UNIT - 3

Normalization →

Design guideline for relational schemas :—

1. Reducing the redundant value in tupples
2. Reducing the null values in tupples
3. Disallow the possibility of generating spurious (duplicate) tupples
4. Semantics of attributes

The aim is to focus on redundancy & consistency

normalisation → It is a process of decomposing a set of relations with anomalies to produce smaller and well structured relations that contain minimum or no redundancy It is the formal process of deciding which attribute should be grouped together in a relation using normalisation any change to the value stored in DB can be achived with the fewest possible update operations.

During normalisation it is ensure that anomalize schemas

1. Does not lose any inf. present in the unnormalize schemas
2. Does not include spurious inf. when the original schema is reconstructed
3. Preserves dependencies present in the original schemas

Normal form → It is a state of relation that results from applys simple rules regarding

functional dependencies to that relation

* Relation is defined as containing two components
  1. Attributes
  2. functional dependencies (fd) b/w them

$R_1 = (\{x, y, z\}, \{x \to y, x \to z\})$

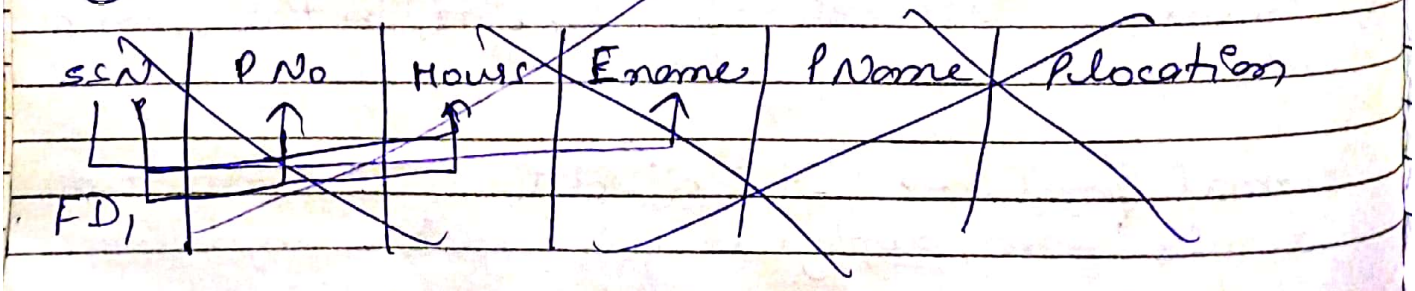
Functional Dependency →

It is a constraints b/w two sets of attributes from dB. ~~functional dep~~

Fd $x \to y$, b/w two set of attributes x,y that are subset of R species a constraint on the ~~poss~~ possible tupple that can form a relation state r of R

Value of y component of a tupple in r depends on, or are determined by the value of x component.
Can be written as, value of x component of a tupple uniquely determined the value of the Y component.

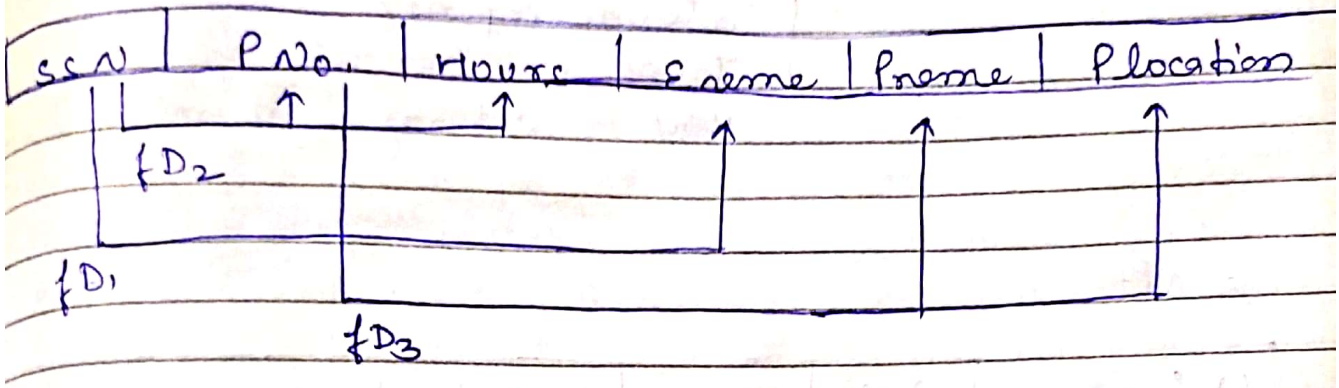
1. fd on b/w to x
2. fd on x to y



fd_1 → value of social security no. can be uniquely identify Employee name and vice versa.

