UNIVERSIDAD DE GRANADA

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

Trabajo del Tema 4. Otros modelos de datos para Sistemas de Información

Asignatura	Diseño y Desarrollo de Sistemas de Información
Grupo Pequeño (prácticas y seminarios)	A1
Grupo de trabajo	A1_ddsimola:D
Integrantes	GUIRADO BAUTISTA, LUIS MIGUEL, IRIGOYEN CORTADI, PABLO, SERRANO VILLENA, MIGUEL ÁNGEL, ZHU, LINQI

1. Breve descripción de la descarga e instalación del SGBD.

➢ **Instalación para un sistema GNU/Linux que utiliza paquetes '.deb' (debian):**

Agregue el repositorio Apache de Cassandra al archivo cassandra.sources.list. La última versión principal es 4.0 y el nombre de distribución correspondiente es 40x(con una "x" como sufijo). Para versiones anteriores, use 311xpara la serie C* 3.11, 30xpara {30_version}, 22x para {22_version} y 21xpara {21_version}. Por ejemplo, para agregar el repositorio para la versión 4.0 (40x):

```
echo "deb https://debian.cassandra.apache.org 41x main" | sudo tee -a
/etc/apt/sources.list.d/cassandra.sources.list
```

Agregue las claves del repositorio de Apache Cassandra a la lista de claves confiables en el servidor:

```
curl https://downloads.apache.org/cassandra/KEYS | sudo apt-key add -
```

Actualice el índice del paquete desde las fuentes:

```
sudo apt-get update
```

Instale Cassandra con APT:

```
sudo apt-get install cassandra
```

Para conectar a la base de datos: `cqlsh`

Para salir de la terminal de texto de Cassandra: `quit`

➢ **Otra opción de instalación: sistema MacOS:**

```
$ brew install cassandra
```

```
$ brew service start cassandra
```

```
$ cqlsh
```

```
$ brew services stop cassandra
```

2. Breve descripción del DDL y DML utilizado.

	SENTENCIAS SQL	SENTENCIAS NoSQL
DDL	CREATE TABLE <name> (<name> <type> [<constraint>], …);	CREATE TABLE [IF NOT EXISTS] <name> (column cql_type, column cql_type, column cql_type, PRIMARY KEY(column, column)) [WITH property = value AND property = value];
	ALTER TABLE <table_name> <modifier>;	ALTER TABLE ALTER TABLE <name> DROP column ALTER TABLE <name> DROP (column, column);
	DROP TABLE <table_name>;	DROP TABLE drop table <name>;
DML	INSERT INTO <table_name> [(column1, column2, …)] VALUES (value1, value2, …);	INSERT INTO <name>(column, column) VALUES(value, value) [USING TTL seconds];
	DELETE [FROM] <table_name> [WHERE <condition>];	DELETE FROM <name> WHERE <condition>;
	UPDATE <table_name> SET <name_field> = ‘new_value’ … [WHERE <condition>];	UPDATE <tableName> [USING TTL seconds] SET column = value, column = value WHERE <condition>;
	SELECT [<field1>,<field2>, … / *] FROM <table_name>;	SELECT column, column FROM <name> WHERE <condition>;

3. Sentencias empleadas para la creación de estructuras, inserción/modificación/borrado de datos, y consultas.

1. Creación de keyspace: es un conjunto de columnas (tabla), índice y otros contenedores, similar al *database* en mysql.

```
$ CREATE KEYSPACE test WITH replication = {'class':
'SimpleStrategy', 'replication_factor' : 3}
$ use test
```

2. Creación de tabla:

```
$ create table IF NOT EXISTS users (
     id bigint primary key,
     username text,
     ) with comment = ‘user info table’;
```

3. Añadir una columna en tabla users:

```
$ alter table users add temp varchar;
borrar una columna
$ alter table users drop temp;
insert
$ insert into users( id, username)values(1, ‘ddsi’);
```

4. Consultas:

```
$ describe table users;
$ select * from users;
```

4. Breve descripción del mecanismo de conexión al SGBD desde una aplicación.

```
from cassandra.cluster import Cluster

# Conecta a Cassandra en localhost:
cluster = Cluster()
session = cluster.connect()

# Conecta a Cassandra en un host específico:
cluster = Cluster(['hostname'])
session = cluster.connect()
```

```
# Ejecuta una consulta
rows = session.execute("SELECT * FROM test.users")
# Procesa el resultado
for row in rows:
    id = row[0]
    value = row[1]
    print("id: {}, value: {}".format(id, value))
```

Otro ejemplo:

```
cluster = Cluster (
    # Completar aquí la dirección del punto de conexión de la base de datos (red pública o intranet), ¡no complete el puerto!
    contact_points=["127.0.0.1"] )
session = cluster.connect()

# Cerrar la sesión:
session.shutdown()

# Cerrar el cluster:
cluster.shutdown()
```

Para instalar la dependencia:

```
pip install cassandra-driver
```

5. Breve discusión sobre si sería adecuado para implementar el SI de la práctica.

Es completamente posible implementar un sistema de información utilizando tanto bases de datos SQL como no SQL. La elección de un tipo de base de datos depende de las necesidades y requisitos del sistema de información en cuestión. En nuestro caso es adecuado implementar con un noSQL pues nos permite una mayor flexibilidad de diseño:

Los SGBD noSQL permiten una mayor flexibilidad en la estructuración de los datos, lo que puede ser útil en situaciones en las que los requisitos de la aplicación pueden cambiar con el tiempo.

