Database Design & Applications (SS ZG 518)

BITS Pilani
Hyderabad Campus

Dr.R.Gururaj
CS&IS Dept.

Functional Dependencies and Normal Forms (Ch.10 & 11)

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.

Good database design is needed for:

- Clarity in understanding the database and
- To formulate good queries

This is achieved by schema refinement.

Functional Dependencies

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

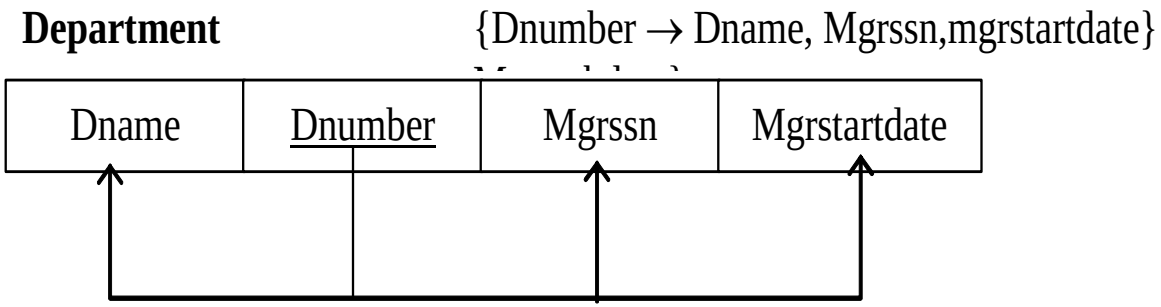
Function Dependency X→Y

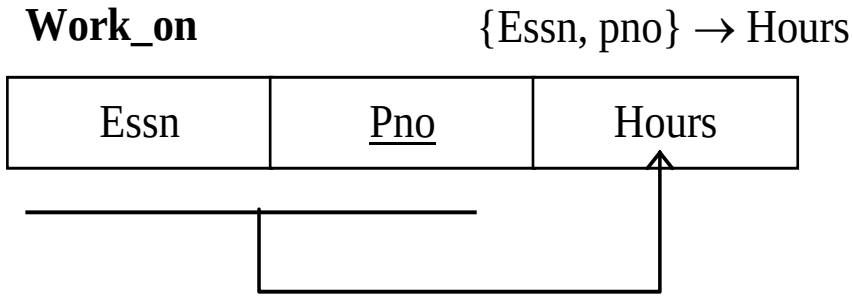
X functionally determines Y in a relation schema R if and only if whenever two tuples of *r*(R) agree on their X values they must necessarily agree on their Y values, but Y $\rightarrow$ X is not true (need not be)

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





Inference Rules for FDs

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

If $X \geq Y$ 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 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

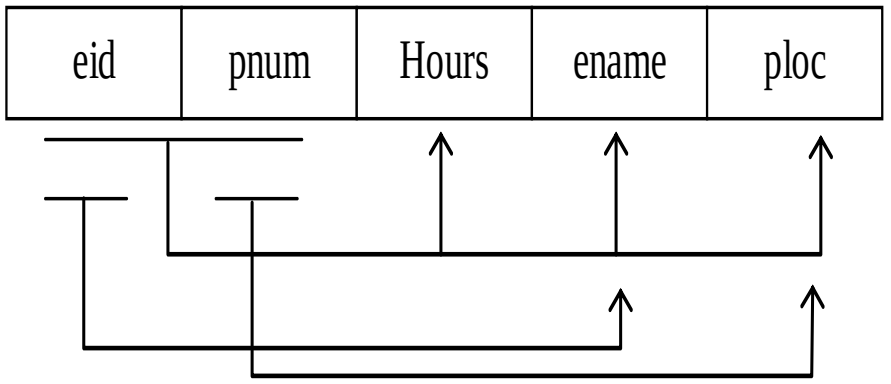
DId	*Dname*	*Dloc*
10	Engg	HYD CHENNAI
20	Mark	HYD MUMBAI

2. Second Normal Form (2NF)

It is based on *full functional dependency*.

