Database Systems

Lecture #09

Sang-Wook Kim
Hanyang University

Objectives

- To learn more complex SQL retrieval queries
  - Nested queries
  - Aggregate functions
  - Grouping

Outline

- Nested Queries
  - Comparison Operators
  - Avoiding Ambiguities
  - EXISTS and UNIQUE Functions
- Aggregate Functions
  - GROUP BY and HAVING Clauses
- Summary of Retrieval Queries

Nested Queries

- Complete select-from-where blocks within a WHERE clause of another query
  - Inner query
  - Outer query

Nested Queries

- Allow multiple levels of nested queries
  - Could cause a performance problem in processing
- Useful to fetch existing values in a database and then use them for a comparison in a WHERE condition
- Any nested queries can be *un-nested !*

Nested Queries

- **Query 4**. Make a list of all project numbers for projects that involve an employee whose last name is 'Smith', either as a worker on the project or as a manager of the department that controls the project

```
Q4A:  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' );
```

Comparison Operators

- IN operator
  - Compares value *v* with a set of values *V*
  - Evaluated TRUE if *v* is one of the elements in *V*
    - *v IN V*

Comparison Operators

- = ANY (or = SOME) operator
  - Returns TRUE if value *v* is equal to some value in set *V*
  - Equivalent to IN
  - Other operators that can be combined with ANY (or SOME)
    - >, >=, <, <=, and <>

Comparison Operators

- = ALL operator
  - Returns TRUE if value *v* is equal to *all* values in set *V*
  - Other operators that can be combined with ANY (or SOME)
    - >, >=, <, <=, and <>

Comparison Operators

- **Query A**. Search for the names of employees whose salary is greater than the salary of all the employees in department 5

```
SELECT    Lname, Fname
FROM      EMPLOYEE
WHERE     Salary > ALL  ( SELECT    Salary
                          FROM      EMPLOYEE
                          WHERE     Dno=5 );
```

Avoiding Ambiguities

- Possible ambiguity among attribute names
  - Attributes of the same name can exist
    - One in a table in the FROM clause of the outer query
    - Another in a table in the FROM clause of the inner query
- Reference rule
  - An *unqualified attribute* refers to the relation declared in the *innermost nested query*

Avoiding Ambiguities

- **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

```
Q16:  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 );
```

EXISTS and UNIQUE Functions

- EXISTS function
  - Check whether the result of a correlated nested query is empty or not
  - TRUE if there exist *at least one tuple* in the result of the nested query
  - FALSE if there are *no tuples* in the result of nested query

EXISTS and UNIQUE Functions

- **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

```
Q16B:  SELECT   E.Fname, E.Lname
       FROM     EMPLOYEE AS E
       WHERE    EXISTS ( SELECT   *
                         FROM     DEPENDENT AS D
                         WHERE    E.Ssn=D.Essn AND E.Sex=D.Sex
                         AND E.Fname=D.Dependent_name);
```

EXISTS and UNIQUE Functions

- NOT EXISTS function
  - Return the opposite result to EXIST function
  - TRUE if there are *no tuples* in the result of nested query
  - FALSE if there is *at least one tuple* in the result of the nested query

EXISTS and UNIQUE Functions

- **Query 6**. Retrieve the names of employees who have no dependents

```
Q6:   SELECT   Fname, Lname
      FROM     EMPLOYEE
      WHERE    NOT EXISTS ( SELECT   *
                            FROM     DEPENDENT
                            WHERE    Ssn=Essn );
```

EXISTS and UNIQUE Functions

- UNIQUE(*Q*) function
  - Check whether the result of query *Q* has any duplicate tuples or not
  - TRUE if there are *no duplicate tuples* in the result of the query *Q*
  - FALSE if there is *at least one duplicate tuple* in the result of query *Q*

Explicit Sets and NULL

- Usage of explicit sets
  - Can use a set of *explicit* values in the WHERE clause

Explicit Sets and NULL

- **Query 17**. Retrieve the social security numbers of all employees who work on project numbers 1, 2, or 3

```
Q17:    SELECT    DISTINCT Essn
        FROM      WORKS_ON
        WHERE     Pno IN (1, 2, 3);
```

