Toward Pay-As-You-Go Data Integration for Healthcare Simulations

Philipp Baumgärtel[1,2], Gregor Endler[1] and Richard Lenz[1]
[1]*Institute of Computer Science 6, Friedrich-Alexander University of Erlangen-Nuremberg, Germany*
[2]*On behalf of the ProHTA Research Group*
{*philipp.baumgaertel,gregor.endler,richard.lenz*}*@fau.de*

Keywords: Data Management, Data Integration, Simulation, Healthcare

Abstract: ProHTA (Prospective Health Technology Assessment) aims at understanding the impact of innovative medical processes and technologies at an early stage. To that end, large scale healthcare simulations are employed to estimate the effects of potential innovations. Simulation techniques are also utilized to detect areas with a high potential for improving the supply chain of healthcare. The data needed for both validating and adjusting these simulations typically comes from various heterogeneous sources and is often preaggregated and insufficiently documented. Thus, new data management techniques are required to cope with these conditions. Because of the high initial integration effort, we propose a pay-as-you-go approach using RDF. Thereby, data storage is separated from semantic annotation. Our proposed system offers automatic initial integration of various data sources. Additionally, it provides methods for searching semantically annotated data and for loading it into the simulation. The user can add annotations to the data in order to enable semantic integration on demand. In this paper, we demonstrate the feasibility of this approach with a prototype implementation. We discuss benefits and remaining challenges.

1 INTRODUCTION

Medical and statistical simulation data in our project ProHTA (Prospective Health Technology Assessment)(Djanatliev et al., 2012) stems from several heterogeneous sources like spreadsheets, relational databases or XML files. Therefore, semantic data integration is an important concern. Although many techniques exist for automatic integration of data (Rahm and Bernstein, 2001), it is still an expensive process. Like Lenz et al. (Lenz et al., 2007), we distinguish between technical and semantic integration. Technical integration enables accessing data. In contrast, semantic integration facilitates understanding the meaning of data. Since our pool of data sources is likely to change frequently, we investigate an integration strategy based on the dataspaces abstraction (Franklin et al., 2005). This approach allows the coexistence of heterogeneous data sources, which are initially integrated only as far as automatically possible (Das Sarma et al., 2008). The system then allows the gradual improvement of the degree of integration in a pay-as-you-go manner (Jeffery et al., 2008).

To achieve pay-as-you-go integration, we developed an architecture with multiple levels of integration based on the five-level schema architecture for federated databases by Sheth and Larson (Sheth and Larson, 1990). The original ANSI/SPARC three-level schema architecture consists of conceptual, internal, and external schema. The conceptual schema describes the data structures and the relationships among them on a logical level. The internal schema contains indexes and information about the physical storage of records. The external schema provides tailored views for separate groups of users. Sheth and Larson (Sheth and Larson, 1990) introduce additional schema levels to support federated databases. The local schema contains the native data model of a data source. The component schema is the translation of the local schema into a canonical data model. For our approach, we adapt the local schema and component schema, which replace the conceptual schema of the ANSI/SPARC architecture. By explicitly storing the native data model of a data source and deferring the translation into a canonical data model, we allow for demand driven integration.

To enable storing heterogeneous data with different data models, our data management system employs a fully generic data schema. As a side effect, following the principle of "design for change" (Parnas, 1994) from the outset ensures evolutionary capabilities, an important factor for success. This is espe-

HEALTHINF 2014 - Proceedings of the International Conference on Health Informatics (DOI: 10.5220/0004734201720177)

cially true for projects in the healthcare sector with its rapidly changing conditions (Lenz, 2009).

