CS349: Project Submission Stage 1

XData Extensions

Satyankar Chandra (22B0967)
Ayan Tanwar (22B0931)
Gourish Garg (22B0915)
Harsh Kumar (22B0973)

1 Project Overview

Aim: Extend XData to support in-browse postgresql execution using pglite.dev. Make sure everything runs from local resources so it can be run for exams without external internet. This allows each user to run in their own database without accessing any central database. The next step is to extend XData system to run the queries in browser rather than on a backend database.

2 Project Proposal

In our initial project proposal, we aimed to extend the XData system and implement the following features:

- Query Profiling and Runtime Analysis
- Evaluation Plans and Query Optimization
- Client Side Autograding
- Support for Concurrent Queries
- Adding Constraints to Queries

Our team has been working on all of these features in parallel. We have made significant progress on the first two features, and we are currently working on the third and fourth features. The fifth feature is still in the planning stage.

3 Current Progress

Below is the current status of our project for each of the features:

3.1 Query Profiling and Runtime Analysis

TODO Ayan

3.2 Evaluation Plans and Query Optimization

PostgreSQL has various options to evaluate a query. For example, for a search query (typically a `SELECT` query), it can perform a `Seq Scan`, `Index Scan`, `Bitmap Index Scan` or `TID Scan`. Similarly there are multiple possibilities for `JOIN`, `SORT` and `GROUP BY` operations. The query planner chooses the best option based on the statistics of the tables involved in the query.

Even though PostgreSQL displays the evaluation plan for a query by using the `EXPLAIN ANALYZE` command, it does not provide the option to choose a specific evaluation plan. We have implemented a feature that allows the user to choose the evaluation plan for a query. The user can select the evaluation plan for each operation in the query and compare the Planning and Execution time for different evaluation plans.

This feature is implemented by setting the `enable_*` flags in PostgreSQL. For example, to enable `Seq Scan`, we set the `enable_seqscan` flag to `on` and all other flags to `off`. This allows us to choose a specific evaluation plan for a query.

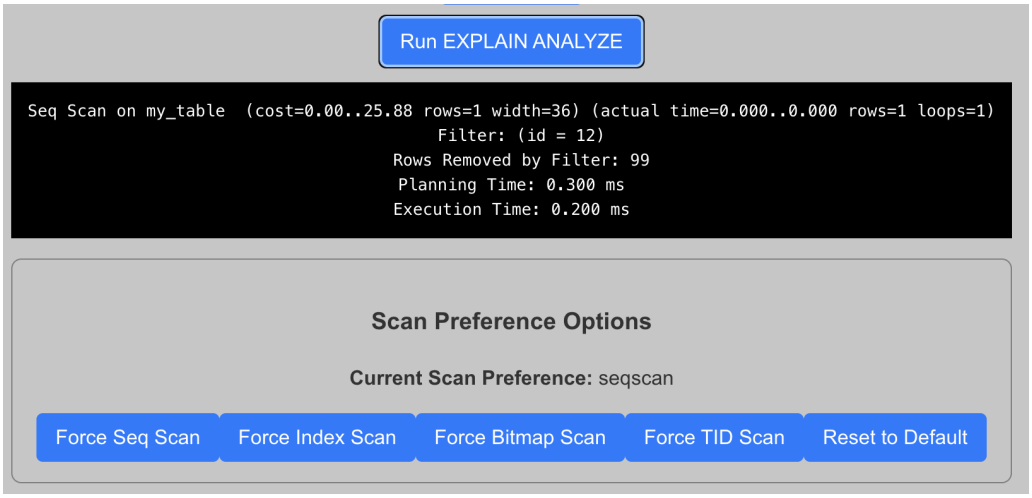

Figure 1: Selecting Sequential Scan for a query

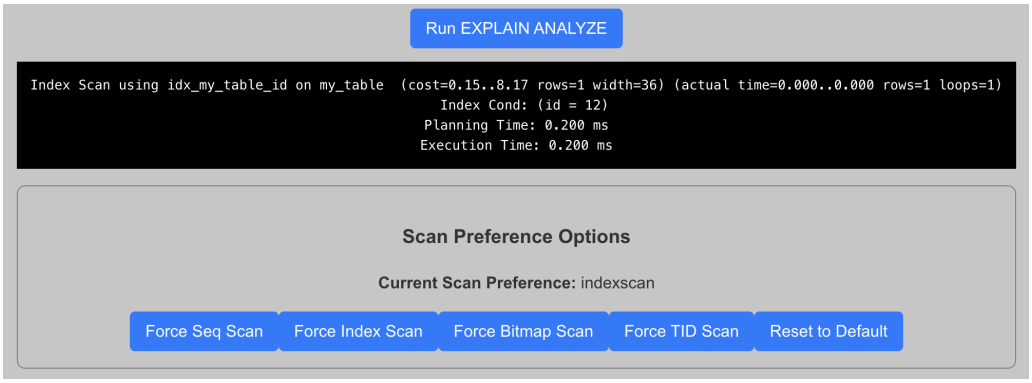

Figure 2: Selecting Index Scan for a query

The main utility of this feature is that sometimes the query planner chooses a suboptimal evaluation plan for a query. This can happen if the statistics of the tables involved in the query are not up to date. In such cases, the user can look at the performance of different evaluation plans and choose the best one. This feature is useful for debugging and optimizing queries.

3.3 Other Features

The other 3 features are still being worked on. We have included them in our future plan of action and provided a rough timeline for their implementation.

4 Future Plan of Action

5 Rough Timeline

6 Integration with XData Code

7 Challenges Faced

8 Conclusion