FoodBroker - Generating Synthetic Datasets for Graph-Based Business Analytics

André Petermann[1,2], Martin Junghanns[1], Robert Müller[2] and Erhard Rahm[1]

[1]University of Leipzig
{petermann,junghanns,rahm}@informatik.uni-leipzig.de
[2]Leipzig University of Applied Sciences
mueller@fbm.htwk-leipzig.de

Abstract. We present FoodBroker, a new data generator for benchmarking graph-based business intelligence systems and approaches. It covers two realistic business processes and their involved master and transactional data objects. The interactions are correlated in controlled ways to enable non-uniform distributions for data and relationships. For benchmarking data integration, the generated data is stored in two interrelated databases. The dataset can be arbitrarily scaled and allows comprehensive graph- and pattern-based analysis.

Keywords: synthetic data generation, benchmarks, graph data management, busines intelligence

1 Introduction

The operations of a company are reflected in its business processes and their different domain objects such as employees, products or purchase orders. Significant goals for business intelligence are to find correlations between such domain objects, to identify how domain objects are involved in business processes and to determine how this affects the success or failure of processes and the company. For example, consider trading processes where unfriendly sales people will have a negative influence while fast logistics companies can have a positive impact. Business information systems supporting the execution of business processes store those domain objects influencing the process outcome as master data. Furtheron, they record transactional data such as sales orders or invoices referencing master data during process execution.

While the data of business information systems is usually stored in relational databases, all process-related domain objects and their references can be abstracted as a graph, where objects are nodes and references are edges. So far, graph data models have been typically used to model natural graph scenarios such as social networks or knowledge models (e.g., ontologies). However, recent research projects aim at using graph models for data integration and analytics, for example in the enterprise [10][13] or health care [9] domains.

Evaluating or benchmarking such analytical graph applications requires datasets that reflect the nature of business information systems. Unfortunately, it is very

difficult to obtain real datasets from enterprises for evaluation purposes so that we see a need to generate appropriate datasets synthetically. Such datasets must meet specific features to be useful for graph-based analytics which are not sufficiently supported by established data generators for warehouse and graph benchmarks. In particular, the scenario covered by the dataset should represent a large number of similar use cases in practice and should allow the analysis of complex relationships and patterns to gain significant business insights. More specifically, we pose the following requirements.

1 - Heterogeneous domain objects Domain objects belong to different classes representing master or transactional data. Example master data classes are employee, customer or product and example transactional classes are sales order, rating or email.

2 - Heterogeneous relationships Domain objects may be related in different semantic ways represented by different relationship types. Relationships have to involve both master and transactional domain objects in any combination.

3 - Support of graph-based analysis The recorded data should allow a more comprehensive, graph-based business analysis than with traditional data warehouses. For enterprise data, it should thus represent different business processes with complex, correlated interactions between master data objects and transactional objects. It should thus be possible to identify and analyze different transactional patterns, e.g., the effect of how an employee interacted with specific customers.

4 - Multiple sources Companies and organizations typically use multiple information systems. To cover data integration tasks, a synthetic dataset needs to imitate multiple data sources.

5 - Scalability For benchmarking, the datasets need to be scalable up to realistic sizes with thousands of master data objects and millions of transactions.

To our knowledge, there is no data generator that provides all of the stated requirements yet. The contribution of this paper is FoodBroker, a data generator for related master and transactional data which are meaningful for a specific domain. FoodBroker is based on simulating processes and process-supporting information systems. The current version of FoodBroker provides an authentic data model representing two interrelated business information systems and generates records by simulating many business process executions (*cases*). FoodBroker considers the correlation between master data instances and their influence on the development and outcome of each case. The resulting data can be integrated using a graph model and used for evaluation and benchmarking of analytical graph systems. The source code of the FoodBroker data generator can be found on GitHub[1] under GPL v3.

The remaining paper is organized as follows. In section 2 we discuss related work about existing benchmark-related data generators. Then in section 3, we introduce our business process and its simulation. The current implementation is described in section 4. Finally, we end up with a summary and provide an outlook in section 5.

