ABSTRACT
Populating the testing environment with relevant data represents a great challenge in software validation, generally requiring expert knowledge about the system under development, as its data critically impacts the outcome of the tests designed to assess the system. Current practices of populating the testing environments generally focus on developing efficient algorithms for generating synthetic data or use the production environment for testing purposes. The latter is an invaluable strategy to provide real test cases in order to discover issues that critically impact the user of the system. However, the production environment generally consists of large amounts of data that are difficult to handle and analyze. Database sampling from the production environment is a potential solution to overcome these challenges.

In this research, we propose two database sampling methods, VFDS and CoDS, with the objective of populating the testing environment. The first method is a very fast random sampling approach, while the latter aims at preserving the distribution of data in order to produce a representative sample. In particular, we focus on the dependencies between the data from different tables and the method tries to preserve the distributions of these dependencies.

Categories and Subject Descriptors
H.2.4 [Systems]: Relational databases; D.2.5 [Testing and Debugging]: Testing tools

General Terms
Algorithms, Design, Experimentation

Keywords
Database sampling, relational database, testing.

1. INTRODUCTION

Testing environments represent a crucial part of the testing process, providing a variety of test cases for discovering anomalies of the system under development in order to successfully deliver the flawless system to its users. Thus, constructing realistic testing environments represents a difficult challenge in software validation, and it generally requires expert knowledge about the system under development.

Existing methods for populating the testing environment commonly generate synthetic data or use some type of random distribution to select the data that must be included in the resulting database [12, 13, 11]. Other generic tools generate synthetic data using the schema only [1, 4]. However, the synthetic data produced is dependent on the parameters given and may result in missing important test cases in the testing environment. In order to avoid this issue, some methods focus on privacy-preserving algorithms [14], facilitating the use of obfuscated operational data. By using the operational data available, the tester ensures that the testing environment contains relevant testing data from the users’ perspective, that serves as an invaluable input for testing the core functionality of the system. However, the production environment generally consists of large amount of data, which are computationally costly to analyze. As databases increase to the point where one terabyte is considered normal, new methods to manage this amount of data need to be devised. Moreover, applications that are generally used to process this data may fail to scale or become costly to use with such large amounts of data (e.g. applying data mining algorithms on a very large dataset).

Database sampling from the large dataset is a potential solution to overcome these challenges. Database sampling has a long history in computer science, starting with Olken’s random sampling approach proving its usefulness in numerous scenarios where using the entire database is infeasible because of the large amounts of data [10]. Existing sampling approaches are generally oriented towards the application area they are applied in (e.g. data mining [5], approximate query evaluation [3], histogram construction [7]), or the type of data to be sampled (e.g. single-table databases [8], graph databases [9], relational databases [6]). Unfortunately, existing sampling approaches that target relational databases are not efficient in time, and they do not aim at preserving the representativeness of the sample database. The representativeness of the sample database has been used as an evaluation criterion before. In [8] the reader is presented two sampling approaches with the objective of tackling large single-table databases, while speeding up the data mining process and maintaining the quality of the mined information. One of the sampling algorithms proposed is the Static Sampling approach which uses the distribution of the sample data as
an evaluation criterion to decide whether the sample reflects the large dataset. This static approach is independent of the following analysis to be performed on the sample. However, the static sampling approach is limited to single-table datasets and to univariate analysis. Other research, [9], proposes a representative sampling approach that aims to handle scalability issues of processing large graphs. The representativeness property of the subgraph of a manageable size is used as an evaluation criterion. However, the novel sampling algorithm is oriented towards graph-structured data.

Sampling from the production environment will determine the sample contain realistic test data, encompassing a variety of scenarios that the user created. Database sampling applied on the production environment aims to: (i) significantly decrease the storage space for the testing environment, (ii) decrease the administration overhead of managing datasets for the testing environment, (iii) decrease the computational cost of running the tests using a smaller database, while (iv) maintaining the accuracy of the results by using a realistic sample of the production environment. In this research, we focus on relational databases, and we aim to find a solution for all the above mentioned objectives. We propose VFDS (Very Fast Database Sampling) for solving the first three objectives. We propose CoDS (Chains of Dependencies-based Sampling), a representative sampling technique for solving all objectives mentioned above.

