Graph-based Data Integration and Business Intelligence with BIIIG

André Petermann
University of Leipzig
petermann@informatik.uni-leipzig.de

Martin Junghanns
University of Leipzig
junghanns@informatik.uni-leipzig.de

Robert Müller
Leipzig University of Applied Sciences
mueller@fbm.htwk-leipzig.de

Erhard Rahm
University of Leipzig
rahm@informatik.uni-leipzig.de

ABSTRACT

We demonstrate BIIIG (Business Intelligence with Integrated Instance Graphs), a new system for graph-based data integration and analysis. It aims at improving business analytics compared to traditional OLAP approaches by comprehensively tracking relationships between entities and making them available for analysis. BIIIG supports a largely automatic data integration pipeline for metadata and instance data. Metadata from heterogeneous sources are integrated in a so-called Unified Metadata Graph (UMG) while instance data is combined in a single integrated instance graph (IIG). A unique feature of BIIIG is the concept of business transaction graphs, which are derived from the IIG and which reflect all steps involved in a specific business process. Queries and analysis tasks can refer to the entire instance graph or sets of business transaction graphs. In the demonstration, we perform all data integration steps and present analytic queries including pattern matching and graph-based aggregation of business measures.

1. INTRODUCTION

In the last decades, technologies for business intelligence have been adopted by many enterprises. Most prevailing are data warehouse and OLAP approaches based on relational databases [3]. By contrast, graph-based business intelligence is a fairly new approach. Compared to traditional approaches, graph data models promise significant benefits in terms of analytical flexibility, in particular to evaluate relationships without having to predefine them in a rather static data warehouse schema. Powerful graph models such as the property graph model [8] are also a promising basis for data integration as they allow a flexible and uniform representation of heterogeneous metadata and instance objects and relationships. Ongoing activities by large vendors such as SAP [9] or Microsoft [10] underline the relevance of representing and analyzing business data within graph models. Other approaches on graph-based business intelligence [4][11][12] focus on specific analytical problems for still simple graph models.

Proceedings of the VLDB Endowment, Vol. 7, No. 13

To enable flexible business analytics, we are developing a new approach called BIIIG (**B**usiness **I**ntelligence with **I**ntegrated **I**nstance **G**raphs) [6] for graph-based data integration and analysis. It supports three main kinds of graphs based on the property graph model:

Unified Metadata Graph: The UMG serves BIIIG as a generic metadata model to combine the metadata from different data sources. Metadata such as database schemas are represented in an intuitive graph model of classes and associations. Classes either represent master (reference) data such as customers or products or transactional data such as purchase orders or invoices. The UMG components are determined semi-automatically per source and integrated with the help of experts. We also generate mappings between data sources and the UMG.

Integrated Instance Graph: The IIG is the main data store of BIIIG where nodes represent data objects and edges relationships. The IIG is generated in a fully automated manner based on the source-UMG mappings. To make the IIG data self-descriptive and to achieve a high semantic expressiveness, the graph elements are associated to metadata classes and associations.

Business Transaction Graphs: From the IIG, we derive so-called BTGs. A BTG represents a single execution of a business process with all its involved master data objects, transactional data traces and their interdependencies. BTGs thus represent valuable units for analysis to provide insights about the operational business of an enterprise. We provide an algorithm to extract BTGs automatically from the IIG.

In the next section, we introduce the components and processing pipeline of BIIIG in more detail. In Section 3 we provide details about the current implementation. Finally, we describe our demonstration scenario to illustrate the data integration process and to show the analytical value of IIG and BTGs.

