Instructions:

1. This is a closed book examination. No additional materials are allowed.
2. Answer all the questions.
3. Answer each question in the space provided.
4. You can use the back of the sheets for rough work.
5. The exam consists of 8 questions and 20 (twenty) pages; make sure you have all of the pages.

Question:	1	2	3	4	5	6	7	8	Total
Points:	10	10	8	6	4	4	4	4	50
Score:									

Q1. (10 points)

Answer the following questions with a few sentences (no longer than the allocated space). Each question is worth 1 point.

(a) What are the three major steps of the database design (data modeling) process? Define each by ***one*** sentence.

Solution:

- Conceptual modeling: The process of identifying the entities, their properties, and the relationships among these entities that are part of the environment that the information system models. In our case, we used E-R model for conceptual modeling.
- Logical modeling: The process of mapping the conceptual model to the primitives of the data model that is used. In the case of relational DBMSs, this is the process of defining the relational schema.
- Physical modeling: The process of defining the physical organization characteristics (e.g., indexes, file structure) of the database.

(b) What types of participation constraints can you have in an E-R model? Define each by ***one*** sentence.

Solution: Two types:

1. Total participation constraint: If entity type E1 is in total participation in relation R with entity type E2, then every entity instance of E1 has to participate via relation R to an entity instance of entity type E2.
2. Partial participation constraint: If, in the above situation, it is acceptable that some instances of E1 participate in relationship R with instances of E2,while others do not have to, then we have a partial participation constraint.

(c) Given relation R(A, B, C, D, E) where (A,B) is the key, and the functional dependencies (A, B) $\rightarrow$ (C, D, E) and B $\rightarrow$ D, is R in Boyce-Codd Normal Form (BCNF)? Justify your answer with ***one*** sentence.

Solution: Relation R is not in BCNF, because D is functionally dependent on part of the key (B) and not the full key.

(d) What is the main difference between relational calculus and relational algebra?

Solution: Relational calculus is a declarative language that requires the user to specify the predicates that need to be satisfied by all the tuples in the result relation. Relational algebra, on the other hand, is procedural where the user specifies how to execute the query by means of relational algebraic operators.

(e) What is referential integrity? How do you represent it in relational model?

Solution: Referential integrity refers to the relationship between two entities such that if there is a referential integrity from entity E1 to entity E2, an instance of E2 has to exist for an instance of E1 to exist. In the relational model, this is represented by foreign keys.

(f) What is the main difference between discretionary access control and mandatory access control?

Solution: From your book "Discretionary access control is based on the concept of rights, or privileges, and mechanism for giving users such privileges. ... Mandaroy access control is based on systemwide policies that cannot be changed by individual users." The fundamental point is that in the first case, the granting of rights to objects are at the discretion of users who already hold rights to those objects, while in the latter case, this is done systemwide.

(g) What are ACID properties of transactions. Explain each by ***one*** sentence.

Solution: **Atomicity:** All actions of a transaction are atomic and either they are all performed or none of the actions are performed.

Consistency: Each transaction, when run alone, must preserve the consistency of the database.

Isolation: Each transaction is isolated (protected) from the effects of other concurrently running transactions.

Durability: Once a transaction successfully completes (commits), its effects on the database will survive any future crashes.

(h) Which of the following are the objectives or features of NoSQL systems? Put down the numbers.

1. Data independence
2. Simplicity (schema, basic API)
3. Scalability and performance
4. ACID transaction support
5. Well defined data schema

Solution: 2,3

(i) Briefly describe how RDD in Spark achieves fault tolerance.

Solution: RDDs maintain lineage information that can be used to reconstruct lost partitions. Lineage is constructed as an object and stored persistently for recovery.

(j) Given a relation R, at most how many clustered index can be built? Briefly explain your answer.

Solution: One. Because there is only one way to layout data physically.

