Data Lineage and Impact Analysis

THE ART OF DATA LINEAGE

Common problems in the field of Business Intelligence, Data Warehousing and Decision Support Systems are the complexity to manage, track and understand data lineage and system component dependencies in long series of data transformation chains. We provide practical solution and technology to capture, calculate and visualize meaningful data transformation and component dependency paths, based on program parsing, heuristic impact analysis, probabilistic rules and semantic technologies. We address different planning and decision support problems for various user profiles like business users, managers, data stewards, system analysts, designers and developers.

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 which usually become a topic of research for system analysts and administrators:

- Where does the data come or go to in/from a specific column, table, view or report?
- When was the data loaded, updated or calculated 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?

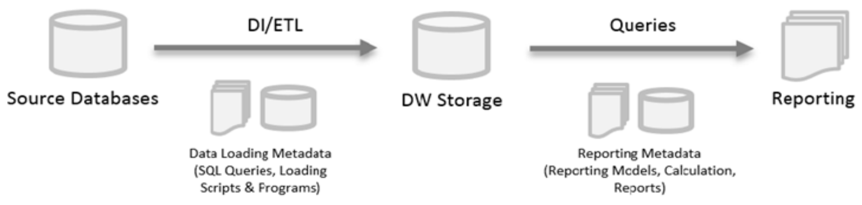

DI/ETL
Queries
Source Databases
DW Storage
Reporting
Data Loading Metadata (SQL Queries, Loading Scripts & Programs)
Reporting Metadata (Reporting Models, Calculation, Reports)

Figure 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.

FEATURES

- **Automates end-to-end data lineage and impact analysis with subsequent transformations, conditions and data mappings**
- **Visualises data lineage and impact analysis in clickable navigable graph**
- **Provides an integrated metadata repository for all business and technical enterprise metadata**
- **Provides a wide range of scanners for various data sources: DB, ETL and BI tools**
- **Aligns business terminology and metrics with IT assets**
- **Reduces evaluation and decision time for project and change management from weeks to minutes or hours**
- **Increases decision-making agility by supplying readily available and current information**
- **Provides a solid and automated system documentation**

SYTEM 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, analyse 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:

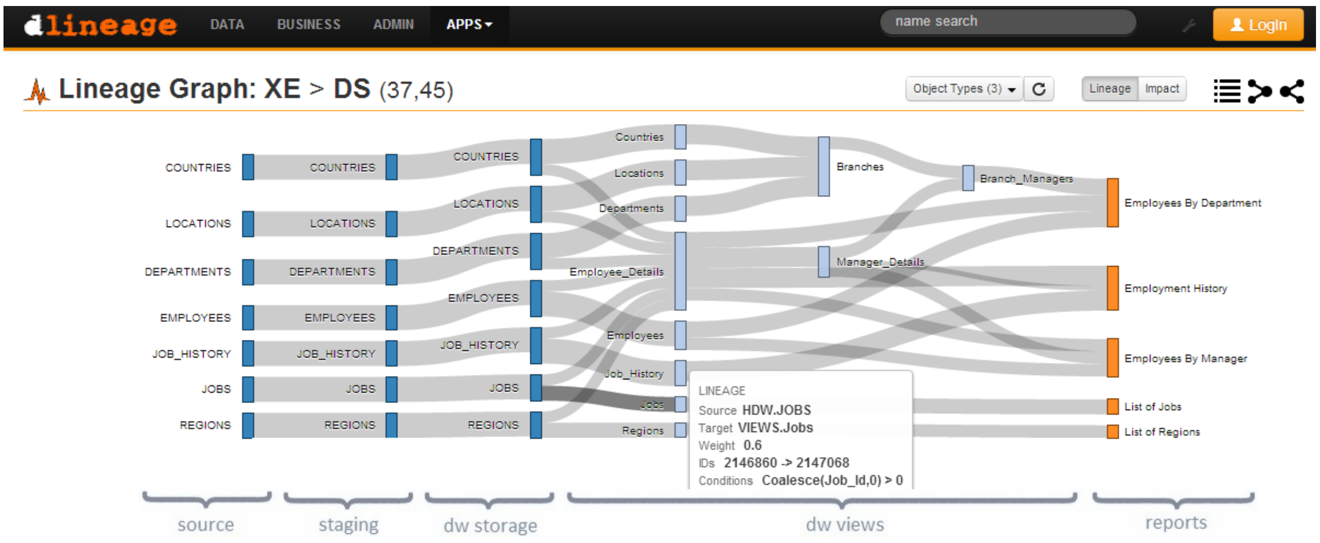
- Scanners collect metadata from different systems that are part of DW data flows (DI/ETL processes, data structures, queries, reports etc.). We build scanners using our xml-based data transformation language and runtime engine XDTL written in Java.
- The SQL parser is based on customized grammars, GoldParser parsing engine and the Java-based XDTL engine.
- 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.
- 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.
- The expression weight calculator applies rules to calculate the meaning of data transformation, join and filter expressions for impact analysis and data flow construction.
- The probabilistic rule-based reasoning engine propagates and aggregates weighted dependencies.
- The directed and weighted sub-graph calculations, visualization and web based UI for data lineage and impact analysis applications.
- The MMX open-schema relational database using PostgreSQL or Oracle for storing and sharing scanned, calculated and derived metadata.

