Rule-Based Impact Analysis for Enterprise Business Intelligence

Kalle Tomingas, Tanel Tammet, Margus Kliimask

Rule-based Impact Analysis for Enterprise Business Intelligence

Kalle Tomingas[1], Tanel Tammet[1], Margus Kliimask[2]

[1] Tallinn University of Technology, Ehitajate tee 5, Tallinn 19086 Estonia
[2] Eliko Competence Center, Teaduspargi 6/2, Tallinn 12618 Estonia

Abstract. We address several common problems in the field of Business Intelligence, Data Warehousing and Decision Support Systems: the complexity to manage, track and understand data lineage and system component dependencies in long series of data transformation chains. The paper presents practical methods to calculate meaningful data transformation and component dependency paths, based on program parsing, heuristic impact analysis, probabilistic rules and semantic technologies. Case studies are employed to explain further data aggregation and visualization of the results to address different planning and decision support problems for various user profiles like business users, managers, data stewards, system analysts, designers and developers.

Keywords: impact analysis, data lineage, data warehouse, rule-based reasoning, probabilistic reasoning, semantics

1 Introduction

Developers and managers are facing similar Data Lineage (DL) and Impact Analysis (IA) problems in complex data integration (DI), business intelligence (BI) and Data Warehouse (DW) environments where the chains of data transformations are long and the complexity of structural changes is high. The management of data integration processes becomes unpredictable and the costs of changes can be very high due to the lack of information about data flows and internal relations of system components. The amount of different data flows and system component dependencies in a traditional data warehouse environment is large. Important contextual relations are coded into data transformation queries and programs (e.g. SQL queries, data loading scripts, open or closed DI system components etc.). Data lineage dependencies are spread between different systems and frequently exist only in program code or SQL queries. This leads to unmanageable complexity, lack of knowledge and a large

amount of technical work with uncomfortable consequences like unpredictable results, wrong estimations, rigid administrative and development processes, high cost, lack of flexibility and lack of trust. We point out some of the most important and common questions for large DW environments (see Fig.1) which usually become a topic of research for system analysts and administrators:

- Where does the data come or go to in a specific column, table, view or report?
- Which components (reports, queries, loadings and structures) are impacted when other components are changed?
- Which data, structure or report is used by whom and when?
- What is the cost of making changes?
- What will break when we change something?

Fig. 1. General scheme of DW/BI data flows.

The ability to find ad-hoc answers to many day-to-day questions determines not only the management capabilities and the cost of the system, but also the price and flexibility of making changes. The dynamics in business, environment and requirements ensure that regular changes are a necessity for every living organization. Due to its reflective nature, the business intelligence is often the most fluid and unsteady part of enterprise information systems.

Obviously, the most promising way to tackle the challenges in such a rapidly growing, changing and labor-intensive field is automation. We claim that efficient automatization in this particular field requires the use of semantic and probabilistic technologies. Our goal is to aid the analysts with tools which can reduce several hard tasks from weeks to minutes, with better precision and smaller costs.

2 Related Work

Impact analysis, traceability and data lineage issues are not new. A good overview of the research activities of the last decade is presented in an article by Priebe et al. [9]. We can find various research approaches and published papers from the early 1990s with methodologies for software traceability [10]. The problem of data lineage tracing in data warehousing environments has been formally founded by Cui and Widom [2]

and [3]. Our recent paper builds upon this theory by introducing the Abstract Mapping representation of data transformations [14].

Other theoretical works for data lineage tracing can be found in [5] and [6]. Fan and Poulovassilis developed algorithms for deriving affected data items along the transformation pathway [5]. These approaches formalize a way to trace tuples (resp. attribute values) through rather complex transformations, given that the transformations are known on a schema level. This assumption does not often hold in practice. Transformations may be documented in source-to-target matrices (specification lineage) and implemented in ETL tools (implementation lineage).

Other practical works that based on conceptual models, ontologies and graphs for data quality and data lineage tracking can be found in [15] and [12]. De Santana proposes the integrated metadatada and the CWM metamodel based data lineage documentation approach [4]. The workflows and the manual annotations based solution proposed by Missier et al. [8].

Priebe et al. [9] concentrates on proper handling of specification lineage, a huge problem in large-scale DWH projects, especially in case different sources have to be consistently mapped to the same target. They propose a business information model (or conceptual business glossary) as the solution and a central mapping point to overcome those issues.

