



INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE COMPUTO



Procesamiento de tablas agregadas utilizando MapReduce

Ingeniería en Sistemas Computacionales
Sistemas Distribuidos

Profesor: Pineda Guerrero Carlos

Alumno: Domínguez Ortega Antonio de Jesús

No. Boleta: 2021630040

Códigos utilizados:

<https://github.com/AntonioDominguezOrtega/Tarea-Extra-Distribuidos>

ÍNDICE

1. Creación de Máquina Virtual en Azure.....	3
2. Instalación del Servidor MySQL.....	4
3. Creación de la Base de Datos "practica.olap"	5
4. Transferencia de Archivos desde Moodle a la Máquina Virtual.....	5
5. Creación de Tablas mediante Script SQL.....	6
6. Carga de Datos a la Tabla sales_data.....	7
7. Carga de Tablas de Dimensiones y Fact Table.....	8
7.1 Carga de tablas de dimensiones simples:.....	8
7.2 Carga de order_date (más compleja):.....	9
7.3 Verificar dimensiones cargadas:.....	10
7.4 Carga de la fact_table.....	10
7.5 Verificación final.....	11
8. Generación del Archivo para Proceso MapReduce.....	13
9. Descarga e Instalación de Apache Hadoop.....	15
10. Descompresión e Instalación de Hadoop.....	16
11. Instalación de Java JDK para Hadoop.....	16
12. Configuración de Java para Hadoop.....	17
13. Desarrollo de Aplicación Hadoop para Agregación de Datos.....	18
13.1 Configuración de Variables de Entorno para Hadoop.....	18
13.2 Creación de Clases MapReduce.....	19
13.3 Compilación y Empaquetado de la Aplicación Hadoop.....	20
13.4 Ejecución del Job MapReduce.....	20
14. Carga de Resultados MapReduce a Base de Datos.....	21
14.1 Transformación del Formato de Salida de Hadoop.....	21
14.2 Conexión a MySQL con Carga Local Habilitada.....	22
14.3 Carga y Verificación de Datos en la Tabla Agregada.....	23
15. Consultas OLAP sobre la Tabla Agregada.....	24
15.1 Cubo OLAP: Acumulado de Ventas por País.....	25
15.2 Cubo OLAP: Acumulado de Ventas por País y Categoría.....	26
19.3 Cubo OLAP: Acumulado de Ventas por País, Categoría y Producto.....	27
Conclusión.....	29

1. Creación de Máquina Virtual en Azure

Se procedió a crear una máquina virtual en Microsoft Azure utilizando la configuración especificada. Se seleccionó el sistema operativo Ubuntu Server, con 2 CPUs virtuales, 4 GB de memoria RAM y disco SSD estándar. La máquina virtual fue nombrada "TE1-2021630040" siguiendo el formato requerido que incluye el número de boleta del estudiante. La implementación se realizó a través del portal web de Azure, configurando los parámetros de red, seguridad y almacenamiento según las mejores prácticas.

The screenshot shows the Azure portal interface for a virtual machine named "TE1-2021630040". The left sidebar has a tree view with "Información general" selected. The main content area shows the following details:

Información esencial	
Grupo de recursos (mover)	: RG-TE1-2021630040
Estado	: En ejecución
Ubicación	: Canada Central
Suscripción (mover)	: Azure for Students
Id. de suscripción	: 9720a544-1a22-428c-af3d-87efbf32ee03
Vista JSON	
Sistema operativo	: Linux (ubuntu 24.04)
Tamaño	: Standard B2s (2 vcpu, 4 GiB de memoria)
IP pública de NIC principal	: 4.206.140.99 1 direcciones IP públicas asociadas
Red virtual/subred	: vnet-canadacentral1/snet-canadacentral-1
Nombre DNS	: Sin configurar
Estado de mantenimiento	: -
Hora de creación	: 17/11/2025, 5:34 UTC

2. Instalación del Servidor MySQL

Se estableció conexión SSH a la máquina virtual y se procedió con la instalación del servidor MySQL. Primero se actualizaron los repositorios del sistema con sudo apt update, luego se instaló el paquete MySQL Server usando sudo apt install mysql-server -y. Se verificó la instalación correcta con mysql --version y se ejecutó el script de seguridad sudo mysql_secure_installation para configurar las políticas de seguridad, eliminar usuarios anónimos y deshabilitar el acceso remoto root. Finalmente, se confirmó que el servicio estuviera activo con sudo systemctl status mysql

Terminal VM:

```
ssh [Usuario]@[Ip_Publica]  
password:
```

```
sudo apt update  
sudo apt install mysql-server -y  
mysql --version
```

```
sudo mysql_secure_installation  
sudo systemctl status mysql
```

```

Antonio@TE1-2021630040:~$ sudo systemctl status mysql
● mysql.service - MySQL Community Server
   Loaded: loaded (/usr/lib/systemd/system/mysql.service; enabled; preset: enabled)
   Active: active (running) since Mon 2025-11-17 05:41:01 UTC; 7min ago
     Process: 2816 ExecStartPre=/usr/share/mysql/mysql-systemd-start pre (code=exited, status=0/SUCCESS)
    Main PID: 2824 (mysqld)
      Status: "Server is operational"
        Tasks: 40 (limit: 4616)
       Memory: 384.8M (peak: 386.4M)
         CPU: 3.374s
      CGroup: /system.slice/mysql.service
              └─2824 /usr/sbin/mysqld

Nov 17 05:41:00 TE1-2021630040 systemd[1]: Starting mysql.service - MySQL Community Server...
Nov 17 05:41:01 TE1-2021630040 systemd[1]: Started mysql.service - MySQL Community Server.

```

Se creó exitosamente el servidor MySQL en la máquina virtual Ubuntu. El servicio se encuentra en estado activo y corriendo, listo para aceptar conexiones.

3. Creación de la Base de Datos "practica.olap"

Se accedió al servidor MySQL y se procedió a crear una nueva base de datos con el nombre "practica.olap". Una vez creada, se verificó que apareciera en el listado de bases de datos disponibles del sistema. Posteriormente, se seleccionó esta base de datos como el contexto activo para las operaciones futuras y se confirmó que estaba siendo utilizada correctamente antes de finalizar la sesión.

Terminal VM:

```
sudo mysql
```

```

CREATE DATABASE practica.olap;
SHOW DATABASES;
USE practica.olap;
SELECT DATABASE();
EXIT;

```

```

mysql> CREATE DATABASE practica.olap;
Query OK, 1 row affected (0.03 sec)

mysql> SHOW DATABASES;
+-----+
| Database      |
+-----+
| information_schema |
| mysql          |
| performance_schema |
| practica.olap  |
| sys            |
+-----+
5 rows in set (0.00 sec)

mysql> USE practica.olap;
Database changed
mysql> SELECT DATABASE();
+-----+
| DATABASE()    |
+-----+
| practica.olap |
+-----+
1 row in set (0.00 sec)

```

La base de datos "practica.olap" fue creada exitosamente y configurada como la base de datos activa. Quedó lista para recibir la estructura de tablas y los datos necesarios para el desarrollo de la práctica OLAP.

4. Transferencia de Archivos desde Moodle a la Máquina Virtual

Los archivos "practica.olap.sql" y "sales_data.csv" fueron descargados previamente desde la plataforma Moodle y almacenados en una carpeta local llamada "Tarea_Extra". Para transferirlos a la máquina virtual en Azure, se utilizó el protocolo SCP (Secure Copy Protocol), copiando ambos archivos desde el equipo local al directorio home del usuario en la máquina virtual remota.

Terminal VM:

```
scp practica.olap.sql sales_data.csv [Usuario]@[Ip_Publica]:~
```

```
antonioortega@antonioortega:~/Documentos/Tarea_Extra$ ls
practica.olap.sql  sales_data.csv
antonioortega@antonioortega:~/Documentos/Tarea_Extra$ scp practica.olap.sql sales_data.csv Antonio@4.206.140.99:~
Antonio@4.206.140.99's password:
practica.olap.sql
sales_data.csv
Antonio@TE1-2021630040:~$ ls
practica.olap.sql  sales_data.csv
Antonio@TE1-2021630040:~$ S
```

Los archivos necesarios para la práctica fueron transferidos exitosamente desde el entorno local hacia la máquina virtual. Ambos archivos quedaron disponibles en el directorio principal del usuario, listos para ser utilizados en la configuración de la base de datos y la carga de datos en los siguientes pasos del proceso.

5. Creación de Tablas mediante Script SQL

Se ejecutó el script "practica.olap.sql" sobre la base de datos "practica.olap" para crear toda la estructura de tablas necesaria. Posteriormente, se accedió al servidor MySQL y se verificó que las tablas hubieran sido creadas correctamente listando todas las tablas existentes en la base de datos.

