

# CS4.301 Data & Applications

Ponnurangam Kumaraguru ("PK")  
#ProfGiri @ IIIT Hyderabad



<https://www.instagram.com/pk.profgiri/>



<https://www.linkedin.com/in/ponguru/>



**Ponnurangam Kumaraguru "PK"** ✓

@ponguru

# Specifying Joined Tables in the FROM Clause of SQL

## Joined table

Permits users to specify a table resulting from a join operation in the FROM clause of a query

### The FROM clause in Q1A

Contains a single joined table. JOIN may also be called INNER JOIN

```
Select fname, lname, address
from (employee join department
on dno=dnumber) where
dname='research';
```

```
[mysql> Select fname, lname, address from (employee |
join department on dno=dnumber) where dname='resear
ch';
```

fname	lname	address
John	Smith	731 Fondren, Houston TX
Franklin	Wong	638 Voss, Houston TX
Joyce	English	5631 Rice, Houston TX
Ramesh	Narayan	975 Fire Oak, Humble TX

```
4 rows in set (0.04 sec)
```

# Different Types of JOINed Tables in SQL

Specify different types of join

- NATURAL JOIN

- Various types of OUTER JOIN (LEFT, RIGHT, FULL )

NATURAL JOIN on two relations R and S

- No join condition specified

- Is equivalent to an implicit EQUIJOIN condition for each pair of attributes with same name from R and S

- The associated tables have one or more pairs of identically named columns

- The columns must be the same data type

- No need for ON

# INNER and OUTER Joins

## INNER JOIN (**versus** OUTER JOIN)

- Default type of join in a joined table

- Tuple is included in the result only if a matching tuple exists in the other relation

## LEFT OUTER JOIN

- Every tuple in left table must appear in result

- If no matching tuple

  - Padded with NULL values for attributes of right table

## RIGHT OUTER JOIN

- Every tuple in right table must appear in result

- If no matching tuple

  - Padded with NULL values for attributes of left table

```

1 SELECT *
2 FROM company
3 INNER JOIN foods
4 ON company.company_id = foods.company_id;

```

Output:

COMPANY_ID	COMPANY_NAME	COMPANY_CITY	ITEM_ID	ITEM_NAME	ITEM_UNIT	COMPANY_ID
16	Akas Foods	Delhi	1	Chex Mix	Pcs	16
15	Jack Hill Ltd	London	6	Cheez-It	Pcs	15
15	Jack Hill Ltd	London	2	BN Biscuit	Pcs	15
17	Foodies.	London	3	Mighty Munch	Pcs	17
15	Jack Hill Ltd	London	4	Pot Rice	Pcs	15
18	Order All	Boston	5	Jaffa Cakes	Pcs	18

```

1 SELECT *
2 FROM company
3 NATURAL JOIN foods;

```

Copy

Output:

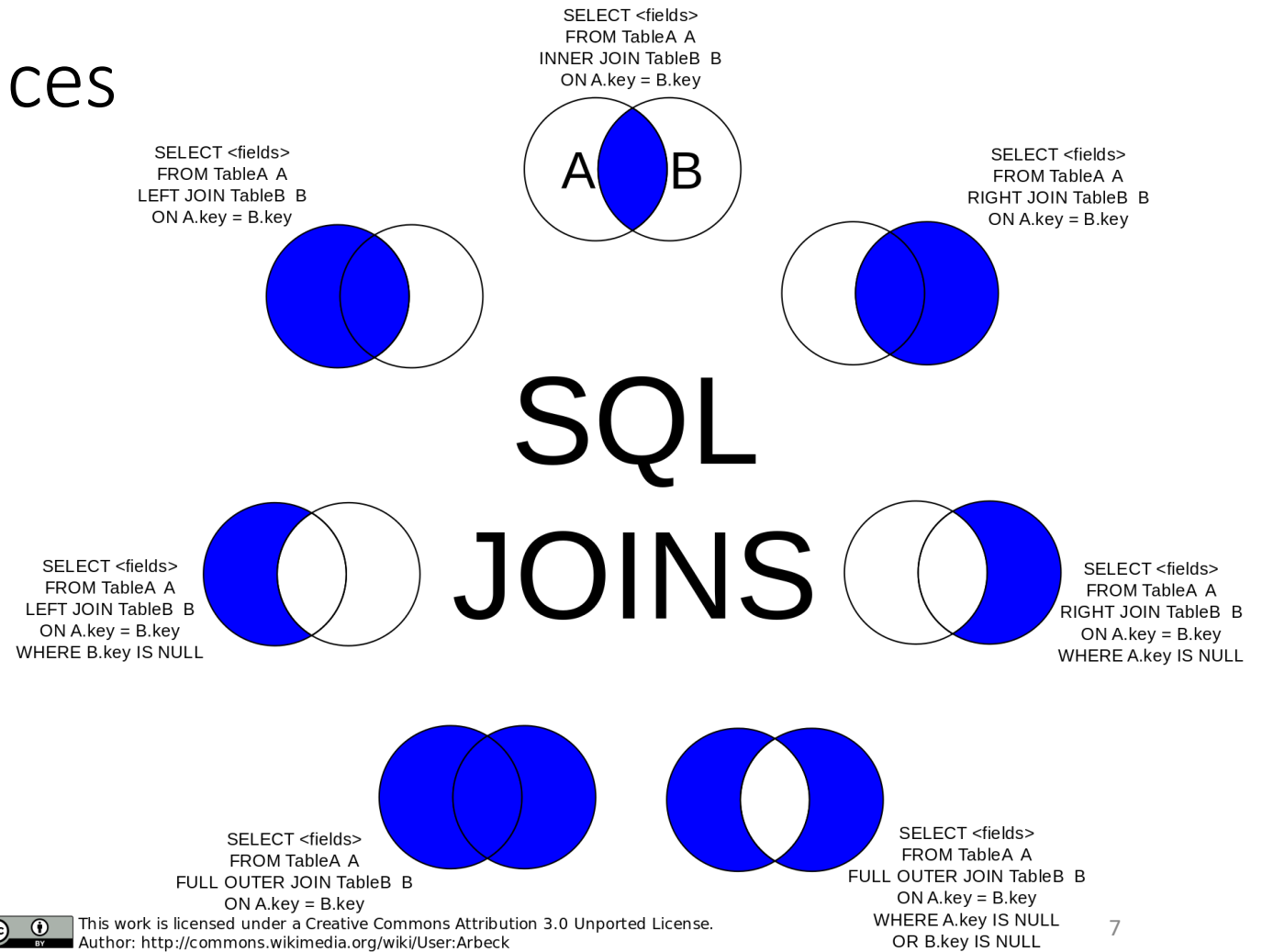
COMPANY_ID	COMPANY_NAME	COMPANY_CITY	ITEM_ID	ITEM_NAME	ITEM_UNIT
16	Akas Foods	Delhi	1	Chex Mix	Pcs
15	Jack Hill Ltd	London	6	Cheez-It	Pcs
15	Jack Hill Ltd	London	2	BN Biscuit	Pcs
17	Foodies.	London	3	Mighty Munch	Pcs
15	Jack Hill Ltd	London	4	Pot Rice	Pcs
18	Order All	Boston	5	Jaffa Cakes	Pcs

