled trees")));
    return queryResult;

   case UPDATE_HIGH_SELECTIVITY_NON_ISSUED:
    collection.updateMany(new Document("DCTERMS_ISSUED", new Document("$exists", false)),
      new Document("$set", new Document("DCTERMS_ISSUED", "0")));
    return queryResult;

   case DELETE_LOW_SELECTIVITY_PAPER_MEDIUM:
    collection.deleteMany(new Document("DCTERMS_MEDIUM", "recycled trees"));
    return queryResult;

   case DELETE_HIGH_SELECTIVIY_NON_ISSUED:
    collection.deleteMany(new Document("DCTERMS_ISSUED", "0"));
    return queryResult;
   }
   break;
  }

  throw new RuntimeException("Something happened");
 }

 @Override
 public String toString() {
  return "MongoDB [getName()=" + getName() + ", getVersion()=" + getVersion() + "]";
 }

 @Override
 public void start() {
  if (Config.THIS_IS_OSX)
   Helpers.terminalLaunchScript("mongodb.sh", 20);
 }

 @Override
 public void stop() {
  if (Config.THIS_IS_OSX)
   Helpers.terminalLaunchScript("mongodb_stop.sh", 10);
 }
}
```

Listing B.150: Java Source Code of the MongoDB Benchmark Implementation.

B.8 ArangoDB Benchmark Implementation

```
package database;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
```

```
import com.arangodb.ArangoConfigure;
import com.arangodb.ArangoDriver;
import com.arangodb.DocumentCursor;
import com.arangodb.entity.BaseDocument;
import com.arangodb.entity.DocumentEntity;
import com.arangodb.entity.EdgeDefinitionEntity;
import com.arangodb.util.AqlQueryOptions;
import util.Codes;
import util.Config;
import util.DataObject;
import util.Dataset;
import util.QueryResult;
import util.QueryScenario;
import util.dumper.Helpers;

public class ArangoDB implements Database {

 @Override
 public String getName() {
  return "ArangoDB";
 }

 @Override
 public String getVersion() {
  return "2.6.9 64bit -- ICU 54.1, V8 4.1.0.27, OpenSSL 1.0.2d 9 Jul 2015 / Java Driver 2.6.8";
 }

 @Override
 public String toString() {
  return "ArangoDB [getName()=" + getName() + ", getVersion()="
    + getVersion() + "]";
 }

 public static void main(String[] args) throws Throwable {
  ArangoDB arangoDb = new ArangoDB();
  arangoDb.start();
  arangoDb.clean();
  arangoDb.setUp();
  arangoDb.load(Dataset.hebis_10000_records);

  QueryScenario testScenario = QueryScenario.
       GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES;

  arangoDb.prepare(testScenario);
  QueryResult queryResult = arangoDb.query(testScenario);

  //System.out.println(queryResult);

  arangoDb.stop();
 }

 ArangoDriver arangoDriver;

 @Override
 public void start() {
  if (Config.THIS_IS_OSX)
   Helpers.terminalLaunchScript("arangodb.sh", 20);

  ArangoConfigure configure = new ArangoConfigure();
  configure.init();
  arangoDriver = new ArangoDriver(configure);
 }

 @Override
 public void stop() {
  if (Config.THIS_IS_OSX)
   Helpers.terminalLaunchScript("arangodb_stop.sh", 10);
 }

 @Override
 public void clean() throws Exception {
  if(arangoDriver.getDatabases().getResult().contains(Config.DATABASE))
   arangoDriver.deleteDatabase(Config.DATABASE);
 }

 @Override
 public void setUp() throws Exception {
```

```
 arangoDriver.createDatabase(Config.DATABASE);
 arangoDriver.setDefaultDatabase(Config.DATABASE);
 arangoDriver.createCollection(Config.DATABASE);
}

@Override
public void load(Dataset dataset) throws Exception {

 Helpers.readRdf(dataset, dataObject -> {
  BaseDocument arangoDoc = new BaseDocument();

  for (Codes code : Codes.values()) {
   if (dataObject.get(code) != null) {
    arangoDoc.addAttribute(code.toString(), dataObject.get(code));
   }
  }

  try {
   arangoDriver.createDocument(Config.DATABASE, arangoDoc);
  } catch (Exception e) {
   System.err.println("Konnte ein ArangoDocument nicht einfuegen.");
   e.printStackTrace();
  }

 } , counter -> {
  if (counter % 50000 == 0)
   System.out.println(counter + " records so far... (ArangoDB load())");
 });

}

private AqlQueryOptions GetAqlQueryOptionsForLongRunningCursor(){
 return new AqlQueryOptions().setCount(false).setFullCount(false).setTtl(60 * 60); // TTL in
      seconds, now 1 hour, doesn't matter if cursors are correctly closed after usage
}

private boolean GraphLoaded = false;
private void InitializeDctermsSubjectsGraph() throws Exception{
 String graphName = Config.DATABASE + "_graph", edgeCollectionName = Config.DATABASE + "
      _hasSubject", subjectsCollection = Config.DATABASE + "_subjects";

 arangoDriver.createCollection(subjectsCollection);
 EdgeDefinitionEntity edgeDefinition = new EdgeDefinitionEntity();
 edgeDefinition.setCollection(edgeCollectionName);
 List<String> from = new ArrayList<>(), to = new ArrayList<>();
 from.add(Config.DATABASE); to.add(subjectsCollection);
 edgeDefinition.setFrom(from); edgeDefinition.setTo(to);
 List<EdgeDefinitionEntity> edgeDefinitions = new ArrayList<>();
 edgeDefinitions.add(edgeDefinition);
 arangoDriver.createGraph(graphName, edgeDefinitions, new ArrayList<String>(), true);

 String getAllRelevantQuery = "for e in loddwhbench filter has(e, 'DCTERMS_SUBJECT') return { '
      id' : e._id, 'DCTERMS_SUBJECT' : e.DCTERMS_SUBJECT }";
 DocumentCursor<BaseDocument> getAllRelevantCursor = arangoDriver.executeDocumentQuery(
      getAllRelevantQuery, null, GetAqlQueryOptionsForLongRunningCursor(), BaseDocument.class);
 HashMap<String, String> existingSubjects = new HashMap<>(); // Map Subject => DocumentHandle (
      temporarily saving to prevent additional queries)
 for(DocumentEntity<BaseDocument> relevantDocument : getAllRelevantCursor){
  BaseDocument document = relevantDocument.getEntity();

  String documentHandle = document.getAttribute("id").toString();
  ArrayList<String> documentSubjects = (ArrayList<String>) document.getAttribute("DCTERMS_SUBJECT
       ");

  for (String subject : documentSubjects) {
   if(!existingSubjects.containsKey(subject)){
    BaseDocument arangoDoc = new BaseDocument();
    arangoDoc.addAttribute("subject", subject);
    DocumentEntity<BaseDocument> newSubjectDocument = arangoDriver.createDocument(
         subjectsCollection, arangoDoc);
    existingSubjects.put(subject, newSubjectDocument.getDocumentHandle());
   }

   String subjectHandle = existingSubjects.get(subject);
   arangoDriver.graphCreateEdge(graphName, edgeCollectionName, documentHandle, subjectHandle,
        null, null);
  }
```

```
 }
 getAllRelevantCursor.close();

 GraphLoaded = true;
}

private DataObject BuildDataObjectFromDocument(BaseDocument document) {
 DataObject dataObject = new DataObject();

 for (Codes code : Codes.values()) {
  if (document.getAttribute(code.toString()) == null) {
   dataObject.set(code, null);
   continue;
  }

  if (code.IS_MULTIPLE){
   for(String value : (ArrayList<String>) document.getAttribute(code.toString())){
    dataObject.putMultiple(code, value);
   }
  }
  else
  {
   dataObject.set(code, document.getAttribute(code.toString()).toString());
  }
 }

 return dataObject;
}

private void PushTwoColumnsToQueryResult(BaseDocument baseDocument, String c1, String c2,
     QueryResult queryResult){
 Object publisherOrDecade = baseDocument.getAttribute(c1), count = baseDocument.getAttribute(c2)
      ;
 String spublisherOrDecade = null, scount = null;
 if(publisherOrDecade != null) spublisherOrDecade = publisherOrDecade.toString();
 if(count != null) scount = String.valueOf((int) Double.parseDouble(count.toString()));
 queryResult.push(spublisherOrDecade, scount);
}

private List<String> EntityRetrievalAndGraphScenariosDcTermsIdentifiers = new ArrayList<>();
private void FillEntityRetrievalAndGraphScenariosDcTermsIdentifiers(BaseDocument doc){
 if(doc.getAttribute("DCTERMS_IDENTIFIER") == null){
  System.err.println("ArangoDB, prepare ENTITY_RETRIEVAL/GRAPH: One element has no
       DCTERMS_IDENTIFIER. Element is ignored. Query-Size will not be 1, 10 or 100.");
  return;
 }
 EntityRetrievalAndGraphScenariosDcTermsIdentifiers.add(doc.getAttribute("DCTERMS_IDENTIFIER").
      toString());
}

@Override
public void prepare(QueryScenario queryScenario) throws Exception {
 String query;
 DocumentCursor<BaseDocument> documentCursor;
 String filterString = "";

 if(queryScenario == QueryScenario.GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY ||
      queryScenario == QueryScenario.GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES) {
//   || queryScenario == QueryScenario.GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES
    || queryScenario == QueryScenario.GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_2HOPS_ONE_ENTITY ||
    queryScenario == QueryScenario.GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_2HOPS_10_ENTITIES ||
    queryScenario == QueryScenario.GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_2HOPS_100_ENTITIES){
  if(!GraphLoaded) InitializeDctermsSubjectsGraph();
  filterString = "FILTER HAS(r, 'DCTERMS_SUBJECT')";
 }

 switch(queryScenario){
  case CONDITIONAL_TABLE_SCAN_ALL_STUDIES:
   arangoDriver.createFulltextIndex(Config.DATABASE, "DCTERMS_TITLE");
   break;

  case ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY:
  case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY:
//   case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_2HOPS_ONE_ENTITY:
   EntityRetrievalAndGraphScenariosDcTermsIdentifiers.clear();
   query = String.format("FOR r IN loddwhbench %s SORT r.DCTERMS_MEDIUM, r.ISBD_P1008, r.
        DCTERM_CONTRIBUTOR, r.DCTERMS_ISSUED, r.DCTERMS_IDENTIFIER LIMIT 1 RETURN { '
```

```
        DCTERMS_IDENTIFIER' : r.DCTERMS_IDENTIFIER }", filterString);
    documentCursor = arangoDriver.executeDocumentQuery(query, null, arangoDriver.
        getDefaultAqlQueryOptions(), BaseDocument.class);
    for (DocumentEntity<BaseDocument> documentEntity : documentCursor) {
     FillEntityRetrievalAndGraphScenariosDcTermsIdentifiers(documentEntity.getEntity());
    }
    break;

   case ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES:
   case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES:
//   case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_2HOPS_10_ENTITIES:
    EntityRetrievalAndGraphScenariosDcTermsIdentifiers.clear();
    query = String.format("FOR r IN loddwhbench %s SORT r.DCTERMS_MEDIUM, r.ISBD_P1008, r.
        DCTERM_CONTRIBUTOR, r.DCTERMS_ISSUED, r.DCTERMS_IDENTIFIER LIMIT 10 RETURN { '
        DCTERMS_IDENTIFIER' : r.DCTERMS_IDENTIFIER }", filterString);
    documentCursor = arangoDriver.executeDocumentQuery(query, null, arangoDriver.
        getDefaultAqlQueryOptions(), BaseDocument.class);
    for (DocumentEntity<BaseDocument> documentEntity : documentCursor) {
     FillEntityRetrievalAndGraphScenariosDcTermsIdentifiers(documentEntity.getEntity());
    }
    break;

   case ENTITY_RETRIEVAL_BY_ID_100_ENTITIES:
   case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES:
//   case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_2HOPS_100_ENTITIES:
    EntityRetrievalAndGraphScenariosDcTermsIdentifiers.clear();
    query = String.format("FOR r IN loddwhbench %s SORT r.DCTERMS_MEDIUM, r.ISBD_P1008, r.
        DCTERM_CONTRIBUTOR, r.DCTERMS_ISSUED, r.DCTERMS_IDENTIFIER LIMIT 100 RETURN { '
        DCTERMS_IDENTIFIER' : r.DCTERMS_IDENTIFIER }", filterString);
    documentCursor = arangoDriver.executeDocumentQuery(query, null, arangoDriver.
        getDefaultAqlQueryOptions(), BaseDocument.class);
    for (DocumentEntity<BaseDocument> documentEntity : documentCursor) {
     FillEntityRetrievalAndGraphScenariosDcTermsIdentifiers(documentEntity.getEntity());
    }
    break;
  }
 }

 @Override
 public QueryResult query(QueryScenario queryScenario) throws Exception {
  QueryResult queryResult = new QueryResult(queryScenario.queryResultType);
  String query;
  DocumentCursor<BaseDocument> documentCursor;

  switch(queryScenario.queryResultType){
   case COMPLETE_ENTITIES:
    switch(queryScenario){
     case CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES:
      query = "for resource in loddwhbench filter 'http://purl.org/dc/terms/BibliographicResource'
            in resource.RDF_TYPE return resource";
      documentCursor = arangoDriver.executeDocumentQuery(query, null, arangoDriver.
          getDefaultAqlQueryOptions(), BaseDocument.class);
      for (DocumentEntity<BaseDocument> documentEntity : documentCursor) {
       queryResult.push(BuildDataObjectFromDocument(documentEntity.getEntity()));
      }
      return queryResult;

     case CONDITIONAL_TABLE_SCAN_ALL_STUDIES:
      query = "for resource in fulltext(loddwhbench, 'DCTERMS_TITLE', 'study,|studie,|prefix:study
          ,|prefix:studie') return resource";
      documentCursor = arangoDriver.executeDocumentQuery(query, null, arangoDriver.
          getDefaultAqlQueryOptions(), BaseDocument.class);
      for (DocumentEntity<BaseDocument> documentEntity : documentCursor) {
       queryResult.push(BuildDataObjectFromDocument(documentEntity.getEntity()));
      }
      return queryResult;

     case CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES:
      query = "for r in loddwhbench filter (contains(r.DCTERMS_TITLE, 'Study', false) || contains(
          r.DCTERMS_TITLE, 'study', false) || contains(r.DCTERMS_TITLE, 'Studie', false) ||
          contains(r.DCTERMS_TITLE, 'studie', false)) && 'http://purl.org/dc/terms/
          BibliographicResource' in r.RDF_TYPE return r";
      documentCursor = arangoDriver.executeDocumentQuery(query, null, arangoDriver.
          getDefaultAqlQueryOptions(), BaseDocument.class);
      for (DocumentEntity<BaseDocument> documentEntity : documentCursor) {
       queryResult.push(BuildDataObjectFromDocument(documentEntity.getEntity()));
      }
```

```
     return queryResult;

    case ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY:
    case ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES:
    case ENTITY_RETRIEVAL_BY_ID_100_ENTITIES:
     for (String dcterms_identifier : EntityRetrievalAndGraphScenariosDcTermsIdentifiers) {
      query = String.format("FOR r IN loddwhbench FILTER r.DCTERMS_IDENTIFIER == '%s' RETURN r",
           dcterms_identifier);
      documentCursor = arangoDriver.executeDocumentQuery(query, null, arangoDriver.
           getDefaultAqlQueryOptions(), BaseDocument.class);
      for (DocumentEntity<BaseDocument> documentEntity : documentCursor) {
       queryResult.push(BuildDataObjectFromDocument(documentEntity.getEntity()));
      }
     }
     return queryResult;
    }
    break;



   case GRAPH:
    switch(queryScenario){
     case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY:
     case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES:
     case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES:
      for (String dcterms_identifier : EntityRetrievalAndGraphScenariosDcTermsIdentifiers) {
       query = String.format("for e in GRAPH_NEIGHBORS('loddwhbench_graph', {DCTERMS_IDENTIFIER:
            '%s'}, {direction: 'outbound', includeData: true}) return e", dcterms_identifier);
       documentCursor = arangoDriver.executeDocumentQuery(query, null, arangoDriver.
            getDefaultAqlQueryOptions(), BaseDocument.class);
       for(DocumentEntity<BaseDocument> documentEntity : documentCursor){
        BaseDocument subjectDocument = documentEntity.getEntity();
        String sharedSubject = subjectDocument.getAttribute("subject").toString();

        String innerQuery = String.format("for e in GRAPH_NEIGHBORS('loddwhbench_graph', {_id: '%s
             '}, {direction: 'inbound', includeData: true}) return { 'DCTERMS_IDENTIFIER': e.
            DCTERMS_IDENTIFIER }", documentEntity.getDocumentHandle());
        DocumentCursor<BaseDocument> innerCursor = arangoDriver.executeDocumentQuery(innerQuery,
             null, arangoDriver.getDefaultAqlQueryOptions(), BaseDocument.class);
        for(DocumentEntity<BaseDocument> documentEntity2 : innerCursor){
         String relatedDctermsIdentifier = documentEntity2.getEntity().getAttribute("
              DCTERMS_IDENTIFIER").toString();
         if(!dcterms_identifier.equals(relatedDctermsIdentifier)){
          queryResult.push(dcterms_identifier, sharedSubject, relatedDctermsIdentifier);
          //System.out.println(String.format("%s => %s <= %s", dcterms_identifier, sharedSubject,
               relatedDctermsIdentifier));
         }
        }
        innerCursor.close();
       }
       documentCursor.close();
      }
      return queryResult;


//     case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_2HOPS_ONE_ENTITY:
//     case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_2HOPS_10_ENTITIES:
//     case GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_2HOPS_100_ENTITIES:
//      for (String dcterms_identifier : EntityRetrievalAndGraphScenariosDcTermsIdentifiers) {
//       query = String.format("for e in GRAPH_NEIGHBORS('loddwhbench_graph', {DCTERMS_IDENTIFIER
    : '%s'}, {direction: 'outbound', includeData: true}) return e", dcterms_identifier);
//       documentCursor = arangoDriver.executeDocumentQuery(query, null, arangoDriver.
    getDefaultAqlQueryOptions(), BaseDocument.class);
//       for(DocumentEntity<BaseDocument> documentEntity : documentCursor){
//        BaseDocument subjectDocument = documentEntity.getEntity();
//        String sharedSubject = subjectDocument.getAttribute("subject").toString();
//
//        String innerQuery = String.format("for e in GRAPH_NEIGHBORS('loddwhbench_graph', {_id:
    '%s'}, {direction: 'inbound', includeData: true}) return { 'DCTERMS_IDENTIFIER': e.
    DCTERMS_IDENTIFIER, 'id': e._id }", documentEntity.getDocumentHandle());
//        DocumentCursor<BaseDocument> innerCursor = arangoDriver.executeDocumentQuery(innerQuery
    , null, arangoDriver.getDefaultAqlQueryOptions(), BaseDocument.class);
//        for(DocumentEntity<BaseDocument> documentEntity2 : innerCursor){
//         String relatedHandle = documentEntity2.getEntity().getAttribute("id").toString();
//         String relatedDctermsIdentifier = documentEntity2.getEntity().getAttribute("
    DCTERMS_IDENTIFIER").toString();
//
```

```
//         if(dcterms_identifier.equals(relatedDctermsIdentifier)) continue;
//
//         String secondInnerQuery = String.format("for e in GRAPH_NEIGHBORS('loddwhbench_graph',
     {_id: '%s'}, {direction: 'outbound', includeData: true}) return e", relatedHandle);
//         DocumentCursor<BaseDocument> secondInnerCursor = arangoDriver.executeDocumentQuery(
    secondInnerQuery, null, arangoDriver.getDefaultAqlQueryOptions(), BaseDocument.class);
//         for(DocumentEntity<BaseDocument> documentEntity3 : secondInnerCursor){
//          String secondSharedSubject = documentEntity3.getEntity().getAttribute("subject").
    toString();
//
//          //if(sharedSubject.equals(secondSharedSubject)) continue;
//
//          String thirdInnerQuery = String.format("for e in GRAPH_NEIGHBORS('loddwhbench_graph',
     {_id: '%s'}, {direction: 'inbound', includeData: true}) return { 'DCTERMS_IDENTIFIER': e.
    DCTERMS_IDENTIFIER }", documentEntity3.getDocumentHandle());
//          DocumentCursor<BaseDocument> thirdInnerCursor = arangoDriver.executeDocumentQuery(
    thirdInnerQuery, null, arangoDriver.getDefaultAqlQueryOptions(), BaseDocument.class);
//          for(DocumentEntity<BaseDocument> documentEntity4 : thirdInnerCursor){
//           String secondRelatedDctermsIdentifier = documentEntity4.getEntity().getAttribute("
    DCTERMS_IDENTIFIER").toString();
//
//           if(relatedDctermsIdentifier.equals(secondRelatedDctermsIdentifier)) continue;
//
//           queryResult.push(dcterms_identifier, sharedSubject, relatedDctermsIdentifier,
    secondSharedSubject, secondRelatedDctermsIdentifier);
//           System.out.println(String.format("%s => %s <= %s => %s <= %s", dcterms_identifier,
    sharedSubject, relatedDctermsIdentifier, secondSharedSubject, secondRelatedDctermsIdentifier)
    );
//          }
//          thirdInnerCursor.close();
//         }
//         secondInnerCursor.close();
//        }
//        innerCursor.close();
//       }
//       documentCursor.close();
//      }
//      return queryResult;
    }
    break;


   case TWO_COLUMNS:
    switch(queryScenario){
     case AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL:
      query = "FOR r IN loddwhbench COLLECT publisher = r.DCTERMS_PUBLISHER WITH COUNT INTO length
            SORT length DESC, publisher ASC RETURN { 'publisher' : publisher, 'count' : length }";
      documentCursor = arangoDriver.executeDocumentQuery(query, null, arangoDriver.
           getDefaultAqlQueryOptions(), BaseDocument.class);
      for (DocumentEntity<BaseDocument> documentEntity : documentCursor) {
       PushTwoColumnsToQueryResult(documentEntity.getEntity(), "publisher", "count", queryResult);
      }
      return queryResult;

     case AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10:
      query = "FOR r IN loddwhbench COLLECT publisher = r.DCTERMS_PUBLISHER WITH COUNT INTO length
            SORT length DESC, publisher ASC LIMIT 10 RETURN { 'publisher' : publisher, 'count' :
           length }";
      documentCursor = arangoDriver.executeDocumentQuery(query, null, arangoDriver.
           getDefaultAqlQueryOptions(), BaseDocument.class);
      for (DocumentEntity<BaseDocument> documentEntity : documentCursor) {
       PushTwoColumnsToQueryResult(documentEntity.getEntity(), "publisher", "count", queryResult);
      }
      return queryResult;

     case AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100:
      query = "FOR r IN loddwhbench COLLECT publisher = r.DCTERMS_PUBLISHER WITH COUNT INTO length
            SORT length DESC, publisher ASC LIMIT 100 RETURN { 'publisher' : publisher, 'count' :
           length }";
      documentCursor = arangoDriver.executeDocumentQuery(query, null, arangoDriver.
           getDefaultAqlQueryOptions(), BaseDocument.class);
      for (DocumentEntity<BaseDocument> documentEntity : documentCursor) {
       PushTwoColumnsToQueryResult(documentEntity.getEntity(), "publisher", "count", queryResult);
      }
      return queryResult;
```

```
  case AGGREGATE_ISSUES_PER_DECADE_ALL:
   query = "FOR r IN loddwhbench COLLECT decade = SUBSTRING(r.DCTERMS_ISSUED, 0, 3) WITH COUNT
        INTO length SORT length DESC, decade ASC RETURN { 'decade' : decade, 'count' : length }
        ";
   documentCursor = arangoDriver.executeDocumentQuery(query, null, arangoDriver.
        getDefaultAqlQueryOptions(), BaseDocument.class);
   for (DocumentEntity<BaseDocument> documentEntity : documentCursor) {
    PushTwoColumnsToQueryResult(documentEntity.getEntity(), "decade", "count", queryResult);
   }
   return queryResult;

  case AGGREGATE_ISSUES_PER_DECADE_TOP10:
   query = "FOR r IN loddwhbench COLLECT decade = SUBSTRING(r.DCTERMS_ISSUED, 0, 3) WITH COUNT
        INTO length SORT length DESC, decade ASC LIMIT 10 RETURN { 'decade' : decade, 'count' :
         length }";
   documentCursor = arangoDriver.executeDocumentQuery(query, null, arangoDriver.
        getDefaultAqlQueryOptions(), BaseDocument.class);
   for (DocumentEntity<BaseDocument> documentEntity : documentCursor) {
    PushTwoColumnsToQueryResult(documentEntity.getEntity(), "decade", "count", queryResult);
   }
   return queryResult;

  case AGGREGATE_ISSUES_PER_DECADE_TOP100:
   query = "FOR r IN loddwhbench COLLECT decade = SUBSTRING(r.DCTERMS_ISSUED, 0, 3) WITH COUNT
        INTO length SORT length DESC, decade ASC LIMIT 100 RETURN { 'decade' : decade, 'count'
        : length }";
   documentCursor = arangoDriver.executeDocumentQuery(query, null, arangoDriver.
        getDefaultAqlQueryOptions(), BaseDocument.class);
   for (DocumentEntity<BaseDocument> documentEntity : documentCursor) {
    PushTwoColumnsToQueryResult(documentEntity.getEntity(), "decade", "count", queryResult);
   }
   return queryResult;
  }
  break;


 case NONE:
  switch(queryScenario){
   case SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY:
   query = "FOR r IN loddwhbench UPDATE r WITH { newfield: 'cheesecake' } IN loddwhbench";
   arangoDriver.executeDocumentQuery(query, null, arangoDriver.getDefaultAqlQueryOptions(),
        BaseDocument.class);
   return queryResult;

   case SCHEMA_CHANGE_INTRODUCE_STRING_OP:
   query = "FOR r IN loddwhbench UPDATE r WITH { idSuffix: SUBSTRING(r.RDF_ABOUT, 29) } IN
        loddwhbench";
   arangoDriver.executeDocumentQuery(query, null, arangoDriver.getDefaultAqlQueryOptions(),
        BaseDocument.class);
   return queryResult;

   case SCHEMA_CHANGE_MIGRATE_RDF_TYPE:
   query = "FOR r IN loddwhbench UPDATE r WITH { manifestation: false, bibresource: false, book
        : false } IN loddwhbench";
   arangoDriver.executeDocumentQuery(query, null, arangoDriver.getDefaultAqlQueryOptions(),
        BaseDocument.class);
   query = "FOR r IN loddwhbench FILTER 'http://purl.org/dc/terms/BibliographicResource' in r.
        RDF_TYPE UPDATE r WITH { bibresource: true } IN loddwhbench";
   arangoDriver.executeDocumentQuery(query, null, arangoDriver.getDefaultAqlQueryOptions(),
        BaseDocument.class);
   query = "FOR r IN loddwhbench FILTER 'http://purl.org/vocab/frbr/core#Manifestation' in r.
        RDF_TYPE UPDATE r WITH { manifestation: true } IN loddwhbench";
   arangoDriver.executeDocumentQuery(query, null, arangoDriver.getDefaultAqlQueryOptions(),
        BaseDocument.class);
   query = "FOR r IN loddwhbench FILTER 'http://purl.org/ontology/bibo/Book' in r.RDF_TYPE
        UPDATE r WITH { book: true } IN loddwhbench";
   arangoDriver.executeDocumentQuery(query, null, arangoDriver.getDefaultAqlQueryOptions(),
        BaseDocument.class);
   return queryResult;

   case SCHEMA_CHANGE_REMOVE_RDF_TYPE:
   query = "FOR r IN loddwhbench UPDATE r WITH { RDF_TYPE: null } IN loddwhbench OPTIONS {
        keepNull: false }";
   arangoDriver.executeDocumentQuery(query, null, arangoDriver.getDefaultAqlQueryOptions(),
        BaseDocument.class);
   return queryResult;
```

```
    case UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM:
     query = "FOR r IN loddwhbench FILTER r.DCTERMS_MEDIUM == 'paper' UPDATE r WITH {
          DCTERMS_MEDIUM: 'recycled trees' } IN loddwhbench";
     arangoDriver.executeDocumentQuery(query, null, arangoDriver.getDefaultAqlQueryOptions(),
          BaseDocument.class);
     return queryResult;

    case UPDATE_HIGH_SELECTIVITY_NON_ISSUED:
     query = "FOR r IN loddwhbench FILTER !HAS(r, 'DCTERMS_ISSUED') UPDATE r WITH {
          DCTERMS_ISSUED: '0' } IN loddwhbench";
     arangoDriver.executeDocumentQuery(query, null, arangoDriver.getDefaultAqlQueryOptions(),
          BaseDocument.class);
     return queryResult;

    case DELETE_LOW_SELECTIVITY_PAPER_MEDIUM:
     query = "FOR r IN loddwhbench FILTER r.DCTERMS_MEDIUM == 'recycled trees' REMOVE r in
          loddwhbench";
     arangoDriver.executeDocumentQuery(query, null, arangoDriver.getDefaultAqlQueryOptions(),
          BaseDocument.class);
     return queryResult;

    case DELETE_HIGH_SELECTIVIY_NON_ISSUED:
     query = "FOR r IN loddwhbench FILTER r.DCTERMS_ISSUED == '0' REMOVE r in loddwhbench";
     arangoDriver.executeDocumentQuery(query, null, arangoDriver.getDefaultAqlQueryOptions(),
          BaseDocument.class);
     return queryResult;
   }
   break;
  }

  throw new RuntimeException("ArangoDB-query() finished without return");
 }

}
```

Listing B.151: Java Source Code of the ArangoDB Benchmark Implementation.

B.9 Helper Methods

```
package util.dumper;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.PrintStream;
import java.io.PrintWriter;
import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.function.Consumer;
import java.util.zip.GZIPInputStream;
import java.util.zip.GZIPOutputStream;
import java.util.zip.ZipException;

import org.marc4j.MarcReader;
import org.marc4j.MarcStreamReader;
import org.marc4j.MarcStreamWriter;
import org.marc4j.MarcWriter;
import org.marc4j.marc.Record;

import redis.clients.jedis.Jedis;
import util.Config;
import util.DataObject;
import util.Dataset;

public abstract class Helpers {

 public static String GetLinkConformString(String value) {
  return value.replace(' ', '-').replace('.', '-');
 }

 public static String DoubleToStringNDecimals(double value, int decimals, boolean
      transferToScientificNotation) {
```

```
 String format = "0.";
 for (int i = 0; i < decimals; i++)
  format += "0";
 if (transferToScientificNotation) {
  format += "E0";
 }
 return new DecimalFormat(format).format(value);
}

public static boolean isNumeric(String str) {
 return str.matches("-?\\d+(\\.\\d+)?"); // match a number with optional
          // '-' and decimal.
}

public static void compare(String name1, ArrayList<String> dump1, String name2, ArrayList<String>
      dump2) throws Exception {
 System.out.println(name1 + " :" + dump1.size());
 System.out.println(name2 + " :" + dump2.size());

 if (dump1.size() > dump2.size()) {
  dump1.removeAll(dump2);
  System.out.println("Did not find ids: " + dump1);
 } else if (dump2.size() > dump1.size()) {
  dump2.removeAll(dump1);
  System.out.println("Did not find ids: " + dump2);
 } else if (dump1.size() == dump2.size()) {
  System.out.println("Both sets are identical :)");
 }
}

public static ArrayList<String> idsFromRdfDump(File rdfDump) throws Exception {
 System.out.println("Scanning " + rdfDump + " for IDs...");
 BufferedReader inputReader = null;

 try {
  inputReader = new BufferedReader(new InputStreamReader(new GZIPInputStream(new FileInputStream(
       rdfDump))));
 } catch (ZipException e) {
  inputReader = new BufferedReader(new InputStreamReader(new FileInputStream(rdfDump)));
 }
 ArrayList<String> ids = new ArrayList<>();

 int counter = 0;
 String inline;
 while ((inline = inputReader.readLine()) != null) {
  if (inline.contains("<dcterms:identifier>")) {
   if (++counter % 100000 == 0) {
    System.out.println(counter + " RDF records so far...");
   }
   String id = (inline.substring(inline.lastIndexOf("<dcterms:identifier>") + 20, inline.
        lastIndexOf("</dcterms:identifier>")));
   ids.add(id.trim());
  }
 }
 System.out.println("... done");

 inputReader.close();
 return ids;
}

public static ArrayList<String> idsFromMarcDump(File marcDump) throws Exception {
 System.out.println("Scanning " + marcDump + " for IDs...");
 InputStream inputStream = null;
 try {
  inputStream = new GZIPInputStream(new FileInputStream(marcDump));
 } catch (ZipException e) {
  inputStream = new FileInputStream(marcDump);
 }
 MarcReader reader = new MarcStreamReader(inputStream);
 ArrayList<String> ids = new ArrayList<>();

 int counter = 0;
 while (reader.hasNext()) {

  if (++counter % 100000 == 0) {
   System.out.println(counter + " MARC records so far...");
  }
```

