BITS Pilani
Hyderabad Campus

Database Systems (CSF 212)

Prof.R.Gururaj
CS&IS Dept.

Relational Database Design
(Ch. 14 &15 of T1)

innovate achieve lead

Content

- ❑*Introduction to Schema Refinement*
- ❑*Functional Dependencies*
- ❑*Inference Rules*
- ❑*Normalization*
- ❑*Normal Forms*
- ❑*Lossless join decomposition*
- ❑*Dependency preserving decomposition*

Introduction to Database Design

A good database design practice is essential to develop good relational schemas at logical level.

Objectives of good Database design:

- ❑Clarity in understanding the database and
- ❑To formulate good/efficient queries

This is achieved by schema refinement.

Relational Schema design guidelines

Informal guidelines used as measures to determine the quality of relational schemas.

- Design a schema so that it is easy to explain the semantics.
- Design schemas so that no insertion, deletion and update anomalies are present. (anomalies)
- Avoid placing attributes into base relations whose values may frequently be NULL. (wastage of storage)
- Design relations in such a way that they can be joined on keys and no spurious tuples are formed. Avoid relations that contain matching attributes that are not PK, FK pairs. (spurious tuples)

Functional Dependencies

Functional Dependency is a constraint between two sets of attributes from the database.

Function Dependency $X \rightarrow Y$

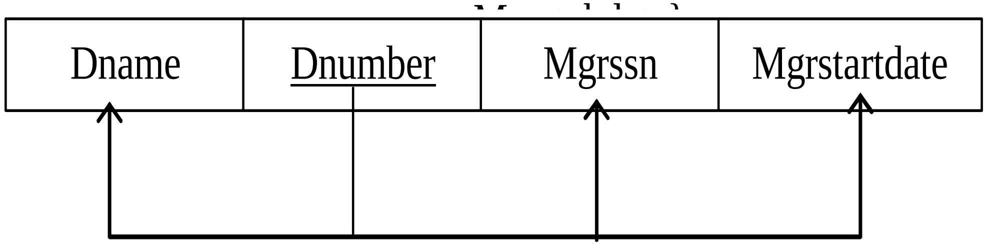
X functionally determines Y in a relation schema R if and only if whenever two tuples of *r*(R) agree on their X values they must necessarily agree on their Y values, but $Y \rightarrow X$ is not true (need not be)

Ex: *ssn* $\rightarrow$ *ename;* *{ssn, pnumber}* $\rightarrow$ *Hours*

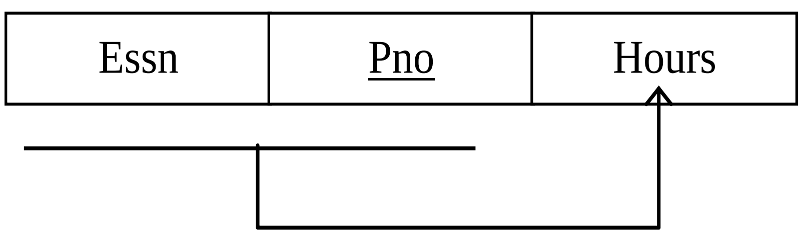
Note: FDs cannot be inferred. They should be defined by someone who knows the semantics of the database very well.

Diagrammatic Notation

Department $\{Dnumber \rightarrow Dname, Mgrssn, mgrstartdate\}$



Work_on $\{Essn, pno\} \rightarrow Hours$



Inference Rules for FDs

Rule 1 $(1R_1)$: (Reflexing)

If $Y \supseteq X$ then $X \rightarrow Y$ otherwise non trivial

Rule 2 $(1R_2)$ (Augumentation)

$X \rightarrow Y$; then $XZ \rightarrow YZ$

Rule 3 $(1R_3)$ (Transitive)

$X \rightarrow Y$; $Y \rightarrow Z$; Then $X \rightarrow Z$;

Rule 4 (IR_4) (Decomposition or projective rule)

$X \rightarrow YZ$ then $X \rightarrow Y$; & $X \rightarrow Z$;

Rule 5 (IR_5) (union rule)

$X \rightarrow A$; $X \rightarrow B$; then $X \rightarrow AB$

Rule 6 (IR_6) (Pseudo transitive)

$X \rightarrow Y$; $WY \rightarrow Z$; then $WX \rightarrow Z$;

We can find the closure F^+ of F, by repeated application of rules IR-1 to IR-3. These rules are called *as Armstrong's Inference rules*.

