Manual de usuario de Apache Cassandra

Miguel Ignacio Rodriguez Suarez.

Contenido

[Introducción 3](#_Toc5964536)

[Arquitectura y características de Cassandra 4](#_Toc5964537)

[Componentes de Cassandra 4](#_Toc5964538)

[Seguridad 5](#_Toc5964539)

[Modelo de Datos de Cassandra 5](#_Toc5964540)

[Comparación entre base de datos relacional y Cassandra 9](#_Toc5964541)

[Lenguaje CQL 9](#_Toc5964542)

[Consultas CQL 9](#_Toc5964543)

[Crear, alterar y eliminar un Keyspace 10](#_Toc5964544)

[Crear, alterar y eliminar una Tabla 11](#_Toc5964545)

[Tipos de Datos en las Tablas 11](#_Toc5964546)

[Insertar y Actualizar Datos 13](#_Toc5964547)

[Eliminación de datos 14](#_Toc5964548)

[Creación y eliminación de Índices 14](#_Toc5964549)

[TRIGGERS 14](#_Toc5964550)

[USERS 14](#_Toc5964551)

[Type 15](#_Toc5964552)

[Cliente remoto 15](#_Toc5964553)

# Introducción

Cassandra es una base de datos de software libre ya que pertenece a la Apache software foundation. Es una base de datos tipo NoSQL (El concepto de NoSQL se explicará más adelante) distribuida y basada en un modelo de almacenamiento de “Clave-Valor”, está escrita en Java por ello es necesario de su uso para que funcione. Permite manejar con grandes volúmenes de datos de forma distribuida. Por ello su uso en el ámbito de Big Data es muy recomendable ya que Cassandra es capaz de manejar un tamaño de datos máximo de 2 billones de registros y en el caso de tener que manejar con más registros, se tienen que particionar para poder trabajar con ellos.

# Arquitectura y características de Cassandra

Cassandra nos proporciona tolerancia a fallos y disponibilidad, pero a cambio de ser eventualmente consistente, tal y como se define en el teorema CAP (ya que la actividad de inserción de datos no es su fuerte, si no su consulta de información de forma rápida).El nivel de consistencia puede ser configurado, según nos interese incluso a nivel de query, dicha configuración está repartida de la siguiente forma:

* Escala Distribuida: lo que quiere decir que la información está repartida a lo largo de los nodos del cluster. Además ofrece alta disponibilidad, de manera que si alguno de lo nodos se cae el servicio no se degradará.
* Escala linealmente: lo que quiere decir que el rendimiento de forma lineal respecto al número de nodos que añadamos. Por ejemplo, si con 2 nodos soportamos 100.000 operaciones por segundo, con 4 nodos soportaremos 200.000. Esto da mucha predictibilidad a nuestros sistemas.
* Escala de forma horizontal: lo que quiere decir que podemos escalar nuestro sistema añadiendo nuevos nodos basados en hardware commodity de bajo coste.

Implementa una arquitectura Peer-to-Peer, lo que quiere decir es que todos los nodos tienen la misma importancia, por lo cual el patrón de maestro-esclavo es inexistente, así que ningún nodo tendrá alguna información de más con respecto al resto. De esta forma cualquier nodo puede tomar el rol de coordinador de una query. Será el driver el que decida qué nodo quiere que sea el coordenador (esta gestión la realiza internamente cassandra).

# Componentes de Cassandra

Una vez que ya sabemos la arquitectura que tiene Cassandra y como funciona internamente, además de las posibilidades que nos brinda vamos a conocer de forma resumida cómo está compuesta para así ser consciente en todo momento qué componentes tiene esta base de datos.

* Cluster: es un componente que contiene uno o más centro de datos (datacenter).
* Centro de datos (datacenter): se trata de una colección que almacena los nodos relacionados.
* Nodo: es el lugar donde se almacena los datos de Cassandra.
  + Commit log: es un fichero en donde se almacena la información sobre los cambios en los datos. Sirve para recuperar los datos en caso de una fallo en el sistema.
  + MemTable: estructura de almacenamiento en memoria. Contiene los datos que aún no han sido escritos en un SSTable.
  + SSTable: es un fichero que almacenan los datos escritos en disco. Cada fichero SSTable es inmutable (que no cambia) una vez creado.
* About internode communications (gossip): es el protocolo de comunicación peer-to-peer para descubrir y compartir información sobre la localización y estado de los nodos en un cluster de Cassandra.
* Partitioner: este componente determina cómo se distribuyen los datos entre los nodos (las copias entre los nodos).
* Replica placement strategy: define la estrategia a seguir para almacenar las copias de los mismo datos en diferentes nodos, de forma que se aseguren la accesibilidad y la tolerancia a fallos. Se pueden definir diferentes estrategias. Hay que tener en cuenta que no existen copias principales ni secundarias, todas las copias, incluida la primera, con réplicas.
* Snitch: define la topología que utilizan las estrategias de replicación para colocar las réplicas y dirigir las consultas de forma eficiente.

