CS4.301 Data & Applications

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

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

Where condition

```
mysql> select fname, lname, address from employee, department where dname='research' AND dno = dnumber;
+----------+---------+-------------------------+
| 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 (0.00 sec)

mysql> select fname, lname, address from employee, department where dno = dnumber AND 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 |
+----------+---------+-------------------------+
4 rows in set (0.01 sec)
```

```
SELECT Pnumber, Dnum, Lname, Address, Bdate
FROM EMPLOYEE, DEPARTMENT, PROJECT
WHERE DNUM=DNUMBER AND Mgr_ssn=Ssn AND
Plocation = 'Stafford';
```

```
mysql> SELECT Pnumber, Dnum, Lname, Address, Bdate
    -> FROM EMPLOYEE, DEPARTMENT, PROJECT
    -> WHERE DNUM=DNUMBER AND Mgr_ssn=Ssn AND
    -> Plocation = '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 (0.00 sec)
```

Unspecified WHERE Clause and Use of the Asterisk

Select Ssn
FROM EMPLOYEE;

SELECT SSN, DNAME
FROM EMPLOYEE, DEPARTMENT;

```
mysql> Select Ssn
    -> FROM EMPLOYEE;
+-----------+
| Ssn       |
+-----------+
| 123456789 |
| 333445555 |
| 453453453 |
| 666884444 |
| 888665555 |
| 987654321 |
| 987987987 |
| 999887777 |
+-----------+
8 rows in set (0.00 sec)
```

Select Ssn
FROM EMPLOYEE;

SELECT SSN, DNAME
FROM EMPLOYEE, DEPARTMENT;

```
mysql> SELECT SSN, DNAME
    -> FROM EMPLOYEE, DEPARTMENT;
+-----------+----------------+
| SSN       | DNAME          |
+-----------+----------------+
| 123456789 | Research       |
| 123456789 | Headquarters   |
| 123456789 | Administration |
| 333445555 | Research       |
| 333445555 | Headquarters   |
| 333445555 | Administration |
| 453453453 | Research       |
| 453453453 | Headquarters   |
| 453453453 | Administration |
| 666884444 | Research       |
| 666884444 | Headquarters   |
| 666884444 | Administration |
| 888665555 | Research       |
| 888665555 | Headquarters   |
| 888665555 | Administration |
| 987654321 | Research       |
| 987654321 | Headquarters   |
| 987654321 | Administration |
| 987987987 | Research       |
| 987987987 | Headquarters   |
| 987987987 | Administration |
| 999887777 | Research       |
| 999887777 | Headquarters   |
| 999887777 | Administration |
+-----------+----------------+
24 rows in set (0.00 sec)
```

Unspecified WHERE Clause and Use of the Asterisk (cont'd.)

Specify an asterisk (*)

- Retrieve all the attribute values of the selected tuples
- The * can be prefixed by the relation name; e.g., EMPLOYEE *

```
SELECT *
FROM EMPLOYEE
WHERE Dno = 5;
```

```
mysql> SELECT *
    -> FROM EMPLOYEE
    -> WHERE Dno = 5;
+----------+-------+---------+-----------+------------+-------------------------+------+--------+-----------+-----+
| 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 |
| Ramesh   | K     | Narayan | 666884444 | 1962-09-15 | 975 Fire Oak, Humble TX | M    |  38000 | 333445555 |   5 |
+----------+-------+---------+-----------+------------+-------------------------+------+--------+-----------+-----+
4 rows in set (0.00 sec)
```

```
SELECT *
FROM EMPLOYEE, DEPARTMENT
WHERE Dname='Research' AND Dno=Dnumber;
```