2. PROPOSED SOLUTIONS

Both approaches target relational databases, in the third normal form, and assume there are no cycles of dependencies between the tables. Both methods create the tables of the sample database identically with the ones from the original database. The sampling methods proposed maintain the data integrity of the database, and both consider a starting table for the sampling process. The starting table is either given by the user, or suggested by the method. In the first case, the user specifies which of the tables from the original database is the most relevant one for the sampling process (e.g. a table connected with most of the tables from the database would significantly impact the resulting sample database and could be a target for sampling). This requires subject matter expertise, and this specification would trigger the system to sample tuples from the rest of tables in the original database based on the already inserted tuples in the starting table from the sample database. If the user lacks such knowledge, the system suggests a starting table. This is a difficult challenge as the starting table critically impacts the sample database. We show in section 3.1 some preliminary results that show that the starting table should be a table that has the most connections with the rest of the tables in the database. This facilitates the control of the tuples sampled from referencing and referenced tables. We refer by directly connected tables to the tables that are referencing (i.e. tables that have a foreign key constraint pointing at the starting table) or referenced by the starting table (i.e. tables pointed by the foreign key constraints of the starting table).

2.1 VFDS

Existing sampling algorithms that propose to maintain the referential integrity of the relational sample database perform expensive computations for selecting the data to be inserted in the sample database. The main contribution of VFDS is the speed of the sampling method. The system produces the sample database in one single pass over the entire original database and does not require the processing of each tuple of the database.

In the first stage, the system randomly selects tuples from the starting table and inserts them in the corresponding table from the sample database. After this is completed, the system ensures the data integrity of the sample database by sampling the referencing and referenced tuples of the starting table. The insertion process is recursive and continues with sampling the referencing and referenced tuples of the previously sampled tuples. The insertion process ends when all of the tables of the sample database have been populated.

2.2 CoDS

The representativeness of the sample is a crucial property as it is feasible to expect that if the sample reflects the original database, applying the analysis on the representative sample will be less time-consuming, less computationally expensive, and most importantly will produce similar results with the ones produced by the original database. CoDS is a representative database sampling method. We define a representative sample database as a sample database where data selection is performed so that the sample database follows the same distribution for specific fields. In particular, the fields analyzed are the foreign key constraints, as they represent enforced links between the data in two tables. During sampling, if such a constraint exists, data from the referenced table needs to be sampled as well in order to keep the referential integrity intact. Thus, the system considers the foreign key constraints invaluable inputs for our system to depict the relationships between data and produce a representative sample. A representative sample of a database should maintain the distribution between all the tables that are connected through foreign keys. For instance, for a database storing banking information, if table Account has a foreign key pointing to table District, the distribution of accounts over districts should be preserved.

The key idea of the proposed solution is to identify the tuples that need to be selected from the starting table. After these tuples have been inserted in the sample database, the method proceeds in inserting the referencing and referenced tuples from the rest of the tables of the original database in the sample database. Similarly to VFDS, the process is recursive and finishes when all of the connected tables have been populated. The challenge stands in identifying which tuples to sample from the starting table in order to produce a representative sample. With this purpose, the method studies the relationships between the starting table and the rest of the tables of the database through various joins when needed. The approach will generate a set of chains of dependencies of the starting table that need to be analyzed before sampling. The chains represent sequence of directly connected tables, through which the relationship between the starting table and the rest of the tables of the chain can be analyzed. Each chain is associated with the distributions of the relationship between the starting table and the rest of the tables of the chain and the goal of CoDS is to maintain these distributions when sampling. The reason for this strategy is the assumption that by preserving the distribution between the starting table and the rest of the tables of the database, the distribution between the tables directly connected would be preserved as well. Furthermore, a key
3. EVALUATION

We consider for evaluating the VFDS sampling approach the following factors: (i) execution time and (ii) sample size. The most important metric considered to evaluate VFDS is the execution time, as the objective of this method is to produce at high speed a sample of desired size of the original database. The second objective of VFDS is to produce a sample of a desired size, thus we also evaluate the approach based on the sample size. As CoDS system is at an initial stage of implementation, we only show preliminary results of the VFDS sampling method in the following subsection.

3.1 PRELIMINARY RESULTS

VFDS represents a random sampling approach. Thus, we performed the experiments five times, and present the sample size error as the average of the absolute value for the sample size error of each run. Each experiment was run with 12GB maximum size of the memory allocation pool on a machine with quad-core 2.5GHz processor, 16GB RAM, and 750GB Serial ATA Drive with 7200 rpm. In each experiment, we applied VFDS on the original database using each table of the database in turn as the starting table.

3.1.1 Metrics

We expect the total number of tuples of the original database, \( O \), to be reduce by \( \alpha \). We define the sample size error as:

\[
sample\_size\_error(S) = \left| \frac{|S| - \alpha \cdot |O|}{\alpha \cdot |O|} \right|
\]

where \( |S| \), and \( |O| \), represents the number of tuples in the sample database, and in the original database.