# Seguridad

La seguridad existente en Cassandra se encuentra a nivel de usuarios como logins con password y permisos de gestión administración vía GRANT/REVOKE, como es utilizada en los sistemas de bases de datos relacionales como por ejemplo ORACLE, a continuación aparece los privilegios que podemos gestionar y a los objetos que son aplicables.

|  |  |
| --- | --- |
| Privilegios | Objetos a los que se le aplican |
| All permissions | All functions |
| Alter | ALL FUNCTIONS IN KEYSPACES “nombre keyspace” |
| Authorize | FUNCTION “nombre de función” |
| Créate | ALL KEYSPACE |
| Describe | KEYSPACE “nombre keyspace” |
| Drop | TABLE “nombre tabla” |
| Execute | ALL ROLES |
| Modify | ROLE “nombre rol” |
| select |  |

A continuación, podremos ver un ejemplo de aplicar /revocar un privilegio:

Asignación de privilegio:

GRANT “privilegio a conceder” ON “objeto al que concede” TO “usuario /rol”;

Revocación de privilegios:

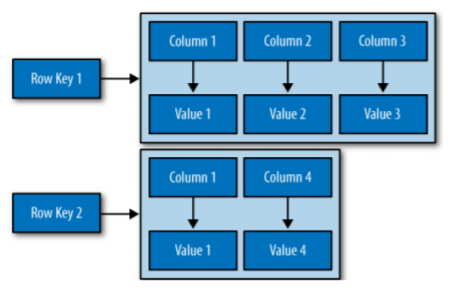
REVOKE "privilegio a revocar" ON "objeto al que revocar" FROM "usuario/rol";

# Modelo de Datos de Cassandra

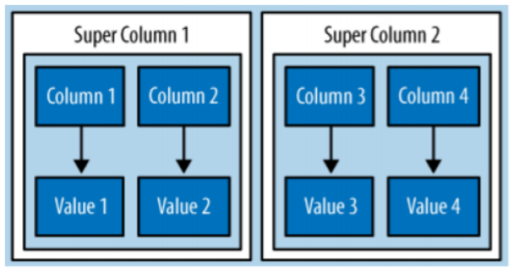
El modelo de datos que se utiliza en Cassandra es algo complejo de entender por cómo se organiza y reparte la información, como ya sabemos, la base de datos de Cassandra se organiza en un sistema de “Clave-Valor” enriquecido, lo que ofrece una visión totalmente diferente a lo que estamos acostumbrado a ver en los sistemas relacionales.

El modelado de datos existente en Cassandra está dividido en diferentes apartados, siendo los siguientes:

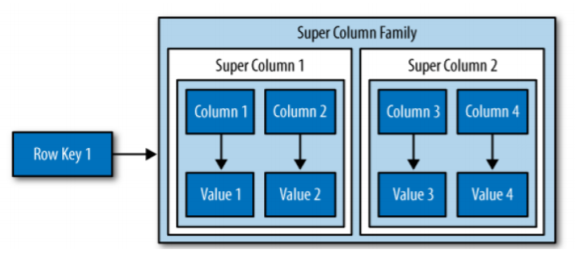
* Columna (Column): es el elemento básico del sistema, es una estructura de tres campos que contienen: el nombre de la columna, el valor de la columna y una marca de tiempo.
  + La clave y el valor se almacenan como datos binarios y el timestamp indica la última vez que se actualizó la columna.
  + Los tres valores son proporcionados por el cliente de alto nivel (por la aplicación), incluido el timestamp. Por ello es necesario tener sincronizado el reloj del sistema con el reloj del clúster.
  + El timestamp se utiliza para la resolución de conflictos y así diferencias cuando se introdujo un dato de otro.
  + El timestamp puede ser de cualquier tipo pero la convención marca que sea el valor en microsegundos.
  + Dicha columna tiene asociada las filas (rows) que es el lugar en donde va a almacenado el valor que hace referencia al nombre de columna.
  + Y la información almacenada que hace referencia a esas columnas se le aplica un hash para generar una key (denominada row-key) que diferencia ese dato con el resto para que en el momento en el que se realice la búsqueda de ese dato, sea capaz de encontrarlo.
  + La row-key es el equivalente a la clave primaria del modelo relacional. y está compuesta por dos partes:
    - Parte del particionado (partition key): el objetivo es identificar en la partición o el nodo en el que se encuentra esa fila/dato
    - Parte del agrupamiento (clustering key): determina el orden físico en el que se almacenan las filas.