[1] `https://github.com/dbs-leipzig/foodbroker`

2 Related Work

Data generators for established OLAP benchmarks such as TPC-H [12] and APB-1 [8] do not fully meet the introduced requirements for graph-based analytics. Although they are scalable and offer heterogenous domain objects their relationship heterogenity is limited. Graph patterns of interest usually involve causal connections among transactional data as well as the involved master data [10]. However, APB-1 and all TPC benchmarks except TPC-DS are focussed on a single class of transactional data and TPC-DS by design provides no relationships in between the different transactional objects. The data generator of BigBench [5] extends the TPC-DS data model by related transactional data from web logs and reviews. However, those relationships are generated without considering the impact of master data instances. Further on, none of the benchmarks involves the problem of integrating multiple sources.

Due to the growing research interest in graph database systems, many benchmark studies have been published comparing those systems to relational database systems as well as among themselves. As there is currently no standard graph benchmark available, varying data generators have been developed, typically focused on a specific use case.

In [14], Vicknair et al. compare a relational and a graph database system. Their generated datasets contain artificial provenance information modeled as a directed acyclic graph and stored in a single data source. Nodes as well as edges are homogeneous and lack specific semantics: nodes carry random payload data, edges have no relationship type and no data attached to them. Graph sizes can vary depending on a user-defined number of nodes.

Holzschuher and Peinl also compare a relational and a graph database system focusing on the evaluation of graph query languages and SQL[7]. The generated datasets resemble the structure and content of online social networks in a property graph and are stored in a single data source. Nodes and edges can be of different types, nodes store realistic data based on dictionaries. The relationship information is taken from a real social network which makes it impossible to scale the graph size unrestricted.

Dominguez-Sal et al.[4] benchmark different GDBMS. They make use of the recursive matrix (R-MAT[3]) algorithm, which was designed to generate directed graphs that are equivalent to real networks in terms of specific graph invariants, like degree distribution and diameter. Using a single parameter, the graph size can be scaled exponentially. The algorithm is focused on generating a realistic structure but lacks the capability to add semantics to nodes or edges.

In [6], Gupta emphasizes that the characteristics of generating data for heterogenous graphs strongly differs from data generators of benchmarks for data warehouses and graphs which are specified by topology characteristics. He proposes a data generator for heterogenous multigraphs with typed nodes and labeled edges representing meaingful data from a drug discovery scenario. Although this approach even provides correlations between domain objects and graph structure, the resulting data does neither consider the characteristics of process-related data nor the scenario of multiple data sources.

Pham et al. propose S3G2, a framework for specifying the generation of graphs using a rule-based approach which leads to plausible structural correlations between graph structures and domain objects [11]. Their focus is the generation of social networks with real-world structural and semantic characteristics using dictionaries and manually defined correlation rules. To achieve scalability in terms of graph size and computing time, the framework is implemented using the MapReduce paradigm. A way to adopt the framework for business analytics could be the definition of domain specific correlation rules. Nevertheless, this would not lead to the generation of complex business processes where the correlation is not only depending on directly connected nodes but on transitive relationships between multiple domain objects.

A promising project, which adopts the S3G2 framework, is the Linked Data Benchmark Council (LDBC) [1][2], which aims at establishing standardized graph-oriented benchmarks and data generators for different application domains. Like us, the authors also highlight the importance of semantic correlations within the data to address business intelligence and graph analytics. Up until now, the project offers data generators for social network and semantic publishing use cases. FoodBroker describes a complementary, business-oriented use case where data objects are interrelated within diverse business process executions. The Foodbroker dataset will thus likely allow different graph-based analysis tasks for business intelligence than in the LDBC use cases. Still we expect the LDBC results as a valuable input for defining a benchmark based on Foodbroker datasets.

3 Simulation

The FoodBroker simulation reflects the core business of a fictive company trading food between producers (vendors) and retailers (customers). The company only offers a brokerage service and has no warehouse. Process-related data is recorded in an enterprise resource planning (ERP) system and a customer issue tracking (CIT) system. The simulation includes the customizable generation of master data as well as transactional data involved in the executions of two interrelated business processes for food brokerage and complaint handling. In the following, we describe the data model and the two simulated business processes. We also discuss how the generated data can be used for graph-based business analytics.

