CS4.301 Data & Applications

Ponnurangam Kumaraguru (“PK”)
#ProfGiri @ IIIT Hyderabad

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

```
mysql> SELECT
    -> D.Dname, E.Lname, E.Fname, P.Pname
    -> FROM
    -> DEPARTMENT AS D, EMPLOYEE AS E, WORKS_ON AS W, PROJECT AS P
    -> WHERE
    -> D.Dnumber = E.Dno AND E.Ssn = W.Essn AND W.Pno = P.Pnumber
    -> ORDER BY
    -> D.Dname, E.Lname, E.Fname;
+----------------+---------+----------+-----------------+
| Dname          | Lname   | Fname    | Pname           |
+----------------+---------+----------+-----------------+
| Administration | Jabbar  | Ahmad    | Computerization |
| Administration | Jabbar  | Ahmad    | Newbenefits     |
| Administration | Wallace | Jennifer | Reorganization  |
| Administration | Wallace | Jennifer | Newbenefits     |
| Administration | Zelaya  | Alicia   | Computerization |
| Administration | Zelaya  | Alicia   | Newbenefits     |
| Headquarters   | Borg    | James    | Reorganization  |
| Research       | English | Joyce    | ProductX        |
| Research       | English | Joyce    | ProductY        |
| Research       | Narayan | Ramesh   | ProductZ        |
| Research       | Smith   | John     | ProductX        |
| Research       | Smith   | John     | ProductY        |
| Research       | Wong    | Franklin | ProductY        |
| Research       | Wong    | Franklin | ProductZ        |
| Research       | Wong    | Franklin | Computerization |
| Research       | Wong    | Franklin | Reorganization  |
+----------------+---------+----------+-----------------+
16 rows in set (0.02 sec)
```

The INSERT Command

Specify the relation name and a list of values for the tuple. All values including nulls are supplied.

U1: **INSERT INTO** EMPLOYEE
VALUES ('Richard', 'K', 'Marini', '653298653', '1962-12-30', '98 Oak Forest, Katy, TX', 'M', 37000, '653298653', 4);

```
mysql> INSERT INTO EMPLOYEE
    -> VALUES ('Richard', 'K', 'Marini', '653298653', '1962-12-30', '98 Oak Forest, Katy, TX', 'M', 37000, '653298653', 4);
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM EMPLOYEE;
+----------+-------+---------+-----------+------------+-------------------------+------+--------+-----------+-----+
| Fname    | Minit | Lname   | Ssn       | Bdate      | Address                 | Sex  | Salary | Super_ssn | Dno |
+----------+-------+---------+-----------+------------+-------------------------+------+--------+-----------+-----+
| John     | B     | Smith   | 123456789 | 1965-01-09 | 731 Fondren, Houston TX | M    |  30000 | 333445555 |   5 |
| Franklin | T     | Wong    | 333445555 | 1965-12-08 | 638 Voss, Houston TX    | M    |  40000 | 888665555 |   5 |
| Joyce    | A     | English | 453453453 | 1972-07-31 | 5631 Rice, Houston TX   | F    |  25000 | 333445555 |   5 |
| Richard  | K     | Marini  | 653298653 | 1962-12-30 | 98 Oak Forest, Katy, TX | M    |  37000 | 653298653 |   4 |
| Ramesh   | K     | Narayan | 666884444 | 1962-09-15 | 975 Fire Oak, Humble TX | M    |  38000 | 333445555 |   5 |
| James    | E     | Borg    | 888665555 | 1937-11-10 | 450 Stone, Houston TX   | M    |  55000 | NULL      |   1 |
| Jennifer | S     | Wallace | 987654321 | 1941-06-20 | 291 Berry, Bellaire TX  | F    |  43000 | 888665555 |   4 |
| Ahmad    | V     | Jabbar  | 987987987 | 1969-03-29 | 980 Dallas, Houston TX  | M    |  25000 | 987654321 |   4 |
| Alicia   | J     | Zelaya  | 999887777 | 1968-01-19 | 3321 Castle, Spring TX  | F    |  25000 | 987654321 |   4 |
+----------+-------+---------+-----------+------------+-------------------------+------+--------+-----------+-----+
9 rows in set (0.00 sec)
```

```
mysql> CREATE TABLE WORKS_ON_INFO (
    ->     Emp_name VARCHAR(150),
    ->     Proj_name VARCHAR(150),
    ->     Hours_per_week DECIMAL(3, 1)
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql> INSERT INTO WORKS_ON_INFO ( Emp_name, Proj_name, Hours_per_week )
    -> SELECT E.Lname, P.Pname, W.Hours
    -> FROM PROJECT P, WORKS_ON W, EMPLOYEE E
    -> WHERE P.Pnumber = W.Pno AND W.Essn = E.Ssn;
Query OK, 16 rows affected (0.02 sec)
Records: 16  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM WORKS_ON_INFO;
+----------+-----------------+----------------+
| Emp_name | Proj_name       | Hours_per_week |
+----------+-----------------+----------------+
| Wong     | Computerization |           10.0 |
| Jabbar   | Computerization |           35.0 |
| Zelaya   | Computerization |           10.0 |
| Wallace  | Newbenefits     |           20.0 |
| Jabbar   | Newbenefits     |            5.0 |
| Zelaya   | Newbenefits     |           30.0 |
| Smith    | ProductX        |           32.5 |
| English  | ProductX        |           20.0 |
| Smith    | ProductY        |            7.5 |
| Wong     | ProductY        |           10.0 |
| English  | ProductY        |           20.0 |
| Wong     | ProductZ        |           10.0 |
| Narayan  | ProductZ        |           40.0 |
| Wong     | Reorganization  |           10.0 |
| Borg     | Reorganization  |           16.0 |
| Wallace  | Reorganization  |           15.0 |
+----------+-----------------+----------------+
16 rows in set (0.00 sec)
```