Q2. (10 points)
You are designing a database for KW Humane Society (the organization that shelters animals). The result is the following set of relations where the type of each relations attribute is given following the attribute (e.g., ID: integer):

Animals(<u>ID: `integer`</u>, Name: `string`, CurrOwner: `integer`, DateAdmitted: `date`, Type: `string`)
Adopter(<u>SIN: `integer`</u>, Name: `string`, Address: `string`, OtherAnimals: `integer`)
Adoption(<u>AnimalID: `integer`, SIN: `integer`</u>, AdoptDate: `date`, chipNo: `integer`)

where

(a) The primary keys are underlined.

(b) Animals relation stores information about the animals currently registered at the Humane Society. Each is given an ID, and their names together with the SIN of their current owners (attribute CurrOwner), and their date of admission is recorded. If an animal is not adopted, its CurrOwner would be NULL. Type refers to the type of animal (dog, cat, etc).

(c) Adopter is the relation that holds information about animal adopters. The attributes are self-descriptive, except OtherAnimals which records the number of other animals that the adopter currently has at home.

(d) AnimalID in Adoption refers to the ID of Animals. Similarly, SIN in Adoption refers to the SIN of Adopter. Attribute chipNo stores the number on the microchip that is implanted on the animal for tracking. CurrOwner in Animals refers to the SIN of Adopter.

Formulate the following queries in SQL; each one is worth 2 points:

(a) Retrieve the total number of dogs that were brought to the Humane Society on 18 April 2000.

Solution:

```
SELECT  COUNT(*)
FROM    Animals
WHERE   Type = ``dog''
AND     DateAdmitted = ``18/04/2000''
```

(b) List the name of the adopter who has adopted every type of animal.

Solution:

```
SELECT  Name
FROM    Adopter
WHERE   NOT EXISTS
         (SELECT *
        FROM    Animals A1
        WHERE   NOT EXISTS
                 (SELECT *
                FROM    Adoption, Animals A2
                WHERE   AnimalID = A2.ID
                AND     A2.Type = A1.Type
                AND     Adoption.SIN = Adopter.SIN))
```

(c) For each animal type, list the animal type and total number of adoptions on 14 June 1999.

Solution:

```
SELECT  Type, COUNT(*)
FROM    Animals, Adoption
WHERE   AdoptDate = ``14/06/1999''
AND     Animals.ID = Adoption.AnimalID
GROUP BY Type;
```

(d) List the types of animals who have not had any adoptions.

Solution:

```
SELECT DISTINCT Type
FROM    Animals
WHERE   NOT EXISTS
         (SELECT *
        FROM    Adoption
        WHERE   Adoption.AnimalID = Animals.ID)
```

(e) For each adopter who has made at least two adoptions, list their names and addresses.

Solution:

```
SELECT  Name, Address
FROM    Adopter, Adoption
WHERE   Adopter.SIN = Adoption.SIN
GROUP BY Adoption.SIN
```

```
HAVING   COUNT(SIN)  >  1;
```

Q3. (8 points)

Given relation R(A, B, C, D, E, F, G) and the set of functional dependencies F={BCD → A, BC → E, A → F, F→G, C→D, A→G}.

(a) (6 points) Decompose R into 3NF. Show your steps. Note: This requires you to first determine the key(s) of R.

Solution: The key of this relation is (B,C). The argument is simple.

> BC→E
> C→D, by augmentation BC→DC, by decomposition BC→D
> BC→D and BCD→A, by pseudotransitivity BC→A
> BC→A and A→F, by transitivity BC→F
> BC→F and F→G, by transitivity BC→G
> Thus, BC→ADEFG and there is (are) no other attribute(s) that functionally determine all attributes of R.

Now we apply the 3NF Synthesize algorithm.

Step 1. We start with result = $\emptyset$

Step 2. Compute minimal cover for F.

