On-Demand Service-Based Big Data Integration: Optimized for Research Collaboration

Pradeeban Kathiravelu[1,2], Yiru Chen[3], Ashish Sharma[4],
Helena Galhardas[1], Peter Van Roy[2], Luís Veiga[1]

[1]INESC-ID / Instituto Superior Técnico, Universidade de Lisboa, Portugal,
[2]Université catholique de Louvain, Belgium,
[3]Peking University, China, [4]Emory University, GA, USA
[1]{pradeeban.kathiravelu, helena.galhardas, luis.veiga}@tecnico.ulisboa.pt,
[2]peter.vanroy@uclouvain.be, [3]chen1ru@pku.edu.cn, [4]ashish.sharma@emory.edu

Abstract. Biomedical research requires distributed access, analysis, and sharing of data from various disperse sources in the Internet scale. Due to the volume and variety of big data, materialized data integration is often infeasible or too expensive including the costs of bandwidth, storage, maintenance, and management. *Óbidos* (**O**n-demand **Bi**g Data **I**ntegration, **D**istribution, and **O**rchestration **S**ystem) provides a novel on-demand integration approach for heterogeneous distributed data. Instead of integrating data from the data sources to build a complete data warehouse as the initial step, *Óbidos* employs a hybrid approach of virtual and materialized data integrations. By allocating unique identifiers as pointers to virtually integrated data sets, *Óbidos* supports efficient data sharing among data consumers. We design *Óbidos* as a generic service-based data integration system, and implement and evaluate a prototype for multimodal medical data.

Keywords: Materialized Data Integration; Virtual Data Integration; Integrated Data Repository; Extract, Transform, Load (ETL); Big Data Integration; DICOM

1 Introduction

Scale and diversity of big data is on rise, with geographically distributed data of exabytes. Integration of data is crucial for various application domains ranging from medical research [1] to transport planning [2], to enable data analysis, finding relationships, and retrieving information, across data from various sources. However, storing the integrated data in a separate data repository can be deferred to query time or avoided altogether. This approach minimizes outdated data in the integrated data repository, and mitigates the maintenance costs associated with frequent data refreshments and data cleaning processes.

Disparate multimodal medical data [3] comprising imaging, clinical, and genomic data is published by data producers such as physicians and health personnel. Data of each of these domains (including clinical, pathology, and other

subdomains of medical data research) are of heterogeneous types and various formats - textual, binary, or a hybrid of both. Medical data science researchers need to consume data sets spanning multiple data sources. This typically requires integration and analysis of multimodal medical data.

Adhering to standard formats such as DICOM (The Digital Imaging and Communications in Medicine) [4] and NIfTI (Neuroimaging Informatics Technology Initiative) [5], each medical image consists of binary raw image data and textual metadata. This textual metadata, present in many file formats that support a large-scale binary data, permit a faster analysis or provide a brief overview of the respective binary data. Most of the medical research studies access or query only the metadata at least till a certain subset is identified for further detailed analysis. Moreover, each medical data research is typically limited to certain sets of data from the data sources. Therefore, the researchers do not need to integrate entirely the data sources, which is also infeasible considering the volume of the data. While the data sets of interest vary between the researchers, the subsets of data also need to be shared among researchers for collaborations and to repeat the experiments for verifiability of scientific and medical research.

Traditional/eager ETL (Extract, Transform, Load) process constructs the entire data warehouse as the initial process. This bootstrapping process comprising the initial loading is lengthy and the study-specific researches are unnecessarily delayed when only a small subset of data is relevant to the queries. Lazy ETL [6] avoids the lengthy bootstrapping process by eagerly loading only the metadata while the actual data is loaded in a lazy manner. The actual data is accessed only at query time. Hence, bootstrapping in lazy ETL is very fast. Nevertheless, lazy ETL approach does not offer scalability to store, access, and share big data efficiently across multiple data consumers.

Motivation: Consider a research to study the effects of a medicine to treat brain tumor in patients of certain age groups against that of a placebo. This research involves a large amount of data consisting of cancer images, clinical records, lab measurements, and observations, collected over past several months. Though the data is huge, the metadata often gives an overall description of the binary data, sufficient for at least the early stages of a research study. When performing data analysis, loading of binary data can be deferred, analysing the textual metadata first to find the required binary data. Hence, loading of cancer images for integration and analysis should be deferred to a latter phase, loading the relevant sets of metadata at first. This approach is more computation and bandwidth efficient, compared to loading all the binary data at once.

The data sets of interest are well-defined by the medical researcher as a small sub set of a much larger data from multiple data sources. The existing data integration approaches are unfit for the study-specific medical research, as these approaches do not leverage the domain knowledge of the researcher, in filtering the data early on. On the other hand, manually searching, downloading, and integrating data is repetitive, time consuming, and expensive. Hence, medical research requires an approach to i) incrementally construct a study-specific integrated data repository on-demand, ii) share interesting subsets of data with

minimal bandwidth consumption, iii) track the loading of data from data sources to load only the changes or updates, and iv) minimize near duplicates due to corrupted or moved data in the integrated data repository, or changes/updates in the data sources.

Contribution: We present *Óbidos*[1], an on-demand data integration system for heterogeneous big data, and implement its prototype for medical research, addressing the identified shortcomings. *Óbidos* follows a hybrid approach in loading and accessing data. Following lazy ETL, it avoids eager loading of binary data, exploiting the metadata in accessing and integrating binary data from data sources.

Óbidos leverages the well-defined hierarchical structure of medical data of formats such as DICOM to construct subsets of data that are of interest. *Óbidos* loads only a subset of metadata, unlike lazy ETL that advocates an eager loading of metadata. First, *Óbidos* finds a superset of data, including that of the data sets of interest. Then it loads the metadata of the chosen superset, for example: i) in a hierarchical data structure such as DICOM, the chosen subset consists of an entire level of granularity that has the data that is of interest to the tenant; ii) in a file system, it may include contents of an entire directory. By leveraging the data loading APIs offered by the respective data source platforms, only the chosen set of metadata is queried, instead of querying the whole data sources, which is not practical and redundant.

The integrated data access tends to be slower in Lazy ETL as it only loads metadata till the query is processed, whereas the associated binary data is much larger in volume and will consume a large time to load from the data sources depending on the network bandwidth. *Óbidos* overcomes this by having a materialized data integration for the data that is the outcome of the queries, in a scalable storage with an efficient approach to access the data. Furthermore, *Óbidos* leverages the metadata in tracking the changes and downloads from the data sources.

