Chapter

SQL DML

Week 4 Readings (SQL):

4.4
5.1

Quiz 1

- Average

Survey

- Labs
- Quizzes

SQL: Structured Query Language.

- SQL is the "standard" query language for relational databases.
- All relational DBMSs implement a standard (or a subset) of SQL.
- SQL can be used interactively (on-line) from a terminal or can be executed in programs.
- SQL is a complete language.
  - DDL (Data Definition Language)
  - **DML (Data Manipulation Language)**
  - DCL (Data Control Language)

AGGREGATE FUNCTI

- Include COUNT, SUM, MAX, MIN, and A
- Apply to a column and return a single re
- **Example:**

```
ritu=> select * fr
 sno | pno | qty
-----+-----+-----
 S1  | P1  | 200
 S2  | P3  | 400
 S2  | P5  | 100
 S3  | P3  | 200
 S3  | P4  | 500
 S4  | P6  | 300
 S5  | P2  | 200
 S5  | P5  | 500
 S5  | P6  | 200
 S5  | P1  | 100
 S5  | P3  | 200
 S5  | P4  | 800
(12 rows)
```

SELECT COUNT(SNO), MAX(QTY), SUM(QTY)
FROM SP;

```
 count | max | sum
-------+-----+------
    12 | 800 | 3700
```

AGGREGATE FUNCT

```
ritu=> select * from
 sno | pno | qty
-----+-----+-----
 S1  | P1  | 200
 S2  | P3  | 400
 S2  | P5  | 100
 S3  | P3  | 200
 S3  | P4  | 500
 S4  | P6  | 300
 S5  | P2  | 200
 S5  | P5  | 500
 S5  | P6  | 200
 S5  | P1  | 100
 S5  | P3  | 200
 S5  | P4  | 800
(12 rows)
```

- **Example:**

SELECT COUNT(DISTINCT SNO), COUNT(*)
FROM SP;

```
 count | count
-------+-------
     5 |    12
(1 row)
```

COUNT(*) gives the number of tuples in the result of FROM (and WHERE if any).

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

```
ritu=> select * from sp;
 sno | pno | qty
-----+-----+-----
 S1  | P1  | 200
 S2  | P3  | 400
 S2  | P5  | 100
 S3  | P3  | 200
 S3  | P4  | 500
 S4  | P6  | 300
 S5  | P2  | 200
 S5  | P5  | 500
 S5  | P6  | 200
 S5  | P1  | 100
 S5  | P3  | 200
 S5  | P4  | 800
(12 rows)
```

How many parts does each supplier supply?

S1 supplies 1 part	P1
S2 supplies 2 parts	P3
	P5
S3 supplies 2 parts	P3
	P4
S4 supplies 1 part	P6
S5 supplies 6 parts	P1
	P2
	P3
	P4
	P5
	P6

GROUP BY

```
ritu=> select * from sp;
 sno | pno | qty
-----+-----+-----
 S1  | P1  | 200
 S2  | P3  | 400
 S2  | P5  | 100
 S3  | P3  | 200
 S3  | P4  | 500
 S4  | P6  | 300
 S5  | P2  | 200
 S5  | P5  | 500
 S5  | P6  | 200
 S5  | P1  | 100
 S5  | P3  | 200
 S5  | P4  | 800
(12 rows)
```

How many parts does each supplier supply?

```
SELECT      SNO, COUNT(PNO)
FROM        SP
GROUP BY SNO;
```

```
 sno | count
-----+-------
 S3  |     2
 S1  |     1
 S4  |     1
 S5  |     6
 S2  |     2
(5 rows)
```

GROUP BY

```
SELECT        SNO, COUNT(PNO)
FROM  SP
GROUP BY SNO;
```

- The result of the SELECT-FROM-WHERE query is grouped according to the values of the attributes listed in GROUP BY
- any aggregation is applied only within each group and gives a single value per group

GROUP BY - restrictions

If an aggregate function is used, then each element of the SELECT list must be either:

- an aggregate function (MAX, MIN, COUNT, AVG, SUM) or

- an attribute on the GROUP BY list.

```
SELECT   SNO, MAX(Qty)
FROM     SP
GROUP    BY SNO;
```

GROUP BY - restrictions

For example, the following queries are invalid.

```
SELECT    SNO, MAX(Qty)
FROM      SP;
```

Valid:

```
SELECT    SNO, MAX(Qty)
FROM      SP
GROUP     BY SNO;
```

```
SELECT    SNO
FROM      SP
WHERE     Qty = MAX(QTY)
```

Valid:

```
SELECT   SNO
FROM     SP
WHERE  Qty = (SELECT MAX(QTY)
              FROM SP);
```

GROUP BY - example

Write a query to list supplier numbers and the total qty they supply, in ascending order of the total.

```
SELECT SNO, SUM(QTY) SQ
FROM SP
GROUP BY SNO
ORDER BY SQ;
```

```
SNO|  SQ
-----+------
 S1  |  200
 S4  |  300
 S2  |  500
 S3  |  700
 S5  | 2000
(5 rows)
```

```
ritu=> select * from sp;
 sno | pno | qty
-----+-----+-----
 S1  | P1  | 200
 S2  | P3  | 400
 S2  | P5  | 100
 S3  | P3  | 200
 S3  | P4  | 500
 S4  | P6  | 300
 S5  | P2  | 200
 S5  | P5  | 500
 S5  | P6  | 200
 S5  | P1  | 100
 S5  | P3  | 200
 S5  | P4  | 800
(12 rows)
```

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)

HAVING - restrictions

- The same requirement as for SELECT clauses with aggregation

HAVING may refer to attributes only if they are either:

- aggregated, or
- an attribute on the GROUP BY list.

HAVING - Example

Write a query to list supplier numbers and the total qty they supply, if the total is more than 1000.

```
ritu=> SELECT SNO, SUM(QTY) SQ
        FROM SP
        GROUP BY SNO
        HAVING SUM(QTY) >1000;
 sno |  sq
-----+------
 S5  | 2000
(1 row)
```

HAVING - Example

Write a query to find all parts that are supplied by more than 1 supplier.

```
SELECT    pno
FROM      SP
GROUP     BY pno
HAVING    COUNT(*)>1;
```

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>]**

Next Class

- Correlated Subqueries