CompSci 516: Homework 1

SQL and RA

Fall 2019

Total Points : 100

- Posted on : Thursday, September 5, 2019 (Part-1 and 2)
- Due on :
  - PART-1: Parser, Q1-Q3: Tuesday, September 17, 2019 11:55 PM
  - PART-2: Q4: Tuesday, September 24, 2019 11:55 PM
  - PART-3: Q5: Tuesday, October 1, 2019 11:55 PM

Course Policy Reminder

Please remember that you need to **strictly** follow the following rules while you are working on the homework:

1. ☐ You can NOT discuss the solution of any question with any other student or can NOT work in a group to solve the problems.
   You CAN only install a software together with other students and test whether toy examples run (not any homework question).
2. ☐ You can NOT show your solution to any question to any other student.
3. ☐ You can NOT see the solution to any question from any other student.
4. ☐ You can NOT find the solution to any question from the Internet or any other sources.
   Note that you can and need to learn programming languages, frameworks, and libraries using online tutorials, and then need to write your own code.
5. ☐ If you are unsure about what is allowed, you need to send the instructor an email and ask about it.
6. ☐ Of course, feel free to post questions on piazza (as many times as you need) if anything is confusing or if you need help! Anonymous questions are fine but do not copy your solutions. If you have specific questions about your solution, you can send private questions on piazza.

1 Overview

In this assignment, you will get a hands-on experience with Postgres working on an end-to-end data processing task. Here is the bird's-eye view of the assignment:

1. acquire raw data
2. pre-process raw data
3. analyze
4. present

You will also practice some more RA queries in this assignment.

2 Download DBLP Dataset

We will be using a DBLP dataset as the raw data. It provides open bibliographic information of major Computer Science journals and proceedings (`http://dblp.uni-trier.de/`).
Download the following XML file and corresponding file-description DTD file from Sakai: Resources → Homework → HW1: dblp-2019-09-05.xml.gz and dblp.dtd.
The formats of the XML file are described in the following document:
`http://dblp.uni-trier.de/xml/docu/dblpxml.pdf`

3 Install Postgres

You should already have postgres from Lab-1 on SQL.

After the installation, create a new user (**dblpuser**) and a new database (**dblp**) in your local machine.

4 Load DBLP Dataset into Postgres

In this section, you will write a program that reads the xml file, cleans the data and stores the pre-processed information into the `dblp` database, which we have created in the previous section.

4.1 Schema

You will create three tables in the `dblp` database: Inproceedings, Article, and Authorship.
Here is the schema of each of the tables. The attribute with an underline is the primary key. Data type is specifed in the parentheses.

- **Inproceedings** (pubkey(TEXT), title(TEXT), booktitle(TEXT), year(INT))
- **Article** (pubkey(TEXT), title(TEXT), journal(TEXT), year(INT))
- **Authorship** (pubkey(TEXT), author(TEXT))

What do these relations stand for?
The DBLP dataset contains publication records in journals, conferences, books, informal publications (like tech report), etc. In this assignment, we will be importing conferences and journals into Postgres.

- Conference papers are stored into the Inproceedings table
- Journal papers are stored into the Article table

What are the attributes?
Consider the following article element in the xml file.

```
<article mdate="2017-05-28" key="journals/spe/WatsonC04">
    <author>Bruce W. Watson</author>
    <author>Loek G. Cleophas</author>
    <title>SPARE Parts: a C++ toolkit for string pattern recognition.</title>
    <pages>697-710</pages>
    <year>2004</year>
    <volume>34</volume>
    <journal>Softw., Pract. Exper.</journal>
    <number>7</number>
    <ee>https://doi.org/10.1002/spe.590</ee>
    <url>db/journals/spe/spe34.html#WatsonC04</url>
</article>
```

We will be using `key, title, journal, year` attributes in the above article element, and each of them will map to `pubkey, title, journal, year` in our Article table accordingly.

Now let's look at an inproceedings element in the xml file.

```
<inproceedings mdate="2015-07-10" key="conf/gecco/Christmas15">
    <author>Jacqueline Christmas</author>
    <title>Genetic C Programming with Probabilistic Evaluation.</title>
    <pages>1371-1372</pages>
    <year>2015</year>
    <booktitle>GECCO (Companion)</booktitle>
    <ee>http://doi.acm.org/10.1145/2739482.2764642</ee>
    <crossref>conf/gecco/2015c</crossref>
    <url>db/conf/gecco/gecco2015c.html#Christmas15</url>
</inproceedings>
```

We will be using `key, title, booktitle, year` attributes in the above inproceedings element, and each of them will map to `pubkey, title, booktitle, year` in our Inproceedings table accordingly.

