UNIT - IV
NORMALIZATION

Schema Refinement:-

Schema Refinement is based on decomposition. The problems that are created by redundant information are listed below. Eventhough the decomposition can eliminate redundancy, it can lead to several problems and should be used carefully.

Problems caused by Redundancy:-

Redundancy is the method of storing the same information redundantly i.e., in more than one place within a database and it can lead to several problems.

* Redundant Storage:- Some information is stored repeatedly
* Update Anomalies:- If one copy of such repeated data is updated an inconsistency is created unless all copies are similarly updated.
* Insertion Anomalies:- It may not be possible to some certain information unless some other unrelated information is stored as well.
* Deletion Anomalies:- It may not be possible to delete certain information without losing some other, unrelated information as well.

Eg:- Consider an Hourly-Employee relation.

Hourly_Employee (eno, ename, salary, rating, hourly-wages, hours_worked).

In the above relation eno is the key attribute.

In addition the hourly-wages is determined by rating attribute i.e., For a given rating value there is only one hourly-wages value. It leads to redundancy in the relation.

Eno	Ename	Salary	rating	hourly_wages	hours_worked
101	A	2000	8	10	40
102	B	5000	8	10	30
103	C	7000	5	7	30
104	D	3500	5	7	32
105	E	7500	8	10	40

Here redundancy is nothing but for a given rating value there is corresponding hourly_wages value.

Redundant Storage:- The rating of 8 corresponds to hourly wages 10, and it is repeated three times.

Update Anomalies:- The hourly wages in the first tuple could be updated without making a similar change in the second tuple.

Insertion Anomalies:- We cannot insert a tuple for an employee unless we know the hourly wage for the employee's rating value.

Deletion Anomalies:- If we delete all tuples with a given rating value, we lose the association between the rating value and its hourly-wages value.

Decomposition:-

Generally redundancy arises when a relational schema enforces an association between two or more attributes. The problems that are created by redundan are solved by replacing a relation with collection of smaller relations. ②

Definition:- A decomposition of relation schema 'R' consists of replacing the relation schema by two or more relation schemas that each contain a subset of the attributes of R and together include all attributes in R

(or)

A relational schema R can be decomposed into a collection of relation schemas $\{R_1, R_2, \ldots, R_m\}$ to eliminate some of anomalies caused by the redundancy in the original relation R such that the relation schemas $R_i \subseteq R$ for $1 \leq i \leq m$ and $R_1 \cup R_2 \ldots \cup R_m = R$.

In simple words "The process of breaking larger relations into smaller relations is known as Decomposition

Eg:- Hourly-Employee(eno, ename, salary, rating, hourly-wages, hours worked) broken into

Hourly-Employee1(eno, ename, salary, rating, hoursworked) and wages (rating, hourly-wages)

As a result, updating any one hourly wages tuples associated with a rating in wages relation involves updating several tuples in hourly Employee1.

Drawbacks or Problems caused by Decomposition:-

A serious drawback of decompositions is that, the original relation may require us to join the decomposed relation. If such queries are many, the performance will be degraded, in this case the decomposition of relation is strictly not acceptable. In this case we allow some of the problems created by redundancy but not to decompose the relation.

Properties of Decomposition:-

Decomposition is the tool that allows us to eliminate redundancy. So it is very important to check that a decomposition does not introduce new problems. In particular we should check, a decomposition allows us to recover the original relation from the decomposed smaller relation known as "Lossless Join decomposition" property and whether it allows us to check integrity constraints efficiently known as "dependancy preserving decomposition" property.

Functional Dependancy:-

A functional dependancy (FD) is a constraint between two sets of attributes from the database. For example if X and Y are attributes of a relation 'R'. Y is functionally dependant on X (denoted as $X \rightarrow Y$) iff when ever two ~~types~~ tuples of relation R agree on their X value, they also agree on their Y value. (Each value of X is associated with exactly one value of Y in R).