Attributes from both tables

```
mysql> SELECT *
    -> FROM EMPLOYEE, DEPARTMENT
    -> WHERE Dname='Research' AND Dno=Dnumber;
+----------+-------+----------+-----------+------------+-------------------------+-----+--------+-----------+-----+----------+---------+-----------+----------------+
| Fname    | Minit | Lname    | Ssn       | Bdate      | Address                 | Sex | Salary | Super_ssn | Dno | Dname    | Dnumber | Mgr_ssn   | Mgr_start_date |
+----------+-------+----------+-----------+------------+-------------------------+-----+--------+-----------+-----+----------+---------+-----------+----------------+
| John     | B     | Smith    | 123456789 | 1965-01-09 | 731 Fondren, Houston TX | M   |  30000 | 333445555 |   5 | Research |       5 | 333445555 | 1988-05-22     |
| Franklin | T     | Wong     | 333445555 | 1965-12-08 | 638 Voss, Houston TX    | M   |  40000 | 888665555 |   5 | Research |       5 | 333445555 | 1988-05-22     |
| Joyce    | A     | English  | 453453453 | 1972-07-31 | 5631 Rice, Houston TX   | F   |  25000 | 333445555 |   5 | Research |       5 | 333445555 | 1988-05-22     |
| Ramesh   | K     | Narayan  | 666884444 | 1962-09-15 | 975 Fire Oak, Humble TX | M   |  38000 | 333445555 |   5 | Research |       5 | 333445555 | 1988-05-22     |
+----------+-------+----------+-----------+------------+-------------------------+-----+--------+-----------+-----+----------+---------+-----------+----------------+
4 rows in set (0.00 sec)
```

SELECT *
FROM EMPLOYEE, DEPARTMENT;

Attributes from both tables

```
mysql> SELECT *
    -> FROM EMPLOYEE, DEPARTMENT;
```

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno	Dname	Dnumber	Mgr_ssn	Mgr_start_date
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston TX	M	30000	333445555	5	Research	5	333445555	1988-05-22
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston TX	M	30000	333445555	5	Administration	4	987654321	1995-01-01
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston TX	M	30000	333445555	5	Headquarters	1	888665555	1981-06-19
Franklin	T	Wong	333445555	1965-12-08	638 Voss, Houston TX	M	40000	888665555	5	Research	5	333445555	1988-05-22
Franklin	T	Wong	333445555	1965-12-08	638 Voss, Houston TX	M	40000	888665555	5	Administration	4	987654321	1995-01-01
Franklin	T	Wong	333445555	1965-12-08	638 Voss, Houston TX	M	40000	888665555	5	Headquarters	1	888665555	1981-06-19
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston TX	F	25000	333445555	5	Research	5	333445555	1988-05-22
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston TX	F	25000	333445555	5	Administration	4	987654321	1995-01-01
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston TX	F	25000	333445555	5	Headquarters	1	888665555	1981-06-19
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble TX	M	38000	333445555	5	Research	5	333445555	1988-05-22
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble TX	M	38000	333445555	5	Administration	4	987654321	1995-01-01
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble TX	M	38000	333445555	5	Headquarters	1	888665555	1981-06-19
James	E	Borg	888665555	1937-11-10	450 Stone, Houston TX	M	55000	NULL	1	Research	5	333445555	1988-05-22
James	E	Borg	888665555	1937-11-10	450 Stone, Houston TX	M	55000	NULL	1	Administration	4	987654321	1995-01-01
James	E	Borg	888665555	1937-11-10	450 Stone, Houston TX	M	55000	NULL	1	Headquarters	1	888665555	1981-06-19
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire TX	F	43000	888665555	4	Research	5	333445555	1988-05-22
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire TX	F	43000	888665555	4	Administration	4	987654321	1995-01-01
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire TX	F	43000	888665555	4	Headquarters	1	888665555	1981-06-19
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston TX	M	25000	987654321	4	Research	5	333445555	1988-05-22
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston TX	M	25000	987654321	4	Administration	4	987654321	1995-01-01
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston TX	M	25000	987654321	4	Headquarters	1	888665555	1981-06-19
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring TX	F	25000	987654321	4	Research	5	333445555	1988-05-22
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring TX	F	25000	987654321	4	Administration	4	987654321	1995-01-01
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring TX	F	25000	987654321	4	Headquarters	1	888665555	1981-06-19

```
24 rows in set (0.00 sec)
```

