Closest Work Comparison Report

Members: Daniel Lindsey, Chase Keller, Long Nguyen

Submission Date: December 7th, 2025

Related Work: Speeding up SQL subqueries via decoupling of non-correlated predicate

By: Dmitrii Radivonchik, Yakob Kuzin, Aanton Chizhov, Dmitriy Scheka, Mikail Firsov, Kirill Smirnov, and George Chernishev

Summary:

This paper discusses a new idea for correlated subquery optimization, albeit only on a subset of correlated subqueries. For subqueries which have a WHERE clause containing both a correlation and a non-correlation, it is then possible to isolate and rewrite the non-correlated section and thus reduce the overall number of evaluations per execution. These cases exist when the WHERE clause features operators such as AND as well as OR.

These queries are rewritten so that the non-correlated part is only evaluated a single time instead of having to be rerun for each row. This is achieved by pulling the non-correlated part out of the WHERE clause and having it be executed first to act as a filter. In the case of the OR clause when the filter evaluates as true the non-correlated section no longer needs to be run, greatly speeding up execution times. In the AND case the non-correlated section is simply executed once, and the correlated section executes repeatedly as normal.

This methodology was tested in both PosDB and PostgreSQL with optimal results showing up to a 5 times reduction in query run-time.

Justification and Description:

This work relates to our project as they are both based around subquery optimization. Our project looked at both correlated and non-correlated subqueries and the differences in query execution times. This paper looked specifically at improving the execution times of correlated queries by rewriting the query to decrease how much of the query needed to be re-executed for each row in the dataset. Both projects evaluated query execution time and provided graphical evidence depicting how modified (decorrelated/unnested or decoupled) subqueries can be more efficient than their base permutation.

The primary difference between the projects is that the related work accomplished much more than what we had sought out to do. Not only did they clearly reduce the execution time of coupled correlated subqueries, but they also did so without being limited by PostgreSQL's optimizer. The other main difference is that they kept their project scope small (focusing only on WHERE clauses with coupling) rather than trying to address all subqueries in general which is what we attempted.