## Example: LEFT OUTER JOIN

```
select E.Lname AS Employee_Name,  
S.Lname as Supervisor_Name from  
EMPLOYEE as E left outer join  
EMPLOYEE as S on  
E.Super_ssn=S.Ssn;
```

```
mysql> select E.Lname AS Employee_Name, S.Lname as Supervisor_Name from E  
EMPLOYEE as E left outer join EMPLOYEE as S on E.Super_ssn=S.Ssn;  
+-----+-----+  
| Employee_Name | Supervisor_Name |  
+-----+-----+  
| Smith         | Wong            |  
| Wong          | Borg            |  
| English       | Wong            |  
| Narayan       | Wong            |  
| Borg          | NULL            |  
| Wallace       | Borg            |  
| Jabbar        | Wallace          |  
| Zelaya        | Wallace          |  
+-----+-----+  
8 rows in set (0.06 sec)
```

# Joins differences



# Multiway JOIN in the FROM clause

FULL OUTER JOIN – combines result if LEFT and RIGHT OUTER JOIN

Can nest JOIN specifications for a multiway join:

```
SELECT Pnumber, Dnum, Lname,  
Address, Bdate FROM ((PROJECT  
JOIN DEPARTMENT ON  
Dnum=Dnumber) JOIN EMPLOYEE  
ON Mgr_ssn=Ssn) WHERE  
Plocation='Stafford';
```

```
mysql> SELECT Pnumber, Dnum, Lname, Address, Bdate FROM ((PROJECT JOIN DE  
PARTMENT ON Dnum=Dnumber) JOIN EMPLOYEE ON Mgr_ssn=Ssn) WHERE Plocatio  
n='Stafford';  
+-----+-----+-----+-----+-----+  
| Pnumber | Dnum | Lname | Address | Bdate |  
+-----+-----+-----+-----+-----+  
| 10 | 4 | Wallace | 291 Berry, Bellaire TX | 1941-06-20 |  
| 30 | 4 | Wallace | 291 Berry, Bellaire TX | 1941-06-20 |  
+-----+-----+-----+-----+-----+  
2 rows in set (0.02 sec)
```



## **CHAPTER 14**

# **Basics of Functional Dependencies and Normalization for Relational Databases**

# 1. Informal Design Guidelines for Relational Databases (1)

What is relational database design?

The grouping of attributes to form "good" relation schemas

Two levels of relation schemas

The logical "user view" level

The storage "base relation" level

Design is concerned mainly with base relations

What are the criteria for "good" base relations?

# Informal Design Guidelines for Relational Databases (2)

We first discuss informal guidelines for good relational design

Then we discuss formal concepts of functional dependencies and normal forms

- 1NF (First Normal Form)
- 2NF (Second Normal Form)
- 3NF (Third Normal Form)
- BCNF (Boyce-Codd Normal Form)

Additional types of dependencies, further normal forms, relational design algorithms by synthesis are discussed in Chapter 15

## 1.1 Semantics of the Relational Attributes must be clear

GUIDELINE 1: Informally, each tuple in a relation should represent one entity or relationship instance. (Applies to individual relations and their attributes).

