SOLUTION

Brought to you by

Database Management Systems

Section – A

Attempt any five out of six.

Each question carries 6 marks.

Soln. 1

The key difference is that RDBMS (relational database management system) applications store data in a tabular form, while DBMS applications store data as files. This

doesn’t mean there are no tables in DBMS. There can be, but there will be no “relation” between the tables, like in a RDBMS. In DBMS, data is generally stored in either a hierarchical form or a navigational form. This means that a single data unit will have one parent node and zero, one or more children nodes. It may even be stored in a graph form, which can be seen in the network model.

In a RDBMS, the tables will have an identifier called primary key. Data values will be stored in the form of tables. The relationships between these data values will be stored in the form of a table as well. Every value stored in the relational database is accessible. This value can be updated by the system. The data in this system is also physically and logically independent.

E.CODDs rules:

Dr Edgar F. Codd, after his extensive research on the Relational Model of database systems, came up with twelve rules of his own, which according to him, a database must obey in order to be regarded as a true relational database.

Rule 1: Information Rule

The data stored in a database, may it be user data or metadata, must be a value of some table cell. Everything in a database must be stored in a table format.

Rule 2: Guaranteed Access Rule

Every single data element (value) is guaranteed to be accessible logically with a combination of table-name, primary-key (row value), and attribute-name (column value). No other means, such as pointers, can be used to access data.

Rule 3: Systematic Treatment of NULL Values

The NULL values in a database must be given a systematic and uniform treatment. This is a very important rule because a NULL can be interpreted as one the following − data is missing, data is not known, or data is not applicable.

Rule 4: Active Online Catalog

The structure description of the entire database must be stored in an online catalog, known as data dictionary,

which can be accessed by authorized users. Users can use the same query language to access the catalog which they use to access the database itself.

Rule 5: Comprehensive Data Sub-Language Rule

A database can only be accessed using a language having linear syntax that supports data definition, data manipulation, and transaction management operations. This language can be used directly or by means of some application. If the database allows access to data without any help of this language, then it is considered as a violation.

Rule 6: View Updating Rule

All the views of a database, which can theoretically be updated, must also be updatable by the system.

Rule 7: High-Level Insert, Update, and Delete Rule

A database must support high-level insertion, updatation, and deletion. This must not be limited to a single row, that is, it must also support union,

intersection and minus operations to yield sets of data records.

Rule 8: Physical Data Independence

The data stored in a database must be independent of the applications that access the database. Any change in the physical structure of a database must not have any impact on how the data is being accessed by external applications.

Rule 9: Logical Data Independence

The logical data in a database must be independent of its user's view (application). Any change in logical data must not affect the applications using it. For example, if two tables are merged or one is split into two different tables, there should be no impact or change on the user application. This is one of the most difficult rules to apply.

Rule 10: Integrity Independence

A database must be independent of the application that uses it. All its integrity constraints can be independently

modified without the need of any change in the application. This rule makes a database independent of the front-end application and its interface.

Rule 11: Distribution Independence

The end-user must not be able to see that the data is distributed over various locations. Users should always get the impression that the data is located at one site only. This rule has been regarded as the foundation of distributed database systems.

