Towards Realistic Sampling: Generating Dependencies in a Relational Database

Teodora Sandra Buda
Lero, Performance Engineering Laboratory
Computer Science and Informatics
University College Dublin
teodora.buda@ucdconnect.ie

John Murphy
Lero, Performance Engineering Laboratory
Computer Science and Informatics
University College Dublin
J.Murphy@ucd.ie

Morten Kristiansen
IBM Collaboration Solutions
IBM Software Group
Dublin, Ireland
Morten.Kristiansen@ie.ibm.com

ABSTRACT

Managing large amounts of information is one of the most expensive, time-consuming and non-trivial activities and it usually requires expert knowledge. In a wide range of application areas, such as data mining, histogram construction, approximate query evaluation, and software validation, handling exponentially growing databases has become a difficult challenge, and a subset of the data is generally preferred. As a solution to the current challenges in managing large amounts of data, database sampling from the operational data available has proved to be a powerful technique. However, none of the existing sampling approaches consider the dependencies between the data in a relational database. In this paper, we propose a novel approach towards constructing a realistic testing environment, by analyzing the distribution of data in the original database along these dependencies before sampling, so that the sample database is representative to the original database.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; H.2.4 [**System**]: Relational databases

General Terms

Algorithms, Measurement, Experimentation

Keywords

Database sampling, test, relational database

1. INTRODUCTION

Identifying critical system production level issues is one of the most expensive, time-consuming and non-trivial steps of the software development cycle. These critical issues represent a priority to be identified and must be solved and controlled with every new release of the product, so that the feasibility of the product is not altered. The core parts of the product can be tested using the operational data available (i.e. the data that are in the database at the time the application is to be tested). Since the production environment encompasses a variety of scenarios that the user created, it represents invaluable data for testing the most common and realistic scenarios, thus, the core functionality of the system from a user's perspective. However, generally these databases consist of large amount of data, which is computationally costly to analyze.

Database sampling is a potential solution to tackle this issue, and it is commonly used in wide range of application areas when using the entire database is not cost-effective or necessary (e.g. data mining, approximate query evaluation). Moreover, the testing environment should contain data that is likely to expose critical faults of the core functionality of the system, and by sampling data from the production environment this would be achieved. However, the existing tools do not consider the dependencies between the data in a relational database, but are limited to random sampling, while maintaining various constraints (e.g. referential integrity constraint, domain constraint)[5], and generally they are oriented towards a specific application area ([10], [7], [12], [8]). The objective of our research is a novel approach for database sampling, which would ensure the sample database respects the same relationships between data as the original database by verifying that both follow the same histograms for specific fields.

This work is based on a collaboration with IBM to determine an efficient and realistic approach to sample a database, with the objective of populating the testing environment in a rapid, automated way, and most important with realistic data.

This paper is organized as follows: the next section presents a short overview of the related work and the limitations of the previous approaches to the problem raised. Section 3 describes the proposed sampling system. In section 4, we present the results we achieved so far. Section 5 concludes the paper. Section 6 describes the future work of the proposed system, and how we can improve our approach.

2. RELATED WORK

Random sampling has been used for solving many database problems where using the entire dataset is computationally infeasible and where a balance between the accuracy of the

results and the computational cost of the analysis is preferred. Existing data population tools for the testing environment focus on populating the resulting database with synthetic data values or use some type of random distribution to select the data that must be included in the resulting database [5].

The closest study we have found associated to our work is the one presented in [4], which focuses on the advantages of constructing prototype databases populated with operational data, and proposes a sampling approach, which selects the data items so that the resulting sample database satisfies some predefined criteria, focusing on the advantage of using prototype databases populated with sampled operational data. Moreover, Olken's major contribution to random sampling from large databases proves sampling to be a powerful technique [11].

Numerous sampling approaches have been formulated in the data mining community, proving that sampling is a powerful technique for achieving a balance between the computational cost of performing data mining on a very large population and the accuracy of the results ([7], [9], [13], [14], [15], [10]).