SSN → Ename

fD_2 → {SSN, project no.} → Hours



fD_3 → P.No → {P.name, P.location}

find an employee whose salary is greater than 25,000

Character function

1. Case manipulation function → upper case, lower case, Initcase
2. character "

length, Instr, Lpad, Rpad, Trim, Replace

1st Normal form

Emp	Age	Dept
Peter	33	
Alex	35	
donald	40	
	33	

A table or the relation is in first NF if the values in the domain of each attribute of relation are atomic. Only one value is associated with each attribute the value is not a set of values or a list of values a relation is in first NF. If it doesnot have any repetative grp.

It disallow multivalue, composite & their combination attributes

2nd NF → A table must be in 1st NF ~~be~~ is and no attributes of the table should be functionally dependent on only one part of a concatinated primary key for eg. there is odrd table

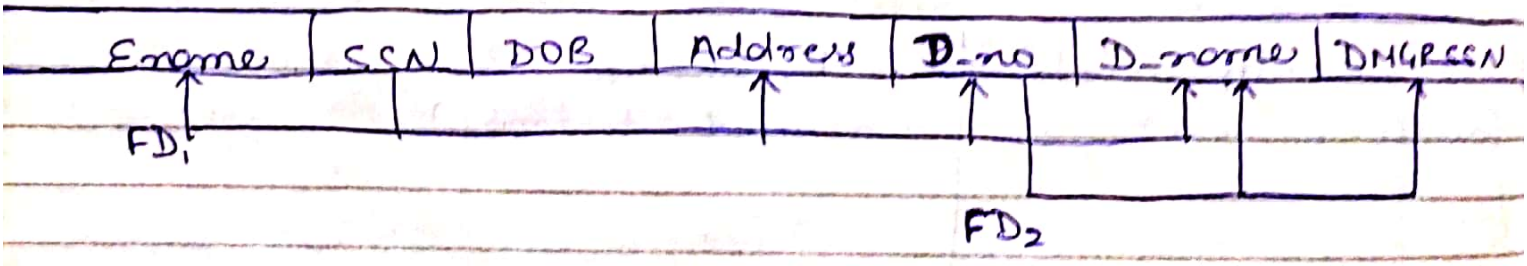
Oder no.	Book-name	no. of Books	Book-Price
1	CN	1	250
1	Graphics	1	275
1	DBMS	2	295
2	Multimedia	1	300
2	Distributed S	1	190
3	DBMS	1	295
3	Multimedia	2	300
3	CN	5	250

II nd normal form based on fully functional dependency. A functional dependency $x \to y$ is the full fd if removal of any attribute a from x, means that the dependency does not hold any more.

A fD $x \to y$ is a partial dependency if some attribute a belong to x ie $a \in x$ can be remove from x in a dependency still holds.

III rd normal form → It is based on transitive dependency, $x \to z$, $z \to y$

a fD $x \to y$ in the relation scheme R is if there is set of attributes z ie neither a candidate key nor a subset of any key of R and both $x \to z$ & $z \to y$ holds.



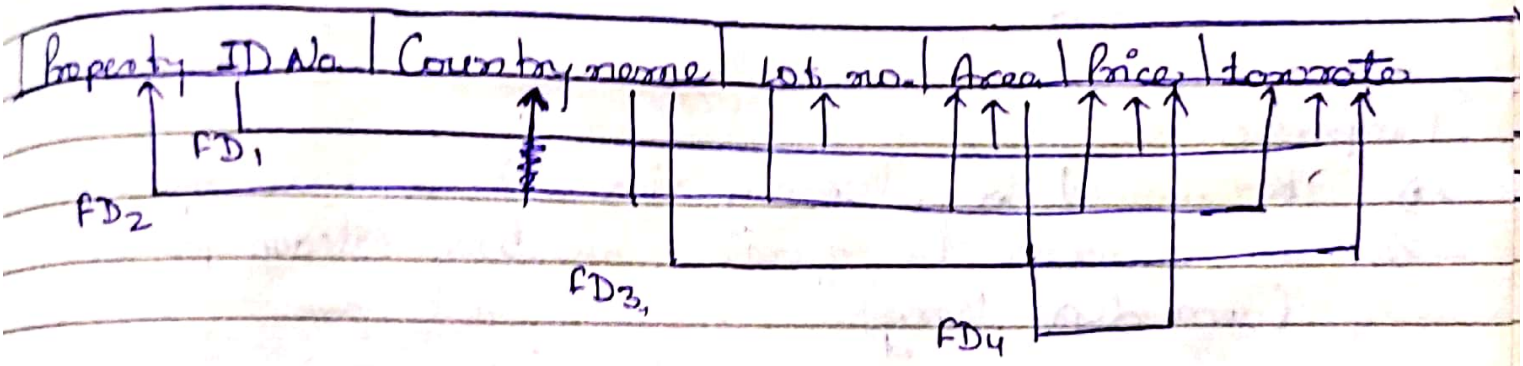
X = SSN
Z = DMGRSSN
Y = DNO.