IR-6

$\{X \rightarrow Y;\ WY \rightarrow Z\} \models WX \rightarrow Z$

$X \rightarrow Y$ (given) -1

$WY \rightarrow Z$ (given) -2

$WX \rightarrow WY$ (use IR2 on 1 add W on both sides)

$WX \rightarrow Z$ (using IR3 ~~and~~ on 3 & 2)

Normalization & Normal forms

Normalization process is first proposed by Raymond *Boyce and Edgar Codd* in 1972.

Normalization of data – is the process of analyzing relation schemas based on their FDs and PKs/Keys to minimize the redundancy

1. First Normal Form (INF)

It states that the domain of any attribute must include only atomic (single / simple/ individual) values.
In the example given below, under the column *Dloc* each row has more than on values.

Ex.: Dept

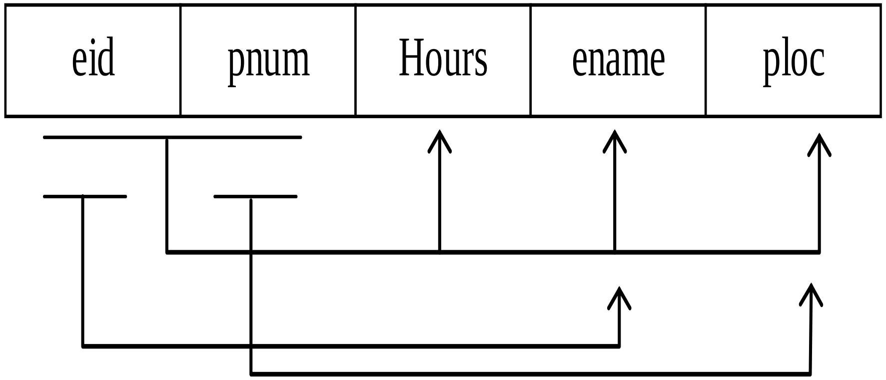
DId	*Dname*	*Dloc*
10	Engg	HYD CHENNAI
20	Mark	HYD MUMBAI

2. Second Normal Form (2NF)

It is based on *full functional dependency*.

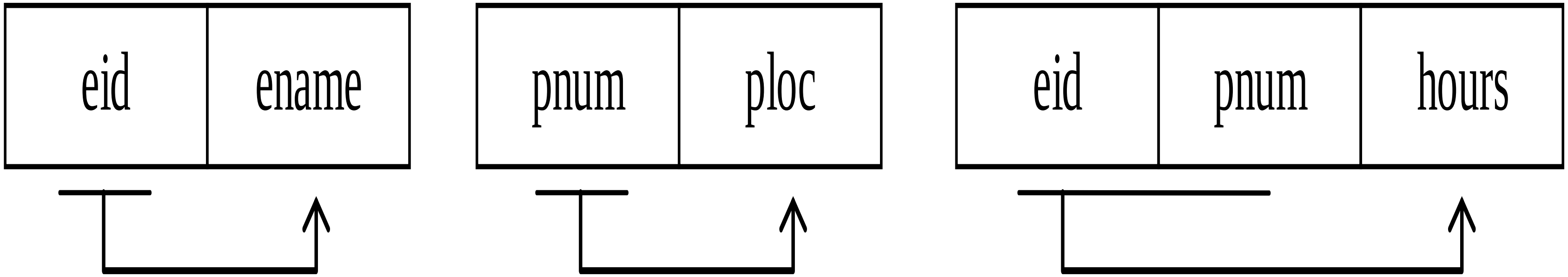
$\{X \rightarrow A\}$ is fully functional if we remove any attribute from X then that FD does not hold anymore.

Condition for 2NF: All non-key attributes are fully functionally dependent on key (or) no non-key attribute should be dependent on part of key(partial dependency).



