Database Systems

Final Assignment (take-home exam)

Exercise I

You have been hired as a database expert by the government of the Tibet Autonomous Region (TAR); your first task is to design a population database that keeps track of marital arrangements. In Tibet, *polyandry* (a marital arrangement in which a woman has several husbands) is the *de facto* norm in rural areas. People in the TAR are identified by an *identification number*.

1. Propose an appropriate E-R diagram design that stores such marital information.
2. Write the corresponding SQL DDL code.
3. The TAR government passes a law that disallows polyandry and establishes monogamy; suggest how all elements of your design should change to reflect the change in the law.

Exercise II

Consider the following self-explanatory database schema. Primary keys are underlined. Assume the natural foreign keys apply (identified by same or similar attribute names). NULL values are forbidden everywhere.

```
object(object, description)
property(object, property, value)
relationship(object1, relationship, object2)
```

We say that "*object1 is in a relationship with object2*" (not that "*object2 is in relationship with object1*"): the position in the schema matters. In other words, relationships, which are given as values of the relationship attribute, are *one-directional*.

Express the following queries in SQL, using your knowledge of primary keys and foreign keys.
In all questions, eliminate duplicates, using DISTINCT, when and only when it is necessary; it is not necessary to eliminate duplicates when duplicates cannot exist.

Hint: *it is a good practice to use **tuple variables**; this practice allows you to refer to more than one tuple of the same table, as in*: SELECT v1.name, v2.name FROM table v1, table v2, …

Question 1. Find all `objects` that are in a relationship with object 'Cynthia' and print only the `description` attribute of these objects.

Question 2. Find all *pairs* of *different* `objects`, such that both `objects` in a pair are in some `relationship` with object 'Cynthia'. Print each pair **only once**. (*Hint: you may need to avoid printing symmetric answers such as ('Ann', 'Bob') and ('Bob', 'Ann'*) *by allowing the members of a pair to appear only in a certain order.*)

Question 3. Print a list of different `descriptions` that contain the keyword 'electronic' and the number of objects having each such a description. Limit your result only to those descriptions that are shared by *more than* 10 objects. (*Hint*: *you can find out if a string contains a keyword using LIKE '%keyword%' in the WHERE condition.*)

Question 4. Find those objects that are in a relationship with ***all*** objects. Do not use any aggregation operation. (*in other words*: find those objects such that there exists no object they are not in a relationship with.)

Question 5. Express the intent and meaning of the following query in simple **but precise** English.

```
SELECT O.object
FROM object O
WHERE O.object NOT IN (SELECT R.object2 FROM relationship R)
```

Question 6. A *reciprocal relationship* is a case where two objects have the same `relationship` mutually with each other (i.e., in both ways). Find all pairs of `objects` that are **only** in reciprocal relationships with each other (i.e., are in at least one reciprocal relationship with each other, and are not in any non-reciprocal relationship with each other). Print each pair **only once**.

Exercise III

Consider the relational scheme R with the set of functional dependencies F.

R = {A, B, C, D}
F = { {A} → {C}, {B,C} → {D}, {A,B} →{C} }

Question 1. Find the candidate keys of R with F. Explain why they are candidate keys, and provide an argument explaining that you have not missed anything.

Question 2. Compute a minimal cover of F. Briefly show the steps of the process.

Question 3. Give a 3NF synthesis of R with F. Briefly show the steps of the process. Explain why this result is lossless and why it is in 3NF.

Question 4. Explain, precisely and briefly, why R with F is not in BCNF.

Question 5. Prove that {A,B} → {D} is in F+, the closure of F.

Using the algorithm of the lecture, we can find *two* decompositions of R in BCNF. The first of those *is not* dependency-preserving, while the second *is* dependency-preserving. In the following, you need to find these two decompositions. Give the steps clearly and concisely. Show the fragments resulting at *each step* and their projected sets of functional dependencies, and indicate whether each of those fragments is in BCNF or not.

Question 6. Show the *first* decomposition you can produce, which **is not** dependency-preserving. Indicate which functional dependency is lost (*hint: start by considering one of the dependencies in the minimal cover of F that violate BCNF; what you will get depends on which dependency you consider first*).

Question 7. Show the *second* decomposition you can produce, which **is** dependency-preserving (*hint: start by considering another dependency in the minimal cover of F that violates BCNF, not the one you considered above; show all steps and the fragments produced in them*).