Óbidos *Orchestration:* For loading and sharing of medical research data, we envision *Óbidos* to be deployed in multiple research institutes. Each such institute is called a data consumer, who needs to integrate and consume research data from multiple data sources, and share among one another. *Óbidos* is to be typically hosted in a cluster of computers in a research institute as a single yet, distributed deployment. Each such deployment is defined as a data consumer deployment, that is used by multiple researchers, who are called the tenants of the *Óbidos* deployment.

Óbidos identifies data that is integrated by the respective tenant. Through the service-based interface of *Óbidos*, tenants can access the data in their respective *Óbidos* deployment, or data integrated in a remote *Óbidos* deployment. Customizable to the policies of the research institutes and the needs of the users,

[1] *Óbidos* is a medieval fortified town that has been patronized by various Portuguese queens. It is known for its sweet wine, served in a chocolate cup.

Óbidos orchestrates the way research data is shared among the users. In addition to colleagues in a research institute sharing data among them, peers from other institutes may invoke the web service interface of *Óbidos* to access the shared data in any *Óbidos* deployment. The service-based approach facilitates protected access through the Internet to the integrated data repository of *Óbidos* data consumer deployment.

The rest of this paper is structured as follows: Section 2 discusses the background behind the multimodal medical data integration and *Óbidos* approach. Section 3 elaborates the deployment and solution architecture of *Óbidos*. Section 4 discusses how *Óbidos* prototype is built for multimodal medical research data. Section 5 evaluates *Óbidos* for its performance. Section 6 discusses the related work on service-based and virtual data integration platforms. Finally, Section 7 provides a summary of the current state and future research work based on *Óbidos*.

2 Background

In this section, we look at the motivations behind *Óbidos* and its contributions, with a background in medical data access, integration, and sharing.

Data integration is performed traditionally by ingesting data from various sources into a data warehouse and processing it afterwards. Due to the proliferation of binary data, the associated textual metadata often replaces the data itself in indexing, identifying, and loading the relevant data. In multidisciplinary research, the number of potentially relevant data sources are monotonically increasing. Therefore, a massive amount of data or metadata need to be loaded initially, if a data warehouse is to be constructed from the entire data sources as the first step. This leads to a waste of time and storage where a large fraction of data that is useless and irrelevant is loaded in the beginning. This can be avoided by improving the lazy loading approach of loading only the metadata with the research-specific knowledge on the data sets of interest. Instead of loading all the metadata from the sources, we advocate loading only the metadata corresponding to the datasets of interest, or a superset of them (rather than loading the entire metadata, which is infeasible). Users may specify all the data sets of interest to load the relevant metadata. Alternatively, it should also be possible to infer their interest from the search queries.

Óbidos attempts to solve two major challenges in research big data integration and sharing: i) the challenges of volume and variety associated with the distributed landscape of medical research data and the data sources, and ii) the need for a scalable implementation to integrate and share the big data between various researchers.

2.1 Integrating Distributed and Disperse Data

Big data integration needs to consider tens of thousands of data sources even in a single domain [7]. Multi-domain and cross-disciplinary researches have left

us with a much larger and diverse number of data sources to be accessed and integrated, compared to the traditional data integration requirements. Velocity, variety, and veracity aspects of big data present further challenges in addition to the volume of the data involved. Hence, if possible, a subset of the data needs to be chosen in advance for the integration, instead of a complete offline warehousing from all the available data sources [7].

The very nature of dispersed data from various data repositories poses a set of issues that need to be addressed. Due to the increasing number and proliferation of data repositories, it is not only unnecessary, but also infeasible to access all the data or even the metadata. The repositories offer protected access to the data they host, typically through a web interface such as web services or web applications. They present a hierarchical or indexed view of the data they host, despite the data itself being often binary and unstructured. These respective interfaces of data repositories need to be leveraged in efficiently accessing and loading data. Traditional approach of loading an entire data or metadata is not applicable in service-based data platforms.

2.2 Scalable Implementation for Big Data Processing

With the increasing scale and complexity of data storage and analysis, it becomes evident that each task of an ETL process should be executed in a parallel manner appropriately, for a faster completion [8]. Distributing the ETL process may reduce latency in accessing, transforming, and loading data into a data warehouse [9], thus preventing main memory from becoming a bottleneck in data transformations at any point. In-Memory Data Grid (IMDG) platforms [10] can be leveraged in data integration and data quality platforms to improve performance and scalability. For example, $\partial u \partial u$ [11] provides a multi-tenant in-memory distributed execution platform for near duplicate detection algorithms for data integration from multiple sources.

The data sources consist of heterogeneous data, often unstructured or semi-structured. Hence, the integrated data should be stored in a scalable storage that supports storing of unstructured data. Thus, typical medical data integration solutions that restrict themselves to well-defined schemas are not suitable for integrating medical research big data. As NoSQL solutions allow storing unstructured data, they should be leveraged along with an architecture that supports indexing of the stored binary data. Hence, even originally unrelated - structured and unstructured data (for example, pathology images and tables of patient records) can be loaded together for a specific research.

The integrated data should be offered with a secured access due to the sensitive nature of medical data. API gateways such as Tyk[2], Kong[3], and API Umbrella[4] offer protected access to the APIs, with authentication and authorization to the APIs that access the underlying data. By offering a service-based

[2] https://tyk.io/
[3] https://getkong.org/
[4] https://apiumbrella.io/

data access and integration, data can be protected from direct access, by leveraging such API gateways, in contrast to offering a direct access to the integrated data repository.

3 *Óbidos* Data Integration System

Óbidos offers an on-demand data integration process for data in a hierarchical structure. It comprises of a scalable storage: i) quicker access to metadata stored in-memory, and ii) much larger distributed physical storage for the materialized binary data. *Óbidos* comprises a set of workflows that are initiated by invoking its APIs.

3.1 *Óbidos* Distributed Storage and Deployment

Óbidos offers a hybrid of materialized and virtual data integration. It stores previously accessed and loaded data of interest in an integrated data repository in Apache Hadoop Distributed File System (HDFS) [12] for subsequent accesses. HDFS is used to store the integrated data, due to its scalability and support for storing unstructured and semi-structured, binary and textual data. It eagerly loads only the metadata for the objects of the chosen subsets of data at a specified granularity, into an In-Memory Data Grid of Infinispan [13]. For example, DICOM is structured from collections, patients, studies, series, to images - from coarse to fine granularity. Hence, if a specific study is chosen, all the metadata associated to the study will be loaded, indicating that all the series and images belonging to the chosen study are of interest to the tenant. The integrated data is shared through public identifiers, among tenants inside or between data consumers, via a web services interface. Furthermore, *Óbidos* offers an efficient and fast access to the integrated and virtually integrated data repository in SQL queries.