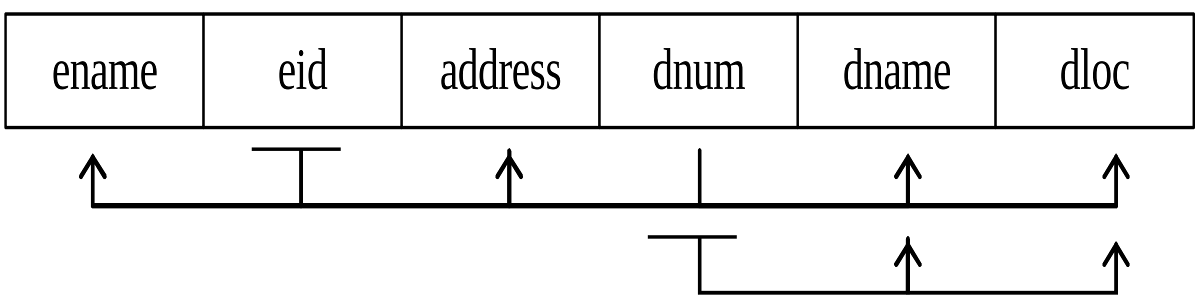
Here, *{ename}* is a non key attribute and determined by *{eid}* which is part of the key. Hence we say that *ename* not fully functionally dependent on key.

The relation shown is not in 2NF. Now we can decompose this in to three relations as shown below.



3. Third Normal form (3NF)

It is based on transitive dependency.
According to this, a relation should not have a non key attribute functionally determined by another non key attribute. i.e., there should be no transitive dependency.



Not in 3NF, because *Dname* is transitively dependent on *eid*.

Now we can decompose the above into 2 relations.

ename	eid	address	dnum

Dnum	Dname	dloc

Condition for 3NF

For each FD, X $\rightarrow$ A in database

i) X must be a superkey or
ii) A is key attribute

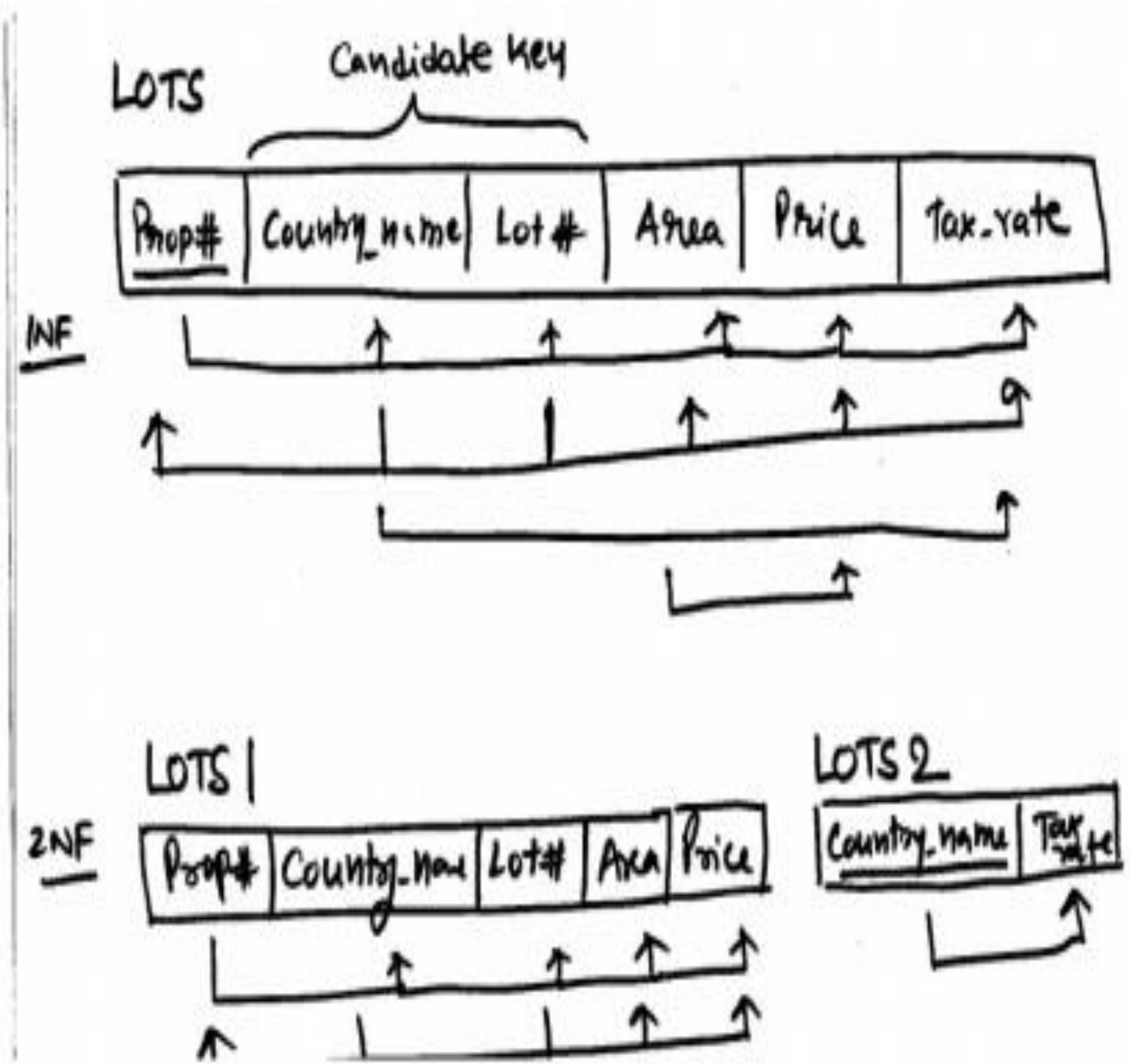
BCNF (Boyce Codd Normal Form)

