Unified Management of Multi-Model Data*
(Vision Paper)

Irena Holubová[1][0000−0003−2113−1539], Martin Svoboda[1][0000−0003−4694−6806], and Jiaheng Lu[2][0000−0003−2067−454X]

[1] Charles University, Faculty of Mathematics and Physics, Prague, Czech Republic
{holubova,svoboda}@ksi.mff.cuni.cz

[2] University of Helsinki, Helsinki, Finland,
jiaheng.lu@helsinki.fi

Abstract. The variety of data is one of the most challenging issues for research and practice in data management. The so-called multi-model data are naturally organized in different and mutually interlinked data formats and logical models, including structured, semi-structured, and unstructured. In this vision paper, we discuss the so far neglected, but for correct and efficient management of multi-model data critical issues and challenges: conceptual modeling of multi-model data, inference of multi-model schemas, unified and conceptual querying, evolution management, and, last but not least, autonomous multi-model data management.

Keywords: multi-model databases · conceptual modeling · schema inference · query languages · evolution management · autonomous systems

1 Introduction and Motivation

In recent years, the Big Data movement has broken down borders of many technologies and approaches that have so far been widely acknowledged as mature and robust. One of the most challenging issues is the *variety* of data. It means that data may be present in multiple types and formats – structured, semi-structured, and unstructured – and independently produced by different sources as well as natively conform to various models, schemas or ontologies.

Although traditional relational databases have been the systems of the first choice for decades, with the arrival of Big Data, their capabilities have become insufficient in many aspects, and so new types of systems, such as NoSQL or NewSQL, have appeared. The variety of *multi-model data* itself brings another dimension of complexity since multiple distinct models must be efficiently supported at a time. Currently, there exist more than 20 representatives of so-called

* This work was partially supported by the Charles University project PROGRES Q48 and the Academy of Finland project number 310321

multi-model databases [15], involving well-known tools, both traditional relational and novel NoSQL (such as Oracle DB, Cassandra, or MongoDB).

The main open problems of these systems are: (1) The level of support for multi-model data varies greatly, with different extent of ability to query across different models, index internal structures or optimize query evaluation plans. (2) Since these systems originate mainly in the IT industry, the existing solutions are determined and significantly limited by the specifics of the original underlying single-model systems. (3) For the same reason, there is a lack of necessary formal background, unified approaches, and generally applicable methods allowing to work with multi-model data in full possible extent.

In this vision paper, we discuss these critical open problems and envision the core research areas closely related to the conceptual modeling and data management that need to be appropriately targeted. Namely, we describe, justify, as well as outline possible solutions for the following five key challenges: (1) proposal of a formal background for conceptual modeling of multi-model data and mapping and transformation of such data into individual models, (2) algorithms for inference of multi-model schemas, (3) unified and conceptual querying over multi-model data, (4) correct propagation of changes to data, schemas, and queries induced by the evolution management, and, finally, (5) autonomous multi-model data management in general.

The rest of the paper is structured as follows: In Section 2, we provide a brief overview of the existing mainly single-model approaches to data management. In Section 3, we discuss the open problems and challenges of multi-model databases, while we conclude in Section 4.

2 Related Work

There are basically two existing general approaches to manipulate and query multi-model data: (1) *polyglot persistence*, and (2) *multi-model databases* [17].

The main strategy of the first kind of systems is to leverage different databases to store different models of data and then develop a mediator to integrate them together. While this idea can be traced back to not only federated databases studied during the 1980s, recently, several research prototypes developed on the polyglot persistence paradigm were also introduced. For example, DBMS+ [12] targets at embracing several database platforms with unified declarative processing, while BigDAWG [9] provides an architecture supporting location transparency and a middleware providing a uniform multi-island query interface.

The second kind of systems incorporates only one single database to manage different data models, and provides a fully integrated backend to handle the system demands for performance, scalability, and fault tolerance [18]. The idea of an integrated system can be traced back to the concept of *object-relational databases*, which borrow and adapt the object-oriented programming principles into the world of relational databases.

