Name: G. Mounika
Roll No: 18A51A0514
Branch: II CSE A
Subject: DBMS
Batch: 03

# ASSIGNMENT-1

1. Consider the following relational schemes for a library databases:

Book (Title, Author, Catalog_no, publisher, year, price)
Collection (Title, Author, Catalog_no)

The following are functional dependencies:

a) Title Author → Catalog_no
b) Catalog_no → Title author publisher year
c) publisher Title year → price
d) Assume {Author, Title} is the key for both schemas.

Apply the appropriate normal form for Book and Collection?

Ans:

Given relational schemes

Book (Title, Author, Catalog_no, publisher, year, price)
Collection (Title, Author, Catalog_no)

and Let us assume {Author, Title} is the key for both schemes

→ The table "Collection" is in BCNF as there is only one functional dependency "Title Author → Catalog_no" and {Author, Title} is key for Collection.

→ Book is not in BCNF because Catalog_no is not a key and there is a functional dependency "Catalog_no → Title Author publisher year".

→ Book is not in 3NF because non-prime attributes (publisher year) are transitively dependent on key [Title, Author].

→ Book is in 2NF because every non-prime attribute of the table is either dependent on the whole of a candidate keys are {Title, Author} and {Catalog_no}. In table Book, non-prime attributes (attributes that do not occur in any Candidate key) are publisher, year and prince.

2. Consider the following transactions with data items p and q initialized to zero:

$T_1$: read(p);
read(q);
If p=0 then q:=q+1;
write(q);

$T_2$: read(q);
read(p);
If q=0 then p:=p+1;
write(p);

Solve and find any non-serial interleaving of $T_1$ and $T_2$ for concurrent execution leads to a serializable schedule or non serializable schedule. Explain?

Ans:

Two or more actions are said to be in Conflict if:
1) The actions belong to different transactions.
2) At least one of the actions is a write operation.
3) The actions access the same object (read or write).

The schedules $S_1$ and $S_2$ are said to be Conflict-equivalent if the following conditions are satisfied:
1) Both schedules $S_1$ and $S_2$ involve the same set of transactions (including ordering of actions within each transaction).
2) The order of each pair of conflicting actions in $S_1$ and $S_2$ are the same.

→ A schedule is said to be Conflict-serializable when the schedule is Conflict-equivalent to one or more serial schedules. In the given scenario, there are two possible serial schedules.
1) $T_1$ followed by $T_2$.
2) $T_2$ followed by $T_1$.

In both of the serial schedules, one of the transactions reads the value written by other transaction as a first step.

Therefore, any non-serial interleaving of $T_1$ and $T_2$ will not be Conflict serializable.

=:

3. Define decomposition and how does it address redundancy? Discuss the problems that may be caused by the use of decompositions?

Ans:

## Decomposition of a Relation:-

The process of breaking up or dividing a single relation into two or more sub relations is called as decomposition of a relation.

(or)

A relation R can be decomposed into a collection of relation schemas to eliminate some of the anomalies in original relation R. This is known as decomposition.

Eg:- Let us take a relation [R of X, Y, Z], R(X, Y, Z)

there can be two ~~sets~~subsets

$R_1(X, Y)$; $R_2(Y, Z)$

Now, we get $R = R_1 \cup R_2$

## Problems with decomposition:-

The problem here is the corresponding information may not be able to get original relation while constructing $R_1 \cup R_2$.

Eg:- Let R be relation with

| Model no | price | Category |
|---|---|---|
| $a_{11}$ | 100 | Canon |
| $s_{20}$ | 200 | nikon |
| $a_{20}$ | 150 | Canon |

→ split it into two relations $R_1$ and $R_2$.

$R_1$ = modelno

$R_2$ = price, category

$R = R_1 \cup R_2$

| modelno | price | Category |
|---|---|---|
| $a_{11}$ | 100 | Canon |
| $a_{11}$ | 150 | Canon |
| $s_{20}$ | 200 | nikon |
| $a_{20}$ | 100 | Canon |
| $a_{20}$ | 150 | Canon |

1) Loss-less Join decomposition :-

Let R is a relation and has a set of FD's (Functional dependencies) 'F' over R.

→ The decomposition of R into $R_1$ and $R_2$ is loss-less coRt F if $R_1 \bowtie R_2 = R$.

2) Lossy decomposition :- In this, it contains extra tuples than the Riginal relation. In that case, we are lossing some important information.

→ R(A, B, C) decomposed into $R_1$(A, B) and $R_2$(B, C). check whether it is loss-less or lossy?

$R_1 \cap R_2 = (B)$ → Common attribute but not a key in either $R_1$ or $R_2$.

∴ It is lossy.

=: