

# Práctica 2, Parte 1: Puesta en marcha de base de datos Cassandra

## Operaciones en Cassandra

- Similar a las bases de datos relacionales en Cassandra utilizamos un lenguaje específico para insertar, borrar, actualizar o consultar datos.
- Este lenguaje se llama Cassandra Query Language (CQL)
- Es muy similar a SQL ya que fue creado para que fuese fácilmente aprendido por los desarrolladores especializados en bases de datos relacionales.



- Utilizaremos el siguiente comando para crear el keyspace:

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

- La clase y el factor de replicación pueden variarse dependiendo de las necesidades de la organización

Elegir columnas clave

```
CREATE TABLE Users_by_artifact (artifact_id uuid, user_id uuid, user_name text,  
email text, areas_of_expertise set<text>, PRIMARY KEY (artifact_id, user_id))
```



Partition key

Clustering key

**Para crear una tabla con contadores:**

```
CREATE TABLE Contadores(pk uuid PRIMARY KEY, contador counter);
```

**En el caso de crear una composite partition key:**

```
PRIMARY KEY ((artifact_id, user_name), user_id, email)
```

Los tipos de datos soportados por Cassandra están en el siguiente enlace:

[https://docs.datastax.com/en/cql-oss/3.x/cql/cql\\_reference/cql\\_data\\_types\\_c.html](https://docs.datastax.com/en/cql-oss/3.x/cql/cql_reference/cql_data_types_c.html)

## Inserción de datos

- Muy similar a las inserciones en SQL.
- Sintaxis:

**INSERT INTO KeyspaceName.TableName(Column1Name, Column2Name, Column3Name . . . ) VALUES (Column1Value, Column2Value, Column3Value . . . )**

- Destacar que si la tabla a insertar contiene contadores/agregaciones, no se puede utilizar la operación INSERT. Se debe usar UPDATE
- A diferencia de SQL, en el caso de que haya datos que correspondan con la primary key, no devolverá error, sino que sobrescribirá los datos.

Actualización de datos

- Sintaxis muy similar a SQL.

```
UPDATE KeyspaceName.TableName
SET ColumnName1=newColumn1Value,
    ColumnName2=newColumn2Value,
    ColumnName3=newColumn3Value,
    ...
WHERE ColumnName=ColumnValue
```

- Al igual que en las inserciones, si se realiza un update donde la cláusula where no tiene ningún valor, insertará los datos
- La clausula WHERE debe contener solo columnas que son clave.
- Si hay clausula WHERE, debe haber un valor para toda columna que pertenezca a la clave primaria, tanto partition como clustering
- En el caso de agregaciones se usa el UPDATE siempre para insertar y actualizar valores:

**UPDATE k1.contadores SET contador = contador +1 WHERE id = 1**

## Borrado de datos

- **Muy similar a SQL. Sintaxis**

```
DELETE FROM KeyspaceName.TableName  
WHERE ColumnName1=ColumnValue
```

- La clausula **WHERE** debe contener solo columnas que son clave.
- Si hay clausula **WHERE** siempre deben estar presentes las **partition key**, las **clustering** son optativas.

## Consulta de datos

- Muy similar a SQL. Sintaxis

**SELECT ColumnNames from KeyspaceName.TableName Where ColumnName1=Column1Value  
AND ColumnName2=Column2Value**

- Al igual que en las operaciones DELETE en la clausula WHERE solo pueden estar columnas que sean clave.
- Si hay clausula WHERE siempre deben estar presentes las partition key, las clustering son optativas.



### Datastax Devcenter

- Gestionar una base de datos exclusivamente desde consola es poco intuitivo y puede llegar a ser complejo para ciertos usuarios
- Aplicación portable disponible para Windows, Mac y Linux que nos permitirá gestionar nuestras bases de datos Cassandra desde un entorno gráfico.
- Nos permitirá realizar conexiones a bases de datos que se encuentran en servidores remotos.
- Gran ventaja con respecto a la administración por consola



<https://downloads.datastax.com/#devcenter>

## Conexión a la base datos

- Necesitamos establecer los siguientes parámetros:
  - Nombre conexión
  - IP del servidor
  - Puerto
  - Usuario y contraseña si se requiere.

Properties for connection1

type filter text

> Basic Settings

### Basic Settings

Enter information about the cluster to connect to.

Connection name:

Contact hosts:  Add Remove

Note: DevCenter uses a white list policy to connect with hosts in your cluster. This means that DevCenter will establish a connection only to the nodes that have been added here.

