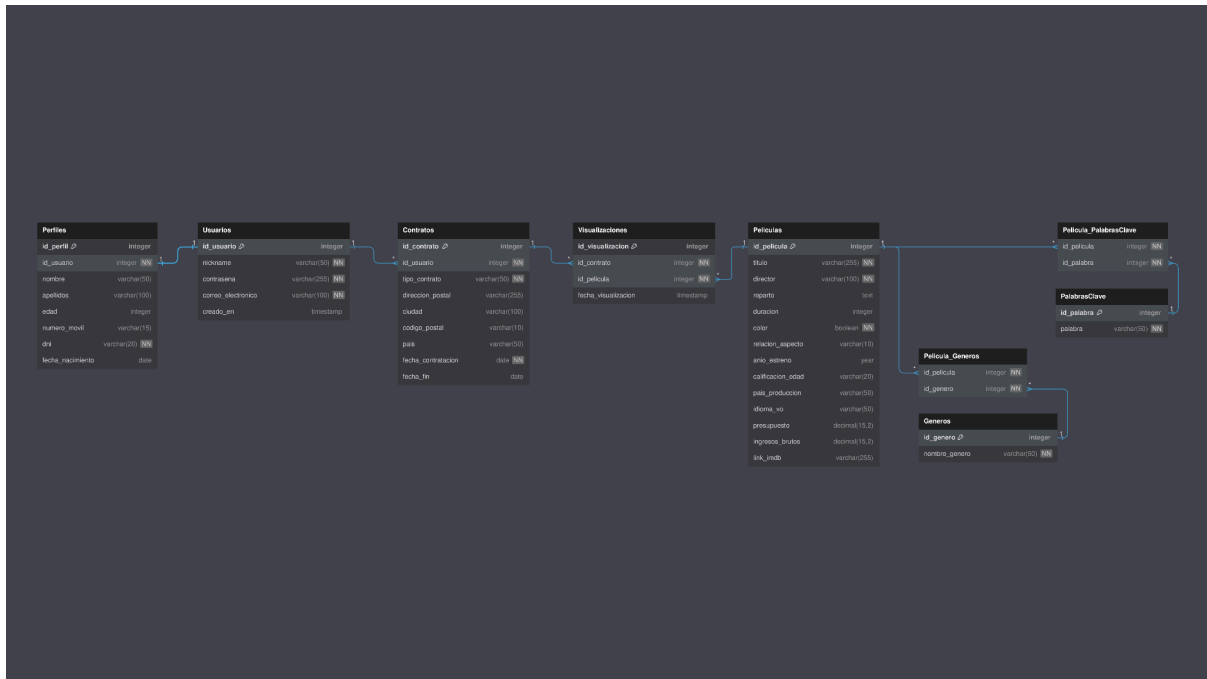


## Memoria del Proyecto: Importación y Exportación de Datos con MariaDB y Apache Sqoop



**Autores:** Pablo Falcón, Miguel Santiesteban, Brian Ramírez

# 1. Grafo Relacional y Decisiones de Diseño



El sistema MovieBind está compuesto por las siguientes tablas y relaciones:

## Tablas Principales

- **usuarios**: Almacena la información básica de los usuarios registrados.
  - Atributos:
    - id\_usuario (INT, PK, autoincrement).
    - nickname (VARCHAR(50), único, no nulo).
    - contrasena (VARCHAR(255), no nulo).
    - correo\_electronico (VARCHAR(100), único, no nulo).
  - Relaciones:
    - Relación uno a uno con perfiles.
    - Relación uno a muchos con contratos.
- **perfiles**: Almacena detalles adicionales del perfil de los usuarios.
  - Atributos:
    - id\_perfil (INT, PK, autoincrement).
    - id\_usuario (INT, FK -> usuarios.id\_usuario).
    - nombre, apellidos, edad, numero\_movil, dni (único y obligatorio), fecha\_nacimiento.
  - Restricciones:
    - dni es obligatorio y único.

- **contratos:** Registra los contratos asociados a los usuarios.
  - Atributos:
    - id\_contrato (INT, PK, autoincrement).
    - id\_usuario (INT, FK -> usuarios.id\_usuario).
    - tipo\_contrato, direccion\_postal, ciudad, codigo\_postal, pais, fecha\_contrato, fecha\_fin\_contrato.
  - Relaciones:
    - Relación uno a muchos con visualizaciones.
  
- **peliculas:** Almacena información sobre las películas.
  - Atributos:
    - id\_pelicula (INT, PK, autoincrement).
    - titulo, director, reparto, duracion, color (Boolean), relacion\_aspecto, anio\_estreno, calificacion\_edad, pais\_produccion, idioma\_vo, presupuesto, ingresos\_brutos, link\_imdb.
  - Relaciones:
    - Relación muchos a muchos con generos mediante pelicula\_generos.
    - Relación muchos a muchos con palabrasclave mediante pelicula\_palabrasclave.
    - Relación uno a muchos con visualizaciones.
  
- **visualizaciones:** Registra las visualizaciones realizadas por usuarios.
  - Atributos:
    - id\_visualizacion (INT, PK, autoincrement).
    - id\_contrato (INT, FK -> contratos.id\_contrato).
    - id\_pelicula (INT, FK -> peliculas.id\_pelicula).
    - fecha\_visualizacion.
  - Relaciones:
    - Claves foráneas con contratos y peliculas.