Tables as Sets in SQL

SQL does not automatically eliminate duplicate tuples in query results

For aggregate operations (See sec 7.1.7) duplicates must be accounted for

Use the keyword **`DISTINCT`** in the `SELECT` clause

Only distinct tuples should remain in the result

Query 11. Retrieve the salary of every employee (Q11) and all distinct salary values (Q11A).

```
Q11:   SELECT    ALL Salary
       FROM      EMPLOYEE;

Q11A:  SELECT    DISTINCT Salary
       FROM      EMPLOYEE;
```

SELECT ALL Salary
FROM EMPLOYEE;

```
mysql> SELECT ALL Salary
    -> FROM EMPLOYEE;
+--------+
| Salary |
+--------+
|  30000 |
|  40000 |
|  25000 |
|  38000 |
|  55000 |
|  43000 |
|  25000 |
|  25000 |
+--------+
8 rows in set (0.01 sec)
```

SELECT DISTINCT Salary
FROM EMPLOYEE;

```
mysql> SELECT DISTINCT Salary
    -> FROM EMPLOYEE;
+--------+
| Salary |
+--------+
|  30000 |
|  40000 |
|  25000 |
|  38000 |
|  55000 |
|  43000 |
+--------+
6 rows in set (0.01 sec)
```

Tables as Sets in SQL (cont'd.)

```
(SELECT DISTINCT Pnumber
FROM PROJECT, DEPARTMENT, EMPLOYEE
WHERE Dnum=Dnumber AND Mgr_ssn=Ssn
      AND Lname='Smith')
UNION
(SELECT DISTINCT Pnumber
FROM PROJECT, WORKS_ON, EMPLOYEE
WHERE Pnumber=Pno AND Essn=Ssn
      AND Lname='Smith');
```

```
mysql> (SELECT DISTINCT Pnumber
    -> FROM PROJECT, DEPARTMENT, EMPLOYEE
    -> WHERE Dnum=Dnumber AND Mgr_ssn=Ssn
    -> AND Lname='Smith')
    -> UNION
    -> (SELECT DISTINCT Pnumber
    -> FROM PROJECT, WORKS_ON, EMPLOYEE
    -> WHERE Pnumber=Pno AND Essn=Ssn
    -> AND Lname='Smith');
+---------+
| Pnumber |
+---------+
|       1 |
|       2 |
+---------+
2 rows in set (0.00 sec)
```

Substring Pattern Matching and Arithmetic Operators

LIKE comparison operator

Used for string **pattern matching**

% replaces an arbitrary number of zero or more characters

underscore (_) replaces a single character

Examples: **WHERE** Address **LIKE** '%Houston,TX%';

WHERE Ssn **LIKE** '_ _ 1_ _ 8901';

BETWEEN comparison operator [ka_ _ _ ka%]

E.g., in Q14 :

WHERE(Salary **BETWEEN** 30000 **AND** 40000)

AND Dno = 5;

Arithmetic Operations

Standard arithmetic operators:

Addition (+), subtraction (–), multiplication (*), and division (/) may be included as a part of **SELECT**

Query 13. Show the resulting salaries if every employee working on the ‘ProductX’ project is given a 10 percent raise.

```
SELECT  E.Fname, E.Lname, 1.1 * E.Salary AS Increased_sal
FROM  EMPLOYEE AS E, WORKS_ON AS W, PROJECT AS P
WHERE  E.Ssn=W.Essn AND W.Pno=P.Pnumber AND P.Pname=‘ProductX’;
```

Ordering of Query Results

Use **`ORDER BY`** clause

- Keyword **`DESC`** to see result in a descending order of values
- Keyword **`ASC`** to specify ascending order explicitly
- Typically placed at the end of the query

```
ORDER BY D.Dname DESC, E.Lname ASC, E.Fname ASC
```

Basic SQL Retrieval Query Block

```
SELECT      <attribute list>
FROM        <table list>
[ WHERE     <condition> ]
[ ORDER BY <attribute list> ];
```

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 );
```

The variation below inserts multiple tuples where a new table is loaded values from the result of a query.

