30231

Escuela de
Ingeniería y Arquitectura
Universidad Zaragoza

BASES DE DATOS 2

Memoria de la práctica 1

Composición del Grupo:
ARIÑO Joaquin
GAJAN Antoine
GARCIA Victor

Coordinator: Carlos Tellería

Tabla de contenido

Introducción

Como parte de la asignatura Bases de datos 2, los estudiantes pondrán en práctica los conocimientos adquiridos previamente sobre el diseño y la administración de bases de datos. Estas competencias les permitirán ser autónomos en los diferentes DBMS presentes en el mercado.

Durante esta primera práctica, los estudiantes deberán descargar desde el entorno que deseen 5 DBMS y realizar comandos básicos desde el terminal para aprender a utilizarlos.

En la memoria se tendrá en cuenta el proceso seguido para lograr el resultado deseado y se señalarán las dificultades y las fuentes utilizadas para remediarlas.

Organización del trabajo y esfuerzos invertidos

★ Organización del Trabajo

Durante la sesión de práctica, cada uno de nosotros instaló y configuró Docker, para luego hacerse cargo de él. Para facilitar la comprensión del tema, decidimos empezar por tratar el primer DBMS (PostgreSQL) juntos. El tratamiento conjunto del primer caso nos permitió compartir nuestros conocimientos y nuestra comprensión de estas nuevas tecnologías.

Esta comprensión compartida nos permitió luego dividir el trabajo de la siguiente manera:
- Víctor: Oracle
- Joaquín: Cassandra, Hbase, Hadoop
- Antoine: IBM DB2, PostgreSQL

Paralelamente a este trabajo, nos hemos repartido las tareas de redacción de la memoria equitativamente. En efecto, cada uno completará en función de las tareas que ha realizado y de los nuevos conocimientos que le han aportado esta práctica.

★ Esfuerzos invertidos

Los miembros del equipo dedicaron el siguiente tiempo a esta primera práctica:

	En clase	En casa (desarrollo de la práctica)	En casa (escritura de la memoria)
Joaquin	3H00	11H00	4H00
Antoine	3H00	9H00	6H00
Victor	3H00	5H00	2H00

★ Configuración del entorno de trabajo (Docker)

En la sesión práctica, empezamos instalando Docker. Para ello, visitamos el sitio web oficial (https://www.docker.com/products/docker-desktop/) y descargamos el archivo. Hemos instalado según la configuración por default Docker Desktop.

★ Instalación y administración de los SGBD

➔ PostgreSQL

El entorno elegido para la instalación de PostgreSQL es Docker.

Primero instalamos PostgreSQL en Docker con la siguiente instrucción y podemos ejecutarlo :

```
$ docker run --name psql_db -e POSTGRES_PASSWORD=mysecretpassword -d postgres
```

Para iniciar sesión en la base de datos:

```
psql -h localhost -U postgres
```

Luego creamos un superusuario de la siguiente manera:

```
CREATE USER name_super_user SUPERUSER;
```

A continuación, podemos conectar con el nuevo superusuario:

```
\c postgres name_super_user;
```

Ahora podemos crear una nueva base de datos cuyo propietario será el superusuario:

```
CREATE DATABASE db_p1;
```

Para conectarse, ejecutamos el siguiente comando:

```
\c db_p1;
```

A continuación, podemos crear varias tablas dentro de esta base de datos. El código se ofrece en el archivo psql_tablas.sql.

Luego podemos crear usuarios y roles. Asignaremos diferentes roles a los usuarios para ver si los accesos definidos son operativos.

```
CREATE USER user1;
CREATE USER user2;
CREATE USER user3;

CREATE ROLE readacces;
```

```
CREATE ROLE writeaccess;

-- Grant access to existing tables
GRANT USAGE ON SCHEMA public TO readaccess;
GRANT SELECT ON ALL TABLES IN SCHEMA public TO readaccess;

GRANT USAGE ON SCHEMA public TO writeaccess;
GRANT SELECT, INSERT ON ALL TABLES IN SCHEMA public TO writeaccess;


-- Grant access to users
GRANT readaccess TO user1;
GRANT writeaccess TO user2;

-- For user3 (mixt of read and write)
GRANT SELECT, INSERT ON Persons IN SCHEMA public TO user3;
GRANT SELECT ON Animales IN SCHEMA public TO user3;
REVOKE ALL PRIVILEGES ON Coches IN SCHEMA public TO user3;
```

Por último, podemos permitir una conexión a la base de datos a todos los usuarios de la red local. Para ello, utilizando tutoriales en Internet, hemos utilizado el siguiente método:

➔ Cassandra

El entorno elegido para la instalación de Cassandra es Docker.

