Lab 1 - Data Pipeline

DBSys 2021

Assigned:	Oct 13th, 2021	Due:	Nov 1st, 2021

This problem set is based on a problem set from CSE 544 - Principles of Database Systems taught @ University of Washington.

1 Preliminaries

For this problem set, you will work **individually**.

You will use a publicly available dataset on research publications. The DBLP data set can be found at `http://dblp.uni-trier.de`. The full dataset can be downloaded for offline analysis at `http://dblp.uni-trier.de/xml/`. A quick documentation of this dataset is at `http://dblp.uni-trier.de/xml/docu/dblpxml.pdf`. You will download the xml, prepare the dataset, load it into a postgres database and later visualize it using any tool of your choice, such as Excel or python - all these steps are detailed in this document.

1.1 Submission

You will package your solution into a folder with the title: 'DBSys-Lab1-*yourLastName*', please replace yourLastName with your real last name. **Zip this folder** and submit it via Moodle.

We will not grade any submission that does not strictly follow the submission rules.

1.2 Install Postgres

You can install postgres from `http://www.postgresql.org/download/`, pick your OS and install the pre-built binary. On Mac OS X, if you have never installed a postgres instance before, consider installing it from here: `http://postgresapp.com/`, see also `http://postgresapp.com/documentation/cli-tools.html` to be able to run the command line tools from Terminal. We do not recommend compiling from source.

Using Linux is recommended, but if you go for Windows, make sure to note the password used for the username *postgres* during the installation. Also, you may want to specify the username using the following command to create a database *createdb -U postgres dblp*, the password provided during the installation will be requested. The username, in this case *postgres*, needs to be provided otherwise it will take as default the username of the windows session.

You will also benefit from installing pgAdmin4 for later assignments: `https://www.pgadmin.org/download/`

1.3 Check your installation works

Create a Database

In your shell (i.e. not inside the psql shell):

```
createdb dblp
```

If for any reason, you want to start fresh, just drop the whole database:

```
dropdb dblp
```

To open the postgres shell at the created dblp database:

```
psql dblp
```

You might need to provide other information like the host (-h localhost), a username or a password if you created ones.
In psql shell, you can directly type SQL commands:

```
create table foo(bar numeric);
select * from foo;
```

Other psql commands can help you such as:

```
\q -- quit (exit psql)
\h -- help
\d -- list relations
\? -- help for internal commands
```

2 Database Design

2.1 Entity-Relationship Diagram

We briefly touch on ER design in class. *Thus, we recommend to read in the textbook the ER chapter if you have problems answering your question.*
Design an ER diagram, consisting of the following entity sets and relationships **[2 points]**.

1. Author - has the following attributes: id (a key; must be unique), name, and homepage (a URL).
2. Publication - has the following attributes: pubid (the key – an integer), pubkey (an alternative key, text; must be unique), title, and year. It has the following subclasses:
   (a) Article - has the following extra attributes: journal, month, volume, number
   (b) Book - has the following extra attributes: publisher, isbn
   (c) Incollection - has the following extra attributes: booktitle, publisher, isbn
   (d) Inproceedings - has the following extra attributes: booktitle, editor
3. There is a many-many relationship Authored from Author to Publication.

Draw the ER 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. Aim to produce a correct schema and feel free to check it with us during the lab.

Turn in: In your submission folder include the solution in a file named **ER.pdf**

2.2 Create the database relations with SQL

Construct the PubSchema with SQL statements such as **[3 points]**:

```
create table author ( ... );
...
```

These statements implement the ER diagram. Note:

- You can choose to use only `int` or `numeric` or `text` for data types.
- Create keys, foreign keys, constraints, etc. Include these here (and in the deliverable) but drop them prior to loading your data and then re-insert them.
- Do not use the inherit/pivot functionality available with postgres (implement subclasses as separate tables instead)
- Store your sql command in a SQL file: `createPubSchema.sql`. Execute all the commands at one go:

  ```
  psql -f createPubSchema.sql dblp
  ```

- You can drop any table as follows:

  ```
  drop table author; -- or drop table author cascade;
                     -- to handle foreign key constraint violations;
  ```

Turn in: In your submission folder include **createPubSchema.sql**

2.3 Data Acquisition & Loading

Download dblp.xml from `http://dblp.uni-trier.de/xml/`. We provide a simple python data transformation wrapper and a script to import DBLP into postgres. File dblp.dtd is provided in the starter code as it is used by the SAX parser in the python wrapper.

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

Inside each publication element there are fields:

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

Run wrapper.py, which is provided in the starter code. You may have to first edit wrapper.py appropriately to point to the correct location of dblp.xml file, and of the output files, pubFile.txt and fieldFile.txt.

Note the wrapper works with python2.7 not python 3!

```
python wrapper.py
```

This step takes several minutes (around 35 mins on a laptop) and produces two large files: pubFile.txt and fieldFile.txt.

The wrapper uses a python XML SAX parser, which is a simple, event driven parser[1].

Also, look inside pubFile.txt and fieldFile.txt by typing:

```
more pubFile.txt
more fieldFile.txt
```

These are tab-separated files, ready to be imported in postgres.

Run createRawSchema.sql, which is provided in the starter code. First, modify it with the absolute file path to pubFile.txt and fieldFile.txt (the path must be absolute, because the copy command is executed by the server, not by the client.). Then run:

```
psql -f createRawSchema.sql dblp
```

This imports the data to postgres. It creates two tables, Pub and Field, which we call the RawSchema, and we call their data the RawData.