The following user interface screenshots illustrate the complementary navigation and analysis views to understand data warehouse component dependencies (i.e. source tables, temporary loading tables, DW storage tables, data access views and reports), which can be either data sources, data consumers or both.

The data lineage graph (Figure.2) presents a classical example of the data flows of a Human Resource (HR) system in the data warehousing process. The left side of the diagram contains the source system objects and the right side contains the reporting system objects. The nodes in the graph represent database objects (e.g. tables, views) or reporting objects (e.g. reports, fields). The directed arcs represent data flows between the nodes: the calculated transformation probabilities are visualized by the weight and width of the arcs.

A directed data lineage graph (Sankey Diagram) is a technique to visualize the complex network of data flows and component dependencies, used to provide an overview and details for different analytical tasks (see details in the Introduction chapter). The dependency graph has one single focus (e.g. schema, table, column etc.): clicking on the node will change the focus and redraw a new graph, giving the user the experience of zooming in and out, similar to browsing a map.

DW metadata collection and impact analysis result can be seen in clickable and navigable graph form (Figure 3), but also in table or tree forms. Dependency table representation gives complementary view to same data and contains two lists of dependent db and reporting system objects that are sorted in order of calculated dependencies (component impact and data lineage) weights. Dependent objects are sorted by calculated Global Dependency Counts which represent the weight or 'gravity' of each dependency. This metric distinguish objects that gives most of the impact or are barely related with selected objects and it helps users easily find the most important ones depends on the goals.



LINEAGE
Source HDW.JOBS
Target VIEWS.Jobs
Weight 0.6
IDs 2146860 -> 2147068
Conditions Coalesce(Job_Id,0) > 0