An evolutionary information system requires special attention as to the keeping of data. A fully generic data schema like EAV/CR (Nadkarni et al., 1999), while perfectly able to manage changing conditions, significantly increases query complexity. For this reason, we adopt an approach using RDF (Resource Description Framework) (Lassila et al., 1999). RDF employs a generic schema consisting of `(subject,predicate,object)` triples to store and link objects. In (Baumgärtel et al., 2013), we developed a Benchmark to compare the performance of several generic data management solutions for simulation input data. With this benchmark, we evaluated a document store, relational databases using an EAV schema, and an RDF triplestore. This benchmark showed that RDF triplestores are slower and not as mature as relational databases. However, RDF triplestores allow concise SPARQL queries (Prud'hommeaux and Seaborne, 2008), whereas SQL queries on an EAV/CR schema prove to be very complex. Therefore, we chose to use RDF triplestores, as their flexibility and query capabilities outweigh their performance drawbacks.

Howe et al. (Howe et al., 2011) identified several key barriers to adopting a data management system for science. They argue that the initial effort of designing a data schema is too complicated in most cases. Therefore, a "data first, structure later" approach should be implemented to reduce the initial effort. Another problem identified by Howe et al. is that scientists sometimes need help to write non-trivial SQL statements. Therefore, we identified several requirements for our simulation data management system:

Automatic Technical Integration: Because of the changing demands of the simulation and the amounts of heterogeneous data, the initial effort of integrating a data source should be minimized. Many sources being available to us have the same format but different semantics. Therefore, the system should enable automatic technical integration and schema import for sources with known formats.

Deferred Semantic Integration: As the data has to be reusable for different simulation studies, the data management system has to provide the means of adding semantic information to stored data. These annotations can then be used for deferred semantic integration on demand implementing a "data first, structure later" approach.

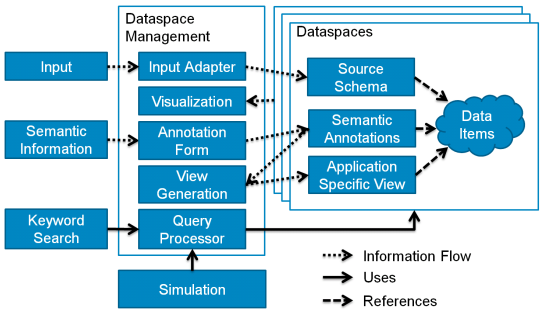


Figure 1: The architecture of our simulation data management system

Querying Simulation Data: The simulation modeler has to be able to find and query data easily. The effort of using a data provider for simulation input data management should be less than manual data input.

2 INTEGRATION

In this section, we propose an approach to simulation data management utilizing pay-as-you-go integration. In a previous paper (Baumgärtel and Lenz, 2012), we developed an ontology for storing data cubes using RDF. We also described a simple domain specific query language assisting simulation modelers to load data into their simulation models. However, the problem of high initial effort for integrating the data remained unsolved, as up-front semantic information about the data sets is necessary.

Therefore, we propose a flexible solution for integrating, annotating and querying simulation input data. Data and metadata are stored in an RDF triplestore using several generic upper ontologies. A web frontend visualizes the data and provides methods for getting user input. Figure 1 depicts the components of our approach. Data is loaded using input adapters for various source types, e.g. spreadsheets, XML files or relational databases. Then, the data is stored utilizing the automatically imported schema of the data source. For example, a spreadsheet is stored using concepts like table, row, column and cell. Initially, the user can query the data utilizing full-text search or SPARQL. Also, the user is able to view the data employing a visualization module for that specific source type. This module also enables the user to add annotations about the semantics of the data to facilitate deferred semantic integration.

