ITESO, Universidad Jesuita de Guadalajara

Cassandra Lab 3 - Python Cassandra APP

Francisco Flores Enríquez

728238

1 / 03 / 2024

Introducción

Este laboratorio documenta el proceso de aprendizaje y desarrollo de un proyecto práctico utilizando Apache Cassandra, Docker y Python. El objetivo principal del proyecto fue adquirir experiencia práctica en el diseño, implementación y manipulación de una base de datos NoSQL. El proyecto consistió en simular un sistema de gestión de datos simplificado, abordando desde la creación de un entorno de desarrollo contenerizado hasta la implementación de una serie de operaciones de base de datos representativas.

Métodos Utilizados para el Desarrollo del Proyecto

Herramientas

- Apache Cassandra: Seleccionada por su modelo de datos flexible y su capacidad para escalar horizontalmente, facilitando el manejo de grandes cantidades de datos distribuidos.
- Docker: Utilizado para contenerizar el entorno de desarrollo y producción, asegurando la consistencia entre diferentes entornos y simplificando el proceso de despliegue.
- Python: Elegido por su sintaxis clara y su extenso ecosistema de librerías, incluyendo el soporte para interactuar con Cassandra a través del driver oficial de DataStax.

Diseño e Implementación

- Diseño de la Base de Datos: Se tomó el análisis del laboratorio 2 para definir las tablas y las relaciones, optimizando el esquema para las consultas más frecuentes y garantizando un acceso eficiente a los datos.
- Contenerización y Despliegue: Se creó un archivo Dockerfile para Cassandra y se utilizó Docker Compose para manejar la aplicación y la base de datos como servicios conectados.
- Desarrollo de Scripts en Python: Se implementaron scripts para la creación de tablas, inserción de datos de prueba y ejecución de consultas, practicando la interacción programática con la base de datos.

Automatización y Pruebas

- Se desarrollaron scripts para automatizar la inicialización de la base de datos, la inserción de datos y la ejecución de consultas, facilitando la repetición de pruebas y la demostración de funcionalidades.

Creación de tablas

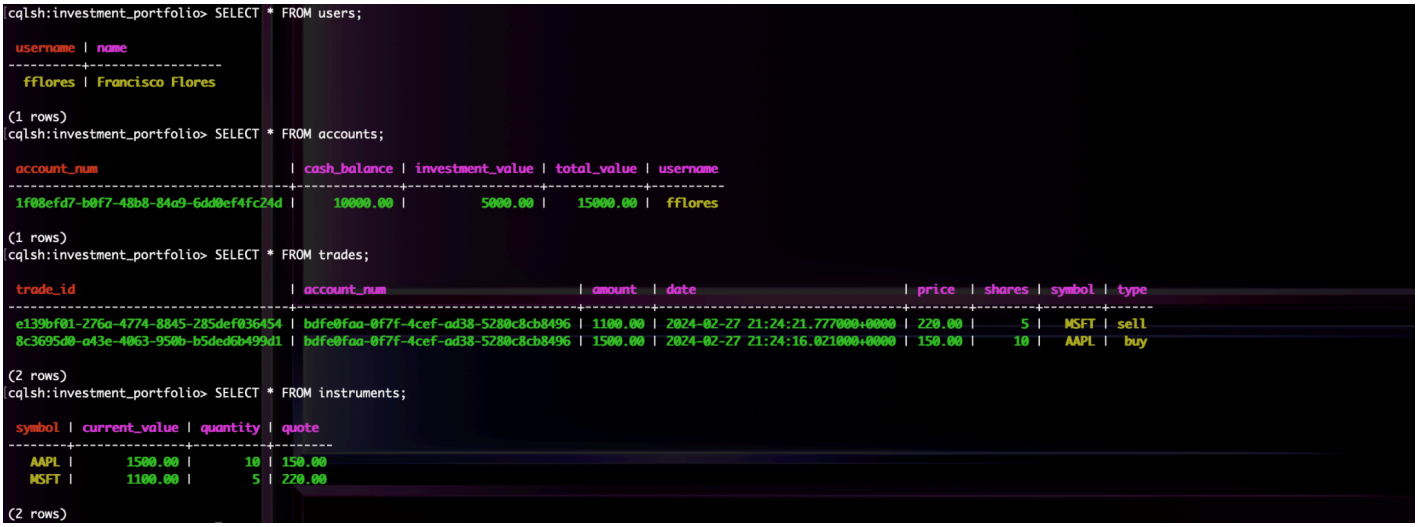
```
20  # Creación de tablas
21  session.execute("""
22  CREATE TABLE IF NOT EXISTS users (
23      username text PRIMARY KEY,
24      name text
25  )
26  """)
27
28  session.execute("""
29  CREATE TABLE IF NOT EXISTS accounts (
30      account_num uuid PRIMARY KEY,
31      username text,
32      cash_balance decimal,
33      investment_value decimal,
34      total_value decimal
35  )
36  """)
37
38  session.execute("""
39  CREATE TABLE IF NOT EXISTS trades (
40      trade_id uuid PRIMARY KEY,
41      account_num uuid,
42      type text,
43      symbol text,
44      shares int,
45      price decimal,
46      amount decimal,
47      date timestamp
48  )
49  """)
50
51  session.execute("""
52  CREATE TABLE IF NOT EXISTS instruments (
53      symbol text PRIMARY KEY,
54      quote decimal,
55      current_value decimal,
56      quantity int
57  )
58  """)
```

