CS4.301 Data & Applications

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

pk.profgiri /in/ponguru @ponguru Ponnurangam.kumaraguru

The ALTER table command

Alter table actions include:

- Adding or dropping a column (attribute)
- Changing a column definition
- Adding or dropping table constraints

Example:

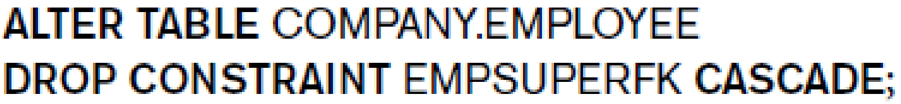
```
ALTER TABLE COMPANY.EMPLOYEE ADD COLUMN Job
VARCHAR(12);
```

Keeping track of jobs of employees

Adding and Dropping Constraints

Change constraints specified on a table

- Add or drop a named constraint



```
ALTER TABLE COMPANY.EMPLOYEE
DROP CONSTRAINT EMPSUPERFK CASCADE;
```

To be dropped, a constraint must have been given a name when it is specified

Dropping Columns, Default Values

To drop a column

- Choose either CASCADE or RESTRICT
- CASCADE would drop the column from views etc. RESTRICT is possible if no views refer to it.

ALTER TABLE COMPANY.EMPLOYEE **DROP COLUMN** Address **CASCADE**;

removes the attribute Address from the employee base table

Default values can be dropped and altered :

ALTER TABLE COMPANY.DEPARTMENT **ALTER COLUMN** Mgr_ssn **DROP DEFAULT**;

ALTER TABLE COMPANY.DEPARTMENT **ALTER COLUMN** Mgr_ssn **SET DEFAULT** ‘333445555’;

The EXISTS Functions in SQL for correlating queries

`EXISTS` function

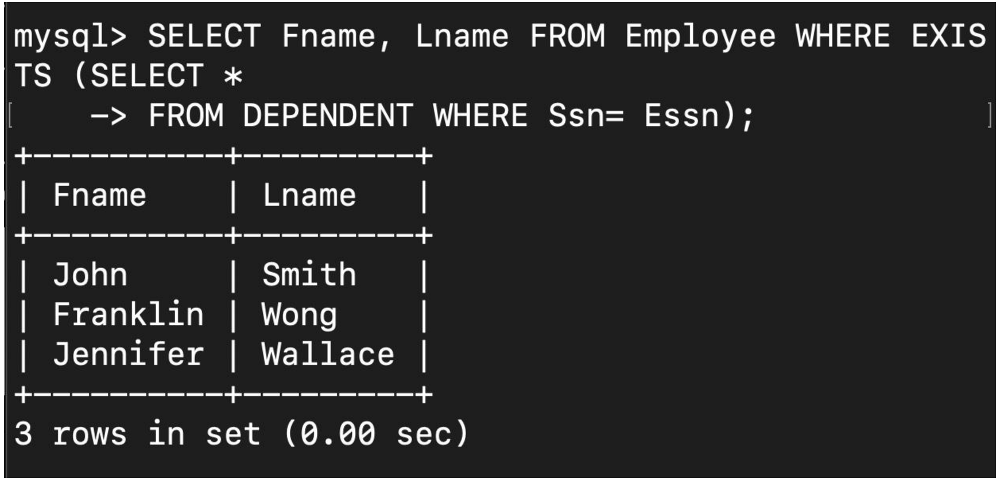
- Check whether the result of a correlated nested query is empty or not. They are Boolean functions that return a TRUE or FALSE result.

`EXISTS` and `NOT EXISTS`

- Typically used in conjunction with a correlated nested query

USE of EXISTS

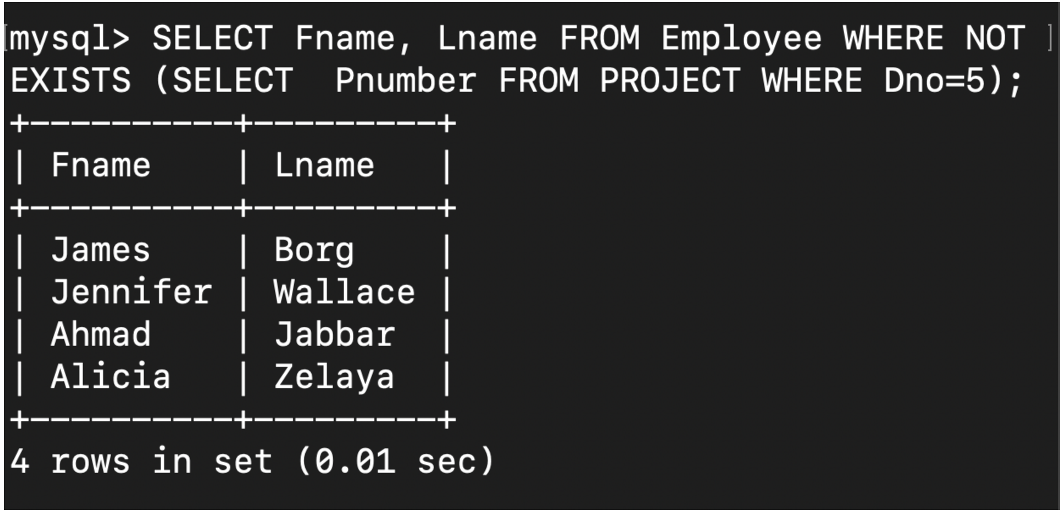
SELECT Fname, Lname FROM Employee WHERE EXISTS (SELECT * FROM DEPENDENT WHERE Ssn= Essn);



```
mysql> SELECT Fname, Lname FROM Employee WHERE EXIS
TS (SELECT *
    -> FROM DEPENDENT WHERE Ssn= Essn);
+----------+---------+
| Fname    | Lname   |
+----------+---------+
| John     | Smith   |
| Franklin | Wong    |
| Jennifer | Wallace |
+----------+---------+
3 rows in set (0.00 sec)
```

USE OF NOT EXISTS

SELECT Fname, Lname FROM Employee WHERE NOT EXISTS (SELECT Pnumber FROM PROJECT WHERE Dno=5);



```
mysql> SELECT Fname, Lname FROM Employee WHERE NOT
EXISTS (SELECT  Pnumber FROM PROJECT WHERE Dno=5);
+----------+---------+
| Fname    | Lname   |
+----------+---------+
| James    | Borg    |
| Jennifer | Wallace |
| Ahmad    | Jabbar  |
| Alicia   | Zelaya  |
+----------+---------+
4 rows in set (0.01 sec)
```

