Data Vaults: a Database Welcome to Scientific File Repositories

Milena Ivanova
Netherlands eScience Center
M.Ivanova@esciencecenter.nl

Yağız Kargın
CWI Amsterdam
Yagiz.Kargin@cwi.nl

Martin Kersten
CWI Amsterdam
Martin.Kersten@cwi.nl

Stefan Manegold
CWI Amsterdam
Stefan.Manegold@cwi.nl

Ying Zhang
CWI Amsterdam
Y.Zhang@cwi.nl

Mihai Datcu
DLR
Mihai.Datcu@dlr.de

Daniela Espinoza Molina
DLR
Daniela.EspinozaMolina@dlr.de

ABSTRACT

Efficient management and exploration of high-volume scientific file repositories have become pivotal for advancement in science. We propose to demonstrate the Data Vault, an extension of the database system architecture that transparently opens scientific file repositories for efficient in-database processing and exploration.

The Data Vault facilitates science data analysis using high-level declarative languages, such as the traditional SQL and the novel array-oriented SciQL. Data of interest are loaded from the attached repository in a just-in-time manner without need for up-front data ingestion.

The demo is built around concrete implementations of the Data Vault for two scientific use cases: seismic time series and Earth observation images. The seismic Data Vault uses the queries submitted by the audience to illustrate the internals of Data Vault functioning by revealing the mechanisms of dynamic query plan generation and on-demand external data ingestion. The image Data Vault shows an application view from the perspective of data mining researchers.

1. INTRODUCTION

Scientific disciplines have entered the era of *data intensive* research [8]. Unprecedentedly large data volumes are generated by advanced observatory instruments or powerful simulations and demand efficient technology for science harvesting. The current approach for navigation and processing over file-based repositories is built upon customized tools, which blend data access, computational analysis and visualization. The increasing data volumes push this approach to the limit revealing various problems and shortcomings. The data-intensive paradigm demands new ways of conducting scientific exploration which provide scalable access, flexibility to incorporate new studies and ability to deploy processing "near the data" [7].

Decades of progress have matured database management systems to provide efficient management and processing of large volumes of business data and offer a flexible declarative query language. However, numerous factors impede their wide adoption in scientific applications. The most important problems concerning scientific file-based repositories are i) too high up-front cost to port data and applications to the DBMS; ii) lack of interfaces to exploit standard science file formats; and iii) limited access to external science libraries during query processing and visualization.

We will demonstrate the *Data Vault*, a true symbiosis between a DBMS and existing (remote) file-based repositories, concept and vision of which was introduced in [11] at SSDBM 2012. With the Data Vault, a user can simply *attach* an external file repository to the DBMS, e.g., using a URI, and let it conduct efficient and flexible query processing over the data of interest. The Data Vault keeps the data in its original format and place, which preserves the possibilities to run existing external tools upon need. At the same time it enables transparent (meta)data access and analysis using database query languages so that scientists can benefit from extended functionality and flexibility. High level declarative query languages (SQL, SciQL) facilitate experimentation with novel science algorithms. Transparent, just-in-time data loading reduces the start-up cost associated with adopting a database solution for existing file repositories.

The Data Vault achieves this by extending the existing database system architecture in several ways, each addressing a problem that impedes using DBMS in science. First, it adds support for *external scientific formats*. This requires understanding of data formats beyond the prevalent CSV files, and a mapping of the external data models to the database model. Second, the Data Vault changes the query processing framework to *access just-in-time* data of interest per current query. This alleviates the burden of the traditional requirement for up-front data ingestion into the database. This

SSDBM '13, Jul 29-31 2013, Baltimore, MD, USA.
ACM 978-1-4503-1921-8/13/07

requirement demands a high initial investment of time and effort and has been recognized as one of the major obstacles in adopting database solutions [17].

Finally, traditional approaches often provide limited in-database processing capabilities for non-standard data types. The Data Vault is developed in the context of MonetDB and its scientific array-query language SciQL [21]. This enables in-database processing of arrays, including both external black-box and internal white-box user defined functions. In this demonstration we focus on the Data Vault as enabling technology for more efficient in-database processing.