$X \rightarrow Y$ read as X functionally determines Y or Y is functionally dependant on X. Here X is called determinant and Y is called dependent.

Ex:-

eno	ename	deptno	salary
101	Sree	10	12000
102	Ram	10	15000
103	Sravan	20	10000
104	Ajay	20	8000
105	Sri	30	7000

In the above relation ename, deptno, salary are functionally dependent on eno. Thus eno → ename, eno → deptno, eno → salary.

Ex:- Rollno → Stuname

R(P, Q, S)

P	Q	S
P_1	q_1	S_1
P_2	q_2	S_2
P_1	q_1	S_2
P_1	q_1	S_3

Here P → Q

For each value of Q there is only one value for P.

Eg:- Every vehicle owner possesses a Licence and a unique Licence number. Each distinct Licence number determines a distinct owner. In otherwords the value of licence number determines the owner entity.

Licenceid → Licence owner

But the converse is not hold true. A person could have a Licence for 2 wheeler and for a 4 wheeler. so

Licence owner ↛ Licenceid.

Characteristics of Functional Dependancy:-

X → Y

* X is called determinant of Y.

* X may or may not be the key attribute in R.
* X and Y may be composite attributes.
* X and Y may be mutually dependent on each other.
* The relationship among attributes in the FDs is most of the time 1:1 and sometimes it may be m:1

101 - Sree	101 - Sree
102 - Ram	102 - Ram
	103 - Sree.

* FD must hold for all times i.e., FD is the property of the relational schema and not the property of a particular instance of the schema.

A	$\rightarrow$ B	A	$\not\rightarrow$ b
a_1	b_1	a_1	b_1
a_2	b_2	a_2	b_2
a_3	b_3	a_1	c_1

* FD's are always non-trivial.

Trivial Functional Dependency:-

A FD is said to be trivial if right hand side attributes are subset of the left hand side attributes.

$X \rightarrow Y$ if $Y \subset X \Rightarrow$ Trivial

$AB \rightarrow B$, $B \subset AB$ so Trivial

$ABC \rightarrow BC$ $BC \subset ABC$ so Trivial

$AB \rightarrow CD$ Non-Trivial.

Non_Trivial FD:- If none of the right hand side attributes are available on lefthand side then it is called non trivial.

$A \rightarrow B$

$AB \rightarrow CD$

$ABC \rightarrow DEF$.

Closure of a set of FD's:-

The set of all FD's implied by a given set of FD's is called the closure of F. By semantics we are able to find out some functional dependencies. Later by applying inference rules we are able to find out some of the additional functional dependencies from existing functional dependencies.

1. Reflexive Rule:- If B is a subset of A, $B \subseteq A$ then $A \rightarrow B$.
2. Augmentation:- If $A \rightarrow B$ then $AC \rightarrow BC$ (adding same attribute on both sides of a dependency results in valid dependency).
3. Transitivity Rule:- If $A \rightarrow B$ and $B \rightarrow C$ then $A \rightarrow C$.

From these three rules we can get four more rules.

4. Self determination Rule:- $A \rightarrow A$.
5. Decomposition Rule:- If $A \rightarrow BC$ then $A \rightarrow B$ and $A \rightarrow C$

$A \rightarrow BC$

$BC \rightarrow C$, $BC \rightarrow B$ (Rule 1)

$A \rightarrow C$, $A \rightarrow B$ (Rule 3).

6. Union rule:- If $A \rightarrow B$ and $A \rightarrow C$ then $A \rightarrow BC$.

$A \rightarrow B$, $A \rightarrow C$

$AA \rightarrow AB \Rightarrow A \rightarrow AB$ (adding same attribute)

$AB \rightarrow CB$ (Rule 2)

$\therefore A \rightarrow BC$.

7. Composition rule:- If $A \rightarrow B$ and $C \rightarrow D$ then $AC \rightarrow BD$.

$A \rightarrow B$

$AC \rightarrow BC$ (Rule 2)

$BC \rightarrow BD$ (Rule 2)

$\therefore AC \rightarrow BD$

Armstrong axiom's are sound and complete.

