🏆 *reviewed paper*

Data-based Collaboration on a Grand Scale

Markus Mayr, Paolo Fogliaroni

(Dipl. Ing. Markus Mayr, TU Vienna Department of Geodesy and Geoinformation, Gusshausstraße 27-29 1040 Vienna, mayr@geoinfo.tuwien.ac.at)
(Dr.-Ing. Paolo Fogliaroni, TU Vienna Department of Geodesy and Geoinformation, Gusshausstraße 27-29 1040 Vienna, paolo@geoinfo.tuwien.ac.at)

1 ABSTRACT

Modern spatial planning strongly relies on computer systems such as Computer-Aided Design tools (CAD) and Geographic Information Systems (GIS). These, in turn, depend upon Database Management Systems: complex computer systems designed to optimize data storage and retrieval. In this paper we try to sketch a short survey of current DBMS technologies for the non-expert by overviewing their history, targets, strengths, and weaknesses. The goal is to make the spatial planning community more aware of the present and developing technologies such that future projects started can take advantage of the most suitable technology.

2 INTRODUCTION

Over the years, methods and techniques employed in spatial planning evolved from good old ink and paper to complex computer systems that allow, for example, for geo-location, automatic computation of light exposure, traffic analysis, and so on.

Modern spatial planning strongly relies on continuously more advanced computer systems such as Computer-Aided Design tools (CAD) and Geographic Information Systems (GIS). Moreover, thanks to technological improvements, the amount of data that these computer systems are able to handle and make available to spatial planners increases quickly and steadily. Nowadays, a planner can easily access information about the population density of an area, the distribution of shops and roads, and average rainfall levels. This abundance of information is helping the planner to make better-informed decisions.

Present trends in the planning field also started to incorporate the idea of public participation and volunteered information (M.Foth et Al. 2009). In such scenarios, laymen are involved in the planning process as they can provide the planner with detailed local information that is typically not present in official data bases (Goodchild, M. F. 2007 and R. Sieber 2006). They can also provide suggestions or expectations about the future development or requalification of a certain area (A. Poplin, 2012).

Thus, on their most basic level, both, present and future spatial planning activities strongly rely on so-called Database Management Systems (DBMS). These are complex computer systems designed to optimize data storage and retrieval.

The scope of this paper is to provide the spatial planning community with an overview of database technologies and to discuss their strengths and weaknesses. The aim is to make the community more aware of present and developing technologies so that future projects started by the community can take advantage of the most suitable technologies, rather than the de-facto standards.

The paper starts in Section 3, sketching the historical evolution and the main principles and motivations underlying the de-facto standard Relational Database Management System (RDBMS). Later, in Section 4, the focus is shifted on more modern demands driven by social and collective projects. We discuss what properties a database should have to support such projects and which particular technology affords for them. At the end of this section we also discuss how the majority of present companies (including big internet colossuses) partially insist in utilizing some parts of RDBMS even though they have been proven to be inadequate to address modern requirements. Section 5, discusses a definitely edge-cutting topic: the semantic web and linked data. These are core topics when dealing with collaborative and public participatory projects. In parallel we discuss graph databases as the more straightforward technology to address the inherent challenges. Finally, Section 6 concludes the paper.

3 RDBMS

In 1970, E.F. Codd introduced the formal foundation for relational database management systems (RDBMS) Codd (1970) . His new model superseded the first and second generation of database management systems

ISBN: 978-3-9503110-8-2 (CD-ROM); ISBN: 978-3-9503110-9-9 (Print)
Editors: M. SCHRENK, V. V. POPOVICH, P. ZEILE, P. ELISEI, C. BEYER

that were based on flat files and hierarchical structures (“Existing noninferential, formatted data systems provide users with tree-structured files or slightly more general network models of the data.”) . This first generations managed to cope with increasing volumes of data and kept them separated from parts of software that was not central to the data storage. But, as Codd (1970) states, “future users of large data banks must be protected from having to know how the data is organized in the machine (the internal representation)“.

This and a couple of additional concepts that increased the reliability of relational systems together with their adoption by big companies like IBM ensured the distribution of this new kind of database system. Over the last decades, RDBMS were heavily optimized, became multi-user compatible and ensured consistency. They were perfectly suited for the monolithic structure of computer systems at that time and also did not stand in contrast of a more hierarchical way of thinking about data. For many applications they are still the best choice.

3.1 Architecture of RDBMS

The typical architecture of a RDBMS is one to multiple tables that are linked with one another to produce results to queries provided in the so-called SQL query language (see figure 1).

