Unit-3

Relational Database Design

Data Normalization & Functional Dependency

A relation in a relational database is based on a relation schema, which consists of a no. of attributes. A relational Database is made up of a no. of relational schemas. In this unit, the major focus is on the issues involved in the design of a database schema using the relational model.

Main Objective of this chapter is to design a good relational schema. Normally, the relational schemas face some undesirable problems such as redundancy & various types of anamolies (Insertion, Updation & Deletion) which (inconsistencies) constitute a bad database design.

Consider one relational schema Supplier (Sname, address, Item, Price)

Supplier

Sname	address	Item	Price
Havell's India Ltd	Civil Lines	Cable	400
Havell's India Ltd	Civil Lines	Switch	300
Bajaj Ltd.	South Ext, Part I	fan	450
Bajaj Ltd.	South Ext, Part I	Switchgear	500
⋮	⋮	⋮	⋮

Here, we can see several problems associated with this relation schema which prove inadequate:

i) Redundancy The aim of the database system is to reduce redundancy meaning that information is to be stored only once, Storing information several times leads to wastage of storage space and an increase in the total size of the data stored. In the above table

relation schema, the address of the supplies are repeated several times for each data item supplied.

ii) Update anamolies - Multiple copies of the same fact may lead to update anamolies or inconsistencies, when an update is made and only some of the multiple copies are updated. Thus, a change in the addres of supplier must be made for consistency; in all the tuples pertaining to the Sname: If one of them is not changed, there will be an inconsistency in the data.

iii) Insertion anamolies -
If this is the only relation in the database showing the association between a Sname and items supplied the fact that we cannot insert an item in the database unless a supplier name is entered in the ~~database~~ relation.

iv) Deletion anamolies
If this is the only relation showing the association between Sname & item supplied, the fact that which item is supplied by which supplier will be lost.

In the relational model, the above problem can be overcome by the process called Decomposition.

Decomposition (Definition) — The Decomposition of a relational schema $R = (A_1, A_2, \ldots A_n)$ is its replacement by a set of relational schemas

$\{R_1, R_2, \ldots R_m\}$ such that $R_i \subseteq R$ for $i = 1$ to m & $R_1 \cup R_2 \cup \ldots \cup R_m = R$

Hence, the problem in the Supplies$^{(R)}$ schema can be resolved only if we can replace it with the following relational schemas:-

SA (Sname, address) - R_1
SIP (Sname, item, price) - R_2

The first schema SA gives the address of each supplier exactly once, hence there is no redundancy.

We can insert an address for a supplier even if it currently has supplies no items.

Similar concepts will be applied to SIP.

where, Supplier = SA $\cup$ SIP &
SA $\subseteq$ Supplier, SIP $\subseteq$ Supplier.

Functional Dependency

- It is a constraint between two sets of attributes from the database.
- A functional dependency (FD) denoted by $X \rightarrow Y$, between two sets of attributes X & Y that are subsets of R specifies a constraint on the possible tuples that can form a relation state r of R. The constraint is that, for any two tuples t_1 & t_2 in r that have $t_1[X] = t_2[X]$, then we must have $t_1[Y] = t_2[Y]$

This means that the values of the Y component of a tuple depend on the values of the X component

or alternately, the values of X-component of a tuple uniquely or functionally determine the value of Y-component.

The FD $X \rightarrow Y$

Y is said to be functionally dependent on X

or

X functionally determines Y.

X is called left hand side attribute also called determinant &
Y " " right " " " "

From the above relation supplier, the following FDs occurs -

Sname → address

Sname, Item → Price.

In other word, whenever 2 tuples of R agree in their X-values, they also agree on their Y-values.

Consider another example..

S

S#	Sname	Status	City
S1	Smith	20	London
S2	Jones	10	Paris
S3	Joe	30	Paris

The following FDs holds:

S# → Sname

S# → Status

S# → Status

but

City ↛ S#.

eg 2

(a) SSN → Ename, specify that the value of an employee's social security no uniquely determine ename.

(b) Pnumber → {Pname, Plocation}

The value of Pnumber uniquely determine the pname & location

(c) {SSN, Pnumber} → hours

a combination of SSN & Pnumber values uniquely determines the no. of hrs.

