Query-driven Data Integration (Short Paper)

Peter K. Schwab, Andreas M. Wahl, Richard Lenz, and Klaus Meyer-Wegener

Friedrich-Alexander-Universität Erlangen-Nürnberg,
Technische Fakultät, Department Informatik,
Lehrstuhl für Informatik 6 (Datenmanagement),
Martensstr. 3, 91058 Erlangen, Germany
{peter.schwab,andreas.wahl,richard.lenz,klaus.meyer-wegener}@fau.de
https://www6.cs.fau.de

Abstract. The paper describes an ongoing project that pursues the idea of query-driven data integration. Instead of first creating a common global schema and fetching, transforming, and loading the data to be integrated, we start with the queries. They are taken as a specification of information need and thus as the overall purpose of integration. Two repositories are being developed, one for all information related to the queries and one for potential data sources to which those queries may refer. Queries may have very different forms, and thus there are many different ways how they can be used to make the integration effort more efficient.

Keywords: Database, Query, Data integration

1 Introduction

Data integration [5] is a task found in many enterprises, causing tremendous amounts of repeated work. Often, an additional data source is considered interesting and thus a complex process of data integration is started. A heavy-weight example is the merger of two companies. Usually, some kind of ETL ("extract – transform – load") process is created with great effort, so that the data from one system can be made available in the other.

Since the effort is substantial and in some cases even prohibiting, other approaches like "pay as you go" have been proposed [4]. They postpone the integration effort to the time when the data are actually needed. However, they still focus on the (incremental) creation of a common global schema.

We continue this line by proposing to look at the queries first. This is our understanding of the term "query-driven". It does not mean that we ignore the other information on the data that may already be available, but the queries are in the focus. A query is regarded as a specification of information need. Ideally, it is given in formal notation already, but we accept other forms as well. The query is written "as if" the new source(s) had already been integrated. So it is not yet executable. The purpose of data integration is then slightly modified to making that query executable and to do only the part of the integration process that is required to achieve that.

There are many variations and options involved in this scenario: the form of the query, the annotations or hints pointing to potential data sources, a repository of data sources found and used in the past, another repository of queries that have already been used for integration purposes, and many more. The purpose of this paper is to define some of these steps and to propose methods that could be used in them. It is assumed that the whole field is by far too large to be handled in just one project, let alone in one paper.

2 Scenarios

We have been asked to help in data-integration tasks many times. Our insistence to first name a few queries caused some confusion in the beginning, because even the users often had the idea to collect lots of data first and to only then think about possible accesses and analyses. It was not so difficult, however, to convince them that taking the queries into account would help to focus the development, and save time and resources.

One of these scenarios is the Aroma-Research Database. Here, we can show an example of a real "query", see Fig. 1, that has been given to us in the very beginning. The elements of this data sheet are all coming from different data sources, some even from remote sites. The details are not important here. Excel is used so far to enter the data into the system, and the users would like to continue to do so. So tools extracting data from Excel tables and formatting them appropriately [7, 10] must be used. The knowledge of the queries allows to restrict the effort to the data that are known to be included in the result.

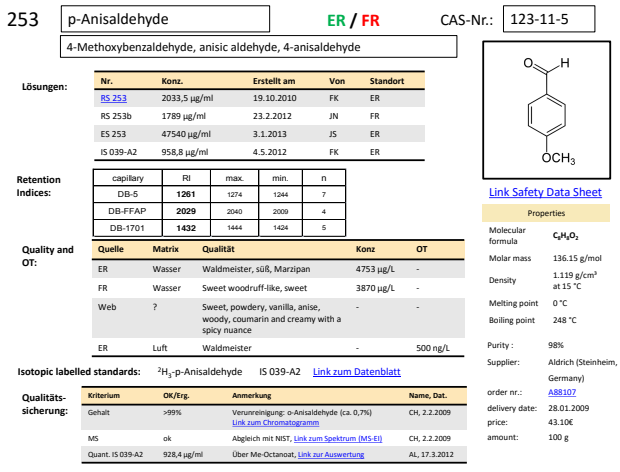

Fig. 1. A "query" given to us by the users of the Aroma-Research Database

A second scenario is currently called "Walhalla DB". The purpose is to allow music researchers to view different aspects of an opera (we use Richard Wagner's "Walküre" – Valkyrie – as an example) simultaneously. These are: a video of some performance, the libretto, and the music sheets. The idea is to allow for flexible navigation in any of the views, and then synchronizing the other views. Here, the queries are defined by the given user interface. It is still not fixed, but already defines a set of accesses to the underlying data sets, which may be calls of a database API as well as database queries. The relational database at the moment has only one table (which is actually the data-entry mode preferred by the music researchers) and will be re-designed in a proper way on the base of the given queries. Images of music sheets are stored separately. And the performance video is accessed via YouTube. The results of the design together with the final queries will be published in due course.

