Homework 7 | JSON and NoSQL

Objectives: To write queries over the semi-structured data model. To manipulate semi-structured data in JSON and use a NoSQL database system (AsterixDB).

What to turn in: A single file for each question (q1.sqlp, q2.sqlp, …). It should contain commands executable by SQL++.

Due date: Friday, May 31, 2024.

Resources

- data: which contains mondial.adm (the entire dataset), country, mountain, and sea (three subsets)
- documentation for AsterixDB
- guide written by former TAs

Assignment Details

In this homework, you will write SQL++ queries over the semi-structured data model implemented in AsterixDB. Asterix is an Apache project on building a DBMS over data stored in JSON or ADM files.

Mondial Dataset

You will run queries over the Mondial database, a geographical dataset aggregated from multiple sources. As is common in real-world aggregated data, the Mondial dataset is *messy*; the schema is occasionally inconsistent, and some facts may conflict. We have provided the dataset in ADM format (see resources above), converted from the XML format available online, for use in AsterixDB.

Setting up AsterixDB

1. Download and install AsterixDB. Download the file asterixdb-0.9.9 and unzip it anywhere you'd like. (If you are on mac you will need to download the 0.9.8 version).
2. (Extra step for some users:) You need to install the **Java 11 SE Development Kit** if you don't already have it. Follow this new link to the download for your system. Many people will have this already if you've installed the tools for writing Java programs. On Windows, just installing the JDK should be enough. For Mac you may need to do some additional steps as described here.

3. Start an instance of AsterixDB using the **start-sample-cluster.sh** (or **.bat** if you are on Windows) located in the **opt/local/bin** folder.
    a. Once you are in the opt/local/bin folder, you can start an instance by typing in **./start-sample-cluster.sh** (or .bat if you are on Windows) in the terminal
4. When your AsterixDB instance is running you can enter the query interface by visiting 127.0.0.1:19001 in your favorite web browser. It may take a few minutes before you can access the page, while the instance starts up.
5. Download the geographical data from the link in the resources above. The data are JSON data files; you can inspect them using your favorite text editor.
6. Create a dataverse of Mondial data. Copy and paste the text below in the Query box of the web interface. Edit the <path to mondial.adm> to be the location of the mondial.adm file on your computer. The format of this line depends on your operating system. 127.0.0.1 is the ip address of the computer you are running this on, and what follows is the directory path.

    FOR WINDOWS: your path will look like the example in the code below, since Windows paths start with the name of the drive like C:

    FOR MAC: your path will most likely look something like this: "path"="127.0.0.1:///Users/XXX/Desktop/hw7/data/mondial.adm" (Then don't forget the ("format"="adm") part as shown below)

        i. Tip: Go to your mondial.adm file in your Finder > right click > hold option > click "Copy "<file>" as Pathname > paste this path in after "path"="127.0.0.1:// (so it looks like the path above, should be a total of three slashes "/")
7. Then press Run

```
DROP DATAVERSE geo IF EXISTS;
CREATE DATAVERSE geo;

CREATE TYPE geo.worldType AS {auto_id:uuid };
CREATE DATASET geo.world(worldType)  PRIMARY KEY auto_id AUTOGENERATED;
LOAD DATASET geo.world USING localfs
(("path"="127.0.0.1:///Users/suciu/asterixdb/hw7/mondial.adm"),("format"="a
dm"));

 /* Edit the absolute path above to point to your copy of mondial.adm. */
 /* You may need to use '\' instead of '/' in a path for Windows. e.g.,
127.0.0.1://C:\\XXX\\hw7\\mondial.adm. */
```

8. Alternatively, you can use the terminal to run queries rather than the web interface. After you have started Asterix, put your query in a file (say **q1.sqlp**), then execute the query by typing the following command in terminal (notice that the port number here is 19002):

```
curl -v --data-urlencode "statement=`cat q1.sqlp`" --data pretty=true
http://localhost:19002/query/service
```

This will print the output on the screen. If there is too much output, you can save it to a file

```
curl -v --data-urlencode "statement=`cat q1.sqlp`" --data pretty=true
http://localhost:19002/query/service  > output.txt
```

You can now view **output.txt** using your favorite text editor.