3.1 Data Model

The data model for the ERP and CIT systems is shown in figure 1. The ERP system stores master data objects of the classes `Employee`, `Product`, `Customer`, `Vendor` and `Logistics`. `Product`s are categozied into different product categories, such as *fruits*, *vegetables* and *nuts*. In the CIT system, the instances of master class `User` refer to employees in the ERP system, while master class `Client` refers to ERP class `Customer`.

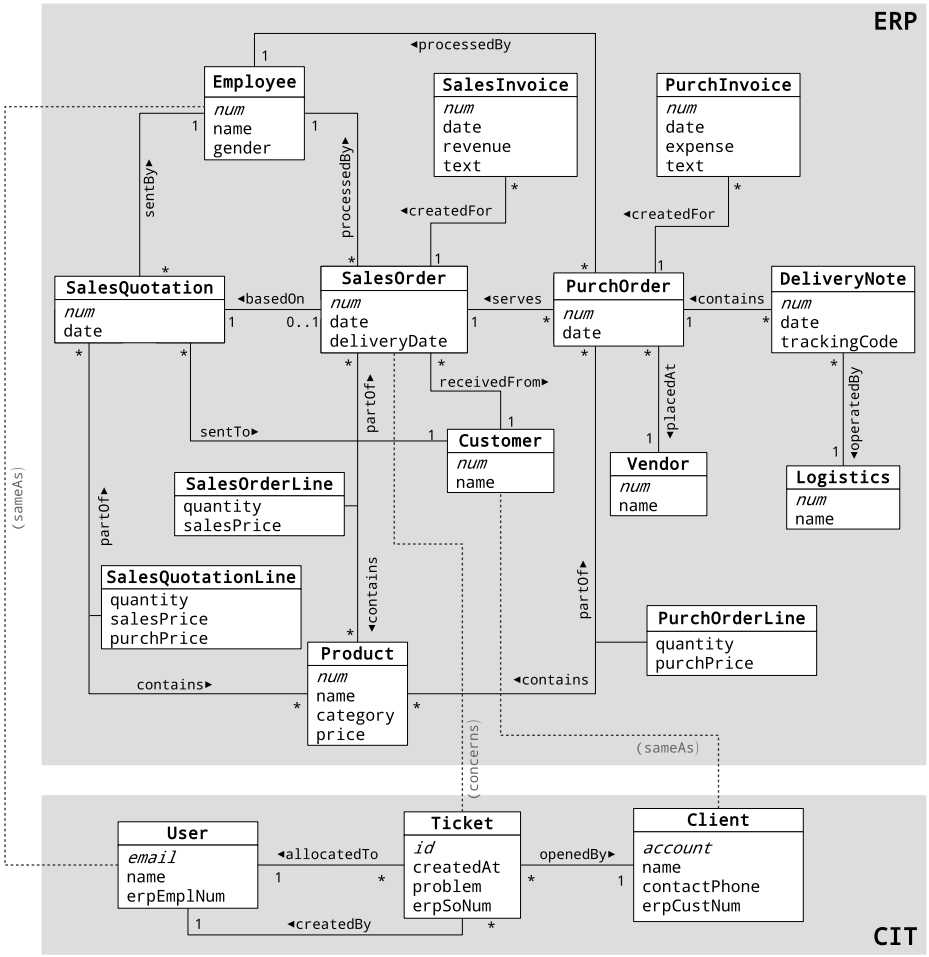

Fig. 1. FoodBroker Data Model : The outer rectangles show the boundaries of two systems `ERP` and `CIT`. Database tables correspond either to classes or n:m associations (`*Line`). Primary keys are highlighted by italic letters. Associations are shown as solid lines. Foreign keys are attached to associations. Implicit associations in between both databases are represented by dotted lines. For each implicit association, there is a corresponding column with prefix `erp`.

For each master data object we provide a quality criterion with one of the values *good*, *normal* or *bad*. We use these criteria in the generation of business processes and achieve thus a correlation between the different kinds of master objects and the structure and outcome of process executions.