Prime Attribute - An attribute is said to be a prime attribute if it is the part of the candidate key. Otherwise, it is called non-prime attribute.

e.g. Student (Rollno, name, address, subject)
here, Rollno is the prime attribute & name, address & subject are called non-prime attribute; because Rollno is the P. key & FD's are Rollno → name
Rollno → address &
Rollno → subject.

SPJ (Sno, Pno, Jno, Qty)
PK
here Sno, Pno, Jno are the prime attribute.
Qty is non-prime attribute as the FD for this is
Sno, Pno, Jno → qty

Different types of functional dependency:

i) Trivial FD: A FD in the form $X \to Y$ is trivial if $Y \subseteq X$

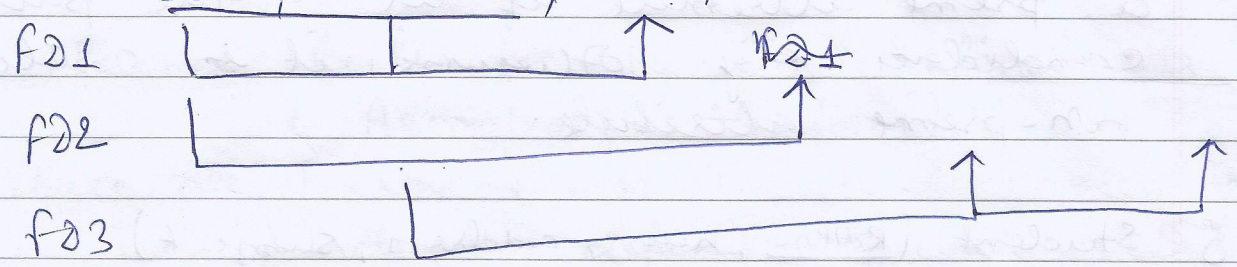
ii) Fully functional dependency
A functional dependency $X \to Y$ is a full functional dependency if removal of any attribute A from X means that the dependency doesn't hold any more.
i.e. for any $A \in X$,
$(X - \{A\})$ doesn't functionally determines Y.

e.g. Student (Rollno, name, address, subject) is an FD Rollno → name is fully FD. But in SPJ (Sno, Pno, Jno, Qty) is an FD
Sno, Pno → qty doesn't hold as Qty is not fully dependent on the primary key which is Sno, Pno & Jno.

Note eg.

EMP-PROJ

SSN, Pnumber, hours, ename, pname, Plocation



fd1: SSN, PNumber → hours
is fully FD.

fd2: SSN → ename
not fully FD because the P key is the combination of SSN & Pnumber.

fd3: Pnumber → Pname, plocation
not fully FD. - same reason.

iii) Partial functional dependency

A FD $X \rightarrow Y$ is a partial dependent if some attribute $A \in X$, can be removed from X and the dependency still holds.

eg. In the above example.

fd2 SSN → ename is partial dependency as SSN is part of the P. key.

fd4 SSN, Pnumber → ename is also partial dependency as SSN → ename holds.

iv) Transitive FD –

A FD $X \rightarrow Y$ in a relational schema R is transitive dependency if there is a set of attributes Z that is neither a candidate

Key not a subset of any key of R & both $X \to Z$ & $Z \to Y$ holds.

e.g.

1. Given $R(A, B, C, D, E)$ & the FDs $F = \{A \to C, B \to D, C \to E, B \to C\}$ then from the above FDs. as $A \to C$ & $C \to E$ then $A \to E$ i.e. E is transitively dependent on A.

2. Prof(Prof-name, Deptname, chairperson) & the FDs are
Prof-name → Deptname &
Deptname → chairperson then
Profname → chairperson holds.

Normalization

~~Primary process of ...~~

Normalization is a process of decomposing the given relation schema into a set of relational schemas based on their FDs and primary keys to achieve the desirable properties of
- minimizing data redundancy
- minimizing the insertion, deletion & update anamolies
- Reduces the memory usage

It also provides a set of normal form tests that can be carried out on individual relation schemas so that the relational

database can be normalized to any desired degree.

