CMPS 180, Final Exam, Fall 2016, Shel Finkelstein

Student Name: _____

Student ID: _____

UCSC Email: _____

Final Points

Part	Max Points	Points
I	18	
II	24	
III	18	
IV	20	
V	21	
Total	101	

The exam is double-sided, so you have extra space to write answers. If you use that extra space, please be sure to write the number of the problem that you're solving next to your answer.

At the end of the exam, please show your UCSC id when you hand in your exam booklet.

Part I: (18 Points, 3 each)

For questions in Part I, use relations with the following schemas. Underlined attributes are keys.

Sailors(sid, sname, rating, age)
// For each sid, gives the sailor's name, rating and age

Boats(bid, bname, color)
// For each bid, give the boat's name and color

Reservations(sid, bid, day)
// For each reservation of a boat by a sailor, gives day sailor reserved boat

Some questions ask you to write SQL queries. Others ask you to write relational algebra expressions.

Question 1: Write a SQL statement that finds the sid and age for sailors who reserved at least one boat.

Answer 1: Here are some of the ways to answer this question.

a)
```
SELECT s.sid, s.age
  FROM Sailors s
  WHERE EXISTS ( SELECT *
                   FROM Reservations r
                   WHERE s.sid = r.sid );
```

b)
```
SELECT DISTINCT s.sid, s.age
  FROM Sailors s, Reservations r
  WHERE s.sid = r.sid;
```

c)
```
SELECT s.sid, s.age
  FROM Sailors s,
  WHERE ( SELECT COUNT(*)
            FROM Reservations r
            WHERE s.sid = r.sid ) > 0;
```

Question 2: Write a relational algebra expression that finds the sid and age for sailors who reserved at least one boat.

Answer 2:

$\Pi_{\text{sailors.sid, sailors.age}}(\sigma_{\text{sailors.sid = reservations.sid}}(\text{Sailors X Reservations}))$

Question 3: Write a SQL statement that finds the bid and color for boats that weren't reserved by any sailor.

Answer 3: Here's the simplest way to write this, similar to Question 1 Answer a):

```
SELECT b.bid, b.color
 FROM Boats b
 WHERE NOT EXISTS ( SELECT *
                    FROM Reservations r
                    WHERE b.bid = r.bid );
```

Some people did additional checks in the subquery to see if the sailor in the reservation wasn't null (r.sid IS NOT NULL). That wasn't required, but it was good.

Question 4: Write a relational algebra expression that finds the bid and color for boats that weren't reserved by any sailor.

Answer 4:

$\Pi_{bid,\ color}$ (Boats) - $\Pi_{Boats.bid,\ color}$ ($\sigma_{Boats.bid=Reservations.bid}$ (Boats X Reservations))

The part of Answer 4 after the "minus" is very similar to Answer 2.

Question 5: Explain what the following relational algebra expression computes, where ⋈ is Natural Join.

$$(\pi_{rating} ((\sigma_{color='red'} \text{Boats}) \bowtie (\sigma_{day='11/07/2016'} \text{Reservations}) \bowtie \text{Sailors}))$$

$$\cap \ (\pi_{rating} ((\sigma_{color='green'} \text{Boats}) \bowtie (\sigma_{day='11/07/2016'} \text{Reservations}) \bowtie \text{Sailors}))$$

Answer 5:

Find the sailor ratings where there's a sailor with that rating who reserved a red boat on 11/07/2016, and there's also a sailor with that rating who reserved a green boat on 11/07/2016.

[Note that for a rating to appear in the answer, there have to be sailors who reserved a red boat and a green boat on 11/07/2016 who have that rating. But they could be different sailors with that rating.]

Question 6: For each equality, answer YES if it is <u>always</u> true, and answer NO if it isn't always true.

6a): $\sigma_{C1}(\sigma_{C2}(R)) = \sigma_{C2}(\sigma_{C1}(R))$

where C1 and C2 are conditions on relation R's attributes.

