Computer Science 143
Spring 2018
*Prof. Ryan Rosario*

Name SUYASH SAXENA UID 404735401

# Midterm Exam

Wednesday, May 9, 2018 – 8am

**Form 2**

| Section: | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | Total |
|---|---|---|---|---|---|---|---|---|---|
| Worth: | 7 | 4 | 10 | 10 | 30 | 15 | 15 | 9 | 100 |
| Earned: | 7 | 3 | 7 | 10 | 20 | 7+5 | 7 | 9 | 75 |

## EXAM TIPS AND GUIDELINES

*8 sections, 110 minutes*
*100 points, 25% of final grade*

1. Look through the entire exam before getting started. If you come a question you are unsure about, skip it and come back to it later; never spend too much time (unproductively) on a single question.
2. Simple and clear solutions are recommended. You may receive as few as 0 points if your response is overcomplicated (exceeds the maximum word limit) or if we cannot understand it.
3. If you make any assumptions when solving a problem, describe your assumptions, though this does not guarantee credit.
4. To receive partial credit, show all work.
5. You are permitted **one** 2-sided reference sheet of the US Letter (8.5in × 11in) standard that is readable without any visual aids.
6. **The reference sheet shall be submitted with your exam.**
7. For test security reasons, there will be no break.
8. Computers, cell phones and any other electronic devices of any kind must be turned off and stored off your desk. The only items permitted on your desk are one or more pencils/pens, one of more erasers, your reference sheet, and this booklet.
9. Aside from your reference sheet, no notes or study materials of any kind are permitted and are strictly prohibited.
10. If you need extra space, use the back of each page.
11. Students are not permitted to leave the room before turning in your exam unless you tell us first. You are not to discuss any part of this exam, either specific content or topics covered, while not in this room until the exam ends.
12. Remember that you are bound by the Academic Integrity Agreement and its terms.
13. **You will need to show your photo ID when turning in your exam.**

## DO NOT OPEN UNTIL WE OFFICIALLY BEGIN THE TEST

7

1. (7 points) **Real or Fake News?** ***Directions:*** Read each statement carefully and mark your answer in the margin to the right. Mark `T` if the statement is always true, otherwise mark `F`. Each item is worth $\frac{1}{2}$ point.

   (a) A natural join is conceptually the same as an inner join. (T) (F)

   (b) RDBMS is a use-case of OLTP. (T) (F)

   (c) Every table is required to have a primary key. (T) (F)

   (d) A GPA, of the fixed format 3.771 is best represented as a `DECIMAL(1, 3)`. (T) (F)

   (e) Procedures use the `CALL` syntax whereas functions are called in a query. (T) (F)

   (f) In general, if we ignore performance, we can use a right join in place of a left join if we just change the order of the tables (and any aliases). (T) (F)

   (g) MySQL enforces foreign keys natively. (T) (F)

   (h) For text, a `VARCHAR(n)` and a `BINARY(n)` are the same because each character is one byte. (T) (F)

   (i) In a relation $R$ that has columns $A$, $B$ and $K$, with superkey $K$, the attributes $K$ and $A$ form a superkey. (T) (F)

   (j) When designing a schema, we should always pick the largest data type we may need so we can be sure data will always fit. (T) (F)

   (k) When computing an aggregate, we must always specify a `GROUP BY`. (T) (F)

   (l) When we do a natural join between two relations with no common attributes, we get the empty set. (T) (F)

   (m) Aggregation functions like `MIN`, `MAX` and `SUM` can be used on both rows and columns. (T) (F)

   (n) The main purpose of foreign keys is to improve join performance. (T) (F)

2. (4 points) **Choices, Choices.** ***Directions:*** Choose the *best* answer. Mark your answer in the margin to the right.