The metadata of the binary data in HDFS is stored in tables hosted in Apache Hive [14] metastore based on HDFS. Hive tables are indexed for efficient user queries to identify and locate the binary data in the HDFS integrated data repository which is typically unstructured. We call this metadata that operates as an index to the unstructured storage of HDFS, **"metametadata"** to differentiate it from the metadata of larger scope that is stored in the virtual data integration layer (as this indexes data and metadata loaded from the sources). Various storages are unified and accessed seamlessly by leveraging Apache Drill [15], as it enables SQL queries on structured, semi-structured, and unstructured data.

Figure 1 illustrates the deployment landscape of data consumers and data sources. Each data consumer consists of a distinct *Óbidos* deployment. Each of the *Óbidos* deployment has multiple tenants accessing, loading, and integrating data. Tenants of *Óbidos* data consumers access data from the data sources to build their study-specific integrated data repository. When a tenant searches or queries the data sources, a Universal Unique Identifier (UUID) is created to point to the subsets of data chosen/accessed by the tenant, spanning the multiple

data sources. These pointers are generated by appending a locally created unique identifier to the public IP address or Uniform Resource Identifier (URI) of the *Óbidos* deployment server. We call these pointers **"replica sets"**. Replica sets can be shared by the tenant that who created them among the other tenants of the same deployment, or to other *Óbidos* data consumer deployments. Hence, the relevant research data can be shared without copying the actual binary data or metadata.

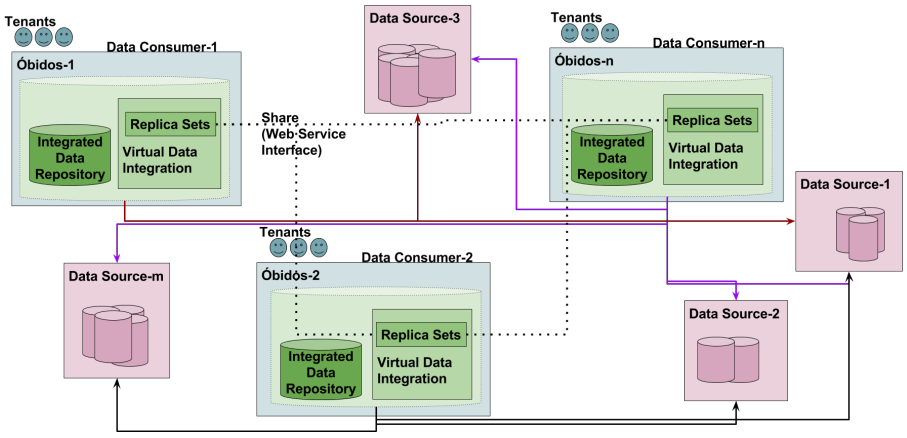

Fig. 1. Landscape of *Óbidos* data consumer deployments, along with the data sources

Data sharing in *Óbidos* works like the Dropbox shared links; no actual data is copied and shared. Only a link that points to the data in data sources, is shared - either directly through the *Óbidos* web service API invocations or outside *Óbidos* (for example, sharing the replica set through other communication media such as email). Once a replica set shared by the tenant is accessed by a remote tenant, the replica set will be resolved into pointers to the actual data sources. The relevant data pointed by the shared replica set can later be loaded by the remote tenants to their deployment. In addition to one-to-one sharing of data between two tenants of the same or different *Óbidos* deployments (deployed in two research institutes), a tenant may opt to create a few replica sets with public access. These public replica sets can be searched and accessed through a public web service API, as they are indexed and stored accordingly.

Tenants can access the data residing in the integrated data repository of their data consumer deployment. Hence, if a replica set is shared among the tenants in the same deployment, it is resolved locally. If the data exists in the integrated data repository, the tenants may access it directly; otherwise, the data is loaded for future accesses.

Óbidos *Virtual Proxies:* The virtual data integration space consists of virtual proxies of metadata. The virtual proxy of an object is essentially a partially loaded metadata of the respective data object. The virtual proxies function as a value holder for the actual metadata, and have sufficient minimal information on metadata for the search query. If only a fraction of metadata is relevant for a search query, that fraction is sufficient as the virtual proxy of the metadata, especially when the metadata itself is of a complex or hierarchical data structure or is large in volume.

When a virtual proxy is accessed by a query, the respective metadata is loaded from the data sources, if it does not resolve locally (means, the complete metadata object is not present yet in the *Óbidos* deployment). This incremental replication approach is motivated by our previous work [16] on achieving efficient replication of larger objects in a wide area network (WAN). This enables *Óbidos* efficient loading of complex metadata faster than the lazy ETL approaches.

Near Duplicate Detection: As the data comes from multiple sources there are possibilities for duplicates. A distributed near duplicate detection algorithm is executed on the metadata periodically, or on-demand, to detect the matches from the heterogeneous sources in building up the integrated data repository. The detected near duplicate pairs are stored in a separate data source, which may be leveraged to improve the accuracy of the detected duplicates. This further allows fixing false positives of detected duplicate pairs in a latter iteration. *Óbidos* compares the timestamps from the metadata of the relevant subsets of data in the data sources against that of the loaded data. Only the updates are loaded when a data set has already been loaded to the *Óbidos* integrated data repository. While previously loaded binary data can be compared with the corresponding data from the source to identify or confirm corruption, typically, the textual metadata is used for the duplicate detection. We exploit our previous work [11] to distribute classic near duplicate detection algorithms such as PPJoin and PPJoin+ [17] in a cluster, to execute and detect near duplicates among the metadata. Thus, we identify duplicates among the respective loaded and source data.

3.2 *Óbidos* Software Architecture

Óbidos is used by i) the tenants (the regular users), ii) administrators that deploy and configure *Óbidos*, and iii) tenants of other *Óbidos* deployments who have a limited web service-based access. Figure 2 depicts a high-level solution architecture of *Óbidos*, representing any *Óbidos* data consumer deployment depicted in Figure 1. Through its northbound, eastbound, southbound, and westbound APIs, *Óbidos* offers capabilities of data access and integration, configuration and management, storage integration, and data sources federation, respectively.