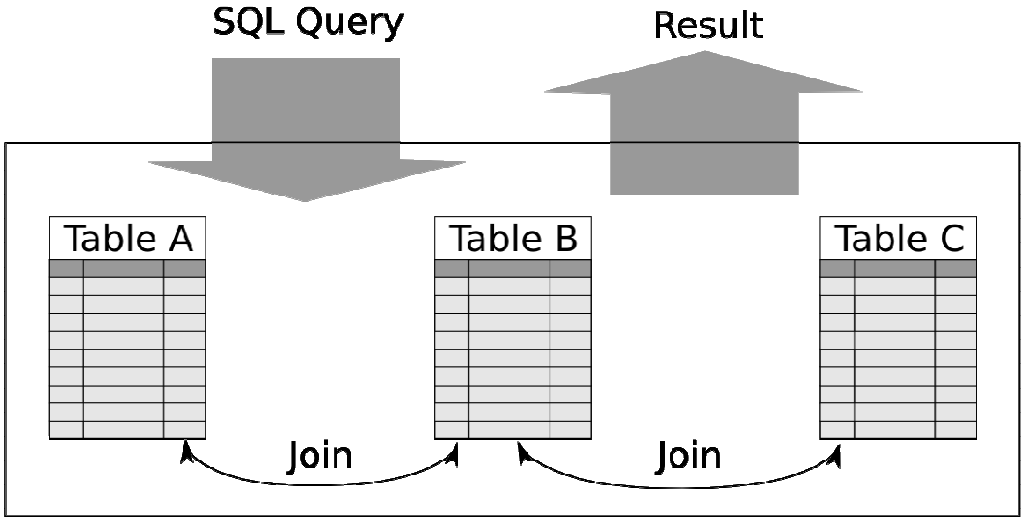

Fig. 1: Requesting data from a traditional RDBMS

These tables are very rigid and their structure should, once defined, not be changed (e.g. removing a column). Doing so could break existing constraints or queries. This means that the data structure which is developed once for an RDBMS is fixed. Later changes are costly.

It is very common that one SQL query requests information that is distributed among multiple tables. These tables have to be joined to query all data to answer the query.

3.1.1 ACID

A well known term that describes properties of a RDBMS is “ACID”. This stands for [A]tomicity, [C]onsistency,[I]solation and [D]urability. Its origins lie in a paper from Gray and others (1981) and are more concretized by Haerder and Reuter (1983) .

- **Atomicity** means that each operation or collection of operations on the database is performed as a whole. If this is not possible, this operation is not performed at all.
- **Consistency** ensures that one operation is executed on a database with a valid status and leaves a valid database behind. Valid in this sense means that the database satisfies all constraints and limitations set up by the database engineer.
- **Isolation** is an important property for multi-user operation. Simply put, it ensures that operations don't interfere with one another.
- **Durability** makes data in the system permanent. An unforeseen event should not result in the loss of any data. There are mechanisms in place that ensure the storage of data even in the event of power loss during a save procedure.

These four properties are the cornerstones to enabling a transaction-based system: Everything that was called an operation above can also be called a transaction.

3.1.2 Transactions

Transactions are the key for multi-user access to a RDBMS and can either read from the database, write to the database, or both. Depending on the kind of transaction, the portion of the database that is accessed during its execution is locked to all other users.