In co-operations with industrial partners, we often found the wish to associate own data with publicly available data, e. g. weather information or tweets. It is interesting how the weather may have influenced sales numbers, or how positive or negative tweets had such an effect. This strongly suggests to take the integration of linked open data [8, 1] into account, as it has already been demonstrated by the OPEN project [2].

So the demand for integration is obvious, and in all these scenarios we envision a benefit from looking at the queries early.

3 Query formulation

Apart from the world of formal query languages, there is also another world even more relevant to many users: result tables or reports, as in the scenario of the Aroma-Research Database. Users can easily sketch a form or report with the data they would like to combine, each field potentially coming from a different source. The data sources are often known, but provide only semi-structured formats (e. g. Excel, PDF, HDFS). They may be located in other departments or even enterprises.

Since experts must transform these specifications into more formal languages anyway, the world of query languages must be investigated, too. Here, queries may now include elements that are not yet found in the given schema as in DrillBeyond [6]. They may be accompanied with hints on the potential sources for these elements. This triggers a process of searching, downloading, transformation, and storage, which is quite usual. The difference now is that the query is already given, so the effort can be tailored to it.

We plan to look at different query languages here, but to get started, we will concentrate on SQL first.

4 Integration with Execution

The degree to which the queries identify data sources may vary. Sometimes the sources are well-known, as in the cases of the Aroma-Research Database and

the Walhalla DB. Sometimes there is the mere hope that appropriate open data sources can be found the World-wide Web. Then searching for data sources must be the first step. Since this may be a significant effort, we propose to keep the information on data sources that have been found and have proven useful in answering former queries. Knowing the query may help further, e. g. if it includes a join with local tables. In this case the join attribute—name as well as type, and values—can give hints on the characteristics of the data source being sought.

Once the source has been found, execution of the query may begin, doing integration on the fly. This can mean fetching data from the source and transforming them to the format required for further processing. It may as well mean to ship some sub-query to the source mostly to select and filter the data before fetching them. The latter requires that the source can execute these sub-queries—and of course the knowledge of the query in the first place. It is a commonplace that this can reduce the amount of data to be transmitted substantially.

In the end, the data coming from the source needs to be transformed to the format required. Again, we see a benefit in having the query at hand and not just creating some part of a common (global) schema, because we transform exactly those parts that will be included in the final processing or the query result.

As far as the query result contributes to the construction of a global schema, it may of course be used for that, too. This, however, is done *after* the execution, so that it does not postpone the delivery.

5 Repository

We are currently designing a repository for queries and any information related to them[1]. SQL as a query language will be dominant in the beginning, but we try to remain open for other query languages and forms. Not surprisingly, the amount of information associated with a single query can be quite substantial. It includes:

- the query expression in text form,
- a query specification in form of example results,
- the abstract syntax tree of the query,
- the relevant parts of the query as separate features (relations used, attributes returned, selection predicates, grouping, etc.),
- a summary of the query result,
- frequency of query execution,
- importance of query execution[2],
- reference to the software that invokes the query,
- execution cost or time of the query,
- the query execution plan in some system at hand,
- and many others.

[1] It is understood that this design takes the queries to be executed on the repository into account.

[2] Some kind of “emergency” query may be very rare, yet very important once executed.

The repository will be designed with evolution in mind, that is, beginning with a rather small subset of attributes and features, but with a potential to grow.

The other repository keeps track of the data sources. It makes sense to remember the data sources that have already been found as well as the experience with them. Their usefulness in at least some context should be saved. This includes a notion of trust and reliability. Also, the mapping and transformation of data elements from that source should be kept.

Both repositories will be linked. This allows to find the data sources that have previously been used by a query. Still, the user may decide to replace them by better sources found in the meantime. And it allows to find all the queries that have used a given data source in the past. Their owners may be informed about changes in that source, up to the fact that it may no longer be available.

6 Related Work

The literature on data integration is tremendous. The book by Doan et al. [5] may be considered the standard work now. Hardly any of the approaches, however, are query-driven. Most of them focus on the construction of a global schema; not necessarily before using the system, but at least before executing a query.