```
  try {
   Record record = reader.next();
   String recordId = record.getControlNumberField().getData().trim();
   ids.add(recordId);
  } catch (Exception e) {
  }
 }

 inputStream.close();

 return ids;
}

public static void readRdf(Dataset dumps, Consumer<DataObject> dataObjectConsumer) throws
     Exception {
 readRdf(dumps, dataObjectConsumer, null);
}

public static void readRdf(Dataset dumps, Consumer<DataObject> dataObjectConsumer, Consumer<
     Integer> intConsumer) throws Exception {
 readRdf(new GZIPInputStream(new FileInputStream(dumps.file)), dataObjectConsumer, intConsumer);
}

public static void readRdf(InputStream inputStream, Consumer<DataObject> dataObjectConsumer,
     Consumer<Integer> intConsumer) {

 try {

  // InputStream inputStream = new GZIPInputStream(new
  // FileInputStream(dumps.file));
  /*
   * Skips "Content-Type: application/rdf+xml; charset=UTF-8
   *
   * " at the beginning of each file - what could possibly go wrong?
   */
  // inputStream.skip(50);
  BufferedReader inputReader = new BufferedReader(new InputStreamReader(inputStream));
  // PrintWriter printWriter = new PrintWriter(new File(target));

  boolean headerWrittenYet = false;
  int counter = 0;// , writes = 0;
  String inline = "", record = "";
  while ((inline = inputReader.readLine()) != null) {
   // if (inline.contains("<dcterms:identifier>")) {
   // if (++counter % 100000 == 0) {
   // System.out.println(counter + " RDF records so far, " + writes
   // + " writes...");
   // }
   // }
   if (!headerWrittenYet && inline.contains("<rdf:Description")) {
    record = "";
    headerWrittenYet = true;
   }

   record += inline.trim() + "\n";

   if (inline.contains("</rdf:Description>")) {
    ++counter;
    DataObject dataObject = new DataObject();
    dataObject.fromRdfString(record);
    dataObjectConsumer.accept(dataObject);

    if (intConsumer != null)
     intConsumer.accept(counter);
    record = "";
   }
  }

  inputReader.close();
 } catch (Exception e) {
  throw new RuntimeException(e);
 }
}

public static int countRdf(Dataset dataset) {
 int counter = 0;
```

```
 try {

  BufferedReader inputReader = new BufferedReader(new InputStreamReader(new GZIPInputStream(new
       FileInputStream(dataset.file))));

  boolean headerWrittenYet = false;
  // , writes = 0;
  String inline = "", record = "";
  while ((inline = inputReader.readLine()) != null) {
   // if (inline.contains("<dcterms:identifier>")) {
   // if (++counter % 100000 == 0) {
   // System.out.println(counter + " RDF records so far, " + writes
   // + " writes...");
   // }
   // }
   if (!headerWrittenYet && inline.contains("<rdf:Description")) {
    record = "";
    headerWrittenYet = true;
   }

   record += inline.trim() + "\n";

   if (inline.contains("</rdf:Description>")) {
    ++counter;
    record = "";
   }
  }

  inputReader.close();
 } catch (Exception e) {
  throw new RuntimeException(e);
 }

 return counter;
}

public static void writeRdfExtract(ArrayList<String> wantedIds, Dataset source, String target)
     throws Exception {
 System.out.println("Scanning " + source + " for entites...");
 BufferedReader inputReader = new BufferedReader(new InputStreamReader(new GZIPInputStream(new
      FileInputStream(source.file))));
 PrintWriter printWriter = new PrintWriter(new File(target));

 boolean headerWrittenYet = false;
 int counter = 0, writes = 0;
 String inline = "", record = "", id = "";
 while ((inline = inputReader.readLine()) != null) {
  if (inline.contains("<dcterms:identifier>")) {
   if (++counter % 100000 == 0) {
    System.out.println(counter + " RDF records so far, " + writes + " writes...");
   }
   id = (inline.substring(inline.lastIndexOf("<dcterms:identifier>") + 20, inline.lastIndexOf("</
        dcterms:identifier>"))).trim();
  }
  if (!headerWrittenYet && inline.contains("<rdf:Description")) {
   printWriter.write(record);
   record = "\n";
  }

  record += inline.trim() + "\n";

  if (inline.contains("</rdf:Description>")) {

   if (wantedIds.contains(id)) {
    printWriter.write(record);
    writes++;
    wantedIds.remove(id);
   }
   record = "";

   if (wantedIds.isEmpty())
    break;
  }

 }
 printWriter.write("\n</rdf:RDF>");
 System.out.println("... done");
```

```
 printWriter.close();
 inputReader.close();
}

public static void writeRdfIntoRedis(Jedis jedis, Dataset dumps) throws Exception {
 BufferedReader inputReader = new BufferedReader(new InputStreamReader(new GZIPInputStream(new
      FileInputStream(dumps.file))));

 boolean headerWrittenYet = false;
 int counter = 0, errors = 0;
 String inline = "", record = "", id = "";
 while ((inline = inputReader.readLine()) != null) {
  if (inline.contains("<dcterms:identifier>")) {
   if (++counter % 100000 == 0) {
    System.out.println("Stored " + counter + " RDF records so far...");
   }
   try {
    id = (inline.substring(inline.lastIndexOf("<dcterms:identifier>") + 20, inline.lastIndexOf("
        </dcterms:identifier>"))).trim();
   } catch (StringIndexOutOfBoundsException e) {
    errors++;
    id = "error";
   }
  }
  if (!headerWrittenYet && inline.contains("<rdf:Description")) {
   jedis.set("RDFHEAD", record);
   record = "";
   headerWrittenYet = true;
  }

  record += inline.trim() + "\n";

  if (inline.contains("</rdf:Description>")) {
   jedis.set(id, record);
   record = "";
  }

 }

 System.out.println("\nWrote " + counter + " records, " + errors + " errors.");
 inputReader.close();
}

public static void writeMarcExtract(ArrayList<String> wantedIds, String target) throws Exception
    {
 writeMarcExtract(wantedIds, Dataset.HeBIS_Hauptbestand_in_MARC_gz.file, target);
}

public static void writeMarcExtract(ArrayList<String> wantedIds, File marcDump, String target)
     throws Exception {
 InputStream inputStream = new GZIPInputStream(new FileInputStream(marcDump));
 // InputStream inputStream = new FileInputStream("untitled");
 MarcReader reader = new MarcStreamReader(inputStream);

 MarcWriter writer = new MarcStreamWriter(new GZIPOutputStream(new FileOutputStream(target)));

 int sizeOfWantedIds = wantedIds.size();

 System.out.println("Scanning MARC records for " + wantedIds.size() + " ids ...");
 ArrayList<String> foundIds = new ArrayList<>();

 int counter = 0;
 while (reader.hasNext()) {
  if (counter++ % 100000 == 0) {
   System.out.println("Scanned " + counter + " so far...");
  }

  try {
   Record record = reader.next();
   String recordId = record.getControlNumberField().getData().trim();

   if (wantedIds.contains(recordId.trim())) {
    foundIds.add(recordId);
    writer.write(record);
    wantedIds.remove(recordId);
```

```
    if (foundIds.size() % 100000 == 0)
     System.out.println("Found \t" + foundIds.size() + " / " + sizeOfWantedIds + ":\t" + recordId
         );

   }
   if (foundIds.size() == sizeOfWantedIds) {
    System.out.println("All records found :)");
    break;
   }
  } catch (Exception e) {
  }
 }
 writer.close();
 inputStream.close();

 if (wantedIds.size() != 0) {
  System.out.println("WARNING: The following IDs could not be found in MARC: " + wantedIds);
 }
}

protected static String minId(ArrayList<String> ids) {
 Integer minId = 999999999;
 for (String id : ids) {

  try {
   Integer idi = new Integer(id);
   if (idi < minId) {
    minId = idi;
   }
  } catch (Exception e) {
  }
 }
 return minId + "";
}

protected static String maxId(ArrayList<String> ids) {
 Integer maxId = 0;
 for (String id : ids) {

  try {
   Integer idi = new Integer(id);
   if (idi > maxId) {
    maxId = idi;
   }
  } catch (Exception e) {
  }
 }
 return maxId + "";
}

protected static void dumpMarc(String minimumId, Dataset marcDump, File file) throws Exception {
 InputStream inputStream = new GZIPInputStream(new FileInputStream(marcDump.file));
 MarcReader reader = new MarcStreamReader(inputStream);

 MarcWriter writer = new MarcStreamWriter(new GZIPOutputStream(new FileOutputStream(file)));

 int counter = 0;
 while (reader.hasNext()) {

  if (++counter % 100000 == 0) {
   System.out.println(counter + " records so far...");
  }

  try {
   Record record = reader.next();
   String recordId = record.getControlNumberField().getData().trim();

   try {
    if (new Integer(recordId) < new Integer(minimumId))
     continue;
   } catch (Exception e) {

   }

   writer.write(record);
  } catch (Exception e) {
  }
```

```
 }

 inputStream.close();
 writer.close();

}

protected static void validate(File file) {
 // Model m = new Mo
 // new Model.read(new GZIPInputStream(new FileInputStream(file)),
 // "www.example.com", "RDF/XML");
}

protected static void produceIdenticalMarcAndRdfDump(Jedis jedis, String target, int quanity)
     throws Exception {
 InputStream inputStream = new GZIPInputStream(new FileInputStream(Dataset.
      HeBIS_Hauptbestand_in_MARC_gz.file));
 MarcReader reader = new MarcStreamReader(inputStream);

 MarcWriter writer = new MarcStreamWriter(new GZIPOutputStream(new FileOutputStream(target + "
      _marc.gz")));
 PrintStream printStream = new PrintStream(new GZIPOutputStream(new FileOutputStream(target + "
      _rdf.gz")), false, "UTF-8");

 // OutputFileStream outputStream = new GZIPOutputStream(new
 // FileOutputStream(new File(target)));

 printStream.print(jedis.get("RDFHEAD") + "\n");
 int counter = 0, written = 0;
 while (reader.hasNext()) {
  if (++counter % 100000 == 0) {
   System.out.println("Passed " + counter + " records so far...");
  }
  if (counter < 10000000) {
   try {
    reader.next();
   } catch (Exception e) {
   }
   continue;
  }

  if (written >= quanity) {
   System.out.println("Wrote " + quanity + " records.");
   break;
  }

  try {
   Record record = reader.next();

   if (Math.random() * 10 < 7)
    continue;

   String recordId = record.getControlNumberField().getData().trim();
   // System.out.println(recordId);
   String rdfRecord = jedis.get(recordId);
   if (rdfRecord != null) {
    written++;
    writer.write(record);
    printStream.print(rdfRecord + "\n");
   }

  } catch (Exception e) {
  }
 }

 printStream.print("\n</rdf:RDF>");
 writer.close();
 inputStream.close();
 printStream.close();

}

public static void terminalLaunchScript(String command, int waitSeconds) {
 if (!Config.THIS_IS_OSX)
  return;

 try {
```

```
   command = Helpers.class.getResource("/shell/" + command).getFile();
   ProcessBuilder processBuilder = new ProcessBuilder("open", "-j", "-g", "-a", "Terminal.app",
        command);
   Process process = processBuilder.start();
   Thread.sleep(waitSeconds * 1000);
  } catch (Exception e) {
   throw new RuntimeException("Cannot launch " + command, e);
  }
 }

 public static void terminalLaunchApp(String... commands) {
  if (!Config.THIS_IS_OSX)
   return;

  try {
   ProcessBuilder processBuilder = new ProcessBuilder(commands);
   Process process = processBuilder.start();
   process.waitFor();
  } catch (Exception e) {
   throw new RuntimeException("Cannot launch " + commands, e);
  }
 }
}
```

Listing B.152: Java Source Code of helper methods to parse data.

B.10 Entity Representation

```
package util;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

import org.apache.commons.lang3.StringEscapeUtils;
import org.marc4j.marc.ControlField;
import org.marc4j.marc.DataField;
import org.marc4j.marc.Record;
import org.marc4j.marc.VariableField;

/**
 * A RAM-loaded entity interpretation. Use / extend as required.
 */
public class DataObject {
 HashMap<Codes, Object> data = new HashMap<>();

 public void putMultiple(Codes code, String value) {
  ArrayList<String> returnValue = (ArrayList<String>) data.get(code);
  if (returnValue == null) {
   returnValue = new ArrayList<String>();
  }

  returnValue.add(value);
  data.put(code, returnValue);
 }

 @Override
 public String toString() {
  String result = "";

  for(Codes outerCode : Codes.values()){
   if(data.containsKey(outerCode)){
    result += outerCode.toString() + ":\t" + data.get(outerCode) + "\n";
   }
  }

  return result;
 }

 public String getId() {
  return (String) data.get(Codes.DCTERMS_IDENTIFIER);
 }

 public void fromMarcRecord(Record record) {
```

```
// Extract subfield a
for (Codes code : Codes.values()) {
 try {
  if (code.MARC_CODE != null && code.SUBFIELD != '0') {
   data.put(code, ((DataField) record.getVariableField(code.MARC_CODE)).getSubfield(code.
       SUBFIELD).getData());
  }
 } catch (Exception e) {
 }
}

// Overwrite special cases:
boolean aBibliographicResource = false, aBook = false;
// Fixed-Length data fields, ftw.
for (ControlField controlField : record.getControlFields()) {
 switch (controlField.getTag()) {
 case "006":
  // data.put(Cocdes.TEST_006, controlField.getData());
  String formOfMaterial = controlField.getData().substring(0, 1);

  switch (formOfMaterial) {
  case "e": // "Cartographic material" - FAll through
  case "c": // c - Notated music
  case "a": // "Language material" -> Book
   aBook = true;
   aBibliographicResource = true;
   break;
  case "m": // Computer file/Electronic resource
  case "j": // "Musical sound recording"
   aBook = false;
   aBibliographicResource = false;
   break;
  case "s": // s - Serial/Integrating resource
   aBook = false;
   aBibliographicResource = true;
   break;
  default:
   // throw new
   // RuntimeException("Dont know this FORM OF MATERIAL: " +
   // formOfMaterial +
   // ", see
   // http://www.loc.gov/marc/bibliographic/bd006.html");
  }
  break;

 case "007": // Might usually not exist
  String unknownProperty = controlField.getData().substring(0, 1);
  switch (unknownProperty) {
  case "q":
  case "a":
   aBibliographicResource = true;
   aBook = true;
  case "h":
   aBibliographicResource = false;
   aBook = true;
   break;
  case "s":
  case "c":
   aBibliographicResource = false;
   aBook = false;
   break;
  default:
   throw new RuntimeException("Dont know this: " + unknownProperty + " @ " + record.
       getControlNumber());
  }

  break;
 }
}

if (aBook) {
 putMultiple(Codes.RDF_TYPE, "http://purl.org/ontology/bibo/Book");
}
if (aBibliographicResource) {
 // This relation is confirmed
 putMultiple(Codes.RDF_TYPE, "http://purl.org/dc/terms/BibliographicResource");
```

```
  data.put(Codes.DCTERMS_MEDIUM, Codes.DCTERMS_MEDIUM.CONSTANT);
  data.put(Codes.DCTERMS_FORMAT, Codes.DCTERMS_FORMAT.CONSTANT);
 }
 putMultiple(Codes.RDF_TYPE, "http://purl.org/vocab/frbr/core#Manifestation");

 // IDs
 // $a(DE-599)DNB860319695 -> DCTERMS_IDENTIFIER 860319695
 data.remove(Codes.DCTERMS_IDENTIFIER);
 String id = record.getControlNumberField().getData();
 data.put(Codes.DCTERMS_IDENTIFIER, id);
 data.put(Codes.RDF_ABOUT, Codes.RDF_ABOUT.CONSTANT + id.substring(0, id.length() - 1));
 data.put(Codes.WDRS_DESCRIBEDBY, Codes.WDRS_DESCRIBEDBY.CONSTANT + id.substring(0, id.length()
     - 1));

 // DNB linkage 035 $a(DE-599)DNB860319695
 data.remove(Codes.OWL_SAMEAS);
 for (VariableField field : record.getVariableFields(Codes.OWL_SAMEAS.MARC_CODE)) {
  String systemControlNumber = ((DataField) field).getSubfield(Codes.OWL_SAMEAS.SUBFIELD).getData
      ();
  if (systemControlNumber.startsWith("(DE-599)DNB")) {
   data.put(Codes.OWL_SAMEAS, "http://d-nb.info/" + systemControlNumber.substring(11));
  }
 }

 // 035 $a(OCoLC)43712277
 Codes code = Codes.BIBO_OCLCNUM;
 data.remove(code);
 for (VariableField field : record.getVariableFields(code.MARC_CODE)) {
  String fieldValue = ((DataField) field).getSubfield(code.SUBFIELD).getData();
  if (fieldValue.startsWith("(OCoLC)")) {
   data.put(code, fieldValue.substring(7));
  }
 }

 // 650 7$0(DE-588)4133806-6$0(DE-603)085487139$aChemische Synthese$2gnd
 // GND == 4133806-6
 code = Codes.DCTERMS_SUBJECT;
 data.remove(code);
 for (VariableField field : record.getVariableFields(code.MARC_CODE)) {
  String systemControlNumber = ((DataField) field).getSubfield(code.SUBFIELD).getData();
  if (systemControlNumber.startsWith("(DE-588)")) {
   putMultiple(code, systemControlNumber.substring(8));
  }
 }
}

public void fromRdfString(String record) {
 for (Codes code : Codes.values()) {
  Pattern rdfPattern = code.rdfPattern;

  try {
   if (rdfPattern != null) {
    Matcher matcher = rdfPattern.matcher(record);

    while (matcher.find()) {
     String value = matcher.group(1).trim();
     value = StringEscapeUtils.unescapeHtml4(value);
     if (code.IS_MULTIPLE)
      putMultiple(code, value);
     else
      data.put(code, value);
    }

   } else {
    System.out.println("HOW TO READ " + code + " ??");
   }
  } catch (Exception e) {
   throw new RuntimeException("HOW TO READ " + code + " ??");
  }
 }
}

public Object get(Codes code) {
 return data.get(code);
}

public void set(Codes key, String value) {
```

```
  this.data.put(key, value);
 }

 @Override
 public int hashCode() {
  final int prime = 31;
  int result = 1;
  result = prime * result + ((data == null) ? 0 : data.hashCode());
  return result;
 }

 @Override
 public boolean equals(Object obj) {
  if (this == obj)
   return true;
  if (obj == null)
   return false;
  if (getClass() != obj.getClass())
   return false;
  DataObject other = (DataObject) obj;
  if (data == null) {
   if (other.data != null)
    return false;
  } else if (!data.equals(other.data))
   return false;

  for (Codes code : Codes.values()) {
   if (this.data.get(code) == null) {
    if (other.data.get(code) != null)
     return false;
   }

   System.out.println(data);
   if (this.data.get(code) != other.data.get(code))
    return false;
  }

  return true;
 }
}
```

Listing B.153: Java Source Code of Entity Object.

B.11 Database Start and Initialisation Scripts

B.11.1 PostgreSQL Initialisation and Start Script

```
#!/bin/bash

rm -R /usr/local/var/loddwhbench
initdb -E UTF8 -D /usr/local/var/loddwhbench

pgtune -i /usr/local/var/loddwhbench/postgresql.conf -o /usr/local/var/loddwhbench/postgresql.
    conf -M 17179869184

pg_ctl -D /usr/local/var/loddwhbench start
#read -p "Press [Enter] to stop PostgreSQL..."
#pg_ctl -D /usr/local/var/loddwhbench stop -m immediate
```

Listing B.154: Bash Source Code of the Start Script for PostgreSQL.

B.11.2 MongoDB Initialisation and Start Script

```
#!/bin/bash

rm -R /data/db/
mkdir /data/db

/usr/local/bin/mongod
```

Listing B.155: Bash Source Code of the Start Script for MongoDB.

B.11.3 ArangoDB Initialisation and Start Script

```
#!/bin/bash

rm -R ~/arangoDB/

sleep 30 && arangosh --javascript.execute-string "db._createDatabase('loddwhbench')" &
/usr/local/Cellar/arangodb/2.6.9/sbin/arangod ~/arangoDB/
```

Listing B.156: Bash Source Code of the Start Script for ArangoDB.

B.11.4 Virtuoso Initialisation and Start Script

```
#!/bin/bash

cd /usr/local/virtuoso-opensource/var/lib/virtuoso/db

rm virtuoso.db
rm virtuoso.log
rm virtuoso.trx
rm virtuoso.pxa
rm virtuoso-temp.db

virtuoso-t +foreground
```

Listing B.157: Bash Source Code of the Start Script for Virtuoso.

B.11.5 Fuseki Initialisation and Start Script

```
#!/bin/bash

cd "$HOME/Ph.D./Applications/DB/apache-jena-fuseki-2.3.0/"