* Súper Columna (SuperColumn): es un elemento que se encuentra compuesto por varias columnas (descrito anteriormente). El aspecto negativo que tiene la súper columna es que no aconsejan el uso de ella ya que afecta al rendimiento en la búsqueda de información ya que “desglosar” las columnas por lo cual tarda más en mostrar el resultado, así que por ello no se recomienda su usabilidad.



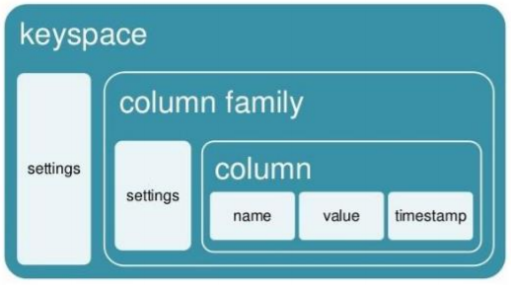
* Familia de Columnas (ColumnFamily): sería un equivalente a una tabla en un esquema de datos relacional. Se trata de un contenedor para una colección ordenada de columnas. Cada entrada (row) se identifica y accede a ella mediante una row-KEY .
  + Cada Familia de Columna es guardada en un fichero separado y es ordenado por la KEY.
  + Los tipos de columnas que almacena la Familia de Columnas es la siguiente:
    - Standard Columns
    - Counter Columns
    - Collection Columns: SET, LIST, MAP
    - User-defined type
    - Tuple-type
    - Timestamp type



* Espacio de clave: es un contenedor para la familia de columnas, más o menos el equivalente a una base de datos (schema) en el modelo relacional. Es una colección ordenada de una familia de columnas o columnas.
  + Los atributos básicos de un Keyspace o Espacio de Clave son:
    - Replication factor: cuánto quieres asignar en rendimiento a favor de consistencia.
    - Replica placement strategy: indica cómo se colocan las réplicas en el anillo:
      * SimpleStrategy (solo un centro de datos).
      * NetworkTopologyStrategy (varios centro de datos).
* Cassandra ofrece soporte para particionado distribuido de datos.
  + RandomPartitioner: ofrece un correcto balanceo de carga.
  + OrderPreservingPartitioner: nos permite ejecutar consultas de rangos, pero exista más trabajo eligiendo node tokens.

○ Además de ello Cassandra tiene consistencia reconfigurable.

* En el momento de escritura, consistency level determina en cuántas réplicas se debe escribir para confirmar a la aplicación cliente.
* Y al leer, también se especifica cuántas réplicas deben responder para retornar los datos a la aplicación cliente.



* Cluster: es un conjunto de máquinas o nodos que ejecutan Cassandra y proporcionan al sistema escalabilidad horizontal, de manera transparente para el cliente.

○ Los datos en Cassandra se guardan en el Clúster (“Ring”) donde se asignan datos a los nodos dentro de un anillo.

○ Un nodo tiene réplicas para diferentes rangos de datos.

○ Si un nodo se cae, su réplica puede responder.

○ El protocolo utilizado en Cassandra es P2P que hace que los datos se repliquen entre nodos acorde a un factor de replicación definido.

# Comparación entre base de datos relacional y Cassandra

A continuación, aparece una pequeña tabla en la cual se muestra las posibles similitudes entre Cassandra y los sistemas de bases de datos relacionales del mercado.

|  |  |
| --- | --- |
| Base de datos relacional | Cassandra |
| Base de datos | Keyspace(Espacio clave) |
| Tabla | Columna Family(Familia de columnas) |
| Clave primaria | Row key |

# Lenguaje CQL

Para poder interactuar con Cassandra mediante CQL tendremos que introducir a través de la sh de la propia base de datos el comando cqlsh, aunque también podemos utilizar herramientas gráficas o a través de los driver soportados por múltiples lenguajes de programación.

## Consultas CQL

Una de las particularidades que posee Cassandra con su lenguaje CQL es que de cara a las consultas que se realizan en ella se hacen de forma muy reducida en cuanto a la condicion de busqueda se refiere, de esta forma las consultas a realizar son del siguiente tipo ( las condiciones tienen que ser especificadas detrás del WHERE ):

* Podemos especificar cláusulas de igualdad a un objeto concreto, como por ejemplo:

SELECT \* WHERE Nombre = 'Pepito';

* Podemos especificar cláusulas de igualdad a varios objetos con el valor “IN”, como por ejemplo:

SELECT \* WHERE Nombre = 'Pepito' and numero IN (2, 3, 4);

* Y también podremos realizar una query cuya cláusula sea a través de rangos, como por ejemplo:

SELECT \* WHERE Nombre = 'Pepito' and numero >=6 and numero <=10;

* Al igual que ocurre en las bases de datos relacionales, podremos realizar consultas con criterios de ordenamiento como es la utilización de un order by (dicho order by se puede realizar si se hace sobre la clustering key, y que en el where aparezca la partition key).

select idusuario from personas where idusuario = '0s66' order by nombre;

* Una de las opciones que nos ofrece Cassandra en el momento de realizar la consulta es la posibilidad de definir el número de filas que deseemos que nos retorne en la consulta, para ello tendremos que introducir el siguiente comando.

select idusuario from personas where idusuario = '0s66' order by nombre limit n;

* Y por último una de las opciones que podemos utilizar es la opción denominada Allow Filtering que nos permite realizar las consultas en las que en el WHERE no haya condiciones en la partition key (aunque no se recomienda su uso, a menos que sea estrictamente necesario que seria en el caso de acceso secuencial a los datos).

select idusuario from personas where mes>9 allow filtering;

## Crear, alterar y eliminar un Keyspace

Como hemos aprendido anteriormente, un keyspace es el contenedor más exterior y debería ser creado antes que cualquier cosa, así que para poder realizar la creación de un keyspace tendremos que introducir el siguiente comando:

CREATE KEYSPACE IF NOT EXISTS "nombre keyspace" WITH replication = { 'class': 'SimpleStrategy', 'replication\_factor': '1'} AND durable\_writes = true;

El comando anterior está indicando que se realice la creación de un keyspace con el nombre deseado en el caso que no exista (si existe, no lo creará), además de ello realizará la replicación en un solo nodo, además de que estamos indicando la estrategia de cómo gestionará la réplica de los datos(este apartado se explica a continuación) y utilizará la política durable\_writes. Hay un dato a tener en cuenta y es que tenemos que definir como deseamos realizar la réplica de los datos, dependiendo de la opción que pongamos, nos realizará un tipo de replica u otro, existen dos opciones de replicado que se explican a continuación.

* SimpleStrategy: se define cuando deseamos que los datos se almacenen en un solo data center.
* NetworkTopologyStrategy: se define si se planea desplegar el cluster en múltiples data centers.

En el caso que deseemos realizar desde fuera de la base de datos (fuera de la shell de CQL) tendremos que realizarlo con el siguiente comando:

cqlsh -u "usuario" -p "password" localhost -e "CREATE KEYSPACE IF NOT EXISTS "nombre keyspace" WITH replication = { 'class': 'SimpleStrategy', 'replication\_factor': '1'} AND durable\_writes = true;"

Si en algún momento deseamos realizar la modificación de ese keyspace tendremos que introducir el siguiente comando que nos permite alterar las características del keyspace:

ALTER KEYSPACE "nombre keyspace" WITH REPLICATION = { 'class' : 'SimpleStrategy', 'replication\_factor' : 3 };

Con esta modificación estamos indicando que deseamos ajustar el número de réplicas disponibles de este keyspace, cambiando el valor de 1 a 3.

En el momento que deseemos usar este keyspace lo único que tendremos que realizar es la introducción del siguiente comando.

USE "nombre de keyspace";

Y si deseamos realizar el borrado de un Keyspace tendremos que introducir el siguiente comando. DROP KEYSPACE "nombre de keyspace";

## Crear, alterar y eliminar una Tabla

Como hemos visto anteriormente el concepto más equivalente en Cassandra de una Tabla en una base de datos relacional es una Familia de Columnas, que dicha familias de columnas almacenan una Super Columna o Columna, las cuales almacena las filas con los datos que son introducidos en la base de datos. Para poder realizar la creación de una “tabla”, tendremos que introducir el siguiente comando en el cual estamos indicando los tipos de campos de las columnas, además de indicar su correspondiente primary key.

CREATE TABLE personas (

id text,

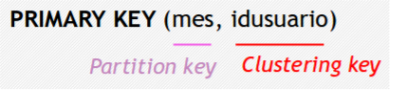
name text,

surname text,

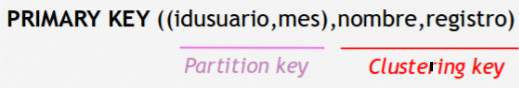
date time,

PRIMARY KEY (id, name));

Es recomendable indicar una fecha de registro e indicar dos primary keys, para que la primera sirva para definir la partitionkey y la segunda sea para definir el clustering key y van en ese orden.