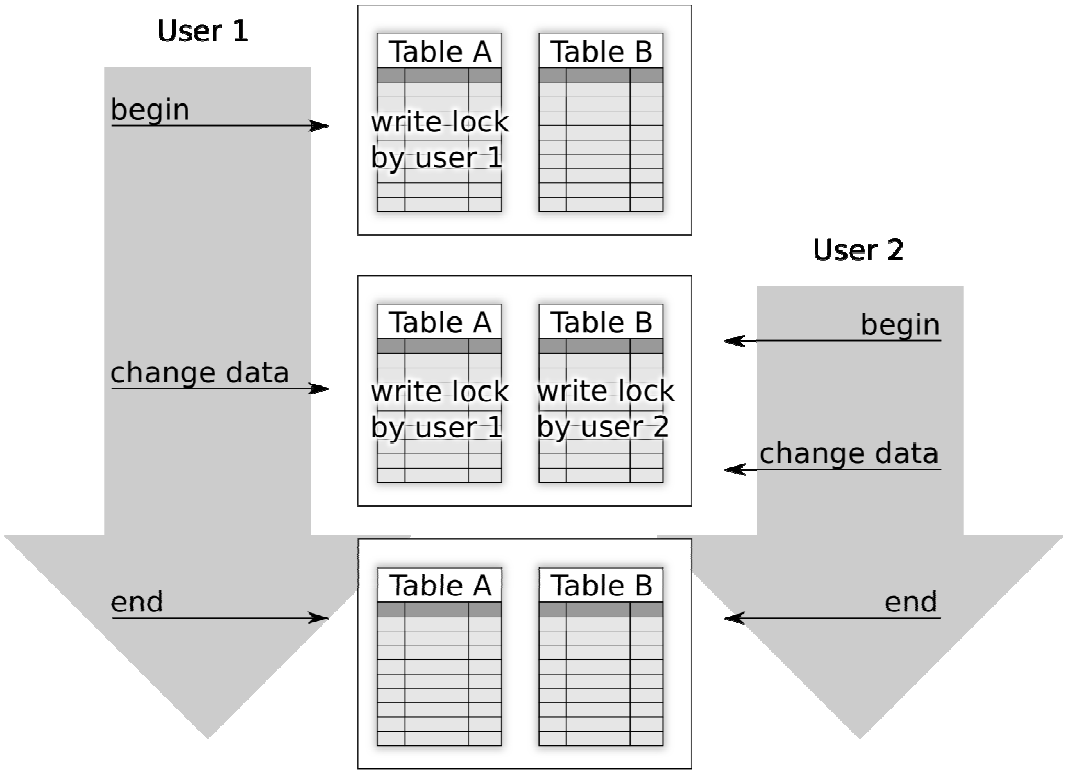

Fig. 2: The transactional locking of a RDBMS

Figure 2 shows a situation where user 1 and user 2 are locking a single table because they want to write to it. There is no problem here, because user 1 is not interested in table B and user 2 is not interested in table A. Each of these two transactions can be performed without interference. If user 2 would want to write to table A while it is locked by user 1, this transaction would have to wait.

In reality there exist multiple locking levels and different strategies to cope with conflicts, but the information presented here suffices to draw a general picture. More information about the transactional locking system is provided by (Kemper & Eickler 2011).

3.2 Problems with traditional RDBMS

There are two central problems that arise when using a traditional RDBMS with many users:

(1) When there is a huge amount of users, they will produce a great number of locks which interfere with other users' access to the database. Also, the locks are managed by a "locking manager" which is a resource intensive process.

(2) As soon as there is either a very high amount of data in the database, it becomes necessary to split it up to be stored on multiple single computers. A relational database system is not built for this. Since its structure depends on relations between tables which have to be checked when performing most queries, the separation of data is reduced to nearly nothing.

Imagine a service developed to support civic participation. It is very successful and is not allowed to have any down time. Otherwise, the image of the system in the public's view would be damaged. It is very difficult to scale a system as such up with an RDBMS to provide more capacity or to increase response times by moving part of the data to servers nearer to the clients while keeping the original structure as it is.

ISBN: 978-3-9503110-8-2 (CD-ROM); ISBN: 978-3-9503110-9-9 (Print)
Editors: M. SCHRENK, V. V. POPOVICH, P. ZEILE, P. ELISEI, C. BEYER

Since technology as a tool has to adapt to changing circumstances and not the other way around, we need two solutions: a more effective way to deal with database locks and more distribute-oriented database management systems.

4 SYSTEMS FOR COLLABORATION

Different database systems that strive for overcoming the deficiencies discussed in this chapter have been developed. Cattell (2011) gives an overview of many of these NoSQL ("not only SQL") databases and describes their peculiarities. There also exist undertakings to adapt existing RDBMS according to some of these concepts (e.g.ScaleBase_Inc. (2014)).

4.1 Theory

First, we will look at some of the theoretical concepts that are beneficial to supporting large scaled collaborative data-based work.

4.1.1 CAP theorem

The CAP theorem was proven in 2002 by Gilbert and Lynch (2002) and describes the fact, that out of three main properties of a distributed database system ([C]onsistency, [A]vailability and [P]artition Tolerance), only two can be achieved at the same time (see figure 3).

