CS 122A: Introduction to Data Management – Winter 2017

Homework 8: NoSQL (100 points)

Due Date: **Friday, Mar 17 (11:59 PM)**

Submission

All HW assignments should contain both your student ID and your name and must be submitted online, formatted in PDF document, through the associated dropbox on EEE. See the table below for the HW submission opportunities. Note that after 11:59 PM on Sunday evening no further HW submissions will be accepted. (We will be releasing the solution at that time so you can study for the third and final midterm exam.) Please strive to get all your work in on time!

Date / Time	**Grade Implications**
Friday, Mar 17 (11:59 PM)	Full credit will be available
Saturday, Mar 19 (11:59 PM)	5 points will be deducted
Sunday, Mar 21 (11:59 PM)	10 points will be deducted

NoSQL (SQL) [100 pts]

Congratulations! If you are reading this, you must have successfully finished reading and running the exercises in the SQL++ Primer. Business is booming, thanks to recent trends in world politics, and TopicalBirds.com needs to scale beyond MySQL's capabilities. In this assignment you will explore some of the virtues of "NoSQL" systems in situations like this. Enter AsterixDB…! As you read through the assignment, answering its questions and writing queries, you will want to refer to two other sources of information -- namely, the solution to HW #1 (the E-R schema) and the corresponding translated relational schema that you used for the SQL-based HW assignments. You may also find that the queries that you're asked to write here seem hauntingly familiar (deja vu!), so you can refer to the relevant earlier assignment(s) if you are wondering if your answers here are on track. (We will also specify the number of expected answer records for each query to help you with that.)

0. Load dataset

To save you time, we have transformed the TopicalBird database schema to ADM (the Asterix Data Model). You need to download our script and populate the dataset. After you've started your sample AsterixDB cluster, the next thing you will need to do is to create the dataverse, datatypes, and datasets for this NoSQL translation of the schema. To do so, you should open `HW8_TableInit.txt` and copy its contents into the AsterixDB web-based query interface. Make sure that the `Query Language` selected there is "SQL++" and then click 'Run'. After execution, you should see something that resembles the following result (except you'll see one fewer Bird):

Now open `HW8_DataInit.txt` and copy its contents into the query interface. As before, execute it by clicking `Run`. You should see something similar to the following result:

The actual query output results should be:

```
{ "ChirpCount": 160 }

{ "TopicsCount": 65 }

{ "WatchersCount": 10 }

{ "BirdsCount": 24 }

{ "AdsCount": 10 }

{ "BirdListenCount": 97 }

{ "TopicListenCount": 41 }
```

Compare these counts with your own output to make sure that you that have the complete set of datasets. All datasets reside in the `TopicalBird` dataverse, so be sure to put `USE TopicalBird;` in front of your queries when you run them. Here is an example of how that looks:

```
USE TopicalBird;

SELECT * FROM Birds WHERE Birds.name.first_name = "Barack";
```

It's now question-answering time! Answer each of the following questions after first following the instructions (if any) that they contain.

1. Which dataset/s in AsterixDB can still be classified as being in 1NF based on the DDL you've been given? Write down the dataset name(s) and briefly explain your answer(s).

ANSWER:

1. **Chirps;**

2. **Topics;**
3. **Ads;**
4. **BirdListen;**
5. **TopicListen.**

Since all these dataset only contain atomic value, they are in 1NF

2. Looking back at the E-R diagram from the conceptual design, compare and contrast the MySQL schema and the AsterixDB schema (given in the DDL). You will find that we have made some different design decisions here in the NoSQL database case. Briefly explain how we have captured the information from the following E-R entities differently in AsterixDB and what the benefits) of our new design probably are:

Peacock:

Difference:

In the **E-R entities**, “Peacock” is been implemented as an Entity separately from the Entity “Bird,” and “Peacock” has a multi-value attribute “variety.” Also, these two entities are connected by “ISA” relation. However, in the **AsterixDB**, “Peacock” is not been implemented separately as an Entity, instead, its attribute “variety” is been implemented as a term inside the Type “BirdType.” Then, “BirdType” is been taken as an parameter to the dataset “Birds,” so that “variety” would become an optional attribute of “Birds”

Benefits:

As we include variety as an optional attribute in Birds, we don’t need to create an entity for “Peacock”, which means, we don’t need to create separate table for “Peacock” and “Variety”. Therefore, when we need to write query to access Peacock, the **speed will increase** and the **cost will be cheaper**.

