[15 hrs] BUILD: Practicum II / Mine a Database

Start Assignment

Due Thursday by 11:59pm **Points** 100 **Submitting** a file upload **File Types** zip
Available until Apr 23 at 11:59pm

> Set reminders of the due date and late submission window to ensure timely submission. Block time in your calendar to work daily on the practicum no later than the module starts.

Motivation

In this practicum you will extract data from an XML document and then store the data relationally in a SQLite database. That database represents a "transactional" database. Then you will extract data from the transactional database and create an "analytical" database using a star schema in MySQL. Finally, you will query facts from the MySQL analytical database. This will require that you connect to two different databases simultaneously -- a common occurrence in practice.

Format

Method: Individual or in pairs (**maximum two**)
Materials: RStudio or RStudio.cloud; SQLite; MySQL

In Part 1 you create a normalized relational OLTP database and populate it with data from an XML document. In Part 2 you will add to the normalized schema fact tables and turn the normalized schema into a de-normalized schema suitable for OLAP. In Part 3 you'll use the OLAP star/snowflake schema to do some (simple) data mining. The parenthesis contain the maximum number of points that can be earned for that question and an estimate of the time in hours.

Part 1 (55 pts) Load XML Data into Database

1. (0 *pts / 0.1 hrs*) Create an R Project named CS5200.PracticumII.*LastNameF* (where *LastName* is your last name and *F* is your first initial, *e.g.*, *CS5200.PracticumII.WuX* or both last names plus initial, *e.g.*, *CS5200.PracticumII.WuX-ChangY,* if you work in a pair) and then within that project create an R Script (*.R* file and not a *.Rmd* Notebook) called *LoadXML2DB.LastNameF.R* for Part 1 (same naming as above if you work in a group); Parts 2 and 3 will be done in different programs.
2. (0 *pts / 0.5 hrs*) Download the following XML file: **pubmed22n0001-tf.xml (https://s3.us-east-2.amazonaws.com/artificium.us/lessons/06.r/l-6-183-extractxml-data-in-r/pubmed-xml-tfm/pubmed22n0001-tf.xml)** . Save the XML files locally in a subfolder named *pubmed-tfm-xml* inside

your project folder, and then inspect the file to familiarize yourself with its content and structure. You might wish to create a subset of the first file for use during development as it will load faster. To download and save an XML file, open the context menu by (generally) right-clicking on the link and selecting "Save As" or a similar choice depending on your operating system.

3. (5 pts / 45 min) Build an external DTD for the XML file above and reference that DTD from the XML downloaded in (2). Fewer points will be awarded if the DTD is internal.
4. (5 *pts / 1 hr*) Create a normalized relational schema that contains the following entities/tables: *Articles*, *Journals*, *Authors*. Use the XML document to determine the appropriate attributes (fields/columns) for the entities (tables). While there may be other types of publications in the XML, you only need to deal with articles in journals. Create appropriate primary and foreign keys. Where necessary, add synthetic surrogate keys. Store all information necessary for Parts 2 and 3. The choice of table structure is up to you. You may wish to visualize the schema using an ERD, although you do not have to submit the ERD.
5. (10 *pts / 1 hr*) Realize the relational schema in SQLite (use CREATE TABLE statements using R functions; you cannot use {sql} chunks in a R Script). You may **not use MySQL** for this step.
6. (5 *pts / 2 hrs*) Load (with validation of the DTD) the XML file into R from a URL (or locally; but fewer points will be awarded). You need to upload the XML and DTD to a hosting service of your choice.
7. (30 *pts / 5 hrs*) Extract and transform the data from the XML file in the folder and then save the data into the appropriate tables in the database. Given the structure of the XML, it is not feasible to only use `xmlToDataFrame` but instead will need parse the XML using a combination of node-by-node tree traversal and *XPath*. It is not feasible to use *XPath* to extract all journals, then all authors, etc. as some are missing and won't match up. You will need to iterate through the top-level nodes. While outside the scope of the course, this task could also be done through XSLT. Do not store duplicate authors or journals. For dates, you need to devise a conversion scheme, document your decision, and convert all dates to your chosen encoding scheme. You may wish to create a smaller XML file for testing rather than processing the larger XML file each time you run the code during development.
8. *(5 pts / 0 hrs)* Practice good coding: write R functions as needed; structure your code to be readable and maintainable; use proper programming practices, thoroughly comment your code; add your name, course information, date to a header in your script.