(a) When creating a table schema, it is most important to

(A) (B) (C) (D) (E)

A. plan how to efficiently swap from/to disk to/from RAM
B. allow redundant data to speed up querying of aggregates
C. use caching
D. use fixed data types for performance
E. plan well, because schema changes break code

(b) The main difference between `TEXT` and `VARCHAR(n)` is

(A) (B) (C) (D) (E)

A. `TEXT` is stored in a variable length record, `VARCHAR(n)` is stored in a fixed length record.
B. `TEXT` can be indexed, `VARCHAR(n)` cannot.
C. `VARCHAR(n)` can be used as a key, `TEXT` cannot.
D. `VARCHAR(n)` can be longer than text for large `n`
E. Nothing; they are the same.

(c) In relational algebra, the closest operator to a SQL `SELECT` is

(A) (B) (C) (D) (E)

A. $\sigma$
B. $\Pi$
C. $\rho$
D. $\times$
E. $\bowtie$

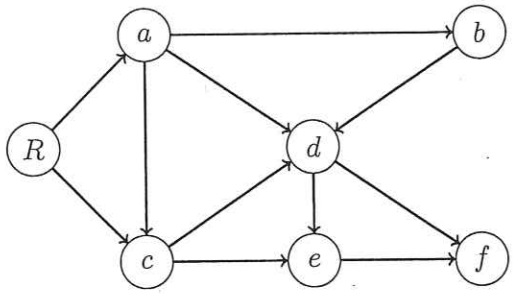
(d) The `CONSTRAINT...CHECK` syntax is a no-op in MySQL. We can enforce constraints with all of the following **except** (choose the *best* answer)

(A) (B) (C) (D) (E)

A. a view
B. modifiers like `NOT NULL` and `UNIQUE`
C. primary and foreign keys
D. a generated column (in newer version of MySQL)
E. a database trigger

3

3. **Access Denied.** We have the following authorization graph for some privilege $p$ granted by root user $R$.




(a) (2 points) Suppose we revoke authorization from user $a$. The user(s) that lose their authorization **under MySQL** with the `CASCADE` option is/are (circle all that apply):

(a) b c d e f

(b) (2 points) Suppose we revoke authorization from user $a$. The user(s) that lose their authorization **under the SQL standard** with `CASCADE` is/are (circle all that apply):

0 (a) (b) c d e f

(c) (2 points) If we instead specify `RESTRICT` mode, what happens if we revoke $p$ from $a$? Mark one answer.

- ○ only $a$ loses the privilege $p$
- ○ $b$, $c$ and $d$ also lose the privilege
- ○ nothing, the revoke is ignored
- ● an error is displayed
- ○ the hard disk suffers a head crash
- ○ vampires invade UCLA

2

(d) (4 points) **Yo Ho, Yo Ho a Spammer's Life for Me.** It's time to put on the pirate hat. Complete the following SQL query so that instead of fetching only your student email address, it fetches email addresses for *all* students at UCLA. Assume the query is not a prepared statement.

4

```
SELECT
uid,
email
FROM student_emails

WHERE uid = " 1 OR UID NOT NULL";
```

4. **A Random Question.** Suppose UCLA Engineering wants to send a survey to random students. We store student information in a table that looks like the following:

```
CREATE TABLE bruin_engineers (
    uid         CHAR(9),
    email       VARCHAR(255),
    major_code  CHAR(4)
    -- i.e. 0201 for COM SCI, 0303 for ELE ENGR etc.
);
```

Further suppose that based on past surveys, we have a table that contains several pieces of data about each UCLA Engineering student and a flag specifying whether or not the student responded (i.e. to model what factors are associated with a student responding or not responding to the survey).

```
CREATE TABLE survey_response_result (
    uid     CHAR(9),
    email   VARCHAR(255),
    ... a bunch of other random stuff...
    responded ENUM('yes', 'no')
);
```

(a) (4 points) Write a query that randomly selects 10% of students (`uid` and `email`) from `bruin_engineers`. Assume you have a function called `RAND()` that generates random numbers from 0 to 1, into a column.

```
SELECT
    uid, email, RAND() AS random
FROM
    bruin_engineers
WHERE
    random > 0.9;
```

4

* Assumes law of large nos.