About:

Difference:

In the **E-R entities**, “About” is been implemented as a Relation that connects “Chirp” and “Topic.” However, in the **AsterixDB**, there is no “Chirp” or “Topic” entities, instead, there are “ChirpType” and “TopicType,” and “about” is been implemented as a term inside “ChirpType.” Then, “ChirpType” is been taken as a parameter to the dataset “Chirp,” so that “about ” would become an optional attribute of “Chirp”

Benefits:

As we include variety as an optional attribute in Birds, we don’t need to create an entity for “About,” which means that we don’t need to create separate table for “About.” Therefore, when we want to write queries to access the topic of chirp, we don't need to refer to the relation table "About," but we can directly access the Chirp's Topic from "Chirp" table(by the attribute "about" which is the topic tid). These is **more fast** and **less expansive** than the previous version.

3. Write a query to print the name, gender, and birthdate of birds who live on "Alicia Pass" street. (Check the DDL of the dataset to determine the attribute names. This doc link might help for the `contains` function that your query will need.) [Result size: 1]

Answer:

1. SQL++ Query:

```
USE TopicalBird;

SELECT b.name, b.gender, b.birth_date

FROM Birds b

WHERE b.address.street = "Alicia Pass";
```

2. Result:

```
{  "name":  {  "first_name":  "Elizabeth",  "last_name":  "Smith"  },  "gender":  "F",
"birth_date": "1972-09-14T03:35:45.000Z" }
```

4. Write a query that, for each watcher, prints the tag of the watcher and the set of Ad id(s) for ads that are owned by that watcher. (You may want to look back at "Query 3: Nested Outer Join" in the SQL++ Primer for reference.) [Result size: 10]

Answer:

1. SQL++ Query:

```
USE TopicalBird;

SELECT w.wtag, (SELECT a.aid

                From Ads a

                WHERE a.wtag = w.wtag) AS aidSet

FROM Watchers w

GROUP BY w.wtag;
```

2. Result:

```
{ "wtag": "anthonylang", "aidSet": [  ] }
{ "wtag": "burtonangel", "aidSet": [ { "aid": 2 } ] }
{ "wtag": "cgregory", "aidSet": [ { "aid": 4 } ] }
{ "wtag": "fosterdeanna", "aidSet": [  ] }
{ "wtag": "gharper", "aidSet": [ { "aid": 7 } ] }
{ "wtag": "hjones", "aidSet": [  ] }
{ "wtag": "matthew99", "aidSet": [  ] }
{ "wtag": "njoyce", "aidSet": [ { "aid": 0 }, { "aid": 6 } ] }
{ "wtag": "rodriguezwilliam", "aidSet": [ { "aid": 9 }, { "aid": 1 }, { "aid": 3 } ] }
{ "wtag": "xbush", "aidSet": [ { "aid": 8 }, { "aid": 5 } ] }
```

5. Write a query that prints the tag, gender, and birthdate of those birds who have chirped about either of the topics “Surface” or “Kindle”. (You can refer to the SQL++ manual to find more information about various useful operators that are available in SQL++). [Result size: 16]

Answer:

1. SQL++ Query:

```
USE TopicalBird;
SELECT DISTINCT b.btag, b.gender, b.birth_date
FROM Birds b, Chirps c, Topics t
WHERE b.btag = c.btag and
        t.tid IN c.about and
        t.name IN ["Surface", "Kindle"];
```