We measure the execution time of the sampling algorithm in seconds.

3.1.2 Database

We have applied VFDS on the Financial\(^1\) database from PKDD’99 Challenge Discovery. The database consists of 8 tables: Account, Card, Client, Disposition, District, Loan, Orders, and Trans. The total size of the database is 1,079,680 tuples. The sizes of the tables range from 77 (table District) to 1,056,320 tuples (table Trans).

3.1.3 Observations

Fig. 1 and Fig. 2 show the sample size error of the sample database and the execution time of the VFDS sampling algorithm respectively.

\(^1\)http://lisp.vse.cz/pkdd99/Challenge/berka.htm

We observe in Fig. 1 that the best results are produced for the starting table Account, with the sample size error ranging between 7.43 and 0.25, for \( \alpha \) between 0.1 and 0.9. The worst results are produced using table Client as the starting table, with the sample size error ranging between 890.3 and 11. We observe a tendency of the sample size error to decrease as the sampling rate, \( \alpha \), increases, independently of the starting table. The reason for this is that as the desired sample size increases, there is less data to be included in the sample database that would influence the sample size error.

We notice in Fig. 2 that the best execution time was achieved using Loan as the starting table, ranging between 4 and 6.6 seconds. The worst execution time for VFDS was achieved using table Trans as the starting table, ranging from 20 to 74.6 seconds. The execution time of VFDS using Account as the starting table ranges from 5.6 to 17.6 seconds. In order to achieve a balance between the execution time and the sample size error, we observe that the best results for the Financial database are produced using Account as the starting table.

3.2 EVALUATION PLAN

We plan to evaluate CoDS according to the previous defined factors together with an additional one: the representativeness of the sample. We consider the representativeness measure the most important factor for the CoDS sampling approach as a realistic sample of the original dataset is expected to be significantly more useful for the following testing than a random one. The representativeness of the sample is computed by comparing the distributions between all the tables that are directly connected through foreign keys.
from the original database with the ones from the sample database.

We plan to evaluate the sampling approaches by applying them on a production environment of our industrial partner and use the sample database as the testing environment of the system under test. The objective is to significantly decrease the time it takes to construct the testing environment, while preserving the accuracy of the results of the tests. More specifically, we plan to measure how many anomalies of the system are discovered using the original dataset in comparison with using the sample database. Moreover, we plan to study how the representativeness of the sample impacts the number of anomalies discovered. In addition, we plan to evaluate how the starting table impacts the number of anomalies discovered of the system under test.

We plan to study how the starting table impacts the representativeness of the sample database, while trying to minimize the sample size error. Furthermore, we plan to apply VFDS and CoDS on different databases and study the impact of the starting table selection on both representativeness and sample size error.

Finally, we plan to compare our approaches with existing sampling methods. In particular, we plan to compare them with Join Synopses [2] (JS), and Linked Bernoulli Synopses [6] (LBS), as they both target relational databases. The objective of JS and LBS is to provide fast approximate query results for join queries. Both approaches maintain the foreign key integrity of the sample database.

4. CONCLUSION AND FUTURE WORK

In this paper we introduced two database sampling approaches targeting relational databases. VFDS aims to significantly decrease the time necessary to produce a sample database while preserving the data integrity of the sample database. CoDS aims to produce a representative sample of the original dataset, while preserving the data integrity of the sample database. The main contribution of VFDS is the speed of the sampling mechanism, while for CoDS the representativeness of the sample. The key idea of both approaches is to choose a starting table, and evaluate the dependencies of the starting table with the rest of the tables of the database and sample accordingly. VFDS randomly samples the data for the resulting database, while CoDS studies the previous mentioned dependencies and produces a representative sample of the original database.

As future work, we plan to extend both VFDS and CoDS to deal with cyclic dependencies of the foreign key constraints in the database. We plan to extend CoDS’ analysis to other fields besides the foreign key constraints, and evaluate the impact of the resulting dataset on the testing environment. Specifically, we plan to use the sample database for functional testing, where a small but representative sample should suffice to discover anomalies of the system, if the sample encapsulates the same test cases as the original dataset. Finally, we plan to run additional experiments both with VFDS and CoDS in other application areas, such as data mining, and approximate query evaluation.

5. ACKNOWLEDGMENTS

This work was supported, in part, by Science Foundation Ireland grant 10/CE/11855 to Lero - the Irish Software Engineering Research Centre (www.lero.ie).

6. REFERENCES