The DELETE Command

Removes tuples from a relation

Includes a `WHERE` clause to select the tuples to be deleted. The number of tuples deleted will vary.

```
U4A:  DELETE FROM  EMPLOYEE
      WHERE        Lname='Brown';

U4B:  DELETE FROM  EMPLOYEE
      WHERE        Ssn='123456789';

U4C:  DELETE FROM  EMPLOYEE
      WHERE        Dno=5;

U4D:  DELETE FROM  EMPLOYEE;
```

```
mysql> UPDATE PROJECT
    -> SET PLOCATION = 'Bellaire', DNUM = 5
    -> WHERE PNUMBER = 10;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM PROJECT;
+-----------------+---------+-----------+------+
| Pname           | Pnumber | Plocation | Dnum |
+-----------------+---------+-----------+------+
| ProductX        |       1 | Bellaire  |    5 |
| ProductY        |       2 | Sugarland |    5 |
| ProductZ        |       3 | Houston   |    5 |
| Computerization |      10 | Bellaire  |    5 |
| Reorganization  |      20 | Houston   |    1 |
| Newbenefits     |      30 | Stafford  |    4 |
+-----------------+---------+-----------+------+
6 rows in set (0.00 sec)
```

UPDATE (contd.)

Example: Give all employees in the 'Research' department a 10% raise in salary.

```
mysql> UPDATE EMPLOYEE
    -> SET SALARY = SALARY * 1.1
    -> WHERE DNO IN (
    ->     SELECT DNUMBER
    ->     FROM DEPARTMENT
    ->     WHERE DNAME = 'Research'
    -> );
Query OK, 4 rows affected (0.02 sec)
Rows matched: 4  Changed: 4  Warnings: 0

mysql> SELECT * FROM EMPLOYEE;
+----------+-------+---------+-----------+------------+--------------------------+------+--------+-----------+-----+
| Fname    | Minit | Lname   | Ssn       | Bdate      | Address                  | Sex  | Salary | Super_ssn | Dno |
+----------+-------+---------+-----------+------------+--------------------------+------+--------+-----------+-----+
| John     | B     | Smith   | 123456789 | 1965-01-09 | 731 Fondren, Houston TX  | M    |  33000 | 333445555 |   5 |
| Franklin | T     | Wong    | 333445555 | 1965-12-08 | 638 Voss, Houston TX     | M    |  44000 | 888665555 |   5 |
| Joyce    | A     | English | 453453453 | 1972-07-31 | 5631 Rice, Houston TX    | F    |  27500 | 333445555 |   5 |
| Richard  | K     | Marini  | 653298653 | 1962-12-30 | 98 Oak Forest, Katy, TX  | M    |  37000 | 653298653 |   4 |
| Ramesh   | K     | Narayan | 666884444 | 1962-09-15 | 975 Fire Oak, Humble TX  | M    |  41800 | 333445555 |   5 |
| James    | E     | Borg    | 888665555 | 1937-11-10 | 450 Stone, Houston TX    | M    |  55000 | NULL      |   1 |
| Jennifer | S     | Wallace | 987654321 | 1941-06-20 | 291 Berry, Bellaire TX   | F    |  43000 | 888665555 |   4 |
| Ahmad    | V     | Jabbar  | 987987987 | 1969-03-29 | 980 Dallas, Houston TX   | M    |  25000 | 987654321 |   4 |
| Alicia   | J     | Zelaya  | 999887777 | 1968-01-19 | 3321 Castle, Spring TX   | F    |  25000 | 987654321 |   4 |
+----------+-------+---------+-----------+------------+--------------------------+------+--------+-----------+-----+
9 rows in set (0.00 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)
```