Specifying Joined Tables in the FROM Clause of SQL

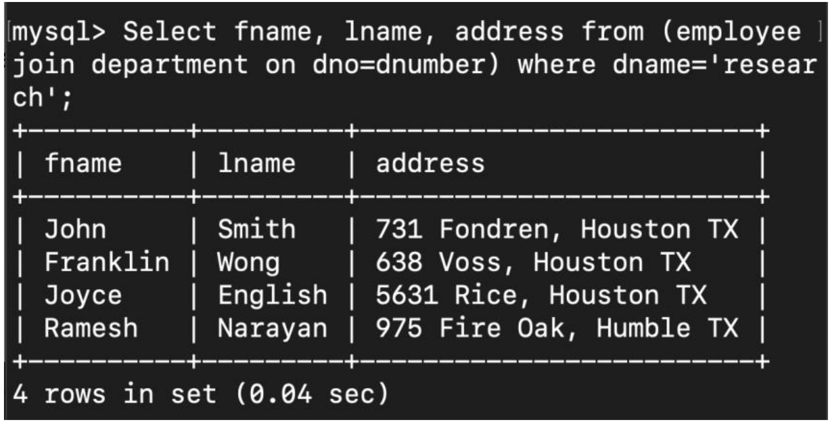
Joined table

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

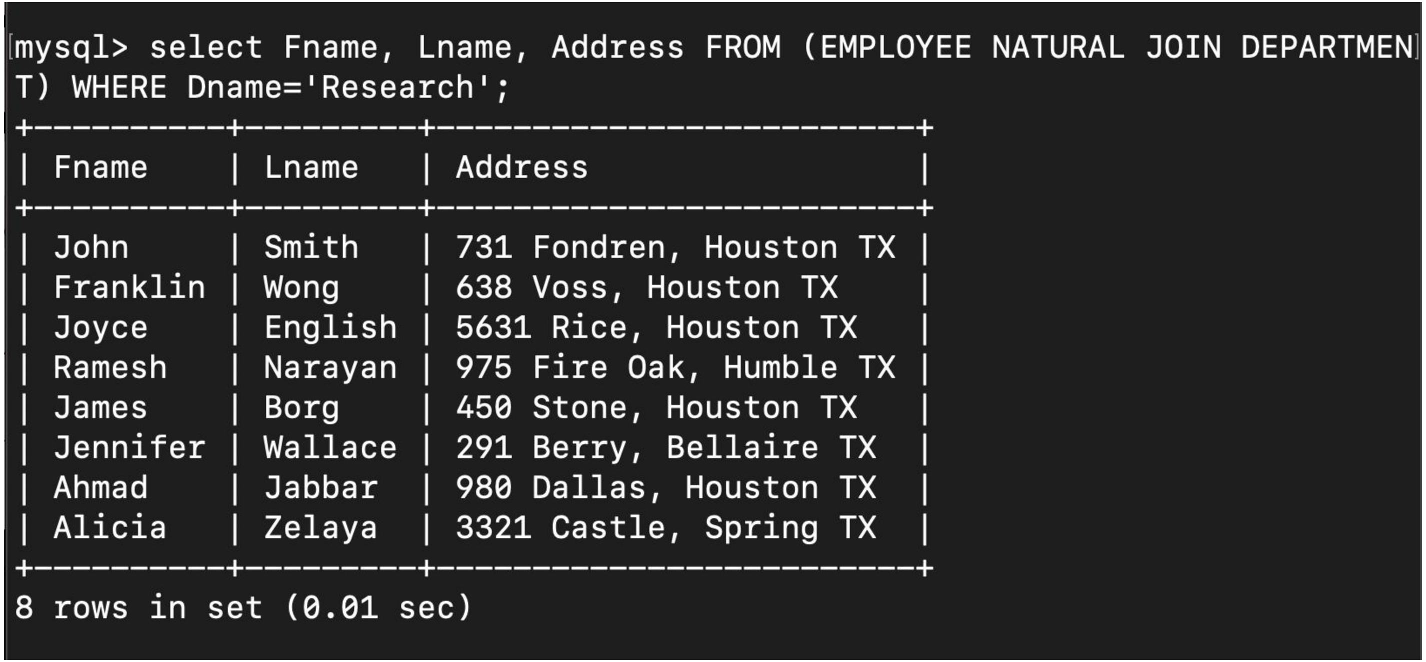
Specify different types of join

- 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

NATURAL JOIN



```
mysql> select Fname, Lname, Address FROM (EMPLOYEE NATURAL JOIN DEPARTMEN
T) WHERE Dname='Research';
+----------+---------+-------------------------+
| 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 |
| James    | Borg    | 450 Stone, Houston TX   |
| Jennifer | Wallace | 291 Berry, Bellaire TX  |
| Ahmad    | Jabbar  | 980 Dallas, Houston TX  |
| Alicia   | Zelaya  | 3321 Castle, Spring TX  |
+----------+---------+-------------------------+
8 rows in set (0.01 sec)
```

INNER and OUTER Joins

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

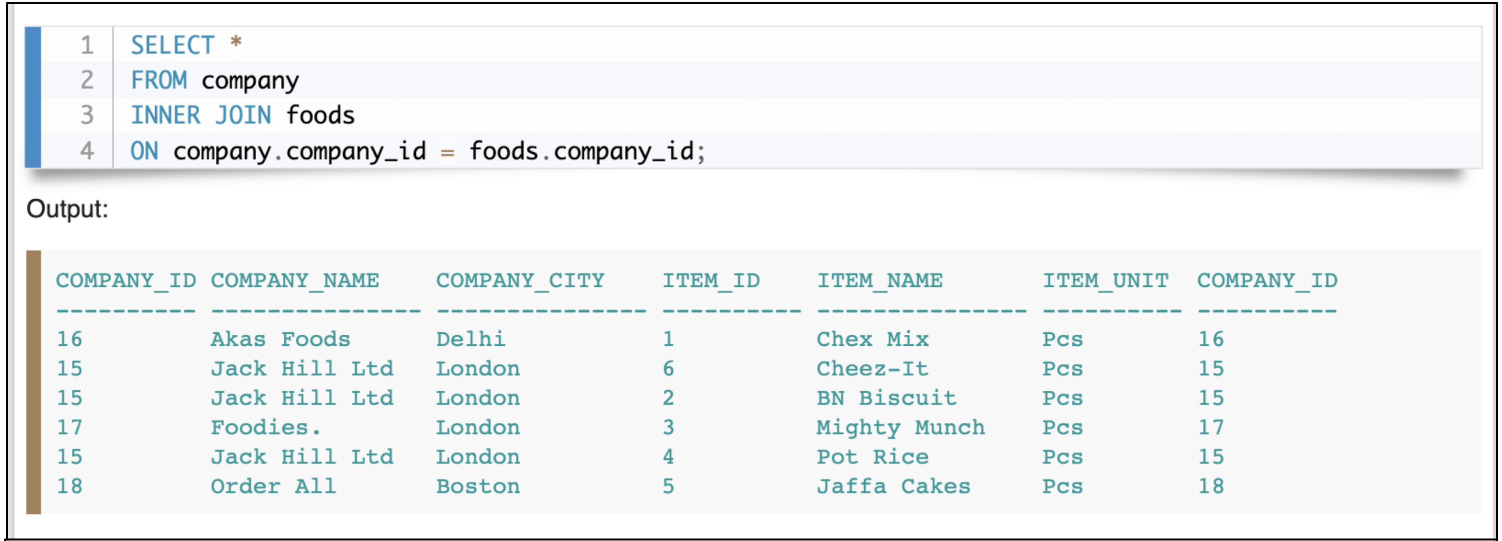
- Default type of join in a joined table
- Tuple is included in the result only if a matching tuple exists in the other relation