Our approach to Impact Analysis and Data Lineage differs from previous work in several aspects. Our aim is to merge technical data lineage [3] with semantic integration approaches [9], [11], using grammar based methods for metadata extraction from program texts and a probabilistic rule-based inference engine for weight calculations and reasoning approaches [7]. We also use the novel and powerful web based data flow and the graph visualization techniques with the multiple view approach [16] to deliver the extraction and the calculation of the result to the end-users.

3 System Architecture

We present a working Impact Analysis solution which can be adopted and implemented in an enterprise environment or provided as a service (SaaS) to manage organization information assets, analyze data flows and system component dependencies. The solution is modular, scalable and extendable. The core functions of our system architecture are built upon the following components presented in the Fig.2.

1. Scanners collect metadata from different systems that are part of DW data flows (DI/ETL processes, data structures, queries, reports etc.). We build scanners using our xml-based data transformation language and runtime engine XDTL [18].
2. The SQL parser is based on customized grammars, GoldParser parsing engine [1] and the Java-based XDTL engine.

3. The rule-based parse tree mapper extracts and collects meaningful expressions from the parsed text, using declared combinations of grammar rules and parsed text tokens.
4. The query resolver applies additional rules to expand and resolve all the variables, aliases, sub-query expressions and other SQL syntax structures which encode crucial information for data flow construction.
5. The expression weight calculator applies rules to calculate the meaning of data transformation, join and filter expressions for impact analysis and data flow construction.
6. The probabilistic rule-based reasoning engine propagates and aggregates weighted dependencies.
7. The directed and weighted sub-graph calculations, visualization and web based UI for data lineage and impact analysis applications.
8. The MMX open-schema relational database using PostgreSQL or Oracle for storing and sharing scanned, calculated and derived metadata [17].

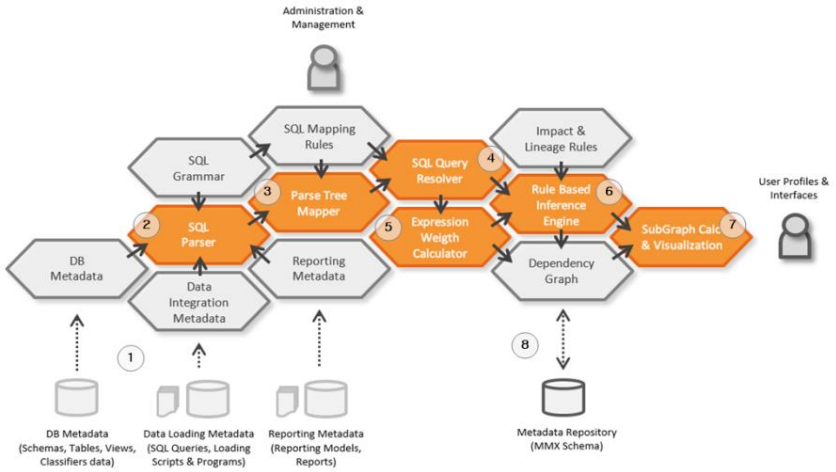

Fig. 2. Impact Analysis system architecture components.

4 Query Parsing and Metadata Extraction

Scanners (No1 in Fig.2) collect metadata from external systems: database data dictionary structures, ETL system scripts and queries, reporting system query models and reports. All the structural information extracted is stored to the metadata database. The scanned objects and their properties are extracted and stored according to the meta-models we have designed: relational database, data integration, reporting and

business terminology models. Meta-models contain ontological knowledge about the metadata and relations collected across different domains and models. The scanner technology (XDTL) and the open-schema metadata database (MMX) design have been described at a more detailed level in our previous work [13],[14].

In order to construct the data flows from the very beginning of the data sources (e.g. the accounting system) to the end points (e.g. the reporting system) we have to be able to find and connect both the identical and the related objects in different systems. In order to connect the objects we have to understand and extract the relations from the SQL queries (e.g. ETL tasks, database views, database procedures), scripts (e.g. loader utility scripts) and expressions (e.g. report structure) collected and stored by scanners. In order to understand the data transformation semantics encoded in the query language statements (e.g. insert, update, select and delete queries) and expressions, we have to involve external knowledge about the syntax and grammatical structure of the query language. Grammar-based parsing functionality is built into the scanner technology. A configurable "parse" command brings semi-structured text parsing and information extraction into the XDTL data integration environment. As the result of SQL parsing step (No2 in Fig.2) we get a large parse tree with every SQL query token assigned a special disambiguated meaning by the grammar.