(b) (6 points) We want to divide the rows of `survey_response_result` into three groups, we use 50% of the data for training (call it group 1), 30% for validation (group 2) and 20% for testing (group 3). Write a query that assigns each row to a particular one of these sets. Again, assume you have a function called `RAND()` like before.

```
SELECT
    *, RAND() as random,
    IF (random <= 0.5, "group 1",
        IF (random <= 0.8, "group 2",
            "group 3")) AS group
FROM
    survey_response_result;
```

6

5. **The Friendly Skies.** Now you work as a safety analyst for Southwest Airlines, where a single aircraft can be used for up to a dozen flights in a day. We have a few tables to study. Note that these tables have some redundant attributes to make this problem simpler. Assume flight numbers uniquely identify a route (origin and destination)

tail, distance, time, dept = same

```
CREATE TABLE equipment_flight (
    departure_time  TIMESTAMP,
    tail            CHAR(6) NOT NULL,
    -- i.e. N1234A, N789SW
    flight          SMALLINT(4),
    PRIMARY KEY(departure_time, flight)
);
-- which individual aircraft services a particular flight on a particular date.

CREATE TABLE flights (
    flight  SMALLINT(4) PRIMARY KEY,
    -- i.e. 0424, 1297
    origin  CHAR(3) NOT NULL,
    destination CHAR(3) NOT NULL,
    -- i.e. DEN, SJC, LAX, BUR, LGA, MMH
    distance SMALLINT NOT NULL
    -- in nautical miles
);
-- flights and their routes

CREATE TABLE snacks (
    flight SMALLINT(4) PRIMARY KEY,
    snack ENUM('pretzel', 'peanut') NOT NULL
);
-- Snacks on a Plane. Only contains rows if snacks are served on the flight.
```

(a) (5 points) Write a query that returns all aircraft (identified by `tail`) that flew/fly more than 5 flights on a particular date. You can assume today (`CURDATE()`) is the particular date.

4

```
SELECT
    tail,
    COUNT(*) AS nflights
FROM
    equipment_flight
WHERE DATE(departure_time) = CURDATE()
GROUP BY tail
HAVING
    nflights > 5;
```

need only 1

+1

(b) (10 points) We need to inspect an aircraft if it flies more than 2,000 nautical miles in 12 hours. For each aircraft (`tail`) and flight departure, write a query to compute the number of miles it flew 12 hours prior to each of its departures. Filter the results to only show the number of miles, per aircraft, for today's (`CURDATE()`) departures only (i.e. using this data, we can plan our inspections for the day, or determine if we missed one). **Hint:** If you use a subquery, you can assign a name to it, like $S$, then you can write your final query in terms of $S$ to avoid repetition and lots of erasing.

```
S = SELECT
      l.tail as aircraft,
      r.distance as distance, dpature_time as dt
    FROM equipmnt_flight l
    JOIN flight r ON
      l.flight = r.flight
```

(6)

Main Qury:

```
SELECT
S1.aircraft, SUM(S1.distance) AS miles
FROM
    S S1, S S2 → wrong join
WHERE
    TIMESTAMPDIFF(S1.dt, S2.dt)
        <= 12 x 60 x 60 x 1000
    AND
    DATE(S1.dt) = CURDATE()
GROUP BY
~~S1.aircraft~~ ok. , flight
```

(-2)

(-2)

(c) (5 points) The safety analyst must make sure that nobody chokes on a pretzel or is allergic to peanuts. Some flights serve snacks, others do not. This is based solely on the flight number. The `snacks` table only contains data about flights that *do serve snacks*. Write a query that returns the routes of today's flights that do/did not serve a snack. *(Your professor needs a snack while flying, so this will help him avoid certain routes)*

```
SELECT
  flight
FROM
  flights
WHERE
  DATE(departure_time) = CURDATE()
  AND flight NOT IN
    (SELECT
      flight
    FROM snacks);
```

need Left/Right join -3

2

(d) (10 points) In lecture we covered the basic relational algebra operators. We also discussed that more complicated operations can be represented using these basic operators. Write a relational algebra expression that *best* represents the query in part (b). **Hint:** If you used a subquery and named it $S$ in part (b), write an expression for $S$ first, and then use it as relation $S$ in the expression for the outer query.