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

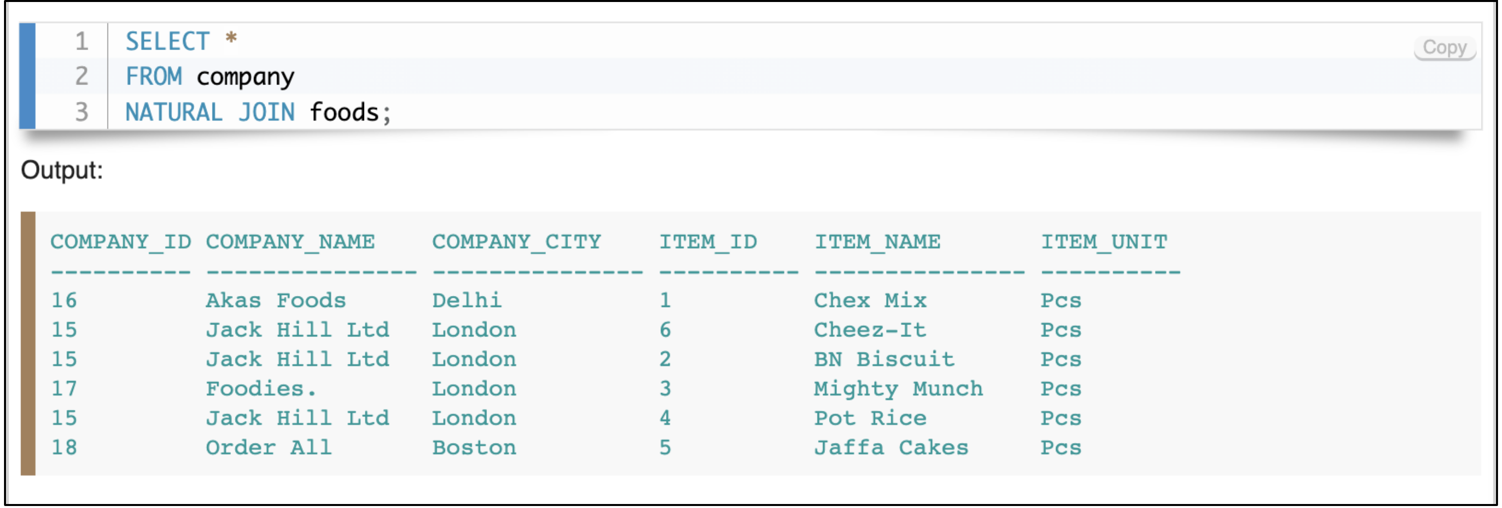
- Every tuple in right table must appear in result
- If no matching tuple
  - Padded with NULL values for attributes of left table



```sql
SELECT *
FROM company
INNER JOIN foods
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
```

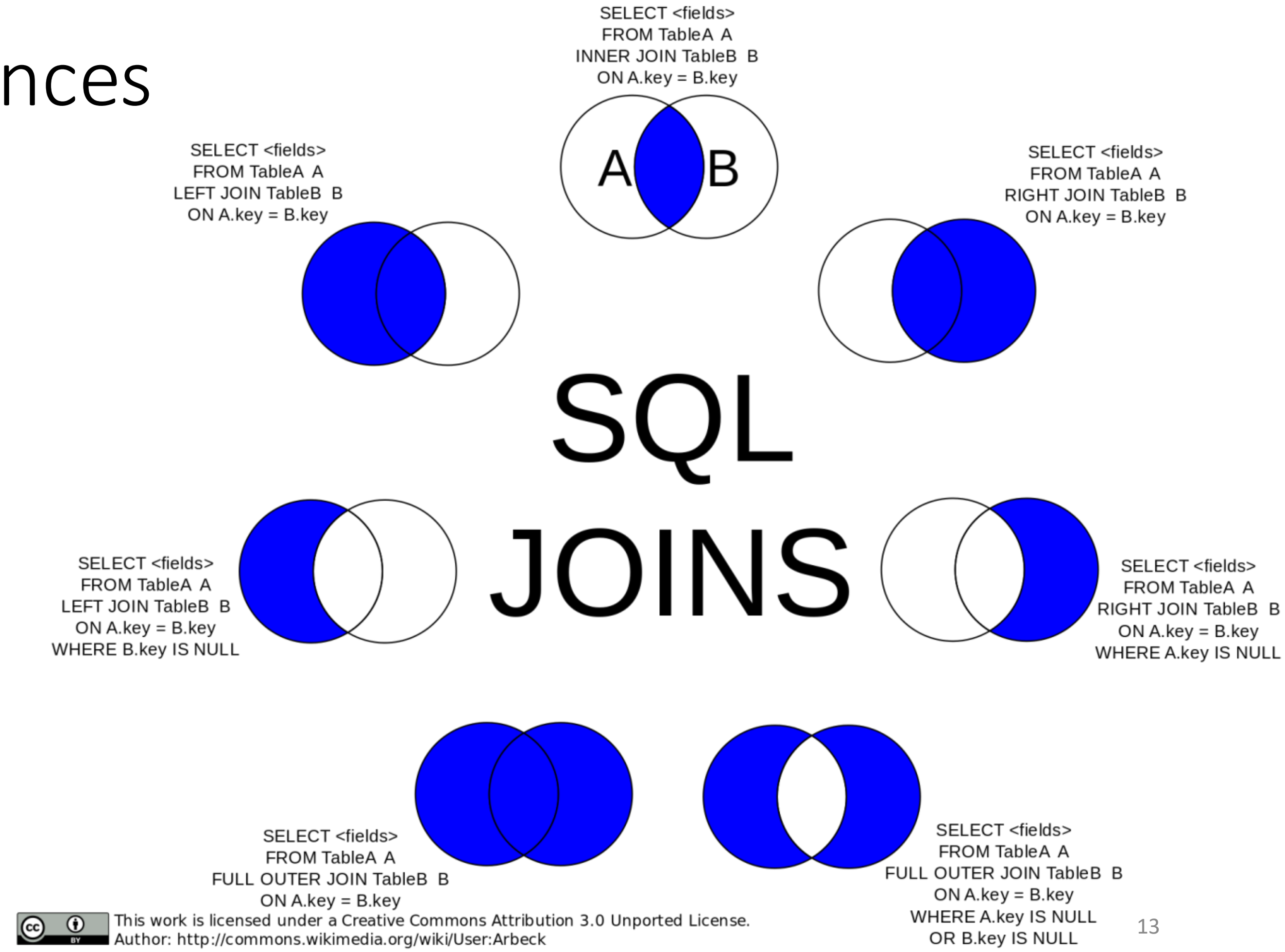


```sql
SELECT *
FROM company
NATURAL JOIN foods;
```

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
```

Joins differences



```
SELECT <fields>
FROM TableA A
INNER JOIN TableB B
ON A.key = B.key
```

```
SELECT <fields>
FROM TableA A
LEFT JOIN TableB B
ON A.key = B.key
```

```
SELECT <fields>
FROM TableA A
RIGHT JOIN TableB B
ON A.key = B.key
```

```
SELECT <fields>
FROM TableA A
LEFT JOIN TableB B
ON A.key = B.key
WHERE B.key IS NULL
```

```
SELECT <fields>
FROM TableA A
RIGHT JOIN TableB B
ON A.key = B.key
WHERE A.key IS NULL
```

```
SELECT <fields>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.key = B.key
```