Before you proceed, make sure you understand what you did. Inspect createRawSchema.sql: you should understand every bit of this file. Then, start an interactive postgres by typing this command:

```
psql dblp
```

and type in some simple queries, like:

```
select * from Pub limit 50;
select * from Field limit 50;
```

Here is one way to play with this data, in its raw format. Go to DBLP and check out one paper, then click on the Bibtex icon for that paper. Say, you check out Christos H. Papadimitriou's DBLP entry, and click on paper:

[1] 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. For a quick illustration, see Example 1-1 in http://oreilly.com/catalog/pythonxml/chapter/ch01.html. A SAX application like wrapper.py needs to be written to process nested elements in a streaming fashion. For each publication element, like <article>...</article>, wrapper.py writes one line into pubFile.txt, and for each field element, like <year>...</year>, it writes one line into fieldFile.txt. 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.

```
@inproceedings{DBLP:conf/stacs/GemiciKMPP19,
  author    = {Kurtulus Gemici and
               Elias Koutsoupias and
               Barnab{\'{e}} Monnot and
               Christos H. Papadimitriou and
               Georgios Piliouras},
  title     = {Wealth Inequality and the Price of Anarchy},
  booktitle = {36th International Symposium on Theoretical Aspects of Computer Science,
               {STACS} 2019, March 13-16, 2019, Berlin, Germany},
  pages     = {31:1--31:16},
  year      = {2019},
  crossref  = {DBLP:conf/stacs/2019},
  url       = {https://doi.org/10.4230/LIPIcs.STACS.2019.31},
  doi       = {10.4230/LIPIcs.STACS.2019.31},
  timestamp = {Thu, 02 May 2019 17:40:17 +0200},
  biburl    = {https://dblp.org/rec/bib/conf/stacs/GemiciKMPP19},
  bibsource = {dblp computer science bibliography, https://dblp.org}
}
```

The key of this entry is conf/stacs/GemiciKMPP19. Use it in the SQL query below:

```sql
select *
from Pub p, Field f
where p.k='conf/stacs/GemiciKMPP19'
and f.k='conf/stacs/GemiciKMPP19';
```

2.4 Queries on raw data

Write SQL Queries to answer the following questions using Pub p and Field f:

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

2. We say that a field 'occurs' in a publication type, if there exists at least one publication of that type having that field. For example, 'publisher occurs in incollection', but 'publisher does not occur in inproceedings' (because no inproceedings entry has a publisher field). Find the fields that occur in **all** publications types. Your query should return a set of field names: for example it may return title, if title occurs in all publication types (article, inproceedings, etc. notice that title does not have to occur in every publication instance, only in some instance of every type), but it should not return publisher (since the latter does not occur in any publication of type inproceedings) **[2 points]**.

Turn in: In your submission folder include **solution-raw.sql**, which would include SELECT-FROM-WHERE queries. In addition, insert into the file all answers to the queries, in form of SQL comments.

3 Data Transformation

Transform the DBLP data from RawSchema to PubSchema **[3 points]**.

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. Here are a few tips:

1. You may create temporary tables (and indices) to speedup the data transformation. Remember to drop all your temp tables when you are done. Keep track of these as well as the drop statements in solution.sql

2. 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...

3. PubSchema requires you to generate an integer key for every author, and for every publication. Use a sequence. For example, try this and see what happens:

   ```
   create table R(a text);
   insert into R values ('a');
   insert into R values ('b');
   insert into R values ('c');
   create table S(id int, a text);

   create sequence q;
   insert into S (select nextval('q') as id, a from R);
   drop sequence q;

   select * from S;
   ```

4. 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.

5. 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.

Turn in: In your submission folder include **transform.sql**, which should include INSERT, CREATE TABLE, DROP TABLE, ALTER TABLE and CREATE INDEX statements.

4 Data Analysis

4.1 Queries

Write SQL queries to answer the following questions:

1. Find the top 20 authors with the largest number of publications **[2 points]**.
2. Find the top 20 authors with the largest number of publications in STOC. Repeat this for one more conferences of your choice (e.g.: SIGMOD or VLDB, careful with spelling the name of the conference) **[2 points]**.
3. Two of the major database conferences are 'PODS' (theory) and 'SIGMOD Conference' (systems). Find (a) all authors who published at least 10 SIGMOD papers but never published a PODS paper, and (b) all authors who published at least 5 PODS papers but never published a SIGMOD paper **[2 points]**.
4. **Extra credit:** A decade is a sequence of ten consecutive years, e.g. 1982, 1983, ..., 1991. For each decade, find the most prolific author in that decade across all venue. Hint: you may want to first compute a temporary table, storing for each decade and each author the number of publications of that author in that decade. Consider computing a temporary table of coauthors. This query will take a few minutes to run **[1 point]**.

Turn in: In your submission folder include **solution-analysis.sql**, which should include your queries and answers as comments.

4.2 Data Visualization

Write a script in any language of your choice (python, java, ...) to connect to your database and submit queries to compute the histogram of the number of publications **[3 points]**.

Start by writing a SQL query that returns a set (k, f(k)), where k=1,2,3,... and f(k) = number of authors that have exactly k publications. Retrieve the results back into your program. From there, either output a CSV file and import it into Excel or use whatever other method you like to produce the graph.

For the histogram, indicate whether the scatter plot is exponential, or a power law, or something else. You may want to try either a log scale, or a log-log scale in your plot to best determine the type of law. Turn in the plots using the best scale that illustrates your finding (e.g. if you used a log-log scale to determine the type of distribution, then your plot should be in log-log).

Turn in: In your submission folder include **vis.sql**, which includes one (or more) SQL queries, **vis.pdf**, which shows your plot, and any **script**.