Aunque también podremos definir la primary key y la clustering key de forma compuesta (que cada una está compuesta por más de uno).



### Tipos de Datos en las Tablas

Además de ello en el momento de la creación de la tabla podremos definir el tipo de dato que vamos a almacenar, por ello tendremos una amplia diversidad de opciones a elegir dependiendo del dato a introducir en esa columna, dichos tipos de datos se clasifican en los siguientes:

* Numéricos:

Int: enteros de hasta 32 bits.

Bigint: enteros de hasta 64 bits.

Smallint: enteros de hasta 2 bytes.

Tinyint: enteros de hasta 1 byte.

Varint: enteros de precisión arbitraria.

Decimal: decimales de precisión variable.

Float: coma-flotante de 32 bits.

Double: coma-flotante de 64 bits.

Counter: contador de enteros de hasta 64 bits.

* Texto:

Ascii: strings US-ASCII.

Text: strings UTF-8.

Varchar: strings UTF-8 de longitud variable.

Inet: strings que almacenan direcciones IPv4 o IPv6.

* Identificación:

UUID: tipo de dato que sigue el estándar UUID. Es único para cada fila insertada en una tabla. Se utiliza para identificar a las filas

Timeuuid: es un UUID que además sirve como timestamp único para cada fila.

* Fecha

Date: fechas desde el 1 de Enero de 1970 en formato yyyy-mm-dd, representada como string.

Time: hora, minutos y segundos en formato hh:mm:ss.sss, representado como string. Timestamp: fecha y hora con precisión al milisegundo en formato yyyy-mm-dd hh:mm:ss.sss. Puede ser representada como string.

* Colección:

List: colección de uno o más elementos ordenados.

Map: colección con pares clave-valor.

Set: colección de uno o más elementos.

* Otros tipos:

Blob: almacena bytes expresados en hexadecimal

En el momento que deseemos revisar la estructura de la tabla creada, tendremos que realizar la introducción del comando describe

DESCRIBE "nombre tabla";

Y si deseamos alterar la tabla creada anteriormente tendremos que realizar la introducción del siguiente comando (en el ejemplo que aparece a continuación añadimos la columna email de tipo texto).

ALTER TABLE personas ADD email text;

Además de añadir columnas, podremos modificar el tipo de dato que es capaz de almacenar esa columna y como en el ejemplo anterior, lo mismo que añadimos columnas nuevas, podremos también eliminarlas. Los dos ejemplos que aparecen a continuación muestran cómo se realizaría la modificación del tipo de columna y la eliminación de la misma.

* Modificación de tipo de columna:

ALTER TABLE personas ALTER email TYPE varchar;

* Eliminación de la columna.

ALTER TABLE personas DROP email;

En el momento que lo hayamos añadido podremos comprobar si se ha realizado correctamente con el comando describe.

Por último si deseamos realizar la eliminación de una TABLE tendremos que realizar la introducción del siguiente comando.

DROP TABLE "nombre tabla";

## Insertar y Actualizar Datos

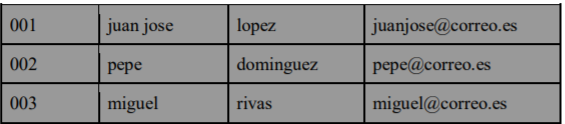
En el caso en el que deseemos insertar datos en una tabla concreta, tendremos que realizarlo de forma similar como lo hacemos en los sistemas de bases de datos relacionales, tal y como aparece a continuación.

INSERT INTO personas (id, name, surname, email) VALUES ('001', 'juan jose', 'lopez', 'juanjose@correo.es');

Un dato de interés a tener en cuenta es que Cassandra por defecto, no comprueba la unicidad de la Primary Key. Si existe otro dato con el mismo valor que el que se está insertando, el dato antiguo será sobreescrito, realizando así una actividad conocida como UPSERT.

Por ejemplo en el caso que deseemos realizar el insert que aparece a continuación sobreescribirá la información de juan josé por antonio, por que antonio que es el que se está insertando, tiene la misma primary key que juan josé.

