Cassandra report - DBLP

MANIP M'EBOBISSE Séthi - MOHAMED Soilhat - Ngatcha Nancy -ROMANO Léa

27 janvier 2019

## How we import the dataset in the NoSQL database

### Schema

First, we need to determine the schema on witch we’ll construct our database. After reading the file and analyzing the different keys and values type, we came to a first schema:

CREATE TYPE pagesType (

    start INT,

    end INT

);

CREATE TYPE journalType (

    series VARCHAR,

    editor VARCHAR,

    volume VARCHAR,

    isbn LIST<VARCHAR>

);

CREATE TABLE IF NOT EXISTS DBLP (

    id VARCHAR,

    type VARCHAR,

    year INT,

    title VARCHAR,

    authors LIST<VARCHAR>,

    pages frozen<pagesType>,

    booktitle VARCHAR,

    journal frozen<journalType>,

    url VARCHAR,

    cites LIST <VARCHAR>,

    PRIMARY KEY(id)

);

### Program for the dataset

We decided to write the loading program in python :

import json

from cassandra.cluster import Cluster

cluster = Cluster(['127.0.0.1'])

session = cluster.connect()

session.execute("CREATE KEYSPACE IF NOT EXISTS DBLP WITH REPLICATION = {'class':'SimpleStrategy','replication\_factor':3};")

session.set\_keyspace('dblp')

session.execute("CREATE TYPE IF NOT EXISTS pagesType ( \

start INT, \

end INT\

);")

session.execute("CREATE TYPE IF NOT EXISTS journalType ( \

series VARCHAR,\

editor VARCHAR,\

volume VARCHAR,\

isbn LIST<VARCHAR>\

);")

session.execute("CREATE TABLE IF NOT EXISTS DBLP ( \

id VARCHAR, \

type VARCHAR,\

year INT, \

title VARCHAR,\

authors LIST<VARCHAR>,\

pages frozen<pagesType>,\

booktitle VARCHAR, \

journal frozen<journalType>,\

url VARCHAR, \

cites LIST<VARCHAR>,\

PRIMARY KEY(id) \

);")

session.execute('TRUNCATE dblp;')

with open('DBLP\_clean.json', 'r') as file:

for data in file.readlines():

dataJSON = json.loads(data.replace("'", "''"))

dataJSON['year'] = int(dataJSON['year'])

if (dataJSON['pages']['start'] != None):

dataJSON['pages']['start']= int(dataJSON['pages']['start'])

if (dataJSON['pages']['end'] != None):

dataJSON['pages']['end']= int(dataJSON['pages']['end'])

data = str(dataJSON)

data = data.replace("'", '"')

data = data.replace('""', "''")

data = data.replace('\n', "")

data = data.replace("\_id", "id")

data = data.replace('None', 'null')

data = data.replace("''\"", "\"''")

data = data.replace("\"'',","''\",")

data = data.replace("u\"","\"")

statement = "INSERT INTO dblp JSON '"+ data + "';"

session.execute(statement)

At each step of the loading we had to add changes to the program so that it cleans all the possible “error” we found.

## Queries

#Simple queries (type exercise 1)

1. Find all the ids.

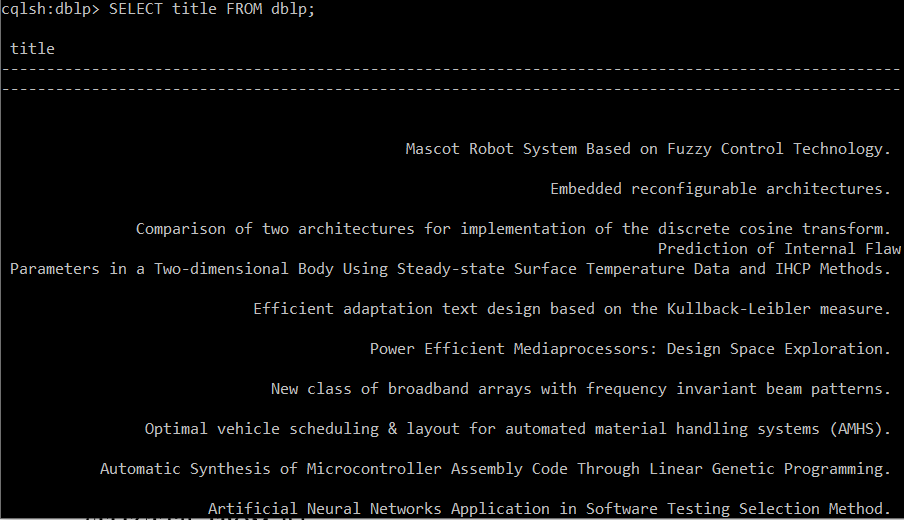
**SELECT id FROM dblp ;**



Here is an extract of the actual output

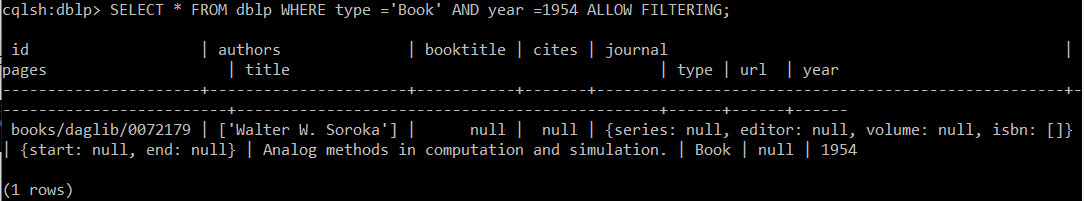
1. Find all the titles of publications

**SELECT title FROM dblp;**



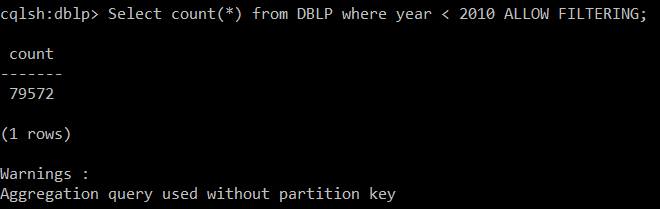
1. Find all books written in 1954.