In [9], the reader is presented with two sampling approaches with the objective of tackling exponentially growing data, while speeding up the data mining process and maintaining the quality of the mined information. The static sampling approach presented uses the distribution of the sample data as an evaluation criterion. This decides whether the sample reflects the large dataset independently of the following analysis to be performed on the sample. The sample is obtained by testing the hypotheses that each field from the sample comes from the same distribution as the original dataset. However, the static sampling approach has various shortcomings, such as its limitation to perform the sampling approach only on datasets containing a single table with various fields. Moreover, the analysis that evaluates the distribution of the sample in comparison to the original dataset is limited to univariate analysis.

A database sampling approach was proposed in [12], to reduce the computational cost of running clustering analysis on a very large population. Its density-biased property assures that even after the sampling has been performed, clustering analysis will find the same clusters as in the original population.

Furthermore, there are a few commercial applications that support sampling from databases. One of them is IBM *Optim*, which is developed to manage the data within many database instances, and has many components that allow the user to archive, compare, extract and manipulate the data by using various built-in functions. Its component, *Move*, can be used for sampling by using the option to select *Every* n^{th} row of each table from the original database. *Optim* ensures that the referential integrity is respected by the sample database ([1]).

As a recognized value of database sampling, Oracle DBMS include sampling mechanisms mainly for gathering statistical information about the data as an aid to improving performance, mainly related with query size estimation or intermediate query result sizes for query optimization algorithms [2]. Moreover, Oracle DBMS supports the possibility to query a sample of a given table instead of the whole table [3].

However, none of the above mentioned approaches consider evaluating the dependencies of the data in a relational database, nor study the distribution of data in the original database along these dependencies and sample accordingly, so that the sample database is representative to the original database.

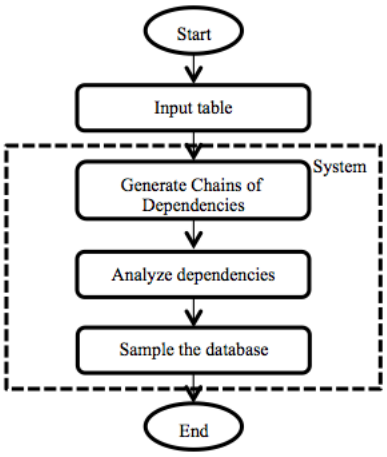

Figure 1: System process overview of the proposed sampling approach.

3. CODS SYSTEM

In this section, we describe the process of the proposed sampling system, CoDs *(Chains of Dependencies-based sampling)*, and how the dependencies between the data from the original database are considered.

3.1 System Process Overview

The proposed system consists of three main stages: generating the dependencies between data from the relational database, analyzing the data set and sampling the data set. Figure 1 shows the sample process diagram. The system is used in the following way:

- The user specifies which of the tables from the original database is the most relevant one for the sampling process (e.g. a large table, occupying critical amount of storage space would significantly impact the resulting sample database and could be a target for sampling).
- The system detects the relationship between the input table and the rest of the tables by following the foreign key (FK) constraints from the metadata. A FK constraint in a database is used to create and enforce a link between the data in two tables. When sampling, if such a constraint exists, data from the referenced table needs to be sampled in order to keep the referential integrity intact. Thus these FK constraints serve as inputs to our system to depict the relationships between data and produce a representative sample. Using the relationships detected, the system automatically generates the chains of dependencies, including in each chain referencing tables. Histograms are used to show the distribution of data along the chains discovered.
- The system analyzes the chains generated by the previous step. A key point is identifying data which has

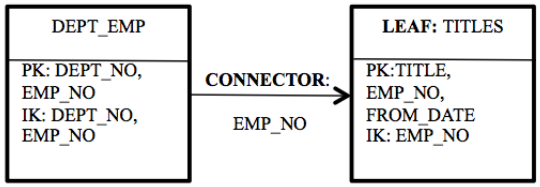

Figure 2: Representation of Chain: *DEPT_EMP, TITLES* for input table: *DEPARTMENTS*.

the same characteristics, as they represent the same scenario. This can be done by identifying groups with the same values in the previous generated histograms. If we focus on one histogram and sample from each group the same percentage of data, this would maintain that specific distribution. We consider all the chains of dependencies discovered, and their associated histograms. Thus, if the data maintains the same characteristics in all graphs (belong to the same group) that means that they represent the same scenario and only some of the records need to be sampled.

