Lab # 3

It is the aim to develop skills to create real databases using **SQL Language**, by means of the most popular open source DBMS **MySQL**.

DBMS software

Client-Server Architecture

Many varieties of modern software use a **client-server** architecture:

- The **client** process requests are sent to the **server** process for execution;
- Database systems divide the work of a DBMS into a **server process** and one or more **client processes**.

In the simplest client/server architecture, the entire DBMS is a server, except for the query interfaces that interact with the user and send queries or other commands across to the server.

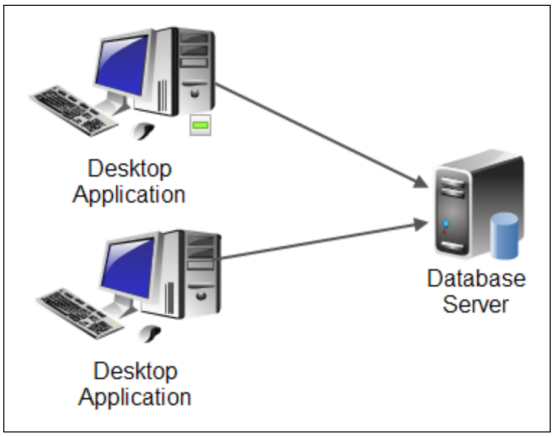

Figure 1: "Client - Server" architecture

The Database Language SQL

Query and modify database languages are used to request information or to modify databases.

The **Structured Query Language** (SQL)[1] is an ANSI standard interactive and programming language for querying and modifying data and managing databases. There are versions of SQL, produced by the principal DBMS vendors, all including these capabilities:

- SQL *Data-Definition Language*: for declaring the database schemas.
- SQL *Data-Manipulation Language*: for querying (asking questions) databases and for modifying tuples (insert, delete, update).

Introduction to SQL

Programming Languages

This section provide the basic terminology [2] of programming languages, aiming to support this introduction to the SQL language.

- **Programming languages** are formal languages for the description and control of the behavior of computers. They are used to realize a precise, reliable, and automatic management and manipulation of data and knowledge.
- A programming language is defined by its *syntax and semantics*.
  - The **syntax** of a programming language is a formal language which defines its correct expressions (**how to write statements**);
  - The **semantics** defines the meaning of the language elements, expressions, and programs (*what statements realise*).
- A programming language usually comes with a **compiler** which translates programs (of the source language) into another programming language (the target language) called **compiled language**. While the source language is typically a high-level programming language (i.e. C, Java, Visual Basic), the target language may be a lower-level language, like assembly language or machine language.
- A programming language is an **interpreted language** when an **interpreter** performs the syntactic and the semantic analysis, and then it either executes the source code, nearly directly or generates some intermediate code which is *immediately executed*.

[1] The current standard is SQL 99.

- A **programming paradigm** is a general approach the programmer can take to describe and solve a problem. This approach is supported by the programming languages of the paradigm which is reflected by the language concepts, i.e. the language constructs and evaluation mechanisms. Programming languages can be classified according to their paradigms.

SQL: declarative language

SQL assumes the **declarative paradigm**, meaning that the language describe "what" is to be computed in contrast for instance with imperative languages that instead describe "how" the computation of the solution for a certain problem takes place.

Distinguishing features of the declarative paradigm are:

- Declarative description of the problem by means of relations and/or constraints, but *the actual solution algorithm is left unspecified* and realised by the underlying language evaluation.
- Computations of declarative programs are stateless and time-independent, i.e. an expression always has the same semantic meaning independent of its context or time.

Relations in SQL

The main evidence of a relations is called table. We shall learn how to declare tables into a database.

