International Journal of Knowledge Based Computer Systems
8 (1 & 2) June & December 2020, 11-16
http://www.publishingindia.com/ijkbcs/

A Survey on the Evolution of Models of Data Integration

Shashank Shrestha[1*] and Subhash Bhalla[2]
[1]Department of Computer and Information Systems, University of Aizu, Aizu-Wakamatsu, Japan.
Email: d8201104@u-aizu.ac.jp
[2]Department of Computer and Information Systems, University of Aizu, Aizu-Wakamatsu, Japan.
Email: bhalla@u-aizu.ac.jp
*Corresponding Author

Abstract: From time to time there have been different models of data integration to manage and analyze data. Also with the emergence of big data, the database community has proposed newer and better solutions to manage such disparate and large data. Also, the changes in the data storage models and massive data repositories on the web have encouraged the need for novel data integration models. In this article, we try to present a case of various trends in integrating data through different models. We present a brief overview of Federated Database Systems, Data Warehouse, Mediators and new proposed Polystore Systems with the evolution of architecture, query processing, distribution, automation and data models supported within those data integration models. The similarities and differences of these models are also presented. Also, the novelty of Polystore Systems with various examples is discussed. This article also highlights the importance of such system for integrating large scale heterogeneous data.

Keywords: Data integration, Multi-database systems, Polystore systems.

I. Introduction

There is a large volume of data on the web which are non- or semi-structured. The data are provided through different scientific, organizational, institutional, and governmental repositories. These data need to be shared, exchanged and integrated. The data on the web usually have catalogs and libraries where there are links to access the data (URL or HTTP). These type of data are called Linked Data [1]. These data require integration to access information through multiple data stores.

Over the past few decades, improvements in the variety of data have given rise to comprehensive research on the management of heterogeneous data. There are problems with the data sets available in various scientific fields in combining data with different models. Integration of data is a method of combining multi-source data. Over all the underlying sources, it should have a single view. Generally, it has two types of architecture. Data Warehousing is a type of physical model where data from multiple sources are copied and stored in a warehouse [2]. Other type of architecture is the virtual architecture. The virtual architecture basically comprises of one of the following- Federated Database Systems (FDBS) [3], Mediation [4] and newly proposed Polystores [5].

The virtual architecture focuses mainly on the management of heterogeneous data with multiple sources/stores. The general term for the virtual architecture is Multi-Database Systems (MDBS). Since it proposes solution to manage heterogeneous data, MDBS can also be called Heterogeneous Database Systems. MDBS is designed to offer the same functionality across various operating systems, data formats, and languages for queries. Database management systems (DBMS) running on heterogeneous computing platforms are usually managed by databases. Information sharing across various channels must also be provided by the MDBSs.

Usually, databases are under different and autonomous supervision. In order to separate themselves from the distributed database architecture, the sources underlying the MDBS must have autonomy. There are various facets of autonomy that must be addressed by MDBS. They are classified as autonomy of design, autonomy of communication, autonomy of implementation and autonomy of associations. The balance between autonomy and heterogeneity [3] is another aspect that must be considered.

- *Design Autonomy:* Design Autonomy is the capacity, regardless of data, query language or conceptualization, to choose the design.
- *Communication Autonomy:* The autonomy of communication is the general operation of DBMSs to communicate with or not with other DBMSs.
- *Execution Autonomy:* Execution Autonomy is the permission of the DBMS component to manage the operations needed by local and external operations.

- *Association Autonomy:* Association Autonomy is the ability of the DBS part to disassociate itself and act as a single DBS from a federation.

The different types of autonomy help give rise to heterogeneity problems while managing MDBS. The heterogeneity in MDBS arise mainly due to design autonomy.

The interoperation or the balance between heterogeneous databases should focus on providing transparency across the data sources. Users should be able to access different databases in the same way as accessing a single database. This requirement is called Distribution transparency. The other requirement is Heterogeneity transparency where the users should be able to access other schemas in the same way as the access to local database using a familiar model and language. The requirement of giving access with heterogeneity transparency for the users can be a daunting task. The task is problematic since there are different types of heterogeneity problems that the database administrator/developers have to address. Source type problem arise when there are different data stores storing different data. Other problem may arise when different stores use different communication protocols. For example, some systems have web interface whereas others use programmable interfaces (APIs). Sometimes the structure of the tables (schema) storing data can be different. Other similar problems arise when type of data or value is different.