Inserción de datos

```python
# Inserción de datos de prueba
# Usuario
username = 'fflores'
session.execute(
    """
    INSERT INTO users (username, name) VALUES (%s, %s)
    """,
    (username, 'Francisco Flores')
)

# Cuenta
account_num = uuid4()
session.execute(
    """
    INSERT INTO accounts (account_num, username, cash_balance, investment_value, total_value) VALUES (%s, %s, 10000, 5000, 15000)
    """,
    (account_num, username)
)

# Trades
trade1_id = uuid4()
session.execute(
    """
    INSERT INTO trades (trade_id, account_num, type, symbol, shares, price, amount, date) VALUES (%s, %s, 'buy', 'AAPL', 10, 150.0, 1500.0, %s)
    """,
    (trade1_id, account_num, datetime.datetime.now())
)

trade2_id = uuid4()
session.execute(
    """
    INSERT INTO trades (trade_id, account_num, type, symbol, shares, price, amount, date) VALUES (%s, %s, 'sell', 'MSFT', 5, 220.0, 1100.0, %s)
    """,
    (trade2_id, account_num, datetime.datetime.now())
)

# Instrumentos
session.execute(
    """
    INSERT INTO instruments (symbol, quote, current_value, quantity) VALUES ('AAPL', 150.0, 1500.0, 10)
    """
)

session.execute(
    """
    INSERT INTO instruments (symbol, quote, current_value, quantity) VALUES ('MSFT', 220.0, 1100.0, 5)
    """
)
```

Resultados

El laboratorio permitió explorar de manera práctica las características distintivas de las bases de datos NoSQL, específicamente Apache Cassandra, en un entorno contenerizado. La implementación de scripts en Python para manipular la base de datos demostró ser una herramienta eficaz para la gestión de datos no relacionales, ofreciendo insights sobre el modelado de datos, la escalabilidad y la consistencia eventual. A través de la automatización, se mejoró la eficiencia del proceso de desarrollo y se facilitó la realización de pruebas consistentes.



```
cqlsh:investment_portfolio> SELECT * FROM users;

 username | name
----------+-----------------
   fflores | Francisco Flores

(1 rows)
cqlsh:investment_portfolio> SELECT * FROM accounts;

 account_num                          | cash_balance | investment_value | total_value | username
--------------------------------------+--------------+------------------+-------------+----------
 1f08efd7-b0f7-48b8-84a9-6dd0ef4fc24d |     10000.00 |          5000.00 |    15000.00 |  fflores

(1 rows)
cqlsh:investment_portfolio> SELECT * FROM trades;

 trade_id                             | account_num                          | amount  | date                            | price  | shares | symbol | type
--------------------------------------+--------------------------------------+---------+---------------------------------+--------+--------+--------+------
 e139bf01-276a-4774-8845-285def036454 | bdfe0faa-0f7f-4cef-ad38-5280c8cb8496 | 1100.00 | 2024-02-27 21:24:21.777000+0000 | 220.00 |      5 |   MSFT | sell
 8c3695d0-a43e-4063-950b-b5ded6b499d1 | bdfe0faa-0f7f-4cef-ad38-5280c8cb8496 | 1500.00 | 2024-02-27 21:24:16.021000+0000 | 150.00 |     10 |   AAPL |  buy

(2 rows)
cqlsh:investment_portfolio> SELECT * FROM instruments;

 symbol | current_value | quantity | quote
--------+---------------+----------+--------
   AAPL |       1500.00 |       10 | 150.00
   MSFT |       1100.00 |        5 | 220.00

(2 rows)
```

En la imágen se puede observar la impresión de las tablas con los datos agregados.