### Tablas Intermedias para Relaciones Muchos a Muchos

- **pelicula\_generos:** Relaciona películas con géneros.
  - Atributos:
    - id\_pelicula (INT, FK -> peliculas.id\_pelicula).
    - id\_genero (INT, FK -> generos.id\_genero).

- **pelicula\_palabrasclave**: Relaciona películas con palabras clave.
  - Atributos:
    - id\_pelicula (INT, FK -> peliculas.id\_pelicula).
    - id\_palabra (INT, FK -> palabrasclave.id\_palabra).

## Tablas Auxiliares

- **generos**: Almacena los géneros filmicos.
  - Atributos:
    - id\_genero (INT, PK, autoincrement).
    - nombre\_genero (VARCHAR(50), único).
- **palabrasclave**: Almacena palabras clave asociadas a las películas.
  - Atributos:
    - id\_palabra (INT, PK, autoincrement).
    - palabra (VARCHAR(50), único).

## Decisiones de Diseño

### Claves Primarias y Foráneas

- Cada tabla cuenta con una clave primaria (como **id\_\***) que se autoincrementa para garantizar que cada registro sea único.
- Las claves foráneas son fundamentales para mantener la integridad entre tablas, asegurando que las relaciones estén correctamente vinculadas.

### Relaciones Muchos a Muchos

- Estas se gestionan mediante tablas intermedias, como **pelicula\_generos** y **pelicula\_palabrasclave**, que sirven para conectar datos entre tablas principales.

### Políticas de Borrado y Actualización

- **ON DELETE CASCADE**:  
Cuando se elimina un registro, como en la tabla **usuarios**, los datos asociados en tablas dependientes como **perfiles** y **contratos** se eliminan automáticamente.
- **ON UPDATE RESTRICT**:  
Restringe cambios en claves primarias que ya están referenciadas por otras tablas, evitando posibles inconsistencias en la base de datos.

## 2. Comandos de Sqoop para Importaciones y Exportaciones

### 2.1 Importación de Tabla Individual sin Filtros

```
sqoop import \  
  --connect jdbc:mariadb://192.168.1.58:3307/moviebind \  
  --username root \  
  --password 'root' \  
  --table usuarios \  
  --target-dir /user/hadoop/text_data/ \  
  --as-textfile \  
  --driver org.mariadb.jdbc.Driver
```

Descripción de flags:

- table:** Tabla específica (usuarios).
- target-dir:** Directorio HDFS donde se almacenan los datos.
- as-textfile:** Define el formato de los datos como texto plano.

### 2.2 Importación con Filtros y Columnas Específicas

```
sqoop import \  
  --connect jdbc:mariadb://192.168.1.58:3307/moviebind \  
  --username root \  
  --password 'root' \  
  --table usuarios \  
  --columns "id_usuario,nickname" \  
  --where "id_usuario > 5" \  
  --target-dir /user/hadoop/text_data/usuarios_filtrados \  
  --as-textfile \  
  --driver org.mariadb.jdbc.Driver
```

Descripción de flags adicionales:

- columns:** Importa solo las columnas id\_usuario y nickname.
- where:** Aplica un filtro (id\_usuario > 5) para limitar los datos importados.

### 2.3 Importación Masiva de Todas las Tablas

Descripción del Proceso:

Se diseñó un script en Bash para automatizar la importación de todas las tablas de la base de datos moviebind. El script funciona leyendo los nombres de las tablas almacenados en un archivo llamado tablas.txt. Este archivo se creó previamente y se transfirió al entorno Linux mediante el comando scp, lo que permitió disponer de la lista necesaria para ejecutar el proceso de importación de forma eficiente.

```
for table in $(cat tablas.txt); do
  sqoop import \
    --connect jdbc:mariadb://192.168.1.58:3307/moviebind \
    --username root \
    --password 'root' \
    --table $table \
    --target-dir /user/hadoop/text_data/$table \
    --as-textfile \
    --driver org.mariadb.jdbc.Driver;
done
```

Explicación del script y flags usados:

**for table in \$(cat tablas.txt):** Lee el archivo tablas.txt línea por línea para obtener el nombre de cada tabla.

**sqoop import:** Ejecuta la importación de datos desde MariaDB hacia HDFS.

**Flags utilizados:**

**--connect:** Define la URL de conexión a la base de datos MariaDB.

**--username y --password:** Credenciales de usuario para autenticar la conexión.

**--table:** Variable \$table para indicar dinámicamente el nombre de la tabla que se está importando.

**--target-dir:** Define el directorio de destino en HDFS donde se almacenan los datos de cada tabla (/user/hadoop/texto\_datos/\$table).

**--as-textfile:** Especifica que los datos deben importarse como archivos de texto plano.

**--driver:** Especifica el driver JDBC de MariaDB.

### 3. Comandos de Apache Hadoop para Leer Datos Comprimidos en Formato Avro y Parquet

#### 3.1 Importación en Formato Avro

Comando utilizado para importar la tabla usuarios en formato Avro:

```
sqoop import \  
  --connect jdbc:mariadb://192.168.1.58:3307/moviebind \  
  --username root \  
  --password 'root' \  
  --table usuarios \  
  --target-dir /user/hadoop/avro_data/ \  
  --as-avrodatafile \  
  --driver org.mariadb.jdbc.Driver
```

Explicación de los flags usados:

**--table:** Nombre de la tabla a importar (usuarios).

**--target-dir:** Define el directorio de destino en HDFS donde se almacenarán los archivos Avro (/user/hadoop/avro\_data/).

