SQL Data Definition II

Database Systems

Michael Pound

This Lecture

- More SQL
  - DROP TABLE
  - ALTER TABLE
  - INSERT, UPDATE, and DELETE
  - The Information Schema
- Further Reading
  - Database Systems, Connolly and Begg, Chapter 6.3
  - The Manga Guide to Databases, Chapter 4

Labs

- Labs start tomorrow, 9am in A32.
- Exercises will be posted each week on the website:
  www.cs.nott.ac.uk/~mpp/G51DBS/
- I will also post information on setting up MySQL at home
- Exercises and Labs are optional, but practice *will* improve your performance in the assessments

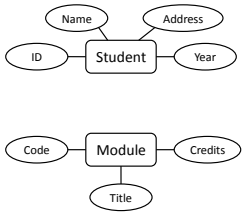
Last Lecture - CREATE TABLE

```
CREATE TABLE <table-name> (
  <col-name 1> <col-def 1>,
  <col-name 2> <col-def 2>,
        :
  <col-name n> <col-def n>,
  <constraint-1>,
        :
  <constraint-k>
);
```

Last Lecture

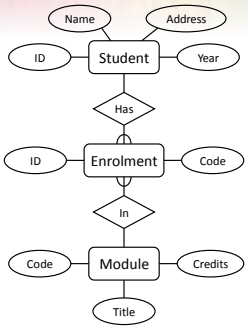
```
CREATE TABLE Student (
  sID INT AUTO_INCREMENT
   PRIMARY KEY,
  sName VARCHAR(50) NOT NULL,
  sAddress VARCHAR(255),
  sYear INT DEFAULT 1
) ENGINE=InnoDB;

CREATE TABLE Module (
  mCode CHAR(6) NOT NULL,
  mCredits TINYINT NOT NULL
   DEFAULT 10,
  mTitle VARCHAR(100) NOT
   NULL,
  CONSTRAINT pk_mod
  PRIMARY KEY (mCode)
) ENGINE=InnoDB;
```



Last Lecture

```
CREATE TABLE Enrolment (
  sID INT NOT NULL,
  mCode CHAR(6) NOT NULL,
  CONSTRAINT en_pk
    PRIMARY KEY (sID, mCode)
  CONSTRAINT en_fk1
    FOREIGN KEY (sID)
    REFERENCES Student (sID)
    ON UPDATE CASCADE
  CONSTRAINT en_fk2
    FOREIGN KEY (mCode)
    REFERENCES Module (mCode)
    ON UPDATE CASCADE
) ENGINE=InnoDB;
```



Deleting Tables

- You can delete tables with the DROP keyword

```
DROP TABLE
 [IF EXISTS]
 table-name;
```

- For example:

```
DROP TABLE Module;
```

- Be ***extremely careful*** using any SQL statement with DROP in it.
  - All rows in the table will also be deleted
  - You won't normally be asked to confirm
  - Undoing a DROP is difficult, sometimes impossible

Deleting Tables

- You can delete multiple tables in a list:

```
DROP TABLE
 IF EXISTS
 Module, Student;
```

- Foreign Key constraints will prevent DROPS under the default RESTRICT option
  - To overcome this, either remove the constraint or drop the tables in the correct order (referencing table first)

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

Altering Columns

- To add a column to a table:

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

OR

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

- To remove a column from a table:

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

- For example:

```
ALTER TABLE Student
 ADD COLUMN sDegree
 VARCHAR(64) NOT NULL;

ALTER TABLE Student
 DROP COLUMN sDegree;
```

Altering Columns

- To change a column's name and/or definition:

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

- To change the definition of a column only:

```
ALTER TABLE <table>
 MODIFY COLUMN
 <col-name>
 <col-definition>
```

Note: Changing the type of a column might have unexpected results. Be careful that the type conversion taking place is appropriate. E.g. INT → VARCHAR is ok, VARCHAR → INT is problematic.

Altering Columns

- To add a constraint:

```
ALTER TABLE <table>
 ADD CONSTRAINT
  <name>
  <definition>
```

- To remove a constraint:

```
ALTER TABLE <table>
 DROP CONSTRAINT
  <name>
```

- To remove a unique key constraint:

```
ALTER TABLE <table>
 DROP INDEX
  <name>
```

- For example:

```
ALTER TABLE Module
 ADD CONSTRAINT
  un_title UNIQUE
  (mTitle)
```

Example

```
CREATE TABLE Module (
  mCode CHAR(6) NOT NULL,
  mCredits TINYINT NOT NULL
   DEFAULT 10,
  mTitle VARCHAR(100) NOT NULL
);
```

What are the SQL command(s) to add a column lecID to the Module table? Followed by a foreign key constraint to reference the lecID column in a Lecturer table?

