CSCI 403: Databases
7 - Aggregate Functions and GROUP BY

Aggregate Functions

SQL has a number of *aggregate* functions - functions which summarize a whole relation or part of a relation. The aggregate functions are:

- COUNT - counts the number of non-NULL entries in a column, or non-NULL tuples in a relation
- SUM - adds the non-NULL entries in a column
- MAX - finds the maximum non-NULL entry in a column
- MIN - finds the minimum non-NULL entry in a column
- AVG - takes the average of the non-NULL entries in a column

Note that each of these results in a scalar result. Examples:

```
SELECT COUNT(*) FROM mines_courses;
SELECT COUNT(instructor) FROM
mines_courses;
SELECT AVG(max_credits) FROM
mines_courses;
```

Grouping

Grouping lets us compute aggregates on subgroups of rows, as organized by distinct values of some subset of attributes of the relation. The general form of a grouping query is:

```
SELECT A1, A2, ..., FN1(A3), FN2(A4),
...
FROM table1, table2, ...
WHERE conditions
GROUP BY A1, A2, ...;
```

In the above, FN represents some aggregate function. In a GROUP BY query, the most important thing to understand is that the set of attributes SELECTed in addition to the aggregate function expressions must be a subset of the attributes in the GROUP BY clause. That is, if we SELECT an attribute, we must also GROUP BY it, or the query is invalid.

The result of the GROUP BY query is a set of tuples with the distinct values of the selected attributes together with aggregate functions computed on the subset of tuples from the relation which share the distinct values. For instance,

```
SELECT instructor, COUNT(*)
FROM mines_courses
GROUP BY instructor;
```

will return all distinct instructors (including NULL as a distinct "value") and the count of tuples in the mines_courses table for each instructor. (In the case of this query, it is a count of the distinct course CRNs associated with an instructor - not exactly a measure of how many courses an instructor teaches.)

Grouping is applied *after* any WHERE conditions are applied.

GROUP BY can be combined with ORDER BY, but only if ORDER BY is applied to either attributes in the GROUP BY clause or to any (not necessarily SELECTed) aggregate functions on the relation. (Also, renaming of aggregate function result columns is often a good idea for readability in the result, and the column alias can be used in the ORDER BY clause.) For example, either of the following is valid:

```
SELECT instructor, COUNT(*)
FROM mines_courses
GROUP BY instructor
ORDER BY instructor;
```

and

```
SELECT instructor, COUNT(*) AS
count
FROM mines_courses
```

```
GROUP BY instructor
ORDER BY count;
```

Another example, showing that you can ORDER BY an aggregate function not SELECTED on:

```
SELECT instructor, COUNT(*)
FROM mines_courses
GROUP BY instructor
ORDER BY SUM(max_credits), instructor;
```

A final example, answering the question: "What MWF timeslots have the most courses scheduled at Mines?"

```
SELECT begin_time, COUNT(*)
FROM mines_courses_meetings
WHERE days = 'MWF'
GROUP BY begin_time
ORDER BY COUNT(*) DESC;
```

HAVING

Sometimes it is desirable to filter grouped data based on some condition which can be applied to the aggregate functions computed on the groups. The HAVING clause acts like a WHERE clause which is applied *after* grouping (as opposed to WHERE, which is applied before grouping). For instance, suppose we want to list only instructors responsible for more than 5 CRNs:

```
SELECT instructor, count(*)
FROM mines_courses
GROUP BY instructor
HAVING count(*) > 5;
```

You can have both WHERE and HAVING conditions in your query.