Given those two elements above, your program should generate the following rows in each of the three tables:

Article

pubkey	title	journal	year
journals/spe/WatsonC04	SPARE Parts: a C++ toolkit for string pattern recognition.	Softw., Pract. Exper.	2004

Inproceedings

pubkey	title	booktitle	year
conf/gecco/Christmas15	Genetic C Programming with Probabilistic Evaluation	GECCO (Companion)	2015

Authorship

pubkey	author
journals/spe/WatsonC04	Bruce W. Watson
journals/spe/WatsonC04	Loek G. Cleophas
conf/gecco/Christmas15	Jacqueline Christmas

4.2 External Libraries

You will need to use external libraries to parse the XML file and to import data into the Postgres database (you can use Java or Python, and any library). **You need to learn how to use these parsers using online tutorials, and then need to write your own code**.

For example, if you are writing a Python script to convert the XML file into Postgres, then you could use the `xml.sax` Python library as a XML parser and `psycopg2` as a Python–Postgres connector.

If you are using Java and eclipse (`https://www.eclipse.org`), you need to include Java Database Connectivity (JDBC) driver, and can use libraries like SAXParsers and JAXB.

5 Analyze DBLP Dataset via SQL Query

In this section, you will run SQL queries and analyze the DBLP dataset stored in the `dblp` database. The questions are described in the following four sections. Please find the submission sheet homework1.xml from sakai and fulfill the question marks. After you've finished your sheet, upload to the Gradescope and it will automatically grade your sql queris. Each question specifies the SQL format as a list of attributes (e.g., $[A_1(TYPE_1), A_2(TYPE_2), ..., A_n(TYPE_n)]$, which means "SELECT ... AS A_1, ... AS A_2, ..., ... AS A_n FROM ..." and each attribute A_i should follow the data type $TYPE_i$ if specified.). Please follow the SQL format, so that the autograder can successfully grade your answers. We also require uploading the query result in case we need to double-check your query. So write your query in the $< sql ><![CDATA[\ldots]] >< /sql >$ block, and paste your query result in the $< result > \cdots < /result >$ block.

Q1. Count the number of tuples

- Q1a. Count the number of tuples in Inproceedings.

SQL format: A single attribute [*cnt*]

- Q1b. Count the number of tuples in Article

  SQL format: A single attribute [*cnt*]

- Q1c. Count the number of tuples in Authorship

  SQL format: A single attribute [*cnt*]

Q2. Change schema (add a column)

- Q2a. Add a column "Area" in the Inproceedings table.

  SQL format: No format requirement

- Q2b. Populate the column "Area" following the table below. If there is no match, then set it to "UNKNOWN". Note: you should use the "**booktitle**" tag of inproceedings entries to do **exact** matching to tell areas. For example, among booktitles "FOCS", "SSDI/FOCS" and "FOCS Workshop", only "FOCS" will be categorized into Theory, and the remaining two are UNKNOWN.

  SQL format: No format requirement

Area	**Conference Name**
Database	SIGMOD Conference VLDB ICDE PODS
Theory	STOC FOCS SODA ICALP
Systems	SIGCOMM ISCA HPCA PLDI
ML-AI	ICML NIPS AAAI IJCAI

Q3. Queries

Write a single SQL query (with sub-queries if needed) to solve each question.

- Q3a. Find the number of authors who published in exactly two of the four areas (do not consider "UNKNOWN").

  SQL format: A single attribute $[cnt]$

- Q3b. Find the number of authors who wrote more journal papers than conference papers (irrespective of research areas). You should take into account the case that an author wrote some journal papers while no conference papers.

  SQL format: A single attribute $[cnt]$

- Q3c. Among the authors who have published at least one "Database" paper (in any year), find the top-5 authors who published the most number of papers (journal OR conference) since the year 2000 (including the year 2000).

  SQL format: The author name and his number of papers $[author, cnt]$ ordered by $[cnt]$ in decreasing order. If any two tuples have the same $[cnt]$, they should be ordered by $[author]$ in increasing order.

Q4. Visualize the query results

Besides the SQL queries and results, you also need to submit q4.pdf including the output graphs (Q4a-c) and table (Q4d) to GradeScope.

You can use Excel or any other software to plot graphs. You should be able to easily copy/export the results of SQL queries to Excel.
Write a single SQL query (with sub-queries if needed) to solve each question.