8

$$S = \Pi_{\text{tail, distance, departure\_time}}(\text{equipment} - \text{flight} \bowtie \text{flight})$$

let $x$ = timestamp condition i.e. Timestamp Diff $(s1.dt, s2.dt) < 12 \times 60$.

$y$ = outside condition i.e. $\text{Date}(s1.dt) = \text{CURDATE}$

$$\Pi_{s1.tail,\ s1.distance}\left(\rho_{s1.tail}\, G_{\text{sum}(s2.distance)}\left(\sigma_{x \wedge y}\left(\rho_{s1}(S) \times \rho_{s2}(S)\right)\right)\right)$$

conds on this -2

6. **Can I Join You Two?** *Directions:* Write a *succint* response to each question about joins. Only correct responses will receive credit. Correct responses should not be long. There is a word maximum for each response. Exceeding this maximum may result in 0 points.

   (a) (5 points) Explain the difference between a `CROSS JOIN` and a `FULL JOIN`. Be specific: use examples or drawings to make your point if you must. (100 word maximum)

5

Cross join returns a cartesian product of two or more relations.

Full join returns the intersection of two or more relations on a join condition, in addition to rows from either relation that did not satisfy condition (in the term a union, one side is NULL)

e.g.

| $K_1$ | $K_2$ |
|---|---|
| a | b |
| c | d |

| $K_3$ | $K_4$ |
|---|---|
| b | d |
| e | f |
| g | h |

FULL JOIN ON $K_2 = K_3$

Cross

| $K_1$ | $K_2$ | $K_3$ | $K_4$ |
|---|---|---|---|
| a | b | b | d |
| a | b | e | f |
| a | b | g | h |
| c | d | b | d |
| c | d | e | f |
| c | d | g | h |

Full

| $K_1$ | $K_2=K_3$ | $K_4$ |
|---|---|---|
| a | b | d |
| c | d | NULL |
| NULL | e | f |
| NULL | g | h |

   (b) (5 points) In class, we discussed extensively the use of `JOIN` in a `SELECT`. Give a situation when we might want to use a `JOIN` in a `DELETE` statement and *also* in an `UPDATE` statement, separately. (50 word maximum)

2

Update: We have normalized schema such that an update to key in $t1$ must trigger a change in $t2$ for select rows on join condition

-1

e.g. Order update for Person in | Id | Person | , | Order | PersonId |

Delete: -2

Deleting record from $t1$ means that connected record in $t2$ also deleted

| Id | Name | ⟷ | Order | Id |

Once name deleted, order history also deleted

(c) (5 points) Suppose we have two very large tables $A$ and $B$ divided between two machines (table $A$ is on machine 1 and table $B$ and machine 2) and neither table fits in RAM. We wish to join both tables on a particular key. Describe how we can perform this join quickly as to minimize runtime, disk I/O, and network I/O. (150 word maximum)

Create a bloom filter on join key of the smaller table and use it to filter out keys on the larger table that are 100% not matching. This vastly reduces scans reading w/ minimal space and network I/O.

We may still have false positives. Can use on hash join filter to check if keys match to completely filter out false positives such that overall runtime, disk I/O is also minimized.