The **Virtual Data Integration Layer** stores the virtually integrated metadata in-memory in an Infinispan data grid. *Óbidos* **Integrated Data Repository** consists of the actual storage behind the **Materialized Data Integration Layer**. When a data item or set that has not been loaded yet into the integrated data repository is accessed, *Óbidos* retrieves the virtual proxies of the relevant

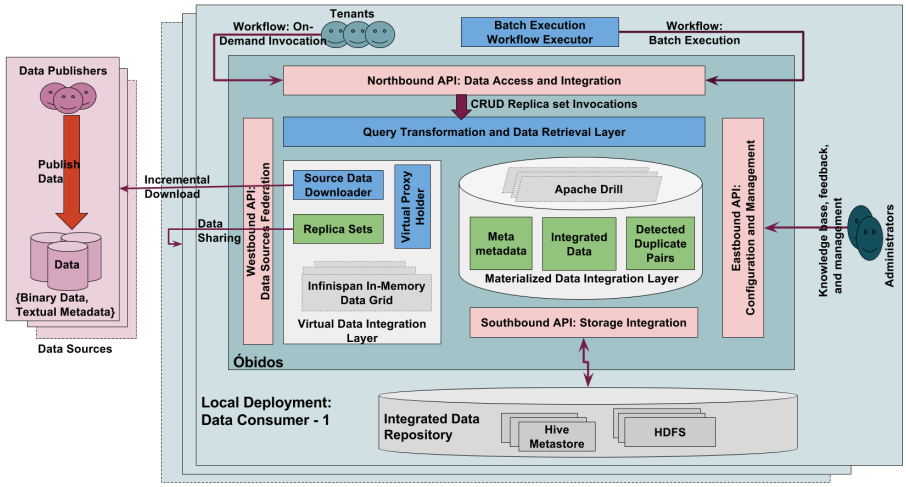

Fig. 2. *Óbidos* Architecture: APIs, Data Integration Layers, and the Integrated Data Repository

data in the virtual data integration layer. If the relevant virtual proxies do not exist or if they are insufficient to offer the results for the query, the data is loaded from the data sources. The integrated data is accessed through a distributed high performance **Query Transformation and Data Retrieval Layer** which has Apache Drill as its core query engine. This allows efficient queries to the data, partially or completely loaded into the integrated data repository.

Location, access, and public API details of the data sources are stored by the **Source Data Downloader** of *Óbidos*. Each data set that has been loaded by the tenant is identified by a replica set which functions as a pointer to the actual data that can resolve back to the search query. *Óbidos* bootstrapping follows a lazy loading approach. It retrieves metadata of the identified matching subsets as virtual proxies. The **Virtual Proxy Holder** offers an indexed and hierarchical data structure to map and store all the virtual proxies.

The components and data corresponding to virtual data integration layer reside in the In-Memory Data Grid to offer quick access to the critical and most accessed data, without requiring any disk access. The exact data matching the user query is fetched and stored in a highly scalable integrated data repository, along with the metametadata and detected duplicate pairs. The integrated data repository stores query outputs as well as data loaded by previous queries. Therefore, queries can be regarded as virtual data sets that can be reaccessed and shared (akin to the materialized view in traditional RDBMS).

The **Northbound API** is the user-facing API, invoked by the tenants to access and integrate the data. Invocation of the northbound API initiates various tenant-centric workflows that are discussed further in Section 4.1. Data sources and other data consumers are connected by the **Westbound API** of *Óbidos*.

While the various data sources offer APIs providing access to their data, tenants create virtual data sets and share them with other tenants. Hence, *Óbidos* itself poses as a hybrid integrated data repository. The data download and sharing mechanisms are offered by the westbound API of *Óbidos*. The westbound API is indirectly invoked on-demand by the tenants through the northbound API or through the **batch execution workflow** that periodically checks and downloads the changesets of relevant data in batch.

System management and a feedback loop to improve the outcomes through a knowledge base are handled by the **Eastbound API**. The eastbound API is invoked by the administrators for data management tasks such as cleaning up orphaned data that is not referred by the replica sets or by periodic batch executions such as leveraging the detected duplicate pairs for subsequent iterative data integrations. The physical storage that functions as the integrated data repository for the materialized data integration layer is configured through the **Southbound API** by the administrators. *Óbidos* integrated data repository consists of HDFS and Hive metastore, deployed in a distributed cluster.

4 *Óbidos* Prototype for Medical Research Data

An *Óbidos* prototype has been implemented with Oracle Java 1.8 to integrate multimodal medical data from various heterogeneous data sources including The Cancer Imaging Archive (TCIA) [18], imaging data hosted in Amazon S3 buckets, medical images accessed through caMicroscope[5], clinical and imaging data hosted in local data sources including relational and NoSQL databases, and file system with files and directories along with CSV files as metadata.

4.1 RESTful Interfaces

Óbidos RESTful interface is designed as CRUD (Create, Retrieve, Update, and Delete) functions on replica sets. The tenant finds subsets of data from data sources by invoking ***create*** *replica set*. This creates a replica set and initiates the data loading workflow. When ***retrieve*** *replica set* is invoked, the relevant data is retrieved and *Óbidos* checks for updates from the data sources pointed by the replica set. Metadata of the replica set is compared against that of the data sources for any corruption or local changes. By running a near duplicate detection between, the local metadata representing the current state of the integrated data, and the metametadata stored at the time of data loading, the corrupted or deleted data is identified, redownloaded, and integrated.

The tenant updates an existing replica set to increase, decrease, or alter its scope, by invoking the ***update*** *replica set*. This may thus invoke parts of create and/or delete workflows, as new data may be loaded while existing parts of data may need to be removed. The tenant deletes existing replica sets by invoking the ***delete*** *replica set*. The virtual data integration layer is updated

[5] http://camicroscope.org

immediately to avoid loading updates to the deleted replica sets. While the virtual data integration layer is multi-tenanted with each tenant having their own virtual isolated space, the integrated data repository is shared among them. Hence, before deleting, the data should be confirmed to be an orphan with no replica sets referring to them from any of the tenants. Deleting the data from the integrated data repository is designed as a separate process that is initiated by the administrators through the eastbound API. When the storage is abundantly available in a cluster, *Óbidos* advocates keeping orphan data in the integrated data repository rather than immediately initiating the cleanup process, and repeating it too frequently.

4.2 Data Representation

Óbidos virtual proxy holder consists of a few easy to index and space efficient data structures. *Óbidos* presents the replica sets internally in a minimal tree-like structure. Depending on how deep the structure is traversed, a replica set can be traced down to its virtual proxy by the virtue of its design as discussed below.