1. The right-hand sides are single attributes, so this step is not necessary.
2. Consider BCD→A

   $(CD)^+$ = C,D
   $(BD)^+$ = B,D
   $(BC)^+$ = A, B,C

   Since A is in $(BC)^+$, we replace BCD→A by BC→A
   Now consider BC→E

   C^+ = C,D
   B^+ = B

   So, nothing happens.
3. Now consider each FD:

   Consider A→F; A^+ w.r.t. G−(AF) = {A, G}
   Consider F→G; F^+ w.r.t. G−(FG) = {F}
   Consider C→D; C^+ w.r.t. G−(CD) = {C}
   Consider A→G; A^+ w.r.t. G−(AG) = {A, F, G}

Therefore A→G is redundant.

Thus, the minimal cover $F' = \{$BC→A, BC→E, A→F, F→G, C→D$\}$.

Step 3. For each $X \to Y \in F'$, create a relation:

R1(B, C, A, E)
R2(A, F)
R3(F, G)
R4(C, D)

(b) (2 points) Is this decomposition also BCNF? Why or why not?

Solution:

This decomposition is also in BCNF. There are a number of ways to reason about this:

1. You can do a BCNF decomposition to show that you can get a decomposition that is the same as this one. However, note that the BCNF decomposition is not unique, so you have to be careful with the particular decomposition – you may need to try a few.
2. You can show that the decomposition is lossless and has no redundancy by showing that elimination of any one of the decomposed relations will make it lossy. Remember that BCNF is guaranteed to be lossless, but not dependency preserving while the 3NF decomposition will be lossless and dependency preserving at the cost of (potentially) adding redundancy. If you can show that there is no redundancy in the resulting decomposition, then it would also be in BCNF.
3. You can get a BCNF decomposition ***from*** the 3NF decomposition. This is based on the understanding that 3NF eliminates transitive dependencies on the superkey, but not partial dependencies (at the same time) while BCNF eliminates both of these (see slide 28). So, if you have a 3NF decomposition, and you can show that there are no partial dependencies in F in that decomposition, the resulting decomposition would be in BCNF.

Note that this decomposition (which is BCNF) is also dependency preserving. If you take the projection of F over R1, R2, R3, and R4 and then take their union, the following FDs will be missing: BCD→A and A→G. However, if you take the closure of the union, BCD→A will be included due to augmentation and decomposition, and A→G will be included due to transitivity. Therefore the closure of the union is equal to the closure of the original FD and thus it is dependency preserving.

Q4. (6 points)

Consider the ER model given in Figure 1. This model represents the operations of a pharmacy chain. Please answer the following questions regarding this model.

(a) (1 point) Can a pharmaceutical company have multiple phone numbers? If not, what do you need to do to allow this?

Solution: Yes, the pharmaceutical company can have multiple phone numbers. This is because the Phone attribute is defined as a multivalued one.

(b) (1 point) If we delete from the database the pharmaceutical company that manufactures a drug, what happens to the drugs that the company manufactures? Justify (in one or two sentences only) your argument.

Solution: All information about these drugs are deleted from the database as well. This is due to the fact that Drug is a weak entity of Pharmaceutical Co. entity and instances of Drug cannot exist without the existence of their strong entity instance.

(c) (1 point) Similar to part (b), but instead of deleting the pharmaceutical company, what if we delete the pharmacy that sells the drug. Do we have to delete the drug too? Why or why not?

Solution: In this case nothing happens, i.e., we do not have to delete the drug. This is because there is no weak entity-strong entity relationship between Drug and Pharmacy. Instances of these entity types can exist on their own..

