Ejercicio de Laboratorio 3: Hive y Beeline

Instituto Politécnico Nacional.
Escuela Superior de Cómputo.
Licenciatura en ciencia de datos.
Bases de Datos Avanzadas

Emiliano López Méndez.

Introduccion

En este ejercicio de laboratorio, se utiliza Apache Hive para crear, poblar y consultar tablas a través del cliente Beeline, ejecutándose en un entorno Docker. Hive es una herramienta clave en el ecosistema de Big Data que permite gestionar grandes volúmenes de datos estructurados y semiestructurados, permitiendo realizar consultas SQL sobre estos datos almacenados en HDFS o en sistemas de archivos locales. A través de este ejercicio, se exploran las funcionalidades básicas de Hive, incluyendo la creación de tablas, la inserción de datos y la realización de consultas relacionales.

Tecnologías empleadas:

- **Apache Hive**: Es un sistema de data warehouse que permite la consulta y gestión de grandes conjuntos de datos almacenados en sistemas distribuidos como HDFS.

- **Beeline**: Es un cliente de línea de comandos utilizado para conectarse a Hive a través de JDBC. Beeline permite ejecutar consultas SQL de forma interactiva y es ampliamente utilizado para acceder a metadatos y datos almacenados en tablas de Hive.

- **Docker**: Se utiliza para crear un entorno aislado y reproducible en el que se despliega Apache Hive y Beeline.

- **PostgreSQL/MySQL** (opcional para Metastore): En muchos casos, Hive utiliza una base de datos como PostgreSQL o MySQL para almacenar los metadatos de las tablas (Metastore).

Desarrollo de la Actividad:

Creando una base de datos con Apache Hive

Empezamos descargando una imagen con este comando

```
docker pull apache/hive:4.0.0-alpha-2
```

```
PS C:\Users\HP\Documents\4to Semestre> docker pull apache/hive:4.0.0-alpha-2
4.0.0-alpha-2: Pulling from apache/hive
1efc276f4ff9: Pull complete
a2f2f93da482: Pull complete
1a2de4cc9431: Pull complete
d2421c7a4bbf: Pull complete
cd30f3d1e783: Downloading   89.3MB/562.2MB
56a130901b37: Downloading  59.92MB/304.4MB
9d948bbdc842: Download complete
60930a9c0895: Download complete
eddb431f2fca: Download complete
239b6597911f: Download complete
b237b9d3d33a: Download complete
ee7a22f107a6: Waiting
4f4fb700ef54: Waiting
```

```
ee7a22f107a6: Pull complete
4f4fb700ef54: Pull complete
Digest: sha256:69e482fdcebb9e07610943b610baea996c941bb36814cf233769b8a4db41f9c1
Status: Downloaded newer image for apache/hive:4.0.0-alpha-2
docker.io/apache/hive:4.0.0-alpha-2

What's next:
    View a summary of image vulnerabilities and recommendations → docker scout quickview apache/hive:4
.0.0-alpha-2
PS C:\Users\HP\Documents\4to Semestre>
```

Una vez ya hecha sacamos nuestra versión de hive y lo configuramos para estar embudada con Metastore

```
See 'docker run --help'.
PS C:\Users\HP\Documents\4to Semestre> $env:HIVE_VERSION = "4.0.0-alpha-2"
PS C:\Users\HP\Documents\4to Semestre> docker run -d -p 10000:10000 -p 10002:10002 --env
SERVICE_NAME=hiveserver2 --name hive4 apache/hive:${HIVE_VERSION}
docker: invalid reference format.
See 'docker run --help'.
PS C:\Users\HP\Documents\4to Semestre> docker run -d -p 10000:10000 -p 10002:10002 --env
SERVICE_NAME=hiveserver2 --name hive4 apache/hive:$env:HIVE_VERSION
daa841c91cec3d86ef4b5177a8c6adc764e2e24098cc49a63f581c1f6047af42
PS C:\Users\HP\Documents\4to Semestre>
```