EMPLOYEE

emp_id	fname	lname	dno
1	John	Doe	10
2	Alice	Lee	20
3	Bob	Kim	10

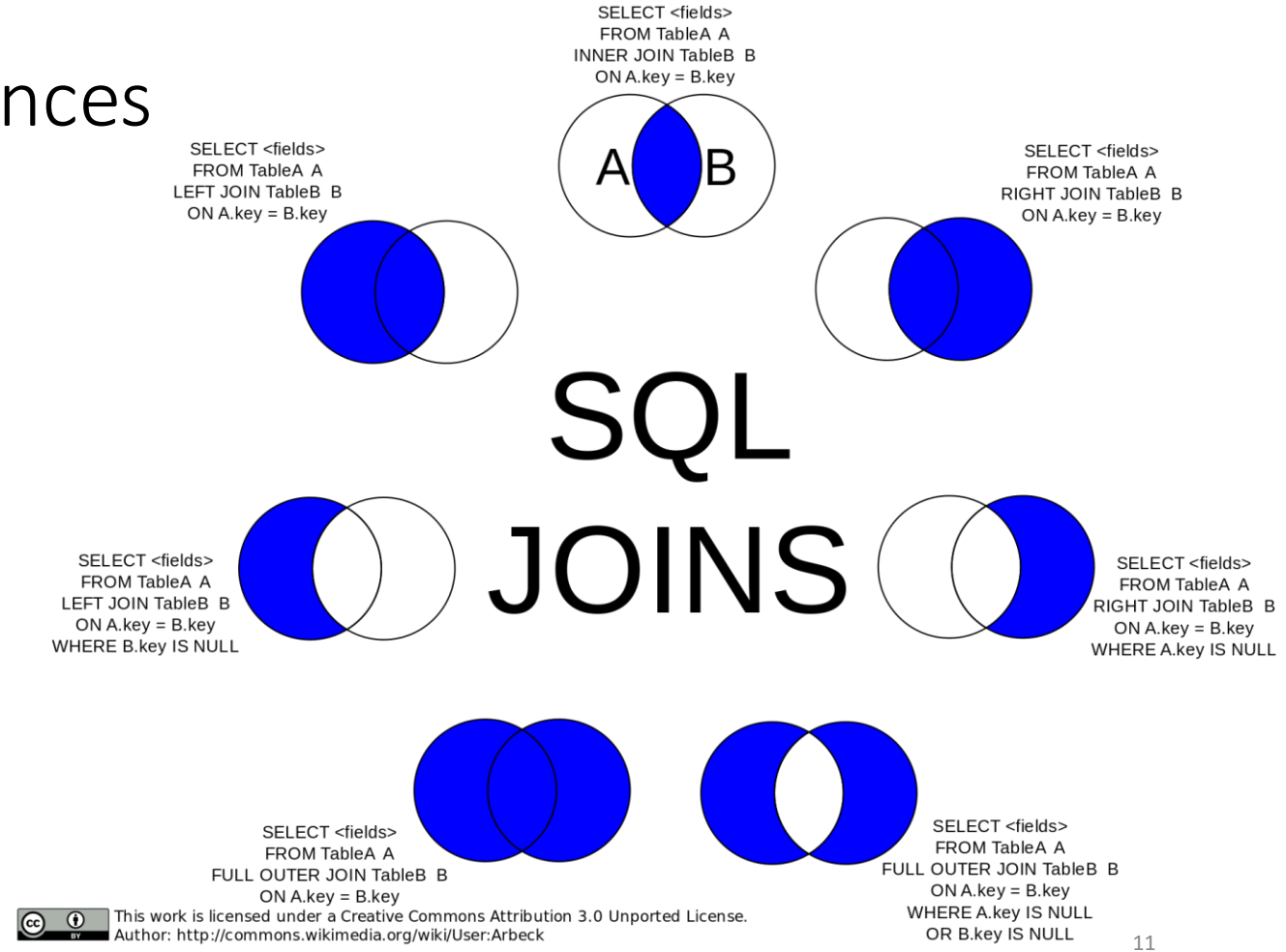
DEPARTMENT

dno	dname	location
10	Research	Delhi
20	HR	Mumbai
30	Finance	Chennai

```
SELECT fname, lname, dname, location
FROM employee NATURAL JOIN department;
```

fname	lname	dname	location
John	Doe	Research	Delhi
Bob	Kim	Research	Delhi
Alice	Lee	HR	Mumbai

