Due Date: MARCH 1st, 2020 at 11.59pm

We do not require you to typeset your homework, although we encourage you to learn how to typeset your works in LaTeX or Microsoft Word. In any case, make sure that your submission is effortlessly **readable**.

You can work and discuss with your classmates, but please submit your **own** work and list the name of your classmates you are working with. **No PLAGIARISM**! If you need any clarification, please post them on Canvas Discussion.

For each of the following questions, please make sure that your answers are written as **concise** as possible. If you are unsure about your answers, feel free to try to justify them for a some small partial credit. Also, if your answer requires calculation, you do not need to simplify your formula.

Problem 1: SQL Programming - 40pt

*** This problem is based on the Assignment 2 of ICCS240 II.2018-2019. ***
*** You may work in group of up to 3 persons, and hand-in a single report for this problem. ***

What to hand in?

- A report of the database schema, complete with primary keys, foreign keys, and relationships between tables.
- A report of all indexes created and a brief description of what each index is intended to help with.
- All source files that you wrote to import the dataset. This should be well-documented and come with enough instruction for someone to rerun it.
- A report of how indexing affects query performance.
- Prepared to demo your program! (03-10 and 03-12 at the end of the class)

Importing into the Databases

Your goal for this part is to download a large dataset, design a database schema to store it in a DBMS, and import the entirety of it into the DBMS. You'll use MySQL or PostgreSQL for this assignment (or a somewhat equivalent free version of MySQL, MariaDB). Your design should make sure that (1) data import performance is acceptable and (2) query performance (see Part II) is excellent. To begin, you will download the Yelp academic dataset from
`https://cs.muic.mahidol.ac.th/courses/db/yelp_academic_2018.tbz2`.
The compressed archive (tbz2) is huge (a few GBs), though. The tbz2 archive contains several files inside. Each is one kind of data. For example, there is a file dedicated to reviews. As a rule of thumb, you should have at least one table for every file here. The data format can be found at `https://www.yelp.com/dataset/documentation/main`. Once you have the design, write a

program to import the whole dataset into the DB. This program can be in any language of your choice. We recommend Python, Java, PHP or Kotlin.

- **Tips#1:** Index your tables appropriately. Too little indexing means horrible query performance. Too much indexing means terrible data import performance —worse yet, queries may still be slow
- **Tips#2:** After your initial design, you may wish to revisit the decisions you've made once you start importing the data and attempted the next portion of the assignment.
- **Tips#3:** It is likely you'll go through this process more than once. For this reason, we recommend that you script everything, from cleaning up existing remains to table creation to inserting in the actual records.

Queries

This portion of the assignment deals with making appropriate queries to the DBMS to answer some questions. Here are some ground rules:

- For each of the questions below, you'll write a function that takes zero or more parameters from **users' inputs** and produces output to screen as indicated below.
- Each question must be answered using a single, possibly nested SQL query. You should use JDBC `PreparedStatement` or equivalent to setup/execute your query.
- For queries that ask for top k, your program should not ask for or obtain more than k entreis from the DB.
- Your solution to each of the following questions should finish in fractions of a second (or at worst, a few seconds).
- You should create a rudimentary interface, so you can demo any combination of these functions without having to recompile each time.

Here is the list of queries you must implement.

1. **Still In Business**
   Your function will be called `stillThere`. The input to your function is a state code (e.g., AZ). You will display the top 10 businesses in this state, sorted in descending order of review counts, that are still in business (as indicated by the attribute `open`). Display for each business:
   `business_id, name, full_address, stars`
2. **Top Reviews**
   Your function will be called `topReviews`. The input to your function is a business ID. Given the business ID, obtain the reviews for that business. Results should be sorted by the review's useful vote counts in descending order. Obtain and show the top 5 reviews. For each review, show the following information:
   `user_id, name_of_user, stars_of_the_review, review_text`
3. **Average Rating**
   Your function will be called `averageRating`. The input to your function is a user ID. Given

that user ID, display the name of the user and the average star rating across all reviews written by this user:

`user_id, name_of_user, average_star_ratings`

4. **Top Business in City**
   Your function will be called topBusinessInCity. The input to your function is (1) a city (input format of your choice) (2) a non-negative number `eliteCount`, and (3) a positive number `topCount`. You will display the businesses in the provided city that have been reviewed by more than eliteCount "elite" users. Your results are ordered by how many "elite" users have reviewed it, in descending order. Display only the top `topCount`. For each business, the following information will be shown:

   `business_id, business_name, review_count, stars, count_of_elite_reviews`

Why Indexing

Since you have a script to re-create everything from scratch, this experiment will be relatively painless to conduct. The overall goal is to appreciate the effect(s) of indexing. Before you attempt this, make sure you're done with the rest of the assignment and have the ability to re-create a perfectly working environment again. For each of the questions above, find some input parameters that produce sensible answers. Note these parameters as we'll use them again soon. You'll conduct this experiment in two stages:

Stage i:

You'll go through every question with the input parameters you noted above. For each question, measure the time it takes to complete the query. (Hint: Don't do this manually. Add timing code to your program)

Stage ii:

Delete all the indexes and repeat the same experiment. Make a table comparing the timing differences (if any) between a database indexed as designed and a database that is not indexed.

At the end of this, restore your database back to a good working condition.

Problem 2: Relational Database Design Theory - 30pt

1. [5pt] A company has started building his database system including information about contract and project, so they come up with the following schema:

   `Contracts(c_no, supp_no proj_no, dept_no, part_no, qty, val)`

   where `c_no`, `supp_no`, `proj_no`, `dept_no`, `part_no`, `qty` and `val` denote contract number, supplier, project, department, equipment part, quantity and value.

   Two DB designers, Alice and Bob, are working over this relation, and discover their own set of constraints over the relation as follows.

   Alice found:

   - A contract number is a key.
   - A project purchases each part using a single contract. (`proj_no part_no` $\rightarrow$ `c_no`)
   - A department purchases at most one part from a supplier. (`supp_no dept_no` $\rightarrow$ `part_no`)

On the other hand, Bob came up with:

- A contract determines project, supplier and department. (`c_no` $\rightarrow$ `proj_no supp_no dept_no`)
- A project purchases each part using a single contract. (`proj_no part_no` $\rightarrow$ `c_no`)
- A supplier, a project and a department, together, are the key of this relation.

Are the findings of the second designer different from those of the first? Show your justification.

already presented in class :)
Idea: given $\mathcal{F}_1$ and $\mathcal{F}_2$, we say that $\mathcal{F}_1$ is equivalent to $\mathcal{F}_2$ if you could show that, for each $f_1 \in \mathcal{F}_1$, f_1 could be inferred from $\mathcal{F}_2$. Similarly, you must also show that, for each $f_2 \in \mathcal{F}_2$, f_2 must also be inferred from $\mathcal{F}_2$. To show that, you can use closure test or Armstrong's Axioms.

2. [5pt] Given three disjoint ordered sets of attributes α, β and γ, consider relations $R = (\alpha, \beta, \gamma)$, and its decomposition: $R_1 = (\alpha, \beta)$ and $R_2 = (\alpha, \gamma)$. That is, $R_1 = \prod_{R_1}(R)$ and $R_2 = \prod_{R_2}(R)$. Assume $R_1 \cap R_2 \rightarrow R_1$ (that is, $\alpha \rightarrow \alpha\beta$). Prove or disprove that $R = R_1 \bowtie R_2$.

   Let a, b, c denote instances of α, β, γ.
   First, we prove that $R \subseteq R_1 \bowtie R_2$, If $(a, b, c) \in R$, then $(a, b) \in R_1$ and $(a, c) \in R_2$. That is, $(a, b, c) \in R_1 \bowtie R_2$.
   Next, we show that $R_1 \bowtie R_2 \subseteq R$. If $(a, b, c) \in R_1 \bowtie R_2$, then $(a, b) \in R_1$ and $(a, c) \in R_2$. That is, there exists c' and b' such that $(a, b, c') \in R$ and $(a, b', c) \in R$. Since $\alpha \rightarrow \beta$, we have that $b = b'$, and so $c' = c$. Thus $(a, b, c) \in R$.

3. [20pt] Consider a relation $R(A, B, C, D)$ with FDs: $AB \rightarrow C$, $BC \rightarrow D$, $CD \rightarrow A$.

