UNIVERSITAS INDONESIA
Veritas, Probitas, Iustitia
FAKULTAS
ILMU KOMPUTER

Slide 4

SQL: Advanced Query (2)

CSF2600700 - BASIS DATA

SEMESTER GENAP 2019/2020

These slides are a modification to the supplementary slide of “Database System”, 7th edition, Elmasri/Navathe, 2015: **Chapter 7 More SQL: Complex Queries, Triggers, Views, and Schema Modification**

Review: SQL yang Sudah Di Pelajari

- DDL
- Basic SQL Query, cartesian product.
- Advanced Query: nested query, join, exists

Outline

More Complex SQL Retrieval Queries: Grouping, Aggregate Function

Views (Virtual Tables) in SQL

Schema Change Statements in 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 17: Retrieve the social security numbers of all employees who work on project number 1, 2, **or** 3.

Q17:

```
SELECT    DISTINCT ESSN
FROM WORKS_ON
WHERE     PNO IN (1, 2, 3);
```

Q17A:

```
SELECT    DISTINCT ESSN
FROM WORKS_ON
WHERE     PNO = ANY (array[1, 2, 3]);
```

Renaming Attribute

In SQL, its possible to rename attribute that appears in the result of a query by adding the qualifier AS followed by the desired new name

Q8A:

```
    SELECT E.Lname AS EMPLOYEE_NAME,
           S.Lname AS SUPERVISOR_NAME
    FROM EMPLOYEE E, EMPLOYEE S
    WHERE E.SUPERSSN = S.SSN;
```

Aggregate Function

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

Query : Find the maximum salary, the minimum salary, and the average salary among all employees.

```
SELECT    MAX(SALARY),MIN(SALARY),AVG(SALARY)
FROM      EMPLOYEE
```

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

AGGREGATE FUNCTION

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

```
SELECT   MAX(SALARY), MIN(SALARY),
         AVG(SALARY)
FROM     EMPLOYEE, DEPARTMENT
WHERE    DNO=DNUMBER AND
         DNAME='Research`;
```

AGGREGATE FUNCTION

Queries : Retrieve the total number of employees in the company (QA), and the number of employees in the 'Research' department (QB).

QA:

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

QB:

```
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

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
```

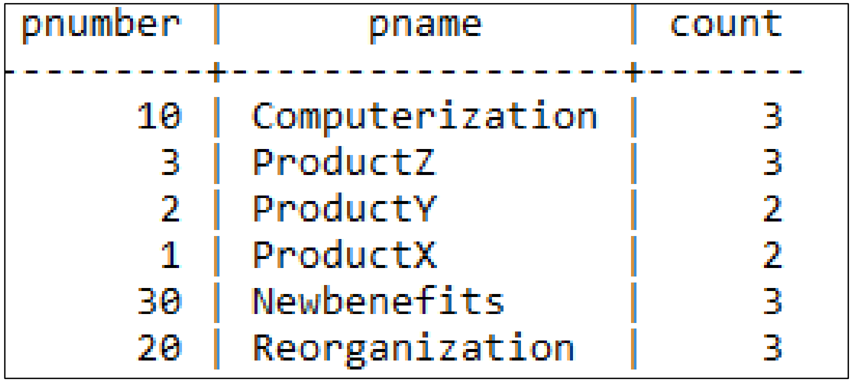
- In Q24, the EMPLOYEE tuples are divided into groups--each group having the same value for the grouping attribute DNO
- The COUNT and AVG functions are applied to each such group of tuples separately
- The SELECT-clause includes only the grouping attribute and the functions to be applied on each group of tuples
- A join condition can be used in conjunction with grouping

GROUPING

Query 25: For each project, retrieve the project number, 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
```

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



