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

Grouping: The GROUP BY and HAVING Clauses (cont'd.)

`HAVING` clause

Provides a condition to select or reject an entire group:

Query 26. For each project *on which more than two employees work,* retrieve the project number, the project name, and the number of employees who work on the project.

```
SELECT Pnumber,
Pname, COUNT(*) FROM
PROJECT, WORKS_ON
WHERE Pnumber=Pno
GROUP BY Pnumber
HAVING COUNT(*) > 2;
```

```
mysql> SELECT Pnumber, Pname, COUNT(*) FROM PROJECT, WORKS_O
N WHERE Pnumber=Pno GROUP BY Pnumber HAVING COUNT(*) > 2;
+---------+-----------------+----------+
| Pnumber | Pname           | COUNT(*) |
+---------+-----------------+----------+
|       2 | ProductY        |        3 |
|      10 | Computerization |        3 |
|      20 | Reorganization  |        3 |
|      30 | Newbenefits     |        3 |
+---------+-----------------+----------+
4 rows in set (0.00 sec)
```

EXPANDED Block Structure of SQL Queries

SELECT <attribute and function list>
FROM <table list>
[**WHERE** <condition>]
[**GROUP BY** <grouping attribute(s)>]
[**HAVING** <group condition>]
[**ORDER BY** <attribute list>];

Views (Virtual Tables) in SQL

Concept of a view in SQL

- Single table derived from other tables called the **defining tables**
- Considered to be a virtual table that is not necessarily populated
- Therefore limits update operations, no limitations in querying
- In COMPANY we may frequently retrieve project name & employee name which is joining employee, works_on, project create a view and query this single table retrieval than multiple tables

Specification of Views in SQL

`CREATE VIEW` command

Give table name, list of attribute names, and a query to specify the contents of the view

Create view works_on1 as select fname, lname, hours from employee, project, works_on where ssn=essn and pno=pnumber;

```
mysql> Create view works_on1 as select fname, lname, hours f
rom employee, project, works_on where ssn=essn and pno=pnumb
er;
Query OK, 0 rows affected (0.05 sec)
```

Data in view, query view

```
select * from works_on1;
```

```
mysql> select * from works_on1;
+----------+---------+-------+
| fname    | lname   | hours |
+----------+---------+-------+
| Franklin | Wong    |  10.0 |
| James    | Borg    |  16.0 |
| Jennifer | Wallace |  15.0 |
| Franklin | Wong    |  10.0 |
| Ahmad    | Jabbar  |  35.0 |
| Alicia   | Zelaya  |  10.0 |
| Jennifer | Wallace |  20.0 |
| Ahmad    | Jabbar  |   5.0 |
| Alicia   | Zelaya  |  30.0 |
| John     | Smith   |  32.5 |
| Joyce    | English |  20.0 |
| John     | Smith   |   7.5 |
| Franklin | Wong    |  10.0 |
| Joyce    | English |  20.0 |
| Franklin | Wong    |  10.0 |
| Ramesh   | Narayan |  40.0 |
+----------+---------+-------+
16 rows in set (0.01 sec)
```

select fname, lname from works_on1 where hours=10;

```
mysql> select fname, lname from works_on1 where hours=10;
+----------+--------+
| fname    | lname  |
+----------+--------+
| Franklin | Wong   |
| Franklin | Wong   |
| Franklin | Wong   |
| Franklin | Wong   |
| Alicia   | Zelaya |
+----------+--------+
5 rows in set (0.01 sec)
```

Specification of Views in SQL (cont'd.)

Once a View is defined, SQL queries can use the View relation in the FROM clause

View is always up-to-date

- Responsibility of the DBMS and not the user

`DROP VIEW` command

- Dispose of a view

View Implementation, View Update, and Inline Views

Complex problem of efficiently implementing a view for querying

Strategy1: Query modification approach

- Compute the view as and when needed. Do not store permanently
- Modify view query into a query on underlying base tables
- Disadvantage: inefficient for views defined via complex queries that are time-consuming to execute

View Materialization

Strategy 2: View materialization

- Physically create a temporary view table when the view is first queried
- Keep that table on the assumption that other queries on the view will follow
- Requires efficient strategy for automatically updating the view table when the base tables are updated

Incremental update strategy for materialized views

- DBMS determines what new tuples must be inserted, deleted, or modified in a materialized view table

View Materialization (contd.)