Para realizar la instalación, desde la terminal se descarga y crea un contenedor con los siguientes comandos:

```
docker pull cassandra
docker run –name test_cassandra -d cassandra
```

Para conectarse al contenedor que se ha creado:

```
docker exec -it test_cassandra cqlsh
```

Para realizar distintas pruebas y así comprobar el funcionamiento de este SGBD, hay que crear un keyspace (“es como llama cassandra a las bd”):

```
CREATE KEYSPACE universidad WITH REPLICATION = {'class' : 'SimpleStrategy', 'replication_factor' : 1};
```

Crea la bd universidad, con un factor de replicación de 1.

Para crear un superusuario, se debe iniciar sesión con las credenciales de superusuario predeterminadas de casandra, ya que este tipo de usuario es el único capaz de crear otro super usuario.

Después se crea un nuevo usuario con el atributo SUPERUSER, y se le conceden todos los permisos:

```
docker exec -it test_cassandra cqlsh -u cassandra -p cassandra

CREATE ROLE administrador WITH PASSWORD = 'admin' AND SUPERUSER = true;
```

Para crear un usuario con solo permisos de lectura:

```
CREATE USER 'usuario' WITH PASSWORD 'passwd';

CREATE ROLE 'readonly' WITH LOGIN = false AND SELECT ON ALL KEYSPACES;

GRANT 'readonly' TO 'usuario' ;
```

O por ejemplo para crear un rol con permisos de inserción pero no de eliminación:

```
CREATE ROLE usuario2 WITH PASSWORD = '<passwd>' AND LOGIN = true;

GRANT INSERT ON TABLE ejemplo_tabla TO usuario2 ;

REVOKE DROP ON TABLE ejemplo_tabla FROM usuario2 ;
```

El siguiente script de cql de ejemplo, formato usado por Cassandra, (disponible en los archivos entregados), crea una tabla, le inserta datos y los consulta.

```
USE universidad;

-- Crear una tabla
CREATE TABLE estudiante (
  id int PRIMARY KEY,
  nombre text,
  edad int
);

-- Insertar datos
INSERT INTO estudiante (id, nombre, edad) VALUES (1, 'Antoine', 21);
```

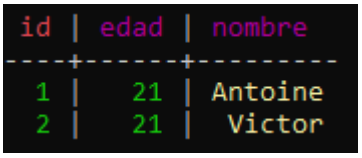
```
INSERT INTO estudiante (id, nombre, edad) VALUES (2, 'Victor', 21);

-- Consultar datos
SELECT * FROM estudiante;
```

Para ejecutar el script:

```
SOURCE '/test_cassandra.cql';
```

El resultado sería el siguiente:



Conectarse desde otra máquina en la misma red si es posible siempre y cuando los puertos del contenedor de docker estén abiertos. Para comprobarlo introducir el siguiente comando, que nos listará todos los contenedores instalados, y entre otra información, los puertos abiertos.

```
docker ps
```

Para conectarte introducir el siguiente comando:

```
cqlsh <IP> <PUERTO>
```

Tras la instalación y pruebas de Cassandra, podemos concluir que se trata de un gestor de bases de datos con una instalación y uso muy simple e intuitivo, a diferencia de otros gestores probados durante esta práctica.

Además, su lenguaje de consulta propio es sencillo de aprender.

Eso sí, si se requiere de una sistema gestor complejo, como por ejemplo que no admite consultas ACID, Cassandra no es la mejor opción.

➔ Apache HBase (ecosistema Hadoop)

Para la instalación de HBase y Hadoop, se ha usado como entorno una máquina virtual (Oracle Virtualbox), más concretamente Ubuntu 20.04. Está distribución ha sido elegida ya que los miembros del grupo la habían usado en otras ocasiones y por tanto, tenían familiaridad con su uso.

Para realizar la instalación, se han seguido mayoritariamente los pasos indicados en Moodle, pero además ha habido que realizar algún otro paso por problemas que han surgido durante la instalación. Estos son los pasos distintos que hemos tenido que realizar:

1. Al intentar ejecutar start-hbase.sh, mostraba un error que indicaba que era necesaria la instalación de Java JDK mayor a 1.7. Java ya había sido instalado al principio de los pasos, y además la ruta de JAVA_HOME había sido modificada en el archivo hadoop-env.sh.

   Tras varios intentos fallidos, se ha solucionado añadiendo la ruta de java al archivo /etc/environment .