Answer 6a): ___YES____

We discussed this in class; both are equivalent to $\sigma_{(C1\ AND\ C2)}(R)$

6b): $\pi_{A1}(\pi_{A2}(R)) = \pi_{A2}(\pi_{A1}(R))$

where A1 and A2 are sets of attributes of relation R.

Answer 6b): ____NO___

Suppose that A2 is (name, address) and A1 is just (name). Then the left-hand side makes sense, producing name. But the right-hand side doesn't make sense. If you do a projection where you only keep name, then you can't project name and address.

6c): $(R - S) - T = R - (S - T)$

where R, S and T are union-compatible relations.

Answer 6c): ____NO___

We drew the Venn diagram for this in class. As an example, suppose that S and T are equal. Then the right-hand side will always equal R, but the left-hand side won't always equal R because of the minus.

Part II: (24 Points, 4 each)

Question 7: A database has the relation Employees, with primary key emp_id, and other attributes giving name, department and salary of an employee.

Employees(emp_id, ename, edept, esalary)

A programmer writes a SQL query intending to find the name, salary and department for employees who make the most money in their departments.

```
SELECT e1.ename, e1.salary, e1.edept
FROM Employees e1
WHERE e1.salary > ALL
        ( SELECT e2.salary
          FROM Employees e2
          WHERE e1.edept = e2.edept );
```

7a): What error has the programmer made in writing this query? Fix the query so that it does what the programmer intended, making as few changes as possible.

Answer 7a):

Every employee is in the same department as herself, but no employee can make more than her own salary. The programmer should have written >= instead of >:

```
SELECT e1.ename, e1.salary, e1.edept
FROM Employees e1
WHERE e1.salary >= ALL
        ( SELECT e2.salary
          FROM Employees e2
          WHERE e1.edept = e2.edept );
```

7b): Suppose that there is only one department ('Marketing') in Employees, and the employees in that department are 'Smith', 'Jones' and 'Brown'. The salary of 'Smith' is 50000 and the salaries of 'Jones' and 'Brown" are NULL. What will be the output of the corrected 7a) query that the programmer intended to write?

Answer 7b):

The output will be empty. Even though Smith makes 50000, a comparison of his salary to a NULL salary yields truth value UNKNOWN. So Smith's salary is not greater or equal to the salary of all the employees in his Marketing department.

Question 8a): If S(A,B,C) is a relation where A's domain is (a1, a2, a3, a4) but A can't be NULL, and B's domain is (b1, b2, b3, b4) but B can be NULL, and C domain is (c1, c2) but c can be NULL, what is the maximum number of different tuples that can be in relation S?

Answer 8a: ____60_____ (4 * 5 * 3 = 60)

8b): Write a SQL statement that updates tuples of S in which B has the value NULL but C does not, setting the B value for those tuples to be b2.

Answer 8b:

```
UPDATE S
SET B = b2
WHERE B IS NULL
    AND C IS NOT NULL;
```

Question 9: Define the Transaction Isolation levels Read Committed, and Repeatable Read, explaining the difference between them.

Answer 9:

For Read Committed, only values written by transactions that committed are read; there are no Dirty Reads, where values written by transactions that have not yet committed are read.

For Repeatable Read, once again only values written by transaction that committed are read. However, with Repeatable Read, if you read a tuple's value multiple times during a transaction, you'll always read exactly the same value for that tuple.

With Read Committed, although you'll only read committed values for a tuple, you might read different values for that tuple during a transaction, e.g., because other transactions changing that tuple committed.

Question 10: Briefly explain two different advantages of using Stored Procedures versus other Application Programming approaches.

Answer 10: Here are the three principal advantages of Stored Procedures over other Application Programming approaches. You were only asked to provide two.

a) Avoid round trips: Performing computation inside the database is faster and cheaper than moving data back and forth between the database and the application.

b) Encapsulation. Once a Stored Procedure has been written, many programs can use that stored procedure. Re-use is often better than having different programmers writing and maintaining code themselves.

c) Security/Authorization. A user may be authorized to see the results of a Stored Procedure (e.g., a procedure that finds the average salary for people in a company), even though that user may not run a query that looks at the salaries of the individuals in the company.

Question 11: For the following table.

```
CREATE TABLE Beers (
        name   CHAR(20) PRIMARY KEY,
        manf   CHAR(20) );

CREATE TABLE Sells (
        bar    CHAR(20),
        beer   CHAR(20),
        price  REAL );
```

Rewrite the CREATE statement for Sells (don't do an ALTER) so that the beer attribute of Sells is a Foreign Key that correspond to name, the primary key of BEERS. Do the rewrite in 2 different ways for parts a) and b):

11a): First, rewrite the CREATE so that a Beers row can't be deleted if there is a bar that sells the beer.
11b) Second, rewrite the CREATE so that if a Beers row is deleted, all the Sells rows for bars that sell that beer also are deleted.

(Don't be concerned about handling of Updates of the name attribute of Beers.)

Answer 11a):

```
CREATE TABLE Sells (
        bar    CHAR(20),
        beer   CHAR(20) REFERENCES Beers(name),
        price  REAL );
```

Answer 11b):

```
CREATE TABLE Sells (
        bar    CHAR(20),
        beer   CHAR(20) REFERENCES Beers(name) ON DELETE CASCADE,
        price  REAL );
```

Question 12:

12a): Explain in words exactly what the regular expression (in bold) for person in the following address book DTD requires.

```
<!DOCTYPE addressbook [
  <!ELEMENT addressbook (person*)>
  <!ELEMENT person
     (name, address?, ( homephone | ( workphone, mobile )* ), email+)>
  <!ELEMENT name          (#PCDATA)>
  <!ELEMENT address       (#PCDATA)>
  <!ELEMENT homephone     (#PCDATA)>
  <!ELEMENT workphone     (#PCDATA)>
  <!ELEMENT mobile        (#PCDATA)>
  <!ELEMENT email         (#PCDATA)>
]>
```

Answer 12a):

According to this DTD, a person:

- must have a single name,
- can have 0 or 1 address,
- has either a single homephone, or else has any number of workphones and mobile phones which can appear in any order … but not both,
- and has 1 or more emails.

12b): Does the following data conform to that DTD? (YES or NO)

```
<person>
        <name> MacNiel, John </name>
        <addr> Rome, OH 98765 </addr>
        <homephone> 718 555 3089 </homephone>
        <workphone> (903) 555 9876 </workphone>
        <mobile> (321) 555 2543 </mobile>
        <email> macniel@ucsc.edu </email>
</person>
```

Answer 12b): ____NO_____

The data does not correspond to the DTD because the person has both a homephone, as well as one or more workphone/mobile numbers (actually, one of each). The DTD says that the person can have either a homephone or else any number of workphone/mobile numbers in any other … but not both.

Part III: (18 points, 3 each):

Answer the following questions with **YES/TRUE** or **NO/FALSE**.

Question 13: For a database with the following relations, with primary keys underlined:

Employees(emp_id, ename, edept, salary)
Departments(dept_id, dmanager, daddress)

Are the following two queries always equivalent?

```
SELECT ename, dmanager, SUM(salary),
FROM Employees, Departments
WHERE edept = dept_id
  AND dmanager<> 'Turing'
GROUP BY ename, dmanager;
```

```
SELECT ename, dmanager, SUM(salary),
FROM Employees, Departments
WHERE edept = dept_id
GROUP BY ename, dmanager
HAVING dmanager <> 'Turing';
```

Answer 13: ___YES_____

Question 14: When optimizing a query tree for execution, it is a good idea to push selection predicates and projections as far down in the tree as possible, to avoid extraneous rows and attributes. But you can't rid of an attribute if it will be needed later to evaluate a predicate, or as part of a query answer.