```
U3B:  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;
```

BULK LOADING OF TABLES

Another variation of **INSERT** is used for bulk-loading of several tuples into tables

A new table TNEW can be created with the same attributes as T and using LIKE and DATA in the syntax, it can be loaded with entire data.

EXAMPLE:

```
CREATE TABLE D5EMPS  LIKE  EMPLOYEE
        (SELECT   E.*
         FROM          EMPLOYEE AS E
        WHERE     E.Dno=5)
 WITH DATA;
```

WITH DATA specifies that the table will be created & loaded with the data specified in the query

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;
```

UPDATE (contd.)

Example: Change the location and controlling department number of project number 10 to 'Bellaire' and 5, respectively

```
U5:  UPDATE   PROJECT
     SET      PLOCATION = 'Bellaire', DNUM = 5
     WHERE    PNUMBER=10
```

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

UPDATE (contd.)

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

```
U6:   UPDATE     EMPLOYEE
SET   SALARY = SALARY *1.1
WHERE      DNO  IN (SELECT        DNUMBER
   FROM    DEPARTMENT
   WHERE  DNAME='Research')
```

In this request, the modified SALARY value depends on the original SALARY value in each tuple

- The reference to the SALARY attribute on the right of = refers to the old SALARY value before modification
- The reference to the SALARY attribute on the left of = refers to the new SALARY value after modification

Comparisons Involving NULL

SQL allows queries that check whether an attribute value is `NULL`

`IS` or `IS NOT NULL`

Query 18. Retrieve the names of all employees who do not have supervisors.

```
Q18:  SELECT  Fname, Lname
      FROM    EMPLOYEE
      WHERE   Super_ssn IS NULL;
```

```
mysql> Select fname, lname from employee where super_ssn IS null;
+-------+-------+
| fname | lname |
+-------+-------+
| James | Borg  |
+-------+-------+
1 row in set (0.00 sec)
```

IS & IS NOT

```
Select fname, lname from employee where super_ssn
IS NOT null;
```

```
mysql> Select fname, lname from employee where super_ssn IS NOT null;
+----------+---------+
| fname    | lname   |
+----------+---------+
| John     | Smith   |
| Franklin | Wong    |
| Joyce    | English |
| Ramesh   | Narayan |
| Jennifer | Wallace |
| Ahmad    | Jabbar  |
| Alicia   | Zelaya  |
+----------+---------+
7 rows in set (0.00 sec)

mysql>
```

This lecture

Query 2. For every project located in 'Stafford', list the project number, the controlling department number, and the department manager's last name, address, and birth date.

Query 2. For every project located in 'Stafford', list the project number, the controlling department number, and the department manager's last name, address, and birth date.

```
Q2:   SELECT   Pnumber, Dnum, Lname, Address, Bdate
      FROM     PROJECT, DEPARTMENT, EMPLOYEE
      WHERE    Dnum=Dnumber AND Mgr_ssn=Ssn AND
               Plocation='Stafford';
```

Query 2. For every project located in 'Stafford', list the project number, the controlling department number, and the department manager's last name, address, and birth date.

```
Q2:   SELECT   Pnumber, Dnum, Lname, Address, Bdate
      FROM     PROJECT, DEPARTMENT, EMPLOYEE
      WHERE    Dnum=Dnumber AND Mgr_ssn=Ssn AND
               Plocation='Stafford';
```

(c)

Pnumber	Dnum	Lname	Address	Bdate
10	4	Wallace	291Berry, Bellaire, TX	1941-06-20
30	4	Wallace	291Berry, Bellaire, TX	1941-06-20