Escalabilidad horizontal: Los SGBD noSQL son buenos para manejar grandes cantidades de datos y tráfico, ya que permiten la distribución de los datos y la carga de trabajo a través de múltiples máquinas. Y nuestro SI necesita guardar a los usuarios y sus trayectos, etc… Por tanto es una gran cantidad de datos.

Alta disponibilidad: Los SGBD noSQL suelen ofrecer alta disponibilidad y tolerancia a fallos, lo que significa que son menos propensos a fallar y pueden recuperarse rápidamente en caso de fallo. Lo que en nuestro caso nos beneficiaría.

6. Creación de tablas noSQL de nuestro SI.

Antes de crear las tablas, es necesario crear el contenedor para las mismas, que es lo que se conoce como **keyspace**. Este no solo tiene como funcionalidad ser el contenedor de las tablas, sino que es el encargado de definir el factor y estrategia de replicación. Vamos entonces a crear nuestro primer keyspace mediante la consola de comandos; para ello, se ejecuta lo siguiente:

```
CREATE KEYSPACE ddsimola WITH replication = { 'class':
'SimpleStrategy', 'replication_factor' : 1};
```

La creación de tablas dentro de **Cassandra** es bastante similar a cuando lo hacemos en modelos relacionales, sin embargo tiene algunas diferencias. Lo primero que hay que hacer y, para ahorrarnos tener que indicar el **keyspace,** es usar la palabra reservada **USE**:

```
USE ddsimola;
```

Al utilizar **USE** todo lo que hagamos de ahora en adelante será dentro de ese *keyspace*, por lo que no debemos preocuparnos por indicarlo, y en la consola de comandos nos debe especificar el **keyspace** en el cual estamos. Ya con esto último verificado, vamos a crear dos tablas de ejemplo para nuestro SI:

- Tabla 1:

```
CREATE TABLE Control_usuarios ( usuario VARCHAR, fecha
TIMESTAMP, PRIMARY KEY (usuario, fecha) );
```

Esta tabla sería interesante para controlar los usuarios que añaden un trayecto como favorito. Esta información puede ser interesante por varios motivos. Entre ellos, cabe mencionar: llevar el recuento del número de trayectos guardados por cada usuario, de cara a establecer una restricción a tal efecto, llevar la cuenta de la fecha de cara a controlar un número máximo de días que el usuario puede tener un trayecto almacenado, etc. Se podría crear un disparador para que utilizase esta tabla para el registro tras las inserciones de los trayectos en el sistema (tabla Usa).

- Para consultar todas las tuplas (sin proyecciones):

```
SELECT * FROM Control_usuarios;
```

- Para realizar una inserción:

```
INSERT INTO Control_usuarios (usuario, fecha) VALUES
('123456', '2023-01-13 23:22:00');
```

❖ Tabla 2:

```
CREATE TABLE Planta_reciclaje ( id_planta_rec VARCHAR,
tipo_res TEXT, capacidad FLOAT, PRIMARY KEY (id_planta_rec) );
```

Esta tabla sería interesante para incorporar también las plantas de reciclaje a nuestro SI, lo que complicaría un poco más el diseño, aunque tendría sentido esta adición puesto que desde la planta de almacenamiento se envían los residuos, según el tipo, a las diferentes plantas de reciclaje.

- Para consultar todas las tuplas (sin proyecciones):

```
SELECT * FROM Planta_reciclaje;
```

- Para realizar una inserción:

```
INSERT INTO Envio_Planta_reciclaje (id_envio, id_planta_alm,
id_planta_rec, fecha_env, fecha_rec, cantidad, tipo_res)
VALUES ('490321', '589320', '440125', '2023-01-14 14:22:00',
'2023-01-14 17:18:00', 5600, 'Plástico');
```

- Para actualizar una tupla:

```
UPDATE Planta_reciclaje SET capacidad = 30000.25 WHERE
id_planta_rec IN ('389023');
```

- Para eliminar una tupla:

```
DELETE FROM Planta_reciclaje WHERE id_planta_rec IN
('389023');
```

- Para eliminar toda la tabla:

  ```
  DROP TABLE Planta_reciclaje;
  ```

- Para eliminar el keyspace:

  ```
  DROP KEYSPACE ddsimola;
  ```