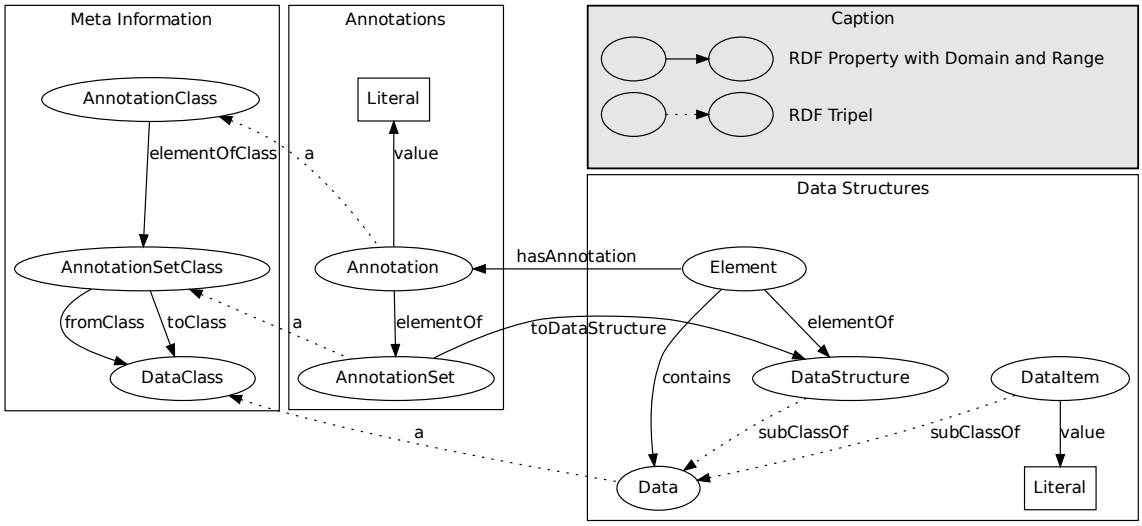


Figure 2: Upper ontology for data structures and annotations

2.1 Example

As a running example, we use the integration of multidimensional statistical data. This kind of data is the most prominent in our simulation project. We have to integrate preaggregated statistical data in pivot tables stored in spreadsheets. Figure 3 shows an example for this kind of data.

To integrate pivot tables as multidimensional data cubes, we need semantic information about the dimensions represented by rows and columns. In the following sections, we will show how our generic framework for demand driven integration enables us to build a pay-as-you-go integration engine to accomplish this.

	1	2	3	4	5	6	7	8	9	10
1			1995-1996	1995-1996	1997-1998	1997-1998	1999-2000	1999-2000	2001-2002	2001-2002
2			M	F	M	F	M	F	M	F
3	Total	Total	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
4	Stroke subtype	CI	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
5	Stroke subtype	PICH	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
6	Stroke subtype	SAH	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
7	Stroke subtype	UNS	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
8	TOAST subtype	LAA	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
9	TOAST subtype	CE	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
10	TOAST subtype	SAO	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

Figure 3: Our web frontend visualizing tabular data

2.2 Technical Integration

For every type of data source an input adapter has to be implemented and an ontology to store the schema of this source type has to be constructed. We developed an upper ontology for describing these schema ontologies. This ontology is depicted in Figure 2. Additionally, this upper ontology enables storing arbitrary data that corresponds to a schema ontology. The data is stored as *Elements* of *Data Structures*. These *Elements* may contain *Data*, which could be a single *Data Item* or another substructure. Therefore, several *Data Structure* ontologies can be assembled. *Elements* can have *Annotations* that are part of a set of annotations for one *Data Structure*.

Now, ontologies to store data cubes or tables can be constructed using this upper ontology. To integrate a data source, the input adapter imports the actual schema of this source and stores it as an instance of a *Data Structure*. Then, the data is imported and stored in the RDF triplestore. This enables querying of all data sources with one common query language. Using the terminology of Sheth and Larson (Sheth and Larson, 1990), the schema of the data source corresponds to the local schema.

To validate this approach, we implemented an adapter for CSV files and an adapter for spreadsheets, as these are the most common file types in our project. To store these files, we constructed an ontology for tabular data using our upper ontology (Figure 4(a)). The input adapter uses this ontology to import the schema of the data sources. All of the classes in the table ontology are derived from classes of our upper ontology. As all data is stored using our upper ontology as a common vocabulary, our system can initially provide a full-text keyword search for these file types. Additionally, the data can be queried using SPARQL. However, our approach is not limited to spreadsheets but also applicable to relational or hierarchical data for example.

2.3 Visualization and Semantic Annotation

For every source type, a visualization module can be implemented. To display the data, we use HTML templates and modules to load and process schema information. We implemented a visualization component for tabular data to visualize CSV files and spreadsheets. Fig. 3 depicts our web frontend displaying a simple table.