```
SELECT <fields>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.key = B.key
WHERE A.key IS NULL
OR B.key IS NULL
```

https://i.stack.imgur.com/3bs7C.png

This Lecture

Multiway JOIN in the FROM clause

Can nest JOIN specifications for a multiway join:

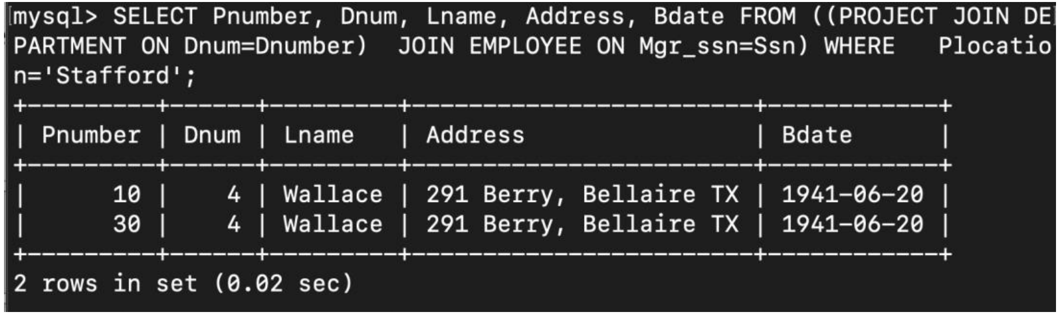
```
SELECT Pnumber, Dnum, Lname,
Address, Bdate FROM ((PROJECT
JOIN DEPARTMENT ON
Dnum=Dnumber) JOIN EMPLOYEE
ON Mgr_ssn=Ssn) WHERE
Plocation='Stafford';
```

Try it yourself!

Multiway JOIN in the FROM clause

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

Informal Design Guidelines for Relational Databases

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

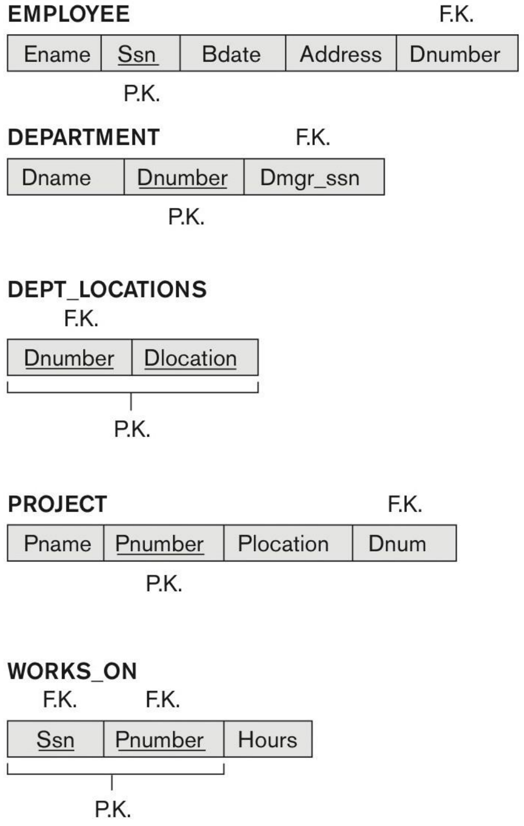
1.1 Semantics of the Relational Attributes must be clear

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

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

Figure 14.1 A simplified COMPANY relational database schema



EMPLOYEE F.K.

Ename | Ssn | Bdate | Address | Dnumber

P.K.

DEPARTMENT F.K.

Dname | Dnumber | Dmgr_ssn

P.K.

DEPT_LOCATIONS

F.K.

Dnumber | Dlocation

P.K.

PROJECT F.K.

Pname | Pnumber | Plocation | Dnum

P.K.

WORKS_ON

F.K. F.K.

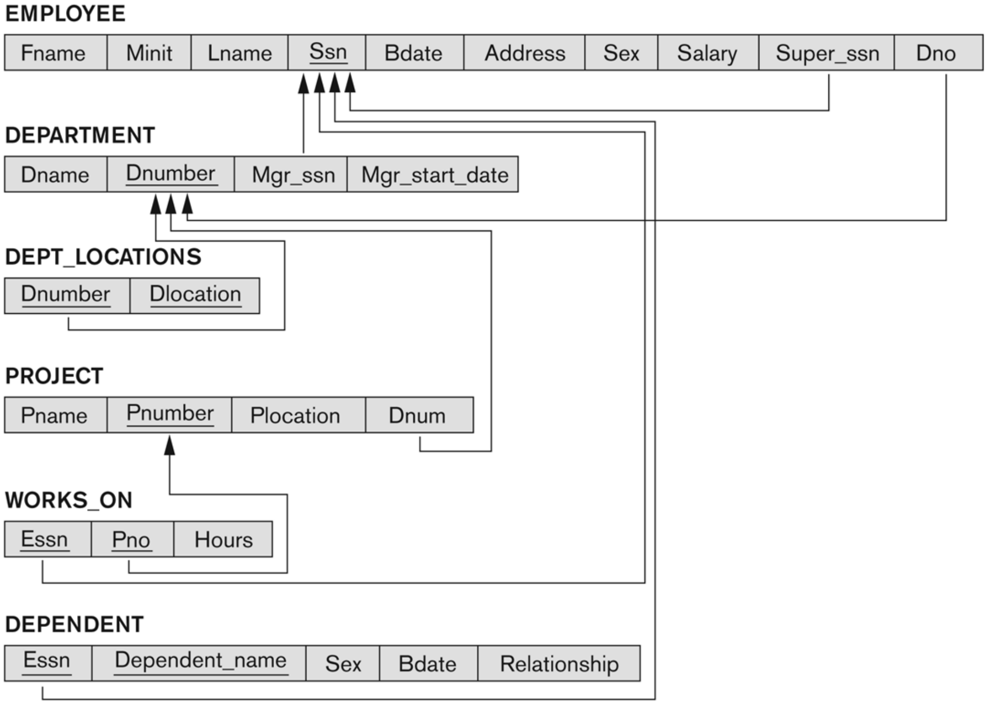
Ssn | Pnumber | Hours

P.K.

Figure 14.1 A simplified COMPANY relational database schema.

Figure 14.1 A simplified COMPANY relational database schema

Figure 5.7
