Multi-model Data Management: What's New and What's Next?

Jiaheng Lu
Department of Computer Science
University of Helsinki, Finland
jiaheng.lu@helsinki.fi

Irena Holubová*
Department of Software Engineering
Charles University, Czech Republic
holubova@ksi.mff.cuni.cz

ABSTRACT

As more businesses realized that data, in all forms and sizes, is critical to making the best possible decisions, we see the continued growth of systems that support massive volume of non-relational or unstructured forms of data. Nothing shows the picture more starkly than the Gartner Magic quadrant for operational database management systems, which assumes that, by 2017, all leading operational DBMSs will offer multiple data models, relational and NoSQL, in a single DBMS platform. Having a single data platform for managing both well-structured data and NoSQL data is beneficial to users; this approach reduces significantly integration, migration, development, maintenance, and operational issues. Therefore, a challenging research work is how to develop efficient consolidated single data management platform covering both relational data and NoSQL to reduce integration issues, simplify operations, and eliminate migration issues. In this tutorial, we review the previous work on multi-model data management and provide the insights on the research challenges and directions for future work. The slides and more materials of this tutorial can be found at http://udbms.cs.helsinki.fi/?tutorials/edbt2017.

1. INTRODUCTION

In recent years the term big data has become a phenomenon that breaks down borders of many technologies and approaches that have so far been acknowledged as mature and robust for any conceivable application. One of the most challenging issues is the "*Variety*" of the data. It may be presented in various types and formats – structured, semi-structured and unstructured – and produced by different sources, and hence natively have various models.

To address the Variety challenge, probably the first type of respective specific database management systems (DBMS) are *NoSQL databases* [34] which can be further classified[1] to *soft* (e.g., object or XML DBMSs), and *core* (e.g., key/value, document, column, or graph DBMSs). From another point of view we can classify them to *single-model* and *multi-model*. The latter type enables to store and process structurally different data, i.e. data with distinct models, which corresponds to the Variety aspect of big data. This approach can be considered as an opposite idea to the "*One Size Does Not Fit All*" argument [39]. However, it can be also understood as a way of re-architecting traditional database models, namely the relational model, to handle new database requirements that were not present during its establishment decades ago [24]. Nothing shows the picture more starkly than the Gartner Magic quadrant for operational database management systems [18], which assumes that, by 2017, all leading operational DBMSs will offer multiple data models, relational and NoSQL, in a single DBMS platform.

In this tutorial, we review the previous work on multi-model data management and give insights on the research challenges and opportunities. First, we show that the idea of multi-model DBMSs is not a brand new approach. It can be traced back to Object-Relational Data Management Systems (ORDBMS) in the early 1990s and in a more broader scope even to federated and integrated DBMSs in the early 1980s. An ORDBMS system can manage different types of data such as relational, object, text and spatial by plugging domain specific data types, functions and index implementations into the DBMS kernels. For instance, PostgreSQL [6] can store relational, spatial and XML data. Recently, we can observe a new trend among NoSQL databases in the support of multiple data models against a single, integrated backend, while meeting the growing requirements for scalability and performance. For example, OrientDB [7] is a graph database extended to support multi-model queries, while ArangoDB [10] is moving from purely document model to the support of also key-value, graph and JSON data.

Second, we dive in three key aspects of technology in a multi-model database system including (1) storage strategies for multi-model data; (2) query languages accessing data across multiple models; and (3) query evaluation and its optimization in the context of multiple data models.

Finally, we provide comparison of features of the existing multi-model DBMSs and we discuss related open problems and remaining challenges.

To the best of our knowledge this is the first tutorial to discuss the state-of-the-art research works and industrial trends in the context of multi-model data management. Recent tutorials related to the big data world include SQL-on-Hadoop Systems [12], open-source on big data [16], knowledge bases

*Supported by the MŠMT ČR grant PROGRES.
[1]http://nosql-database.org/

in big data analytics [40], or big time-series data management [35], i.e., different aspects of big data challenges.

