Differentially Private Query Optimization

Sara Alam, Chenghong Wang, Johes Bater

1. Introduction:

- Relational database systems use query optimizers to find low-cost join orders
- For example, consider the relations A, B and C in figure 1 below. Say a query requires the joins (A,B) and (A,C).

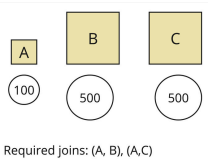
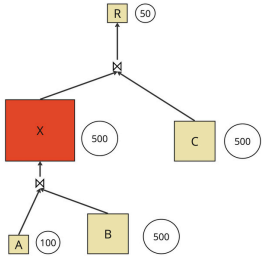


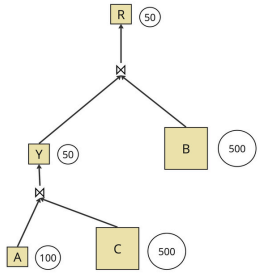
Figure 1: Example relations and joins

- There are two possible orders in which the joins can be done, as shown by figures 2 and 3.



Cost = 1600

Figure 2: Join order 1



Cost = 1150

Figure 3: Join order 2

- Optimizers use **data-dependent** information stored in the **system catalog**.
- Costs are computed and the minimum cost join order is picked
- For our example, let the cost be the sum of intermediate cardinalities. Say, system catalog information used is the fraction of relation C that has a match in relation A.

Join order	Cost (without system catalog) (records processed)	Cost (with system catalog) (records processed)
1	100 + 500 + 500 + 500 = 1600	100 + 500 + 500 + 500 = 1600
2	100 + 500 + 500 + 500 = 1600	100 + 500 + (**50%** * 100) + 500 = **1150**

Table 1: Example join order costs

2. Problem

- Problem: The system catalog leaks information to an **untrusted** server
- Key observation: Cost computation is **probabilistic** and **inherently noisy**.
- Under certain conditions, applying differential privacy, which provides proven probabilistic privacy guarantees, does not degrade query performance.
- **Differential Privacy** is widely-used privacy definition that utilizes various mechanisms like the Laplace Mechanism.
- **Question:** When and how should we add noise to satisfy our privacy goals while ensuring that the optimizers's performance is maintained?

3. Proposed Solution:

We propose the following system:

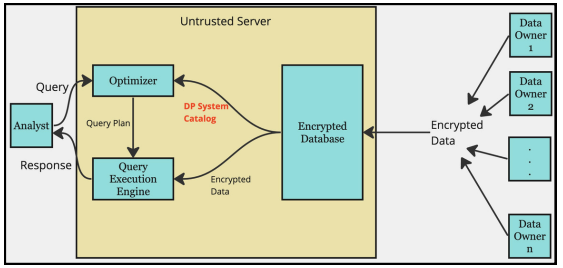


Figure 4: System Design

4. Preliminary Results:

Query	No. of joins	Rank Difference	Private Runtime (ms)	Original runtime (ms)
1	4	0	3581	3636
2	5	1	54081	54648
3	6	0	112	88

Table 2: Noisy rank difference

The table above shows the results for specific TPCH queries, their number of joins, change in rank of top-ranked plan after noise injection using a privacy budget ϵ = 0.1, execution time for top-ranked plan and the execution time for PostgreSQL-recommended plan.