Referential integrity constraints displayed on the COMPANY relational database schema.



EMPLOYEE

Fname | Minit | Lname | Ssn | Bdate | Address | Sex | Salary | Super_ssn | Dno

DEPARTMENT

Dname | Dnumber | Mgr_ssn | Mgr_start_date

DEPT_LOCATIONS

Dnumber | Dlocation

PROJECT

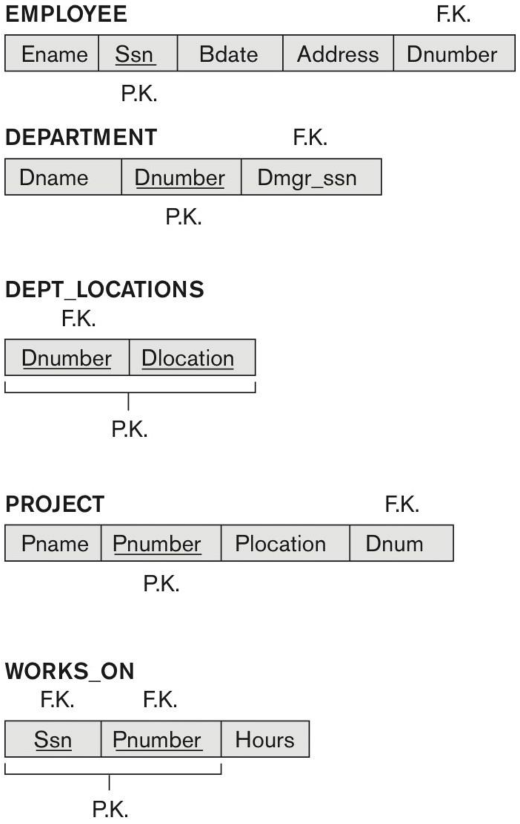
Pname | Pnumber | Plocation | Dnum

WORKS_ON

Essn | Pno | Hours

DEPENDENT

Essn | Dependent_name | Sex | Bdate | Relationship



EMPLOYEE F.K.

Ename | Ssn | Bdate | Address | Dnumber

P.K.

DEPARTMENT F.K.

Dname | Dnumber | Dmgr_ssn

P.K.

DEPT_LOCATIONS

F.K.

Dnumber | Dlocation

P.K.

PROJECT F.K.

Pname | Pnumber | Plocation | Dnum

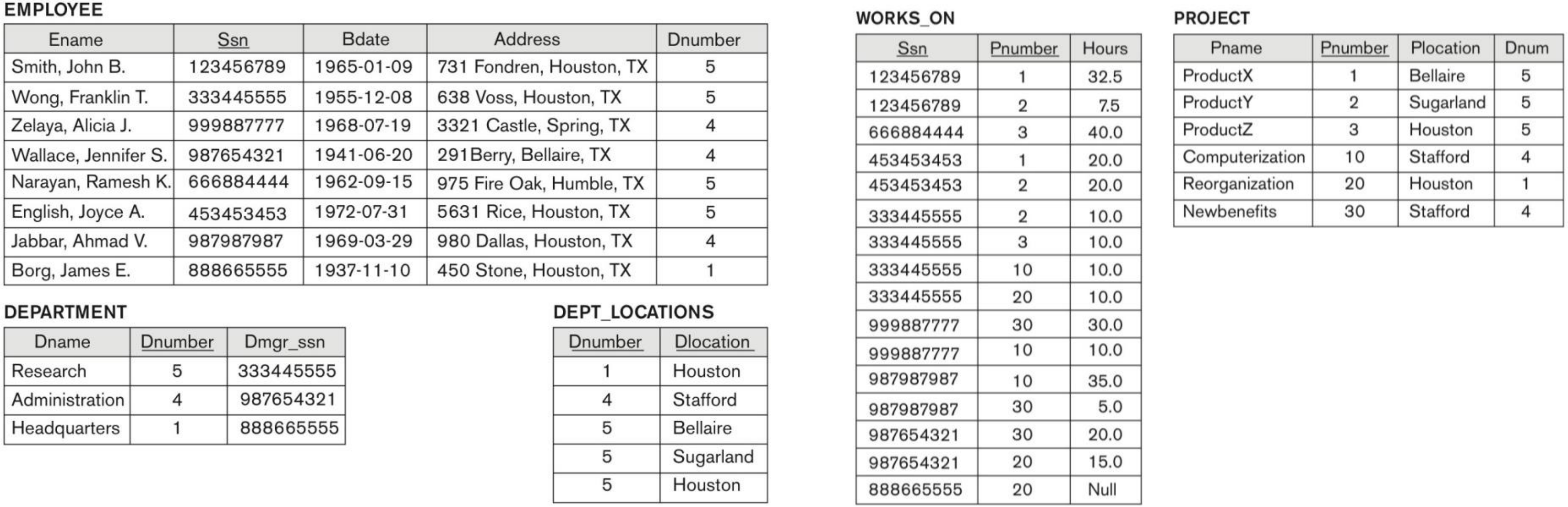
P.K.

WORKS_ON

F.K. F.K.

Ssn | Pnumber | Hours

P.K.



EMPLOYEE

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

DEPARTMENT

Dname	Dnumber	Dmgr_ssn
Research	5	333445555
Administration	4	987654321
Headquarters	1	888665555

DEPT_LOCATIONS

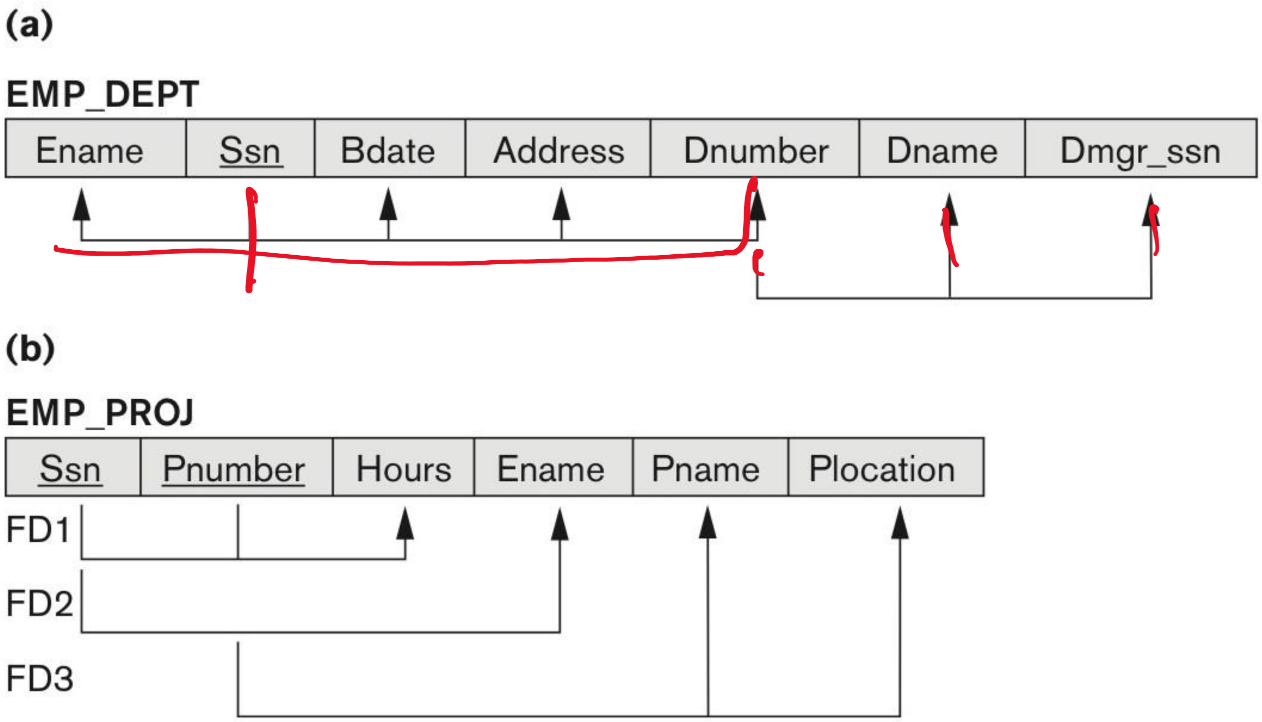
Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

WORKS_ON

Ssn	Pnumber	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	Null

PROJECT

Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4



(a)

EMP_DEPT

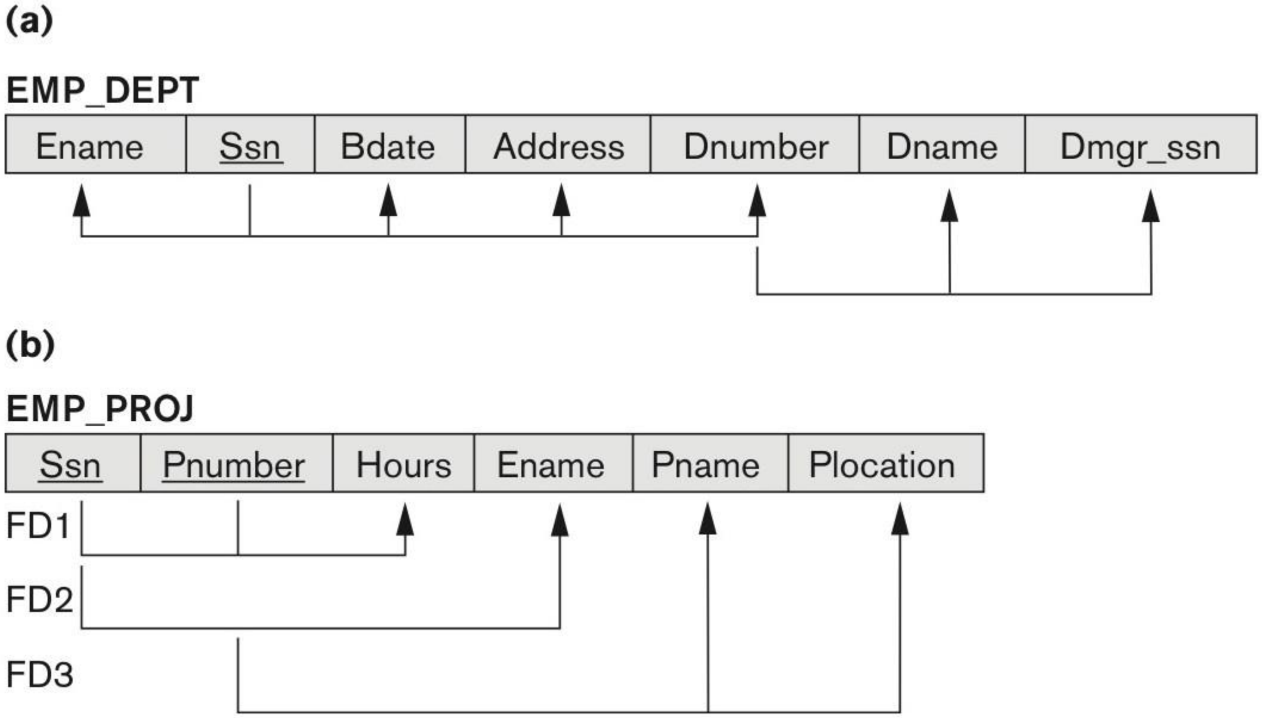
Ename	Ssn	Bdate	Address	Dnumber	Dname	Dmgr_ssn

(b)

EMP_PROJ

Ssn	Pnumber	Hours	Ename	Pname	Plocation

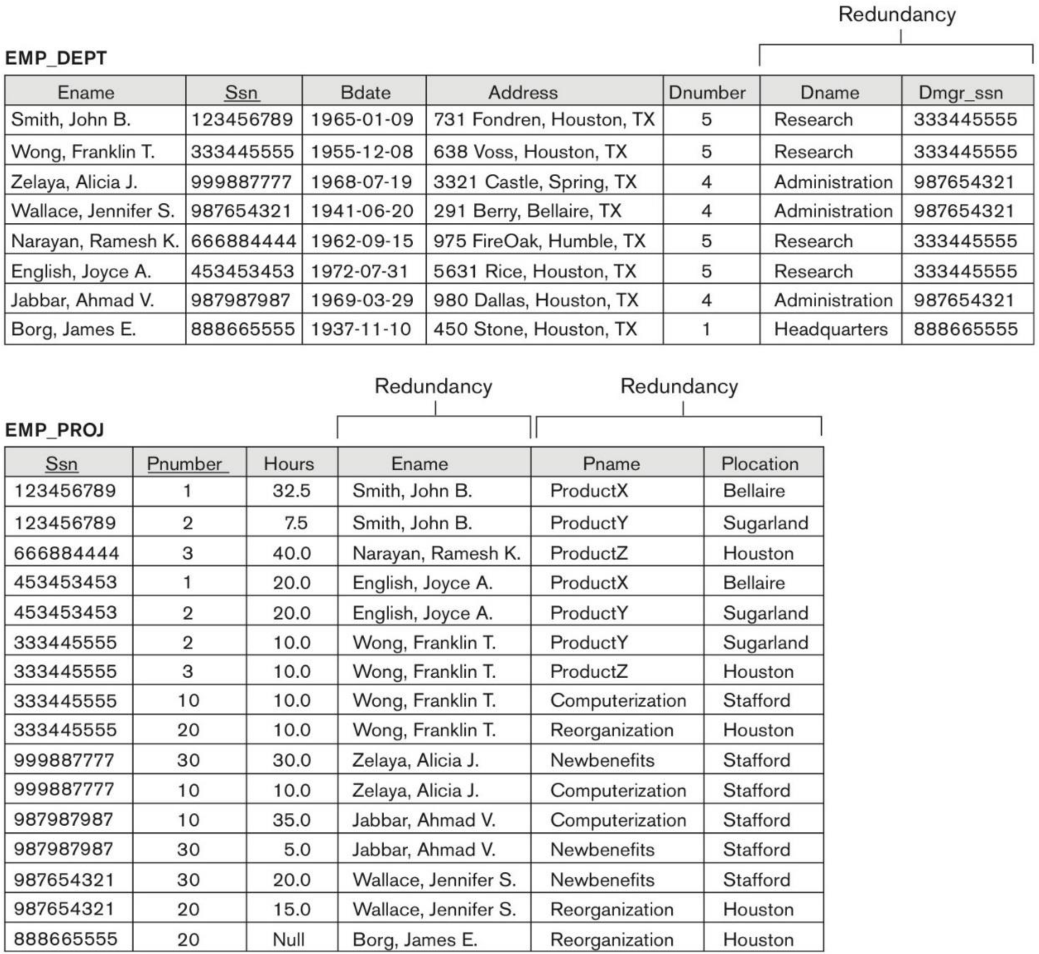
Any concerns here?



Any concerns here?

EMP_DEPT: mixing attributes of employees & departments