2. Al ejecutar start-hbase también daba otro error que indicaba que no tenía permiso para crear directorios en /hbase.

   La solución ha sido dar permisos para la creación de estos directorios con:

   ```
   sudo chmod -R 777 usr/local/hbase
   ```

   (primero se ejecutó sin -R, pero esto no lo soluciona ya que la R significa recursiva, y afecta tanto al directorio como subdirectorios)

Para iniciar la shell de hbase (para salir usar el comando exit):

```
hbase shell
```

Comandos para:

```
#Crear una tabla con dos columnas
create 'Estudiante', 'PID', 'Nombre'
#Insertar datos
put 'Estudiante', '1', 'Antoine'
put 'Estudiante', '2', 'Victor'
#Obtener datos de una fila
get 'Estudiante', '2'
#Eliminar datos
delete 'Estudiante', '2', 'Victor'
#Listar todas las tablas de Hbase
list
```

Para crear usuarios con distintos roles se debe usan los siguientes comandos:

```
hbase> grant <user> <permissions> [ @<namespace> [ <table>[ <column
```

```
family>[ <column qualifier> ] ] ] ]

hbase> revoke <user> [ @<namespace> [ <table> [ <column family> [
<column qualifier> ] ] ]

#Por ejemplo:
grant 'Victor', 'put', 'Estudiante'
grant 'Antoine', 'scan', 'Estudiante'
```

Tras la realización de instalación y pruebas, se puede concluir que se trata de una base de datos que se puede escalar fácilmente y manejar grandes conjuntos de datos con velocidad, especialmente comparado con otras BBDD analizadas como Cassandra, ya que es su principal cometido.

Aunque no se trata de una base de datos SQL, sus instrucciones son sencillas de aprender, eso sí, puede quedarse corta si se desea hacer consultas más complejas que inserciones, modificaciones y eliminaciones.

➔ IBM DB2

El entorno elegido para la instalación de IBM DB2 es Docker desde una máquina virtual Linux Ubuntu.

Para realizar la instalación, desde la terminal se descarga y crea un contenedor con los siguientes comandos:

```
mkdir ~/database

docker run -itd --name mydb2 --privileged=true -p 50000:50000 -e
LICENSE=accept -e DB2INST1_PASSWORD=my_password -e DBNAME=testdb -v
~/database:/database ibmcom/db2
```

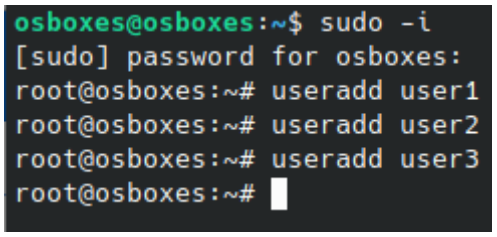
Para conectarse al contenedor que se ha creado:

```
docker exec -ti mydb2 bash -c "su - db2inst1"
```

Para conectarse a la base de datos :

```
db2
connect to testdb
```

En otro terminal, vamos a crear 3 usuarios. Para ello, nos ponemos en modo root con sudo -i y luego creamos los 3 usuarios.



En primer lugar, se permite a los usuarios conectarse a la base de datos creada.

```
grant connect to database to user user1
grant connect to database to user user2
grant connect to database to user user3
```

Se define al usuario 1 como un superusuario :

```
grant dbadm to database to user user1
```

Comenzamos creando la estructura de la base de datos mediante la creación de un esquema de la siguiente manera:

```
CREATE SCHEMA schema_test AUTHORIZATION user1
```

Luego creamos las tablas con la sintaxis SQL:

```
CREATE TABLE STORE ( id INTEGER NOT NULL, name VARCHAR(30) NOT NULL);
INSERT INTO STORE VALUES (1, 'Alcampo');
INSERT INTO STORE VALUES (2, 'Carrefour');
```

Podemos crear roles para definir los permisos de los usuarios:

```
-- Create roles
CREATE ROLE readaccess;
CREATE ROLE writeaccess;

-- Define roles and privileges to each role
GRANT SELECT ON STORE TO ROLE readaccess;
REVOKE INSERT, UPDATE ON STORE FROM ROLE readaccess;
GRANT SELECT, INSERT, UPDATE ON STORE TO ROLE writeaccess;
```

```
-- Give roles and privileges to users
GRANT ROLE readaccess TO USER user1;
GRANT ROLE writeaccess TO USER user2;
REVOKE ALL PRIVILEGES FROM USER user3;
```

El código anterior se encuentra en el anexo del informe “db2.sql”. Para ejecutarlo es necesario:

```
db2 -tvmf db2.sql |tee db2.sql.out
```

Con los usuarios creados, usted podrá con cada uno de ellos leer y escribir, si sus derechos lo permiten, datos de la base de datos.