The frontend facilitates the annotation of individual elements of a dataset to add semantic information. To this end, the frontend automatically generates annotation forms using annotation ontologies. These annotation ontologies describe sets of possible annotations. To define these annotation ontologies, we developed an upper annotation ontology (Figure 2). These semantic annotations can then be utilized for semantic integration.

For tabular data, the elements that can be annotated are rows, columns and cells. The user can select a set of possible semantic annotations he would like to add. For example, the user can annotate rows and columns of a spreadsheet containing multidimensional data with information about dimensions and granularities. Figure 4(b) depicts an annotation ontology that can be used to store information about multidimensional data in tables.

To validate the modularity of our approach, we implemented an additional annotation ontology to describe spreadsheets containing multiple subtables. For this new annotation ontology, the annotation forms are generated automatically and the table can be annotated. Because of the independence of the modules, there is no need to change any code when adding new annotation ontologies. However, modules to process the annotations for semantic integration have to be implemented.

2.4 Semantic Integration

When there is sufficient semantic information about a dataset, the system can translate the data into the canonical data model, which is represented by another data-structure ontology. For each set of annotations and each source type, a semantic integration module can be implemented.

We implemented a module to integrate multidimensional data using our data cube ontology (Baumgärtel and Lenz, 2012) as the canonical data model. This component utilizes annotations to tables (Figure 4(b)) and constructs new multidimensional data structures. That way, we can integrate pivot tables as multidimensional data cubes. Now, the data can be queried using our domain specific query language for multidimensional data (Baumgärtel and Lenz, 2012). Other domain specific languages can be added to our query processor as well.

2.5 Searching and Querying Simulation Input Data

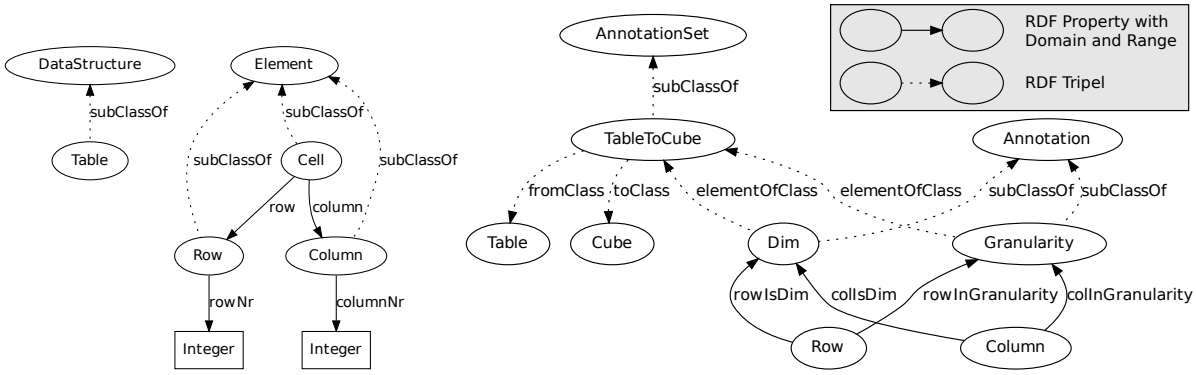
After data is stored in our system, it offers a full-text keyword search. As SPARQL queries tend to be very complex, we do not expect the simulation modelers to write SPARQL queries themselves. In future work, we will add automatic query generation to the graphical user interface. However, writing SPARQL queries remains possible and enables the user to aggregate and manipulate the data. As data is annotated with further information, the data can be queried using high level domain specific query languages. For example, we developed a domain specific query language for agent based simulation models to query and aggregate data that corresponds to steps in a UML activity diagram (Baumgärtel et al., 2014).

3 VALIDATION

We validated our concepts by integrating CSV-files and spreadsheets stemming from the German Federal Statistical Office[1]. Most of these data files contained multidimensional data as pivot tables.