**--as-avrodatafile:** Especifica que los datos se importarán en formato Avro.

**--driver:** Indica el driver JDBC de MariaDB necesario para la conexión.

**Verificación en HDFS:** Se ejecuta el siguiente comando para comprobar los archivos generados en HDFS:

```
hadoop fs -ls /user/hadoop/avro_data
```

**Resultado:**

```
hadoop@ubuntucasa:~$ hadoop fs -ls /user/hadoop/avro_data/  
Found 5 items  
-rw-r--r-- 1 hadoop supergroup 0 2024-12-14 13:21 /user/hadoop/avro_d  
ata/_SUCCESS  
-rw-r--r-- 1 hadoop supergroup 733 2024-12-14 13:21 /user/hadoop/avro_d  
ata/part-m-00000.avro  
-rw-r--r-- 1 hadoop supergroup 678 2024-12-14 13:21 /user/hadoop/avro_d  
ata/part-m-00001.avro  
-rw-r--r-- 1 hadoop supergroup 678 2024-12-14 13:21 /user/hadoop/avro_d  
ata/part-m-00002.avro  
-rw-r--r-- 1 hadoop supergroup 732 2024-12-14 13:21 /user/hadoop/avro_d  
ata/part-m-00003.avro
```

Los archivos part-m-00000.avro, part-m-00001.avro, etc., se generan correctamente en el directorio /user/hadoop/avro\_data/, como se muestra en la imagen.

## 3.2 Visualización del Formato Avro

**3.2.1 Descarga de la herramienta avro-tools:** Para visualizar los archivos Avro, es necesario descargar avro-tools:

```
wget http://archive.apache.org/dist/avro/avro-1.10.2/java/avro-tools-1.10.2.jar -O avro-tools.jar
```

### Explicación:

wget descarga el archivo avro-tools desde el repositorio oficial de Apache y lo guarda localmente como avro-tools.jar.

#### 3.2.2 Copia de los archivos Avro desde HDFS a Local:

Se utiliza el siguiente comando para copiar un archivo Avro desde HDFS al sistema de archivos local:

```
hadoop fs -copyToLocal /user/hadoop/avro_data/part-m-000000.avro .
```

### Explicación:

**hadoop fs -copyToLocal:** Copia el archivo part-m-000000.avro desde HDFS al directorio local actual.

#### 3.2.3 Visualización del contenido Avro en JSON:

Inicialmente, si intentamos visualizar el archivo Avro usando cat, el contenido no es legible:

```
cat part-m-000000.avro
```

Por ello, utilizamos avro-tools para convertir el archivo Avro a formato JSON:

```
java -jar avro-tools.jar tojson part-m-000000.avr
```

### Explicación:

- java -jar avro-tools.jar: Ejecuta la herramienta Avro descargada.
- tojson: Convierte el archivo Avro al formato JSON legible.
- part-m-000000.avro: Nombre del archivo Avro que se desea visualizar.



## Resultado:

```
hadoop@ubuntu:~/Descargas$ cat part-m-00000.avro
Objavro.schema("type":"record","name":"usuarios","doc":"Sqoop import of usuarios","fields":[{"name":"id_usuario","type":["null","int"],"default":null,"columnName":"id_usuario","sqlType":"4"},
{"name":"nickname","type":["null","string"],"default":null,"columnName":"nickname","sqlType":"12"},{"name":"contrasena","type":["null","string"],"default":null,"columnName":"contrasena","sqlType":"12"},{"name":"correo_electronico","type":["null","string"],"default":null,"columnName":"correo_electronico","sqlType":"12"}],"tableName":"usuarios"},"s
,sarawilson@example.net,riosnicholas#c2M48Ufj",jonathan22@example.net,berryriley@example.org"},"s
hadoop@ubuntu:~/Descargas$
hadoop@ubuntu:~/Descargas$
hadoop@ubuntu:~/Descargas$ java -jar avro-tools.jar tojson part-m-00000.avro
24/12/15 04:53:32 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
{"id_usuario":{"int":1},"nickname":{"string":"stonejoseph"},"contrasena":{"string":"@dIXfv3%("},"correo_electronico":{"string":"sarawilson@example.net"}}
{"id_usuario":{"int":2},"nickname":{"string":"riosnicholas"},"contrasena":{"string":"#c2M48Ufj"},"correo_electronico":{"string":"jonathan22@example.net"}}
{"id_usuario":{"int":3},"nickname":{"string":"berryriley"},"contrasena":{"string":"!Dmj@xyhvl"},"correo_electronico":{"string":"jacobsriley@example.org"}}
hadoop@ubuntu:~/Descargas$ ^C
hadoop@ubuntu:~/Descargas$
```

El contenido del archivo Avro se muestra correctamente en formato JSON, como se puede ver en la última imagen. El resultado incluye los datos de la tabla usuarios con los atributos id\_usuario, nickname, contrasena y correo\_electronico.

## Resumen de la visualización:

1. Importación en formato Avro con Sqoop.
2. Copia del archivo desde HDFS a local.
3. Conversión del archivo Avro a JSON utilizando avro-tools.

