Database Management Systems

BITS Pilani
Hyderabad Campus

Dr.R.Gururaj
CS&IS Dept.

Lecture Session-10

Schema Refinement-2

innovate achieve lead

Content

- *3 NF and BCNF*
- *Decomposition requirements*
- *Lossless join decomposition*
- *Dependency preserving decomposition*
- *Examples*

Recap of 1NF and 2 NF

innovate achieve lead

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

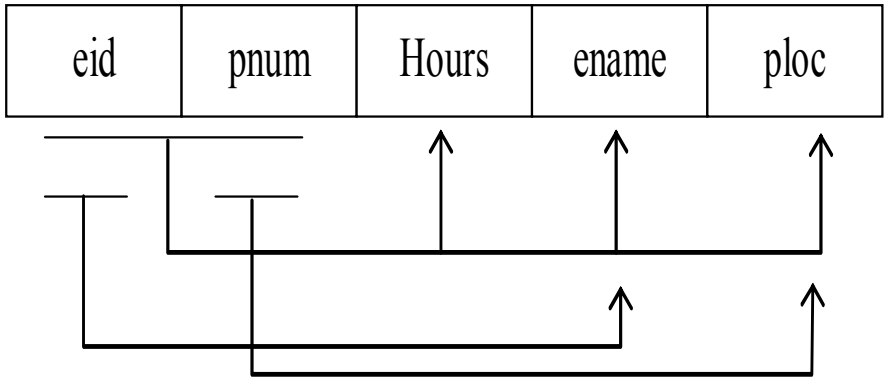
DId	*Dname*	*Dloc*
10	Engg	HYD CHENNAI
20	Mark	HYD MUMBAI

2. Second Normal Form (2NF)

It is based on *full functional dependency*.

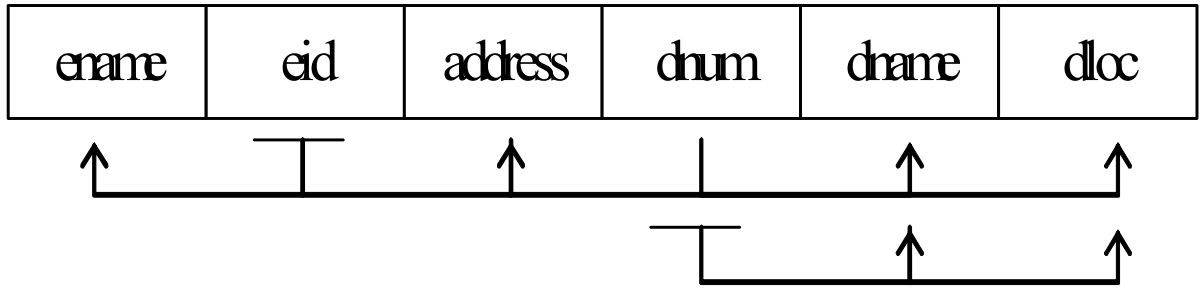
{X → A} is fully functional if we remove any attribute from X then that FD does not hold anymore.

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).



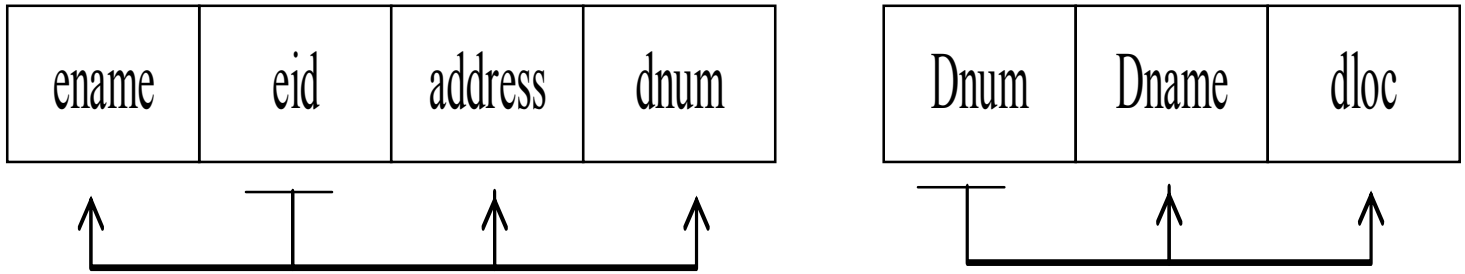
3. Third Normal form (3NF)

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 → 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 superkey

4th NF: Is based on multivalued dependency

5th NF: Is based on join dependency normally database designers go up to 3NF only, and 4NF & 5NF are beyond the scope of our discussion.

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

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)

(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.

This dependency preserving condition makes sure that no FD in original relation is lost as a result of decomposition. The FDs represent constraints (business logic).

Note:

- Not every BCNF is dependency preserving
- Limited amount of redundancy in 3NF in the form of transitive dependency is better than losing FDs as result of bringing 3NF to BCNF.

Summary

- ✓ *Recap of 2NF*
- ✓ *What is 3NF and BCNF*
- ✓ *Decomposition into 3NF and BCNF*
- ✓ *Lossless join decomposition*
- ✓ *Dependency preserving decomposition*