Attributes of different entities (EMPLOYEEs, DEPARTMENTs, PROJECTs) should not be mixed in the same relation

Only foreign keys should be used to refer to other entities

Entity and relationship attributes should be kept apart as much as possible.

Bottom Line: *Design a schema that can be explained easily relation by relation. The semantics of attributes should be easy to interpret.*

(a)

**EMP\_DEPT**

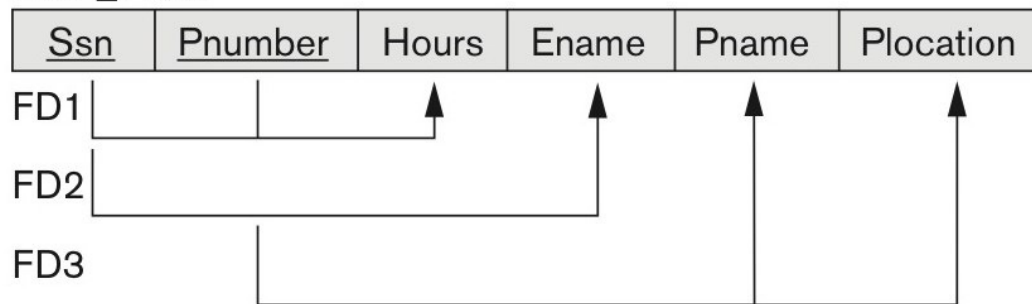


Any concerns here?

EMP\_DEPT: mixing attributes of employees & departments

(b)

**EMP\_PROJ**



EMP\_PROJ: mixes attributes of employees, projects & works\_on

**Figure 14.4**  
Sample states for EMP\_DEPT and EMP\_PROJ resulting from applying NATURAL JOIN to the relations in Figure 14.2. These may be stored as base relations for performance reasons.

EMP_DEPT					Redundancy	
Ename	Ssn	Bdate	Address	Dnumber	Dname	Dmgr_ssn
Smith, John B.	123456789	1965-01-09	731 Fondren, Houston, TX	5	Research	333445555
Wong, Franklin T.	333445555	1955-12-08	638 Voss, Houston, TX	5	Research	333445555
Zelaya, Alicia J.	999887777	1968-07-19	3321 Castle, Spring, TX	4	Administration	987654321
Wallace, Jennifer S.	987654321	1941-06-20	291 Berry, Bellaire, TX	4	Administration	987654321
Narayan, Ramesh K.	666884444	1962-09-15	975 FireOak, Humble, TX	5	Research	333445555
English, Joyce A.	453453453	1972-07-31	5631 Rice, Houston, TX	5	Research	333445555
Jabbar, Ahmad V.	987987987	1969-03-29	980 Dallas, Houston, TX	4	Administration	987654321
Borg, James E.	888665555	1937-11-10	450 Stone, Houston, TX	1	Headquarters	888665555

EMP_PROJ						Redundancy	Redundancy
Ssn	Pnumber	Hours	Ename	Pname	Plocation		
123456789	1	32.5	Smith, John B.	ProductX	Bellaire		
123456789	2	7.5	Smith, John B.	ProductY	Sugarland		
666884444	3	40.0	Narayan, Ramesh K.	ProductZ	Houston		
453453453	1	20.0	English, Joyce A.	ProductX	Bellaire		
453453453	2	20.0	English, Joyce A.	ProductY	Sugarland		
333445555	2	10.0	Wong, Franklin T.	ProductY	Sugarland		
333445555	3	10.0	Wong, Franklin T.	ProductZ	Houston		
333445555	10	10.0	Wong, Franklin T.	Computerization	Stafford		
333445555	20	10.0	Wong, Franklin T.	Reorganization	Houston		
999887777	30	30.0	Zelaya, Alicia J.	Newbenefits	Stafford		
999887777	10	10.0	Zelaya, Alicia J.	Computerization	Stafford		
987987987	10	35.0	Jabbar, Ahmad V.	Computerization	Stafford		
987987987	30	5.0	Jabbar, Ahmad V.	Newbenefits	Stafford		
987654321	30	20.0	Wallace, Jennifer S.	Newbenefits	Stafford		
987654321	20	15.0	Wallace, Jennifer S.	Reorganization	Houston		
888665555	20	Null	Borg, James E.	Reorganization	Houston		

## 1.2 Redundant Information in Tuples and Update Anomalies

Information is stored redundantly

- Wastes storage

- Causes problems with update anomalies

  - Insertion anomalies

  - Deletion anomalies

  - Modification anomalies

# EXAMPLE OF AN INSERT ANOMALY

Consider the relation:

EMP\_PROJ(Emp#, Proj#, Ename, Pname, No\_hours)

Insert Anomaly:

Cannot insert a project unless an employee is assigned to it.

Conversely

Cannot insert an employee unless an he/she is assigned to a project.