Rule 12: Non-Subversion Rule

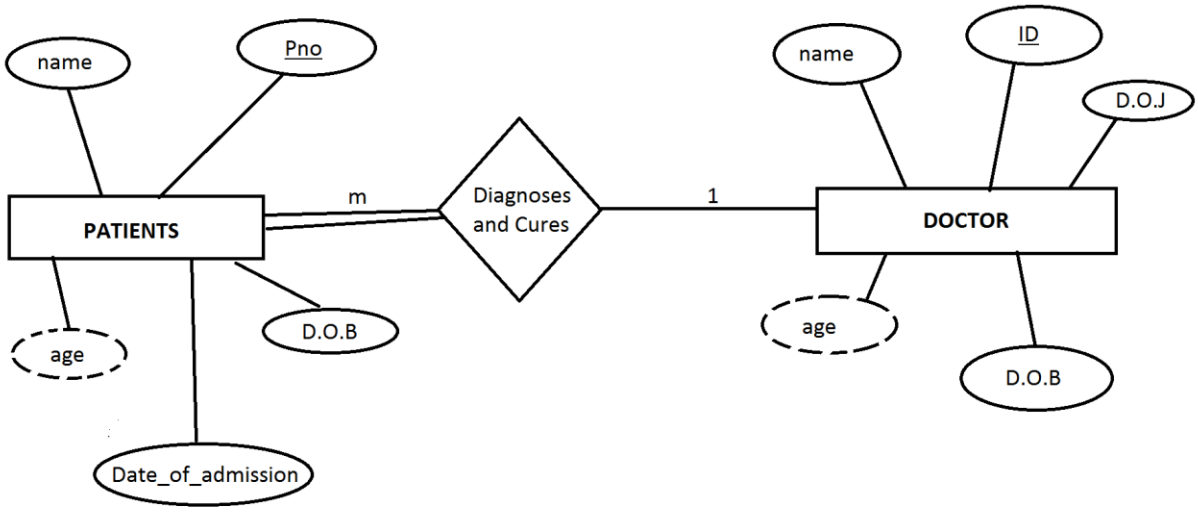
If a system has an interface that provides access to low-level records, then the interface must not be able to subvert the system and bypass security and integrity constraints.

Soln. 2:

ER Data Model is based on the real world objects and their relationship. In other words, each and everything, either living or non-living things in this world forms the

object in database world. The ER model defines the conceptual view of a database. It works around real-world entities and the associations among them. At view level, the ER model is considered a good option for designing databases.

For this purpose, we use ER diagrams where we plan the database pictorially. ER diagram basically breaks requirement into entities, attributes and relationship.



Soln. 3:

(a) Weak entity: weak entity is one that can only exist when owned by another one. For example: a ROOM can only exist in a BUILDING.
Strong entity: String entity is one that can exist

independently. For example: A TIRE might be considered as a strong entity because it also can exist without being attached to a CAR.

(b) Database Languages: Database languages are used for read, update and store data in a database. There are several such languages that can be used for this purpose; one of them is SQL (Structured Query Language).
Types: DDL, DML, DCL, TCL.

(c) Data Abstraction: It's the ability to reduce a particular body. Only the essential information is shown and rest is hidden.

Soln. 4:

(a) Primary Key and Unique Key are Entity integrity constraints. Primary key allows each row in a table to be uniquely identified and ensures that no duplicate rows exist and **no NULL values are entered.** Whereas, unique key, just as primary key identifies an entity, but **can take NULL values.** Foreign keys are used to reference unique columns in another table. So, for example, a

foreign key can be defined on one table A, and it can reference some unique column(s) in another table B. Why would you want a foreign key? Well, whenever it makes sense to have a relationship between columns in two different tables.

(b) A relation, R, is in 3NF iff for every nontrivial FD (X->A) satisfied by R at least ONE of the following conditions is true:

(a) X is a superkey for R, or

(b) A is a key attribute for RBCNF requires (a) but doesn't treat (b) as a special case of its own. In other words BCNF requires that every nontrivial determinant is a superkey even its dependent attributes happen to be part of a key.
A relation, R, is in BCNF iff for every nontrivial FD (X->A) satisfied by R the following condition is true:

(a) X is a superkey for R

BCNF is therefore more strict.

Soln. 5:
S = (V, W, X, Y, Z);

FDs:

Z→V, W→Y, XY→Z, V→WX

Closure sets:

→V = V, W, X, Y, Z => V is a super key.

→XY = X, Y, Z, V, W => XY is a super key.

→Z = Z, V, W, X, Y => Z is a super key.

All of them a candidate keys too.

For a decomposition to be lossless 3 conditions should be satisfied:

. Condition 1: **S1 U S2 = S**

. Condition 2: **S1 Λ S2 ≠ Ø**

. Condition 3: **S1 Λ S2 should be a SK**

(i) S1 = (V, W, X)
S2 = (V, Y, Z)

(V, W, X) U (V, Y, Z) = (X, Y, Z, V, W) = S //cond 1 satisfied.

