Bases de Datos Orientada a Columnas: Cassandra

Máster en Business Analytics y Big Data

****

**Asignatura**: Bases de Datos No Convencionales

**Módulo**: Tecnologías de Big Data/Gestión de Datos

**Coordinador**: Elena García Barriocanal, [elenagarcia@campusciff.net](mailto:elenagarcia@campusciff.net)

**Profesores:** Jordi Conesa [jordiconesa@campusciff.net](mailto:jordiconesa@campusciff.net)

Elena García Barriocanal, [elenagarcia@campusciff.net](mailto:elenagarcia@campusciff.net)

Contenido

[Arrancar el servicio de Cassandra 2](#_Toc412042077)

[Parar el servicio Cassandra 2](#_Toc412042078)

[Conexión a Shell de Cassandra 2](#_Toc412042079)

[Crear una base de datos 2](#_Toc412042080)

[Creación de tablas (familias de columnas) 2](#_Toc412042081)

[Creación de columnas *contadores* 4](#_Toc412042082)

[Inserción de datos en tablas 4](#_Toc412042083)

[Modificación de tablas 4](#_Toc412042084)

[Columnas “colección” 5](#_Toc412042085)

[Crear índices 6](#_Toc412042086)

[Consulta de filas 6](#_Toc412042087)

[Eliminar tablas 7](#_Toc412042088)

[Eliminar la base de datos 7](#_Toc412042089)

[Documentación 7](#_Toc412042090)

# Arrancar el servicio de Cassandra

sudo service cassandra start

# Parar el servicio Cassandra

sudo service cassandra stop

# Conexión a Shell de Cassandra

cqlsh

# Crear una base de datos

En el momento de la creación hay que establecer valor para los atributos:

* Replica placement strategy:
  + SimpleStrategy: La primera réplica se sitúa en el nodo indicado y las siguientes en los nodos colindantes del anillo siguiendo las agujas del reloj.
  + NetworkTopologyStrategy: Situa un número de réplicas concretos en distintos data centers.
* Replication factor: número de nodos que contendrán réplicas de cada fila.

CREATE KEYSPACE tweetssandra WITH REPLICATION = { 'class' : 'SimpleStrategy', 'replication\_factor' : '1' };

DESCRIBE KEYSPACES;

SELECT \* FROM system.schema\_keyspaces;

USE tweetsssandra;

DESCRIBE TABLES;

# Creación de tablas (familias de columnas)

CREATE TABLE users (

username text PRIMARY KEY,

password text

);

CREATE TABLE friends (

username text,

friend text,

since timestamp,

PRIMARY KEY (username, friend)

);

CREATE TABLE followers (

username text,

follower text,

since timestamp,

PRIMARY KEY (username, follower)

);

Las tablas followers y friends tienen claves compuestas. Se crean especificando dos columnas como *primary key* (clave primaria compuesta):

* La clave de partición controla cómo los datos se dispersan en el cluster.
* La clave de agrupación controla como se ordenan los datos en el disco: se crea un índice para esa columna y los datos se guardan para la clave de partición agrupados por ese índice.

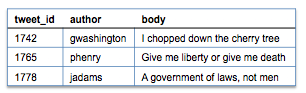
CREATE TABLE tweets (

tweet\_id uuid PRIMARY KEY,

username text,

body text

);



CREATE TABLE userline (

username text,

time timeuuid,

body text,

tweet\_id uuid,

PRIMARY KEY (username, time)

) WITH CLUSTERING ORDER BY (time DESC);

CREATE TABLE timeline (

user\_id text,

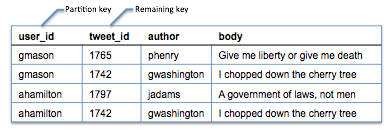
tweet\_id uuid,

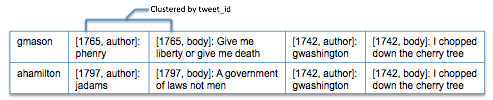
author text,

body text,

PRIMARY KEY (user\_id, tweet\_id)

);





SELECT \* FROM timeline WHERE user\_id = gmason

ORDER BY tweet\_id DESC LIMIT 50;

# Creación de columnas *contadores*

CREATE TABLE page\_view\_counts

(counter\_value counter,

url\_name varchar,

page\_name varchar,

PRIMARY KEY (url\_name, page\_name)

);

UPDATE page\_view\_counts

SET counter\_value = counter\_value + 1

WHERE url\_name='www.miPagina.com' AND page\_name='home';

# Inserción de datos en tablas

INSERT INTO followers (username, follower, since)

VALUES ('gwashington', 'gmason', '2013-01-28 15:30');

INSERT INTO followers (username, follower, since)

VALUES ('gwashington', 'ahamiltom', '2013-01-27 21:15');

# Modificación de tablas

ALTER TABLE users ADD name text;

INSERT INTO users (username, password, name)

VALUES ('egarcia', 'pswegarcia', 'Eva Garcia');