rm -Rf run
rm -Rf DB
mkdir DB
./fuseki-server --timeout=3600000,3600000 --loc=DB /DB
```

Listing B.158: Bash Source Code of the Start Script for Fuseki.

B.12 Evaluation Report for Test Series Tiny

B.12.1 Evaluation Overview, all Databases, Test Series TINY

QueryScenario	Phase	sqlite4java	SQLite-Xerial	PostgreSQL	Virtuoso	Fuseki	MongoDB	ArangoDB
Set up		19.21 ms	95.77 ms	257.17 ms	0.94 ms	**0.00 ms**	5.88 ms	20.65 ms
Load		329.39 ms	266.72 ms	500.93 ms	**121.22 ms**	1061.05 ms	580.58 ms	726.13 ms
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	Prepare	**6.24 ms**	7.66 ms	15.99 ms	101.60 ms	234.38 ms	15.66 ms	11.10 ms
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	Query	1.38 ms	**0.23 ms**	1.02 ms	63.13 ms	34.00 ms	0.82 ms	1.80 ms
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	Prepare	**1.59 ms**	6.90 ms	30.35 ms	6.08 ms	87.40 ms	6.68 ms	4.96 ms
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	Query	1.10 ms	**0.47 ms**	2.53 ms	167.30 ms	38.52 ms	6.15 ms	22.13 ms
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	Prepare	**2.31 ms**	7.20 ms	28.53 ms	12.82 ms	149.42 ms	8.13 ms	13.81 ms
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	Query	2.68 ms	**2.05 ms**	5.19 ms	358.24 ms	133.77 ms	55.90 ms	17.48 ms
AGG._PUBLICATIONS_PER_PUBLISHER_TOP10	Prepare	0.48 ms	4.12 ms	13.05 ms	2.13 ms	1.09 ms	**0.00 ms**	**0.00 ms**
AGG._PUBLICATIONS_PER_PUBLISHER_TOP10	Query	0.68 ms	**0.28 ms**	0.99 ms	0.46 ms	28.55 ms	4.75 ms	8.35 ms
AGG._PUBLICATIONS_PER_PUBLISHER_TOP100	Prepare	0.54 ms	3.92 ms	27.86 ms	1.39 ms	0.64 ms	**0.00 ms**	**0.00 ms**
AGG._PUBLICATIONS_PER_PUBLISHER_TOP100	Query	0.62 ms	**0.31 ms**	1.39 ms	0.59 ms	12.56 ms	1.95 ms	3.47 ms
AGG._PUBLICATIONS_PER_PUBLISHER_ALL	Prepare	0.48 ms	4.00 ms	23.87 ms	1.66 ms	0.71 ms	**0.00 ms**	**0.00 ms**
AGG._PUBLICATIONS_PER_PUBLISHER_ALL	Query	0.62 ms	**0.28 ms**	0.93 ms	0.62 ms	13.50 ms	1.86 ms	3.29 ms
AGG._ISSUES_PER_DECADE_TOP10	Prepare	3.95 ms	6.76 ms	27.34 ms	1.79 ms	0.71 ms	**0.00 ms**	**0.00 ms**
AGG._ISSUES_PER_DECADE_TOP10	Query	**0.89 ms**	1.70 ms	1.17 ms	2.87 ms	19.63 ms	1.85 ms	21.49 ms
AGG._ISSUES_PER_DECADE_TOP100	Prepare	0.62 ms	5.74 ms	28.94 ms	1.56 ms	0.62 ms	**0.00 ms**	**0.00 ms**
AGG._ISSUES_PER_DECADE_TOP100	Query	**0.91 ms**	1.55 ms	1.22 ms	2.88 ms	14.80 ms	1.84 ms	12.92 ms
AGG._ISSUES_PER_DECADE_ALL	Prepare	0.57 ms	5.77 ms	28.31 ms	1.74 ms	0.54 ms	**0.00 ms**	**0.00 ms**
AGG._ISSUES_PER_DECADE_ALL	Query	**0.96 ms**	1.55 ms	1.34 ms	2.84 ms	11.95 ms	1.84 ms	13.95 ms
CONDITIONAL_TABLE_SCAN_ALL_STUDIES	Prepare	0.61 ms	4.54 ms	12.11 ms	6.96 ms	2.18 ms	**0.00 ms**	45.35 ms
CONDITIONAL_TABLE_SCAN_ALL_STUDIES	Query	**2.29 ms**	2.89 ms	2.45 ms	231.78 ms	133.41 ms	2.97 ms	3.39 ms
COND._T._S._ALL_BIB._RES.	Prepare	0.57 ms	1.82 ms	11.78 ms	8.22 ms	1.60 ms	**0.00 ms**	**0.00 ms**
COND._T._S._ALL_BIB._RES.	Query	15.26 ms	16.23 ms	14.46 ms	1962.31 ms	757.57 ms	**8.54 ms**	47.13 ms
COND._T._S._ALL_BIB._RES._AND_STUDIES	Prepare	0.65 ms	1.68 ms	28.68 ms	7.66 ms	1.34 ms	**0.00 ms**	**0.00 ms**
COND._T._S._ALL_BIB._RES._AND_STUDIES	Query	**2.25 ms**	3.36 ms	2.45 ms	231.65 ms	288.08 ms	3.15 ms	13.36 ms
GRAPH_L._REL._BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY	Prepare	209.98 ms	186.85 ms	42.63 ms	3.20 ms	13.00 ms	**0.80 ms**	121.40 ms
GRAPH_L._REL._BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY	Query	0.10 ms	**0.09 ms**	0.80 ms	0.72 ms	7.91 ms	2.78 ms	4.97 ms
GRAPH_L._REL._BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES	Prepare	2.55 ms	2.45 ms	11.49 ms	1.83 ms	10.83 ms	**1.32 ms**	8.11 ms

QueryScenario	Phase	sqlite4java	SQLite-Xerial	PostgreSQL	Virtuoso	Fuseki	MongoDB	ArangoDB
GRAPH_L._REL._BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES	Query	**0.10 ms**	0.12 ms	0.72 ms	1.13 ms	8.35 ms	19.65 ms	31.05 ms
GRAPH_L._REL._BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES	Prepare	**1.05 ms**	2.28 ms	11.94 ms	1.82 ms	10.88 ms	1.25 ms	11.50 ms
GRAPH_L._REL._BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES	Query	**0.10 ms**	0.12 ms	0.96 ms	1.12 ms	6.82 ms	17.61 ms	212.67 ms
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	Prepare	3.56 ms	1.21 ms	8.94 ms	1.55 ms	3.16 ms	**0.00 ms**	**0.00 ms**
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	Query	0.95 ms	**0.83 ms**	32.83 ms	158.15 ms	236.66 ms	10.27 ms	53.24 ms
SCHEMA_CHANGE_INTRODUCE_STRING_OP	Prepare	0.66 ms	4.21 ms	14.26 ms	2.69 ms	1.22 ms	**0.00 ms**	**0.00 ms**
SCHEMA_CHANGE_INTRODUCE_STRING_OP	Query	7.52 ms	8.60 ms	27.75 ms	**2.17 ms**	7.66 ms	187.42 ms	66.43 ms
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	Prepare	0.90 ms	1.69 ms	26.15 ms	5.73 ms	3.11 ms	**0.00 ms**	**0.00 ms**
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	Query	**12.17 ms**	16.79 ms	135.54 ms	1010.09 ms	368.64 ms	17.36 ms	234.75 ms
SCHEMA_CHANGE_REMOVE_RDF_TYPE	Prepare	1.18 ms	1.85 ms	10.30 ms	2.20 ms	1.04 ms	**0.00 ms**	**0.00 ms**
SCHEMA_CHANGE_REMOVE_RDF_TYPE	Query	4.43 ms	7.41 ms	16.27 ms	**0.21 ms**	5.33 ms	3.38 ms	58.49 ms
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	Prepare	0.49 ms	4.54 ms	15.56 ms	1.31 ms	0.99 ms	**0.00 ms**	**0.00 ms**
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	Query	5.32 ms	9.39 ms	40.54 ms	20.29 ms	33.38 ms	**3.30 ms**	57.54 ms
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	Prepare	0.44 ms	4.12 ms	15.46 ms	1.67 ms	23.21 ms	**0.00 ms**	**0.00 ms**
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	Query	1.00 ms	0.78 ms	2.04 ms	2.08 ms	73.33 ms	**0.58 ms**	9.66 ms
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	Prepare	0.34 ms	4.05 ms	29.06 ms	1.56 ms	1.24 ms	**0.00 ms**	**0.00 ms**
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	Query	**2.56 ms**	5.32 ms	2.73 ms	512.00 ms	5.31 ms	5.45 ms	4.21 ms
DELETE_HIGH_SELECTIVIY_NON_ISSUED	Prepare	0.31 ms	2.50 ms	27.70 ms	27.79 ms	0.72 ms	**0.00 ms**	**0.00 ms**
DELETE_HIGH_SELECTIVIY_NON_ISSUED	Query	0.06 ms	**0.04 ms**	1.28 ms	2.14 ms	3.67 ms	0.26 ms	0.69 ms

B.12.2 Test Series Datasets

Test Series TINY	Dataset Size
hebis_1000_records	0.077 MB

B.12.3 Evaluation Details for sqlite4java, Version 392 with SQLite 3.8.7, Test Series TINY

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
Set up		19.21 ms			
Load		329.39 ms			
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	Prepare	6.24 ms			
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	Query	3.77 ms	0.19 ms	0.16 ms	1.38 ms
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	Prepare	1.59 ms			
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	Query	0.85 ms	0.72 ms	1.75 ms	1.10 ms
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	Prepare	2.31 ms			
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	Query	3.07 ms	2.55 ms	2.43 ms	2.68 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10	Prepare	0.48 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10	Query	0.89 ms	0.60 ms	0.55 ms	0.68 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100	Prepare	0.54 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100	Query	0.66 ms	0.63 ms	0.59 ms	0.62 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL	Prepare	0.48 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL	Query	0.64 ms	0.61 ms	0.62 ms	0.62 ms
AGGREGATE_ISSUES_PER_DECADE_TOP10	Prepare	3.95 ms			
AGGREGATE_ISSUES_PER_DECADE_TOP10	Query	0.94 ms	0.87 ms	0.88 ms	0.89 ms
AGGREGATE_ISSUES_PER_DECADE_TOP100	Prepare	0.62 ms			
AGGREGATE_ISSUES_PER_DECADE_TOP100	Query	0.93 ms	0.91 ms	0.90 ms	0.91 ms
AGGREGATE_ISSUES_PER_DECADE_ALL	Prepare	0.57 ms			
AGGREGATE_ISSUES_PER_DECADE_ALL	Query	0.94 ms	0.86 ms	1.07 ms	0.96 ms
CONDITIONAL_TABLE_SCAN_ALL_STUDIES	Prepare	0.61 ms			
CONDITIONAL_TABLE_SCAN_ALL_STUDIES	Query	2.50 ms	2.29 ms	2.09 ms	2.29 ms
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES	Prepare	0.57 ms			
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES	Query	17.67 ms	14.48 ms	13.62 ms	15.26 ms
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES	Prepare	0.65 ms			
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES	Query	2.49 ms	2.18 ms	2.09 ms	2.25 ms
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY	Prepare	209.98 ms			
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY	Query	0.10 ms	0.09 ms	0.11 ms	0.10 ms
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES	Prepare	2.55 ms			
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES	Query	0.11 ms	0.10 ms	0.10 ms	0.10 ms
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES	Prepare	1.05 ms			
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES	Query	0.11 ms	0.10 ms	0.10 ms	0.10 ms
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	Prepare	3.56 ms			

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	Query	0.95 ms			
SCHEMA_CHANGE_INTRODUCE_STRING_OP	Prepare	0.66 ms			
SCHEMA_CHANGE_INTRODUCE_STRING_OP	Query	7.52 ms			
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	Prepare	0.90 ms			
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	Query	12.17 ms			
SCHEMA_CHANGE_REMOVE_RDF_TYPE	Prepare	1.18 ms			
SCHEMA_CHANGE_REMOVE_RDF_TYPE	Query	4.43 ms			
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	Prepare	0.49 ms			
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	Query	5.32 ms			
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	Prepare	0.44 ms			
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	Query	1.00 ms			
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	Prepare	0.34 ms			
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	Query	2.56 ms			
DELETE_HIGH_SELECTIVIY_NON_ISSUED	Prepare	0.31 ms			
DELETE_HIGH_SELECTIVIY_NON_ISSUED	Query	0.06 ms			

B.12.4 Evaluation Details for SQLite-Xerial, Version 3.8.11, Test Series TINY

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
Set up		95.77 ms			
Load		266.72 ms			
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	Prepare	7.66 ms			
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	Query	0.41 ms	0.15 ms	0.14 ms	0.23 ms
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	Prepare	6.90 ms			
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	Query	0.52 ms	0.51 ms	0.38 ms	0.47 ms
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	Prepare	7.20 ms			
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	Query	2.03 ms	2.03 ms	2.08 ms	2.05 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10	Prepare	4.12 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10	Query	0.31 ms	0.27 ms	0.27 ms	0.28 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100	Prepare	3.92 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100	Query	0.34 ms	0.31 ms	0.30 ms	0.31 ms

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL	Prepare	4.00 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL	Query	0.30 ms	0.27 ms	0.26 ms	0.28 ms
AGGREGATE_ISSUES_PER_DECADE_TOP10	Prepare	6.76 ms			
AGGREGATE_ISSUES_PER_DECADE_TOP10	Query	2.20 ms	1.47 ms	1.42 ms	1.70 ms
AGGREGATE_ISSUES_PER_DECADE_TOP100	Prepare	5.74 ms			
AGGREGATE_ISSUES_PER_DECADE_TOP100	Query	2.03 ms	1.36 ms	1.26 ms	1.55 ms
AGGREGATE_ISSUES_PER_DECADE_ALL	Prepare	5.77 ms			
AGGREGATE_ISSUES_PER_DECADE_ALL	Query	2.01 ms	1.38 ms	1.26 ms	1.55 ms
CONDITIONAL_TABLE_SCAN_ALL_STUDIES	Prepare	4.54 ms			
CONDITIONAL_TABLE_SCAN_ALL_STUDIES	Query	3.24 ms	2.61 ms	2.80 ms	2.89 ms
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES	Prepare	1.82 ms			
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES	Query	15.93 ms	15.55 ms	17.21 ms	16.23 ms
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES	Prepare	1.68 ms			
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES	Query	3.94 ms	2.95 ms	3.20 ms	3.36 ms
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY	Prepare	186.85 ms			
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY	Query	0.10 ms	0.09 ms	0.08 ms	0.09 ms
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES	Prepare	2.45 ms			
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES	Query	0.13 ms	0.12 ms	0.12 ms	0.12 ms
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES	Prepare	2.28 ms			
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES	Query	0.12 ms	0.12 ms	0.11 ms	0.12 ms
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	Prepare	1.21 ms			
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	Query	0.83 ms			
SCHEMA_CHANGE_INTRODUCE_STRING_OP	Prepare	4.21 ms			
SCHEMA_CHANGE_INTRODUCE_STRING_OP	Query	8.60 ms			
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	Prepare	1.69 ms			
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	Query	16.79 ms			
SCHEMA_CHANGE_REMOVE_RDF_TYPE	Prepare	1.85 ms			
SCHEMA_CHANGE_REMOVE_RDF_TYPE	Query	7.41 ms			
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	Prepare	4.54 ms			
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	Query	9.39 ms			
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	Prepare	4.12 ms			
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	Query	0.78 ms			
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	Prepare	4.05 ms			

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	Query	5.32 ms			
DELETE_HIGH_SELECTIVIY_NON_ISSUED	Prepare	2.50 ms			
DELETE_HIGH_SELECTIVIY_NON_ISSUED	Query	0.04 ms			

B.12.5 Evaluation Details for PostgreSQL, Version PostgreSQL 9.4.4 on x86_64-apple-darwin14.3.0, compiled by Apple LLVM version 6.1.0 (clang-602.0.53) (based on LLVM 3.6.0svn), 64-bit / 9.4-1201-jdbc41, Test Series TINY

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
Set up		257.17 ms			
Load		500.93 ms			
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	Prepare	15.99 ms			
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	Query	1.95 ms	0.59 ms	0.51 ms	1.02 ms
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	Prepare	30.35 ms			
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	Query	3.97 ms	2.05 ms	1.55 ms	2.53 ms
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	Prepare	28.53 ms			
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	Query	6.95 ms	4.09 ms	4.53 ms	5.19 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10	Prepare	13.05 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10	Query	1.84 ms	0.62 ms	0.51 ms	0.99 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100	Prepare	27.86 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100	Query	2.43 ms	0.98 ms	0.77 ms	1.39 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL	Prepare	23.87 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL	Query	1.63 ms	0.59 ms	0.57 ms	0.93 ms
AGGREGATE_ISSUES_PER_DECADE_TOP10	Prepare	27.34 ms			
AGGREGATE_ISSUES_PER_DECADE_TOP10	Query	2.07 ms	0.75 ms	0.69 ms	1.17 ms
AGGREGATE_ISSUES_PER_DECADE_TOP100	Prepare	28.94 ms			
AGGREGATE_ISSUES_PER_DECADE_TOP100	Query	2.10 ms	0.74 ms	0.80 ms	1.22 ms
AGGREGATE_ISSUES_PER_DECADE_ALL	Prepare	28.31 ms			
AGGREGATE_ISSUES_PER_DECADE_ALL	Query	2.04 ms	0.91 ms	1.06 ms	1.34 ms
CONDITIONAL_TABLE_SCAN_ALL_STUDIES	Prepare	12.11 ms			
CONDITIONAL_TABLE_SCAN_ALL_STUDIES	Query	3.81 ms	1.80 ms	1.75 ms	2.45 ms
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES	Prepare	11.78 ms			

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES	Query	14.45 ms	14.44 ms	14.51 ms	14.46 ms
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES	Prepare	28.68 ms			
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES	Query	3.82 ms	1.82 ms	1.71 ms	2.45 ms
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY	Prepare	42.63 ms			
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY	Query	0.94 ms	0.80 ms	0.65 ms	0.80 ms
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES	Prepare	11.49 ms			
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES	Query	1.05 ms	0.60 ms	0.50 ms	0.72 ms
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES	Prepare	11.94 ms			
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES	Query	1.17 ms	0.84 ms	0.87 ms	0.96 ms
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	Prepare	8.94 ms			
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	Query	32.83 ms			
SCHEMA_CHANGE_INTRODUCE_STRING_OP	Prepare	14.26 ms			
SCHEMA_CHANGE_INTRODUCE_STRING_OP	Query	27.75 ms			
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	Prepare	26.15 ms			
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	Query	135.54 ms			
SCHEMA_CHANGE_REMOVE_RDF_TYPE	Prepare	10.30 ms			
SCHEMA_CHANGE_REMOVE_RDF_TYPE	Query	16.27 ms			
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	Prepare	15.56 ms			
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	Query	40.54 ms			
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	Prepare	15.46 ms			
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	Query	2.04 ms			
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	Prepare	29.06 ms			
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	Query	2.73 ms			
DELETE_HIGH_SELECTIVIY_NON_ISSUED	Prepare	27.70 ms			
DELETE_HIGH_SELECTIVIY_NON_ISSUED	Query	1.28 ms			

B.12.6 Evaluation Details for Virtuoso, Version 07.20.3214 / Virtuoso JDBC 4.1, Test Series TINY

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
Set up		0.94 ms			
Load		121.22 ms			

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	Prepare	101.60 ms			
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	Query	181.26 ms	4.33 ms	3.82 ms	63.13 ms
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	Prepare	6.08 ms			
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	Query	165.51 ms	154.80 ms	181.59 ms	167.30 ms
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	Prepare	12.82 ms			
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	Query	378.78 ms	350.67 ms	345.26 ms	358.24 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10	Prepare	2.13 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10	Query	0.57 ms	0.42 ms	0.37 ms	0.46 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100	Prepare	1.39 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100	Query	0.59 ms	0.62 ms	0.57 ms	0.59 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL	Prepare	1.66 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL	Query	0.66 ms	0.61 ms	0.59 ms	0.62 ms
AGGREGATE_ISSUES_PER_DECADE_TOP10	Prepare	1.79 ms			
AGGREGATE_ISSUES_PER_DECADE_TOP10	Query	2.84 ms	3.07 ms	2.71 ms	2.87 ms
AGGREGATE_ISSUES_PER_DECADE_TOP100	Prepare	1.56 ms			
AGGREGATE_ISSUES_PER_DECADE_TOP100	Query	3.02 ms	2.86 ms	2.77 ms	2.88 ms
AGGREGATE_ISSUES_PER_DECADE_ALL	Prepare	1.74 ms			
AGGREGATE_ISSUES_PER_DECADE_ALL	Query	2.96 ms	2.77 ms	2.78 ms	2.84 ms
CONDITIONAL_TABLE_SCAN_ALL_STUDIES	Prepare	6.96 ms			
CONDITIONAL_TABLE_SCAN_ALL_STUDIES	Query	231.79 ms	238.46 ms	225.08 ms	231.78 ms
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES	Prepare	8.22 ms			
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES	Query	1961.22 ms	1948.46 ms	1977.24 ms	1962.31 ms
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES	Prepare	7.66 ms			
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES	Query	222.93 ms	206.68 ms	265.33 ms	231.65 ms
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY	Prepare	3.20 ms			
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY	Query	1.42 ms	0.38 ms	0.36 ms	0.72 ms
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES	Prepare	1.83 ms			
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES	Query	2.68 ms	0.34 ms	0.38 ms	1.13 ms
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES	Prepare	1.82 ms			
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES	Query	2.66 ms	0.36 ms	0.34 ms	1.12 ms
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	Prepare	1.55 ms			
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	Query	158.15 ms			
SCHEMA_CHANGE_INTRODUCE_STRING_OP	Prepare	2.69 ms			

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
SCHEMA_CHANGE_INTRODUCE_STRING_OP	Query	2.17 ms			
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	Prepare	5.73 ms			
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	Query	1010.09 ms			
SCHEMA_CHANGE_REMOVE_RDF_TYPE	Prepare	2.20 ms			
SCHEMA_CHANGE_REMOVE_RDF_TYPE	Query	0.21 ms			
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	Prepare	1.31 ms			
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	Query	20.29 ms			
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	Prepare	1.67 ms			
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	Query	2.08 ms			
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	Prepare	1.56 ms			
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	Query	512.00 ms			
DELETE_HIGH_SELECTIVIY_NON_ISSUED	Prepare	27.79 ms			
DELETE_HIGH_SELECTIVIY_NON_ISSUED	Query	2.14 ms			

B.12.7 Evaluation Details for Fuseki, Version 2.3.0 2015-07-25T17:11:28+0000 / jena-libs 2.13.0, Test Series TINY

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
Set up		0.00 ms			
Load		1061.05 ms			
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	Prepare	234.38 ms			
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	Query	65.55 ms	18.27 ms	18.20 ms	34.00 ms
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	Prepare	87.40 ms			
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	Query	45.87 ms	35.32 ms	34.36 ms	38.52 ms
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	Prepare	149.42 ms			
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	Query	156.12 ms	124.47 ms	120.72 ms	133.77 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10	Prepare	1.09 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10	Query	62.18 ms	12.84 ms	10.64 ms	28.55 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100	Prepare	0.64 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100	Query	11.91 ms	10.78 ms	15.00 ms	12.56 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL	Prepare	0.71 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL	Query	16.92 ms	10.11 ms	13.48 ms	13.50 ms

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
AGGREGATE_ISSUES_PER_DECADE_TOP10	Prepare	0.71 ms			
AGGREGATE_ISSUES_PER_DECADE_TOP10	Query	23.03 ms	17.83 ms	18.01 ms	19.63 ms
AGGREGATE_ISSUES_PER_DECADE_TOP100	Prepare	0.62 ms			
AGGREGATE_ISSUES_PER_DECADE_TOP100	Query	12.77 ms	21.76 ms	9.86 ms	14.80 ms
AGGREGATE_ISSUES_PER_DECADE_ALL	Prepare	0.54 ms			
AGGREGATE_ISSUES_PER_DECADE_ALL	Query	9.11 ms	11.06 ms	15.68 ms	11.95 ms
CONDITIONAL_TABLE_SCAN_ALL_STUDIES	Prepare	2.18 ms			
CONDITIONAL_TABLE_SCAN_ALL_STUDIES	Query	168.15 ms	131.36 ms	100.71 ms	133.41 ms
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES	Prepare	1.60 ms			
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES	Query	880.06 ms	656.41 ms	736.24 ms	757.57 ms
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES	Prepare	1.34 ms			
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES	Query	292.72 ms	295.55 ms	275.96 ms	288.08 ms
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY	Prepare	13.00 ms			
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY	Query	9.48 ms	7.44 ms	6.81 ms	7.91 ms
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES	Prepare	10.83 ms			
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES	Query	8.14 ms	8.80 ms	8.10 ms	8.35 ms
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES	Prepare	10.88 ms			
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES	Query	6.79 ms	7.08 ms	6.58 ms	6.82 ms
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	Prepare	3.16 ms			
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	Query	236.66 ms			
SCHEMA_CHANGE_INTRODUCE_STRING_OP	Prepare	1.22 ms			
SCHEMA_CHANGE_INTRODUCE_STRING_OP	Query	7.66 ms			
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	Prepare	3.11 ms			
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	Query	368.64 ms			
SCHEMA_CHANGE_REMOVE_RDF_TYPE	Prepare	1.04 ms			
SCHEMA_CHANGE_REMOVE_RDF_TYPE	Query	5.33 ms			
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	Prepare	0.99 ms			
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	Query	33.38 ms			
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	Prepare	23.21 ms			
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	Query	73.33 ms			
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	Prepare	1.24 ms			
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	Query	5.31 ms			
DELETE_HIGH_SELECTIVIY_NON_ISSUED	Prepare	0.72 ms			

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
DELETE_HIGH_SELECTIVIY_NON_ISSUED	Query	3.67 ms			

B.12.8 Evaluation Details for MongoDB, Version 3.0.6, Test Series TINY

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
Set up		5.88 ms			
Load		580.58 ms			
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	Prepare	15.66 ms			
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	Query	1.20 ms	0.65 ms	0.62 ms	0.82 ms
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	Prepare	6.68 ms			
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	Query	6.45 ms	6.00 ms	6.00 ms	6.15 ms
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	Prepare	8.13 ms			
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	Query	61.39 ms	52.48 ms	53.83 ms	55.90 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10	Prepare	0.00 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10	Query	11.66 ms	1.22 ms	1.37 ms	4.75 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100	Prepare	0.00 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100	Query	2.32 ms	1.93 ms	1.60 ms	1.95 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL	Prepare	0.00 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL	Query	2.49 ms	1.40 ms	1.70 ms	1.86 ms
AGGREGATE_ISSUES_PER_DECADE_TOP10	Prepare	0.00 ms			
AGGREGATE_ISSUES_PER_DECADE_TOP10	Query	2.19 ms	1.80 ms	1.58 ms	1.85 ms
AGGREGATE_ISSUES_PER_DECADE_TOP100	Prepare	0.00 ms			
AGGREGATE_ISSUES_PER_DECADE_TOP100	Query	2.11 ms	1.60 ms	1.81 ms	1.84 ms
AGGREGATE_ISSUES_PER_DECADE_ALL	Prepare	0.00 ms			
AGGREGATE_ISSUES_PER_DECADE_ALL	Query	2.15 ms	1.62 ms	1.74 ms	1.84 ms
CONDITIONAL_TABLE_SCAN_ALL_STUDIES	Prepare	0.00 ms			
CONDITIONAL_TABLE_SCAN_ALL_STUDIES	Query	3.36 ms	2.78 ms	2.78 ms	2.97 ms
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES	Prepare	0.00 ms			
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES	Query	10.48 ms	7.52 ms	7.62 ms	8.54 ms
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES	Prepare	0.00 ms			
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES	Query	3.51 ms	2.98 ms	2.97 ms	3.15 ms

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY	Prepare	0.80 ms			
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY	Query	4.03 ms	2.34 ms	1.97 ms	2.78 ms
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES	Prepare	1.32 ms			
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES	Query	22.19 ms	18.74 ms	18.02 ms	19.65 ms
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES	Prepare	1.25 ms			
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES	Query	17.41 ms	17.69 ms	17.73 ms	17.61 ms
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	Prepare	0.00 ms			
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	Query	10.27 ms			
SCHEMA_CHANGE_INTRODUCE_STRING_OP	Prepare	0.00 ms			
SCHEMA_CHANGE_INTRODUCE_STRING_OP	Query	187.42 ms			
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	Prepare	0.00 ms			
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	Query	17.36 ms			
SCHEMA_CHANGE_REMOVE_RDF_TYPE	Prepare	0.00 ms			
SCHEMA_CHANGE_REMOVE_RDF_TYPE	Query	3.38 ms			
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	Prepare	0.00 ms			
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	Query	3.30 ms			
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	Prepare	0.00 ms			
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	Query	0.58 ms			
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	Prepare	0.00 ms			
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	Query	5.45 ms			
DELETE_HIGH_SELECTIVIY_NON_ISSUED	Prepare	0.00 ms			
DELETE_HIGH_SELECTIVIY_NON_ISSUED	Query	0.26 ms			

B.12.9 Evaluation Details for ArangoDB, Version 2.6.9 64bit – ICU 54.1, V8 4.1.0.27, OpenSSL 1.0.2d 9 Jul 2015 / Java Driver 2.6.8, Test Series TINY

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
Set up		206.65 ms			
Load		726.13 ms			
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	Prepare	18.10 ms			
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	Query	3.36 ms	2.66 ms	2.39 ms	2.80 ms

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	Prepare	9.96 ms			
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	Query	22.44 ms	23.33 ms	20.61 ms	22.13 ms
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	Prepare	13.81 ms			
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	Query	179.61 ms	170.19 ms	170.65 ms	173.48 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10	Prepare	0.00 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10	Query	2.47 ms	2.40 ms	2.17 ms	2.35 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100	Prepare	0.00 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100	Query	2.69 ms	2.39 ms	2.33 ms	2.47 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL	Prepare	0.00 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL	Query	2.29 ms	2.25 ms	2.32 ms	2.29 ms
AGGREGATE_ISSUES_PER_DECADE_TOP10	Prepare	0.00 ms			
AGGREGATE_ISSUES_PER_DECADE_TOP10	Query	31.67 ms	17.44 ms	15.35 ms	21.49 ms
AGGREGATE_ISSUES_PER_DECADE_TOP100	Prepare	0.00 ms			
AGGREGATE_ISSUES_PER_DECADE_TOP100	Query	12.07 ms	13.90 ms	12.79 ms	12.92 ms
AGGREGATE_ISSUES_PER_DECADE_ALL	Prepare	0.00 ms			
AGGREGATE_ISSUES_PER_DECADE_ALL	Query	15.40 ms	13.65 ms	12.81 ms	13.95 ms
CONDITIONAL_TABLE_SCAN_ALL_STUDIES	Prepare	45.35 ms			
CONDITIONAL_TABLE_SCAN_ALL_STUDIES	Query	4.10 ms	3.19 ms	2.86 ms	3.39 ms
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES	Prepare	0.00 ms			
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES	Query	54.89 ms	43.66 ms	42.85 ms	47.13 ms
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES	Prepare	0.00 ms			
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES	Query	14.64 ms	13.03 ms	12.42 ms	13.36 ms
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY	Prepare	121.40 ms			
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY	Query	6.48 ms	4.56 ms	3.87 ms	4.97 ms
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES	Prepare	8.11 ms			
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES	Query	32.09 ms	31.94 ms	29.12 ms	31.05 ms
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES	Prepare	11.50 ms			
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES	Query	218.05 ms	213.27 ms	206.68 ms	212.67 ms
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	Prepare	0.00 ms			
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	Query	53.24 ms			
SCHEMA_CHANGE_INTRODUCE_STRING_OP	Prepare	0.00 ms			
SCHEMA_CHANGE_INTRODUCE_STRING_OP	Query	66.43 ms			
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	Prepare	0.00 ms			

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	Query	234.75 ms			
SCHEMA_CHANGE_REMOVE_RDF_TYPE	Prepare	0.00 ms			
SCHEMA_CHANGE_REMOVE_RDF_TYPE	Query	58.49 ms			
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	Prepare	0.00 ms			
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	Query	57.54 ms			
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	Prepare	0.00 ms			
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	Query	9.66 ms			
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	Prepare	0.00 ms			
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	Query	4.21 ms			
DELETE_HIGH_SELECTIVIY_NON_ISSUED	Prepare	0.00 ms			
DELETE_HIGH_SELECTIVIY_NON_ISSUED	Query	0.69 ms			

B.13 Evaluation Report for Test Series Small

B.13.1 Evaluation Overview, all Databases, Test Series SMALL

QueryScenario	P.	sqlite4java	SQLite-Xerial	PostgreSQL	Virtuoso	Fuseki	MongoDB	ArangoDB
Set up		18.47 ms	71.12 ms	257.27 ms	0.93 ms	**0.00 ms**	3.68 ms	179.59 ms
Load		21536.39 ms	21137.56 ms	28130.98 ms	**7939.37 ms**	30072.75 ms	31006.20 ms	37685.71 ms
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	P.	302.13 ms	377.61 ms	**208.78 ms**	887.70 ms	2966.76 ms	666.65 ms	1697.89 ms
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	Q.	1.38 ms	**0.31 ms**	1.14 ms	566.57 ms	41.63 ms	61.52 ms	122.41 ms
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	P.	**147.21 ms**	425.49 ms	197.98 ms	150.30 ms	2338.34 ms	587.62 ms	1659.06 ms
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	Q.	0.94 ms	**0.66 ms**	2.84 ms	6164.64 ms	48.79 ms	451.85 ms	1283.15 ms
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	P.	**155.56 ms**	651.40 ms	198.67 ms	170.64 ms	2747.84 ms	614.22 ms	1684.18 ms
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	Q.	2.38 ms	**2.06 ms**	4.37 ms	54452.35 ms	153.49 ms	4643.60 ms	12976.55 ms
AGG_PUBLICATIONS_PER_PUBLISHER_TOP10	P.	0.63 ms	193.46 ms	80.62 ms	1.82 ms	0.84 ms	**0.00 ms**	**0.00 ms**
AGG._PUBLICATIONS_PER_PUBLISHER_TOP10	Q.	134.64 ms	32.45 ms	41.04 ms	**21.84 ms**	174.15 ms	83.63 ms	158.19 ms
AGG._PUBLICATIONS_PER_PUBLISHER_TOP100	P.	0.57 ms	218.41 ms	96.21 ms	1.24 ms	6.81 ms	**0.00 ms**	**0.00 ms**
AGG._PUBLICATIONS_PER_PUBLISHER_TOP100	Q.	143.94 ms	32.09 ms	41.83 ms	**27.45 ms**	81.54 ms	83.09 ms	145.58 ms
AGG._PUBLICATIONS_PER_PUBLISHER_ALL	P.	0.72 ms	208.74 ms	99.40 ms	1.53 ms	0.70 ms	**0.00 ms**	**0.00 ms**
AGG._PUBLICATIONS_PER_PUBLISHER_ALL	Q.	154.91 ms	**40.26 ms**	50.25 ms	116.82 ms	458.24 ms	110.77 ms	301.44 ms
AGG._ISSUES_PER_DECADE_TOP10	P.	3.74 ms	389.29 ms	218.90 ms	1.39 ms	0.50 ms	**0.00 ms**	**0.00 ms**
AGG_ISSUES_PER_DECADE_TOP10	Q.	174.19 ms	225.04 ms	**46.10 ms**	136.46 ms	166.90 ms	127.21 ms	399.28 ms
AGG._ISSUES_PER_DECADE_TOP100	P.	0.63 ms	406.49 ms	219.27 ms	2.12 ms	0.48 ms	**0.00 ms**	**0.00 ms**
AGG._ISSUES_PER_DECADE_TOP100	Q.	164.77 ms	228.12 ms	**44.40 ms**	133.34 ms	144.22 ms	114.53 ms	383.72 ms
AGG._ISSUES_PER_DECADE_ALL	P.	0.52 ms	422.42 ms	213.19 ms	1.39 ms	0.47 ms	**0.00 ms**	**0.00 ms**
AGG._ISSUES_PER_DECADE_ALL	Q.	161.50 ms	226.51 ms	**46.88 ms**	143.11 ms	112.12 ms	114.41 ms	375.78 ms
COND._TABLE_SCAN_ALL_STUDIES	P.	0.57 ms	362.61 ms	159.15 ms	7.14 ms	0.48 ms	**0.00 ms**	1437.42 ms
COND._TABLE_SCAN_ALL_STUDIES	Q.	207.82 ms	299.94 ms	**130.40 ms**	16675.65 ms	7521.54 ms	217.29 ms	152.86 ms
COND._TABLE_SCAN_ALL_BIB._RES.	P.	0.56 ms	2.69 ms	111.25 ms	7.12 ms	0.70 ms	**0.00 ms**	**0.00 ms**
COND._TABLE_SCAN_ALL_BIB._RES.	Q.	980.37 ms	1189.01 ms	1031.39 ms	2.99E5 ms	65087.94 ms	**515.95 ms**	3203.18 ms
COND._T._S._ALL_BIB._RES._AND_STUDIES	P.	0.58 ms	2.10 ms	339.09 ms	6.93 ms	2.85 ms	**0.00 ms**	**0.00 ms**
COND._T._S._ALL_BIB._RES._AND_STUDIES	Q.	250.47 ms	339.87 ms	**128.86 ms**	13504.67 ms	23235.87 ms	267.11 ms	747.21 ms
GRAPH_L._REL._BY_SUBJ._1HOP_ONE_ENTITY	P.	19101.57 ms	19982.08 ms	178.16 ms	**35.90 ms**	661.71 ms	79.33 ms	18441.61 ms
GRAPH_L._REL._BY_SUBJ._1HOP_ONE_ENTITY	Q.	0.31 ms	**0.19 ms**	0.82 ms	0.78 ms	136.78 ms	269.81 ms	73.29 ms
GRAPH_L._REL._BY_SUBJ._1HOP_10_ENTITIES	P.	81.12 ms	103.30 ms	75.85 ms	**38.47 ms**	496.71 ms	78.34 ms	827.13 ms

QueryScenario	P.	sqlite4java	SQLite-Xerial	PostgreSQL	Virtuoso	Fuseki	MongoDB	ArangoDB
GRAPH_L._REL._BY_SUBJ._1HOP_10_ENTITIES	Q.	0.62 ms	**0.54 ms**	7.73 ms	1.43 ms	7.61 ms	3042.83 ms	679.41 ms
GRAPH_L._REL._BY_SUBJ._1HOP_100_ENTITIES	P.	108.27 ms	103.52 ms	79.40 ms	**63.02 ms**	614.32 ms	80.58 ms	1601.55 ms
GRAPH_L._REL._BY_SUBJ._1HOP_100_ENTITIES	Q.	15.95 ms	13.22 ms	**8.55 ms**	13.61 ms	122.48 ms	41676.83 ms	6267.49 ms
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	P.	3.33 ms	2.18 ms	9.37 ms	2.14 ms	4.05 ms	**0.00 ms**	**0.00 ms**
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	Q.	1.67 ms	**0.82 ms**	680.37 ms	16271.80 ms	12291.21 ms	730.76 ms	7259.30 ms
SCHEMA_CHANGE_INTRODUCE_STRING_OP	P.	0.83 ms	212.58 ms	138.84 ms	191.79 ms	1.46 ms	**0.00 ms**	**0.00 ms**
SCHEMA_CHANGE_INTRODUCE_STRING_OP	Q.	776.19 ms	899.90 ms	4984.46 ms	**489.95 ms**	524.28 ms	10512.26 ms	6907.28 ms
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	P.	0.81 ms	2.31 ms	191.10 ms	5.32 ms	3.92 ms	**0.00 ms**	**0.00 ms**
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	Q.	**1619.26 ms**	2103.30 ms	14700.13 ms	91227.25 ms	29899.91 ms	2156.42 ms	28700.01 ms
SCHEMA_CHANGE_REMOVE_RDF_TYPE	P.	1.04 ms	2.14 ms	10.33 ms	1.15 ms	0.99 ms	**0.00 ms**	**0.00 ms**
SCHEMA_CHANGE_REMOVE_RDF_TYPE	Q.	442.17 ms	576.15 ms	20.84 ms	**0.19 ms**	8.06 ms	404.88 ms	5876.80 ms
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	P.	0.44 ms	160.42 ms	165.19 ms	1.01 ms	2.31 ms	0.01 ms	**0.00 ms**
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	Q.	529.73 ms	908.66 ms	2348.41 ms	3837.52 ms	1964.35 ms	**346.10 ms**	6928.84 ms
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	P.	0.53 ms	214.09 ms	220.00 ms	1.45 ms	17.98 ms	**0.00 ms**	**0.00 ms**
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	Q.	82.10 ms	**28.92 ms**	105.60 ms	328.60 ms	4643.03 ms	60.63 ms	1073.74 ms
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	P.	0.46 ms	215.56 ms	181.27 ms	Error	1.35 ms	**0.00 ms**	**0.00 ms**
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	Q.	305.24 ms	617.56 ms	423.15 ms	Error	**6.33 ms**	442.88 ms	1213.52 ms
DELETE_HIGH_SELECTIVIY_NON_ISSUED	P.	0.41 ms	18.81 ms	121.94 ms	883.94 ms	0.92 ms	**0.00 ms**	**0.00 ms**
DELETE_HIGH_SELECTIVIY_NON_ISSUED	Q.	8.94 ms	4.64 ms	**2.14 ms**	11342.42 ms	3.56 ms	5.95 ms	18.37 ms

B.13.2 Test Series Datasets

Test Series SMALL	Dataset Size
hebis_100000_records	7.865 MB

B.13.3 Evaluation Details for sqlite4java, Version 392 with SQLite 3.8.7, Test Series SMALL

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
Set up		18.47 ms			
Load		21536.39 ms			
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	Prepare	302.13 ms			
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	Query	3.86 ms	0.16 ms	0.12 ms	1.38 ms
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	Prepare	147.21 ms			
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	Query	0.89 ms	0.59 ms	1.35 ms	0.94 ms
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	Prepare	155.56 ms			
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	Query	3.09 ms	2.09 ms	1.96 ms	2.38 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10	Prepare	0.63 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10	Query	136.63 ms	127.70 ms	139.60 ms	134.64 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100	Prepare	0.57 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100	Query	143.88 ms	145.09 ms	142.86 ms	143.94 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL	Prepare	0.72 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL	Query	166.11 ms	151.97 ms	146.63 ms	154.91 ms
AGGREGATE_ISSUES_PER_DECADE_TOP10	Prepare	3.74 ms			
AGGREGATE_ISSUES_PER_DECADE_TOP10	Query	160.60 ms	175.80 ms	186.17 ms	174.19 ms
AGGREGATE_ISSUES_PER_DECADE_TOP100	Prepare	0.63 ms			
AGGREGATE_ISSUES_PER_DECADE_TOP100	Query	163.42 ms	166.55 ms	164.34 ms	164.77 ms
AGGREGATE_ISSUES_PER_DECADE_ALL	Prepare	0.52 ms			
AGGREGATE_ISSUES_PER_DECADE_ALL	Query	161.16 ms	161.75 ms	161.60 ms	161.50 ms
COND._T._S._ALL_STUDIES	Prepare	0.57 ms			
COND._T._S._ALL_STUDIES	Query	209.07 ms	209.88 ms	204.51 ms	207.82 ms
COND._T._S._ALL_BIBLIOGRAPHIC_RESOURCES	Prepare	0.56 ms			
COND._T._S._ALL_BIBLIOGRAPHIC_RESOURCES	Query	1000.50 ms	978.26 ms	962.34 ms	980.37 ms
COND._T._S._ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES	Prepare	0.58 ms			
COND._T._S._ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES	Query	244.27 ms	267.94 ms	239.20 ms	250.47 ms
GRAPH_L._REL._BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY	Prepare	19101.57 ms			
GRAPH_L._REL._BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY	Query	0.60 ms	0.18 ms	0.15 ms	0.31 ms
GRAPH_L._REL._BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES	Prepare	81.12 ms			
GRAPH_L._REL._BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES	Query	0.67 ms	0.65 ms	0.54 ms	0.62 ms
GRAPH_L._REL._BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES	Prepare	108.27 ms			
GRAPH_L._REL._BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES	Query	18.06 ms	15.61 ms	14.17 ms	15.95 ms
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	Prepare	3.33 ms			

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	Query	1.67 ms			
SCHEMA_CHANGE_INTRODUCE_STRING_OP	Prepare	0.83 ms			
SCHEMA_CHANGE_INTRODUCE_STRING_OP	Query	776.19 ms			
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	Prepare	0.81 ms			
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	Query	1619.26 ms			
SCHEMA_CHANGE_REMOVE_RDF_TYPE	Prepare	1.04 ms			
SCHEMA_CHANGE_REMOVE_RDF_TYPE	Query	442.17 ms			
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	Prepare	0.44 ms			
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	Query	529.73 ms			
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	Prepare	0.53 ms			
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	Query	82.10 ms			
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	Prepare	0.46 ms			
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	Query	305.24 ms			
DELETE_HIGH_SELECTIVIY_NON_ISSUED	Prepare	0.41 ms			
DELETE_HIGH_SELECTIVIY_NON_ISSUED	Query	8.94 ms			

B.13.4 Evaluation Details for SQLite-Xerial, Version 3.8.11, Test Series SMALL

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
Set up		71.12 ms			
Load		21137.56 ms			
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	Prepare	377.61 ms			
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	Query	0.49 ms	0.24 ms	0.20 ms	0.31 ms
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	Prepare	425.49 ms			
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	Query	0.74 ms	0.70 ms	0.54 ms	0.66 ms
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	Prepare	651.40 ms			
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	Query	2.40 ms	1.87 ms	1.91 ms	2.06 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10	Prepare	193.46 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10	Query	34.53 ms	32.86 ms	29.95 ms	32.45 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100	Prepare	218.41 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100	Query	34.42 ms	31.26 ms	30.60 ms	32.09 ms

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL	Prepare	208.74 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL	Query	44.44 ms	38.17 ms	38.17 ms	40.26 ms
AGGREGATE_ISSUES_PER_DECADE_TOP10	Prepare	389.29 ms			
AGGREGATE_ISSUES_PER_DECADE_TOP10	Query	227.74 ms	231.59 ms	215.78 ms	225.04 ms
AGGREGATE_ISSUES_PER_DECADE_TOP100	Prepare	406.49 ms			
AGGREGATE_ISSUES_PER_DECADE_TOP100	Query	219.87 ms	236.89 ms	227.60 ms	228.12 ms
AGGREGATE_ISSUES_PER_DECADE_ALL	Prepare	422.42 ms			
AGGREGATE_ISSUES_PER_DECADE_ALL	Query	239.62 ms	216.98 ms	222.93 ms	226.51 ms
COND._T._S._ALL_STUDIES	Prepare	362.61 ms			
COND._T._S._ALL_STUDIES	Query	305.87 ms	282.15 ms	311.80 ms	299.94 ms
COND._T._S._ALL_BIBLIOGRAPHIC_RESOURCES	Prepare	2.69 ms			
COND._T._S._ALL_BIBLIOGRAPHIC_RESOURCES	Query	1191.71 ms	1162.55 ms	1212.76 ms	1189.01 ms
COND._T._S._ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES	Prepare	2.10 ms			
COND._T._S._ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES	Query	342.64 ms	348.16 ms	328.82 ms	339.87 ms
GRAPH_L._REL._BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY	Prepare	19982.08 ms			
GRAPH_L._REL._BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY	Query	0.27 ms	0.16 ms	0.15 ms	0.19 ms
GRAPH_L._REL._BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES	Prepare	103.30 ms			
GRAPH_L._REL._BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES	Query	0.62 ms	0.48 ms	0.53 ms	0.54 ms
GRAPH_L._REL._BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES	Prepare	103.52 ms			
GRAPH_L._REL._BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES	Query	14.09 ms	13.30 ms	12.29 ms	13.22 ms
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	Prepare	2.18 ms			
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	Query	0.82 ms			
SCHEMA_CHANGE_INTRODUCE_STRING_OP	Prepare	212.58 ms			
SCHEMA_CHANGE_INTRODUCE_STRING_OP	Query	899.90 ms			
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	Prepare	2.31 ms			
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	Query	2103.30 ms			
SCHEMA_CHANGE_REMOVE_RDF_TYPE	Prepare	2.14 ms			
SCHEMA_CHANGE_REMOVE_RDF_TYPE	Query	576.15 ms			
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	Prepare	160.42 ms			
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	Query	908.66 ms			
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	Prepare	214.09 ms			
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	Query	28.92 ms			
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	Prepare	215.56 ms			

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	Query	617.56 ms			
DELETE_HIGH_SELECTIVIY_NON_ISSUED	Prepare	18.81 ms			
DELETE_HIGH_SELECTIVIY_NON_ISSUED	Query	4.64 ms			

B.13.5 Evaluation Details for PostgreSQL, Version PostgreSQL 9.4.4 on x86_64-apple-darwin14.3.0, compiled by Apple LLVM version 6.1.0 (clang-602.0.53) (based on LLVM 3.6.0svn), 64-bit / 9.4-1201-jdbc41, Test Series SMALL

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
Set up		257.27 ms			
Load		28130.98 ms			
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	Prepare	208.78 ms			
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	Query	2.32 ms	0.58 ms	0.52 ms	1.14 ms
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	Prepare	197.98 ms			
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	Query	3.85 ms	2.32 ms	2.34 ms	2.84 ms
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	Prepare	198.67 ms			
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	Query	5.98 ms	3.44 ms	3.69 ms	4.37 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10	Prepare	80.62 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10	Query	51.24 ms	35.30 ms	36.57 ms	41.04 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100	Prepare	96.21 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100	Query	58.11 ms	30.32 ms	37.05 ms	41.83 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL	Prepare	99.40 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL	Query	67.38 ms	40.30 ms	43.06 ms	50.25 ms
AGGREGATE_ISSUES_PER_DECADE_TOP10	Prepare	218.90 ms			
AGGREGATE_ISSUES_PER_DECADE_TOP10	Query	54.93 ms	40.83 ms	42.54 ms	46.10 ms
AGGREGATE_ISSUES_PER_DECADE_TOP100	Prepare	219.27 ms			
AGGREGATE_ISSUES_PER_DECADE_TOP100	Query	54.15 ms	36.61 ms	42.44 ms	44.40 ms
AGGREGATE_ISSUES_PER_DECADE_ALL	Prepare	213.19 ms			
AGGREGATE_ISSUES_PER_DECADE_ALL	Query	54.68 ms	43.04 ms	42.91 ms	46.88 ms
CONDITIONAL_TABLE_SCAN_ALL_STUDIES	Prepare	159.15 ms			
CONDITIONAL_TABLE_SCAN_ALL_STUDIES	Query	142.50 ms	122.09 ms	126.61 ms	130.40 ms
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES	Prepare	111.25 ms			

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES	Query	958.00 ms	1050.21 ms	1085.96 ms	1031.39 ms
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES	Prepare	339.09 ms			
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES	Query	143.39 ms	119.93 ms	123.25 ms	128.86 ms
GRAPH_L._REL._BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY	Prepare	178.16 ms			
GRAPH_L._REL._BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY	Query	1.20 ms	0.67 ms	0.60 ms	0.82 ms
GRAPH_L._REL._BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES	Prepare	75.85 ms			
GRAPH_L._REL._BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES	Query	10.42 ms	7.20 ms	5.57 ms	7.73 ms
GRAPH_L._REL._BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES	Prepare	79.40 ms			
GRAPH_L._REL._BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES	Query	9.66 ms	7.76 ms	8.23 ms	8.55 ms
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	Prepare	9.37 ms			
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	Query	680.37 ms			
SCHEMA_CHANGE_INTRODUCE_STRING_OP	Prepare	138.84 ms			
SCHEMA_CHANGE_INTRODUCE_STRING_OP	Query	4984.46 ms			
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	Prepare	191.10 ms			
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	Query	14700.13 ms			
SCHEMA_CHANGE_REMOVE_RDF_TYPE	Prepare	10.33 ms			
SCHEMA_CHANGE_REMOVE_RDF_TYPE	Query	20.84 ms			
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	Prepare	165.19 ms			
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	Query	2348.41 ms			
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	Prepare	220.00 ms			
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	Query	105.60 ms			
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	Prepare	181.27 ms			
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	Query	423.15 ms			
DELETE_HIGH_SELECTIVIY_NON_ISSUED	Prepare	121.94 ms			
DELETE_HIGH_SELECTIVIY_NON_ISSUED	Query	2.14 ms			

B.13.6 Evaluation Details for Virtuoso, Version 07.20.3214 / Virtuoso JDBC 4.1, Test Series SMALL

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
Set up		0.93 ms			
Load		7939.37 ms			

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	Prepare	887.70 ms			
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	Query	1171.36 ms	253.50 ms	274.84 ms	566.57 ms
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	Prepare	150.30 ms			
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	Query	6024.43 ms	6295.79 ms	6173.71 ms	6164.64 ms
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	Prepare	170.64 ms			
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	Query	53637.47 ms	54787.61 ms	54931.98 ms	54452.35 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10	Prepare	1.82 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10	Query	22.24 ms	20.61 ms	22.68 ms	21.84 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100	Prepare	1.24 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100	Query	23.65 ms	29.19 ms	29.52 ms	27.45 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL	Prepare	1.53 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL	Query	131.73 ms	105.53 ms	113.18 ms	116.82 ms
AGGREGATE_ISSUES_PER_DECADE_TOP10	Prepare	1.39 ms			
AGGREGATE_ISSUES_PER_DECADE_TOP10	Query	138.20 ms	139.31 ms	131.87 ms	136.46 ms
AGGREGATE_ISSUES_PER_DECADE_TOP100	Prepare	2.12 ms			
AGGREGATE_ISSUES_PER_DECADE_TOP100	Query	128.13 ms	136.77 ms	135.11 ms	133.34 ms
AGGREGATE_ISSUES_PER_DECADE_ALL	Prepare	1.39 ms			
AGGREGATE_ISSUES_PER_DECADE_ALL	Query	146.84 ms	147.02 ms	135.47 ms	143.11 ms
CONDITIONAL_TABLE_SCAN_ALL_STUDIES	Prepare	7.14 ms			
CONDITIONAL_TABLE_SCAN_ALL_STUDIES	Query	17287.27 ms	16243.25 ms	16496.42 ms	16675.65 ms
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES	Prepare	7.12 ms			
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES	Query	3.00E5 ms	3.00E5 ms	2.98E5 ms	2.99E5 ms
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES	Prepare	6.93 ms			
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES	Query	13460.39 ms	13480.38 ms	13573.24 ms	13504.67 ms
GRAPH_L._REL._BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY	Prepare	35.90 ms			
GRAPH_L._REL._BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY	Query	1.50 ms	0.42 ms	0.41 ms	0.78 ms
GRAPH_L._REL._BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES	Prepare	38.47 ms			
GRAPH_L._REL._BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES	Query	3.00 ms	0.60 ms	0.70 ms	1.43 ms
GRAPH_L._REL._BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES	Prepare	63.02 ms			
GRAPH_L._REL._BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES	Query	26.98 ms	7.01 ms	6.85 ms	13.61 ms
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	Prepare	2.14 ms			
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	Query	16271.80 ms			
SCHEMA_CHANGE_INTRODUCE_STRING_OP	Prepare	191.79 ms			

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
SCHEMA_CHANGE_INTRODUCE_STRING_OP	Query	489.95 ms			
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	Prepare	5.32 ms			
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	Query	91227.25 ms			
SCHEMA_CHANGE_REMOVE_RDF_TYPE	Prepare	1.15 ms			
SCHEMA_CHANGE_REMOVE_RDF_TYPE	Query	0.19 ms			
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	Prepare	1.01 ms			
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	Query	3837.52 ms			
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	Prepare	1.45 ms			
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	Query	328.60 ms			
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	Prepare	Error			
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	Query	Error			
DELETE_HIGH_SELECTIVIY_NON_ISSUED	Prepare	883.94 ms			
DELETE_HIGH_SELECTIVIY_NON_ISSUED	Query	11342.42 ms			

B.13.7 Evaluation Details for Fuseki, Version 2.3.0 2015-07-25T17:11:28+0000 / jena-libs 2.13.0, Test Series SMALL

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
Set up		0.00 ms			
Load		30072.75 ms			
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	Prepare	2966.76 ms			
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	Query	93.75 ms	15.56 ms	15.58 ms	41.63 ms
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	Prepare	2338.34 ms			
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	Query	65.68 ms	36.84 ms	43.86 ms	48.79 ms
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	Prepare	2747.84 ms			
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	Query	174.35 ms	146.93 ms	139.20 ms	153.49 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10	Prepare	0.84 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10	Query	285.79 ms	134.20 ms	102.46 ms	174.15 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100	Prepare	6.81 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100	Query	86.31 ms	79.80 ms	78.49 ms	81.54 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL	Prepare	0.70 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL	Query	500.25 ms	410.13 ms	464.33 ms	458.24 ms

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
AGGREGATE_ISSUES_PER_DECADE_TOP10	Prepare	0.50 ms			
AGGREGATE_ISSUES_PER_DECADE_TOP10	Query	190.50 ms	158.43 ms	151.75 ms	166.90 ms
AGGREGATE_ISSUES_PER_DECADE_TOP100	Prepare	0.48 ms			
AGGREGATE_ISSUES_PER_DECADE_TOP100	Query	156.28 ms	146.36 ms	130.02 ms	144.22 ms
AGGREGATE_ISSUES_PER_DECADE_ALL	Prepare	0.47 ms			
AGGREGATE_ISSUES_PER_DECADE_ALL	Query	116.42 ms	105.16 ms	114.78 ms	112.12 ms
CONDITIONAL_TABLE_SCAN_ALL_STUDIES	Prepare	0.48 ms			
CONDITIONAL_TABLE_SCAN_ALL_STUDIES	Query	7400.73 ms	7518.27 ms	7645.61 ms	7521.54 ms
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES	Prepare	0.70 ms			
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES	Query	66308.89 ms	67642.58 ms	61312.35 ms	65087.94 ms
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES	Prepare	2.85 ms			
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES	Query	22817.02 ms	27546.35 ms	19344.25 ms	23235.87 ms
GRAPH_L._REL._BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY	Prepare	661.71 ms			
GRAPH_L._REL._BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY	Query	142.34 ms	132.45 ms	135.53 ms	136.78 ms
GRAPH_L._REL._BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES	Prepare	496.71 ms			
GRAPH_L._REL._BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES	Query	8.86 ms	7.22 ms	6.75 ms	7.61 ms
GRAPH_L._REL._BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES	Prepare	614.32 ms			
GRAPH_L._REL._BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES	Query	126.00 ms	121.21 ms	120.22 ms	122.48 ms
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	Prepare	4.05 ms			
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	Query	12291.21 ms			
SCHEMA_CHANGE_INTRODUCE_STRING_OP	Prepare	1.46 ms			
SCHEMA_CHANGE_INTRODUCE_STRING_OP	Query	524.28 ms			
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	Prepare	3.92 ms			
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	Query	29899.91 ms			
SCHEMA_CHANGE_REMOVE_RDF_TYPE	Prepare	0.99 ms			
SCHEMA_CHANGE_REMOVE_RDF_TYPE	Query	8.06 ms			
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	Prepare	2.31 ms			
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	Query	1964.35 ms			
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	Prepare	17.98 ms			
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	Query	4643.03 ms			
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	Prepare	1.35 ms			
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	Query	6.33 ms			
DELETE_HIGH_SELECTIVIY_NON_ISSUED	Prepare	0.92 ms			

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
DELETE_HIGH_SELECTIVIY_NON_ISSUED	Query	3.56 ms			

B.13.8 Evaluation Details for MongoDB, Version 3.0.6, Test Series SMALL

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
Set up		3.68 ms			
Load		31006.20 ms			
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	Prepare	666.65 ms			
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	Query	85.59 ms	53.75 ms	45.23 ms	61.52 ms
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	Prepare	587.62 ms			
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	Query	464.24 ms	443.16 ms	448.14 ms	451.85 ms
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	Prepare	614.22 ms			
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	Query	4670.32 ms	4709.89 ms	4550.59 ms	4643.60 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10	Prepare	0.00 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10	Query	91.06 ms	79.20 ms	80.64 ms	83.63 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100	Prepare	0.00 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100	Query	85.32 ms	83.39 ms	80.54 ms	83.09 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL	Prepare	0.00 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL	Query	116.34 ms	110.74 ms	105.21 ms	110.77 ms
AGGREGATE_ISSUES_PER_DECADE_TOP10	Prepare	0.00 ms			
AGGREGATE_ISSUES_PER_DECADE_TOP10	Query	139.54 ms	122.39 ms	119.72 ms	127.21 ms
AGGREGATE_ISSUES_PER_DECADE_TOP100	Prepare	0.00 ms			
AGGREGATE_ISSUES_PER_DECADE_TOP100	Query	118.73 ms	110.58 ms	114.27 ms	114.53 ms
AGGREGATE_ISSUES_PER_DECADE_ALL	Prepare	0.00 ms			
AGGREGATE_ISSUES_PER_DECADE_ALL	Query	113.39 ms	120.23 ms	109.62 ms	114.41 ms
CONDITIONAL_TABLE_SCAN_ALL_STUDIES	Prepare	0.00 ms			
CONDITIONAL_TABLE_SCAN_ALL_STUDIES	Query	212.12 ms	230.27 ms	209.48 ms	217.29 ms
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES	Prepare	0.00 ms			
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES	Query	508.71 ms	508.30 ms	530.83 ms	515.95 ms
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES	Prepare	0.00 ms			
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES	Query	278.40 ms	265.55 ms	257.37 ms	267.11 ms

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
GRAPH_L._REL._BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY	Prepare	79.33 ms			
GRAPH_L._REL._BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY	Query	276.54 ms	263.89 ms	269.00 ms	269.81 ms
GRAPH_L._REL._BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES	Prepare	78.34 ms			
GRAPH_L._REL._BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES	Query	2985.14 ms	2947.35 ms	3195.99 ms	3042.83 ms
GRAPH_L._REL._BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES	Prepare	80.58 ms			
GRAPH_L._REL._BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES	Query	40148.83 ms	40680.99 ms	44200.69 ms	41676.83 ms
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	Prepare	0.00 ms			
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	Query	730.76 ms			
SCHEMA_CHANGE_INTRODUCE_STRING_OP	Prepare	0.00 ms			
SCHEMA_CHANGE_INTRODUCE_STRING_OP	Query	10512.26 ms			
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	Prepare	0.00 ms			
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	Query	2156.42 ms			
SCHEMA_CHANGE_REMOVE_RDF_TYPE	Prepare	0.00 ms			
SCHEMA_CHANGE_REMOVE_RDF_TYPE	Query	404.88 ms			
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	Prepare	0.01 ms			
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	Query	346.10 ms			
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	Prepare	0.00 ms			
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	Query	60.63 ms			
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	Prepare	0.00 ms			
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	Query	442.88 ms			
DELETE_HIGH_SELECTIVIY_NON_ISSUED	Prepare	0.00 ms			
DELETE_HIGH_SELECTIVIY_NON_ISSUED	Query	5.95 ms			

B.13.9 Evaluation Details for ArangoDB, Version 2.6.9 64bit – ICU 54.1, V8 4.1.0.27, OpenSSL 1.0.2d 9 Jul 2015 / Java Driver 2.6.8, Test Series SMALL

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
Set up		179.59 ms			
Load		37685.51 ms			
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	Prepare	1697.89 ms			
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	Query	125.45 ms	124.48 ms	132.30 ms	127.41 ms

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	Prepare	1659.46 ms			
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	Query	1307.65 ms	1276.96 ms	1264.83 ms	1283.15 ms
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	Prepare	1684.48 ms			
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	Query	13171.69 ms	13084.39 ms	12673.58 ms	12976.55 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10	Prepare	0.00 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10	Query	150.34 ms	154.38 ms	169.85 ms	158.19 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100	Prepare	0.00 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100	Query	152.41 ms	145.49 ms	138.85 ms	145.58 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL	Prepare	0.00 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL	Query	329.57 ms	286.56 ms	288.18 ms	301.44 ms
AGGREGATE_ISSUES_PER_DECADE_TOP10	Prepare	0.00 ms			
AGGREGATE_ISSUES_PER_DECADE_TOP10	Query	370.46 ms	394.55 ms	434.94 ms	399.98 ms
AGGREGATE_ISSUES_PER_DECADE_TOP100	Prepare	0.00 ms			
AGGREGATE_ISSUES_PER_DECADE_TOP100	Query	399.75 ms	377.16 ms	374.25 ms	383.72 ms
AGGREGATE_ISSUES_PER_DECADE_ALL	Prepare	0.00 ms			
AGGREGATE_ISSUES_PER_DECADE_ALL	Query	390.39 ms	373.43 ms	363.52 ms	375.78 ms
CONDITIONAL_TABLE_SCAN_ALL_STUDIES	Prepare	1437.42 ms			
CONDITIONAL_TABLE_SCAN_ALL_STUDIES	Query	155.27 ms	153.46 ms	149.84 ms	152.86 ms
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES	Prepare	0.00 ms			
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES	Query	3314.04 ms	3171.26 ms	3124.24 ms	3203.18 ms
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES	Prepare	0.00 ms			
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES	Query	798.11 ms	716.55 ms	726.96 ms	747.21 ms
GRAPH_L._REL._BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY	Prepare	18441.61 ms			
GRAPH_L._REL._BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY	Query	75.34 ms	71.55 ms	72.98 ms	73.29 ms
GRAPH_L._REL._BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES	Prepare	827.13 ms			
GRAPH_L._REL._BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES	Query	681.31 ms	690.01 ms	666.91 ms	679.41 ms
GRAPH_L._REL._BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES	Prepare	1601.05 ms			
GRAPH_L._REL._BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES	Query	6253.46 ms	6248.20 ms	6300.79 ms	6267.49 ms
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	Prepare	0.00 ms			
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	Query	7259.30 ms			
SCHEMA_CHANGE_INTRODUCE_STRING_OP	Prepare	0.00 ms			
SCHEMA_CHANGE_INTRODUCE_STRING_OP	Query	6907.28 ms			
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	Prepare	0.00 ms			

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	Query	28700.81 ms			
SCHEMA_CHANGE_REMOVE_RDF_TYPE	Prepare	0.00 ms			
SCHEMA_CHANGE_REMOVE_RDF_TYPE	Query	5876.80 ms			
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	Prepare	0.00 ms			
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	Query	6928.84 ms			
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	Prepare	0.00 ms			
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	Query	1073.74 ms			
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	Prepare	0.00 ms			
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	Query	1213.62 ms			
DELETE_HIGH_SELECTIVIY_NON_ISSUED	Prepare	0.00 ms			
DELETE_HIGH_SELECTIVIY_NON_ISSUED	Query	18.37 ms			

B.14 Evaluation Report for Test Series Medium

B.14.1 Evaluation Overview, all Databases, Test Series MEDIUM

QueryScenario	P.	sqlite4java	SQLite-Xerial	PostgreSQL	Virtuoso	Fuseki	MongoDB	ArangoDB
Set up		19.80 ms	131.28 ms	542.26 ms	**4.89 ms**	Error	7.42 ms	185.40 ms