Es posible conectarse desde un equipo de la red local ejecutando la siguiente serie de comandos:

```
curl -u admin:password -k -O
https://sds801va1_hostname_or_IPaddress/custfile_mgmt/download?idstools/idscfgremotedb.zip # Download remote connexion package

unzip idscfgremotedb.zip # Unzip package

jar -xvf idscfgremotedb.zip

groupadd dbsysadm # Create admin group
usermod -G root,dbsysadm root # Give admin privileges

useradd -d /home/db2inst1 -g dbsysadm -G dbsysadm -m -s /usr/bin/ksh db2inst1

passwd db2inst1 # Define password of user

./idscfgremotedb -c -u db2inst1 -w passwd -p /opt/ibm/db2/V10.5 -s 6512 -t ldapdb -l
/home/db2inst1 # Allows remote connexion

./idscfgremotedb -c -u db2inst1 -w passwd -p /opt/ibm/db2/V10.5 -s 6512 -t ldapclog #
Create remote database
```

Por último, podrá probar con consultas SQL si los diferentes usuarios tienen acceso a las tablas para las que se les han concedido derechos.

➔ ORACLE

El entorno elegido para la instalación de IBM DB2 es Docker desde una máquina virtual Linux Ubuntu.

Para realizar la instalación, primero se descarga la imagen desde la página oficial de oracle, junto con las carpetas del siguiente github para la instalación:
https://github.com/oracle/docker-images/tree/main/OracleDatabase

Ejecutamos el código BuildContainerImage.sh del código de github con la versión correspondiente y la imagen aparecerá en docker.

Posteriormente ejecutamos run de la imagen para crear el contenedor desde el propio docker o con el comando: $ docker run -d –name

Desde terminal se pueden modificar parámetros, por ejemplo:
docker run -d --name oracle19 -p 1521:1521 -p 5500:5500 -e ORACLE_PWD=pass oracle/database:19.3.0-se2

Podemos conectar a la base de datos desde terminal con los comandos:
$ docker exec -it dbname sqlplus / as sysdba
$ docker exec -it dbname sqlplus sys/cdb-user-password@cdb-sid as sysdba
$ docker exec -it dbname sqlplus system/cdb-user-password@cdb-sid
$ docker exec -it dbname sqlplus pdbadmin/pdb-user-password@pdbname

crear usuario:

```
CREATE USER miusuario
  IDENTIFIED GLOBALLY AS 'CN=analyst, OU=division1, O=oracle, C=US'
  DEFAULT TABLESPACE example
  QUOTA 5M ON example;
```

En primer lugar, se permite a los usuarios conectarse a la base de datos creada:

$ GRANT CONNECT TO miusuario;
$ GRANT CREATE SESSION GRANT ANY PRIVILEGE TO books_admin;
$ GRANT UNLIMITED TABLESPACE TO books_admin;

Dar permisos a un usuario sobre una tabla.
GRANT SELECT, INSERT, UPDATE, DELETE ON persons TO miusuario;

Dar permisos de lectura y escritura;

grant read on orders to miusuario;

grant writeon orders to miusuario;

Se define al usuario 1 como un superusuario :
grant SYSDBA to miusuario;

Para crear una tabla usamos la sintaxis de oracle en sqlplus

```
CREATE TABLE ot.persons(
   person_id NUMBER GENERATED BY DEFAULT AS IDENTITY,
   first_name VARCHAR2(50) NOT NULL,
   last_name VARCHAR2(50) NOT NULL,
   PRIMARY KEY(person_id)
);
```

Conectarse desde otra máquina en la misma red si es posible siempre y cuando los puertos del contenedor de docker estén abiertos.

Para conectarse desde fuera del contenedor usando SQL*Plus, use uno de los siguientes comandos, donde dbname es el nombre de la base de datos, cdb-user-password es la contraseña para un usuario de la base de datos con privilegios de sistema SYSDBA o SYSTEM en el CDB, cdb-sid es el identificador del sistema del CDB, pdb-password es un usuario con los privilegios del sistema PDBADMIN y pdbname es el nombre del PDB al que se está conectando. En este ejemplo, el puerto Docker expuesto para la base de datos es 1521.

```
$ sqlplus sys/cdb-user-password@//localhost:1521/cdb-sid as sysdba
$ sqlplus system/cdb-user-password@//localhost:1521/cdb-sid
```

★ Comentarios acerca de las licencias

Las diferentes licencias de DBMS permiten realizar tareas similares de manera más o menos compleja. Mientras que PostgreSQL parece ser el más intuitivo para crear usuarios, roles y grupos, otros DBMS nos parecen más fáciles de usar cuando se trata de establecer una conexión remota o ejecutar scripts SQL.

