iv) **Deletion Anomalies** : Due to deletion of particular record some other important information associated with the deleted record get deleted and thus we may lose some other important information from the schema.

Q.2 Define functional dependency. AU : Dec 04,05, May 05,14,15

Ans. : Let P and Q be sets of columns, then : P functionally determines Q, written P → Q if and only if any two rows that are equal on (all the attributes in) P must be equal on (all the attributes in) Q.

In other words, the functional dependency holds if

T1.P = T2.P, then T1.Q=T2.Q

Where notation T1.P projects the tuple T1 onto the attribute in P.

Q.3 Why certain functional dependencies are called trivial functional dependencies ? AU : May-06,12

Ans. :

- A functional dependency FD : X → Y is called trivial if Y is a subset of X. This kind of dependency is called trivial because it can be derived from common sense. If one "side" is a subset of the other, it's considered trivial. The left side is considered the determinant and the right the dependent.
- For example - {A,B} –> B is a trivial functional dependency because B is a subset of A,B. Since {A,B} –> B includes B, the value of B can be determined. It's a trivial functional dependency because determining B is satisfied by its relationship to A,B

AU : May -14

Q.4 Define normalization.

Ans. : Normalization is the process of reorganizing data in a database so that it meets two basic requirements :

1) There is **no redundancy** of data (all data is stored in only one place), and
2) data **dependencies** are logical (all related data items are stored together)

AU : Dec.-15

Q.5 State anomalies of 1NF.

Ans. : All the insertion, deletion and update anomalies are in 1NF relation.

AU : Dec.-12

Q.6 Show that if a relation is in BCNF, then it is also in 3NF.

Ans. :

- Boyce and Codd Normal Form is a **higher version** of the Third Normal form.
- A 3NF table which **does not have multiple overlapping** candidate keys is said to be in BCNF. When the table is in BCNF then it doesn't have partial functional dependency as well as transitive dependency.

Hence it is true that if relation is in BCNF then it is also in 3NF.

2.7 Why it is necessary to decompose a relation ?

AU : May-0

ns. :

- Decomposition is the process of breaking down one table into multiple tables.
- The decomposition is used for eliminating redundancy.

2.8 Explain atleast two desirable properties of decomposition.

AU : May-03,17,19, Dec

ns. : There are two properties associated with decomposition and those are –

- **Loss-less Join or non Loss Decomposition :** When all information found in the ori database is preserved after decomposition, we call it as loss less or non decomposition.
- **Dependency Preservation :** This is a property in which the constraints on the or table can be maintained by simply enforcing some constraints on each of the sr relations.

2.9 'Boyce-Codd normal is found to be stricter than third normal form'. Justify the stater

AU :

ns. : Refer example 3.6.18.

(ii) Shrinking Phase : It is a phase in which the transaction may release the locks but does not obtain any new lock.

Q.15 What is the difference between shared lock and exclusive lock? AU : May-18

vs.	Shared Lock	Exclusive Lock
	Shared lock is used for when the transaction wants to perform read operation.	Exclusive lock is used when the transaction wants to perform both read and write operation.
	Multiple shared lock can be set on a transactions simultaneously.	Only one exclusive lock can be placed on a data item at a time.
	Using shared lock data item can be viewed.	Using exclusive lock data can be inserted or deleted.

Q.16 What type of lock is needed for insert and delete operations. AU : May-17

Ans. : The exclusive lock is needed to insert and delete operations.

Q.17 What benefit does strict two-phase locking provide ? What disadvantages result ?

AU : May-06, 07, Dec.-07

Ans. : Benefits :

1. This ensure that any data written by an uncommitted transaction are locked in exclusive mode until the transaction commits and preventing other transaction from reading that data.
2. This protocol solves dirty read problem.

Disadvantage :

1. Concurrency is reduced.

Q.18 What is rigorous two phase locking protocol ? AU : Dec.-13

Ans. : This is stricter two phase locking protocol. Here all locks are to be held until the transaction commits.

Q.19 Differentiate strict two phase locking and rigourous two phase locking protocol.

AU : May-16

Q.7

Ans. : A mechanism which ensures that simultaneous execution of more than one transactions does not lead to any database inconsistencies is called concurrency control mechanism. AU : Dec.-15

Q.8 State the need for concurrency control.

