

Module 3

Databases

SQL-99: Advanced Queries

Relational Database Schema--Figure 5.5

EMPLOYEE

FNAME	MINIT	LNAME	<u>SSN</u>	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
-------	-------	-------	------------	-------	---------	-----	--------	----------	-----

DEPARTMENT

DNAME	<u>DNUMBER</u>	MGRSSN	MGRSTARTDATE
-------	----------------	--------	--------------

DEPT_LOCATIONS

<u>DNUMBER</u>	<u>DLOCATION</u>
----------------	------------------

PROJECT

PNAME	<u>PNUMBER</u>	PLOCATION	DNUM
-------	----------------	-----------	------

WORKS_ON

<u>ESSN</u>	<u>PNO</u>	HOURS
-------------	------------	-------

DEPENDENT

<u>ESSN</u>	<u>DEPENDENT_NAME</u>	SEX	BDATE	RELATIONSHIP
-------------	-----------------------	-----	-------	--------------



Populated Database--Fig.5.6

EMPLOYEE	FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	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	1968-07-19	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	MGRSSN	MGRSTARTDATE
	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

NESTING OF QUERIES


- A complete SELECT query, called a *nested query* , can be specified within the WHERE-clause of another query, called the *outer query*
- Many of the previous queries can be specified in an alternative form using nesting
- Query 1: Retrieve the name and address of all employees who work for the 'Research' department.

Q1: **SELECT FNAME, LNAME, ADDRESS**
 FROM EMPLOYEE
 WHERE DNO IN (SELECT DNUMBER
 FROM DEPARTMENT
 WHERE DNAME='Research')

NESTING OF QUERIES (cont)

- Employees whose salary is greater than the salary of the employees in department 5.

Q2: SELECT LNAME, FNAME FROM EMPLOYEE
WHERE Salary > (SELECT Salary FROM EMPLOYEE WHERE Dno=5);



Nested query returns
more than 1 rows

NESTING OF QUERIES (cont)

- Employees whose salary is greater than the salary of all the employees in department 5.

Q2a: **SELECT LNAME, FNAME FROM EMPLOYEE**
 WHERE Salary > ALL (SELECT Salary FROM EMPLOYEE WHERE Dno=5);

	LNAME	FNAME
1	Wallace	Jennifer
2	Borg	James

- Employees whose salary is greater than the salary of any employee in department 5.

Q2b: **SELECT LNAME, FNAME FROM EMPLOYEE**
 WHERE Salary > ANY (SELECT Salary FROM EMPLOYEE WHERE Dno=5);

	LNAME	FNAME
1	Borg	James
2	Wallace	Jennifer
3	Wong	Franklin
4	Narayan	Ramesh
5	Smith	John

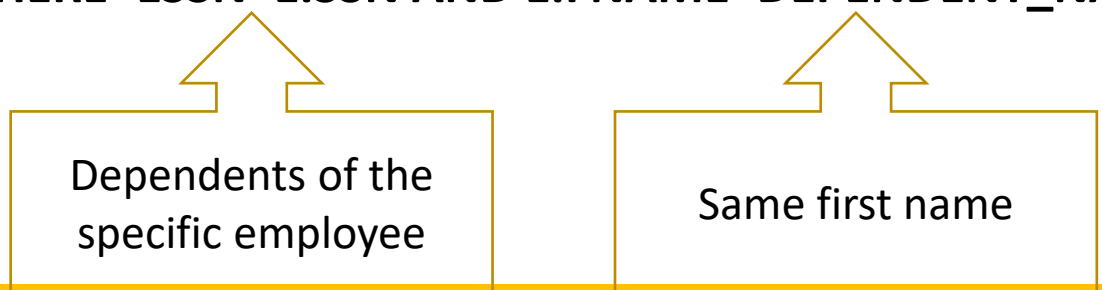
NESTING OF QUERIES (cont.)

- The nested query selects the number of the 'Research' department
- The outer query select an EMPLOYEE tuple if its DNO value is in the result of either nested query
- The comparison operator **IN** compares a value v with a set (or multi-set) of values V , and evaluates to **TRUE** if v is one of the elements in V
- In general, we can have several levels of nested queries
- A reference to an *unqualified attribute* refers to the relation declared in the *innermost nested query*
- In this example, the nested query is *not correlated* with the outer query

CORRELATED NESTED QUERIES

- If a condition in the WHERE-clause of a *nested query* references an attribute of a relation declared in the *outer query*, the two queries are said to be *correlated*
- The result of a correlated nested query is *different for each tuple (or combination of tuples) of the relation(s) the outer query*
- Query 2: Retrieve the name of each employee who has a dependent with the same first name as the employee.

```
Q2: SELECT E.FNAME, E.LNAME  
      FROM EMPLOYEE AS E  
      WHERE E.SSN IN (SELECT ESSN  
                      FROM DEPENDENT  
                      WHERE ESSN=E.SSN AND E.FNAME=DEPENDENT_NAME)
```



Dependents of the
specific employee

Same first name

CORRELATED NESTED QUERIES (cont.)

