Database Management Systems

BITS Pilani
Hyderabad Campus

Dr.R.Gururaj
CS&IS Dept.

Lecture Session-9

Schema Refinement -1

Content

- *Introduction to Schema Refinement*
- *Functional Dependencies*
- *Inference Rules*
- *Normalization*
- *Normal Forms (1NF and 2NF)*

Schema Refinement (Database Design)

All database applications have certain constraints that must hold for the data.

These set of constraints help to make the system to accept correct and valid data.

A DBMS must provide facilities for defining and enforcing these constraints.

<u>Types of Integrity Constraints for Relational data model</u>

Domain constraints – Data type, Null, Check for certain range.

Entity constraints – Primary key and Unique key

Referential integrity – Foreign key

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 performed on the conceptual schema which is the result of mapping high-level conceptual schema(ER) to Data model specific conceptual schema(relational schema)

Functional Dependencies

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

If a relational database schema has *n* attributes A_1, A_2, A_3,….., A_n, then think of it as a universal database schema $R = \{A_1, A_2, A_3,\ldots\ldots A_4\}$.

This is not a real table, this is conceptual for developing formal theory of data dependencies.

Function Dependency

Denoted by $X \rightarrow Y$ between two sets of attributes in R, and specifies a constraint on the possible tuples that can form a relational instance *r* of R.

Values of Y component are determined by X component.

(or) Y is functionally dependent on X.

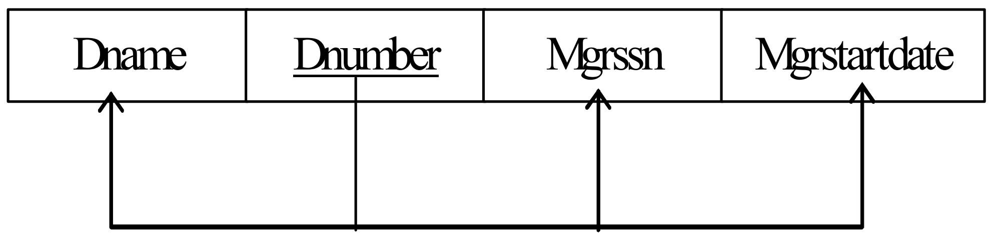
Thus, X functionally determines Y in a relation schema R if and only if whenever two tuples of *r*(R) agree on their X values they must necessarily agree on their Y values, but $Y \rightarrow X$ is not true (need not be)

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

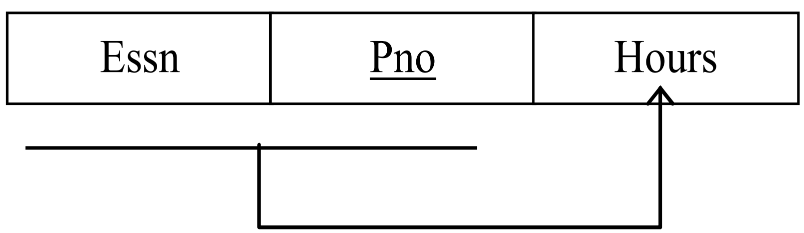
Note: FDs cannot be inferred. They should be defined by someone who knows the semantics of the database very well.

Diagrammatic Notation

Department {Dnumber → Dname, Mgrssn,mgrstartdate}



Work_on {Essn, pno} → Hours



Inference rules for FDs

If F denotes a set of FDs, we can infer some new FDs from specified FDs, set of all possible functional dependencies is called as *closure of F* and denoted as (F^+).

If F = { ssn $\rightarrow$ {Ename, address, dnumber},
Dnumber $\rightarrow$ {Dname, dlocation}
}

We can infer new FDs as below

ssn $\rightarrow$ {Dname, Dlocation}
ssn $\rightarrow$ ssn
Dnumber $\rightarrow$ Dname

If X $\rightarrow$ Z we can say that XY $\rightarrow$ Z.

Inference Rules for FDs

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

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

Rule 2 ($1R_2$) (Argumentation)

$X \rightarrow Y$; 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 clusure F^+ of F, by repeated application of rules IR-1 to IR-3. These rules are called *as Armstrong's Inference rules*.

Equivalence of sets of FDs

F covers E if every FD in E is in F^+
F and E are equivalent if $E^+ = F^+$

A set of FDs F is minimal if it satisfies the following conditions.