Figure 2. Data Lineage and Impact Analysis sample graph view

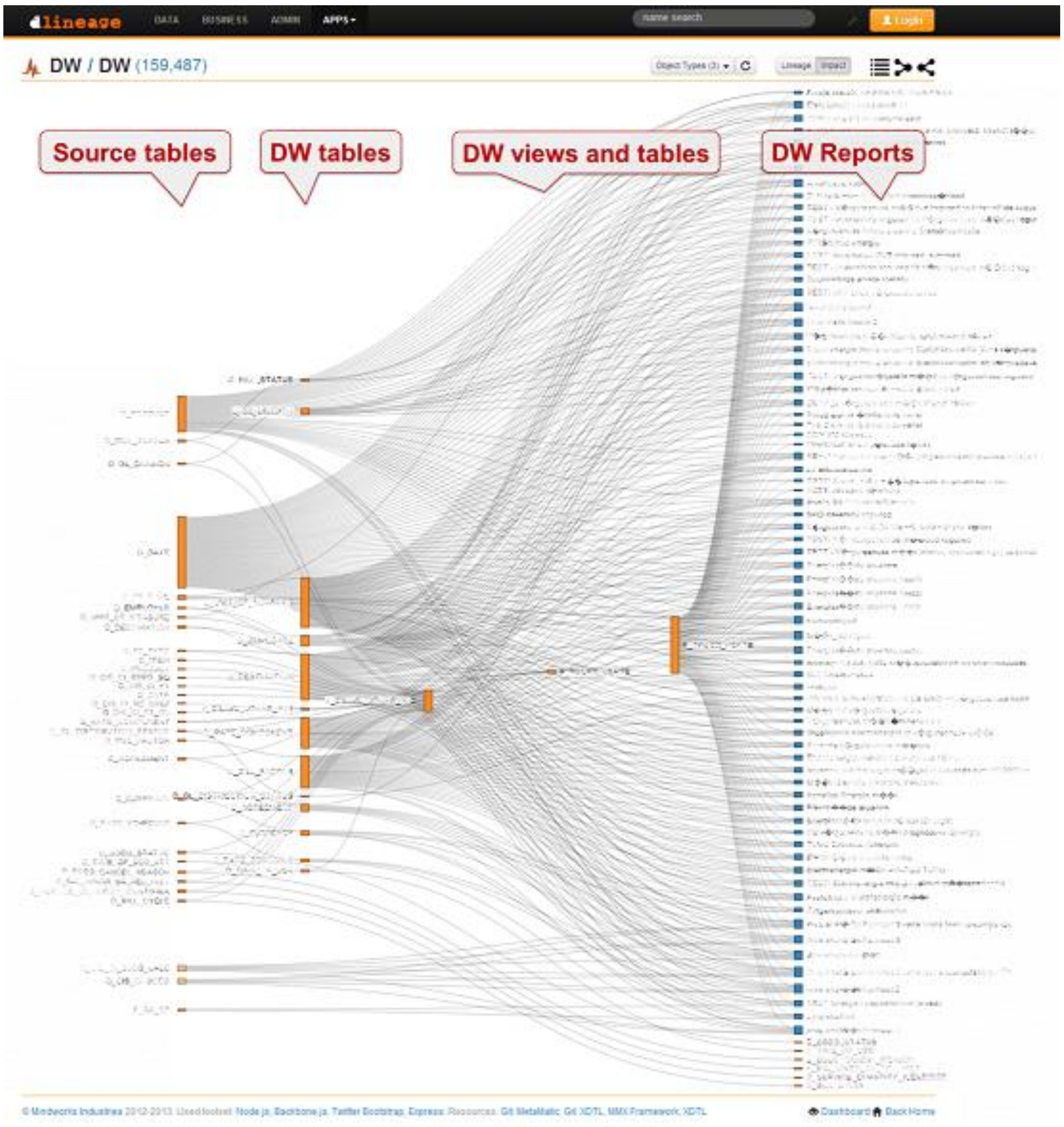


Figure 3. Impact Analysis application screenshot

QUERY PARSING AND METADATA EXTRACTION

Scanners collect external systems metadata like database data dictionary structures, ETL system scripts and queries or reporting system query models and reports, and all structural information extracted and stored to metadata database. All scanned objects and their properties are extracted and stored according to defined meta-models like relational database, data integration, reporting and business terminology models. Meta-models contain ontological knowledge about collected metadata and relations across different domains and models.

To construct 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 should be able to connect same and related objects in different systems.

To connect objects we have to understand and extract the relations from SQL queries (e.g. ETL tasks, database views, database procedures) and scripts (e.g. loader utility scripts) and expressions (e.g. report structure) that are collected and stored by scanners. In order to understand data transformation semantics that are captured into the query language statements (e.g. insert, update, select and delete queries) and expressions, we have to involve external knowledge about the query language syntax and grammatical structure. We use a general purpose Java based GoldParser engine and we have developed a custom SQL grammar which is written in Extended Backus-Naur Form (EBNF). Our grammar is based on ANSI/SQL syntax, but it contains a big set of dialect specific notations, syntax elements and functions that are developed and trained using large real life SQL query corpuses from the field of data warehousing. The current grammar edition is based on Teradata, Oracle, Greenplum, Vertica, Postgres, Sybase and MsSQL dialects.

REAL LIFE CASE STUDY

The following real-life Enterprise Dependency Graph example (Figure 4) is an illustration of the complex dependency structures between the DW storage scheme, access views and user reports. The example is generated using only 3-4 data lineage layers (sources and ETL layers are not present here) and has details at object level (not at column level). When going to column and report filed level then the full data lineage graph would be about 10 times bigger and much more complex to fully visualize.

The sample Data Lineage Graph from Data Warehouse tables to views and user reports has:

- 5 000 nodes (tables, views, reports)
- 20 00 links (data transformations in views and queries)

TECHNICAL SPECIFICATIONS

Data Lineage works with a wide variety of databases, applications, languages, and other tools. Scanners are designed as open scripts or components and arbitrary data sources can be added in each implementation process.

