Querying Cassandra

NoSQL

A4- S8

ESILV
jihane.mali (at) devinci.fr

Chapter

Contents

Chapter 1

Create a database on Cassandra

1.1 Files Transfer

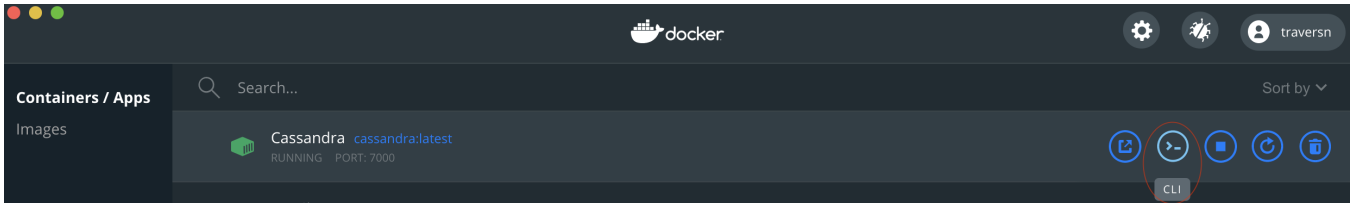
1.1.1 Transfer the file to the container. in your command line (OS console/shell not CLI console):

```
docker cp PATH_TO_FOLDER/DBLP.tar.gz Cassandra:/
```

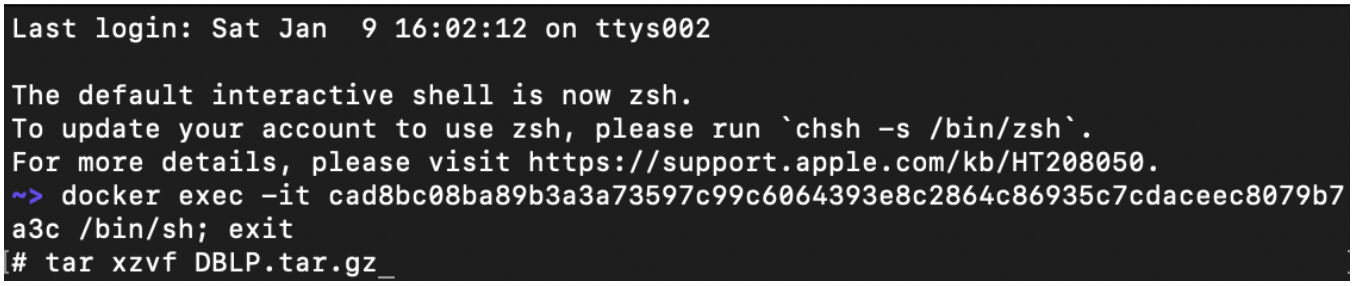
Of course "PATH_TO_FOLDER" is the folder where you have put the DBLP.tar.gz file. You can drag and drop it into the shell.

"Cassandra" is the name of the container. If you did not named it like this, please change to the corresponding name.

1.1.2 Open container's `Docker CLI`, available on the Docker dashboard (docker console)



1.1.3 In the `Docker CLI` environment, unarchive the file in the container:




```
tar xzvf DBLP.tar.gz
```

Sometimes, it happens that your file is already unarchived (but not untared). For that, apply simply: "tar xvf DBLP.tar" (without the 'z' for unzipping).

1.1.4 Open a CQLSH console for the following steps:

```
cqlsh
```

1.2 Create Your Keyspace

```
CREATE KEYSPACE IF NOT EXISTS DBLP
     WITH REPLICATION =
     { 'class' : 'SimpleStrategy', 'replication_factor': 3 };
```

Here we create a "DBLP" database for which the replication factor is set to 3, in order to manage falt tolerance.

To select the database:

```
USE DBLP;
```

1.3 Create Tables and Import Data

All those instructions has to be applied in the *Docker CLI* environment in CQLSH (the COPY and SOURCE instructions are not recognized by TablePlus).

1.3.1 Publications

Create the "publications" table:

```
CREATE TABLE publications (
      art_id TEXT, type TEXT, title text, pages_start INT, pages_end int, booktitle text,
      journal_series text, journal_editor text, journal_volume int, journal_isbn text,
      url text, year int,
      PRIMARY KEY (art_id)
);
ALTER TABLE publications WITH GC_GRACE_SECONDS = 0;
CREATE INDEX btree_publi_type on publications(type);
```

NB GC_GRACE_SECONDS␣=␣0;
Since the practice works is done locally with only ONE server, no replication can be made on the network. Thus, it is necessary to say Cassandra not to wait for aknowledgement of updates.