Ambiguous Attribute Names

Same name can be used for two (or more) attributes in different relations

- As long as the attributes are in different relations
- Must **qualify** the attribute name with the relation name to prevent ambiguity

```
Q1A:    SELECT    Fname, EMPLOYEE.Name, Address
        FROM      EMPLOYEE, DEPARTMENT
        WHERE     DEPARTMENT.Name='Research' AND
                  DEPARTMENT.Dnumber=EMPLOYEE.Dnumber;
```

Aliasing, and Renaming

Aliases or **tuple variables**

Declare alternative relation names E and S to refer to the EMPLOYEE relation twice in a query:

Query 8. For each employee, retrieve the employee's first and last name and the first and last name of his or her immediate supervisor.

```
SELECT  E.Fname, E.Lname, S.Fname, S.Lname
FROM    EMPLOYEE AS E, EMPLOYEE AS S
WHERE E.Super_ssn=S.Ssn;
```

Recommended practice to abbreviate names and to prefix same or similar attribute from multiple tables.

E.Fname	E.Lname	S.Fname	S.Lname
John	Smith	Franklin	Wong
Franklin	Wong	James	Borg
Alicia	Zelaya	Jennifer	Wallace
Jennifer	Wallace	James	Borg
Ramesh	Narayan	Franklin	Wong
Joyce	English	Franklin	Wong
Ahmad	Jabbar	Jennifer	Wallace

Aliasing,Renaming and Tuple Variables (contd.)

The attribute names can also be renamed

```
EMPLOYEE AS E(Fn, Mi, Ln, Ssn, Bd, Addr, Sex,
Sal, Sssn, Dno)
```

Note that the relation EMPLOYEE now has a variable name E which corresponds to a tuple variable

The “AS” may be dropped in most SQL implementations

Nested Queries, Tuples, and Set/Multiset Comparisons

Nested queries

- Complete select-from-where blocks within WHERE clause of another query
- **Outer query and nested subqueries**

Comparison operator `IN`

- Compares value *v* with a set (or multiset) of values *V*
- Evaluates to `TRUE` if *v* is one of the elements in *V*

Nested Queries (cont'd.)

```
SELECT   DISTINCT Pnumber
FROM     PROJECT
WHERE    Pnumber IN
         (SELECT   Pnumber
         FROM      PROJECT, DEPARTMENT, EMPLOYEE
         WHERE     Dnum = Dnumber AND
         Mgr_ssn = Ssn and Lname = 'Smith')
         OR
         Pnumber IN
         (SELECT   Pno
         FROM      WORKS_ON, EMPLOYEE
         WHERE     Essn = Ssn AND Lname = 'Smith');
```

```
+---------+
| Pnumber |
+---------+
|       1 |
|       2 |
+---------+
2 rows in set (0.01 sec)
```

Nested Queries (cont'd.)

```
SELECT      DISTINCT    Pnumber
FROM        PROJECT
WHERE       Pnumber IN
            (SELECT     Pnumber
            FROM        PROJECT, DEPARTMENT, EMPLOYEE
            WHERE       Dnum = Dnumber AND
            Mgr_ssn = Ssn and Lname = 'Smith')
            OR
            Pnumber IN
            (SELECT     Pno
            FROM        WORKS_ON, EMPLOYEE
            WHERE       Essn = Ssn AND Lname = 'Smith');
```

```
mysql> (SELECT DISTINCT Pnumber
    -> FROM PROJECT, DEPARTMENT, EMPLOYEE
    -> WHERE Dnum=Dnumber AND Mgr_ssn=Ssn
    -> AND Lname='Smith')
    -> UNION
    -> (SELECT DISTINCT Pnumber
    -> FROM PROJECT, WORKS_ON, EMPLOYEE
    -> WHERE Pnumber=Pno AND Essn=Ssn
    -> AND Lname='Smith');
+---------+
| Pnumber |
+---------+
|       1 |
|       2 |
+---------+
2 rows in set (0.00 sec)
```

Nested Queries (cont'd.)