Complete:- Repeated application of these rules will generate all FD's in the closure F^+.

Sound:- Any dependency that is computed using these rules will holds on every relation.

Eg:-① R(ABC) and FD's are $A \rightarrow B$, $B \rightarrow C$.

By applying Rule(3) we get $A \rightarrow C$.

From augmentation we get.

$AC \rightarrow BC$, $AB \rightarrow AC$

then $AB \rightarrow BC$.

② Suppose we are given a relation R(ABCDEF) and the FD's are $A \rightarrow BC$, $B \rightarrow E$, $CD \rightarrow EF$. Now check whether $AD \rightarrow F$ holds or not.

1. $A \rightarrow BC$ (Given)
2. $A \rightarrow C$ (Decomposition Rule)
3. $AD \rightarrow CD$ (Augmentation by adding D)
4. $CD \rightarrow EF$ (given)
5. $AD \rightarrow EF$ (Transitivity of 3 and 4)
6. $AD \rightarrow F$ (Decomposition of 5)

Attribute Closure:-

If we want to check whether a given dependency $X \rightarrow Y$, is in the closure of a set F of FD's then we have to compute the attribute closure X^+ with respec to F, which is the set of attributes A such that $X \rightarrow A$ can be computed using the Armstrong axioms.

Algorithm for Computing attribute closure:-

1. Select attribute or attributes from LHS of FD and equate it to X. (X is the set of attributes that become a closure).
2. Select the FD's one by one. If LHS of FD is sub set of X, if RHS of FD is not in X, then add RHS of FD to X.
3. Repeat step ② till to cover all FD's. Here the result is called as closure of attribute. If the closure covers all the attributes of the relation then it is called as Candidate key.

Eg:- 1. Consider a relation R(ABCD) with following FD's
$AB \to C$, $C \to D$, $D \to A$..

AB^+ $AB \; C \subset AB$ (Add C to AB)
$X = AB$

ABC ($C \subset ABC$ so add D).

$ABCD$

$\therefore AB^+ = ABCD$. So AB is a Candidate key.

$BD^+ = BD$ $BC^+ = BC$
$= BDA$ $= BCD$
$= BDAC$ $= BCDA$

So the candidate keys are AB, BC and BD.

2. Consider a relational schema R(ABCDE) and FD's are $A \to BC$, $CD \to E$, $B \to D$, $E \to A$. Find out A^+ and E^+.

$X = A^+$ $E^+ = EA$ ($E \to A$)
$= ABC$ ($A \to BC$) $= EABC$ ($A \to BC$)
$= ABCD$ ($B \to D$) $= EABCD$ ($B \to D$)
$= ABCDE$ ($CD \to E$)   Candidate keys are BE, A, E, CD, BC

3. Consider $R(ABCDEFGHI)$. FD's are $ABD \to E$, $AB \to G$, $B \to F$, $C \to J$, $CJ \to I$, $G \to H$. Find out a) AB^+ b) CJ^+ c) ABD^+ d) $ABCD^+$ e) $ABCG^+$ f) $ABCJ^+$.

a) $AB^+ = ABG \quad (AB \to G)$
$= ABGH \quad (G \to H)$
$= ABGHF \quad (B \to F)$

b) $CJ^+ = CJ \quad (CJ \to I)$
$= CJI$

c) $ABD^+ = ABD$
$= ABDE \quad (ABD \to E)$
$= ABDEG \quad (AB \to G)$
$= ABDEFG \quad (B \to F)$
$= ABDEFGH \quad (G \to H)$

d) $ABCD^+ = ABCDE \quad (ABD \to E$
$= ABCDEG \quad (AB \to G$
$= ABCDEFG \quad (B \to F$
$= ABCDEFGH \quad (G \to H$
$= ABCDEFGHI \quad (C \to$
$= ABCDEFGHIJ \quad (CJ \to$

(d) $ABCG^+ = ABCFGHIJ$

e) $ABCJ^+ = ABCFGHIJ$

$\therefore$ Candidate key is $ABCD$.