Part 2 (25 pts) Create Star/Snowflake Schema

1. (0 *pts / 0.1 hrs*) Create a new R Script for Part 2 named *LoadDataWarehouse.LastNameF.R* (use same naming as before if you are in a group).
2. (5 *pts / 1 hr)* Create a MySQL database using either a local or a cloud MySQL instance. Connect to the database. If you use SQLite for this step, no credit will be awarded for this question.
3. (20 *pts / 5 hrs*) Create and populate a star schema for journal facts. You may wish to visualize the star schema using an ERD, although you do not have to submit the ERD. Each row in this fact table will represent one journal fact. It must include the journal PK (as you've designed in in Part 1 above), journal title, number of articles published in the journal per quarter and per year, and number of unique authors who published per quarter and per month. Load the data from the SQLite Database

created in Part 1 and populate the fact table via R. Do not load the entire tables from SQLite into R as that does not scale. Use SQL to get the data you need. Note that there is not a single way to create the fact table -- you may use dimension tables or you may collapse the dimensions into the fact table. Remember that the goal of fact tables is to make interactive analytical queries fast through pre-computation and storage -- more storage but better performance. This requires thinking and creativity -- there is not a single best solution. Make sure your code scales -- imagine that your transactional database (the SQLite database) contains hundreds of millions of rows per table -- would your code still work and still performance well? ETL code is not always the fastest but it must scale. This **chalk talk** **(https://youtu.be/H6no78vdNUM)** on how to build fact tables and why might help...

Your star schema must support analytical queries such as these:

What the are number of articles published in every journal in 2012 and 2013?
What is the number of articles published in every journal in each quarter of 2012 through 2015?
How many articles were published each quarter (across all years)?
How many unique authors published articles in each year for which there is data?

This is the fact table you will need for Part 3.2. Of course, the years in the examples above are arbitrary and other analytical queries are possible, but these are examples to help you define your fact tables.

Part 3 (20 pts) Explore and Mine Data

1. (0 pts / 0.5 hrs) Create an R Notebook named *AnalyzeData.LastNameF.Rmd* within your R Project (use same naming as before if you are in a group).
2. (15 pts / 4 hrs) In the notebook, use markdown to write a "report" which shows the results of the following analytical queries against your MySQL data warehouse from Part 2 (which might go to some manager as part of a weekly report):

   Analytical Query I*: Top five journals with the most articles published in them for the time period. Think about how you would best present this information so it is easily usable by an analyst or manager.*

   Analytical Query II: *Number of articles per journal per year broken down by quarter. Again, think of a good way to show this. Is a table better than a visualization or is it better to use a visualization.*

   Of course, be sure that your report uses the data from the fact tables.

   If you use SQLite for this step, maximum half the credit will be awarded for this question. Each of the analytics use cases is equally weighted. Thoughtfulness and creativity do count and we want you to think about your presentation and report. Apply yourself a bit.

Hints

- see the hints from Practicum I -- many apply here as well
- if you cannot connect to a local MySQL, be sure that the MySQL Server is actually running and that you do not have firewall software running that blocks port 3306
- if you cannot batch save data with `dbWriteTable()`, then you need to enable the setting with `dbSendQuery(db, "SET GLOBAL local_infile = true")`
- if you cannot install MySQL locally, use AWS RDS or db4free.net -- see instructions on how to use these services in Practicum I; note the restriction on db4free.net that you cannot write data with `dbWriteTable()`; see Practicum I on tutorials for bulk loading
- hashmaps can speed up look-ups (they are a kind of index); R doesn't directly support them, but they can be "faked": **https://www.r-bloggers.com/2015/03/hash-table-performance-in-r-part-i (https://www.r-bloggers.com/2015/03/hash-table-performance-in-r-part-i)** or investigate the functions from the package **r2r** which implements hashmaps
- saving large data frames and other objects can save time... look up how to save objects in binary: **https://rstudio-education.github.io/hopr/dataio.html (https://rstudio-education.github.io/hopr/dataio.html)**
- build a smaller XML file for development; it'll speed things up during dev
- you may use AI assistants as needed but let us know which ones you used and for what; add acknowledgments in your code

Submission

Submit all files: *.R*, *.Rmd* , *.xml*, and *.dtd* in a *zip* file. So, all of your programs, the modified XML that references the external DTD and the file containing the DTD. One submission per pair (max two per "group") is sufficient; list both members of the group in all submission files and in a submission comment. Your code must run. Do not use absolute paths for your XML files or the database; make all paths relative to your project folder. Do not do any work outside of R.

Schedule your (required) demo. Failing to demo your code will result in a 0 for the practicum.

Your code has to run, obviously, but it also has to run somewhat efficiently... if everyone else's code runs in 10-30 minutes but yours takes several hours then clearly is due to poor programming and not due to the inherent complexity of the problem... follow common coding strategies for writing efficient code such as factoring out invariants from loops, not calling functions repeatedly, pre-allocating memory, not copying objects needlessly, not calling expensive functions when simpler ones will do (e.g., *call substring() instead of doing regular expressions), use which() when searching and don't use sqldf, and so forth. These practices are not specific to R, although there are R specific performance issues, but those are less likely to be a concern here.*