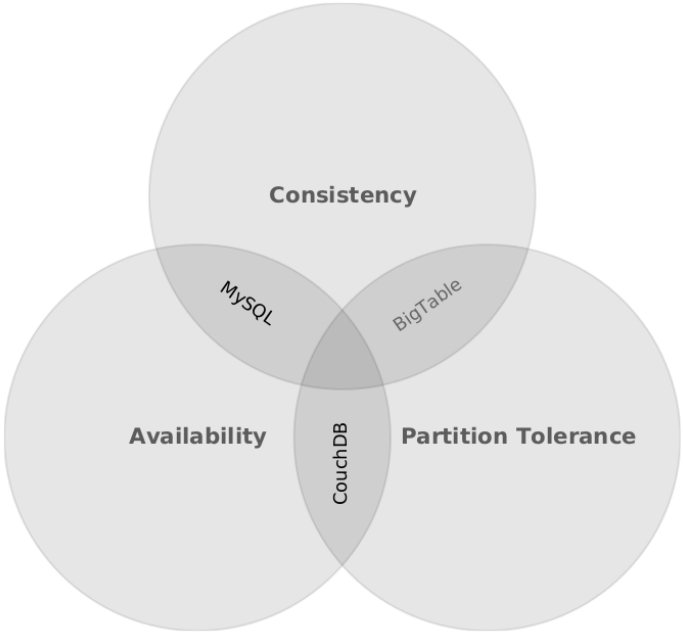

Fig. 3: The CAP Theorem and its relation to three different database-systems

- **Consistency** describes the status that every single instance of a distributed database reflects the same state. There is no difference in data on each of these systems. This is a challenge when there are lots of changes to the database. In a consistent system, all these changes have to be transferred to the other instances before these are allowed to operate again. Already in 1979, the problem was given a lot of thought by Lindsay et al. (1979).
- **Availability** simply means that the system has to be able to respond to requests all the time.
- **Partition Tolerance** assures that the system is not affected by the loss of one single instance of the distributed database.

Traditional RDBMS, like the widely used "MySQL" database, can provide consistency and availability, but fail to provide partition tolerance. Other database-systems like the document-based "CouchDB" are unable to guarantee consistency. What they offer instead is called eventual consistency which means, that over an infinite span of time, all database instances will be consistent – but not at once. In many applications, strong consistency is not needed (e.g. in social networks it does not matter if a comment pops up one second later).

4.1.2 Deatomization

One principle of RDBMS is that data should be stored in atomized form. This means that every kind of information should be stored in a table unique to this type of information. While in reality, mostly because of practical and performance reasons, this is not exercised to its absolute extent, an example for this would be a dataset of inhabitants: In the first table, each person is assigned an ID number. In a second table, the person's names are related to their ID. To store addresses, a third table is used in the same manner. Later, at query time, the different parts that belong together are assembled from these different tables and presented as result. When dealing with a decentralized system, it is imperative to avoid the need to assemble data from different tables (which is called "join"). Every join means more database instances that need to be queried, more time that has to be waited for a result and more traffic for the network connection.

Instead of splitting information, the target is to keep together data that belongs together. Kinds of database-systems that make heavy use of this method are document-based databases. One single document, which corresponds to one database entry, contains all information about a single entry. In case of the previous example of a database about inhabitants this would mean that every person is represented by one single document. This document contains all information about one person. Deatomization plays a major role when performing queries in distributed systems, as we will discuss in later sections.

4.1.3 Multi Version Concurrency Control

The Multi Version Concurrency Control mechanism (MVCC) is an alternative to the transaction-lock system. The key attribute is that readers never block writers and writers never block readers. This method is explained by looking at a document based NoSql databases named CouchDB.

Anderson, Lehnardt and Slater (2010) explain this system: "Documents in CouchDB are versioned (...). If you want to change a value in a document, you crate an entire new version of that document and save it over the old one. After doing this, you end up with two versions of the same document, one old and one new. ". The new document then is compared to the old one and, if the old one was not changed by another user in the meantime, the new document replaces the old one. If the old document was changed, an error message is produced and the database programmer has to implement a method to resolve this conflict. This system is compared with the transactional-lock of RDBMS by Anderson et al. (2010): "Under high load, a relational database can spend more time figuring out who is allowed to do what, and in which order, than it does doing any actual work.".

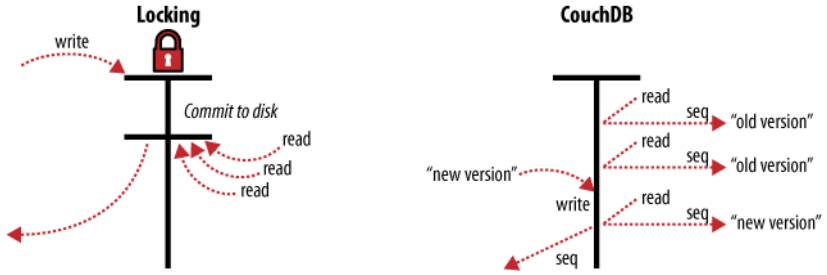

Fig. 4: A transactional locking mechanism vs. a multi-version concurrency control (image souce: (Anderson et al. 2010))

In figure 4 the two systems are visualized. On the left side, one can see that requests are blocked until a lock is released. On the right side, access to one database entry is always possible. When storing a new version, the old version is replaced after its version has been verified.

A Multi Version Concurrency Control mechanism can lift a lot of load on a database server when dealing with lots of concurrent users. The downside of this mechanic is that a method has to be implemented on the client to resolve possible conflicts. The database server is not influenced by this.