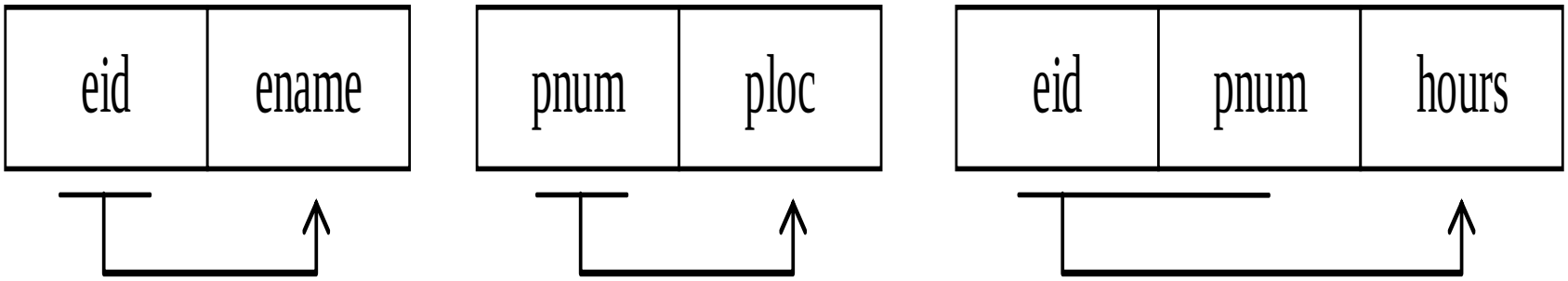
{X → A} is fully functional if we remove any attribute from X then that FD does not hold anymore.

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).



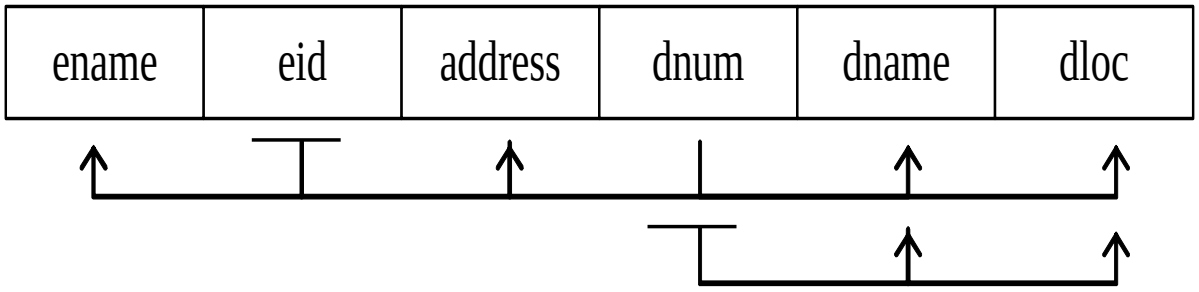
Here, *{ename}* is a non key attribute and determined by *{eid}* which is part of the key. Hence we say that *ename* not fully functionally dependent on key.

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



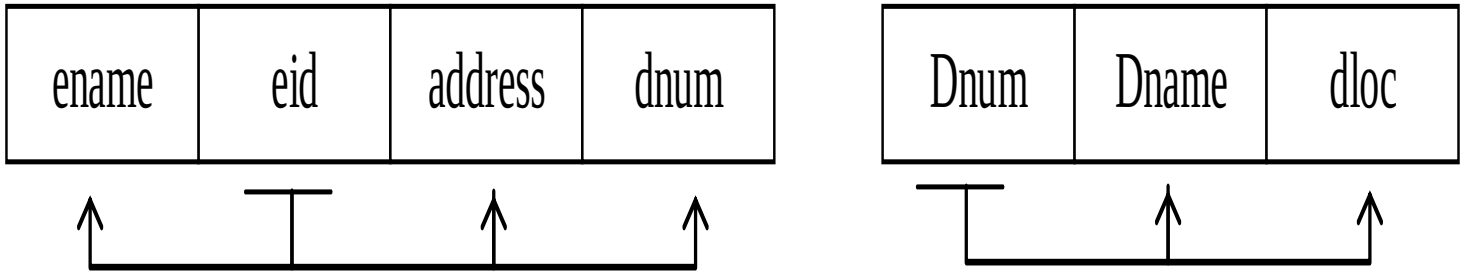
3. Third Normal form (3NF)

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.