- Finally, the system proceeds to sampling according to the data gathered from the previous analysis, while maintaining the distribution of the data by sampling accordingly from each group identified.

This paper focuses on the detection of dependencies stage of the system, which needs to be automated and is vital for achieving a realistic sample from the original database.

3.2 Generating Chains of Dependencies in a Relational Database

Histograms are an extremely useful tool for analyzing the data, and can be viewed as an approximation of the underlying data distribution. The proposed system uses histograms to show the distribution of the data along the dependencies between the data in the original database. These histograms serve as an invaluable input for the sampling algorithm. Taking into considerations the distribution of the data for the sampling technique assures that the sample database will follow the same distribution, and the system will output a representative sample.

We define a chain of dependencies as a list of tables that either reference or are referenced by the original table sent as an input from the user (starting table). Furthermore, we define a connector as either a primary key (PK) or an imported key (IK) from a table, which represents a link with another table from the database. Finally, we also define a leaf as a table that either does not have a connector with any other table from the database or has only one connector with a table that has already been included in a chain of dependencies. See Figure 2 for better understanding.

The list of tables from the chain of dependencies represents the dependencies of the input table, and must be considered when sampling. Our algorithm (see *Simplified Algorithm*) analyzes the metadata of the database and searches for tables that import the PK of the input table. This analysis is iterative and continues with the detection of the tables that import the PK of the discovered dependent tables until it reaches a leaf table. An example of this type of chain is presented in Figure 2, where the table to be sampled is

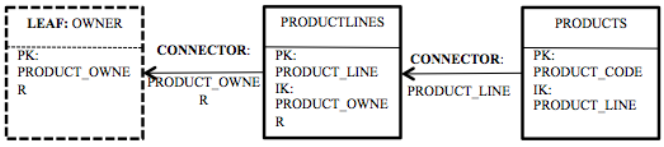

Figure 3: Representation of Chain: *OWNER, PRODUCTLINES, PRODUCTS* for input table: *PRODUCTS*.

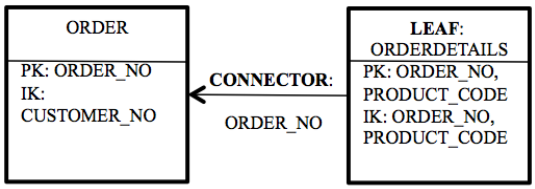

Figure 4: Representation of Chain: *ORDER,ORDERDETAILS* for input table: *PRODUCTS*.

the *DEPARTMENTS* table, and the relationship between titles of employees and their associated departments can be analyzed.

Moreover the algorithm searches whether the starting table imports any keys itself, and if it does, it will generate the chains of dependencies by searching for the tables that are referenced by the input table. This process is iterative as well, and it further analyzes which tables reference the discovered dependent tables, and will continue until a leaf table is reached. Figure 3 shows an example of this type of chain, where the input table to be sampled is *PRODUCTS*, with PK: *PRODUCT_CODE*. The system searches whether the input table imports any key, and it will discover the *PRODUCT_LINE* IK. Further it will search whether the table discovered, *PRODUCTLINES*, imports any other key and if it does, the system will add the discovered table to the chain of dependencies and studies its referenced tables. This process finishes when it reaches a leaf table. As we are analyzing the data dependencies through the FK relationships, if the PK of the leaf table is imported by the previous table in the chain, we can use the information stored in the previous table for analyzing its relationship with the starting table, eliminating the need to add the leaf table to the chain (see dashed line in Figure 3).

Finally, the algorithm searches for dependencies in the database by combining the two methods described above, by searching for tables that reference the input table's PKs and tables that are referenced by the discovered tables. Figure 4 shows an example of such a chain, where *PRODUCTS* is the input table to be sampled, *ORDERDETAILS* table imports *PRODUCTS*' PK, and *ORDER* table with PK *ORDER_NO* imports *CUSTOMER_NO*. Through this chain of dependencies, the relationship between *PRODUCTS* and *CUSTOMERS* can be analyzed.