In order to convert different texts into the tree structure, to reduce tokens and to convert the tree back to the meaningful expressions (depending on search goals), we use a declarative rule set presented in the Json format, combining token and grammar rules. A configurable grammar and a synchronized reduction rule set makes the XDTL parse command suitable for general purpose information extraction and captures the resource hungry computation steps into one single parse-and-map step with the flat table outcome. The Parse Tree Mapper (No3 in Fig.2) uses three different rule sets with more than eighty rules to map the parse tree to data transformation expressions.

After extraction and mapping of each SQL query statement into a series of expressions we execute the SQL Query Resolver (No4 in Fig.2) which contains a series of functions to resolve the SQL query structure:

- Solve source and target object aliases to full qualified object names;
- Solve sub-query aliases to context specific source and target object names;
- Solve sub-query expressions and identifiers to expand all the query level expressions and identifiers with fully qualified and functional ones;
- Solve syntactic dissymmetry in different data transformation expressions (e.g. insert statement column lists, select statement column lists and update statement assign list etc.);
- Extract quantitative metrics from data transformation, filter and join expressions to calculate expression weights (e.g. number of columns in expression, functions, predicates, string constants, number constants etc.).

5 Data Transformation Weight Calculation

Data structure transformations are parsed, extracted from queries and stored as formalized, declarative mappings in the system. To add additional quantitative measures to each column transformation or column usage in the join and filter conditions we evaluate each expression and calculate transformation and filter weights for those.

Expression Weight Calculation (No5 in Fig.2) is based on the idea that we can evaluate column data "transformation rate" and column data "filtering rate" using data structure and structure transformation information captured from SQL queries. Such a heuristic evaluation enables us to distinguish columns and structures used in the transformation expressions or in filtering conditions or both, and gives probabilistic weights to expressions without the need to understand the full semantics of each expression. We have defined two measures that we further use in our probabilistic rule system for deriving new facts:

Definition 1. A primitive data transformation operation $O(X,Y,M1,F1,W_t)$ is a data transformation between a source column X and a target column Y in a transformation set M (mapping or query) having expression similarity weight W_t and having conditions set F1.

Definition 2. Column transformation weight W_t is based on the similarity of each source column and column transformation expression: the calculated weight expresses the source column transfer rate or strength. The weight is calculated on scale [0,1] where 0 means that the source data are not transformed to target (e.g. constant assignment in a query) and 1 means that the source is directly copied to the target (no additional column transformations).

Definition 3. Column filter weight W_f is based on the similarity of each filter column in the filter expression and the calculated weight expresses the column filtering rate or strength. The weight is calculated on scale [0,1] where 0 means that the column is not used in the filter and 1 means that the column is directly used in the filter predicate (no additional expressions).

The general column weight W algorithm in each expression for W_t and W_f components is calculated as a column count ratio over all the expression component counts (e.g. column count, constant count, function count, predicate count):

```
W=IdCount/IdCount+FncCount+StrCount+NbrCount+PrdCount.
```

5.1 Rule System

The defined figures, operations and weights are used in combination with the declarative probabilistic inference rules to calculate the possible relations and dependencies between data structures and software components. Applying the rule system to the extracted query graphs we calculate and produce a full dependency graph that is used for data lineage and impact analysis.

The basic operations used in the rules for the dependency graph calculations are the following:

- The primitive data transformation is the elementary operation between the source column X and the target column Y in the query mapping id set M1 with the filter condition set F1 and the transformation weight W_t (see Definition 1). This is represented by the predicate `O(X,Y,M1,F1,Wt);`
- The predicate `member(X,F1,Wt)` is used in the filter impact calculation rule to detect that the column X is a member of the filter id set F1 with the filter weight Wt;
- The predicate `disjoint(M1,M2)` is used in the impact aggregation rule to detect that two query mapping id sets M1 and M2 are disjoint. The disjointness condition is necessary for aggregating the data transformation relations and weights in case more than one path from different queries connects the same column pairs;
- The predicate `parent(X0,X1)` is used in the parent aggregation rule to detect that the table X0 is the owner or parent object of the column X1. This condition is necessary for aggregating all the column level relations and the weights between two tables for the table level impact relation;
- The function `union(M1,M2)` is used to calculate the impact relations over two disjoint query id sets M1 and M2: it returns the distinct id lists of two sets M1 and M2;
- The function `sum(W1,W2)` is used to calculate the aggregated impact relation weight in case the basic operations are disjoint, i.e. stem from independent queries. The weight calculation is based on non-mutually-exclusive event probabilities (two independent queries means there could be an overlap between two events) and is calculated as the sum of probabilities of W1 and W2: `sum(W1,W2)=(W1+W2)-(W1*W2);`
- The function `avg(W1,W2)` is used to calculate the parent impact weight when the basic operations have the same parent structures. The weight is calculated as the arithmetic mean of W1 and W2: `avg(W1,W2)=(W1+W2)/2;`

