

COM SCI 143 Midterm Forms 1-5

Jordan Xian Yue Soon

TOTAL POINTS

71 / 100

QUESTION 1

1 Total **71 / 100**

+ **37 pts** 71

+ **71** Point adjustment

UNIVERSITY OF CALIFORNIA, LOS ANGELES
Department of Computer Science

Computer Science 143
Spring 2019

Prof. Ryan Rosario

MIDTERM EXAM

Wednesday, May 1, 2019 - 8am

Form 2

Name Jordan Xian Yue Soon UID 004909667

Section:	1	2	3	4	5	6	7	Total
Worth:	14	7	18	28	10	18	5	100
Earned:	14	5	6	18	10	14	4	71

EXAM TIPS AND GUIDELINES

*7 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 assumptions when solving a problem, describe them, 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, is prepared by you and only you, and is handwritten.
6. For test security reasons, there will be no break.
7. 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 or more erasers, your reference sheet, and this booklet.
8. Aside from your reference sheet, no notes or study materials of any kind are permitted and are strictly prohibited.
9. Students are not permitted to leave the room before finishing the exam **without first signing out with a TA and giving them your exam**. 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. Only one student shall be allowed to leave at a time.
10. Remember that you are bound by the Academic Integrity Agreement and its terms.
11. **You will need to show your photo ID when turning in your exam.**

DO NOT OPEN UNTIL WE OFFICIALLY BEGIN THE TEST

1. (14 points) **Real or Fake News?** *Directions:* Read each statement carefully and mark your answer in the margin to the right. For some statement to be true, it must be universally true across all possible relation instances under all data and constraint assumptions. Otherwise, it is false. Mark T for True and F for False.

(a) A theta join explicitly specifies a join condition.

F

(b) The main purpose of foreign keys is to enforce referential integrity.

F

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

T

(d) When specifying a GROUP BY clause, we must always also use an aggregate function within the SELECT.

T

(e) A data warehouse only stores aggregates.

T

(f) A natural join in relational algebra and SQL can only be an inner join.

T

(g) For convenience, any column aliases created in a SELECT statement can be used in a WHERE and GROUP BY clause.

T

(h) When user U has some privilege revoked, all of its descendants also lose the privilege.

T

(i) Suppose R is a relation with primary key $K = (A, B, C)$, where K is also the minimal candidate key. Another relation S also contains C . C is a foreign key referencing R .

T

(j) A VIEW improves query performance by only storing references to data in a table.

T

(k) A full outer join gives a Cartesian product.

T

(l) A SELECT clause in a SQL query is best represented by Π in the relational algebra.

F

(m) The standard DISTINCT applies across all attributes in a SELECT and returns a set of unique tuples.

F

(n) An inner join between relations R and S can have a one-to-one, one-to-many, or many-to-many relationship between join keys in R and S , assuming the join key in R is a primary key.

T

2. (7 points) **Choices, Choices.** Choose the *best* answer. Mark your answer in the margin to the right.

(a) The main difference between a `varchar` and `varchar(n)` in PostgreSQL is

A B C D E

- A. using `varchar` is bad style
- B. `a` requires more space than `b` for small n
- C. `varchar` is varying, but `varchar(n)` is the same as `char(n)`.
- D. PostgreSQL will throw an error if the length of `b` exceeds n .
- E. There is no difference.

(b) A subquery can appear in all of the following parts of a query **except**

A B C D E

- A. FROM, JOIN
- B. SET
- C. WHERE, HAVING
- D. GROUP BY
- E. INSERT, DELETE

(c) All of the following are OLAP operations **except**

A B C D E

- A. cube
- B. pivot
- C. slice and dice
- D. rollup
- E. drilldown

(d) One main purpose of sharding in a distributed database system is to

A B C D E

- A. provide fast access to aggregates
- B. protect data against natural disasters
- C. reduce latency between the user and the RDBMS
- D. enforce referential integrity
- E. provide fast lookup in a cache

(e) When we want to specify that a query should run when some condition occurs, we would want to use `a(n)`

A B C D E

- A. stored procedure
- B. user-defined function
- C. check constraint
- D. assertion
- E. trigger

(f) All of the following are ways to protect user privacy in a database **except**

A B C D E

- A. hashing private data
- B. using two-factor authentication to connect to the physical server
- C. imposing a strict retention policy on a table containing private data
- D. constructing queries using prepared statements
- E. using views instead of tables

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

A B C D E

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

3. It's So Logical. Answer the following questions about the relational algebra.

For items (a) through (d), consider a relation $R(A, B)$ that contains r tuples and a relation $S(B, C)$ that contains s tuples. Assume that $r > 0, s > 0$. Make no assumptions about keys. For each of the following expressions, state in terms of r and s the minimum and maximum number of tuples that could be in the result of the expression. **Note:** The correct response may require a function be applied to r and s .