The work on incremental integration is also helpful when doing it query-driven. The Data Tamer system [12] (now a product named "Tamr") includes many techniques for that. The already-mentioned systems OPEN and DrillBeyond [2, 6] are query-driven to some extent, and provide very useful mechanisms for the integration of external data sources, which can be adapted to work with our approach. Query-driven schema expansion as described in [11] is similar to our approach, but concentrates on a rather specific form of external data, namely ratings based on crowd sourcing. The example used is the rating of movies. Database reverse engineering and SQL tracking [3] can be used to feed a query repository. Collaborative query management [9] also creates a repository, but for a different purpose: User are supported in writing queries by searching for similar queries.

7 Summary and Outlook

The impetus of the project described in this paper is to take queries into account when designing a data management system that integrates heterogeneous data sources. This has many aspects that still need to be investigated in more detail. Queries can be specified in many different ways with different properties, and their execution can be prepared and finally done along many different paths. It is our belief that it is worthwhile to follow at least some of them.

We have already taken a first step as documented in [13]. The repository can be initialized with the help of query logs. The article proposes a particular approach to evaluate query logs in the context of data integration. The information required to do this will also be available in the repository sketched here. The query log is used to extract knowledge about data sources and thus contributes

to the contents of the second repository. This knowledge can then be utilized by the data scientists interacting with the data-integration system. The queries may contain fictional tables and attributes, and the system helps to find appropriate data sources. This generates a kind of kernel for the first repository.

Please note that it is not necessary to take a puristic view. The query-driven approach can easily be combined with the classical approach (that we may now call schema-driven). The message here is simply to consider the queries as well. Whatever knowledge about the data and their preliminary structure is available, it should certainly not be ignored.

Acknowledgement: The authors would like to thank the anonymous reviewers for their valuable remarks.

References

1. Bizer, C., Heath, T., Berners-Lee, T.: Linked data – the story so far. Int. Journal on Semantic Web & Information Systems 5(3), 1–22 (2009)
2. Braunschweig, K., Eberius, J., Thiele, M., Lehner, W.: OPEN - enabling non-expert users to extract, integrate, and analyze open data. Datenbank-Spektrum 12(2), 121–130 (2012)
3. van den Brink, H.J., van der Leek, R.C.: Quality metrics for SQL queries embedded in host languages. In: Proc. Special Session on System Quality and Maintainability (SQM, March 20) in conjunction with 11th European Conf. on Software Maintenance and Reengineering: "Software Evolution in Complex Software Intensive Systems" (CSMR, Amsterdam, the Netherlands, March 21-23). p. 2 (2007)
4. Das Sarma, A., Dong, X., Halevy, A.: Bootstrapping pay-as-you-go data integration systems. In: Proc. SIGMOD. pp. 861–874. ACM, New York, NY, USA (2008)
5. Doan, A., Halevy, A., Ives, Z.: Principles of Data Integration. Morgan Kaufmann, Waltham, MA, USA (2012)
6. Eberius, J., Thiele, M., Braunschweig, K., Lehner, W.: DrillBeyond: Processing multi-result open world SQL queries. In: Proc. 27th Int. Conf. on SSDBM. pp. 16:1–16:12. ACM (2015)
7. Eberius, J., Werner, C., Thiele, M., Braunschweig, K., Dannecker, L., Lehner, W.: DeExcelerator: a framework for extracting relational data from partially structured documents. In: Proc. CIKM. pp. 2477–2480 (2013)
8. Heath, T., Bizer, C.: Linked Data: Evolving the Web into a Global Data Space. Synthesis Lectures on the Semantic Web: Theory and Technology, Morgan & Claypool (2011)
9. Khoussainova, N., Balazinska, M., Gatterbauer, W., Kwon, Y., Suciu, D.: A case for a collaborative query management system. In: Proc. CIDR (2009)
10. Le, V., Gulwani, S.: FlashExtract: A framework for data extraction by examples. In: Proc. PLDI (Edinburgh, United Kingdom, June 9-11). pp. 542–553 (2014)
11. Selke, J., Lofi, C., Balke, W.T.: Pushing the boundaries of crowd-enabled databases with query-driven schema expansion. PVLDB 5(6), 538–549 (2012)
12. Stonebraker, M., Bruckner, D., Ilyas, I.F., Beskales, G., Cherniack, M., Zdonik, S.B., Pagan, A., Xu, S.: Data curation at scale: The Data Tamer system. In: Proc. CIDR (2013)
13. Wahl, A.M.: A minimally-intrusive approach for query-driven data integration systems. In: Proc. IEEE 32nd ICDE Workshops (ICDEW) (2016)