**SELECT \* FROM dblp WHERE type =’Book’ AND year = 1954 ALLOW FILTERING;**



4. Find the number of books or articles written before 2010

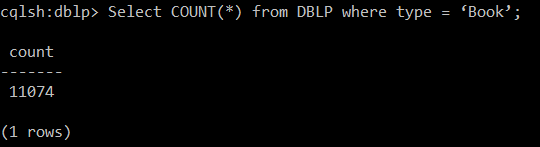
**SELECT count(\*) FROM dblp WHERE year < 2010 ALLOW FILTERING;**



5. Find the number of books in the database

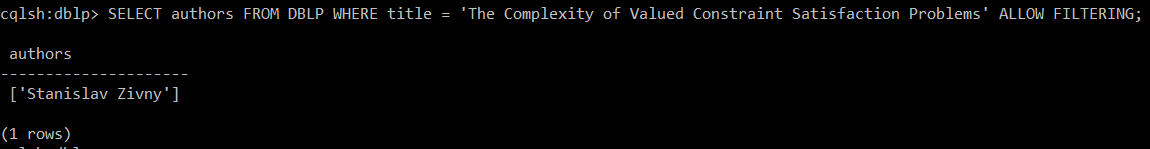
Such as we already create an index on type we don’t need to allow filtereing.

**SELECT COUNT(\*) FROM dblp WHERE type = ‘Book’;**



6. Find the authors of the book "The Complexity of Valued Constraint Satisfaction Problems »

**SELECT authors FROM DBLP WHERE title = 'The Complexity of Valued Constraint Satisfaction Problems' ALLOW FILTERING;**



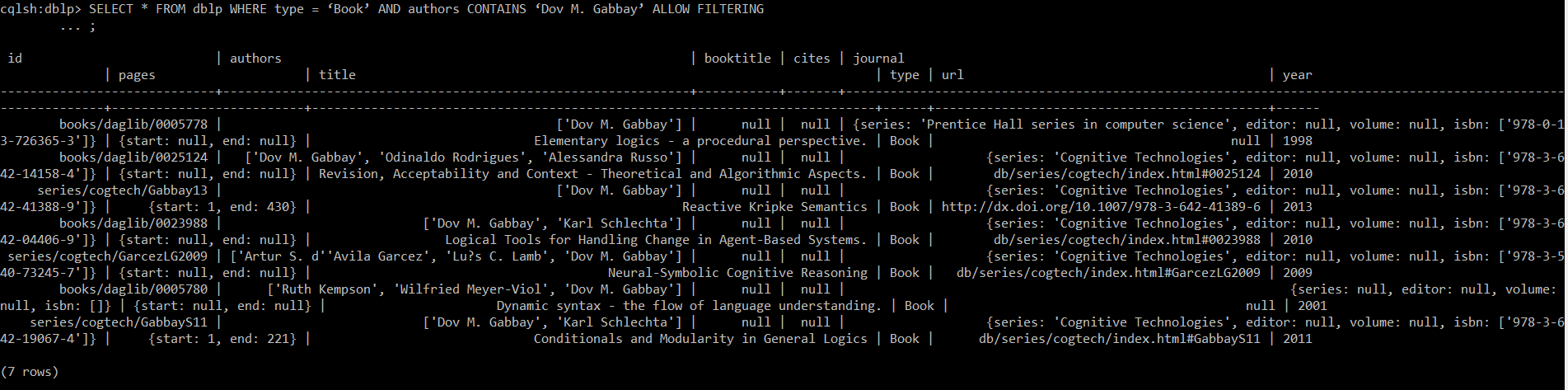
#Complex queries (type exercice 2)

1. Find all books written by Dov M. Gabbay.

**CREATE INDEX ON dblp (authors);**

**CREATE INDEX ON dblp(type);**

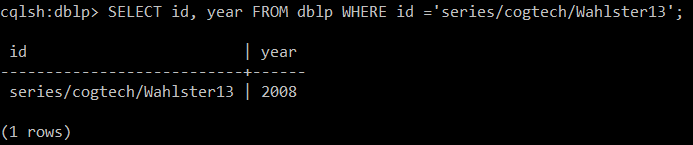
**SELECT \* FROM dblp WHERE type = ‘Book’ AND authors CONTAINS ‘Dov M. Gabbay’ ALLOW FILTERING;**



2. Update the year of a book depending on it’s id.

**UPDATE DBLP SET year = 2008 WHERE id ='series/cogtech/Wahlster13';**





#Hard query (type exercice 3)

1. Create a new UDA to produce an equivalence to ”*GROUP BY + COUNT*” on textual attributes

## CREATE OR REPLACE FUNCTION state\_group(state map<text, int>, type text) CALLED ON NULL INPUT RETURNS map<text, int> LANGUAGE java AS ' Integer val = (Integer) state.get(type); if (val == null) val = 0; else val++; state.put(type, val); return state; ' ;

## CREATE OR REPLACE AGGREGATE state\_group\_and\_max(text) SFUNC state\_group STYPE map<text, int> INITCOND {};

## We had trouble testing this function on Cassandra but normally it works.