Inference Rules for Functional Dependencies (Armstrong's Axioms)

- There are following rules that logically implied functionally dependency with a set of FD's F & suppose X, Y & Z denotes sets of attributes over a relational schema R.
- also called Armstrong's axioms.

We use the notation $F \models X \to Y$ i.e. $X \to Y$ is inferred from F.

R1. Reflexivity rule

If X is a set of attributes & $Y \subseteq X$, then $X \to Y$ holds if $F \models X \to Y$

R2. Augmentation rule

If $X \to Y$ holds & Z is a set of attributes, then $XZ \to YZ$ or $F \models XZ \to YZ$

R3 Transitivity Rule.

If $X \to Y$ & $Y \to Z$ then $X \to Z$ holds. i.e. $F \models X \to Y$

R4. Union or Additive Rule

If $X \to Z$ & $X \to Y$, then $X \to YZ$ holds. or $F \models X \to YZ$

R5 Decomposition or Projective Rule

If $X \to YZ$ holds then $X \to Y$ holds & $X \to Z$ holds.

or $F \models (X \to Y, X \to Z)$

Contd...

Inference rules for FDs (Functional Dependencies)

R_6 Pseudo-transitivity rule

If $X \to Y$ holds & $WY \to Z$ holds
then $WX \to Z$ holds.

Example: Prove the following:

1) $X \to Y$ & $Z \subseteq Y \models \{X \to Z\}$

Proof: i) $X \to Y$ given.
ii) As $Z \subseteq Y \Rightarrow Y \to Z$ (R1, Reflexivity)
iii) $X \to Y, Y \to Z \models X \to Z$ (Transitivity R3)

2) $\{W \to Y, X \to Z\} \models \{WX \to Y\}$

i) $W \to Y$ given
$\Rightarrow WX \to YX$ (R2, Augmentation)
$YX \to Y$ as $Y \subseteq YX$ (R1, Reflexivity)
$\therefore WX \to Y$ (Transitivity R3).

classmate
Date
Page

Normal forms

The normal forms are used to ensure that the various types of anamdies & inconsist-entries are not introduced into the databas. For determing whether a particular relation is in normal forms or not, the fDs between the attributes in the relation are examined & not the current contents of the relation. Various types of normal forms are used in relational databas are as follows:

1. first normal form (1NF)
2. Second " " (2NF)
3. Theird " " (3NF)
4. Boyce-Codd " " (BCNF)
5. Fourth " " (4NF)
6. Fifth " " (5NF).

A relational schema is said to be in a particular relax normal forms if it satisfies a certain prescribed set of conditions as in discuss different normal form. To explain the various normal forms, first consider an example of Unnormalized relation.

P

Prjno	Prjname	Eno	Ename	Category	hrate
P001	FIS	E001	Smith	A	700
"	"	E002	Robin	B	500
P002	EIS	E006	John	A	700
"	"	E007	Joe	B	300

fig (a)

Here, this unnormalized rel^n P contains non-atomic values.

(i) First normal form (1NF) - A relation is said to be in 1NF, if the values in the domain of each attribute of the relation are atomic i.e. only one value is associated with each attribute and the value is not a set of values or list of values.

Hence, the above relation P can be normalized into a relation (1NF)

ex-1 P

Prjno	Prjname	Eno	Ename	Category	hrsals
P001	FIS	E001	Smith	A	700
P001	FIS	E002	Robin	B	500
P002	EIS	E006	John	A	700
P002	EIS	E007	Joe	B	300

info Emp-proj (Prjno, Prjname) PK & Emp (eno, Prjno, ename, Category, hrsals) PK

A table is in 1NF if
- there are no repeating groups.
- all the key attributes are defined
- all attributes are dependent on a Primary key.

example 2.