The need to dynamically access external data from inside the database system has been addressed by the external table feature provided by all major DBMS vendors. It supports predominantly external tabular data, such as flat CSV files and data from other SQL-server vendors. Advanced query processing over such data is offered by solutions such as NoDB [2] as demonstrated in [1]. The limitation to tabular external data has been addressed by the Universal File Interface extensions [20] which allow for indexing and querying of more complex scientific formats, e.g., NetCDF, HDF5.

However, in all of the above cases the user has to specify an explicit mapping of external data into in-database tabular structures. For a real-life repository of thousands and even millions of files, such as ORFEUS [14], such explicit enumeration of external data items limits scalability and requires separate detection of data sets of interest. In contrast, the Data Vault provides a view over external data, including complex, non-tabular, and compressed sources, in an automatic and transparent way. It hides the details from the user, who has to solely attach a file repository.

Modern object-relational and middleware systems offer various extensions to store and process complex (raw) data and their associated metadata. Examples are middleware-based RasDaMan for multi-dimensional data and geo-spatial extenders, such as Oracles' Spatial and Graph and PostGIS. These systems require explicit and up-front data ingestion before query processing can take place.

We propose to demonstrate two scientific use cases of the Data Vault. The seismology Data Vault provides access to seismic data stored in files in mSEED format [16]. The GeoTIFF [15] images Data Vault serves the needs of data mining applications over Earth observations. The Data Vaults 'understand' the external mSEED and GeoTIFF formats and allow the user to attach repositories, which provides easy browsing and navigation facility over repository metadata. In-database processing and analysis of the external data is performed using declarative queries in a language appropriate for the data model (SQL or SciQL).

2. DATA VAULT ARCHITECTURE

The Data Vault architecture is designed in the context of MonetDB [13], an open-source column-store database system developed at CWI. The design was driven by the following main principles: **(1)** transparent access to repository metadata; **(2)** dynamic, on-demand data loading driven by query needs; **(3)** symbiotic query processing combining the benefits of available science libraries with in-database processing; **(4)** transparent synchronization of in-database content with the external repository.

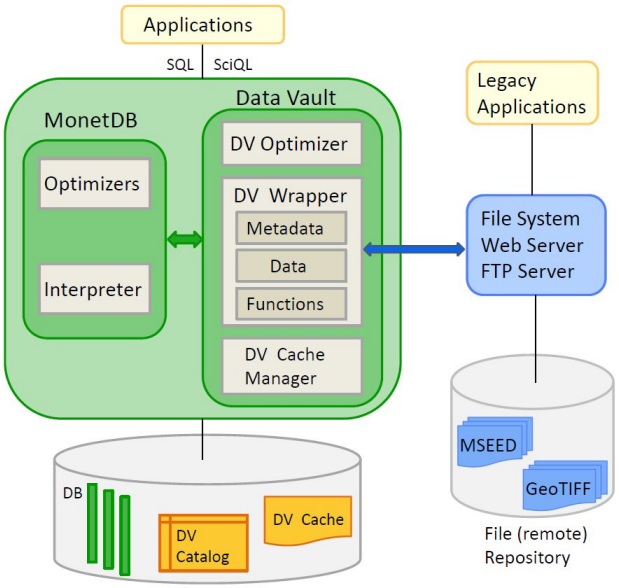

Figure 1: MonetDB Data Vault (DV) architecture

Figure 1 illustrates the Data Vault (DV) architecture [11]. It extends the MonetDB software architecture with three components: a wrapper, an optimizer, and a cache manager. The *Data Vault wrapper* is responsible for the communication with the file repository and is built around data model mapping, also applying conversions from the model of file format to the database schema model. It has components to access data, metadata, and external libraries and tools. The metadata wrapper accesses self-descriptive metadata of external files and populates the *Data Vault catalog* in the database to facilitate repository browsing, query formulation, and data management.