Native Protocol port:

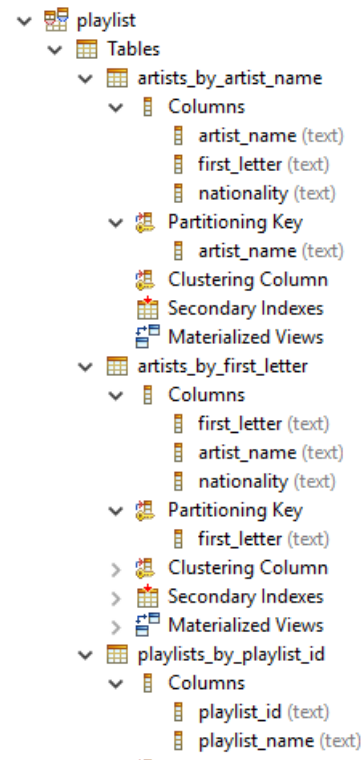
Use compression: ☐ None ☒ Snappy ☐ LZ4

[Try to establish a connection](#) using these settings.

OK Cancel

## Esquema base de datos

- Una vez establecida la conexión se nos muestra la información de nuestra base de datos dividida en keyspaces.
- En cada keyspace se nos muestra la información de las tablas que lo componen
- Por cada table tenemos información de las columnas y cuales de esas columnas son partition key o clustering key.



## Inserción y consulta de datos

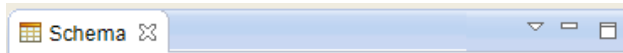
- Se pueden ejecutar todos tipo de sentencias CQL, incluidas la creación de keyspaces, tablas y modificaciones de datos.
- La consulta de datos a través de sentencias SELECT se muestra en la parte inferior de la aplicación mostrando las filas y las columnas.
- También se muestra el tiempo de ejecución de la operación.

```
1 INSERT INTO artists_by_artist_name (artist_name,  
2     first_letter, nationality  
3 ) values ('nombre', 'n', 'español');  
4 SELECT * from artists_by_artist_name;
```

Results			Query Trace		
artist_name			first_letter		
nombre			n		
nationality			español		

1 selected statement successfully executed in 116 ms. Retrieved 1 row

## Nuevo keyspace



New Keyspace... Ctrl+Alt+Shift+K



New Keyspace

Basic Settings

Define the keyspace and properties. Click Next to review summary.

Connection: connection3 New...

Keyspace name: keyspace1

Replication Strategy: SimpleStrategy SimpleStrategy NetworkTopologyStrategy

Replication factor: 1

☐ Durable writes: ☒

Reset properties above to their [default](#) values.

See more details about keyspace properties and their default values [here](#).

▼ CQL Preview:

```
CREATE KEYSPACE keyspace1
WITH replication = {
  'class' : 'SimpleStrategy',
  'replication_factor' : 1
};
```



# Creación Tabla

## Tabla Productos

Precio (K)

Id\_Producto (C)

Nombre

Existencias

New Table

**Basic Settings**

Define the table's basic structure below. Click Next to fine-tune the table's primary key structure.

Connection: connection3 New...

Keyspace: keyspace1 New...

Table name: Productos

Columns:

Name	Type	Primary Key	Static
Precio	float	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Id_Producto	int	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Nombre	text	<input type="checkbox"/>	<input type="checkbox"/>
Existencias	int	<input type="checkbox"/>	<input type="checkbox"/>

Add Remove Up Down

Comment:

▼ CQL Preview:

```
CREATE TABLE keyspace1."Productos" (  
  "Precio" float,  
  "Id_Producto" int,  
  "Nombre" text,  
  "Existencias" int,  
  PRIMARY KEY ("Precio", "Id_Producto")  
);
```

Help < Back Next > Last >> Cancel

# Organización clave primaria

New Table

**Primary Key Settings**

Define the table's primary key structure below. Click Next to set more advanced properties.

Available columns:

Nombre
Existencias

Partition keys:

Precio

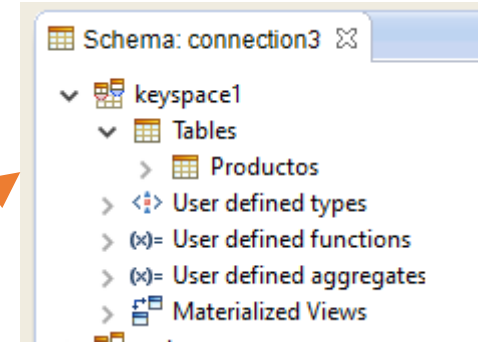
Clustering columns:

Name	Order
Id_Producto	ASC

QCL Preview:

```
CREATE TABLE keyspace1."Productos" (  
  "Precio" float,  
  "Id_Producto" int,  
  "Nombre" text,  
  "Existencias" int,  
  PRIMARY KEY ("Precio", "Id_Producto")  
);
```

