DATA 514 Homework 6: JSON, NoSQL, and AsterixDB

Objectives: To practice writing queries over the semi-structured data model. To be able to manipulate semistructured data in JSON and practice using a NoSQL database system (AsterixDB).

Assigned date: March 4, 2019

Due date: Monday, March. 11, 2019. You have 1 week for this homework.

What to turn in:

A single file for each question, i.e., `hw6-q1.sqlp`, `hw6-q2.sqlp` etc in the `submission` directory. It should contain commands executable by SQL++, and should contain comments for text answers (delimited by `--` as in SQL).

Resources

- Starter code: which contains `monidal.adm` (the entire dataset), `country`, `mountain`, and `sea` (three subsets)
- documentation for AsterixDB

Assignment Details

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

A. Setting up AsterixDB (0 points)

AsterixDB does not work well on Windows, so option 1 below discuss how to run AsterixDB on a Linux VM. Option 1 is likely to be easiest, so that approach is recommended for Windows users. For Mac users, option 2 is likely to be easiest, so that approach is recommended for Mac users.

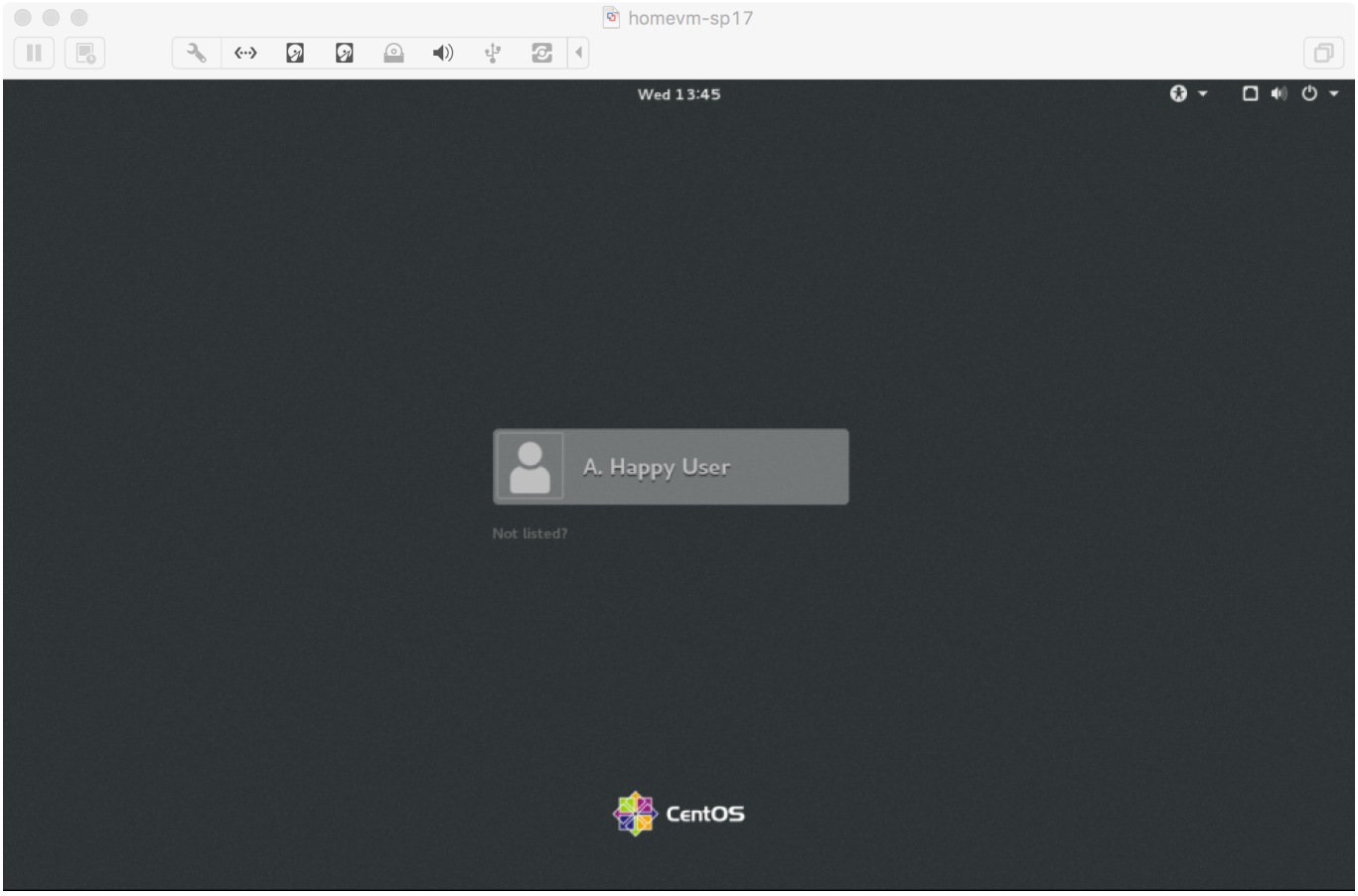
Option 1 (Recommended for Windows users)