A multi-map (named 'tenant multi-map') lists the replica sets for each of the tenant. It stores replica sets belonging to each tenant, with the keys representing the tenant ID and each of the values representing one or more 'replica set maps'. As replica sets include pointers to data sets of interest from various data sources, each replica set map indicates the relevant data sources by each of the replica set, having replica set ID as the key and the list of data source names as the value. Each data source belonging to a replica set is internally represented by n maps. Each of the n maps represents one of the granularities in the data source (hence, $n = 4$ for TCIA), and a boolean array A_b of length n, each element $A_b[i]$ of the array representing the existence of a non-null entry in the i^{th} map of granularity. Thus, the boolean flags in A_b indicate the existence (or lack thereof) of the data set in any granularity. If an entire level of granularity is included in the replica set by the tenant, the relevant flag is set to true.

Figure 3 illustrates the data representation of *Óbidos* and how it stores the replica sets and virtual proxies in a tree-like data structure. The maps representing each granularity resolve to store the identifier of the metadata or a virtual proxy. Hence details of data sets of interest from TCIA imaging metadata is stored in 4 maps - one each for collections, patients, studies, and series.

4.3 Implementation

Apache Velocity 1.7 [19] is leveraged to generate the application templates of the *Óbidos* web interface. Hadoop 2.7.2 and Hive 1.2.0 are used to store the integrated data and the metametadata respectively. Hive-jdbc package writes the metametadata into the Hive metastore. SparkJava 2.5[6] compact Java-based web framework is leveraged to expose the *Óbidos* APIs as RESTful services. The APIs are managed and made available to the relevant users through API gateways.

[6] sparkjava.com/

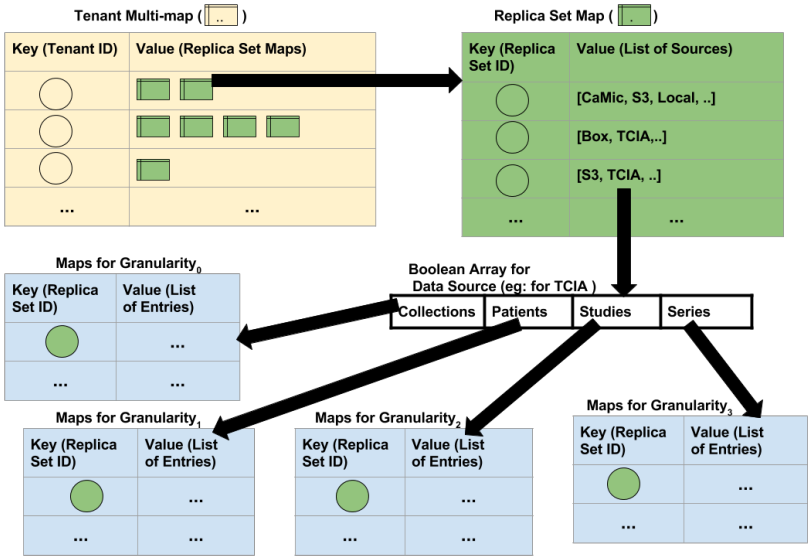

Fig. 3. Data Representation: *Óbidos* Virtual Data Integration Layer

API Umbrella is deployed as the default API gateway. Embedded Tomcat 7.0.34 is used to deploy *Óbidos* with minimal deployment and operational overhead. The Embedded Tomcat-based web application exposes *Óbidos* APIs and offers a simple web-based user interface to the data consumers to access the data sources, and create, modify, and share replica sets. Infinispan 8.2.2 is used as the In-Memory Data Grid where its distributed streams support distributed execution of *Óbidos* workflows across the *Óbidos* clustered deployment. The virtual proxy holder and the replica sets are represented by instances of Infinispan Cache class, which is a Java implementation of distributed HashMap.

Drill 1.7.0 is exploited for the SQL queries on the integrated data repository, with drill-jdbc offering JDBC API to interconnect with Drill from the Query Transformation and Data Retrieval Layer. Drill cannot query nested arrays inside data in unstructured formats such as JSON (or NoSQL data sources generated from them), unless its $flatten$ function is used. However, $flatten$ alters the data structure and flattens the objects in the query results to make multiple entries instead of a single object with a nested array. This may be unfit for the requirements of a data consumer. We also observed $flatten$ to be slow on large nested arrays. Hence, when this situation is expected, *Óbidos* deployment is configured by the administrators to either i) avoid Drill and query the data integration layer directly from the query transformation and data retrieval layer, or ii) configure the southbound such that the nested arrays are converted into an array of maps at the time of data loading into the integrated data repository.

5 Evaluation

Óbidos has been benchmarked against implementations of eager ETL and lazy ETL approaches, using microbenchmarks derived from medical research queries on cancer imaging and clinical data. While *Óbidos* can integrate textual and binary (based on their metadata) data efficiently, we limit our focus to DICOM images to benchmark against the traditional data loading and sharing approaches in a fair manner.

The DICOM imaging data used in the evaluation consists of collections of various volume, as shown by Figure 4. The data consists of large scale binary images (in the scale of a few thousands GB, up to 10 000 GB) along with a smaller scale textual metadata (in the range of MBs).

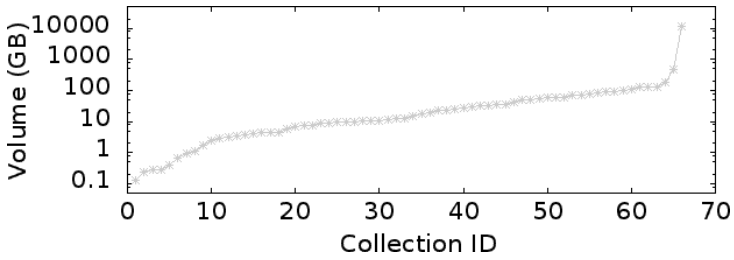

Fig. 4. Evaluated DICOM Imaging Collections (Sorted by Total Volume)

Various granularities of DICOM are leveraged in the evaluations. Figure 5 illustrates the number of patients, studies, series, and images in each of the collection. Collections are sorted according to their total volume. Each collection consists of multiple patients; each patient has one or more studies; each study has one or more series; and each series has multiple images. Figure 5 indicates that the total volume of a collection does not necessarily reflect the number of entries in it.

5.1 Loading and Querying Data

Óbidos was benchmarked for its performance and efficiency in loading the data and average query time from the integrated data repository. Initially, to avoid the influence of bandwidth or transfer speed in the evaluation, data sources were replicated to a local cluster in the first two iterations, illustrated by Figure 6 and 7. Figure 6 shows the loading time for different total volume of data sources for the same data sets of interest of the tenant. Figure 7 shows the time taken for the same total volume of data sources while increasing the number of studies of interest.

