Computer Science 143
Spring 2018
Prof. Ryan Rosario

[redacted]

MIDTERM EXAM

Wednesday, May 9, 2018 – 8am

Form 4

Section:	1	2	3	4	5	6	7	8	Total
Worth:	7	4	15	10	30	10	15	9	100
Earned:	6	4	10	10	23	6	8+4	0	71

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. 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.
11. Remember that you are bound by the Academic Integrity Agreement and its terms.
12. **You will need to show your photo ID when turning in your exam.**

DO NOT OPEN UNTIL WE OFFICIALLY BEGIN THE TEST

6

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

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

defaults to a single group as long as query isn't ambiguous

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

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

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

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

(h) 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)

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

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

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

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

X (m) MySQL enforces foreign keys natively. (T) (F)

requires InnoDB storage engine

X (n) 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)

pick the smallest type and promote as needed to save space

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

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

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

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

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

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

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

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

A. σ
B. Π
C. ρ
D. $\times$
E. $\bowtie$

4

3. **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

$\sim \log_2 (1{,}000{,}000{,}000)$

(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?

1

~~$\sim \log_{100} (1{,}000{,}000{,}000)$~~

$\sim \log_{50} (1{,}000{,}000{,}000)$ when each node is only half-full

(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 λ 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.

1

~~$1{,}000{,}000{,}000 \times \sum_{x=0}^{k-1} \lambda e^{-\lambda x}$~~

$1{,}000{,}000{,}000 \times \int_0^k \lambda e^{-\lambda x} \, dx$

$1{,}000{,}000{,}000 \times \left[-e^{-\lambda x}\right]_{x=0}^{k}$

$1{,}000{,}000{,}000 \times (1 - e^{-\lambda k})$

(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: a Bloom Filter takes less space

Disadvantage: a Bloom Filter can give false positives

6

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.

4

```
Select
  S.uid,
  S.email
from (
  Select
    uid,
    email,
    rand() as r
  from bruin_engineers
) as S
where S.R < 0.1;
```

better:

```
select
  uid,
  email
from bruin_engineers
where rand() < 0.1;
```

(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.

6

```
Select
  S.uid,
  S.email, ... ("random stuff")...
  if(S.R between 0 and 0.5, "training", if (S.R between 0.5 and 0.8, "validation", "testing"))
from (
  Select
    uid,
    email, ... ("random stuff")...
    rand() as r
  from survey_response_results
) as S
```

better:

```
Select
  uid,
  email,
  case
    when rand() < 0.5 then "training"
    when rand() < 0.8 then "validation"
    else "testing"
  end
from survey_response_results
```

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)

```
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.

5

```
select
   tail
from equipment_flight
where date(departure_time) = curdate()
group by tail
having count(distinct flight) > 5 ;
```

not (departure_time, flight)!

(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.

```
select
    equipment_flight.departure_time,
    equipment_flight.tail,
    coalesce(sum(S.distance),0) as recent_distance
from equipment_flight
left join S on timediff(equipment_flight.departure_time,
        S.departure_time) between "00:00:01" and "12:00:00"
    and equipment_flight.tail = S.tail
where date(equipment_flight.departure_time) = curdate()
group by tail, flight
```

```
S = (select
    departure_time,
    tail,
    equipment_flight.flight as flight,
    distance
from equipment_flight
join flights on flights.flight = equipment_flight.flight)
```

equipment_flight.departure_time,
equipment_flight.tail,
equipment_flight.flight

One inequijoin (and missing) −1

(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
    flights.flight,
    flights.origin,
    flights.destination,

from flights
left join snacks on snacks.flight = flights.flight
join equipment_flight on equipment_flight.flight = flights.flight
where date(departure_time) = curdate()
  and isnull(snacks.snack)
```

Don't need 2 joins

correct

-1 Snacks is null

3

(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.

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

Π ~~$distance\ G_{tail}\ (\sigma_{date(equipment\_flight.departure\_time) = curdate()}\ (equipment\_flight \bowtie timediff(equipment\_flight.departure\_time, S.departure\_time)\ between\ "00:00:00"\ and\ "12:00:00"\ and\ equipment\_flight.tail = S.tail)\ S)$~~

-2 -2

$\Pi_{equipment\_flight.departure\_time,\ equipment\_flight.tail,\ distance}\ [$

$equipment\_flight.departure\_time,\ equipment\_flight.tail\ G_{sum(s.distance)}\ [$

$\sigma_{\theta}\ [\ equipment\_flight \times S\ ]$

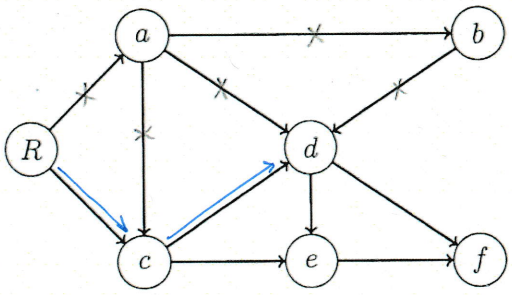
$]$

$]$

θ = date(equipment_flight.departure_time) = curdate()
^ timediff(equipment_flight.departure_time, S.departure_time) between "00:00:00" and "12:00:00"
^ equipment_flight.tail = S.tail

6

6. **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):

2 (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):

1 (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.

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

(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.

3 correct

```
SELECT
uid,
email
FROM student_emails

WHERE uid = "???"  or  "1" = "1";
```

7. **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

A cross join is a Cartesian product while a full join has a condition. For example:

Table X

key	value
A	P
B	Q

Table Y

key	value
A	R
C	S

Cross Join Result

X.key	X.value	Y.key	Y.value
A	P	A	R
A	P	C	S
B	Q	A	R
B	Q	C	S

Full Join Result (equijoin on key)

key	X.value	Y.value
A	P	R
B	Q	∅
C	∅	S

(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)

3

in a delete: when you want to delete row(s) in one table based on a value corresponding to that row given in another table

in an update: when you want to update row(s) in one table based on a value corresponding to that row given in another table

in the example above, deleting rows in X if the corresponding Y.value is R.

in the example above, updating rows in X if the corresponding Y.value is R.

(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)

Machine A sends a Bloom filter for the keys it has to machine B. Machine B checks the Bloom filter (against an index in RAM) to fetch from disk and send over the network only those records matching the Bloom filter. Machine A or a third machine checks for false positives and completes the join.

hashing-based

hashing?

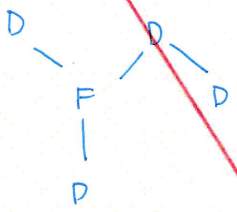
+4

Half of this page is intentionally blank.

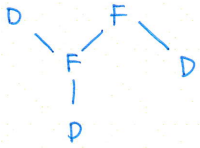
8. (9 points) **Cubism for Computer Scientists.** List the *three* most common schemas used in OLAP. Draw an example graph representing each schema. Denote a dimension table as D and a fact table as F.

- star

- snowflake



- constellation



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.**