CS585 Midterm exam

2016-10-14

Duration: 1 hour.

Last Name: ________________________________

First Name: ________________________________

Student ID: ________________________________

Email: ________________________________

Hello! There are 9 questions below (8 plus a bonus), one per page. Please read each question carefully before answering. You don't to elaborate on anything, so you won't need additional sheets.

The exam is CLOSED book/notes/devices/neighbors(!) but 'open mind' :)

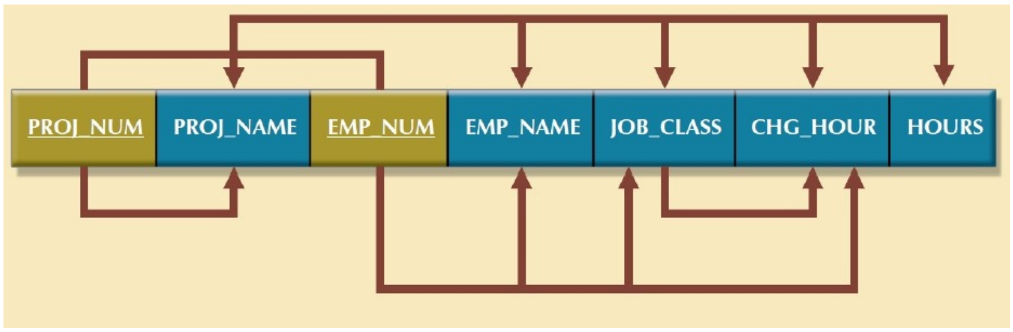
If you are observed cheating, or later discovered to have cheated in any manner, you will get a 0 on the test and also be reported to SJACS - so please don't!

When we announce that the time is up, you NEED to stop writing immediately, and turn in what you have; if you continue working on the exam, we will not grade it (ie. you will get a 0).

Have fun, and good luck! Hope you do well.

Saty

Q1 (1+3=4 points). A 1NF table, such as the one shown below (we covered this in class on great detail), is analyzed to detect problems (related to unwanted dependencies), which are then systematically eliminated using a diagram such as the shown below (the table is converted to 2NF, then 3NF), in a process called 'normalization'.



a. What is the diagram (shown above) called?

b. How does the diagram aid in normalization? Explain briefly, using the above diagram (you can mark it up if you want).

Q2 (4 points). Two-phase locking (2PL) is a popular concurrency control scheme for managing transactions. Unfortunately, however, it cannot entirely prevent deadlocks from occurring. Using two transactions T1 and T2, show (using a sequence of events you come up with) how a deadlock can occur between them.

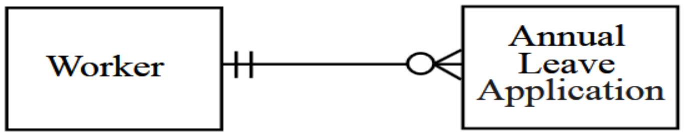
Q3 (4 points). In 1970, Ed Codd at IBM published a landmark paper that totally redefined the state of art of databases at the time. What was Codd's breakthrough? Explain, using your own words and diagrams. You need to have at least 4 sentences in your answer.

Q4 (2+2=4 points). For a while now, NASA has been conceptualizing a network called the Interplanetary Internet, which could come in handy 'someday' when we colonize Mars [when pigs fly out of our butts :)]. If that were to come to fruition, Eric Brewer's 'CAP theorem' would be highly relevant and applicable to such a distributed system of nodes. As per the CAP theorem, 'you can't always get what you want' (at least not C,A,P all at once, all equally guaranteed).

In an Interplanetary Internet, how would you rank C,A,P in terms of concerns? In other words, which would we worry about most, and relatively which, the least? You need to state why (justify your ordering).

Where might nodes be located, for an Interplanetary Internet? And, what disaster scenarios can you envision (that affect the network)?

Q5 (4 points). Consider the following relation:



As per the above, any employee in a certain company could file annual leave applications (submit vacation requests). The management institutes a new policy, stating that only full-time employees (not part-timers or contractors) can do so. How would you reflect the policy change via an updated diagram?

Q6 (2 points). Below is a table that tracks projects. There are several steps (phases) in each project (workorder), and for each step, its completion status is maintained (C means completed, A means awaiting completion).

```
CREATE TABLE Projects
(workorder_id CHAR(5) NOT NULL,
step_nbr INTEGER NOT NULL CHECK (step_nbr BETWEEN 0 AND 1000),
step_status CHAR(1) NOT NULL
CHECK (step_status IN ('C', 'A')),
PRIMARY KEY (workorder_id, step_nbr));
```

Here is some sample data conforming to the definition above:

```
Projects
workorder_id step_nbr step_status
=================================
'O100'          0         'C'
'O100'          1         'A'
'O100'          2         'A'
'O200'          0         'A'
'O200'          1         'A'
'O300'          0         'C'
'O300'          1         'C'
```