También hemos notado que PostgreSQL parece el más fácil de usar debido a la importante documentación en internet. De hecho, fue más fácil encontrar recursos en PostgreSQL que en Cassandra.

★ Generación de datos y pruebas

En los pasos anteriores, nos tomamos el tiempo para generar datos de prueba y verificar la validez de las pruebas realizadas.

Descubrimientos, dificultades encontradas y lecciones aprendidas

★ Descubrimientos

Esta práctica nos ha permitido descubrir los diferentes DBMS y sus diferencias. En efecto, no todos movilizan los mismos conceptos. Algunos diferencian entre usuarios, roles y grupos, mientras que otros no. Algunos DBMS son más complejos de manejar que otros y menos intuitivos, como DB2.

Durante esta práctica, descubrimos las diferentes maneras de establecer una conexión remota en una red local. Hemos observado que no todos los DBMS ofrece esta herramienta, o al menos no son tan restrictivos. De hecho, en algunos DBMS es difícil establecer una conexión solo a la red local.

★ Dificultades encontradas

➔ Crear superusuarios en IBM DB2

Para realizar el trabajo solicitado con IBM DB2, primero utilizamos el entorno Docker en Windows 11. Sin embargo, después de buscar en Internet, nos dimos cuenta de que alcanzar los objetivos solicitados sería complicado. De hecho, mientras que otros DBMS permite crear usuarios dentro de la base de datos sin crearlos en la máquina del usuario, no podemos crear un nuevo superusuario aquí. Con IBM DB2, debe configurarse en el sistema operativo del host.

Por esta razón, decidimos cambiar a una máquina virtual de Ubuntu Linux, que nos permitirá crear nuevos usuarios más fácilmente.

➔ Instalación Máquina Virtual para Hbase y Hadopop

Para la instalación de hbase y hadoop, se ha usado como entorno una máquina virtual (Oracle Virtual Box). Esta ha dado varios problemas ya que nada más completar la instalación, e iniciar la máquina, primero la terminal no se abría. Una vez solucionado este problema que aparentemente es común, al entrar en la terminal e intentar ejecutar comandos como sudo, daba el siguiente error: “Username is not in the sudoers file”.

Tras solucionar este problema también, se siguieron los pasos para instalar hbase y hadoop indicados en moodle, pero constantemente se tenía que resolver problemas de permisos. Al terminar la instalación, no se podía ejecutar el comando start-hbase.sh, ya que se nos denegaba el permiso de ejecutarlo.

Tras intentar solucionar de múltiples maneras, se instaló la máquina virtual de nuevo, esta vez haciendo la instalación completamente manual, sin la ayuda de Oracle VirtualBox que permite hacer la instalación más rápida.

➔ Acceder a sqlplus

Al intentar acceder a sqlplus daba el error: ORA-12162: TNS:net service name is incorrectly specified

Su solución consistía en especificar la variable ORACLE_SID y realizar export de esta.

Tutoriales utilizados

Dado que el tema es relativamente abierto y requiere nociones de sintaxis propias del DBMS, hemos recurrido a numerosos tutoriales y videos de explicación de Youtube.

Utilizamos :

AB Tutorials: IBM DB2 in a Docker Container. www.youtube.com, https://www.youtube.com/watch?v=qo3t6PqUado. (24/02/2023)

Connecting to a Remote PostgreSQL Database - NetIQ Identity Manager Setup Guide for Windows. https://www.netiq.com/documentation/identity-manager-47/setup_windows/data/connecting-to-a-remote-postgresql-database.html. (24/02/2023)

Conclusión

Así, esta primera práctica fue para nosotros la oportunidad de tomar el control del universo Docker y familiarizarnos con diferentes DBMS. Hemos podido aprovechar los conocimientos teóricos aportados por los cursos magistrales sobre los temas de administración de bases de datos.

Además, también hemos hecho uso de máquinas virtuales de Linux, y por tanto, hemos aprendido no solo a instalarlas, sino a configurarlas correctamente.

Hemos puesto en práctica la creación de usuarios, de roles, el diseño de esquemas. Esta práctica fue para nosotros la oportunidad de entender cómo permitir el acceso de la red local a una base de datos.

Estos conocimientos nos serán útiles en la continuación de nuestro recorrido universitario y podrán movilizarse en las prácticas en empresa.