Load		4.87E5 ms	4.47E5 ms	6.13E5 ms	**2.12E5 ms**	Error	6.88E5 ms	8.61E5 ms
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	P.	15549.49 ms	16427.42 ms	**4591.04 ms**	7599.10 ms	Error	15847.46 ms	58226.17 ms
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	Q.	2.73 ms	**0.52 ms**	1.21 ms	9382.19 ms	Error	999.25 ms	3235.56 ms
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	P.	**3300.80 ms**	11481.62 ms	3814.14 ms	4082.31 ms	Error	12924.41 ms	57183.48 ms
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	Q.	2.23 ms	**1.68 ms**	2.17 ms	1.89E5 ms	Error	9527.43 ms	32524.23 ms
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	P.	3950.24 ms	10487.82 ms	**3793.26 ms**	7698.99 ms	Error	12830.62 ms	56942.40 ms
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	Q.	10.85 ms	**2.59 ms**	3.91 ms	1.37E6 ms	Error	99333.86 ms	3.23E5 ms
AGG._PUBLICATIONS_PER_PUBLISHER_TOP10	P.	1.62 ms	5657.30 ms	3302.00 ms	6.06 ms	Error	**0.00 ms**	**0.00 ms**
AGG._PUBLICATIONS_PER_PUBLISHER_TOP10	Q.	3958.27 ms	795.35 ms	1138.18 ms	**737.28 ms**	Error	4043.91 ms	5209.33 ms
AGG._PUBLICATIONS_PER_PUBLISHER_TOP100	P.	0.90 ms	6328.80 ms	3290.99 ms	2.54 ms	Error	**0.00 ms**	**0.00 ms**
AGG._PUBLICATIONS_PER_PUBLISHER_TOP100	Q.	3992.14 ms	838.87 ms	1174.99 ms	**724.22 ms**	Error	3870.72 ms	5229.33 ms
AGG._PUBLICATIONS_PER_PUBLISHER_ALL	P.	4.52 ms	6177.23 ms	3416.82 ms	2.46 ms	Error	**0.00 ms**	**0.00 ms**
AGG._PUBLICATIONS_PER_PUBLISHER_ALL	Q.	4273.47 ms	**1112.13 ms**	1592.18 ms	3018.93 ms	Error	5229.01 ms	8573.41 ms
AGG._ISSUES_PER_DECADE_TOP10	P.	0.93 ms	9436.19 ms	5484.88 ms	2.72 ms	Error	**0.00 ms**	**0.00 ms**
AGG._ISSUES_PER_DECADE_TOP10	Q.	3992.82 ms	5587.86 ms	**975.43 ms**	2456.79 ms	Error	3063.77 ms	8667.55 ms
AGG._ISSUES_PER_DECADE_TOP100	P.	0.83 ms	9695.52 ms	5048.58 ms	2.34 ms	Error	**0.00 ms**	**0.00 ms**
AGG._ISSUES_PER_DECADE_TOP100	Q.	4269.83 ms	5626.93 ms	**1034.89 ms**	2489.62 ms	Error	2758.35 ms	8746.80 ms
AGG._ISSUES_PER_DECADE_ALL	P.	0.98 ms	9649.27 ms	5204.96 ms	1.84 ms	Error	**0.00 ms**	**0.00 ms**
AGG._ISSUES_PER_DECADE_ALL	Q.	4025.16 ms	5580.76 ms	**1010.88 ms**	2562.25 ms	Error	2920.50 ms	8668.04 ms
CONDITIONAL_TABLE_SCAN_ALL_STUDIES	P.	1.33 ms	7273.73 ms	4308.43 ms	9.11 ms	Error	**0.00 ms**	26747.31 ms
CONDITIONAL_TABLE_SCAN_ALL_STUDIES	Q.	3842.06 ms	4965.95 ms	1989.96 ms	3.42E5 ms	Error	3676.14 ms	**1140.15 ms**
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES	P.	1.35 ms	4.47 ms	2237.44 ms	9.23 ms	Error	**0.00 ms**	**0.00 ms**
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES	Q.	25422.12 ms	27719.41 ms	29016.58 ms	9.01E6 ms	Error	**15985.97 ms**	79574.47 ms
COND._T._S._ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES	P.	3.26 ms	4.08 ms	6118.23 ms	14.56 ms	Error	**0.00 ms**	**0.00 ms**
COND._T._S._ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES	Q.	4796.04 ms	5815.28 ms	**2164.36 ms**	3.52E5 ms	Error	4907.33 ms	14780.01 ms
GRAPH_L._REL._BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY	P.	4.63E5 ms	4.13E5 ms	5319.84 ms	**1398.54 ms**	Error	3626.48 ms	1.21E6 ms
GRAPH_L._REL._BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY	Q.	69.35 ms	32.01 ms	**0.82 ms**	2.30 ms	Error	7374.98 ms	1993.19 ms
GRAPH_L._REL._BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES	P.	2057.81 ms	2781.29 ms	**1386.84 ms**	1456.04 ms	Error	3878.45 ms	26666.50 ms

QueryScenario	P.	sqlite4java	SQLite-Xerial	PostgreSQL	Virtuoso	Fuseki	MongoDB	ArangoDB
GRAPH_L._REL._BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES	Q.	119.51 ms	81.21 ms	**0.90 ms**	27.33 ms	Error	87055.70 ms	16144.05 ms
GRAPH_L._REL._BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES	P.	1883.00 ms	2668.29 ms	**1359.20 ms**	1497.29 ms	Error	3824.36 ms	54174.19 ms
GRAPH_L._REL._BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES	Q.	1702.85 ms	1436.59 ms	2412.83 ms	**336.56 ms**	Error	7.24E5 ms	1.49E5 ms
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	P.	7.26 ms	4.11 ms	10.58 ms	Error	Error	**0.00 ms**	**0.00 ms**
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	Q.	1.14 ms	**0.99 ms**	25215.62 ms	Error	Error	17724.65 ms	3.44E5 ms
SCHEMA_CHANGE_INTRODUCE_STRING_OP	P.	1.15 ms	4678.78 ms	3456.47 ms	Error	Error	**0.00 ms**	**0.00 ms**
SCHEMA_CHANGE_INTRODUCE_STRING_OP	Q.	**21181.29 ms**	23606.17 ms	1.69E5 ms	Error	Error	2.33E5 ms	3.35E5 ms
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	P.	7.29 ms	6.13 ms	4841.35 ms	Error	Error	**0.00 ms**	Error
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	Q.	**66071.02 ms**	77057.79 ms	5.04E5 ms	Error	Error	66910.28 ms	Error
SCHEMA_CHANGE_REMOVE_RDF_TYPE	P.	2.33 ms	4.08 ms	24.04 ms	Error	Error	**0.00 ms**	Error
SCHEMA_CHANGE_REMOVE_RDF_TYPE	Q.	13795.24 ms	17329.30 ms	**57.96 ms**	Error	Error	15041.54 ms	Error
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	P.	0.72 ms	8249.80 ms	3809.45 ms	Error	Error	**0.00 ms**	Error
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	Q.	17342.08 ms	25041.33 ms	1.57E5 ms	Error	Error	**15549.93 ms**	Error
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	P.	6.50 ms	5910.08 ms	5607.18 ms	179.28 ms	Error	**0.00 ms**	Error
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	Q.	3341.25 ms	**2984.77 ms**	12806.84 ms	50413.15 ms	Error	3800.26 ms	Error
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	P.	2.72 ms	5132.60 ms	3404.44 ms	7.33 ms	Error	**0.00 ms**	Error
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	Q.	7494.29 ms	15184.78 ms	15042.79 ms	**2.57 ms**	Error	16511.32 ms	Error
DELETE_HIGH_SELECTIVIY_NON_ISSUED	P.	0.85 ms	323.05 ms	1920.53 ms	Error	Error	**0.00 ms**	Error
DELETE_HIGH_SELECTIVIY_NON_ISSUED	Q.	281.05 ms	217.26 ms	**97.68 ms**	Error	Error	659.22 ms	Error

B.14.2 Test Series Datasets

Test Series MEDIUM	Dataset Size
hebis_10147116_13050073_rdf_gz	206.285 MB

B.14.3 Evaluation Details for sqlite4java, Version 392 with SQLite 3.8.7, Test Series MEDIUM

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
Set up		19.80 ms			

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
Load		4.87E5 ms			
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	Prepare	15549.49 ms			
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	Query	7.74 ms	0.27 ms	0.20 ms	2.73 ms
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	Prepare	3300.80 ms			
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	Query	4.67 ms	1.52 ms	0.50 ms	2.23 ms
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	Prepare	3950.24 ms			
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	Query	27.96 ms	2.22 ms	2.38 ms	10.85 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10	Prepare	1.62 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10	Query	4091.66 ms	3925.19 ms	3857.96 ms	3958.27 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100	Prepare	0.90 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100	Query	3940.71 ms	3940.56 ms	4095.16 ms	3992.14 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL	Prepare	4.52 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL	Query	4372.55 ms	4272.08 ms	4175.79 ms	4273.47 ms
AGGREGATE_ISSUES_PER_DECADE_TOP10	Prepare	0.93 ms			
AGGREGATE_ISSUES_PER_DECADE_TOP10	Query	3916.88 ms	4104.62 ms	3956.97 ms	3992.82 ms
AGGREGATE_ISSUES_PER_DECADE_TOP100	Prepare	0.83 ms			
AGGREGATE_ISSUES_PER_DECADE_TOP100	Query	4239.59 ms	4456.20 ms	4113.70 ms	4269.83 ms
AGGREGATE_ISSUES_PER_DECADE_ALL	Prepare	0.98 ms			
AGGREGATE_ISSUES_PER_DECADE_ALL	Query	4025.15 ms	4002.77 ms	4047.56 ms	4025.16 ms
CONDITIONAL_TABLE_SCAN_ALL_STUDIES	Prepare	1.33 ms			
CONDITIONAL_TABLE_SCAN_ALL_STUDIES	Query	3896.76 ms	3810.43 ms	3818.99 ms	3842.06 ms
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES	Prepare	1.35 ms			
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES	Query	24721.64 ms	26283.99 ms	25260.73 ms	25422.12 ms
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES	Prepare	3.26 ms			
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES	Query	4780.61 ms	4891.59 ms	4715.94 ms	4796.04 ms
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY	Prepare	4.63E5 ms			
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY	Query	137.51 ms	35.70 ms	34.84 ms	69.35 ms
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES	Prepare	2057.81 ms			
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES	Query	211.34 ms	72.80 ms	74.39 ms	119.51 ms
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES	Prepare	1883.00 ms			
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES	Query	1920.72 ms	1580.46 ms	1607.38 ms	1702.85 ms
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	Prepare	7.26 ms			
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	Query	1.14 ms			

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
SCHEMA_CHANGE_INTRODUCE_STRING_OP	Prepare	1.15 ms			
SCHEMA_CHANGE_INTRODUCE_STRING_OP	Query	21181.29 ms			
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	Prepare	7.29 ms			
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	Query	66071.02 ms			
SCHEMA_CHANGE_REMOVE_RDF_TYPE	Prepare	2.33 ms			
SCHEMA_CHANGE_REMOVE_RDF_TYPE	Query	13795.24 ms			
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	Prepare	0.72 ms			
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	Query	17342.08 ms			
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	Prepare	6.50 ms			
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	Query	3341.25 ms			
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	Prepare	2.72 ms			
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	Query	7494.29 ms			
DELETE_HIGH_SELECTIVIY_NON_ISSUED	Prepare	0.85 ms			
DELETE_HIGH_SELECTIVIY_NON_ISSUED	Query	281.05 ms			

B.14.4 Evaluation Details for SQLite-Xerial, Version 3.8.11, Test Series MEDIUM

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
Set up		131.28 ms			
Load		4.47E5 ms			
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	Prepare	16427.42 ms			
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	Query	1.21 ms	0.20 ms	0.17 ms	0.52 ms
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	Prepare	11481.62 ms			
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	Query	3.39 ms	0.93 ms	0.73 ms	1.68 ms
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	Prepare	10487.82 ms			
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	Query	3.06 ms	2.28 ms	2.41 ms	2.59 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10	Prepare	5657.30 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10	Query	788.82 ms	810.48 ms	786.76 ms	795.35 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100	Prepare	6328.80 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100	Query	838.30 ms	839.70 ms	838.60 ms	838.87 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL	Prepare	6177.23 ms			

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL	Query	1136.18 ms	1115.80 ms	1084.41 ms	1112.13 ms
AGGREGATE_ISSUES_PER_DECADE_TOP10	Prepare	9436.19 ms			
AGGREGATE_ISSUES_PER_DECADE_TOP10	Query	5656.21 ms	5577.21 ms	5530.15 ms	5587.86 ms
AGGREGATE_ISSUES_PER_DECADE_TOP100	Prepare	9695.52 ms			
AGGREGATE_ISSUES_PER_DECADE_TOP100	Query	5786.55 ms	5513.64 ms	5580.60 ms	5626.93 ms
AGGREGATE_ISSUES_PER_DECADE_ALL	Prepare	9649.27 ms			
AGGREGATE_ISSUES_PER_DECADE_ALL	Query	5655.68 ms	5536.46 ms	5550.16 ms	5580.76 ms
CONDITIONAL_TABLE_SCAN_ALL_STUDIES	Prepare	7273.73 ms			
CONDITIONAL_TABLE_SCAN_ALL_STUDIES	Query	4967.76 ms	4846.15 ms	5083.95 ms	4965.95 ms
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES	Prepare	4.47 ms			
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES	Query	27866.32 ms	27506.99 ms	27784.92 ms	27719.41 ms
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES	Prepare	4.08 ms			
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES	Query	5937.04 ms	5730.20 ms	5778.59 ms	5815.28 ms
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY	Prepare	4.13E5 ms			
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY	Query	35.12 ms	28.02 ms	32.89 ms	32.01 ms
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES	Prepare	2781.29 ms			
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES	Query	89.91 ms	81.11 ms	72.60 ms	81.21 ms
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES	Prepare	2668.29 ms			
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES	Query	1457.29 ms	1418.60 ms	1433.86 ms	1436.59 ms
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	Prepare	4.11 ms			
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	Query	0.99 ms			
SCHEMA_CHANGE_INTRODUCE_STRING_OP	Prepare	4678.78 ms			
SCHEMA_CHANGE_INTRODUCE_STRING_OP	Query	23606.17 ms			
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	Prepare	6.13 ms			
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	Query	77057.79 ms			
SCHEMA_CHANGE_REMOVE_RDF_TYPE	Prepare	4.08 ms			
SCHEMA_CHANGE_REMOVE_RDF_TYPE	Query	17329.30 ms			
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	Prepare	8249.80 ms			
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	Query	25041.33 ms			
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	Prepare	5910.08 ms			
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	Query	2984.77 ms			
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	Prepare	5132.60 ms			
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	Query	15184.78 ms			

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
DELETE_HIGH_SELECTIVIY_NON_ISSUED	Prepare	323.05 ms			
DELETE_HIGH_SELECTIVIY_NON_ISSUED	Query	217.26 ms			

B.14.5 Evaluation Details for PostgreSQL, Version PostgreSQL 9.4.4 on x86_64-apple-darwin14.3.0, compiled by Apple LLVM version 6.1.0 (clang-602.0.53) (based on LLVM 3.6.0svn), 64-bit / 9.4-1201-jdbc41, Test Series MEDIUM

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
Set up		542.26 ms			
Load		6.13E5 ms			
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	Prepare	4591.04 ms			
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	Query	2.43 ms	0.57 ms	0.63 ms	1.21 ms
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	Prepare	3814.14 ms			
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	Query	3.84 ms	1.42 ms	1.26 ms	2.17 ms
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	Prepare	3793.26 ms			
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	Query	5.50 ms	3.11 ms	3.11 ms	3.91 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10	Prepare	3302.00 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10	Query	1339.26 ms	1038.67 ms	1036.60 ms	1138.18 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100	Prepare	3290.99 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100	Query	1327.32 ms	1075.72 ms	1121.94 ms	1174.99 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL	Prepare	3416.82 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL	Query	1820.37 ms	1520.27 ms	1435.92 ms	1592.18 ms
AGGREGATE_ISSUES_PER_DECADE_TOP10	Prepare	5484.88 ms			
AGGREGATE_ISSUES_PER_DECADE_TOP10	Query	1287.32 ms	813.11 ms	825.87 ms	975.43 ms
AGGREGATE_ISSUES_PER_DECADE_TOP100	Prepare	5048.58 ms			
AGGREGATE_ISSUES_PER_DECADE_TOP100	Query	1213.48 ms	990.76 ms	900.43 ms	1034.89 ms
AGGREGATE_ISSUES_PER_DECADE_ALL	Prepare	5204.96 ms			
AGGREGATE_ISSUES_PER_DECADE_ALL	Query	1300.03 ms	896.77 ms	835.86 ms	1010.88 ms
CONDITIONAL_TABLE_SCAN_ALL_STUDIES	Prepare	4308.43 ms			
CONDITIONAL_TABLE_SCAN_ALL_STUDIES	Query	2236.56 ms	1893.35 ms	1839.98 ms	1989.96 ms
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES	Prepare	2237.44 ms			
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES	Query	34506.99 ms	28414.42 ms	24128.33 ms	29016.58 ms

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES	Prepare	6118.23 ms			
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES	Query	2459.65 ms	2031.49 ms	2001.93 ms	2164.36 ms
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY	Prepare	5319.84 ms			
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY	Query	1.28 ms	0.60 ms	0.58 ms	0.82 ms
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES	Prepare	1386.84 ms			
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES	Query	1.36 ms	0.64 ms	0.69 ms	0.90 ms
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES	Prepare	1359.20 ms			
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES	Query	3638.70 ms	3533.27 ms	66.53 ms	2412.83 ms
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	Prepare	10.58 ms			
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	Query	25215.62 ms			
SCHEMA_CHANGE_INTRODUCE_STRING_OP	Prepare	3456.47 ms			
SCHEMA_CHANGE_INTRODUCE_STRING_OP	Query	1.69E5 ms			
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	Prepare	4841.35 ms			
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	Query	5.04E5 ms			
SCHEMA_CHANGE_REMOVE_RDF_TYPE	Prepare	24.04 ms			
SCHEMA_CHANGE_REMOVE_RDF_TYPE	Query	57.96 ms			
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	Prepare	3809.45 ms			
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	Query	1.57E5 ms			
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	Prepare	5607.18 ms			
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	Query	12806.84 ms			
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	Prepare	3404.44 ms			
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	Query	15042.79 ms			
DELETE_HIGH_SELECTIVIY_NON_ISSUED	Prepare	1920.53 ms			
DELETE_HIGH_SELECTIVIY_NON_ISSUED	Query	97.68 ms			

B.14.6 Evaluation Details for Virtuoso, Version 07.20.3214 / Virtuoso JDBC 4.1, Test Series MEDIUM

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
Set up		4.89 ms			
Load		2.12E5 ms			
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	Prepare	7599.10 ms			

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	Query	13874.17 ms	7202.89 ms	7069.50 ms	9382.19 ms
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	Prepare	4082.31 ms			
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	Query	1.66E5 ms	1.94E5 ms	2.06E5 ms	1.89E5 ms
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	Prepare	7698.99 ms			
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	Query	1.51E6 ms	1.30E6 ms	1.30E6 ms	1.37E6 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10	Prepare	6.06 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10	Query	760.02 ms	729.15 ms	722.66 ms	737.28 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100	Prepare	2.54 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100	Query	715.86 ms	727.20 ms	729.60 ms	724.22 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL	Prepare	2.46 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL	Query	3133.11 ms	2925.65 ms	2998.03 ms	3018.93 ms
AGGREGATE_ISSUES_PER_DECADE_TOP10	Prepare	2.72 ms			
AGGREGATE_ISSUES_PER_DECADE_TOP10	Query	2433.03 ms	2449.24 ms	2488.10 ms	2456.79 ms
AGGREGATE_ISSUES_PER_DECADE_TOP100	Prepare	2.34 ms			
AGGREGATE_ISSUES_PER_DECADE_TOP100	Query	2494.96 ms	2484.53 ms	2489.37 ms	2489.62 ms
AGGREGATE_ISSUES_PER_DECADE_ALL	Prepare	1.84 ms			
AGGREGATE_ISSUES_PER_DECADE_ALL	Query	2559.47 ms	2574.13 ms	2553.15 ms	2562.25 ms
CONDITIONAL_TABLE_SCAN_ALL_STUDIES	Prepare	9.11 ms			
CONDITIONAL_TABLE_SCAN_ALL_STUDIES	Query	3.64E5 ms	3.42E5 ms	3.18E5 ms	3.42E5 ms
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES	Prepare	9.23 ms			
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES	Query	9.01E6 ms	Error	Error	9.01E6 ms
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES	Prepare	14.56 ms			
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES	Query	3.58E5 ms	3.49E5 ms	3.49E5 ms	3.52E5 ms
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY	Prepare	1398.54 ms			
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY	Query	5.02 ms	0.93 ms	0.95 ms	2.30 ms
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES	Prepare	1456.04 ms			
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES	Query	28.82 ms	28.72 ms	24.44 ms	27.33 ms
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES	Prepare	1497.29 ms			
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES	Query	459.45 ms	278.16 ms	272.06 ms	336.56 ms
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	Prepare	Error			
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	Query	Error			
SCHEMA_CHANGE_INTRODUCE_STRING_OP	Prepare	Error			
SCHEMA_CHANGE_INTRODUCE_STRING_OP	Query	Error			

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	Prepare	Error			
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	Query	Error			
SCHEMA_CHANGE_REMOVE_RDF_TYPE	Prepare	Error			
SCHEMA_CHANGE_REMOVE_RDF_TYPE	Query	Error			
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	Prepare	Error			
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	Query	Error			
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	Prepare	179.28 ms			
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	Query	50413.15 ms			
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	Prepare	7.33 ms			
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	Query	2.57 ms			
DELETE_HIGH_SELECTIVIY_NON_ISSUED	Prepare	Error			
DELETE_HIGH_SELECTIVIY_NON_ISSUED	Query	Error			

B.14.7 Evaluation Details for Fuseki, Version 2.3.0 2015-07-25T17:11:28+0000 / jena-libs 2.13.0, Test Series MEDIUM

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
Set up		Error			
Load		Error			
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	Prepare	Error			
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	Query	Error	Error	Error	Error
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	Prepare	Error			
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	Query	Error	Error	Error	Error
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	Prepare	Error			
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	Query	Error	Error	Error	Error
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10	Prepare	Error			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10	Query	Error	Error	Error	Error
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100	Prepare	Error			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100	Query	Error	Error	Error	Error
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL	Prepare	Error			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL	Query	Error	Error	Error	Error
AGGREGATE_ISSUES_PER_DECADE_TOP10	Prepare	Error			

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
AGGREGATE_ISSUES_PER_DECADE_TOP10	Query	Error	Error	Error	Error
AGGREGATE_ISSUES_PER_DECADE_TOP100	Prepare	Error			
AGGREGATE_ISSUES_PER_DECADE_TOP100	Query	Error	Error	Error	Error
AGGREGATE_ISSUES_PER_DECADE_ALL	Prepare	Error			
AGGREGATE_ISSUES_PER_DECADE_ALL	Query	Error	Error	Error	Error
CONDITIONAL_TABLE_SCAN_ALL_STUDIES	Prepare	Error			
CONDITIONAL_TABLE_SCAN_ALL_STUDIES	Query	Error	Error	Error	Error
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES	Prepare	Error			
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES	Query	Error	Error	Error	Error
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES	Prepare	Error			
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES	Query	Error	Error	Error	Error
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY	Prepare	Error			
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY	Query	Error	Error	Error	Error
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES	Prepare	Error			
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES	Query	Error	Error	Error	Error
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES	Prepare	Error			
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES	Query	Error	Error	Error	Error
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	Prepare	Error			
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	Query	Error			
SCHEMA_CHANGE_INTRODUCE_STRING_OP	Prepare	Error			
SCHEMA_CHANGE_INTRODUCE_STRING_OP	Query	Error			
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	Prepare	Error			
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	Query	Error			
SCHEMA_CHANGE_REMOVE_RDF_TYPE	Prepare	Error			
SCHEMA_CHANGE_REMOVE_RDF_TYPE	Query	Error			
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	Prepare	Error			
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	Query	Error			
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	Prepare	Error			
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	Query	Error			
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	Prepare	Error			
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	Query	Error			
DELETE_HIGH_SELECTIVIY_NON_ISSUED	Prepare	Error			

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
DELETE_HIGH_SELECTIVIY_NON_ISSUED	Query	Error			

B.14.8 Evaluation Details for MongoDB, Version 3.0.6, Test Series MEDIUM

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
Set up		7.42 ms			
Load		6.88E5 ms			
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	Prepare	15847.46 ms			
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	Query	1064.41 ms	940.48 ms	992.86 ms	999.25 ms
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	Prepare	12924.41 ms			
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	Query	9480.23 ms	9674.86 ms	9427.21 ms	9527.43 ms
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	Prepare	12830.62 ms			
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	Query	98832.38 ms	99950.01 ms	99219.20 ms	99333.86 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10	Prepare	0.00 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10	Query	4166.53 ms	4162.24 ms	3802.95 ms	4043.91 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100	Prepare	0.00 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100	Query	3851.31 ms	3842.43 ms	3918.42 ms	3870.72 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL	Prepare	0.00 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL	Query	5047.42 ms	5338.55 ms	5301.05 ms	5229.01 ms
AGGREGATE_ISSUES_PER_DECADE_TOP10	Prepare	0.00 ms			
AGGREGATE_ISSUES_PER_DECADE_TOP10	Query	3121.95 ms	3083.72 ms	2985.64 ms	3063.77 ms
AGGREGATE_ISSUES_PER_DECADE_TOP100	Prepare	0.00 ms			
AGGREGATE_ISSUES_PER_DECADE_TOP100	Query	2867.48 ms	2687.86 ms	2719.70 ms	2758.35 ms
AGGREGATE_ISSUES_PER_DECADE_ALL	Prepare	0.00 ms			
AGGREGATE_ISSUES_PER_DECADE_ALL	Query	2800.85 ms	2847.12 ms	3113.52 ms	2920.50 ms
CONDITIONAL_TABLE_SCAN_ALL_STUDIES	Prepare	0.00 ms			
CONDITIONAL_TABLE_SCAN_ALL_STUDIES	Query	3642.24 ms	3765.52 ms	3620.67 ms	3676.14 ms
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES	Prepare	0.00 ms			
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES	Query	16023.49 ms	16218.87 ms	15715.56 ms	15985.97 ms
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES	Prepare	0.00 ms			
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES	Query	4753.62 ms	5081.69 ms	4886.67 ms	4907.33 ms

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY	Prepare	3626.48 ms			
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY	Query	7385.30 ms	7368.93 ms	7370.72 ms	7374.98 ms
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES	Prepare	3878.45 ms			
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES	Query	83988.05 ms	86836.24 ms	90342.80 ms	87055.70 ms
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES	Prepare	3824.36 ms			
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES	Query	7.25E5 ms	7.08E5 ms	7.38E5 ms	7.24E5 ms
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	Prepare	0.00 ms			
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	Query	17724.65 ms			
SCHEMA_CHANGE_INTRODUCE_STRING_OP	Prepare	0.00 ms			
SCHEMA_CHANGE_INTRODUCE_STRING_OP	Query	2.33E5 ms			
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	Prepare	0.00 ms			
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	Query	66910.28 ms			
SCHEMA_CHANGE_REMOVE_RDF_TYPE	Prepare	0.00 ms			
SCHEMA_CHANGE_REMOVE_RDF_TYPE	Query	15041.54 ms			
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	Prepare	0.00 ms			
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	Query	15549.93 ms			
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	Prepare	0.00 ms			
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	Query	3800.26 ms			
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	Prepare	0.00 ms			
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	Query	16511.32 ms			
DELETE_HIGH_SELECTIVIY_NON_ISSUED	Prepare	0.00 ms			
DELETE_HIGH_SELECTIVIY_NON_ISSUED	Query	659.22 ms			

B.14.9 Evaluation Details for ArangoDB, Version 2.6.9 64bit – ICU 54.1, V8 4.1.0.27, OpenSSL 1.0.2d 9 Jul 2015 / Java Driver 2.6.8, Test Series MEDIUM

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
Set up		185.46 ms			
Load		8.61E5 ms			
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	Prepare	58226.17 ms			
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	Query	3226.04 ms	3228.97 ms	3251.67 ms	3235.56 ms

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	Prepare	57183.48 ms			
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	Query	32326.06 ms	32403.50 ms	32843.19 ms	32524.25 ms
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	Prepare	56942.40 ms			
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	Query	3.21E5 ms	3.25E5 ms	3.25E5 ms	3.23E5 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10	Prepare	0.00 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10	Query	5318.30 ms	5155.03 ms	5154.67 ms	5209.33 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100	Prepare	0.00 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100	Query	5159.26 ms	5286.73 ms	5242.06 ms	5229.35 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL	Prepare	0.00 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL	Query	8627.81 ms	8449.40 ms	8643.00 ms	8573.41 ms
AGGREGATE_ISSUES_PER_DECADE_TOP10	Prepare	0.00 ms			
AGGREGATE_ISSUES_PER_DECADE_TOP10	Query	8609.02 ms	8814.36 ms	8579.26 ms	8667.55 ms
AGGREGATE_ISSUES_PER_DECADE_TOP100	Prepare	0.00 ms			
AGGREGATE_ISSUES_PER_DECADE_TOP100	Query	8711.70 ms	8441.43 ms	9087.28 ms	8746.80 ms
AGGREGATE_ISSUES_PER_DECADE_ALL	Prepare	0.00 ms			
AGGREGATE_ISSUES_PER_DECADE_ALL	Query	8752.36 ms	8620.36 ms	8631.40 ms	8668.04 ms
CONDITIONAL_TABLE_SCAN_ALL_STUDIES	Prepare	26747.31 ms			
CONDITIONAL_TABLE_SCAN_ALL_STUDIES	Query	1163.44 ms	1138.12 ms	1118.89 ms	1140.15 ms
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES	Prepare	0.00 ms			
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES	Query	79621.34 ms	79662.87 ms	79439.21 ms	79574.47 ms
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES	Prepare	0.00 ms			
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES	Query	14642.05 ms	15018.80 ms	14679.19 ms	14780.01 ms
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY	Prepare	1.21E6 ms			
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY	Query	2019.36 ms	1956.50 ms	2003.71 ms	1993.19 ms
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES	Prepare	26666.50 ms			
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES	Query	16102.13 ms	16170.04 ms	16159.96 ms	16144.05 ms
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES	Prepare	54174.19 ms			
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES	Query	1.49E5 ms	1.49E5 ms	1.49E5 ms	1.49E5 ms
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	Prepare	0.00 ms			
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	Query	3.44E5 ms			
SCHEMA_CHANGE_INTRODUCE_STRING_OP	Prepare	0.00 ms			
SCHEMA_CHANGE_INTRODUCE_STRING_OP	Query	3.35E5 ms			
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	Prepare	Error			

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	Query	Error			
SCHEMA_CHANGE_REMOVE_RDF_TYPE	Prepare	Error			
SCHEMA_CHANGE_REMOVE_RDF_TYPE	Query	Error			
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	Prepare	Error			
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	Query	Error			
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	Prepare	Error			
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	Query	Error			
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	Prepare	Error			
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	Query	Error			
DELETE_HIGH_SELECTIVIY_NON_ISSUED	Prepare	Error			
DELETE_HIGH_SELECTIVIY_NON_ISSUED	Query	Error			

B.15 Evaluation Report for Test Series Large

B.15.1 Evaluation Overview, all Databases, Test Series LARGE

QueryScenario	Phase	sqlite4java	SQLite-Xerial	PostgreSQL	Virtuoso	Fuseki	MongoDB	ArangoDB
Set up		26.01 ms	146.01 ms	545.21 ms	Error	Error	**5.30 ms**	168.11 ms
Load		1.27E6 ms	**1.19E6 ms**	1.64E6 ms	Error	Error	1.93E6 ms	2.32E6 ms
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	Prepare	56115.56 ms	57742.43 ms	**17664.76 ms**	Error	Error	36047.30 ms	1.58E5 ms
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	Query	9.44 ms	**1.12 ms**	2.93 ms	Error	Error	2604.03 ms	8014.89 ms
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	Prepare	**9214.12 ms**	22439.78 ms	9965.09 ms	Error	Error	32303.63 ms	1.58E5 ms
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	Query	6.26 ms	**2.50 ms**	2.80 ms	Error	Error	24540.70 ms	81197.56 ms
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	Prepare	10215.23 ms	26944.78 ms	**10197.67 ms**	Error	Error	32706.44 ms	1.61E5 ms
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	Query	6.75 ms	**4.33 ms**	10.19 ms	Error	Error	2.46E5 ms	8.58E5 ms
AGG._PUBLICATIONS_PER_PUBLISHER_TOP10	Prepare	1.32 ms	15639.44 ms	8887.14 ms	Error	Error	**0.00 ms**	**0.00 ms**
AGG._PUBLICATIONS_PER_PUBLISHER_TOP10	Query	10823.61 ms	**2151.83 ms**	2919.83 ms	Error	Error	9599.36 ms	14408.54 ms
AGG._PUBLICATIONS_PER_PUBLISHER_TOP100	Prepare	1.41 ms	16973.25 ms	8950.74 ms	Error	Error	**0.00 ms**	**0.00 ms**
AGG._PUBLICATIONS_PER_PUBLISHER_TOP100	Query	11486.73 ms	**2283.26 ms**	3088.30 ms	Error	Error	9597.52 ms	14173.63 ms
AGG._PUBLICATIONS_PER_PUBLISHER_ALL	Prepare	1.90 ms	17295.63 ms	8819.23 ms	Error	Error	Error	**0.00 ms**
AGG._PUBLICATIONS_PER_PUBLISHER_ALL	Query	11590.62 ms	**3124.37 ms**	3912.75 ms	Error	Error	Error	22478.56 ms
AGG._ISSUES_PER_DECADE_TOP10	Prepare	5.29 ms	25100.92 ms	13177.38 ms	Error	Error	**0.00 ms**	**0.00 ms**
AGG._ISSUES_PER_DECADE_TOP10	Query	11872.51 ms	14790.62 ms	**2714.04 ms**	Error	Error	7096.29 ms	20785.09 ms
AGG._ISSUES_PER_DECADE_TOP100	Prepare	1.11 ms	24692.90 ms	13137.28 ms	Error	Error	**0.00 ms**	**0.00 ms**
AGG._ISSUES_PER_DECADE_TOP100	Query	12492.21 ms	14176.74 ms	**2675.42 ms**	Error	Error	7056.18 ms	20471.41 ms
AGG._ISSUES_PER_DECADE_ALL	Prepare	1.07 ms	24580.52 ms	13170.99 ms	Error	Error	**0.00 ms**	**0.00 ms**
AGG._ISSUES_PER_DECADE_ALL	Query	12842.47 ms	14162.48 ms	**2685.78 ms**	Error	Error	7208.70 ms	20592.58 ms
CONDITIONAL_TABLE_SCAN_ALL_STUDIES	Prepare	16.84 ms	19839.92 ms	13272.60 ms	Error	Error	**0.00 ms**	82644.62 ms
COND._T._S._ALL_STUDIES	Query	14250.52 ms	13250.13 ms	7171.27 ms	Error	Error	10205.19 ms	**3115.19 ms**
COND._T._S._ALL_BIBLIOGRAPHIC_RESOURCES	Prepare	2.01 ms	3.44 ms	5485.72 ms	Error	Error	**0.00 ms**	**0.00 ms**
COND._T._S._ALL_BIBLIOGRAPHIC_RESOURCES	Query	64200.50 ms	55987.28 ms	70342.26 ms	Error	Error	**29403.51 ms**	1.76E5 ms
COND._T._S._ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES	Prepare	6.28 ms	4.09 ms	19235.51 ms	Error	Error	**0.00 ms**	1.15 ms
COND._T._S._ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES	Query	12945.22 ms	13167.53 ms	**6119.93 ms**	Error	Error	10084.06 ms	39115.80 ms
GRAPH_L._REL._BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY	Prepare	1.23E6 ms	1.01E6 ms	12145.97 ms	Error	Error	**7303.20 ms**	2.03E6 ms
GRAPH_L._REL._BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY	Query	206.78 ms	48.64 ms	**1.20 ms**	Error	Error	17971.03 ms	4869.37 ms
GRAPH_L._REL._BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES	Prepare	10129.62 ms	5552.17 ms	**3570.32 ms**	Error	Error	7379.75 ms	60645.37 ms

QueryScenario	Phase	sqlite4java	SQLite-Xerial	PostgreSQL	Virtuoso	Fuseki	MongoDB	ArangoDB
GRAPH_L._REL._BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES	Query	376.90 ms	**135.84 ms**	1620.94 ms	Error	Error	2.07E5 ms	44759.81 ms
GRAPH_L._REL._BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES	Prepare	10393.55 ms	6487.42 ms	**3542.21 ms**	Error	Error	6839.11 ms	1.70E5 ms
GRAPH_L._REL._BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES	Query	4006.99 ms	**3084.81 ms**	7341.36 ms	Error	Error	1.81E6 ms	4.38E5 ms
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	Prepare	20.36 ms	6.70 ms	11.72 ms	Error	Error	**0.00 ms**	**0.00 ms**
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	Query	3.51 ms	**1.02 ms**	64333.54 ms	Error	Error	45648.67 ms	1.39E6 ms
SCHEMA_CHANGE_INTRODUCE_STRING_OP	Prepare	4.00 ms	12871.51 ms	8941.02 ms	Error	Error	**0.00 ms**	**0.00 ms**
SCHEMA_CHANGE_INTRODUCE_STRING_OP	Query	75640.81 ms	**66496.15 ms**	5.24E5 ms	Error	Error	6.18E5 ms	1.45E6 ms
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	Prepare	2.10 ms	11.67 ms	17463.10 ms	Error	Error	**0.00 ms**	0.06 ms
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	Query	2.16E5 ms	1.94E5 ms	1.43E6 ms	Error	Error	**1.69E5 ms**	1.86E7 ms
SCHEMA_CHANGE_REMOVE_RDF_TYPE	Prepare	2.46 ms	3.46 ms	81.16 ms	Error	Error	**0.00 ms**	0.08 ms
SCHEMA_CHANGE_REMOVE_RDF_TYPE	Query	1.12E5 ms	39956.90 ms	**156.75 ms**	Error	Error	43014.13 ms	8.39E6 ms
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	Prepare	10.53 ms	11626.21 ms	12493.90 ms	Error	Error	**0.00 ms**	0.08 ms
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	Query	53093.87 ms	55434.40 ms	3.48E5 ms	Error	Error	**30071.94 ms**	8.00E6 ms
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	Prepare	6.67 ms	13914.45 ms	18606.37 ms	Error	Error	**0.00 ms**	0.07 ms
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	Query	9160.28 ms	**5676.63 ms**	46110.89 ms	Error	Error	9358.86 ms	5.41E5 ms
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	Prepare	3.78 ms	13697.71 ms	8688.58 ms	Error	Error	**0.00 ms**	**0.00 ms**
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	Query	**17561.26 ms**	34682.37 ms	29052.12 ms	Error	Error	35961.72 ms	4.40E6 ms
DELETE_HIGH_SELECTIVIY_NON_ISSUED	Prepare	5.07 ms	3486.75 ms	6470.05 ms	Error	Error	**0.00 ms**	0.01 ms
DELETE_HIGH_SELECTIVIY_NON_ISSUED	Query	1950.12 ms	775.04 ms	**666.58 ms**	Error	Error	2821.43 ms	34492.86 ms

B.15.2 Test Series Datasets

Test Series LARGE	Dataset Size
hebis_10147116_13050073_rdf_gz	206.285 MB
hebis_29873806_36057474_rdf_gz	298.021 MB

B.15.3 Evaluation Details for sqlite4java, Version 392 with SQLite 3.8.7, Test Series LARGE

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
Set up		26.01 ms			
Load		1.27E6 ms			
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	Prepare	56115.56 ms			
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	Query	27.58 ms	0.37 ms	0.38 ms	9.44 ms
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	Prepare	9214.12 ms			
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	Query	7.56 ms	3.03 ms	8.18 ms	6.26 ms
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	Prepare	10215.23 ms			
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	Query	9.52 ms	4.73 ms	6.02 ms	6.75 ms
AGG._PUBLICATIONS_PER_PUBLISHER_TOP10	Prepare	1.32 ms			
AGG._PUBLICATIONS_PER_PUBLISHER_TOP10	Query	10815.27 ms	10977.17 ms	10678.40 ms	10823.61 ms
AGG._PUBLICATIONS_PER_PUBLISHER_TOP100	Prepare	1.41 ms			
AGG._PUBLICATIONS_PER_PUBLISHER_TOP100	Query	13046.37 ms	10666.59 ms	10747.23 ms	11486.73 ms
AGG._PUBLICATIONS_PER_PUBLISHER_ALL	Prepare	1.90 ms			
AGG._PUBLICATIONS_PER_PUBLISHER_ALL	Query	11526.99 ms	11648.15 ms	11596.71 ms	11590.62 ms
AGG._ISSUES_PER_DECADE_TOP10	Prepare	5.29 ms			
AGG._ISSUES_PER_DECADE_TOP10	Query	10750.54 ms	11567.81 ms	13299.18 ms	11872.51 ms
AGG._ISSUES_PER_DECADE_TOP100	Prepare	1.11 ms			
AGG._ISSUES_PER_DECADE_TOP100	Query	13081.84 ms	10395.37 ms	13999.41 ms	12492.21 ms
AGG._ISSUES_PER_DECADE_ALL	Prepare	1.07 ms			
AGG._ISSUES_PER_DECADE_ALL	Query	11196.83 ms	14329.78 ms	13000.81 ms	12842.47 ms
CONDITIONAL_TABLE_SCAN_ALL_STUDIES	Prepare	16.84 ms			
CONDITIONAL_TABLE_SCAN_ALL_STUDIES	Query	14092.84 ms	12175.49 ms	16483.22 ms	14250.52 ms
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES	Prepare	2.01 ms			
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES	Query	84387.37 ms	56750.68 ms	51463.44 ms	64200.50 ms
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES	Prepare	6.28 ms			
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES	Query	13033.33 ms	12916.79 ms	12885.55 ms	12945.22 ms
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY	Prepare	1.23E6 ms			
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY	Query	510.27 ms	51.76 ms	58.29 ms	206.78 ms
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES	Prepare	10129.62 ms			
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES	Query	847.98 ms	140.22 ms	142.49 ms	376.90 ms
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES	Prepare	10393.55 ms			
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES	Query	5315.40 ms	3374.11 ms	3331.46 ms	4006.99 ms
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	Prepare	20.36 ms			

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	Query	3.51 ms			
SCHEMA_CHANGE_INTRODUCE_STRING_OP	Prepare	4.00 ms			
SCHEMA_CHANGE_INTRODUCE_STRING_OP	Query	75640.81 ms			
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	Prepare	2.10 ms			
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	Query	2.16E5 ms			
SCHEMA_CHANGE_REMOVE_RDF_TYPE	Prepare	2.46 ms			
SCHEMA_CHANGE_REMOVE_RDF_TYPE	Query	1.12E5 ms			
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	Prepare	10.53 ms			
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	Query	53093.87 ms			
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	Prepare	6.67 ms			
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	Query	9160.28 ms			
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	Prepare	3.78 ms			
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	Query	17561.26 ms			
DELETE_HIGH_SELECTIVIY_NON_ISSUED	Prepare	5.07 ms			
DELETE_HIGH_SELECTIVIY_NON_ISSUED	Query	1950.12 ms			

B.15.4 Evaluation Details for SQLite-Xerial, Version 3.8.11, Test Series LARGE

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
Set up		146.01 ms			
Load		1.19E6 ms			
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	Prepare	57742.43 ms			
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	Query	0.48 ms	0.18 ms	2.70 ms	1.12 ms
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	Prepare	22439.78 ms			
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	Query	0.54 ms	3.49 ms	3.45 ms	2.50 ms
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	Prepare	26944.78 ms			
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	Query	5.43 ms	2.53 ms	5.03 ms	4.33 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10	Prepare	15639.44 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10	Query	2173.07 ms	2127.87 ms	2154.55 ms	2151.83 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100	Prepare	16973.25 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100	Query	2289.82 ms	2277.15 ms	2282.80 ms	2283.26 ms

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL	Prepare	17295.63 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL	Query	3141.29 ms	3152.03 ms	3079.78 ms	3124.37 ms
AGGREGATE_ISSUES_PER_DECADE_TOP10	Prepare	25100.92 ms			
AGGREGATE_ISSUES_PER_DECADE_TOP10	Query	14989.25 ms	14891.03 ms	14491.59 ms	14790.62 ms
AGGREGATE_ISSUES_PER_DECADE_TOP100	Prepare	24692.90 ms			
AGGREGATE_ISSUES_PER_DECADE_TOP100	Query	14196.63 ms	14164.37 ms	14169.21 ms	14176.74 ms
AGGREGATE_ISSUES_PER_DECADE_ALL	Prepare	24580.52 ms			
AGGREGATE_ISSUES_PER_DECADE_ALL	Query	14112.35 ms	14280.15 ms	14094.96 ms	14162.48 ms
CONDITIONAL_TABLE_SCAN_ALL_STUDIES	Prepare	19839.92 ms			
CONDITIONAL_TABLE_SCAN_ALL_STUDIES	Query	13326.35 ms	13169.14 ms	13254.91 ms	13250.13 ms
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES	Prepare	3.44 ms			
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES	Query	55579.45 ms	56727.61 ms	55654.79 ms	55987.28 ms
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES	Prepare	4.09 ms			
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES	Query	13114.77 ms	13109.29 ms	13278.53 ms	13167.53 ms
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY	Prepare	1.01E6 ms			
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY	Query	50.68 ms	45.48 ms	49.76 ms	48.64 ms
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES	Prepare	5552.17 ms			
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES	Query	142.17 ms	135.91 ms	129.43 ms	135.84 ms
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES	Prepare	6487.42 ms			
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES	Query	3146.62 ms	3030.41 ms	3077.39 ms	3084.81 ms
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	Prepare	6.70 ms			
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	Query	1.02 ms			
SCHEMA_CHANGE_INTRODUCE_STRING_OP	Prepare	12871.51 ms			
SCHEMA_CHANGE_INTRODUCE_STRING_OP	Query	66496.15 ms			
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	Prepare	11.67 ms			
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	Query	1.94E5 ms			
SCHEMA_CHANGE_REMOVE_RDF_TYPE	Prepare	3.46 ms			
SCHEMA_CHANGE_REMOVE_RDF_TYPE	Query	39956.90 ms			
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	Prepare	11626.21 ms			
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	Query	55434.40 ms			
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	Prepare	13914.45 ms			
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	Query	5676.63 ms			
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	Prepare	13697.71 ms			

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	Query	34682.37 ms			
DELETE_HIGH_SELECTIVIY_NON_ISSUED	Prepare	3486.75 ms			
DELETE_HIGH_SELECTIVIY_NON_ISSUED	Query	775.04 ms			

B.15.5 Evaluation Details for PostgreSQL, Version PostgreSQL 9.4.4 on x86_64-apple-darwin14.3.0, compiled by Apple LLVM version 6.1.0 (clang-602.0.53) (based on LLVM 3.6.0svn), 64-bit / 9.4-1201-jdbc41, Test Series LARGE

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
Set up		545.21 ms			
Load		1.64E6 ms			
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	Prepare	17664.76 ms			
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	Query	5.58 ms	0.71 ms	2.52 ms	2.93 ms
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	Prepare	9965.09 ms			
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	Query	4.75 ms	1.34 ms	2.33 ms	2.80 ms
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	Prepare	10197.67 ms			
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	Query	14.22 ms	8.10 ms	8.25 ms	10.19 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10	Prepare	8887.14 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10	Query	3441.43 ms	2650.93 ms	2667.13 ms	2919.83 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100	Prepare	8950.74 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100	Query	3729.64 ms	2800.07 ms	2735.18 ms	3088.30 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL	Prepare	8819.23 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL	Query	4471.19 ms	3662.40 ms	3604.65 ms	3912.75 ms
AGGREGATE_ISSUES_PER_DECADE_TOP10	Prepare	13177.38 ms			
AGGREGATE_ISSUES_PER_DECADE_TOP10	Query	3209.76 ms	2447.67 ms	2484.68 ms	2714.04 ms
AGGREGATE_ISSUES_PER_DECADE_TOP100	Prepare	13137.28 ms			
AGGREGATE_ISSUES_PER_DECADE_TOP100	Query	3102.20 ms	2432.18 ms	2491.87 ms	2675.42 ms
AGGREGATE_ISSUES_PER_DECADE_ALL	Prepare	13170.99 ms			
AGGREGATE_ISSUES_PER_DECADE_ALL	Query	3320.64 ms	2430.87 ms	2305.84 ms	2685.78 ms
CONDITIONAL_TABLE_SCAN_ALL_STUDIES	Prepare	13272.60 ms			
CONDITIONAL_TABLE_SCAN_ALL_STUDIES	Query	11207.82 ms	5190.82 ms	5115.15 ms	7171.27 ms
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES	Prepare	5485.72 ms			

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES	Query	81131.34 ms	71789.50 ms	58105.95 ms	70342.26 ms
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES	Prepare	19235.51 ms			
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES	Query	7066.93 ms	5792.11 ms	5500.77 ms	6119.93 ms
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY	Prepare	12145.97 ms			
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY	Query	2.35 ms	0.65 ms	0.59 ms	1.20 ms
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES	Prepare	3570.32 ms			
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES	Query	1811.15 ms	1528.35 ms	1523.33 ms	1620.94 ms
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES	Prepare	3542.21 ms			
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES	Query	7409.75 ms	7152.22 ms	7462.12 ms	7341.36 ms
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	Prepare	11.72 ms			
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	Query	64333.54 ms			
SCHEMA_CHANGE_INTRODUCE_STRING_OP	Prepare	8941.02 ms			
SCHEMA_CHANGE_INTRODUCE_STRING_OP	Query	5.24E5 ms			
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	Prepare	17463.10 ms			
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	Query	1.43E6 ms			
SCHEMA_CHANGE_REMOVE_RDF_TYPE	Prepare	81.16 ms			
SCHEMA_CHANGE_REMOVE_RDF_TYPE	Query	156.75 ms			
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	Prepare	12493.90 ms			
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	Query	3.48E5 ms			
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	Prepare	18606.37 ms			
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	Query	46110.89 ms			
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	Prepare	8688.58 ms			
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	Query	29052.12 ms			
DELETE_HIGH_SELECTIVIY_NON_ISSUED	Prepare	6470.05 ms			
DELETE_HIGH_SELECTIVIY_NON_ISSUED	Query	666.58 ms			

B.15.6 Evaluation Details for Virtuoso, Version 07.20.3214 / Virtuoso JDBC 4.1, Test Series LARGE

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
Set up		Error			
Load		Error			

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	Prepare	Error			
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	Query	Error	Error	Error	Error
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	Prepare	Error			
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	Query	Error	Error	Error	Error
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	Prepare	Error			
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	Query	Error	Error	Error	Error
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10	Prepare	Error			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10	Query	Error	Error	Error	Error
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100	Prepare	Error			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100	Query	Error	Error	Error	Error
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL	Prepare	Error			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL	Query	Error	Error	Error	Error
AGGREGATE_ISSUES_PER_DECADE_TOP10	Prepare	Error			
AGGREGATE_ISSUES_PER_DECADE_TOP10	Query	Error	Error	Error	Error
AGGREGATE_ISSUES_PER_DECADE_TOP100	Prepare	Error			
AGGREGATE_ISSUES_PER_DECADE_TOP100	Query	Error	Error	Error	Error
AGGREGATE_ISSUES_PER_DECADE_ALL	Prepare	Error			
AGGREGATE_ISSUES_PER_DECADE_ALL	Query	Error	Error	Error	Error
CONDITIONAL_TABLE_SCAN_ALL_STUDIES	Prepare	Error			
CONDITIONAL_TABLE_SCAN_ALL_STUDIES	Query	Error	Error	Error	Error
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES	Prepare	Error			
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES	Query	Error	Error	Error	Error
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES	Prepare	Error			
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES	Query	Error	Error	Error	Error
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY	Prepare	Error			
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY	Query	Error	Error	Error	Error
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES	Prepare	Error			
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES	Query	Error	Error	Error	Error
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES	Prepare	Error			
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES	Query	Error	Error	Error	Error
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	Prepare	Error			
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	Query	Error			
SCHEMA_CHANGE_INTRODUCE_STRING_OP	Prepare	Error			

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
SCHEMA_CHANGE_INTRODUCE_STRING_OP	Query	Error			
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	Prepare	Error			
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	Query	Error			
SCHEMA_CHANGE_REMOVE_RDF_TYPE	Prepare	Error			
SCHEMA_CHANGE_REMOVE_RDF_TYPE	Query	Error			
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	Prepare	Error			
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	Query	Error			
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	Prepare	Error			
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	Query	Error			
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	Prepare	Error			
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	Query	Error			
DELETE_HIGH_SELECTIVIY_NON_ISSUED	Prepare	Error			
DELETE_HIGH_SELECTIVIY_NON_ISSUED	Query	Error			

B.15.7 Evaluation Details for Fuseki, Version 2.3.0 2015-07-25T17:11:28+0000 / jena-libs 2.13.0, Test Series LARGE

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
Set up		Error			
Load		Error			
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	Prepare	Error			
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	Query	Error	Error	Error	Error
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	Prepare	Error			
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	Query	Error	Error	Error	Error
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	Prepare	Error			
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	Query	Error	Error	Error	Error
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10	Prepare	Error			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10	Query	Error	Error	Error	Error
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100	Prepare	Error			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100	Query	Error	Error	Error	Error
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL	Prepare	Error			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL	Query	Error	Error	Error	Error

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
AGGREGATE_ISSUES_PER_DECADE_TOP10	Prepare	Error			
AGGREGATE_ISSUES_PER_DECADE_TOP10	Query	Error	Error	Error	Error
AGGREGATE_ISSUES_PER_DECADE_TOP100	Prepare	Error			
AGGREGATE_ISSUES_PER_DECADE_TOP100	Query	Error	Error	Error	Error
AGGREGATE_ISSUES_PER_DECADE_ALL	Prepare	Error			
AGGREGATE_ISSUES_PER_DECADE_ALL	Query	Error	Error	Error	Error
CONDITIONAL_TABLE_SCAN_ALL_STUDIES	Prepare	Error			
CONDITIONAL_TABLE_SCAN_ALL_STUDIES	Query	Error	Error	Error	Error
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES	Prepare	Error			
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES	Query	Error	Error	Error	Error
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES	Prepare	Error			
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES	Query	Error	Error	Error	Error
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY	Prepare	Error			
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY	Query	Error	Error	Error	Error
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES	Prepare	Error			
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES	Query	Error	Error	Error	Error
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES	Prepare	Error			
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES	Query	Error	Error	Error	Error
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	Prepare	Error			
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	Query	Error			
SCHEMA_CHANGE_INTRODUCE_STRING_OP	Prepare	Error			
SCHEMA_CHANGE_INTRODUCE_STRING_OP	Query	Error			
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	Prepare	Error			
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	Query	Error			
SCHEMA_CHANGE_REMOVE_RDF_TYPE	Prepare	Error			
SCHEMA_CHANGE_REMOVE_RDF_TYPE	Query	Error			
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	Prepare	Error			
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	Query	Error			
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	Prepare	Error			
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	Query	Error			
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	Prepare	Error			
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	Query	Error			
DELETE_HIGH_SELECTIVIY_NON_ISSUED	Prepare	Error			

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
DELETE_HIGH_SELECTIVIY_NON_ISSUED	Query	Error			

B.15.8 Evaluation Details for MongoDB, Version 3.0.6, Test Series LARGE

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
Set up		5.30 ms			
Load		1.93E6 ms			
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	Prepare	36047.30 ms			
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	Query	2808.91 ms	2459.08 ms	2544.10 ms	2604.03 ms
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	Prepare	32303.63 ms			
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	Query	24500.49 ms	24500.06 ms	24621.56 ms	24540.70 ms
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	Prepare	32706.44 ms			
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	Query	2.44E5 ms	2.48E5 ms	2.46E5 ms	2.46E5 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10	Prepare	0.00 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10	Query	9732.69 ms	9519.92 ms	9545.45 ms	9599.36 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100	Prepare	0.00 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100	Query	9763.15 ms	9564.52 ms	9464.89 ms	9597.52 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL	Prepare	Error			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL	Query	Error	Error	Error	Error
AGGREGATE_ISSUES_PER_DECADE_TOP10	Prepare	0.00 ms			
AGGREGATE_ISSUES_PER_DECADE_TOP10	Query	7066.40 ms	7113.03 ms	7109.45 ms	7096.29 ms
AGGREGATE_ISSUES_PER_DECADE_TOP100	Prepare	0.00 ms			
AGGREGATE_ISSUES_PER_DECADE_TOP100	Query	7100.20 ms	7065.28 ms	7003.07 ms	7056.18 ms
AGGREGATE_ISSUES_PER_DECADE_ALL	Prepare	0.00 ms			
AGGREGATE_ISSUES_PER_DECADE_ALL	Query	7210.33 ms	7117.53 ms	7298.23 ms	7208.70 ms
CONDITIONAL_TABLE_SCAN_ALL_STUDIES	Prepare	0.00 ms			
CONDITIONAL_TABLE_SCAN_ALL_STUDIES	Query	10352.71 ms	10108.08 ms	10154.78 ms	10205.19 ms
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES	Prepare	0.00 ms			
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES	Query	29900.74 ms	29160.27 ms	29149.54 ms	29403.51 ms
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES	Prepare	0.00 ms			
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES	Query	10160.21 ms	10146.96 ms	9945.00 ms	10084.06 ms

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY	Prepare	7303.20 ms			
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY	Query	18153.86 ms	17780.34 ms	17978.89 ms	17971.03 ms
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES	Prepare	7379.75 ms			
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES	Query	2.16E5 ms	2.07E5 ms	1.97E5 ms	2.07E5 ms
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES	Prepare	6839.11 ms			
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES	Query	1.95E6 ms	1.78E6 ms	1.72E6 ms	1.81E6 ms
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	Prepare	0.00 ms			
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	Query	45648.67 ms			
SCHEMA_CHANGE_INTRODUCE_STRING_OP	Prepare	0.00 ms			
SCHEMA_CHANGE_INTRODUCE_STRING_OP	Query	6.18E5 ms			
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	Prepare	0.00 ms			
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	Query	1.69E5 ms			
SCHEMA_CHANGE_REMOVE_RDF_TYPE	Prepare	0.00 ms			
SCHEMA_CHANGE_REMOVE_RDF_TYPE	Query	43014.13 ms			
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	Prepare	0.00 ms			
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	Query	30071.94 ms			
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	Prepare	0.00 ms			
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	Query	9358.86 ms			
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	Prepare	0.00 ms			
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	Query	35961.72 ms			
DELETE_HIGH_SELECTIVIY_NON_ISSUED	Prepare	0.00 ms			
DELETE_HIGH_SELECTIVIY_NON_ISSUED	Query	2821.43 ms			

B.15.9 Evaluation Details for ArangoDB, Version 2.6.9 64bit – ICU 54.1, V8 4.1.0.27, OpenSSL 1.0.2d 9 Jul 2015 / Java Driver 2.6.8, Test Series LARGE

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
Set up		168.01 ms			
Load		2.32E6 ms			
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	Prepare	1.58E5 ms			
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY	Query	8024.96 ms	7961.33 ms	8056.56 ms	8014.28 ms

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	Prepare	1.58E5 ms			
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES	Query	81408.45 ms	81224.10 ms	80960.20 ms	81197.58 ms
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	Prepare	1.61E5 ms			
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES	Query	8.42E5 ms	8.60E5 ms	8.71E5 ms	8.58E5 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10	Prepare	0.00 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10	Query	14513.76 ms	14264.49 ms	14446.78 ms	14408.34 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100	Prepare	0.00 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100	Query	14115.78 ms	14143.68 ms	14260.52 ms	14173.33 ms
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL	Prepare	0.00 ms			
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL	Query	23568.51 ms	22361.28 ms	21507.10 ms	22478.96 ms
AGGREGATE_ISSUES_PER_DECADE_TOP10	Prepare	0.00 ms			
AGGREGATE_ISSUES_PER_DECADE_TOP10	Query	21161.78 ms	20572.97 ms	20621.72 ms	20785.49 ms
AGGREGATE_ISSUES_PER_DECADE_TOP100	Prepare	0.00 ms			
AGGREGATE_ISSUES_PER_DECADE_TOP100	Query	20404.66 ms	20640.17 ms	20369.41 ms	20471.41 ms
AGGREGATE_ISSUES_PER_DECADE_ALL	Prepare	0.00 ms			
AGGREGATE_ISSUES_PER_DECADE_ALL	Query	20767.96 ms	19887.98 ms	21121.81 ms	20592.58 ms
CONDITIONAL_TABLE_SCAN_ALL_STUDIES	Prepare	82644.62 ms			
CONDITIONAL_TABLE_SCAN_ALL_STUDIES	Query	3099.61 ms	3120.00 ms	3125.97 ms	3115.19 ms
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES	Prepare	0.00 ms			
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES	Query	1.92E5 ms	1.75E5 ms	1.60E5 ms	1.76E5 ms
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES	Prepare	1.15 ms			
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES	Query	39014.97 ms	39515.09 ms	38817.33 ms	39115.80 ms
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY	Prepare	2.03E6 ms			
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY	Query	4861.68 ms	4927.80 ms	4818.62 ms	4869.37 ms
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES	Prepare	60645.37 ms			
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES	Query	41907.43 ms	46065.05 ms	46306.94 ms	44759.81 ms
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES	Prepare	1.70E5 ms			
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES	Query	4.39E5 ms	4.40E5 ms	4.34E5 ms	4.38E5 ms
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	Prepare	0.00 ms			
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY	Query	1.39E6 ms			
SCHEMA_CHANGE_INTRODUCE_STRING_OP	Prepare	0.00 ms			
SCHEMA_CHANGE_INTRODUCE_STRING_OP	Query	1.45E6 ms			
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	Prepare	0.06 ms			

QueryScenario	Phase	1st Exec	2nd Exec	3rd Exec	Average time
SCHEMA_CHANGE_MIGRATE_RDF_TYPE	Query	1.86E7 ms			
SCHEMA_CHANGE_REMOVE_RDF_TYPE	Prepare	0.08 ms			
SCHEMA_CHANGE_REMOVE_RDF_TYPE	Query	8.39E6 ms			
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	Prepare	0.08 ms			
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM	Query	8.00E6 ms			
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	Prepare	0.07 ms			
UPDATE_HIGH_SELECTIVITY_NON_ISSUED	Query	5.41E5 ms			
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	Prepare	0.00 ms			
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM	Query	4.40E6 ms			
DELETE_HIGH_SELECTIVIY_NON_ISSUED	Prepare	0.01 ms			
DELETE_HIGH_SELECTIVIY_NON_ISSUED	Query	34492.86 ms			

B.16 Database Console Printouts

B.16.1 ArangoDB Error Log - Execution of SCHEMA_CHANGE_MIGRATE_RDF_TYPE in MEDIUM Test Series