   - Is R in BCNF? Explain why or why not?
     No, R is not in BCNF since the $CD \rightarrow A$ violates the condition. That is, CD is not a superkey of R.
   - If R is not in BCNF, decompose it into collections of relations that are in BCNF.
     Decompose using $CD \rightarrow A$, we have $R_1(A, C, D)$ and $R_2(B, C, D)$.
   - Is the decomposition of R into $R_1(A, B, C)$ and $R_2(A, C, D)$ lossless? Explain why.
     Consider $R = \{(1, 2, 3, 4), (1, 4, 3, 2)\}$. After decomposition, we have $R_1 = \{(1, 2, 3), (1, 4, 3)\}$ and $R_2 = \{(1, 3, 4), (1, 3, 2)\}$.
     Consider that $R_1 \bowtie R_2$ gives $\{(1, 2, 3, 4), (1, 2, 3, 2), (1, 4, 3, 4), (1, 4, 3, 2)\} \neq R$.
     Thus, the decomposition is not lossless.
   - Is R in 3NF? If not, decompose R into a collection of relations that are in 3NF.
     Yes.

Problem 3: Storage and Indexing - 20pt

1. [5pt] Consider a relation $R(A, B, C, D)$ containing 1 million records. where each page of the relation holds a number of records. R is organized as a heap file with unclustered indexes, and the records in R are randomly ordered. Assume that attribute A is a candidate key for

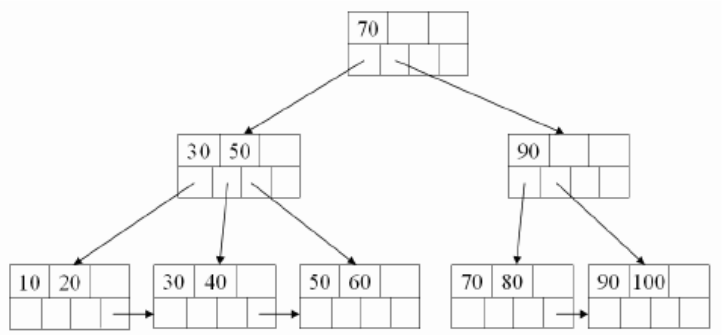
R, with values lying in the range 0 to 999,999. For each of the following queries, amongst the two approaches:
(1) using B+ tree index on $R.A$,
(2) using hash index on attribute $R.A$, and
(3) scanning through the whole heap file for R,
which would most likely require the fewest I/Os for processing the query.

- Find all tuples of R.
- Find all R tuples such that $R.A \in [0, 100)$
- Find all R tuples such that $R.A = 100$

Also assume that after accessing the page, it takes one more disk access to get the actual record.

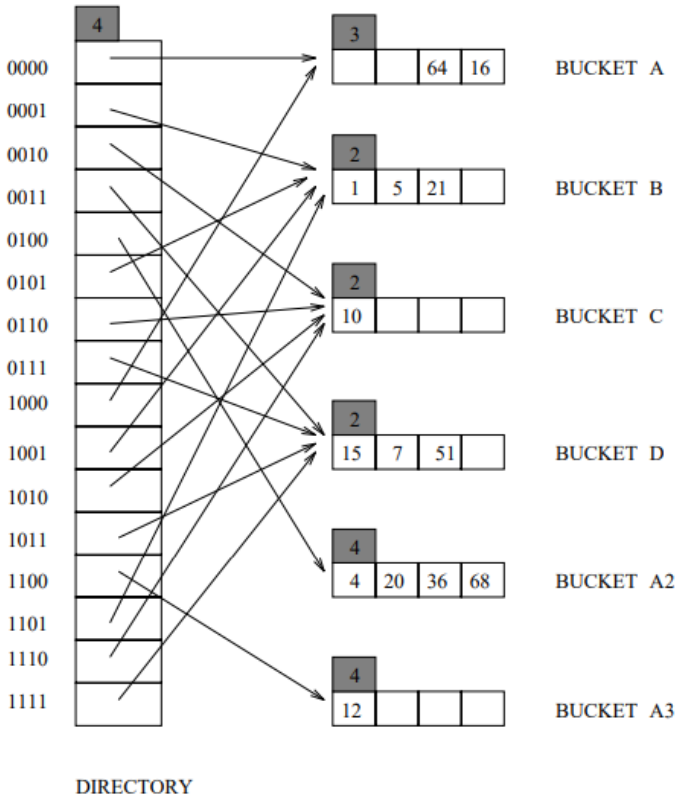
Let h be the height of B+ tree and M be the number of data entries per page ($M > 100$). Let c be the occupancy facdtor in hash indexing. Heap file: $10^6/100 = 10^4$ for all queries.
B+Tree: $h + 10^6/M + 10^6$, $h + 100/M + 100$, $h + 1$.
Hash: $10^6/cM + 10^6$, 2×100, 2×1.
... Hence, Heap is good for fetching all tuples. B+Tree is likely to be better for range query. Hash is likely to be good for equality search.