## 3.3 Importación en Formato Parquet

Comando utilizado para importar la tabla usuarios en formato Parquet

```
sqoop import \
  --connect
  jdbc:mariadb://192.168.1.58:3307/moviebind \
  --username root \
  --password 'root' \
  --table usuarios \
  --target-dir /user/hadoop/parquet_data/ \
  --as-parquetfile \
  --driver org.mariadb.jdbc.Driver
```

## Explicación de los flags utilizados

- table:** Nombre de la tabla a importar (usuarios).
- target-dir:** Define el directorio de destino en HDFS (/user/hadoop/parquet\_data/).
- as-parquetfile:** Especifica que los datos se importarán en formato Parquet.
- driver:** Indica el driver JDBC de MariaDB.

## Verificación en HDFS

Una vez completada la importación, se utiliza el siguiente comando para listar los archivos generados en HDFS:

```
hadoop fs -ls /user/hadoop/parquet_data/
```

### Resultado:

Se generaron varios archivos Parquet en el directorio /user/hadoop/parquet\_data/, como se puede observar en la imagen.

```
hadoop@ubuntuCasa:~/Descargas$ hadoop fs -ls /user/hadoop/parquet_data/
Found 5 items
drwxr-xr-x   - hadoop supergroup          0 2024-12-15 04:10 /user/hadoop/parquet_data/.metadata
-rw-r--r--   1 hadoop supergroup       1109 2024-12-15 04:11 /user/hadoop/parquet_data/59feddf9-8253-448a-b08c-a56b74eb68c0.parquet
-rw-r--r--   1 hadoop supergroup       1168 2024-12-15 04:11 /user/hadoop/parquet_data/7c38fbcf-f871-42e1-babd-10749de915d4.parquet
-rw-r--r--   1 hadoop supergroup       1166 2024-12-15 04:11 /user/hadoop/parquet_data/90145c07-39dc-470a-8a01-3925a455e176.parquet
-rw-r--r--   1 hadoop supergroup       1107 2024-12-15 04:11 /user/hadoop/parquet_data/c1fb8ea8-cf54-4734-8df2-92c0d069ccf.parquet
hadoop@ubuntuCasa:~/Descargas$
```

### Visualización del Formato Parquet

#### 1. Descarga de la herramienta parquet-tools

Para poder visualizar archivos en formato Parquet, es necesario descargar la herramienta parquet-tools:

```
wget https://repo1.maven.org/maven2/org/apache/parquet/parquet-tools/1.11.1/parquet-tools-1.11.1.jar
```

**Explicación:** wget descarga la versión 1.11.1 de parquet-tools desde Maven Central.

#### 2. Ejecución de parquet-tools para visualizar datos

Para visualizar los datos del archivo Parquet, se ejecuta el siguiente comando:

```
java -cp parquet-tools-1.11.1.jar:/opt/hadoop/share/hadoop/common/hadoop-common-3.3.6.jar:/opt/hadoop/share/hadoop/common/lib/*:/opt/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-client-core-3.3.6.jar \
org.apache.parquet.tools.Main cat 59feddf9-8253-448a-b08c-a56b74eb68c0.parquet
```

### Explicación del comando:

- **java -cp:** Especifica el classpath para ejecutar **parquet-tools.parquet-tools-1.11.1.jar**: El archivo jar de la herramienta descargada.
- **/opt/hadoop/share/...:** Incluye las dependencias de Hadoop necesarias para leer archivos Parquet.

- **org.apache.parquet.tools.Main cat:** Visualiza el contenido del archivo Parquet en formato legible.
- **59feddf9-8253-448a-b08c-a56b74eb68c0.parquet:** Nombre del archivo Parquet a visualizar.

## Resultado:

```
hadoop@ubuntu:~/Descargas$ java -cp parquet-tools-1.11.1.jar:/opt/hadoop/share/hadoop/common/hadoop-common-3.3.6.jar:/opt/hadoop/share/hadoop/common/lib/*:/opt/hadoop/share/hadoop/mapred
uce/hadoop-mapreduce-client-core-3.3.6.jar org.apache.parquet.tools.Main cat 59feddf9-8253-448a-b08c-a56b74eb68c0.parquet
id_usuario = 6
nickname = michael15
contrasena = "lu^KIG8gG
correo_electronico = alexander24@example.com

id_usuario = 7
nickname = peter49
contrasena = I*3k02mXP^
correo_electronico = jasminestewart@example.org

hadoop@ubuntu:~/Descargas$
```

El contenido del archivo Parquet se muestra en formato legible, como se observa en la imagen.

Se visualizan los registros de la tabla usuarios con los campos id\_usuario, nickname, contrasena y correo\_electronico.

## 4 Uso del Codec Snappy para Importar y Visualizar Datos

### 4.1 Importación de datos con compresión Snappy

Para realizar la importación de datos desde la tabla usuarios de MariaDB a HDFS utilizando Sqoop y el codec de compresión Snappy, se emplea el siguiente comando:

```
sqoop import \
  --connect jdbc:mariadb://192.168.1.58:3307/moviebind \
  --username root \
  --password 'root' \
  --table usuarios \
  --target-dir /user/hadoop/text_data_snp/ \
  --as-textfile \
  --compression-codec org.apache.hadoop.io.compress.SnappyCodec \
  --driver org.mariadb.jdbc.Driver
```

### Explicación del comando:

- **--connect jdbc:mariadb://192.168.1.58:3307/moviebind:** Indica la conexión con la base de datos MariaDB que contiene la tabla **usuarios**.
- **--username y --password:** Credenciales para acceder a la base de datos.
- **--table usuarios:** Especifica la tabla a importar.
- **--target-dir /user/hadoop/text\_data\_snp/:** Define el directorio en HDFS donde se almacenarán los datos importados.
- **--as-textfile:** Indica que los datos se guardarán como archivos de texto plano.