```
2015-11-17T03:19:22Z [1085] INFO Authentication is turned off
2015-11-17T03:19:22Z [1085] INFO ArangoDB (version 2.6.9 [darwin]) is ready for business. Have
     fun!
2015-11-17T03:19:45Z [1085] INFO creating database 'loddwhbench', directory '/Users/th/arangoDB/
     databases/database-26629072'
2015-11-17T03:19:45Z [1085] INFO created application directory '/usr/local/var/lib/arangodb-apps/
     _db/loddwhbench' for database 'loddwhbench'
2015-11-17T03:19:45Z [1085] INFO collection '_users' does not exist, no authentication available
2015-11-17T03:19:45Z [1085] INFO In database 'loddwhbench': No version information file found in
     database directory.
2015-11-17T03:19:45Z [1085] INFO In database 'loddwhbench': Found 24 defined task(s), 19 task(s)
     to run
2015-11-17T03:19:45Z [1085] INFO In database 'loddwhbench': state prod/standalone/init, tasks
     setupUsers, createUsersIndex, addDefaultUser, updateUserModel, setupGraphs, createModules,
     createRouting, createKickstarterConfiguration, insertRedirectionsAll, setupAal,
     createAalIndex, setupAqlFunctions, createStatistics, createConfiguration, systemAppEndpoints,
      setupQueues, setupJobs, moveFoxxApps, dropAal
2015-11-17T03:19:45Z [1085] INFO In database 'loddwhbench': init successfully finished
2015-11-17T03:20:13Z [1085] INFO dropping database 'loddwhbench', directory '/Users/th/arangoDB/
     databases/database-26629072'
2015-11-17T03:20:13Z [1085] INFO creating database 'loddwhbench', directory '/Users/th/arangoDB/
     databases/database-44454864'
2015-11-17T03:20:13Z [1085] INFO collection '_users' does not exist, no authentication available
2015-11-17T03:20:13Z [1085] INFO In database 'loddwhbench': No version information file found in
     database directory.
2015-11-17T03:20:13Z [1085] INFO In database 'loddwhbench': Found 24 defined task(s), 19 task(s)
     to run
2015-11-17T03:20:13Z [1085] INFO In database 'loddwhbench': state prod/standalone/init, tasks
     setupUsers, createUsersIndex, addDefaultUser, updateUserModel, setupGraphs, createModules,
     createRouting, createKickstarterConfiguration, insertRedirectionsAll, setupAal,
     createAalIndex, setupAqlFunctions, createStatistics, createConfiguration, systemAppEndpoints,
      setupQueues, setupJobs, moveFoxxApps, dropAal
2015-11-17T03:20:13Z [1085] INFO In database 'loddwhbench': init successfully finished
2015-11-17T04:44:21Z [1085] ERROR cannot create json file '/Users/th/arangoDB/SHUTDOWN.tmp': Too
     many open files
2015-11-17T04:44:21Z [1085] ERROR unable to write WAL state file '/Users/th/arangoDB/SHUTDOWN'
2015-11-17T04:44:25Z [1085] WARNING got unexpected error in CollectorThread::collect: no journal
2015-11-17T04:44:25Z [1085] ERROR cannot create datafile '/Users/th/arangoDB/databases/database
     -44454864/collection-60773328/temp-1175062270928.db': Too many open files
2015-11-17T04:44:25Z [1085] WARNING got unexpected error in CollectorThread::collect: no journal
2015-11-17T04:44:25Z [1085] ERROR cannot create datafile '/Users/th/arangoDB/databases/database
     -44454864/collection-60773328/temp-1175062270928.db': Too many open files
2015-11-17T04:44:25Z [1085] WARNING got unexpected error in CollectorThread::collect: no journal
2015-11-17T04:44:25Z [1085] ERROR cannot create datafile '/Users/th/arangoDB/databases/database
     -44454864/collection-60773328/temp-1175062270928.db': Too many open files
2015-11-17T04:44:25Z [1085] WARNING got unexpected error in CollectorThread::collect: no journal
2015-11-17T04:44:25Z [1085] ERROR cannot create datafile '/Users/th/arangoDB/databases/database
     -44454864/collection-60773328/temp-1175062270928.db': Too many open files
2015-11-17T04:44:25Z [1085] WARNING got unexpected error in CollectorThread::collect: no journal
2015-11-17T04:44:25Z [1085] ERROR cannot create datafile '/Users/th/arangoDB/databases/database
     -44454864/collection-60773328/temp-1175062270928.db': Too many open files
2015-11-17T04:44:25Z [1085] WARNING got unexpected error in CollectorThread::collect: no journal
2015-11-17T04:44:25Z [1085] ERROR cannot create datafile '/Users/th/arangoDB/journals/logfile
     -1329070625744.db': Too many open files
2015-11-17T04:44:25Z [1085] ERROR unable to create logfile '/Users/th/arangoDB/journals/logfile
     -1329070625744.db': system error
2015-11-17T04:44:25Z [1085] ERROR unable to create logfile: system error
2015-11-17T04:44:25Z [1085] ERROR unable to create new WAL reserve logfile
```

Listing B.159: ArangoDB Error Log

B.16.2 MongoDB Error Log - Execution of AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL in LARGE Test Series

