DrillBeyond: Enabling Business Analysts to Explore the Web of Open Data

Julian Eberius, Maik Thiele, Katrin Braunschweig and Wolfgang Lehner
Technische Universität Dresden
Faculty of Computer Science, Database Technology Group
01062 Dresden, Germany
[firstname.lastname]@tu-dresden.de

ABSTRACT

Following the *Open Data* trend, governments and public agencies have started making their data available on the Web and established platforms such as `data.gov` or `data.un.org`. These Open Data platforms provide a huge amount of data for various topics such as demographics, transport, finance or health in various data formats. One typical usage scenario for this kind of data is their integration into a database or data warehouse in order to apply data analytics. However, in today's business intelligence tools there is an evident lack of support for so-called situational or ad-hoc data integration. In this demonstration we will therefore present *DrillBeyond*, a novel database and information retrieval engine which allows users to query a local database as well as the Web of Open Data in a seamless and integrated way with standard SQL. The audience will be able to pose queries to our DrillBeyond system which will be answered partly from local data in the database and partly from datasets that originate from the Web of Data.
We will show how such queries are divided into known and unknown parts and how missing attributes are mapped to open datasets. We will demonstrate the integration of the open datasets back into the DBMS in order to apply its analytical features.

1. INTRODUCTION

The concept of *Open Data* describes information that is freely available and can be used as well as republished by everyone without restrictions from copyright or patents. The goal of the Open Data movement is to open all non-personal and non-commercial data, especially (but not exclusively) all data collected and processed by government organizations. This applies to all data which has an infrastructural role essential for the civil use or the scientific endeavor or which was generated with the help of public money, for example, maps, medical science, government spendings or environmental pollution data. Ideally, making this data available on the web would lead to more transparency, participation and innovation throughout society. The essence of Open Data is to enable as many people as possible to reuse this public asset, to bring it into a new context, to mix it with other data sources and to create knowledge and value out of it.
To make productive use of the variety of Open Data, two things are required: first, a way to integrate Open Data into a common representation, second a way to analyze the integrated data to make it usable. From a data management perspective this is typically done using a data warehouse stack with its ETL processes (extract, transform and load) and well-defined analytical database schemata. However, these processes and schemata are rather inflexible and may only be modified within well-defined development cycles, which prevents situational and ad-hoc integration of arbitrary web data sources.
Consider a business user performing OLAP style analysis on a data warehouse. For our demo we use, amongst others, the well known TPC-H benchmark schema. Let us assume that the users are analyzing the orders with respect to the nations they originate from and they would like to classify them by some criterion that is not part of the local schema, e.g., the nation's population. The new open data repositories of countries or organizations are useful sources for this kind of structured information, e.g., the World Bank provides the portal `data.worldbank.org` with global per-country economic indicators. However, searching these repositories and integrating the acquired datasets with local data is a tedious and error-prone task left to the user.
In this demonstration we therefore propose our Open World approach DrillBeyond, which enables users to query a local database together with the Web of Open Data. DrillBeyond unburdens the users from tasks such as finding the best open datasets that fits their needs, downloading and integrating them into the local database schema and rewriting their own queries. Our demo will show the potential of a combined database and information retrieval engine in order to perform powerful queries over databases and the Web of Data in an efficient, easy-to-use and seamless manner.

2. OPEN-WORLD QUERIES

The system we want to demonstrate offers a novel way of dealing with the scenario described above. We present a modified relational database management system that is able to answer so-called open-world queries which are not restricted to the schema of the local database, i.e. the user is allowed to use arbitrary attribute names that do not appear in the original schema. An open-world query on the `Nation`

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Articles from this volume were invited to present their results at The 38th International Conference on Very Large Data Bases, August 27th - 31st 2012, Istanbul, Turkey.
Proceedings of the VLDB Endowment, Vol. 5, No. 12
Copyright 2012 VLDB Endowment 2150-8097/12/08... $ 10.00.