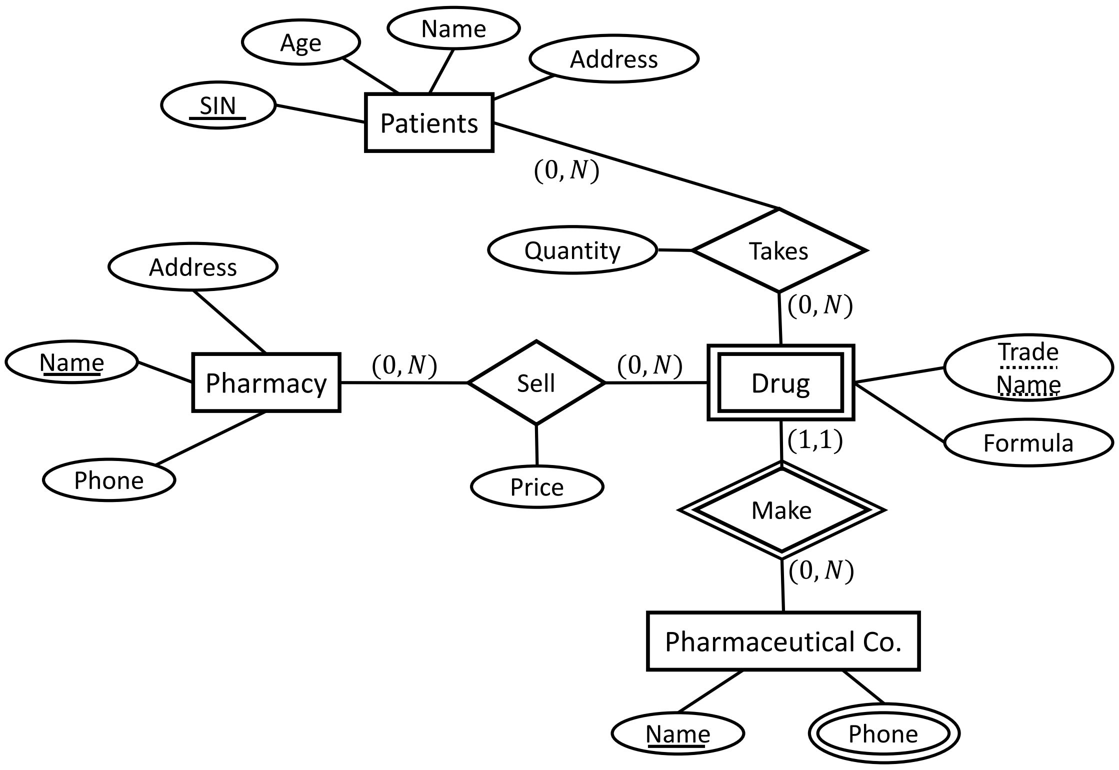


Figure 1: Figure for Question 4

(d) (3 points) Modify the model (by adding to Figure 1, ***not*** by drawing another figure) so that you can represent the following

- Each patient has to have one and only one primary physician, who can be identified by SIN. Each physician has at least one patient. In addition, we want to record the specialty and the date of entry into the profession of each physician.
- Instead of modeling only the fact that a patient takes certain drugs, model the fact that a patient takes drugs that are prescribed by a physician and the prescription date.
- Pharmaceutical companies have long-term contracts with pharmacies. A pharmaceutical company can contract with several pharmacies, and a pharmacy can contract with several pharmaceutical companies. For each contract we want to store a start date, an end date.