(V, W, X) **Λ** (V, Y, Z) = V **≠ Ø**//cond 2 satisfied

(V, W, X) **Λ** (V, Y, Z) = V is a SK//cond 3 satisfied.

This decomposition is lossless.

(ii) S1 = (V, W, X)
S2 = (X, Y, Z)

(V,W, X) U (X, Y, Z) = (X, Y, Z, V, W) = S//cond 1 satisfied.

(V,W, X) **Λ** (X, Y, Z) = X **≠ Ø**//cond 2 satisfied.
(V,W, X) **Λ** (X, Y, Z) = X is NOT a SK. //cond 3 NOT satisfied.

This decomposition is NOT lossless.

Soln. 6:

Yes. The FD ACDF → G holds because for G to be determined. We need EF. We already have F. So we need to determine E using ACD. Since ABCD → E, and A →B. We can determine both E and F. Therefore, The FD ACDF → G holds true.

Section – B

Attempt any two out of three questions.

Each question carries 10 marks.

Soln. 7:

(a) Super Key: A super key of a relation schema R = {A1, A2, ... , An} is a set of attributes S ⊆ R with the property that no two tuples t1 and t2 in any legal relation state r of R will have t1[S] = t2[S]. A key K is a super key with the additional property that removal of any attribute from K will cause K not to be a super key anymore.

(b) Candidate Key: A nominee for primary key field is known as candidate key.

(c) Triggers: Triggers are stored programs, which are automatically executed or fired when some events occur. Triggers are, in fact, written to be executed in response to any of the following events:

→A database manipulation (DML) statement (DELETE, INSERT, or UPDATE).

→A database definition (DDL) statement (CREATES, ALTER, or DROP).

→A database operation (SERVERERROR, LOGON, LOGOFF, STARTUP, or SHUTDOWN).

Soln. 8: // Will be updated.

Soln. 9:

T31	T32
Read_lock(A)	**Read_lock(B)**
Read(A)	Read(B)
Read_lock(B)	**Read_lock(A)**
Read(B)	Read(A)
If A = 0 then B := B+1	If B = 0 then A := A+1
Write_lock(B)	**Write_lock(A)**
Write(B)	Write(A)
Unlock(A)	Unlock(B)
Unlock(B)	Unlock(A)

Dead lock can arise when both are requesting an exclusive lock.

Section – C

All questions are compulsory

Soln. 10:

(a)

(i)
```
SELECT degcode
FROM candidate
order by degcode;
EXCEPT
SELECT degcode
FROM DEGREE
ORDER BY degcode;
```

(ii)
```
SELECT name
FROM degree, marks
WHERE marks.papercode = 'PHYS'
ORDER BY name;
```

(iii)
```
SELECT name, subject, count(*)
FROM degree, candidates
WHERE count(candidates.degcode) < 5;
```

(iv)
```
SELECT name
FROM candidate, marks
WHERE (count(*) WHERE Marks.marks <
40) = 5;
```

(v)
```
SELECT name
FROM candidates, Marks
WHERE Marks.marks= MAX(Marks.marks);
```

(b) //Will be updated

(c) In a multi-process system, deadlock is an unwanted situation that arises in a shared resource environment, where a process indefinitely waits for a resource that is held by another process.

Deadlock occurs when each transaction T in a set of two or more transactions is waiting for some item that is locked by some other transaction T in the set. Hence, each transaction in the set is in a waiting queue, waiting for one of the other transactions in the set to release the lock on an item. But because the other transaction is also waiting, it will never release the lock.

A simple way to detect a state of deadlock is for the system to construct and maintain a wait-for graph. Another simple scheme to deal with deadlock is the use of timeouts. This method is practical because of its low overhead and simplicity.

Deadlock Avoidance

Aborting a transaction is not always a practical approach. Instead, deadlock avoidance mechanisms can be used to detect any deadlock situation in advance.

--------X--------