- In Q2, the nested query has a different result *for each tuple* in the outer query
- A query written with nested SELECT... FROM... WHERE... blocks and using the = or IN comparison operators can ***always*** be expressed as a single block query. For example, Q2 may be written as in Q2A

Q2A:

```
SELECT  E.FNAME, E.LNAME
FROM    EMPLOYEE E, DEPENDENT D
WHERE   E.SSN=D.ESSN AND E.FNAME=D.DEPENDENT_NAME
```

THE EXISTS FUNCTION

- EXISTS is used to check whether the result of a correlated nested query is empty (contains no tuples) or not
- We can formulate Query 2 in an alternative form that uses EXISTS as Q12B below
- Query 2: Retrieve the name of each employee who has a dependent with the same first name as the employee.

Q2B:

```
SELECT FNAME, LNAME
FROM EMPLOYEE
WHERE EXISTS (SELECT *
               FROM DEPENDENT
               WHERE SSN=ESSN AND FNAME=DEPENDENT_NAME)
```

THE EXISTS FUNCTION (cont.)

- Query 3: Retrieve the names of employees who have no dependents.

**Q3: SELECT FNAME, LNAME
FROM EMPLOYEE
WHERE NOT EXISTS (SELECT*
FROM DEPENDENT
WHERE SSN=ESSN)**

- In Q3, the correlated nested query retrieves all DEPENDENT tuples related to an EMPLOYEE tuple. If *none exist*, the EMPLOYEE tuple is selected
- EXISTS is necessary for the expressive power of SQL

EXPLICIT SETS

- It is also possible to use an **explicit (enumerated) set of values** in the WHERE-clause rather than a nested query
- Query 4: Retrieve the social security numbers of all employees who work on project number 1, 2, or 3.

Q4: **SELECT DISTINCT ESSN**
 FROM WORKS_ON
 WHERE PNO IN (1, 2, 3)

NEGATIVE QUERIES

- Makes sure that the expression proceeded does not have any of the values present in the arguments.
- `elem NOT IN (v1, v2...)` is true if `elem` does not belong to `(v1, v2...)`
- `NOT EXISTS (tuples_set)` is true if `tuples_set` is empty
- Q5: Find the employees which are not working on Project number 2

**Q5: SELECT FNAME, LNAME
 FROM EMPLOYEE, WORKS_ON
 WHERE SSN = ESSN AND PNO !=2**

1	John	Smith
2	Franklin	Wong
3	Franklin	Wong
4	Franklin	Wong
5	Joyce	English
6	Ramesh	Narayan
7	James	Borg
8	Jennifer	Wallace
9	Jennifer	Wallace
10	Ahmad	Jabbar
11	Ahmad	Jabbar
12	Alicia	Zelava
13	Alicia	Zelava

John Smith works on
projects 1 and 2

NEGATIVE QUERIES

Q5a: **SELECT FNAME, LNAME**
 FROM EMPLOYEE
 WHERE SSN NOT IN (SELECT ESSN FROM WORKS_ON
 WHERE PNO = 2); Employees working on Project 2

	FNAME	LNAME
1	Alicia	Zelava
2	Jennifer	Wallace
3	Ramesh	Naravan
4	Ahmad	Jabbar
5	James	Borg

AGGREGATE FUNCTIONS

- Include **COUNT**, **SUM**, **MAX**, **MIN**, and **AVG**
- Query 5: Find the maximum salary, the minimum salary, and the average salary among all employees.

**Q5: SELECT MAX(SALARY), MIN(SALARY), AVG(SALARY)
FROM EMPLOYEE**

- Some SQL implementations *may not allow more than one function* in the SELECT-clause

AGGREGATE FUNCTIONS (cont.)

- Query 6: Find the maximum salary, the minimum salary, and the average salary among employees who work for the 'Research' department.

**Q6: SELECT MAX(SALARY), MIN(SALARY), AVG(SALARY)
FROM EMPLOYEE, DEPARTMENT
WHERE DNO=DNUMBER AND DNAME='Research'**

AGGREGATE FUNCTIONS (cont.)

- Queries 7 and 8: Retrieve the total number of employees in the company (Q7), and the number of employees in the 'Research' department (Q8).