Import data from the CSV file:

```
COPY publications(art_id,type,year,title,pages_start,pages_end,booktitle,journal_series,
     journal_editor,journal_volume,journal_isbn,url)
     FROM 'DBLP/DBLP_publis.csv' WITH HEADER = true AND DELIMITER=';';
```

1.3.2 Authors

Create the "authors" table:

```
CREATE TABLE authors (
     art_id TEXT, author TEXT, pos INT,
     PRIMARY KEY ((author), art_id)
);
ALTER TABLE authors WITH GC_GRACE_SECONDS = 0;
CREATE INDEX btree_authors_art_id on authors(art_id);
CREATE INDEX btree_authors_pos on authors(pos);
```

Import data from the CSV file:

```
COPY authors(art_id,author,pos) FROM 'DBLP/authors.csv' WITH HEADER = true AND DELIMITER=';';
```

1.3.3 Authors publications

Create the "authors_publis" table:

```
CREATE TABLE authors_publis (
      art_id TEXT, author TEXT, type TEXT, title text, pages_start INT, pages_end int,
      booktitle text, journal_series text, journal_editor text, journal_volume int,
      journal_isbn text, url text, year int, pos int,
      PRIMARY KEY ((author), art_id)
);
ALTER TABLE authors_publis WITH GC_GRACE_SECONDS = 0;

CREATE INDEX btree_authors_publi_type on authors_publis(type);
CREATE INDEX btree_authors_publi_title on authors_publis(title);
```

Import data from the CSV file:

```
COPY authors_publis(art_id,author,type,year,title,pages_start,pages_end,booktitle,
      journal_series,journal_editor,journal_volume,journal_isbn,url,pos)
      FROM 'DBLP/authors_publis.csv' WITH HEADER = true AND DELIMITER=';';
```

Chapter 2

Querying Cassandra

Now we have a database, we can query it. However, since we use a query language very close to SQL, we will try to use it in this way. This practice work will be a little bit disturbing. The goal is to show how to use Cassandra in the proper way, not in the way to use traditional databases.

The queries can be either executed in "*CQLSH*" (in the `Docker CLI` environment) or in '*TablePlus*".

Remind that to connect "TablePlus" to CQL, your container MUST have the redirected port 9042. Otherwise, it cannot find cassandra. This step has been done during the container installation.

2.1 CQL : Simple Queries

The Cassandra query Language CQL is available here: `https://cassandra.apache.org/doc/latest/cql/dml.html#select`. It is inspired from SQL. In the following, express the following queries in CQL:

2.1.1 List of publications,

2.1.2 List of publications titles,

2.1.3 Booktitle of publications id "*series/sci/2008-156*",

2.1.4 Number of "*Book*" publications,

2.1.5 Number of publications WHERE booktitle is equal to "*HICSS*",

2.1.6 Use "`ALLOW FILTERING`" to execute the query,

2.1.7 The good approach is to create a seconday index on the "*booktitle*" attribute to be more efficient. Execute the query again without "`ALLOW FILTERING`",

2.1.8 Number of publications where type is "*Article*" and booktitle is equal to "*HICSS*",

2.1.9 Number of `authors` whose position is equal to 3,

2.1.10 Number of `authors` whose position is above to 3,

Bonus: The Token Hash Function Give the number of publications for which "*token(art_id)*" is below 0 (and above). You can also give the token and arti_id of each publication.

2.2 Complex Queries: Aggregates

Some grouping queries are available on the *partitioning key* (the complex part of the modelization).

2.2.1 Count the number of publications per author,

2.2.2 Count the number of publications per author when they are in third position,

2.2.3 Try to count the number of authors per position.
Hint: Choose an other partitioning key.

2.2.4 Distribution of positions for author "*Oscar Castillo*",

2.3 Complex Queries: Joins & Denormalization

2.3.1 Give authors' name for publication which title is "*Medical imaging archiving: A comparison between several NoSQL solutions.*". Join between tables *publications* and *authors*,

2.3.2 There is no way to do a join in CQL[1]. A first denormalization step has been done on this dataset with table "*authors_publis*". Try the previous query on this denormalized table.

