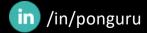
# CS4.301 Data & Applications

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









```
SFLECT
        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)
```

### 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');

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

Use other comparison operators to compare a single value v

= ANY (or = SOME) operator

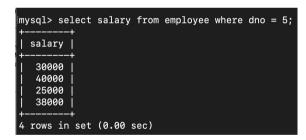
Returns TRUE if the value v is equal to some value in the set V and is hence equivalent to IN

Other operators that can be combined with ANY (or SOME): >, >=, <, <=, and <>

ALL: value must exceed all values from nested query

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

mysql> Sele			•		
here salary		тест затал	ry iroili	embrokee	wne
re dno = 5	);				
	, 		L		
lname	fname	salary			
<del> </del>		+	+		
i Dame	7	I	i		
, ,	James				
Wallace	Jennifer	43000	l		
		•	•		
i .		•			
2 rows in s	set (0.00 s	ec)			



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);

### 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;
                    supervisor_name
  employee_name |
  Smith
                    Wong
  Wong
                    Borg
  English
                    Wong
  Narayan
                    Wong
  Wallace
                    Borg
  Jabbar
                    Wallace
  Zelaya
                    Wallace
7 \text{ rows in set } (0.00 \text{ 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(Salary) 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 Highel
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.

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

```
      Imysql> SELECT SUM(Salary), MAX(Salary), MIN(Salary), AVG(Salary) 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(*) |
+-----+
1 row in set (0.02 sec)

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

## Group BY example

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

# Group BY example

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

<pre>[mysql&gt; SELECT Pnumber, Pname, COUNT(*) FROM PROJECT, WORKS_O N WHERE Pnumber=Pno GROUP BY Pname; +</pre>					
Pnumber		COUNT(*)	Ĭ		
10	Computerization	3	Ī		
30	Newbenefits	3	İ		
1	ProductX	2	İ		
2	ProductY	3	İ		
3	ProductZ	2	İ		
20	Reorganization	3	Ì		
+	<del></del>	+	+		
6 rows in set (0.01 sec)					

# This Lecture

### Administrativia

Guest lecture on Nov 7<sup>th</sup> evening

# 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_0
 WHERE Pnumber=Pno GROUP BY Pnumber HAVING COUNT(*) > 2;
                              COUNT(*)
  Pnumber
            Pname
            ProductY
                                      3
            Computerization
       10
                                      3
            Reorganization
       20
                                      3
            Newbenefits
       30
                                      3
4 rows in set (0.00 sec)
```

### **EXPANDED Block Structure of SQL Queries**

```
SELECT <attribute and function list>
FROM 
[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, Iname, hours from employee, project, works\_on where ssn=essn and pno=pnumber;

```
mysql> Create view works_on1 as select fname, lname, hours fl
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;
```

select fname, Iname from works\_on1 where hours=10;

		Raille 311
		16 rows in
	ct fname, lname from works_o	on1 where hours=10;
   fname 	 lname   +	
Franklin   Franklin   Franklin   Franklin   Alicia	Wong   Wong	
5 rows in se	et (0.01 sec)	

mysql> select \* from works\_on1; fname lname hours Franklin | Wong 10.0 16.0 James Borg Jennifer Wallace 15.0 Franklin i Wong 10.0 Jabbar 35.0 Ahmad Alicia Zelaya 10.0 Jennifer | Wallace 20.0 Jabbar Ahmad 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 Narayan 40.0 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 timeconsuming 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:

**lazy 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



f Ponnurangam.kumaraguru

in /in/ponguru

ponguru

Thank you for attending the class!!!

pk.guru@iiit.ac.in