- **--compression-codec org.apache.hadoop.io.compress.SnappyCodec:** Utiliza el codec de compresión Snappy para reducir el tamaño del archivo.
- **--driver org.mariadb.jdbc.Driver:** Especifica el controlador JDBC de MariaDB necesario para la conexión.

## 4.2 Resultados de la importación

```

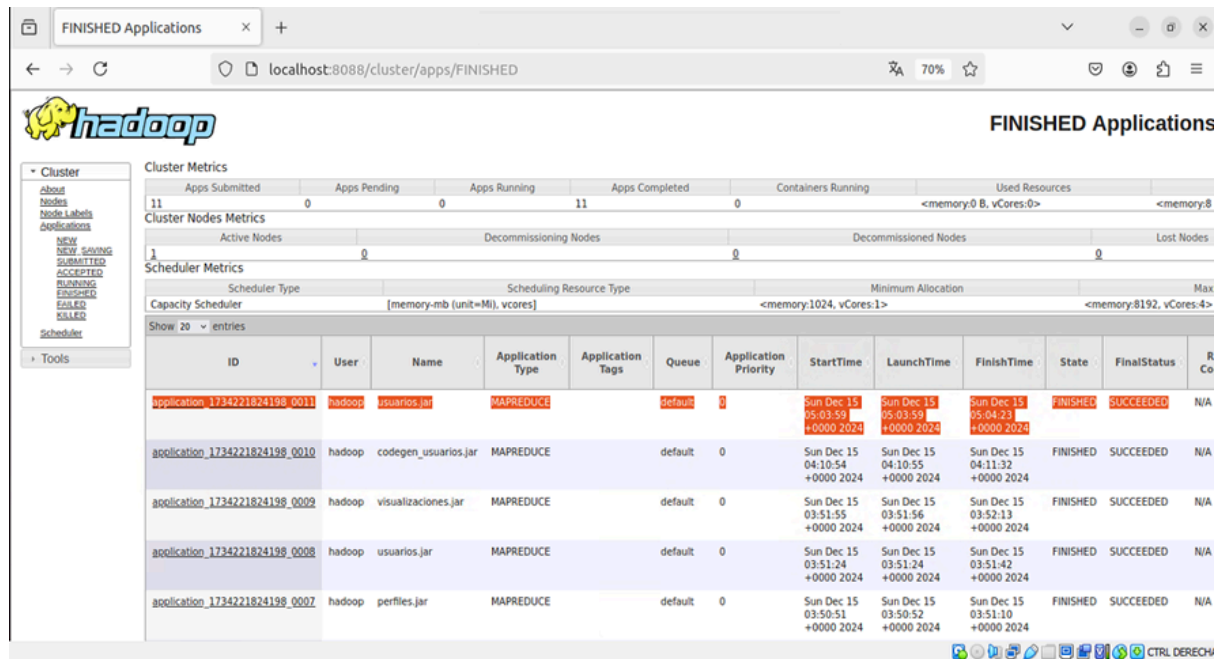
2024-12-15 05:04:08,443 INFO mapreduce.Job: map 0% reduce 0%
2024-12-15 05:04:20,884 INFO mapreduce.Job: map 25% reduce 0%
2024-12-15 05:04:22,953 INFO mapreduce.Job: map 75% reduce 0%
2024-12-15 05:04:23,965 INFO mapreduce.Job: map 100% reduce 0%
2024-12-15 05:04:24,991 INFO mapreduce.Job: Job job_1734221824198_0011 completed successfully
2024-12-15 05:04:25,099 INFO mapreduce.Job: Counters: 34
  File System Counters
    FILE: Number of bytes read=0
    FILE: Number of bytes written=1137940
    FILE: Number of read operations=0
    FILE: Number of large read operations=0
    FILE: Number of write operations=0
    HDFS: Number of bytes read=442
    HDFS: Number of bytes written=506
    HDFS: Number of read operations=24
    HDFS: Number of large read operations=0
    HDFS: Number of write operations=8
    HDFS: Number of bytes read erasure-coded=0
  Job Counters
    Killed map tasks=1
    Launched map tasks=4
    Other local map tasks=4
    Total time spent by all maps in occupied slots (ms)=42726
    Total time spent by all reduces in occupied slots (ms)=0
    Total time spent by all map tasks (ms)=42726
    Total vcore-milliseconds taken by all map tasks=42726
    Total megabyte-milliseconds taken by all map tasks=43751424
  Map-Reduce Framework
    Map input records=10
    Map output records=10
    Input split bytes=442
    Spilled Records=0
    Failed Shuffles=0
    Merged Map outputs=0
    GC time elapsed (ms)=876
    CPU time spent (ms)=4830
    Physical memory (bytes) snapshot=920391680
    Virtual memory (bytes) snapshot=10503184384
    Total committed heap usage (bytes)=670040064
    Peak Map Physical memory (bytes)=232648704
    Peak Map Virtual memory (bytes)=2628603904
  File Input Format Counters
    Bytes Read=0
  File Output Format Counters
    Bytes Written=506
2024-12-15 05:04:25,108 INFO mapreduce.ImportJobBase: Transferred 506 bytes in 34,6727 seconds (14,5936 bytes/sec)
2024-12-15 05:04:25,110 INFO mapreduce.ImportJobBase: Retrieved 10 records.
hadoop@ubuntucasa:~/Descargas$

hadoop@ubuntucasa:~/Descargas$ hadoop fs -ls /user/hadoop/text_data_snp
Found 5 items
-rw-r--r-- 1 hadoop supergroup          0 2024-12-15 05:04 /user/hadoop/text_data_snp/_SUCCESS
-rw-r--r-- 1 hadoop supergroup       151 2024-12-15 05:04 /user/hadoop/text_data_snp/part-m-00000.snappy
-rw-r--r-- 1 hadoop supergroup         91 2024-12-15 05:04 /user/hadoop/text_data_snp/part-m-00001.snappy
-rw-r--r-- 1 hadoop supergroup        106 2024-12-15 05:04 /user/hadoop/text_data_snp/part-m-00002.snappy
-rw-r--r-- 1 hadoop supergroup        158 2024-12-15 05:04 /user/hadoop/text_data_snp/part-m-00003.snappy
hadoop@ubuntucasa:~/Descargas$