Solution: See Figure 2

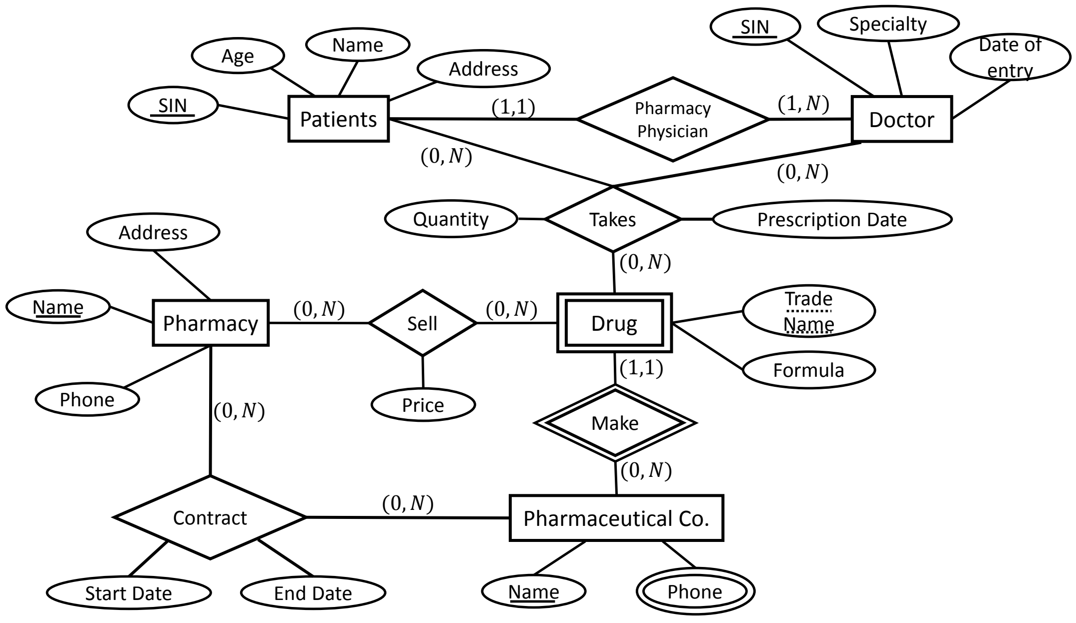


Figure 2: Solution to Question 4d

Q5. (4 points)
Consider histories H_1 and H_2 given below:

$$H_1 = r_1(x), r_2(z), r_1(z), r_3(x), r_3(y), w_1(x), w_3(y), r_2(y), w_2(z), w_2(y)$$
$$H_2 = r_1(x), r_2(z), r_3(x), r_1(z), r_2(y), r_3(y), w_1(x), w_2(z), w_3(y), w_2(y)$$

These histories are generated by the following transactions:

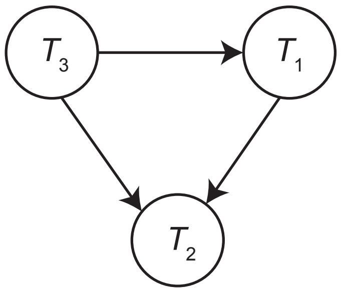
$$T_1 = r_1(x), r_1(z), w_1(x)$$
$$T_2 = r_2(z), r_2(y), w_2(z), w_2(y)$$
$$T_3 = r_3(x), r_3(y), w_3(y)$$

(a) (2 points) Draw the serialization graph for H_1 and state whether or not it is serializable. If it is serializable, give the equivalent serial history.

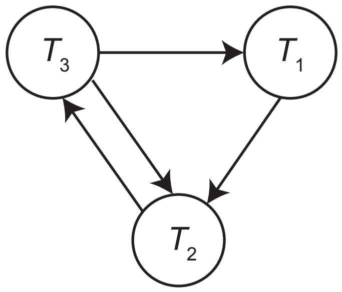
Solution: The serialization graph for H_1 is given in Figure 3a. H_1 is serializable and the serialization order is $T_3 \rightarrow T_1 \rightarrow T_2$.

(b) (2 points) Draw the serialization graph for H_2 and state whether or not it is serializable. If it is serializable, give the equivalent serial history.

Solution: The serialization graph for H_2 is given in Figure 3b. Since there is a cycle in the graph, H_2 is not serializable.



(a) Solution to Question 5a



(b) Solution to Question 5b

Figure 3: Solutions to Question 5b

Q6. (4 points)
Consider the following schema:

Suppliers(<u>sid: `integer`</u>, sname: `string`, address: `string`)
Parts(<u>pid: `integer`</u>, pname: `string`, colour: `string`)

Catalog(<u>sid: `integer`, pid: `integer`</u>, cost: `real`)

The Catalog relation lists the prices charged for parts by Suppliers.

For each of the following transactions state the SQL isolation level that you would use and explain why you chose it.

(a) (1 point) A transaction that adds a new part to a supplier's catalog.