4.1.4 Queries

In 2008 Dean and Ghemawat (2008) published a new query method developed by Google to support distributed database systems. They named it "map-reduce". It is very well explained by their abstract: "Users specify a map function that processes a key/value pair to generate a set of intermediate key/value pairs, and a reduce function that merges all intermediate values associated with the same intermediate key.".

ISBN: 978-3-9503110-8-2 (CD-ROM); ISBN: 978-3-9503110-9-9 (Print)
Editors: M. SCHRENK, V. V. POPOVICH, P. ZEILE, P. ELISEI, C. BEYER

Given a dataset, a function specified by the user is executed for each entry in this dataset. This function may do whatever the user pleases and produces one to many output entries. These resulting entries are then processed by the optional reduce function that is performing a summarizing activity, also specified by the user. This technique benefits greatly from a deatomized dataset.

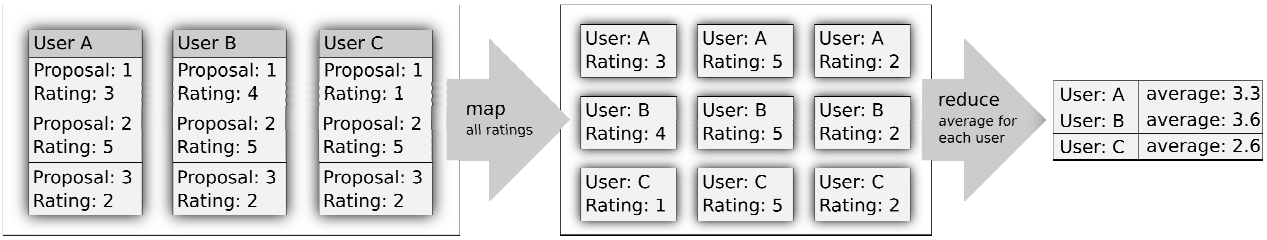

Fig. 5: Requesting a user's average voting by map-reduce

To illustrate this procedure, think of a civic involvement process where any user can rate three proposed planning drafts by using the numbers 1 through 5 (see figure 5). This data is stored in a document-based database where each document is identified by the user's name as key ("User A", "User B", "User C") and a list of ratings for each proposal as value. We want to know what the average rating of each user is to find out which persons are in general more likely to accept any planning procedure. We perform this query by first using a map function that produces a collection of key/value pairs, each describing one rating (the value) for one user (the key). Next, we run a reduce function that, per user, counts all the ratings and devides them by their total sum. The result is the average rating per user.

The catch of key/value queries is that they can be performed on a collection of data without the need to join any external dataset. If the data is split among two computers, each can perform a map without the other. The results can be collected later.

The map-reduce method enables querying on distributed systems. One does not have to think about how to construct a database query to avoid pitfalls like too many joins. Map-reduce queries are designed especially for this purpose.

4.2 Examples

We will look at some of the companies that deal with similar problems: many users, high availability and fast queries.

4.2.1 Google

In their research paper Chang et al. (2008) present a database system used and developed by Google named Bigtable. It is built to store vast amounts of information up to the amount of petabytes. It is used by Google to power most of their services, ranging from their web indexer to Google Earth, and many more. While it is an oversimplification, one can describe a Bigtable database as a single table without joins that is optimized for a huge amount of data. It supports multi-dimensional indexing through the inclusion of time as a third dimension.

This system is proprietary and only used by Google internally. There exist similar open source implementations called Hbase and Hypertable. Khetrapal and Ganesh (2006) did a comparison of these and preferred Hbase because of its stability.

4.2.2 Facebook

Facebook uses a relational database management system as backbone (namely MySQL). To achieve the possibility to distribute their data stores even with an RDBMS, they avoid joins and store data in a way that made little use of the features a RDBMS has to offer. While there is no statement about this from Facebook directly, it seems like the company started with well known web-technologies but was overrun by its own success. The technology chosen did not scale natively with the increasing user base all over the world.

While Facebook is heavily modifying the system chosen to adapt it to their circumstances, they use different non-relational database engines for different purposes. Originally only for their inbox search, a unique solution named Cassandra was developed (Lakshman & Malik 2010). This system is published as OpenSource and also used by other companies. It was developed to be expecially fast and fault tolerant in a

distributed network and is optimized for reading data from the database. For its Facebook Messages application, a system called Hadoop is used (Borthakur et al. 2011) which is somewhat similar to Googles Bigtable solution. For analysis of its immense cloud of data, a combination of specialized database software is used, as stated by Thusoo et al. (2010): “A lot of different non-RDBMS components come together to provide a comprehensive platform for processing and providing data at Facebook.”

4.2.3 Twitter

Similarly to Facebook, Twitter started out with the traditional RDBMS MySQL. Soon, its limits were reached. Facebooks database system Cassandra was implemented together with an abstraction layer for MySQL called Gizzard, enabling this database to be distributed by providing more than a simple RDBMS (Cole 2011) . For other different scenarios, various other database systems are used.