SSN → DNO
DNO. → DMGRSSN
SSN → DMGRSSN

2nd & 3rd

According to Codd's definition a relation R is in 3rd normal form if it satisfies 2nd normal form & no non prime attributes of R is transitively dependent on primary key.

A relation schema R is in 3rd normal form if whenever a non trivial functional dependency $X \rightarrow A$ holds in R, either
a) X is superkey of R or
b) A is a prime attribute of R



Multivalue Dependency & 4th normal form

MVD is FD where dependency may be to a set and not just a single value is defined as $X \rightarrow\rightarrow Y$, relation R(X, Y, Z) if each X value is associated with a set of Y values in a way that does not depend in Z values & Y both are subset of R a relation is said to be in 4NF if it is in BCNF (3rd NF) & for every non trivial MVD $X \rightarrow Y$, X is superkey of R.

Assignment → 5th NF

5th Normal Form / Project Normal Form

A relation R is in 5th normal form if and only if every join dependency in R is implied by the candidate key of R.

A relation decomposed into two relation must have LOSS-LESS join property which ensures that no spurious or extra tupples are generated, when relations are reunited through natural join

Properties :—

1) R should be already in 4th normal form
2) It cannot be further no loss decomposed (join dependency)

for eg:

Agent	Company	Product
A_1	PQR	nut
A_1	PQR	Bolt
A_1	XYZ	nut
A_1	XYZ	Bolt
A_2	PQR	nut

This relation will decompose into three relations

Agent	Company
A_1	PQR
A_1	XYZ
A_2	PQR

Agent	Product
A_1	nut
A_1	Bolt
A_2	nut

Company	Product
PQR	nut
PQR	Bolt
XYZ	nut
XYZ	Bolt

Acid properties :—

Atomicity Consistency Isolation Durability

Active — the initial state, transaction stays in this state when it is executing.

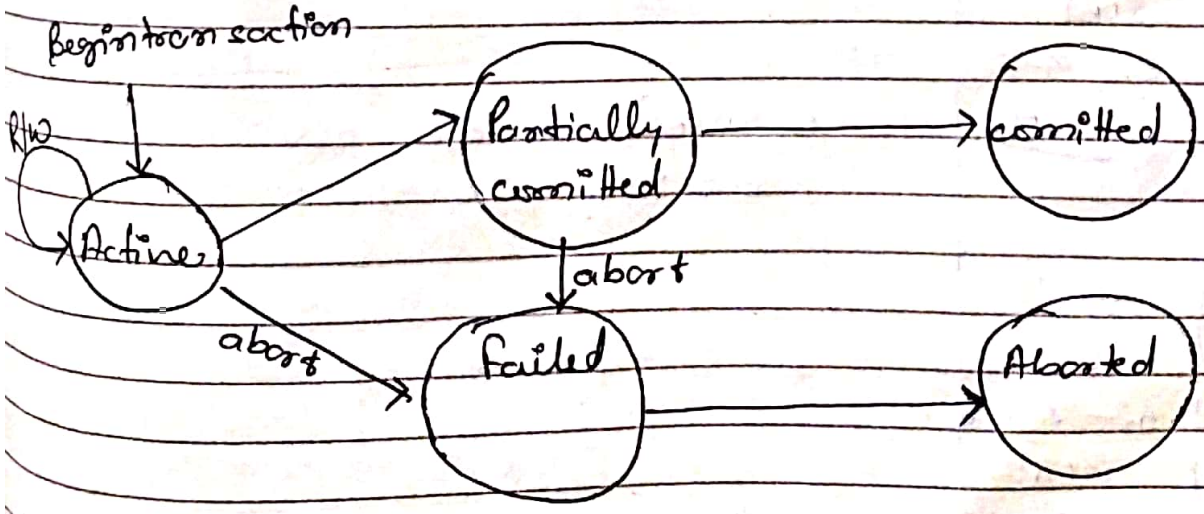
Partially Comitted — after the final statement has been executed.

