Optimization II

Database Administration
Lab Guide 5

2025/2026

Steps

1. Follow the instructions in the appendix to install DuckDB.
2. Populate the PostgreSQL database using the benchmark, with a scale of 20:

```
java -jar target/benchmark-1.0-SNAPSHOT.jar \
  -d jdbc:postgresql://localhost/testdb \
  -U postgres -P postgres -p -s 20
```

3. Populate the *Invoice* table with random data:

```
INSERT INTO invoice
SELECT i,
  (random() * 1048575)::int, (random() * 1048575)::int,
  random() * 10,
  repeat(md5(random()::text), 32)
FROM generate_series(1, 2000000) t(i);
```

4. Create a database in DuckDB: `duckdb test.db`
5. With the DuckDB client, populate the `test.db` database using the `postgres` extension:

```
INSTALL postgres;
LOAD postgres;

CREATE TABLE Client AS
SELECT * FROM postgres_scan(
  'host=localhost port=5432 dbname=testdb user=postgres password=postgres',
  'public',
  'client'
);

CREATE TABLE Product AS
SELECT * FROM postgres_scan(
  'host=localhost port=5432 dbname=testdb user=postgres password=postgres',
  'public',
  'product'
);
```

```
CREATE TABLE Invoice AS
SELECT * FROM postgres_scan(
  'host=localhost port=5432 dbname=testdb user=postgres password=postgres',
  'public',
  'invoice'
);
```

6. Create indexes and run `ANALYZE` on both systems:

   ```
   CREATE INDEX client_id ON Client (id);
   CREATE INDEX product_id ON Product (id);
   CREATE INDEX invoice_clientid ON Invoice (clientid);
   CREATE INDEX invoice_productid ON Invoice (productid);
   ANALYZE;
   ```

7. Visualize the execution plans of the following queries in both systems (`EXPLAIN ANALYZE`):

   (a)
   ```
   SELECT p.id, p.description
   FROM Product p
   JOIN Invoice i ON i.productid = p.id
   WHERE i.clientid = 12345;
   ```
   (b)
   ```
   SELECT i.productid
   FROM invoice i
   WHERE i.clientid = 12345;
   ```
   (c)
   ```
   SELECT p.id, count(i.id)
   FROM product p
   JOIN invoice i on i.productid = p.id
   GROUP BY p.id
   ORDER BY count(i.id) DESC
   LIMIT 10;
   ```

Questions

1. How does query 7a compare in terms of plan and execution time in both systems? What about 7b?
2. How does the selection of table *Invoice* in 7a compare against the one in 7b in DuckDB? Try to rewrite 7a in order to use the same selection algorithm as 7b.
3. How does query 7c compare in terms of plan and execution time in both systems? What causes the difference in execution time?
4. Can the performance of 7c be improved in both systems? Explain.
5. What conclusions can be taken away about the performance of both systems?

Learning Outcomes Get familiarized with DuckDB. Compare execution plans and performance of systems with different execution and storage engines.

DuckDB Instructions

Install

- Linux/MacOS: `curl https://install.duckdb.org | sh`
- Windows (`winget`): `winget install DuckDB.cli`
- Other: `https://duckdb.org/install`

Usage

- Open client:

  ```
  # uses an in-memory database
  duckdb

  # uses (or creates if it does not exist) a file to persist the database
  duckdb <file>
  ```

- Show schema: `.schema`
- Quit client: `.q`
- More options: `.help`