The data wrapper component accesses the external data and creates their internal representation in the *Data Vault cache*. Formally, the Data Vault cache is presented to the user as a virtual table or array, which allows query formulation over external data before their actual ingestion. The functionality wrapper provides access to external libraries and tools. It defines mappings between external functions' input parameters and results to valid database representations. The Data Vault cache contains snippets of repository data imported into the database as a part of query processing. This component leverages our work on recycling intermediates [10]. The *Data Vault cache manager* is responsible for keeping the database cache in sync with the repository. The cached data can also be transformed into persistent structures in the database. This transformation can be adaptively driven by the workload or explicitly induced by the user.

The purpose of the *Data Vault optimizer* is the symbiotic query processing. Using the Data Vault catalog, it detects operations over repository data and makes decisions about their execution locations based on a cost model. It might

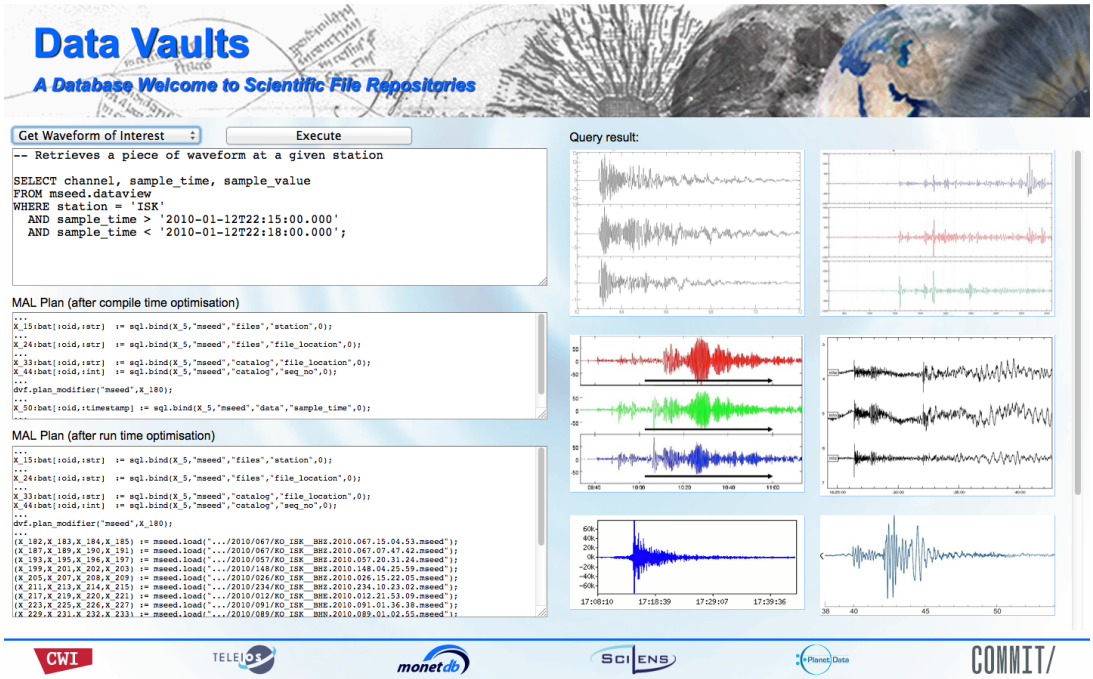

Figure 2: GUI of the mSEED Data Vault.

delegate processing to an external tool and correspondingly inject a call to the functional data wrapper. If in-database processing is deemed more efficient, the optimizer applies a two-phase optimization, compile-time and run-time, to facilitate automatic on-demand loading of data from files [12].

At compile-time, the optimizer's challenge is to reorganize the plan so that the selection predicates on the metadata are to be evaluated first, to determine the actual files to be loaded. Otherwise, the actual data to be processed for that query is not known. Once this part of the plan is executed, a rewriting operator is triggered. This operator uses plan introspection provided by the MonetDB system and modifies the rest of the plan at run-time, mainly replacing all references to relational tables representing external data with operators (data wrapper calls) that load the necessary files. Internally these operators use external scientific library calls to extract the data out of the specific file formats.

Figure 1 also illustrates the extended opportunities provided by the Data Vault. Existing legacy applications using middleware solutions and scientific libraries can operate over the file repository as before. New applications can rely on declarative SQL and SciQL query requests to MonetDB's Data Vault to boost analysis of external scientific data.

