CS 360 Internet Programming

Web Database Applications

MySQL Basics

Daniel Zappala
1 Databases and Tables
   - Creating Databases and Tables
   - Attributes and Keys
   - Deleting Databases and Tables

2 Inserting, Deleting, and Updating Data
   - Inserting Data
   - Deleting Data
   - Updating Data

3 Querying with SQL SELECT
   - WHERE Clauses
   - Sorting and Grouping
   - Join Queries
Starting the Command Interpreter

1. `% mysql -h ilab.cs.byu.edu -u name -p`
2. `> source statements.sql`

> will prompt for password
1  > `CREATE DATABASE juicestore;`
2
3  > `use juicestore;`
Creating Tables

```
CREATE TABLE customer ( // name the table
  cust_id int(5) NOT NULL, // specify attributes
  surname varchar(50),
  firstname varchar(50),
  initial char(1),
  title_id int(3),
  address varchar(50),
  city varchar(50),
  state varchar(20),
  zipcode varchar(10),
  country_id int(4),
  phone varchar(15),
  birth_date char(10),
  PRIMARY KEY (cust_id) // create the primary key
) type=MyISAM; // does not support transactions
```
Attribute Types

- **int(length)**: integer with a maximum length
- **decimal(width[,decimal_digits])**: float
- **datetime**: date and time in the format YYYY-MM-DD HH:MM:SS
- **time**: time in the format HH:MM:SS
- **date**: date in the format YYYY-MM-DD.
- **timestamp**: date and time in the format YYYYMMDDHHMMSS.
  - first-occurring timestamp attribute in a row is set to the current date and time when that created or modified.
  - timestamp will also be updated if you set it to NULL
- **varchar(length)**: unpadded, variable-length string
- **char(length)**: padded, fixed-length string
- **blob**: stores up to 64 KB of data
Attribute Modifiers

- **NOT NULL**: attribute must have a value
- **DEFAULT**: default value
- **zerofill**: left-pads a number with zeros
- **unsigned**: only positive values, doubles the maximum positive value
- **auto_increment**: automatically increments to next integer when set to NULL
Keys

- **primary key**: uniquely identifies a record
- can add additional keys
  - database will create an index for each key to provide faster lookups based on the key
  - each index takes additional space and must be updated for each insert, delete, modify operation
Deleting Databases and Tables

```
1 DROP TABLE customer;
2
3 DROP DATABASE juicestore;
4
5 DROP DATABASE IF EXISTS juicestore;
6
7 DROP TABLE IF EXISTS customer;
```
Inserting Data

1. INSERT INTO customer VALUES (1, 'Williams', 'Lucy', 'E', 3, 
2. '272 Station St', 'Carlton North', 'VIC', '3054', 12, '(613)83008460', 
3. '2002–07–02');

- number of values inserted must match the number of attributes
- must know ordering of attributes in table: use SHOW COLUMNS FROM customer
- may include NULL if the attribute allows this value
- may insert multiple rows at a time
Inserting Data

```
1  INSERT INTO customer SET cust_id = 1, surname = 'Williams',
2       firstname = 'Lucy', initial = 'E', title_id = 3,
3       address = '272 Station St', city = 'Carlton North',
4       state = 'VIC', zipcode = '3054', country_id = 12,
5       phone = '(613)83008460', birth_date = '2002-07-10';
```

- list attribute names explicitly
- may skip some attributes
- may use a different attribute order
Default Values and Auto-Increment

- **default values**
  - if attribute is not included in INSERT, it is set to DEFAULT value if specified
  - if no DEFAULT value and NOT_NULL is not set, the value is set to NULL
  - if no DEFAULT and NOT_NULL is set, then integers are set to 0, and strings to ""

- **auto increment**
  - insert NULL as value for an attribute with auto increment set
  - only one attribute in a table may have this feature
Deleting Data

1. `DELETE FROM customer;`
   - deletes all records in customer table

2. `DELETE FROM customer WHERE cust_id = 1;`

3. `DELETE FROM customer WHERE surname = 'Smith';`
   - deletes only matching records
Updating Data

1. `UPDATE customer SET state = upper(state);`
2. 
3. `UPDATE customer SET state = upper(state), city = upper(city);`

- updates all records in customer table

1. `UPDATE customer SET surname = 'Smith' WHERE cust_id = 7;`
2. 
3. `UPDATE customer SET zipcode = '3001' WHERE city = 'Melbourne';`
### Basic Query

1. SELECT surname, firstname FROM customer;

<table>
<thead>
<tr>
<th>surname</th>
<th>firstname</th>
</tr>
</thead>
<tbody>
<tr>
<td>Marzalla</td>
<td>Dimitria</td>
</tr>
<tr>
<td>LaTrobe</td>
<td>Anthony</td>
</tr>
<tr>
<td>Fong</td>
<td>Nicholas</td>
</tr>
<tr>
<td>Stribling</td>
<td>James</td>
</tr>
</tbody>
</table>

4 rows in set (0.04 sec)