Answer 14: ____YES_____

Question 15: The 5 relational algebra operators product (x), projection (π), selection (σ), union ($\cup$) and difference (–) are independent of each other, meaning that none of those can be expressed using the other. However, it's possible to express join, natural join, intersection and division using the 5 relational algebra operators.

Answer 15: ____YES_____

Question 16: Is every binary relation R(A,B) in Boyce Codd Normal Form, no matter what functional dependencies R has?

Answer 16: ___YES_____

Question 17: A set F of Functional Dependencies (FDs) implies an FD F (written $F \models F$) if every instance r that satisfies FDs F must also satisfy the FD F.

Now suppose that we find an instance r that satisfies F and that also satisfies F. YES or NO, does that prove that $F \models F$?

Answer 17: ____NO_____

Question 18: For OLAP, with a star schema, there is a foreign key constraint between the Fact table and each of the Dimension tables. However, there can be combinations of Dimension table values for which there is no row in the Fact table.

Answer 18: ___YES_____

Part IV: (20 points, 5 each):

The questions in Part III are about the following tables, which should be familiar:

Stores (store_id, name, address, manager)
Customers (customer_id, name, address, email)
Products (product_id, name, category, manufacturer)
Sales (store_id, customer_id, product_id, purchase_date, quantity, unit_price, shipped)

Question 19: Write a SQL statement that creates the Products table with product_id as a primary key. Ensure that there can't be two products that have both the same manufacturer and name. The default manufacturer should be 'Apple'. Category can be NULL, but the other attributes can't be NULL. product_id is an integer, and name category and manufacturer are character strings of length 20.

Answer 19:

```
CREATE TABLE Products
 (  product_id INT PRIMARY KEY,
    name CHAR(20) NOT NULL,
    category CHAR(20),
    manufacturer CHAR(20) NOT NULL DEFAULT 'Apple',
    UNIQUE (name, manufacturer)
  );
```

Question 20:

Here is a Create Statement for the Sales Table.

```
CREATE TABLE Sales
( store_id INT,
  customer_id INT,
  product_id INT,
  purchase_date DATE,
  quantity INT,
  unit_price DECIMAL(6,2),
  shipped BOOLEAN,
  PRIMARY KEY(store_id, customer_id, product_id, purchase_date) );
```

Rewrite this CREATE statement (don't do an ALTER) so that store_id, customer_id and product_id are Foreign Keys that correspond to the keys of Stores, Customers and Products, respectively. The policies for referential integrity should be:

- Deleting a customer also deletes all Sales involving that customer.
- Products can't be deleted if there are Sales of that product.
- Deleting a store deletes all Sales that were made in that store.

Also, the total price for any sale (which is quantity multiplied by unit_price) must be less than 2000.

Answer 20:

```
CREATE TABLE Sales
  ( store_id int references Stores(store_id) ON DELETE CASCADE,
    customer_id int references Customers(customer_id) ON DELETE CASCADE,
    product_id int references Products(product_id),
    purchase_date date,
    quantity INT,
    unit_price DECIMAL(6,2),
    shipped BOOLEAN,
    PRIMARY KEY(store_id, customer_id, product_id, purchase_date),
    CHECK (quantity * unit_price < 2000)
  );
```

Question 21:

21a): Write a view BacklogProds that finds the product_id and product name for all products whose manufacturer is 'Kelloggs' where that product has at least one sale in Sales that has not been shipped. Each product should be listed only once, even if there are multiple unshipped sales of that product.

Answer 21a):

```
CREATE VIEW BacklogProds AS
    SELECT p.product_id, p.name
    FROM Products p
    WHERE manufacturer = 'Kelloggs'
        AND EXISTS ( SELECT * FROM Sales s
                        WHERE p.product_id = s.product_id
                        AND NOT shipped );
```

21b): Write a query that does the following: For each product that is in the BacklogProds view, give the product_id, product name, and the total quantity that **has** been shipped for that product.