Create a Database

1. Create and Use a Data Base

```
CREATE DATABASE dbname;

USE dbname;
```

2. Create a Table

The **CREATE TABLE** statement declares the relation schema.

```
CREATE TABLE TableName
(
ColumnName DataType(size) [PRIMARY KEY],
ColumnName DataType(size) [DEFAULT value],
ColumnName DataType(size) [NOT NULL | NULL],
ColumnName DataType(size) [CHECK (condition)],
....,
[PRIMARY KEY (ColumnName List)]
[FOREIGN KEY (ColumnName) REFERENCES TableName(ColumnName)]
);
```

3. Grant Privileges

GRANT allows user to access and use databases or simple tables created by another user.

```
GRANT privilege type ON dbname/tablename TO user;
```

4. Modify the table structure

```
ALTER TABLE table-name
ADD column-name data-type;
```

```
ALTER TABLE table-name
ADD PRIMARY KEY (attributes);
```

Manage a Database

In order to insert data into a database, and in case to update, we approach the **SQL Data Manipulation Language (DML)**.

Insert Data (a new Tuple)

```
INSERT INTO TableName [(col_name,...)]
{VALUES, VALUE}
(value_list, {NULL});
```

Update Data (given a Tuple)

```
UPDATE TableName
SET col_name = expr [, col_name = expr]
[WHERE col_name = expr ];
```

Script

Frequently one or more SQL statements are saved as file into a **SQL Script**, that is a text file recognized by the extension "`.sql`" (for example `recipe.sql`).

Scripts are generally used to automate massive processes of database creation, insertion, modification, or query.

```sql
CREATE DATABASE recipe;
USE recipe;

CREATE TABLE Course(
    name VARCHAR(100) PRIMARY KEY,
    courseType VARCHAR(100),
    difficulty VARCHAR(10),
    preparationTime NUMERIC(4)
);

CREATE TABLE Ingredient(
    name VARCHAR(100) PRIMARY KEY,
    description VARCHAR(50),
    category VARCHAR(20)
);

CREATE TABLE Recipe(
    courseName VARCHAR(100),
    ingredientName VARCHAR(100),
    quantity NUMERIC(4),
    FOREIGN KEY (courseName) REFERENCES course(name),
    FOREIGN KEY (ingredientName) REFERENCES Ingredient(name)
);
```

Figure 2: Script - book-recipe.sql

Ex. # 1

Aiming to manage a list of courses we have the following table.

```
CREATE TABLE Course(
    name VARCHAR(20) PRIMARY KEY,
    courseType VARCHAR(10),
    difficulty VARCHAR(10) DEFAULT 'N/A',
    preparationTime NUMERIC(4,0) CHECK (preparationTime > 0)
);

GRANT ALL ON Course TO ''@localhost;
```

Follows few statements in order to enter data. **Executing them we observe that some data are not registered or there are, but incorrectly!**

```
INSERT INTO Course(name, courseType, difficulty, preparationTime)
VALUES ('Lasagne', 'Pasta','Medium',90);
INSERT INTO Course(name, courseType)
VALUES ('Tiramisù', 'Dessert');
INSERT INTO Course(name, courseType, difficulty, preparationTime)
VALUES ('Lasagne', 'Pasta','Low', 120);
INSERT INTO Course
VALUES ('Cotoletta', 'Meat', 'Low', 0);
INSERT INTO Course
VALUES ('Cotoletta alla Milanese', 'Meat', NULL, NULL);
INSERT INTO Course
VALUES ('Cotoletta alla Milan', 'Meat', NULL, 35);
INSERT INTO Course
VALUES ('Risotto con funghi', 100);
INSERT INTO Course
VALUES ('Risotto con funghi', NULL, NULL, 100);

UPDATE Course
      SET difficulty = 'Low'
WHERE name = 'Lasagne';
UPDATE Course
      SET name ='Cotoletta Mil.'
WHERE name = 'Cotoletta alla Milan';
```

Ex. # 2

The table tells the percentage of recyclable material we take out from electronic devices.