2. COVERED TOPICS

2.1 Background, History and Classification

In the first part of the tutorial we first provide a motivating example of a multi-model application and briefly describe most common data models used in the world of multi-model DBMSs (mainly key/value, relational, JSON, XML, and graph). Next, we focus on their history and classification.

The world of multi-model DBMSs can be divided into *single-database* and *multi-database* (see Figure 1), depending on whether the multiple models are handled in a single DBMS or there exist a number of cooperating or centrally managed DBMSs, each handling own data model(s).

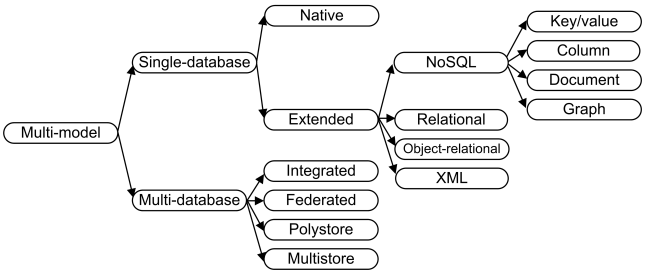

Figure 1: Classification of multi-model data management systems

The first approaches towards multi-model multi-database data management can be seen in *integrated DBMSs* [37] and *federated DBMSs* [20, 36]. Both types of systems can be characterized as a meta-DBMS consisting of a collection of (possibly) heterogeneous DBMSs which can differ in data models, constraints, query languages, and/or transaction management. The data integration is usually based on the idea of *mediators* [43]. The main difference is that in federated systems the DBMSs are autonomous and cooperate. Thus federated databases provide a compromise between no integration (where the users must explicitly interface with multiple autonomous DBMSs) and total integration (where the users can access data through a single global interface but cannot directly access a DBMS as a local user) [36].

Recently there has appeared a successor of federated databases – so-called *polystore systems* [38]. The key representative, system BigDAWG [17], also enables users to pose declarative queries that span several DBMSs. However, it consists of *islands of information*, i.e. collections of DBMSs accessed with a single query language (e.g., relational or array). Cross-island queries are supported using casting (e.g., tables to arrays or vice versa).

Another recent related approach from the area of big data analytics represent so-called *multistore systems* [23, 44]. For example system MISO [23] involves two types of data stores – a parallel relational data warehouse and a system for massive data storage and analysis (namely HDFS with Apache Hive). The aim is to combine their capabilities in order to gain more efficient query processing.

Multi-model single-database DBMSs can also be further classified. Probably the most natural classification is according their origin [2] (see Figure 1). Similarly to XML databases, we can distinguish *native* and *extended* DBMSs depending on whether the support for multiple models was the initial feature of the system, or it was added later. In the latter case we can find representatives amongst all four core types of NoSQL databases as well as traditional DBMS.

2.2 Overview and Comparison

In the second part of the tutorial we take a closer look at particular multi-model single-database DBMSs from the point of view of three key aspects of a database system.

The first database challenge is to develop a strategy to store distinct data models. Approaches used in the existing multi-model DBMSs can be classified according to the combination of used models. The main group (systems such as, e.g., PostgreSQL or Microsoft SQL Server [9]) is naturally represented by the (object-)relational model extended towards other data models, such as JSON, XML etc. From the set of NoSQL databases we can observe the tendency towards multi-model data management among column stores [4], key/value stores [11], or graph databases [7]. And there are also representatives of native hierarchical data stores [5] which support other types of data models.

The second database challenge is a query language capable of accessing and combining data having distinct models. Naturally, having a single language for managing queries over both (semi-)structured and NoSQL data is convenient to users. And again, in general, this is not a new feature of a query language, as we can see, e.g., in the case of the SQL/XML [21] extension of SQL. Most of the current NoSQL multi-model databases across the spectrum of storage strategies [6, 4, 7] support an SQL-like language. However, as we will show, despite this approach is natural and user-friendly, there are significant differences as well as persisting limitations. There also exist XML or JSON query language extensions towards other data models (e.g., MarkLogic's XPath for JSON [3]), as well as specific languages like, e.g., SQL++[31], JSONiq [33], or FSD domain-specific language [24]. In a more broader scope paper [32] identifies a subset of SQL for access to NoSQL systems or paper [13] evaluates the possibilities of using declarative structures in NoSQL data processing. We also discuss other techniques, like, e.g., [14, 32, 41].