# EXAMPLE OF A DELETE ANOMALY

Consider the relation:

EMP\_PROJ(Emp#, Proj#, Ename, Pname, No\_hours)

Delete Anomaly:

When a project is deleted, it will result in deleting all the employees who work on that project.

Alternately, if an employee is the sole employee on a project, deleting that employee would result in deleting the corresponding project.

# EXAMPLE OF AN UPDATE ANOMALY

Consider the relation:

EMP\_PROJ(Emp#, Proj#, Ename, Pname, No\_hours)

Update Anomaly:

Changing the name of project number P1 from “Billing” to “Customer-Accounting” may cause this update to be made for all 100 employees working on project P1.

# Guideline for Redundant Information in Tuples and Update Anomalies

## GUIDELINE 2:

Design a schema that does not suffer from the insertion, deletion and update anomalies.

If there are any anomalies present, then note them so that applications can be made to take them into account.

## 1.3 Null Values in Tuples

### GUIDELINE 3:

Relations should be designed such that their tuples will have as few NULL values as possible

Attributes that are NULL frequently could be placed in separate relations (with the primary key)

### Reasons for nulls; different meanings for null:

Attribute not applicable or invalid [visa status to US students]

Attribute value unknown [DOB of an employee]

Value is known but absent; it has not been recorded yet [phone # of employee]

# This Lecture

## 1.4 Generation of Spurious Tuples – avoid at any cost

Bad designs for a relational database may result in erroneous results for certain JOIN operations

### GUIDELINE 4:

No spurious tuples should be generated by doing a natural-join of any relations.

(a)

**EMP\_LOCS**

<u>Ename</u>	<u>Plocation</u>
--------------	------------------

P.K.

**EMP\_PROJ1**

<u>Ssn</u>	<u>Pnumber</u>	Hours	Pname	Plocation
------------	----------------	-------	-------	-----------

P.K.

(b)

**EMP\_LOCS**

Ename	Plocation
Smith, John B.	Bellaire
Smith, John B.	Sugarland
Narayan, Ramesh K.	Houston
English, Joyce A.	Bellaire
English, Joyce A.	Sugarland
Wong, Franklin T.	Sugarland
Wong, Franklin T.	Houston
Wong, Franklin T.	Stafford
Zelaya, Alicia J.	Stafford
Jabbar, Ahmad V.	Stafford
Wallace, Jennifer S.	Stafford
Wallace, Jennifer S.	Houston
Borg, James E.	Houston

**EMP\_PROJ1**

Ssn	Pnumber	Hours	Pname	Plocation
123456789	1	32.5	ProductX	Bellaire
123456789	2	7.5	ProductY	Sugarland
666884444	3	40.0	ProductZ	Houston
453453453	1	20.0	ProductX	Bellaire
453453453	2	20.0	ProductY	Sugarland
333445555	2	10.0	ProductY	Sugarland
333445555	3	10.0	ProductZ	Houston
333445555	10	10.0	Computerization	Stafford
333445555	20	10.0	Reorganization	Houston
999887777	30	30.0	Newbenefits	Stafford
999887777	10	10.0	Computerization	Stafford
987987987	10	35.0	Computerization	Stafford
987987987	30	5.0	Newbenefits	Stafford
987654321	30	20.0	Newbenefits	Stafford
987654321	20	15.0	Reorganization	Houston
888665555	20	NULL	Reorganization	Houston

	Ssn	Pnumber	Hours	Pname	Plocation	Ename
*	123456789	1	32.5	ProductX	Bellaire	Smith, John B.
	123456789	1	32.5	ProductX	Bellaire	English, Joyce A.
	123456789	2	7.5	ProductY	Sugarland	Smith, John B.
*	123456789	2	7.5	ProductY	Sugarland	English, Joyce A.
*	123456789	2	7.5	ProductY	Sugarland	Wong, Franklin T.
	666884444	3	40.0	ProductZ	Houston	Narayan, Ramesh K.
*	666884444	3	40.0	ProductZ	Houston	Wong, Franklin T.
*	453453453	1	20.0	ProductX	Bellaire	Smith, John B.
	453453453	1	20.0	ProductX	Bellaire	English, Joyce A.
*	453453453	2	20.0	ProductY	Sugarland	Smith, John B.
	453453453	2	20.0	ProductY	Sugarland	English, Joyce A.
*	453453453	2	20.0	ProductY	Sugarland	Wong, Franklin T.
*	333445555	2	10.0	ProductY	Sugarland	Smith, John B.
*	333445555	2	10.0	ProductY	Sugarland	English, Joyce A.
	333445555	2	10.0	ProductY	Sugarland	Wong, Franklin T.
*	333445555	3	10.0	ProductZ	Houston	Narayan, Ramesh K.
	333445555	3	10.0	ProductZ	Houston	Wong, Franklin T.
	333445555	10	10.0	Computerization	Stafford	Wong, Franklin T.
*	333445555	20	10.0	Reorganization	Houston	Narayan, Ramesh K.
	333445555	20	10.0	Reorganization	Houston	Wong, Franklin T.