Explicit Sets and NULL

- IS NULL function
  - Check whether an attribute value is NULL
  - TRUE if the attribute value is NULL
  - FALSE if the attribute value is not NULL
  - Each NULL value is *not equal* to other NULL values
    - Not possible to use = or <> to compare a value to NULL
    - While doing a JOIN operation, pairs of tuples with NULL values in the join attribute are *removed* from the result

Explicit Sets and 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;
```

Renaming Attributes

- Use qualifier AS followed by a desired new name
  - Rename any attribute that appears in the result of a query

Renaming Attributes

- **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

```
Q8A:    SELECT    E.Lname AS Employee_name, S.Lname AS Supervisor_name
        FROM      EMPLOYEE AS E, EMPLOYEE AS S
        WHERE     E.Super_ssn=S.Ssn;
```

Aggregate Functions

- Used to summarize information from multiple tuples into a single-tuple summary
- Built-in aggregate functions
  - COUNT, SUM, MAX, MIN, and AVG
  - COUNT returns the number of tuples specified in a query
  - SUM, MAX, MIN and AVG returns the sum, maximum value, minimum value, and average of tuples specified in a query

Aggregate Functions

- **Query 19**. Find the sum of the salaries of all employees, the maximum salary, the minimum salary, and the average salary

```
Q19:    SELECT    SUM (Salary), MAX (Salary), MIN (Salary), AVG (Salary)
        FROM      EMPLOYEE;
```

Aggregate Functions

- **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

```
Q20:    SELECT    SUM (Salary), MAX (Salary), MIN (Salary), AVG (Salary)
        FROM      (EMPLOYEE JOIN DEPARTMENT ON Dno=Dnumber)
        WHERE     Dname='Research';
```

Aggregate Functions

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

```
Q21:    SELECT    COUNT (*)
        FROM      EMPLOYEE;

Q22:    SELECT    COUNT (*)
        FROM      EMPLOYEE, DEPARTMENT
        WHERE     DNO=DNUMBER AND DNAME='Research';
```

Grouping

- Partition a relation into multiple groups of tuples
  - Apply an aggregate function to each group independently
- Grouping attribute(s)
  - Attributes used as the criteria for grouping
  - Each group consists of the tuples that have the same value for grouping attributes
  - Specified by GROUP BY clause
  - Should also appear in the SELECT clause

Grouping

- **Query 24**. For each department, retrieve the department number, the number of employees in the department, and their average salary

```
Q24:    SELECT    Dno, COUNT (*), AVG (Salary)
        FROM      EMPLOYEE
        GROUP BY  Dno;
```

Fname	Minit	Lname	Ssn	...	Salary	Super_ssn	Dno
John	B	Smith	123456789		30000	333445555	5
Franklin	T	Wong	333445555		40000	888665555	5
Ramesh	K	Narayan	666884444		38000	333445555	5
Joyce	A	English	453453453	...	25000	333445555	5
Alicia	J	Zelaya	999887777		25000	987654321	4
Jennifer	S	Wallace	987654321		43000	888665555	4
Ahmad	V	Jabbar	987987987		25000	987654321	4
James	E	Bong	888665555		55000	NULL	1

Grouping EMPLOYEE tuples by the value of Dno

Dno	Count (*)	Avg (Salary)
5	4	33250
4	3	31000
1	1	55000

Result of Q24

Grouping

- **Query 25**. For each project, retrieve the project number, the project name, and the number of employees who work on that project

```
Q25:    SELECT      Pnumber, Pname, COUNT (*)
        FROM        PROJECT, WORKS_ON
        WHERE       Pnumber=Pno
        GROUP BY    Pnumber, Pname;
```

HAVING clause

- Provides a condition on the summary information regarding the group of tuples
  - In conjunction with a GROUP BY clause

HAVING clause

- **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