Terminal VM (mysql):

```
sudo mysql practica.olap < ~/practica.olap.sql
sudo mysql
USE practica.olap;
SHOW TABLES;
```

```

mysql> USE practica_olap;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> SHOW TABLES;
+-----+
| Tables_in_practica_olap |
+-----+
| category
| country
| country_category_product
| country_quarter_category
| country_weekday_product
| customer
| employee
| fact_table
| month
| order_date
| product
| quarter
| region
| sales_data
| semester
| weekday
+-----+
16 rows in set (0.00 sec)

```

El script se ejecutó exitosamente, creando todas las tablas de dimensiones, la fact table y las tablas agregadas definidas en el esquema de la práctica OLAP. La verificación confirmó la presencia de todas las tablas requeridas, incluyendo country, category, product, sales_data, fact_table y las tablas agregadas, estableciendo la estructura completa para el procesamiento de datos.

6. Carga de Datos a la Tabla sales_data

Se procedió a cargar los datos del archivo CSV "sales_data.csv" en la tabla sales_data utilizando la funcionalidad de carga local de MySQL. Se configuraron los parámetros de formato para coincidir con la estructura del archivo CSV, incluyendo delimitadores de campos y líneas, y se especificó ignorar la primera fila que contenía los encabezados. Posteriormente, se realizaron verificaciones para confirmar la integridad de la carga.

Terminal VM (mysql):

```

sudo mysql --local-infile=1 practica_olap
LOAD DATA LOCAL INFILE '/home/Antonio/sales_data.csv'
INTO TABLE sales_data
FIELDS TERMINATED BY ','
ENCLOSED BY ""
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;

-- Contar total de registros cargados
SELECT COUNT(*) AS total_registros FROM sales_data;

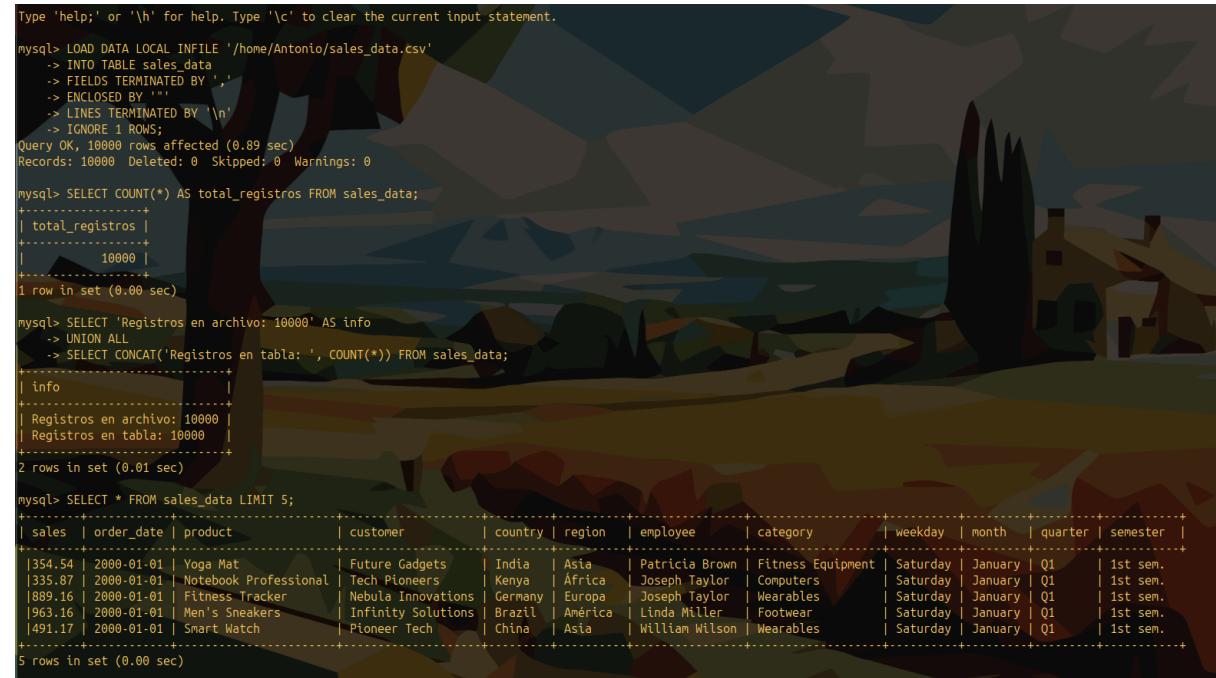
-- Verificar que coincida con el archivo (debería ser 10000, ignorando el header)
SELECT 'Registros en archivo: 10000' AS info
UNION ALL

```

```
SELECT CONCAT('Registros en tabla: ', COUNT(*)) FROM sales_data;
```

-- Ver algunos registros de ejemplo

```
SELECT * FROM sales_data LIMIT 5;
```



```
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> LOAD DATA LOCAL INFILE '/home/Antonio/sales_data.csv'
    -> INTO TABLE sales_data
    -> FIELDS TERMINATED BY ','
    -> ENCLOSED BY ""
    -> LINES TERMINATED BY '\n'
    -> IGNORE 1 ROWS;
Query OK, 10000 rows affected (0.89 sec)
Records: 10000 Deleted: 0 Skipped: 0 Warnings: 0

mysql> SELECT COUNT(*) AS total_registros FROM sales_data;
+-----+
| total_registros |
+-----+
|      10000 |
+-----+
1 row in set (0.00 sec)

mysql> SELECT 'Registros en archivo: 10000' AS info
    -> UNION ALL
    -> SELECT CONCAT('Registros en tabla: ', COUNT(*)) FROM sales_data;
+-----+
| info          |
+-----+
| Registros en archivo: 10000 |
| Registros en tabla: 10000 |
+-----+
2 rows in set (0.01 sec)

mysql> SELECT * FROM sales_data LIMIT 5;
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| sales | order_date | product | customer | country | region | employee | category | weekday | month | quarter | semester |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 354.54 | 2000-01-01 | Yoga Mat | Future Gadgets | India | Asia | Patricia Brown | Fitness Equipment | Saturday | January | Q1 | 1st sem.
| 335.87 | 2000-01-01 | Notebook Professional | Tech Pioneers | Kenya | Africa | Joseph Taylor | Computers | Saturday | January | Q1 | 1st sem.
| 1889.16 | 2000-01-01 | Fitness Tracker | Nebula Innovations | Germany | Europa | Joseph Taylor | Wearables | Saturday | January | Q1 | 1st sem.
| 963.16 | 2000-01-01 | Men's Sneakers | Infinity Solutions | Brazil | America | Linda Miller | Footwear | Saturday | January | Q1 | 1st sem.
| 1491.17 | 2000-01-01 | Smart Watch | Pioneer Tech | China | Asia | William Wilson | Wearables | Saturday | January | Q1 | 1st sem.
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)
```

La carga se completó exitosamente, insertando 10,000 registros en la tabla sales_data. Las verificaciones confirmaron que la cantidad de registros en la tabla coincidía exactamente con los datos originales del archivo CSV. La tabla quedó poblada con toda la información de ventas necesaria para el posterior procesamiento y transformación en el modelo dimensional.

7. Carga de Tablas de Dimensiones y Fact Table

Una vez establecida la estructura de la base de datos, se procedió a poblar las tablas de dimensiones y la fact table para construir el modelo dimensional completo necesario para el análisis OLAP. Se ejecutó lo siguiente:

7.1 Carga de tablas de dimensiones simples:

Se procedió a poblar las tablas de dimensiones básicas extrayendo los valores únicos directamente desde la tabla sales_data. Esto incluyó dimensiones independientes como región, cliente, empleado, semestre, categoría y día de la semana, que no dependían de otras dimensiones para su carga.