As described by Metz (2014),by now Twitter has developed an own solution called Manhattan. For this system, the previously discussed topics were taken into account. One main point Manhattan makes is to allow the user to select whether strong consistency or eventual consistency is preferred.

4.2.4 Amazon

As described by DeCandia et al. (2007), Amazon uses an own database system called Dynamo (in its newest version named DynamoDB). The affordances for such a system are described as follows: "Reliability is one of the most important requirements because even the slightest outage has significant financial consequences and impacts customer trust. In addition, to support continuous growth, the platform needs to be highly scalable.". To achieve this kind of reliability, the Dynamo database system sacrifices consistency by making it optional. In their paper, DeCandia et al. (2007) make it clear that the topmost priority is partition tolerance. There is no failure allowed. The shopping chart has to be available all the time, even when a tornado destroys a single data center. Vogels (2009) gives an in-depth report on how this eventual consistency is managed at Amazon. The set up of a system with this level of reliability and scalability is greatly facilitated by embracing the theoretical points mentioned before.

5 THE META-LEVEL: MODELLING COLLABORATION

We have looked at which techniques are available to cope with vast amounts of data coming from many users. There is yet a second aspect: when looking at collaborative networks, there exists a special kind of data structure that is suited for modelling network structures: So-called Graphs.

5.1 Neo4j

A very prominent example for graph databases is a rather young company called Neo Technology. They offer a pure graph database called Neo4j. What is so unique about this database is that it does not only model its data as a graph but actually stores it as such. Robinson, Webber and Eifrem (2013) describe this system in depth.

The big gains of using this technique are the profoundly different analytical methods offered by a graph database. It becomes possible to query for relations of the graph in a very efficient manner. This is a time consuming task when using RDBMS. An example of such a query would be “all people that have up to a third grade relative who likes to go biking”.

An alternative to Neo4j is Cayley, a true open source graph database developed by Peters et al. (2014). Its origins lie within the technology used by the Google Freebase engine (Bollacker,Evans,Paritosh,Sturge & Taylor 2008), "(...) a practical, scalable tuple database used to structure general human knowledge.".

5.2 Resource Description Framework (RDF)

When talking about graphs, collaboration and huge amounts of information, is becomes unavoidable to mention RDF. Nearly any kind of information can be stored by so-called "rdf triples". These triples constitute the base-granularity of all data to be stored and are one prerequisite to achive a maximum of interoperability between different sources of data. Abadi,Marcus,Madden and Hollenbach (2009) say about this: "(…) the ‘Resource Description Framework’, or RDF, represents data as statements about resources using a graph connecting resource nodes and their property values with labeled arcs representing properties. Syntactically, this graph can be represented using XML syntax (RDF/XML). This is typically the format for

ISBN: 978-3-9503110-8-2 (CD-ROM); ISBN: 978-3-9503110-9-9 (Print)
Editors: M. SCHRENK, V. V. POPOVICH, P. ZEILE, P. ELISEI, C. BEYER

RDF data exchange; however, structurally, the graph can be parsed into a series of triples, each representing a statement of the form < subject, property, object > (...)". Many efforts are made to make distributed sorage and processing of triples as efficient as possible.

The driving force behind research about RDF is the liked data project. As said by Wood,Zaidman,Ruth and Hausenblas (2014), "Linked Data makes the World Wide Web into a global database that we call the Web of Data.". (figure 6 shows the current state of the linked data cloud with each node representing one single source)