Use tuples of values in comparisons

Place them within parentheses

```
Select distinct essn
From works_on
Where (pno, hours) IN
(Select pno, hours from
works_on where essn =
'123456789');
```

```
mysql> Select pno, hours from works_on where essn = '123456789';
+-----+-------+
| pno | hours |
+-----+-------+
|   1 |  32.5 |
|   2 |   7.5 |
+-----+-------+
2 rows in set (0.04 sec)
```

```
mysql> Select distinct essn
    -> From works_on
    -> Where (pno, hours) IN (Select pno, hours from works_on where essn = '123456789');
+-----------+
| essn      |
+-----------+
| 123456789 |
+-----------+
1 row in set (0.01 sec)
```

Nested Queries (cont'd.)

Use other comparison operators to compare a single value *v*

- `= ANY` (or `= SOME`) operator
  - Returns `TRUE` if the value *v* is equal to some value in the set *V* and is hence equivalent to `IN`
- Other operators that can be combined with `ANY` (or `SOME`): >, >=, <, <=, and <>
- `ALL`: value must exceed all values from nested query

```
mysql> select salary from employee where dno = 5;
+--------+
| salary |
+--------+
|  30000 |
|  40000 |
|  25000 |
|  38000 |
+--------+
4 rows in set (0.00 sec)
```

Select lname, fname, salary from employee where salary > all (select salary from employee where dno = 5);

```
mysql> Select lname, fname, salary from employee w
here salary > all (select salary from employee whe
re dno = 5);
+---------+----------+--------+
| lname   | fname    | salary |
+---------+----------+--------+
| Borg    | James    |  55000 |
| Wallace | Jennifer |  43000 |
+---------+----------+--------+
2 rows in set (0.00 sec)
```

```
mysql> Select lname, fname, salary from employee wh
ere salary > any (select salary from employee where
 dno = 5);
+---------+----------+--------+
| lname   | fname    | salary |
+---------+----------+--------+
| Smith   | John     |  30000 |
| Wong    | Franklin |  40000 |
| Narayan | Ramesh   |  38000 |
| Borg    | James    |  55000 |
| Wallace | Jennifer |  43000 |
+---------+----------+--------+
5 rows in set (0.00 sec)
```

Nested Queries (cont'd.)

Avoid potential errors and ambiguities

Create tuple variables (aliases) for all tables referenced in SQL query

Query 16. Retrieve the name of each employee who has a dependent with the same first name and is the same sex as the employee.

Nested Queries (cont'd.)

Avoid potential errors and ambiguities

Create tuple variables (aliases) for all tables referenced in SQL query

Query 16. Retrieve the name of each employee who has a dependent with the same first name and is the same sex as the employee.

Select e.fname, e.lname from employee as e where e.ssn in (select essn from dependent as d where e.fname=d.dependent_name and e.sex=d.sex);

```
mysql> Select e.fname, e.lname from employee as e w
here e.ssn in (select essn from dependent as d wher
e e.fname=d.dependent_name and e.sex=d.sex);
Empty set (0.00 sec)
```

Correlated Nested Queries

Queries that are nested using the = or IN comparison operator can be collapsed into one single block, last query can be changed like ???? Ideas?

```
Select e.fname, e.lname from
employee as e where e.ssn in
(select essn from dependent as
d where
e.fname=d.dependent_name
and e.sex=d.sex);
```

Correlated nested query

Evaluated once for each tuple in the outer query

Correlated Nested Queries

Queries that are nested using the = or IN comparison operator can be collapsed into one single block, last query can be changed like

```
Select e.fname, e.lname from
employee as e where e.ssn in
(select essn from dependent as
d where
e.fname=d.dependent_name
and e.sex=d.sex);
```

```
SELECT E.Fname, E.Lname
FROM EMPLOYEE AS E,
DEPENDENT AS D WHERE
E.Ssn=D.Essn AND E.Sex=D.Sex
AND
E.Fname=D.Dependent_name;
```