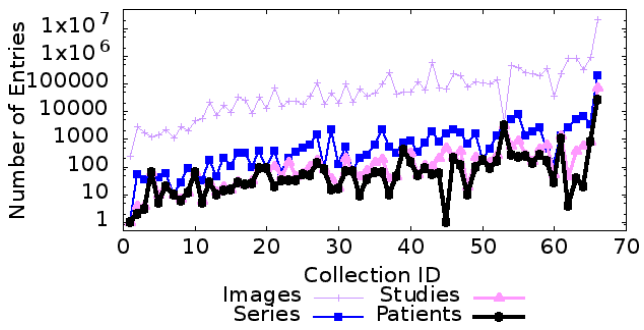

Fig. 5. Various Entries in Evaluated Collections (Sorted by Total Volume)

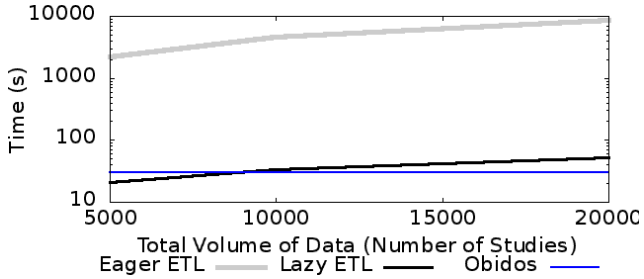

Fig. 6. Data load time: change in total volume of data sources (Same query and same data set of interest)

Finally, data was loaded from the remote data sources through their web service APIs, to evaluate the influence of data downloading and bandwidth consumption associated with it. The total volume of data was changed, while keeping the data sets of interest constant. Figure 8 shows the time taken for *Óbidos* and the other approaches. Eager ETL performed poor as binary data had to be downloaded over the network. Lazy ETL too performed slow for large volumes as it must eagerly load the metadata (which itself grows with scale) over the network.

Since *Óbidos* loads the metadata of only the data sets of interest, the loading time remains constant for the experiments as illustrated by Figure 6 and Figure 8, where it increases with the number of studies of interest in the experiment depicted by Figure 7. As only the data sets of interest are loaded, *Óbidos* uses bandwidth conservatively, loading no irrelevant data or metadata.

Query completion time depends on the number of entries in the queried data rather than the size of the entire integrated data repository. Hence, varying amounts of data, measured by the number of studies, were queried. The query completion time is depicted in Figure 9. The sample queries used are made sure to be satisfied by the data that have already been loaded, in case of lazy ETL

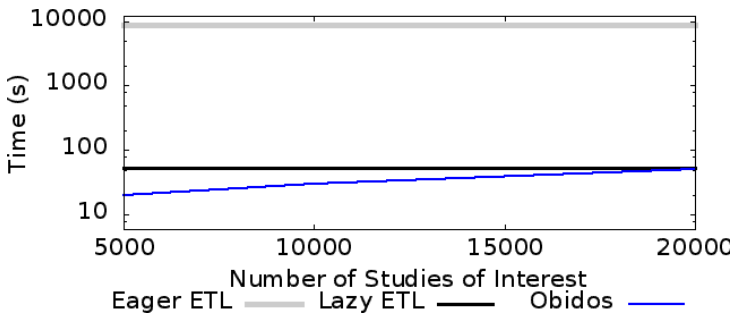

Fig. 7. Data load time: varying number of studies of interest (same query and constant total data volume)

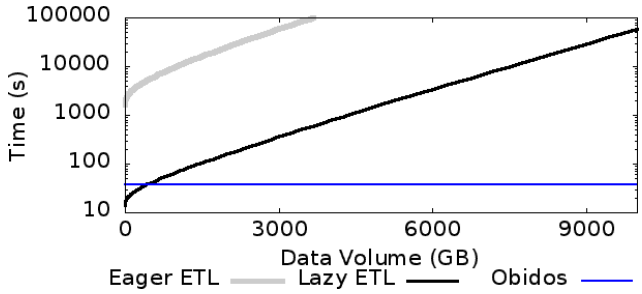

Fig. 8. Load time from the remote data sources

and *Óbidos*. In the sample deployments, eager ETL and lazy ETL offered similar performance for various queries. *Óbidos* showed a speedup compared to the lazy ETL, which can be attributed to the efficiency of the query transformation and data retrieval layer. The unstructured data in HDFS was very efficiently queried as in a relational database through the distributed query execution of Drill with its SQL support for NoSQL data sources. Eager ETL would outperform both lazy ETL and *Óbidos* for queries that access data not yet loaded (in lazy ETL and *Óbidos*), as eager ETL would have constructed an entire data warehouse beforehand. *Óbidos* attempts to minimize this by leveraging the virtual proxies in addition to the metadata. Furthermore, with the domain knowledge of the medical data researcher, the relevant subsets of data are loaded timely. The time required to construct such a data warehouse would prevent any benefits of eager loading from being prominent. If data is also not loaded beforehand in eager ETL, it would consume much longer to construct the entire data warehouse before starting the query. Moreover, loading everything beforehand is irrelevant and impractical or even impossible for study-specific researches due to the scale and distribution of sources.

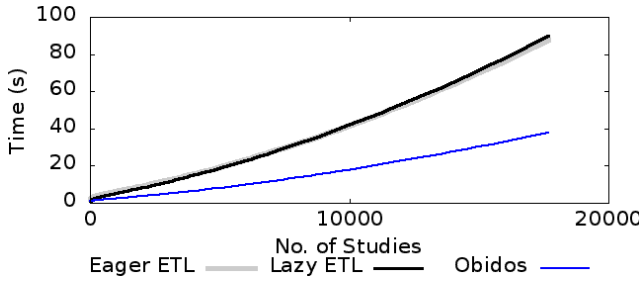

Fig. 9. Query completion time for the integrated data repository

Overall, in all the relevant use cases, lazy ETL and *Óbidos* significantly outperformed eager ETL as the need to build a complete data warehouse is avoided in them. As *Óbidos* loads only the subset of metadata, and does not even eagerly load the metadata unlike eager ETL, for large volumes, *Óbidos* also significantly outperformed lazy ETL. While eager ETL performs better for data that has not been loaded prior by *Óbidos* and lazy ETL, *Óbidos* assumes that the researcher is aware of the data to load the data sets of interest appropriately.

