CS 411: Database Systems

Fall 2018

Homework 5 (Due by 23:59 CST on 12/9)

Logistics

1. The homework is due on Dec 9 23:59. **We DO NOT accept late homework submissions.**
2. **(VERY IMPORTANT)** You will be using Gradescope to submit your solutions. For your solutions, use the template in **this link**. The template is read-only, make a copy of it (*File* > *Make a copy...*). Answer each question in the specified placeholder for that question. Do not modify the headers or remove a page or add a new page, we provided extra pages for questions that might need more than one page. For submission, download the pdf file (*File* > *Download as* > *PDF Document (.pdf)*) and submit the file on Gradescope (your pdf file should have 10 pages, if not you didn't follow the previous guidelines).
3. Please submit your homework to Gradescope using your illinois.edu email. There should be an invitation sent out.
4. Please write all answers electronically. We won't grade handwritten/hand-drawn versions. If you are looking for tools to create ER-diagrams etc, consider https://www.draw.io or GraphViz.
5. Feel free to talk to other members of the class in doing the homework. You should, however, write down your solutions yourself. Also, list the names of everyone you worked with at the top of your submission.
6. Please use Piazza if you have questions about the homework but do not post answers. Feel free to use private posts or come to office hours.

Section 1. Query Optimization [30 pts]

Part 1. [14 points]

Given the following Bank database schema:

Customer = (cid, cName, cCity)
Account = (aNumber, balance, interest_rate, cid, bid)
Branch = (bid, bCity, bName)

1) For each equivalence shown below, state whether it is true or false in general. If your answer is "true", justify the equivalency by explaining which RA rule/rules can be used to transform one query expression into the other. If your answer is "false", give a sample instance of the database where the two expressions are not equal. (7 points)

a)

$$\pi_{Cus-cid}(\sigma_{cName="Alex" AND\ Cus-cid=Acc-cid\ AND\ balance>1000\ AND\ Acc-bid=Bra-bid\ AND\ bCity="Urbana"}((Branch \times Account) \times Customer))$$

≡

$$\pi_{Cus-cid}(\sigma_{Acc-bid=Bra-bid}(\sigma_{Cus-cid=Acc-cid}(\sigma_{cName="Alex"}(Customer) \times \sigma_{balance>1000}(Account)) \times \sigma_{bCity="Urbana"}(Branch))$$

b) $\pi_{Cus-cid}(\sigma_{cName="Alex" AND\ Cus-cid=Acc-cid\ AND\ balance>1000\ AND\ Acc-bid=Bra-bid\ AND\ bCity="Urbana"}((Branch \times Account) \times Customer))$

≡

$$\pi_{Cus-cid}(\sigma_{Cus-cid=Acc-cid}(\pi_{Acc-cid}(\sigma_{city="Urbana"}(Branch) \times \sigma_{balance>1000}(Account)) \times \sigma_{cName="Alex"}(Customer)))$$

c)

$$\pi_{Cus-cid}(\sigma_{cName="Alex" AND\ Cus-cid=Acc-cid\ AND\ balance>1000\ AND\ Acc-bid=Bra-bid\ AND\ bCity="Urbana"}((Branch \times Account) \times Customer))$$

≡

$$\pi_{Cus-cid}(\sigma_{Cus-cid=Acc-cid}(\sigma_{cName="Alex"}(Customer) \times \sigma_{balance>1000}(Account)) \bowtie \pi_{bid}(\sigma_{bcity="Urbana"}(Branch)))$$

2) Rewrite the following queries to make the execution most efficient (that is, pushing selections and projections as close to the base relation as possible, limiting the projected columns to only the ones required, and using appropriate joins). (7 points)

a) Find the names of all customers who live in “Champaign” with an account at some branch in “Champaign” or “Urbana”.

$$\pi_{cName}(\sigma_{cCity="Champaign" AND\ Cus-cid=Acc-cid\ AND\ Acc-bid=Bra-bid\ AND\ (bCity="Urbana"\ OR\ bCity="Champaign")}((Branch \times Account) \times Customer))$$

b) Find the names of all customer with an account at some branch located in “Champaign”, whose account interest rate is higher than 2% and the account balance is larger than 2000$.

$$\pi_{cName}(\sigma_{\ Cus-cid=Acc-cid\ AND\ Acc-bid=Bra-bid\ AND\ bCity="Champaign" AND\ balance\ >\ 2000\ AND\ interest-rate>2}((Branch \times Account) \times Customer))$$

Part 2. [6 points]

Consider the following selection operation on the relation R(A, B, C). If the relation R contains 1000 tuples, where attribute A has 10 different values, attribute B has 20 different values, and attribute C has 30 different values, what is the maximum possible size of the selection for each of the following queries?