2. OVERVIEW OF BIIIG

In this section, we provide an overview of BIIIG; more details about the approach can be found in [6]. In contrast to data warehouses, we do not require defining a global schema for data integration such as a star or snowflake schema. While such an approach serves many OLAP queries, it is often too inflexible as it can only evaluate facts according to the predefined dimensions and relationships. For example, to better understand in which way employees or customers contribute to the profit of an enterprise it is beneficial to

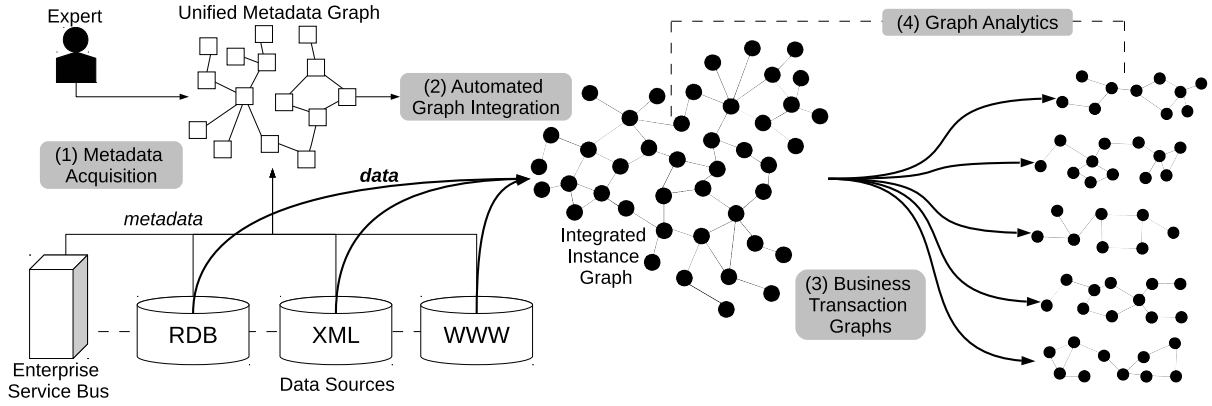

Figure 1: Conceptual Overview of BIIIG

evaluate their involvement and relationships within business processes. BIIIG thus aims at supporting the analysis of relationships between business entities in addition to standard analysis tasks. For this purpose, we follow a bottom-up data integration approach that combines metadata and instance data from relevant data sources in flexible and generic graph models that preserve existing relationships for later analysis.

As illustrated in Fig. 1, there are four main steps in the BIIIG processing pipeline that we will discuss next: metadata acquisition and integration resulting in the UMG, instance integration resulting in the IIG, generation of BTGs, and graph analytics.

2.1 Metadata integration

For every data source such as a database, we first extract the schema of its objects and relationships and translate it to the generic graph format of the *Unified Metadata Graph* (UMG) describing a source in terms of classes and associations. In the UMG, classes are represented as nodes and associations as edges. UMG class definition include the class name, the originating data source, an id attribute as well as a list of class attributes. Furthermore, classes are categorized into transactional or master data. Associations have a name indicating the relationship type as well as a set of relationship attributes. Assocations can link classes from different sources. The UMG also contains a mapping per class and association describing how their instances can be derived from the data sources.

For relational database sources, the generation of the UMG components is relatively straight-forward based on the definitions of tables and foreign key references and largely automatic [6]. Mappings can be expressed as SQL statements. Manual interaction is needed to categorize classes as master or transactional data or to rename classes and associations for improved understandability. Furthermore, cross-system associations need to be defined, e.g. to refer to master data in another source or to link redundant master data classes by *sameAs* associations. *SameAs* associations indicate matching classes; they will be used to identify and fuse together matching instances (see below). The semi-automatic generation of the UMG will be part of the demonstration.

2.2 Instance integration

The main data store of BIIIG is called *integrated instance graph* (IIG). In this graph each data object is represented by a node and each relationship by an edge. Both nodes and edges have mandatory metadata properties as well as arbitrarily many instance properties. Nodes provide either one *source identifier* (concatenated from a class and instance identifier) or a set of such identifiers in the case of fused objects, to enable tracing back any object to its originating source(s).

Instance integration is fully automated based on the source-UMG mappings. This process entails three steps. First, the mappings of all classes are evaluated and a new node is derived for each object. A node is assigned a single source identifier as well as a class name and category. Attribute-value pairs from the source are added as properties. In the second step, the mappings of all associations are evaluated to derive edges for each relationship. Generated edges of the dedicated type *sameAs* connect matching objects and are processed in the last integration step. We resolve these edges by fusing the connected nodes and deleting *sameAs* edges subsequently. A fused node combines source identifiers, properties and relationships of the original nodes.

