Question Booklet

W4111 Introduction to Databases
Fall 2018
Midterm Exam Solutions
Instructor: Eugene Wu

Closed Book, 1 page notes: 8.5x11" letter paper, both sides
Duration: 75 minutes
501 Northwest Corner if the last digit of your UNI is 0,1,2,3,4,5
Pupin 329 if the last digit of your UNI is 6, 7, 8, or 9

Instructions

This is the question booklet, which contains questions for the exam.
There is a separate answer booklet for your answers.

1. You are supposed to write your answers on the answer sheets.
2. The staff will ignore text written on the question sheets.
3. You will submit the Quesion AND Answer booklets at the end of the exam. If we do not receive both booklets with your UNI, **you will receive a zero**.

Your Name: Alice H. Acker
Your UNI: aa0000

For regrade requests If we added your score incorrectly, submit a regrade request on Gradescope.
If there is an error in the solutions, please let us know with a **private piazza message**.
If you want a regrade of a question, we will *regrade the entire exam* carefully.

1 (16 points) Equivalences

For each of the following pairs of queries (in relational algebra or SQL), you will write the contents of two database instances. The databases have the following schemas:

```
A(a int, b int);
B(a int, b int)
```

(2 Points) The first database instance should be populated with one or more rows so that Q1 and Q2 output different results. If Q1 and Q2 are equivalent, then write "identical" next to the empty tables instead.

(2 Points) The second database instance should be populated with one or more rows so that Q1 and Q2 return the same results. If this is not possible, write "not possible" next to the empty tables instead.

1.1 (4 Points, 2 Per Database Instance)

Q1: $A \bowtie_a B$
Q2: $A \bowtie B$

> Different: Any non-empty table is acceptable. The queries return different schemas.
>
> Same: Not possible

1.2 (4 Points, 2 Per Database Instance)

Q1: $A \bowtie_a B$
Q2: $A \times \sigma_{B.a='a'}(B)$

> Different: Any non-empty table is acceptable. The queries return different schemas. Stating that Q2 is an error is also acceptable
>
> Same: Not possible

1.3 (4 Points, 2 Per Database Instance)

Q1: $\sigma_{\$1=\$3}(A \times B)$
Q2: `SELECT * FROM A, B WHERE A.a = B.a`

> Different: Note that SQL is multiset semantics, so database instances that allow for duplicate values will produce different query results. A contains (1,2), (1,2). B contains (1,4)
>
> Same: database instances that do not result in duplicates are acceptable. A trivial instance is one where the result is empty: A contains (1,2), B contains (3,4)

1.4 (4 Points, 2 Per Database Instance)

Q1:
```
SELECT * FROM A JOIN B ON A.a = B.a WHERE B.a = 1 or B.b = 2
```
Q2:
```
SELECT * FROM A JOIN B ON A.a = B.a WHERE B.a = 1
 UNION ALL
SELECT * FROM A JOIN B ON A.a = B.a WHERE B.b = 2
```

Different: UNION ALL preserves duplicates, thus any instances where the subqueries in Q2 return overlapping tables is acceptable.

Same: Any instances where the subqueries in Q2 do not overlap in records is acceptable.

2 (18 points) Entity-Relationship Models

2.1 Constraints (3 Points)

Your friend downloaded the following CSV file and shared it with you. What constraints, if any, can be inferred from this dataset? If there are none, simply write "none" and explain why in one sentence:

name	age	state
amy	18	TX
amy	18	CA
amy	18	FL
joe	20	MA
joe	20	NY

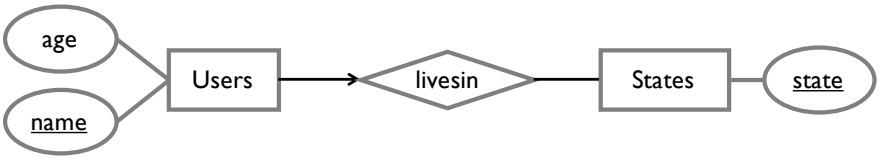
Table 1: Table for Problem 2

Constraints are a property of EVERY database instance. Constraints cannot be inferred from a single database instance.
We did not deduct points if you only statement the following constraint: primary key consists of all three attributes.

2.2 ER Constraints

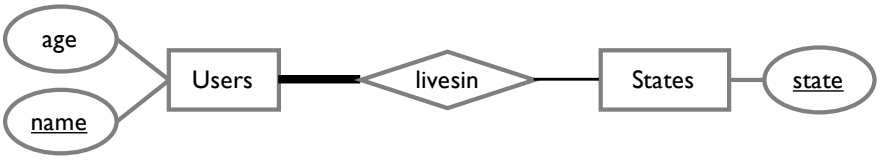
For each of the following ER diagrams, select TRUE if Table 1 satisfies the constraints depicted in the diagram, and FALSE otherwise. If FALSE, write a short sentence about why.

2.2.1 (3 Points)



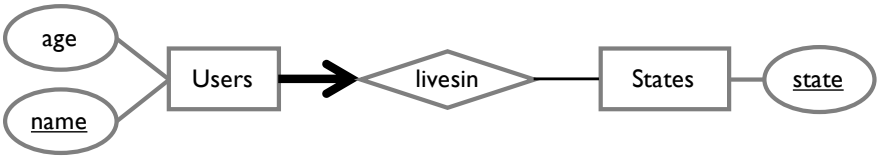
False. Violates at most one constraint.

2.2.2 (3 Points)



True

2.2.3 (3 Points)



False. Violates at most one constraint.

2.3 ER to SQL

2.3.1 (3 Points)

Translate the ER diagram from Problem 2.2.1 into SQL.

This can be translated into two tables or three. We looked for the correct primary key. Common mistakes include stating that states should be unique.

```
CREATE TABLE users_livesin(
  name text primary key,
  age int,
  state text references States
);
CREATE TABLE state(
  state text primary key
)
```

or

```
CREATE TABLE users(
  name text primary key,
  age int,
);
CREATE TABLE livesin(
  name text references users,
  state text not null references state,
  primary key (name)
)
CREATE TABLE state(
  state text primary key
)
```

2.3.2 (3 Points)

Translate the ER diagram from Problem 2.2.3 into SQL.

Can only be expressed by merging users and livesin. We looked for the correct primary key and NOT NULL constraint.

```
CREATE TABLE users_livesin(
  name text primary key,
  age int,
  state text NOT NULL references States
);
CREATE TABLE state(
  state text primary key
)
```

3 (8 points) Triggers

Consider the following database schema, and each table is empty:

```
CREATE TABLE A(a int);
CREATE TABLE B(b int);
```

In this question, we will present two implementations of triggers. Assuming the two tables are initially empty, and you run the following `INSERT` statements, you will write the final contents of `A` and `B` in the answer sheet.

```
INSERT INTO a VALUES(1);
INSERT INTO a VALUES(2);
INSERT INTO a VALUES(3);
```

3.1 Triggers

3.1.1 (4 Points, 2/table)

```
CREATE FUNCTION UDF1() RETURNS trigger
AS $$
BEGIN
  INSERT INTO b VALUES(NEW.*);
  RETURN NEW;
END;
$$ language plpgsql;
```

```
CREATE TRIGGER T1
BEFORE INSERT ON a
FOR EACH ROW
  EXECUTE PROCEDURE UDF1();
```

Both tables contain 1,2,3. This is because the UDF returns a non-null value, and the UDF inserts each new row into b.

3.1.2 (4 Points, 2/table)

```
CREATE FUNCTION UDF2() RETURNS trigger
AS $$
BEGIN
  INSERT INTO b VALUES(NEW.*);
  RETURN null;
END;
$$ language plpgsql;
```

```
CREATE TRIGGER T2
AFTER INSERT ON a
FOR EACH ROW
  EXECUTE PROCEDURE UDF2();
```

Both tables contain 1,2,3. The UDF runs AFTER the insert, so its null return value has no effect.

4 (10 points) Misc. Questions

4.1 (2 Points)

In at most 2 short sentences, describe the significance of integrity constraints in database management systems as compared to writing code to check constraints within the application.

Constraints are stored together with the data and always enforced by the database system. It is very difficult to check every piece of data that may change the database within the application code.

4.2 (2 Points)

List 2 important properties that the relational model provides that the Network/Hierarchical model does not provide. 4 words MAX for each property.

Data independence
Declarative query interface
Avoids redundancy sometimes.

4.3 (2 Points)

In at most 2 short sentences, describe the difference between `VIEW` and `WITH` in SQL.

WITH defines a (possibly recursive) temporary table that lasts for the duration of a single query. VIEWs are tables defined as a query that persists until the view is dropped, and is evaluated by rewriting references to the VIEW with its query definition.