We compared the traditional data management workflow using spreadsheets or manual input (Robertson and Perera, 2002) to the workflow using our system. To evaluate the traditional workflow, we employed a document management system to store spreadsheets. In both scenarios, data is stored by uploading files to a web frontend. Both the document management system and our prototype offer means of searching data by keywords. In the traditional workflow the data is viewed using a spreadsheet application. In contrast, our system offers integrated means of viewing spreadsheet data. In the traditional workflow, data sets are loaded into the simulation model by simulation framework specific adapters for spreadsheets. Using our approach, data sets are loaded into the simulation using SPARQL. The effort of defining the desired rows and columns of the data set using the simulation framework specific adapters was equal to using SPARQL. However, semantic annotation and integration enable the use of domain specific query languages with additional features. Therefore, the initial effort of storing data is the same in both scenar-

[1] http://www.destatis.de



(a) Ontology for tables (b) Ontology to annotate pivot tables

Figure 4: RDF classes to describe tables and the mapping of pivot tables to data cubes

ios. However, our system offers techniques to gradually improve the semantic integration of the data and therefore its reusability.

4 RELATED WORK

In this section, we will give a brief overview about existing work on simulation data management.

SciDB (Rogers et al., 2010) is a tool for scientific data management storing large multidimensional array data and supporting efficient data processing. In contrast to SciDB, our main challenge is not to handle very large array data, but to handle many small data sets in heterogeneous formats.

SIMPL (Reimann et al., 2011) is a framework supporting the ETL-process (extract, transform and load) for simulation data. The schema of data sources is described in XML, which is flexible enough for heterogeneous data.

DaltOn (Jablonski et al., 2009) is a data integration framework for scientific applications. Integration is done using descriptions of the data sources.

Arthofer et al. (Arthofer et al., 2012) developed an ontology based tool to integrate medical data. They also provide an automatically generated ontology based web frontend to add additional information with focus on data quality.

However, with all these systems the high initial effort of data integration remains. Howe et al. (Howe et al., 2011) describe a relational scientific dataspace system with support for schema free storage of tables. The main aspect of the system is the automatic suggestion of SQL queries and not the semantic integration of the tables. In contrast, our system facilitates semantic integration and can be extended to support other data formats than tables.

5 CONCLUSIONS

In this paper, we proposed a data management system for healthcare simulations. We clarified the importance of a pay-as-you-go data integration approach for simulation data management. After the discussion of the basic concepts, we described our prototypical implementation and the validation of our concepts. Finally, we discussed related work.

Our modular framework for heterogeneous data and input adapters facilitates automatic initial technical integration. Additionally, visualization and annotations enable deferred semantic integration. This "data first, structure later" approach minimizes the initial integration effort. Searching and querying the data is possible utilizing a keyword search and SPARQL. Finally, semantic integration enables enhanced query possibilities like domain specific query languages. We validated our approach with a prototypical implementation of our framework.

In future work, we will improve our framework by adding support for various other types of data sources and improving usability. Additionally, we will add facilities for automatic query generation to our frontend to support simulation modelers. Finally, we will evaluate our approach in the context of our simulation project.

ACKNOWLEDGEMENTS

This project is supported by the German Federal Ministry of Education and Research (BMBF), project grant No. 13EX1013B.

REFERENCES

Arthofer, K., Girardi, D., and Giretzlehner, M. (2012). Ein ontologiebasiertes system zum extrahieren, transformieren und laden in krankenanstalten. In *Proceedings of the eHealth2012*.

Baumgärtel, P., Endler, G., and Lenz, R. (2013). A benchmark for multidimensional statistical data. In Catania, B., Guerrini, G., and Pokorn, J., editors, *Advances in Databases and Information Systems*, volume 8133 of *Lecture Notes in Computer Science*, pages 358–371. Springer Berlin Heidelberg.

Baumgärtel, P. and Lenz, R. (2012). Towards data and data quality management for large scale healthcare simulations. In Conchon, E., Correia, C., Fred, A., and Gamboa, H., editors, *Proceedings of the International Conference on Health Informatics*, pages 275–280. SciTePress - Science and Technology Publications.