Multiple ways to handle materialization:

immediate update strategy updates a view as soon as the base tables are changed

lazy update strategy updates the view when needed by a view query

periodic update strategy updates the view periodically (in the latter strategy, a view query may get a result that is not up-to-date). This is commonly used in Banks, Retail store operations, etc.

Schema Change Statements in SQL

Schema evolution commands

DBA may want to change the schema while the database is operational

Does not require recompilation of the database schema

The DROP Command

`DROP` command

Used to drop named schema elements, such as tables, domains, or constraint

Drop behavior options:

`CASCADE` and `RESTRICT`

RESTRICT – schema will be dropped only if it has no elements in it

Example:

```
DROP SCHEMA COMPANY CASCADE;
```

This removes the schema and all its elements including tables, views, constraints, etc.

```
DROP TABLE DEPENDENT CASCADE;
```

If we no longer wish to track the dependents

The DROP Command

Not only deletes all the records in the table if successful, removes the table definition from catalog

This Lecture

Last class question

DROP Schema – Only the mentioned schema dropped; there may be other schema in the DB which don’t get dropped

DROP Database – All schemas in DB dropped

The ALTER table command [Name for today?]

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 and UNIQUE 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

SQL function `UNIQUE(Q)`

Returns `TRUE` if there are no duplicate tuples in the result of query Q

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

Rename attributes of one relation so it can be joined with another using NATURAL 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;

The above works with EMPLOYEE.Dno = DEPT.Dno as an implicit join condition

```
mysql> select Fname, Lname, Address FROM (EMPLOYEE NATURAL JOIN DEPARTME
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

Natural join & Inner join difference

Number of columns returned

```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;
```

Copy

Output:

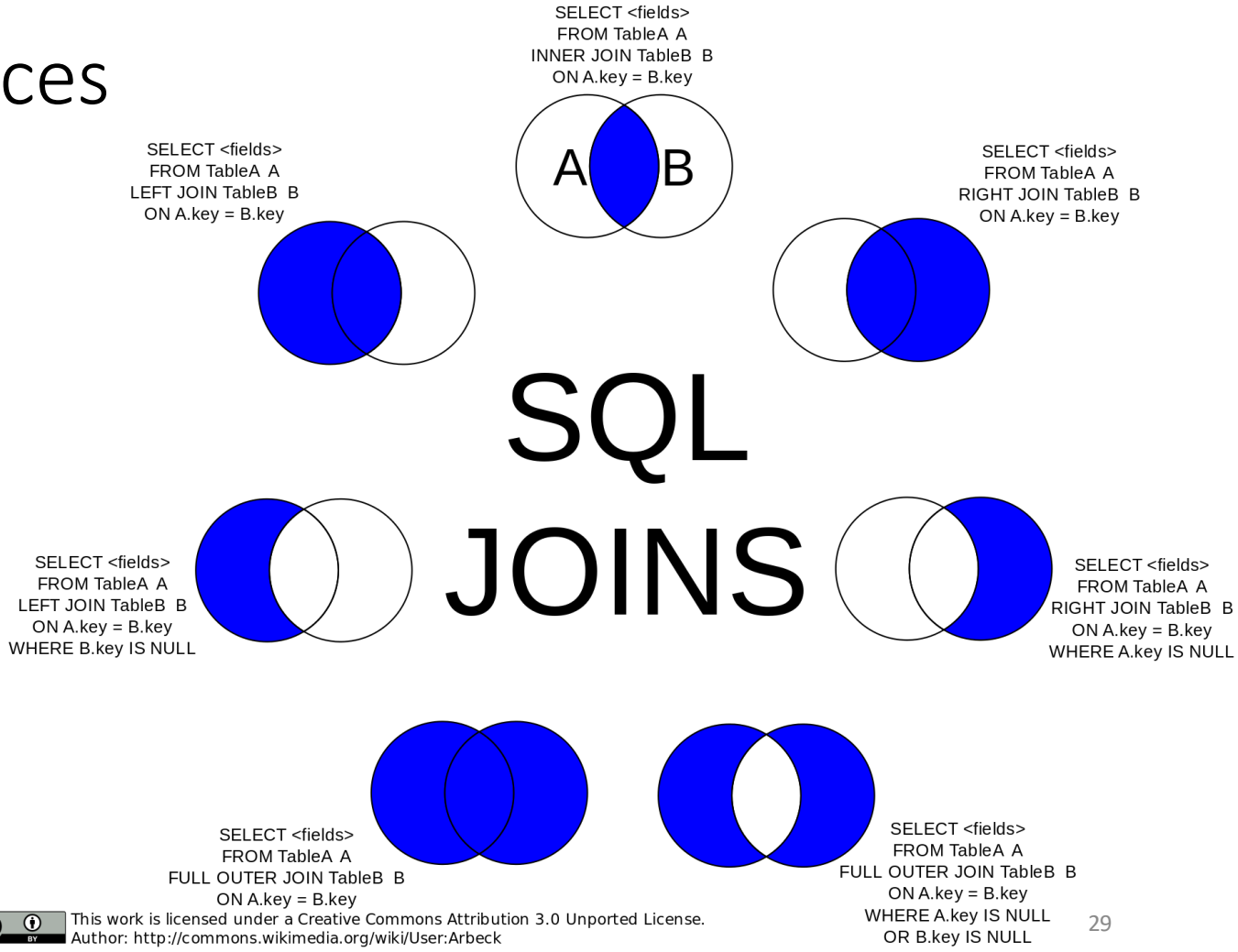
```
COMPANY_ID COMPANY_NAME    COMPANY_CITY    ITEM_ID    ITEM_NAME       ITEM_UNIT
---------- --------------- --------------- ---------- --------------- ----------
16         Akas Foods      Delhi           1          Chex Mix        Pcs
15         Jack Hill Ltd   London          6          Cheez-It        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
MPLOYEE 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