```

Una vez ejecutado el comando, la importación se completa exitosamente, como se observa en los logs generados por Sqoop. Esto incluye información como el número de registros importados, el tiempo empleado y las estadísticas de la tarea MapReduce.

### 4.3 Visualización de la tarea generada



Al finalizar la importación, se puede observar la nueva tarea en la interfaz de Hadoop, que muestra las métricas relacionadas con el proceso de MapReduce, incluyendo el estado final "FINISHED" y los recursos utilizados.

### 4.4 Descompresión y lectura de archivos con Snappy

Para visualizar los datos comprimidos en formato Snappy, se utiliza el siguiente comando de Hadoop:

```
hadoop fs -text /user/hadoop/text_data_snp/part-m-000000.snappy
```

Explicación del comando:

- **hadoop fs -text**: Descomprime y muestra el contenido del archivo comprimido en un formato legible para humanos.
- **/user/hadoop/text\_data\_snp/part-m-000000.snappy**: Especifica el archivo comprimido a descomprimir.

El resultado de este comando permite verificar que los datos importados desde la tabla **usuarios** son accesibles y legibles, mostrando los registros en texto plano.

```
hadoop@ubuntucasa:~/Descargas$ hadoop fs -text /user/hadoop/text_data_snp/part-m-00000.snappy
2024-12-15 05:12:21,162 INFO compress.CodecPool: Got brand-new decompressor [.snappy]
1,stonejoseph,@d1XFfv$%(,sarawilson@example.net
2,riosnicholas,#c2M4BUfj^,jonathan22@example.net
3,berrymichael,!Dmj@xyhv1,jacobsriley@example.org
hadoop@ubuntucasa:~/Descargas$
```

## 5 Importación Compatible con Hive

Para realizar una importación de datos compatible con Apache Hive, utilizamos el formato Avro combinado con el codec de compresión Snappy. Este método asegura que los datos sean eficientes para su manejo en Hive y en otras herramientas del ecosistema Hadoop.

El comando de Sqoop para esta importación es el siguiente:

```
sqoop import \
  --connect jdbc:mariadb://192.168.1.58:3307/moviebind \
  --username root \
  --password 'root' \
  --table usuarios \
  --target-dir /user/hadoop/avro_data_snp/ \
  --as-avrodatafile \
  --compression-codec org.apache.hadoop.io.compress.SnappyCodec \
  --driver org.mariadb.jdbc.Driver
```

### 5.1 Explicación del comando:

- **--as-avrodatafile**: Especifica que los datos se almacenarán en formato Avro, que es compatible con Apache Hive.
- **--compression-codec org.apache.hadoop.io.compress.SnappyCodec**: Utiliza el codec Snappy para comprimir los datos, optimizando el espacio y la velocidad de procesamiento.
- **--target-dir /user/hadoop/avro\_data\_snp/**: Define el directorio de destino en HDFS donde se almacenarán los datos importados.

### 5.2 Resultado de la Importación

Una vez ejecutado el comando, la tarea de importación se completa exitosamente. En la consola se muestran los detalles de la tarea, incluyendo el número de registros importados, el tiempo empleado y las estadísticas de la tarea MapReduce.

```

2024-12-15 05:34:42,504 INFO mapreduce.Job: map 0% reduce 0%
2024-12-15 05:34:52,861 INFO mapreduce.Job: map 25% reduce 0%
2024-12-15 05:34:54,889 INFO mapreduce.Job: map 75% reduce 0%
2024-12-15 05:34:55,899 INFO mapreduce.Job: map 100% reduce 0%
2024-12-15 05:34:55,914 INFO mapreduce.Job: Job job_1734221824198_0012 completed successfully
2024-12-15 05:34:56,016 INFO mapreduce.Job: Counters: 33
  File System Counters
    FILE: Number of bytes read=0
    FILE: Number of bytes written=1139720
    FILE: Number of read operations=0
    FILE: Number of large read operations=0
    FILE: Number of write operations=0
    HDFS: Number of bytes read=442
    HDFS: Number of bytes written=2912
    HDFS: Number of read operations=24
    HDFS: Number of large read operations=0
    HDFS: Number of write operations=8
    HDFS: Number of bytes read erasure-coded=0
  Job Counters
    Launched map tasks=4
    Other local map tasks=4
    Total time spent by all maps in occupied slots (ms)=31913
    Total time spent by all reduces in occupied slots (ms)=0
    Total time spent by all map tasks (ms)=31913
    Total vcore-milliseconds taken by all map tasks=31913
    Total megabyte-milliseconds taken by all map tasks=32678912
  Map-Reduce Framework
    Map input records=10
    Map output records=10
    Input split bytes=442
    Spilled Records=0
    Failed Shuffles=0
    Merged Map outputs=0
    GC time elapsed (ms)=716
    CPU time spent (ms)=4780
    Physical memory (bytes) snapshot=992923648
    Virtual memory (bytes) snapshot=10507128832
    Total committed heap usage (bytes)=7298080896
    Peak Map Physical memory (bytes)=253853696
    Peak Map Virtual memory (bytes)=2627751936
  File Input Format Counters
    Bytes Read=0
  File Output Format Counters
    Bytes Written=2912
2024-12-15 05:34:56,023 INFO mapreduce.ImportJobBase: Transferred 2,8438 KB in 25,746 seconds (113,105 bytes/sec)
2024-12-15 05:34:56,029 INFO mapreduce.ImportJobBase: Retrieved 10 records.

