Fuzzy SQL

Recently, aggregate queries are used as workloads to evaluate the utility of synthetic data1. The generated query is run on both real and synthetic datasets. The query results from both datasets are compared and the relative error is calculated to serve as a utility measure. However, the current studies assume pre-determined query formulation such as that proposed by Li et al where specific categorical variables are chosen for aggregation2. This deterministic approach leads to biased results since a synthesizer may learn the underlying distribution of some variables better than the others. To tackle this problem, we are inspired by the randomness introduced by SQL Fuzzing techniques also known as Fuzzers.

SQL Fuzzers are mainly used to test database management systems (DBMSs) for any bugs or vulnerabilities3,4. Before executing an SQL query, a DBMS performs two levels of checks. First the SQL statement is checked for any syntactic error such as grammatical errors. Secondly, the query is checked semantically e.g. a call is made to a non-existent table. Once the DBMS performs the necessary checks, the SQL statement is executed according to the best execution plan5. Fuzzers usually generate large amount of queries that do not pass the aforementioned checks. . For instance, American Fuzzy Lop (AFL)6 , a widely used fuzzer, has only 30% out of it generated queries passing the syntax check while only 4% can pass the semantic check5. While research attempts are made to focus on finding DBMS logic errors rather than semantic and syntactic errors such techniques generate both queries and the test datasets7. Other researchers8 propose an approach to generate queries that ensure fetching a randomly selected row and hence avoid syntactic and semantic error injection.

To ensure unbiased representation of synthetic data utility, we propose a Fuzzy SQL technique that have the following features:

* Aggregate queries shall be randomly generated, i.e. grouping may be executed using any combination of the available categorical data.
* Aggregation may take place across any of the available continuous variables.
* A random condition may be imposed on the aggregate queries. In such case, the values used in the WHERE clause shall be randomly sampled from the real data and equally executed on both the real and synthetic data.
* Datasets are available in tabular formal and may include categorical, continuous and date variables.
* A proper metric shall be established to compare the results from the real and synthetic data.

It is important to pay a special attention to the metric to be used. For instance, Fan et al proposes1 to arrange the data with one of the categorical variables as a dependent variable. Then two classifiers are trained using training examples from the real and synthetic data respectively. Finally the testing examples from the real data are used to test both models using traditional metric such as F1. The F1 scores for both models are compared. Clearly, this approach will highly depend on the selection of the dependent variable especially if the dataset mostly incudes categorical variables.

1. Fan J, Liu T, Li G, Chen J, Shen Y, Du X. Relational Data Synthesis using Generative Adversarial Networks: A Design Space Exploration. Published online August 28, 2020. Accessed August 30, 2022. http://arxiv.org/abs/2008.12763

2. Li K, Zhang Y, Li G, Tao W, Yan Y. Bounded Approximate Query Processing. *IEEE Trans Knowl Data Eng*. 2019;31(12):2262-2276. doi:10.1109/TKDE.2018.2877362

3. SQLsmith: Randomized SQL Testing in CockroachDB. Cockroach Labs. Published 00:00 + UTC . Accessed August 31, 2022. https://www.cockroachlabs.com/blog/sqlsmith-randomized-sql-testing/

4. Seltenreich A. SQLsmith. Published online August 29, 2022. Accessed August 30, 2022. https://github.com/anse1/sqlsmith

5. Zhong R, Chen Y, Hu H, Zhang H, Lee W, Wu D. SQUIRREL: Testing Database Management Systems with Language Validity and Coverage Feedback. Published online June 3, 2020. doi:10.48550/arXiv.2006.02398

6. american fuzzy lop. Accessed August 30, 2022. https://lcamtuf.coredump.cx/afl/

7. Ghit B, Poggi N, Rosen J, Xin R, Boncz P. SparkFuzz: searching correctness regressions in modern query engines. In: ; 2020:1-6. doi:10.1145/3395032.3395327

8. Rigger M, Su Z. Testing Database Engines via Pivoted Query Synthesis. Published online January 13, 2020. Accessed August 31, 2022. http://arxiv.org/abs/2001.04174