EMP_PROJ: mixes attributes of employees, projects & works_on



Redundancy
Redundancy
Redundancy

EMP_DEPT

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

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

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.

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]

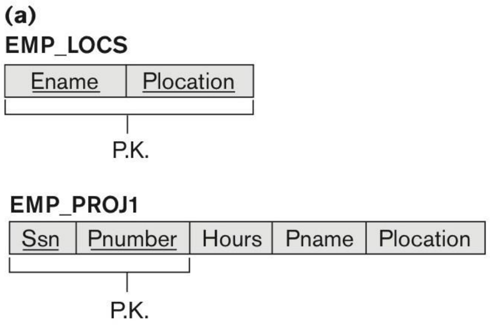
1.4 Generation of Spurious Tuples – avoid at any cost

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.

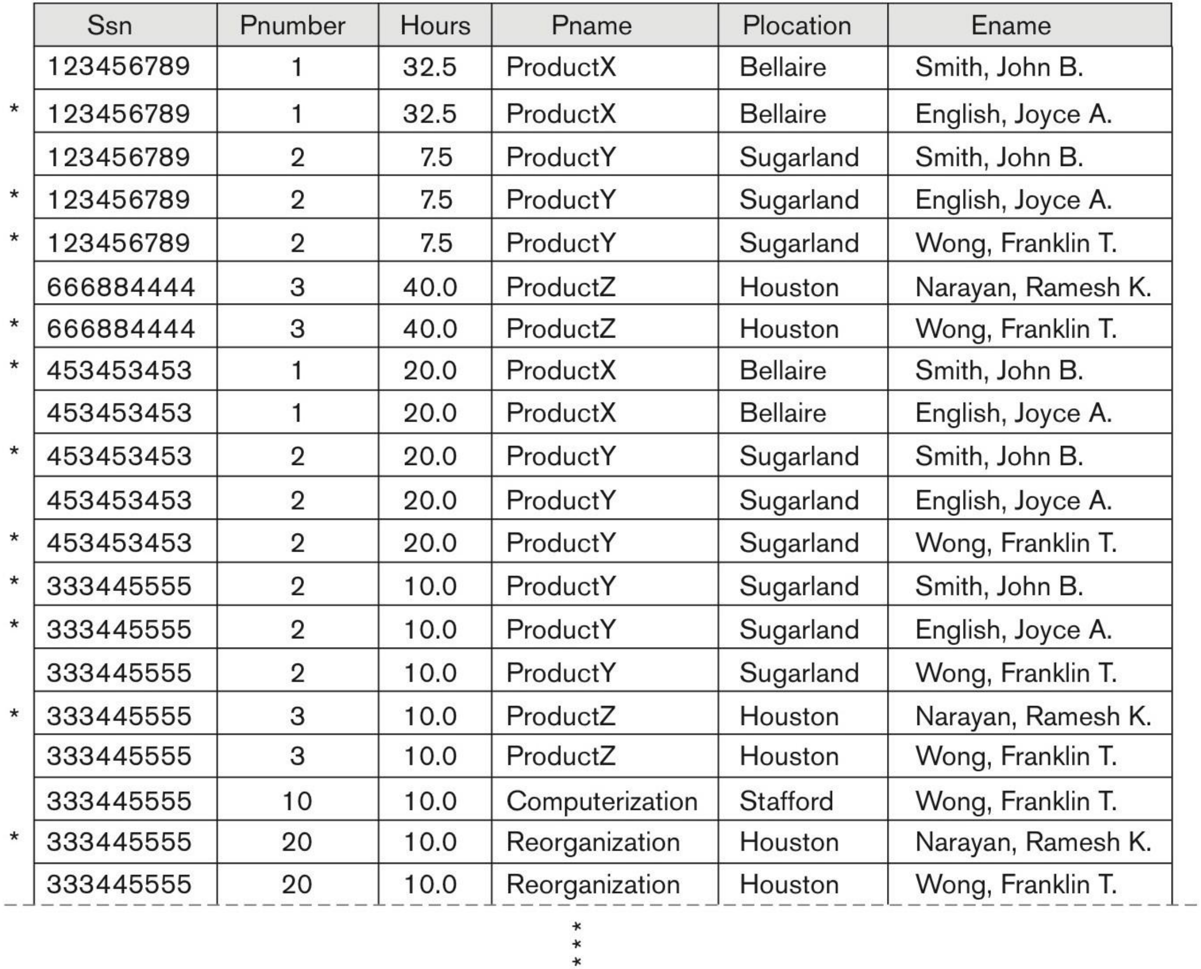
(b)

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

For any two tuples t1 and t2 in any relation instance r(R): If t1[X]=t2[X], *then* t1[Y]=t2[Y]

X → Y in R specifies a *constraint* on all relation instances r(R)

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

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

Examples of FD constraints (1)

Social security number determines employee name

SSN → ENAME

Project number determines project name and location

PNUMBER → {PNAME, PLOCATION}

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

{SSN, PNUMBER} → HOURS

Examples of FD constraints (1)

Social security number determines employee name

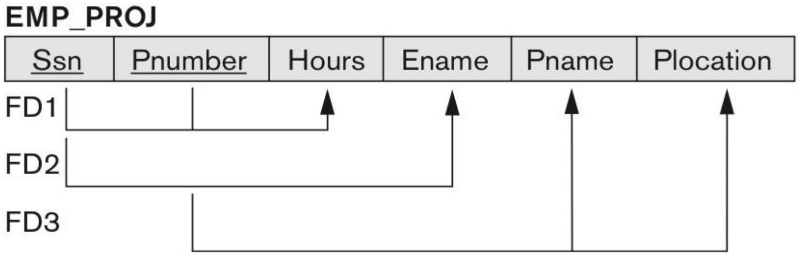
SSN → ENAME

Project number determines project name and location

PNUMBER → {PNAME, PLOCATION}

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