2. [8pt] Consider constructing a B+tree of order 3 (i.e., n = 3).
   - Show the resulting B+tree after inserting keys $10, 20, 30, 40, 50, 60, 70, 80, 90, 100$ in this order. (4pt)



   - Is it possible to reorder the insertion of the above keys to get a shorter B+tree, e.g., one that has smaller height? If no, explain why. Otherwise, show a new insertion order, and the resulting tree according to your give order. (4pt)
     Yes. We can change ordering so that more leaf nodes are full. For example, 10 20 40 50 70 80 30 60 90 100

3. [7pt] Consider the Extendible Hashing index shown in Figure 1. Answer the following questions about this index.
   (a) Show an index after inserting an entry with hash value 68. (5pt)



(b) From the original index shown in Figure 1, find the minimal set of entries to be deleted from the index that would trigger a merge. (2pt)

Remove 2 entries from Bucket A if allowed to merge data from split image A2. Otherwise, remove 4 entries from Bucket A2.

Exam Revisit - 7pt (exam extra credit)

1. [1pt] Suppose we have a relation `exam(score)` representing scores of all students taking ICCS240 exams. Explain why we should not use a set-model when calculate the ICCS240 average exam score?

   possible multiple scores with the same value.

2. [2pt] Consider the following relations:
   `Computer(maker, model, type, price)`,
   `PC(model, speed, ram, storage)`,
   `Laptop(model, speed, ram, storage, screen)`.

   Both `PC`s, and `Laptop`s are `Computer`s identified by `type` field.

   Without using `MAX()` and `MIN()` functions, find the maker of the fastest PC.
   Warning: it is possible that multiple PCs may have the same speed.

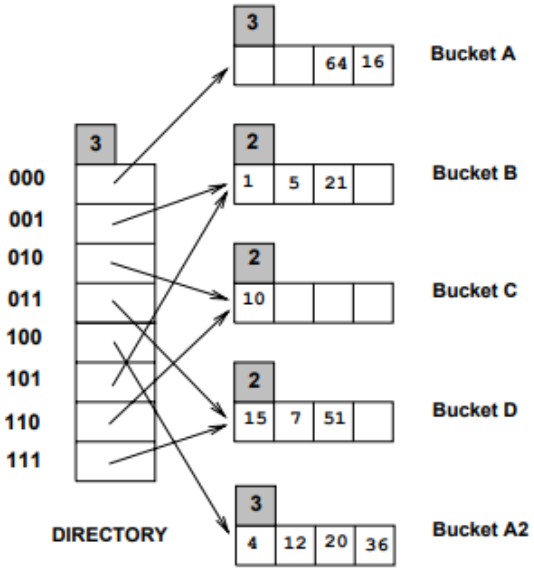


Figure 1: Extendible Hashing index

```
SELECT maker FROM Computer WHERE model = (SELECT model FROM PC WHERE speed >=
ALL (SELECT speed FROM PC));
```

3. [2pt] Given relation $R(A, B, C, D)$ and the following set of functional dependencies $\mathcal{F}$:

$$AC \to B, \quad B \to A, \quad BD \to C, \quad D \to A$$

Find all the *candidate* keys for this relation. Explain your answers.

Consider that $\{BD\}^+ = \{A, B, C, D\}$, $\{CD\}^+ = \{A, B, C, D\}$. Hence, BD and CD are (super)keys. Note that no closure of each of A, B, C or D is the same as $\{A, B, C, D\}$. That is, no $X \subseteq \{BD\}$ and $X \subseteq \{CD\}$ are keys. We have BD and CD are minimal, and so they are candidate keys (and only them since no other closure of two attributes are keys).

4. [2pt] Correctly answer all of the following questions:

- Why $\mathcal{G} : \{A \to BC\}$ is not a minimal basis of $\mathcal{F} : \{A \to B, A \to C\}$?
  R.H.S. contains more than one attribute.
- Is $\mathcal{G}$ a basis of $\mathcal{F}$, and vice versa?
  Yes.
- Find the minimal basis of $\mathcal{F}$.
  $\mathcal{F}$