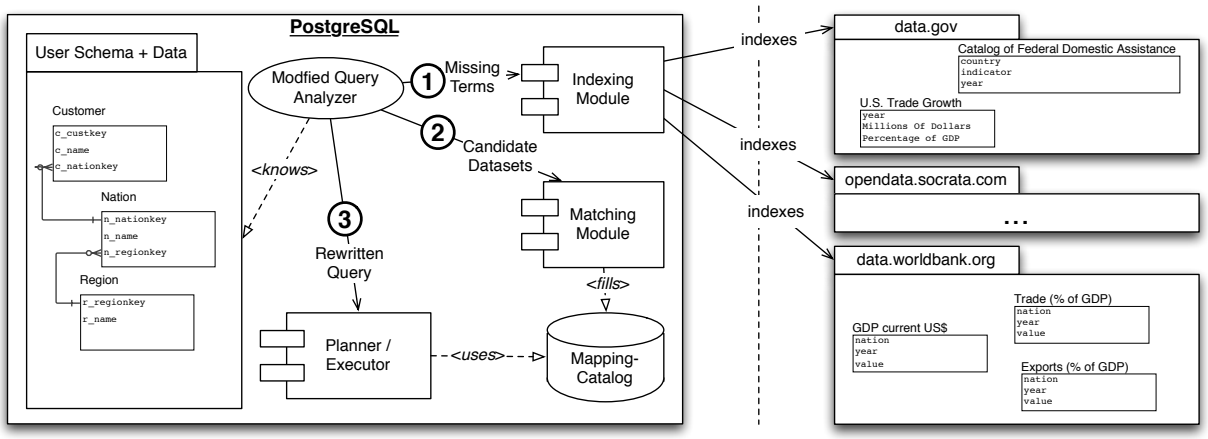

Figure 1: Query Processing and relations to remote data sources

table might look like this:

```
select population, n_name, avg(o_totalprice)
    from nation
    join customer on n_nationkey=c_nationkey
    join orders on c_custkey=o_custkey
group by population, n_name
order by population
```

The `population` attribute which is used in the `Select` and `Order By` clauses is not part of the TPC-H schema and therefore has to be processed in a different way. Missing attributes are translated into keyword queries that are run against an index of open datasets on the web. The DrillBeyond system will answer the query by substituting the missing attribute values, e.g. for *population*, with values from the retrieved open datasets. In contrast to pure search systems, DrillBeyond is able to exploit the local schema and data as well as the context given by the SQL query to find the best matching open datasets. As usual with IR style approaches, the system will not be able to find the optimal dataset completely automatically, but instead needs to present the user with a ranked result list. Our system handles this ambiguities on two levels: 1) in the query processing and the schema of the query result and 2) through a web GUI tool that guides the user through query answering process.
Apart from identifying open datasets that can be potentially used to answer the open-world SQL, the challenges lie in identifying the attributes of the open datasets that contain the correct values, as well as identifying those attributes that can be used to join the local table and the open data table. For our running example it would be possible to retrieve a dataset from `data.worldbank.org` which contains the population values for each country, and join it with TPC-H's nation table using the `n_name` attribute. Then, the query can be answered like a regular SQL query. The architecture of our DrillBeyond system, which is able to process this new type of query, is outlined in the following section.

3. SYSTEM OVERVIEW

The DrillBeyond system is implemented inside the open source RDBMS PostgreSQL in its current version 9.1. The following changes and additions to the system were made:

1. **Modified Query Analyzer and Rewriter**
2. **Additional IR-style Indexing Module**
3. **Additional Schema/Instance Matching Module**

We will describe these components and the query preprocessing they perform in the following paragraphs. All other components of the system, such as the query optimizers, or join implementations, can be reused unmodified, as the output of the preprocessing steps is a standard SQL query referencing standard database objects. Figure 1 gives a global overview of the demo system, including the new DBMS components and query processing steps.