Failed — after the discovery that normal execution can no longer proceed.

Aborted — After the transaction has been rolled back and the database to restored to it's state prior to the start of transaction. two options after it has been aborted

- restart the transaction :—
  — can be done if no logical internal error
- kill the transaction

Comitted after successful completion



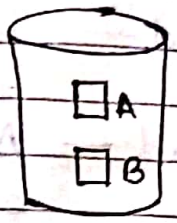
Lost update

T_1	T_2
R(A)	
W(A)	
	W(A)
!	C
!	
C	

Dirty Read Problem

T_1	T_2
R(A)	
W(A)	
	W(A)
!	C
!	
failure	

T_1	T_2	
r(A)	w(A)	conflicting
w(A)	r(A)	
w(A)	w(A)	non conflicting
r(A)	w(B)	
r(B)	w(A)	
w(B)	r(B)	



A = 100 ; B = 200

T_1	T_2	
r(A)		A = 250, B = 190
A := A * 50		
w(A)		
	r(B)	
	B := B - 10	
	w(B)	B(190+50) or B = 250
r(B)		B = 240
B := B * 50		
w(B)		
	r(A)	A = 50 + 10 = 60
	A := A + 10	
	w(A)	

not equivalent and ~~not~~ conflicting

View serializable

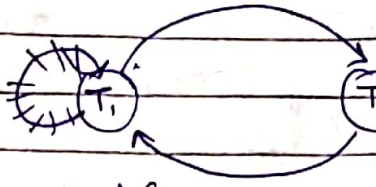
Q- Consider two schedule S & S' where the same set of transaction participates in both schedules. the schedule S & S' are said to be view equivalent if the following three conditions met.

① Initial Read :- for each data item Q (a,b) if transition Ti reads the initial value of Q in schedule S then transaction Ti must in schedule S', also reads the initial value of Q

② Final write :- for each data item Q if transaction Ti executes read Q in schedule S, and if that value ~~also~~ was produced via write Q operation executed by transition Tj, then the read Q operation of transition Ti must, in schedule S', also read the value of Q that were produced by the same write Q operation of transaction Tj.

③ Update Read :- for each data item Q the transaction that perform the final write Q operation in schedule S must perform the final write Q operation in schedule S'

~~Note~~ 99



cycle
∴ non conflict

if conflict then view serizable
if not conflict then may or may not view

* Log based recovery :—
The structure for recording db modifications is called log, it is the sequence of log records recording all the updates, activites in the database an update log record describes a single db wright. It has the following fields :—

① Transaction identifier → It is unique identifier of transiction that perform the wright operation
② Date item identifier → It is a location on disk of date item
③ Old value → Value of date item prior to the right
④ New value → value that the date item will have after the right here the various types of log records

① <Ti start> — Transaction
② <Ti, Xj, Vi, V2>
③ <Ti commit>
④ <Ti abort>

In recovery methods we use two methods
Undo (Ti) & Redo (Ti)

① Undo (Ti) → restores value of all data items updated by transiction Ti to the old value

② Redo (Ti) → sets the value of all data items updated by transiction Ti to the new value.

we need to
undo a transaction T only when log contains the record <T start> for eg.

T_0	T_1
Read (A)	
A = A - 50	
write (A)	
Read (B)	
B = B + 50	
write B	
	Read (C)
	C = C - 100
	write (C)

$R_A = 1000$
$R_B = 2000$
$W_A = 950$
$W_B = 2050$
$C = 600$

Deferred Database modification →
It ensures transaction atomicity by recording all db modifications in the log but deferring (postpone) execution of all write operation of transaction until the transaction partially commit

<T_1 start>
<T_1 C 600>
<T_1 commit>

<T_0 start>
<T_0 A 950>
<T_0 B 2050>
<T_0 commit>

Write ~~sql~~ SQL query for following statements.

1. write a command for selecting all tables from db
2. find average salary of x employee from employee table
3. find employee name whose first name starts with L and end with P
4. Select the maximum salary of those employees whose name starts from R

1. Command to show all tables in database are
   (i) Select tablename from dba_tables
   (ii) show tables
   (iii) eg ~~select table~~

   eg select employee from all tables

2. Select AVG(salary), COUNT(*) from employee

   eg select AVG(salary), count(*) from x

3.