## 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,

select \* from publications;

2.1.2 List of publications titles,

select title from publications;

2.1.3 Booktitle of publications id “series/sci/2008-156”,

select title from publications where art\_id='series/sci/2008-156';

2.1.4 Number of “Book” publications,

select count(\*) from publications where type='Book';

**6178**

2.1.5 Number of publications WHERE booktitle is equal to “HICSS”,

select count(\*) from publications where booktitle='HICSS';

2.1.6 Use “ALLOW FILTERING” to execute the query,

select count(\*) from publications where booktitle='HICSS' ALLOW FILTERING;

2.1.7 The good approach is to create a seconday index on the “booktitle” attribute to be more efficient.

CREATE INDEX btree\_publi\_booktitle on publications(booktitle);

select count(\*) from publications where booktitle='HICSS';

Execute the query again without “ALLOW FILTERING”,

2.1.8 Number of publications where type is “Article” and booktitle is equal to “HICSS”,

select count(\*) from publications where type='Article' AND booktitle='HICSS' ALLOW FILTERING;

**1271**

2.1.9 Number of authors whose position is equal to 3,

select count(\*) from authors where pos=3;

2.1.10 Number of authors whose position is above to 3,

select count(\*) from authors where pos>3 ALLOW FILTERING;

**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.**

select count(\*) from publications where token(art\_id)<0;

select count(\*) from publications where token(art\_id)>0;

select art\_id, token(art\_id) as token\_number from publications;

## 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,

select author, count(\*) as n\_pubs from authors\_publis group by author;

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

select author, count(\*) as n\_pubs from authors\_publis where pos=3 group by author ALLOW FILTERING ;

2.2.3 Try to count the number of authors per position.

Hint: Choose an other partitioning key.

**drop table if exists temp;**

**create table temp(**

**author text,**

**art\_id text,**

**pos int,**

**PRIMARY KEY ((pos), author)**

**);**

**ALTER TABLE temp with Gc\_grace\_Seconds=0;**

**create index btree\_temp\_author on temp(author);**

**create index btree\_temp\_pos on temp(pos);**

**COPY temp(author, art\_id, pos)**

**FROM 'DBLP/authors.csv' WITH HEADER = true AND DELIMITER=';';**

**select pos, count(\*) as n\_authors from temp GROUP BY pos;**

2.2.4 Distribution of positions for author “Oscar Castillo”,

**select pos, author, count(\*) from temp where author='Oscar Castillo' GROUP BY pos;**

# 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,

select author from author join publications where title='Medical imaging archiving: A comparison between several NoSQL solutions.';

2.3.2 There is no way to do a join un CQL1

. A first denormalization step has been done on this dataset with table

“authors\_publis”. Try the previous query on this denormalized table.

select author from authors\_publis where title='Medical imaging archiving: A comparison between several NoSQL solutions.';

2.3.3 Give titles and position of publications from “Oscar Castillo”,

select author, title, pos from authors\_publis where author='Oscar Castillo';

2.3.4 Give authors’ name who published with “Oscar Castillo”,

select distinct(author) from author\_publis inner join (select author, title, pos from authors\_publis where author='Oscar Castillo') using(title);

-- Does not work

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:

create type journal (

series text,

editor text,

volume int,

isbn list<text>

);

create table publicationsNorm (

id text,

type text,

year int,

title text,

authors set<text>,

pages map<text, int>,

booktitle text,

journal FROZEN<journal>,

url text,

cites set<text>,

PRIMARY KEY ((id), title)

);

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'; -- It worked! :D

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

create index btree\_pubnorm\_title on publicationsNorm(title);

2.3.8 Give authors’ name for publication “Data Quality” in this new table,

select authors from publicationsNorm where title='Data Quality';

2.3.9 Give the journal’s series of this publication,

select journal.series from publicationsNorm where title='Data Quality';

2.3.10 Give the pages’ end of this publication,

select pages['end'] from publicationsNorm where title='Data Quality';

2.3.11 Give the first author of this publication,

2.3.12 Give title’s publications where authors’ name is “Oscar Castillo”,

select title, authors from publicationsNorm where authors contains 'Oscar Castillo' ALLOW FILTERING;

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

create index btree\_pubnorm\_pages\_end on publicationsNorm(pages);

select title, pages['start'] from publicationsNorm where pages['end'] = 99 ALLOW FILTERING;

2.3.14 Give titles of journal series: “Advances in Database Systems”