Today
- Group by and Having (wrap up)
- Views and Prepared Statements

Assignments
- HW5 out
- Quiz next Tues
Example Tables

account

<table>
<thead>
<tr>
<th>number</th>
<th>owner</th>
<th>balance</th>
<th>type</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>J. Smith</td>
<td>1000.00</td>
<td>checking</td>
</tr>
<tr>
<td>102</td>
<td>W. Wei</td>
<td>2000.00</td>
<td>checking</td>
</tr>
<tr>
<td>103</td>
<td>J. Smith</td>
<td>5000.00</td>
<td>savings</td>
</tr>
<tr>
<td>104</td>
<td>M. Jones</td>
<td>1000.00</td>
<td>checking</td>
</tr>
<tr>
<td>105</td>
<td>H. Martin</td>
<td>10000.00</td>
<td>checking</td>
</tr>
</tbody>
</table>

deposit

<table>
<thead>
<tr>
<th>account</th>
<th>transaction_id</th>
<th>date</th>
<th>amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>102</td>
<td>1</td>
<td>10/22/11</td>
<td>500.00</td>
</tr>
<tr>
<td>102</td>
<td>2</td>
<td>10/29/11</td>
<td>200.00</td>
</tr>
<tr>
<td>104</td>
<td>3</td>
<td>10/29/11</td>
<td>1000.00</td>
</tr>
<tr>
<td>105</td>
<td>4</td>
<td>11/2/11</td>
<td>10000.00</td>
</tr>
</tbody>
</table>

check

<table>
<thead>
<tr>
<th>account</th>
<th>check_number</th>
<th>date</th>
<th>amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>924</td>
<td>10/23/11</td>
<td>125.00</td>
</tr>
<tr>
<td>101</td>
<td>925</td>
<td>10/24/11</td>
<td>23.98</td>
</tr>
<tr>
<td>102</td>
<td>746</td>
<td>10/25/11</td>
<td>51.73</td>
</tr>
</tbody>
</table>

FK's:

- deposit.account → account.number
- check.account → account.number
Group By and Having

**GROUP BY clause creates one row per “group”**

- For example:

```
SELECT owner, AVG(balance)
FROM account
GROUP BY owner
```

- each distinct owner value forms a “group”
- we compute the average balance over each group

<table>
<thead>
<tr>
<th>owner</th>
<th>AVG(balance)</th>
</tr>
</thead>
<tbody>
<tr>
<td>H. Martin</td>
<td>10000</td>
</tr>
<tr>
<td>J. Smith</td>
<td>3000</td>
</tr>
<tr>
<td>M. Jones</td>
<td>1000</td>
</tr>
<tr>
<td>W. Wei</td>
<td>2000</td>
</tr>
</tbody>
</table>

**A HAVING clause filters groups**

- For example:

```
SELECT owner, AVG(balance)
FROM account
GROUP BY owner
HAVING COUNT(*) > 1
```

- each group must satisfy the having clause

<table>
<thead>
<tr>
<th>owner</th>
<th>AVG(balance)</th>
</tr>
</thead>
<tbody>
<tr>
<td>J. Smith</td>
<td>3000</td>
</tr>
</tbody>
</table>
Group By attributes

What can appear in a SELECT clause under GROUP BY?

- the grouping attributes (attributes in GROUP BY clause) ... (*) see below
- or aggregate operators (which are applied to the group)

This query is not legal

```sql
SELECT name
FROM customer
GROUP BY salesperson;
```

Q: Why not?

- name is not a grouping attribute, and is not an aggregate
- there could be many names for each group!

Q: What if we group by a primary key?

- then, there could only be one value for other attributes
- e.g.: `SELECT name FROM customer GROUP BY number`
- (*) this is legal in SQL (actually, any functional dependency; more later)

Multiple attributes can appear in a GROUP BY

- Same rule applies: No two rows in a group have different values for attributes in the GROUP BY
What can appear in a HAVING clause?