```

El directorio `/user/hadoop/avro_data_snp/` en HDFS contiene los archivos Avro resultantes, comprimidos con el codec Snappy. Estos archivos pueden ser utilizados directamente en Apache Hive, ya que son completamente compatibles con su ecosistema.

Además, la interfaz web de Hadoop confirma la finalización exitosa de la tarea, detallando el estado "FINISHED" y los recursos utilizados durante el proceso.

**FINISHED Applications**

**Cluster Metrics**

| Apps Submitted | Apps Pending | Apps Running | Apps Completed | Containers Running | Used Resources         | Total Resources         |
|----------------|--------------|--------------|----------------|--------------------|------------------------|-------------------------|
| 12             | 0            | 0            | 12             | 0                  | <memory:0 B, vCores:0> | <memory:8 GB, vCores:8> |

**Cluster Nodes Metrics**

| Active Nodes | Decommissioning Nodes | Decommissioned Nodes | Lost Nodes |
|--------------|-----------------------|----------------------|------------|
| 1            | 0                     | 0                    | 0          |

**Scheduler Metrics**

| Scheduler Type     | Scheduling Resource Type     | Minimum Allocation      | Maximum Allocation      |
|--------------------|------------------------------|-------------------------|-------------------------|
| Capacity Scheduler | [memory-mb (unit=M), vcores] | <memory:1024, vCores:1> | <memory:8192, vCores:4> |

Show 20 entries

| ID                             | User   | Name         | Application Type | Application Tags | Queue   | Application Priority | StartTime                      | LaunchTime                     | FinishTime                     | State    | FinalStatus | Running Containers |
|--------------------------------|--------|--------------|------------------|------------------|---------|----------------------|--------------------------------|--------------------------------|--------------------------------|----------|-------------|--------------------|
| application_1734221824198_0012 | hadoop | usuarios.jar | MAPREDUCE        |                  | default | 0                    | Sun Dec 15 05:34:36 +0000 2024 | Sun Dec 15 05:34:37 +0000 2024 | Sun Dec 15 05:34:54 +0000 2024 | FINISHED | SUCCEEDED   | N/A                |



## 6 Exportación de Datos a MariaDB

### 6.1 Exportación de Datos de una Sola Tabla

En este caso, los datos almacenados en la tabla **usuarios** de HDFS serán exportados a la tabla equivalente en MariaDB utilizando el siguiente comando:

```
sqoop export \  
  --connect jdbc:mariadb://192.168.1.58:3307/moviebind \  
  --username root \  
  --password 'root' \  
  --table usuarios \  
  --export-dir /user/hadoop/text_data \  
  --input-fields-terminated-by ',' \  
  --input-lines-terminated-by '\n' \  
  --driver org.mariadb.jdbc.Driver
```

### 6.2 Explicación del comando:

- **--table usuarios:** Especifica la tabla destino en MariaDB.
- **--export-dir /user/hadoop/text\_data:** Define el directorio en HDFS que contiene los datos a exportar.
- **--input-fields-terminated-by ',':** Indica que los campos en los datos están separados por comas.
- **--input-lines-terminated-by '\n':** Indica que las líneas están terminadas con un salto de línea.
- **--driver org.mariadb.jdbc.Driver:** Especifica el controlador JDBC para conectar a MariaDB.

