CS 457/657 Database Management Systems

Programming Assignment 3: Table Joins

Overview

In this assignment you will write a program that allows a database user to make queries over multiple tables. That is, you will implement table joins using techniques we discussed in the class (e.g., nested loop). This assignment assumes the metadata and data manipulations on singular tables are already available, which should have been implemented in the first two programming assignments.

System Design

- You will decide how to implement the join operations
    - In lectures: nested loop
    - As always, you are free (in fact, encouraged) to come up with your own design
- In the design document, you should clearly explain how your program joins tuples from two source tables. If in your first assignment you decided to use the mapping of directory-database and file-table, and if in your second assignment you followed the following physical layout for an example table ***Product (pid, name, price)***

    pid int | name varchar(20) | price float
    1 | laptop | 1999.99
    2 | mobile phone | 899.99
    3 | monitor | 1399.99

    then the pseudocode for a join could be (omitting the column headers):

```
for each line of table_file1
  for each line of table_file2
    check the condition parsed from the given query
```

Implementation

- The program should not use external database library/application.
- The program should persist metadata and data somewhere (e.g., local file system, cloud storage).
- Any programming language is acceptable, e.g., Python, Java, C/C++, Go
    - Just pick one(s) that you are most comfortable/proficient with
    - But keep in mind we will test your code in Linux with OS-level utilities (e.g., files)
        - So, probably not: C#, Object-C, JavaScript, Prolog…
- Functionalities:
    - SQL: inner join, left outer join

Interface

- A similar interface than Sqlite3
- Please contact the TA if you are unsure about your programming interface.

Testing

- We will test your program on Ubuntu (version 14 or above)
- If the TA cannot compile your code, you will be asked to demo its compilation and execution
    - Try not to use many exotic libraries and/or heavy frameworks.
- A full test script will be provided
    - # ~/cs457/pa3/[your_program] < PA3_test.sql (if your program supports redirection)
    - # [expected output]
    - You don't need to parse the comment lines (i.e., starting with "- -")
    - We will not to test your programs with any other scripts
        - It's always good to consider more corner cases

Grading (20 points)

- This is an individual assignment
- Design document that clarifies the followings: (5 points)
    - At a very high level, how you implement different joins.
    - Be very specific on how to compile and execute your code
- Source code (15 points)
    - Coding style and clarity, 5 points
        - Appropriate parenthesis locations, indention, etc.
        - Always write comments at the beginning of any files
            - Author, date, history, etc.
        - Always write comments at the beginning of any non-trivial class/function
            - What this class/function does, high-level algorithm if needed
        - Write in-line comments for non-trivial blocks of code
    - Functionality, 10 points
        - Refer to the test script for detailed breakdowns

Submission

- WebCampus
- Compress all your source code and documents into one package in this format:
    - <your_netid>_pa3
- Late penalty: 10% per day