```
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES
```

```
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL
Fehler bei MongoDB, AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL
AGGREGATE_ISSUES_PER_DECADE_TOP10
com.mongodb.MongoCommandException: Command failed with error 16819: 'exception: Sort exceeded
     memory limit of 104857600 bytes, but did not opt in to external sorting. Aborting operation.
     Pass allowDiskUse:true to opt in.' on server localhost:27017. The full response is { "errmsg"
      : "exception: Sort exceeded memory limit of 104857600 bytes, but did not opt in to external
     sorting. Aborting operation. Pass allowDiskUse:true to opt in.", "code" : 16819, "ok" : 0.0 }
 at com.mongodb.connection.ProtocolHelper.getCommandFailureException(ProtocolHelper.java:77)
 at com.mongodb.connection.CommandProtocol.createCommandResult(CommandProtocol.java:140)
 at com.mongodb.connection.CommandProtocol.receiveMessage(CommandProtocol.java:131)
 at com.mongodb.connection.CommandProtocol.execute(CommandProtocol.java:84)
 at com.mongodb.connection.DefaultServer$DefaultServerProtocolExecutor.execute(DefaultServer.java
      :155)
 at com.mongodb.connection.DefaultServerConnection.executeProtocol(DefaultServerConnection.java
      :219)
 at com.mongodb.connection.DefaultServerConnection.command(DefaultServerConnection.java:146)
 at com.mongodb.operation.CommandOperationHelper.executeWrappedCommandProtocol(
      CommandOperationHelper.java:166)
 at com.mongodb.operation.CommandOperationHelper.executeWrappedCommandProtocol(
      CommandOperationHelper.java:157)
```

Listing B.160: MongoDB Error Log

B.16.3 Virtuoso Error Log - Load of LARGE Test Series

```
09:30:19 OpenLink Virtuoso Universal Server
09:30:19 Version 07.20.3214-pthreads for Darwin as of Aug 18 2015
09:30:19 uses parts of OpenSSL, PCRE, Html Tidy
09:30:20 SQL Optimizer enabled (max 1000 layouts)
09:30:20 Compiler unit is timed at 0.000513 msec
09:30:23 Server received signal 28. Continuing with the default action for that signal.
09:30:25 Checkpoint started
09:30:25 Roll forward started
09:30:25 Roll forward complete
09:30:26 Checkpoint started
09:30:26 Checkpoint finished, log reused
09:30:26 Checkpoint started
09:30:26 Checkpoint finished, log reused
09:30:26 Checkpoint started
09:30:26 Checkpoint finished, log reused
09:30:26 Checkpoint started
09:30:26 Checkpoint finished, log reused
09:30:26 Checkpoint started
09:30:27 Checkpoint finished, log reused
09:30:27 Checkpoint started
09:30:27 Checkpoint finished, log reused
09:30:29 HTTP/WebDAV server online at 8890
09:30:29 Server online at 1111 (pid 1788)
09:31:49 PL LOG: Loader started
09:31:49 PL LOG: File /Users/th/Ph.D./hebisdumps/noHeader/hebis-29873806-36057474.rdf.gz error
     22007 XM033: XML parser detected an error:
 ERROR : Attribute name without value is allowed in HTML but not in XML
at li
09:31:49 PL LOG: No more files to load. Loader has finished,
```

Listing B.161: Virtuoso Error Log

B.16.4 Virtuoso Error Log - Query of SMALL Test Series

```
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL
AGGREGATE_ISSUES_PER_DECADE_TOP10
AGGREGATE_ISSUES_PER_DECADE_TOP100
AGGREGATE_ISSUES_PER_DECADE_ALL
CONDITIONAL_TABLE_SCAN_ALL_STUDIES
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES
```

```
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY
SCHEMA_CHANGE_INTRODUCE_STRING_OP
SCHEMA_CHANGE_MIGRATE_RDF_TYPE
SCHEMA_CHANGE_REMOVE_RDF_TYPE
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM
UPDATE_HIGH_SELECTIVITY_NON_ISSUED
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM
Fehler bei Virtuoso, DELETE_LOW_SELECTIVITY_PAPER_MEDIUM
DELETE_HIGH_SELECTIVIY_NON_ISSUED
virtuoso.jdbc4.VirtuosoException: FRVEC: array in for vectored over max vector length 1290028 >
     1000000
 at virtuoso.jdbc4.VirtuosoResultSet.process_result(VirtuosoResultSet.java:340)
 at virtuoso.jdbc4.VirtuosoResultSet.getMoreResults(VirtuosoResultSet.java:84)
 at virtuoso.jdbc4.VirtuosoPreparedStatement.sendQuery(VirtuosoPreparedStatement.java:87)
 at virtuoso.jdbc4.VirtuosoPreparedStatement.executeUpdate(VirtuosoPreparedStatement.java:117)
 at database.Virtuoso.query(Virtuoso.java:387)
 at main.Benchmark.main(Benchmark.java:86)
Done with BenchmarkObject [database=Virtuoso, prepareQueryScenarioResults=
[...]
Wrote Report for Testserie SMALL
```

Listing B.162: ArangoDB Error Log

B.16.5 Virtuoso Error Log - Query of MEDIUM Test Series

```
ENTITY_RETRIEVAL_BY_ID_ONE_ENTITY
ENTITY_RETRIEVAL_BY_ID_TEN_ENTITIES
ENTITY_RETRIEVAL_BY_ID_100_ENTITIES
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP10
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_TOP100
AGGREGATE_PUBLICATIONS_PER_PUBLISHER_ALL
AGGREGATE_ISSUES_PER_DECADE_TOP10
AGGREGATE_ISSUES_PER_DECADE_TOP100
AGGREGATE_ISSUES_PER_DECADE_ALL
CONDITIONAL_TABLE_SCAN_ALL_STUDIES
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES
Testserie MEDIUM, Virtuoso: QueryScenario CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES took
      too long (over 2000000ms) to finish at execution 1. Cancelling the remaining executions.
CONDITIONAL_TABLE_SCAN_ALL_BIBLIOGRAPHIC_RESOURCES_AND_STUDIES
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_ONE_ENTITY
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_10_ENTITIES
GRAPH_LIKE_RELATED_BY_DCTERMS_SUBJECTS_1HOP_100_ENTITIES
SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY
Fehler bei Virtuoso, SCHEMA_CHANGE_INTRODUCE_NEW_PROPERTY
virtuoso.jdbc4.VirtuosoException: FRVEC: array in for vectored over max vector length 2000000 >
     1000000
 at virtuoso.jdbc4.VirtuosoResultSet.process_result(VirtuosoResultSet.java:340)
 at virtuoso.jdbc4.VirtuosoResultSet.getMoreResults(VirtuosoResultSet.java:84)
 at virtuoso.jdbc4.VirtuosoPreparedStatement.sendQuery(VirtuosoPreparedStatement.java:87)
 at virtuoso.jdbc4.VirtuosoPreparedStatement.executeUpdate(VirtuosoPreparedStatement.java:117)
 at database.Virtuoso.query(Virtuoso.java:387)
 at main.Benchmark.main(Benchmark.java:86)
SCHEMA_CHANGE_INTRODUCE_STRING_OP
Fehler bei Virtuoso, SCHEMA_CHANGE_INTRODUCE_STRING_OP
SCHEMA_CHANGE_MIGRATE_RDF_TYPE
virtuoso.jdbc4.VirtuosoException: SR325: Transaction aborted because it's log after image size
     went above the limit
 at virtuoso.jdbc4.VirtuosoResultSet.process_result(VirtuosoResultSet.java:340)
 at virtuoso.jdbc4.VirtuosoResultSet.getMoreResults(VirtuosoResultSet.java:84)
 at virtuoso.jdbc4.VirtuosoPreparedStatement.sendQuery(VirtuosoPreparedStatement.java:87)
 at virtuoso.jdbc4.VirtuosoPreparedStatement.executeUpdate(VirtuosoPreparedStatement.java:117)
 at database.Virtuoso.query(Virtuoso.java:387)
 at main.Benchmark.main(Benchmark.java:86)
Fehler bei Virtuoso, SCHEMA_CHANGE_MIGRATE_RDF_TYPESCHEMA_CHANGE_REMOVE_RDF_TYPE

virtuoso.jdbc4.VirtuosoException: SR325: Transaction aborted because it's log after image size
     went above the limit
 at virtuoso.jdbc4.VirtuosoResultSet.process_result(VirtuosoResultSet.java:340)
 at virtuoso.jdbc4.VirtuosoResultSet.getMoreResults(VirtuosoResultSet.java:84)
 at virtuoso.jdbc4.VirtuosoPreparedStatement.sendQuery(VirtuosoPreparedStatement.java:87)
 at virtuoso.jdbc4.VirtuosoPreparedStatement.executeUpdate(VirtuosoPreparedStatement.java:117)
 at database.Virtuoso.query(Virtuoso.java:387)
```

```
 at main.Benchmark.main(Benchmark.java:86)
Fehler bei Virtuoso, SCHEMA_CHANGE_REMOVE_RDF_TYPE
UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM
virtuoso.jdbc4.VirtuosoException: FRVEC: array in for vectored over max vector length 2000000 >
     1000000
 at virtuoso.jdbc4.VirtuosoResultSet.process_result(VirtuosoResultSet.java:340)
 at virtuoso.jdbc4.VirtuosoResultSet.getMoreResults(VirtuosoResultSet.java:84)
 at virtuoso.jdbc4.VirtuosoPreparedStatement.sendQuery(VirtuosoPreparedStatement.java:87)
 at virtuoso.jdbc4.VirtuosoPreparedStatement.executeUpdate(VirtuosoPreparedStatement.java:117)
 at database.Virtuoso.query(Virtuoso.java:387)
 at main.Benchmark.main(Benchmark.java:86)
Fehler bei Virtuoso, UPDATE_LOW_SELECTIVITY_PAPER_MEDIUM
UPDATE_HIGH_SELECTIVITY_NON_ISSUED
virtuoso.jdbc4.VirtuosoException: FRVEC: array in for vectored over max vector length 2000000 >
     1000000
 at virtuoso.jdbc4.VirtuosoResultSet.process_result(VirtuosoResultSet.java:340)
 at virtuoso.jdbc4.VirtuosoResultSet.getMoreResults(VirtuosoResultSet.java:84)
 at virtuoso.jdbc4.VirtuosoPreparedStatement.sendQuery(VirtuosoPreparedStatement.java:87)
 at virtuoso.jdbc4.VirtuosoPreparedStatement.executeUpdate(VirtuosoPreparedStatement.java:117)
 at database.Virtuoso.query(Virtuoso.java:387)
 at main.Benchmark.main(Benchmark.java:86)
DELETE_LOW_SELECTIVITY_PAPER_MEDIUM
DELETE_HIGH_SELECTIVIY_NON_ISSUED
Fehler bei Virtuoso, DELETE_HIGH_SELECTIVIY_NON_ISSUED
virtuoso.jdbc4.VirtuosoException: FRVEC: array in for vectored over max vector length 2000000 >
     1000000
 at virtuoso.jdbc4.VirtuosoResultSet.process_result(VirtuosoResultSet.java:340)
 at virtuoso.jdbc4.VirtuosoResultSet.getMoreResults(VirtuosoResultSet.java:84)
 at virtuoso.jdbc4.VirtuosoPreparedStatement.sendQuery(VirtuosoPreparedStatement.java:87)
 at virtuoso.jdbc4.VirtuosoPreparedStatement.executeUpdate(VirtuosoPreparedStatement.java:117)
 at database.Virtuoso.query(Virtuoso.java:387)
 at main.Benchmark.main(Benchmark.java:86)

[...]
Wrote Report for Testserie MEDIUM
```

Listing B.163: Virtuoso Error Log

B.16.6 Fuseki Error Log - Load of MEDIUM and LARGE Test Series

```
[2015-11-16 10:16:04] HttpAction WARN Transaction still active in endWriter - no commit or abort
     seen (forced abort)
[2015-11-16 10:16:14] Fuseki   WARN [2] RC = 500 : Java heap space
java.lang.OutOfMemoryError: Java heap space
 at java.util.concurrent.ConcurrentHashMap.putVal(ConcurrentHashMap.java:1019)
 at java.util.concurrent.ConcurrentHashMap.put(ConcurrentHashMap.java:1006)
 at sun.util.resources.ParallelListResourceBundle.loadLookupTablesIfNecessary(
      ParallelListResourceBundle.java:169)
 at sun.util.resources.ParallelListResourceBundle.handleKeySet(ParallelListResourceBundle.java
      :134)
 at sun.util.resources.ParallelListResourceBundle.keySet(ParallelListResourceBundle.java:143)
 at sun.util.resources.ParallelListResourceBundle.containsKey(ParallelListResourceBundle.java
      :129)
 at sun.util.resources.ParallelListResourceBundle$KeySet.contains(ParallelListResourceBundle.java
      :208)
 at sun.util.resources.ParallelListResourceBundle.containsKey(ParallelListResourceBundle.java
      :129)
 at sun.util.locale.provider.LocaleResources.getDecimalFormatSymbolsData(LocaleResources.java
      :181)
 at java.text.DecimalFormatSymbols.initialize(DecimalFormatSymbols.java:616)
 at java.text.DecimalFormatSymbols.<init>(DecimalFormatSymbols.java:113)
 at sun.util.locale.provider.DecimalFormatSymbolsProviderImpl.getInstance(
      DecimalFormatSymbolsProviderImpl.java:85)
 at java.text.DecimalFormatSymbols.getInstance(DecimalFormatSymbols.java:180)
 at java.util.Formatter.getZero(Formatter.java:2283)
 at java.util.Formatter.<init>(Formatter.java:1892)
 at java.util.Formatter.<init>(Formatter.java:1914)
 at java.lang.String.format(String.java:2927)
 at org.apache.jena.riot.SysRIOT.fmtMessage(SysRIOT.java:50)
 at org.apache.jena.riot.system.ErrorHandlerFactory$ErrorLogger.logWarning(ErrorHandlerFactory.
      java:77)
 at org.apache.jena.riot.system.ErrorHandlerFactory$ErrorHandlerStd.warning(ErrorHandlerFactory.
      java:121)
 at org.apache.jena.riot.lang.LangRDFXML$ErrorHandlerBridge.warning(LangRDFXML.java:240)
 at org.apache.jena.rdfxml.xmlinput.impl.ARPSaxErrorHandler.warning(ARPSaxErrorHandler.java:42)
```

