UCLA
Computer Science Department
Fall 2008

Instructor: J. Cho

Student Name and ID: _____

CS143 Midterm: Closed Book, 90 minutes

(IMPORTANT PLEASE READ **):**

- There are 3 problems on 10 pages for a total of 85 points to be completed in 90 minutes. *You should look through the entire exam before getting started, in order to plan your strategy.*
- *Simplicity and clarity of your solutions will count.* You may get as few as 0 point for a problem if your solution is far more complicated than necessary, or if we cannot understand your solution.
- If you need to make any assumption to solve a question, please write down your assumptions.
- To get partial credits, you may want to write down how you arrived at your answer step by step.
- You may use one-page double-sided cheat-sheet during exam. You are also allowed to use a calculator.
- Please, write your answers neatly. Attach extra pages as needed. Write your name and ID on the extra pages.

Problem	**Score**	
1	25	
2	30	
3	30	
Total	85	

Problem 1 (Queries): 25 points

1. The table `Sensor(date, time, temp, humidity)` stores the temperature and the humidity values every few hours in the Los Angeles airport. Write a relational algebra query that returns the highest temperature value of each day in the table. That is, your query should return (date, highest temperature of the day) pairs for each day in the table. Make your query simple and clear. Your answer will be graded based on simplicity as well as correctness. (10 points)

   ANSWER:
   $\pi_{date,temp}(Sensor)$
   $- \pi_{R1.date,R1.temp}(\sigma_{R1.date=R2.date \wedge R1.temp<R2.temp}(\rho_{R1}(Sensor) \times \rho_{R2}(Sensor)))$

2. Consider the following database tables on car ownership and car accidents:

   `Person(`<u>`SSN`</u>`, name, address)`
   `Car(`<u>`license`</u>`, year, model)`
   `Accident(`<u>`license`</u>`, `<u>`accident-date`</u>`, driver, damage-amount)`
   `Owns(`<u>`SSN`</u>`, `<u>`license`</u>`)`

   Note that the driver involved in a car accident may not always be the owner of the car. We assume that a car cannot get involved in more than one accident at a certain date.

   For the following English statements, choose all queries in SQL or relational algebra that return the proper answer. Note that each question may have zero, one or more correct answers. List only the ones that are correct. Write 'None' if there is no correct answer.

   (a) Find the SSN of all persons who have had all of their cars involved in an accident. (5 points)

   i.
   ```
   (SELECT SSN FROM Owns, Accident
   WHERE Owns.license = Accident.license)
   EXCEPT
   (SELECT SSN FROM Owns
   WHERE license NOT IN (SELECT license FROM Accident))
   ```

   ii. $\pi_{SSN,license}(Owns \bowtie Accident)/\pi_{license}(Accident)$

   Correct queries:
   ANSWER:
   `(i) only. (ii) returns the owner(s) who own every car that has been in an accident.`

(b) Who is the driver who participated in the most costly accident? Return the driver and the amount of damage. (5 points)

i.
```
SELECT driver, damage-amount FROM Accident
WHERE damage-amount IN (SELECT MAX(damage-amount) FROM Accident)
```

ii.
```
SELECT driver, damage-amount FROM Accident
WHERE damage-amount = MAX(damage-amount)
```

iii.
```
(SELECT driver, damage-amount FROM Accident)
EXCEPT
(SELECT A1.driver, A1.damage-amount
FROM Accident A1, Accident A2
WHERE A1.damage-amount < A2.damage-amount)
```

Correct queries:
ANSWER:
(i), (iii) only. (ii) generates error because of MAX() in WHERE clause.

(c) Find the license number of all cars that have been involved in more than one accident (DO NOT RETURN DUPLICATES) (5 points)

i.
```
SELECT license FROM Accident
GROUP BY license
HAVING COUNT(accident-date) > 1
```

ii.
```
SELECT A1.license FROM Accident A1, Accident A2
WHERE A1.license = A2.license AND A1.accident-date <> A2.accident-date
```

iii.
```
SELECT DISTINCT A1.license FROM Accident A1
WHERE A1.license IN (SELECT A2.license
                     FROM Accident A2
                     WHERE A1.accident-date <> A2.accident-date)
```

Correct queries:
ANSWER:
(i), (iii) only. (ii) may return duplicate license numbers.

Problem 2 (Integrity, View, and Authorization): 30 points

For all questions in this problem, we refer to three tables created by the following statements:

```
CREATE TABLE Vendor(
    vid  INT,
    name CHAR(20),
    PRIMARY KEY(vid));

CREATE TABLE Product(
    pid  INT,
    name CHAR(20),
    vid  INT,
    PRIMARY KEY(pid),
    FOREIGN KEY(vid) REFERENCES Vendor(vid)
        ON DELETE CASCADE
        ON UPDATE CASCADE);

CREATE TABLE Sales(
    pid        INT,
    price      INT,
    quantity   INT,
    sales-time DATETIME
    PRIMARY KEY(pid, sales-time),
    FOREIGN KEY(pid) REFERENCES Product(pid));
```

As you can imagine from their names, Vendor table contains the vendor information, Product table contains the product information and its vendor, and the tuples in the Sales table correspond to the actual sales of a product made at a particular date and time.