3.1 Modified Query Analyzer and Query Rewriting

In a regular DBMS, the query analyzer maps tokens from the SQL query to objects in the database, e.g., the token `n_name` to the corresponding attribute in the nation table. If a token can not be mapped to a database object, an error is raised. For this demo, we modified PostgreSQL's query analyzer to trigger additional query processing steps when it encounters an unknown token. Specifically, the query processor not only the unknown token as input for the Indexing Module, but also the token's *context*, i.e. the database objects and the instance data that are related to the token. For example, if the unknown token is used in the `select` clause, it must be related to one of the relations given in the `from` clause. Continuing the running example, when the token `population` appears in the query's select clause as shown in our example query, then the system concludes that `population` must be an attribute of the local relation `nation`. In this case, this can easily be deduced from the fact that there is only one relation given in the query. If more relations are given, the users can prefix the relation name, e.g., `nation.population`), as they would do in regular SQL.
The second part of the query context extracted by the analyzer is instance data. In our example, if we have concluded that `nation` is the table related to `population`, we use instance data from the `nation` table to aid the search for a fitting join partner in the DrillBeyond index as described the following sections. After collecting this context information, the analyzer calls the two new modules, the Matching- and

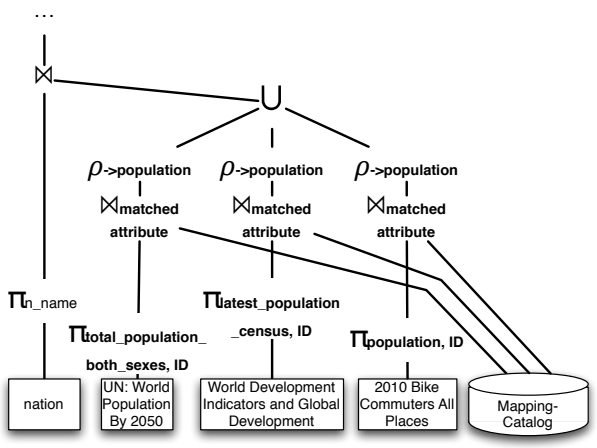

Figure 2: Query Rewrite for $\pi_{n\_name,population}(Nation)$

the Indexing Module, performs query rewriting and finally passes control to the regular optimizer and the executor. These new steps in query processing are depicted as number 1 to 3 in Figure 1. First, the extracted keyword token and its context are passed to the Indexing Module, which returns a list of candidate datasets (details in Section 3.2). This list is passed to the Matching Module (Step 2) that checks which of those candidates can be joined with the local data at all, and if possible saves mappings in the catalog (details in Section 3.3). In a third step the original, unoptimized query plan is rewritten to include a new attribute created from the candidates. Figure 2 shows an example of such a rewrite for the selection of the `population` attribute from the nation table. Notice that for each of the identified candidates, the value attribute is selected as chosen by the Indexing Module. Furthermore, a data source ID is attached to each tuple which will be used to structure the result presented to the user. Then each dataset is joined with the mapping catalog table, which contains matching value pairs from the matching columns of the local and the open datasets, as determined by the Matching Module. The attributes are renamed to satisfy the outer query and the union of the join results is created. Finally, a simple join can be performed to construct the result of the query. The following table shows an excerpt from such a result table:

n_name	population	source_ID
France	55.0	1
Germany	77.0	1
Russia	148.5	1
France	21	2
Germany	15	2
Russia	9	2
France	57,000,000	3
Germany	82,000,000	3
Russia	*NULL*	3

Notice that it contains several values for the each tuple of the local table, representing the possible ways of adding the missing attribute. If such an rewrite occurs inside a more complex query, some constraints apply: aggregation or sort operators further up in the plan need to include the source ID attribute in their grouping or ordering predicates to produce one correct result per external data source. In addition, `Limit` clauses on the whole query have to be propagated to the individual subqueries. Furthermore, in order to support the `Union` operator, the data types of the respective value attributes must be aligned. The partitioned result table can then be investigated by the user to find the best candidate directly, or by using more advanced GUI tools as we will show in Section 3.4. Since the raw SQL result does not contain any metadata about the data sources, we modified the PostgreSQL's `explain` statement to provide metadata for a given query, e.g., the source URLs and metadata and schemata of the datasets used for the query. This additional information helps the users to decide for the optimal candidate.