4. Consider a relation $R(ABCDEFG)$ then test the FD $ACF \to DG$ can be derived from existing dependencies

$A \to B$
$BC \to DE$
$AEG \to G$

$ACF^+ = ABCF \quad (A \to B)$
$= ABCDEF \quad (BC \to DE)$

Since all the attributes $(ACFDG)$ are not in ACF^+ so $ACF \to DG$ cannot be derived from existing dependencies.

5. Consider $R(ABCDEFGH)$ and FD's are $A \to BC$, $CD \to E$, $E \to C$, $D \to AEH$, $ABH \to BD$, $DH \to BC$. Find out whether $BCD \to H$ holds or not.

$BCD^+ = BCDE \; (CD \rightarrow E)$

$= ABCDEH \; (D \rightarrow AEH)$

All the attributes (BCDH) are in BCD^+ so it $BCD \rightarrow H$ holc

$AB \rightarrow C$ holds.

Equivalence of FD's:-

Consider two set of FDs A and B. They are said to be equivalent iff $A^+ = B^+$. Equivalent means every FD in A can be inferred from B and every FD in B car be inferred from A. If A is equivalent to B then we can say that A covers B and B covers A.

Eg:- Consider the following sets of FD's

$F = \{A \rightarrow C, AC \rightarrow D, E \rightarrow AD, E \rightarrow H\}$

$G = \{A \rightarrow CD, E \rightarrow AH\}$

First set:-

$A^+ = A$

$= ACD$

So $A \rightarrow CD$ is possible

$AC^+ = AC$

$= ACD$

$AC \rightarrow CD$ is possible

$E \rightarrow AD$

$E^+ = E$

$= EAD$

$= EADC$

$= EADCH$

$E \rightarrow AH$ is possible.

Second set:-

$A \rightarrow CD$

$A^+ = ACD$

So $A \rightarrow C$, $AC \rightarrow D$ is possible

$E^+ = EAH$

$= EAHCD$

$E \rightarrow AD, E \rightarrow H$ is possible.

So F and G are equivalence.

$F = \{A \rightarrow C, AC \rightarrow D, E \rightarrow AD, E \rightarrow H\}$

$G = \{A \rightarrow CD, E \rightarrow A\}$ are not equivalent.

Prime or Key attributes:- The attributes that are part of keys are called as prime or key attributes.

Non-Prime or Non-Key attributes:- The attributes that are not part of keys are called as Non-prime or Non-key attributes.

Eg:- R = ABCDEH, FD's are $A \rightarrow BC$, $CD \rightarrow E$, $E \rightarrow C$, $AH \rightarrow D$.

Prime attributes : AH

Non prime attributes : BCDE.

Types of Functional Dependencies:-

1. **Full Functional Dependency:-** A FD $X \rightarrow Y$ is said to be full FD, if removal of any attribute A from X means that the dependency does not hold any more i.e., for any attribute $A \in X$, $(X - \{A\})$ doesnot functionally determine Y.

2. **Partial Functional Dependency:-**

A FD $X \rightarrow Y$ is partial FD, if some attribute $A \in X$, $(X - \{A\})$ functionally determines Y.

or

A FD in which one or more non key attributes are Functionally dependent on part of the primary key.

Eg:- R(ABCD) and FD's are $AB \rightarrow C$, $B \rightarrow D$.

AB is primary key. Non key attributes are C, D. As D is depending on only B (part of the primary key) then $B \rightarrow D$ is the partial FD.

3. Transitive dependencies:-

If there is a FD between 2 or more non prime attributes then it is called Transitive functional dependency

Eg:- R(ABCDE)

$AB \rightarrow C$, $B \rightarrow D$, $C \rightarrow E$

AB is the key for the relation.