The ERP system further records transactional data about trading and refunds, represented by the classes `SalesQuotation`, `SalesOrder`, `PurchOrder`, `DeliveryNote`, `SalesInvoice` and `PurchInvoice`. The line classes `SalesQuotationLine`, `SalesOrderLine` and `PurchOrderLine` represent n:m associations between the respective transactional classes and `Product`. The CIT system has only the transactional class `Ticket` representing customer complaints. All transactional classes have associations to other transactional and master data classes.

3.2 Food Brokerage

A food brokerage starts with a `SalesQuotation` sent by a random `Employee` to a random `Customer`. A quotation has `SalesQuotationLine`s referring to random products. Each `SalesQuotationLine` provides a `salesPrice` that is determined by adding a sales margin to the products `purchPrice`. A `SalesQuotation` can be either confirmed or rejected. To simulate the interaction of employee and customer, the probability of confirmation as well as the sales margin will be significantly higher if a *good* `Employee` and a *good* `Customer` are involved and correspondingly lower for *normal* or *bad* master data objects.

A confirmed quotation results in a `SalesOrder` and a set of `SalesOrderLine`s. The `SalesOrder` includes a reference to the underlying `SalesQuotation` as well as a `deliveryDate`. To reflect partial confirmations, there may be fewer `SalesOrderLine`s than `SalesQuotationLine`s. While the `Customer` is the same as the one of the `SalesQuotation`, a new `Employee` is processing the `SalesOrder`.

For each `SalesOrder`, one ore more `PurchOrder`s, each with one or more `PurchOrderLine`s, are placed at random `Vendor`s. A random `Employee` (purchaser) is associated per `PurchOrder`. Actual purchase prices are subject to variations. To simulate the interaction of purchaser and vendor, a *good* `Employee` and a *good* `Vendor` will lead to a lower `purchPrice` as compared to *normal* or *bad* ones. Furtheron, a *good* `Employee` will place `PurchOrder`s faster.

After a `PurchOrder` is processed by the `Vendor`, the company receives information about the `DeliveryNote`, in particular `date` of delivery, operating `Logistics` company and operator-specific `trackingCode`. The delivery time is influenced by the quality of both `Vendor` and `Logistics` company such that *good* business partners will lead to faster delivery than *bad* or *normal* ones.

Finally, one `SalesInvoice` per `SalesOrder` will be sent to the `Customer` and one `PurchInvoice` per `PurchOrder` received from the corresponding `Vendor`. All transactional data objects created within cases of food brokerage refer to their predecessor (except `SalesQuotation`) and provide a `date` property with a value greater or equal than the one of the predecessor.

3.3 Complaint Handling

For every customer complaint an instance of `Ticket` referring the corresponding `SalesOrder` is created. For the first complaint per `Customer`, additionally a `Client` instance is created. The employee handling the complaint is recorded in class `User`. There are two `problems` which may cause complaints: late delivery and bad product quality. Late delivery complaints occur if the `date` of a `DeliveryNote` is greater than the agreed `deliveryDate` of the `SalesOrder`, e.g., due to a *bad* `Employee` processing the `SalesOrder` or a `PurchaseOrder`, a *bad* `Vendor`, a *bad* `Logistics` company or combinations of those. Bad quality complaints may be caused by *bad* `Products` , a *bad* `Vendor`, a *bad* `Logistics` company or combinations of such.

A `Ticket` may lead to refunds which are recorded as `SalesInvoice` objects with negative `revenue`. While the constellation of a *good* `Employee` allocated to the `Ticket` and a *good* `Customer` may lead to low or no refund, *bad* ones lead to higher refund. If the problem was caused by the `Vendor`, there will be also a refund for the company in the form of a `PurchInvoice` with negative expenses. While the constellation of a *good* `Employee` and a *good* `Vendor` may lead to high refund, *bad* ones lead to lower or no refund.