Terminal VM (mysql):

-- 1. Cargar region

```
INSERT INTO region (region)
```

```
SELECT DISTINCT region FROM sales_data;
```

-- 2. Cargar country (con región)

```
INSERT INTO country (country, id_region)
```

```
SELECT DISTINCT s.country, r.id_region
```

```

FROM sales_data s
JOIN region r ON s.region = r.region;

-- 3. Cargar customer
INSERT INTO customer (customer)
SELECT DISTINCT customer FROM sales_data;

-- 4. Cargar employee
INSERT INTO employee (employee)
SELECT DISTINCT employee FROM sales_data;

-- 5. Cargar semester
INSERT INTO semester (semester)
SELECT DISTINCT semester FROM sales_data;

-- 6. Cargar quarter (con semester)
INSERT INTO quarter (quarter, id_semester)
SELECT DISTINCT s.quarter, sem.id_semester
FROM sales_data s
JOIN semester sem ON s.semester = sem.semester;

-- 7. Cargar month (con quarter)
INSERT INTO month (month, id_quarter)
SELECT DISTINCT s.month, q.id_quarter
FROM sales_data s
JOIN quarter q ON s.quarter = q.quarter;

-- 8. Cargar weekday
INSERT INTO weekday (weekday)
SELECT DISTINCT weekday FROM sales_data;

-- 9. Cargar category
INSERT INTO category (category)
SELECT DISTINCT category FROM sales_data;

-- 10. Cargar product (con category)
INSERT INTO product (product, id_category)
SELECT DISTINCT s.product, c.id_category
FROM sales_data s
JOIN category c ON s.category = c.category;

```

7.2 Carga de order_date (más compleja):

La dimensión order_date requirió un proceso más elaborado al necesitar integrar múltiples relaciones. Se cargó combinando información de la fecha original con las claves foráneas correspondientes al día de la semana y al mes, asegurando la correcta vinculación con las demás dimensiones temporales.

Terminal VM (mysql):

```
-- 11. Cargar order_date (con weekday y month)
INSERT INTO order_date (order_date, id_weekday, id_month)
SELECT DISTINCT
    s.order_date,
    w.id_weekday,
    m.id_month
FROM sales_data s
JOIN weekday w ON s.weekday = w.weekday
JOIN month m ON s.month = m.month;
```

7.3 Verificar dimensiones cargadas:

Se realizó una verificación integral mediante una consulta que mostró el conteo de registros en cada tabla de dimensión. Esta validación confirmó que todas las dimensiones fueron pobladas correctamente y estaban listas para ser utilizadas en la fact table.

Terminal VM (mysql):

```
-- Verificar conteos
SELECT 'region' as tabla, COUNT(*) as total FROM region
UNION ALL SELECT 'country', COUNT(*) FROM country
UNION ALL SELECT 'customer', COUNT(*) FROM customer
UNION ALL SELECT 'employee', COUNT(*) FROM employee
UNION ALL SELECT 'semester', COUNT(*) FROM semester
UNION ALL SELECT 'quarter', COUNT(*) FROM quarter
UNION ALL SELECT 'month', COUNT(*) FROM month
UNION ALL SELECT 'weekday', COUNT(*) FROM weekday
UNION ALL SELECT 'category', COUNT(*) FROM category
UNION ALL SELECT 'product', COUNT(*) FROM product
UNION ALL SELECT 'order_date', COUNT(*) FROM order_date;
```

```

mysql> SELECT 'region' as tabla, COUNT(*) as total FROM region
-> UNION ALL SELECT 'country', COUNT(*) FROM country
-> UNION ALL SELECT 'customer', COUNT(*) FROM customer
-> UNION ALL SELECT 'employee', COUNT(*) FROM employee
-> UNION ALL SELECT 'semester', COUNT(*) FROM semester
-> UNION ALL SELECT 'quarter', COUNT(*) FROM quarter
-> UNION ALL SELECT 'month', COUNT(*) FROM month
-> UNION ALL SELECT 'weekday', COUNT(*) FROM weekday
-> UNION ALL SELECT 'category', COUNT(*) FROM category
-> UNION ALL SELECT 'product', COUNT(*) FROM product
-> UNION ALL SELECT 'order_date', COUNT(*) FROM order_date;
+-----+-----+
| tabla | total |
+-----+-----+
| region |     4 |
| country |    16 |
| customer |   15 |
| employee |   10 |
| semester |    2 |
| quarter |    4 |
| month |   12 |
| weekday |    7 |
| category |   24 |
| product |   51 |
| order_date | 1571 |
+-----+-----+
11 rows in set (0.00 sec)

```

7.4 Carga de la fact_table

Una vez pobladas todas las dimensiones, se procedió a cargar la fact_table integrando toda la información mediante joins entre la tabla sales_data y todas las tablas de dimensiones creadas. Esto permitió establecer las relaciones correctas mediante claves foráneas para cada dimensión del modelo.

Terminal VM (mysql):

```

-- 12. Cargar fact_table con todas las claves foráneas
INSERT INTO fact_table (sales, id_order_date, id_product, id_customer, id_country, id_employee)
SELECT
    s.sales,
    od.id_order_date,
    p.id_product,
    c.id_customer,
    co.id_country,
    e.id_employee
FROM sales_data s
JOIN order_date od ON s.order_date = od.order_date
JOIN product p ON s.product = p.product
JOIN customer c ON s.customer = c.customer
JOIN country co ON s.country = co.country
JOIN employee e ON s.employee = e.employee;

```

```

mysql> INSERT INTO fact_table (sales, id_order_date, id_product, id_customer, id_country, id_employee)
-> SELECT
->     s.sales,
->     od.id_order_date,
->     p.id_product,
->     c.id_customer,
->     co.id_country,
->     e.id_employee
-> FROM sales_data s
-> JOIN order_date od ON s.order_date = od.order_date
-> JOIN product p ON s.product = p.product
-> JOIN customer c ON s.customer = c.customer
-> JOIN country co ON s.country = co.country
-> JOIN employee e ON s.employee = e.employee;
Query OK, 10000 rows affected (1.43 sec)
Records: 10000  Duplicates: 0  Warnings: 0

```

7.5 Verificación final

Se realizó una verificación completa del modelo dimensional, confirmando que la fact_table contenía exactamente los mismos 10,000 registros que la tabla sales_data original. Además, se validó mediante consultas de ejemplo que las relaciones entre la fact_table y las dimensiones funcionaban correctamente, mostrando datos coherentes y consistentes.

Terminal VM (mysql):

```

-- Verificar fact_table
SELECT COUNT(*) AS total_fact_rows FROM fact_table;

-- Ver resumen completo
SELECT
    'sales_data' as tabla, COUNT(*) as registros FROM sales_data
UNION ALL
SELECT 'fact_table', COUNT(*) FROM fact_table
UNION ALL
SELECT 'TOTAL TABLAS', 13;

-- Ver ejemplo de datos en fact_table
SELECT
    f.sales,
    od.order_date,
    p.product,
    c.customer,
    co.country,
    e.employee
FROM fact_table f
JOIN order_date od ON f.id_order_date = od.id_order_date
JOIN product p ON f.id_product = p.id_product
JOIN customer c ON f.id_customer = c.id_customer
JOIN country co ON f.id_country = co.id_country
JOIN employee e ON f.id_employee = e.id_employee
LIMIT 5;

```

```

mysql> SELECT COUNT(*) AS total_fact_rows FROM fact_table;
+-----+
| total_fact_rows |
+-----+
|      10000 |
+-----+
1 row in set (0.01 sec)

mysql> SELECT
->      'sales_data' as tabla, COUNT(*) as registros FROM sales_data
-> UNION ALL
->      SELECT 'fact_table', COUNT(*) FROM fact_table
-> UNION ALL
->      SELECT 'TOTAL TABLAS', 13;
+-----+-----+
| tabla    | registros |
+-----+-----+
| sales_data |      10000 |
| fact_table |      10000 |
| TOTAL TABLAS |      13 |
+-----+-----+
3 rows in set (0.01 sec)

mysql> SELECT
->      f.sales,
->      od.order_date,
->      p.product,
->      c.customer,
->      co.country,
->      e.employee
->  FROM fact_table f
->  JOIN order_date od ON f.id_order_date = od.id_order_date
->  JOIN product p ON f.id_product = p.id_product
->  JOIN customer c ON f.id_customer = c.id_customer
->  JOIN country co ON f.id_country = co.id_country
->  JOIN employee e ON f.id_employee = e.id_employee
->  LIMIT 5;
+-----+-----+-----+-----+-----+
| sales | order_date | product | customer | country | employee |
+-----+-----+-----+-----+-----+
| 176.06 | 2000-01-01 | Electric Guitar | Ultra Tech | Japan | Patricia Brown |
| 248.30 | 2000-01-01 | Smartphone Alpha | Pinnacle Tech | India | Patricia Brown |
| 354.54 | 2000-01-01 | Yoga Mat | Future Gadgets | India | Patricia Brown |
| 657.62 | 2000-01-02 | LED Desk Lamp | Infinity Solutions | Brazil | Patricia Brown |
| 224.57 | 2000-01-05 | Air Purifier | Elite Tech | Mexico | Patricia Brown |
+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)

```

8. Generación del Archivo para Proceso MapReduce

Se generó el archivo de entrada para el proceso MapReduce mediante una consulta SQL que extrajo los campos id_country, id_category, id_product y sales de la fact_table. Debido a las restricciones de seguridad de MySQL, el archivo se exportó al directorio permitido /var/lib/mysql-files/. Posteriormente, se copió el archivo al directorio /tmp y se ajustaron los permisos para facilitar su acceso durante el procesamiento con Hadoop.