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

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 Noferferferfewrmal 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.*

Figure 14.1 A simplified COMPANY relational database schema

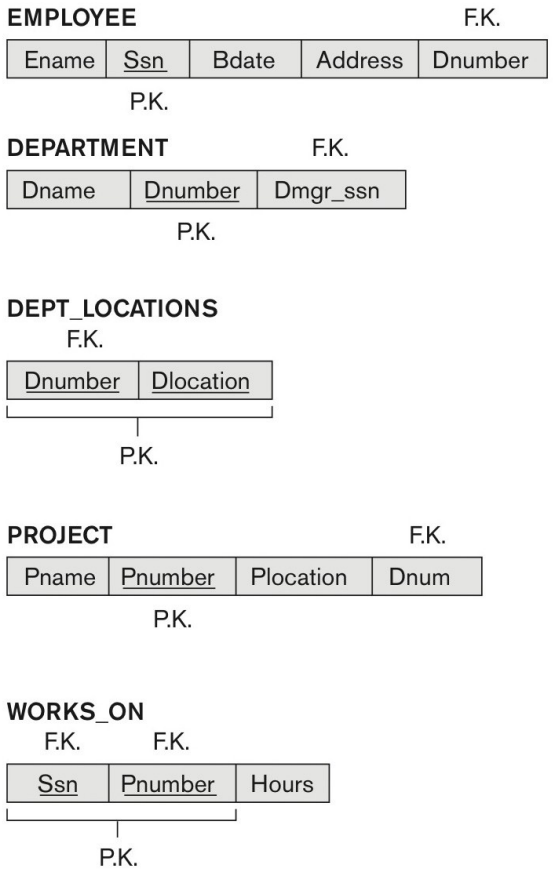


Figure 14.1 A simplified COMPANY relational database schema.