\*  
\*  
\*

Additional  
tuples that  
were not there  
in Emp\_proj is  
here, they are  
called spurious  
tuples



## 2. Functional Dependencies

### Functional dependencies (FDs)

Are used to specify *formal measures* of the "goodness" of relational designs

And keys are used to define **normal forms** for relations

Are **constraints** that are derived from the *meaning* and *interrelationships* of the data attributes

A set of attributes *X functionally determines* a set of attributes *Y* if the value of *X* determines a unique value for *Y*

## 2.1 Defining Functional Dependencies

$X \rightarrow Y$  holds if whenever two tuples have the same value for  $X$ , they *must have* the same value for  $Y$

For any two tuples  $t_1$  and  $t_2$  in any relation instance  $r(R)$ : If  $t_1[X]=t_2[X]$ , *then*  $t_1[Y]=t_2[Y]$

$X \rightarrow Y$  in  $R$  specifies a *constraint* on all relation instances  $r(R)$

Written as  $X \rightarrow Y$ ; can be displayed graphically on a relation schema as in Figures; denoted by the arrow  $\rightarrow$

FDs are derived from the real-world constraints on the attributes

# Examples of FD constraints (1)

Social security number determines employee name

$SSN \rightarrow ENAME$

Project number determines project name and location

$PNUMBER \rightarrow \{PNAME, PLOCATION\}$

Employee ssn and project number determines the hours per week that the employee works on the project

$\{SSN, PNUMBER\} \rightarrow HOURS$

# Examples of FD constraints (1)

Social security number determines employee name

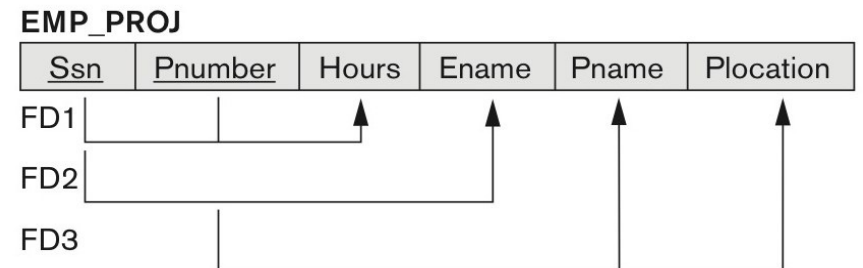
$SSN \rightarrow ENAME$

Project number determines project name and location

$PNUMBER \rightarrow \{PNAME, PLOCATION\}$

Employee ssn and project number determines the hours per week that the employee works on the project

$\{SSN, PNUMBER\} \rightarrow HOURS$



## Examples of FD constraints (2)

An FD is a property of the attributes in the schema  $R$

The constraint must hold on *every* relation instance  $r(R)$

If  $K$  is a key of  $R$ , then  $K$  functionally determines all attributes in  $R$   
(since we never have two distinct tuples with  $t_1[K]=t_2[K]$ )

# Defining FDs from instances

Note that in order to define the FDs, we need to understand the meaning of the attributes involved and the relationship between them.

Given the instance (population) of a relation, all we can conclude is that an FD may exist between certain attributes.

What we can definitely conclude is – that certain FDs do not exist because there are tuples that show a violation of those dependencies.

# Ruling Out FDs

Note that given the state of the TEACH relation, we can say that the FD: Text  $\rightarrow$  Course may exist. However, the FDs Teacher  $\rightarrow$  Course, Teacher  $\rightarrow$  Text and Course  $\rightarrow$  Text are ruled out.

**TEACH**

Teacher	Course	Text
Smith	Data Structures	Bartram
Smith	Data Management	Martin
Hall	Compilers	Hoffman
Brown	Data Structures	Horowitz

What FDs may exist?

A relation  $R(A, B, C, D)$  with its extension.  
Which FDs may exist in this relation?

A	B	C	D
a1	b1	c1	d1
a1	b2	c2	d2
a2	b2	c2	d3
a3	b3	c4	d3



# What FDs may exist?

A relation  $R(A, B, C, D)$  with its extension.

Which FDs may exist in this relation?

A	B	C	D
a1	b1	c1	d1
a1	b2	c2	d2
a2	b2	c2	d3
a3	b3	c4	d3

$B \rightarrow C$ ;  $C \rightarrow B$ ;  $\{A, B\} \rightarrow C$ ;  $\{A, B\} \rightarrow D$ ;  $\{C, D\} \rightarrow B$

How about  $A \rightarrow B$ ?  $B \rightarrow A$ ?  $D \rightarrow C$ ?

# Normal Forms Based on Primary Keys

Normalization of Relations

Practical Use of Normal Forms

Definitions of Keys and Attributes Participating in Keys

First Normal Form

Second Normal Form

Third Normal Form

## 3.1 Normalization of Relations (1)

### **Normalization:**

The process of decomposing unsatisfactory "bad" relations by breaking up their attributes into smaller relations

### **Normal form:**

Condition using keys and FDs of a relation to certify whether a relation schema is in a particular normal form

# Normalization of Relations (2)

2NF, 3NF, BCNF

based on keys and FDs of a relation schema

4NF

based on keys, multi-valued dependencies: MVDs;

5NF

based on keys, join dependencies: JDs

Additional properties may be needed to ensure a good relational design  
(lossless join, dependency preservation; see Chapter 15)

## 3.2 Practical Use of Normal Forms

**Normalization** is carried out in practice so that the resulting designs are of high quality and meet the desirable properties

The practical utility of these normal forms becomes questionable when the constraints on which they are based are *hard to understand* or to *detect*

The database designers *need not* normalize to the highest possible normal form  
(usually up to 3NF and BCNF. 4NF rarely used in practice.)