$AB \rightarrow C$ – Full F.D

$B \rightarrow D$ – Partial F.D

$C \rightarrow E$ – Transitive F.D.

Lossless Join Decomposition:-

Let R be a relation schema and let F be a set of FD's over R. A decomposition of R into two schemas with attribute sets X and Y is said to Lossless Join decomposition with respect to F if, for every instance of r of R.

$\pi_X(r) \bowtie \pi_Y(r) = r$. In other words if we recover the original relation from the decomposed relations then it is Lossless Join Decomposition.

Eg:- R(ABC)

A	B	C
a_1	b_1	c_1
a_2	b_2	c_2
a_3	b_1	c_3

We are decomposing R into $R_1(AB)$ and $R_2(BC)$

R_1

A	B
a_1	b_1
a_2	b_2
a_3	b_1

R_2

B	C
b_1	c_1
b_2	c_2
b_1	c_3

$R_1 \bowtie R_2$:-

a_1	b_1	c_1	
a_1	b_1	c_3 ✓	Spurious tuples.
a_2	b_2	c_2	
a_3	b_1	c_1 ✓	
a_3	b_1	c_3	

The above is Lossy Join Decomposition.

(2).

S#	status	city
S_1	10	Banglore
S_4	30	HYD
S_6	20	Chennai

R_1

S#	Status
S_1	10
S_4	30
S_6	20

R_2

S#	city
S_1	Banglore
S_4	Hyd
S_6	chennai

$R_1 \bowtie R_2$

S_1	10	Banglore
S_4	30	Hyd
S_6	20	chennai

The above is example of Lossless Join decompositio

Dependency preserving decomposition:-

Consider a relation R with attributes s' and Set of FD's F. R is decomposed into releations R_1, R_2, ----, R_n with the FD's $F_1, F_2, ---, F_n$. This decomposition of R is dependency preserving if

$$(F_1 \cup F_2 \cup \cdots \cup F_n)^+ = F^+.$$

The dependencies in the original relation can be implied by decomposition relation dependencies.

Eg:- ① $R(ABCD)$ with FD's $F = \{A \rightarrow B, A \rightarrow C, C \rightarrow D\}$, $R_1(ABC)$ with FD's $F_1 = \{A \rightarrow B, A \rightarrow C\}$, $R_2(CD)$ with FD's $F_2 = \{C \rightarrow D\}$

In the above decomposition all the original FD's can be logically derived from F_1 and F_2. Hence the decomposition is dependency preserving.

② $R(ABCD)$, $F = \{A \rightarrow B, A \rightarrow C, A \rightarrow D\}$

$R_1(ABD)$, $F_1 = \{A \rightarrow B, A \rightarrow D\}$, $R_2(BC)$ with no FD's.

The above decomposition is not dependency preserving

Normal Forms:-

Normal forms are rules for structuring relations to eliminate anomalies like modification (update), deletion, addition (insert).

Normalization:-

The process of deciding which attributes should be grouped together in decomposing a given relation into smaller relations. This normalization is based on the primary keys and FD's

(or)

Normalization is a tool to validate and improve the logical design of the database.

(or)

The process of reducing redundancy is called Normalization.

Advantages:-

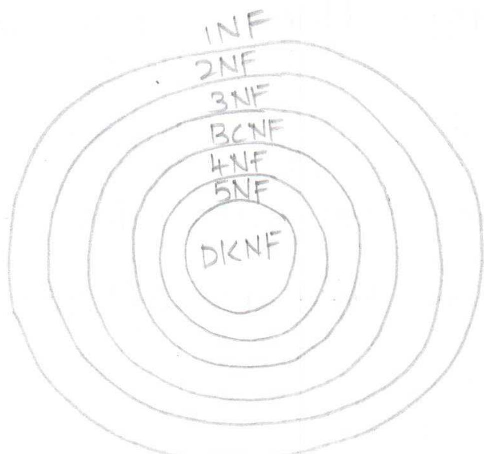
* It eliminates insertion, deletion and update anomalies.