Answer 21b):

```
SELECT b.product_id, b.name, SUM(quantity)
FROM BacklogProds b, Sales s
WHERE b.product_id = s.product_id
     AND shipped
GROUP BY b.product_id, b.name;
```

Question 22:

22a): Write a statement that creates an index on the attributes purchase_date and unit_price of the Sales table.

Answer 22a):

```
CREATE INDEX SalesDatePrice
ON Sales(purchase_date, unit_price);
```

22b): For which of these queries would the index you created in part a) work best?

Query 1:

```
SELECT *
FROM Sales
WHERE purchase_date = '11/29/2016'
   AND unit_price < 20;
```

Query 2:

```
SELECT *
FROM Sales
WHERE unit_price = 20
   AND purchase_date < '11/29/2016';
```

Answer 22b): ___Query 1_____

22c: Indexes help process queries faster, but they also have significant disadvantages. Provide two different disadvantages associated with having indexes.

Answer 22c): Here are several disadvantages of index; you only had to provide 2.

- When you have indexes, they have to be modified when values in underlying tables change (insert, update, delete), extra work that takes extra time.
- Accessing indexes uses up additional memory (DRAM, L3, L2, L1).
- Indexes take up additional space on disk (or other durable storage).

Part V: (21 points, 7 each):

Question 23: Suppose that you have a relation R(A,B,C), and that an instance of R has rows (1,2,3), (4,2,3) and (5,3,3).

23a) Put an X next to a listed functional dependency if you can determine that it does not hold for R? Otherwise, leave the line blank.

_____ A → B

__X___ BC → A

_____ B → C

23b): Are there any functional dependencies that you can determine do hold for R? If so, give an example. If not, explain why not.

Answer 23b):

There are two good answers to this question:

NO: Even if those are all the tuples in the instance, you can't tell whether a non-trivial Functional Dependency holds by looking at a single instance. A Functional Dependency is a rule that must hold across all instances, and even if that rule holds for one instance, it may not necessarily hold for other instances.

YES: Trivial Functional Dependencies must hold for any database. A trivial dependency is a Functional Dependency of the form X → A, where A is one of the attributes in X.

[This question should have begun: "Are there any non-trivial functional dependencies that …".]

Question 24:

24a): As part of design theory, we discussed anomalies, which database designers often try to avoid. Using an example, explain the deletion anomaly problem.

Answer 24a): Consider an Employee relation with the columns shown below, where eid is the primary key, but also where rank → salary_scale (rank functionally determines salary_scale).

eid	name	addr	rank	salary_scale
34-133	Jane	Elm St.	6	70-90
33-112	Hugh	Pine St.	3	30-40
26-002	Gary	Elm St.	4	35-50
51-994	Ann	South St.	4	35-50
45-990	Jim	Main St.	6	70-90
98-762	Paul	Walnut St.	4	35-50

The tuples with eid 33-112 (name is Hugh) tells us that people who have rank 3 must have salary_scale 30-40. But if that tuple was deleted, then the relation would no longer have information about the salary_scale associated with rank 3. That's the Deletion Anomaly discussed in the first of the two lectures on Design Theory.

24b): If you have a relation R and a set of functional dependencies F, what is a lossless-join decomposition for R with respect to F?

Answer 24b): Here's the precise definition from our Lecture slides, starting with the definition of decomposition. Answers could be more informal than this.

A **Decomposition of a relation R** is defined by sets of attributes $X_1, \ldots, X_k$ (which don't have to be disjoint) such that:

1. Each $X_i \subseteq attr(R)$
2. $X_1 \cup X_2 \cup \ldots \cup X_k = attr(R)$

For a decomposition, we will write $p_{Xi}(R)$ as R_i, with instance of R written as r and instances of R_i written as r_i.