Exercise IV

Consider the following SQL query (from now on, we call it “the query”):

```
SELECT B.Title
FROM Book B, Purchasing P
WHERE B.BId = P.BId AND B.Category = 'CS' AND P.Month = 'Mar'
```

The query is evaluated over a relational database instance with the schema:

```
Customer (CId, Name)
Book (BId, Title, ISBN, Year, Category)
Purchasing (CId, BId, Month)
```

You are given the following statistics information of the database instance:

- `Book` has 2400 tuples and `Purchasing` has 21600 tuples;
- `Book` has 6 distinct values over the Category attribute;

- `Purchasing` has 10 distinct values over the Month attribute;
- Values of non-key attributes are **uniformly distributed**;
- `Book` is stored in a file of 1200 disk blocks; `Purchasing` in a file 1800 disk blocks;
- A memory buffer of 82 pages is available for I/O; one of these pages is used for output;
- `Book` has a 2-level clustered B^+-tree on `Category`; this index allows the tuples of any Category value to be accessed using **two** disk block accesses on the index, plus any number of disk block accesses required for reading the data;
- `Purchasing` has a 2-level clustered B^+-tree on `Month`; this index behaves following the same principle described above for the other B^+-tree.

Answer the following questions, which are building on each other in a sequence:

1. How many tuples are there in `Book` for each category value? (*hint: remember the fact that values of non-key attributes are uniformly distributed*)

2. How many disk blocks are needed to store the `Book` tuples for each category value?

3. How many tuples in `Purchasing` qualify the query's condition on Month? (*hint: use the specific information provided in the above*)

4. How many tuples of `Purchasing` are there in one disk block?

5. How many disk blocks should we reach to obtain all tuples in `Purchasing` that qualify the query's condition on Month?

6. Assume we use the B^+-tree index on `Book` to process the query's condition on a book's Category. How many disk blocks have to be read for that purpose? (*hint: you may need to count the disk blocks for going through the B^+-tree, plus the disk blocks for reading tuples*)

7. Similarly, assume we use the B^+-tree index on `Purchasing` to process the query's condition on Month. How many disk blocks have to be read for that purpose? (*hint: again, you may need to count the disk blocks as you did above*)

8. Consider these two ways of getting the result of the given query:
    a. We use 80 pages of memory for the input of `Book`, 1 page for the input of `Purchasing`, and 1 page for output. We load the qualifying blocks in `Book` in iterations (*i.e.*, steps), using the 80 pages of allocated memory (i.e., we load as many as fit in memory in each iteration). In each iteration, we find all qualifying tuples in `Purchasing`, for each qualifying tuple in `Book` that is in memory.
    b. We use 80 pages of memory for the input of `Purchasing`, 1 page for the input of `Book`, and 1 page for output. We load the qualifying blocks in `Purchasing` in iterations (*i.e.*, steps), using the 80 pages of allocated memory (i.e., we load as many as fit in memory in each iteration). In each iteration, we find all qualifying tuples in `Book`, for each qualifying tuple in `Purchasing` that is in memory.

    Any variables the program needs to make the process run do not occupy space in the memory buffer reserved for I/O. Describe how each of the above two suggested ways can use B^+-trees. Estimate how many disk blocks each of them accesses in total.

9. Which of the two ways suggested above would you prefer?

10. Suggest a mathematical rule that decides, in the *general case*, how to process a join query that involves a condition on one attribute of each of two joined tables, when we have a B^+-tree on each of these two attributes, as we did in the above case. (*Hint: You may want to introduce certain parameters and define your rule based on those parameters' values.*)

Exercise V

You are given the following transaction schedule S. In the notation used below, R stands for a Read, W stands for a Write, Qn(A), where Q is either R or W, means that operation Q is done by transaction *n* on data item A. For example, R1(C) means that transaction 1 Reads item C. The full schedule is the following:

S: R1(A), W1(B), R2(C), R1(C), R2(B), W3(B), R2(A), W3(A)

Can this schedule be executed in a way that satisfies the 2-Phase Locking (2PL) Protocol?

If the answer is *no*, then:

1. Explain why it is so.
2. Suggest a change that would make the schedule executable by the 2PL protocol.

If the answer is *yes*, then:

1. Present a complete version of the schedule, with Lock and Unlock operations inserted. You may denote such operations with S for Shared Lock, X for Exclusive Lock, and U for Unlock. Explain why the 2-Phase Locking Protocol is satisfied by your solution.

2. Can you suggest a change in this schedule that keeps each transaction unchanged, yet makes it *impossible* to execute the new, changed version of the schedule, by the 2PL protocol? If *yes*, do propose such a change. If *not*, explain why not.

Submission Format

Type your solutions. Submit a single (no multiple) **pdf** file (no image files or other formats) in blackboard. Write your name **both** inside the submitted document and in the file name. Name the file after your surname with "A4" appended, for example, andersenA4.pdf. You are allowed a **single submission**, just like in a regular written exam.