* Obtains database consistency.
* reduces duplication hence disk size.

Disadvantages:-

Retrieving of information from the normalized table is quite complex and consumes lot of time or slow down the process why because we need to join multiple tables.

There are different types of normal forms and the below figure shows the relationship among them.



The above diagram indicates if a relation is in 2NF means then it is already in 1NF. If it is in 3NF, it is already in 1NF, 2NF and so on.

First Normal Form (1NF):-

A relation schema is said to be in 1NF if the values in the domain of each attribute of the relation are atomic. In other words only one value is associated with each attribute and the value is not a set of values or list of values.

To transform the unnormalised table into 1NF we have to identify and remove the repeating group

Repeating group:-

A repeating group is an attribute or group of attributes with in a table that occurs with multiple values for single occurrence.

Eg:-

Fno	Fname	Phone
101	Sree	12481
		12642
		12662
102	Ram	12661
		12741
		12721

The above relation is not in INF.

Fno	Fname	Phone
101	Sree	12481
101	Sree	12642
101	Sree	12662
102	Ram	12661
102	Ram	12741
102	Ram	12721

A relation that satisfies the INF must meet the following requirements.

* The cells of the table must have a single value.
* Neither repeating groups nor arrays are allowed as values.
* All entries in any column must be of the same kind.
* No two rows in a table may be identical.

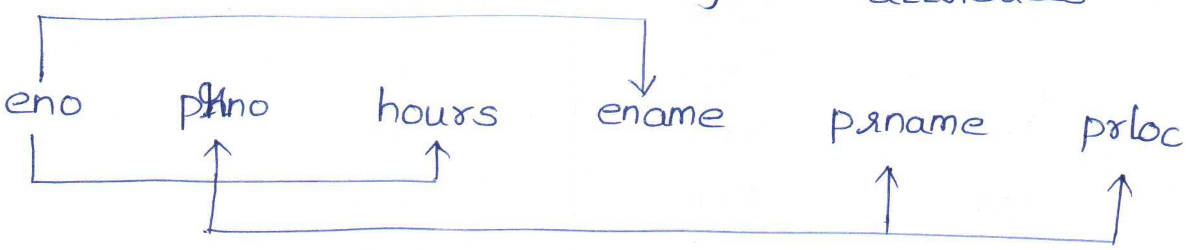
Drawback:- Redundancy of data.

Second Normal Form (2NF):-

A relation is said to be in 2NF, if it is in INF and every non key attribute is fully functionally dependant on the primary key i.e., no attribute is dependent on only a part of the key. 2NF deals with partial dependencie (It eliminates partial dependencies).

A relation is said to be in 2NF, if any one of the condition is satisfied.

* Primary key consists only one attribute.
* If all attributes in a relation are in the primary or non key attributes exists in the table.
* Every nonkey attribute is fully functionally dependant on primary key.
* If the relation consists only two attributes.



FD's are <u>eno prno</u> $\longrightarrow$ hours

<u>eno</u> $\longrightarrow$ ename

prno $\longrightarrow$ prname prloc

The above relation is in 1NF, but not in 2NF. The key attribute for the above relation is enoprno. But ename is depending on part of the primary key (i.e., eno $\longrightarrow$ ename) In the same way prnameprloc is also dependent on part of the primary key (prno). The above relation can be decomposed into smaller relations.

<u>eno prno</u> hours.

<u>eno</u> ename

<u>prno</u> prname prloc

Eg:- ① Consider a relation R(ABCDEFGHIJ) and FD's are $AB \rightarrow C$, $A \rightarrow DE$, $B \rightarrow F$, $F \rightarrow GH$, $D \rightarrow IJ$. What is the key attribute of R and decompose R into 2NF.

AB is the key.