Terminal VM (mysql):

```

SELECT a.id_country, b.id_category, a.id_product, a.sales
INTO OUTFILE '/var/lib/mysql-files/country_category_product.csv'
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY ""
LINES TERMINATED BY '\n'
FROM fact_table a, product b

```

```
WHERE a.id_product = b.id_product;  
EXIT
```

```
mysql> SHOW VARIABLES LIKE 'secure_file_priv';  
+-----+  
| Variable_name | Value |  
+-----+  
| secure_file_priv | /var/lib/mysql-files/ |  
+-----+  
1 row in set (0.01 sec)  
  
mysql> SELECT a.id_country, b.id_category, a.id_product, a.sales  
-> INTO OUTFILE '/var/lib/mysql-files/country_category_product.csv'  
-> FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '\"'  
-> LINES TERMINATED BY '\n'  
-> FROM fact_table a, product b  
-> WHERE a.id_product = b.id_product;  
Query OK, 10000 rows affected (0.02 sec)
```

Terminal VM:

```
# Verificar que el archivo se creó  
sudo ls -la /var/lib/mysql-files/country_category_product.csv
```

```
# Ver las primeras líneas  
sudo head -n 5 /var/lib/mysql-files/country_category_product.csv
```

```
# Contar las líneas (debería ser 10000)  
sudo wc -l /var/lib/mysql-files/country_category_product.csv
```

```
Antonio@TE1-2021630040:~$ sudo ls -la /var/lib/mysql-files/country_category_product.csv  
-rw-r----- 1 mysql mysql 147286 Nov 17 07:27 /var/lib/mysql-files/country_category_product.csv  
Antonio@TE1-2021630040:~$ sudo head -n 5 /var/lib/mysql-files/country_category_product.csv  
1,1,1,354.54  
13,1,1,963.59  
3,1,1,32.70  
8,1,1,559.87  
10,1,1,781.53  
Antonio@TE1-2021630040:~$ sudo wc -l /var/lib/mysql-files/country_category_product.csv  
10000 /var/lib/mysql-files/country_category_product.csv
```

Se confirmó la creación exitosa del archivo verificando su existencia, revisando las primeras líneas de contenido y contando el total de registros exportados. El archivo contenía exactamente 10,000 registros, correspondientes a todos los datos necesarios para el proceso de agregación MapReduce.

Dado que el directorio original de MySQL tiene restricciones de acceso, se copió el archivo al directorio /tmp que es más accesible para el usuario durante el proceso con Hadoop. Se ajustaron los permisos del archivo para garantizar que pudiera ser leído correctamente por las aplicaciones MapReduce sin problemas de acceso.

Terminal VM:

```
# Copiar a /tmp  
sudo cp /var/lib/mysql-files/country_category_product.csv /tmp/
```

```
# Cambiar permisos para poder usarlo  
sudo chmod 644 /tmp/country_category_product.csv
```

```
# Verificar en /tmp  
ls -la /tmp/country_category_product.csv
```

```
# Ver contenido  
head -n 5 /tmp/country_category_product.csv  
wc -l /tmp/country_category_product.csv
```

```
Antonio@TE1-2021630040:~$ sudo cp /var/lib/mysql-files/country_category_product.csv /tmp/  
Antonio@TE1-2021630040:~$ sudo chmod 644 /tmp/country_category_product.csv  
Antonio@TE1-2021630040:~$ ls -la /tmp/country_category_product.csv  
-rw-r--r-- 1 root root 147286 Nov 17 07:31 /tmp/country_category_product.csv  
Antonio@TE1-2021630040:~$ head -n 5 /tmp/country_category_product.csv  
1,1,1,354.54  
13,1,1,963.59  
3,1,1,32.70  
8,1,1,559.87  
10,1,1,781.53  
Antonio@TE1-2021630040:~$ wc -l /tmp/country_category_product.csv  
10000 /tmp/country_category_product.csv  
Antonio@TE1-2021630040:~$ █
```

9. Descarga e Instalación de Apache Hadoop

Se descargó el paquete de Apache Hadoop versión 3.4.0 desde los servidores oficiales de Apache. La descarga se realizó utilizando el comando wget, obteniendo la distribución binaria comprimida específica para la arquitectura ARM64 (aarch64) de la máquina virtual.

Terminal VM:

```
wget https://dlcdn.apache.org/hadoop/common/hadoop-3.4.0/hadoop-3.4.0-aarch64.tar.gz
```

```
Antonio@TE1-2021630040:~$ wget https://dlcdn.apache.org/hadoop/common/hadoop-3.4.0/hadoop-3.4.0-aarch64.tar.gz  
--2025-11-17 07:41:02-- https://dlcdn.apache.org/hadoop/common/hadoop-3.4.0/hadoop-3.4.0-aarch64.tar.gz  
Resolving dlcdn.apache.org (dlcdn.apache.org)... 151.101.2.132, 2a04:4e42:1:64  
Connecting to dlcdn.apache.org (dlcdn.apache.org)|151.101.2.132|:443... connected.  
HTTP request sent, awaiting response...  
200 OK  
Length: 965757299 (921M) [application/x-gzip]  
Saving to: "hadoop-3.4.0-aarch64.tar.gz"  
  
hadoop-3.4.0-aarch64.tar.gz          100%[=====] 921.02M  41.9MB/s   in 9.1s  
2025-11-17 07:41:35 (102 MB/s) - 'hadoop-3.4.0-aarch64.tar.gz' saved [965757299/965757299]  
Antonio@TE1-2021630040:~$  
Antonio@TE1-2021630040:~$ ls  
hadoop-3.4.0-aarch64.tar.gz  practica_olap.sql  sales_data.csv  
Antonio@TE1-2021630040:~$
```

El archivo hadoop-3.4.0-aarch64.tar.gz fue descargado exitosamente y almacenado en el directorio home del usuario. Este paquete contiene todos los componentes necesarios de Hadoop para ejecutar el proceso MapReduce, incluyendo las librerías y herramientas requeridas para el procesamiento distribuido de los datos de ventas.

10. Descompresión e Instalación de Hadoop

Se procedió a descomprimir el archivo de Hadoop descargado utilizando el comando gunzip para eliminar la compresión, seguido del comando tar para extraer todos los archivos del paquete. Una vez completada la extracción, se verificó que el directorio de Hadoop se creó correctamente y que contenía toda la estructura de archivos y carpetas necesarias para su funcionamiento.

Terminal VM:

Descomprimir

```
gunzip hadoop-3.4.0-aarch64.tar.gz
```

```
tar xvfhadoop-3.4.0-aarch64.tar
```

Verificar que se descomprimió

```
ls -la hadoop-3.4.0/
```

```
Antonio@TE1-2021630040:~$ ls -la hadoop-3.4.0/
total 116
drwxr-xr-x 10 Antonio Antonio 4096 Mar  4 2024 .
drwxr-x---  5 Antonio Antonio 4096 Nov 17 07:43 ..
-rw-r--r--  1 Antonio Antonio 23756 Mar  4 2024 LICENSE-binary
-rw-r--r--  1 Antonio Antonio 15696 Mar  4 2024 LICENSE.txt
-rw-r--r--  1 Antonio Antonio 27165 Mar  4 2024 NOTICE-binary
-rw-r--r--  1 Antonio Antonio 1541 Mar  4 2024 NOTICE.txt
-rw-r--r--  1 Antonio Antonio 175 Mar  4 2024 README.txt
drwxr-xr-x  2 Antonio Antonio 4096 Mar  4 2024 bin
drwxr-xr-x  3 Antonio Antonio 4096 Mar  4 2024 etc
drwxr-xr-x  2 Antonio Antonio 4096 Mar  4 2024 include
drwxr-xr-x  3 Antonio Antonio 4096 Mar  4 2024 lib
drwxr-xr-x  4 Antonio Antonio 4096 Mar  4 2024 libexec
drwxr-xr-x  2 Antonio Antonio 4096 Mar  4 2024 licenses-binary
drwxr-xr-x  3 Antonio Antonio 4096 Mar  4 2024 sbin
drwxr-xr-x  4 Antonio Antonio 4096 Mar  4 2024 share
```

Se descomprimió exitosamente el paquete de Hadoop, generando el directorio hadoop-3.4.0 con toda la estructura completa del framework. El directorio incluyó todos los componentes necesarios como los binarios, archivos de configuración, librerías y documentación, dejando Hadoop listo para su configuración y uso en el proceso MapReduce.