With the dawn of Big Data, the challenge of handling variety has recently inspired a new generation of dedicated *multi-model databases*, capable of stor-

ing and processing structurally different data by supporting several data models within just a single database. This way of solving the polyglot persistence problem offers advantages in data modeling, allowing to represent data in its most native from. While this approach can be considered as opposite to the *one size does not fit all* argument [26], it can also be understood as a way of re-architecting traditional database models to address new requirements [13]. If nothing else, it was (correctly) assumed that, by 2017, the majority of leading database systems would offer multiple data models within just a single platform.

3 Research Challenges

While the existing multi-model databases pursue the bottom-up design principles, and so essentially represent kind of a trade-off solution, where a core model is more-or-less painfully adapted to additional new models, a top-down approach that would provide a systematically designed and robust conceptual multi-model solution backed by a precise formalism is still missing. In particular, we see the following main issues:

1. *Formal background definition:* There is a need for a complex formal apparatus for multi-model data representation, storing and querying, including proofs of its features and complexity of algorithms.
2. *Data processing unification:* Unified and generally applicable methods and approaches for data processing tasks at the conceptual level (together with necessary mappings and extensions to the logical level) need to be proposed.
3. *Practical impact preservation:* All the proposed languages, methods, and algorithms must still preserve a tight relation to the existing systems so that they can be exploited in real-world scenarios and implementations.

In this section, we discuss in detail five particular key areas we see as the primary research targets for the conceptual modeling and database communities.

3.1 Conceptual Modeling of Multi-Model Data

When data across distinct models are to be processed together, their schemas inferred, or query expressions evaluated, kind of a unified data abstraction has to be established first. These models often mutually share a couple of the same principles on the one hand, while can also have certain specifics on the other.

For this purpose, widely used modeling languages ER [6] and UML [22] could be utilized and in a *top-down* way adjusted to the needs of individual logical models. While the former language exists in several notations yet provides more complex constructs better grasping the real-world relationships among entities, the latter one is standardized but, unfortunately, only too data-oriented and concealing important details (e.g., weak entity types). On the contrary, *bottom-up* approaches could find an inspiration in NoSQL AbstractModel [5], a system-independent model for so called *aggregate-oriented* databases.

Regardless of the adopted strategy, the theory of categories [14], associative arrays [11], or description logics [1] could be utilized to internally model the data in a formal, abstract, and rigorous way. Complex non-relational systems often involve a variety of heterogeneous and interrelated models – models that are, unfortunately, expressed using several modeling languages. Moreover, if there are only a few solutions targeting at conceptual modeling of NoSQL databases in general, modeling of graph databases is even more non-trivial [21].

The key aspect of multi-model data is mutual links between the distinct models. Their semantics and features can differ depending on the types of interlinked models. Also, within the single-model systems, these links can have different representations, involving, e.g., foreign keys in the relational model, pointers in the object model, or embedding and references in document models.

To sum up, the first core issue of multi-model data management is to define and formally describe a way how multi-model data can be modeled and further processed at a conceptual level in a unified means abstracting specific features and technical details of individual models. Next, mapping rules and transformation operations need to be defined so that the proposed conceptual constructs can be mapped to data structures provided by individual logical models, as well as data directly transformed from / to at least the widely used models.

3.2 Inference of Multi-Model Schemas

With multi-model data and databases, we may distinguish several levels of schema support ranging from *schema-full* (where a schema description is provided explicitly and its requirements must be satisfied) to *schema-less* (where a schema is neither provided nor required).

In reality, however, even in schema-less databases, there typically exists an implicit schema, i.e., kind of an agreed structure of the data expected by the application. Hence, the idea of schemalessness is often rather characterized as *schema-on-demand*. This observation motivates the necessity of research in the area of multi-model schema inference.