INSERT INTO personas (id, name, surname, email) VALUES ('001', 'antonio', 'jimenez', 'antonio@correo.es');



Además del dato de interés anterior tenemos diferentes opciones que nos ofrece Cassandra a la hora de realizar la inserción de datos, siendo las siguientes opciones:

* USING TTL: esta opción nos permite realizar un insert por tiempo limitado, dicho tiempo se define en segundo y en el momento en el que se cumpla ese tiempo, Cassandra automáticamente eliminará ese dato introducido

INSERT INTO personas (id, name, surname, email) VALUES ('001', 'juan jose', 'lopez', 'juanjose@correo.es') USING TTL 87900;

* USING TIMESTAMP: esta opción está indicada en microsegundos y permite realizar un insert en el cual si no se usa la el tiempo especificado en microsegundos , se usará el tiempo en el que se hizo la escritura.

INSERT INTO personas (id, name, surname, email) VALUES ('001', 'juan jose', 'lopez', 'juanjose@correo.es') USING TIMESTAMP 123456789;

* IF NOT EXISTS: esta opción permite evitar la inserción de esa fila en el caso que exista otra fila con la misma Primary Key (evitando un UPSERTS). Existe el problema que disminuye el rendimiento de dicho insert ya que va comprobandolo fila por fila.

Y si se da el caso de tener que actualizar unos datos existentes para que aparezcan correctamente en la base de datos tendremos que introducir el siguiente comando para actualizar el dato correctamente.

UPDATE personas SET email='correojuanjose@correo.com' WHERE id='001';

## Eliminación de datos

En el caso de la eliminación de registros en una tabla concreta se puede realizar con la cláusula DELETE, utilizándose así de la misma forma que en los sistemas de bases de datos relacionales, pero en el caso de Cassandra puede eliminar la fila o el contenido de una columna concreta, a continuación se ven unos ejemplos que nos permitirá entender mejor como va todo el tema de la eliminación de datos.

* Esta opción elimina el contenido de la columna “nombre” de la fila con el “id=001”

DELETE nombre FROM personas WHERE id='001';

* Esta segunda opción nos permite eliminar la fila con el “id=001”

DELETE FROM personas WHERE id='001';

## Creación y eliminación de Índices

Al igual que ocurre en los sistemas de bases de datos relacionales, podremos realizar a creación de índices en Cassandra. Dichos índices se pueden aplicar sobre columnas o tablas, para su creación se tiene que introducir el siguiente comando.

CREATE INDEX idx\_nombreusuario ON personas(nombre);

Y si deseamos eliminar dicho index que hemos creado, solamente tendremos que introducir el siguiente comando, que nos permitirá quitarlo de nuestra base de datos Cassandra, para ello vamos a ver un ejemplo de ese comando.

DROP INDEX idx\_nombreusuario;

## TRIGGERS

Creación/Eliminación de trigger: podremos realizar la creación de triggers o eliminado de triggers con los siguientes comandos.

Creación: CREATE TRIGGER IF NOT EXISTS trigger\_name ON table\_name USING 'java\_class'; Eliminación:

DROP TRIGGER trigger\_name;

## USERS

Creación/Eliminación de usuarios: podremos realizar la creación o eliminado de usuario para así gestionar la base de datos y su contenido, todo ello se realiza con los siguientes comandos.

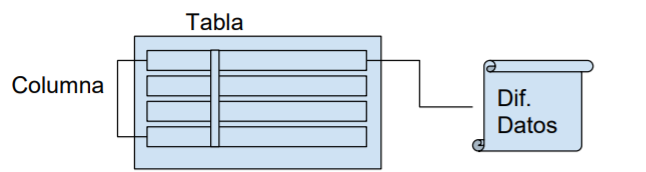
Creación:

CREATE USER IF NOT EXISTS "nombre usuario" WITH PASSWORD 'password usuario' NOSUPERUSER | SUPERUSER;

Eliminación: DROP USER "nombre usuario";

## Type

Otro de los elementos que puede ser administrado por Cassandra, son los datos TYPE. Estos datos TYPE no son más que espacios proporcionados al usuario (cuya creación y gestión es parecida a la de una tabla) que puede crear y usar tipos de datos definidos por él mismo. Puede crear un tipo de datos para manejar múltiples campos. Este tipo de componente se suele utilizar para asignarle a una columna de una tabla, que el tipo de columna sea de colección (collection) para que sean almacenados todos esos datos en un solo registro.



* Creación/Eliminación de TYPE: para realizar la creación o eliminación de los TYPE tendremos que introducir los siguientes comandos.