The past models of data integration have provided different solutions to handle disparate data. However, the lack of providing a uniform/single query language from multiple heterogeneous data sources have limited their scope. We can fairly assume that the different approaches proposed by past models of data integration are now failing to manage heterogeneous data with the emergence of big data in recent years. One of the main reasons they have failed is that, there is increased interest in the database community on managing the new forms of disparate data. The "one size fits all" approach adopted by the past models are no longer relevant in modern day database engineering [6]. Also, all the underlying DBMSs require a complete autonomous functionality for better query optimization.

Current trends in database management require the use of multiple models and data stores. The old FDBS and data warehouse architecture works fine for relational data but managing variety of data (arrays, graphs, and images etc.) becomes impossible. Different data stores managing variety of data have their own local language. Polystore architecture allows the underlying DBMSs to create a communication protocol via island of information where the information from each data store is federated which generates a single query language understandable by the query system [7]. Also, location transparency must be preserved by the middleware supporting any particular island. This middleware can be an API, mediator or wrapper.

The article will present a brief overview of various models of data integration in Section II. In Section III, a short evaluation of the models with similarities and differences are explained. The article finally summarizes and concludes in Section IV.

II. Models of Data Integration

A DBMS can be categorized as either centralized or distributed. A centralized framework maintains a single database, while multiple databases are operated by distributed systems. In a DBMS, a component DBS can be centralized or distributed. Depending on the autonomy of the component DBS as federated and non-federated, a multiple DBS (MDBS) can be divided into two groups. An integration of part DBMS that is not autonomous is a non-federated database system. In order to allow partial and managed sharing of their data, a federated database system consists of component DBS that are independent but participate in a federation. Based on the levels of integration with the component database systems and the scope of services provided by the federation, federated architectures vary. The FDBS can be classified as systems that are loosely or closely coupled.

- *Loosely Coupled:* Component databases are allowed by Loosely Coupled to create their own federated schema. Using a multi-domain language, a user can usually access other component database systems, but this eliminates some levels of location confidentiality, requiring the user to have direct knowledge of the federated schema. A consumer imports from other component databases the data they need and combines it with their own to create a federated schema.
- *Tightly Coupled:* Tightly Coupled system consists of component systems that use independent processes to construct and publicize an integrated federated schema.

Multiple DBS can be defined by three dimensions: distribution, heterogeneity and autonomy, of which FDBS is a particular form. Another characterization may be based on the networking dimension, such as single databases or multiple LAN or WAN databases. Over the years there have been numerous architectures and models for incorporating heterogeneous data. The size, growth and varieties have increased with the advent of big data. Thus several new database techniques and architectures for incorporating such data have arisen to deal with this transition. This section specifies certain models and introduces them.

A. Federated Database Systems (FDBS)

FDBS is a collection of databases of co-operating but autonomous components. FDBS enables each local DB, where the control is decentralized, to have more control over the shareable information. It is necessary to complete the integration of FDBS, but it depends on users' needs.

A Federated Database Management System handles FDBS (FDBMS). FDBMS is a middleware that runs on top of many local DBMSs and offers a seamless interface with separately developed DBMS schemas for disparate systems.

As shown in Fig. 1, FDBS has fundamental device components of the data management architecture. The processor, command, data, schema, mapping of information and database are the components. These components help build the FDBS schema, which is commonly referred to as the Five-level FDBS schema architecture. FDBS's five-level schema architecture incorporates the following:

- *Local Schema:* The conceptual definition embodied in the primary data model of the component DBMS is the local Schema.
- *Component Schema:* By converting the local schema into a format called the canonical data model or common data model, the component schema is derived. They are useful when the part integrates semantics that are missed in the local schema. They assist in data integration for closely coupled FDBS.
- *Export Schema:* The export schema is a subset of a schema component accessible to the FDBS. Access control details about its use by individual users of the federation may be included. The export schema assists in the management of the data control flow.
- *Federated Schema:* The aggregation of several export schemes is a federated schema. It includes data distribution information that is generated when export schemes are integrated.
- *External Schema:* The External Schema specifies a schema for a user/app or a user/app class. While accurately reflecting the state of the art in data integration, the above Five Level Schema Architecture suffers from a major disadvantage, namely the look and feel imposed by IT. Modern users of data need control over how data is presented; their interests are somewhat in conflict with such bottom-up data integration approaches.