Identifying the tables that reference the table to be sampled is a vital step towards achieving a realistic sample. If one of the variants is omitted, and we do not choose carefully the dependent tables, the sampling process is affected and will not produce a representative sample. The proposed system automatically detects these dependencies and analyzes

them in order to produce a representative sample.

Simplified Algorithm: Simplified Generating Chains of Dependencies

Result: Chains of dependencies for the input table
Initialization:
Starting parameters(ST): input table's PKs;
Related tables list: tables from the database that have a connector with the input table, or with the discovered related tables;

for *Table T1 in Related Tables list* **do**
  for *Table T2 in Related Tables list* **do**
    if *T1!=T2* **and** *ST.notIn(T2's IKs)* **then**
      Connector = retrieveConnector(T1,T2);
      if *Connector!=null* **and** *Connector!= ST* **then**
        if *Connector is one of T2's IKs* **then**
          ADD T2 to T1's list of Next Tables in Chain(NTC);
        else
          if *Connector is one of T2's PKs* **and** *(T2's IKs is not empty* **or** *T2's PKs.size()>1)* **then**
            ADD T1 to T2's list of NTC;

for *Table T in Related tables list* **do**
  if *T contains as IK or as PK one of the ST* **then**
    CREATE a new Table Chain(TC);
    ADD T in the new TC;
    ADD T's list of NTC in new TCs for each table;
    while *not reached a leaf table* **do**
      if *any of the tables have a list of NTC* **then**
        ADD the discovered tables in TC;
        ADD their list of NTC in new TCs;

4. RESULTS

We have applied the technique on a production environment available from our industrial partner IBM, however, for this paper, we consider an example database[1] from MySQL database management system (DBMS), with the EER diagram presented in Figure 5. The database is called *"Classic Models"* and it contains typical business data, such as customers, products, sale orders, and payments data. The tables mentioned in this section are: *CUSTOMERS* (customers' data), *PRODUCTS* (a list of scale model cars), *PRODUCTLINES* (a list of product line category), *ORDERS* (orders placed by customers), *ORDERDETAILS* (order line items in each order), *PAYMENTS* (payments made by customers based on their account), and *EMPLOYEES* (all employee information).

We consider the following scenario: The tester wants to sample the *PRODUCTS* table, as this will trigger sampling the data associated by the referential integrity with *PRODUCTLINES* and *ORDERDETAILS* tables, and following

[1]http://www.mysqltutorial.org/mysql-sample-database.aspx

Figure 5: The *"Classic Models"* EER Diagram.

the FK constraints from the schema, with the rest of the tables as well. Thus, we can easily deduct from this case study that the essential question to answer is which data should we sample from the *PRODUCTS* table so that the sample database will follow the same distribution as the original database for the relationship between the table to be sampled and the associated tables. For this scenario, the chains of dependencies are:

- Chain 1: *PRODUCTS*. Through this chain, we can analyze the relationship between the products and the product lines they belong to.
- Chain 2:*PAYMENTS,ORDERS,ORDERDETAILS*. Through this chain, we can analyze the relationship between the customers and the payments they made, between the orders and their associated customers, and between the orders and the products they contain. Thus, several linked dependencies can be analyzed through this chain, such as between customers and products, payments and products, and orders and products.
- Chain 3: *EMPLOYEES, CUSTOMERS, ORDERS, ORDERDETAILS*. Through this chain, we can analyze the relationship between the employees and the offices they work in, between the customers and their associated sales representatives, between the orders and their associated customers, and between the orders and the products they contain. Thus, several linked dependencies can be analyzed through this chain, such as between offices and products, customers and products, orders and customers, and orders and products.

Figure 6, 7, and 8 show the distribution of data along the second chain. The first graph shows the number of customers with certain number of payments, the second graph shows that a certain number of customers have made a certain number of orders, while the third graph shows the num-

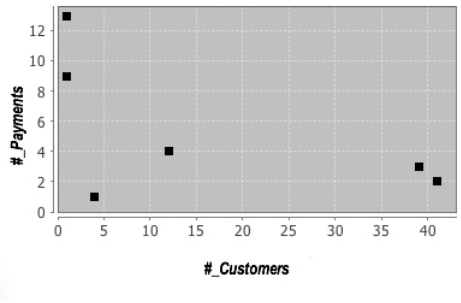

Figure 6: Relationship between *PAYMENTS* and *CUSTOMERS* in table: *PAYMENTS*.