**Denormalization:**

The process of storing the join of higher normal form relations as a base relation—which is in a lower normal form

### 3.3 Definitions of Keys and Attributes Participating in Keys (1)

A **superkey** of a relation schema  $R = \{A_1, A_2, \dots, A_n\}$  is a set of attributes  $S$  *subset-of*  $R$  with the property that no two tuples  $t_1$  and  $t_2$  in any legal relation state  $r$  of  $R$  will have  $t_1[S] = t_2[S]$

A **key**  $K$  is a **superkey** with the *additional property* that removal of any attribute from  $K$  will cause  $K$  not to be a superkey any more.

## Definitions of Keys and Attributes      Participating in Keys (2)

If a relation schema has more than one key, each is called a **candidate** key.

One of the candidate keys is *arbitrarily* designated to be the **primary key**, and the others are called **secondary keys**.

A **Prime attribute** must be a member of *some* candidate key

A **Nonprime attribute** is not a prime attribute—that is, it is not a member of any candidate key.

## 3.4 First Normal Form

Disallows

- composite attributes

- multivalued attributes

- nested relations**; attributes whose values for an *individual tuple* are non-atomic

Considered to be part of the definition of a relation

Most RDBMSs allow only those relations to be defined that are in First Normal Form




# Normalization into 1NF

(a)

DEPARTMENT

Dname	<u>Dnumber</u>	Dmgr_ssn	Dlocations



(b)

DEPARTMENT

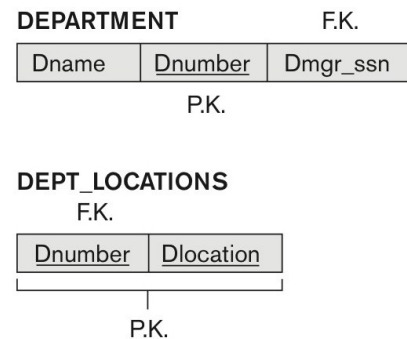
Dname	<u>Dnumber</u>	Dmgr_ssn	Dlocations
Research	5	333445555	{Bellaire, Sugarland, Houston}
Administration	4	987654321	{Stafford}
Headquarters	1	888665555	{Houston}

**Figure 14.9**

Normalization into 1NF. (a)  
A relation schema that is  
not in 1NF. (b) Sample  
state of relation  
DEPARTMENT

## Ways to make it make it 1NF?

# 1NF



DEPARTMENT			
Dname	<u>Dnumber</u>	Dmgr_ssn	<u>Dlocation</u>
Research	5	333445555	Bellaire
Research	5	333445555	Sugarland
Research	5	333445555	Houston
Administration	4	987654321	Stafford
Headquarters	1	888665555	Houston

Redundancy

If the maximum number of values (n) for location is known, replace it with n attributes  
e.g. Only 3 locations for the company – Dlocation1, Dlocation2, Dlocation3  
Introducing NULL if most departments have fewer than 3 locations  
Hard to query, e.g. List the departments that have 'Bellaire' as one of the locations  
1<sup>st</sup> option is commonly used one

# Normalizing nested relations into 1NF

(a)

EMP_PROJ		Projs	
Ssn	Ename	Pnumber	Hours

(b)

EMP_PROJ			
Ssn	Ename	Pnumber	Hours
123456789	Smith, John B.	1	32.5
		2	7.5
666884444	Narayan, Ramesh K.	3	40.0
453453453	English, Joyce A.	1	20.0
		2	20.0
333445555	Wong, Franklin T.	2	10.0
		3	10.0
		10	10.0
		20	10.0
999887777	Zelaya, Alicia J.	30	30.0
		10	10.0
987987987	Jabbar, Ahmad V.	10	35.0
		30	5.0
987654321	Wallace, Jennifer S.	30	20.0
		20	15.0
888665555	Borg, James E.	20	NULL

(c)

EMP_PROJ1	
Ssn	Ename

EMP\_PROJ2

Ssn	Pnumber	Hours
-----	---------	-------

Remove the nested relation attributes into a new relation and propagate primary key

This idea can be applied recursively to a relation with multiple-level nesting to unnest

BLOB, CLOB – atomic, single-valued so 1NF

**Figure 14.10**

Normalizing nested relations into 1NF. (a) Schema of the EMP\_PROJ relation with a nested relation attribute PROJS. (b) Sample extension of the EMP\_PROJ relation showing nested relations within each tuple. (c) Decomposition of EMP\_PROJ into relations EMP\_PROJ1 and EMP\_PROJ2 by propagating the primary key.