2. Result:

```
{ "btag": "judy60", "gender": "F", "birth_date": "1976-02-08T04:19:50.000Z" }
{ "btag": "thenry", "gender": "F", "birth_date": "1961-12-25T06:29:58.000Z" }
{ "btag": "larry82", "gender": "M", "birth_date": "1964-09-20T23:01:00.000Z" }
{ "btag": "austin73", "gender": "M", "birth_date": "1964-12-16T10:23:56.000Z" }
{ "btag": "qsanchez", "gender": "F", "birth_date": "1968-08-16T22:26:39.000Z" }
{ "btag": "natalie49", "gender": "M", "birth_date": "1969-12-21T02:05:03.000Z" }
{ "btag": "uatkinson", "gender": "M", "birth_date": "1971-12-03T19:53:31.000Z" }
{ "btag": "margaret47", "gender": "M", "birth_date": "1994-06-18T00:41:15.000Z" }
{ "btag": "bentonmichael", "gender": "F", "birth_date": "1978-06-11T15:22:46.000Z" }
{ "btag": "bishopcheyenne", "gender": "F", "birth_date": "1986-12-11T09:57:44.000Z" }
{ "btag": "vchang", "gender": "M", "birth_date": "1969-05-06T15:27:41.000Z" }
{ "btag": "dfowler", "gender": "M", "birth_date": "1981-06-17T23:19:17.000Z" }
{ "btag": "laura43", "gender": "F", "birth_date": "1972-09-14T03:35:45.000Z" }
{ "btag": "bnavarro", "gender": "M", "birth_date": "1950-08-24T22:36:48.000Z" }
{ "btag": "jessewilson", "gender": "M", "birth_date": "1955-01-07T04:07:24.000Z" }
{ "btag": "colemancraig", "gender": "M", "birth_date": "1973-03-19T17:44:25.000Z" }
```

6. Write a query that lists the 5 biggest parrots (i.e., the top 5 birds based on their parroted chirp counts) by printing their bird tags along with their associated parrot counts in descending order. [Result size: 5]

Note: Recall from Friday's lecture that AsterixDB, by providing open datatyping, gives developers more flexibility when working with different data formats. In our TopicalBird dataset, we snuck one such attribute into the Chirps dataset so that you can experience this feature first-hand. The 'parrot_from' Chirp attribute is not defined in ChirpType, yet the records carry that attribute and they can be inserted smoothly into the Chirps dataset. Also, you will see that you can query this attribute just as you can the predefined attributes. To help you understand what to expect, here's what the full definition of the attribute would be if it had actually been predefined:

```
parrot_from: {
        btag: string,
        cno: int
}
```

Answer:

1. SQL Query:

```
USE TopicalBird;
SELECT c.btag, count(*) as count
FROM Chirps c
WHERE c.parrot_from IS NOT NULL
GROUP BY c.btag
ORDER BY count DESC
LIMIT 5;
```

2. Result:

```
{ "btag": "jessewilson", "count": 6 }
{ "btag": "judy60", "count": 6 }
{ "btag": "austin73", "count": 6 }
{ "btag": "swolf", "count": 6 }
{ "btag": "bishopcheyenne", "count": 5 }
```

7. To help in identifying potential Russian bot-birds, write a query that prints the tags and email addresses of those birds who have done nothing but parrot chirps from bird “realDonaldTrump” (i.e., birds all of whose chirps are parroted Trump chirps). [Result size: 2]

Answer:

1. SQL Query:

```
USE TopicalBird;
SELECT b.btag, b.email
FROM Birds b
WHERE NOT EXISTS(
SELECT *
FROM Chirps c
WHERE b.btag = c.btag AND c.parrot_from.btag IS NULL)
AND NOT EXISTS(
SELECT *
FROM Chirps c
WHERE b.btag = c.btag AND c.parrot_from.btag != 'realDonaldTrump');
```

2. Result:

```
{ "btag": "jessewilson", "email": "jessewilson@garcia.com" }
{ "btag": "BarackObama", "email": "BarackObama@white-house.gov" }
{ "btag": "swolf", "email": "swolf@fuller-lester.com" }
```

[EXTRA CREDIT]

8. (20 points) Come up with an interesting/creative question of your own about this collection of data -- something that was not asked in the previous SQL-based assignments -- and then write an AsterixDB query to answer it. Show both the English and SQL++ versions of your query as well as the results that you obtain. [Result size: Up to you!]

Question(expected result: 1):

TopicalBird recently receive some Sponsorship money from a food market located in **CN**, as promised, TopicalBird needs to send advertisement to at **least** one Watcher per month. After discussion, TopicalBird Decides to send advertisement through email to the Watcher who pay the **least membership fee**. Find this watcher, who pay the least membership fee and live in **CN**. Print his/her **tag**, along with his/her **business name** and **email address**.

1. SQL++ Query:

```
USE TopicalBird;

SELECT  w.wtag, w.bname, w.email

FROM Watchers w

WHERE w.address.country = 'CN'

ORDER BY w.fee

ASC

LIMIT 1;
```

2. RESULT:

```
{ "wtag": "burtonangel", "bname": "Maynard-Morrison", "email": "burtonangel@perez.com"
}
```