4.4 (2 Points)

In ONE sentence, explain multiset semantics.

Extends relational algebra's set semantics to allow duplicates.

4.5 (2 Points)

Write a creative example of joins In Real Life by filling in the sentence in the answer sheet. Most creative answer (subjectively judged by the staff) gets 2 extra credit points.

Accepted any answer that is sensible and can actually be viewed as a join. A common mistake was writing a filter as the join condition. Another was to fill the blanks with words that seem similar. Another was to join individual entities rather than actual SETS of entities.

5 (14 points) Pass the SQL

The Warriors is the dominant basketball team in the National Basketball Association (NBA). Legend states that their dominance is not due to having a team of four (now five) NBA all stars, but instead due to their focus on passing and unselfish ball handling. Is this really the case? This problem will walk through an analysis to study how long players hold the ball before they pass to their teammates. When a player holds the ball, we say that the player *possesses* the ball.

Consider the following database schema, where `Players` contains information about each basketball player, `Teams` contains information about each team, and `Possessions` contains information about each time a player held the ball and how long the player possessed the ball.

```
Players(
  pid int primary key,
  tid int not null
      references Teams,
  name text not null,
  age int not null

)
```

```
Teams(
  tid int primary key,
  name text not null,
  westcoast bool not null
)
```

```
Possessions(
  id int primary key,
  pid int not null
      references Players,
  -- when the possession started
  time timestamp not null,
  -- number of seconds the player
  -- held the ball
  held int not null
)
```

5.1 Sports Never Ages (2 Points)

Write the SQL query to find the average age across all players on west coast teams (westcoast is true). The output should be the average age.

```
SELECT avg(age)
FROM Players AS P JOIN Teams AS T ON P.tid = T.tid
WHERE westcoast
```

5.2 Hold It (4 Points)

Fill in the `CREATE TABLE TimeHeld(pid, name, held` statement by writing a query that computes the average possession time for each player. The average possession time is defined as the average number of seconds that a player possesses the ball before the ball is passed to a teammate. Return the `pid` and `name` of each player, and the player's average possession time.

```
CREATE TABLE TimeHeld(pid, name, held) AS
  SELECT pid, name, avg(held)
    FROM Possessions
GROUP BY pid, name
```

5.3 Teams That Pass (4 Points)

Fill in the `CREATE TABLE TeamPasses(passer, passee)` statement so it contains the `pids` of the players that passed the ball (`passer`) and the teammate that received the pass (`passee`).

This is defined as `Possessions` where the passer that is in possession of the ball is on the same team as the passee that receives the ball, *and* if the time of the passee's possession is equal to the time of the passer's possession plus the amount of time that the passer held onto the ball. You may assume that `TimeHeld` has been correctly created and may use it in your answer if it helps.

```
CREATE TABLE TeamPasses(passer, passee) AS
  SELECT P1.pid, P2.pid
    FROM Possessions AS P1, Players AS L1,
         Possessions AS P2, Players AS L2
   WHERE P1.pid = L1.pid AND
         P2.pid = L2.pid AND
         L1.tid = L2.tid AND
         P1.time + P1.held = P2.time
```

5.4 The Control Tower (6 Points)

Fill in the `CREATE TABLE Control(tid, pid)` statement to identify the players on each team that have passed the ball to every teammate at least once. Return the team `tid` and player `pid`. You may assume that `TimeHeld` and `TeamPasses` have been correctly created and may use it in your answer if it helps.

```
CREATE TABLE Control(tid, pid) AS
 SELECT tid, pid
   FROM Players AS L1
  WHERE NOT EXISTS (
   SELECT *
     FROM Players AS L2
    WHERE L1.tid = L2.tid AND
          L1.pid != L2.pid AND
          NOT EXISTS (
            SELECT *
              FROM TeamPasses AS TP
             WHERE TP.passer = L1.pid AND
                   TP.passee = L2.pid AND
       )
```

5.5 The Passing-est (4 Points)

Let the team possession time be the average of the average possession time over all players on the team. Write the SQL query that returns the name of the team with the lowest team possession time. You may assume that `TimeHeld`, `TeamPasses`, and `Control` have been correctly created and may use it in your answer if it helps.

```
SELECT T.name
  FROM TimeHeld AS TH, Players AS L, Teams T
 WHERE TH.pid = L.pid AND T.tid = L.tid
```

```
GROUP BY T.tid, T.name
ORDER BY avg(TH.held) ASC
   LIMIT 1
```