11. Instalación de Java JDK para Hadoop

Se actualizaron los repositorios de paquetes del sistema para asegurar la disponibilidad de las últimas versiones de software. Posteriormente, se procedió a instalar el Java Development Kit (JDK) versión 17 en su variante headless, que incluye todas las librerías necesarias para ejecutar Hadoop sin componentes gráficos. Finalmente, se verificó la instalación correcta comprobando la versión de Java y su ubicación en el sistema.

Terminal VM:

Actualizar e instalar Java

```
sudo apt-get update
```

```
sudo apt install openjdk-17-jdk-headless -y
```

Verificar instalación de Java

java -version
which java

```
Antonio@TE1-2021630040:~$ java -version
openjdk version "17.0.16" 2025-07-15
OpenJDK Runtime Environment (build 17.0.16+8-Ubuntu-0ubuntu124.04.1)
OpenJDK 64-Bit Server VM (build 17.0.16+8-Ubuntu-0ubuntu124.04.1, mixed mode, sharing)
Antonio@TE1-2021630040:~$ which java
/usr/bin/java
Antonio@TE1-2021630040:~$
```

Java JDK 17 fue instalado exitosamente en la máquina virtual. La verificación confirmó que Java está correctamente configurado y disponible en el sistema, cumpliendo con el requisito fundamental para el funcionamiento de Apache Hadoop. La ruta del ejecutable de Java fue identificada y quedó lista para ser referenciada en la configuración de Hadoop.

12. Configuración de Java para Hadoop

Se determinó la ubicación del ejecutable de Java en el sistema mediante el comando *which*, identificando que se encontraba en */usr/bin/java*. Con esta información, se procedió a editar el archivo de configuración de entorno de Hadoop para establecer la variable *JAVA_HOME*. Se localizó la línea correspondiente en el archivo de configuración y se descomentó, asignándole el valor */usr* que corresponde al directorio padre donde se encuentra el directorio bin de Java.

Terminal VM:
which java

```
Antonio@TE1-2021630040:~$ which java
/usr/bin/java
```

Terminal VM:
nano hadoop-3.4.0/etc/hadoop/hadoop-env.sh

Busca:
export JAVA_HOME=

Cambiar por:
export JAVA_HOME=/usr

```

# Technically, the only required environment variable is JAVA_HOME.
# All others are optional. However, the defaults are probably not
# preferred. Many sites configure these options outside of Hadoop,
# such as in /etc/profile.d

# The java implementation to use. By default, this environment
# variable is REQUIRED on ALL platforms except OS X!
export JAVA_HOME=/usr

# The language environment in which Hadoop runs. Use the English
# environment to ensure that logs are printed as expected.
export LANG=en_US.UTF-8

# Location of Hadoop. By default, Hadoop will attempt to determine
# this location based upon its execution path.
# export HADOOP_HOME=

# Location of Hadoop's configuration information. i.e., where this
# file is living. If this is not defined, Hadoop will attempt to
# locate it based upon its execution path.
#
# NOTE: It is recommend that this variable not be set here but in

```

^G Help ^O Write Out ^W Where Is ^K Cut
 ^X Exit ^R Read File ^\ Replace ^U Paste ^T Execute
 ^C Location ^J Justify ^/ Go To Line

Terminal VM:

```
grep "JAVA_HOME" hadoop-3.4.0/etc/hadoop/hadoop-env.sh
```

```

Antonio@TE1-2021630040:~$ grep "JAVA_HOME" hadoop-3.4.0/etc/hadoop/hadoop-env.sh
# JAVA_HOME=/usr/java/testing hdfs dfs -ls
# Technically, the only required environment variable is JAVA_HOME.
export JAVA_HOME=/usr

```

La variable de entorno JAVA_HOME fue configurada exitosamente en Hadoop, apuntando correctamente a /usr. Esta configuración es esencial para que Hadoop pueda localizar y utilizar la instalación de Java durante la ejecución de los procesos MapReduce, estableciendo la base para el correcto funcionamiento del framework.

13. Desarrollo de Aplicación Hadoop para Agregación de Datos

Con los componentes de Hadoop instalados y configurados correctamente, se procedió a desarrollar una aplicación MapReduce personalizada. El objetivo de esta aplicación fue procesar el archivo "country_category_product.csv" para generar una tabla agregada que acumulara las ventas agrupadas por país, categoría y producto, implementando así el proceso de agregación distribuida característico del modelo MapReduce.

13.1 Configuración de Variables de Entorno para Hadoop

Se establecieron las variables de entorno necesarias para el funcionamiento de Hadoop en la sesión actual. Se definió JAVA_HOME apuntando al directorio de instalación de Java, HADOOP_HOME especificando la ruta donde se descomprimió Hadoop, y se actualizó la variable PATH para incluir los

directorios bin de ambas herramientas, permitiendo su ejecución desde cualquier ubicación en el sistema.

Terminal VM:

```
export JAVA_HOME=/usr  
export HADOOP_HOME=/home/Antonio/hadoop-3.4.0  
export PATH=$PATH:$JAVA_HOME/bin:$HADOOP_HOME/bin  
  
# Verificar que se establecieron correctamente  
echo $JAVA_HOME  
echo $HADOOP_HOME  
echo $PATH
```

```
Antonio@TE1-2021630040:~$ export JAVA_HOME=/usr  
Antonio@TE1-2021630040:~$ export HADOOP_HOME=/home/Antonio/hadoop-3.4.0  
Antonio@TE1-2021630040:~$ export PATH=$PATH:$JAVA_HOME/bin:$HADOOP_HOME/bin  
Antonio@TE1-2021630040:~$  
Antonio@TE1-2021630040:~$ echo $JAVA_HOME  
/usr  
Antonio@TE1-2021630040:~$ echo $HADOOP_HOME  
/home/Antonio/hadoop-3.4.0  
Antonio@TE1-2021630040:~$ echo $PATH  
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/usr/games:/usr/local/games:/snap/bin:/usr/bin:/home/Antonio/hadoop-3.4.0/bin  
Antonio@TE1-2021630040:~$ █
```

Las variables de entorno críticas para Hadoop fueron configuradas exitosamente. JAVA_HOME, HADOOP_HOME y PATH quedaron correctamente definidas, permitiendo que Hadoop pueda localizar todas sus dependencias y que los comandos de Hadoop estén disponibles globalmente en el sistema para el desarrollo y ejecución de la aplicación MapReduce.

13.2 Creación de Clases MapReduce

Se creó el directorio "prueba" para organizar el proyecto de la aplicación Hadoop. Dentro de este directorio, se generaron los tres archivos Java esenciales para el proceso MapReduce: AggregationMapper.java, AggregationReducer.java y AggregationDriver.java, que contienen la lógica para el mapeo, reducción y control del job respectivamente.

Los archivos fueron poblados con el código fuente correspondiente a cada componente del framework MapReduce. El código completo de estas clases está disponible en el repositorio de GitHub: <https://github.com/AntonioDominguezOrtega/Tarea-Extra-Distribuidos>

Terminal VM:

```
mkdir prueba  
nano prueba/AggregationMapper.java  
nano prueba/AggregationReducer.java  
nano prueba/AggregationDriver.java
```

```
Antonio@TE1-2021630040:~$ mkdir prueba  
Antonio@TE1-2021630040:~$ ls  
hadoop-3.4.0  hadoop-3.4.0-aarch64.tar  practica_olap.sql  prueba  sales_data.csv  
Antonio@TE1-2021630040:~$ nano prueba/AggregationMapper.java  
Antonio@TE1-2021630040:~$ nano prueba/AggregationReducer.java  
Antonio@TE1-2021630040:~$ nano prueba/AggregationDriver.java
```

Se estableció la estructura completa del proyecto MapReduce con las tres clases fundamentales implementadas. Los archivos quedaron listos para ser compilados y empaquetados en el siguiente paso, formando la base de la aplicación de agregación que procesará los datos de ventas.

13.3 Compilación y Empaquetado de la Aplicación Hadoop

Se procedió a compilar las clases Java del proyecto MapReduce utilizando el classpath de Hadoop para incluir todas las dependencias necesarias. La compilación generó los archivos .class correspondientes a cada componente. Posteriormente, se creó un archivo JAR ejecutable que empaqueta todas las clases compiladas en un único archivo distribuible.