2.3 Business Transaction Graphs

A main feature of BIIIG is the generation of *business transaction graphs* (BTG) representing interrelated business activities. We observed that certain relationships reflect *causal connections* in terms of business activities, for example a sales order can relate to a preceding quotation. Relationships of that kind are represented as edges between transactional nodes in the IIG. Longer paths of such edges can also be considered as causal connections For example, a quotation may cause a sales order and later an invoice for the order. By contrast, relationships or paths involving master data are generally no hint for a causal connection. For example, two quotations involving the same product can be completely independent. Consequentially, we can consider subgraphs of causally connected transactional nodes as BTGs. We isolate BTGs by an algorithm which starts with an arbitrary transactional node, traverses all causal connections and stops on master data nodes. However, BTGs also include those master data nodes because of their fundamental analytical value. Our algorithm ensures that any node or edge is traversed only once and thus performs in linear time complexity.

2.4 Graph Analytics

Both the IIG and the set of BTGs can be the basis for a comprehensive and flexible business analytics. The current implementation of BIIIG is the foundation of our ongoing research on novel graph-based business analytics including graph mining and the evaluation of relationship patterns. We will also generate relational output from the graphs to leverage existing OLAP approaches for multidimensional analysis in addition to the graph-based evaluations. Currently, BIIIG already offers browsing and querying the different graphs. Analysts can visually navigate through the graphs to access any piece of recorded data with its relationships. Especially BTGs enable an informative view on interrelated business activities recorded in different data sources. Since our current implementation is based on Neo4j (see next section), we can already leverage the declarative query language Cypher [1] on the integrated graph data. Hence, BIIIG already supports analytical graph queries including pattern matching and the aggregation of business measures.

3. IMPLEMENTATION

The architecture of our initial implementation of BIIIG is shown in Figure 2. It consists of three layers that we describe bottom up in the following.

Databases: Unified metadata graph (UMG), integrated instance graph (IIG) and the set of business transaction graphs (BTGs) are stored in separate graph databases. At the moment, we use Neo4j [2] in version 2.0 for all graphs. As Neo4j lacks in support for managing graph collections, we implemented the set of BTGs as a single graph database in which every BTG is represented by an isolated subgraph including redundant master data. To express unique BTG memberships of nodes, all nodes in that database contain a dedicated property `btg_id`.

Back End: The back end of BIIIG is implemented in Java and covers all tasks of actual data processing. The current implementation includes tasks for *metadata acquisition* from relational databases, automated *instance integration* and *BTG isolation*, which are implementions of the corresponding algorithms of [6]. In future developments we will add further tasks, for example for graph mining. The backend provides a REST API to trigger task execution remotely. We access Neo4j using the native Java API which is known to provide the best performance [5].

Front End: The front end provides easy-to-use administrative and analytical facilities for the end user. It is implemented as a Ruby on Rails web application so that all interfaces are accessible using a web browser. Hence, front end and back end services and applications can run on different machines and multiple users can use BIIIG concurrently. For administrative tasks, the *source manager* allows the type-specific configuration of data source connections and the *job scheduler* provides control about data processing tasks of the back end. The *UMG editor* allows manipulating the UMG suggested by metadata acquisition. By this interface, an expert knowing the data sources can enhance the UMG as described in Section 2.1. Finally, analysts can explore all graphs using the *graph browser* and submit analytical queries using the *query interface* with either tabular or visual output. In the current version, the front end of BIIIG integrates the graphical user interface provided by the Neo4j server for both analytical facilities.

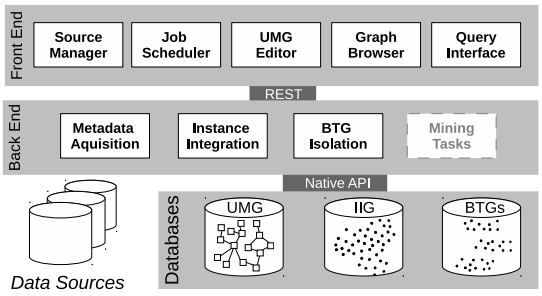

Figure 2: Architecture of BIIIG

4. DEMONSTRATION

During the demonstration, we will present BIIIG as an end-to-end solution for graph-based data integration and business intelligence. As data sources we will use real ERP as well as synthetic data sets generated by the FoodBroker simulation [7]. The FoodBroker data sets provide realistic characteristics and can be scaled to different sizes without introducing and disclosing too many enterprise-specific details. On site, we will demonstrate the data integration process and execute analytical queries on the resulting graphs.