**Half of this page is intentionally blank.**

7. **If you Liked it Then You Should Have Put an *Index* on It.**

Suppose we want to use various indexing methods to answer the following query across 1 billion records. Assume all keys in the index are unique.

```
SELECT *
FROM r
WHERE attribute = value;
```

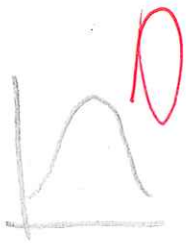
(a) (2 points) Suppose we use a naive complete binary tree to index into blocks on the disk. In the worst case, how many disk block reads must we perform?

$2 \log_2(1 \text{ billion})$ ✓

(b) (2 points) Now suppose we instead use a B+-tree with $n = 100$. In the worst case, how many disk block reads must we perform?

$2 \log_{50}(1 \text{ billion})$ ✓

(c) (5 points) Now suppose we have an ordered index where each key $x$ comes from a particular (exponential) distribution, where $f(x|\lambda) = \lambda e^{-\lambda x}$ where $\lambda$ is a constant and $x \in [0, \infty)$. We want to find out where we should begin a binary search over the keys in this ordered index to find a particular key value $k$. Write an expression in terms of $k$ that finds the location in the index where we should begin the search.



Location to start searching from $= \int_0^k \lambda e^{-\lambda x} \, dx$

(d) (6 points) Based on what we have discussed, *think about* how a Bloom Filter differs from a hashing index structure. List *one* advantage of a Bloom Filter over a hashing index structure and list *one* disadvantage of a Bloom Filter vs. an index. *There is no need to exceed the given lines.*

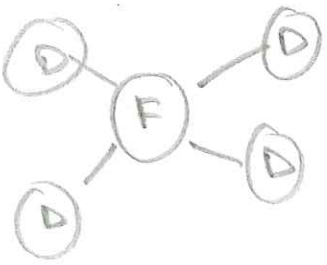
Advantage: Space efficient - huge data is compactly represented in a bit array << No of records of size

3

Disadvantage: More collisions and false positives in Bloom by nature of being a compacted representation. Need a hash bucket for each key in index.

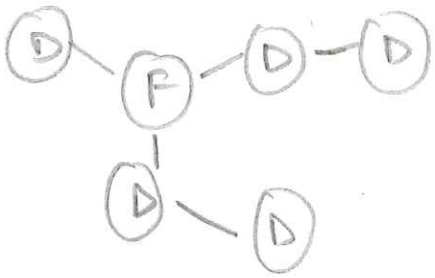
8. (9 points) **Cubism for Computer Scientists.** List the *three* most common schemas used in OLAP. Draw an example graph representing each schema. Denote a dimension table as $D$ and a fact table as $F$.

Star



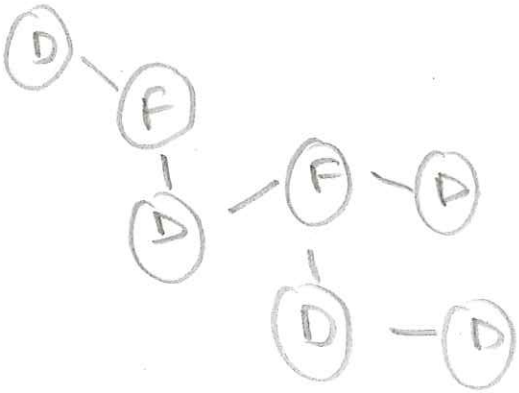

One fact table
Many dimension

Snow flake




i.e. A normalized form of Star

Constellation




i.e. Multiple fact tables + dimension tables

## Page Intentionally Left Blank

## END OF EXAM

1. Make sure that you have written your name and UID on page 1.
2. Make sure you marked all of your answers for questions 1 and 2 in the right hand margin of pages 2 and 3. Marks should be distinct, with errors erased completely so we do not mistake them as intentional responses.
3. Make sure your responses are legible and that errors are cleanly erased or crossed out.
4. **Your reference sheet *must* be handed in with your exam. Please put your UID and name on it.**
5. **Prepare to show a photo ID when handing in your exam.**