Creación:

CREATE TYPE direcciones (

calle text,

ciudad text,

cod\_postal int,

movil set );

Eliminación:

DROP TYPE direcciones;

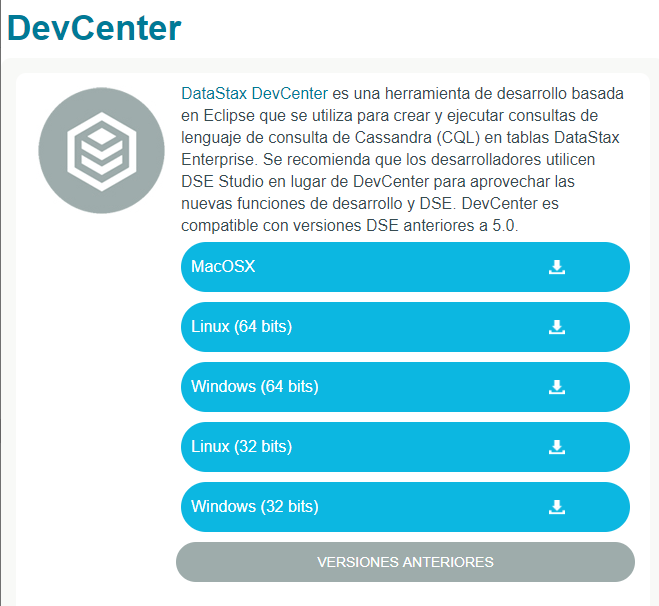
## Cliente remoto

Ya que sabemos como es el funcionamiento de Cassandra procederemos a conectar nuestra maquina virtual con un cliente remoto que funcione para Cassandra las mas conocidas son :

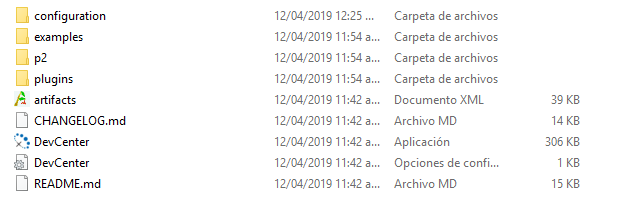
* DataStax: es la versión certificada de Cassandra preparada para entornos de producción intensiva. Integra servicios y herramientas de gestión con Cassandra, y utiliza tecnología de otros proyectos Apache, como el Hadoop, además de permitir búsquedas utilizando Solr.
* TablePlus :soporta además de cassandra ,oraclee ,maríaDB,mongoDB,etc.
* DataStax DevCenter: es una herramienta de desarrollo basada en Eclipse que se utiliza para crear y ejecutar consultas de lenguaje de consulta de Cassandra (CQL) en tablas DataStax Enterprise. Se recomienda que los desarrolladores utilicen DSE Studio en lugar de DevCenter para aprovechar las nuevas funciones de desarrollo y DSE. DevCenter es compatible con versiones DSE anteriores a 5.0.

Para la realización de este manual se escogio DevCenter por su fácil funcionamiento ,para la realización de esta manual se explicara desde la descarga hasta la instalación a continuación:

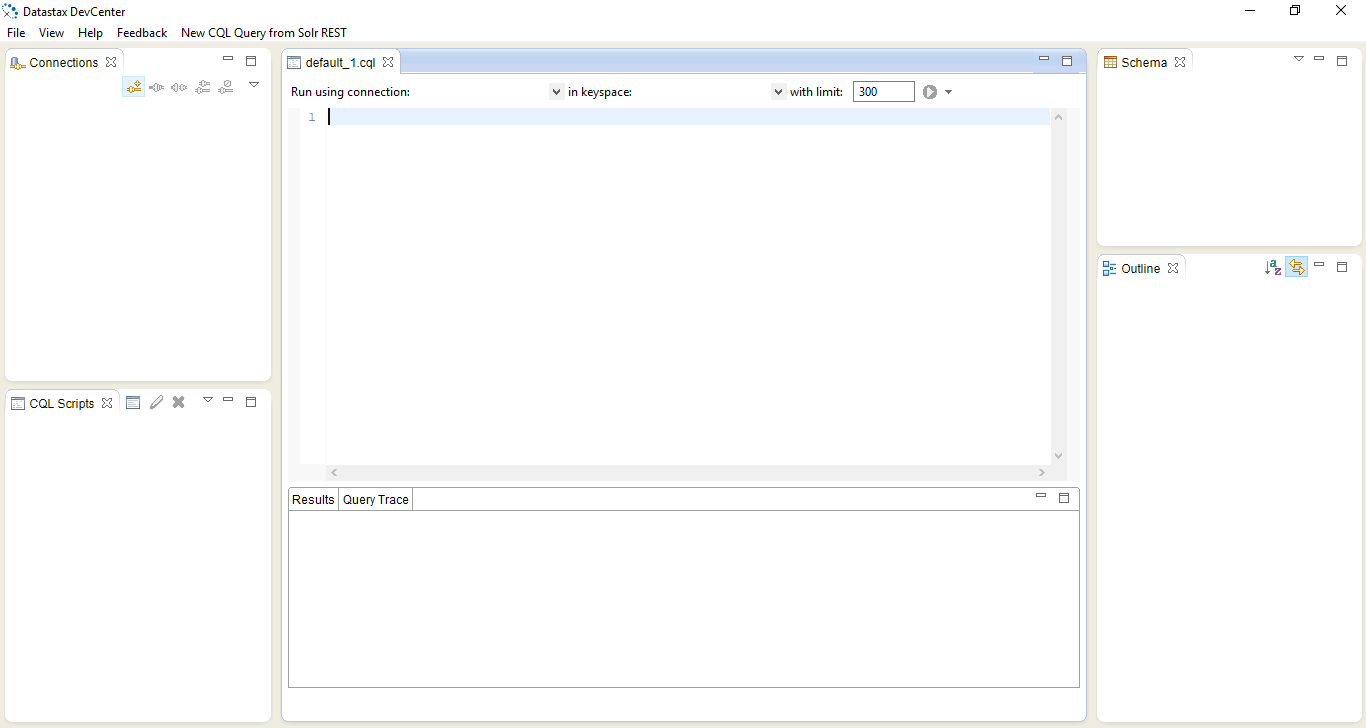
Ingresaremos a la pagina oficial de data Stax( <https://academy.datastax.com/downloads>)y descargaremos la versión para Windows.



