Static analysis of SQL queries

1 Description

Dependency graphs (also called Precedence graphs) are a standard technique to implement concurrency control mechanism for RDBMS such as PostgreSQL, MySQL, ... and also to implement static analysis to ensure the correctness of concurrent executions of transactions [BG16, BBE19].

Last year a group of our students developed a tool that computes the dependency graph of a given set of SQL queries, and that allows its users to interact with the graph via a nice GUI.

The tool tough lacks important functionalities, such as

(a) parsing real world MySQL or PostgresSQL queries;

(b) computing dependency graphs according to different definitions (compare Definition 9 in [BG16] and pag. 299 in [BBE19]);

(c) finding critical cycles in the graphs.

In practice the last functionality is the most important one, yet the most difficult to implement.

If you are interested in the semantics of SQL and in software verification you can decide to implement any of the above functionalities, thereby extending the existing tool.

2 Mile-stones

- The first mile-stone is of course to download the existing code and get acquainted with.
- If you decide to work on either functionality (a) or (b) the second mile-stone is to understand the necessary formalism.
- The subsequent mile-stones depend on the particular project that you pick, and have to be discussed.

References

[BBE19] Sidi Mohamed Beillahi, Ahmed Bouajjani, and Constantin Enea. Checking robustness against snapshot isolation. In Isil Dillig and Serdar Tasiran, editors, *Computer Aided Verification - 31st International Conference, CAV 2019, New York City, NY, USA, July 15-18, 2019, Proceedings, Part II*, volume 11562 of *Lecture Notes in Computer Science*, pages 286–304. Springer, 2019.

[BG16] Giovanni Bernardi and Alexey Gotsman. Robustness against consistency models with atomic visibility. In Josée Desharnais and Radha Jagadeesan, editors, *27th International Conference on Concurrency Theory, CONCUR 2016, August 23-26, 2016, Québec City, Canada*, volume 59 of *LIPIcs*, pages 7:1–7:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.