One easy way to get access to a Linux machine when you have Windows is to use VMware. VMware allows you to run Linux (a competing operating system to Windows) as an ordinary Windows program. VMware does this by running the real Linux operating system in a virtual environment, where all of the I/O devices (disks, network, keyboard, etc.) are simulated by VMware. For example, what Linux thinks is a hard disk drive is actually just a file that VMware creates in Windows.

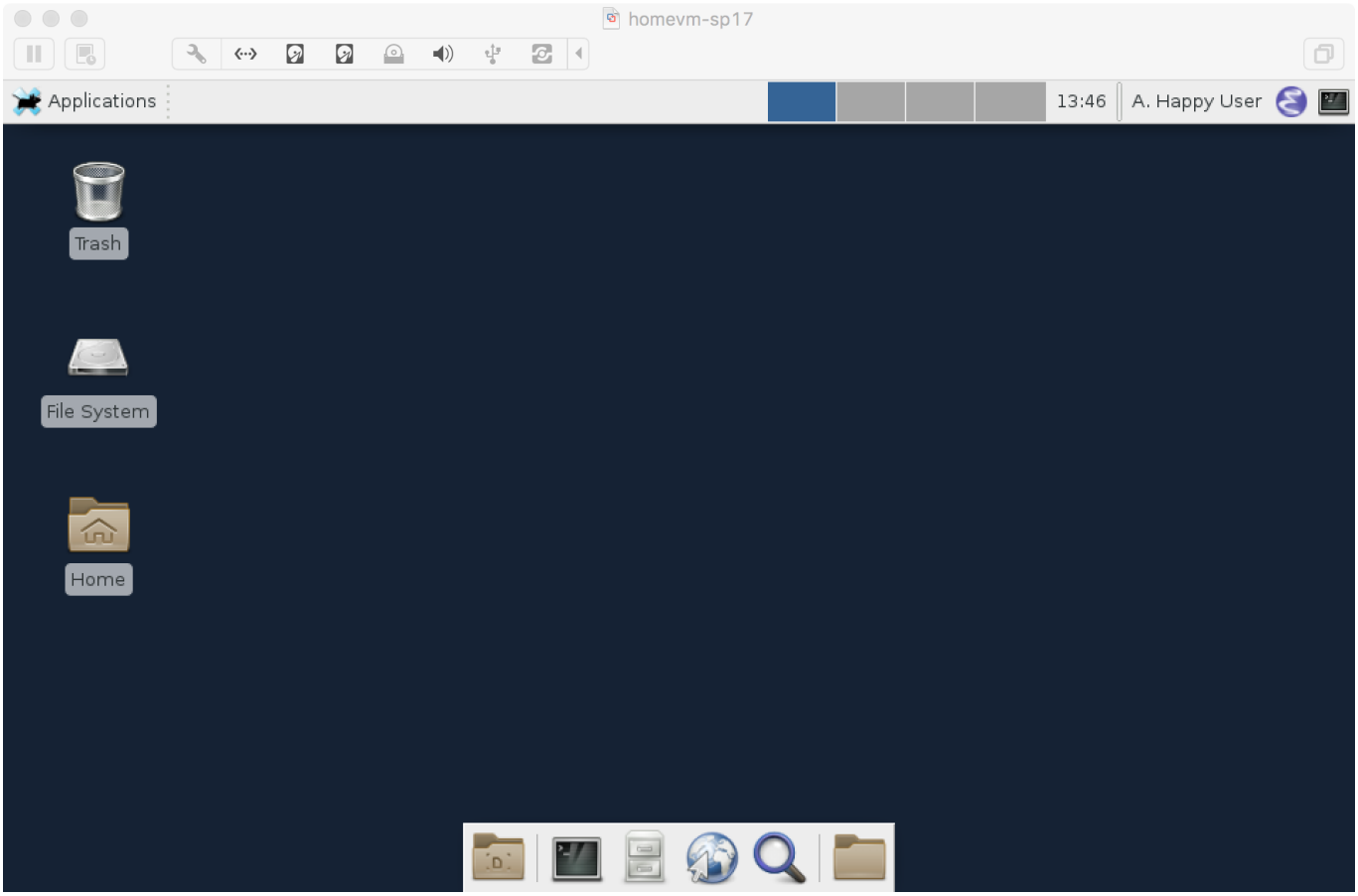
As a student in CSE 514, you should be able to download and install VMware from this web site (using your UW email address) at no cost. For Windows users, install the most recent version of "VMware Workstation". For Mac users, install the most recent version of "VMware Fusion".