The third challenge corresponds to query evaluation and optimization. As expected, the world of multi-model DBMSs exploits and extends verified database approaches such as indices (B+ tree, inverted, range, spatial, full text, etc.), views and materialization, hashing etc. In this part of the tutorial we overview and compare the query optimization technologies used in the previously discussed systems. We also introduce the related area of benchmarking multi-model database systems. As more and more platforms are proposed to deal with multi-model data, it becomes important to have benchmarks specific for this next generation of database systems. We mention several systems for benchmarking big data systems including YCSB [15], TPCx-BB [19], Bigframe [22], and UniBench [25].

We conclude this part with comparison of features of the state-of-the-art systems in the form of system-feature matrices and a timeline demonstrating their evolution.

2.3 Open Problems and Challenges

In the last part of the tutorial we focus on open problems

that must be addressed to ensure the success of multi-model DBMSs. The key areas to be discussed involve:

- Unified query processing and index structures,
- Multi-model main memory structure,
- Multi-model schema extraction, design, and optimization, especially in the context of schema-less DBMSs,
- Evolution management and model extensibility,
- Benchmarking and standardization.

In each of these areas we first briefly overview the solutions in the world of single-model DBMSs as well as eventually existing (partial) solutions among multi-model DBMSs. Then we explain the related problems in the context of multi-model databases, eventually with existing preliminary solutions. We assume that this part will raise questions to be discussed in the end of the tutorial.

3. TUTORIAL ORGANIZATION

The tutorial is planned for 1.5 hours and will have the following structure:

Motivation (5'). We motivate the need for multi-model data management by several examples in the era of big data.

History and classification (10'). We introduce the history and classification of multi-model databases, including ORDBMS [9], NoSQL databases [7, 10] and Polyglot persistence [38, 43].

Multi-model data storage (10'). We introduce various methods to store multi-model data, including object-relational model, graph model, document model and native hierarchical model.

Multi-model data query languages (15'). We compare languages for multi-model data processing, such as AQL [10], SQL++ [31], OrientDB SQL [7], and SQL/XML [21].

Multi-model query processing (15'). We overview the multi-model extensions of traditional query processing appraoches and indexes, such as B+ tree [1, 30], inverted index [8], schema discovery [42, 24], and cross-model query processing [10, 7].

Multi-model database benchmarking (15'). We introduce the previous and on-going benchmark systems for multi-model data, such as TPCx-BB [19], Bigframe [22], YCSB [15], or UniBench [25].

Open problem and challenges (20'). We conclude with a discussion of open problems and challenges for database research in the area of multi-model data management [29].

4. GOALS OF THE TUTORIAL

4.1 Learning Outcomes

The main learning outcomes of this tutorial are as follows:

- Motivation, classification and historical evolution of multi-model DBMSs.
- An overview of technologies and algorithms used by the current multi-model DBMSs including storing, query languages, and query optimization.
- Comparison of features of current multi-model DBMSs.
- A discussion of research challenges and open problems of multi-model data management.

4.2 Intended Audience

This tutorial is intended for a wide scope of audience, e.g. for developers and architects to get insights from the emerging industrial trends and its connections to scientific research, for stakeholders to make wise and informed decisions on investments in multi-model DBMS products, for motivated researchers and developers to select new topics and contribute their expertise on multi-model data, and, of course, for new developers and students to quickly gain a comprehensive picture and understand the new trends and the state-of-art techniques in this field.

Basic knowledge in relational and NoSQL databases is sufficient to follow the tutorial. Some background in semi-structured and graph query optimization would be useful, but is not necessary.