The inference rules with the basic operations and the weighs for the dependency graph calculations are the following:

- The basic impact calculation rule for the operation O with no additional filters produces the impact predicate I:
  `O(X,Y,M1,F1,Wt) => I(X,Y,M1,F1,Wt);`
- The basic impact calculation rule for the operation O with a related filter condition produces the impact predicate I with multiplied weights:
  `O(X,Y,M1,F1,Wt)& member(X,F1,Wf) => I(X,Y,M1,F1,Wt*Wf);`
- The transitivity rule is used to calculate the sequences of the consecutive impact relations:

```
I(X,Y,M1,F1,W1) & I(Y,Z,M2,F2,W2) & disjoint(M1,M2) =>
          I(X,Z,union(M1,M2),union(F1,F2),W1*W2);
```

- The column aggregation rule is used when multiple different paths from the different queries connect the same columns: calculate the impact relations with aggregated query id-s and the aggregated weights:

```
I(X,Z,M1,F1,W1) & I(X,Z,M2,F2,W2) & disjoint(M1,M2) =>
          I(X,Z,union(M1,M2),union(F1,F2),sum(W1,W2));
```

- The parent aggregation rule is used when multiple different impact relations connect the column pairs of the same tables: calculate the table level impact relations with aggregated query id-s and aggregated weights:

```
I(X1,Y1,M1,F1,W1) & I(X2,Y2,M2,F2,W2) & parent(X0,X1)& par-
ent(X0,X2) & parent(Y0,Y1) & parent(Y0,Y2) =>
          I(X0,Y0,union(M1,M2),union(F1,F2),avg(W1,W2)).
```

5.2 Dependency Score Calculation

We can use the derived dependency graph to solve different business tasks by calculating the selected component(s) lineage or impact over available layers and details chosen details. Business questions like: "What reports are using my data?", "Which components should be changed or tested?" or "What is the time and cost of change?" are converted to directed sub-graph navigation and calculation tasks. The following definitions add new quantitative measures to each component or node (e.g. table, view, column, etl task, report etc.) in the calculation. We use those measures in the UI to sort and select the right components for specific tasks.

Definition 4. Local Lineage Dependency % (LLD) is calculated as the ratio over the sum of the local source and target Lineage weights W_t:

$LLD = SUM(source(W_t)/source(W_t)+target(W_t))$.

Local Lineage Dependency 0 % means that there are no data sources detected for the object. Local Lineage Dependency 100 % means that there are no data consumers (targets) detected for the object. Local Lineage Dependency about 50 % means that there are equal numbers of weighted sources and consumers (targets) detected for the object.

Definition 5. Local Impact Dependency % (LID) is calculated as the ratio over the sum of local source and target impact weights $W(W_t,W_f)$:

$LID=SUM(source(W) /source(W)+target(W))$.

Local Impact Dependency 0 % means that there are no dependent data sources detected for the object. Local Dependency 100 % means that there are no dependent data consumers (targets) detected for the object. Local Impact Dependency about 50 % means that there are equal numbers of weighted dependent sources and consumers (targets) detected for the object.

Definition 6. Global Dependency Count (GDC) is the sum of all source and target Lineage and Impact relations counts: `GDC=GSC+GTC`.

The Global Dependency Count is a good differential metric that allows us to see clear distinctions in the dependencies of each object. We can take the GDC metric as a sort of "gravity" of the object that can be used to develop new rules, to infer the time and cost of changes of object(s) (e.g. database table, view, data loading programs or report).

6 Conclusions and Future Work

The previously described architecture and algorithms have been used to implement an integrated toolset dLineage (http://dlineage.com). Both the scanners and web-based tools of dLineage have been enhanced and tested in real-life projects and environments to support several popular DW database platforms (e.g. Oracle, Greenplum, Teradata, Vertica, PostgreSQL, MsSQL, Sybase), ETL tools (e.g. Pentaho, Oracle Data Integrator, SQL scripts and different data loading utilities) and BI tools (e.g. SAP Business Objects, Microstrategy). The dLineage dynamic visualization and graph navigation tools are implemented in Javascript using the d3.js graphics libraries.

