Project 1: Querying Databases Efficiently

Objectives:

- To get familiar with the main components of the database design: schema design, data acquisition, data transformation, querying, and indexing.
- To gain experience with databases containing a reasonable amount of data (and to have a taste of writing efficient SQL queries for a real-life database)

Tools:

MySQL, Excel, and a programming language of your choice.

Due date:

Friday, Oct 12, 2018.

In this project you need to finish the following tasks:

1. Schema Design and Data Acquisition (10 marks)

1.1 schema design

Design and create a database schema about publications **based on the data and queries** (see subsequent part of the project description). We will refer to this schema as PubSchema, and to the data as PubData.

1. **E/R Diagram.** Design the E/R diagram. Some potential entity sets and relationships are given below.
   - `Author` – it may have the following attributes: id (a key; must be unique), and name.
   - `Publication` – it may have the following attributes: pubid (the key -- an integer), pubkey (an alternative key, text; must be unique), title, and year. It may have the following subclasses:
     - `Article`
     - `Book`
     - `Incollection`
     - `Inproceedings`
     - `proceedings`
   - There is a many-many relationship `Authored` from `Author` to `Publication`.

Draw the E/R diagram for this schema. Identify all keys in all entity sets, and indicate the correct type of all relationships (many-many or many-one); make sure you use the ISA box where needed.

Note that the data contains more classes/attributes. You can decide what you want to include in your schedule. But at least your scheme should be able to handle these queries in Section 2 of this project. For projects, we hope to give students the flexibility to explore what they want.

2. **Choose a Database.**

    - **You can install the database on your computer.** Depending on your luck and OS, the installation may go perfectly smoothly, or you may have to uninstall and reinstall a few times and get your fingers very dirty; (check the README file and Manual)

      Note that MySQL database system allows you more options for setting index.

3. **Create the Publication Schema in SQL** We will refer to this schema as **PubSchema** in the following sections.

You need to submit in your report:

1.1) an E/R Diagram; and

1.2) all commands for creating tables.

1.2. Data Acquisition.

This step consists of downloading data, or extracting it with a software tool, or all of the above. Then it involves writing and running some tools that reformats the data into some CSV format that we can upload to the database. You need to download the DBLP data yourself.

Note that this might be the first time for you to process a large file. It will be challenging and fun. You may need several rounds of trials before you succeed! Please treat this as a great learning experience.

1. **Import DBLP into the database system.**
    - Download dblp.dtd and dblp.xml.gz from http://dblp.uni-trier.de/xml/. We don't need dblp.dtd, but the SAX parser (to be explained) may needs it, so you better download it.

    - Before you proceed, make sure you understand dblp.xml. Look inside by typing:

      ```
      more dblp.xml
      ```

      The file looks like this. There is a giant root element:

      ```
      <dblp> . . . . </dblp>
      ```

      Inside there are publication elements:

      ```
      <article> . . . </article>
      <proceedings> . . . </proceedings>

      <inproceedings> . . . </inproceedings>
      ```
      etc

Inside each publication element there are fields:

```
<author> . . . </author>
<title> . . . </title>
<year> . . . </year>
```
etc

- Here is one way to play with this data, in its raw format. Go to DBLP (http://www.informatik.uni-trier.de/~ley/db/) and check out any paper, then click on the XML icon for that paper.

For example, a proceeding is like

```
<dblp>
<proceedings key="conf/kdd/2017" mdate="2017-08-15">
<title>
Proceedings of the 23rd ACM SIGKDD International Conference on
Knowledge Discovery and Data Mining, Halifax, NS, Canada, August
13 - 17, 2017
</title>
<booktitle>KDD</booktitle>
<publisher>ACM</publisher>
<year>2017</year>
<isbn>978-1-4503-4887-4</isbn>
<ee>http://doi.acm.org/10.1145/3097983</ee>
<url>db/conf/kdd/kdd2017.html</url>
</proceedings>
</dblp>
```

The key of this entry is `conf/kdd/2017`.