14. SELECT * FROM region;
WHERE Clauses

1. `SELECT region_name FROM region WHERE region_id <= 3;`
2. +________________________________________+
3. | region_name |
4. +________________________________________+
5. | All |
6. | Goulburn Valley |
7. | Rutherglen |
8. +________________________________________+
9. 3 rows in set (0.01 sec)
Complex WHERE Clauses

```
1  SELECT cust_id FROM customer
2    WHERE (surname='Marzalla' AND firstname LIKE 'M%') OR
3           birth_date='1980-07-14';

<table>
<thead>
<tr>
<th>cust_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>440</td>
</tr>
<tr>
<td>493</td>
</tr>
</tbody>
</table>

2 rows in set (0.01 sec)
```
### Sorting Output

```
1 SELECT surname, firstname, initial FROM customer
2 WHERE city = 'Coonawarra' OR city = 'Longwood'
3 ORDER BY surname, firstname, initial;

<table>
<thead>
<tr>
<th>surname</th>
<th>firstname</th>
<th>initial</th>
</tr>
</thead>
<tbody>
<tr>
<td>Archibald</td>
<td>Belinda</td>
<td>Q</td>
</tr>
<tr>
<td>Chester</td>
<td>Marie</td>
<td>S</td>
</tr>
<tr>
<td>Dalion</td>
<td>Marie</td>
<td>C</td>
</tr>
<tr>
<td>Eggelston</td>
<td>Martin</td>
<td>E</td>
</tr>
<tr>
<td>Florenini</td>
<td>Melinda</td>
<td>O</td>
</tr>
<tr>
<td>Holdenson</td>
<td>Jasmine</td>
<td>F</td>
</tr>
<tr>
<td>Mellaseca</td>
<td>Craig</td>
<td>Y</td>
</tr>
<tr>
<td>Mockridge</td>
<td>Dimitria</td>
<td>I</td>
</tr>
</tbody>
</table>
```
Grouping Output

- group matching rows
- report number of rows in each group
- COUNT(), SUM(), MAX(), MIN(), AVG()

```
1 SELECT city, COUNT(*) FROM customer GROUP BY city;
2
3  | city      | COUNT(*) |
4  | Alexannda | 14       |
5  | Armidale   | 7        |
6  | Athlone    | 9        |
7  | Bauple     | 6        |
8  | Belmont    | 11       |
9  | Bentley    | 10       |
10 | Beralal    | 9        |
11 | Broadmeadows | 11 |
```
Combining Clauses

```
1 SELECT city, surname, firstname, \textit{count}(\ast) \text{ FROM} \ customer
2 \hspace{1em} \textbf{WHERE} \ state = 'VIC'
3 \hspace{1em} \textbf{GROUP BY} \ surname, firstname \textbf{HAVING} \ count(\ast) \geq 2
4 \hspace{1em} \textbf{ORDER BY} \ city;

6
7 \hspace{1em} \begin{tabular}{|c|c|c|c|}
8 \hline
9 \text{city} & \text{surname} & \text{firstname} & \text{count}(\ast) \\
10 \text{Broadmeadows} & \text{Mellaseca} & \text{Anthony} & 2 \\
11 \text{Eleker} & \text{Leramonth} & \text{Harry} & 2 \\
12 \text{Kalimna} & \text{Galti} & \text{Nicholas} & 2 \\
13 \text{Lucknow} & \text{Mellili} & \text{Derryn} & 2 \\
14 \text{McLaren} & \text{Chester} & \text{Betty} & 2 \\
15 \hline
16 \end{tabular}
17 5 \textit{rows in set} (0.00 \text{ sec})
```
Join Queries

- match rows from tables based on relationship
- example: which customers that live in Australia have placed orders

```
SELECT juicery_name, region_name FROM juicery, region
ORDER BY juicery_name, region_name;
```

<table>
<thead>
<tr>
<th>juicery_name</th>
<th>region_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anderson and Sons Premium Juices</td>
<td>All</td>
</tr>
<tr>
<td>Anderson and Sons Premium Juices</td>
<td>Barossa Valley</td>
</tr>
<tr>
<td>Anderson and Sons Premium Juices</td>
<td>Coonawarra</td>
</tr>
<tr>
<td>Anderson and Sons Premium Juices</td>
<td>Goulburn Valley</td>
</tr>
<tr>
<td>Anderson and Sons Premium Juices</td>
<td>Lower Hunter Valley</td>
</tr>
</tbody>
</table>

- displays all possible combinations of juiceries and regions
Natural Joins

- want to output `juicery_name` and `region_name` values by matching rows from the `juicery` and `region` tables.
- Query below automatically matches `region_id` attributes.

```sql
1 SELECT juicery_name, region_name FROM juicery NATURAL JOIN region
2 ORDER BY juicery_name;
```

<table>
<thead>
<tr>
<th>juicery_name</th>
<th>region_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anderson and Sons Premium Juices</td>
<td>Coonawarra</td>
</tr>
<tr>
<td>Anderson and Sons Juices</td>
<td>Coonawarra</td>
</tr>
<tr>
<td>Anderson Brothers Group</td>
<td>Rutherglen</td>
</tr>
<tr>
<td>Anderson Creek Group</td>
<td>Riverland</td>
</tr>
<tr>
<td>Anderson Daze Group</td>
<td>Rutherglen</td>
</tr>
</tbody>
</table>