- ❑ Every dependency in F has a single attribute for its RHS.
- ❑ We can't replace any dependency $X \rightarrow A$ in F with any dependency $Y \rightarrow A$ where Y is proper subset of X, and still have a set of dependencies that are equivalent to F.
- ❑ We can't remove any dependency from F and still have equivalent FD to F.

Normalization & Normal forms

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

Normalization of data – is the process of analyizing relation schemas based on their FDs and PKs/Keys to achieve the desirable properties of –(i)Minimal redundancy (ii)Minimal anomalies

In the process of normalization, unsatisfactory relations that do not meet the requirements are de-composed into smaller relations.

Every level of NF need to satisfy certain conditions.

Normal Form (NF) of a relation refers to highest NF condition that it satisfies.

Schema Refinement (Database Design) encompasses
(i) Normalization – bringing the database to the desired level of NF
(ii) Checking for other desired properties like –

- Lossless join property
- Dependency preserving property

The above properties are desirable during the process of decomposition.

Some definitions useful in database design

Key: Is a minimal superkey is also called as *candidate key*. One of these becomes *PK*. Other candidate keys are called *alternate keys*.

key Attribute: An attribute which is part of some key (any Candidate key)

We study the general definitions of NF in terms of keys, not just PK.
A relation can have any number of keys but ha only one PK.

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

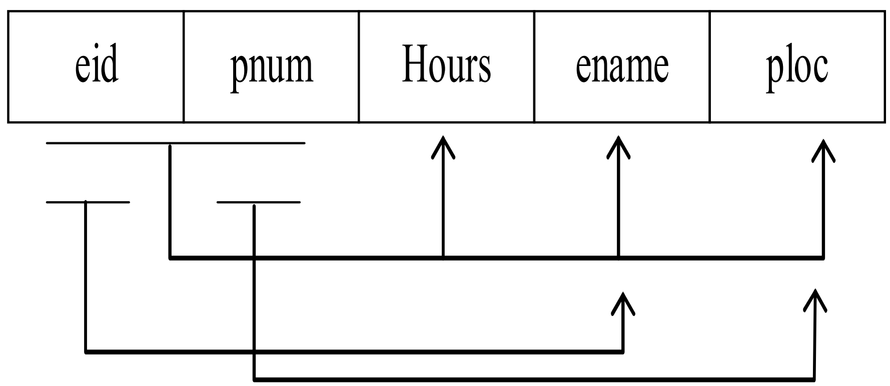
DId	*Dname*	*Dloc*
10	Engg	HYD CHENNAI
20	Mark	HYD MUMBAI

2. Second Normal Form (2NF)

It is based on *full functional dependency*.

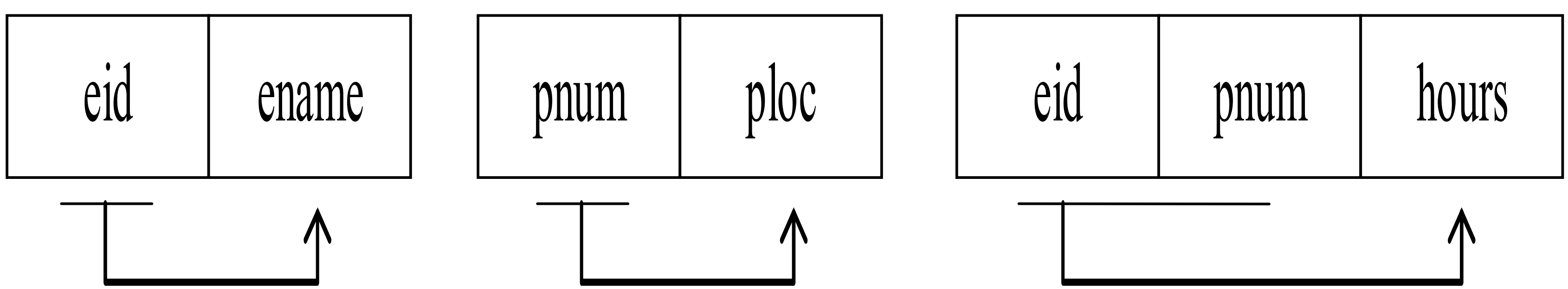
$\{X \rightarrow A\}$ is fully functional if we remove any attribute from X then that FD does not hold anymore.

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.



Summary

- ✓ *What is the Schema Refinement process*
- ✓ *Functional Dependencies*
- ✓ *What are the Inference Rules*
- ✓ *What is Normalization & 1 NF and 2NF*