OR AU : Dec.-17

Q.9 Why is it necessary to have control of concurrent execution of transactions ? How is it made possible ?

Ans. : Following are the purposes of concurrency control – AU : Dec.-02

- To ensure isolation :
- To resolve read-write or write-write conflicts :
- To preserve consistency of database :

Q.10 List commonly used concurrency control techniques. AU : Dec.-11

Ans. : The commonly used concurrency control techniques are –

i) Lock ii) Timestamp iii) Snapshot Isolation

Q.11 What is meant by serializability? How it is tested? AU : May-14,18, Dec.-14,16

Ans. : Serializability is a concept that helps to identify which non serial schedule and find the transaction equivalent to serial schedule.

It is tested using precedence graph technique.

Q.12 What is serializable schedule ? AU : May-17

Ans. : The schedule in which the transactions execute one after the other is called serial schedule. It is consistent in nature. For example : Consider following two transactions T1 and T2

Q.13 Define two phase locking. AU : May-13

Ans. : The two phase locking is a protocol in which there are two phases :

i) **Growing Phase (Locking Phase) :** It is a phase in which the transaction may obtain but does not release any lock.

ii) **Shrinking Phase) :** It is a phase in which the transac

Concept of extraneous ....

Definition : An attribute of a functional dependency is said to be extraneous if we can remove it without changing the closure of the set of functional dependencies. The formal definition of extraneous attributes is as follows :

Consider a set F of functional dependencies and the functional dependency $\alpha \rightarrow \beta$ in F

- Attribute A is extraneous in α if $A \in \alpha$ and F logically implies
  $(F - \{\alpha \rightarrow \beta\}) \cup \{(\alpha - A) \rightarrow \beta\}$
- Attribute A is extraneous in β if $A \in \beta$ and the set of functional dependencies
  $(F - \{\alpha \rightarrow \beta\}) \cup \{(\alpha \rightarrow (\beta - A)\}$ logically implies F.

Algorithm for computing canonical cover for set of functional dependencies F

$F_C = F$

Repeat

Use the union rule to replace any dependencies in F_C of the form

$\alpha 1 \rightarrow \beta 1$ and $\alpha 1 \rightarrow \beta 2$ and $\alpha 1 \rightarrow \beta 1\ \beta 2$

Find a functional dependency $\alpha \rightarrow \beta$ in F_C with an extraneous attribute either in

or in β.

/* The test for extraneous attributes is done using F_C, not F */

If an extraneous attribute is found, delete it from $\alpha \rightarrow \beta$ in F_C.

until (F_C does not change)

Example 3.4.4 *Consider the following functional dependencies over the attribute set R(ABCDE) for finding minimal cover FD = {A -> C, AC -> D, B -> ADE}.*

Solution :

Step 1 : Split the FD such that R.H.S contain single attribute. Hence we get

A -> C

AC -> D

B -> A

B -> D

B -> E

Step 2 : Find the redundant entries and delete them. This can be done as follows.

- **For A -> C :** We find $(A)^+$ by assuming that we delete A -> C temporarily. W $(A)^+$=(A). Thus from A it is not possible to obtain C by deleting A -> C. This mea we can not delete A -> C.
- **For AC -> D :** We find $(AC)^+$ by assuming that we delete AC -> D temporarily. W get $(AC)^+$= (AC). Thus by such deletion it is not possible to obtain D. This means w can not delete AC -> D.
- **For B -> A :** We find $(B)^+$ by assuming that we delete B -> A temporarily. We g $(B)^+$= (BDE). Thus by such deletion it is not possible to obtain A. This means we ca not delete B -> A.
- **For B -> D :** We find $(B)^+$ by assuming that we delete B -> D temporarily We ge $(B)^+$= (BEACD). This shows clearly that even if we delete B -> D we can obtain D This means we can delete B -> A. Thus it is redundant.
- **For B -> E :** We find $(B)^+$ by assuming that we delete B -> E temporarily. We ge $(B)^+$= (BDAC). Thus by such deletion it is not possible to obtain E. This means w can not delete B->E.