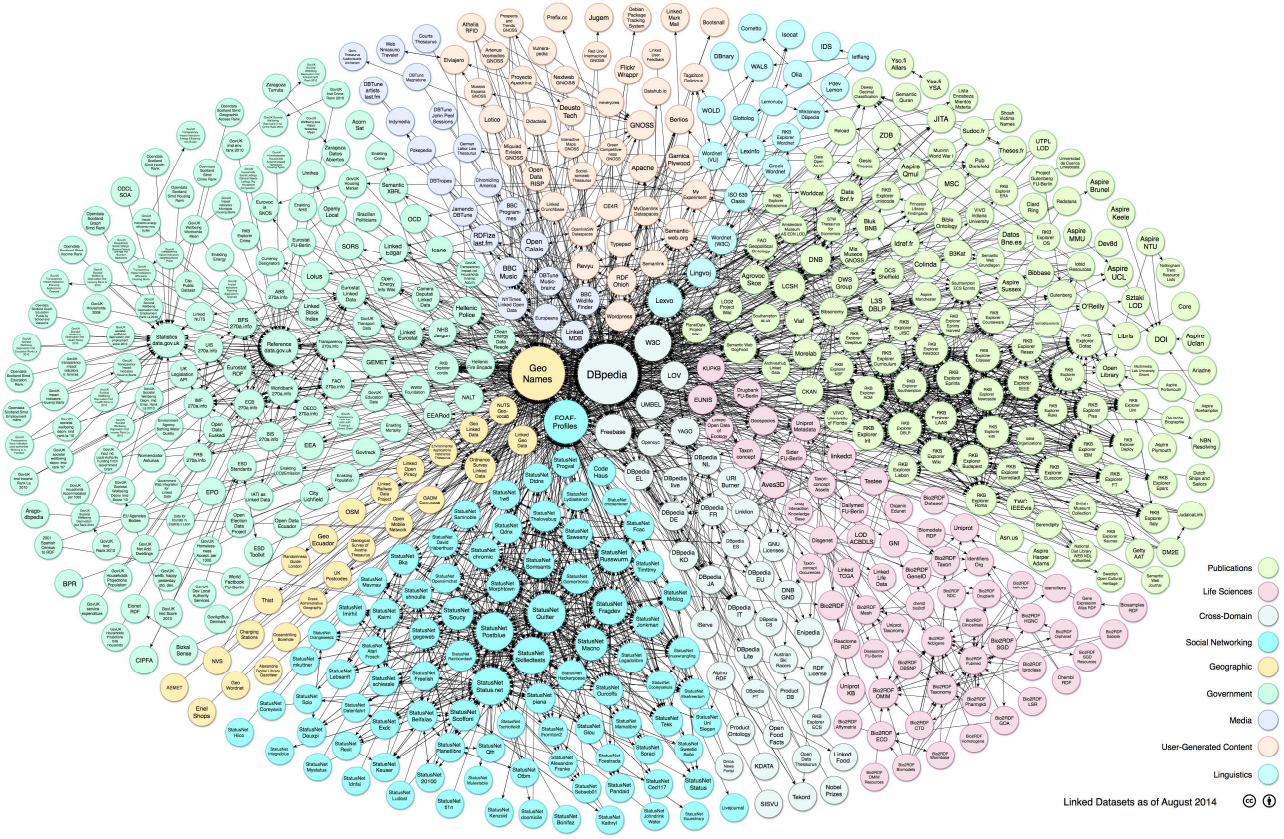

Fig. 6: Linking Open Data cloud diagram 2014, by Max Schmachtenberg, Christian Bizer, Anja Jentzsch and Richard Cyganiak. http://lod-cloud.net/

They further explain that "The term Linked Data refers to a set of best practices for publishing and connecting structured data on the Web using international standards of the World Wide Web Consortium." . To explain all the aspects involved is not the scope of this paper, but one can state that this is a huge project with great implications on how information might be handeld in the future. From a technological perspective it is still very young but its implications concerning data handling, data processing and data sharing are huge.

6 CONCLUSION

It becomes clear that the demands on database systems have changed: from stand-alone monolithic solutions needed in 1970s, today we are in need of more flexible and distributed alternatives that can cope with structured and, especially, low- or un-structured data.

The solution is not to "overload" present techniques by developing cumbersome workarounds to enable unsupported features. Rather, one should take advantage of the whole arsenal of new methods and techniques that are natively able to cope with problems arising when building collaborative and public participatory systems.

Obviously, one has to know about them in order to make a wise and informed decision, as was the aim of this paper: to make the spatial planning community more aware of the state-of-the-art and uprising technologies.

7 REFERENCES

ABADI, D. J.; MARCUS, A.; MADDEN, S. R.; HOLLENBACH, K.: SW-Store: A Vertically Partitioned DBMS for Semantic Web Data Management. In: The VLDB Journal Vol. 18, pp. 385-406. 2009

ANDERSON, J. C.; LENHARDT, J.; SLATER, N.: CouchDB: The definitive guide. O'Reilly Media Inc., 2010

BOLLACKER, K.; EVANS, C.; PARITOSH, P.; STURGE, T.;TAYLOR, J.: Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge. In: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, pp. 1247-1250. 2008

BORTHAKUR, D.; GRAY, J.; SARMA, J. S.; MUTHUKKARUPPAN, K.; SPIEGELBERG, N.; KUANG, H.; RANGANATHAN, K.; MOLKOV, D.; MENON, A.; RASH, S.; SHMIDT, R.;AIYER, A.: Apache Hadoop Goes Realtime at Facebook. In: Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data, pp 1071-1080. 2011

CATTELL, R.: Scalable SQL and NoSQL Data Stores, SIGMOD Rec., Vol 39, pp. 12-27. 2011

CHANG, F.; DEAN, J.; GHEMAWAT, S.; HSIEH, W. C.; WALLACH, D. A.; BURROWS, M.; CHANDRA, T.; FIKES, A.; GRUBER, R. E.: Bigtable: A Distributed Storage System for Structured Data, ACM Trans. Comput. Syst., Vol 26, pp. 1-26. 2008

CODD, E. F.: A Relational Model of Data for Large Shared Data Banks, Commun. ACM, Vol 13, pp. 377-387. 1970

COLE, J.: Big and Small Data at @Twitter. In: O'Reilly MySQL CE. 2011

DEAN, J.; GHEMAWAT, S.: MapReduce: simplified data processing on large clusters. In: Communications of the ACM, Vol 51, pp. 107-113. 2008

DECANDIA, G.; HASTORUM, D.; JAMPANI, M.; KAKULAPATI, G.; LAKSHMAN, A.; PILCHIN, A.; SIVASUBRAMANIAN, S.; VOSSHALL, P.; VOGELS, W.: Dynamo: Amazon's Highly Available Key-value Store. In: SIGOPS Oper. Syst. Rev., Vol. 41, pp. 205-220. 2007

FOTH, M.; BAJRACHARYA B.; BROWN R.; HEARN G.: The Second Life of urban planning? Using NeoGeography tools for community engagement. In: Journal of Location Based Services, Vol 3, Issue 2, pp. 97-117. 2009

GILBERT, S.; LYNCH, N.: Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. In: SIGACT News, Vol. 33, pp. 51-59. 2002

GOODCHILD, M. F.: Citizens as sensors: the world of volunteered geography. In: GeoJournal, Vol 69, pp. 211-221. 2007

GRAY, J.: The transaction concept: Virtues and limitations. In: Proceedings of the Seventh International Conference on Very Large Data Bases, Vol 7, pp. 144-154. Cannes, 1981

HAERDER, T.; REUTER, A.: Principles of Transaction-oriented Database Recovery. In: ACM Comput. Surv., Vol. 15, pp. 287-317. 1983

KEMPER, A. and EICKLER, A.:Datenbanksysteme: Eine Einführung. Oldenbourg Verlag, 2011

KHETRAPAL, A. and GANESH, V.: HBase and Hypertable for large scale distributed storage systems. Dept. of Computer Science Purdue University, 2006

LAKSHMAN, A.; MALIK, P.: Cassandra: A Decentralized Structured Storage System. In: SIGOPS Oper. Syst. Rev., Vol. 44, pp. 35-40. 2010

LINDSAY, B. G.; SELINGER, P. G.; GALTIERI, C. A.; GRAY, J. N.; LORIE, R. A.; PRICE, T. G.; PUTZOLU, F.; TRAIGER, I. L.; WADE, B. W.: Notes on Distributed Databases. IBM, 1979

METZ, C.: This is what you build to juggle 6,000 Tweets a second. In: Wired Magazine Online, URL: http://www.wired.com/2014/04/twitter-manhattan . 2014

PETERS, A.; MICHENER, B.; LEENDERS, B.; GRAVES, J.; JAY, J.; UZAMERE, P.; KORTSCHAK, R. D.; ARMSTRONG, T.; LI, Z.: Cayley - An open-source graph database, URL: https://github.com/google/cayley . 2014

POPLIN, A.: Playful public participation in urban planning: A case study for online serious games. In: Computers, Environment and Urban Systems, Vol. 36, pp. 195-206. 2012

ROBINSON, I.; WEBBER, J.;EIFREM, E.: Graph databases. O'Reilly Media, Inc. 2013

SCALEBASE INC.: ScaleBase - A Distributed MySQL Database, URL: https://www.scalebase.com . 2014

SIEBER, R.: Public Participation Geographic Information Systems: a Literature Review and Framework. In: Annals of the association of American Geographers, Vol. 96, Issue 3, pp. 491-507. 2006

THUSOO, A.; SHAO, Z.; ANTHONY, S.; BORTHAKUR, D.; JAIN, N.; SEN SARMA, J.; MURTHY, R.; LIU, H.: Data Warehousing and Analytics Infrastructure at Facebook. In: Proceedings of the 2010 ACM SIGMOD International Conference on Management of data, pp. 1013-1020. 2010

VOGELS, W.: Eventually Consistent. In: Commun. ACM, Vol. 52, pp. 40-44. 2009

WOOD, D.; ZAIDMAN, M.; RUTH, L. and HAUSENBLAS, M.: Linked Data. Manning Publications Co. 2014

ISBN: 978-3-9503110-8-2 (CD-ROM); ISBN: 978-3-9503110-9-9 (Print)
Editors: M. SCHRENK, V. V. POPOVICH, P. ZEILE, P. ELISEI, C. BEYER