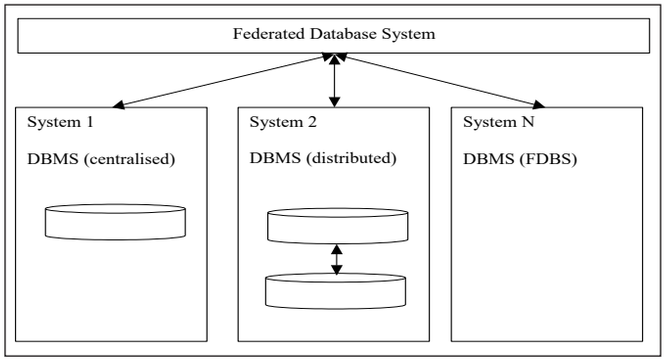

Fig. 1: FDBS Architecture

FDBS provides processors that are required for information mapping between the underlying data sources. It is required for transforming internal command language to a local query language, access the control specified in export schema to limit allowable operations submitted to corresponding component schemas and for the query decomposition and data merging. These processes are called transforming, filtering and constructing respectively.

B. Data Warehouse(DW)

A data warehouse is a federated repository within an organizational context for all data generated by the different operating structures. DW gathers data for transactional processing rather than from multiple sources of analysis and access. [8] Typically, a data warehouse is a relational database located on the mainframe or cloud of an enterprise. On-Line Transaction Processing (OLTP) allows the organization for business intelligence activities and decision support. Data warehouses are typically defined by four basic components.

- *Subject Oriented:* Data is defined by subject matter such as sales, customer etc. which helps to analyze the data efficiently.
- *Integrated:* Data is gathered from multiple sources and all data in the warehouse must be compatible with each other regardless of type or location.
- *Nonvolatile:* Nonvolatile means that, once entered into the warehouse, data should not change.
- *Time Variant:* All data contains a reference to time so that the age of each piece of data can be determined.

As data warehouse is the most common and popular method of data integration; various architectural models have been proposed overtime. The general DW architecture focus on business analysis, models (virtual warehouse, data mart and enterprise warehouse), load manager, query manager, warehouse manager and detailed information. The business or organization must choose the best architecture according to their business requirements.

The basic data warehouse architecture shown in Fig. 2 comprises of data sources, a central warehouse and a platform where the users can use the data for analysis, reporting and/or mining. The basic data warehouse architecture can be customized for different groups in the organization. With the addition of data marts, the systems can be designed for particular line of business. Moreover, the architecture can also include On-Line Analytical Processing (OLAP) engine for purposes of querying, data collection, reporting and mining [9]. OLAP conducts multidimensional business data analysis and has the ability to complicate measurements, analysis of patterns and advanced data modeling.

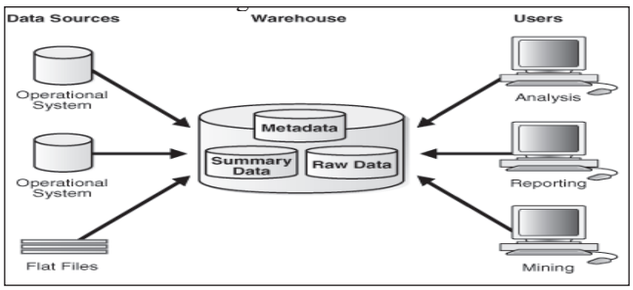

Fig. 2: Data Warehouse Architecture

C. Mediators

Mediator is a virtual view over the data. The data is only stored at the sources. Mediator has a virtual schema which combines all the schemas from the sources. The Mediator provides metadata defining the integration schema and the external schemas of each data resource of the federation.

Mediator architecture also uses data mapping for querying and matching the data sources. The mapping and querying is achieved through the implementation of wrappers. The general mediator architecture is shown in Fig. 3.

Mediator architecture allows system to map a user query to other multiple queries from different data sources. This function is called data mapping [10]. Whenever a query is generated, it is sent to its source. The data source evaluates the query and return the results. Finally, the results are merged together and passed to the end-user.