5.2 Sharing Efficiency of Medical Research Data

Various image series of an average uniform size are shared between tenants inside an *Óbidos* deployment and between remote deployments. Figure 10 benchmarks the data shared in these *Óbidos* data sharing approaches against the typical binary data transfers for its bandwidth efficiency. Inside *Óbidos* deployment, regardless of the size of data sets to be shared, only an identifier to the replica set, the replica set ID, is shared. The ID is interpreted directly by the receiving tenant from *Óbidos*. Hence, the shared data is of a constant size. When data is shared between multiple *Óbidos* deployments, replica sets are shared. Replica sets are minimal in size as pointers to actual data. However, they grow linearly when more data of same granularity is shared. Minimal data was shared in both cases compared to sharing actual data.

This also avoids the need for manually sharing the locations of the data sets, which is an alternative bandwidth-efficient approach to sharing the integrated data. As the pointers are shared, no actual data is copied and shared. This enables data sharing with zero redundancy.

6 Related Work

Virtual data integration and service-based data integration approaches are two major motivations for *Óbidos*. *Óbidos* leverages both approaches together for an enhanced functionality and performance.

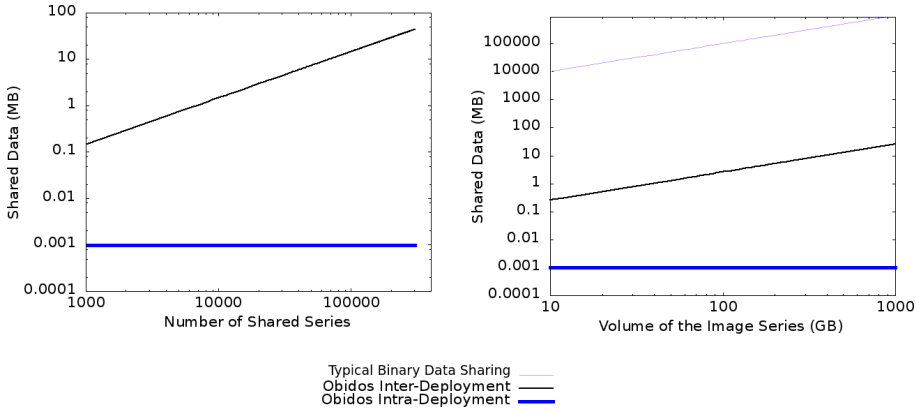

Fig. 10. Actual data shared in *Óbidos* data sharing use cases vs. in regular binary data sharing

Service-Based Data Integration: OGSA-DAI (Open Grid Services Architecture - Data Access and Integration) [20] facilitates federation and management of various data sources through its web service interface. The Vienna Cloud Environment (VCE) [21] offers service-based data integration of clinical trials and consolidates data from distributed sources. VCE offers data services to query individual data sources and to provide an integrated schema atop the individual data sets. This is similar to *Óbidos*, though *Óbidos* leverages virtual data integration for an efficient federation and sharing of data and avoids replicating data excessively.

EUDAT [22] is a platform to store, share, and access multidisciplinary scientific research data. EUDAT hosts a service-based data access feature B2FIND [23], and a sharing feature B2SHARE [24]. When researchers access these cross-disciplinary research data sources, they already know which of the repositories they are interested in, or can find them by the search feature. Similar to the motivation of *Óbidos*, loading the entire data from all the sources is irrelevant in EUDAT. Hence, choosing and loading certain sets of data is supported by these service-based data access platforms. *Óbidos* can be leveraged to load related cross-disciplinary data from the eScience data sources such as EUDAT.

Heterogeneity in data and sources, and the distribution of data sources, have imposed various challenges to the data integration. Multi-agent systems have been proposed for data integration in wireless sensor networks [25]. *Óbidos* approach can be used in conjunction with multi-agent systems where metadata can be loaded instead of an eager loading of data, while still caching the data loaded by the distributed agents.

Lazy ETL: Lazy ETL [6] demonstrates how metadata can be efficiently used for study-specific queries without actually constructing an entire data warehouse beforehand, by using files in SEED [26] standard format for seismological research.

The hierarchical structure and metadata of SEED is similar to that of DICOM medical imaging data files that are accessed by the *Óbidos* prototype. Thus, we note that while we prototype *Óbidos* for medical research, the approach is applicable to various research and application domains.

LigDB [27] is similar to *Óbidos* as both focus on a query-based integration approach as opposed to having a data warehouse constructed as the first step, and it efficiently handles unstructured data with no schema. However, *Óbidos* differs as it indeed has a scalable storage, and does not periodically evict the stored data unlike LigDB.

Medical Data Integration: Leveraging Hadoop ecosystem for management and integration of medical data is not entirely new, and our choices are indeed motivated by previous work [28]. However, the existing approaches failed to extend the scalable architecture offered by Hadoop and the other big data platforms to create an index to the unstructured integrated data, manage the data in-memory for quicker data manipulations, and share results and data sets efficiently with peers. *Óbidos* attempts to address these shortcomings with its novel approach and architecture, combining service-based big data integration with virtual data integration, designed for research collaboration.

Our previous work, Data Café [29] offers a dynamic warehousing platform for medical research data integration. It lets the medical big data researchers to construct a data warehouse, integrating data from heterogeneous data sources, consisting of unstructured or semi-structured data, even without knowing the data schema a priori. *Óbidos* leverages and extends Data Café in materialized data integration layer to store the loaded binary data. *Óbidos* also shares the (Apache Hive and Drill-based) data query architecture of Data Café to efficiently index and query the integrated data repository.

7 Conclusion and Future Work

Óbidos offers service-based big data integration to ensure fast and resource efficient data analysis. It stores the loaded binary data in a distributed and scalable integrated data repository, and the metadata representing a larger set of data in an in-memory cluster. By implementing and evaluating *Óbidos* for medical research data, we demonstrated how *Óbidos* offers a hybrid data integration solution for big data, and lets the researchers share data sets with zero redundancy.

We built our case on the reality that data sources are proliferating, and cross-disciplinary researches such as medical data research often require access and integration of data sets spanning across the multiple data sources in the Internet. We further presented how a service-based data integration and access approach fits well for the requirement to have a protected access of sensitive data by various researchers. We leveraged the respective APIs offered by the data sources in consuming and loading the data, while offering our own service-based access to the *Óbidos* integrated data repository. We further envisioned that various distributed deployments of *Óbidos* will be able to collaborate and

coordinate to construct and share study-specific integrated data sets internally and between one another.

As a future work, we aim to deploy *Óbidos* approach to consume data from various research repositories such as EUDAT to find and integrate research data. Thus, we will be able to conduct a usability evaluation of *Óbidos* based on different use cases and data sources. We also propose to leverage the network proximity among the data sources and the data consumer deployments for efficient data integration and sharing, in the future work. Thus, we aim to build virtual distributed data warehouses - data partially replicated and shared across various research institutes.