3.4 Graph Analytics

The FoodBroker datasets provide complex correlations between master data instances and their involvement within the execution of the described business processes making it a good basis for graph-based business analytics. In particular it is possible to analyze the influence of the different master objects (employees, customers,vendors, etc.) on the financial outcome of business processes by case-wise aggregating all revenue- and expense-related properties, e.g. in `SalesInvoice` and `PurchInvoice` records of the same case, as well as determining the impact of refunds due to complaints.

Since all objects involved in the same case are interconnected in the generated dataset, it is possible to analyze the resulting graphs representing business process executions. Hence it is not only possible to aggregate profit-related measures but also to analyze the transactional correlation patterns underlying positive or negative cases to find frequent patterns in such business process executions. For example, it could be found that cases with outstandingly high profit frequently contain patterns such as
`Customer:ACME <-receivedFrom- SalesOrder -processedBy-> Employee:Alice`
saying that employee Alice was processing the sales order by customer ACME.

Finding such correlations is a non-trivial task due to many-to-many associations between cases and master data objects of the same class. For example, in a single case multiple `Employees` can be involved in different ways but also in the same way, e.g. multiple purchaser employees for different items.

Class	Configuration	Default Value
Master Data		
`Employee`	number of instances	$30 + 10 \times SF$
	proportions of good/normal/bad inst.	0.1/0.8/0.1
`Product`	number of instances	$1000 + 10 \times SF$
	proportions of good/normal/bad inst.	0.1/0.8/0.1
	list price range	0.5..8.5
`Customer`	number of instances	$50 + 20 \times SF$
	proportions of good/normal/bad inst.	0.1/0.8/0.1
`Vendor`	number of instances	$10 + 5 \times SF$
	proportions of good/normal/bad inst.	0.1/0.8/0.1
`Logistics`	number of instances	$10 + 0 \times SF$
	proportions of good/normal/bad inst.	0.1/0.8/0.1
Food Brokerage		
`Process`	number of cases	$10000 \times SF$
	date range	2014-01-01..2014-12-31
`SalesQuotation`	products per quotation (lines)	1..20
	quantity per product	1..100
	sales margin/*IM*	0.05/0.02
	confirmation probability/*IM*	0.6/0.2
	line confirmation probability	0.9
	confirmation delay/*IM*	0..20/5
`SalesOrder`	agreed delivery delay/*IM*	2..4/1
	purchase delay/*IM*	0..2/2
	invoice delay/*IM*	0..3/-2
`PurchOrder`	price variation/*IM*	0.01/-0.02
	delivery delay/*IM*	0..1/1
	invoice delay/*IM*	2..5/3
Complaint Handling		
`Ticket`	probability of bad quality/*IM*	0.05/-0.1
	sales refund/*IM*	0.1/-0.05
	purchase refund/*IM*	0.1/0.05

SF abbreviates **s**cale **f**actor
number of instance configurations are defined by linear functions $a + b \times SF$

IM abbreviates **i**mpact per **m**aster data instance
$IM > 0$ means good/bad master data instances increase/decrease value
$IM < 0$ means good/bad master data instances decrease/increase value

Table 1. FoodBroker Configuration Parameters

4 Implementation

Our current implementation stores the generated dataset in a MySQL database dump with separate databases for the `ERP` and `CIT` systems. In both databases every class has a dedicated table with the same name. 1:1 and 1:m associations correspond to foreign keys in class tables where the column names represent the relationship type. M:n associations are stored in separate tables.

The simulation is implemented as a Java console application. The database dump includes SQL statements for the creation of tables. All instances of a class are inserted into the tables corresponding to their class. Domain-specific string values such as employee or product names are provided by an embedded SQLite database.

Scalability in terms of different data sizes is controlled by a scale factor (SF). This makes it possible to create datasets with equal criteria regarding distributions, value ranges and probabilities but with a different number of instances. The number of master data instances and the number of simulated cases are defined by linear functions. The simulation function has a default slope of 10,000 per scale factor. We take into account that master data instances do not scale proportionally to cases. For example, there is only a limited amount of logistics companies. Thus, the functions of all master data classes beside a specific slope also have a y-intercept to specify a minimum number of instances.