Consultas

- Q1. Consulta de cuentas de un usuario

```
SELECT * FROM accounts WHERE username = 'user2' ALLOW FILTERING;
```

	account_num	cash_balance	investment_value	total_value	username
1	dfe4017b-b7fc-4fea-be76-7c2144207fbf	20000	15000	35000	user2

❖ Q2. Consulta del balance de cada cuenta.

```
SELECT cash_balance, investment_value, total_value FROM accounts WHERE
account_num = dfe4017b-b7fc-4fea-be76-7c2144207fbf ALLOW FILTERING;
```

	cash_balance	investment_value	total_value
1	20000	15000	35000

❖ Q3.1 y 3.2 Consultas de todas las transacciones (con opción de rango de fechas):

```
SELECT * FROM trades WHERE account_num =
dfe4017b-b7fc-4fea-be76-7c2144207fbf
                    AND date >= '2024-02-28 04:27:30.140'
                    AND date <= '2024-02-29 01:00:26.870' allow
filtering;
```

	account_num	date	trade_id	amount	price	shares	symbol	type
1	dfe4017b-b7fc-4fea-be76-7c2144207fbf	2024-02-29 01:00:26.870	5b69a93d-dd0d-4264-aa4f-f5a777635b7b	14400	1809	2	GOOGL	buy
2	dfe4017b-b7fc-4fea-be76-7c2144207fbf	2024-02-28 04:27:30.162	ddaca31f-19d0-4650-927e-76220e2cb642	12400	3100	4	AMZN	sell
3	dfe4017b-b7fc-4fea-be76-7c2144207fbf	2024-02-28 04:27:30.140	70747f0e-fb3d-4c84-b8f7-4681ff987ead	14400	1800	8	GOOGL	buy

❖ Q3.2 Consulta de transacciones filtrando el tipo de transacción (buy/sell). (Con opción de rango de fechas)

```
SELECT * FROM trades WHERE account_num =
dfe4017b-b7fc-4fea-be76-7c2144207fbf
                    AND date >= '2024-02-28 04:27:30.140'
                    AND date <= '2024-02-29 01:00:26.870'
                    AND type = 'buy' allow filtering;
```

	account_num	date	trade_id	amount	price	shares	symbol	type
1	dfe4017b-b7fc-4fea-be76-7c2144207fbf	2024-02-29 01:00:26.870	5b69a93d-dd0d-4264-aa4f-f5a777635b7b	14400	1809	2	GOOGL	buy
2	dfe4017b-b7fc-4fea-be76-7c2144207fbf	2024-02-28 04:27:30.140	70747f0e-fb3d-4c84-b8f7-4681ff987ead	14400	1800	8	GOOGL	buy

❖ Q3.3 Consulta de transacciones filtrando el tipo de transacción e instrumento de inversión. (Con opción de rango de fechas)

```
SELECT * FROM trades WHERE account_num =
dfe4017b-b7fc-4fea-be76-7c2144207fbf
                        AND date >= '2024-02-28 04:27:30.140'
                        AND date <= '2024-02-29 01:00:26.870'
                        AND type = 'sell'
                        AND symbol = 'AMZN'ALLOW FILTERING;
```

account_num	date	trade_id	amount	price	shares	symbol	type
dfe4017b-b7fc-4fea-be76-7c2144207fbf	2024-02-28 04:27:30.162	ddaca31f-19d0-4650-927e-76220e2cb642	12400	3100	4	AMZN	sell