{SSN, PNUMBER} → 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

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.

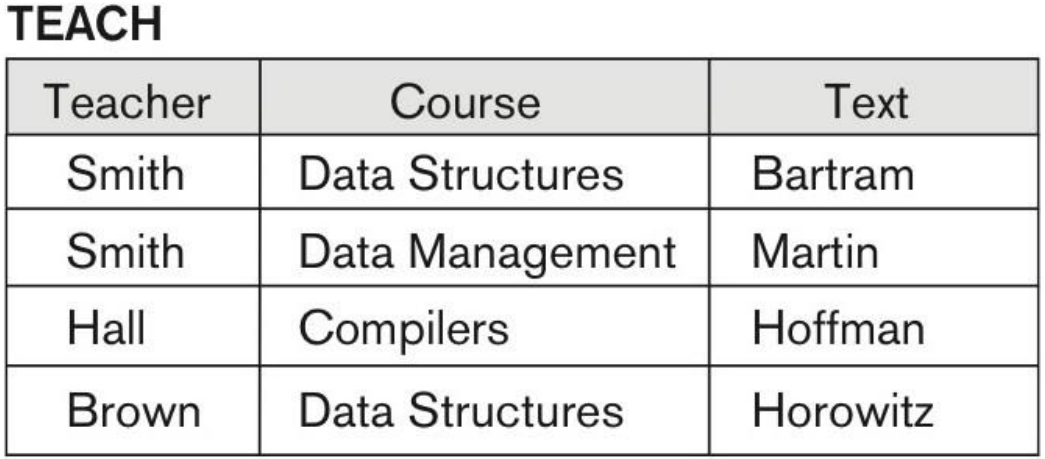
Ruling Out FDs

~~Teach → Course, Text~~

T, T → C

Text → Course

T, C → Text

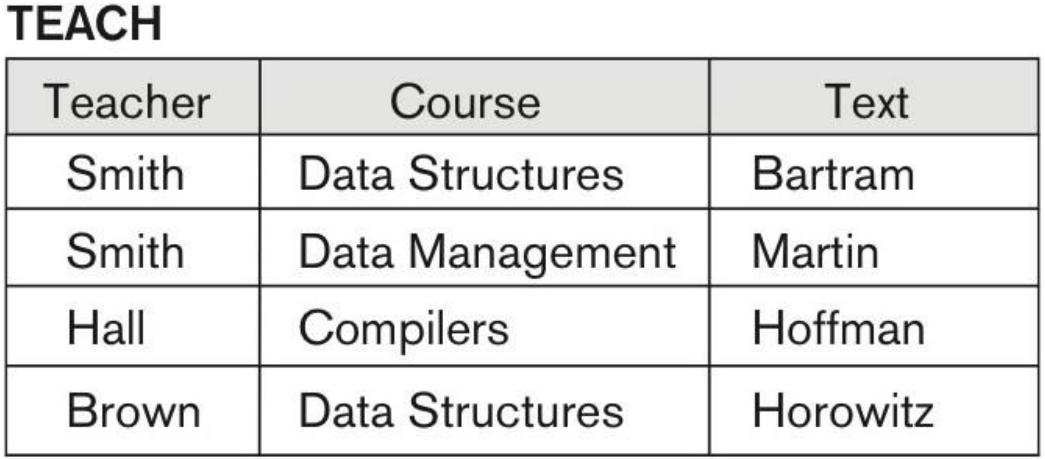


TEACH

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

Ruling Out FDs

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



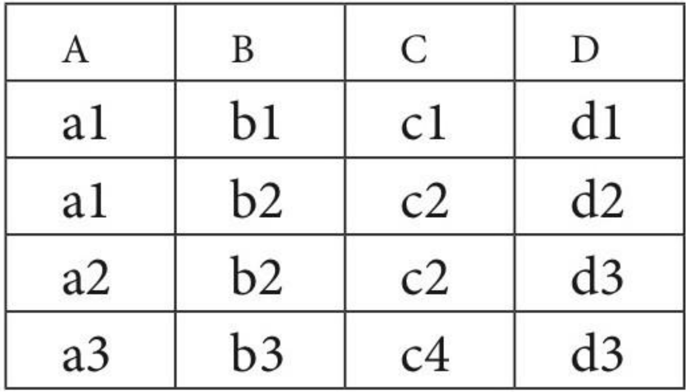
TEACH

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?

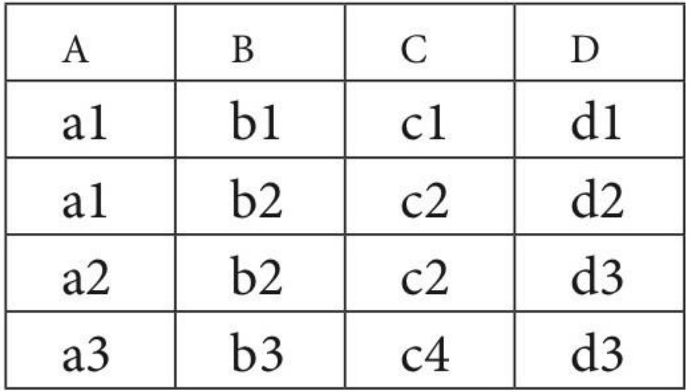


A	B	C	D
a1	b1	c1	d1
a1	b2	c2	d2
a2	b2	c2	d3
a3	b3	c4	d3

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 → C; C → B; {A,B} → C; {A,B} → D; {C,D}→ B

How about A → B? B→ A? D → 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

Administrativia

Quiz?

All marks done except Q3?

Quiz on 18th Topics between last quiz and topics to be covered on 18th also

End Sem on 25th 9 – 1~~2~~ ~~noon~~, entire syllabus

Quiz on 30th Entire syllabus – Please fill the form for taking the quiz

HARD

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 = {A1, A2, ..., An} is a set of attributes S *subset-of* R with the property that no two tuples t1 and t2 in any legal relation state r of R will have t1[S] = t2[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

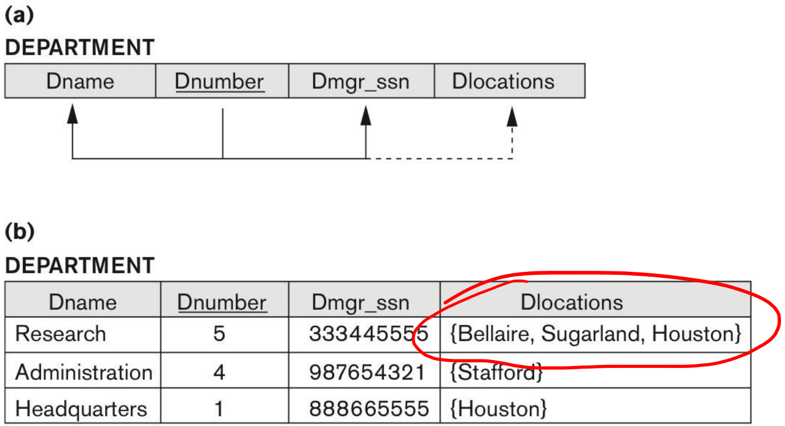
Disallows

- 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	Dnumber	Dmgr_ssn	Dlocations