Given the above, what does the following query do?

```
SELECT workorder_id
FROM Projects AS P1
WHERE step_nbr = 0
AND step_status = 'C'
AND 'W' = ALL (SELECT step_status
FROM Projects AS P2
WHERE step_nbr <> 0
AND P1.workorder_id = P2.workorder_id);
```

Q7 (4 points). Here are a pair of tables - a PRODUCTS table that lists products a company sells, and SALES, which records sales of the products (each unit of a product that is sold, gets a separate row in SALES):

```
PRODUCTS(PRODUCT_ID, PRODUCT_NAME);
SALES(SALE_ID, YEAR, PRODUCT_ID, PRICE);
```

Consider the following three queries, we're calling them Q1, Q2, Q3. In Q2, fyi, 'SELECT 1' returns a 1, which we can ignore (it is not essential to our query).

```
SELECT S.PRODUCT_ID,SUM(PRICE)
FROM SALES S
 JOIN
 PRODUCTS P
 ON (S.PRODUCT_ID = P.PRODUCT_ID)
GROUP BY S.PRODUCT_ID;

SELECT S.PRODUCT_ID,SUM(PRICE)
FROM SALES S
WHERE EXISTS
(
 SELECT  1
 FROM PRODUCTS P
 WHERE P.PRODUCT_ID = S.PRODUCT_ID
)
GROUP BY S.PRODUCT_ID;

SELECT S.PRODUCT_ID,SUM(PRICE)
FROM SALES S
WHERE PRODUCT_ID IN
(
 SELECT  PRODUCT_ID
 FROM PRODUCTS P
)
GROUP BY S.PRODUCT_ID;
```

Circle the correct choice below:

```
a. Q1, Q2, Q3 are all different (they produce different results)
b. Q1, Q2, Q3 are all identical
c. Q1 and Q2 are identical
d. Q1 and Q3 are identical
e. Q2 and Q3 are identical
```

Q8 (4 points). The following is a table that tracks sales made by salespeople across several districts (eg. at a car dealership).

```
CREATE TABLE SalesData
(district_nbr INTEGER NOT NULL,
sales_person CHAR(10) NOT NULL,
sales_id INTEGER NOT NULL,
sales_amt DECIMAL(5,2) NOT NULL);
```

Here is some conforming data:

```
SalesData
district_nbr sales_person sales_id sales_amt
==========================================
1          'Curly'        5          3.00
1          'Harpo'        11         4.00
1          'Larry'        1          50.00
1          'Larry'        2          50.00
1           'Larry'       3          50.00
1          'Moe'          4          5.00
2          'Dick'         8          5.00
2          'Fred'         7          5.00
2          'Harry'        6          5.00
2          'Tom'          7          5.00
3          'Irving'       10         5.00
3          'Melvin'       9          7.00
4          'Jenny'        15         20.00
4          'Jessie'       16         10.00
4          'Mary'         12         50.00
4          'Oprah'        14         30.00
4          'Sally'        13         40.00
```

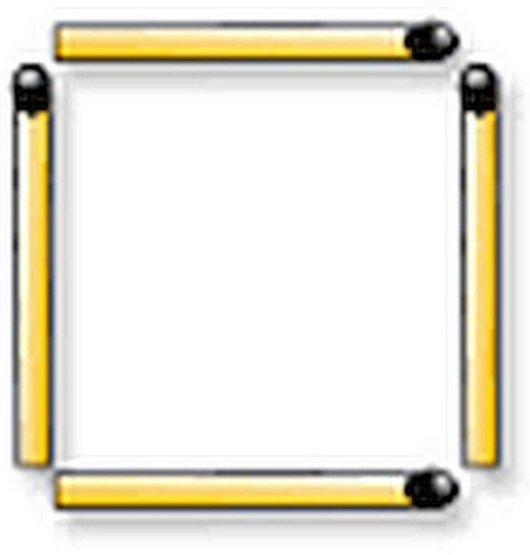
Here are a pair of queries against the table shown earlier.

```
SELECT *
FROM SalesData AS S0
WHERE sales_amt IN (SELECT S1.sales_amt
FROM SalesData AS S1
WHERE S0.district_nbr = S1.district_nbr
AND S0.sales_amt <= S1.sales_amt
HAVING COUNT(*) <= 3)
ORDER BY S0.district_nbr, S0.sales_person, S0.sales_id,
S0.sales_amt;
```

```
SELECT DISTINCT district_nbr, sales_person
FROM SalesData AS S0
WHERE sales_amt <= (SELECT MAX(S1.sales_amt)
FROM SalesData AS S1
WHERE S0.district_nbr = S1.district_nbr
AND S0.sales_amt <= S1.sales_amt
HAVING COUNT(DISTINCT S0.sales_amt) <= 3);
```

Question: how are the two queries similar, and how are they different? In other words, what does each do, which makes them be alike and also distinct?

Bonus question (1 point).



How would you add 4 matches to the above square (using matches identical in size to the ones above), that result in four triangles and two squares? No need to bend/break.. any match.