$A \rightarrow DE$ is partial FD

$B \rightarrow F$ is partial FD

$A^+ = ADEIJ$, $B^+ = BFGH$.

$R_1 = A^+$

$R_2 = B^+$

$R_3 = ABC$

② Consider R(ABCDEFGHIJ) and FD's are $AB \rightarrow C$, $BD \rightarrow EF$, $AD \rightarrow GH$, $A \rightarrow I$, $H \rightarrow J$. Find out primary key and normalize upto 2NF.

ABD is the primary key.

$AB^+ = ABCI$ $\quad R_1$

$BD^+ = BDEF$ $\quad R_2$

$AD^+ = ADGHJI$ $\quad R_3$

$A^+ = AI$

$R_1 <$ ABC R_1', AI R_1''

R_2 BDEF

$R_3 <$ ADGHJ R_3', AI R_3''

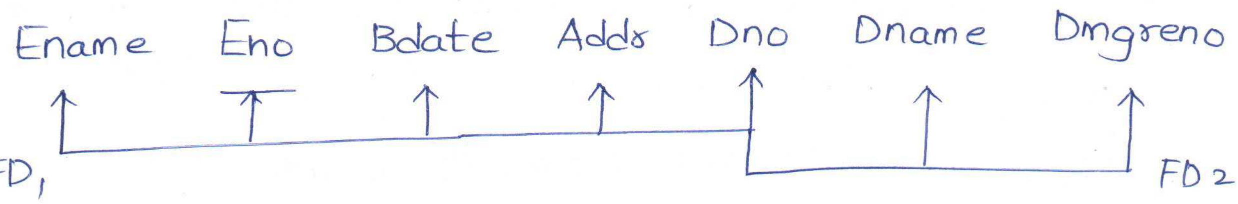
The decomposed relations are ABC, AI, BDEF, ADGHJ, ABD.

Third Normal Form (3NF):-

A relation is said to be in 3NF if it is in 2NF and no transitive dependency exists in the table.

A transitive dependency is the FD between the two non-key attributes.

Eg:- ①



Since, $Eno \rightarrow Dno$
$Dno \rightarrow Dmgreno$

Through transitivity $eno \rightarrow Dmgreno$. It is also true for Dname. Now the above relation is decomposed into 2 relations.

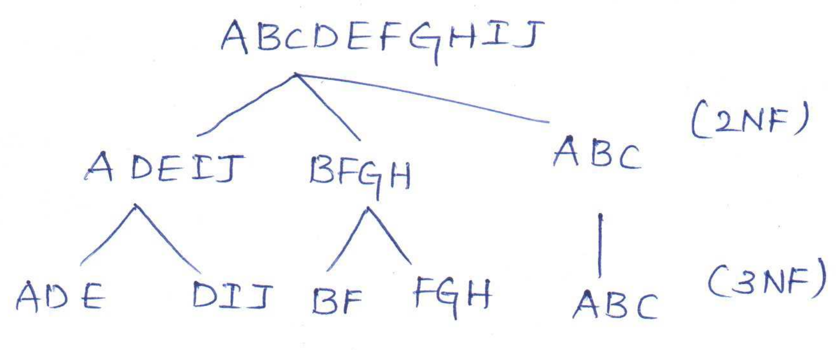
Ename <u>Eno</u> Bdate Addr Dno

<u>Dno</u> Dname Dmgreno.

② Consider R(ABCDEFGHIJ), FD's are $AB \rightarrow C, A \rightarrow DE, B \rightarrow F$
$F \rightarrow GH, D \rightarrow IJ$.

AB is the Key attribute.