Solution: First note that this is very similar to the problem that you solved in Assignment 3.

Because we are inserting a new row in the table Catalog, we do not need any lock on the existing rows. So it may appear that using READ UNCOMMITTED would be sufficient. However, note that this isolation level is only allowable for read-only queries. Therefore, READ COMMITTED would need to be used.

(b) (1 point) A transaction that increases the price that a supplier charges for a part.

Solution: Because we are updating one existing row in the table Catalog, we need an exclusive access to the row which we are updating. So we would use READ COMMITTED.

(c) (1 point) A transaction that determines the total number of items for a given supplier.

Solution: To prevent other transactions from inserting or updating the table Catalog while we are reading from it (known as the phantom problem), we would need to use SERIALIZABLE.

(d) (1 point) A transaction that shows, for each part, the supplier that supplies the part at the lowest price.

Solution: Same as (c).

Q7. (4 points)
(Exercise 15.2 in Textbook) Consider the following two transactions:

$$\begin{aligned} T_1 : \ & read(A) \\ & read(B) \\ & if\ A = 0\ then\ B := B + 1 \\ & write(B) \end{aligned}$$

$$\begin{aligned} T_2 : \ & read(B) \\ & read(A) \\ & if\ B = 0\ then\ A := A + 1 \\ & write(A) \end{aligned}$$

(a) (2 points) Add lock and unlock instructions to transaction T_1 and T_2, so that they observe the two-phase locking protocol.

Solution:

$$\begin{aligned} T_1 : &\ lock - S(A) \\ &\ read(A) \\ &\ lock - X(B) \\ &\ read(B) \\ &\ if\ A = 0\ then\ B := B + 1 \\ &\ write(B) \\ &\ unlock(A) \\ &\ unlock(B) \end{aligned}$$

$$\begin{aligned} T_2 : &\ lock - S(B) \\ &\ read(B) \\ &\ lock - X(A) \\ &\ read(A) \\ &\ if\ B = 0\ then\ A := A + 1 \\ &\ write(A) \\ &\ unlock(B) \\ &\ unlock(A) \end{aligned}$$

(b) (2 points) Can the execution of these transactions (using two-phase locking) result in a deadlock? Explain your answer.

Solution: Yes, a deadlock can happen, as shown in the following schedule:

T_1	T_2
lock-S(A)	
	lock-S(B)
	read(B)
read(A)	
lock-X(B)	
	lock-X(A)

Q8. (4 points)
Given the geographic locations (latitude and longitude) of a number of road intersections, and the locations of a set of gas stations, find the nearest gas station for each intersection if their geometric distance is less than 5km. Assumptions:

- There are total N intersections, each can be uniquely identified by i_k, where $k = 1, ..., N$;
- There are total M gas stations, each can be uniquely identified by g_j, where $j = 1, ..., M$;

Design a MapReduce job to do this task. Note that you are not asked to write MapReduce code. Pseudo code or elaborating the input and output of Map and Reduce functions are sufficient. Specifically, you need to show how (*key, value*) pairs are generated and aggregated in your design. Please put down your assumption if there is any.

(a) (2 points) Map function:

Solution:

- Input $\langle id, la, lo \rangle$
  1. If it is a intersection, map it to a key-value pair $\langle i_k, (i_k, la, lo) \rangle$;
  2. If it is a gas station, map it to N key-value pairs: $\langle i_k, (g_j, la, lo) \rangle$, where $k = 1, ..., N$
- Output $\langle i_k, (id, la, lo) \rangle$

(b) (2 points) Reduce function:

Solution:

- Input $\langle i_k$, list$(id, la, lo) \rangle$
  1. For each i_k, examine each gas station g_j in its list of values, compute and filter their geometric distance to obtain $\langle i_k$, list$(g_j, dist) \rangle$, where $dist \leq 5$km;
  2. For each i_k, search and keep g_j with the minimum *dist*
- Output $\langle i_k, g_j \rangle$