❖ Q3.4 Consulta de transacciones filtrando el instrumento de inversión. (Con opción de rango de fechas)

```
SELECT * FROM trades WHERE account_num =
dfe4017b-b7fc-4fea-be76-7c2144207fbf
                        AND date >= '2024-02-28 04:27:30.140'
                        AND date <= '2024-02-29 01:00:47.139'
                        AND symbol = 'AMZN'ALLOW FILTERING;
```

account_num	date	trade_id	amount	price	shares	symbol	type
dfe4017b-b7fc-4fea-be76-7c2144207fbf	2024-02-29 01:00:47.139	b9f80b56-3e37-4a67-b408-cee8aa5a9982	12400	3110	9	AMZN	sell
dfe4017b-b7fc-4fea-be76-7c2144207fbf	2024-02-28 04:27:30.162	ddaca31f-19d0-4650-927e-76220e2cb642	12400	3100	4	AMZN	sell

Así se verían las consultas en el script de Python

```python
#Consultas
def query_accounts_by_username(username):
    query = "SELECT * FROM accounts WHERE username = %s ALLOW FILTERING;"
    rows = session.execute(query, (username,))
    for row in rows:
        print(row)

def query_balance_by_account_uuid(account_uuid):
    query = "SELECT cash_balance, investment_value, total_value FROM accounts WHERE account_num = %s;"
    rows = session.execute(query, (UUID(account_uuid),))  # Conversión de string a UUID
    for row in rows:
        print(row)

def query_trades_by_date_range(account_uuid, start_date, end_date):
    query = """
    SELECT * FROM trades WHERE account_num = %s
    AND date >= %s AND date <= %s ALLOW FILTERING;
    """
    rows = session.execute(query, (UUID(account_uuid), start_date, end_date))
    for row in rows:
        print(row)

def query_type_trades_by_date_range(account_uuid, start_date, end_date):
    query = """
    SELECT * FROM trades WHERE account_num = %s
    AND date >= %s AND date <= %s AND type = 'buy' ALLOW FILTERING;
    """
    rows = session.execute(query, (UUID(account_uuid), start_date, end_date))
    for row in rows:
        print(row)

def query_type_instrument_trades_by_date_range(account_uuid, start_date, end_date):
    query = """
    SELECT * FROM trades WHERE account_num = %s
    AND date >= %s AND date <= %s AND type = 'buy' AND symbol = 'AMZN'
    ALLOW FILTERING;
    """
    rows = session.execute(query, (UUID(account_uuid), start_date, end_date))
    for row in rows:
        print(row)

def query_instrument_trades_by_date_range(account_uuid, start_date, end_date):
    query = """
    SELECT * FROM trades WHERE account_num = %s
    AND date >= %s AND date <= %s AND symbol = 'AMZN' ALLOW FILTERING;
    """
    rows = session.execute(query, (UUID(account_uuid), start_date, end_date))
    for row in rows:
        print(row)
```

Y su ejecución

```
account_uuid = 'dfe4017b-b7fc-4fea-be76-7c2144207fbf'
start_date = '2024-02-28 04:27:30.140000'
end_date = '2024-02-29 01:00:47.139000'

# Conversión de string a datetime
start_date_dt = datetime.datetime.strptime(start_date, '%Y-%m-%d %H:%M:%S.%f')
end_date_dt = datetime.datetime.strptime(end_date, '%Y-%m-%d %H:%M:%S.%f')

# Ejecución las consultas
query_accounts_by_username('user2')
query_balance_by_account_uuid(account_uuid)
query_trades_by_date_range(account_uuid, start_date_dt, end_date_dt)
query_type_trades_by_date_range(account_uuid, start_date_dt, end_date_dt)
query_type_instrument_trades_by_date_range(account_uuid, start_date_dt, end_date_dt)
query_instrument_trades_by_date_range(account_uuid, start_date_dt, end_date_dt)

# Cerrar la conexión
cluster.shutdown()

print("Termino correctamente")
```

Conclusión

El desarrollo de este proyecto en el contexto de la materia de Bases de Datos No Relacionales me dejó una valiosa oportunidad para aplicar teorías y conceptos a un escenario práctico, profundizando en el entendimiento de las bases de datos NoSQL como Apache Cassandra. La experiencia adquirida en la contenerización con Docker y la automatización con Python no solo mejoró mi habilidad usándolo sino que también me dejó perspectivas sobre el diseño y gestión de bases de datos modernas. Este proyecto marca la importancia de combinar herramientas y tecnologías actuales para enfrentar los desafíos de manejo de datos en grandes volúmenes y distribuidos, un conocimiento esencial para el desarrollo de aplicaciones en el mundo real.