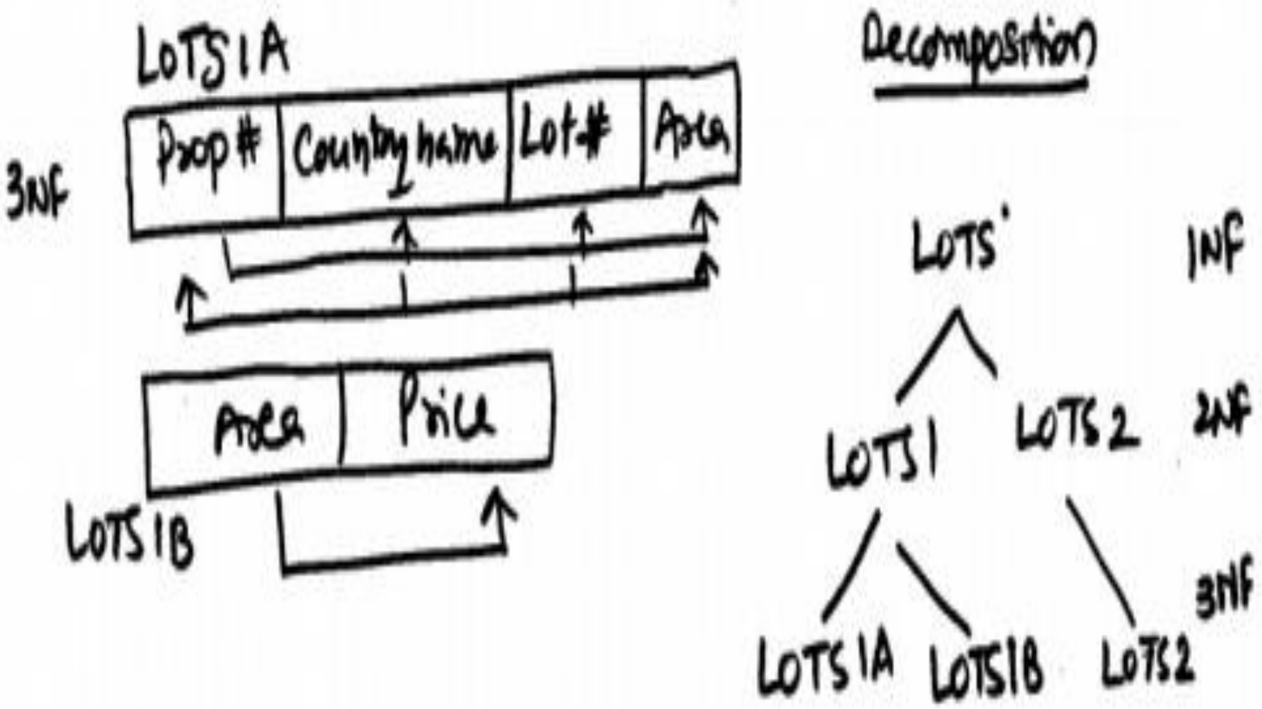
It is a stricter form of 3NF

Condition

For each FD $X \rightarrow A$

X must be a super key


LOTS
Candidate key
Prop#
Country_name
Lot#
Area
Price
Tax_rate
1NF
LOTS1
2NF
Prop#
Country_name
Lot#
Area
Price
LOTS2
Country_name
Tax_rate


LOTS1A
3NF
Prop#
County name
Lot#
Area
Area
Price
LOTS1B
Decomposition
LOTS
1NF
LOTS1
LOTS2
2NF
3NF
LOTS1A
LOTS1B
LOTS2

3NF to BCNF Decomposition

If $\alpha \to \beta$ not in BCNF
Split the R into
① $(\alpha \cup \beta)$ ② $(R-(\beta-\alpha))$

Ex R

Student	Course	Instructor

$\alpha \to \beta$
Instr $\to$ Course
violates BCNF.

Now split R

$(\alpha \cup \beta)$

Instr	Course

Student	Instr

Ex Problem (Normalization)

Find the highest NF satisfied and bring to BCNF

(i) R1(A,B, C, D) { AB→D; AB→ C; C→B; B→D }

(ii) R2(A,B, C, D) { AB→C; AB→ D; C→D }

Decomposition and Desirable properties

As we have seen, decomposition (of a bigger relation R into smaller ones), is a major step in the process of normalization.

But during this activity of decomposition, we need to make sure that the decomposition is *lossless* and *dependency preserving*

Loss-less join Decomposition

Let C represent a set of constraints on the database. A decomposition $\{R_1, R_2, R_3, \ldots\ldots\ldots R_4\}$ of a relation schema R is a lossless join decomposition for R if all relation instances *r* on R that are legal under C.

$$r = \prod_{R_1}(r) \quad * \quad \prod_{R_2}(r) \ldots\ldots\ldots\ldots = r$$

$\prod_{R_1}(r)$ = projection of *r* on R_1

r – relation instance in R

F = FDs on R

(or) $\{R\} \rightarrow \{R_1, R_2\}$

R(eid,name,sal, city, dno,dname,dloc)

r

eid	name	sal	city	dno	dname	dloc
102	Kiran	50000	HYD	10	HR	DEL
104	Mohan	38000	DEL	20	ACCT	CHEN
105	John	45000	HYD	20	ACCT	CHEN

R1(eid,name,sal, city, dno)

$\Pi_{R1}(r)$

eid	name	sal	city	dno
102	Kiran	50000	HYD	10
104	Mohan	38000	DEL	20
105	John	45000	HYD	20

R2(dno,dname,dloc)

$\Pi_{R2}(r)$

dno	dname	dloc
10	HR	DEL
20	ACCT	CHEN
20	ACCT	CHEN

$\Pi_{R1}(r) \; * \; \Pi_{R2}(r) \; = \; r$ Hence lossless

R(eid,name,sal, city, dno,dname,dloc)

r

eid	name	sal	city	dno	dname	dloc
102	Kiran	50000	HYD	10	HR	DEL
104	Mohan	38000	DEL	20	ACCT	CHEN
105	John	45000	HYD	20	ACCT	CHEN

R1(eid,name,sal, city)

$\Pi_{R1}(r)$

eid	name	sal	city
102	Kiran	50000	HYD
104	Mohan	38000	DEL
105	John	45000	HYD

R2(dno,dname,dloc,city)

