Testing Concurrency in Databases still Matters

by

Dimitrios Liarokapis, Univ. of Ioannina, Greece

Elizabeth O’Neil, UMASS Boston, USA

Patrick O’Neil, UMASS Boston, USA

Proceedings of the 2020 IEEE International Conference on Information Technologies (InfoTech-2020) 17-18 September 2020, St. St. Constantine and Elena, Bulgaria

Note

This is a short and abstract presentation focusing on the motivation of the paper and communicating the findings at a higher level (avoiding technical details)

Motivation

- We wanted to reinvigorate research we did in the past about testing the implementation of Isolation Levels in Database Management Systems.
- Some developments in the industry that could rekindle interest.
  - Database products are adopting multiversion concurrency control (similar to snapshot isolation)
  - A proposal for providing a serializable snapshot isolation level has started being adopted in the industry (Berkley DB, PostgreSQL, MySQL)

Original Work

- Implemented HISTEX (HISTory Exerciser)
  - More details in another paper in this conference
- HISTEX was used systematically in testing database systems utilizing single version concurrency control relying on locking (IBM DB2, Informix)
- Used also for ad hoc examination of DB systems utilizing multiversion concurrency control such as Snapshot Isolation (Oracle)

Current Approach

- We upgraded HISTEX to run with current versions of database systems we examined in the past (IBM DB2, Informix, Oracle)
- We rerun a test suite that was implemented to test products relying on single version concurrency control based on locking
- We inspected and analyzed the outcomes.

Summary of Findings

- We noticed differences in one of the two products we tested.
- The differences are related mainly to the introduction of multiversion concurrency control.
- We could classify the differences in three categories (fictitious issues, improved precision, potential errors)

Fictitious Issues

- There were differences in the output that could appear as errors but were actually caused by the introduction of mulitversion concurrency control and the tool’s behavior not reporting “Not Found” data.
- This can be addressed by extending the tool to treat these cases with improved reporting.

Improved Precision

- There were cases where the product improved its precision compared to the original versions we had tested in the past.
- There used to be cases where a concurrent operation would block but it was not necessary for a given combination of isolation levels.
- In the recent test this blocking does not occur indicating improved precision.

Potential Errors

- We discovered a case where a database product would behave differently depending on the pattern of the access code.
  - When accessing the database by utilizing cursors and relying on dynamic sql (sql code that is not precompiled) there would be unessary blocking of read operations.
  - The blocking would not occur with explicit sql statements or by relying on cursors with static SQL

Importance of Findings

- The potential error is something that could be important to applications relying on the database product.
- If there is no resolution or workarounds, applications could suffer from reduced performance due to the unnecessary blocking that could occur in case the application relies on dynamic SQL.

Thank you!

dili@uoi.gr