(a) (2 points) $\Pi_{A,C}(R \bowtie S)$

Minimum: ~~MAX(rs)~~, Maximum: ~~rs~~ ^{ok}

(b) (2 points) $\Pi_B(R) - (\Pi_B(R) - \Pi_B(S))$

Minimum: 0, Maximum: ~~r~~

(c) (2 points) $(R \bowtie R) \bowtie R$

Minimum: ~~r~~ ^{ok}, Maximum: ~~r~~ ^{ok}

(d) (2 points) $\sigma_{A>B}(R) \cup \sigma_{A<B}(R)$

Minimum: 0, Maximum: r

Sorry, you were first to be graded :)

(e) (5 points) We learned some basic relational algebra operators and discussed that more complex operators could be constructed based on them. Let R and S be relations. Write a relational algebra expression for R left semijoin S . The left semijoin essentially uses values of S to filter and keep values in R that match on the join key, and then restricts the result to a subset of attributes in R . Thus, the semijoin is sometimes called *restriction*. Write the relational algebra expression for the left semijoin using the operators we learned in class.

$R \ltimes S =$ _____

(f) (5 points) The *antijoin* performs the converse operation of the semijoin. It extracts tuples in R , that do not have a match in S . Write a relational algebra expression for the antijoin. You may use your answer from the previous part in your response.

$R \triangleright S =$ _____

4. **The Not So Friendly Skies.** Suppose you work as a safety analyst for Southwest Airlines. We have a few tables to study. Due to the number of flights operated each day, flight numbers may be recycled, including on the same day. For example, flight 528 on Wednesdays starts in Albany and stops in Orlando, Raleigh, Atlanta, Houston, El Paso, Los Angeles and terminates in San Jose. Assume that each flight number uses the same aircraft (denoted by tail, which is like a license plate) throughout its journey. Assume that no flights span two days (depart before midnight, arrive after midnight).

You will notice that there are no foreign keys defined. We ignore them for simplicity, and assume that these tables are static with no inserts, updates or deletes.

Special Note: *NONE of the queries in this section require using more than one subquery, or more than one level of subquery. Complicated queries that use additional subqueries will not receive credit. Please use standard JOIN queries rather than using multiple tables in FROM.*

```
CREATE TABLE eq_flt (
    fltno          smallint,
    tail          char(6) NOT NULL,
    -- i.e. N1234A, N789SW
    PRIMARY KEY(fltno, depart)
    -- minimal key, no flight number can depart at same time to different destination.
);
-- which individual aircraft (tail) services a particular flight on a particular date/time.
```

```
CREATE TABLE flt (
    fltno          smallint,
    -- i.e. 424, 1297
    sch_dep        timestamptz, -- scheduled departure
    sch_arr        timestamptz, -- scheduled arrival
    orig           char(3) NOT NULL,
    dest           char(3) NOT NULL,
    -- i.e. DEN, SJC, LAX, BUR, LGA, MMH
    dist           smallint, -- distance in nautical miles (redundancy for simplicity)
    PRIMARY KEY(fltno, sch_dep)
);
-- flights and their routes
```

```
CREATE TYPE model AS enum ('B737', 'B738', 'B38M');
-- B737 = 737-700, B738 = 737-800, B38M = 737 MAX 8
CREATE TABLE actype (
    tail          char(6) PRIMARY KEY,
    ac            model NOT NULL
);
-- mapping tails to aircraft model type
```

REST OF PAGE INTENTIONALLY LEFT BLANK

- (a) (5 points) Management has decided to cut costs and only restock snacks and drinks on an aircraft when it either has no more flights that day, or once it flies more than 2,000 nautical miles in a day. We want to get a head start, so we want to find out which aircraft (tails) will exceed the 2,000 nautical mile mark tomorrow. Write a query to answer this question. Assume you have a function TOMORROW() that returns tomorrow's date and a function DATE() that extracts the date part of a timestamp.

```

SELECT ef.tail
FROM (SELECT fltno, SUM(dist) AS total
      FROM flt
      WHERE DATE(sch-dep) = DATE(NOW())
      AND DATE(sch-arr) <= TOMORROW()
      GROUP BY fltno
      HAVING total > 2000) temp
JOIN eq-flt ef
ON ef.fltno = temp.fltno;