```
 pnumber |       pname       | count
---------+-------------------+-------
      10 | Computerization   |     3
       3 | ProductZ          |     3
       2 | ProductY          |     2
       1 | ProductX          |     2
      30 | Newbenefits       |     3
      20 | Reorganization    |     3
```

THE HAVING CLAUSE

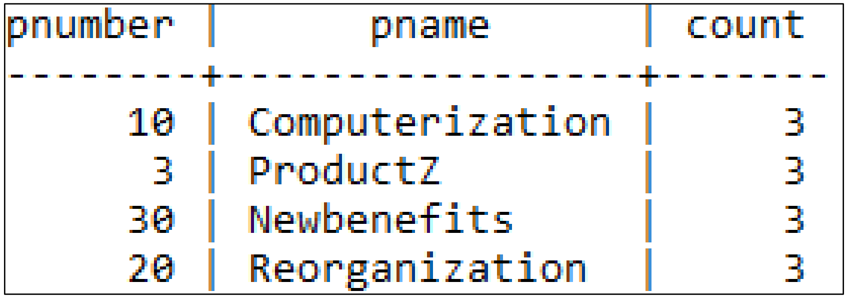
Sometimes we want to retrieve the values of these functions for only those *groups that satisfy certain conditions*

The HAVING-clause is used for specifying a selection condition on groups (rather than on individual tuples)

THE HAVING CLAUSE

Query 26: 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.

```
Q26:    SELECT      PNUMBER, PNAME, COUNT (*)
        FROM        PROJECT, WORKS_ON
        WHERE       PNUMBER=PNO
        GROUP BYPNUMBER, PNAME
        HAVING      COUNT (*) > 2
```



```
pnumber |        pname        | count
---------+---------------------+-------
      10 | Computerization     |     3
       3 | ProductZ            |     3
      30 | Newbenefits         |     3
      20 | Reorganization      |     3
```

VIEWS in SQL

A view is a “virtual” table that is derived from other tables

Allows for limited update operations (since the table may not physically be stored)

Allows full query operations

A convenience for expressing certain operations

Specification of Views

SQL command: `CREATE VIEW`

- a table (view) name
- a possible list of attribute names (for example, when arithmetic operations are specified or when we want the names to be different from the attributes in the base relations)
- a query to specify the table contents

SQL Views: An Example

Specify a different WORKS_ON table

```
CREATE VIEW WORKS_ON_NEW AS
SELECT FNAME, LNAME, PNAME, HOURS
        FROM EMPLOYEE, PROJECT, WORKS_ON
        WHERE SSN=ESSN AND PNO=PNUMBER
```

Using a Virtual Table

We can specify SQL queries on a newly create table (view):

```
SELECT FNAME, LNAME FROM WORKS_ON_NEW
WHERE PNAME='Seena';
```

When no longer needed, a view can be dropped:

```
DROP VIEW WORKS_ON_NEW;
```

Efficient View Implementation

Query modification: present the view query in terms of a query on the underlying base tables

- disadvantage: inefficient for views defined via complex queries (especially if additional queries are to be applied to the view within a short time period)

Efficient View Implementation

View materialization: involves physically creating and keeping a temporary table

- assumption: other queries on the view will follow
- concerns: maintaining correspondence between the base table and the view when the base table is updated
- strategy: incremental update

View Update

Update on a single view without aggregate operations: update may map to an update on the underlying base table

Views involving joins: an update *may* map to an update on the underlying base relations

- not always possible

Un-updatable Views

Views defined using groups and aggregate functions are not updateable

Views defined on multiple tables using joins are generally not updateable

`WITH CHECK OPTION`: must be added to the definition of a view if the view is to be updated

- to allow check for updatability and to plan for an execution strategy

CREATE VIEW authors_CA AS (SELECT * FROM Authors WHERE state='CA') WITH CHECK OPTION

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

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

Exercise

1. Tampilkan rata-rata salary pada pegawai perempuan.
2. Tampilkan jumlah project yang dikerjakan oleh setiap pegawai.
3. Tampilkan nama departemen dan jumlah project yang ditangani department tersebut.
4. Tampilkan nama depan semua pegawai yang bekerja di sebuah department yang memiliki pegawai dengan salary tertinggi.
5. Untuk setiap department, tampilkan nama departemen dan rata-rata salary pegawai yang bekerja pada department tersebut.
6. Untuk setiap department yang rata-rata salary pegawainya kurang dari 50000, tampilkan nama department beserta jumlah pegawainya.
7. Untuk setiap supervisor, tampilkan nama depan supervisor dan akumulasi jumlah jam kerja employee yang disupervisinya. Tampilkan hanya yang akumulasi jumlah jam kerjanya employee lebih dari 50 jam.
8. Buatlah sebuah view bernama EMPLOYEE_DEPENDENT yang berisi daftar nama depan pegawai yang memiliki dependent dan jumlah dependant yang dimilikinya.