Assume that the three tables contain the following tuples:

Vendor

vid	name
1	Motorola
2	LG
3	Samsung

Product

pid	name	vid
1	Razr	1
2	Vue	2
3	Blackjack	3
4	Slide	3

Sales

pid	price	quantity	sales-time
1	100	1	2008-10-25 10:00:00
3	150	1	2008-10-26 11:00:00

1. Write a *single* SQL constraint or SQL trigger statement that ensures that every vendor in the Vendor table should always have at least one associated product in the Product table. If your answer is a PRIMARY KEY, a FOREIGN KEY, or a CHECK constraint, just write the constraint definition part and the name of the table for which the constraint is defined. If your answer is either an assertion or a trigger, write its full definition. (5 points)

   ANSWER:
   ```
   CREATE ASSERTION VERIFY-VENDOR CHECK( NOT EXISTS(SELECT * FROM Vendor WHERE
   vid NOT IN (SELECT vid FROM Products)))
   ```

2. Write a *single* SQL constraint or SQL trigger statement that ensures that the sales-time of any newly added Sales tuple should not be before that of any existing Sales tuple with the same pid. That is, any new sale of a product cannot precede a prior sale of the product. Note that the aggregate function MAX() returns NULL if it is applied to an empty set of tuples. If your answer is a PRIMARY KEY, a FOREIGN KEY, or a CHECK constraint, just write the constraint definition part and the name of the table for which the constraint is defined. If your answer is either an assertion or a trigger, write its full definition. (5 points)

   ANSWER:
   ```
   CREATE TRIGGER NO-EARLIER-SALE AFTER INSERT ON Sales REFERENCING NEW ROW
   AS nr FOR EACH ROW WHEN (nr.sales-time < (SELECT MAX(sales-time) FROM Sales
   WHERE pid = nr.pid) DELETE FROM Sales WHERE pid = nr.pid AND sales-time
   = nr.sales-time
   ```

3. For each of the following statement, briefly state what will happen when the statement is issued. If some tuples are deleted or updated due to the statement, clearly indicate the exact list of tuples that are affected (and their new values if they are updated). If the statement is rejected, briefly explain why. Do *NOT* assume the constraints/triggers that you defined for questions 1 and 2. For each statement, assume the table instances given at the beginning of this problem. Consider each statement separately from others.

   (a) `DELETE FROM Vendor WHERE vid = 3;` (2.5 points)

   ANSWER:
   The statement is rejected because it will try to delete (3, 'Blackjack', 3) from Product table, but this will violate the FOREIGN KEY constraint in Sales table.

   (b) `DELETE FROM Vendor WHERE name = 'LG';` (2.5 points)

   ANSWER:
   (2, 'LG') is deleted from the Vendor table. (2, 'Vue', 2) is also deleted from the Product table

   (c) `UPDATE Vendor SET vid = 4 WHERE vid = 1;` (2.5 points)

   ANSWER:
   (1, 'Motorola') is updated to (4, 'Motorola') in Vendor. (1, 'Razr', 1) is also updated to (1, 'Razr', 4) in Product.

   (d) `UPDATE Product SET pid = 5, vid = 3 WHERE pid = 2;` (2.5 points)

   ANSWER:
   (2, 'Vue', 2) is updated to (5, 'Vue', 3)

4. Assume that u_1 is the owner of all three tables, Vendor, Product, and Sales. Now u_1 wants to allow another user u_2 to create the ProductSales view by executing the following statement:

```
CREATE VIEW ProductSales AS
    SELECT pid, SUM(quantity*price) total_sales
    FROM Sales
    GROUP BY pid
```

   Assume that u_1 wants to allow u_2 to give out the SELECT privilege for the ProductSales view to any other user that u_2 chooses. Write a single GRANT statement that provides the *minimum* set of privileges that have to be given to u_2 to allow this. That is, u_2 should be able to run the above statement and allow other users to SELECT on ProductSales view. (5 points)

   ANSWER:

```
GRANT SELECT ON Sales TO 'u2' WITH GRANT OPTION;
```

5. For this question, assume that u_2 has already created ProductSales view after obtaining the necessary privileges from u_1. In addition, another user, u_3, has also created the following TotalSales view after u_2 granted the necessary privileges to u_3:

```
CREATE VIEW TotalSales AS
    SELECT SUM(total_sales) revenue
    FROM ProductSales
```

   Now u_1 wants to control the access to the Sales table more tightly and decides to run the following REVOKE command:

```
REVOKE GRANT OPTION FOR SELECT ON Sales FROM 'u2' CASCADE;
```

