POSTGRESQL DATABASE HANDS-ON SESSION 1

Valter Uotila

OUTLINE

Presentation about PostgreSQL

Demo on PostgreSQL

Hands-on exercises

Welcome!

Welcome to the university to start your first (or *n*th year like me)!

General recommendations for all courses when doing practical assignments or larger projects on yourselves:

- We encourage to cooperate and work together
- There are multiple channels to ask help and have a discussions:
  - Also, good old email works: `firstname.lastname@helsinki.fi`
  - For this session I am using HackMD: https://hackmd.io/@valterUo/rkeRRp7zD/edit

CONCEPT VS. FACT

Another video on Youtube: Marty Lobdell - Study Less Study Smart.

Facts

- Top 4 most used databases: Oracle, MySQL, Microsoft SQL Server, PostgreSQL

Concepts

- How does a relational database work? What are fundamental building blocks of it and how different operations work?

CONCEPT VS. FACT

- Facts you can google
- Concepts are important. They stay with you if you understand them
- So instead of memorizing, understanding has the main role

MOTIVATION

- Databases are important part of many systems
- The relational databases are still the most used databases
- With small amount of work you should be able to integrate PostgreSQL with your future projects, for example, as a part of your full stack program
- All relational databases have similar overall design and working principles: If you know how to execute queries on PostgreSQL, you should be able to use MySQL, Oracle, SQLite etc. databases with minor modifications. Besides, even many NoSQL query languages look like SQL.

LEARNING OBJECTIVES

- Querying Data
  - SELECT, SELECT DISTINCT
  - Column aliases
  - ORDER BY
- Filtering Data
  - WHERE, LIMIT, FETCH, IN, BETWEEN, LIKE, IS NULL
- Joining Multiple Tables
  - Natural, inner, left, full outer, self, cross joins
  - Table aliases
- Understanding PostgreSQL Constraints
  - Primary and foreign key
  - CHECK, UNIQUE, NOT NULL constraints

FUNDAMENTALS OF RELATIONAL DATABASE SYSTEM

- Theoretical foundation on E. F. Codd's paper *A Relational Model of Data for Large Shared Data Banks*
- Surprisingly, NoSQL databases are lacking similar rigorous unifying theory

BASIC OPERATIONS

- Select: $\sigma_{\phi}(R)$
- Project: $\Pi_{a_1,\ldots,a_n}(R)$
- Cartesian product (flat-version):

$$R \times S = \{(r_1, \ldots, r_n, s_1, \ldots, s_t) \mid (r_1, \ldots, r_n) \in R, (s_1, \ldots, s_t) \in S\}$$

- Rename: $\rho_{a/b}(R)$

SELECT OPERATION

Example 1

```
1 SELECT first_name FROM customer;
```

Example 2

```
1 SELECT
2     first_name, last_name, email
3 FROM
4     customer;
```

SELECT ALL

Example 3

```
1 SELECT * FROM customer;
```

COLUMN ALIASES

Example 4

```
1 SELECT
2     first_name,
3     last_name AS surname
4 FROM customer;
```

COLUMN ALIASES

”AS” is optional in PostgreSQL:

Example 5

```
1 SELECT
2     first_name ,
3     last_name surname
4 FROM customer;
```

If an alias contain space, the alias needs to be wrapped in quotation marks.

ORDER BY OPERATION

Example 6

```
1 SELECT
2         first_name, last_name
3 FROM
4         customer
5 ORDER BY
6         first_name;
```

Options are ascending order (`ASC`) or descending order (`DESC`). Default is ascending order.

ORDER BY OPERATION

Example 7

```
1 SELECT
2         first_name , last_name
3 FROM
4         customer
5 ORDER BY
6         first_name ASC;
```

ORDER BY OPERATION

Example 8

```
1 SELECT
2         first_name , last_name
3 FROM
4         customer
5 ORDER BY
6         first_name DESC;
```

ORDER BY OPERATION

Example 9

```
1  SELECT
2          first_name , last_name
3  FROM
4          customer
5  ORDER BY
6          first_name ASC;
7          last_name DESC;
```

First sorts by first_name, then sorts the sorted values by last_name.