In case of a single-model schema inference, there exists a number of techniques. As a consequence of Gold's theorem [10], e.g., XML schema languages are not identifiable from positive examples only (i.e., sample data). Hence, either an *identifiable* subclass of such a language has to be inferred, or heuristics must be utilized. Naturally, a large set of inference approaches, both heuristic [20] and grammar-inferring [3], can be found for XML data. With the dawn of NoSQL databases, there appeared approaches inferring, e.g., (big) JSON data [2] or general approaches for aggregate-oriented databases [24].

When dealing with multi-model schema inference, we can primarily focus on heuristic approaches. Apart from multi-model extensions of the existing verified single-model approaches, mutual links between records across the models can bring another piece of important information. Inference approaches may thus benefit from information extracted from related data in distinct models.

The second issue of multi-model data management can hence be summarized as the need of a universal multi-model schema-inference method that would

provide near real-world schemas and which would be able to infer a correct schema at least for the majority of real-world use cases.

3.3 Multi-Model Data Querying

There already exist proposals of proprietary multi-model query languages [16]. For example, AQL provided by ArangoDB enables one to access both graph and document data. However, these languages have numerous limitations, often lack the desired level of documentation and formalism, and not only because of that, it is still an open challenge to develop a full-fledged query language for multi-model data.

In pursuit of such a language, it is only natural to take into account features of the existing languages used both in multi-model as well as single-model databases. Despite they assume different data models and thus have certain specifics, some of their aspects are rather surprisingly shared by more of them. For example, results of SPARQL and Neo4j Cypher query expressions are tables analogous to the relational model, even though these languages are intended for RDF triples and property graphs respectively. Expressions of the majority of languages are often decomposed into clauses, yet their structure and order are fixed in case of SQL, while in Cypher these clauses can almost arbitrarily be chained together. If usage of sub-queries in SQL is straightforward, not all the languages support such a concept. In XQuery for XML data, expressions of all kinds act like functions, and so can be arbitrarily embedded into each other, on the contrary. Even expressions at a higher level of abstraction based on lambda functions are provided in case of XQuery.

It is apparent that in a long-range perspective, it is highly unlikely that such a variety of models and query languages could reasonably be maintained and harnessed. And while the integration at the level of data has already begun as plenty of formerly single-model systems are being enriched with additional data formats, proposal of robust, unified, and even conceptual query languages with appropriate expressive power should obviously be considered as the next step, while other challenges, such as, e.g., multi-model indexing techniques, efficient query evaluation and optimization etc., will in turn follow.

Even though the idea of conceptual querying is not new [28, 4], contemporary multi-model databases require a new point of view. Therefore, the third challenging issue is to overcome the outlined obstacles and research on the possibilities of introducing such a unified, well-formalized, and still user-friendly query language for the multi-model environment, so that the data could be processed uniformly from a conceptual perspective concealing representation details of individual logical models and their physical implementations.

3.4 Evolution Management in Multi-Model Environment

Efficient management of schema evolution and propagation of changes to relevant parts of a database system, such as data instances, queries, indices, or even storage strategies, is a difficult task in general. In smaller applications, a company

can rely on a skilled database administrator, but in most cases, it is still a complicated and error-prone job.

Currently, there exists a number of approaches dealing with single-model systems or systems with closely related models, namely aggregate-oriented NoSQL databases [23, 27]. There also exists a nontrivial set of approaches focusing primarily on the evolution management of XML documents, as well as comprehensive analyses of changes of real-world database schemas over time.

In the case of multi-model databases, this task is even more subtle and difficult, not only because we need to distinguish between *intra-model* and *inter-model* changes. In the former case, we can re-use the existing single-model approaches, while in the latter one, however, these cannot be straightforwardly applied. In addition, the challenge of query rewriting [7, 19], i.e., propagation of changes to queries, also becomes more complex in case of inter-model changes, which then require changes in data access constructs.