In the following sections, we will give more details on the indexing and matching modules as well as on the web front used in the demo.

3.2 Indexing Module

The Index Module keeps an index of datasets available on open data repositories on the web. It indexes the datasets metadata found on the repository, such as title, description and tags. Furthermore, it indexes each dataset's attributes, and the metadata given for those. For this demo, the index is realized using PostgreSQL's native text search capabilities which support normalization / stemming and boolean keyword queries. We added synonym expansion via WordNet[1] to be able to identify additional candidate datasets. A lookup in the index is performed using the unknown tokens and their context, as passed from the query analyzer (see Section 3.1). The result ranking is a mixture of classical keyword-search ranking, e.g., comparing the query tokens to dataset metadata and attribute names, but also instance-based techniques, which are applied in the Matching Module to refine the ranking. Continuing our running example, the index lookup will identify the queried terms and its context in several datasets, and pass them to the next stage.

3.3 Matching Module

Since there are no foreign-key relationships between the local and open datasets, a join can only be performed when at least one matching column pair of the local table and the respective open dataset can be identified. Therefore, the Matching Module employs a set of classic schema and instance matching techniques, such as string similarity measures and external knowledge such as synonym dictionaries. By doing this we are able to rank the result candidates produced by the index lookup and to prune all datasets which can not be joined. The ranking is influenced mainly by the quality of the mapping, e.g., the monogamy and coverage of the created mapping between two datasets. It is also determined by data quality aspects such as the number of null values in the value attribute in the matched tuples of the open dataset. If a join candidate is found, the instance level mappings between the matching attributes are stored in the mapping catalog to establish a foreign key relationship between the two datasets. The stored mappings are later used to perform the joins to produce the actual query result. Continuing our running example, the Matching Module will match the country names from the Worldbank population tables to the `n_name` attribute in the TPC-H `nation` table.

[1] http://wordnet.princeton.edu/

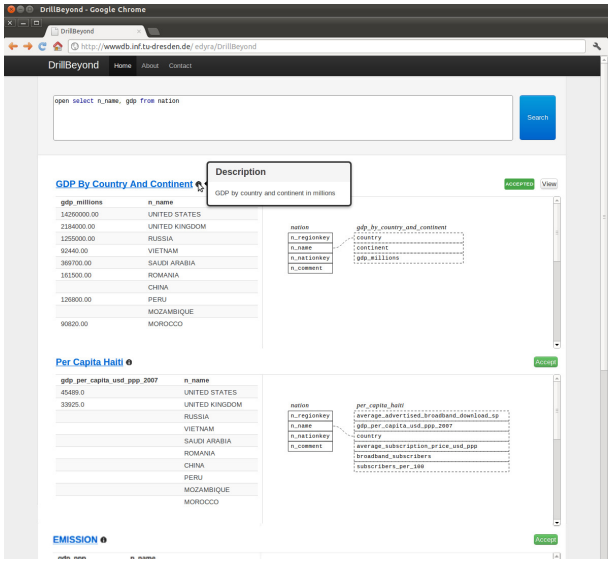

Figure 3: Screenshot of the web front end used for the demonstration

In this example, the small variations in the nation names (e.g. `RUSSIA` in TPC-H versus `Russian Federation` in the WorldBank dataset) can be solved using string similarity measures.