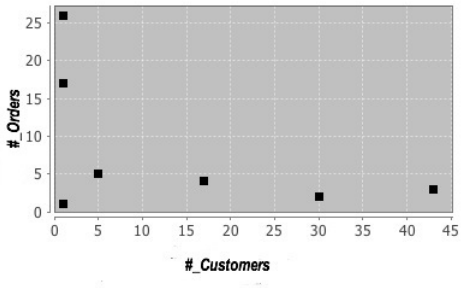

Figure 7: Relationship between *CUSTOMERS* and *ORDERS* in table: *ORDERS*.

ber of products that belong to a certain number of orders. By linking the information from these graphs, we can determine for example the number of products that a customer has requested through an order.

The other graphs reveal information regarding the product lines of a product, the sales representative for a customer, and the number of employees per office. Our objective is maintain the distribution of these relationships in the sample database (e.g. number of employees per office, which product is popularly bought) and ensure that the isolated cases will be sampled as well. Considering the scenario where a customer has made only two payments for two different orders each containing only one, but different product, when sampling, these two different products need to be selected. Another scenario to consider is if a order has only two products that can not be found in any other order belonging to the same product line containing only these two products. Not sampling these products might disturb the distribution between orders and products, or between product lines and products, and the uniqueness of data for a test case, thus an analysis of these relationships must be done. By sampling accordingly, the system assures that the sample will be representative to the original database.

Further, we will compare the presented approach with two existing sampling techniques that are closest to our work.

Bisbal's database prototyping approach is oriented towards relational databases and presented in detail in [6]. It performs the data selection by following a predefined set of criteria, generally the satisfaction of sets of integrity constraints(e.g. domain constraints, cardinality constraints, referential constraints), to evaluate the consistency of the resulting database. The algorithm randomly picks an entry

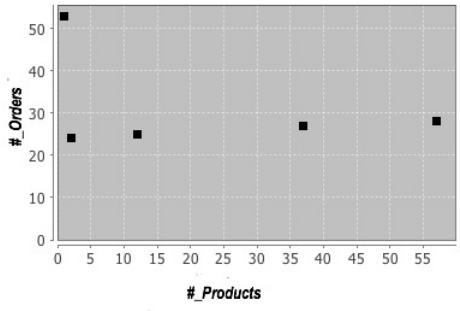

Figure 8: Relationship between *ORDERS* and *PRODUCTS* in table: *ORDERDETAILS*.

from the original database and verifies if it satisfies the predefined set of criteria to maintain both inter-sampler consistency and intra-sampler consistency. If the inserted tuples require the insertion of other referred tuples, the algorithm will insert the necessary remaining tuples. Entries are randomly picked until the minimum number of instances, given as an input to the algorithm, is reached for each entity. The algorithm represents one step further towards achieving a representative sample by following the previous mentioned constraints. However, the algorithm proposed involves random sampling, and does not produce a representative sample from a statistical point of view. For constructing a realistic sample, statistical information regarding the content of the original database should be maintained.

John's static sampling approach presented in [9] maintains the distribution of the original database in the sample. However, it is limited to univariate analysis, and the approach presented is focused on datasets containing a single table with various attributes. We extend this analysis through our approach to relational databases where various relationships between the data need to be maintained. Furthermore, they focus on obtaining high accuracy for the results of the data mining algorithm applied on the sample and they propose a dynamic sampling approach that offers better results, as it is specific to the data mining algorithm used, providing little detail on how the static representative sample is obtained.

5. CONCLUSION

This paper proposed a tool to generate a representative sample of a database by taking into consideration the dependencies between data in a relational database. In addition, this paper formulated the problem of finding the chains of dependencies of a table, in a database, which will be used as input to the sampling algorithm. The chains of dependencies generated by the system, suggest which other tables have to be considered when sampling. The sampling algorithm aims to significantly decrease the size of the original database and the computational cost of running the tests on the original database while maintaining the same output for the results of the tests, and keeping the data realistic.

