Benchmark

Database Administration
Lab Guide 0

2025/2026

Consider a simplified invoice processing system, assuming that each transaction sells one item of a single product. Use the following tables:

Client: Id, Name, Address, Data.

Product: Id, Description, Data.

Invoice: Id, ProductId, ClientId, Data.

This application should provide the following operations:

Sell: Add invoice record.

Account: List names of products sold to some client.

Top10: List currently 10 most sold products

Client and product tables should be pre-populated with `MAX` $= 2^n$ items, for some n (`scale` parameter of the benchmark). Generate client and product identifiers with `rand.nextInt(MAX)|rand.nextInt(MAX)` when creating new invoices. Use a long string ($>$ 1000 characters) for Data fields in each table.

Steps

1. Implement the proposed system using an RDBMS, starting with a tool to initialize and populate the database.
2. Design and run queries for each proposed operation. Test them in the `psql` client.
3. Implement a workload generator, that invokes random operations (e.g., 50% sell, 25% account, 25% top10) with a varying number of client threads. Collect response time and throughput statistics.
4. Run the benchmark for 1, 2, 4, 8, ... client threads and record *throughput* and *response time* results. Plot response time vs. throughput to obtain a *scalability curve*.

Questions

1. After running the benchmark for some time, what are the numerical identifiers of the top 10 products?
2. How are the product sales distributed?
3. What is the maximum throughput of your system? What is the expected response time?
4. How does the throughput and response time vary with the increasing number of client threads?

Learning Outcomes Deploy and run an instance of PostgreSQL server. Recall relational database programming with SQL and JDBC. Compute and recognize the significance of key performance statistics. Evaluate and explain performance and scalability using a scalability curve.

PostgreSQL HowTo

With Docker

1. Create the container:

   ```
   $ docker run --name postgres -e POSTGRES_PASSWORD=postgres \
       -p 5432:5432 -d postgres:17
   ```

2. Access the psql client:

   ```
   $ docker exec -it postgres psql -U postgres
   ```

3. Create a new database `testdb`:

   ```
   # in psql
   psql> create database testdb;

   # or using createdb
   docker exec -it postgres createdb -U postgres testdb
   ```

4. Connect to the new database:

   ```
   psql> \c testdb
   ```

5. Get the list of relations:

   ```
   psql> \d
   ```

6. To restart the server:

   ```
   $ docker restart postgres
   ```

7. To stop the container:

   ```
   $ docker stop postgres
   ```