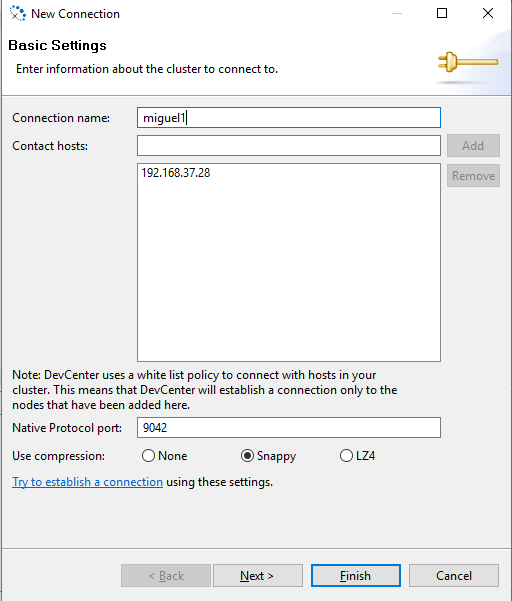
Una vez descargado el .zip de DevCenter lo descomprimiremos.



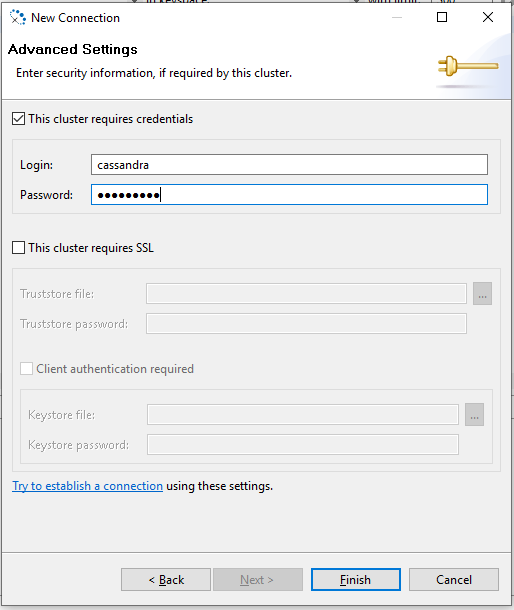
Una vez descomprimido ejecutaremos el DevCenter el cual nos mostrara la siguiente ventana.



Una vez ejecutado iremos a file e iremos a new para crear una nueva conexión ingresando el nombre de la conexión y el host que en este caso es 192.168.37.28 como se ve en la siguiente imagen.



Ya que ingresamos el nombre y el host le daremos click a next en el cual permitiremos la opción “This cluster requires credentials” para asi poder ingresar el usuario “cassandra” y contraseña “cassandra” nativos de cassandra como se ve en la siguiente imagen.



Una ves ingresado el usuario y la contraseña daremos click en finish para finalizar la conexión y nos aparecerá la conexión en la parte izquierda de la pantalla en este caso es miguel1 como se ve en la siguiente imagen.