Correlated nested query

Evaluated once for each tuple in the outer query

Explicit Sets and Renaming of Attributes in SQL

Can use explicit set of values in WHERE clause

SELECT DISTINCT Essn FROM WORKS_ON WHERE Pno IN (1, 2, 3);

```
mysql> SELECT DISTINCT Essn FROM WORKS_ON WHERE Pno
 IN (1, 2, 3);
+-----------+
| Essn      |
+-----------+
| 123456789 |
| 453453453 |
| 333445555 |
| 666884444 |
+-----------+
4 rows in set (0.00 sec)
```

Explicit Sets and Renaming of Attributes in SQL

Use qualifier AS followed by desired new name

Rename any attribute that appears in the result of a query

Select e.lname as employee_name, s.lname as supervisor_name from employee as e, employee as s where e.super_ssn = s.ssn;

```
mysql> Select e.lname as employee_name, s.lname as
supervisor_name from employee as e, employee as s w
here e.super_ssn = s.ssn;
+---------------+-----------------+
| employee_name | supervisor_name |
+---------------+-----------------+
| Smith         | Wong            |
| Wong          | Borg            |
| English       | Wong            |
| Narayan       | Wong            |
| Wallace       | Borg            |
| Jabbar        | Wallace         |
| Zelaya        | Wallace         |
+---------------+-----------------+
7 rows in set (0.00 sec)
```

Aggregate Functions in SQL

Used to summarize information from multiple tuples into a single-tuple summary

Built-in aggregate functions

COUNT, **SUM**, **MAX**, **MIN**, and **AVG**

Grouping

Create subgroups of tuples before summarizing

To select entire groups, `HAVING` clause is used

Aggregate functions can be used in the `SELECT` clause or in a `HAVING` clause

Renaming Results of Aggregation

```
SELECT SUM(Salary),
MAX(Salary),
MIN(Salary), AVG(Salary)
FROM EMPLOYEE;
```

```
mysql> SELECT SUM(Salary), MAX(Salary), MIN(Salary), AVG(Sal
ary) FROM EMPLOYEE;
+-------------+-------------+-------------+-------------+
| SUM(Salary) | MAX(Salary) | MIN(Salary) | AVG(Salary) |
+-------------+-------------+-------------+-------------+
|      281000 |       55000 |       25000 |  35125.0000 |
+-------------+-------------+-------------+-------------+
1 row in set (0.00 sec)
```

Renaming Results of Aggregation

```
SELECT SUM(Salary) AS
Total_Sal, MAX(Salary)
AS Highest_Sal,
MIN(Salary) AS
Lowest_Sal, AVG(Salary)
AS Average_Sal FROM
EMPLOYEE;
```

```
mysql> SELECT SUM(Salary) AS Total_Sal, MAX(Salary) AS Highe
st_Sal, MIN(Salary) AS Lowest_Sal, AVG(Salary) AS Average_Sa
l FROM EMPLOYEE;
+-----------+-------------+------------+-------------+
| Total_Sal | Highest_Sal | Lowest_Sal | Average_Sal |
+-----------+-------------+------------+-------------+
|    281000 |       55000 |      25000 |  35125.0000 |
+-----------+-------------+------------+-------------+
1 row in set (0.00 sec)
```

Aggregate Functions in SQL (cont'd.)

Query 20. Find the sum of the salaries of all employees of the 'Research' department, as well as the maximum salary, the minimum salary, and the average salary in this department.

Aggregate Functions in SQL (cont'd.)