- Q4a. Plot a linegraph with two lines, one for the number of journal papers and the other for the number of conference papers in every decade starting from 1950.

  SQL format: The decade, the number of journal papers and the number of conference papers $[decade(INT), num\_journals, num\_confs]$

  - The decades will be 1950-1959, 1960-1969, ... , 2000-2009, 2010-. Please use the start year to indicate the decade. For example, use 1950 to represent 1950-1959.

- Q4b. Plot a barchart showing how the average number of **"collaborators"** varied in each decade starting from 1950 in each of the four areas.

  SQL format: The decade, the area and the average [decade(INT), area, avgcollab(DOUBLE PRECISION)]

  - **collaborator** = Two authors are collaborators, if they were both authors in a journal and conference papers (but there can be other authors in the paper). In the "article" example on page 3, Bruce W. Watson and Loek G. Cleophas are collaborators. Note that two collaborators can write multiple journal or conference papers together.
  - Again, the decades will be 1950-1959, 1960-1969, ... , 2010-.
  - And for each decade, there will be four bars each representing one of the four areas (do not consider "UNKNOWN").
  - The height of the bars will represent the average number of collaborators.

  - **Definition of an author (say X)'s collaborators in a decade (say 1960-1969) in an area (say Database)**: if the author X published some INPROCEEDINGS papers in the decade (1960-1969) and the area (Database), all other authors who were collaborators of X (in either conference or journal paper) with this author in the same decade (regardless of areas) are considered this author's collaborators in the decade (1960-1969) and in the area (Database).
  - If author Y published multiple papers together with author X in a decade then Y only counts as one collaborator for X in that decade.
  - Intuitively, fix the decade and area first, then for all authors who published any conference (INPROCEEDINGS) paper in the fixed decade and area, count the number of their collaborators (according to the previous definition), and then compute the average number for the decade and the area. So theoretically you may write 4 (for areas) * 7 (for decades) = 28 queries to get all average values, **although this won't be a valid solution to the question!** (think of group by-s and sub-queries).

- Q4c. Using a single SQL query, compute the **"trend"** in the average number of authors per paper over each decade starting from 1950 in each of the four areas, in terms of **the slopes in "linear regression"**. Show the answers as a table.
  Which two areas had the highest and lowest slopes?

  SQL format: The area and its slope [area, slope(DOUBLE PRECISION)]

  - Recall that the linear regression fits the best fitting straight line $y = mx + b$ with minimum least square error, and gives the following formula for slope for n points $(x_1, y_1), \cdots, (x_n, y_n)$:

$$m = \frac{n(\sum_i x_i y_i) - (\sum_i x_i)(\sum_i y_i)}{n(\sum_i {x_i}^2) - (\sum_i x_i)^2}$$

  - Try yourself (**no need to submit**)! How this formula for m is obtained applying least square error on n points.

6 RA Queries

Q5. Recall the schema from your Lab-2 on RA on 09/12 about bars, beers, and bar-goers a.k.a. drinkers.

- drinker(name, address)
- bar(name, address)
- beer(name, brewer)
- frequents(drinker, bar, times_a_week)
- likes(drinker, beer)
- serves(bar, beer, price)

Now answer these four RA questions in RA.
You would have to submit your answer in the RADB language, but you should find the solution first in RA.
You solved (a)-(e) in the RA Lab. Now you will solve (f)-(i).

- (f) Find the **cheapest and second cheapest** drinks in town and **where they are served**; i.e., return the 'serves' rows with the lowest and second lowest prices. If there are multiple serves rows with these two lowest prices, return all of them. (For the example, suppose the two lowest prices are \$0.01 and \$0.02, and there are 3 serves rows with \$0.01 and 1 serves row with \$0.02; then, the query should return a total of 4 rows.)
- (g) For each drinker, find names of bars frequented by the drinker that serve none of the beers liked by the drinker. Format your output as list of (drinker, bar) pairs.
- (h) Find names of all drinkers who frequent only those bars that serve some beers they like.
- (i) Find names of all drinkers who frequent every bar that serves some beers they like.

7 Submission

TO BE ANNOUNCED ON PIAZZA.

Points division:

Total : 100 points

- Code to sparse XML data and load into Postgres: **(10 points)**
- Q1: **(3 points)** – somewhat free, we will post these numbers on piazza :)
- Q2: Q2a **(3 points)**, Q2b **(4 points)**
- Q3: (**3 $\times$ 6 =18 points**)
- Q4: (**3 $\times$ 10 = 30 points**)
- Q5: (**4 $\times$ 8 = 32 points**)