9. To reference the geodatabase, use the statement USE geo; before each of your queries to declare the geo namespace. Alternatively, prefix every dataset with geo. Try this query to see if things are running correctly:

```
SELECT y.`-car_code` as code, y.name as name
FROM geo.world x, x.mondial.country y
ORDER BY y.name;
```

10. For practice, run, examine, modify these queries. They contain useful templates for the questions on the homework: make sure you understand them.

```
-- return the set of countries
SELECT x.mondial.country FROM geo.world x;

-- return each country, one by one (see the difference?)
SELECT y as country FROM geo.world x, x.mondial.country y;

-- return just their codes, and their names, alphabetically
-- notice that -car_code is not a legal field name, so we enclose in ` ...
`
SELECT y.`-car_code` as code, y.name as name
FROM geo.world x, x.mondial.country y order by y.name;

-- this query will NOT run...
SELECT z.name as province_name, u.name as city_name
FROM geo.world x, x.mondial.country y, y.province z, z.city u
WHERE  y.name='Hungary';
-- ...because some provinces have a single city, others have a list of
cities; fix it:
```

```
SELECT z.name as province_name, u.name as city_name
FROM geo.world x, x.mondial.country y, y.province z,
            CASE  WHEN is_array(z.city) THEN z.city
                  ELSE [z.city] END u
WHERE  y.name='Hungary';

-- same, but return the city names as a nested collection;
-- note correct treatment of missing cities
-- also note the convenient LET construct (see SQL++ documentation)
SELECT z.name as province_name, (select u.name from cities u) as cities
FROM geo.world x, x.mondial.country y, y.province z
LET cities = CASE  WHEN z.city is missing THEN []
                   WHEN is_array(z.city) THEN z.city
                   ELSE [z.city] END
WHERE  y.name='Hungary';
```

11. To shutdown Asterix, simply run **stop-sample-cluster.sh** in the terminal. The script is located in **opt/local/bin** (or **opt\local\bin\stop-sample-cluster.bat** on windows).

Problems (100 points)

Notes:

- **For all questions asking to report free response-type questions, please leave your responses in comments**
- **The order of the keys can differ due to how AsterixDB works internally.**
- **To count rows, you can copy the entirety of the UI output into a text editor and note the line number.**

Use only the mondial.adm dataset for problems 1-5.

1. Retrieve the names of all cities located in Peru, sorted alphabetically.

   Name your output attribute **city**.

   [Result Size: 30 rows of **{"city":...}**]

2. For each country, return its name, its population, and the number of religions sorted alphabetically by country. Report 0 religions for countries without religions.

   Name your output attributes **country**, **population**, **num_religions**.

[Result Size: 238 rows of **{"num_religions":..., "country":..., "population":...}**]

3. For each religion return the number of countries where it occurs; order them in a decreasing number of countries.

   Name your output attributes **religion**, **num_countries**.

   [Result size: 37 of **{"religion':..., "num_countries":...}**]

4. For each ethnic group, return the number of countries where it occurs, as well as the total population world-wide of that group. Hint: you need to multiply the ethnicity's percentage with the country's population. Use the functions **float(x)** and/or **int(x)** to convert a **string** to a **float** or to an **int**.

   Name your output attributes **ethnic_group**, **num_countries**, **total_population**. You can leave your final **total_population** as a **float** if you like.

   [Result Size: 262 of **{"ethnic_group":..., "num_countries":..., "total_population":...}**]

5. Find all countries bordering two or more seas. Here you need to join the "sea" collection with the "country" collection. For each country in your list, return its code, its name, and the list of bordering seas, in decreasing order of the number of seas.

   Name your output attributes **country_code**, **country_name**, **seas**. The attribute **seas** should be a list of objects, each with the attribute sea.

   [Result Size: 74 rows of **{"country_code":..., "country_name":..., "seas": [{"sea":...}, {"sea":...}, ...]}**]

Submission Instructions

Write your answers in a file for each question: **q1.sqlp, … , q5.sqlp** to Gradescope.