**Q7: SELECT COUNT (*)
FROM EMPLOYEE**

**Q8: SELECT COUNT (*)
FROM EMPLOYEE, DEPARTMENT
WHERE DNO=DNUMBER AND DNAME='Research'**

GROUPING

- In many cases, we want to apply the aggregate functions *to subgroups of tuples in a relation*
- Each subgroup of tuples consists of the set of tuples that have *the same value* for the *grouping attribute(s)*
- The function is applied to each subgroup independently
- SQL has a **GROUP BY**-clause for specifying the grouping attributes, which *must also appear in the SELECT-clause*

GROUPING (cont.)

- Query 9: For each department, retrieve the department number, the number of employees in the department, and their average salary.

**Q9: SELECT DNO, COUNT (*), AVG (SALARY)
 FROM EMPLOYEE
 GROUP BY DNO**

	DNO	COUNT(*)	AVG(SALARY)
1	1	1	55000
2	5	4	33250
3	4	3	31000

- In Q9, the EMPLOYEE tuples are divided into groups--each group having the same value for the grouping attribute DNO
- The COUNT and AVG functions are applied to each such group of tuples separately
- The SELECT-clause includes only the grouping attribute and the functions to be applied on each group of tuples
- A join condition can be used in conjunction with grouping

GROUPING (cont.)

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
James	E	Borg	888665555	10/11/27	450 Stone, Houston, TX	M	55000	(null)	1
Jennifer	S	Wallace	987654321	20/06/31	291 Berry, Bellaire, TX	F	43000	888665555	4
Ahmad	V	Jabbar	987987987	29/03/59	980 Dallas, Houston, TX	M	25000	987654321	4
Alicia	J	Zelaya	999887777	19/07/58	3321 Castle, Spring, TX	F	25000	987654321	4
Ramesh	K	Narayan	666884444	15/09/52	975 Fire Oak, Humble, TX	M	38000	333445555	5
John	B	Smith	123456789	09/01/55	731 Fondren, Houston, TX	M	30000	333445555	5
Joyce	A	English	453453453	31/07/62	5631 Rice, Houston, TX	F	25000	333445555	5
Franklin	T	Wong	333445555	08/12/45	638 Voss, Houston, TX	M	40000	888665555	5

Count	Avg
1	55000
3	31000
4	33250



GROUPING (cont.)

- Query 10: For each project, retrieve the project number, project name, and the number of employees who work on that project.

**Q10: SELECT PNUMBER, PNAME, COUNT (*)
FROM PROJECT, WORKS_ON
WHERE PNUMBER=PNO
GROUP BY PNUMBER, PNAME**

	⚡ PNUMBER	⚡ PNAME	⚡ COUNT(*)
1	1	ProductX	2
2	2	ProductY	3
3	3	ProductZ	2
4	10	Computerization	3
5	20	Reorganization	3
6	30	Newbenefits	3

- In this case, the grouping and functions are applied *after* the joining of the two relations

THE HAVING-CLAUSE

- Sometimes we want to retrieve the values of these functions for only those *groups that satisfy certain conditions*
- The HAVING-clause is used for specifying a selection condition on groups (rather than on individual tuples)

THE HAVING-CLAUSE (cont.)

- Query 11: For each project *on which more than two employees work* , retrieve the project number, project name, and the number of employees who work on that project.

	PNUMBER	PNAME	COUNT(*)
1	1	ProductX	2
2	2	ProductY	3
3	3	ProductZ	2
4	10	Computerization	3
5	20	Reorganization	3
6	30	Newbenefits	3

Q11: **SELECT PNUMBER, PNAME, COUNT(*)**
FROM PROJECT, WORKS_ON
WHERE PNUMBER=PNO
GROUP BY PNUMBER, PNAME
HAVING COUNT (*) > 2

	PNUMBER	PNAME	COUNT(*)
1	2	ProductY	3
2	30	Newbenefits	3
3	20	Reorganization	3
4	10	Computerization	3

Summary of SQL Queries

- A query in SQL can consist of up to six clauses, but only the first two, SELECT and FROM, are mandatory. The clauses are specified in the following order:

SELECT <attribute list>
FROM <table list>
[WHERE <condition>
[GROUP BY <grouping attribute(s)>
[HAVING <group condition>
[ORDER BY <attribute list>

Summary of SQL Queries (cont.)

- The SELECT-clause lists the attributes or functions to be retrieved
- The FROM-clause specifies all relations (or aliases) needed in the query but not those needed in nested queries
- The WHERE-clause specifies the conditions for selection and join of tuples from the relations specified in the FROM-clause
- GROUP BY specifies grouping attributes
- HAVING specifies a condition for selection of groups
- ORDER BY specifies an order for displaying the result of a query
- A query is evaluated by first applying the WHERE-clause, then GROUP BY and HAVING, and finally the SELECT-clause