The fourth issue, therefore, is a proposal of a solution dealing with multi-model evolution management covering both intra-model and inter-model changes and ensuring their correct and complete (at least) semi-automatic propagation to all affected parts of the system. This requires a definition of a set of schema modification operations, their precise semantics, as well as the corresponding algorithms for their correct and efficient propagation to not only data instances.

3.5 Autonomous Multi-Model Data Management

Autonomous data management provides special features that enable databases to self-tune and self-heal [8, 25]. This service relieves database administrators of the remaining operational tasks (that include advanced tuning functions, database security, and troubleshooting), and so they can focus more time on design and development activities instead of administering the database installation and configuration.

Considering the environment of multi-model data, one application can store data in one data model, whereas later the same data can be queried by another application using a different model via *multi-model data views* [14]. Hence, multi-model data transformation can exploit the genuine value of multi-model databases which enable applications requiring different data models to share the same platform. Multi-model databases are supposed to transparently provide different access interfaces (views) of the same data adaptive to each application requirement. Autonomous multi-model databases can recommend suitable data models as such, while at a more advanced level, they can also provide data model virtualization via controlling of physical multi-model data materialization and transformation adaptively.

To conclude, the fifth issue is a proposal of a solution building autonomous multi-model databases to automatically handle the evolution of data models, selecting the best models for physical storage of data, and performing automatic transformations between the involved models. In general, it is the responsibility of databases (not users) to find the best way to organize and store the data in order to fulfill and optimize inter-data model queries and modification requests.

4 Conclusion

As the current trends indicate, multi-model databases represent a dignified and promising successor of the traditional approaches for the newly emerging and challenging use cases. Yet they first need to gain solid foundations and reach the same level of both applied and theoretical maturity in order to become a robust alternative to the relational databases.

We hope to entice the database and conceptual modeling communities to deal with the identified multi-model data management challenges related to the conceptual view of this domain. In particular and as we hope we have shown and argued in this paper, we believe especially the following areas are calling for attention and should be appropriately tackled so that the envisioned functionality of database systems could be pursuit:

- Conceptual modeling of multi-model data enabling their further unified processing, while abstracting specific features of widely used logical models and still preserving the practical usability.
- Universal multi-model schema inference methods that will be able to provide near real-world schemas for at least the majority of real-world use cases and widely used constructs.
- User-friendly, yet well-formalized query language allowing for the unified processing of multi-model data at a conceptual layer concealing details of individual logical models.
- Evolution management covering both intra-model and inter-model schema changes and ensuring their correct and complete propagation to all the affected parts of the multi-model system.
- Autonomous multi-model database management solution allowing to select suitable logical models, handle the evolution of schemas, as well as transformation of both data and query expressions.

References

1. F. Baader, D. Calvanese, D. McGuinness, P. Patel-Schneider, and D. Nardi. *The description logic handbook: Theory, implementation and applications*. Cambridge university press, 2003.
2. M.-A. Baazizi, D. Colazzo, G. Ghelli, and C. Sartiani. Parametric schema inference for massive JSON datasets. *The VLDB Journal*, Jan. 2019.
3. G. J. Bex, F. Neven, T. Schwentick, and S. Vansummeren. Inference of concise regular expressions and DTDs. *ACM Trans. Database Syst.*, 35(2):11:1–11:47, 2010.
4. A. C. Bloesch and T. A. Halpin. ConQuer: a conceptual query language. In *ER 1996*, pages 121–133. Springer, 1996.
5. F. Bugiotti, L. Cabibbo, P. Atzeni, and R. Torlone. Database design for NoSQL systems. In *Conceptual Modeling*, pages 223–231, Cham, 2014. Springer.
6. P. Chen. The entity-relationship model – toward a unified view of data. *ACM Transactions on Database Systems*, 1(1):9–36, Mar. 1976.
7. C. A. Curino, H. J. Moon, and C. Zaniolo. Graceful database schema evolution: The PRISM workbench. *Proc. VLDB Endow.*, 1(1):761–772, Aug. 2008.