```
 at org.apache.jena.rdfxml.xmlinput.impl.XMLHandler.warning(XMLHandler.java:191)
 at org.apache.jena.rdfxml.xmlinput.impl.XMLHandler.warning(XMLHandler.java:173)
 at org.apache.jena.rdfxml.xmlinput.impl.XMLHandler.warning(XMLHandler.java:168)
 at org.apache.jena.rdfxml.xmlinput.impl.ParserSupport.warning(ParserSupport.java:194)
 at org.apache.jena.rdfxml.xmlinput.impl.ParserSupport.checkString(ParserSupport.java:105)
 at org.apache.jena.rdfxml.xmlinput.impl.ARPString.<init>(ARPString.java:46)
 at org.apache.jena.rdfxml.xmlinput.states.WantLiteralValueOrDescription.endElement(
     WantLiteralValueOrDescription.java:82)
 at org.apache.jena.rdfxml.xmlinput.impl.XMLHandler.endElement(XMLHandler.java:121)
 at org.apache.xerces.parsers.AbstractSAXParser.endElement(Unknown Source)
 at org.apache.xerces.impl.XMLNamespaceBinder.handleEndElement(Unknown Source)
[2015-11-16 10:16:14] Fuseki   INFO [2] 500 Java heap space (1,510.840 s)
```

Listing B.164: Fuseki Error Log

Appendix C

Framework Appendix

C.1 LODicity Warehouse

```
package com.github.heussd.lodicity.store;

import java.io.Closeable;
import java.util.Arrays;
import java.util.List;
import java.util.function.Consumer;
import java.util.logging.Level;

import org.apache.commons.io.IOUtils;
import org.hibernate.Criteria;
import org.hibernate.EntityMode;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.cfg.Configuration;
import org.hibernate.cfg.Environment;
import org.hibernate.criterion.Criterion;
import org.hibernate.criterion.Projections;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import com.github.heussd.lodicity.data.MetaData;
import com.github.heussd.lodicity.model.DataObject;
import com.github.heussd.lodicity.model.DataObjectIterable;
import com.github.heussd.lodicity.model.Schema;

public class Warehouse implements Closeable {

 private static final Logger LOGGER = LoggerFactory.getLogger(Warehouse.class);
 private SessionFactory factory;
 private Session session;
 private Transaction transaction;

 public Warehouse() {
  this(false, DataObject.class);
 }

 @SafeVarargs
 public Warehouse(Class<? extends DataObject>... dataObjectClasses) {
  this(false, dataObjectClasses);
 }

 /**
  *
  * @param clear
  *      Remove all existing structures (true) or re-use existing
  * @param dataObjectClasses
  */
 @SafeVarargs
 public Warehouse(boolean clear, Class<? extends DataObject>... dataObjectClasses) {
  LOGGER.debug("T.H. LODicity Warehouse");
  java.util.logging.Logger.getLogger("org.hibernate").setLevel(Level.FINEST);
  try {
   Configuration configuration = new Configuration().configure();
   configuration.setProperty(Environment.DEFAULT_ENTITY_MODE, EntityMode.MAP.toString());
   configuration.setProperty(Environment.SHOW_SQL, "false");
   configuration.setInterceptor(new DataObjectInterceptor());

   if (clear) {
```

```
   LOGGER.warn("Clear-Flag set, will erase existing data structures");
   configuration.setProperty(Environment.HBM2DDL_AUTO, "create");
  }

  for (Class<? extends DataObject> dataObjectClass : dataObjectClasses) {
   LOGGER.debug("Registering DataObject Type {}", dataObjectClass.getSimpleName());

   // Make a Hibernate mapping based on Schema information
   configuration.addInputStream(IOUtils.toInputStream(Schema.generateHibernateMapping(
        dataObjectClass), "UTF-8"));
  }

  factory = configuration.buildSessionFactory();
  session = factory.openSession();

 } catch (Throwable e) {
  throw new RuntimeException("Failed to create Warehouse", e);
 }
}

public void persist(DataObject... dataObjects) {
 persist(Arrays.asList(dataObjects));
}

@SuppressWarnings("unchecked")
public Iterable<? extends DataObject> all(Class<? extends DataObject> dataObjectClass) {
 return new DataObjectIterable(dataObjectClass, session.createCriteria(dataObjectClass).list());
}

public void forEach(Class<? extends DataObject> dataObjectClass, Consumer<? super DataObject>
     consumer) {
 all(dataObjectClass).forEach(consumer);
}

@Override
public void close() {
 assert session != null : "Session is null";
 assert session.isOpen() : "Session is not open";
 session.close();
}

public void persist(List<? extends DataObject> dataObjects) {
 assert session != null : "Session is null";
 assert dataObjects.size() != 0 : "No DataObject(s) given";
 LOGGER.info("Persisting {} items...", dataObjects.size());

 long start = System.nanoTime();
 Transaction transaction = session.beginTransaction();
 for (DataObject dataObject : dataObjects) {

  session.saveOrUpdate(dataObject);
 }
 long end = System.nanoTime();
 LOGGER.info("Persisted items in {} milliseconds", (end - start) / 100000);
 transaction.commit();
}

public void update(DataObject dataObject) {
 assert session != null : "Session is null";
 assert dataObject != null : "No DataObject given";

 Transaction transaction = session.beginTransaction();
 session.merge(dataObject);
 transaction.commit();
}

public Iterable<? extends DataObject> query(Filter... filters) {
 assert filters.length != 0 : "At least one filter is required";
 Criteria criteria = criteriaFromFilters(filters);

 LOGGER.debug("Firing query with critera {}", criteria.toString());
 return new DataObjectIterable(filters[0].getDataObjectClass(), criteria.list());
}

public Long count(Class<? extends DataObject> dataObjectClass) {
 return count(new Filter(dataObjectClass));
}
```

```
public Long count(Filter... filters) {
 assert filters.length > 0 : "At least one filter is required";
 Criteria criteria = criteriaFromFilters(filters);

 LOGGER.debug("Firing query with critera {}", criteria.toString());
 return (Long) criteria.setProjection(Projections.rowCount()).uniqueResult();
}

private Criteria criteriaFromFilters(Filter[] filters) {
 Criteria criteria = session.createCriteria(filters[0].getDataObjectClass());
 Arrays.asList(filters).forEach(filter -> {
  Criterion criterion = filter.getCriterion();
  if (criterion != null)
   criteria.add(criterion);
 });
 return criteria;
}

public void openTransaction() {
 this.transaction = session.beginTransaction();
}

public void massUpdate(DataObject dataObject) {
 assert session != null : "Session is null";
 assert dataObject != null : "No DataObject given";
 assert this.transaction != null : "No transaction";

 session.merge(dataObject);
}

public void commit() {
 if (this.transaction != null)
  this.transaction.commit();
}

public MetaData getMetaData(String identifer) {
 List<MetaData> list = session.createQuery("FROM MetaData metaData WHERE metaData.
      dataSourceIdentifier= :identifier")
   .setParameter("identifier", identifer).list();
 if (list.size() > 0) {
  return list.get(0);
 } else {
  return new MetaData(identifer);
 }
}

public void persistMetaData(MetaData metaData) {
 Transaction t = session.beginTransaction();
 session.saveOrUpdate(metaData);
 t.commit();
}

}
```

Listing C.1: Java Source Code of Warehouse.

C.2 Schema

```
package com.github.heussd.lodicity.model;

import java.io.File;
import java.io.FileInputStream;
import java.io.InputStream;
import java.io.StringWriter;
import java.io.UnsupportedEncodingException;
import java.io.Writer;
import java.net.URL;
import java.util.ArrayList;
import java.util.Collection;
import java.util.Collections;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Map;
import java.util.Map.Entry;
```

```
import java.util.Set;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.transform.OutputKeys;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerException;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;

import org.apache.poi.ss.usermodel.Row;
import org.apache.poi.ss.usermodel.Sheet;
import org.apache.poi.ss.usermodel.Workbook;
import org.apache.poi.ss.usermodel.WorkbookFactory;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.w3c.dom.Document;
import org.w3c.dom.Element;

/**
 * Offers a number of convenience methods to access the Schema for {@link DataObject} or to
     validate {@link DataObject} instances.
 *
 * @author Timm Heuss, Hochschule Darmstadt - University of Applied Sciences, 2013
 */
public class Schema {

 private static final Logger LOGGER = LoggerFactory.getLogger(Schema.class);
 private final static String SCHEMA_NAME = "lodicity.schema.xlsx";

 private final static Schema INSTANCE;

 static {
  try {
   File schemaFile = new File(Schema.class.getResource("/" + SCHEMA_NAME).toURI());
   LOGGER.info("Using schema definition from {}", schemaFile);
   if (!schemaFile.exists())
    throw new RuntimeException(schemaFile.getName() + " could not be found.");
   INSTANCE = new Schema(schemaFile);
  } catch (Exception e) {
   throw new RuntimeException("Cannot load Schema", e);
  }
 }

 public static Schema getInstance() {
  return INSTANCE;
 }

 /**
  * <b>How to use</b>: The schema is implemented with three nested {@link HashMap}s. Their keys
      have the following hierarchy:<br>
  * <code>type -> attribute -> {@link SchemaProperty} -> property value</code> <br>
  * for example:<br/>
  * <code>Resource -> creator -> SchemaProperty.IS_LIST_TYPE -> true</code>
  */
 private Map<String, HashMap<String, HashMap<SchemaProperty, Object>>> schemaModel;

 private Schema(File file) {
  try {
   schemaModel = new HashMap<>();

   Workbook workbook = null;
   try (InputStream inputStream = new FileInputStream(file)) {
    workbook = WorkbookFactory.create(inputStream);
   }

   for (int i = 0; i < workbook.getNumberOfSheets(); i++) {
    Sheet sheet = workbook.getSheetAt(i);

    // Make sure the selected sheet has the right structure
    Row firstRow = sheet.getRow(0);
    if (!firstRow.getCell(SchemaProperty.ATTRIBUTE_NAME.cellIndex).toString().equals("Attribute"))
     continue;

    assert firstRow.getCell(SchemaProperty.CARDINALITY.cellIndex).toString().equals("Cardinality")
```

```
        : "Invalid sheet structure in sheet \""
    + sheet.getSheetName() + "\": Cardinality not found";

   assert firstRow.getCell(SchemaProperty.DATATYPE.cellIndex).toString().equals("Datatype") : "
        Invalid sheet structure in sheet \""
    + sheet.getSheetName() + "\": Datatype not found";

   assert firstRow.getCell(SchemaProperty.APPLICATION.cellIndex).toString().equals("Application")
        : "Invalid sheet structure in sheet \""
    + sheet.getSheetName() + "\": Application not found";

   // Produce a random access structure of the selected sheet
   HashMap<String, HashMap<SchemaProperty, Object>> attributeDefintion = new HashMap<>();
   for (int rowNumber = 1; rowNumber <= sheet.getLastRowNum(); rowNumber++) {
    Row attributeRow = sheet.getRow(rowNumber);

    HashMap<SchemaProperty, Object> attributes = new HashMap<>();

    // There might be empty rows in the schema, skip them here
    if (attributeRow == null)
     continue;

    // The getStringCellValue method explicitly validates a cell
    // to be of type String - this is pretty much of use here...
    String attribute = attributeRow.getCell(SchemaProperty.ATTRIBUTE_NAME.cellIndex) != null
      ? attributeRow.getCell(SchemaProperty.ATTRIBUTE_NAME.cellIndex).getStringCellValue() : null
          ;

    if (attribute == null)
     continue;

    // Read all properties from excel
    for (SchemaProperty schemaProperty : SchemaProperty.values()) {
     // Only consider properties with a valid cell index
     if (schemaProperty.cellIndex > 0) {
      attributes.put(schemaProperty,
        attributeRow.getCell(schemaProperty.cellIndex) != null ? attributeRow.getCell(
            schemaProperty.cellIndex).toString() : null);
     }
    }

    // Interpret cardinality, create field IS_LIST_TYPE
    if (attributes.get(SchemaProperty.CARDINALITY) != null) {
     String cardinality = (String) attributes.get(SchemaProperty.CARDINALITY);
     assert (cardinality.length() == 3 || cardinality.length() == 4) : "Invalid cardinality: \""
         + cardinality + "\"";

     attributes.put(SchemaProperty.IS_LIST_TYPE, cardinality.substring(cardinality.length() - 1,
         cardinality.length()).equals("*"));
    }

    // Interpret enum entries, create field VALUES
    if (attributes.get(SchemaProperty.DATATYPE) != null) {
     String datatype = (String) attributes.get(SchemaProperty.DATATYPE);

     if (datatype.length() > 4 && datatype.substring(0, 4).equals("enum")) {
      Collection<String> values = new ArrayList<>();
      Collections.addAll(values, (datatype.substring(5, datatype.length() - 1)).split(", "));
      attributes.put(SchemaProperty.VALUES, values);

      // Change DATATYPE
      attributes.put(SchemaProperty.DATATYPE, "String");

     }
    }
    attributeDefintion.put(attribute, attributes);
   }
   schemaModel.put(sheet.getSheetName(), attributeDefintion);
  }
 } catch (Throwable t) {
  throw new RuntimeException("Cannot initialize schema", t);
 }
}

/**
 * Returns all valid attribute-keys defined for the given {@link DataObject} -specialization.
 *
```

```
 * @param class1
 * @return
 */
public static Set<String> getAttributes(DataObject dataObject) {
 return getInstance().schemaModel.get(dataObject.getClass().getSimpleName()).keySet();
}

/**
 * Returns all valid attribute-keys defined for the given {@link DataObject} -specialization.
 *
 * @param class1
 * @return
 */
public static Set<String> getAttributes(String dataObject) {
 HashMap<String, HashMap<SchemaProperty, Object>> dataObjectModel = getInstance().schemaModel.
     get(dataObject);
 if (dataObjectModel == null) {
  throw new RuntimeException("No schema definition found for dataObject \"" + dataObject + "\"");
 }
 return dataObjectModel.keySet();
}

/**
 * Returns all valid attribute-keys for the given application defined for the given {@link
     DataObject} -specialization.
 *
 * @param class1
 * @return
 */
public static Set<String> getAttributes(DataObject dataObject, String application) {
 HashMap<String, HashMap<SchemaProperty, Object>> dataObjectAttributes = getInstance().
     schemaModel.get(dataObject.getClass().getSimpleName());
 Set<String> applicationKeySet = new HashSet<>();

 // iterate through all attributes of the dataObject
 for (Entry<String, HashMap<SchemaProperty, Object>> stringHashMapEntry : dataObjectAttributes.
     entrySet()) {
  Entry<String, HashMap<SchemaProperty, Object>> pairs = (Entry) stringHashMapEntry;

  // get SchemaPropertys
  HashMap<SchemaProperty, Object> schemaPropertys = pairs.getValue();

  // get application
  String applicationSchemaProperty = (String) schemaPropertys.get(SchemaProperty.APPLICATION);

  // schema property for the given application == null means for both (museum and library)
  if (applicationSchemaProperty == null || applicationSchemaProperty.equals(application)) {
   applicationKeySet.add((String) pairs.getKey());
  }

  // it.remove(); // avoids a ConcurrentModificationException
 }
 return applicationKeySet;
}

/**
 * Returns all properties of a given attribute of a given type.
 *
 * @param dataObject
 * @param attribute
 * @return
 */
public static Map<SchemaProperty, Object> getAttributeDefinition(DataObject dataObject, String
     attribute) {
 assert getInstance().schemaModel.containsKey(dataObject.getClass().getSimpleName()) : "Schema
     definition not found for type \""
   + dataObject.getClass().getSimpleName() + "\"";

 Map<SchemaProperty, Object> attributeDefinition = getInstance().schemaModel.get(dataObject.
     getClass().getSimpleName()).get(attribute);

 assert attributeDefinition != null : "No attribute definition found for attribute \"" +
     attribute + "\" in type \""
   + dataObject.getClass().getSimpleName() + "\"";

 return attributeDefinition;
}
```

```
/**
 * Indicates if a given attribute of a given {@link DataObject} is a List-type.
 *
 * @param dataObject
 * @param attribute
 * @return
 */
public static boolean isListType(DataObject dataObject, String attribute) {
 Map<SchemaProperty, Object> attributeDefinition = getAttributeDefinition(dataObject, attribute)
      ;

 return attributeDefinition.get(SchemaProperty.IS_LIST_TYPE) != null && (boolean)
      attributeDefinition.get(SchemaProperty.IS_LIST_TYPE);
}

/**
 * Indicates if a given attribute of a given {@link DataObject} is a List-type.
 *
 * @param dataObject
 * @param attribute
 * @return
 */
public static boolean isListType(Class<? extends DataObject> dataObjectClass, String attribute) {
 assert dataObjectClass != null : "DataObjectClass is null";
 assert attribute != null : "Attribute is null";

 Map<SchemaProperty, Object> attributeDefinition = getInstance().schemaModel.get(dataObjectClass
      .getSimpleName()).get(attribute);

 assert attributeDefinition != null : "no attribute definition found for attribute " + attribute
       + " in type " + dataObjectClass.getSimpleName();
 return attributeDefinition.get(SchemaProperty.IS_LIST_TYPE) != null && (boolean)
      attributeDefinition.get(SchemaProperty.IS_LIST_TYPE);
}

public static boolean isMandatory(DataObject dataObject, String attribute) {
 Map<SchemaProperty, Object> attributeDefinition = getAttributeDefinition(dataObject, attribute)
      ;

 return attributeDefinition.get(SchemaProperty.CARDINALITY) != null && attributeDefinition.get(
      SchemaProperty.CARDINALITY).toString().startsWith("1");
}

public static boolean isOptional(DataObject dataObject, String attribute) {
 Map<SchemaProperty, Object> attributeDefinition = getAttributeDefinition(dataObject, attribute)
      ;

 return attributeDefinition.get(SchemaProperty.CARDINALITY) != null && attributeDefinition.get(
      SchemaProperty.CARDINALITY).toString().startsWith("0");
}

/**
 * @deprecated Use {@link #getDataType(DataObject, String)} instead.
 */
public static String getSchemaDefinedDataType(DataObject dataObject, String attribute) {
 return getInstance().schemaModel.get(dataObject.getClass().getSimpleName()).get(attribute).get(
      SchemaProperty.DATATYPE).toString();
}

public static boolean isValid(DataObject dataObject, String attribute, Object value) {
 // Special logic applies to this attribute:
 if (attribute.equals("_class_") && dataObject.getClass().getSimpleName().equals(value))
  return true;

 try {
  Map<SchemaProperty, Object> attributeDefinition = Schema.getAttributeDefinition(dataObject,
       attribute);

  if (value == null) {
   /*
    * Does the schema allow the value to be empty or null?
    */
   if (Schema.isOptional(dataObject, attribute))
    return true;

   if (Schema.isMandatory(dataObject, attribute)) {
```

```
    return false;
   }

   // Value is null, but not cardinality is given -> OK
   return true;
  }

  String schemaDefinedDataType = Schema.getDataType(dataObject, attribute);

  if (Schema.isListType(dataObject, attribute) || Schema.isPairType(dataObject, attribute)) {
   // List types go in here...
   // Special validation case for Pair* types: Validate
   // them as if they are lists

   assert value instanceof Collection<?> : "Invalid attribute type \"" + value.getClass().
        getSimpleName() + "\", expected type was \"Collection\"";

   if (((Collection) value).size() == 0 && isMandatory(dataObject, attribute))
    throw new RuntimeException("Mendatory list has zero elements");

   for (Object innerValue : (Collection<?>) value) {

    if (schemaDefinedDataType != null && schemaDefinedDataType.startsWith("Pair")) {
     // Special validation case for Pair* types: Validate
     // them as if they are lists
     assert (value.getClass().getSimpleName()
       .equals(ArrayList.class.getSimpleName())) : "Attribute of type \"Pair\" does not contain
            an ArrayList, but \""
         + value.getClass().getSimpleName() + "\"";
    } else {
     assert schemaDefinedDataType == null
       || innerValue.getClass().getSimpleName().equals(schemaDefinedDataType) : "Invalid list
            entity type \""
         + innerValue.getClass().getSimpleName() + "\", expected type was \"" +
              schemaDefinedDataType + "\"";

     if (attributeDefinition.containsKey(SchemaProperty.VALUES)) {
      // Schema defines the exact inner-list values that
      // are allowed for this attribute
      assert ((ArrayList<String>) attributeDefinition.get(SchemaProperty.VALUES))
        .contains(innerValue) : "Invalid inner-list attribute value \"" + innerValue + "\",
            expected any of "
          + attributeDefinition.get(SchemaProperty.VALUES);
     }
    }
   }
  } else {
   // Non-List-type attribute, make sure it is of the
   // schema-defined type
   if (schemaDefinedDataType != null && !schemaDefinedDataType.equals("null")) {
    // Most likely because of the JSON framework, number types
    // are sometimes mixed up, e.g. an Integer is read as Long.
    // We don't seem to have influence on that, so we do not
    // verify their current data type, but if they are castable
    // into the Schema-defined type.
    try {
     switch (schemaDefinedDataType) {
     case "Float":
      new Float(value + "");
      break;
     case "Integer":
      new Integer(value + "");
      break;
     case "URL":
      new URL(value.toString());
      break;
     case "Boolean":
      String v = value.toString();
      if (!(v.equals("true") || v.equals("false"))) {
       throw new ClassCastException("Cannot convert the value " + v + " to Boolean");
      }
      break;
     default:
      // General purpose datatype validation
      assert value.getClass().getSimpleName().equals(schemaDefinedDataType) : "Invalid attribute
           type \""
        + value.getClass().getSimpleName() + "\", expected type was \"" + schemaDefinedDataType +
```

```
            "\"";
     break;
     }
    } catch (Exception e) {
     throw new RuntimeException("Cannot cast type \"" + value.getClass().getSimpleName() + "\" to
          \"" + schemaDefinedDataType + "\"", e);
    }
   }

   if (attributeDefinition.containsKey(SchemaProperty.VALUES)) {
    // Schema defines the exact values that are allowed for this
    // attribute
    assert ((ArrayList<String>) attributeDefinition.get(SchemaProperty.VALUES)).contains(value) :
         "Invalid attribute value \"" + value
      + "\", expected any of " + attributeDefinition.get(SchemaProperty.VALUES);
   }

  }

  return true;
 } catch (Throwable e) {
  throw new RuntimeException(
    "Validation failed for attribute \"" + attribute + "\", value \"" + value + "\" in type \"" +
         dataObject.getClass().getSimpleName() + "\"",
    e);
 }

}

/**
 * Validates the given {@link DataObject} against the schema. In case of validation failures, a
     {@link RuntimeException} is thrown.
 */
public static void validate(DataObject dataObject) {
 assert dataObject != null : "Cannot validate a null object";

 for (String attribute : getAttributes(dataObject)) {
  // Make a validated get
  dataObject.get(attribute);
 }
}

public static boolean isPairType(DataObject dataObject, String attribute) {
 String schemaDefinedDataType = Schema.getDataType(dataObject, attribute);
 if (schemaDefinedDataType != null)
  return schemaDefinedDataType.startsWith("Pair");

 return false;
}

public static String getDataType(DataObject dataObject, String attribute) {
 if (attribute.equals("_class_")) {
  return "String";
 }

 try {
  return (String) getInstance().schemaModel.get(dataObject.getClass().getSimpleName()).get(
       attribute).get(SchemaProperty.DATATYPE);
 } catch (Exception e) {
  throw new RuntimeException("Cannot read data type of attribute " + attribute + " of type " +
       dataObject.getClass().getSimpleName());
 }
}

public static String getDataType(Class<? extends DataObject> dataObjectClass, String attribute) {
 return (String) getInstance().schemaModel.get(dataObjectClass.getSimpleName()).get(attribute).
      get(SchemaProperty.DATATYPE);
}

public static String generateHibernateMapping(Class<? extends DataObject> dataObjectClass)
  throws UnsupportedEncodingException, TransformerException, ParserConfigurationException {
 DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
 DocumentBuilder builder = dbf.newDocumentBuilder();
 Document doc = builder.newDocument();

 // create the root element node
 Element root = doc.createElement("hibernate-mapping");
```

```
 doc.appendChild(root);

 Element entity = doc.createElement("class");
 root.appendChild(entity);

 entity.setAttribute("entity-name", dataObjectClass.getCanonicalName() == null ? dataObjectClass
      .getSimpleName() : dataObjectClass.getCanonicalName());
 entity.setAttribute("table", dataObjectClass.getSimpleName().toUpperCase());

 // Meta section
 Element meta = doc.createElement("meta");
 meta.setAttribute("attribute", "class-description");
 meta.insertBefore(doc.createTextNode("Hibernate mapping for the LODicity entity " +
      dataObjectClass.getSimpleName()), meta.getLastChild());
 entity.appendChild(meta);

 // Meta section
 // <id name="id" column="modelId" length="32" type="string">
 // <generator class="uuid.hex" />
 // </id>

 Element id = doc.createElement("id");
 id.setAttribute("name", "hibernateInternalId");
 id.setAttribute("type", "string");
 id.setAttribute("column", "HIBERNATEINTERNALID");
 Element generator = doc.createElement("generator");
 generator.setAttribute("class", "native");
 id.appendChild(generator);
 entity.appendChild(id);

 for (String attribute : getAttributes(dataObjectClass.getSimpleName())) {
  Element property = doc.createElement("property");
  property.setAttribute("name", attribute);
  property.setAttribute("column", attribute);
  // property.setAttribute("index", "IDX_" + dataObjectClass.getSimpleName().toUpperCase() + "
       _SHARED, IDX_" + dataObjectClass.getSimpleName() + "_" + attribute);
  String type = getHibernateType(getDataType(dataObjectClass, attribute));
  property.setAttribute("type", type);
  entity.appendChild(property);
 }

 // create text for the node
 // itemElement.insertBefore(doc.createTextNode("text"), itemElement.getLastChild());

 Transformer tf = TransformerFactory.newInstance().newTransformer();
 tf.setOutputProperty(OutputKeys.ENCODING, "UTF-8");
 tf.setOutputProperty(OutputKeys.INDENT, "yes");
 Writer out = new StringWriter();
 tf.transform(new DOMSource(doc), new StreamResult(out));

 LOGGER.info("Produced Hibernate type definition for {} \n{}", dataObjectClass.getSimpleName(),
      out.toString());
 return out.toString();
}

public final static boolean isTrivialType(String type) {
 switch (type) {
 case "Enum":
 case "URL":
 case "Float":
 case "String":
 case "Integer":
 case "Boolean":
  return true;
 default:
  return false;
 }
}

private static String getHibernateType(String dataType) {
 // TODO Should be warned or something
 // assert dataType != null : "dataType is null";
 if (dataType == null)
  return "string";

 switch (dataType) {
 case "Enum":
```

```
  case "URL":
  case "String":
   return "string";
  case "Float":
   return "float";
  case "Integer":
   return "int";
  case "Boolean":
   return "boolean";
  default:
   // Most likely, we have to handle complex datatypes here.

   return "string";

  // throw new RuntimeException("Cannot assign a hibernate type for data type " + dataType);
  }
 }

}
```

Listing C.2: Java Source Code of Schema.

C.3 DataObject

```
package com.github.heussd.lodicity.model;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collection;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import org.json.JSONArray;
import org.json.JSONObject;

public class DataObject extends HashMap<String, Object> implements Serializable {

 public static final String HIBERNATE_INTERNAL_ID = "hibernateInternalId";

 private final static List<String> SCHEMA_IGNORED_ATTRIBUTES = Arrays.asList("_class_",
      HIBERNATE_INTERNAL_ID);

 private static final long serialVersionUID = 1L;

 public String hibernateInternalId;

 @Override
 public Object put(String attribute, Object value) {
  return put(attribute, value, true);
 }

 public Object put(String attribute, Object value, boolean validate) {
  Object previousValue = super.get(attribute);

  if (previousValue != null && previousValue instanceof List) {
   if (!(value instanceof List)) {
    // Comfort feature: You are setting a scalar value where lists are expected. Add it on the fly
        .
    Object v = value;
    value = this.<ArrayList<Object>>get(attribute);
    ((List<String>) value).add((String) v);
   }
  }

  if (!SCHEMA_IGNORED_ATTRIBUTES.contains(attribute)) {
   if (validate)
    assert Schema.isValid(this, attribute, value);
  }

  super.put(attribute, value);
  return previousValue;
 }
```

```
public <T> T get(String attribute) {
 return get(attribute, true);
}

@SuppressWarnings("unchecked")
public <T> T get(String attribute, int index) {
 assert Schema.isListType(this, attribute) : attribute + " is no list type, thus, the n-th list
      element cannot be retrieved.";
 return (T) this.<List<Object>> get(attribute).get(index);
}

@SuppressWarnings("unchecked")
protected <T> T get(String attribute, boolean validate) {
 Object value = super.get(attribute);

 if (Schema.isListType(this.getClass(), attribute)) {
  // Value might already be converted
  if (value != null && value instanceof Collection) {
   return (T) value;
  }

  // Deserialize list from JSON object
  if (!(value instanceof ArrayList)) {
   String jsonString = (String) value;
   List<String> list = new ArrayList<String>();

   if (jsonString != null && !jsonString.equals("")) {
    // https://stackoverflow.com/questions/3395729/convert-json-array-to-normal-java-array
    JSONArray jsonArray = new JSONArray(jsonString);
    if (jsonArray != null) {
     int len = jsonArray.length();
     for (int i = 0; i < len; i++) {
      list.add(jsonArray.get(i).toString());
     }
    }
   }
   value = list;
  }
 }

 if (validate && !SCHEMA_IGNORED_ATTRIBUTES.contains(attribute)) {
  // System.out.println(attribute + " " + value);
  assert Schema.isValid(this, attribute, value);
 }

 // Make sure lists always return != null
 if (value == null && Schema.isListType(this, attribute)) {
  value = new ArrayList<>();
 }

 return (T) value;
}

public static String generateId(String... identifierStrings) {
 return String.join("_", Arrays.asList(identifierStrings));
}

public String toJson() {
 return JSONObject.valueToString(this);
}

public DataObject(Map<String, Object> map) {
 super(map);
}

// Needed for Hibernate

public void setHibernateInternalId(String id) {
 this.hibernateInternalId = id;
}

public String getHibernateInternalId() {
 return hibernateInternalId;
}

// Deprecated?
```

```
 public DataObject() {
  addClassAttribute();

  // /* Make sure lists are empty by default after DataObject initialization. */
  // for (String attribute : Schema.getAttributes(this)) {
  // if (!attribute.equals("_class_")) {
  // if (Schema.isListType(this, attribute)) {
  // this.setWithoutValidation(attribute, new ArrayList<>());
  // }
  // }
  // }
 }

 public void addClassAttribute() {
  this.put("_class_", this.getClass().getSimpleName());
 }

 public boolean isValid(String attribute, String value) {
  return Schema.isValid(this, attribute, value);
 }

 public void validate() {
  Schema.validate(this);
 }

 // Kept for compatiblity reasons
 public void set(String attribute, Object value) {
  put(attribute, value);
 }

 public <T> T getWithoutValidation(String attribute) {
  return this.<T> get(attribute);
 }

}
```

Listing C.3: Java Source Code of DataObject.

C.4 DataObject Interceptor

```
package com.github.heussd.lodicity.store;

import java.io.Serializable;
import java.util.HashMap;
import java.util.List;

import org.hibernate.EmptyInterceptor;
import org.hibernate.type.Type;
import org.json.JSONArray;

import com.github.heussd.lodicity.model.DataObject;
import com.github.heussd.lodicity.model.Schema;

public class DataObjectInterceptor extends EmptyInterceptor {
 private static final long serialVersionUID = 1L;

 /**
  * Make sure that {@link DataObject}-instances are correctly identified as {@link HashMap}
      instances. This is essential for using a dynamic model based on {@link DataObject}s.
  * The original idea is presented in <a href="https://forum.hibernate.org/viewtopic.php?f=1&t
      =992446">https://forum.hibernate.org/viewtopic.php?f=1&t=992446</a>.
  *
  * @param object
  * @return
  */
 @Override
 public String getEntityName(Object object) {
  if (object != null && object instanceof DataObject) {
   return object.getClass().getName();
  } else {
   return super.getEntityName(object);
  }
 }

 @Override
```

```
public boolean onSave(Object entity, Serializable id, Object[] state, String[] propertyNames,
     Type[] types) {
 state = embedListsAsJson(entity, state, propertyNames, types);
 state = embedDataObjectsAsJson(entity, state, propertyNames, types);
 return super.onSave(entity, id, state, propertyNames, types);
}