4.1 Data Integration

We will start demonstrating data integration by acquiring metadata from sources to generate an UMG proposal. Then, we will add cross-system associations manually and rename some technical into user-friendly terms in our graphical user interface. Afterwards, we will start automated instance integration. We will then visually browse through the IIG and inspect sample master and transactional data as well as causal connections. Afterwards we will start extracting business transaction graphs. We will visually present selected BTGs to demonstrate their analytical potential. A screenshot of a sample BTG is shown in Fig. 3.

4.2 Business Intelligence

We will demonstrate a variety of analytical queries on the IIG and BTGs exploiting the graph structure and returning result graphs or aggregated relationship patterns and measures. Query suggestions by conference attendees can also be demonstrated. A few example queries are as follows.

IIG 1 - Count Employee Activities: Determine the number of different kinds of business activities per employee. An example result row could be :
$\langle$`SalesOrder`, `processedBy`, `Alice`, 29$\rangle$.

IIG 2 - Customer Interaction Overview: Start at a specific `Customer` node and traverse all paths via transactional to other master data. The result graph contains all business activities involving this customer (sales orders, tickets, ...) including the related master data (products, employees, ...). Browsing this graph, provides a visual overview about the selected customers interaction history.

BTG 1 - Net Profit: For each *BTG*, sum all profit-related properties representing expenses and revenue to determine the net profit of the business process. An example result row could be :
$\langle btg\_id : 456, rev : 82,000, exp : 71,000, profit : 11,000\rangle$

BTG 2 - Complaint Analysis: Find all BTGs having one or more `Ticket` nodes (customer complaints) and determine the involved products and employees.