(a) Department

Dname	Dno	Dmgr-SSN	Dloc

not normalized

(b)

Dname	Dno	Dmgr-SSN	Dloc
Research	5	11325	(Delhi, Banglore, Noida) → Multivalued
Admin	4	52325	(Delhi, Chennai) → Multivalued.
IT	1	23432	Noida

→

State of Relation Department

Date to 1NF

(c) Department (with P. Key Dno + Dloc)

Dname	Dno	Dmgr-ssn	Dloc	
Research	5	11325	Delhi	atomic value
Research	5	11325	Banglore	
Research	5	11325	Noida	
Admin	4	52325	Delhi	atomic value
Admin	4	52325	Chennai	
IT	1	23432	Noida	

8. Normalized the relation EMP-Proj into 1NF.

EMP-Proj (SSN , ename , Pno , hours).

↓ PK

EMP-Proj 1 (SSN , ename) &
EMP-Proj 2 (SSN , Pno , hours)
PK

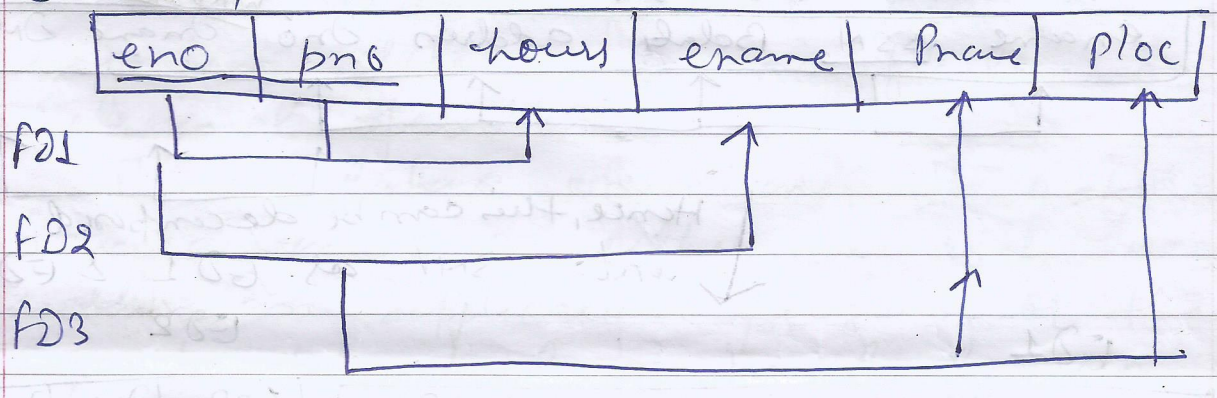
classmate
Date
Page

(2) Seeend Normal form (2NF)

- The 2NF is based on the Coreept of full functional dependency.
- A Relation is in 2NF if it is in 1NF and every non-prime attributes are fully functionally dependent on the Primary key of R.

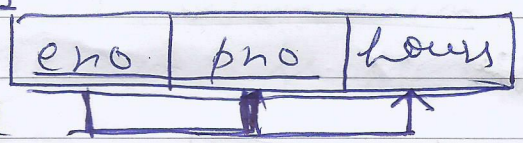
eg:

EMP-Proj

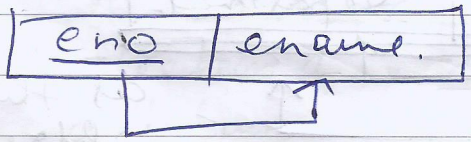


can be decomposed into 3 ~~tab~~ relations as;