```

subquery -3

- (b) (8 points) A journalist believes that Southwest was flying the beleaguered Boeing 737 MAX 8 for longer distances without an inspection than it should have been. The journalist wants to take each MAX 8 aircraft (tail), and all of the distances that they flew (one distance for every takeoff/landing) and then compute the 95th percentile of those distances. Fill in the window function below to answer this question. Assume you have a function PERCENTILE() that computes the percentile associated with each distance. You may assume that one flight will be the exact 95th percentile and that it is denoted by the value 95. **Hint:** Columns produced in joins are available to the window function.

```

SELECT
  ef.tail,
  dist,
  PERCENTILE(SUM(dist)) OVER (
    PARTITION BY ef.tail
    ORDER BY sch-dep
  ) AS mileage_percentiles
FROM flt f
JOIN eq-flt ef ON f.fltno = ef.fltno
JOIN actype act ON ef.tail = act.tail
WHERE act.ac = 'B38M' AND mileage_percentiles = 95 -- percentile
ORDER BY dist DESC;

```

- (c) (10 points) A crew fuels an aircraft before it takes off, and records the amount of fuel, and a separate crew checks the fuel level after a flight arrives at the gate. Suppose it's not uncommon for a mechanic to forget to record the landing fuel level, particularly if the tanks are known to be near empty at arrival. These recordings are stored in two tables shown below. Write a query that computes the average liters of fuel consumed per route (origin, destination pair). Only consider routes that the airline actually flies (not all pairs of origin/destination). You can assume that if the mechanic did not enter a fuel reading when the plane arrived, then the total fuel consumed shall be the amount of fuel that was present before takeoff for that particular journey. **Hint:** Be very careful. You may want to sketch out this query on the back of the page before writing a response.

```
CREATE TABLE to_fuel (
  fltno smallint,
  sch_dep timestampz,
  orig char(3),
  dest char(3),
  liters integer,
  PRIMARY KEY(fltno, sch_dep)
```

```
CREATE TABLE arr_fuel (
  fltno smallint,
  sch_dep timestampz,
  orig char(3),
  dest char(3),
  liters integer,
  PRIMARY KEY(fltno, sch_dep)
```

```
);
SELECT to.orig, to.dest, AVG(to.liters - COALESCE(arr.liters, 0)) AS avg_fuel
FROM to_fuel to
JOIN flt f
  ON to.fltno = f.fltno AND to.sch_dep = f.sch_dep AND to.orig = f.orig AND to.dest = f.dest
  -- To Make sure record is an actual flight in flt
LEFT JOIN arr_fuel arr
  ON to.fltno = arr.fltno AND to.sch_dep = arr.sch_dep to.orig = arr.orig AND to.dest = arr.dest
  -- Missing landing fuel results in NULLs
GROUP BY to.orig, to.dest;
```

not needed

missing subquery

7

- (d) (5 points) In this problem we stored the fuel measurements in two different tables. An alternative approach would be to have one record per takeoff/landing. Briefly explain which should be preferred and why. (50 word maximum)

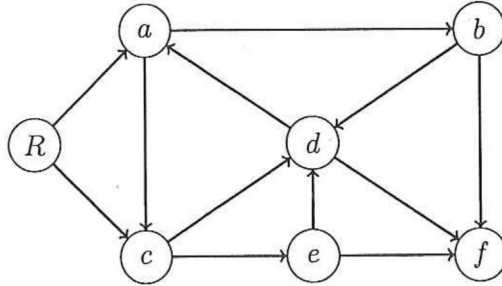
One record would be preferred because there'll be less redundant data across tables, requiring less joins. A non-recorded landing fuel level could just be 'recorded' as a NULL.

is that good?

Updates?