Condition for 3NF

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

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

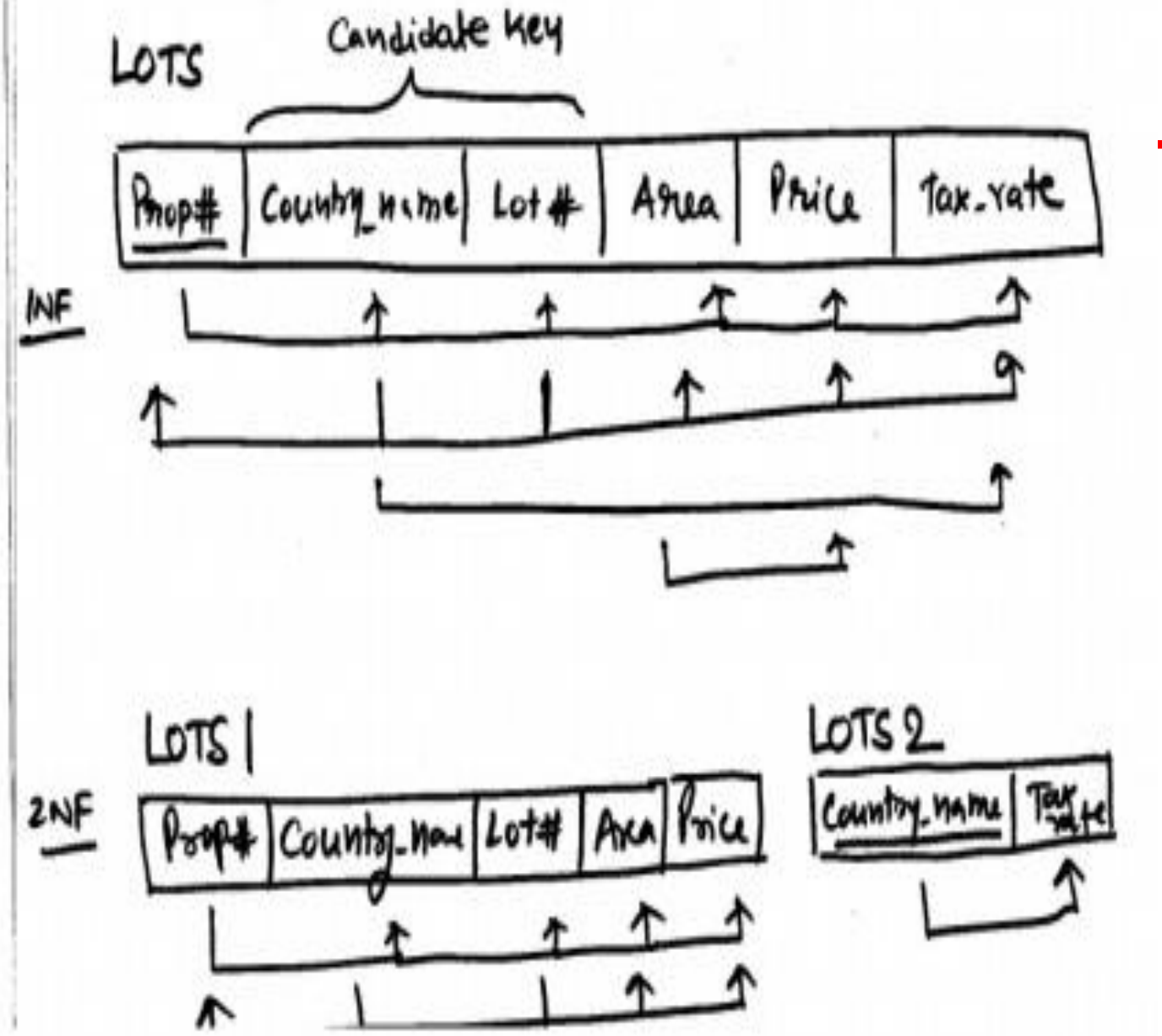
BCNF (Boyce Codd Normal Form)