Beside the scale factor and the resulting data growth, the generation of master data but also the process simulation is customizable by several configuration parameters. For example, one can set the confirmation probability of quotations or the influence of master data quality criteria on that probability. All configurations are set within a single file. An overview about the configuration parameters is provided in table 1. While using FoodBroker for benchmarks requires fixed configurations and variable scale, variable configurations at a fixed scale can be used to evaluate business intelligence applications at different scenarios. Table 2 shows dataset measures for different scale factors using the standard configuration. With respect to runtime, our current implementation shows a linear behavior for increasing scale factors. All datasets were generated on a workstation containing an Intel Xeon Quadcore, 8GB RAM and a standard HDD.

	Master Data		Transactional Data			
SF	Objects	Time	Objects	Relationships	Time	Dump Size
1	1.1K	4s	73K	380K	4s	42MB
10	1.7K	4s	725K	3.8M	25s	426MB
100	6.8K	4s	7.2M	38M	4min	4.1GB
1000	67K	8s	68M	360M	35min	39GB

Table 2. Measures of FoodBroker datasets for different scale factors (SF)

.

5 Summary

We presented the FoodBroker data generator for graph-based analytics based on simulated business process executions. FoodBroker creates domain-specific master and transactional data as well as meaningful relationships in between them. Quality criteria provided for all master data objects influence the execution of business processes and thus the creation of transactional data and process outcome. The complex correlations between master and transactional objects within graph-like business process representations make the Foodbroker datasets a good candidate for a variety of graph- and pattern-based analysis tasks.

Data generation is customizable and can be scaled to large sizes by a simple scale factor. In future work, we want to define a benchmarking workload for Foodbroker datasets and use it to evaluate different systems including current GDBMS and our own BIIIG[2] approach [10].

6 Acknowledgments

This work is partly funded within the EU program *Europa fördert Sachsen* of the European Social Fund.

References

1. R. Angles et. al. The Linked Data Benchmark Council: a Graph and RDF industry benchmarking effort. *ACM SIGMOD Record*, 43(1), 2014.
2. P. Boncz. LDBC: benchmarks for graph and RDF data management. In *Proc. of the 17th Int. Database Engineering & Applications Symposium*. ACM, 2013.
3. D. Chakrabarti, Y. Zhan, and C. Faloutsos. R-mat: A recursive model for graph mining. In *SDM*, volume 4, pages 442–446. SIAM, 2004.
4. D. Dominguez-Sal et. al. Survey of graph database performance on the hpc scalable graph analysis benchmark. In *Web-Age Information Management*. Springer, 2010.
5. A. Ghazal et. al. Bigbench: Towards an industry standard benchmark for big data analytics. In *Proc. of the 2013 int. conf. on Management of data*. ACM.
6. A. Gupta. Generating large-scale heterogeneous graphs for benchmarking. In *Specifying Big Data Benchmarks*, pages 113–128. Springer, 2014.
7. F. Holzschuher and R. Peinl. Performance of graph query languages: comparison of cypher, gremlin and native access in neo4j. In *Proceedings of the Joint EDBT/ICDT 2013 Workshops*. ACM, 2013.
8. OLAP Council. APB-1 OLAP Benchmark. `http://www.olapcouncil.org/research/bmarkly.htm`.
9. Y. Park et. al. Graph Databases for Large-Scale Healthcare Systems: A Framework for Efficient Data Management and Data Services. In *Data Engineering Workshops (ICDEW), IEEE 30th Int. Conf. on*, 2014.
10. 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.

[2] `http://www.biiig.org`

11. M.-D. Pham et. al. S3g2: A scalable structure-correlated social graph generator. In *Selected Topics in Performance Evaluation and Benchmarking*. Springer, 2013.
12. Transaction Processing Performance Council. TPC Benchmarks. `http://www.tpc.org/information/benchmarks.asp`.
13. E. Vasilyeva et. al. Leveraging flexible data management with graph databases. In *1st Int. Workshop on Graph Data Man. Experiences and Systems*. ACM, 2013.
14. C. Vicknair et al. A comparison of a graph database and a relational database: a data provenance perspective. In *Proc. of the 48th ann. Southeast reg. conf.* ACM, 2010.