7

5. **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 with the CASCADE option is/are (mark all that apply):

(b) (2 points) After the previous revoke succeeds, suppose we then revoke authorization from user c . The user(s) that lose their authorization with the CASCADE option after all (including those in part a) of these revocations is/are (mark all that apply):

(c) (6 points) **Yo Ho, Yo Ho a Hacker's Life for Me.** It's time to put on the pirate hat. Suppose we have a web app that retrieves contact info for a particular UCLA student using a table `directory`. Suppose in the same schema there is a relation `bol` containing BOL usernames and passwords (with attributes `username` and `password`). Assume the user has read access to both `bol` and `directory` tables and, for simplicity, that both columns in both tables are `text`. Perform a SQL injection to steal all usernames and passwords using the following query. The query must **only** returns usernames and passwords. Note that you may not need all of the lines provided.

```

SELECT
  name,
  email
FROM directory
WHERE name = 'jordan' AND name = 'impossible'; SELECT username, password
FROM bol;

```

Clever

3. More Queries and Joins. These problems deal with recursion and joins.

- (a) (8 points) **Repeating it Over, and Over, and Over, ... and Over.** In class, we saw examples of a recursive query that traverses a tree down to its leaves – an implicit termination condition. We can use an *explicit* termination condition by using variables, or by making clever use of constant value columns. Suppose we have the following table containing CS classes and their prerequisites.

```
CREATE TABLE prereqs (  
    course varchar(8),    -- i.e. CS 1, CS 280RA  
    prereq varchar(8)  
);
```

Assuming we can take unlimited classes each quarter, write a recursive query that will give us a list of courses that we could complete within 2 (and only 2) quarters of completing CS 111 (Operating Systems).

```
WITH RECURSIVE two_qtrs_later AS (  
    -- Non-recursive term:  
    SELECT  
        0 AS qtrs_since_111,    -- 0 is the number zero  
        course,  
        prereq  
    FROM prereqs  
    WHERE course = 'CS 111' |  
  
    -- Connector for Non-Recursive and Recursive Terms:  
    UNION |  
    -- Recursive term:  
    SELECT tq1.qtrs_since_111 + 1 AS qtrs_since_111, |  
        course p.course, |  
        p.prereq |  
    FROM prereqs p  
    INNER JOIN two_qtrs_later tq1 |  
    ON tq1.course = p.prereq |  
    WHERE qtrs_since_111 <= 2 |  
  
)  
SELECT  
    DISTINCT qtrs_since_111, course  
FROM two_qtrs_later  
WHERE qtrs_since_111 > 0  
ORDER BY qtrs_since_111, course;
```

8

Directions: Write a *succinct* response to each question. Only correct responses will receive credit. Correct responses should not be long. There is a word maximum for each response and responses should not exceed the provided lines. Exceeding this maximum may result in 0 points.

- (b) (5 points) The recursive term in the CTE from part (a) cannot contain any OUTER JOINS. Explain why you believe this must be the case. What would happen if the recursive term were allowed to contain an OUTER JOIN? Your response should fit on the lines. (50 words maximum)

If it allowed an OUTER JOIN, there would be nothing ensuring that we only move down the tree. We might end up moving up as well such that a class like CS 35L or CS 33 (which are CS 111's) prereqs get outputted in by the query.

- (c) (5 points) Can I Join You Two? Give a specific example of an application when we would want to intentionally compute a CROSS JOIN between two distinct relations R and S . A response that simply repeat the definition of a CROSS JOIN without a specific application will not receive credit. (50 words maximum)

We might wish to examine all potential roommate pairings to compute suitability or compatibility ratings. We would need to pair every tuple with each other in R and S to compute the scores.

REST OF PAGE INTENTIONALLY LEFT BLANK

7. (5 points) **It's Purely Functional.** Look at the following distributed SQL query and schema (for say, Hive or Spark SQL). The table below lists several parts of the query. Put a (✓) in the box to represent which phase in a MapReduce job each part of the query is executed in.

```

CREATE TYPE party AS enum ('D', 'R');
CREATE TABLE reddit_comments (
    political_party party,
    comment          text
);

SELECT
    political_party,
    AVG(SENTIMENT(comment)) AS average_sentiment
FROM reddit_comments
WHERE SENTIMENT(comment) IS NOT NULL
GROUP BY political_party;

```

Component	Map Phase?	Reduce Phase?
SELECT political_party, comment	✓	
SENTIMENT(comment)	✓	
AVG(...)		✓
WHERE	✓	
GROUP BY		✓

✗

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. Prepare to show a photo ID when handing in your exam.