Let R be a relation schema and F be a set of FDs over R. A decomposition of R into *k* schemas, with attribute sets $X_1, \ldots, X_k$, is a **Lossless Join Decomposition** *with respect to* F if for every instance r of R that satisfies F, we have:

$r = \pi_{X1}(r) \bowtie \ldots \bowtie \pi_{Xk}(r)$
$= r_1 \bowtie r_2 \bowtie \ldots \bowtie r_k$ where $\bowtie$ is Natural Join.

The following isn't the definition of Lossless Join Decomposition, but answers saying this received most of the credit for this problem. (Generalizing from 2 to k was even better, although we didn't do that in class.)

Let R be a relation and F be set of FDs that hold over R. A decomposition of R into relation schemas R_1 and R_2 is Lossless if and only if F^+ contains either:

1. $R_1 \cap R_2 \rightarrow R_1$, or
2. $R_1 \cap R_2 \rightarrow R_2$

That is, the intersection of the attributes of R_1 and R_2 is a superkey of either R_1 or R_2

24c): Give an example of a relation R with no functional dependencies and a decomposition of R that is <u>not</u> a lossless-join decomposition.

Answer 24c): There are many good answers to this question that give a decomposition with an example that is not lossless. Here's one of the examples that was discussed in class, where when you do the Natural Join of the relation instances formed from the original instance by the decomposition, the result has tuples in it that were not in the original instance.

Let R(A,B,C) be a relation schema with no functional dependencies. We can decompose R into R1(A,B) and R2(B,C), but the decomposition is not lossless, as we can see from this example with instance ry.

The instance on the far right should be labeled: $\pi_{A,B}(ry) \bowtie \pi_{B,C}(ry)$

Instance ry

A	B	C
a1	b1	c1
a2	b1	c2

$\pi_{A,B}(ry)$

A	B
a1	b1
a2	b1

$\pi_{B,C}(ry)$

B	C
b1	c1
b1	c2

$\pi_{A,B}(ry)$ 1 $\pi_{B,C}(ry)$

A	B	C
a1	b1	c1
a1	b1	c2
a2	b1	c1
a2	b1	c2

Lossy!

The tuples (a1, b1, c2) and (a2, b1, c2) were not in the original instance ry, so the decomposition is lossy.

Question 25: Suppose that you have a relation R(A,B,C,D,E), with the following functional dependencies: A → B, BC → E, and ED → A.

25a): Is R in BCNF? Explain your answer.

Answer 25a):

No, R is not in BCNF. The FD A → B is not trivial, and A^+ is AB, so the left-hand side of A → B is not a superkey. That's enough to show that R is not in BCNF. But you could also prove this by looking at $(BC)^+$, which is BCE, or by looking at $(ED)^+$, which is ABDE.

None of the left-hand sides of the FDs are superkeys, and none of them is trivial. Hence any one of these FDs suffices to show that R is not in BCNF; we didn't have to look at all three.

25b): Is R in 3NF? Explain your answer.

Yes, R is in 3NF.

First, note that each of the following is a key for R: ACD, BCD and CDE. Why is each a key? Because the closure for each of them is ABCDE, which makes them superkeys, and we see from the following that no subset of any of them is a superkey, so they are keys for R.

$(AC)^+$ = ABCE
$(AD)^+$ = ABD
$(CD)^+$ = CD
$(BC)^+$ = BCE
$(BD)^+$ = BD
$(CE)^+$ = CE
$(DE)^+$ = ABDE

Although none of the 3 FDs for R is trivial, and none of their left-hand sides are superkeys (as we saw in 25a), the right-hand sides of all the FDs are parts of keys for R.

For A → B, B is part of the key BCD.
For BC → E, E is part of the key CDE.
For ED → A, A is part of the key ACD.

25c): For each of the following, indicate whether it's a candidate key for R by writing YES or NO in the blank.

Answer 25c):

___NO_____	ABC	Not a key because $(ABC)^+ = ABCE$.
___NO_____	ACDE	Not a key because a subset, ACD, is a superkey.
___YES____	BCD	BCD is a key, as we showed in 25b).