$\Pi_{R2}(r)$

city	dno	dname	dloc
HYD	10	HR	DEL
DEL	20	ACCT	CHEN
HYD	20	ACCT	CHEN

$\Pi_{R1}(r) * \Pi_{R2}(r) \neq r$ Hence Lossy

Test for Lossless join property

(a) R={SSN, ENAME, PNUMBER, PNAME, PLOCATION, HOURS} $D=\{R_1, R_2\}$

R_1=EMP_LOCS={ENAME, PLOCATION}

R_2=EMP_PROJ1={SSN, PNUMBER, HOURS, PNAME, PLOCATION}

F={SSN→ENAME;PNUMBER→{PNAME, PLOCATION} ;{SSN,PNUMBER}→HOURS}

	SSN	ENAME	PNUMBER	PNAME	PLOCATION	HOURS
R_1	b_{11}	a_2	b_{13}	b_{14}	a_5	b_{16}
R_2	a_1	b_{22}	a_3	a_4	a_5	a_6

(no changes to matrix after applying functional dependencies)

(b)

(c) R={SSN, ENAME, PNUMBER, PNAME, PLOCATION, HOURS} $D=\{R_1, R_2, R_3\}$

R_1=EMP={SSN, ENAME}

R_2=PROJ={PNUMBER, PNAME, PLOCATION}

R_3=WORKS_ON={SSN, PNUMBER, HOURS}

F={SSN→{ENAME;PNUMBER→{PNAME, PLOCATION} ;{SSN,PNUMBER}→HOURS}

	SSN	ENAME	PNUMBER	PNAME	PLOCATION	HOURS
R_1	a_1	a_2	b_{13}	b_{14}	b_{15}	b_{16}
R_2	b_{21}	b_{22}	a_3	a_4	a_5	b_{26}
R_3	a_1	b_{32}	a_3	b_{34}	b_{35}	a_6

(original matrix S at start of algorithm)

	SSN	ENAME	PNUMBER	PNAME	PLOCATION	HOURS
R_1	a_1	a_2	b_{13}	b_{14}	b_{15}	b_{16}
R_2	b_{21}	b_{22}	a_3	a_4	a_5	b_{26}
R_3	a_1	$\cancel{b_{32}}$ a_2	a_3	$\cancel{b_{34}}$ a_4	$\cancel{b_{35}}$ a_5	a_6

(matrix S after applying the first two functional dependencies - last row is all "a" symbols, so we stop)

Ex Problem (Decomposition)

Suppose that we decompose the relation ***R=(A,B,C,D,E)*** into three relations- ***R1(A, C, E)*** , ***R2(B, D)*** and ***R3(A, B, E)*** . Find whether this decomposition is lossless, if the following set ***F*** of functional dependencies holds.

F={ A→CE; B→D;}

Note: Detailed working is required with final answer.

Dependency Preserving Decomposition

Given a set of dependencies F on R, the projection of F on R_i denoted by

(where R_i is a subset of R); is the set of FDs $X \rightarrow Y$ in F^+ such that the attributes in $X \cup Y$ are contained in R_i.

$$\left(\Pi_{R_1}(F) \cup \Pi_{R_2}(F) \cup \ldots, \Pi_{R_m}(F)\right)^+ = F^+$$

Then it is dependency preserving decomposition.

$\Pi_{R_1}(f)$ - is projection of F on R_1.

Ex Problem (Decomposition)

Suppose that we decompose the relation ***R=(A,B,C,D)*** into three relations-
R1(A, C, D) ,and ***R2(A, B)***. Find whether this decomposition is lossless and Dependency preserving, if the following set ***F*** of functional dependencies holds.
F={ A→B; B→C; C→D; A→D}

Note: Detailed working is required with final answer.

Exercise

If a relation ***R(A,B,C,D,E)*** with functional dependencies ***F={AB→CD; D→E}*** is decomposed into ***R1(A,B,E)***, ***R2(A,B,D)***, and ***R3(C,D,E)***, check if the decomposition is-

(i) ***dependency preserving*** or not.

(ii) ***Lossless*** or not.

[*Note*: Give complete working with all steps].

Summary

- ✓ *Introduction to Database Design*
- ✓*Functional Dependencies and Inference Rules*
- ✓*Concepts in Normalization*
- ✓*Normal Forms (1NF,2NF,3 NF and BCNF)*
- ✓*Desirable properties of Decomposition (requirements)*
- ✓*Lossless join decomposition and tests*
- ✓*Dependency preserving decomposition and tests*