@Override
public boolean onFlushDirty(Object entity, Serializable id, Object[] currentState, Object[]
     previousState, String[] propertyNames, Type[] types) {
 currentState = embedListsAsJson(entity, currentState, propertyNames, types);
 currentState = embedDataObjectsAsJson(entity, currentState, propertyNames, types);
 return super.onFlushDirty(entity, id, currentState, previousState, propertyNames, types);
}

private Object[] embedListsAsJson(Object entity, Object[] states, String[] propertyNames, Type[]
      types) {
 if (entity instanceof DataObject) {
  for (int i = 0; i < propertyNames.length; i++) {
   String propertyName = propertyNames[i];

   if (Schema.isListType((DataObject) entity, propertyName)) {
    Object value = states[i];
    value = (String) new JSONArray((List<String>) value).toString();
    states[i] = value;
   }
  }
 }
 return states;
}

private Object[] embedDataObjectsAsJson(Object entity, Object[] states, String[] propertyNames,
     Type[] types) {
 if (entity instanceof DataObject) {
  for (int i = 0; i < propertyNames.length; i++) {
   Object value = states[i];

   if (value != null && value instanceof DataObject) {
    value = (String) ((DataObject) value).toJson();
    states[i] = value;
   }
  }
 }
 return states;
 }
}
```

Listing C.4: Java Source Code of DataObjectInterceptor.

C.5 Patch of the Third-Party SQLite Hibernate Dialect

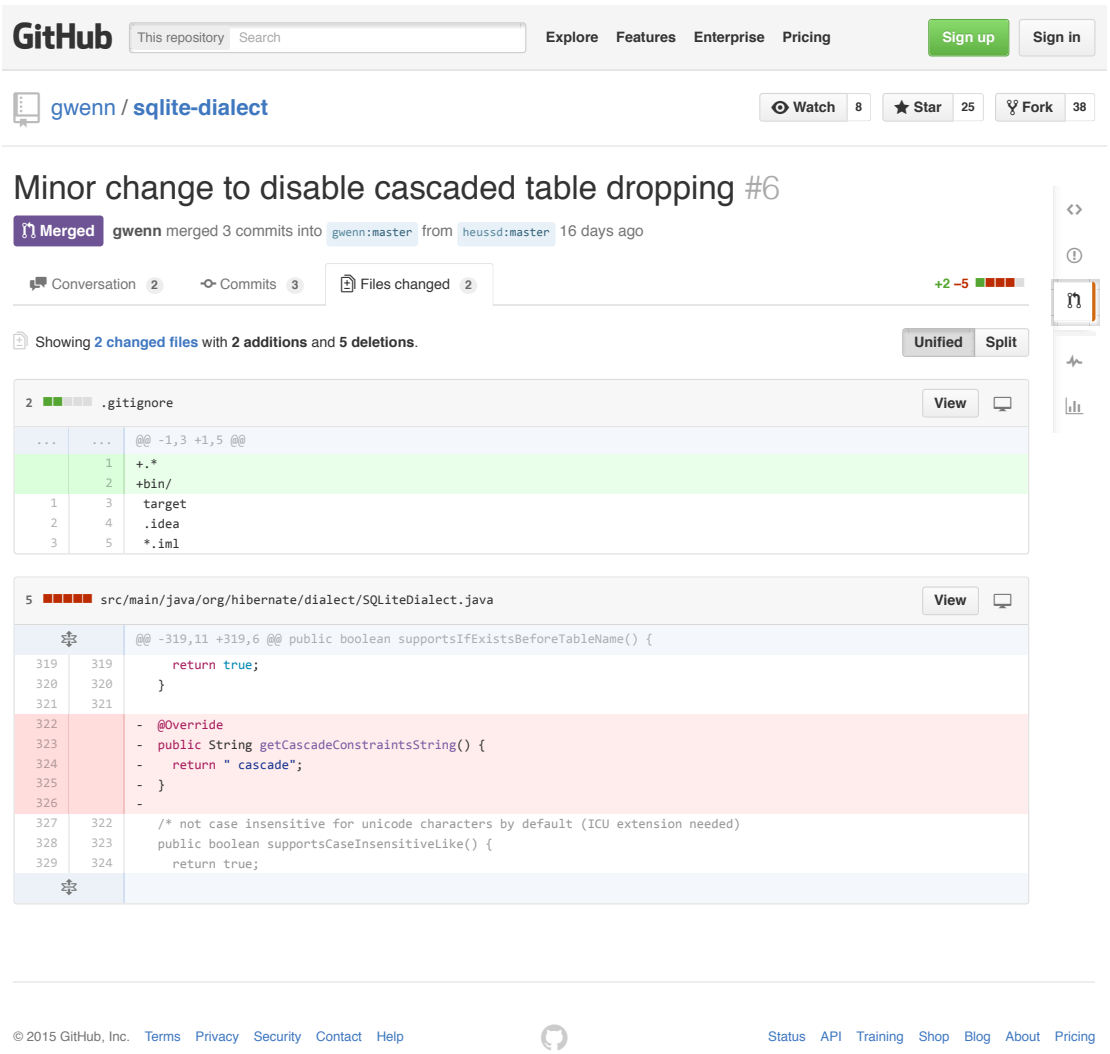


Figure C.1: SQLite Dialect GitHub Pull Request: Changed Files `https://github.com/gwenn/sqlite-dialect/pull/6/files`, last access on 2015-11-07

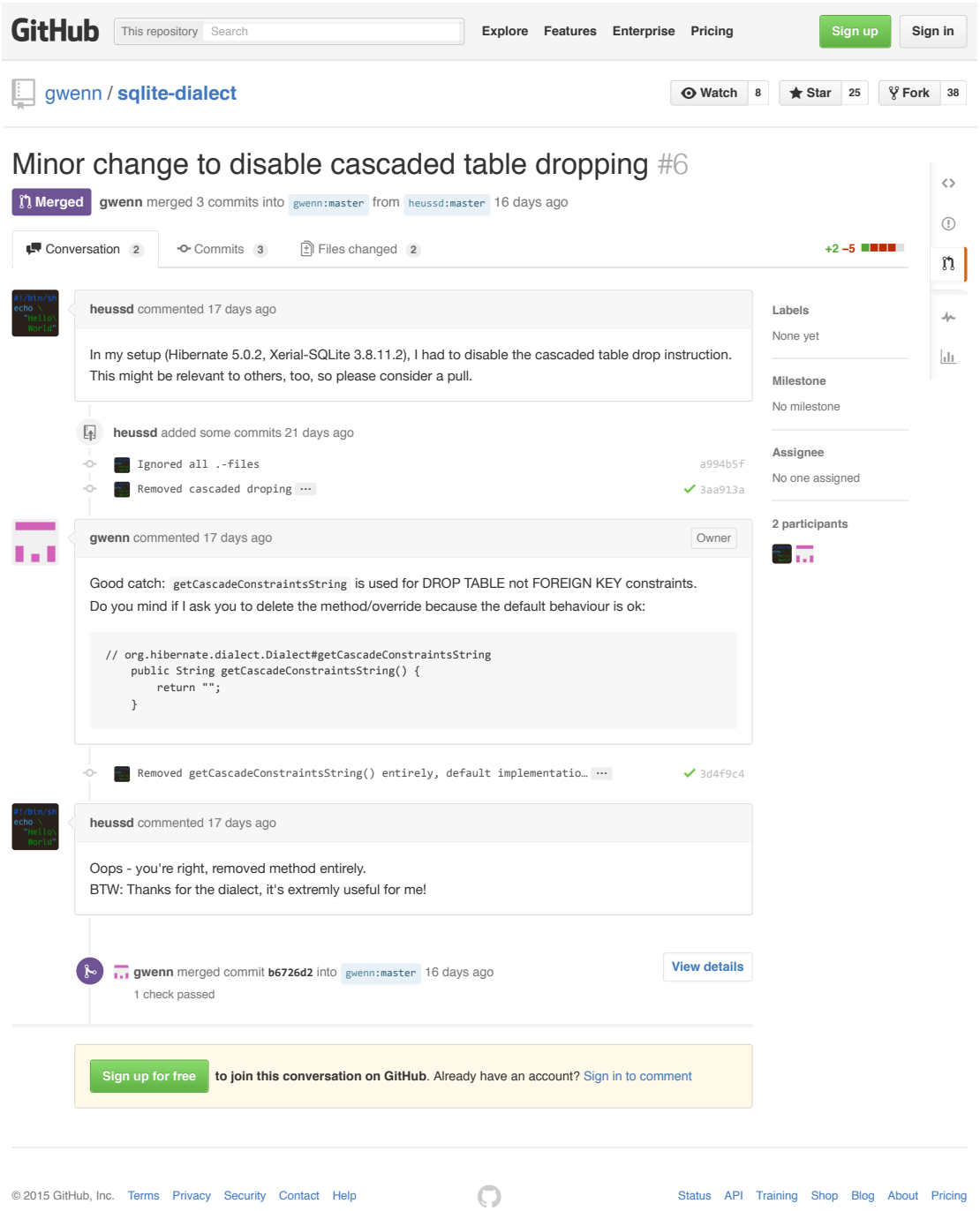


Figure C.2: SQLite Dialect GitHub Pull Request: Conversation `https://github.com/gwenn/sqlite-dialect/pull/6`, last access on 2015-11-07

Appendix D

Proof of Concept Appendix

D.1 ETL Process of the Original Mediaplatform

```
package de.h_da.mediaplatform.service;

import com.google.common.base.Stopwatch;
import com.google.common.collect.Lists;
import de.h_da.mediaplatform.dataobject.common.Person;
import de.h_da.mediaplatform.dataobject.common.Resource;
import de.h_da.mediaplatform.dataobject.common.Term;
import de.h_da.mediaplatform.datastore.common.ResourceStore;
import de.h_da.mediaplatform.datastore.common.TermStore;
import de.h_da.mediaplatform.datastore.common.openthesaurus.OpenThesaurusLoader;
import de.h_da.mediaplatform.datastore.common.searchengine.LuceneWriteEngine;
import de.h_da.mediaplatform.datastore.library.ResourceStoreTermStoreEnrichment;
import de.h_da.mediaplatform.datastore.library.gnd.GndLoader;
import de.h_da.mediaplatform.datastore.library.gnd.GndTermsInterpreter;
import de.h_da.mediaplatform.datastore.library.gnd.TermRanker;
import de.h_da.mediaplatform.datastore.library.hebis.HebisLoader;
import de.h_da.mediaplatform.datastore.museum.StaedelLoader;
import de.h_da.mediaplatform.guide.library.LibraryCacheBuilder;
import de.h_da.mediaplatform.util.Places.Places;
import org.apache.log4j.PropertyConfigurator;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import java.io.IOException;
import java.nio.file.DirectoryStream;
import java.nio.file.Files;
import java.nio.file.Path;
import java.util.List;
import java.util.concurrent.TimeUnit;

import static com.google.common.base.Preconditions.checkArgument;

/**
 * The LoadCoordinator starts and manages the ETL process. It should only be called when
 * an update to the databases is required.
 *
 * In order for the LoadCoordinator to work, the environment variable
 * MEDIAPLATFORM_HOME
 * must be set and the directory structure existing exactly as required.
 * It is recommended that only authorized persons be able to run and create
 * the Lucene Indices.
 *
 * The LoadCoordinator makes use of three directories within MEDIAPLATFOR_HOME:
 *   old/ new/ prod/
 * During the load, all prod/ data is moved to old/, the new indices
 * are created in new/ and afterwards moved into prod/. This is to ensure
 * that a backup of the lucene data is available in old/ at all times.
 *
 * @author Christian Greppmeier, Hochschule Darmstadt - University of Applied Sciences, 2013
 * @author Christian Broomfield, Hochschule Darmstadt - University of Applied Sciences, 2013
 */
public class LoadCoordinator {

  // Class variables (static):
  private static final Logger LOGGER = LoggerFactory.getLogger(LoadCoordinator.class);

  // Constructors:

  public LoadCoordinator() {
```

```
  PropertyConfigurator.configure(Places.PROPERTIES.properties("log4j_loader.properties"));
  assertThatNewIsEmpty();
}

// Class methods (static):

public static void main(final String[] args) {
  LoadCoordinator lc = new LoadCoordinator();

  try {
    lc.loadAll();
} catch (Exception e) {
 e.printStackTrace();
 LOGGER.error(e.getMessage());
 System.exit(-1);
}
}

// Public instance methods:

private void loadAll() {
  Stopwatch etlStopwatch = Stopwatch.createStarted();

  createDataStores();

  LOGGER.info("ETL finished: " +etlStopwatch.elapsed(TimeUnit.MINUTES) +" minutes, " + (
   etlStopwatch.elapsed(TimeUnit.SECONDS)%60) + " seconds");
}

private void assertThatNewIsEmpty() {
  List<Places> newDataStores = Lists.newArrayList(Places.DATA_STORE_NEW_PERSON,
                          Places.DATA_STORE_NEW_RESOURCE,
                          Places.DATA_STORE_NEW_TERM);

  for(Places newDataStore : newDataStores)
    assertDirectoryIsEmptyOrHasPlaceholder(newDataStore.directory());
}

private void assertDirectoryIsEmptyOrHasPlaceholder(final Path directory) {
  checkArgument(Files.isDirectory(directory), "The path does not lead to a directory, but to '%
   s' instead", directory);

  List<String> placeholders = Lists.newArrayList("readme.txt", ".DS_Store");
  try (DirectoryStream<Path> directoryStream = Files.newDirectoryStream(directory)) {
    for (Path candidate : directoryStream)
      failIfNotPlaceholder(candidate, placeholders);

    for (String placeholder : placeholders)
      Files.deleteIfExists(directory.resolve(placeholder));
  } catch (IOException e) {
    throw new RuntimeException("Could not access directory '" +directory +"'");
  }
}

private void failIfNotPlaceholder(final Path path, final List<String> placeholders) {
  String pathFileName = path.getFileName().toString();

  for (String placeholder : placeholders)
    if (pathFileName.equalsIgnoreCase(placeholder))
      return;

  String message = String.format("The directory %s is not empty! Clean it and try again.", path
   .getParent());
  throw new IllegalArgumentException(message);
}

private void createDataStores() {
LOGGER.info("Creating new indices in new/");

LuceneWriteEngine<Term> termStoreWriteEngine = LuceneWriteEngine.createTermStoreWriteEngine();
LuceneWriteEngine<Resource> resourceStoreWriteEngine = LuceneWriteEngine.
     createResourceStoreWriteEngine();
LuceneWriteEngine<Person> personStoreWriteEngine = LuceneWriteEngine.
     createPersonStoreWriteEngine();

OpenThesaurusLoader openThesaurusLoader = new OpenThesaurusLoader(termStoreWriteEngine);
openThesaurusLoader.load();
```

```
  HebisLoader hebisLoader = new HebisLoader(resourceStoreWriteEngine);
  hebisLoader.loadResources();

  GndLoader gndLoader = new GndLoader(termStoreWriteEngine, new TermRanker(termStoreWriteEngine,
       resourceStoreWriteEngine));
  gndLoader.addGndEntryInterpreter(new GndTermsInterpreter());
  gndLoader.load();

  StaedelLoader staedelLoader = new StaedelLoader(resourceStoreWriteEngine,
       personStoreWriteEngine);
  staedelLoader.loadAll();

  LibraryCacheBuilder libCacheBuilder = new LibraryCacheBuilder(resourceStoreWriteEngine,
       termStoreWriteEngine);

    ResourceStoreTermStoreEnrichment resourceStoreTermStoreEnrichment = new
     ResourceStoreTermStoreEnrichment(termStoreWriteEngine, resourceStoreWriteEngine,
     libCacheBuilder);
    resourceStoreTermStoreEnrichment.enrich();

    termStoreWriteEngine.close();
    resourceStoreWriteEngine.close();
    personStoreWriteEngine.close();

    libCacheBuilder = new LibraryCacheBuilder(new ResourceStore(Places.DATA_STORE_NEW_RESOURCE.
     directory()), new TermStore(Places.DATA_STORE_NEW_TERM.directory()));
    libCacheBuilder.buildFullIndex();

//  new LibraryCacheBuilder(new ResourceStore(Places.DATA_STORE_RESOURCE.directory()), new
     TermStore(Places.DATA_STORE_TERM.directory())).buildFullIndex();

  }
}
```

Listing D.1: Java Source Code of the Original Mediaplatform ETL Process

D.2 ETL Process of the LODicity-enhanced Mediaplatform

```
package de.h_da.mediaplatform.service;

import static com.google.common.base.Preconditions.checkArgument;

import java.io.IOException;
import java.nio.file.DirectoryStream;
import java.nio.file.Files;
import java.nio.file.Path;
import java.util.List;
import java.util.concurrent.TimeUnit;

import org.apache.log4j.PropertyConfigurator;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import com.github.heussd.lodicity.data.LoadManager;
import com.github.heussd.lodicity.store.Warehouse;
import com.google.common.base.Stopwatch;
import com.google.common.collect.Lists;

import de.h_da.mediaplatform.dataobject.common.Person;
import de.h_da.mediaplatform.dataobject.common.Resource;
import de.h_da.mediaplatform.dataobject.common.Term;
import de.h_da.mediaplatform.datastore.common.ResourceStore;
import de.h_da.mediaplatform.datastore.common.openthesaurus.OpenThesaurusLoader;
import de.h_da.mediaplatform.datastore.common.searchengine.LuceneWriteEngine;
import de.h_da.mediaplatform.datastore.library.ResourceStoreTermStoreEnrichment;
import de.h_da.mediaplatform.datastore.library.gnd.GndLoader;
import de.h_da.mediaplatform.datastore.library.gnd.TermRanker;
import de.h_da.mediaplatform.datastore.library.hebis.HebisLoader;
import de.h_da.mediaplatform.datastore.museum.StaedelLoader;
import de.h_da.mediaplatform.guide.library.LibraryCacheBuilder;
import de.h_da.mediaplatform.util.Places.Places;

/**
 * The LoadCoordinator starts and manages the ETL process. It should only be called when an
```

```
     update to the databases is required.
 *
 * In order for the LoadCoordinator to work, the environment variable MEDIAPLATFORM_HOME must be
     set and the directory structure existing exactly as required. It is recommended
 * that only authorized persons be able to run and create the Lucene Indices.
 *
 * The LoadCoordinator makes use of three directories within MEDIAPLATFOR_HOME: old/ new/ prod/
     During the load, all prod/ data is moved to old/, the new indices are created in
 * new/ and afterwards moved into prod/. This is to ensure that a backup of the lucene data is
     available in old/ at all times.
 *
 * @author Christian Greppmeier, Hochschule Darmstadt - University of Applied Sciences, 2013
 * @author Christian Broomfield, Hochschule Darmstadt - University of Applied Sciences, 2013
 */
public class LoadCoordinator {

 // Class variables (static):
 private static final Logger LOGGER = LoggerFactory.getLogger(LoadCoordinator.class);
 private Warehouse warehouse = new Warehouse(true, Resource.class, Person.class, Term.class);

 // Constructors:

 public LoadCoordinator() {
  PropertyConfigurator.configure(Places.PROPERTIES.properties("log4j_loader.properties"));
  assertThatNewIsEmpty();
 }

 // Class methods (static):

 public static void main(final String[] args) {

  LoadCoordinator lc = new LoadCoordinator();

  try {
   lc.loadAll();
  } catch (Exception e) {
   e.printStackTrace();
   LOGGER.error(e.getMessage());
   System.exit(-1);
  }
 }

 // Public instance methods:

 private void loadAll() {
  Stopwatch etlStopwatch = Stopwatch.createStarted();

  createDataStores();

  LOGGER.info("ETL finished: " + etlStopwatch.elapsed(TimeUnit.MINUTES) + " minutes, " + (
      etlStopwatch.elapsed(TimeUnit.SECONDS) % 60) + " seconds");
 }

 private void assertThatNewIsEmpty() {
  List<Places> newDataStores = Lists.newArrayList(Places.DATA_STORE_NEW_PERSON, Places.
      DATA_STORE_NEW_RESOURCE, Places.DATA_STORE_NEW_TERM);

  for (Places newDataStore : newDataStores)
   assertDirectoryIsEmptyOrHasPlaceholder(newDataStore.directory());
 }

 private void assertDirectoryIsEmptyOrHasPlaceholder(final Path directory) {
  checkArgument(Files.isDirectory(directory), "The path does not lead to a directory, but to '%s'
       instead", directory);

  List<String> placeholders = Lists.newArrayList("readme.txt", ".DS_Store");
  try (DirectoryStream<Path> directoryStream = Files.newDirectoryStream(directory)) {
   for (Path candidate : directoryStream)
    failIfNotPlaceholder(candidate, placeholders);

   for (String placeholder : placeholders)
    Files.deleteIfExists(directory.resolve(placeholder));
  } catch (IOException e) {
   throw new RuntimeException("Could not access directory '" + directory + "'");
  }
 }
```

```
 private void failIfNotPlaceholder(final Path path, final List<String> placeholders) {
  String pathFileName = path.getFileName().toString();

  for (String placeholder : placeholders)
   if (pathFileName.equalsIgnoreCase(placeholder))
    return;

  String message = String.format("The directory %s is not empty! Clean it and try again.", path.
      getParent());
  throw new IllegalArgumentException(message);
 }

 private void createDataStores() {
  LOGGER.info("Creating new indices in new/");

  // Integration
  LoadManager loadManager = new LoadManager(warehouse);
  loadManager.register(new OpenThesaurusLoader(Places.DATA_SOURCES_OPENTHESAURUS.file("
      openthesaurus.txt")));
  loadManager.register(new HebisLoader());
  loadManager.register(new GndLoader(), new TermRanker());
  loadManager.register(new StaedelLoader());
  loadManager.loadAll();

  // Customization / Delivery
  LuceneWriteEngine<Resource> resourceStoreWriteEngine = LuceneWriteEngine.
      createResourceStoreWriteEngine();

  warehouse.all(Resource.class).forEach(resource -> resourceStoreWriteEngine.add((Resource)
      resource));
  resourceStoreWriteEngine.commit();

  LibraryCacheBuilder libCacheBuilder = new LibraryCacheBuilder(resourceStoreWriteEngine,
      warehouse);

  ResourceStoreTermStoreEnrichment resourceStoreTermStoreEnrichment = new
      ResourceStoreTermStoreEnrichment(warehouse, resourceStoreWriteEngine,
    libCacheBuilder);
  resourceStoreTermStoreEnrichment.enrich();

  LuceneWriteEngine<Person> personStoreWriteEngine = LuceneWriteEngine.
      createPersonStoreWriteEngine();
  warehouse.all(Person.class).forEach(person -> personStoreWriteEngine.add((Person) person));

  personStoreWriteEngine.commit();
  resourceStoreWriteEngine.commit();

  libCacheBuilder = new LibraryCacheBuilder(new ResourceStore(Places.DATA_STORE_NEW_RESOURCE.
      directory()), warehouse);
  libCacheBuilder.buildFullIndex();

 }
}
```

Listing D.2: Java Source Code of the LODicity-Integrated Mediaplatform ETL Process

D.3 CLOC Output Original Mediaplatform

language	filename	blank	comment	code
XML	webapp/WEB-INF/lib/export.xml	0	0	11104
Java	java/de/h_da/mediaplatform/datastore/library/hebis/HebisLoader.java	214	239	769
Java	java/de/h_da/mediaplatform/datastore/museum/StaedelLoader.java	151	172	540
Java	java/de/h_da/mediaplatform/datastore/common/searchengine/LuceneSearchEngine.java	84	115	333
Java	java/de/h_da/mediaplatform/dataobject/common/DataObject.java	53	198	294
Java	java/de/h_da/mediaplatform/guide/library/LibraryGuide.java	66	33	269
Java	java/de/h_da/mediaplatform/datastore/museum/AdlibLoader.java	59	58	225
Java	java/de/h_da/mediaplatform/dataobject/common/Resource.java	54	57	222
Java	java/de/h_da/mediaplatform/dataobject/common/Schema.java	65	123	222
Java	java/de/h_da/mediaplatform/datastore/museum/StaedelCMSLoader.java	43	77	219
Java	java/de/h_da/mediaplatform/util/Places/Places.java	80	147	218
Java	java/de/h_da/mediaplatform/guide/library/LibraryCacheBuilder.java	48	33	201
Java	java/de/h_da/mediaplatform/service/museum/MuseumService.java	45	18	181
Java	java/de/h_da/mediaplatform/datastore/library/gnd/GndEntryReader.java	51	41	171
Java	java/de/h_da/mediaplatform/datastore/common/searchengine/LuceneWriteEngine.java	32	4	136
Java	java/de/h_da/mediaplatform/service/MediaService.java	39	33	130
Java	java/de/h_da/mediaplatform/guide/library/LibraryCache.java	32	22	123
Java	java/de/h_da/mediaplatform/datastore/library/gnd/GndEntry.java	36	34	121
Java	java/de/h_da/mediaplatform/datastore/common/ResourceStore.java	41	51	112
Java	java/de/h_da/mediaplatform/util/XmlReader.java	29	90	110
Java	java/de/h_da/mediaplatform/guide/museum/MuseumGuide.java	33	66	102
Java	java/de/h_da/mediaplatform/service/LoadCoordinator.java	34	24	100
Java	java/de/h_da/mediaplatform/datastore/library/gnd/GndLoader.java	22	7	100
JSON	resources/configuration/LibraryStartTopics.json	0	0	94
Java	java/de/h_da/mediaplatform/guide/library/GNDNode.java	22	23	91
Java	java/de/h_da/mediaplatform/service/library/LibraryService.java	25	18	90
Java	java/de/h_da/mediaplatform/datastore/library/ResourceStoreTermStoreEnrichment.java	33	7	86
Java	java/de/h_da/mediaplatform/datastore/library/gnd/GNDSCTaxonomy.java	20	14	77
Java	java/de/h_da/mediaplatform/datastore/library/gnd/TermRanker.java	26	11	68
Java	java/de/h_da/mediaplatform/datastore/common/TermStore.java	21	76	62
Java	java/de/h_da/mediaplatform/datastore/common/searchengine/LuceneSearchParams.java	15	3	60
Java	java/de/h_da/mediaplatform/datastore/common/openthesaurus/OpenThesaurusLoader.java	17	16	56
Java	java/de/h_da/mediaplatform/datastore/common/file/FileService.java	27	43	54
Java	java/de/h_da/mediaplatform/util/CsvReader.java	19	40	47
Java	java/de/h_da/mediaplatform/dataobject/common/container/SearchResult.java	13	8	46
Java	java/de/h_da/mediaplatform/dataobject/common/Term.java	21	18	45
Java	java/de/h_da/mediaplatform/guide/library/LibraryCacheResult.java	9	6	38
Java	java/de/h_da/mediaplatform/util/Converter.java	9	20	37
Java	java/de/h_da/mediaplatform/guide/library/AbstractLibraryCache.java	14	7	36
Java	java/de/h_da/mediaplatform/dataobject/common/Person.java	18	18	32
XML	webapp/WEB-INF/web.xml	13	0	31
Java	java/de/h_da/mediaplatform/datastore/library/hebis/SubjectGroupCsvLoader.java	8	18	31
Java	java/de/h_da/mediaplatform/dataobject/common/Result.java	7	6	30
Java	java/de/h_da/mediaplatform/dataobject/common/Topic.java	8	9	30
JSON	resources/configuration/MuseumStartTopics.json	0	0	27
Java	java/de/h_da/mediaplatform/util/Places/FilePropertiesLoader.java	10	14	25
Java	java/de/h_da/mediaplatform/datastore/library/gnd/GndTermsInterpreter.java	4	5	16
Java	java/de/h_da/mediaplatform/datastore/common/PersonStore.java	14	13	13
Java	java/de/h_da/mediaplatform/datastore/library/gnd/GndEntryType.java	4	5	11
Java	java/de/h_da/mediaplatform/datastore/common/searchengine/LuceneDefaultSimilarity.java	2	3	9
Java	java/de/h_da/mediaplatform/util/Places/PropertiesLoader.java	2	12	7
Java	java/de/h_da/mediaplatform/datastore/library/gnd/GndEntryInterpreter.java	2	14	6
JSP	webapp/index.jsp	0	0	5
Java	java/de/h_da/mediaplatform/dataobject/common/Request.java	2	6	3
Java	java/de/h_da/mediaplatform/ClassTemplate.java	9	10	3
XML	resources/data_sources/museum/Staedel/Staedel.Adlib.xml	0	0	1
Total		1705	2085	17269

D.4 CLOC Output LODicity-enhanced Mediaplatform

language	filename	blank	comment	code
XML	resources/data_sources/museum/Staedel/Staedel.CMS.xml	0	0	11104
Java	java/de/h_da/mediaplatform/datastore/library/hebis/HebisLoader.java	213	239	772
Java	java/de/h_da/mediaplatform/datastore/museum/StaedelLoader.java	152	171	544
Java	java/de/h_da/mediaplatform/datastore/common/searchengine/LuceneSearchEngine.java	84	116	333
Java	java/de/h_da/mediaplatform/guide/library/LibraryGuide.java	66	33	269
Java	java/de/h_da/mediaplatform/dataobject/common/Resource.java	57	55	227
Java	java/de/h_da/mediaplatform/datastore/museum/AdlibLoader.java	59	58	225
Java	java/de/h_da/mediaplatform/datastore/museum/StaedelCMSLoader.java	43	77	219
Java	java/de/h_da/mediaplatform/util/Places/Places.java	80	147	218
Java	java/de/h_da/mediaplatform/dataobject/common/EntityFactory.java	29	65	206
Java	java/de/h_da/mediaplatform/guide/library/LibraryCacheBuilder.java	51	27	195
Java	java/de/h_da/mediaplatform/service/museum/MuseumService.java	45	18	179
Java	java/de/h_da/mediaplatform/datastore/library/gnd/GndEntryReader.java	51	41	171
Java	java/de/h_da/mediaplatform/datastore/common/searchengine/LuceneWriteEngine.java	33	4	138
Java	java/de/h_da/mediaplatform/service/MediaService.java	40	33	130
Java	java/de/h_da/mediaplatform/guide/library/LibraryCache.java	32	22	123
Java	java/de/h_da/mediaplatform/datastore/library/gnd/GndEntry.java	38	34	121
Java	java/de/h_da/mediaplatform/datastore/common/ResourceStore.java	41	51	112
Java	java/de/h_da/mediaplatform/util/XmlReader.java	29	90	110
Java	java/de/h_da/mediaplatform/datastore/library/gnd/GndLoader.java	25	6	104
Java	java/de/h_da/mediaplatform/guide/museum/MuseumGuide.java	33	65	99
Java	java/de/h_da/mediaplatform/service/LoadCoordinator.java	35	18	98
JSON	resources/configuration/LibraryStartTopics.json	0	0	94
Java	java/de/h_da/mediaplatform/guide/library/GNDNode.java	22	23	91
Java	java/de/h_da/mediaplatform/datastore/library/ResourceStoreTermStoreEnrichment.java	30	7	87
Java	java/de/h_da/mediaplatform/service/library/LibraryService.java	26	18	86
Java	java/de/h_da/mediaplatform/datastore/library/gnd/GNDSCTaxonomy.java	20	14	77
Java	java/de/h_da/mediaplatform/datastore/common/searchengine/LuceneSearchParams.java	15	3	60
Java	java/de/h_da/mediaplatform/datastore/common/openthesaurus/OpenThesaurusLoader.java	20	16	60
Java	java/de/h_da/mediaplatform/datastore/common/file/FileService.java	29	43	55
Java	java/de/h_da/mediaplatform/dataobject/common/Term.java	24	18	50
Java	java/de/h_da/mediaplatform/util/CsvReader.java	19	40	47
Java	java/de/h_da/mediaplatform/dataobject/common/container/SearchResult.java	13	8	46
Java	java/de/h_da/mediaplatform/datastore/library/gnd/TermRanker.java	18	11	46
Java	java/de/h_da/mediaplatform/guide/library/LibraryCacheResult.java	9	6	38
Java	java/de/h_da/mediaplatform/util/Converter.java	9	20	37
Java	java/de/h_da/mediaplatform/dataobject/common/Person.java	20	18	37
Java	java/de/h_da/mediaplatform/guide/library/AbstractLibraryCache.java	14	7	36
Java	java/de/h_da/mediaplatform/dataobject/common/Topic.java	10	9	31
XML	webapp/WEB-INF/web.xml	13	0	31
Java	java/de/h_da/mediaplatform/datastore/library/hebis/SubjectGroupCsvLoader.java	8	18	31
Java	java/de/h_da/mediaplatform/dataobject/common/Result.java	8	6	31
JSON	resources/configuration/MuseumStartTopics.json	0	0	27
Java	java/de/h_da/mediaplatform/util/Places/FilePropertiesLoader.java	10	14	25
Java	java/de/h_da/mediaplatform/datastore/library/gnd/GndTermsInterpreter.java	5	5	16
Java	java/de/h_da/mediaplatform/datastore/common/PersonStore.java	14	13	13
Java	java/de/h_da/mediaplatform/datastore/library/gnd/GndEntryType.java	4	5	11
Java	java/de/h_da/mediaplatform/datastore/common/searchengine/LuceneDefaultSimilarity.java	2	3	9
Java	java/de/h_da/mediaplatform/util/Places/PropertiesLoader.java	2	12	7
Java	java/de/h_da/mediaplatform/datastore/library/gnd/GndEntryInterpreter.java	2	14	6
JSP	webapp/index.jsp	0	0	5
Java	java/de/h_da/mediaplatform/dataobject/common/Request.java	3	6	4
Java	java/de/h_da/mediaplatform/ClassTemplate.java	9	10	3
XML	resources/data_sources/museum/Staedel/Staedel.Adlib.xml	0	0	1
Total		1614	1737	16895

D.5 Surefire Test Reports Original Mediaplatform

D.5.1 Summary

Tests	Errors	Failures	Skipped	Success Rate	Time
87	0	0	2	97.701%	3.596

D.5.2 Package List

Package	Tests	Errors	Failures	Skipped	Success Rate	Time
de.h_da.mediaplatform.util	2	0	0	0	100%	0
de.h_da.mediaplatform.dataobject.common	15	0	0	0	100%	1.268
de.h_da.mediaplatform	1	0	0	1	0%	0
de.h_da.mediaplatform.datastore.common	20	0	0	0	100%	0.464
de.h_da.mediaplatform.datastore.museum	6	0	0	0	100%	0.377
de.h_da.mediaplatform.service	42	0	0	1	97.619%	1.065
de.h_da.mediaplatform.guide.museum	1	0	0	0	100%	0.422

D.6 Surefire Test Report Mediaplatform + LODicity

D.6.1 Summary

Tests	Errors	Failures	Skipped	Success Rate	Time
87	0	0	2	97.701%	2.848

D.6.2 Package List

Package	Tests	Errors	Failures	Skipped	Success Rate	Time
de.h_da.mediaplatform.util	2	0	0	0	100%	0.001
de.h_da.mediaplatform.dataobject.common	15	0	0	0	100%	0.77
de.h_da.mediaplatform	1	0	0	1	0%	0
de.h_da.mediaplatform.datastore.common	20	0	0	0	100%	0.343
de.h_da.mediaplatform.datastore.museum	6	0	0	0	100%	0.263
de.h_da.mediaplatform.service	42	0	0	1	97.619%	1.453
de.h_da.mediaplatform.guide.museum	1	0	0	0	100%	0.018