Acknowledgements: This work was supported by NCI U01 [1U01CA187013-01], Resources for development and validation of Radiomic Analyses & Adaptive Therapy, Fred Prior, Ashish Sharma (UAMS, Emory), national funds through Fundação para a Ciência e a Tecnologia with reference UID/CEC/50021/2013, PTDC/EEI-SCR/6945/2014, a Google Summer of Code project, and a PhD grant offered by the Erasmus Mundus Joint Doctorate in Distributed Computing (EMJD-DC).

References

1. Lee, G., Doyle, S., Monaco, J., Madabhushi, A., Feldman, M.D., Master, S.R., Tomaszewski, J.E.: A knowledge representation framework for integration, classification of multi-scale imaging and non-imaging data: Preliminary results in predicting prostate cancer recurrence by fusing mass spectrometry and histology. In: 2009 IEEE International Symposium on Biomedical Imaging: From Nano to Macro, IEEE (2009) 77–80
2. Huang, Z.: Data Integration For Urban Transport Planning. Citeseer (2003)
3. Sujansky, W.: Heterogeneous database integration in biomedicine. Journal of biomedical informatics **34**(4) (2001) 285–298
4. Mildenberger, P., Eichelberg, M., Martin, E.: Introduction to the dicom standard. European radiology **12**(4) (2002) 920–927
5. Whitcher, B., Schmid, V.J., Thornton, A.: Working with the dicom and nifti data standards in r. Journal of Statistical Software **44**(6) (2011) 1–28
6. Kargín, Y., Ivanova, M., Zhang, Y., Manegold, S., Kersten, M.: Lazy ETL in action: ETL technology dates scientific data. Proceedings of the VLDB Endowment **6**(12) (2013) 1286–1289
7. Dong, X.L., Srivastava, D.: Big data integration. In: Data Engineering (ICDE), 2013 IEEE 29th International Conference on, IEEE (2013) 1245–1248
8. Rustagi, A.: Parallel processing for ETL processes (2007) US Patent App. 11/682,815.
9. Porter, D.L., Swanholm, D.E.: Distributed extract, transfer, and load (ETL) computer method (2006) US Patent 7,051,334.
10. Rimal, B.P., Choi, E., Lumb, I.: A taxonomy and survey of cloud computing systems. INC, IMS and IDC (2009) 44–51
11. Kathiravelu, P., Galhardas, H., Veiga, L.: $\partial u \partial u$ multi-tenanted framework: Distributed near duplicate detection for big data. In: OTM Confederated International Conferences" On the Move to Meaningful Internet Systems", Springer (2015) 237–256
12. White, T.: Hadoop: The definitive guide. " O'Reilly Media, Inc." (2012)
13. Marchioni, F., Surtani, M.: Infinispan data grid platform. Packt Publishing Ltd (2012)

14. Thusoo, A., Sarma, J.S., Jain, N., Shao, Z., Chakka, P., Anthony, S., Liu, H., Wyckoff, P., Murthy, R.: Hive: a warehousing solution over a map-reduce framework. Proceedings of the VLDB Endowment **2**(2) (2009) 1626–1629
15. Hausenblas, M., Nadeau, J.: Apache drill: interactive ad-hoc analysis at scale. Big Data **1**(2) (2013) 100–104
16. Veiga, L., Ferreira, P.: Incremental replication for mobility support in obiwan. In: Distributed Computing Systems, 2002. Proceedings. 22nd International Conference on, IEEE (2002) 249–256
17. Xiao, C., Wang, W., Lin, X., Yu, J.X., Wang, G.: Efficient similarity joins for near-duplicate detection. ACM Transactions on Database Systems (TODS) **36**(3) (2011) 15
18. Clark, K., Vendt, B., Smith, K., Freymann, J., Kirby, J., Koppel, P., Moore, S., Phillips, S., Maffitt, D., Pringle, M., et al.: The cancer imaging archive (tcia): maintaining and operating a public information repository. Journal of digital imaging **26**(6) (2013) 1045–1057
19. Gradecki, J.D., Cole, J.: Mastering Apache Velocity. John Wiley & Sons (2003)
20. Antonioletti, M., Atkinson, M., Baxter, R., Borley, A., Chue Hong, N.P., Collins, B., Hardman, N., Hume, A.C., Knox, A., Jackson, M., et al.: The design and implementation of grid database services in ogsa-dai. Concurrency and Computation: Practice and Experience **17**(2-4) (2005) 357–376
21. Borckholder, C., Heinzel, A., Kaniovskyi, Y., Benkner, S., Lukas, A., Mayer, B.: A generic, service-based data integration framework applied to linking drugs & clinical trials. Procedia Computer Science **23** (2013) 24–35
22. Lecarpentier, D., Wittenburg, P., Elbers, W., Michelini, A., Kanso, R., Coveney, P., Baxter, R.: Eudat: A new cross-disciplinary data infrastructure for science. International Journal of Digital Curation **8**(1) (2013) 279–287
23. Widmann, H., Thiemann, H.: Eudat b2find: A cross-discipline metadata service and discovery portal. In: EGU General Assembly Conference Abstracts. Volume 18. (2016) 8562
24. Ardestani, S.B., Håkansson, C.J., Laure, E., Livenson, I., Stranák, P., Dima, E., Blommesteijn, D., van de Sanden, M.: B2share: An open escience data sharing platform. In: e-Science (e-Science), 2015 IEEE 11th International Conference on, IEEE (2015) 448–453
25. Qi, H., Iyengar, S., Chakrabarty, K.: Multiresolution data integration using mobile agents in distributed sensor networks. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) **31**(3) (2001) 383–391
26. Ahern, T., Casey, R., Barnes, D., Benson, R., Knight, T.: Seed standard for the exchange of earthquake data reference manual format version 2.4. Incorporated Research Institutions for Seismology (IRIS), Seattle (2007)
27. Milchevski, E., Michel, S.: ligdb-online query processing without (almost) any storage. In: EDBT. (2015) 683–688
28. Lyu, D.M., Tian, Y., Wang, Y., Tong, D.Y., Yin, W.W., Li, J.S.: Design and implementation of clinical data integration and management system based on hadoop platform. In: Information Technology in Medicine and Education (ITME), 2015 7th International Conference on, IEEE (2015) 76–79
29. Kathiravelu, P., Sharma, A.: A dynamic data warehousing platform for creating and accessing biomedical data lakes. In: VLDB Workshop on Data Management and Analytics for Medicine and Healthcare, Springer (2016) 101–120