Using Cassandra with cqlsh

To start working with Cassandra, let's start the **Cassandra Query Language** (**CQL**) shell . The shell interface will allow us to execute CQL commands to define, query, and modify our data. As this is a new cluster and we have turned on authentication and authorization, we will use the default cassandra and cassandra username and password, as follows:

Copy

**bin/cqlsh 192.168.0.101 -u cassandra -p cassandra**

**Connected to PermanentWaves at 192.168.0.101:9042.**

**[cqlsh 5.0.1 | Cassandra 3.11.2 | CQL spec 3.4.4 | Native protocol v4]**

**Use HELP for help.**

**cassandra@cqlsh>**

First, let's tighten up security. Let's start by creating a new superuser to work with.

New users can only be created if authentication and authorization are properly set in the cassandra.yaml file:

Copy

**cassandra@cqlsh> CREATE ROLE cassdba WITH PASSWORD='flynnLives' AND LOGIN=true and SUPERUSER=true;**

Now, set the default cassandra user to something long and indecipherable. You shouldn't need to use it ever again:

Copy

**cassandra@cqlsh> ALTER ROLE cassandra WITH PASSWORD='dsfawesomethingdfhdfshdlongandindecipherabledfhdfh';**

Then, exit cqlsh using the exit command and log back in as the new cassdba user:

Copy

**cassandra@cqlsh> exit**

**bin/cqlsh 192.168.0.101 -u cassdba -p flynnLives**

**Connected to PermanentWaves at 192.168.0.101:9042.**

**[cqlsh 5.0.1 | Cassandra 3.11.2 | CQL spec 3.4.4 | Native protocol v4]**

**Use HELP for help.**

**cassdba@cqlsh>**

Now, let's create a new keyspace where we can put our tables, as follows:

Copy

**cassdba@cqlsh> CREATE KEYSPACE packt WITH replication =**

**{'class': 'NetworkTopologyStrategy', 'ClockworkAngels': '1'}**

**AND durable\_writes = true;**

Note

For those of you who have used Cassandra before, you might be tempted to build your local keyspaces with SimpleStrategy. SimpleStrategy has no benefits over NetworkTopologyStrategy, and is limited in that it cannot be used in a plural data center environment. Therefore, it is a good idea to get used to using it on your local instance as well.

With the newly created keyspace, let's go ahead and use it:

Copy

**cassdba@cqlsh> use packt;**

**cassdba@cqlsh:packt>**

Note

The cqlsh prompt changes depending on the user and keyspace currently being used.

Now, let's assume that we have a requirement to build a table for video game scores. We will want to keep track of the player by their name, as well as their score and game on which they achieved it. A table to store this data would look something like this:

Copy

**CREATE TABLE hi\_scores (name TEXT, game TEXT, score BIGINT,**

**PRIMARY KEY (name,game));**

Next, we will INSERT data into the table, which will help us understand some of Cassandra's behaviors:

Copy

**INSERT INTO hi\_scores (name, game, score) VALUES ('Dad','Pacman',182330);**

**INSERT INTO hi\_scores (name, game, score) VALUES ('Dad','Burgertime',222000);**

**INSERT INTO hi\_scores (name, game, score) VALUES ('Dad','Frogger',15690);**

**INSERT INTO hi\_scores (name, game, score) VALUES ('Dad','Joust',48150);**

**INSERT INTO hi\_scores (name, game, score) VALUES ('Connor','Pacman',182330);**

**INSERT INTO hi\_scores (name, game, score) VALUES ('Connor','Monkey Kong',15800);**

**INSERT INTO hi\_scores (name, game, score) VALUES ('Connor','Frogger',4220);**

**INSERT INTO hi\_scores (name, game, score) VALUES('Connor','Joust',48850);**

**INSERT INTO hi\_scores (name, game, score) VALUES ('Avery','Galaga',28880);**

**INSERT INTO hi\_scores (name, game, score) VALUES ('Avery','Burgertime',1200);**

**INSERT INTO hi\_scores (name, game, score) VALUES ('Avery','Frogger',1100);**

**INSERT INTO hi\_scores (name, game, score) VALUES ('Avery','Joust',19520);**

Now, let's execute a CQL query to retrieve the scores of the player named Connor:

Copy

**cassdba@cqlsh:packt> SELECT \* FROM hi\_scores WHERE name='Connor';**

**name | game | score**

**--------+-------------+--------**

**Connor | Frogger | 4220**

**Connor | Joust | 48850**

**Connor | Monkey Kong | 15800**

**Connor | Pacman | 182330**

**(4 rows)**

That works pretty well. But what if we want to see how all of the players did while playing the Joust game, as follows:

Copy

**cassdba@cqlsh:packt> SELECT \* FROM hi\_scores WHERE game='Joust';**

**InvalidRequest: Error from server: code=2200 [Invalid query] message="Cannot execute this query as it might involve data filtering and thus may have unpredictable performance. If you want to execute this query despite the performance unpredictability, use ALLOW FILTERING"**

Note

As stated in the preceding error message, this query could be solved by adding the ALLOW FILTERING directive. Queries using ALLOW FILTERING are notorious for performing poorly, so it is a good idea to build your data model so that you do not use it.

Evidently, Cassandra has some problems with that query. We'll discuss more about why that is the case later on. But, for now, let's build a table that specifically supports querying high scores by game:

Copy

**CREATE TABLE hi\_scores\_by\_game (name TEXT, game TEXT, score BIGINT,**

**PRIMARY KEY (game,score)) WITH CLUSTERING ORDER BY (score DESC);**

Now, we will duplicate our data into our new query table:

Copy

**INSERT INTO hi\_scores\_by\_game (name, game, score) VALUES ('Dad','Pacman',182330);**

**INSERT INTO hi\_scores\_by\_game (name, game, score) VALUES ('Dad','Burgertime',222000);**

**INSERT INTO hi\_scores\_by\_game (name, game, score) VALUES ('Dad','Frogger',15690);**

**INSERT INTO hi\_scores\_by\_game (name, game, score) VALUES ('Dad','Joust',48150);**

**INSERT INTO hi\_scores\_by\_game (name, game, score) VALUES ('Connor','Pacman',182330);**

**INSERT INTO hi\_scores\_by\_game (name, game, score) VALUES ('Connor','Monkey Kong',15800);**

**INSERT INTO hi\_scores\_by\_game (name, game, score) VALUES ('Connor','Frogger',4220);**

**INSERT INTO hi\_scores\_by\_game (name, game, score) VALUES ('Connor','Joust',48850);**

**INSERT INTO hi\_scores\_by\_game (name, game, score) VALUES ('Avery','Galaga',28880);**

**INSERT INTO hi\_scores\_by\_game (name, game, score) VALUES ('Avery','Burgertime',1200);**

**INSERT INTO hi\_scores\_by\_game (name, game, score) VALUES ('Avery','Frogger',1100);**

**INSERT INTO hi\_scores\_by\_game (name, game, score) VALUES ('Avery','Joust',19520);**

Now, let's try to query while filtering on game with our new table:

Copy

**cassdba@cqlsh:packt> SELECT \* FROM hi\_scores\_by\_game**

**WHERE game='Joust';**

**game | score | name**

**-------+-------+--------**

**Joust | 48850 | Connor**

**Joust | 48150 | Dad**

**Joust | 19520 | Avery**

**(3 rows)**

As mentioned previously, the following chapters will discuss why and when Cassandra only allows certain PRIMARY KEY components to be used in the WHERE clause. The important thing to remember at this point is that in Cassandra, tables and data structures should be modeled according to the queries that they are intended to serve.

[Previous Section](https://subscription.packtpub.com/book/big_data_and_business_intelligence/9781789131499/1/ch01lvl1sec14/cassandra-cluster-manager)[Next Section](https://subscription.packtpub.com/book/big_data_and_business_intelligence/9781789131499/1/ch01lvl1sec16/shutting-down-cassandra)

 Not using [Hotjar](https://www.hotjar.com/incoming-feedback?utm_source=client&utm_medium=incoming_feedback&utm_campaign=insights" \t "_blank) yet?

Select an element on the page.