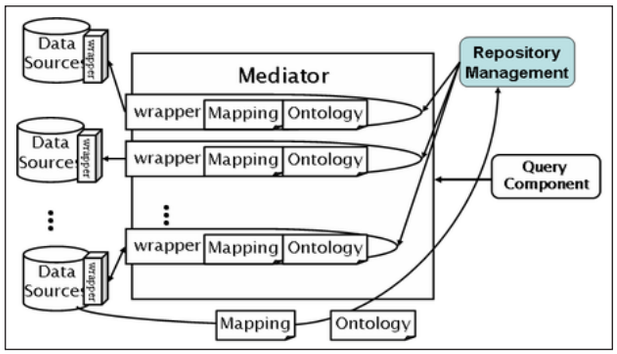

Fig. 3: Mediator Architecture

D. Polystore Systems

A polystore is a framework that supports heterogeneous data that, through a single interface, exposes multiple storage engines [11]. Big Data's handling difficulties are caused by the range, volume and speed at which the data is delivered to the system. A fresh view of federated databases, Polystore handles various data models in multiple data stores (as shown in Fig. 4). Over multiple data models, it offers a cohesive single query language. Some academic work on this new method has been ongoing for the past few years. Several research institutions and major corporations have been working on Polystores. The BigDAWG [5] architecture of MIT offers island queries over multiple data models for the MIMIC II medical dataset [12]. Similarly, UCB's SparkSQL [13] offers a nested data model for SQL as a query language where SQL can be executed on top of Apache Spark [14]. Microsoft's PolyBase [15] is a new technology that integrates Microsoft's MPP product, SQL Server Parallel Data Warehouse (PDW), managed with Hadoop through RDBMS. The Hadoop Distributed File System (HDFS) module is capable of querying Hadoop from RDBMS.

Polystores is a relatively recent subject and goes back to multidatabase systems or federated database systems to trace its roots. They are often referred to as multi-store systems. Polystores also provide integrated access to various, heterogeneous stores of cloud data. CloudMdsQL Polystore [16] introduces all query building blocks as functions for querying multiple data sources (relational, NoSQL and HDFS) in the cloud with a functional SQL-like query language.

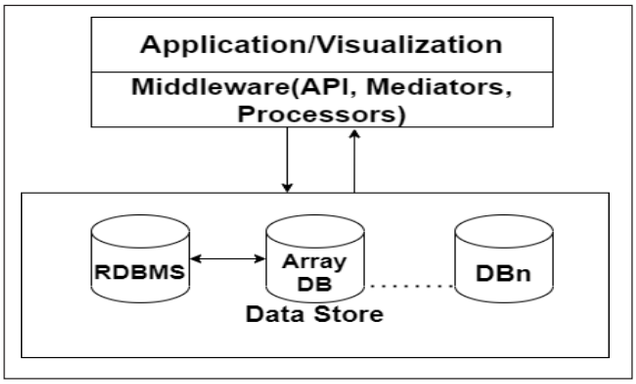

Fig. 4: Polystore Architecture

Other Polystores applications can be represented through the mediator/wrapper architecture. Mediator provides applications and consumers with a Global View (GVA). In order to handle Mediator communication, wrappers can encapsulate the DBMS component information. To move data within applications using multiple stores, native APIs or any other general APIs can be deployed as a wrapper.

III. Evaluation

In this section, we evaluate the similarities and differences between the various models of data integration as discussed in the previous section. These models can be evaluated with various criteria such as architecture, distribution, automation, query processing and data models supported. As the models focus on integrating the heterogeneous data through various data stores, there are some similarities in the features. Table I specifies the similarities and differences between these models.

FDBS, mediators and Polystore have architectural similarities as all of those are virtual models of data integration. These models are based on data virtualization. The data is virtualized using wrappers and query mapping is utilized. Also these models are variants of the distributed database systems.

TABLE I: COMPARISON OF THE DATA MODELS

Features	FDBS	DW	Mediator	Polystore
Architecture	Virtual	Physical	Virtual	Virtual
Distribution	Distributed	Central	Distributed	Central
Automation	Mapping	ETL	Wrappers	Wrappers/ Mapping
Query Processor	Federated Query Agents (FQA)	OLAP	Mapping and Merging	Island, HDFS bridge and more
Data Models	Single	Single	Multiple	Multiple

On the other hand, Data Warehouse is implemented through physical architecture where the automation is achieved through Extract-Transform-Load procedure. Data Warehouse is controlled by a central data store or warehouse.

Although there are similarities between the virtual models of data integration, the main differences lie in the query processing. FDBS uses a query processing method of Federated Query Agents (FQA). These agents can act as mediators to store and execute queries. The mediator model uses wrappers to map and merge queries whereas Polystore Systems use different varieties of query processors. For example, BigDAWG (MIT) uses island query processor whereas Polybase (Microsoft) uses a Hadoop Distributed File System (HDFS) bridge. The query processors are designed as per the variety and volume of data. Other major difference between the virtual models is the support for multiple data models. FDBS supports a single data model mainly relational data whereas the mediators and Polystore system support multiple data models.