5. SHORT BIBLIOGRAPHIES

Jiaheng Lu is an Associate Professor at the University of Helsinki, Finland. He received Ph.D. degree at the National University of Singapore in 2007. He did two-year Postdoctoral research at the University of California, Irvine. His main research interests lie in the big data management and database systems, and specifically in the challenge of efficient data processing from real-life, massive data repository and Web. He has published more than sixty journal and conference papers. He has extensive experiences of the industrial cooperations with IBM, Microsoft and Huawei for the projects of NoSQL databases and performance tuning on distributed systems. He has published several books, on XML [27], Hadoop [28] and NoSQL databases [26]. His book [28] on Hadoop is one of the top-10 best-selling books in the category of computer software in China in 2013.

Irena Holubová is an Associate Professor at the Charles University, Prague, Czech Republic, where she received Ph.D. degree in 2007. Her current main research interests include big data management and NoSQL databases, big data generators and benchmarking, evolution and change management of database applications, analysis of real-world data, and schema inference. She has published more than 80 conference and journal papers; her works gained 4 awards. She has also published 2 books on XML technologies and NoSQL databases. She serves as an independent expert for evaluation and monitoring of EU FP7 and H2020 projects.

6. REFERENCES

[1] *Improving Secondary Index Write Performance in 1.2.* DataStax, Inc., 2013.

[2] *Neither Fish Nor Fowl: the Rise of Multi-Model Databases.* The 451 Group, 2013.

[3] *Application Developer's Guide – Chapter 18 Working With JSON.* MarkLogic Corporation, 2016.

[4] *Cassandra: Manage Massive Amounts of Data, Fast, without Losing Sleep.* The Apache Software Foundation, 2016.

[5] *MarkLogic: The World's Best Database for Integrating Data From Silos.* MarkLogic Corporation, 2016.

[6] *The Official Site for PostgreSQL, the World's Most Advanced Open Source Database.* The PostgreSQL Global Development Group, 2016.

[7] *OrientDB – a 2nd Generation Distributed Graph Database.* OrientDB, 2016.

[8] *PostgreSQL 9.5.3 Documentation – Chapter 61. GIN Indexes*. The PostgreSQL Global Development Group, 2016.

[9] *SQL Server 2016*. Microsoft, 2016.

[10] *Three major NoSQL data models in one open-source database*. ArangoDB, 2016.

[11] *Vertica Advanced Analytics*. Hewlett Packard Enterprise, 2016.

[12] D. Abadi, S. Babu, F. Ozcan, and I. Pandis. Tutorial: SQL-on-Hadoop Systems. *PVLDB*, 8(12):2050–2061, 2015.

[13] M. Bach and A. Werner. Standardization of NoSQL Database Languages. In *BDAS*, pages 50–60, 2014.

[14] F. Bugiotti, L. Cabibbo, P. Atzeni, and R. Torlone. Database Design for NoSQL Systems. In *ER*, pages 223–231, 2014.

[15] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears. Benchmarking Cloud Serving Systems with YCSB. In *SoCC*, pages 143–154, 2010.

[16] C. Douglas and C. Curino. Blind Men and an Elephant Coalescing Open-source, Academic, and Industrial Perspectives on BigData. In *ICDE*, pages 1523–1526, 2015.

[17] J. Duggan, A. J. Elmore, M. Stonebraker, M. Balazinska, B. Howe, J. Kepner, S. Madden, D. Maier, T. Mattson, and S. Zdonik. The BigDAWG Polystore System. *SIGMOD Rec.*, 44(2):11–16, Aug. 2015.

[18] D. Feinberg, M. Adrian, N. Heudecker, A. M. Ronthal, and T. Palanca. Gartner Magic Quadrant for Operational Database Management Systems, 12 October 2015.

[19] A. Ghazal, T. Rabl, M. Hu, F. Raab, M. Poess, A. Crolotte, and H. Jacobsen. BigBench: Towards an Industry Standard Benchmark for Big Data Analytics. In *ACM SIGMOD*, pages 1197–1208, 2013.

[20] M. Hammer and D. McLeod. *On Database Management System Architecture*. MIT/LCS/TM. Mass. Inst. of Technology, Laboratory for Computer Science, 1979.

[21] ISO. ISO/IEC 9075-14:2011 Information technology – Database languages – SQL – Part 14: XML-Related Specifications (SQL/XML), 2011.