Baumgärtel, P., Tenschert, J., and Lenz, R. (2014). A query language for workflow instance data. In Catania, B., Cerquitelli, T., Chiusano, S., Guerrini, G., Kmpf, M., Kemper, A., Novikov, B., Palpanas, T., Pokorn, J., and Vakali, A., editors, *New Trends in Databases and Information Systems*, volume 241 of *Advances in Intelligent Systems and Computing*, pages 79–86. Springer International Publishing.

Das Sarma, A., Dong, X., and Halevy, A. (2008). Bootstrapping pay-as-you-go data integration systems. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, SIGMOD '08, pages 861–874, New York, NY, USA. ACM.

Djanatliev, A., Kolominsky-Rabas, P., Hofmann, B. M., and German, R. (2012). Hybrid simulation with loosely coupled system dynamics and agent-based models for prospective health technology assessments. In *Proceedings of the 2012 Winter Simulation Conference*.

Franklin, M., Halevy, A., and Maier, D. (2005). From databases to dataspaces: a new abstraction for information management. *SIGMOD Rec.*, 34:27–33.

Howe, B., Cole, G., Souroush, E., Koutris, P., Key, A., Khoussainova, N., and Battle, L. (2011). Database-as-a-service for long-tail science. In Bayard Cushing, J., French, J., and Bowers, S., editors, *Scientific and Statistical Database Management*, volume 6809 of *Lecture Notes in Computer Science*, pages 480–489. Springer Berlin / Heidelberg.

Jablonski, S., Volz, B., Rehman, M., Archner, O., and Cur, O. (2009). Data integration with the dalton framework - a case study. In Winslett, M., editor, *Scientific and Statistical Database Management*, volume 5566 of *Lecture Notes in Computer Science*, pages 255–263. Springer Berlin / Heidelberg.

Jeffery, S. R., Franklin, M. J., and Halevy, A. Y. (2008). Pay-as-you-go user feedback for dataspace systems. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, SIGMOD '08, pages 847–860, New York, NY, USA. ACM.

Lassila, O., Swick, R. R., Wide, W., and Consortium, W. (1999). Resource description framework (rdf) model and syntax specification. `http://www.w3.org/TR/1999/REC-rdf-syntax-19990222`.

Lenz, R. (2009). Information systems in healthcare - state and steps towards sustainability. *IMIA Yearbook 2009*, 1:63–70.

Lenz, R., Beyer, M., and Kuhn, K. A. (2007). Semantic integration in healthcare networks. *International Journal of Medical Informatics*, 76(2-3):201 – 207.

Nadkarni, P. M., Marenco, L., Chen, R., Skoufos, E., Shepherd, G., and Miller, P. (1999). Organization of heterogeneous scientific data using the eav/cr representation. *Journal of the American Medical Informatics Association*, 6(6):478–493.

Parnas, D. L. (1994). Software aging. In *Proceedings of the 16th international conference on Software engineering*, ICSE '94, pages 279–287, Los Alamitos, CA, USA. IEEE Computer Society Press.

Prud'hommeaux, E. and Seaborne, A. (2008). Sparql query language for rdf. `http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/`.

Rahm, E. and Bernstein, P. A. (2001). A survey of approaches to automatic schema matching. *The VLDB Journal*, 10:334–350. 10.1007/s007780100057.

Reimann, P., Reiter, M., Schwarz, H., Karastoyanova, D., and Leymann, F. (2011). Simpl - a framework for accessing external data in simulation workflows. In *Datenbanksysteme für Business, Technologie und Web (BTW)*.

Robertson, N. and Perera, T. (2002). Automated data collection for simulation? *Simulation Practice and Theory*, 9(6-8):349 – 364.

Rogers, J., Simakov, R., Soroush, E., Velikhov, P., Balazinska, M., DeWitt, D., Heath, B., Maier, D., Madden, S., Patel, J., Stonebraker, M., Zdonik, S., Smirnov, A., Knizhnik, K., and Brown, P. G. (2010). Overview of scidb, large scale array storage, processing and analysis. In *Proceedings of the SIGMOD'10*.

Sheth, A. P. and Larson, J. A. (1990). Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Comput. Surv.*, 22:183–236.