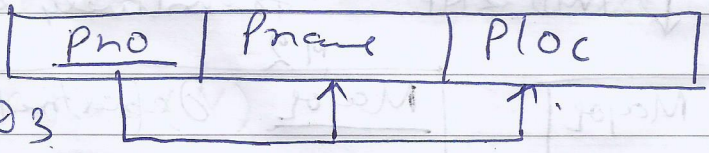
EP1



EP2



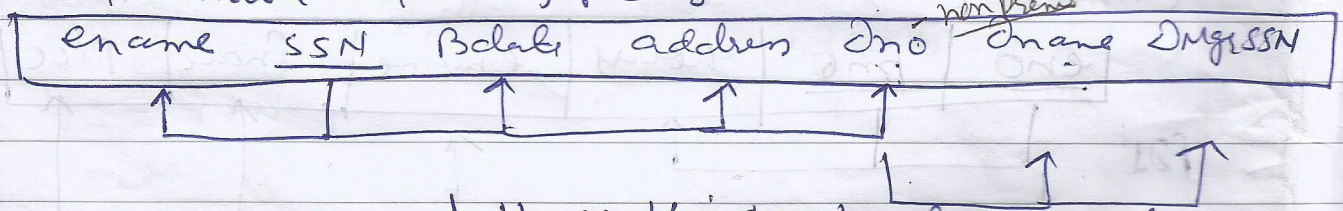
EP3



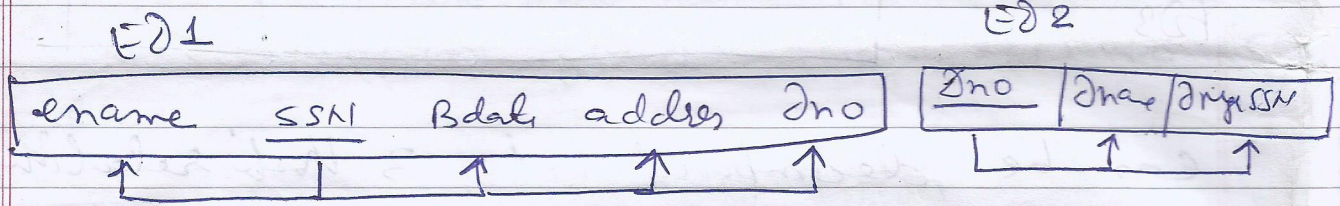
(3) Third Normal form (3NF).

- 3NF is based on the concept of transitive dependency.
- A relation schema R is in 3NF if it is in 2NF and no non-prime attribute of R is transitively dependent on the primary key.

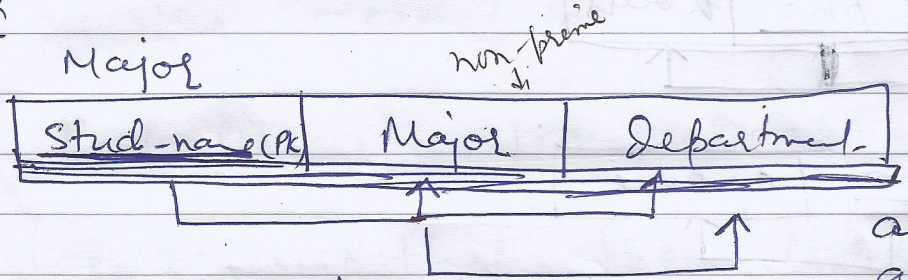
(1) EMP_dept - is not in 3NF because of the transitive dependency of ∂MgrSSN on SSN via ∂no.



Hence, this can be decomposed into 3NF as ED1 & ED2.



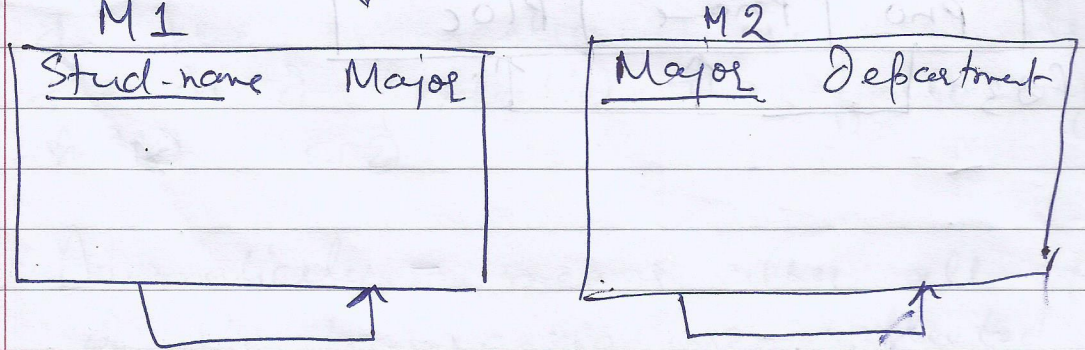
E.g.2



as the non-prime attribute Major is transitively dependent on R.

∞

into 3NF



∞ i.e Stud-name → Major, Major → Department ⇒ Stud-name → Department should n't occurs.