Joins differences



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

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)
```

Activity

Try all the queries all yourself if you have not tried it

Write 3 join statements using any of the Company DB
Submit the query and results

Try the Multiway JOIN statement with any of the Company DB

Submit the query and results

This Lecture

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)

1.1 Semantics of the Relational Attributes must be clear

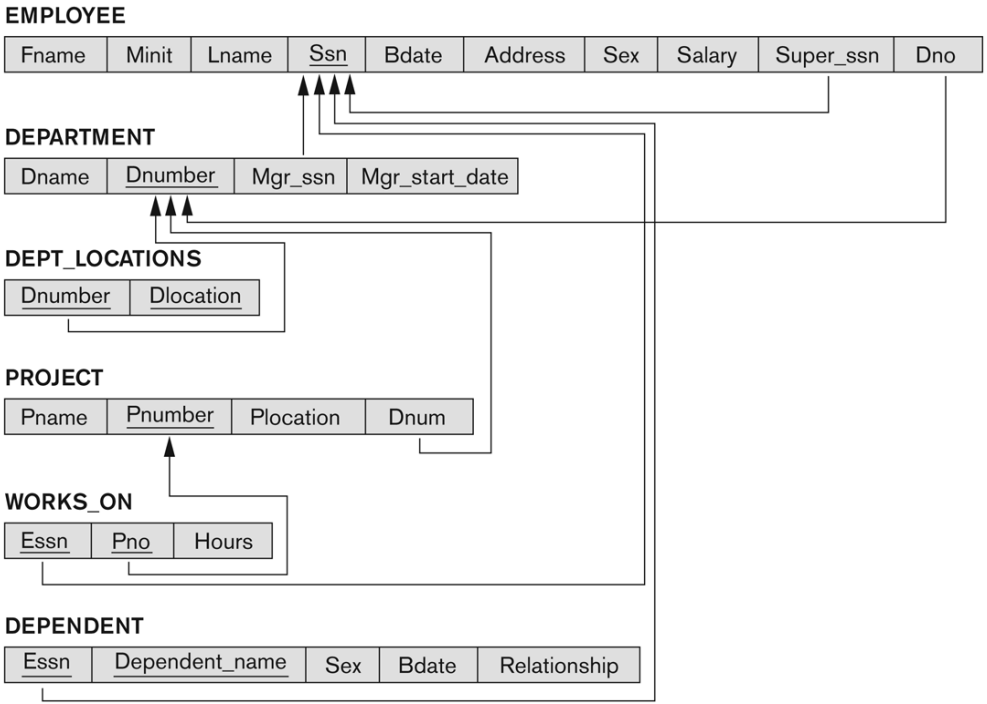
GUIDELINE 1: Informally, each tuple in a relation should represent one entity or relationship instance. (Applies to individual relations and their attributes).

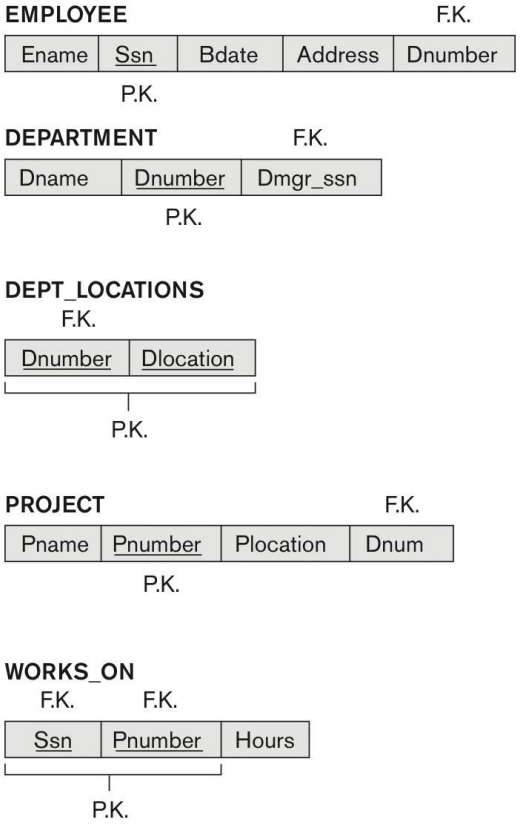
- Attributes of different entities (EMPLOYEEs, DEPARTMENTs, PROJECTs) should not be mixed in the same relation
- Only foreign keys should be used to refer to other entities

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

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





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

