Structured Query Language (SQL) - 2

Databases and Interfaces

Matthew Pike

This Lecture

- Introduction to SQL
  - What is SQL?
  - How to:
    - INSERT data into tables
    - SELECT data from tables
    - UPDATE data in tables
  - Also:
    - DELETE
    - ALTER

Recap

- Thus far we have
  - Converted a written (English) description to ER
  - Converted an ER representation to SQL
- Next
  - We need to add and query data

INSERT

INSERT

- Inserts rows into the database with the specified values

```
INSERT INTO
   table-name
   (col1, col2, …)
    VALUES
   (val1, val2, …);
```

- The number of columns and the number of values must be the same
- If you are adding a value to every column, you don't have to list them
- If you don't list columns, be careful of the ordering

INSERT

Employee		
ID	Name	Salary
1	John	25000

```
INSERT INTO Employee
    (ID, Name, Salary)
    VALUES (2, 'Mary', 26000);

INSERT INTO Employee
    (Name, ID)
    VALUES ('Mary', 2);

INSERT INTO Employee
    VALUES (2, 'Mary', 26000);
```

INSERT

```
INSERT INTO Employee
    (ID, Name, Salary)
    VALUES (2, 'Mary', 26000);
```

2	Mary	26000

```
INSERT INTO Employee
    (Name, ID)
    VALUES ('Mary', 2);
```

2	Mary	

```
INSERT INTO Employee
    VALUES (2, 'Mary', 26000);
```

2	Mary	26000

So Far ...

```
CREATE TABLE Student (
  sID INTEGER PRIMARY KEY,
  sName VARCHAR(50) NOT NULL,
  sAddress VARCHAR(255),
  sYear INT DEFAULT 1
);
```

INSERT

```
INSERT INTO Student (sName, sAddress, sYear)
       VALUES ('Smith', NULL, 2);

INSERT INTO Student  (sID, sName, sAddress, sYear)
       VALUES (1, 'Smith', '5 Arnold Close', 1);

INSERT INTO Student (sName, sAddress)
       VALUES
            ('Smith', '5 Arnold Close'),
            ('Brooks', '7 Holly Ave.');
```

INSERT Example

```
INSERT INTO Student
        (sID, sName, sAddress, sYear)
VALUES
        (1, 'Smith', '5 Arnold Close', 1);
```

Student

sID	sName	sAddress	sYear
1	John	5 Arnold Close	1

```
INSERT INTO Student
        (sName, sAddress, sYear)
      VALUES
        ('Smith', NULL, 2);
```

Student

sID	sName	sAddress	sYear
1	Smith	NULL	2

```
INSERT INTO Student
        (sName, sAddress)
      VALUES
          ('Smith', '5 Arnold Close'),
          ('Brooks', '7 Holly Ave.');
```

Student

sID	sName	sAddress	sYear
1	Smith	5 Arnold Close	1
2	Brooks	7 Holly Ave	1

INSERT

```
INSERT INTO Student
   VALUES
   ('Smith', '5 Arnold Close', 1);
```

→ Error!

```
INSERT INTO Student
   VALUES
   ('Smith', '5 Arnold Close');
```

→ Error!

SELECT

SQL SELECT

- SELECT is the type of query you will use most often.
  - Queries one or more table(s) and returns the result as a table
  - Lots of options, which will be covered over the next few lectures
  - Usually queries can be achieved in a number of ways

Simple SELECT

```
SELECT columns
    FROM table-name;
```

- `columns` can be
  - A single column
  - A comma-separated list of columns
  - * for ‘all columns’

Simple SELECT

```
SELECT * FROM Student;
```

Student			
sID	sName	sAddress	sYear
1	Smith	5 Arnold Close	2
2	Brooks	7 Holly Avenue	2
3	Anderson	15 Main Street	3
4	Evans	Flat 1a, High Street	2
5	Harrison	Newark Hall	1
6	Jones	Southwell Hall	1

Simple SELECT

```
SELECT sName FROM Student;
```

Simple SELECT

```
SELECT sName FROM Student;
```

sName
Smith
Brooks
Anderson
Evans
Harrison
Jones

Simple SELECT

```
SELECT sName, sAddress FROM Student;
```

sName	sAddress
Smith	5 Arnold Close
Brooks	7 Holly Avenue
Anderson	15 Main Street
Evans	Flat 1a, High Street
Harrison	Newark Hall
Jones	Southwell Hall

$$\pi_{\text{sName, sAddress}}(\textbf{Student})$$

Listing Tables

- To list all of your tables using:

```
.tables
```

Update

UPDATE

- Changes values in specified rows based on WHERE conditions

```
UPDATE table-name
    SET col1 = val1
        [,col2 = val2…]
    [WHERE
        condition]
```

