Paper Review:

TritonSort: A Balanced and Energy-Efficient Large-Scale Sorting System

Braden D. Licastro
Department of Computer Science
Allegheny College
licastb@allegheny.edu
http://www.fullforceapps.com/

April 2, 2013

1 Summary

This paper describes an effective and efficient method of testing relational database schema for validity. sorting algorithm coined Triton Sort. The authors created a system, coined SchemaAnalyst, which operates at competitive or better generation times which outperform a commonly used, free, and open source system called DBMonster in terms of constraint coverage and mutation score. [1] Often times when creating a relational schema mistakes can go unnoticed for varying lengths of time that can lead to data integrity problems and corruption down the line. The program and methods proposed combat this problem my verifying and testing the given schema constraints.

SchemaAnalyst was designed to create randomized data to be inserted into the database that would both violate the schema and also satisfy it. To successfully test this, data would be generated that should satisfy the constraints and be inserted into the empty database. If one of the correct inserts would not be inserted there is more than likely a constraint error in the database. The next pass discussed breaks the constraints down piece by piece. Through each iteration of insert statement the data generated will satisfy all schema requirements but one constraint. This could be a unique, not null, primary key, and so on. By violating one constraint at a time each constraint in the schema may be validated.

Lastly, a set of mutation operators was used to test the integrity constraints of the schema. [1] This segment of the research mutated the constraints one part at a time. Examples could include removing a NOT NULL constraint or adding one where there previously wasn't. This will test the schema in a way that would verify that only correct data is allowable therefore verifying the schema. Another method the authors implemented was the removal, addition, or replacement of a primary or foreign key constraint.

2 Critique

Throughout the paper, the authors restricted the validations to the universal data types, namely *Boolean, DateTime, Date, Numeric, String, Time and* TimeStamp [1] due to the variance of allowable data types in various DBMS implementations. In the event where a data type is used that cannot be mapped to one of these universal types, one could assume validation would not be possible. I believe that by focusing on one DBMS which both SchemaAnalyst and DBMonster are compatible with, a more in-depth validation program would be possible. In addition, the generation of data is done based on the provided schema and is therefore tested based on the schema provided. Could it be possible that a specific type or severity would alter the generated data so severely that the expected data appears nothing like the generated data? In this case it could be theoretically possible to find minor constraint errors but test using invalid data.

3 Synthesis

After reading the research paper I concluded that additional research could be performed in several ways, primarily creating a testing environment that would be fully automated and determine why a certain mutant wasn't fixed. Currently human interaction is necessary for processing the results. By locating the highest mutation scores returned and analyzing the SQL relating to that score the bug can be pinpointed. If this process could be reliably replicated through an extension of the program it could be made significantly more effective and reduce the time spent pinpointing a specific problem.

References

[1] Gregory M. Kapfhammer, Phil McMinn, and Chris J. Wright. Search-based testing of relational schema integrity constraints across multiple database management systems. In *International Conference on Software Testing, Verification and Validation (ICST 2013)*. IEEE, March 2013.