Practice Questions for Final Exam
CMPS 182, Summer 2016, UCSC

These are sample questions for the post-midterm material (but remember the final exam is cumulative.)

Question 1: What is an advantage of using Stored Procedures (such as Persistent Stored Modules, or PSMs)?

Answer 1: Some examples of good answers:

Processing of a stored procedure can be done in the database, performing operations that may be hard or impossible to do in a single SQL statement, without having to do multiple calls to the database.

By filtering data using a stored procedure, less data has to be moved from the database to the client application.

A client may be authorized to execute a stored procedure and see the result, without being authorized to see the data that the stored procedure uses (or modifies).

Question 2: What is an advantage of XML over relational models?

Answer 2: Some examples of good answers:

XML can represent semistructured or unstructured data, where you don't know (or don't fully know) what the structure/schema of the data is before you see the data. It is sometimes described as "data first", whereas relational is described as "schema first".

XML also allows data that isn't in First Normal Form (that is, with non-primitive types), so a value may be a complex type (such as a set, bag, sequence, array, ...) which relational doesn't permit.

XML is similar to hierarchical data models, encouraging access and storage of data based on nesting of data within XML documents, which can be efficient for storage and access.

Question 3: Assume that relation R(A,B,C,D) includes the row (1,2,3,4), and that R has Functional Dependencies {A,B} → {C} and {C} → {D}. (Note: by convention, these could also be written without the curly braces.) Mark each row below with a check if it might also be in R, and mark each row below with an X if it cannot also be in R.

Answer 3:

__X___ (1,2,4,7) __√__ (1,3,8,9) __X___ (1,2,3,5) ___√__ (2,2,4,6)

Question 4: Is the following relation with the specified Functional Dependencies in **Third Normal Form**? The following are both keys: {zip, street} and {city, street}.

(Important caveat: to be complete and certain, one needs to consider *all* functional dependencies, not just those that are given in the problem statement. There may well be other FDs that are implied by the given ones. However, to avoid doing too much "busy work", for this problem we assume that it is sufficient to test the given FDs.)

R(city, street, zip)
{city, street} → {zip}
{zip} → {city}

Answer 4: **YES**

Question 5: Is the same relation with the specified Functional Dependencies in **Boyce Codd Normal Form**? (Same relation, keys, and FDs as in question 4. Also, same caveat applies.)

Answer 5: **NO**. The FDs aren't trivial, and although (city,street) is a superkey, {zip} is not a superkey.

Question 6: Here are declarations of two relations R and S:

```
CREATE TABLE S(
    c INT PRIMARY KEY,
    d INT
);
CREATE TABLE R(
    a INT PRIMARY KEY,
    b INT,
    CHECK(b IN (SELECT c FROM S))
);
```

R(a,b) currently contains the four tuples (0,4), (1,5), (2,4), and (3,5).
S(c,d) currently contains the four tuples (2,10), (3,11), (4,12), and (5,13).

For each of the following modifications, circle YES or NO to indicate whether or not the modification is allowable. (In other words, YES it *would* be allowed, or NO it *would not* be allowed.) Note: Each of these 4 modifications is a separate problem, starting with the 8 tuples listed above. That is, this does **not** represent a sequence of modifications.

[✓ YES] NO Inserting (7,3) into R.
YES [✓ NO] Inserting (3,1) into S.
YES [✓ NO] Updating (3,5) in R to be (3,1).
[✓ YES] NO Deleting (5,13) from S.

Note: This question was based on one of our Gradiance questions. Reminder: if you revisit the Gradiance questions after their due dates, the system shows you a more complete explanation.

Question 7: Relation R has schema:

```
CREATE TABLE R (
    a INT PRIMARY KEY,
    b INT DEFAULT 0,
    c INT NOT NULL
);
```

R is currently empty.

For each of the following insertions, circle YES or NO to indicate whether or not the modification is allowable. Note: Each of these 4 modifications is a separate problem, starting with R empty. That is, this does **not** represent a sequence of modifications.

YES NO INSERT INTO R VALUES(1,NULL,2);
YES NO INSERT INTO R(c,a,b) VALUES(3,4,5);
YES NO INSERT INTO R(b,a) VALUES(5,6);
YES NO INSERT INTO R(b,c) VALUES(3,4);

Question 8a:

```
CREATE TABLE Employee (
    empNum INT PRIMARY KEY,
    name CHAR(30) NOT NULL,
    deptNum INT,
    salary INT
 );
```

How would you change the CREATE statement above if we wanted to enforce Referential Integrity by making deptNum in the Employee table be a foreign key referring to a deptNum attribute in a Department table?