## 3.5 Second Normal Form (1)

Uses the concepts of **FDs, primary key**

Definitions

**Prime attribute:** An attribute that is member of the primary key K

**Full functional dependency:** a FD  $Y \rightarrow Z$  where removal of any attribute from Y means the FD does not hold any more

Examples:

$\{SSN, PNUMBER\} \rightarrow HOURS$  is a full FD since neither  $SSN \rightarrow HOURS$  nor  $PNUMBER \rightarrow HOURS$  hold

$\{SSN, PNUMBER\} \rightarrow ENAME$  is not a full FD (it is called a partial dependency ) since  $SSN \rightarrow ENAME$  also holds

## Second Normal Form (2)

A relation schema R is in **second normal form (2NF)** if every non-prime attribute A in R is fully functionally dependent on the primary key

R can be decomposed into 2NF relations via the process of 2NF normalization or “second normalization”

# Fully functional dependency

If X and Y are an attribute set of a relation, Y is fully functional dependent on X, if Y is functionally dependent on X but not on any proper subset of X.

e.g. In the relation  $ABC \rightarrow D$ , attribute D is fully functionally dependent on ABC and not on any proper subset of ABC. That means that subsets of ABC like AB, BC, A, B, etc cannot determine D.

supplier_id	item_id	price
1	1	540
2	1	545
1	2	200
2	2	201
1	1	540
2	2	201
3	1	542

{ supplier\_id , item\_id }  $\rightarrow$  price

# Partial dependency

A functional dependency  $X \rightarrow Y$  is a partial dependency if  $Y$  is functionally dependent on  $X$  and  $Y$  can be determined by any proper subset of  $X$ .

e.g. we have a relationship  $AC \rightarrow B$ ,  $A \rightarrow D$ , and  $D \rightarrow B$ .

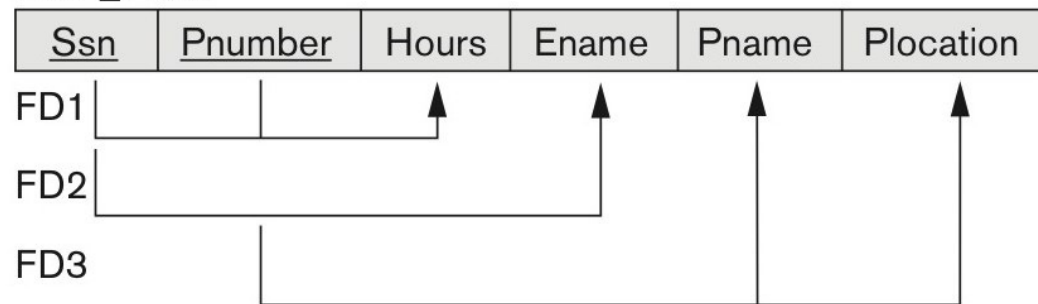
name	roll_no	course
Ravi	2	DBMS
Tim	3	OS
John	5	Java

$\{\text{name}\} \rightarrow \text{course}$

$\{\text{roll\_no}\} \rightarrow \text{course}$

# 2NF

## EMP\_PROJ

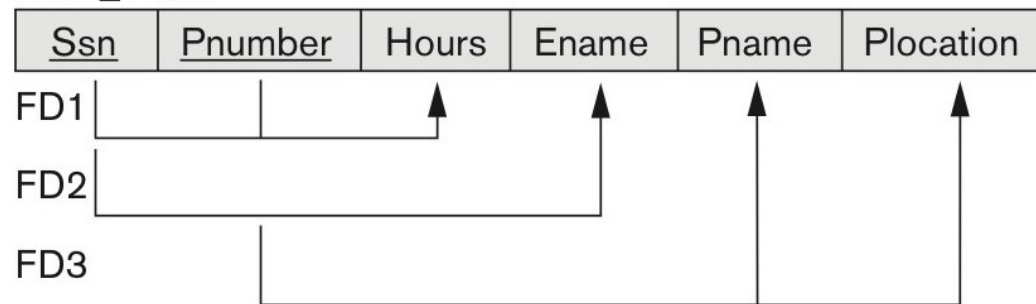


Any violation?



## 2NF

### EMP\_PROJ



Any violation?

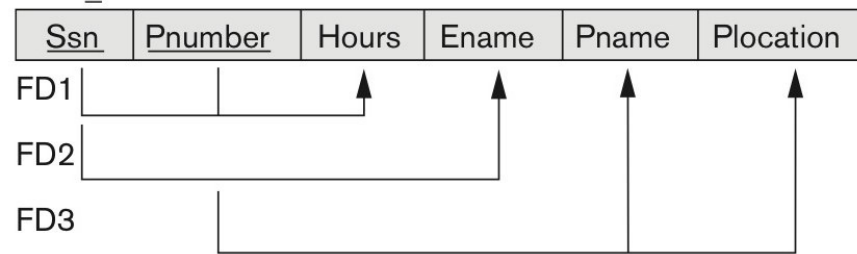
FD2, FD3 violates 2NF

$\{ssn\} \rightarrow \{Ename\}$ ,  $\{pnumber\} \rightarrow \{pname, plocation\}$  &  $\{ssn, pnumber\}$  are primary keys; not fully functionally dependent