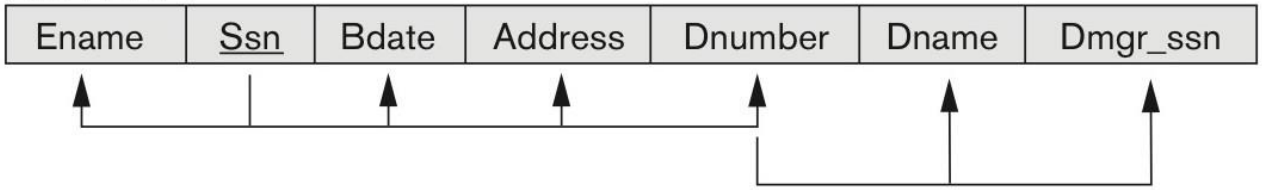
Ssn	Pnumber	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	Null

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

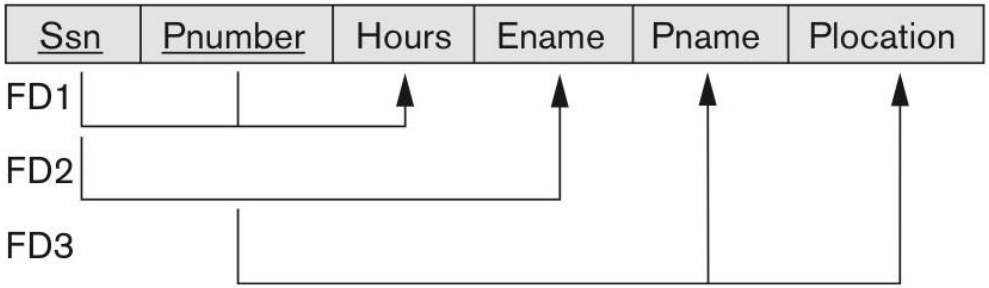
(a)

EMP_DEPT



(b)

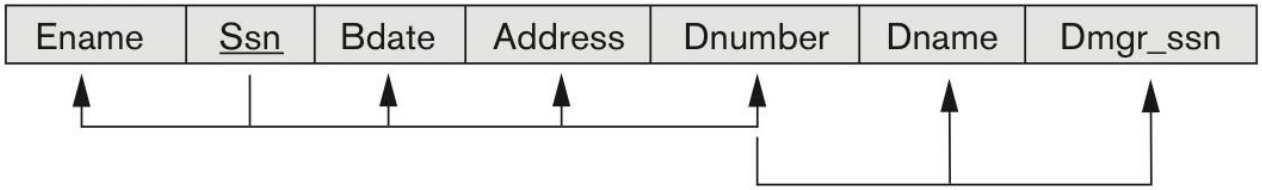
EMP_PROJ



Any concerns here?

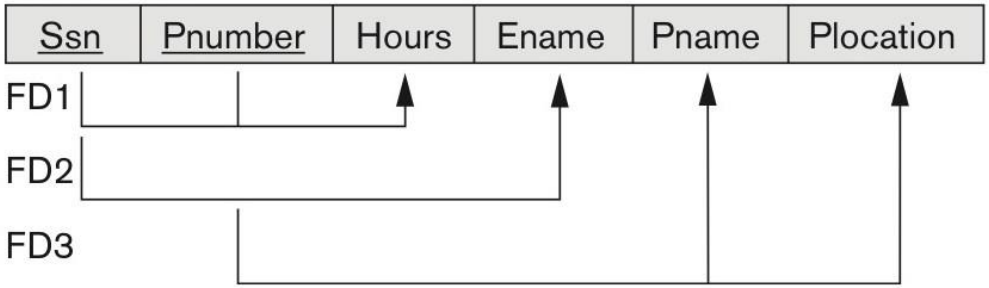
(a)

EMP_DEPT



(b)

EMP_PROJ



Any concerns here?

EMP_DEPT: mixing attributes of employees & departments

EMP_PROJ: mixing 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.

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

Redundancy Redundancy

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

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.

EXAMPLE OF AN UPDATE ANOMALY

Emp_ID	Emp_Name	Dept_Name	Dept_Location
101	Ravi	HR	Delhi
102	Neha	HR	Delhi
103	Amit	IT	Mumbai

EXAMPLE OF AN UPDATE ANOMALY

Emp_ID	Emp_Name	Dept_Name	Dept_Location
101	Ravi	HR	Delhi
102	Neha	HR	Delhi
103	Amit	IT	Mumbai

Emp_ID	Emp_Name	Dept_Name	Dept_Location
101	Ravi	HR	Delhi ❌
102	Neha	HR	Noida ✅
103	Amit	IT	Mumbai