Terminal VM:

Compilar las clases

```
javac -classpath "`$HADOOP_HOME/bin/hadoop classpath`" -d prueba prueba/*.java
```

Crear el JAR

```
jar -cvf Aggregation.jar -C prueba .
```

```
Antonio@TE1-2021630040:~$ javac -classpath "$HADOOP_HOME/bin/hadoop classpath" -d prueba prueba/*.java
Antonio@TE1-2021630040:~$ jar -cvf Aggregation.jar -C prueba .
added manifest
adding: AggregationDriver.class(in = 1518) (out= 835)(deflated 44%)
adding: AggregationDriver.java(in = 1135) (out= 432)(deflated 61%)
adding: AggregationMapper.class(in = 2313) (out= 968)(deflated 58%)
adding: AggregationMapper.java(in = 1233) (out= 466)(deflated 62%)
adding: AggregationReducer.class(in = 1892) (out= 819)(deflated 56%)
adding: AggregationReducer.java(in = 746) (out= 360)(deflated 51%)
```

La compilación se completó exitosamente, generando los bytecodes de las clases MapReduce. El archivo Aggregation.jar fue creado correctamente, conteniendo toda la aplicación Hadoop empaquetada y lista para ser ejecutada en el cluster. Este JAR representa la aplicación completa de agregación que será ejecutada por el framework Hadoop para procesar los datos de ventas.

13.4 Ejecución del Job MapReduce

Se creó el directorio de entrada para la aplicación Hadoop y se copió el archivo CSV con los datos de ventas. Antes de ejecutar el job, se aseguró que no existiera un directorio de salida previo para evitar conflictos. Finalmente, se lanzó el job de Hadoop especificando la clase principal, el directorio de entrada y el directorio de salida.

Terminal VM:

Crear directorio de entrada

```
mkdir prueba/input
```

Copiar el archivo CSV

```
cp /tmp/country_category_product.csv prueba/input/
```

Eliminar directorio de salida si existe

```
rm -rf prueba/output
```

Ejecutar el job de Hadoop

```
hadoop jar Aggregation.jar AggregationDriver prueba/input prueba/output
```

```
Antonio@TE1-2021630040:~$ mkdir prueba/input
Antonio@TE1-2021630040:~$ cp /tmp/country_category_product.csv prueba/input/
Antonio@TE1-2021630040:~$ rm -rf prueba/output
Antonio@TE1-2021630040:~$ hadoop jar Aggregation.jar AggregationDriver prueba/input prueba/output
2025-11-17 08:28:58,348 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
2025-11-17 08:28:58,762 INFO impl.MetricsConfig: Loaded properties from hadoop-metrics2.properties
2025-11-17 08:28:58,912 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second(s).
2025-11-17 08:28:58,912 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2025-11-17 08:28:59,088 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
2025-11-17 08:28:59,161 INFO input.FileInputFormat: Total input files to process : 1
2025-11-17 08:28:59,227 INFO mapreduce.JobSubmitter: number of splits:1
2025-11-17 08:28:59,465 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_local2088582238_0001
2025-11-17 08:28:59,465 INFO mapreduce.JobSubmitter: Executing with tokens: []
2025-11-17 08:28:59,681 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
2025-11-17 08:28:59,682 INFO mapreduce.Job: Running Job: job_local2088582238_0001
2025-11-17 08:28:59,683 INFO mapred.LocalJobRunner: OutputCommitter set in config null
2025-11-17 08:28:59,694 INFO output.PathOutputCommitterFactory: No output committer factory defined, defaulting to FileOutputCommitterFactory
2025-11-17 08:28:59,695 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2025-11-17 08:28:59,695 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: false
2025-11-17 08:28:59,696 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapreduce.lib.output.FileOutputCommitter
2025-11-17 08:28:59,747 INFO mapred.LocalJobRunner: Waiting for map tasks
2025-11-17 08:28:59,748 INFO mapred.LocalJobRunner: Starting task: attempt_local2088582238_0001_m_000000
2025-11-17 08:28:59,769 INFO output.PathOutputCommitterFactory: No output committer factory defined, defaulting to FileOutputCommitterFactory
2025-11-17 08:28:59,770 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2025-11-17 08:28:59,771 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup _temporary folders under output directory:false, ignore cleanup failures: false
2025-11-17 08:28:59,816 INFO mapred.Task: Using ResourceCalculatorProcessTree : [ ]
2025-11-17 08:28:59,823 INFO mapred.MapTask: Processing split: file:/home/Antonio/prueba/input/country_category_product.csv:0+147286
2025-11-17 08:28:59,878 INFO mapred.MapTask: (EQUATOR) 0 kvi 26214396(104857584)
2025-11-17 08:28:59,878 INFO mapred.MapTask: mapreduce.task.io.sort.mb: 100
2025-11-17 08:28:59,879 INFO mapred.MapTask: soft limit at 83886080
2025-11-17 08:28:59,879 INFO mapred.MapTask: bufstart = 0; bufvoid = 104857600
2025-11-17 08:28:59,879 INFO mapred.MapTask: kvstart = 26214396; length = 6553600
2025-11-17 08:29:00,057 INFO mapred.LocalJobRunner:
2025-11-17 08:29:00,059 INFO mapred.MapTask: Starting flush of map output
2025-11-17 08:29:00,059 INFO mapred.MapTask: Spilling map output
2025-11-17 08:29:00,059 INFO mapred.MapTask: bufstart = 0; bufend = 147286; bufvoid = 104857600
2025-11-17 08:29:00,059 INFO mapred.MapTask: kvstart = 26214396(104857584); kvend = 26174400(104697600); length = 39997/6553600
```

```
Antonio@TE1-2021630040:~/prueba/output$ ls
SUCCESS  part-r-00000
```

El job MapReduce se ejecutó exitosamente, generando dos archivos en el directorio de salida (prueba/output). El archivo _SUCCESS confirmó la finalización correcta del proceso, mientras que part-r-00000 contenía los resultados de la agregación con las ventas acumuladas por país, categoría y producto. Estos archivos representan la tabla agregada generada mediante el procesamiento distribuido de Hadoop.

14. Carga de Resultados MapReduce a Base de Datos

Con los resultados del proceso MapReduce generados exitosamente, se procedió a preparar y cargar los datos agregados a la base de datos MySQL. Esta etapa involucró la transformación del formato de salida de Hadoop y la posterior inserción en la tabla agregada "country_category_product" para su utilización en consultas OLAP.

14.1 Transformación del Formato de Salida de Hadoop

Se identificó que el archivo de resultados generado por Hadoop utilizaba tabuladores como separadores entre las claves y valores. Para adaptarlo al formato CSV requerido por MySQL, se ejecutó un comando de sustitución que reemplazó todos los tabuladores por comas. Posteriormente, se verificó visualmente que el cambio se aplicó correctamente en las primeras líneas del archivo.

Terminal VM:

```
# Cambiar tabuladores por comas en el archivo resultante
sed -i 's/\t/,/g' ~/prueba/output/part-r-00000
```

```
# Verificar el cambio
echo "==== Primeras 5 líneas después del cambio ==="
```

```
head -n 5 ~/prueba/output/part-r-00000
# Verificar que ahora usa comas como separador
echo "==== Verificando separadores ==="
head -n 1 ~/prueba/output/part-r-00000 | cat -A
```

```
Antonio@TE1-2021630040:~/prueba/output$ sed -i 's/\t/,/g' ~/prueba/output/part-r-00000
Antonio@TE1-2021630040:~/prueba/output$ echo "==== Primeras 5 líneas después del cambio ==="
==== Primeras 5 líneas después del cambio ===
Antonio@TE1-2021630040:~/prueba/output$ head -n 5 ~/prueba/output/part-r-00000
1,1,1,7187.64
1,1,11,5791.529999999999
1,10,17,4492.640000000001
1,10,32,7046.429999999999
1,11,19,4471.83
Antonio@TE1-2021630040:~/prueba/output$ echo "==== Verificando separadores ==="
==== Verificando separadores ===
Antonio@TE1-2021630040:~/prueba/output$ head -n 1 ~/prueba/output/part-r-00000 | cat -A
1,1,1,7187.64$
Antonio@TE1-2021630040:~/prueba/output$ 
```

El archivo part-r-00000 fue convertido exitosamente de formato clave-valor con separadores de tabulación a formato CSV con separadores de coma. Esta transformación permitió que el archivo fuera compatible con los procedimientos de carga de MySQL, preparándolo para su inserción en la base de datos.

14.2 Conexión a MySQL con Carga Local Habilitada

Se estableció una conexión al servidor MySQL activando específicamente la opción "local-infile", que permite la carga de archivos locales en la base de datos. Esta configuración es necesaria para importar el archivo de resultados generado por Hadoop directamente desde el sistema de archivos de la máquina virtual.