To summarize we get now

A -> C

AC -> D

B -> A

B -> E

Thus R.H.S gets simplified.

Step 3 : Now we will simplify L.H.S.

Consider AC -> D. Here we can split A and C. For that we find closure set of A and C

$$(A)^+ = (AC)$$

$$(C)^+ = (C)$$

Thus C can be obtained from both A as well as C. That also means we need not have t ve AC on L.H.S. Instead, only A can be allowed and C can be eliminated. Thus after plification we get

A -> D

o summarize we get now

A -> C

B -> A

B -> D

B -> E

A -> D

B -> A

B -> E

Thus L.H.S gets simplified.

Step 3 : The simplified L.H.S. and R.H.S can be combined together to form

A -> CD

B -> AE

This is a **minimal cover** or **canonical cover** of functional dependencies.

Example 3.4.5 *A relation R(A,C,D,E,H) satisfies the following FDs A -> C, AC -> D, E -> AD, E -> H. Find the canonical cover for this set of FD's.*

Solution : For obtaining canonical cover we have to find the redundant entries from both LHS and RHS and eliminate them.

Step 1 : Suppose we minimize LHS first, then go through each production rule one by one considering LHS.

A -> C, Keep it as it is.

AC -> D, Here A -> C and A -> D, So we remove A -> C , hence A -> D is kept by eliminating C from LHS.

E -> AD, keep it as it is as E is a single attribute at LHS.

E -> H, keep it as it is

Step 2 : Now we will minimize RHS.

A -> C , keep it as it is

A -> D, keep it as it is

E -> AD. That means E -> A and E -> D.

As A -> D is already present in the FD, we got E -> A and A -> D. Thus E -> D is

Definition : A transaction can be defined as a group of tasks that form a single logical unit.

For example - Suppose we want to withdraw ₹ 100 from an account then we will follow following operations :

1) Check account balance
2) If sufficient balance is present request for withdrawal.
3) Get the money
4) Calculate Balance = Balance –100
5) Update account with new balance.

The above mentioned **four steps** denote **one transaction.**

In a database, each transaction should maintain ACID property to meet the consistency and integrity of the database.

Review Question

1. Write a short note on – Transaction Concept. **AU : Dec.-14, Marks 4**

4.2 Properti ; **AU : May-14,18,19, Marks 15**

1) Atomicity :

- This property states that each transaction must be considered as a **single unit** and **must be completed fully or not completed at all.**
- No transaction in the database is left half completed.
- Database should be in a state either before the transaction execution or after the transaction execution. It **should not be in a state 'executing'.**
- For example - In above mentioned withdrawal of money transaction all the five steps must be completed fully or none of the step is completed. Suppose if transaction gets failed after step 3, then the customer will get the money but the balance will not be updated accordingly. The state of database should be either at before ATM withdrawal (i.e customer without withdrawn money) or after ATM withdrawal (i.e. customer with money and account updated). This will make the system in consistent state.

2) Consistency :

- The database must remain in consistent state after performing any transaction.
- For example : In ATM withdrawal operation, the balance must be updated appropriately after performing transaction. Thus the database can be in consistent state.

3) Isolation :

- In a database system where more than one transaction are being executed simultaneously and in parallel, the property of isolation states that all the transactions will be carried out and executed as if it is the only transaction in the system.
- No transaction will affect the existence of any other transaction.
- **For example :** If a bank manager is checking the account balance of particular customer, then manager should see the balance either before withdrawing the money or after withdrawing the money. This will make sure that each individual transaction is completed and any other dependent transaction will get the consistent data out of it. Any failure to any transaction will not affect other transaction in this case. Hence it makes all the transactions consistent.

4) Durability :

- The database should be **strong enough** to handle any system failure.
- If there is any set of insert /update, then it should be able to handle and commit to the database.
- If there is any **failure**, the database should be able to **recover** it to the consistent state.
- For example : In ATM withdrawal example, if the system failure happens after Customer getting the money then the system should be strong enough to update Database with his new balance, after system recovers. For that purpose the system has to keep the **log of each transaction and its failure.** So when the system recovers, it should be able to know when a system has failed and if there is any pending transaction, then it should be updated to Database.

Review Questions