INSERT INTO users (username, password)

VALUES ('gmason', 'pswgmason');

UPDATE users SET name = 'George Mason' WHERE username='gmason';

En versiones posteriores a Cassandra 2.0 se pueden crear tipos definidos por el usuario

CREATE TYPE fullname (

firstname text,

lastname text

);

ALTER TABLE users ADD name frozen <fullname>;

INSERT INTO users (username, password, name)

VALUES ('ahamilton', 'pswahamilton', {firstname: 'Albert', lastname: 'Hamilton'});

INSERT INTO users (username, password)

VALUES ('gmason', 'pswgmason');

UPDATE users SET name = {firstname: 'George', lastname: 'Mason'} WHERE username='gmason';

# Columnas “colección”

Se pueden crear 3 tipos de columnas de colección:

* Set: Una columna de este tipo contiene un conjunto de valores únicos no ordenados.

CREATE TABLE users (

user\_id text PRIMARY KEY,

first\_name text,

last\_name text,

emails set<text>

);

INSERT INTO users (user\_id, first\_name, last\_name, emails)

VALUES('frodo', 'Frodo', 'Baggins', {'f@baggins.com', 'baggins@gmail.com'});

UPDATE users SET emails = emails + {'fb@friendsofmordor.org'} WHERE user\_id = 'frodo';

UPDATE users SET emails = emails - {'fb@friendsofmordor.org'} WHERE user\_id = 'frodo';

UPDATE users SET emails = {} WHERE user\_id = 'frodo';

DELETE emails FROM users WHERE user\_id = 'frodo';

Modificar la tabla de usuarios para añadir varias direcciones de correo electrónico y comprobar la modificación.

* List: Una columna de este tipo contiene un conjunto ordenado de valores, que pueden estar repetidos.

ALTER TABLE users ADD top\_places list<text>;

UPDATE users SET top\_places = [ 'rivendell', 'rohan' ] WHERE user\_id = 'frodo';

UPDATE users SET top\_places = [ 'the shire' ] + top\_places WHERE user\_id = 'frodo';

UPDATE users SET top\_places = top\_places + [ 'mordor' ] WHERE user\_id = 'frodo';

UPDATE users SET top\_places[2] = 'riddermark' WHERE user\_id = 'frodo';

DELETE top\_places[3] FROM users WHERE user\_id = 'frodo';

UPDATE users SET top\_places = top\_places - ['riddermark'] WHERE

user\_id = 'frodo';

Modificar la tabla de usuarios para añadir varios temas de interés ordenados descendentemente por orden de interés.

* Map: Una columna de este tipo contiene pares de datos.

ALTER TABLE users ADD todo map<timestamp, text>;

INSERT INTO users (user\_id, todo) VALUES ('frodo', {'2013-9-22

12:01' : 'birthday wishes to Bilbo', '2013-10-1 18:00': 'Check

into Inn of Pracing Pony'}) ;

UPDATE users SET todo = {'2012-9-24': 'enter mordor',

'2014-10-2 12:00':'throw ring into mount doom'}

WHERE user\_id = 'frodo';

UPDATE users SET todo = todo + {'2013-9-22 12:01' : 'birthday

wishes to Bilbo', '2013-10-1 18:00': 'Check into Inn of

Pracing Pony'}

WHERE user\_id='frodo';

UPDATE users SET todo['2014-10-2 12:00'] = 'throw my precious

into mount doom'

WHERE user\_id = 'frodo';

DELETE todo['2013-9-22 12:01'] FROM users WHERE user\_id='frodo';

UPDATE users SET todo=todo - {'2013-9-22 12:01','2013-10-01

18:00'} WHERE user\_id='frodo';

Modificar la tabla de usuarios para añadir una colección de otros usuarios favoritos donde se indique el username del usuario favorito y en qué momento pasó a serlo.

Las colecciones son útiles cuando se quiere desnormalizar una pequeña cantidad de datos.

# Crear índices

CREATE INDEX ON tweets (username);

En Cassandra 2.1 se pueden indexar colecciones.

# Consulta de filas

SELECT \* FROM timeline WHERE user\_id = gmason

ORDER BY tweet\_id DESC LIMIT 50;

Para consultar filas en columnas de algún tipo “colección” se puede utilizar la condición CONTAINS en la cláusula WHERE:

CREATE TABLE users (

user\_id text PRIMARY KEY,

first\_name text,

last\_name text,

emails set<text>

top\_places list<text>

);

CREATE INDEX ON users (top\_places); 🡨 para versiones de Cassandra posteriores a la 2.1

SELECT first\_name, last\_name, emails FROM users WHERE top\_places CONTAINS 'the shire';

# Eliminar tablas

DELETE name FROM users WHERE username='egarcia';

DELETE FROM users WHERE username='egarcia';

DROP TABLE users;

# Eliminar la base de datos

DROP KEYSPACE tweetssandra;

# Documentación

<http://www.datastax.com/documentation/cassandra/2.1/cassandra/cql.html>