We have tested our solution during two main case studies involving a thorough analysis of large international companies in the financial and the energy sectors. Both case studies analyzed thousands of database tables and views, tens of thousands of data loading scripts and BI reports. Those figures are far over the capacity limits of human analysts not assisted by the special tools and technologies.

We have presented several algorithms and techniques for quantitative impact analysis, data lineage and change management. The focus of these methods is on automated analysis of the semantics of data conversion systems followed by employing probabilistic rules for calculating chains and sums of impact estimations. The algorithms and techniques have been successfully employed in two large case studies, leading to practical data lineage and component dependency visualizations.

We are planning to continue this research by considering a more abstract, conceptual/business level in addition to the current physical/technical level of data representation.

Acknowledgments

This research has been supported by European Union through European Regional Development Fund.

References

1. Cook,D. (2010). Gold parsing system. URL: http://www.goldparser.org.
2. Cui, Y., Widom, J., & Wiener, J. L. (2000). Tracing the lineage of view data in a warehousing environment. ACM Transactions on Database Systems (TODS), 25(2), 179-227.
3. Cui, Y., & Widom, J. (2003). Lineage tracing for general data warehouse transformations. The VLDB Journal—The International Journal on Very Large Data Bases, 12(1), 41-58.
4. de Santana, A. S., & de Carvalho Moura, A. M. (2004). Metadata to support transformations and data & metadata lineage in a warehousing environment. In Data Warehousing and Knowledge Discovery (pp. 249-258). Springer Berlin Heidelberg.
5. Fan, H., & Poulovassilis, A. (2003, November). Using AutoMed metadata in data warehousing environments. In Proceedings of the 6th ACM international workshop on Data warehousing and OLAP (pp. 86-93). ACM.
6. Giorgini, P., Rizzi, S., & Garzetti, M. (2008). GRAnD: A goal-oriented approach to requirement analysis in data warehouses. Decision Support Systems, 45(1), 4-21.
7. Luberg, A., Tammet, T., & Järv, P. (2011). Smart City: A Rule-based Tourist Recommendation System. In Information and Communication Technologies in Tourism 2011 (pp. 51-62). Springer Vienna.
8. Missier, P., Belhajjame, K., Zhao, J., Roos, M., & Goble, C. (2008). Data lineage model for Taverna workflows with lightweight annotation requirements. In Provenance and Annotation of Data and Processes (pp. 17-30). Springer Berlin Heidelberg.
9. Priebe, T., Reisser, A., & Hoang, D. T. A. (2011). Reinventing the Wheel?! Why Harmonization and Reuse Fail in Complex Data Warehouse Environments and a Proposed Solution to the Problem.
10. Ramesh, B., & Jarke, M. (2001). Toward reference models for requirements traceability. Software Engineering, IEEE Transactions on, 27(1), 58-93.
11. Reisser, A., & Priebe, T. (2009, August). Utilizing Semantic Web Technologies for Efficient Data Lineage and Impact Analyses in Data Warehouse Environments. In Database and Expert Systems Application, 2009. DEXA'09, pp. 59-63.
12. Skoutas, D., & Simitsis, A. (2007). Ontology-based conceptual design of ETL processes for both structured and semi-structured data. International Journal on Semantic Web and Information Systems (IJSWIS), 3(4), 1-24.
13. Tomingas,K., Kliimask,M., Tammet,T. (2014). Mappings, Rules and Patterns in Template Based ETL Construction. The 11th International Baltic DB&IS2014 Conference.
14. Tomingas,K., Kliimask,M., Tammet,T. (2014). Data Integration Patterns for Data Warehouse Automation. The 18th East-European ADBIS2014 Conference.
15. Vassiliadis, P., Simitsis, A., & Skiadopoulos, S. (2002, November). Conceptual modeling for ETL processes. In Proceedings of the 5th ACM international workshop on Data Warehousing and OLAP (pp. 14-21). ACM.
16. Wang Baldonado, M. Q., Woodruff, A., & Kuchinsky, A. (2000, May). Guidelines for using multiple views in information visualization. In Proceedings of the working conference on Advanced visual interfaces (pp. 110-119). ACM.
17. MMX Metadata Framework, URL: http://mmxframework.org
18. XDTL Data Transformation Language, URL: http://xdtl.org