A publication or paper in the proceeding is like:

```
<dblp>
<inproceedings key="conf/kdd/FayyadCRPCL17" mdate="2017-08-25">
<author>Usama M. Fayyad</author>
<author>Arno Candel</author>
<author>Eduardo Ariño de la Rubia</author>
<author>Szilárd Pafka</author>
<author>Anthony Chong</author>
<author>Jeong-Yoon Lee</author>
<title>
Benchmarks and Process Management in Data Science: Will We Ever
Get Over the Mess?
</title>
<pages>31-32</pages>
<year>2017</year>
<booktitle>KDD</booktitle>
<ee>https://doi.org/10.1145/3097983.3120998</ee>
<crossref>conf/kdd/2017</crossref>
<url>db/conf/kdd/kdd2017.html#FayyadCRPCL17</url>
</inproceedings>
</dblp>
```

`Please note the relationship between proceeding and inproceedings.`

- Choose a programming language (e.g., Java, C#, Python) to read, parse, and extract data from the XML file dblp.xml. In general, there are two ways to parse an XML file, using DOM or SAX parser. The advantage of a SAX parser (over a DOM parser) is that it can process the XML data in a streaming fashion, without storing it in main memory. So SAX parser is recommended because the DOM parser will read the whole XML file into memory, and the memory in your computer may not be large enough. If you use Java, you could refer to http://docs.oracle.com/javase/tutorial/jaxp/sax/parsing.html

  If you use python, you can google to find documents about this. For a quick illustration, see Example 1-1 in page 13 (Search the book "Python and XML" by Christopher A. Jones, Fred L. Drak through Google Books)

- You can also google to find some example when you write your SAX parser. Alternatively, **you can also find tools for reading and processing XML documents from the Web**.

  If you write your SAX parse, note that A SAX application needs to be written to process nested elements in a streaming fashion. Notice how `startElement` handles differently a publication element from a field element; also notice that most of the useful work (writing to the files) is done by `endElement`.

- The extracted files should be in CSV format, which can be imported into databases. If you use SQL server, the following page is a start on how to prepare data for bulk import. (but SCSE database lab may not allow the permission to use BULK INSERT)

  http://technet.microsoft.com/en-us/library/ms188609.aspx

Note: why do we populate the database with a large amount of data? Your database should contain a large amount of data so that you will experience of implementing SQL queries on a reasonable large database.

Challenge: Note that a challenge here is that the extracted files from the XML files may not contain all the required data for the database tables in PubSchema (what you designed in question 1). For example, you may generate two files pubFile.txt and fieldFile.txt by the following steps: For each publication element, like `<article>...</article>`, write one line into a file, say pubFile.txt, and for each field element, like `<year>...</year>`, it writes one line into a file, say fieldFile.txt. One possible solution is as follows.

- Import the extract files into intermediate database tables. Then you transform the data in these intermediate tables into the tables in your database scheme.
  Your transformation will consist of several SQL queries, one per PubSchema table. For example, to populate your Article table, you will likely run a query like:

```
insert into Article (select ... from Pub, Field ... where ...)
```

Since PubSchema is a well designed schema (you designed it yourself!), you will need to go through some trial and error to get the transformation right: use SQL interactively to get a sense of RawData, and find how to map it to PubData. We give you a few hints:

- You may create temporary tables (and indices) to speedup the data transformation. Remember to drop all your temp tables when you are done.
- Databases are notoriously inefficient at bulk inserting into a table that contains a foreign key, because they need to check the foreign key constraint after each insert. Hint: do not declare foreign keys in PubSchema; instead, populate the tables first, then run the ALTER TABLE command (see \h ALTER TABLE in postgres). Way faster...
- What if a publication in RawData has two titles? Or two publishers? Or two years? (You *will* encounter duplicate fields, but not necessarily these ones.) Your PubSchema is textbook-perfect, and does not allow multiple attributes or other nonsense; if you try inserting, should get an error at some point. There are only few repeated fields, but they prevent you from uploading PubSchema, so you must address them. It doesn't matter how you resolve these conflicts, but your data should load into PubSchema correctly.
- Once you are done loading PubData, make sure you add all foreign keys and unique constraints that you have omitted for performance reasons. Hint: use ALTER TABLE.

You need to submit: 1.3) all the codes for this step (as an appendix to your report).