Answer 8a:

```
CREATE TABLE Employee (
    empNum INT PRIMARY KEY,
    name CHAR(30) NOT NULL,
    deptNum INT REFERENCES Department(deptNum),
    salary INT
 );
```

[It's also possible to do this by adding a FOREIGN KEY specification at the end of the CREATE statement, e.g., FOREIGN KEY (deptNum) REFERENCES Department(deptNum).]

Question 8b: Now assume that the following table declaration is used to create the Department table mentioned in Question 8a:

```
CREATE TABLE Department (
    deptNum INT,
    deptName CHAR(30) NOT NULL
);
```

What change to this CREATE statement is needed to make your foreign key, in your answer to Question 8a, legal?

Answer 8b:

```
CREATE TABLE Department (
    deptNum INT PRIMARY KEY,
    deptName CHAR(30) NOT NULL
);
```

Question 8c: Describe two (out of the three) different actions that might be taken if a department that had employees in it was deleted from the Department table.

The three actions that may be taken if a department that had employees in it were deleted from the Department table are:

- Default action: Reject the deletion
- Cascade: Delete employees in the department (matching deptNum) when the department is deleted.
- Set NULL: For all employees who were in the deleted department (matching deptNum), set deptNum to NULL.

Question 9: Do Textbook Exercise 7.1.1, all parts.

Answer for 7.1.1

a)

```
CREATE TABLE Movies (
title           CHAR(100),
year            INT,
length          INT,
genre           CHAR(10),
studioName      CHAR(30),
producerC#      INT,
PRIMARY KEY (title, year),
FOREIGN KEY (producerC#)  REFERENCES MovieExec(cert#)
);
```

or

```
CREATE TABLE Movies (
title           CHAR(100),
year            INT,
length          INT,
genre           CHAR(10),
studioName      CHAR(30),
producerC#      INT     REFERENCES MovieExec(cert#),
PRIMARY KEY (title, year)
);
```

b)

```
CREATE TABLE Movies (
title           CHAR(100),
year            INT,
length          INT,
genre           CHAR(10),
studioName      CHAR(30),
producerC#      INT     REFERENCES MovieExec(cert#)
ON DELETE SET NULL
ON UPDATE SET NULL,
PRIMARY KEY (title, year)
);
```

c)

```
CREATE TABLE Movies (
title           CHAR(100),
year            INT,
length          INT,
genre           CHAR(10),
studioName      CHAR(30),
producerC#      INT     REFERENCES MovieExec(cert#)
ON DELETE CASCADE
ON UPDATE CASCADE,
PRIMARY KEY (title, year)
);
```

d)

```
CREATE TABLE StarsIn (
movieTitle      CHAR(100)   REFERENCES  Movie(title),
movieYear       INT,
starName        CHAR(30),
PRIMARY KEY (movieTItle, movieYear, starName)
);
```

e)

```
CREATE TABLE StarsIn (
movieTitle      CHAR(100)   REFERENCES  Movie(title)
        ON DELETE CASCADE,
movieYear       INT,
starName        CHAR(30),
PRIMARY KEY (movieTItle, movieYear, starName)
);
```

Question 10: Do Textbook Exercise 7.2.1.

Answer for 7.2.1

```
a)
year            INT     CHECK (year >= 1915)

b)
length          INT     CHECK (length >= 60 AND length <= 250)

c)
studioName      CHAR(30)
     CHECK (studioName IN (‘Disney’, Fox’, ‘MGM’, ‘Paramount’) )
```

Question 11: Do Textbook Exercise 7.2.4.

Answer for 7.2.4

```
a)
        CHECK (speed >= 2.0 OR price <= 600)

b)
        CHECK (screen >= 15 OR hd >= 40 OR price <= 1000)
```

Question 12: Do Textbook Exercise 7.5.2.

Answer for 7.5.2

```
a)
CREATE TRIGGER LowPricePCTrigger
AFTER UPDATE OF price ON PC
REFERENCING
        OLD ROW AS OldRow,
        OLD TABLE AS OldStuff,
        NEW ROW AS NewRow,
        NEW TABLE AS NewStuff
FOR EACH ROW
WHEN (NewRow.price < ALL
        (SELECT PC.price FROM PC
         WHERE PC.speed = NewRow.speed))
BEGIN
        DELETE FROM PC
        WHERE (model, speed, ram, hd, price) IN NewStuff;
        INSERT INTO PC
                (SELECT * FROM OldStuff);
END;
```

```
b)
CREATE TRIGGER NewPrinterTrigger
AFTER INSERT ON Printer
REFERENCING
        NEW ROW AS NewRow,
        NEW TABLE AS NewStuff
FOR EACH ROW
WHEN (NOT EXISTS (SELECT * FROM Product
                  WHERE Product.model = NewRow.model))
DELETE FROM Printer
        WHERE (model, color, type, price) IN NewStuff;
```

Note: These complete trigger problems are for practice. Any trigger problem on the exam will be smaller and more focused than this.