SELECT DISTINCT OPERATION

- Selects distinct rows i.e. removes duplicate rows from the result table

FILTERING DATA: LOGICAL OPERATORS

Operator	Description
=	Equal
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
<> or !=	Not equal
AND	Logical operator AND
OR	Logical operator OR
IN	Return true if a value matches any value in a list
BETWEEN	Return true if a value is between a range of values
LIKE	Return true if a value matches a pattern
IS NULL	Return true if a value is NULL
NOT	Negate the result of other operators

FILTERING DATA

Example 10

```
1  SELECT
2          last_name,
3          first_name
4  FROM
5          customer
6  WHERE
7          first_name = 'Jamie';
```

FILTERING DATA

Example 11

```
1  SELECT
2          last_name,
3          first_name
4  FROM
5          customer
6  WHERE
7          first_name = 'Jamie' AND
8          last_name = 'Rice';
```

FILTERING DATA

Example 12

```
1  SELECT
2          first_name,
3          last_name
4  FROM
5          customer
6  WHERE
7          last_name = 'Rodriguez' OR
8          first_name = 'Adam';
```

FILTERING DATA

Example 13

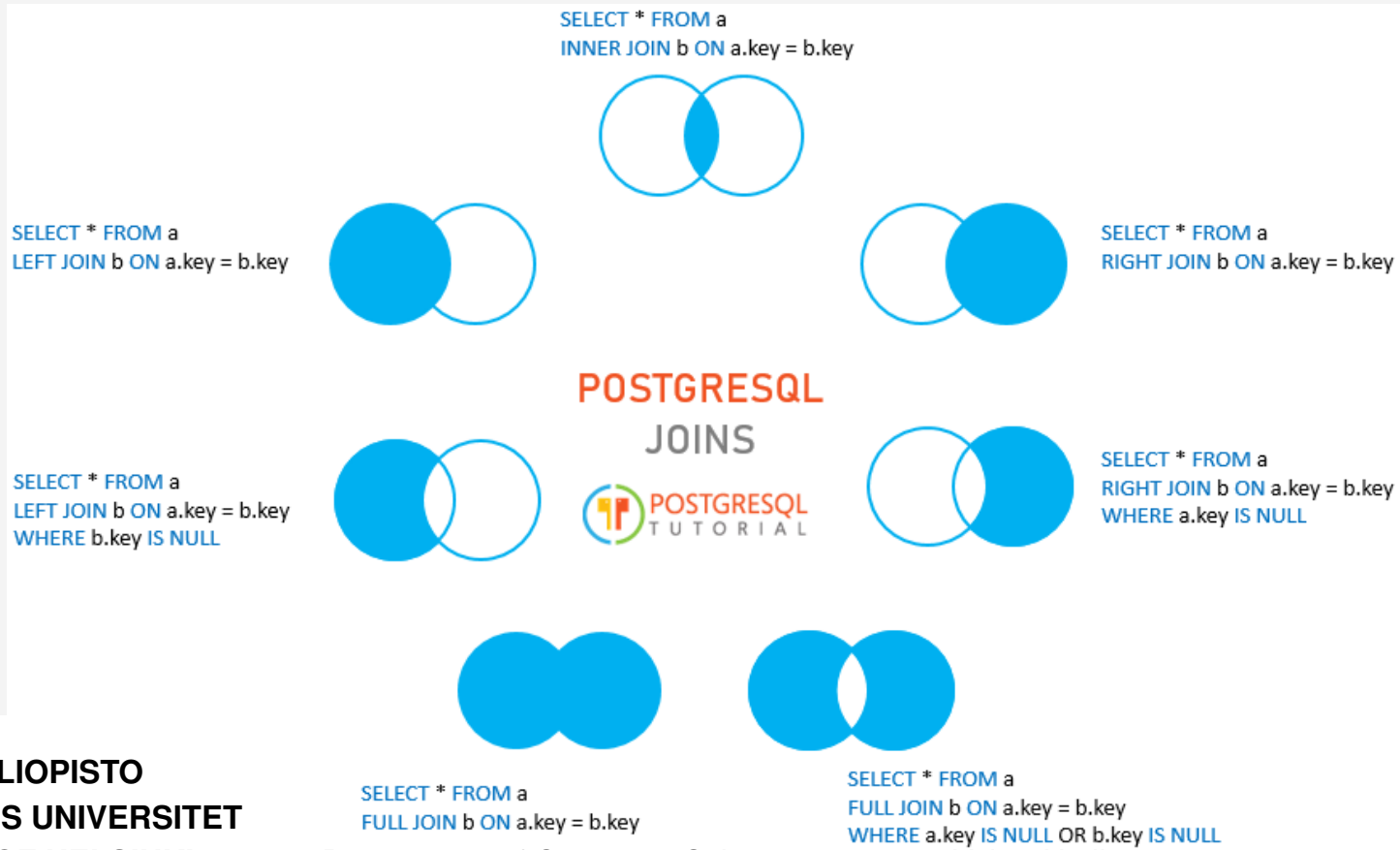
```
1  SELECT
2          first_name,
3          LENGTH(first_name) name_length
4  FROM
5          customer
6  WHERE
7          first_name LIKE 'A%' AND
8          LENGTH(first_name) BETWEEN 3 AND 5
9  ORDER BY
10         name_length;
```