[22] M. Kunjir, P. Kalmegh, and S. Babu. Thoth: Towards Managing a Multi-System Cluster. *PVLDB*, 7(13):1689–1692, 2014.

[23] J. LeFevre, J. Sankaranarayanan, H. Hacigumus, J. Tatemura, N. Polyzotis, and M. J. Carey. MISO: Souping Up Big Data Query Processing with a Multistore System. In *ACM SIGMOD*, pages 1591–1602, 2014.

[24] Z. H. Liu and D. Gawlick. Management of Flexible Schema Data in RDBMSs – Opportunities and Limitations for NoSQL. In *CIDR*, 2015.

[25] J. Lu. Towards Benchmarking Multi-Model Databases http://udbms.cs.helsinki.fi/?projects/ubench. In *CIDR*, 2017.

[26] J. Lu. *Big data challenge and NoSQL databases*. House of Electrical Industry in China, ISBN:978-7-121-19660-7, 423 pages, April, 2013.

[27] J. Lu. *An Introduction to XML Query Processing and Keyword Search*. Springer Berlin Heidelberg, ISBN: 978-3-642-34554-8, 201 pages, March 16, 2013.

[28] J. Lu. *Programming on Hadoop*. China Industrial Press, ISBN: 978-7-111-35944-9, 441 pages, October, 2011.

[29] J. Lu, Z. H. Liu, P. Xu, and C. Zhang. UDBMS: road to unification for multi-model data management. *CoRR*, abs/1612.08050, 2016.

[30] P. E. O'Neil. The SB-tree: An Index-sequential Structure for High-performance Sequential Access. *Acta Inf.*, 29(3):241–265, June 1992.

[31] K. W. Ong, Y. Papakonstantinou, and R. Vernoux. The SQL++ Unifying Semi-structured Query Language, and an Expressiveness Benchmark of SQL-on-Hadoop, NoSQL and NewSQL Databases, 2016.

[32] J. Rith, P. S. Lehmayr, and K. Meyer-Wegener. Speaking in Tongues: SQL Access to NoSQL Systems. In *SAC*, pages 855–857, 2014.

[33] J. Robie, G. Fourny, M. Brantner, D. Florescu, T. Westmann, and M. Zaharioudakis. The JSON Query Language, 2016.

[34] P. J. Sadalage and M. Fowler. *NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence*. Addison-Wesley Professional, 1st edition, 2012.

[35] Y. Sakurai, Y. Matsubara, and C. Faloutsos. Mining and Forecasting of Big Time-series Data. In *Proceedings of the 2015 ACM SIGMOD*, pages 919–922, 2015.

[36] A. P. Sheth and J. A. Larson. Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. *ACM Comput. Surv.*, 22(3):183–236, Sept. 1990.

[37] J. M. Smith, P. A. Bernstein, U. Dayal, N. Goodman, T. Landers, K. W. T. Lin, and E. Wong. Multibase: Integrating Heterogeneous Distributed Database Systems. In *AFIPS '81*, pages 487–499, New York, NY, USA, 1981. ACM.

[38] M. Stonebraker. The Case for Polystores, 2015.

[39] M. Stonebraker and U. Cetintemel. "One Size Fits All": An Idea Whose Time Has Come and Gone. In *ICDE '05*, pages 2–11, Washington, DC, USA, 2005. IEEE Computer Society.

[40] F. M. Suchanek and G. Weikum. Knowledge Bases in the Age of Big Data Analytics. *PVLDB*, 7(13):1713–1714, 2014.

[41] D. Tahara, T. Diamond, and D. J. Abadi. Sinew: a SQL System for Multi-structured Data. In *SIGMOD*, pages 815–826, 2014.

[42] D. A. Teich. Database Schemas Still Needed, Despite Hadoop and NoSQL Pretensions, 2016.

[43] G. Wiederhold. Mediators in the Architecture of Future Information Systems. *Computer*, 25(3):38–49, Mar. 1992.

[44] Y. Xu, P. Kostamaa, and L. Gao. Integrating Hadoop and Parallel DBMs. In *SIGMOD '10*, pages 969–974, New York, NY, USA, 2010. ACM.