```
UPDATE Employee
SET Dept_Location = 'Noida'
WHERE Emp_ID = 102;
```

EXAMPLE OF AN UPDATE ANOMALY

Partial or inconsistent updates (human or system error)

Multiple users or applications

Real-world DBs are complex

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]

1.3 Null Values in Tuples

Poor design:

Employee(EmpID, Name, Department, Salary, Commission, Bonus)

Why? Better design?

1.3 Null Values in Tuples

Poor design:

Employee(EmpID, Name, Department, Salary, Commission, Bonus)

Why? Better design?

Employee(EmpID, Name, Department, Salary)

SalesCommission(EmpID, Commission)

BonusPayment(EmpID, Bonus)

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

Ename	Plocation

P.K.

EMP_PROJ1

Ssn	Pnumber	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

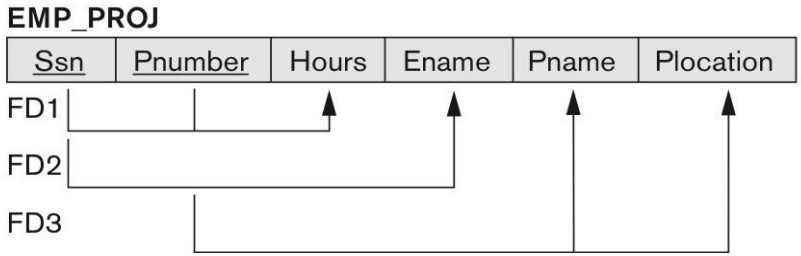
SSN → ENAME

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.

What can we say about FDs?

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), 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), 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

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

Activity: 10th Nov

Infinium, Develop 2 examples relation where where mixing of attributes exist?

Give an example of INSERT, DELETE, UPDATE anomaly in Infinium DB

Write at least 2 FDs that may exist in Infinium DB
Write at least 2 FDs that may not exist in Infinium DB

Write 2 prime attributes in Infinium DB
Write 2 non-prime attributes in Infinium DB

Administrativia

Quiz 3 on 17th Nov

How many will take the Bonus quiz? We will do best of 4; Bonus will be all topics covered and expected to be hard

Bibliography / Acknowledgements

Instructor materials from Elmasri & Navathe 7e


pk.profgiri


Ponnurangam.kumaraguru


/in/ponguru


ponguru

Thank you for attending the class!!!


pk.guru@iiit.ac.in