IV. SUMMARY AND CONCLUSIONS

The emergence of big data has rendered FDBS and Data Warehouse obsolete since these models can only integrate databases with a single data model. Also the volume and the velocity of the increase of the data cannot be handled. These models have cost as well as performance issues while dealing with such data. For this reason, data virtualization via mediation is required. Thus, Polystore System provides a perfect architecture which supports multiple data models which can be stored in various heterogeneous data stores.

REFERENCES

[1] M. Ceriani, and P. Bottoni, "A dataflow platform for applications based on linked data," *International Journal of Computational Science and Engineering,* vol. 16, no. 4, pp. 419-429, 2018.

[2] C. R. Musick, T. Critchlow, M. Ganesh, T. Slezak, and K. Fidelis, "System and method for integrating and accessing multiple data sources within a data warehouse architecture," U.S. Patent No. 7,152,070, Dec. 19, 2006.

[3] A. P. Sheth, and J. A. Larson, "Federated database systems for managing distributed, heterogeneous, and autonomous databases," *ACM Computing Surveys*, vol. 22, no. 3, pp. 183-236, 1990.

[4] S. Suwanmanee, et al., "Wrapping and integrating heterogeneous databases with OWL," *7th International Conference on Enterprise Information Systems (ICIES 2005)*, 2005.

[5] V. Gadepally, P. Chen, J. Duggan, A. Elmore, B. Haynes, ..., and M. Stonebraker, "The BigDAWG polystore system and architecture," *2016 IEEE High Performance Extreme Computing Conference (HPEC)*, IEEE, Waltham, MA, USA, Sep. 13-15, 2016.

[6] M. Stonebraker, and U. Çetintemel, ""One size fits all": An idea whose time has come and gone," *Making Databases Work: The Pragmatic Wisdom of Michael Stonebraker*, 2018, pp. 441-462.

[7] Z. She, S. Ravishankar, and J. Duggan, "BigDAWG polystore query optimization through semantic equivalences," *2016 IEEE High Performance Extreme Computing Conference (HPEC),* IEEE, Waltham, MA, USA, Sep. 13-15, 2016.

[8] D. L. Moody, and M. A. R. Kortink, "From enterprise models to dimensional models: A methodology for data warehouse and data mart design," *Proceedings of the International Workshop on Design and Management of Data Warehouses (DMDW'2000),* Stockholm, Sweden, Jun. 5-6, 2000.

[9] S. Chaudhuri, and U. Dayal, "An overview of data warehousing and OLAP technology," *ACM Sigmod Record,* vol. 26, no. 1, pp. 65-74, 1997.

[10] G. J. L. Kemp, N. Angelopoulos, and P. M. D. Gray, "Architecture of a mediator for a bioinformatics database federation," *IEEE Transactions on Information Technology in Biomedicine*, vol. 6, no. 2, pp. 116-122, 2002.

[11] J. Duggan, A. J. Elmore, M. Stonebraker, M. Balazinska, B. Howe, ..., and S. Z. Brown, "The BigDAWG polystore system," *ACM Sigmod Record*, vol. 44, no. 2, pp. 11-16, 2015.

[12] Mohd. Saeed, M. Villarroel, A. T. Reisner, G. Clifford, L.-W. Lehman, ..., and R. G. Mark, "Multiparameter Intelligent Monitoring in Intensive Care II (MIMIC-II): A public-access intensive care unit database," *Critical Care Medicine*, vol. 39, no. 5, pp. 952-960, 2011.

[13] M. Armbrust, et al., "Spark SQL: Relational data processing in spark," *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, 2015.

[14] M. Zaharia, R. S. Xin, P. Wendell, T. Das, M. Armbrust, ….., and I. Stoica, "Apache spark: A unified engine for big data processing," *Communications of the ACM*, vol. 59, no. 11, pp. 56-65, 2016.

[15] D. J. DeWitt, A. Halverson, R. Nehme, S. Shankar, ..., and J. Gramling, "Split query processing in polybase," *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data,* 2013.

[16] B. Kolev, P. Valduriez, C. Bondiombouy, R. Jimenez-Peris, R. Pau, and J. Pereira, "CloudMdsQL: Querying heterogeneous cloud data stores with a common language," *Distributed and Parallel Databases*, vol. 34, no. 4, pp. 463-503, 2016.