8. A. K. Elmagarmid, M. Rusinkiewicz, A. Sheth, and A. Sheth. *Management of heterogeneous and autonomous database systems*. Morgan Kaufmann, 1999.
9. Elmore et al. A demonstration of the BigDAWG polystore system. *PVLDB*, 8(12):1908–1911, 2015.
10. E. M. Gold. Language identification in the limit. *Information and Control*, 10(5):447–474, 1967.
11. J. Kepner, V. Gadepally, D. Hutchison, H. Jananthan, T. Mattson, S. Samsi, and A. Reuther. Associative array model of SQL, NoSQL, and NewSQL databases. In *HPEC 2016*, pages 1–9. IEEE, 2016.
12. H. Lim, Y. Han, and S. Babu. How to fit when no one size fits. In *CIDR 2013*. www.cidrdb.org, 2013.
13. Z. H. Liu and D. Gawlick. Management of flexible schema data in RDBMSs - opportunities and limitations for NoSQL. In *CIDR 2015*. www.cidrdb.org, 2015.
14. Z. H. Liu, J. Lu, D. Gawlick, H. Helskyaho, G. Pogossiants, and Z. Wu. Multi-model database management systems - A look forward. In *VLDB 2018 Workshops, Poly and DMAH*, pages 16–29, 2018.
15. J. Lu and I. Holubová. Multi-model data management: What's new and what's next? In *EDBT 2017*, pages 602–605, 2017.
16. J. Lu and I. Holubová. Multi-model databases: A new journey to handle the variety of data. *ACM Comput. Surv. (accepted)*, 2019.
17. J. Lu, I. Holubová, and B. Cautis. Multi-model databases and tightly integrated polystores: Current practices, comparisons, and open challenges. In *CIKM 2018*, pages 2301–2302, 2018.
18. J. Lu, Z. H. Liu, P. Xu, and C. Zhang. UDBMS: Road to unification for multi-model data management. *CoRR*, abs/1612.08050:285–294, 2016.
19. P. Manousis, P. Vassiliadis, and G. Papastefanatos. Automating the adaptation of evolving data-intensive ecosystems. In *ER 2013*, pages 182–196, 2013.
20. I. Mlýnková and M. Nečaský. Heuristic methods for inference of XML schemas: Lessons learned and open issues. *Informatica, Lith. Acad. Sci.*, 24(4):577–602, 2013.
21. J. Pokorný. Conceptual and database modelling of graph databases. In *IDEAS 2016*, pages 370–377, New York, NY, USA, 2016. ACM.
22. J. Rumbaugh, I. Jacobson, and G. Booch. *Unified modeling language reference manual*. Pearson Higher Education, 2004.
23. S. Scherzinger, T. Cerqueus, and E. C. de Almeida. Controvol: A framework for controlled schema evolution in NoSQL application development. In *ICDE 2015*, pages 1464–1467. IEEE Computer Society, 2015.
24. D. Sevilla Ruiz, S. F. Morales, and J. García Molina. Inferring versioned schemas from NoSQL databases and its applications. In *Conceptual Modeling*, pages 467–480, Cham, 2015. Springer.
25. A. P. Sheth and J. A. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys (CSUR)*, 22(3):183–236, 1990.
26. M. Stonebraker and U. Cetintemel. "One size fits all": An idea whose time has come and gone. In *ICDE 2005*, pages 2–11, Washington, DC, USA, 2005. IEEE Computer Society.
27. U. Störl, D. Müller, M. Klettke, and S. Scherzinger. Enabling efficient agile software development of nosql-backed applications. In *BTW 2017*, pages 611–614, 2017.
28. A. H. ter Hofstede, H. A. Proper, and T. P. Van Der Weide. Formal definition of a conceptual language for the description and manipulation of information models. *Information Systems*, 18(7):489–523, 1993.