Question 13: Assume that a JDBC connection myCon has been established to a database that has a table Sells(bar, beername, price), where bar and beername are character strings and price is float.

Print out all the beernames and prices that are sold at 'GoodBar'. Don't bother with including libraries or variable declarations, and you can have an informal print statement if you want. Here 's an outline of what you need to write:

```
// Execute the query
// Loop through the results
   // For each value in the result, get the values of beername and price, and print them
```

Answer 13 (not the only way):

```
// Execute the query
  Statement stmt = myCon.createStatement();
  resultset = stmt.executeQuery(
              'SELECT beername, price FROM Sells WHERE bar = 'GoodBar');

// Loop through the results
  while (resultset.next()) {
     // For each value in result, get values of beername and price, and print them
     System.out.println(resultset.getString(1), resultset.getFloat(2));
  }
```

Question 14a: Refer to Figure 11.5 in Textbook Chapter 11. Show how you would add the facts that Star Wars was directed by George Lucas and produced by Gary Kurtz. (If you need new tags, you can make them up.)

Question 14b: In addition, add elements for 2 more movies: *Empire Strikes Back and Return of the Jedi,* including the facts that Carrie Fisher and Mark Hamill appeared in both of these. (It's OK to omit the Year elements for these 2 films.)

Answer for Question 14 both parts:

```
<? xml version = "1.0" encoding = "utf-8" standalone = "yes" ?>
<StarMovieData>
        <Star starID = "cf" starredIn = "sw esb roj">
                <Name>Carrie Fisher</Name>
                <Address>
                        <Street>123 Maple St.</Street>
                        <City>Hollywood</City>
                </Address>
                <Address>
                        <Street>5 Locust Ln.</Stree>
                        <City>Malibu</City>
                </Address>
        </Star>
        <Star starID = "mh" starredIn = "sw esb roj">
                <Name>Mark Hamill</Name>
                <Address>
                        <Street>456 Oak Rd.</Street>
                        <City>Brentwood</City>
                </Address>
        </Star>
        <Movie movieID = "sw" starsOf = "cf mh">
                <Title>Star Wars</Title>
                <Year>1977</Year>
                <Director>George Lucas</Director>
                <Producer>Gary Kurtz</ Producer>
        </Movie>
        <Movie movieID = "esb" starsOf = "cf mh">
                <Title>The Empire Strikes Back</Title>
        </Movie>
        <Movie movieID = "roj" starsOf = "cf mh">
                <Title>Return of the Jedi</Title>
        </Movie>
</StarMovieData>
```

Question 15: does the XML document shown in Textbook Figure **11.3** conform to the XML SCHEMA declarations shown in Textbook Figures 11.19 and 11.20?

YES [✓ NO]

If not, describe something in the document that is a violation of one or both of the schemas, and explain what's the problem.

Any of these would be okay:

1. The document makes use of an element called StarMovieData, which is not defined in either schema.
2. In the document, the Star element includes Street, City, which are not present in either schema. (Instead, the Fig. 20 schema has element Address.)
3. In the document, Movie elements appear nested with in StarMovieData, whereas in the Fig. 19 schema they are nested within Movies.

Question 16: does the XML document shown in Textbook Figure **11.5** conform to the XML SCHEMA declarations shown in Textbook Figures 11.19 and 11.20?

YES NO

If not, describe something in the document that is a violation of one or both of the schemas, and explain what's the problem.

Any of these would be okay:

1. The document makes use of an element called StarMovieData, which is not defined in either schema.
2. In the document, the Star element includes Street and City, which are not defined in either schema.
3. In the document, one of the Star elements includes an Address element with 2 subelements. Instead, the Fig. 20 schema defines element Address that contains a string.)
4. In the document, Movie elements appear nested with in StarMovieData, whereas in the Fig. 19 schema they are nested within Movies.
5. The document makes use of attributes movieID, starredIn and starsOf, none of which are defined in either schema.

Question 17: Normal forms help avoid anomalies. We discussed 3 types of anomalies (Update, Delete and Insert) in class. Explain 2 of these 3 anomalies, giving examples of the problems that could arise.

You may use this table Employee(eid, name, addr, rank, salary-scale), with the Functional Dependency rank → salary-scale, to discuss the anomalies, if you want.

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

Answer 17:

Update anomaly: If Ann becomes rank 6, and her salary-scale doesn't also become 70-90, then the table is inconsistent, because the FD is violated.

Delete anomaly: If Hugh is deleted, then the information that rank 3 is associated with salary scale 30-40 is lost.

Insert anomaly: If Mary is entered with rank 6 but her salary-scale isn't set to 70-90, then the table is inconsistent, because the FD is violated.