Relational databases : Querying

- Having modeled your data allows you to store every information properly
- To modify and retrieve data, we need to operate **ISUD** operations
- **ISUD** stands for `INSERT`, `SELECT`, `UPDATE`, `DELETE`
- In the previous lectures, we learnt how to create tables with the `CREATE` verb, and modify any structural information with `ALTER`. Now we will act on data

Relational databases : Inserting with "INSERT"

- To insert new data in a table, you have to know its structure
- Hereafter is a sample `INSERT` statement

```
INSERT into facilities(facid, name, membercost, guestcost, initialoutlay, monthlymaintenance)
          values (9, 'Swimming Pool', 8, 16, 3000,3000)
```

- Even if the first parenthesis is optional, it is strongly recommended to use it, it gives a clear idea of what are the data inserted

Relational databases : Updating with "UPDATE"

- To update lines of a table, you have to know its structure
- Hereafter is a sample `UPDATE` statement

```
UPDATE facilities SET membercost=10 WHERE name='Swimming Pool';
```

- notice the `SET` operator that targets a column, and replace the old value
- some where clause can be used to precise the target

> Before using an `UPDATE` query, always do a select to make like a backup of data before the update

Relational databases : deleting with "DELETE"

- To delete lines of a table, you can use the `DELETE`, as shown below

```
DELETE from facilities WHERE name='Swimming Pool';
```

- Some where clause can be used to precise the target

> Before using a `DELETE` query, always do a select to make like a backup of data before the update

Relational databases : Querying

- `SELECT` verb in combination with `FROM` operator
- The result of this select is a relation containing tuples
- One can select specific columns of an existing table, specified by `FROM`
- Some filters can be applied thanks to the `WHERE` clause
- logical & arithmetical operators can be used (`=`, `<`, `>`, `<>`, `AND`, `OR`) to combine several conditions

SQL Queries : practising

- Launch the query tool from your pgAdmin console
- Load the data present in the attached clubdata (credits : https://pgexercises.com/)
- Run the query:

```
select * from bookings;
```

> ***Tip*** : if you have "relation not found", you have to update the search path
>
> ```
> show search_path ;
> set search_path = "$user", public, "cd";
> ```

Select with WHERE clauses : practising

- You can refine the previous query by adding a where clause

```
SELECT * FROM facilities
WHERE monthlymaintenance < 1000;
```

- will produce

facid	name	membercost	guestcost	initialoutlay	monthlymaintenance
0	Tennis Court 1	5	25	10000	200
1	Tennis Court 2	5	25	8000	200
2	Badminton Court	0	15.5	4000	50
3	Table Tennis	0	5	320	10
6	Squash Court	3.5	17.5	5000	80
7	Snooker Table	0	5	450	15
8	Pool Table	0	5	400	15

JOIN operators

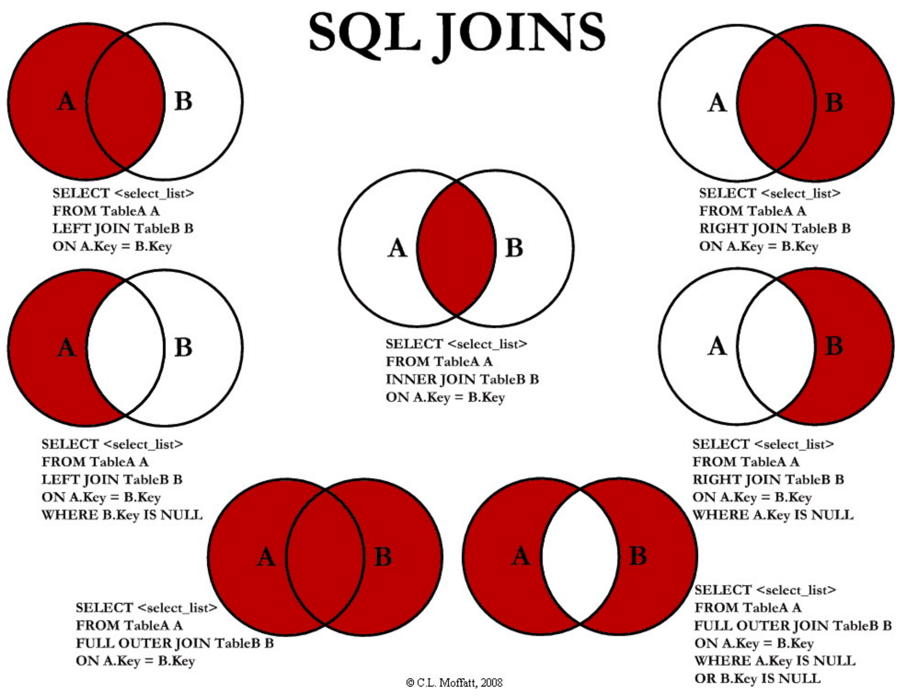
- You can combine data coming from different tables by usign the `JOIN` operator, for example

```
SELECT * FROM facilities
JOIN bookings on bookings.facid = facilities.facid
WHERE monthlymaintenance < 15;
```

- will produce

facid	name	membercost	guestcost	initialoutlay	monthlymaintenance	bookid	facid-2	memid	starttime	slots
3	Table Tennis	0	5	320	10	0	3	1	2012-07-03 11:00:00	2
6	Squash Court	3.5	17.5	5000	80	2	6	0	2012-07-03 18:00:00	2
7	Snooker Table	0	5	450	15	3	7	1	2012-07-03 19:00:00	2
8	Pool Table	0	5	400	15	4	8	1	2012-07-03 10:00:00	1
8	Pool Table	0	5	400	15	5	8	1	2012-07-03 15:00:00	1

Different types of JOIN



```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```

```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```

```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```

credits visual representation of SQL-Joins

Exercise : practising Select with joins

- Download and analyse this sql dump : tables
- Convert this SQL Script into a PostgreSQL script, and import it in a new database
- Try all the sorts of SQL Joins, as shown in the previous schema

Exercise : Going further with SQL Queries

- To practise better the PostgreSQL environment, achieve the following exercises
- pgexercises

Integrating with Programming languages

- With Python

```
import psycopg2
connection = psycopg2.connect(user = "postgres",
                              password = "postgres",
                              host = "192.168.137.50",
                              port = "10532",
                              database = "postgres_db")
```

> The driver is included in psycopg2

Integrating with Programming languages (2)

- with Nodejs

```
const { Client } = require('pg');
const client = new Client({
    user: 'postgres',
    host: '192.168.137.50',
    database: 'postgres_db',
    password: 'postgres',
    port: 10532,
});
client.connect()
```

Integrating with Programming languages(3)

- with Java

```
import java.sql.Connection;
import java.sql.SQLException;
import java.sql.DriverManager;

public class Launcher {
   public static void main(String args[]) {
      Class.forName("org.postgresql.Driver");
      try (Connection c = DriverManager
            .getConnection("jdbc:postgresql://192.168.137.50:10432/postgres_db",
            "postgres", "postgres");){
        System.out.println("connected to schema : " +connection.getSchema());
      } catch (SQLException e) {
         e.printStackTrace(); //TODO use a logger
      }
   }
}
```