In order to launch Linux in VMware, you will need a virtual machine image to run. Grab the one from this page. In addition to Linux, it also has AsterixDB already installed for you! Once you download and unzip the virtual machine image file, you can run it in VMware by clicking File > "Open and Run" on the menu. (You may see an error message about an ide disk not being found. Just ignore that.)

Once the VM instance launches, you should see a login window like this one:



Click on "A. Happy User" to log in. Once you do so, you will see a Linux desktop, which looks like this:



At the bottom of the screen, there is a row of icons. The second from the left is the terminal program. The fourth from the left is the web browser. Those are the only two that you will need for this assignment.

Open the terminal program, and execute this command:

```
ssh-keygen -t rsa -P ""
```

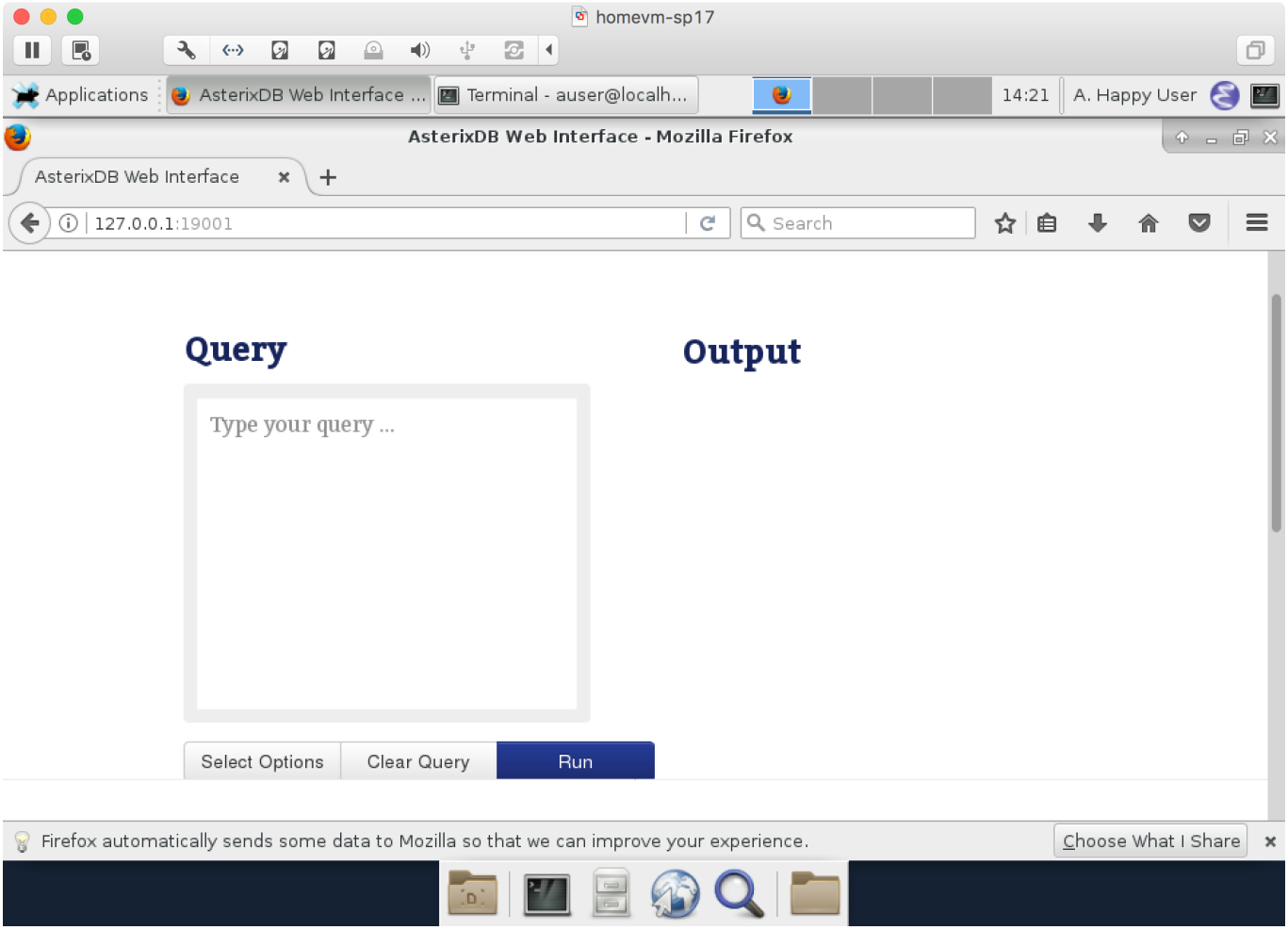
It will ask you if you want to name the file it creates id_rsa. Just press ENTER and accept the default name.