3. DEMONSTRATION

The demonstration offers two use cases of the Data Vault implementation for seismology and remote-sensing images.

3.1 Seismology Data Vault

In seismology, SEED [16] is the most widely used standard file format to exchange waveform data among seismograph networks. A SEED volume mainly contains the waveform time series as highly compressed data records. A SEED repository requires up to 10 times the original storage size when loaded into a database [12]. Additionally, several ASCII control headers are included in a SEED file. The control headers contain the metadata consisting of identification and configuration information about the actual data (i.e., data records). In this demo we use the Mini-SEED (mSEED) variant, which reduces the SEED metadata to the most widely used subset. The sizes of mSEED files commonly vary from 4 KB to several MBs. Millions of them are stored in remote file repositories with direct FTP access [14].

As part of the COMMIT project [3], the Royal Dutch Meteorological Institute (KNMI) conducts research on discovery and comprehension of seismic events over seismic waveform data collected from the European-Mediterranean region. Seismologists can greatly benefit from the Data Vault while hunting for 'interesting' seismic events and analyzing them. Seismologists do not have to wait for lengthy up-front extraction and loading. They enjoy the just-in-time data access by specifying their interests as query predicates. They can run any declarative query for their analysis tasks, such as computing *Short Term Averages*, *Long Term Averages*, and/or retrieving the data of a record of interest for visualization and visual analysis, etc. A more detailed view on mSEED data and the seismology Data Vault is given in [12].

The implementation of the mSEED Data Vault uses the open source library LIBMSEED [9]. The demonstration scenario allows to zoom into the details of the Data Vault architecture and functioning. In particular, the attendees can experiment with (1) attaching an mSEED repository of any size and browsing the metadata, (2) inspecting the data content of an mSEED file as with their traditional software, but now as in-database tabular structures, (3) declarative querying through the metadata and actual data to locate time se-

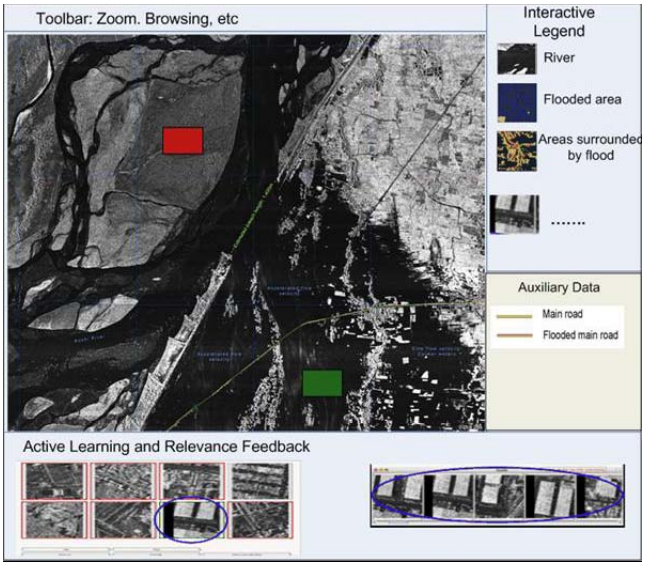

Figure 3: Example GUI. Users select positive (green box) & negative (red box) samples for classification.

ries of interest, and (4) visualizing the resulting time series as seismograms, while observing the modifications done on the query plans for realization of just-in-time actual data access. In addition, there are also predefined queries for demonstration purposes that can be used when looking for interesting seismic events. To illustrate, a snapshot of the GUI is shown in Figure 2.

3.2 Data Mining Remote Sensing Images

TELEIOS is an EU project to build a virtual observatory for Earth observation data [18]. As part of the project, the German Space Agency (DLR) conducts research on knowledge discovery and data mining (*KDD*) over Earth observation images. The source data are high-resolution TerraSAR-X radar images [5] in GeoTIFF format accompanied by metadata specification in XML format, and geo-spatial data sources. The data sizes vary from 300 MB to 2 GB per image depending on the product type.