- Comparisons to only those attributes in the GROUP BY clause
  - same rules apply w.r.t. keys as well
- Comparisons to expressions (like aggregates) that result in a single value for the group

HAVING clause without a GROUP BY

- Treats result of FROM and WHERE as a single group

Q: How does this differ to a WHERE clause?

- WHERE applies to each row
- HAVING applies to the whole group
More on MySQL

Creating Views

• Views provide a way to name a query

• The query can be used in a table within another query

• But, the query result is never stored (not “materialized”)

Example table:

<table>
<thead>
<tr>
<th>id</th>
<th>common_name</th>
<th>heat</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>bell</td>
<td>mild</td>
</tr>
<tr>
<td>2</td>
<td>jalapeno</td>
<td>hot</td>
</tr>
<tr>
<td>3</td>
<td>banana</td>
<td>mild</td>
</tr>
<tr>
<td>4</td>
<td>cayenne</td>
<td>hot</td>
</tr>
<tr>
<td>5</td>
<td>waxy</td>
<td>medium</td>
</tr>
<tr>
<td>6</td>
<td>habanero</td>
<td>very hot</td>
</tr>
</tbody>
</table>

Example of creating a view using MySQL:

```sql
mysql> CREATE VIEW hot_pepper AS
    -> SELECT id, common_name
    -> FROM pepper
    -> WHERE heat='hot' OR heat='very hot';
```

```sql
mysql> show tables;
+-------------------------+
| Tables_in_bowers_schema |
+-------------------------+
| hot_pepper              |
| pepper                  |
+-------------------------+
```
Example continued ...

```
mysql> SELECT * FROM hot_pepper;
+----+-------------+
| id | common_name |
+----+-------------+
|  2 | jalapeno    |
|  4 | cayenne     |
|  6 | habanero    |
+----+-------------+

mysql> INSERT INTO pepper VALUES (7, 'serrano', 'hot');

mysql> SELECT * FROM hot_pepper;
+----+-------------+
| id | common_name |
+----+-------------+
|  2 | jalapeno    |
|  4 | cayenne     |
|  6 | habanero    |
|  7 | serrano     |
+----+-------------+
```
More features

In MySQL, you can use LIMIT to restrict number of rows returned in a query:

```sql
mysql> SELECT * FROM hot_pepper LIMIT 1;
+----+------------+
| id | name      |
+----+------------+
| 2  | jalapeno   |
+----+------------+
```

You can also specify a starting offset row ... limit to y rows starting at row x:

```sql
mysql> SELECT * FROM hot_peppers LIMIT 1, 2;
+----+------------+
| id | name      |
+----+------------+
| 4  | cayenne    |
| 6  | habanero   |
+----+------------+
```

Often useful with ORDER BY ... ("top-k" queries)

```sql
mysql> SELECT * FROM hot_peppers ORDER BY common_name LIMIT 1;
+----+------------+
| id | name      |
+----+------------+
| 4  | cayenne    |
+----+------------+
```
ORDER BY can be in ascending (ASC) or descending order (DESC)

- ascending order is the default
- can also order by multiple attributes e.g., ORDER BY office, name
- can mix and match order direction, e.g., ORDER BY office, name DESC

Removing a view is similar to dropping a table:

```
mysql> drop view hot_peppers;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from hot_peppers;
ERROR 1146 (42S02): Table 'bowers.hot_peppers' doesn't exist
```

Prepared statements

- Name a parameterized query
- Store the query on the server

```
mysql> PREPARE get_pepper_name FROM
-> "SELECT name FROM ingredients WHERE id=?";

mysql> SET @my_var = 3;

mysql> EXECUTE get_pepper_name USING @my_var;
```

```
+-------------+
| common_name |
+-------------+
| bell        |
+-------------+
```

- Prepared statements are “compiled” (parsed, analyzed, optimized)