- **Databases:** Oracle, Teradata, Greenplum, PostgreSQL, Vertica, Sybase, MSSQL, DB2, MySQL, MS Access
- **BI Tools:** Business Objects, Microstrategy*
- **ETL Tools:** IBM DataStage*, Microsoft DTS/SSIS, Oracle ODI (Sunopsis), Pentaho, Loader scripts (Teradata FastLoad, MultiLoad, Oracle SQL*Loader), SQL scripts*, Informatica PowerCenter*
- **Modeling Tools:** Erwin, Enterprise Architect (XMI)
- **Other Structures:** CSV, DBF, Excel, XML, JSON*

** work in progressor and/or requires customisation for each implementation*

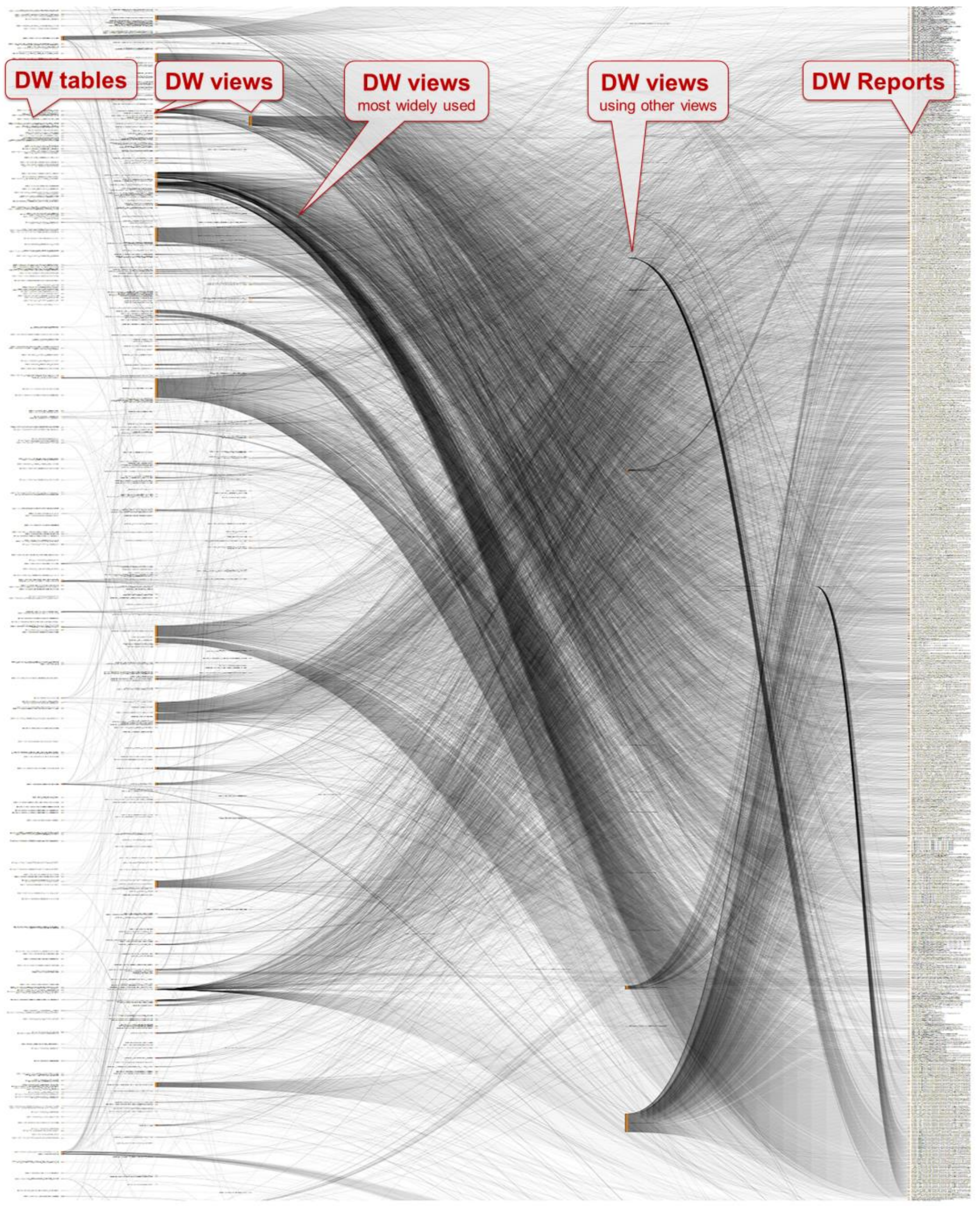


Figure 4. Data Lineage graph with dependencies between DW tables, views and reports.