Terminal VM:

```
# Conectarse a MySQL con la opción local-infile habilitada
sudo mysql --local-infile=1 practica.olap
```

```
Antonio@TE1-2021630040:~/prueba/output$ sudo mysql --local-infile=1 practica.olap
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 27
Server version: 8.0.43-0ubuntu0.24.04.2 (Ubuntu)

Copyright (c) 2000, 2025, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

La sesión de MySQL se inició exitosamente con los permisos de carga local habilitados, creando el entorno adecuado para ejecutar el comando de importación del archivo CSV. La conexión quedó establecida con la base de datos "practica.olap" lista para recibir los datos agregados del proceso MapReduce.

14.3 Carga y Verificación de Datos en la Tabla Agregada

Se ejecutó el comando de carga para importar el archivo transformado a la tabla country_category_product, especificando el formato CSV con separadores de coma. Una vez completada la carga, se realizaron múltiples verificaciones para garantizar la integridad de los datos, incluyendo el conteo total de registros cargados, la revisión de registros de ejemplo y la validación de que no existieran valores nulos en las columnas críticas.

Terminal VM(mysql):

```
-- Cargar el archivo a la tabla country_category_product
LOAD DATA LOCAL INFILE '/home/Antonio/prueba/output/part-r-00000'
INTO TABLE country_category_product
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
(id_country, id_category, id_product, sales);

-- Verificar cuántos registros se cargaron
SELECT COUNT(*) AS total_registros_cargados FROM country_category_product;

-- Ver los primeros 10 registros
SELECT * FROM country_category_product LIMIT 10;

-- Verificar que no hay valores nulos
SELECT
    COUNT(*) as total,
    SUM(CASE WHEN id_country IS NULL THEN 1 ELSE 0 END) as nulos_country,
    SUM(CASE WHEN id_category IS NULL THEN 1 ELSE 0 END) as nulos_category,
    SUM(CASE WHEN id_product IS NULL THEN 1 ELSE 0 END) as nulos_product,
    SUM(CASE WHEN sales IS NULL THEN 1 ELSE 0 END) as nulos_sales
FROM country_category_product;
```

```

mysql> TRUNCATE TABLE country_category_product;
Query OK, 0 rows affected (0.31 sec)

mysql> LOAD DATA LOCAL INFILE '/home/Antonio/prueba/output/part-r-00000'
-> INTO TABLE country_category_product
-> FIELDS TERMINATED BY ','
-> LINES TERMINATED BY '\n'
-> (id_country, id_category, id_product, sales);
Query OK, 816 rows affected, 369 warnings (0.07 sec)
Records: 816 Deleted: 0 Skipped: 0 Warnings: 369

mysql> SELECT COUNT(*) AS total_registros_cargados FROM country_category_product;
+-----+
| total_registros_cargados |
+-----+
| 816 |
+-----+
1 row in set (0.01 sec)

mysql> SELECT * FROM country_category_product LIMIT 10;
+-----+-----+-----+-----+
| id_country | id_category | id_product | sales |
+-----+-----+-----+-----+
| 1 | 1 | 1 | 7187.64 |
| 1 | 1 | 11 | 5791.53 |
| 1 | 10 | 17 | 4492.64 |
| 1 | 10 | 32 | 7046.43 |
| 1 | 11 | 19 | 4471.83 |
| 1 | 11 | 43 | 6772.27 |
| 1 | 12 | 20 | 7562.84 |
| 1 | 12 | 30 | 3522.75 |
| 1 | 13 | 21 | 6034.70 |
| 1 | 13 | 38 | 9237.73 |
+-----+-----+-----+-----+
10 rows in set (0.00 sec)

mysql> SELECT
->     COUNT(*) as total,
->     SUM(CASE WHEN id_country IS NULL THEN 1 ELSE 0 END) as nulos_country,
->     SUM(CASE WHEN id_category IS NULL THEN 1 ELSE 0 END) as nulos_category,
->     SUM(CASE WHEN id_product IS NULL THEN 1 ELSE 0 END) as nulos_product,
->     SUM(CASE WHEN sales IS NULL THEN 1 ELSE 0 END) as nulos_sales
->   FROM country_category_product;
+-----+-----+-----+-----+-----+
| total | nulos_country | nulos_category | nulos_product | nulos_sales |
+-----+-----+-----+-----+-----+
| 816 | 0 | 0 | 0 | 0 |
+-----+-----+-----+-----+
1 row in set (0.00 sec)

```

La carga se completó exitosamente con 816 registros insertados en la tabla agregada. Las verificaciones confirmaron que todos los datos fueron cargados correctamente sin valores nulos en las columnas esenciales. La tabla `country_category_product` quedó poblada con los resultados del proceso MapReduce, conteniendo las ventas agregadas por país, categoría y producto, lista para ser utilizada en las consultas OLAP finales.

15. Consultas OLAP sobre la Tabla Agregada

Con los datos agregados cargados exitosamente en la tabla `country_category_product`, se procedió a ejecutar las consultas OLAP que demuestran las capacidades de análisis multidimensional. Estas consultas aprovechan la estructura pre-agregada de la tabla para generar cubos de datos con diferentes

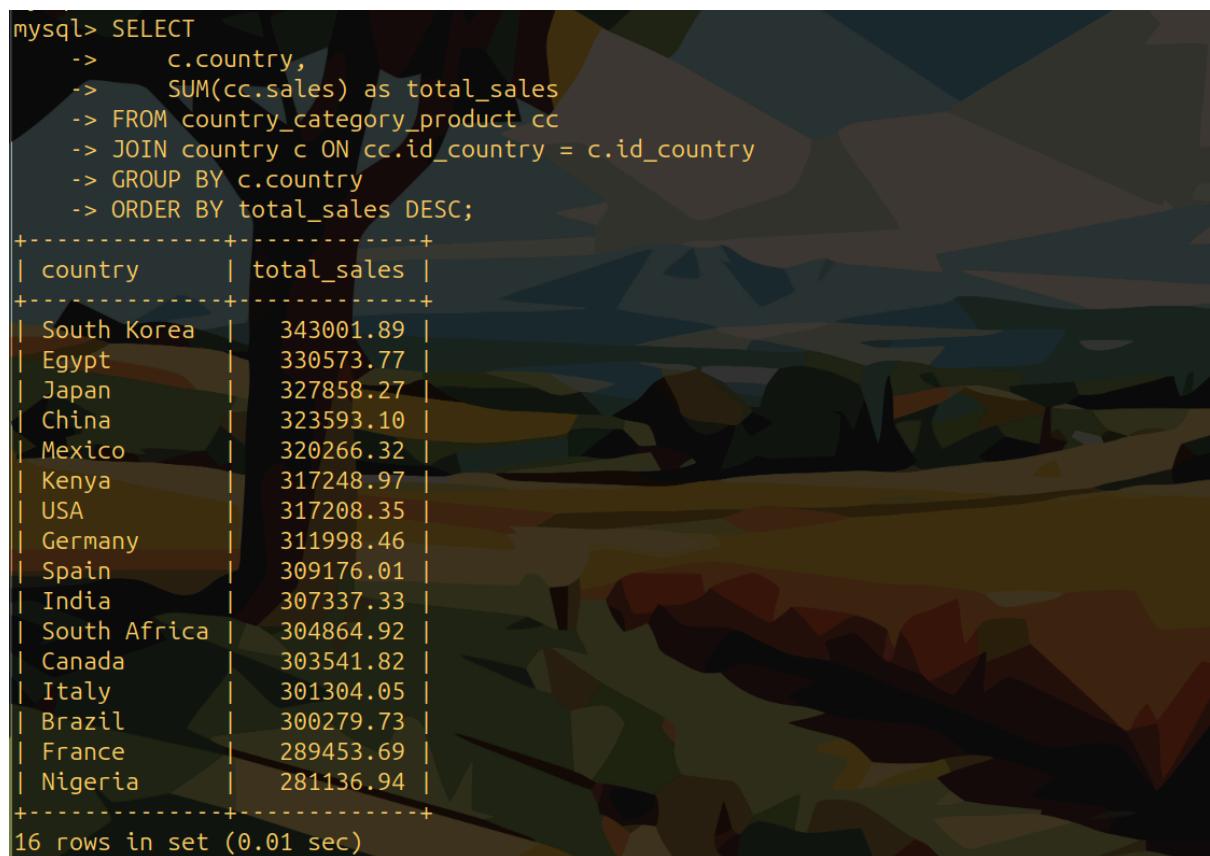
niveles de granularidad, optimizando el rendimiento al evitar el procesamiento directo sobre la fact-table más grande.

15.1 Cubo OLAP: Acumulado de Ventas por País

Se ejecutó una consulta de agregación que calculó el total de ventas agrupado por país. La consulta utilizó la tabla agregada country_category_product y realizó un join con la dimensión country para obtener los nombres de los países. Los resultados se ordenaron de forma descendente para identificar rápidamente los países con mayores volúmenes de ventas.

Terminal VM(mysql):