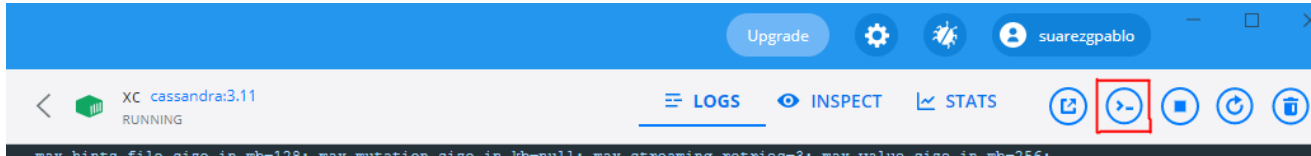
Help < Back **Next >** Last >> Cancel



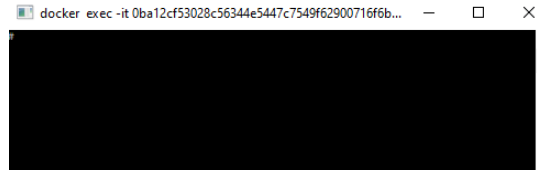


- Crearemos las tablas de forma manual utilizando el cliente de Docker en caso de que lo estamos usando.
- Si usamos Cassandra instalado en local nos debería bastar con ejecutar el comando “cqlsh”.

- Tras hacer click en el contenedor, clickamos en el siguiente botón:



- Debería aparecer la siguiente ventana:



- Ejecutamos cqlsh en ella y debería mostrarse lo siguiente:

```
docker exec -it 0ba12cf53028c56344e5447c7549f62900716f6b...  
# cqlsh  
Connected to Test Cluster at 127.0.0.1:9042.  
[cqlsh 5.0.1 | Cassandra 3.11.10 | CQL spec 3.4.4 | Native protocol v4]  
Use HELP for help.  
cqlsh>
```

- Primero abra una terminal.
- En ella asegúrese que la máquina Docker se está ejecutando con el siguiente comando:

**sudo docker ps**

- Debería ver algo similar a la siguiente captura:

```
[UNIVERSIDADVIU\pablo.suarez@a-2nqewuzyjo3sz ~]$ sudo docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
2dfab598e901	cassandra:3.11	"docker-entrypoint.s..."	7 days ago	Up 28 hours	7000-7001/tcp, 7199/tcp, 9160/tcp, 0.0.0.0:9042->9042/tcp, :::9042->9042/tcp	cassandra

- En caso de que no se esté ejecutando, ejecute el siguiente comando siendo “cassandra” el nombre de su contenedor. Cambie dicho valor por el correspondiente:

**sudo docker start cassandra**

- Posteriormente ejecute los siguientes comandos

**sudo docker exec -it cassandra bash  
cqlsh**

- A partir de este momento hemos entrado en cqlsh y podremos iniciar la creación de estructuras.
- Primero creamos el keyspace:

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

- Para entrar en él ejecutamos el siguiente comando, siendo “nombreKeyspace” el nombre del keyspace que vamos a usar:

```
USE nombreKeyspace
```

- Dentro del keyspace podremos empezar a crear las tablas así como ejecutar operaciones de modificación y consulta de datos.

```
CREATE TABLE practica2.productos (  
    precio float,  
    idproducto int,  
    existencias int,  
    nombre text,  
    PRIMARY KEY (precio, idproducto)  
) WITH CLUSTERING ORDER BY ( idproducto ASC );
```

- Insertar 5 filas en la tabla Productos
- Actualizar los valores del Nombre y de las existencias
- Borrar 1 fila
- Actualizar el precio en 1 fila. ¿Cómo lo hacemos?
- Crear las tablas de las consultas 1, 2 y 3 diseñadas en la Práctica 1
- Crea una tabla en la que se consulten los Productos por su precio

### Consulta 1: cliente\_pedidos

Column	Primary Key
pedido_fecha	Partition key
pedido_idpedidos	Clustering Key
cliente_nombre	
cliente_dni	
cliente_dirección	
cliente_idcliente	

### Consulta 3: cliente\_producto

Column	Primary Key
cliente_nombre	PK
cliente_dni	Clustering Key
producto_idproducto	Clustering Key
pedido_id	
producto_precio	
producto_existencia	

### Consulta 2: numpedidos

Column	Primary Key
Pedido_fecha	Partition key
numpedidos	+

### Consulta Extra: productos

Column	Primary Key
precio	PK
idproducto	Clustering Key
nombre	
existencia	