Now run the follwowing commands. They should each run without error.

```
cp .ssh/id_rsa.pub .ssh/authorized_keys
sudo chmod 777 /usr/local/asterixdb/clusters/local
sudo chmod 666 /usr/local/asterixdb/clusters/local/*.xml
sudo chmod 666 /usr/local/asterixdb/clusters/demo/*.xml
sudo chmod 666 /usr/local/asterixdb/conf/managix-conf.xml
managix configure
managix validate
managix create -n my_asterix -c
/usr/local/asterixdb/clusters/local/local.xml
```

That last command may take some time to complete. After it runs, you should see Status: ACTIVE, which tells you that AsterixDB is now running in your Linux virtual machine.

Finally, click on the browser window icon at the bottom of the screen. And navigate to the URL http://127.0.0.1:19001. You should see a page like this one where you can type in queries.



If you want to test that it works, you can try running the following query:

```
SELECT VALUE ds FROM Metadata.`Dataset` ds;
```

This query will print out information about all of the datasets that already exist in your AsterixDB instance. You will add your own datasets below.

If you later shut down and restart VMware, you can use the following commands to restart AsterixDB:

```
managix stop -n my_asterix
managix start -n my_asterix
```

Option 2 (Recommended for Mac Users)

For Mac users, it is also possible to install AsterixDB directly on your machine. To do so, start by downloading AsterixDB from this page. Click on the downloaded file to unzip it in your Downloads directory.

Next, run the following command in the Terminal to start your Asterix cluster:

```
cd Downloads/apache-asterixdb-0.9.4
cd opt/local/bin
```

```
./start-sample-cluster.sh
```

Now, navigate in your browser to http://localhost:19001, and you should see a web page where you can type in SQL++ queries.

If you want to test that it works, you can try running the following query:

```
SELECT VALUE ds FROM Metadata.`Dataset` ds;
```

This query will print out information about all of the datasets that already exist in your AsterixDB instance. You will add your own datasets below.