A. $\sigma_{((A=a1)AND(B=b2))OR(C=c3)}(R)$

B. $\sigma_{(A=a1)AND(B=b2)AND(C=c3)}(R)$

Part 3. Dynamic Programming [15 points]

Consider relations A, B, C and D with the following information.

A(x,y)	B(z,y,w)	C(v,x)	D(w,x,z)
T(A) = 5000 V(A, x) = 100 V(A, y) = 50	T(B) = 6000 V(B, z) = 200 V(B, y) = 150 V(B, w) = 50	T(C) = 2000 V(C, v) = 100 V(C, x) = 250	T(D) = 2500 V(D, w) = 100 V(D, x) = 200 V(D, z) = 50

Note that T(R) is number of tuples in relation R and V(R, a) is number of distinct values of attribute a in relation R. We want to compute A ⋈ B ⋈ C ⋈ D as efficiently as possible. Determine the most efficient way to do the join. Clearly state any assumptions you have made. Show your work by completing the following table (each step in the dynamic programming algorithm should be one row):

Subquery	Size	Cost	Plan
...	...	...	...

Section 2. MongoDB [35 points]

Install MongoDB on your machine. The installation instructions of MongoDB can be found here

Download the database provided here which has been acquired from the YELP database.

The database has two CSV files: *review.csv* and *business.csv*.

- *reviews.csv:* This file contains reviews of YELP users on a set of business where each review is associated with the user id of the users who have submitted the review as well as the business id of the business that the review is about. The original YELP dataset contains 5200000 reviews. We randomly sampled 500000 reviews to generate the reviews.csv file.
- *business.csv*: Each line of the file contains a business id along with some attributes associated with the target business.

Load the .csv files into your database and write and execute the following queries.

NOTE

For each query, please include your query and also **top-10 documents ordered (in descending order)** by the corresponding id attribute in the pdf. You do not need to include the entire results in your submission.

1) List all businesses with more than 200 reviews. (The output documents should contain a single field: business_id)
2) List all users with at least one review for a restaurant in Arizona (AZ). Use the “categories” attribute to find the businesses belonging to the category “Restaurants”. (The output documents should contain a single field: user_id)
3) Display the total number of users who have at least one review with rating score 5.
4) Display the total number of users with at least two reviews for a business located in California (CA).
5) List all users who have at least five reviews on businesses with a delivery option. (The output documents should contain a single field: user_id)
6) Compute and display the average star rating (computed from reviews) of businesses in Illinois grouped by business id. (The output documents should contain two fields: business_id, average_star_rating)

Bonus (12 points)

7) Display the total number of users who have reviewed businesses in more than two distinct states. (5 points)

8) For each business b, display the total number of reviews from users who have at least five reviews outside the state of business b. (The output documents should contain two fields: business_id, total_number_of_reviews) (7 points)

Section 3. Neo4j [35 pts]

Install Neo4j on your machine. The installation instructions of Neo4j can be found here

You will use the same database that used in Section 2. Use Neo4j’s browser (http://localhost:7474/browser/) to load the data and write your queries. The graph database should have three types of nodes: user, business, and reviews.

Once you created the graph database, use the Neo4j browser to write and execute the same queries from Section2. **Query #7 is a mandatory question for this section.**

Query #8 is counted as extra credit for this section too. You don't need to return the query results for this one. However, we strongly recommend you to check your query correctness on a smaller dataset before submission. (7 points).

NOTE

For each query, please provide your query and also **top-10 documents ordered (in descending order)** by the corresponding id attribute in the pdf.

Dear students,
Students that have difficulties to import the CSV files to a Neo4j graph can use the provided database dump. To load the dump please follow these steps:

Before starting, we strongly recommend using the Community Server version that is available for Mac/Linux/Windows as ZIP files here https://neo4j.com/download-center/#releases (some other versions don't have neo4j-admin tool that we need to load the dump)

1 - Download the graph.dump from https://www.dropbox.com/s/lmr2u6ntkvpfnlq/graph.dump?dl=0

2 - Shutdown the database.

3 - In [NEO4J-HOME]/bin directory, execute the following command:

./neo4j-admin load --from=path/to/graph.dump --database=graph.db --force

4 - Start the database.

More details about Neo4j Dump/Load commands:
https://neo4j.com/docs/operations-manual/current/tools/dump-load/

This article is a good start for query tuning in Neo4j: https://neo4j.com/blog/neo4j-2-2-query-tuning/