- All rows where the condition is true have the columns set to the given values
- If no condition is given all rows are changed so BE CAREFUL
- Values are constants or can be computed from columns

UPDATE

Employee		
ID	**Name**	**Salary**
1	John	25000
2	Mary	26000
3	Mark	18000
4	Anne	22000

```
UPDATE Employee
SET Salary = 15000,
    Name = 'Jane'
WHERE ID = 4;
```

```
UPDATE Employee
SET Salary =
    Salary * 1.05;
```

UPDATE

Consider how "**WHERE**" could be used in a **SELECT** command.

Employee		
ID	**Name**	**Salary**
1	John	25000
2	Mary	26000
3	Mark	18000
4	Anne	22000

```
UPDATE Employee
SET Salary = 15000,
    Name = 'Jane'
WHERE ID = 4;
```

Employee		
ID	**Name**	**Salary**
1	John	25000
2	Mary	26000
3	Mark	18000
4	Jane	15000

```
UPDATE Employee
SET Salary =
    Salary * 1.05;
```

Employee		
ID	**Name**	**Salary**
1	John	26250
2	Mary	27300
3	Mark	18900
4	Anne	23100

DELETE

DELETE

- Removes all rows, or those which satisfy a condition
- If no condition is given then ALL rows are deleted - BE CAREFUL

```
DELETE FROM
    table-name
    [WHERE condition]
```

DELETE

Employee		
ID	**Name**	**Salary**
1	John	25000
2	Mary	26000
3	Mark	18000
4	Anne	22000

```
DELETE FROM
     Employee
WHERE
     Salary > 20000;
```

```
DELETE FROM Employee;
```

DELETE

Employee		
ID	**Name**	**Salary**
1	John	25000
2	Mary	26000
3	Mark	18000
4	Anne	22000

```
DELETE FROM
    Employee
WHERE
    Salary > 20000;
```

Employee		
ID	**Name**	**Salary**
3	Mark	18000

```
DELETE FROM Employee;
```

Employee		
ID	**Name**	**Salary**

Being Careful

- When using DELETE and UPDATE
  - You need to be careful to have the right WHERE clause
  - You can check it by running a SELECT statement with the same WHERE clause first

Before running:

```
DELETE FROM Student
    WHERE sYear = 3;
```

Run:

```
SELECT * FROM Student
    WHERE sYear = 3;
```

ALTER

Altering Columns

- To add a column to a table:

```
ALTER TABLE table-name
    ADD COLUMN col-name
    col-def
```

- Or

```
ALTER TABLE table-name
            ADD COLUMN
col-name
    FIRST | AFTER col2
```

- To remove a column from a table:

```
ALTER TABLE table-name
DROP COLUMN col-name
```

- For Example

```
ALTER TABLE Student DROP
COLUMN sDegree;
```

Changing Tables

- Sometimes you want to change the structure of an existing table
  - One way is to **`DROP`** it then rebuild it
  - This is dangerous, so there is the **`ALTER TABLE`** command instead
- **`ALTER TABLE`** can
  - Add a new column
  - Remove an existing column
  - Add a new constraint
  - Remove an existing constraint
  - Change column name and/or definition

- Note – SQLite has limited support for ALTER operations
- More Info - http://sqlite.org/lang_altertable.html

Altering Columns

- To change a column's name (and definition):

```
ALTER TABLE table-name
CHANGE COLUMN
    col-name
    new-col-name
    col-definition
```

- To change the definition of a column only:

```
ALTER TABLE table-name
    MODIFY COLUMN
        col-name new-col-
        definition
```

Takeaways

1. SQL - Structured Query Language
2. We use MySQL as DBMS
3. Create
    a. Database and Tables
    b. Data types / column definition
    c. Constraints (Primary and Foreign keys)
4. Manipulating tables
    a. DROP TABLE
    b. ALTER TABLE
    c. INSERT, UPDATE, and DELETE
    d. Take Care with these destructive operations
5. Retrieve information
    a. SELECT FROM

Questions?