### 6.3 Visualización de la Tarea Finalizada

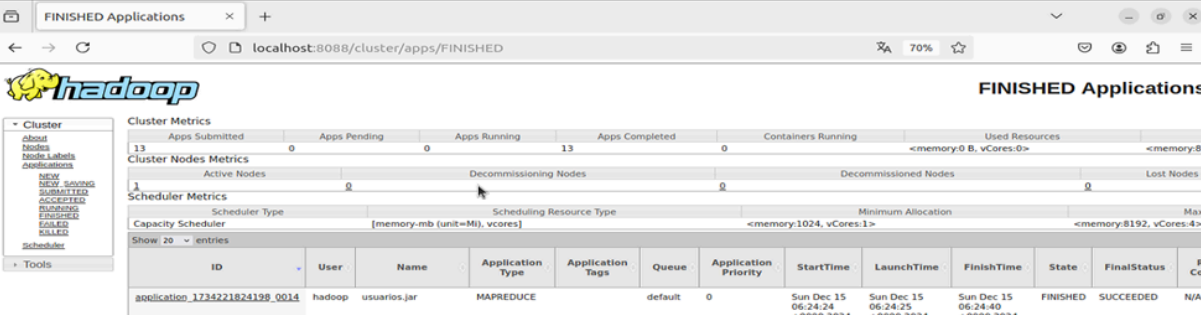
El progreso y la finalización de la tarea de exportación pueden observarse en la interfaz web de Hadoop, que indica que la tarea fue completada con éxito.



```

2024-12-15 06:24:31,542 INFO mapreduce.Job: map 0% reduce 0%
2024-12-15 06:24:40,738 INFO mapreduce.Job: map 100% reduce 0%
2024-12-15 06:24:42,757 INFO mapreduce.Job: Job job_1734221824198_0014 completed successfully
2024-12-15 06:24:42,846 INFO mapreduce.Job: Counters: 33
File System Counters
  FILE: Number of bytes read=0
  FILE: Number of bytes written=852753
  FILE: Number of read operations=0
  FILE: Number of large read operations=0
  FILE: Number of write operations=0
  HDFS: Number of bytes read=1280
  HDFS: Number of bytes written=0
  HDFS: Number of read operations=21
  HDFS: Number of large read operations=0
  HDFS: Number of write operations=0
  HDFS: Number of bytes read erasure-coded=0
Job Counters
  Launched map tasks=3
  Data-local map tasks=3
  Total time spent by all maps in occupied slots (ms)=21449
  Total time spent by all reduces in occupied slots (ms)=0
  Total time spent by all map tasks (ms)=21449
  Total vcore-milliseconds taken by all map tasks=21449
  Total megabyte-milliseconds taken by all map tasks=21963776
Map-Reduce Framework
  Map input records=10
  Map output records=10
  Input split bytes=630
  Spilled Records=0
  Failed Shuffles=0
  Merged Map outputs=0
  GC time elapsed (ms)=419
  CPU time spent (ms)=2590
  Physical memory (bytes) snapshot=653221888
  Virtual memory (bytes) snapshot=7858171904
  Total committed heap usage (bytes)=521666560
  Peak Map Physical memory (bytes)=219176960
  Peak Map Virtual memory (bytes)=2619965440
File Input Format Counters
  Bytes Read=0
File Output Format Counters
  Bytes Written=0
2024-12-15 06:24:42,852 INFO mapreduce.ExportJobBase: Transferred 1,25 KB in 26,3687 seconds (48,5424 bytes/sec)
2024-12-15 06:24:42,855 INFO mapreduce.ExportJobBase: Exported 10 records.
hadoop@ubuntuCasa:~/Descargas$

```



The screenshot shows the Hadoop web interface at localhost:8088. The 'FINISHED Applications' tab is selected, displaying a table of completed jobs. The table includes columns for ID, User, Name, Application Type, Application Tags, Queue, Application Priority, Start Time, Launch Time, Finish Time, State, Final Status, and Resource. A single job is listed with ID 'application\_1734221824198\_0014', User 'hadoop', Name 'usuarios.jar', Application Type 'MAPREDUCE', and a state of 'FINISHED'.

| ID                             | User   | Name         | Application Type | Application Tags | Queue   | Application Priority | StartTime                      | LaunchTime                     | FinishTime                     | State    | FinalStatus | Resource |
|--------------------------------|--------|--------------|------------------|------------------|---------|----------------------|--------------------------------|--------------------------------|--------------------------------|----------|-------------|----------|
| application_1734221824198_0014 | hadoop | usuarios.jar | MAPREDUCE        |                  | default | 0                    | Sun Dec 15 06:24:24 +0000 2024 | Sun Dec 15 06:24:25 +0000 2024 | Sun Dec 15 06:24:40 +0000 2024 | FINISHED | SUCCEEDED   | N/A      |

## 7 Exportación Masiva de Datos a Todas las Tablas

Para exportar datos a todas las tablas de la base de datos, se creó un script Bash que automatiza el proceso. Este script se coloca en el mismo directorio donde previamente se realizó el script de importación. El contenido del script es el siguiente:

```
for table in $(cat tablas.txt); do
  echo "Exportando tabla: $table"
  sqoop export \
    --connect jdbc:mariadb://192.168.1.58:3307/moviebind \
    --username root \
    --password 'root' \
    --table $table \
    --export-dir /user/hadoop/texto_datos/$table \
    --input-fields-terminated-by ',' \
    --input-lines-terminated-by '\n' \
    --driver org.mariadb.jdbc.Driver
done
```

## 7.1 Explicación del Script:

- **for table in \$(cat tablas.txt):** Itera sobre las tablas listadas en el archivo tablas.txt.
- **echo "Exportando tabla: \$table":** Imprime un mensaje en la consola indicando la tabla en proceso.
- **--table \$table:** Utiliza cada tabla iterada en el comando de Sqoop.
- **--export-dir /user/hadoop/texto\_datos/\$table:** Define el directorio de HDFS donde están los datos de cada tabla.

## 7.2 Ejecución del Script

Antes de ejecutar el script, es necesario otorgarle permisos de ejecución con el comando:

```
chmod +x exportar_tablas.sh
```

Luego, se ejecuta con:

```
./exportar_tablas.sh
```

Tras su ejecución, MariaDB se actualizará con los datos exportados desde HDFS.