Y nos conectamos a Beeline

```
See 'docker run --help'.
PS C:\Users\HP\Documents\4to Semestre> docker exec -it hive4 beeline -u 'jdbc:hive2://localhost:10000/
'
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/hive/lib/log4j-slf4j-impl-2.18.0.jar!/org/slf4j/impl/StaticLogg
erBinder.class]
SLF4J: Found binding in [jar:file:/opt/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.30.jar!/org/sl
f4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/hive/lib/log4j-slf4j-impl-2.18.0.jar!/org/slf4j/impl/StaticLogg
erBinder.class]
SLF4J: Found binding in [jar:file:/opt/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.30.jar!/org/sl
f4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Connecting to jdbc:hive2://localhost:10000/
Connected to: Apache Hive (version 4.0.0-alpha-2)
Driver: Hive JDBC (version 4.0.0-alpha-2)
Transaction isolation: TRANSACTION_REPEATABLE_READ
Beeline version 4.0.0-alpha-2 by Apache Hive
0: jdbc:hive2://localhost:10000/>
```

Luego vamos a crear una tabla como ejemplo

```
Beeline version 4.0.0-alpha-2 by Apache Hive
0: jdbc:hive2://localhost:10000> show tables;
INFO  : Compiling command(queryId=hive_20241013040925_c890c87d-d22d-40fd-aad5-ae788ac9fb9f): show tabl
es
INFO  : Semantic Analysis Completed (retrial = false)
INFO  : Created Hive schema: Schema(fieldSchemas:[FieldSchema(name:tab_name, type:string, comment:from
 deserializer)], properties:null)
INFO  : Completed compiling command(queryId=hive_20241013040925_c890c87d-d22d-40fd-aad5-ae788ac9fb9f);
 Time taken: 1.671 seconds
INFO  : Concurrency mode is disabled, not creating a lock manager
INFO  : Executing command(queryId=hive_20241013040925_c890c87d-d22d-40fd-aad5-ae788ac9fb9f): show tabl
es
INFO  : Starting task [Stage-0:DDL] in serial mode
INFO  : Completed executing command(queryId=hive_20241013040925_c890c87d-d22d-40fd-aad5-ae788ac9fb9f);
 Time taken: 0.058 seconds
+------------+
| tab_name   |
+------------+
+------------+
```

Como podemos ver no tenemos ninguna tabla por el momento

Con este comando creamos la tabla hive_example

```
No rows selected (2.093 seconds)
0: jdbc:hive2://localhost:10000> create table hive_example(a string, b int) partitioned by(c int);
```

```
show tables;
create table hive_example(a string, b int) partitioned by(c int);
alter table hive_example add partition(c=1);
insert into hive_example partition(c=1) values('a', 1), ('a', 2),('b',3);
select count(distinct a) from hive_example;
select sum(b) from hive_example;
```

Y con los próximos comandos vamos a alterar la tabla e insertar datos para demostrar que la tabla este poblada

```
INFO  : Status: Running (Executing on YARN cluster with App id application_1728792617810_0001)

INFO  : Completed executing command(queryId=hive_20241013041122_3b8d8bc9-ca9f-40d4-ac60-125edaaf4e44);
 Time taken: 0.993 seconds
+------+
| _c0  |
+------+
| 2    |
+------+
```

```
INFO  : Status: Running (Executing on YARN cluster with App id application_1728792617810_0001)

INFO  : Completed executing command(queryId=hive_20241013041218_cc30b849-c1a1-44ce-8562-b71eaf999ce5);
 Time taken: 0.534 seconds
+------+
| _c0  |
+------+
| 6    |
+------+
```