Device	Material	%
laptop	plastic	45%
	iron	25%
	silicon	30%
smartphone	plastic	65%
	silicon	35%

Create by means of a `SQL statement` a table holding the given data and **refer** to the table `Object` which holds some related data.

```
CREATE TABLE Object(
   name VARCHAR(20) PRIMARY KEY,
   description VARCHAR(100),
   recyclable BOOLEAN
);

CREATE TABLE Composition(
   object VARCHAR(20),
   material VARCHAR(30),
   percentage NUMERIC(5,2),
   FOREIGN KEY (object) REFERENCES Object(name)
);
```

Ex. # 3

Since 2010, offers for voice and sms services have been collected and stored in the table below. Consider to store also the data of the picture below. It looks that the table shall be altered in order to store it. Modify properly the SQL statements avoiding to loose data/information already registered. Further since now a provider is identified by its name and its country modify `PRIMARY KEY` declaration.

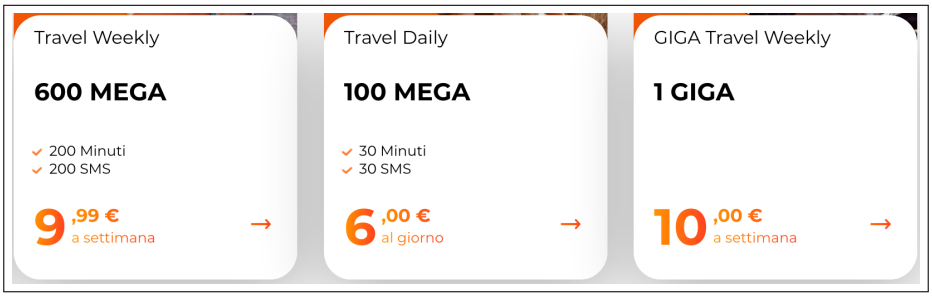

Figure 3: source: **www.windtre.it**: Internet offers to travel abroad

```
CREATE TABLE Offer(
   provider VARCHAR(10) PRIMARY KEY,
   country CHAR(2),
   offer_name VARCHAR(10),
   minutes NUMERIC(6,2),
   sms NUMERIC(5,2),
   price NUMERIC(4,1)
);
```

```
CREATE TABLE Offer(
   provider VARCHAR(10) PRIMARY KEY,
   country CHAR(2),
   offer_name VARCHAR(10),
   minutes NUMERIC(6,2),
   sms NUMERIC(5,2),
   price NUMERIC(4,1)
);
```

```
CREATE TABLE Offer(
   provider VARCHAR(10),
   country CHAR(2),
   offer_name VARCHAR(30),
   minutes NUMERIC(6,2),
   sms NUMERIC(6,2),
   internet NUMERIC(4,0),
   int_unit CHAR(4),
   price NUMERIC(4,2),
   frequency VARCHAR(20),
   PRIMARY KEY (provider, country)
);
```

Alternative

```
CREATE TABLE Offer(
   provider VARCHAR(10) PRIMARY KEY,
   country CHAR(2),
   offer_name VARCHAR(10),
   minutes NUMERIC(6,2),
   sms NUMERIC(5,2),
   price NUMERIC(4,1)
);
```

```
ALTER TABLE Offer
ADD internet NUMERIC(4,0);

ALTER TABLE Offer
ADD int_unit CHAR(4);

ALTER TABLE Offer
ADD frequency VARCHAR(20);

ALTER TABLE Offer
DROP PRIMARY KEY;

ALTER TABLE Offer
ADD PRIMARY KEY (provider, country);
```

Ex. # 4

Assuming a database for the regional railway transport system few **relations** were designed. The SQL statement to create the corresponding tables are reported.

```
CREATE TABLE Train(
    num INTEGER PRIMARY KEY,
    type VARCHAR(30),
    maxSpeed NUMERIC(6,3)
);

CREATE TABLE Station(
    name VARCHAR(30) PRIMARY KEY,
    place VARCHAR(30),
    num_platform INTEGER
);
```

