Execution

Database Administration
Lab Guide 2

2025/2026

Consider the EXPLAIN statement that displays the execution plan and statistics using the following alternatives:

- `EXPLAIN SELECT ...`
- `EXPLAIN ANALYZE SELECT ...`
- `EXPLAIN (ANALYZE,BUFFERS) SELECT ...` [1]

Steps

1. Follow the cloud virtual machine instructions in the appendix.
2. In `psql`, observe the execution plans for each read query with `EXPLAIN ANALYZE`. (Disregard cost information.)
3. Observe the system's resource usage of queries with `EXPLAIN ANALYZE` and with operating system tools while the benchmark is running.
4. Disable operators in the configuration file and repeat the observation and benchmark. For example:[2]

   ```
   psql> alter system set enable_hashjoin to false;
   psql> select pg_reload_conf();
   ```

5. Follow the bucket instructions in the appendix.
6. With *User Defined Functions* in Python (check the appendix for more information):

   (a) Create a table function that generates random strings:

   ```
   CREATE FUNCTION random_str(n_strings integer, str_length integer)
   RETURNS setof varchar AS $$
       import random, string
       chars = string.ascii_letters + string.digits
       for _ in range(n_strings):
           yield ''.join(random.choices(chars, k=str_length))
   $$ LANGUAGE plpython3u;
   ```

   (b) Analyze the execution plans of the following queries in terms of I/O (ensure *Product* has data):

   ```
   SELECT * FROM product;
   SELECT * FROM product LIMIT 10;
   SELECT * FROM random_str(100000, 10);
   SELECT * FROM random_str(100000, 10) LIMIT 10;
   ```

   (c) Create a table function to retrieve foreign exchange data (replace `BUCKET_FOREX_EUR_LINK` with the `forex_eur.json` link created by the bucket):

[1] `BUFFERS` is automatically enabled since PostgreSQL 18.

[2] List of operator-specific parameters: `https://www.postgresql.org/docs/18/runtime-config-query.html#RUNTIME-CONFIG-QUERY-ENABLE`

```
CREATE TYPE forex AS (
  code varchar,
  name varchar,
  rate decimal,
  extra varchar
);

CREATE OR REPLACE FUNCTION current_forex() RETURNS setof forex AS $$
  import requests, json
  response = requests.get('BUCKET_FOREX_EUR_LINK')
  object = json.loads(response.content.decode('utf8'))
  for code, info in object.items():
    yield {
      'code': code,
      'name': info['name'],
      'rate': info['rate'],
      'extra': '0' * 100000
    }
$$ LANGUAGE plpython3u;
```

(d) With the `VERBOSE` flag, compare the *Output* of `Function Scan` with the `Sequential Scan` in the query that determines the total money spent by the client with `id=123` in the currency `code=usd`:

```
EXPLAIN (ANALYZE, BUFFERS, VERBOSE)
SELECT sum(price * rate) AS total
FROM invoice, current_forex()
WHERE clientid = 123 AND code = 'usd';
```

Questions

1. What are the effects on the plan and on the response time of disabling the default join operator in the "top products" query? What about also disabling the alternative operator used?
2. Why do database systems have different physical implementations of logical operators?
3. What is the impact of the join algorithm in memory usage and I/O?
4. What is the impact of the grouping algorithm in memory usage and I/O?
5. Discuss the execution semantics of `Sequential Scan` vs `Function Scan`.

Learning Outcomes Describe execution plans for abstract relational operations. Relate query plan with resource usage. Relate resource usage with performance. Compare the execution of table and function scans. Get familiarized with cloud environments.

Additional Tools

Plan visualizer

- `https://explain.depesz.com`
- `https://explain.dalibo.com`
- `http://pgbadger.darold.net`

Hardware usage

- `htop` – `https://linux.die.net/man/1/htop`
- `psutil` – `https://github.com/giampaolo/psutil`

Google Cloud virtual machine instructions

IMPORTANT: provisioning a virtual machine in Google Cloud has an associated hourly cost. Be sure to delete the instance at the end as each student has a limited amount of credits.

1. Prepare a zip file with the benchmark (`skelbench.zip`) and the installation script (`install.sh`);
2. Go to `https://console.cloud.google.com`
3. If no project has been created yet, select *Select a project* - *New project*. Give it a name and select the previously claimed education credits.
4. Go to `https://console.cloud.google.com/apis/library/compute.googleapis.com` and enable the Compute Engine API.
5. Open the navigation menu on the top-left side, and access *Compute Engine*;
6. On the left menu, open *Metadata*; Select *SSH Keys* - *Edit* - *Add Item* and add your public SSH key - *Save*;
7. Back in *Compute Engine*, select *Create Instance* and create the virtual machine as follows:
   (a) *Machine Configuration* – N2, Custom, 4 vCPU, 4 GB RAM;
   (b) *OS and storage* – *Change* - Ubuntu, Ubuntu 25.04 LTS x86/64, SSD persistent disk, 50 GB;
   (c) *Advanced* – Provisioning model: Spot (machines created with this option can be randomly turned off, but are available at a lower price. More information: `https://cloud.google.com/compute/docs/instances/spot`);
8. Move the auxiliary zip file to the cloud instance. E.g.: `scp` *files.zip user*`@`*ip*`:` (the public IP can be found in the *Compute Engine* page; the *user* comes from the SSH key used; do not forget the ":" at the end of the command);
9. Access the virtual machine through SSH and install everything:

```
# access the virtual machine and extract the files
ssh user@ip
sudo apt install unzip -y
unzip files.zip
cd files

# installs postgres and java
sudo chmod +x install.sh
./install.sh

# check that postgres is running (\q to exit)
PGPASSWORD=postgres psql -U postgres
```

```
# test the benchmark
cd skelbench
mvn package
java -jar target/benchmark-1.0-SNAPSHOT.jar \
    -d jdbc:postgresql://localhost/testdb \
    -U postgres -P postgres -p -x -R 15 -s 15 -c 8
```

10. After completing the guide, **delete the virtual machine** by accessing the *Compute Engine* page, clicking the three dots on the respective instance, and selecting *Delete*.

Google Cloud storage bucket instructions

IMPORTANT: Delete unused data after completing the guide to avoid unwanted costs. It is not necessary to delete the bucket.

1. Create a new bucket:
   (a) (On the left menu) Select Cloud Storage → Buckets → Create;
   (b) Name the bucket → Continue;
   (c) *Choose where to store your data* → Region → us-east1 → Continue;
   (d) *Choose how to store your data* → Continue;
   (e) *Choose how to control access to objects* → Disable "Enforce public access prevention on this bucket" → Set "Fine-grained" on "Access Control" → Continue;
   (f) Select Create.
2. Add the `forex_eur.json` file to the bucket, by clicking Upload Files or by dragging the files directly to the bucket's page (Cloud Storage → Buckets → *Bucket name*).
3. Select the three dots on the right side of the file row → Select "Edit access" → Select "Add Entry" → Set the new row to "Public", "allUsers", and "Reader" → Save.
4. Confirm everything is working: copy the public link and execute
   `curl <copied_link>`
   (e.g., `curl https://storage.googleapis.com/bucket/forex_eur.json`).
   The output should show the json data.

User Defined Functions in Python

1. Install the packages in the server/container running Postgres (already installed in the `install.sh` script):

   ```
   apt install -y postgresql-plpython3-18 python3-requests
   ```

2. Create the `plpython3u` extension in the target database:

   ```
   CREATE EXTENSION plpython3u;
   ```

3. Documentation is available at `https://www.postgresql.org/docs/18/plpython-funcs.html`.