Como podemos ver ya creamos la tabla hive_example

```
1 row selected (0.045 seconds)
0: jdbc:hive2://localhost:10000> show tables;
INFO  : Compiling command(queryId=hive_20241013041231_29d9d57f-6bb3-4c04-aee3-44e540abd5f1): show tabl
es
INFO  : Semantic Analysis Completed (retrial = false)
INFO  : Created Hive schema: Schema(fieldSchemas:[FieldSchema(name:tab_name, type:string, comment:from
 deserializer)], properties:null)
INFO  : Completed compiling command(queryId=hive_20241013041231_29d9d57f-6bb3-4c04-aee3-44e540abd5f1);
 Time taken: 0.004 seconds
INFO  : Concurrency mode is disabled, not creating a lock manager
INFO  : Executing command(queryId=hive_20241013041231_29d9d57f-6bb3-4c04-aee3-44e540abd5f1): show tabl
es
INFO  : Starting task [Stage-0:DDL] in serial mode
INFO  : Completed executing command(queryId=hive_20241013041231_29d9d57f-6bb3-4c04-aee3-44e540abd5f1);
 Time taken: 0.048 seconds
+---------------+
|   tab_name    |
+---------------+
| hive_example  |
+---------------+
1 row selected (0.069 seconds)
0: jdbc:hive2://localhost:10000>
```

```
7 rows selected (0.04 seconds)
0: jdbc:hive2://localhost:10000> describe hive_example;
INFO  : Compiling command(queryId=hive_20241013041358_0e886322-9216-469e-b6ba-d5800a1d8c5d): describe hive_example
INFO  : Semantic Analysis Completed (retrial = false)
INFO  : Created Hive schema: Schema(fieldSchemas:[FieldSchema(name:col_name, type:string, comment:from deserializer), FieldSchema(na
me:data_type, type:string, comment:from deserializer), FieldSchema(name:comment, type:string, comment:from deserializer)], propertie
s:null)
INFO  : Completed compiling command(queryId=hive_20241013041358_0e886322-9216-469e-b6ba-d5800a1d8c5d); Time taken: 0.012 seconds
INFO  : Concurrency mode is disabled, not creating a lock manager
INFO  : Executing command(queryId=hive_20241013041358_0e886322-9216-469e-b6ba-d5800a1d8c5d): describe hive_example
INFO  : Starting task [Stage-0:DDL] in serial mode
INFO  : Completed executing command(queryId=hive_20241013041358_0e886322-9216-469e-b6ba-d5800a1d8c5d); Time taken: 0.011 seconds
+--------------------------+------------+----------+
|         col_name         | data_type  | comment  |
+--------------------------+------------+----------+
| a                        | string     |          |
| b                        | int        |          |
| c                        | int        |          |
|                          | NULL       | NULL     |
| # Partition Information  | NULL       | NULL     |
| # col_name               | data_type  | comment  |
| c                        | int        |          |
+--------------------------+------------+----------+
7 rows selected (0.037 seconds)
0: jdbc:hive2://localhost:10000>
```

Vemos al tabla poblada

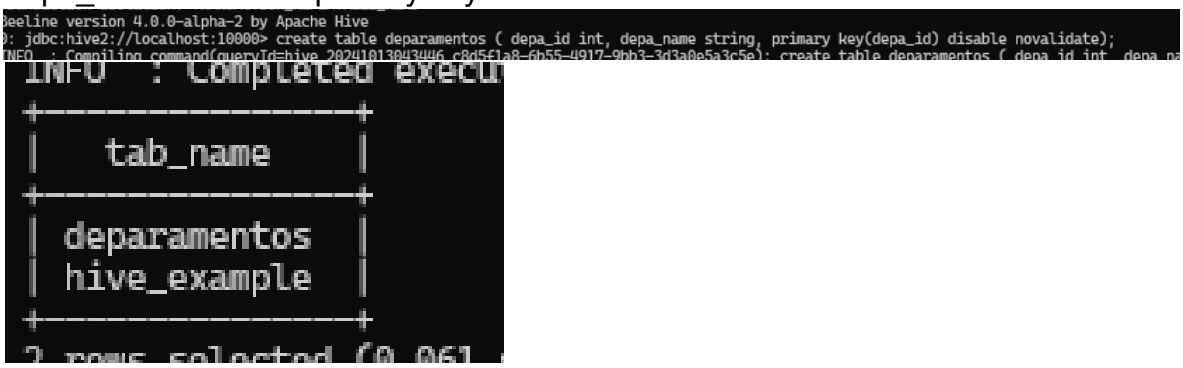
¿Es posible crear una tabla a pesar de que no se definió una base de datos como tal?

Sí, es posible crear una tabla en Apache Hive sin haber definido explícitamente una base de datos, porque Hive utiliza una base de datos predeterminada llamada **default** si no se especifica otra. De hecho, cuando no defines una base de datos con un comando USE, cualquier tabla que crees será almacenada dentro de esta base de datos predeterminada.