2. Queries and Optimizing Queries

(40 marks)

Finally, now comes the fun part. Write SQL queries to answer the following questions: Note that some hints are given, but it is not necessary to follow the hints.

1. For each type of publication, count the total number of publications of that type between 2000-2017. Your query should return a set of (publication-type, count) pairs. For example (article, 20000), (inproceedings, 30000), ... (not the real answer).

2. Find all the conferences that have ever published more than 200 papers in one year and are held in July. Note that one conference may be held every year (e.g., KDD runs many years, and each year the conference has a number of papers).

3. a) Find the publications of author = "X" (replace X with a real name in your database) at year 2015 (List all the available information of each publication).

   b) Find the publications of author = "X" (replace X with a real name in your database) at year "Y" at conference "Z" (replace Y and Z with real value so that the query will return some tuples as results.)

c) Find authors who published at least 2 papers at conference "Z" at year "Y"

4. Find (a) all authors who published at least 10 PVLDB papers and published at least 10 SIGMOD papers, and (b) all authors who published at least 15 PVLDB papers but never published a KDD paper. (Note that you need to do some digging to find out how DBLP spells the name of conferences and journals).

5. For each 10 consecutive years starting from 1970, i.e., [1970, 1979], [1980, 1989],…, [2010, 2019], compute the total number of conference publications in DBLP in that 10 years. Hint: for this query you may want to compute a temporary table with all distinct years.

6. Find the most collaborative authors who published in a conference or journal whose name contains "data" (e.g., `ACM SIGKDD International Conference on Knowledge Discovery and Data Mining`). That is, for each author determine its number of collaborators, and then find the author with the most number of collaborators. Hint: for this question you may want to compute a temporary table of coauthors.

7. Data analytics and data science are very popular topics. Find the top 10 authors with the largest number of publications that are published in conferences and journals whose titles contain word "Data" in the last 5 years.

8. List the name of the conferences such that it has ever been held in June, and the corresponding proceedings (in the year where the conf was held in June) contain more than 100 publications.

9. (a)Find authors who have published at least 1 paper every year in the last 30 years, and whose family name start with 'H'. (b) Find the names and number of publications for authors who have the earliest publication record in DBLP.

10. Design a join query that is not in the above list

For each of the above query, you may want to test if your query gives correct results. Please perform the following tasks:

a. Record the running time.
b. Cut the size of the database that you used by half, re-run the queries, and record the new running time
c. Further cut the size of the database in b) by half, re-run the queries, and record the new running time.

You can then use Excel to draw a figure and analyze **the effect of database size on the query time**, for each query.

You need to submit in your report:

2.1) the SQL queries,

2.2) screen captures of the results of these SQL queries, and

2.3) a report including the figures and your analysis.

3. Building an Index and Studying the Effect of the Index (40 marks)

Some of your queries above may be slow. Speed them up by creating appropriate indexes, using the CREATE INDEX statement. Think if you can write a more efficient query when you have the indexes.

For each index that you created, experiment and analyze whether it help some of your queries. Hint, for each index that you built, you should run the queries on the data you used in 2a, and compare with the running time when the index is not built.

You need to submit in your report:

3.1) your CREATE INDEX statements; and

3.2) a report of the analysis on whether some of your indexes help some of your queries.

4. Advanced Part: Study the Effect of Cache (10 marks)

In practice, the cache is very important factor for the performance of database system. Different DBMS provides different ways to set up the size for cache. For example, MySQL has *innodb_buffer_pool_size* system variable specifies the size of the buffer pool.

You can choose some queries (with and without indexes) to see how the cache size affects the performance of databases.

You need to submit in your report: the analysis on the effect of cache size on the performance of some of your queries.

Summary on what to submit:

Hardcopy:

Except for 1.3, all the others should be submitted in hard copy (do **NOT** print out your code).

Softcopy:

Everything should be submitted in softcopy through NTULearn (*which has some function to check if two reports are very similar*).

More information on project report submission (both hardcopy and softcopy) will be announced later!

More about Evaluation

Note that Project 1 will take 18% (out of the total of 25%) of the final coursework assessment mark. Late submission can be accepted if you have valid reasons.