# Normalizing into 2NF

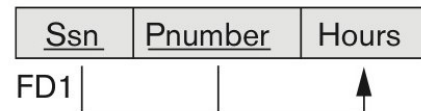
(a)

EMP\_PROJ

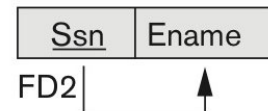


2NF Normalization

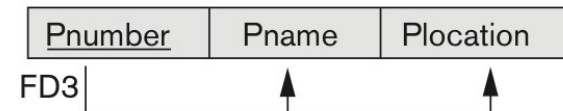
EP1



EP2



EP3



EP1, EP2, EP3 are fully functionally dependent

**Figure 14.11**

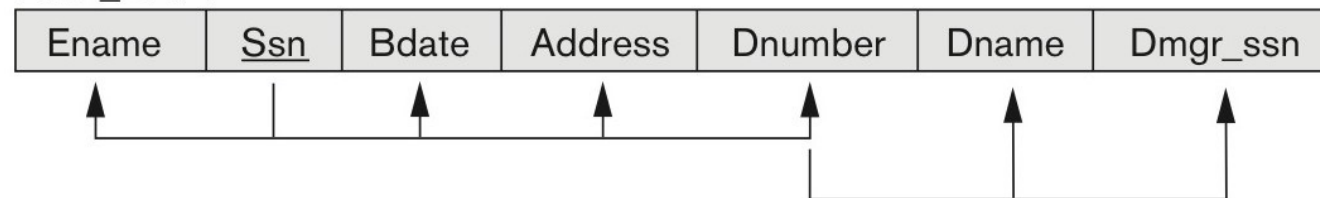
Normalizing into 2NF and 3NF.  
(a) Normalizing EMP\_PROJ into 2NF relations.

# Third normal form

## Transitive Dependency

$X \rightarrow Y$  in  $R$  is transitive dependency, if there exists a set of attributes  $Z$  in  $R$  that is neither a candidate key nor a subset of any key of  $R$  and both  $X \rightarrow Z$  &  $Z \rightarrow Y$  hold.

### EMP\_DEPT



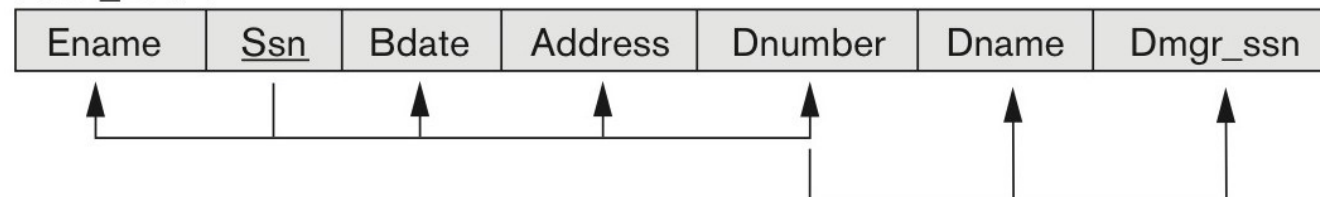
Any transitivity?

# Third normal form

## Transitive Dependency

$X \rightarrow Y$  in R is transitive dependency, if there exists a set of attributes Z in R that is neither a candidate key nor a subset of any key of R and both  $X \rightarrow Z$  &  $Z \rightarrow Y$  hold.

### EMP\_DEPT



Any transitivity?

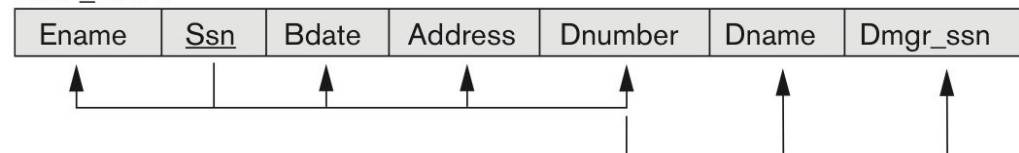
$Ssn \rightarrow dmgr\_ssn$  is transitive through  $dnumber$

Both  $ssn \rightarrow dnumber$  &  $dnumber \rightarrow dmgr\_ssn$  hold &  $dnumber$  is neither a key nor a subset of a key

# Third normal form

R is in 3NF if it satisfies 2NF and no nonprime attribute or R is transitively dependent on the primary key

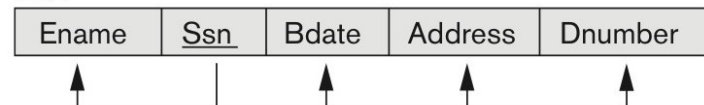
EMP\_DEPT



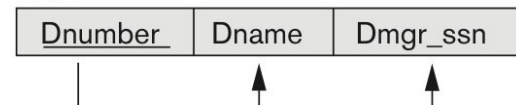
Normalizing EMP\_DEPT into 3NF relations.

3NF Normalization

ED1



ED2



ED1 & ED2 represent independent facts about employees & departments