   Briefly explain what will happen to all views, ProductSales and TotalSales, after the above command is executed. If the command generates an error, briefly explain why. (5 points)

   ANSWER:

```
ProductSales stays.  TotalSales gets dropped.
```

Problem 3 (Disk, File, and Index): 30 points

Consider the Sales table created with the following statement:

```
CREATE TABLE Sales(
    sales-id     INT,
    product-id   INT,
    product-name CHAR(80),
    price        INT,
    quantity     INT,
    sales-time   TIME,
    PRIMARY KEY(sales-id))
```

We store one million tuples in the Sales table in the following disk:

- 1 platter (2 surfaces)
- 2,000 cylinders
- 100 sectors per track
- 2048 bytes per sector
- 6,000 RPM rotational speed
- 10ms average seek time

One disk sector corresponds to one disk block. INT and TIME datatypes require 4 bytes each and CHAR(n) datatype requires n bytes. Each Sales tuple is stored as a fixed-length record and the tuples are *NOT SPANNED*. We do not store any additional header in each block, so all block space is available to store tuples.

1. What is the (burst) transfer rate of our disk? (5 points)
   ANSWER:
   `(2048 bytes/sector) * (100 sector/track) * (6,000 track/min) * (1 min/60 sec) = 20M bytes/sec.`

2. What is the minimum number of blocks that we need to store the Sales table? (5 points)
   ANSWER:
   $\lfloor \frac{2048 bytes/block}{100 bytes/tuple} \rfloor = 20 tuples/block.$ $\lceil \frac{1{,}000{,}000 tuple/table}{20 tuple/block} \rceil = 50{,}000 block/table.$ `We need at least 50,000 blocks.`

3. We need to retrieve the sales tuple with '`sales-id = 307`' by scanning the table. The tuples are *NOT* sequenced by sales-id. Assume that the Sales table is stored in 100,000 disk blocks. The disk blocks for the table are allocated sequentially. Assume that the sustained transfer rate of our disk is 20MB/sec. Note that the sustained transfer rate is the average rate at which we can sequentially read a large volume of data that may span over multiple tracks. (These numbers may or may not be the same as your previous answers.) For other parameters, assume the numbers we gave in the beginning. How long does it take to retrieve the tuple? Compute the *expected (or average)* time. (5 points)

   ANSWER:
   `It takes 10ms for the initial seek and 5ms for the rotational delay. Once we start reading the table block, we will have to read half of the table on average to find the tuple. Half of the table (50,000 block) is 100MB, so it takes 100/20 = 5 sec to read the half. In total, it takes 5.015 sec (5 sec) on average.`

4. Now we create a B+tree on the sales-id attribute (integer of 4 bytes) of the Sales table. The leaf nodes of the tree contain only the search keys (sales-id) and the pointers to the tuples. They do not contain actual tuples. Each pointer to a tuple is of size 8 bytes. The B+tree is dense. That is, the leaf nodes of the B+tree have pointers to all tuples in the table. Each B+tree node is stored in one disk block. As we learned in the class, a node in a B+tree has n pointers and $(n-1)$ search keys. What is the value of n (or the maximum number of pointers that we can store in a node) for this tree? (5 points)

   ANSWER:
   $8n + 4(n-1) \leq 2048.$ $n \leq 2052/12 = 171.$

5. For this question, assume that the n value for the B+tree is 200. (This number may or may not be the same as the previous answer.) Assuming that each node in the B+tree contains the *minimum* possible number of pointers, how many levels of nodes does the B+tree have? How many nodes are needed for each level of the B+tree? Again, the B+tree is dense. (5 points)

   ANSWER:
   Height: 3, Level 1: 10,000 nodes, Level 2: 100 nodes, Level 3: 1 node.

   When $n = 200$, leaf nodes should have at least 100 pointers to tuples (101 pointers including the next node pointer). Non-leaf nodes should have at least 100 pointers to their children. Since the B+tree is dense, we have to store 1,000,000 index entries. Therefore, we need at least $\lfloor 1,000,000/100 \rfloor = 10,000$ nodes at leaf level. At one level above, we need $\lfloor 10,000/100 \rfloor = 100$ nodes. At one level above, we have a single root node.

6. We now retrieve the Sales tuple with '`sales-id = 307`' using the constructed B+tree. That is, we first look up the B+tree, get the pointer to the tuple, and then read the actual tuple from the disk. How long does it take to retrieve the tuple? Compute the expected time (or average). For this question, assume that the B+tree has four levels of nodes and the burst transfer rate of the disk is 20MB/sec. (These numbers may or may not be the same as your previous answers.) For other parameters, assume the numbers that we listed in the beginning. (5 points)

   ANSWER:
   To follow the tree, we have to read 4 disk blocks and to access the tuple, we have to read 1 disk block. In total, we have to do 5 random disk block read, which takes $5 \times (10ms + 5ms + 0.1ms) = 75.5ms$.