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

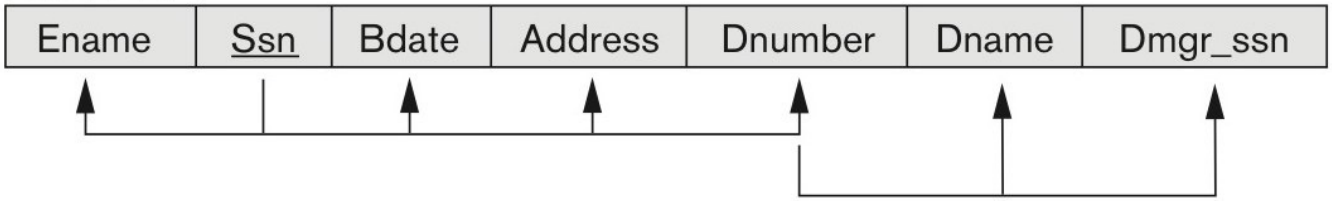
Ssn	Pnumber	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	Null

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

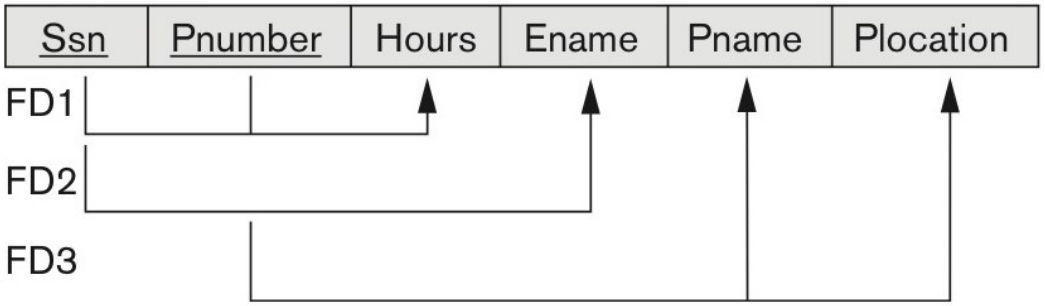
(a)

EMP_DEPT



(b)

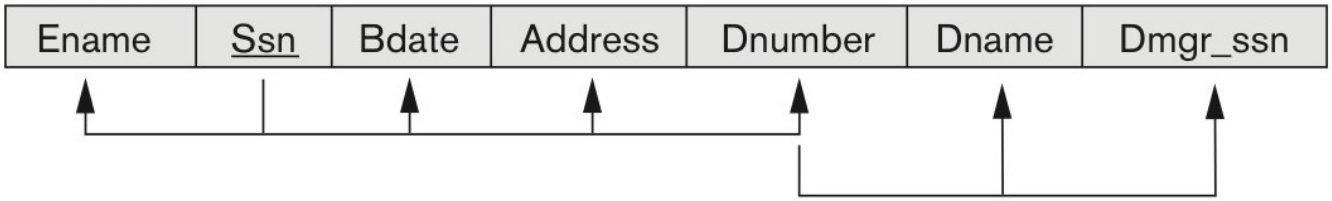
EMP_PROJ



Any concerns here?

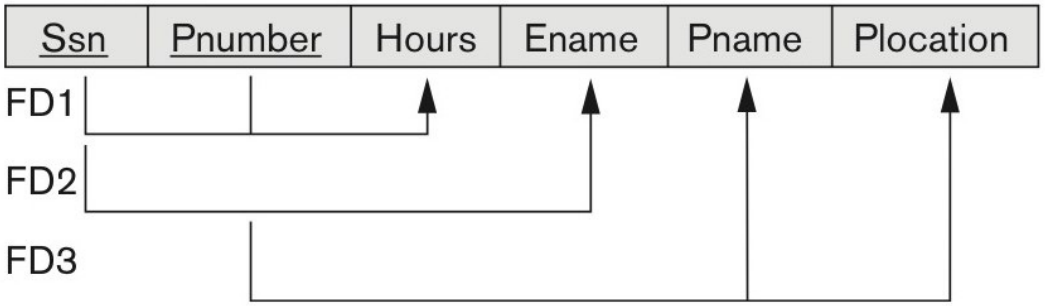
(a)

EMP_DEPT



(b)

EMP_PROJ



Any concerns here?

EMP_DEPT: mixing attributes of employees & departments

EMP_PROJ: 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.

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.

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.

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