2.3.3 Give titles and position of publications from "*Oscar Castillo*",

2.3.4 Give authors' name who published with "*Oscar Castillo*",

2.3.5 To answer this query, it requires a new denormalization with SET, MAP, LIST, TYPES or TUPLE. Create a table "*publicationsNorm*" which can insert documents in file "`DBLP.json`". An example is given below:

```
INSERT INTO publicationsNorm JSON
 '{"id":"series/cogtech/BrandhermSNL13", "type":"Article", "year":2013,
 "title":"A SemProM Use Case: Health Care and Compliance.",
 "authors":["Boris Brandherm","Michael Schmitz","Robert Ne?elrath","Frank Lehmann"],
 "pages":{"start":349, "end":361}, "booktitle":"SemProM",
 "journal":{"series":"", "editor":"", "volume":0, "isbn":[ "" ]},
 "url":"db/series/cogtech/364237376.html#BrandhermSNL13", "cites":[ "" ]}';
```

2.3.6 Once this sample can be inserted, import the whole dataset with this command (in the *Docker CLI* environment):

```
SOURCE '/DBLP.json';
```

2.3.7 Create an index on attribute 'title' of this new table,

2.3.8 Give authors' name for publication "Data Quality" in this new table,

2.3.9 Give the journal's series of this publication,

Bonus:

2.3.10 Give the pages' end of this publication,

2.3.11 Give the first author of this publication,

2.3.12 Give title's publications where authors' name is "*Oscar Castillo*",

2.3.13 Give titles and the starting page of publications which ends at page 99 while using an index,

2.3.14 Give titles of journal series : "*Advances in Database Systems*"

Comments on Cassandra Cassandra is used to manipulate mostly simple tables or very few denormalization. For instance, a sensor management system is really interesting with *Partitioning Keys* like in question 2.3.2.

To model and manipulate tuples, it is preferable to avoid TYPE and TUPLE. Use MAP or LIST to maximize the queries you wish to apply.

2.4 Hard Queries: User Define Aggregate functions

To agregate values from different rows, we need to create *User Define Aggregate* functions (UDA).

[1]Since it is a distributed database, a wide and costly broadcast is necessary and has to be avoided absolutely.

2.4.1 Enable user_define_function

To achieve this, we need first to activate this functionnality with parameter "*user_defined_functions_enabled*" in Cassandra.

For this:

- In the shell (eg., Docker CLI), update the dependancies: `apt-get␣update`
- Install a shell editor (*vim* or *nano*): `apt-get␣install␣vim` (or *nano* as you wish)
- Edit the file "`cassandra.yaml`" (config Cassandra folder): `vim␣/etc/cassandra/cassandra.yaml`
- Find "`user_defined`": `/user_defined_functions`
- Modify the parameter with '`true`' value (a space is mandatory between ':' and 'true'),
  - Switch *on* modify mode: `i`
  - Modify *false* in *true*
  - Exiting modify mode: `escape` (the key 'esc' on the keyboard)
  - Save and quit mode: `:wq`
- Restart the Cassandra server (relaunch the docker container on *Docker Desktop*).

2.4.2 Create your Own Function

Then, we can create the UDA:

2.4.1 Create the *state* function:

```
CREATE OR REPLACE FUNCTION avgState ( state tuple<int,bigint>, val int )
 CALLED ON NULL INPUT RETURNS tuple<int,bigint> LANGUAGE java
 AS 'if (val !=null) { state.setInt(0, state.getInt(0)+1);
    state.setLong(1, state.getLong(1)+val.intValue()); }
    return state;';
```

2.4.2 Create *final* function:

```
CREATE OR REPLACE FUNCTION avgFinal ( state tuple<int,bigint> )
 CALLED ON NULL INPUT RETURNS double LANGUAGE java
 AS 'double r = 0;
    if (state.getInt(0) == 0) return null;
    r = state.getLong(1);
    r/= state.getInt(0);
    return Double.valueOf(r);';
```

2.4.3 Create the *UDA* function

```
CREATE AGGREGATE IF NOT EXISTS average ( int )
 SFUNC avgState STYPE tuple<int,bigint>
 FINALFUNC avgFinal INITCOND (0,0);
```

2.4.4 Compute the average *position* of "*Oscar Castillo*" in his publications,

2.4.5 Idem with the average number of pages,

2.4.6 **Bonus:** Create a new UDA to produce an equivalence to *"GROUP BY + COUNT"* on textual attributes, like for:

```
SELECT countGroup(pos) FROM authors_publis;
```

The type of the "state" parameter must be a "*MAP<int, int>*".