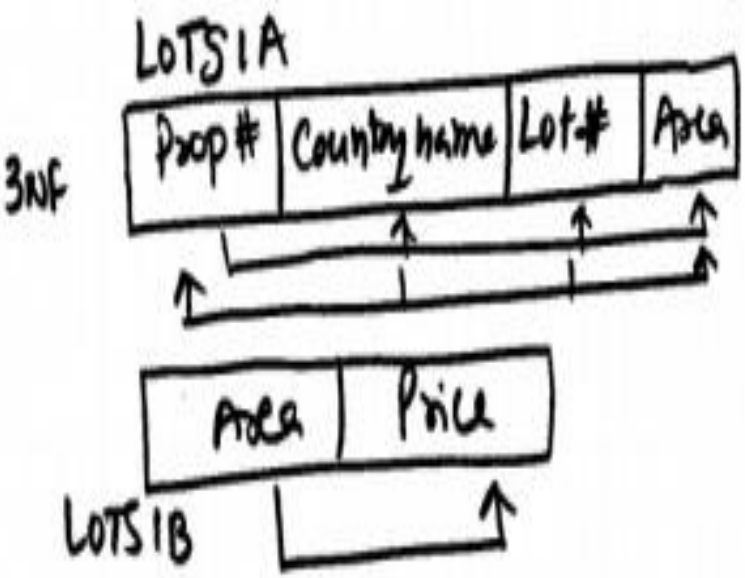
It is a stricter form of 3NF

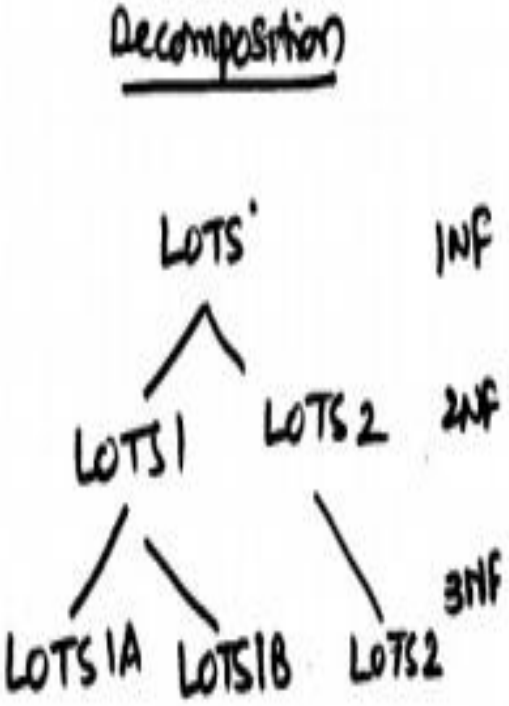
Condition

For each FD $X \rightarrow A$

X must be a super key


LOTS
Candidate key
Prop#
Country_name
Lot#
Area
Price
Tax_rate
INF
LOTS 1
2NF
Prop#
County_nam
Lot#
Area
Price
LOTS 2
Country_name
Tax


LOTS1A
Prop #
County name
Lot #
Area
3NF
Area
Price
LOTS1B


Decomposition
LOTS'
1NF
LOTS1
LOTS2
2NF
3NF
LOTS1A
LOTS1B
LOTS2

3NF to BCNF Decomposition

If $\alpha \rightarrow \beta$ not in BCNF

Split the R into

① $(\alpha \cup \beta)$ ② $(R - (\beta - \alpha))$

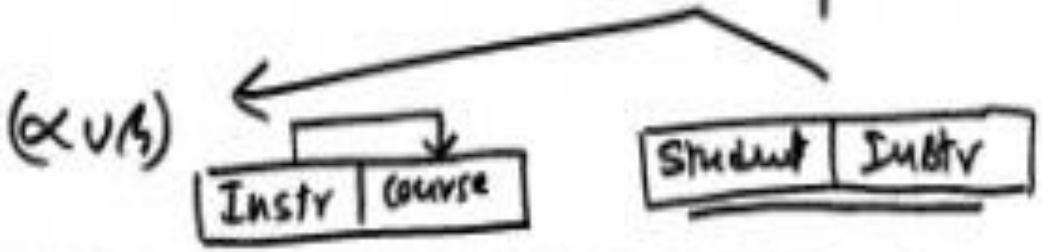
Ex R

$\alpha \rightarrow \beta$

Instr → Course

violates BCNF.

Now Split R



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 $\{R_1, R_2, R_3, \ldots\ldots R_4\}$ 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) $\{R\} \rightarrow \{R_1, R_2\}$

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	~~b_{32}~~ a_2	a_3	~~b_{34}~~ a_4	~~b_{35}~~ a_5	a_6

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

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.

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

Then it is dependency preserving decomposition.

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

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*