6. FUTURE WORK

The presented sampling system described in the previous sections is partly implemented, and we are currently exploring how to use the chains of dependencies and histograms

generated by our system and described in section 3, in order to achieve the generation of a representative sample from the original database. We plan to validate the results by measuring how representative the sample database is to the original database, by verifying that both follow the same distributions for the discovered chains of dependencies. Moreover, another validation method we will use is by running the same tests from the original testing environment, on the sample database. We will measure the accuracy of the results by observing the queries' performance characteristics in the sample database. Since the sample database is expected to be representative to the original database and follows the same distributions, we are expecting the same results for running the tests on the sample database, however observing a significant decrease in the storage space needed for the testing environment.

Moreover, one improvement to the system would be to eliminate the need of specifying any parameters by the user, leading to reducing the uncertainty of our results. Through a static analysis of the database, the system can automatically detect the referenced tables that do not import information from any other table in the database. These tables can be used as starting tables for our sampling approach, since other tables reference their data, and the application can use these tables to generate the chains of dependencies that need to be analyzed, thus eliminating the needed input table for our system.

7. ACKNOWLEDGMENTS

This work was supported, in part, by Science Foundation Ireland grant 10/CE/I1855 to Lero - the Irish Software Engineering Research Centre (www.lero.ie).

8. REFERENCES

[1] IBM Optim Integrated Data Management. `http://www-01.ibm.com/software/data/data-management/optim-solutions/`.

[2] Oracle Database Performance Tuning Guide. `http://docs.oracle.com/cd/B19306_01/server.102/b14211/stats.htm#PFGRF003`.

[3] SELECT - Oracle Documentation. `http://docs.oracle.com/cd/B19306_01/server.102/b14200/statements_10002.htm`.

[4] J. Bisbal and J. Grimson. Consistent database sampling as a database prototyping approach. *Journal of Software Maintenance*, 14(6):447–459, Nov. 2002.

[5] J. Bisbal, J. Grimson, and D. Bell. A formal framework for database sampling. *Inf. Softw. Technol.*, 47(12):819–828, Sept. 2005.

[6] J. Bisbal, B. Wu, D. Lawless, and J. Grimson. Building consistent sample databases to support information system evolution and migration. In *Proceedings of the 9th International Conference on Database and Expert Systems Applications (DEXA'98), volume 1460 of Lecture Notes in Computer Science*, pages 196–205. Springer-Verlag, 1998.

[7] V. T. Chakaravarthy, V. Pandit, and Y. Sabharwal. Analysis of sampling techniques for association rule mining. In *Proceedings of the 12th International Conference on Database Theory*, ICDT '09, pages 276–283, New York, NY, USA, 2009. ACM.

[8] S. Chaudhuri, G. Das, and U. Srivastava. Effective use of block-level sampling in statistics estimation. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, SIGMOD '04, pages 287–298, New York, NY, USA, 2004. ACM.

[9] G. John and P. Langley. Static versus dynamic sampling for data mining. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 367–370. AAAI Press, 1996.

[10] H. Köhler, X. Zhou, S. Sadiq, Y. Shu, and K. Taylor. Sampling dirty data for matching attributes. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, SIGMOD '10, pages 63–74, New York, NY, USA, 2010. ACM.

[11] F. Olken. *Random Sampling from Databases*. PhD thesis, University of California at Berkeley, 1993.

[12] C. R. Palmer and C. Faloutsos. Density biased sampling: an improved method for data mining and clustering. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, SIGMOD '00, pages 82–92, New York, NY, USA, 2000. ACM.

[13] F. Provost, D. Jensen, and T. Oates. Efficient progressive sampling. In *Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining*, pages 23–32. ACM Press, 1999.

[14] H. Toivonen. Sampling large databases for association rules. In *Proceedings of the 22th International Conference on Very Large Data Bases*, VLDB '96, pages 134–145, San Francisco, CA, USA, 1996. Morgan Kaufmann Publishers Inc.

[15] M. J. Zaki, S. Parthasarathy, W. Li, and M. Ogihara. Evaluation of sampling for data mining of association rules. Technical report, Rochester, NY, USA, 1996.