```
Q26:    SELECT      Pnumber, Pname, COUNT (*)
        FROM        PROJECT, WORKS_ON
        WHERE       Pnumber=Pno
        GROUP BY    Pnumber, Pname
        HAVING      COUNT (*) > 2;
```

HAVING clause

- **Query 26**

Pname	Pnumber	...	Essn	Pno	Hours
ProductX	1		123456789	1	32.5
ProductX	1		453453453	1	20.0
ProductY	2		123456789	2	7.5
ProductY	2		453453453	2	20.0
ProductY	2		333445555	2	10.0
ProductZ	3		666884444	3	40.0
ProductZ	3		333445555	3	10.0
Computerization	10	...	333445555	10	10.0
Computerization	10		999887777	10	10.0
Computerization	10		987987987	10	35.0
Reorganization	20		333445555	20	10.0
Reorganization	20		987654321	20	15.0
Reorganization	20		888665555	20	NULL
Newbenefits	30		987987987	30	5.0
Newbenefits	30		987654321	30	20.0
Newbenefits	30		999887777	30	30.0

These groups are not selected by the HAVING condition of Q26.

After applying the WHERE clause but before applying HAVING

HAVING clause

- **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

Pname	Pnumber	...	Essn	Pno	Hours
ProductY	2		123456789	2	7.5
ProductY	2		453453453	2	20.0
ProductY	2		333445555	2	10.0
Computerization	10		333445555	10	10.0
Computerization	10	...	999887777	10	10.0
Computerization	10		987987987	10	35.0
Reorganization	20		333445555	20	10.0
Reorganization	20		987654321	20	15.0
Reorganization	20		888665555	20	NULL
Newbenefits	30		987987987	30	5.0
Newbenefits	30		987654321	30	20.0
Newbenefits	30		999887777	30	30.0

After applying the HAVING clause condition

Pname	Count (*)
ProductY	3
Computerization	3
Reorganization	3
Newbenefits	3

Result of Q26
(Pnumber not shown)

HAVING clause

- Note that the HAVING and WHERE clauses provide a common function specifying conditions
  - WHERE: To choose *tuples*
  - HAVING: To choose *groups of tuples*

Substring Pattern Matching

- LIKE comparison operator
  - Used for string pattern matching
  - '%' replaces an arbitrary number of zero or more characters
  - '_' (underscore) replaces a *single* character

Substring Pattern Matching

- **Query 12**. Retrieve all employees whose address is in Houston, Texas

```
Q12:    SELECT    Fname, Lname
        FROM      EMPLOYEE
        WHERE     Address LIKE '%Houston,TX%';
```

Substring Pattern Matching

- **Query 12A**. Find all employees who were born during the 1950s

```
Q12:    SELECT    Fname, Lname
        FROM      EMPLOYEE
        WHERE     Bdate LIKE '_ _ 5 _ _ _ _ _ _ _ _';
```

Arithmetic Operators

- Standard arithmetic operators:
  - Addition (+), subtraction (–), multiplication (*), and division (/)

Arithmetic Operators

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

```
Q13:    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

- The user can make orderings of tuples in the result of a query
  - By the values of one or more of the attributes that appear in the query result
  - Using ORDER BY clause

Ordering of Query Results

- Specifying order
  - Use DESC keyword for descending order
  - Use ASC keyword for ascending order
  - Ascending order by default
  - Example

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

Ordering of Query Results

- **Query 15**. Retrieve a list of employees and the projects they are working on, ordered by department and, within each department, ordered alphabetically by last name, then first name

```
Q15:    SELECT    D.Dname, E.Lname, E.Fname, P.Pname
        FROM      DEPARTMENT D, EMPLOYEE E, WORKS_ON W,
                  PROJECT P
        WHERE     D.Dnumber= E.Dno AND E.Ssn= W.Essn AND
                  W.Pno= P.Pnumber
        ORDER BY  D.Dname, E.Lname, E.Fname;
```

Summary of Retrieval Queries

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

Summary

- More complex retrieval queries
  - Nested queries
  - Aggregate functions
  - Grouping

References

1. Reisner, Phyllis. "Use of Psychological Experimentation as an Aid to Development, of Language." *IEEE Transactions on Software Engineering* 3.3, 1977.
2. Date, C. J. "A critique of the SQL database language." *ACM SIGMOD Record* 14.3 (1984): 8-54.
3. Date, Chris J., and Hugh Darwen. "A Guide to the SQL Standard", 1993.
4. American National Standards Institute: **The Database Language SQL**, Document ANSI X3.135, 1986.
5. Melton, Jim, and Alan R. Simon. "Understanding the New SQL: A Complete Guide Morgan Kaufmann Publishers." *San Francisco, CA*, 1993.
6. Horowitz, Bruce M. "A run-time execution model for referential integrity maintenance." *Data Engineering, 1992. Proceedings. Eighth International Conference on*. IEEE, 1992.

Have a nice day!