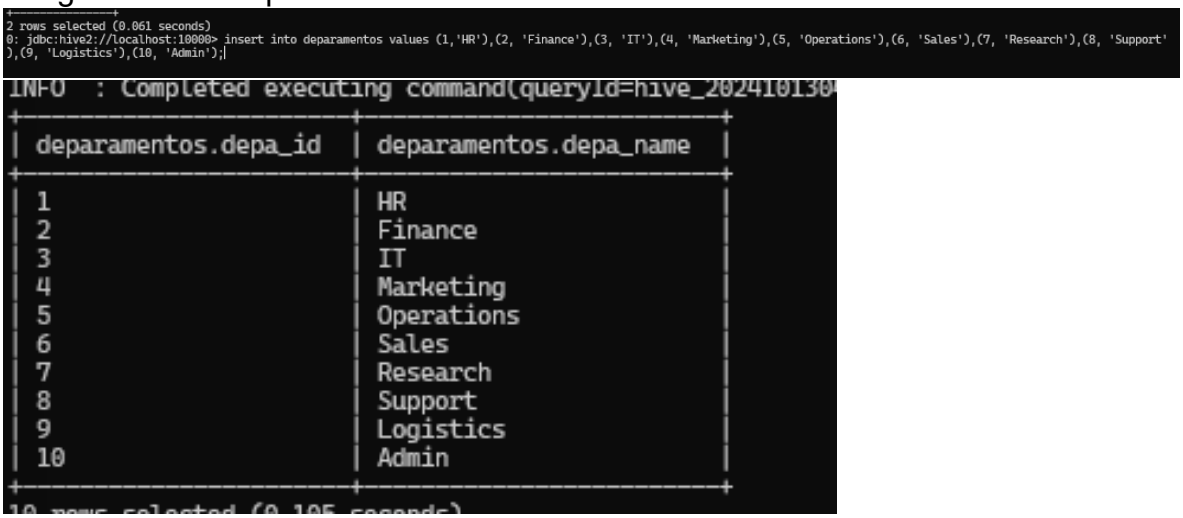
Parte 2 Creando dos tablas pobladas con relacion

Creando primera tabla departamentos

Empezamos creando una tabla de departamentos depa_id y depa_name en donde depa_id será nuestra primary key.



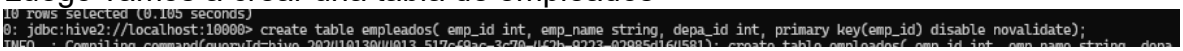
Luego la vamos a poblar:



Aquí vemos que si se poblo la tabla de deparamentos.

Creando segunda tabla empleados

Luego vamos a crear una tabla de empleados



```
INFO  : Completed exe
+---------------+
|   tab_name    |
+---------------+
| deparamentos  |
| empleados     |
| hive_example  |
+---------------+
```

Y poblamos la tabla

```
INFO  : Completed executing command(queryId=hive_20241013044439_dedb0a
+-------------------+---------------------+--------------------+
| empleados.emp_id  | empleados.emp_name  | empleados.depa_id  |
+-------------------+---------------------+--------------------+
| 101               | Alice               | 1                  |
| 102               | Bob                 | 2                  |
| 103               | Charlie             | 3                  |
| 104               | David               | 4                  |
| 105               | Eve                 | 5                  |
| 106               | Frank               | 6                  |
| 107               | Grace               | 7                  |
| 108               | Hank                | 8                  |
| 109               | Ivy                 | 9                  |
| 110               | Jack                | 10                 |
+-------------------+---------------------+--------------------+
10 rows selected (0.075 seconds)
0: jdbc:hive2://localhost:10000>
```