When you want to stop your AsterixDB cluster, run this command:

```
opt/local/bin/stop-sample-cluster.sh
```

Loading Data

Copy, paste, and edit the red text in the Query box, the press Run:

```
DROP DATAVERSE hw6 IF EXISTS;
CREATE DATAVERSE hw6;
USE hw6;
CREATE TYPE worldType AS {auto_id:uuid };
CREATE DATASET world(worldType)  PRIMARY KEY auto_id AUTOGENERATED;
LOAD DATASET world USING localfs
        (("path"="127.0.0.1:///<path to mondial.adm>, e.g.,
/514/hw6/mondial.adm"),("format"="adm"));
/* Edit the absolute path above to point to your copy of mondial.adm. */
/* Use '/' instead of '\' in a path for Windows. e.g.,
C:/514/hw6/mondial.adm. */
```

```
/* Note: if you type one command at a time, then end it with a ";" */
USE hw6;
```

Test queries

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
USE hw6;
```

```
SELECT x.mondial.country FROM world x;
```

```
-- return each country, one by one (see the difference?)
USE hw6;
SELECT y as country FROM 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 ` … `
USE hw6;
SELECT y.`-car_code` as code, y.name as name
FROM world x, x.mondial.country y order by y.name;
```

```
-- this query will NOT run...
USE hw6;
SELECT z.name as province_name, u.name as city_name
FROM 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:

USE hw6;
SELECT z.name as province_name, u.name as city_name
FROM 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)

USE hw6;
SELECT z.name as province_name, (select u.name from cities u)
FROM 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';
```

B. Problems (100 points)

1. (5 points) Retrieve all the names of all cities located in Peru, sorted alphabetically. [Result Size: 30 rows]

2. (10 points) For each country return its name, its population, and the number of religions, sorted alphabetically by country. [Result Size: 238 rows]

3. (10 points) For each religion return the number of countries where it occurs; order them in decreasing number of countries. [Result size: 37]

4. (10 points) 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 x to a float or to an int. [Result Size: 262]

5. (10 points) Compute the list of all mountains, their heights, and the countries where they are located. Here you will join the "mountain" collection with the "country" collection, on the country code. You should return a list consisting of the mountain name, its height, the country code, and country name, in descending order of the height. [Result Size: 272 rows]

6. (10 points) Compute a list of countries with all their mountains. This is similar to the previous problem, but now you will group the moutains for each country; return both the mountain name and its height. Your query should return a list where each element consists of the country code, country name, and a list of mountain names and heights; order the countries by the number of mountains they contain [Result Size: 238]

7. (10 points) 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 lits list of bordering seas, in decreasing order of the number of seas. [Result Size: 74]

8. (10 points) Return all landlocked countries. A country is landlocked if it borders no sea. Order your answers in decreasing order of the country's area. Note: this should be an easy query to derive from the previous one. [Result Size: 45]

9. (10 points) For this query you should also measure and report the runtime; it may be approximate (expect it around 10'-30') . Find all distinct pairs of countries that share both a mountain and a sea. Your query should return a list of pairs of country names. Avoid including a country with itself, like in (France,France), and avoid listing both (France,Korea) and (Korea,France) (not a real answer). [Result Size: 7]

10. (13 points) Create a new dataverse called hw6index, then run the following commands:

```
USE hw6index;
CREATE TYPE countryType AS OPEN {
    `-car_code`: string,
    `-area`: string,
    population: string
};
CREATE DATASET country(countryType)
   PRIMARY KEY `-car_code`;
CREATE INDEX countryID ON country(`-car_code`) TYPE BTREE;
LOAD DATASET country USING localfs
    (("path"="127.0.0.1://<path to country.adm>, e.g.,
/514/hw6/country.adm"),("format"="adm"));
```

This created the type `countryType`, the dataset `country`, and a `BTREE` index on the attribute `-car_code`, which is also the primary key. Both types are OPEN, which means that they may have other fields besides the three required fields `-car_code`, `-area`, and population.

Create two new types: `mountainType` and `seaType`, and two new datasets, `mountain` and `sea`. Both should have two required fields: `-id` and `-country`. Their key should be autogenerated, and of type `uuid` (see how we did it for the mondial dataset). Create an index of type `KEYWORD` (instead of `BTREE`) on the `-country` field (for both `moutnain` and `sea`). Turn in the complete sequence of commands for creating all three types, datasets, and indices (for `country`, `mountain`, `sea`).

11. (1 points) Re-run the query from 9. (“pairs of countries that share both a mountain and a sea”) on the new dataverse `hw6index`. Report the new runtime. [Result Size: 7]

12. (1 points) Modify the query from 11. to return, for each pair of countries, the list of common mountains, and the list of common seas. [Result Size: 7]