Taller I: Definición de Datos en SQL

Objetivos

a) Introducción a docker para bases de datos
b) Conocer las principales funcionalidades del entorno de trabajo *pgAdmin IV*.
c) Crear, documentar y guardar scripts SQL.
d) Familiarizarse con los siguientes comandos del lenguaje SQL:

```
CREATE DATABASE                 | DROP DATABASE
CREATE TABLE                    | DROP TABLE
INSERT INTO ... VALUES          | DELETE FROM ...
COPY
```

1. *(Instalación)* Descargar e instalar Docker y docker-compose, de la pagina oficial
   a) En una terminal, verificamos que los tengamos instalados correctamente, corriendo los siguientes comandos:

```
$ docker -v
Docker version 24.0.2, build cb74dfc
$ docker-compose -v
Docker Compose version v2.19.1
```

   b) Como recomendación, creen una carpeta donde vamos a generar todo lo relacionado a la base de datos.
   c) Tenemos que crear dos archivos, primero es el `Dockerfile` con el siguiente contenido:

```
FROM postgres:14.1-alpine

LABEL author="BDD-Fiuba"
LABEL description="Postgres Image for BDD-FIUBA"
LABEL version="1.0"

COPY *.sql /docker-entrypoint-initdb.d/
```

d) El segundo es el `docker-compose.yaml` con el siguiente contenido:

```
version: '3.9'

services:
# Servicio para PostgreSQL
postgres:
build:
  context: .
  dockerfile: Dockerfile
image: postgres:14
container_name: bdd_postgres_db
restart: always
environment:
  POSTGRES_DB: bdd_db
  POSTGRES_USER: admin
  POSTGRES_PASSWORD: admin123
ports:
  - "5432:5432"
volumes:
  - ./data/postgres:/var/lib/postgresql/data

# Servicio para PgAdmin
pgadmin:
image: dpage/pgadmin4:7.5
container_name: bdd_pg_admin
restart: always
environment:
  PGADMIN_DEFAULT_EMAIL: admin@gmail.com
  PGADMIN_DEFAULT_PASSWORD: admin123
ports:
  - "5050:80"
```

e) Estos dos archivos nos van a permitir descargar la imagen de PostgresSQL que posee el motor de base de datos y además pgadmin para poder administrar la base de datos. Por lo tanto, tenemos que iniciar nuestros containers:

- Tenemos que correr el siguiente comando para poder crear los containers de estos dos servicios:

```
$ docker-compose up -d
```

- Si queremos apagar los containers:

```
$ docker-compose down
```

f) Tenemos otro comando útil de docker

- Para listar todos los containers que tenemos corriendo:

```
$ docker ps -a
```

2. *(Establecimiento de una conexión).* Ingrese al administrador *pgAdmin*, nuestro docker-compose expone una pagina para acceder al mimso http://localhost:5050/
   En el `docker-compose.yaml` se definio el user y el password

```
...
environment:
  PGADMIN_DEFAULT_EMAIL: admin@gmail.com
  PGADMIN_DEFAULT_PASSWORD: admin123
...
```

   Tómese unos minutos para explorar sus íconos principales y menúes. Cree una conexión al servidor de *PostgreSQL* local a su computadora, que escucha por defecto en el puerto `5432` de `localhost`.

Nota

Si nos gusta más la consola, o no tenemos una interfaz visual, podemos siempre conectarnos utilizando comandos. En PostgreSQL, contamos con el comando `psql` el cual nos permite conectarnos por consola a la base de datos.
Como nuestra base vive dentro de un docker vamos a entrar al docker con el siguiente comando:

```
$ docker exec -it <CONTAINER_NAME> <COMAND>
$ docker exec -it bdd_postgres_db bash
```

Aquí si vamos a tener todo instalado (gracias a la imagen de docker). Podemos ejecutar el siguiente comando:

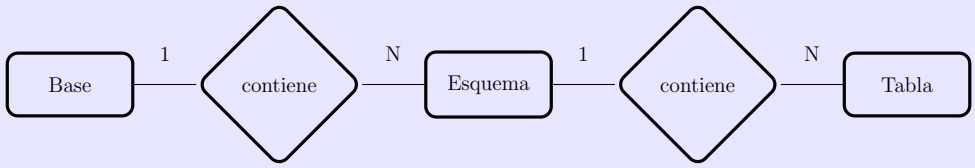
```
root@123:/# psql -h <host> -p <port> -U <username> <dbName>
root@123:/# psql -h localhost -p 5432 -U admin schooldb
psql (14.8 (Debian 14.8-1.pgdg120+1))
Type "help" for help.

schooldb=#
```

Por default siempre va a existir la base `postgres`.

3. *(Creación de una nueva base)* Utilice el comando *New Database...* para crear una nueva base de nombre `SchoolDB`. Una vez creada, navegue por la estructura *SchoolDB* → *Schemas* → *Public* → *Tables*.