JOINS

- Previous examples were using selection and projection. Joins are using Cartesian product.
- All the joins can be defined with Cartesian product, projection and selection but generally evaluation is not using Cartesian product because it is expensive to evaluate.
- There exist relational algebra and relational calculus.

JOINS

Because usually data is distributed along multiple tables, we need to join tables to obtain full results.



NATURAL JOINS AND SELF JOIN

- Besides the joins in the previous figure, we have natural joins and self join
- INNER, LEFT and RIGHT JOIN have two identical columns in the result table
- Natural join removes this

INNER JOIN

```
1 SELECT
2     f.title, l.name
3 FROM
4     language l
5 INNER JOIN
6     film f
7 ON l.language_id = f.language_id;
```

TABLE ALIASES

- We defined column aliases with AS keyword. Similarly we can define table aliases which are useful when we refer the tables in the query.

Example 14

```
1  SELECT
2      c.customer_id, first_name, amount
3  FROM
4          customer AS c
5  INNER JOIN payment AS p
6      ON p.customer_id = c.customer_id;
```

CONSTRAINTS

- While creating tables, we can define constraints which ensures the accuracy and reliability of the data
- NOT NULL constraint: ensures that a column cannot have NULL value.
- UNIQUE constraint: ensures that all values in a column are different.
- PRIMARY key: uniquely identifies each row/record in a database table.
- FOREIGN key: constrains data based on columns in other tables.
- CHECK constraint: CHECK constraint ensures that all values in a column satisfy certain conditions.

PRIMARY AND FOREIGN KEYS

Example

```
1 CREATE TABLE country (
2     country_id INT PRIMARY KEY,
3     country VARCHAR (50) NOT NULL,
4     last_update VARCHAR (50)
5 );
```

PRIMARY AND FOREIGN KEYS

Example

```
1  CREATE TABLE city (
2          city_id INT PRIMARY KEY,
3          city VARCHAR (50),
4      country_id INT NOT NULL,
5      last_update VARCHAR (50),
6      CONSTRAINT fk_country
7        FOREIGN KEY(country_id)
8              REFERENCES country(country_id)
9  );
```

POSTGRESQL INSTALLATION

- Download Postgres from www.postgresql.org
- Start the installation. It should be easy and succeed by just clicking ”next”.
- When you create a password, remember to memorize. It will be needed.

CREATE NEW DATABASE

Open the command line and start PostgreSQL server:

```
1 psql -U postgres
```

- `-U` sets user to `postgres`
- When the password is asked, use the password you defined during the installation.

```
1 CREATE DATABASE dvdrental;
2 exit;
```

CREATE NEW DATABASE

Download the demo database from https://www.postgresqltutorial.com/postgresql-sample-database/. Extract the .zip file so you will get access to .tar file. Navigate to the bin folder:

```
cd C:\Program Files\PostgreSQL\12\bin
```

Use `pg_restore` tool to upload the dvdrental data into the database you just created:

```
pg_restore -U postgres -d dvdrental "C:\path_to_file\dvdrental.tar"
```

Make sure the path is correct. The password is asked again. The quotation marks around the file path might be necessary.

CONNECTING TO DATABASE

Again, open the command line and start the server

```
psql -U postgres
```

Use the meta-command (not part of SQL but Postgres):

```
\connect dvdrental;
```

or equivalently

```
\c dvdrental;
```

Now you should be able to query the dvdrental database.

BREAKOUT ROOMS AND EXERCISES

- Zoom will randomly divide you into breakout rooms where you work together
- Please see the questions in HackMD page: https://hackmd.io/@valterUo/rkeRRp7zD/edit
- I will go around the rooms and see if you have any questions
- You can also write questions and comments to HackMD page