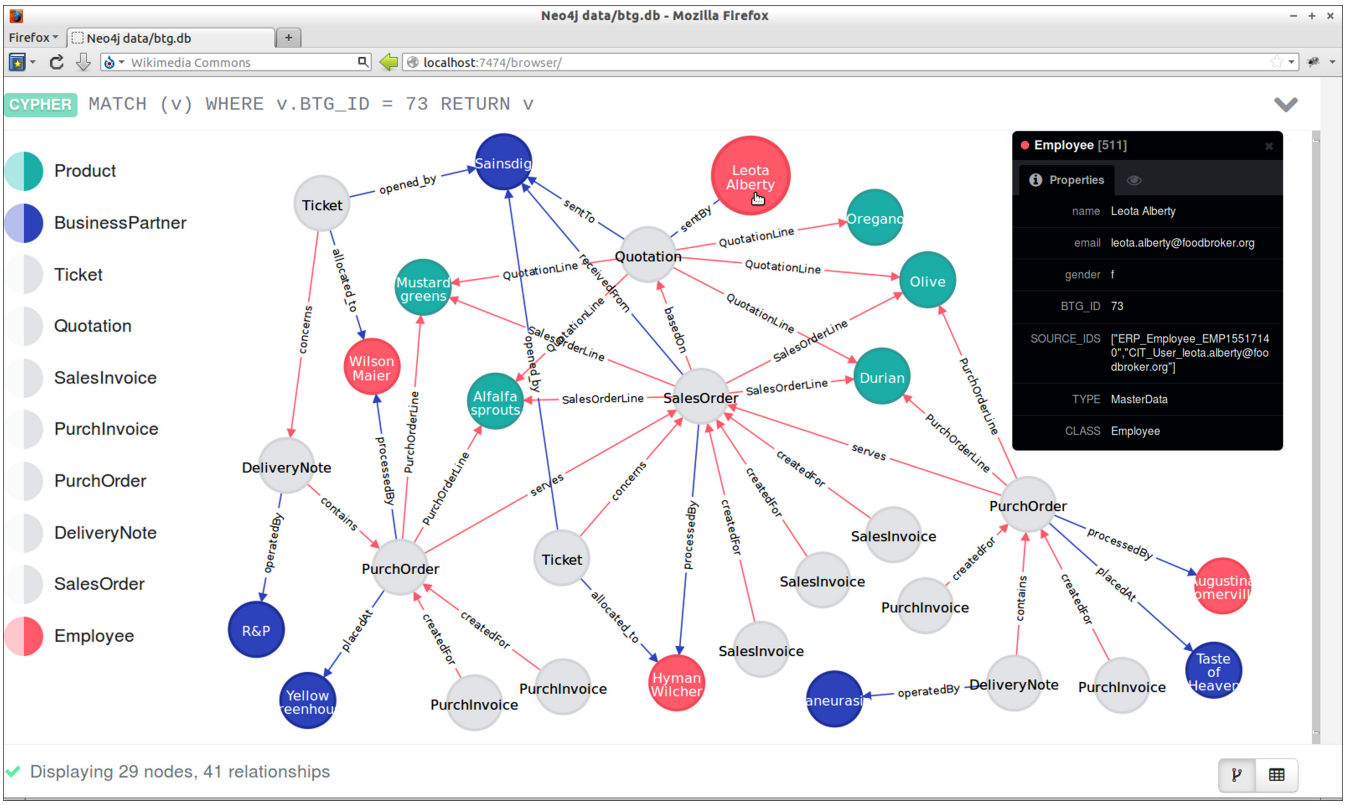

Figure 3: Sample Business Transaction Graph

The screenshot shows a single business transaction graph extracted from a FoodBroker dataset. Transactional nodes are colored in gray and labeled with their class name (metadata). Master data nodes are colored depending on their class and labeled with the value of their `name` property (instance data). Edges are labeled with corresponding relationship types (metadata). Causal connections are highlighted in red color. The dark rectangle shows properties of the node selected by the hand-shaped pointer (employee *Leota Alberty*). Recognizable by multiple values of `SOURCE_IDS`, the node holds data originating from two data sources.

5. REFERENCES

[1] Cypher query language. `http://docs.neo4j.org/chunked/2.0.1/cypher-query-lang.html`.

[2] Neo4j graph database. `http://www.neo4j.org`.

[3] S. Chaudhuri, U. Dayal, and V. Narasayya. An overview of business intelligence technology. *Communications of the ACM*, 54(8):88–98, 2011.

[4] C. Chen et al. Graph OLAP: Towards online analytical processing on graphs. In *Data Mining. ICDM'08. Eighth IEEE Int. Conf. on*, 2008.

[5] F. Holzschuher and R. Peinl. Performance of graph query languages. In *Proc. of the Joint EDBT/ICDT 2013 Workshops*, pages 195–204. ACM, 2013.

[6] A. Petermann, M. Junghanns, R. Müller, and E. Rahm. BIIIG : Enbabling Business Intelligence with Integrated Instance Graphs. In *Data Engineering Workshops (ICDEW), IEEE 30th Int. Conf. on*, 2014.

[7] A. Petermann, M. Junghanns, R. Müller, and E. Rahm. FoodBroker - Generating Synthetic Datasets for Graph-Based Business Analytics. In *Big Data Benchmarking (WBDB), 5th Workshop on*, 2014.

[8] M. A. Rodriguez and P. Neubauer. Constructions from dots and lines. *Bulletin of the Amer. Society for Inf. Sci. and Tec.*, 36(6), 2010.

[9] M. Rudolf et. al. The graph story of the SAP HANA database. In *BTW*, pages 403–420, 2013.

[10] H. Wang. Graph query and analytics with trinity. `http://www.cse.unsw.edu.au/~iwgdm/2013/Slides/Haixun.pdf`.

[11] M. Yin, B. Wu, and Z. Zeng. HMGraph OLAP: a novel framework for multi-dimensional heterogeneous network analysis. In *Proceedings of the 15th int. workshop on Data warehousing and OLAP*, 2012.

[12] P. Zhao et. al. Graph cube: on warehousing and olap multidimensional networks. In *SIGMOD Conf.*, 2011.