Query 20. Find the sum of the salaries of all employees of the 'Research' department, as well as the maximum salary, the minimum salary, and the average salary in this department.

```
SELECT SUM(Salary),
MAX(Salary),
MIN(Salary), AVG(Salary)
FROM (EMPLOYEE join
department on
dno=dnumber) where
dname='research';
```

```
mysql> SELECT SUM(Salary), MAX(Salary), MIN(Salary), AVG(Sal
ary) FROM (EMPLOYEE join department on dno=dnumber) where dn
ame='research';
+-------------+-------------+-------------+-------------+
| SUM(Salary) | MAX(Salary) | MIN(Salary) | AVG(Salary) |
+-------------+-------------+-------------+-------------+
|      133000 |       40000 |       25000 |  33250.0000 |
+-------------+-------------+-------------+-------------+
1 row in set (0.00 sec)
```

Aggregate Functions in SQL (cont'd.)

Queries 21 and 22. Retrieve the total number of employees in the company (Q21) and the number of employees in the 'Research' department (Q22).

Select count(*) from employee;

Select count(*) from employee, department where dno=dnumber and dname='research';

```
mysql> Select count(*) from employee;
+----------+
| count(*) |
+----------+
|        8 |
+----------+
1 row in set (0.02 sec)

mysql> Select count(*) from employee, department where dno=d
number and dname='research';
+----------+
| count(*) |
+----------+
|        4 |
+----------+
1 row in set (0.00 sec)
```

Aggregate Functions on Booleans

SOME and ALL may be applied as functions on Boolean Values.

SOME returns true if at least one element in the collection is TRUE (similar to OR)

ALL returns true if all of the elements in the collection are TRUE (similar to AND)

Grouping: The GROUP BY Clause

Partition relation into subsets of tuples

- Based on **grouping attribute(s)**
- Apply function to each such group independently

`GROUP BY` clause

- Specifies grouping attributes

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	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

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

WORKS_ON

Essn	Pno	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

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

Group BY example

SELECT Dno, COUNT(*), AVG(Salary) FROM EMPLOYEE GROUP BY Dno;

```
mysql> SELECT Dno, COUNT(*), AVG(Salary) FROM EMPLOYEE GROUP
 BY Dno;
+-----+----------+-------------+
| Dno | COUNT(*) | AVG(Salary) |
+-----+----------+-------------+
|   5 |        4 |  33250.0000 |
|   1 |        1 |  55000.0000 |
|   4 |        3 |  31000.0000 |
+-----+----------+-------------+
3 rows in set (0.01 sec)
```

Group BY example

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

```
mysql> SELECT Pnumber, Pname, COUNT(*) FROM PROJECT, WORKS_O
N WHERE Pnumber=Pno GROUP BY Pname;
+---------+-----------------+----------+
| Pnumber | Pname           | COUNT(*) |
+---------+-----------------+----------+
|      10 | Computerization |        3 |
|      30 | Newbenefits     |        3 |
|       1 | ProductX        |        2 |
|       2 | ProductY        |        3 |
|       3 | ProductZ        |        2 |
|      20 | Reorganization  |        3 |
+---------+-----------------+----------+
6 rows in set (0.01 sec)
```

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

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

In-Class Activity

- Schema & Data: https://github.com/tolgahanakgun/Elmasri-Database

a. Retrieve the names of all employees who work in the department that has the employee with the highest salary among all employees.

b. Retrieve the names of all employees whose supervisor's supervisor has ''888665555' for ssn

c. Retrieve the names of employees who make at least 10,000 USD more than the employee who is paid the least in the company

d. Create a view that has the department name, manager name, and manager salary for every department

e. Create a view that has the employee name, supervisor name, and employee salary for each employee who works in the 'research' department

Summary

SQL

- A Comprehensive language for relational database management
- Data definition, queries, updates, constraint specification, and view definition

Covered :

- Data definition commands for creating tables
- Commands for constraint specification
- Simple retrieval queries
- Database update commands

Bibliography / Acknowledgements

Instructor materials from Elmasri & Navathe 7e

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


pk.profgiri


Ponnurangam.kumaraguru


/in/ponguru


ponguru

Thank you for attending the class!!!


pk.guru@iiit.ac.in