3.4 Web Frontend

In addition to the PostgreSQL back end, we implemented a web front end which enables the interactions with the user, such as presentation of the search results and selection of a fitting dataset from the candidates. Figure 3 gives an impression of this interface. The users can enter regular SQL queries, and browse candidate datasets in a standard search engine layout. In contrast to document search engines, for each candidate, the front end will display the dataset's schema, the attributes matching the local table, the attributes (potentially) containing the missing values as well as sample rows of the query result when the respective candidate is chosen. Finally, for each open dataset the associated metadata as delivered by the dataset's source can be viewed. This allows the users to make a more informed choice about which open datasets to use to complement their data.

4. DEMO WALKTHROUGH

In this live demonstration, users will be able to get a feeling for the potential of Open Data in data analytics by posing SQL queries including undefined attributes to the DrillBeyond system. Using an console or the web interface, they will be able to choose from different preloaded local databases to perform analysis on. The preloaded schemata include two well known warehouse schemata, TPC-H and FoodMart. Then, the users can enter SQL queries on the chosen schemata, using open attributes as they see fit.
The demo system will consult its index of open datasets, which for this demo, contains about 5,000 indicators from `data.worldbank.org`, 5,400 datasets from `data.un.org`, as well as about 16,000 datasets from `opendata.socrata.com`. Depending on the tool used, the audience will be able to study query results as one raw SQL result table on the console, as shown in Section 3.1, or as individual query results depending on a selected candidate when using the web interface as shown in Figure 3. In the web front end they also have access to the metadata, schema and sample rows of the candidate datasets. A screencast demonstrating both the raw SQL console as well as the web front end is available at `http://wwwdb.inf.tu-dresden.de/edyra/DrillBeyond`.

5. RELATED WORK

Enabling keyword-based search over relational data is a current field of interest in the database research community, see for example [1]. These systems answer keyword queries on local relational data by translating the keyword queries to SQL and are usually used for point or fact queries. In contrast, our system uses keyword lookups to identify join candidates in a collection of open datasets. Bernstein et al. [2] provide a recent review of the methods of schema matching which are applied in the Mapping Module. CrowdDB [3] is a very similar system in that it also enables open-world queries in a classic relational context. It employs a crowdsourcing approach to complement missing values or tuples. In contrast to that, the DrillBeyond system retrieves them in a semi-automatic fashion from structured open datasets.

6. CONCLUSION & FUTURE WORK

We have presented the *DrillBeyond* system, an extended RDBMS that is able to process SQL queries with unknown attributes. In contrast to regular SQL queries, these queries allow the usage of attribute names which do not appear in the existing local schema. The queries are processed by looking up open data sets on the web and matching selected datasets to the local data in such a way that the query can be answered. In the demo, users can pose SQL queries to the system using various local schemata and open data repositories as basis for their queries, using both a raw SQL console or a web tool that aides the process of finding, integrating and querying open datasets. In this way they can explore the potential of open data for business intelligence and data analysis in general. Future work is planned on extending the index and match components which are using off-the-shelf components at the moment. Furthermore, the query rewriting constraints discussed in Section 3.1 have to be explored further and possible optimization rules for plans containing open attributes must be considered. Another aspect would be to extend the system to allow distributed query processing, offloading parts of the matching and candidate retrieval to the data sources themselves, eliminating the need to fetch all candidate datasets for matching.

7. REFERENCES

[1] S. Bergamaschi, E. Domnori, F. Guerra, R. Trillo Lado, and Y. Velegrakis. Keyword search over relational databases: a metadata approach. SIGMOD, pages 565–576, 2011.

[2] P. A. Bernstein, J. Madhavan, and E. Rahm. Generic schema matching, ten years later. *VLDB*, pages 695–701, 2011.

[3] M. J. Franklin, D. Kossmann, T. Kraska, S. Ramesh, and R. Xin. CrowdDB: answering queries with crowdsourcing. SIGMOD, pages 61–72, 2011.