```
SELECT
    c.country,
    SUM(cc.sales) as total_sales
FROM country_category_product cc
JOIN country c ON cc.id_country = c.id_country
GROUP BY c.country
ORDER BY total_sales DESC;
```



country	total_sales
South Korea	343001.89
Egypt	330573.77
Japan	327858.27
China	323593.10
Mexico	320266.32
Kenya	317248.97
USA	317208.35
Germany	311998.46
Spain	309176.01
India	307337.33
South Africa	304864.92
Canada	303541.82
Italy	301304.05
Brazil	300279.73
France	289453.69
Nigeria	281136.94

16 rows in set (0.01 sec)

La consulta generó exitosamente el cubo OLAP que muestra la distribución de ventas totales por país. Este análisis permitió identificar los mercados más rentables y comprender el comportamiento de ventas a nivel geográfico, proporcionando información valiosa para la toma de decisiones comerciales estratégicas basadas en el desempeño por país.

15.2 Cubo OLAP: Acumulado de Ventas por País y Categoría

Se ejecutó una consulta de agregación multidimensional que calculó el total de ventas agrupado por la combinación de país y categoría. La consulta utilizó joins con las dimensiones country y category para enriquecer los datos con información descriptiva. Los resultados se organizaron primero por país y luego por monto de ventas en orden descendente para facilitar el análisis comparativo.

Terminal VM(mysql):

```
SELECT
    c.country,
    cat.category,
    SUM(cc.sales) as total_sales
FROM country_category_product cc
JOIN country c ON cc.id_country = c.id_country
JOIN category cat ON cc.id_category = cat.id_category
GROUP BY c.country, cat.category
ORDER BY c.country, total_sales DESC;
```

```

mysql> SELECT
    ->     c.country,
    ->     cat.category,
    ->     SUM(cc.sales) as total_sales
    -> FROM country_category_product cc
    -> JOIN country c ON cc.id_country = c.id_country
    -> JOIN category cat ON cc.id_category = cat.id_category
    -> GROUP BY c.country, cat.category
    -> ORDER BY c.country, total_sales DESC;
+-----+-----+-----+
| country | category | total_sales |
+-----+-----+-----+
| Brazil  | Mobile Devices | 29729.86 |
| Brazil  | Audio Equipment | 18667.11 |
| Brazil  | Computers | 18065.85 |
| Brazil  | Outdoor Gear | 16445.10 |
| Brazil  | Storage Devices | 16362.15 |
| Brazil  | Books | 14778.31 |
| Brazil  | Home Appliances | 14632.82 |
| Brazil  | Wearables | 14229.26 |
| Brazil  | Computer Accessories | 13150.81 |
| Brazil  | Fitness Equipment | 12146.79 |
| Brazil  | Musical Instruments | 11985.49 |
| Brazil  | Footwear | 11520.12 |
| Brazil  | Monitors | 11286.43 |
| Brazil  | Pet Supplies | 11149.34 |
| Brazil  | Art | 10795.02 |
| Brazil  | Office Furniture | 10677.60 |
| Brazil  | School Supplies | 10476.29 |
| Brazil  | Kitchen Appliances | 10416.94 |
| Brazil  | Home Decor | 9979.56 |
| Brazil  | Home Electronics | 8975.39 |
| Brazil  | Electronics | 7327.16 |
| Brazil  | Video Games | 6993.22 |
| Brazil  | Fashion Accessories | 6346.25 |
| Brazil  | Lighting | 4142.86 |
| Canada  | Mobile Devices | 28941.26 |
| Canada  | Computers | 26274.21 |
| Canada  | Outdoor Gear | 21454.57 |
| Canada  | Audio Equipment | 18771.06 |
| Canada  | Monitors | 15675.46 |
| Canada  | Video Games | 13524.32 |
| Canada  | Books | 13214.34 |
| Canada  | Electronics | 12913.53 |
| Canada  | Kitchen Appliances | 12722.01 |
| Canada  | Pet Supplies | 12181.23 |
| Canada  | Footwear | 11300.50 |
| Canada  | Fitness Equipment | 10148.36 |
| Canada  | Musical Instruments | 10043.48 |
| Canada  | Home Appliances | 10015.54 |
| Canada  | Storage Devices | 9821.57 |
| Canada  | Art | 9621.60 |
| Canada  | Fashion Accessories | 9573.08 |
| Canada  | Wearables | 9039.73 |
| Canada  | Office Furniture | 8835.74 |
| Canada  | Home Electronics | 8554.66 |
| Canada  | School Supplies | 8446.38 |
| Canada  | Lighting | 7828.56 |
| Canada  | Home Decor | 7614.34 |
| Canada  | Computer Accessories | 7026.29 |
| China   | Mobile Devices | 27282.98 |
| China   | Computers | 26709.26 |
| China   | Outdoor Gear | 24410.76 |
| China   | Books | 15999.24 |
| China   | Home Decor | 15496.07 |
| China   | Monitors | 14858.96 |
| China   | Video Games | 13279.69 |
| China   | Fashion Accessories | 13164.82 |
| China   | Office Furniture | 13133.31 |
| China   | Art | 12896.48 |

```

La consulta generó exitosamente el cubo OLAP que muestra el desglose de ventas por país y categoría de producto. Este nivel de detalle permitió identificar qué categorías de productos tienen mejor desempeño en cada país, revelando patrones de preferencia regional y oportunidades de optimización de inventario y estrategias de marketing específicas por mercado.

19.3 Cubo OLAP: Acumulado de Ventas por País, Categoría y Producto

Se ejecutó una consulta de agregación con el máximo nivel de detalle, calculando el total de ventas agrupado por la combinación de país, categoría y producto específico. La consulta integró tres

dimensiones mediante joins con las tablas country, category y product para obtener la información descriptiva completa. Los resultados se limitaron a 20 registros para facilitar la visualización, organizándose por país, categoría y monto de ventas descendente.

Terminal VM(mysql):

SELECT

```
c.country,  
cat.category,  
p.product,  
SUM(cc.sales) as total_sales  
FROM country_category_product cc  
JOIN country c ON cc.id_country = c.id_country  
JOIN category cat ON cc.id_category = cat.id_category  
JOIN product p ON cc.id_product = p.id_product  
GROUP BY c.country, cat.category, p.product  
ORDER BY c.country, cat.category, total_sales DESC  
LIMIT 20; -- Limitar para no saturar la pantalla
```

country	category	product	total_sales
Brazil	Art	Landscape Painting	6189.27
Brazil	Art	Sculpture	4605.75
Brazil	Audio Equipment	Bluetooth Speaker	10545.97
Brazil	Audio Equipment	Noise Cancelling Headphones	8121.14
Brazil	Books	Science Fiction Novel	7945.72
Brazil	Books	Biography Book	6832.59
Brazil	Computer Accessories	USB Keyboard	9574.01
Brazil	Computer Accessories	Wireless Mouse	3576.80
Brazil	Computers	Notebook Professional	5990.04
Brazil	Computers	Desktop Beta	4582.72
Brazil	Computers	Notebook Basic	3863.20
Brazil	Computers	Desktop Alpha	3629.89
Brazil	Electronics	Virtual Reality Headset	4852.41
Brazil	Electronics	Drone with Camera	2474.75
Brazil	Fashion Accessories	Wristwatch	3790.02
Brazil	Fashion Accessories	Sunglasses	2556.23
Brazil	Fitness Equipment	Yoga Mat	7534.52
Brazil	Fitness Equipment	Dumbbell Set	4612.27
Brazil	Footwear	Men's Sneakers	6520.40
Brazil	Footwear	Women's Sandals	4999.72

La consulta generó exitosamente el cubo OLAP más granular, mostrando el desempeño de ventas a nivel de producto específico dentro de cada categoría y país. Este nivel de detalle proporcionó información invaluable para identificar los productos más rentables en cada mercado, permitiendo estrategias de gestión de inventario, pricing y promociones altamente específicas y basadas en datos concretos.

Conclusión

Esta práctica demostró la funcionalidad de combinar diferentes tecnologías para el análisis de datos. Hadoop permitió procesar eficientemente grandes volúmenes de información, mientras que MySQL facilitó el análisis detallado mediante consultas OLAP.

El proceso completo, desde la configuración inicial hasta las consultas finales, evidenció cómo estas herramientas se complementan para transformar datos en información valiosa. Los resultados obtenidos proporcionan insights accionables sobre el comportamiento de ventas que pueden guiar decisiones comerciales estratégicas.

La integración exitosa de estas tecnologías establece un patrón replicable para futuros proyectos de análisis de datos a escala.