Nota

Una base en *Postgres* está conformada por un conjunto de `esquemas (schemas)`, y un esquema está formado por un conjunto de `tablas` (aunque un esquema también contiene otros objetos, como funciones, vistas, tipos de dato, etc.).



La diferencia entre bases y esquemas es que una conexión a un servidor de *PostgreSQL* se realiza a una base específica, aunque puede trabajar con más de un esquema de dicha base. Los esquemas son una separación lógica de las tablas, mientras que tablas en bases distintas no pueden verse entre sí.
Cuando instalamos *PostgreSQL*, automáticamente se configura una base `postgres` con un esquema `public`.

4. *(Creación de una tabla)* Reconozca el ícono que abre el editor de SQL. Escriba un script con consultas de tipo `CREATE TABLE` en lenguaje SQL a los efectos de crear las siguientes dos tablas:

   - `teams(team, players_used, avg_age, possession, games, goals, assists, cards_yellow, cards_red)`
   - `matches(team1, team2, goals_team1, goals_team2, stage)`

Sugerencias

Escriba sus consultas SQL en varias líneas y utilizando tabulaciones, de manera que sean más legibles. El Anexo incluye una consulta de creación de tabla a modo de ejemplo. Puede introducir comentarios al final de las líneas con los caracteres '`--`' para documentar su script.

Guarde el script en un archivo con extensión `.sql`. Visualice una de las tablas creadas en el explorador de objetos. Observe las opciones que ofrece el menú contextual, en particular las opciones dentro de *New Object*, *Delete/Drop*, *Scripts* y *View Data*.

5. *(Eliminación de tablas)* Mejore el script anterior anteponiéndole al comando `CREATE TABLE`, el comando `DROP TABLE` condicional para eliminar las tablas. Ejecute el script.

6. *(Inserción manual de datos)* Abra un nuevo script utilizando la funcionalidad
   Scripts → *INSERT Script*
   y complételo para agregar una fila de datos a la tabla `teams`. Guarde el script en un archivo, ejecútelo, y utilice la funcionalidad *View Data* para ver la tabla con los datos insertados.

Nota

En SQL los *strings* se delimitan con comillas simples (''').

7. *(Eliminación de datos)* Abra un nuevo script utilizando la funcionalidad
   Scripts → *DELETE Script*
   y complételo para eliminar la fila insertada. Guarde el script en un archivo, ejecútelo, y utilice la funcionalidad *View Data* para verificar que la tabla esté ahora vacía.

8. *(Carga de datos desde archivos)* El comando `COPY` de *PostgreSQL* es un comando *no estándar* que permite cargar una tabla desde un archivo *.csv* y viceversa. Utilice el comando `COPY` para cargar en cada una de las tablas los datos de los archivos provistos en el Campus para este taller: `team_data.csv` y `matches.csv`. Luego utilice la funcionalidad *View Data* para examinar las tablas.

9. *(SQL dump y exportación de datos)* Exporte cada una de las tablas creadas a un archivo `.csv` con el comando `COPY`. Exporte luego toda la base de datos a un *SQL dump* utilizando el comando `pg_dump` de *PostgreSQL*, y observe la estructura del archivo que se generó.
   Nota: Un *SQL dump* es un script con consultas SQL que permite permite reconstruir la base de datos desde cero. Sirve entre otras cosas como backup de la misma.

Anexo

Para el script de creación de tablas puede guiarse por el siguiente ejemplo base:

```
DROP TABLE IF EXISTS mi_tabla;
CREATE TABLE mi_tabla (
    c1      VARCHAR(10)       NOT NULL,
    c2      INT               NOT NULL,
    c3      NUMERIC(15,12)    NOT NULL,
    c4      INT,
    c5      ...               ...,
    ...     ...               ...
);
```

Y el siguiente es un uso básico del comando `COPY`:

```
COPY mi_tabla
FROM path_archivo
DELIMITER ';'
CSV HEADER          --para indicar que saltee la primera línea
ENCODING 'LATIN1';
```

Le sugerimos también leer las siguientes entradas de la documentación de *PostgreSQL*, y en particular los ejemplos que las mismas incluyen:

- https://www.postgresql.org/docs/current/static/datatype.html (DATATYPE-TABLE)
- https://www.postgresql.org/docs/current/static/sql-createtable.html (CREATE TABLE)
- https://www.postgresql.org/docs/current/static/sql-insert.html (INSERT INTO)
- https://www.postgresql.org/docs/current/static/sql-copy.html (COPY)

Links utiles:

- https://docs.docker.com/get-started/docker_cheatsheet.pdf (docker Cheat sheet)
- https://www.postgresqltutorial.com/postgresql-cheat-sheet/ (psql Cheat sheet)