A unir las dos tablas en base a sus llaves primarias y foráneas

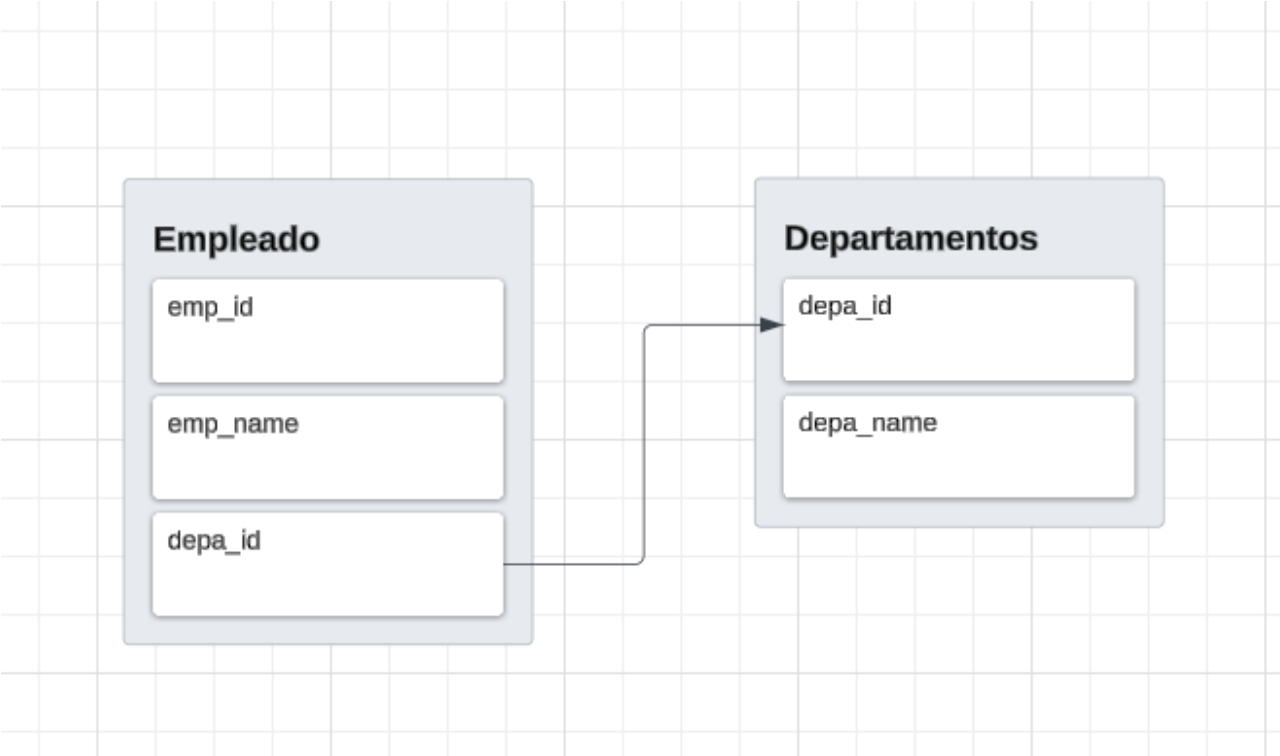
Mediante esta consulta de join

```
10 rows selected (0.075 seconds)
0: jdbc:hive2://localhost:10000> select e.emp_id, e.emp_name, d.depa_name from empleados e join deparamentos d on e.depa_id = d.depa_id;
INFO  : Compiling command(queryId=hive_20241013044742_c1570a87-5703-4f8c-8bbd-bff60007cd5b): select e.emp_id, e.emp_name, d.depa_name from emplea
```

Lograrmeos unir las dos tablas

```
INFO  : Completed executing command(queryId=hive_2024
+-----------+--------------+----------------+
| e.emp_id  | e.emp_name   | d.depa_name    |
+-----------+--------------+----------------+
| 101       | Alice        | HR             |
| 102       | Bob          | Finance        |
| 103       | Charlie      | IT             |
| 104       | David        | Marketing      |
| 105       | Eve          | Operations     |
| 106       | Frank        | Sales          |
| 107       | Grace        | Research       |
| 108       | Hank         | Support        |
| 109       | Ivy          | Logistics      |
| 110       | Jack         | Admin          |
+-----------+--------------+----------------+
```

Esquema relacional de las tablas creadas



Parte 3 Investigacion

Como se le podría dar persistencia al servidor?

- **Mediante la persistencia de volúmenes en Docker**: Para asegurar que los datos y configuraciones del servidor persistan incluso después de reiniciar o detener el contenedor Docker, es necesario utilizar **volúmenes de Docker**. Los volúmenes permiten almacenar los datos en el sistema de archivos del host, lo que garantiza que no se pierdan cuando el contenedor se reinicie o elimine.
- **Metastore**: Si el servidor utiliza una base de datos (como PostgreSQL o MySQL) para almacenar los metadatos de las tablas, es importante montar un volumen para persistir los datos de la base de datos.
- **Archivos de datos**: En caso de que el servidor maneje archivos de datos localmente (por ejemplo, tablas en formato de archivos), es fundamental crear un volumen para el directorio donde se almacenan estos datos.
- **Uso de HDFS** (si aplica): En entornos distribuidos como Hadoop, el sistema de archivos HDFS también puede beneficiarse de volúmenes Docker para mantener persistencia en los nodos de NameNode y DataNode.

Conclusión

En este ejercicio de laboratorio, se utiliza Apache Hive para crear, poblar y consultar tablas a través del cliente Beeline, ejecutándose en un entorno Docker. Hive es una herramienta clave en el ecosistema de Big Data que permite gestionar grandes volúmenes de datos estructurados y semiestructurados, permitiendo realizar consultas SQL sobre estos datos almacenados en HDFS o en sistemas de archivos locales. A través de este ejercicio, se exploran las funcionalidades básicas de Hive, incluyendo la creación de tablas, la inserción de datos y la realización de consultas relacionales.