$\left.\begin{array}{l} A \rightarrow DE \\ B \rightarrow F \end{array}\right\}$ PFD's $\qquad \left.\begin{array}{l} F \rightarrow GH \\ D \rightarrow IJ \end{array}\right\}$ Transitive FD's

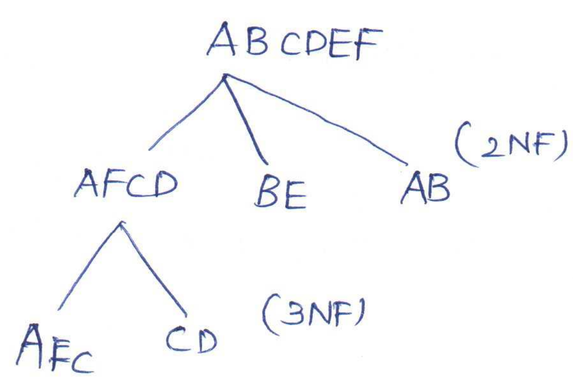


③. R(ABCDEF), FD's are $A \rightarrow FC, C \rightarrow D, B \rightarrow E$.

Key attribute is AB.

$\left.\begin{array}{l} A^+ = AFCD \\ B^+ = BE \end{array}\right\}$ 2NF.

Key attribute -- AB



Boyce-Codd Normal Form (BCNF):-

Database relations are designed so that they have neither partial nor transitive dependencies, because they result in modification anomalies. To avoid these anomalies we normalize table to 2NF to eliminate PDs and to 3NF to eliminate TD's. But we cannot take into account other candidate keys of a relation. Hence even after application of 2NF & 3NF there is a possibility for additional redundancy caused by dependency that violates all candidate keys. so inorder to take care of this weakness in 3NF we are using BCNF and it is based on FD's that take into account all candidate keys. BCNF is stronger than 3NF. Every relation in BCNF is also in 3NF. But a relation in 3NF may not necessarily in BCNF.

3NF is not dealing with the case of a relation with following properties.

* having 2 or more candidate keys.
* Candidate keys overlap i.e., have atleast one attribute in common.

If the above conditions are there in a relation then the relation violates BCNF and causes some redundancy.

Definition:-

A relation is said to be in BCNF iff every (non-trivial irreducible FD has a candidate key as its determinant) determinant is a candidate key.

Eg:- Consider student(Rollno, name, phno, Branch) where Rollno is unique, student identification no and where name, phno assumed to be unique. Then the FD's are.

Rollno → Branch	phno → Branch	name → Branch
Rollno → name	phno → name	name → Rollno
Rollno → phno	phno → Rollno	name → phno.

The relation student is in BCNF. Since each FD involves a candidate key as its determinant.

(2). R(ABCD), FD's are A → B, BC → D, A → C.

Primary key is A. The relation is in 2NF. but not in 3NF (BC → D - T.D)

R_1 = ABC, R_2 = BCD.

Now the relation is in 3NF. Since all determinants are candidate keys it is in BCNF.

If a table contains only one candidate key or only non composite keys then 3NF & BCNF are equivalent.

Multivalued dependency:-

Represents a dependency between attributes (say A, B, C) in a relation such that for each value of A there are set of values for B and a set of values for C. However the set of values for B and C are independent of each other.

The multivalued dependency may be trivial or non-trivial

Trivial MVD:- $A \twoheadrightarrow B$ in a relation R is defined as trivial

a) B is subset of A.

If neither (a) nor (b) is satisfied then it is called as nontrivial MVD.

Fourth Normal Form (4NF):-

A relation is said to be in 4NF iff it is in BCNF and has no multivalued dependency.

The normalization of BCNF relations to 4NF involves the removal of MVD from the relation by placing the attributes in a new relation along with the copy of its determinants.

Eg:- Assume an employee works on more than one project and having more than one dependent.

Ename	Pname	dependent
Sree	x	Ram
	y	Sam

Ename	Pname	dependent
Sree	x	Ram
Sree	x	Sam
Sree	y	Ram
Sree	y	Ram

Ename →→ pname

Ename →→ dependent.

The emp relation is not in 4NF and therefore we decompose emp into

Emp-project(Ename, Pname)

Emp-dependent(Ename, Dname)

Sree	x	Sree	Ram
Sree	y	Sree	Sam