In the design process we have identified a **many-to-many relationship** between **train** and **station**. Write the corresponding SQL statement to create the relationship, further write SQL statements to insert data (choose data as you like): a train, a station and their relationship.

```
CREATE TABLE Train_stop(
    num INTEGER,
    name VARCHAR(30),
    FOREIGN KEY (num) REFERENCES Train(num),
    FOREIGN KEY (name) REFERENCES Station(name)
);

INSERT INTO Train VALUES(2064, 'RV', 180.5);
INSERT INTO Station VALUES('Lambrate', 'Milano', 12);
INSERT INTO Train_stop VALUES(2064, 'Lambrate');
```

Ex. # 5

"Climate change has direct impacts on global temperature increase, rising sea levels and more extreme weather conditions." Eurostat databank provides annual European temperature deviation (globally) in the recent years.

Annual European temperature deviation [Degree Celsius]			
SOURCE/TIME	2011	2015	2019
Met Office Hadley Centre and Climatic Research Unit	1.57	1.89	2.02
NASA Goddard Institute for Space Studies	1.73	1.99	2.12
National Oceanic and Atmospheric Administration	1.68	1.98	2.18

Table 1: European Environment Agency (EEA)

The goal is to arrange data in a data set having (**only**) the following variables.

Source	Time	Temperature Deviation	Unit

Assume to converge the resulting **data set** into a single "relation". Write the SQL statement that creates the relation with the advice to define a 'natural' key (avoid to use ID), and furthermore write SQL statements to enter at least two tuples.

```
CREATE TABLE Climate(
    source VARCHAR(100),
    time INTEGER,
    tDev NUMERIC(4,2),
    unit VARCHAR(15),
    PRIMARY KEY(source, time)
);

INSERT INTO Climate
VALUES ('Met Office Hadley Centre and Climatic Research Unit', 2011, 1.57, 'Degree Celsius');
INSERT INTO Climate
VALUES ('Met Office Hadley Centre and Climatic Research Unit', 2015, 1.89, 'Degree Celsius');
```

Web source: Scraping and Database

Currently the Web could be considered as the most important source of data and information, but unfortunately generally embodied into HTML language sections. Basically by scraping process we are able to gather data from web pages and arrange them into a structured format (data set). Considering that typically we aim to collect huge amount of data should be a good practice to insert the whole data set into a relational database (properly designed).

Considering skills in Scraping process and in SQL language, we could set up a process in order to flow data from the Web to DBMS system.

We assume the question: "How luxury cloths for kid discount varies in UK cloths market?"

We consider as source of data and information **YOOX shop online**.

For this reason a process is set up: kids cloth discounted offers are selected on the shop online, then scraped and finally arranged in a structured format (relation), hence stored into a database by means of DBMS MySQL. Here below a piece of the process implemented in R language which transforms **data frame rows** into **INSERT SQL statements**.

```
# SQL INSERT INTO
table <- "Cloths"
s <- c()
for (r in 1:nrow(df)) {
  stat <- paste("INSERT INTO ", table, " VALUES(", r, sep="")
  for (c in 1:ncol(df)) {
    if (is.na(df[r,c])) {
      value <- 'NULL'}
    else if (is.numeric(df[r,c])){
      value <- df[r,c]
    } else {
      value <- paste("'", df[r,c], "'", sep="")
    }

    stat <- paste(stat, value, sep=",")
  }
  stat <- paste(stat, ");", sep="")
  s <- c(s, stat)
}
```

The outcome of the process is therefore a `.sql` script file holding the list of `INSERT INTO` statements to run by means of MySQL.

```
INSERT INTO Cloths VALUES(1,'N°21','Jumpers',189,63,69);
INSERT INTO Cloths VALUES(2,'BRUNELLO CUCINELLI','Boots',498,67,160);
INSERT INTO Cloths VALUES(3,'DSQUARED2','Denim jackets',498,53,230);
```

References

1. http://www.w3schools.com/sql/
2. https://www.mysql.it/