(b)

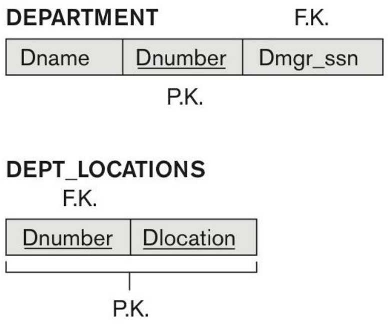
DEPARTMENT

Dname	Dnumber	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 F.K.

Dname	Dnumber	Dmgr_ssn

P.K.

DEPT_LOCATIONS

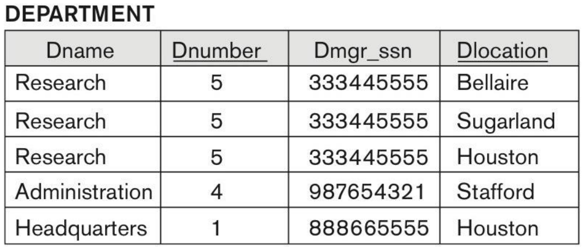
F.K.

Dnumber	Dlocation

P.K.

1 = Two 1NF relations

2 = Redundancy, Dnumber & Dlocation primary key

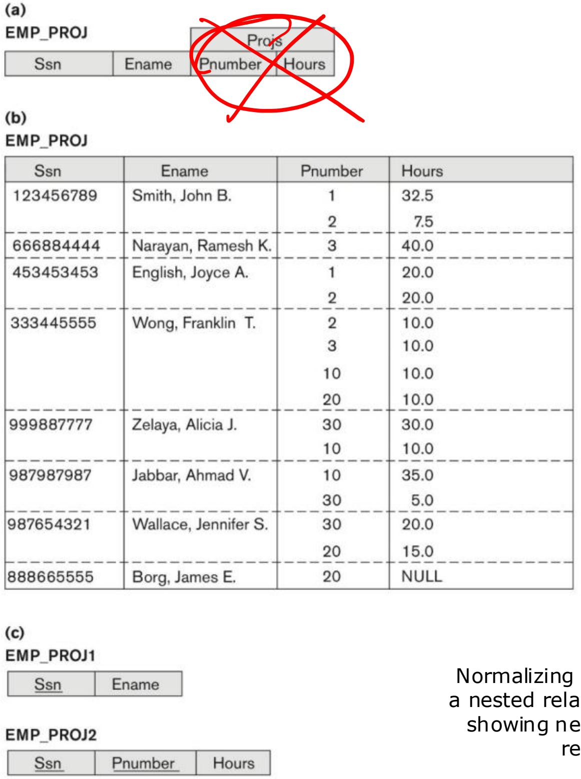


DEPARTMENT

Dname	Dnumber	Dmgr_ssn	Dlocation
Research	5	333445555	Bellaire
Research	5	333445555	Sugarland
Research	5	333445555	Houston
Administration	4	987654321	Stafford
Headquarters	1	888665555	Houston

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

(c)

EMP_PROJ1

Ssn	Ename

EMP_PROJ2

Ssn	Pnumber	Hours

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.

Ssn is the primary key, Pnumber is the partial key

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

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 -> Z where removal of any attribute from Y means the FD does not hold any more

Examples:

- {SSN, PNUMBER} -> HOURS is a full FD since neither SSN -> HOURS nor PNUMBER -> HOURS hold
- {SSN, PNUMBER} -> ENAME is not a full FD (it is called a partial dependency) since SSN -> 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”

Bibliography / Acknowledgements

Instructor materials from Elmasri & Navathe 7e

pk.profgiri

Ponnurangam.kumaraguru

/in/ponguru

ponguru

pk.guru@iiit.ac.in

Thank you
for attending
the class!!!