To support KDD over TerraSAR-X images [4], low-level features are extracted from image chunks and mapped into semantic classes and symbolic representations meaningful for the end user. The features extracted by various methods are stored in a database and used as input for a higher level image analysis, such as classification and content-based image retrieval. The initial image processing (patching and feature extraction) is carried out by customized software tools that operate directly over image files in file repositories.

Due to the unique properties of radar images, e.g., grayscale, high resolution and size, feature extraction from them is an active research topic by itself. This requires continual and tedious changes of the processing pipelines while searching for the best method. Scientists can greatly benefit from a higher degree of flexibility when experimenting with the quality and efficiency of various image processing methods. Such flexibility can be provided if patch construction and feature extraction are specified as declarative queries executed inside the database kernel. The Data Vault enables this opportunity by providing transparent access to external image repositories.

The implementation of the GeoTIFF Data Vault uses the open-source libraries LIBTIFF [19] and LIBGEOTIFF [6] to access image data and metadata in GeoTIFF format. The application scenario is built around data mining over TerraSAR-X Earth observation images. For demonstration purposes, the Data Vault is already attached, a number of images loaded, and processed with the feature extraction algorithms. The application uses a GUI, illustrated in Figure 3, to visualize a set of image patches and to select positive and negative examples for an image class. The features of the selected images are then provided as input to the Support Vector Machine for classification. The results can be used to automatically classify new images.

Acknowledgments

The work reported here was supported by the EU-FP7-ICT project TELEIOS and the Dutch national project COMMIT. We wish to thank our partners in both projects, in particular our colleagues at KNMI, for constructive guidance on the functionality and implementation of the MonetDB Data Vault.

4. REFERENCES

[1] I. Alagiannis, R. Borovica, M. Branco, S. Idreos, and A. Ailamaki. NoDB in Action: Adaptive Query Processing on Raw Data. *PVLDB*, 5(12), 2012.
[2] I. Alagiannis et al. NoDB: Efficient Query Execution on Raw Data Files. In *SIGMOD*, 2012.
[3] COMMIT Project. `http://www.commit-nl.nl/`, 2013.
[4] C. O. Dumitru et al. TELEIOS WP3: KDD concepts and methods proposal: report and design recommendations. `http://www.earthobservatory.eu/deliverables/FP7-257662-TELEIOS-D3.1.pdf`.
[5] T. Fritz et al. TerraSAR-X Ground Segment: Basic Product Specification Document, TX-GS-DD-3302, 2008.
[6] Libgeotiff. `http://trac.osgeo.org/geotiff/`, 2013.
[7] J. Gray et al. Scientific Data Management in the Coming Decade. *SIGMOD Record*, 34(4), 2005.
[8] T. Hey et al. *The Fourth Paradigm: Data-Intensive Scientific Discovery*. Microsoft Research, 2009.
[9] IRIS. libmseed: Mini-SEED Software Library, 2011.
[10] M. Ivanova et al. An Architecture for Recycling Intermediates in a Column-store. In *SIGMOD*, 2009.
[11] M. Ivanova et al. Data Vaults: a Symbiosis between Database Technology and Scientific File Repositories. In *SSDBM*, 2012.
[12] Y. Kargin et al. Instant-On Scientific Data Warehouses — Lazy ETL for Data-Intensive Research. In *BIRTE*, 2012.
[13] MonetDB. `http://www.monetdb.org/`, 2013.
[14] ORFEUS. Seismology Event Data (1988 - now). `ftp://www.orfeus-eu.org/pub/data/POND/`.
[15] N. Ritter and M. Ruth. GeoTIFF format specification, Revision 1.0. `http://trac.osgeo.org/geotiff/`.
[16] SEED. Standard for the exchange of earthquake data, 2010. `www.iris.edu/manuals/SEEDManual_V2.4.pdf`.
[17] M. Stonebraker et al. Requirements for Science Data Bases and SciDB. In *CIDR*, 2009.
[18] TELEIOS. `http://www.earthobservatory.eu/`, 2013.
[19] Libtiff. `http://www.libtiff.org/`, 2013.
[20] Universal File Interface . `http://www.barrodale.com/universal-file-interface-ufi`, 2013.
[21] Y. Zhang et al. SciQL: Bridging the Gap between Science and Relational DBMS. In *IDEAS*, 2011.