Module

mCode	mCredits	mTitle
G51DBS	10	Database Systems
G51PRG	20	Programming
G51IAI	10	Artificial Intelligence
G52ADS	10	Algorithms

Example

To add a lecID column:

```
ALTER TABLE Module
  ADD COLUMN lecID INT NULL;
```

Module

mCode	mCredits	mTitle	lecID
G51DBS	10	Database Systems	NULL
G51PRG	20	Programming	NULL
G51IAI	10	Artificial Intelligence	NULL
G52ADS	10	Algorithms	NULL

Example

To create a Foreign Key:

```
ALTER TABLE Module
  ADD CONSTRAINT fk_mod_lec
  FOREIGN KEY (lecID) REFERENCES Lecturer (lecID);
```

Module

mCode	mCredits	mTitle	lecID
G51DBS	10	Database Systems	NULL
G51PRG	20	Programming	NULL
G51IAI	10	Artificial Intelligence	NULL
G52ADS	10	Algorithms	NULL

INSERT, UPDATE, DELETE

- **INSERT** - add a row to a table
- **UPDATE** - change row(s) in a table
- **DELETE** - remove row(s) from a table
- **UPDATE** and **DELETE** should make use of '**WHERE** clauses' to specify which rows to change or remove
- ***BE CAREFUL*** with these - an incorrect or absent **WHERE** clause can destroy lots of data

INSERT

- Inserts rows into the database with the specified values

```
INSERT INTO
  <table>
  (col1, col2, …)
  VALUES
  (val1, val2, …);
```

- The number of columns and 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 values, be careful of the ordering

INSERT

Employee

ID	Name	Salary
1	John	25000

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

Employee

ID	Name	Salary
1	John	25000
2	Mary	26000

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

Employee

ID	Name	Salary
1	John	25000
2	Mary	

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

Employee

ID	Name	Salary
1	John	25000
2	Mary	26000

INSERT

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

Student

sID	sName	sAddress	sYear
1	Smith	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
 ('John', '5 Arnold Close'),
 ('Brooks', '7 Holly Ave.');
```

Student

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

UPDATE

- Changes values in specified rows based on SET conditions

```
UPDATE <table>
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
```

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

- Removes all rows, or those which satisfy a condition

```
DELETE FROM
 <table>
 [WHERE
  <condition>]
```

- If no condition is given then ALL rows are deleted - BE CAREFUL
- You might also use **TRUNCATE TABLE** which is like **DELETE FROM** without a **WHERE** but is often quicker

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
4	Jane	15000

```
DELETE FROM Employee;
          Or
TRUNCATE TABLE
  Employee;
```

Employee

ID	Name	Salary

SQL SELECT

- SELECT is the type of query you will use most often.
  - Queries one or more tables 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>;
```

<columns> can be

- A single column
- A comma-separated list of columns
- ***** for 'all columns'

Sample SELECTs

```
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

Sample SELECTs

```
SELECT sName FROM Student;
```

Student

sName
Smith
Brooks
Anderson
Evans
Harrison
Jones

Sample SELECTs

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

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

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 Year = 3;
```

run

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

Information Schema

- SQL '92 defines a set of system views that can be used to access metadata
- Metadata is 'data about data'. In this case, data about all of your tables
- This system database is called the information_schema
- Lots of DBMSs support this, but as usual support varies
- MySQL also gives us a few custom commands as well:
  - SHOW – Shows information on tables
  - DESCRIBE – Shows information on the columns in a specific table
- These are fine, but remember SHOW is MySQL specific, so don't become too reliant on it

Listing Tables

- To list all of your tables using SHOW:

```
SHOW tables;
```

- To use the information_schema:

```
SELECT TABLE_NAME
 FROM information_schema.tables
WHERE TABLE_SCHEMA = 'mpp';
```

More Information

- The information_schema has other columns that might be of use:

```
SELECT TABLE_NAME, TABLE_TYPE,
  TABLE_ROWS, ENGINE
 FROM information_schema.tables
 WHERE TABLE_SCHEMA = 'mpp';
```

- TABLE_TYPE – Table or view
- TABLE_ROWS – Current row count
- ENGINE – What storage engine the table uses

Showing Columns

- You can describe a table using:

```
DESCRIBE <table-name>;
```

```
mysql> DESCRIBE Student;
+----------+--------------+------+-----+---------+-------+
| Field    | Type         | Null | Key | Default | Extra |
+----------+--------------+------+-----+---------+-------+
| sID      | int(11)      | NO   | PRI | NULL    |       |
| sName    | varchar(64)  | NO   |     | NULL    |       |
| sAddress | varchar(255) | YES  |     | NULL    |       |
| sYear    | tinyint(4)   | NO   |     | 1       |       |
+----------+--------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

Showing More Detail

- You can show a little more information, including constraints with:

```
SHOW CREATE TABLE <table-name>;
```

```
mysql> show create table Student;
| Student | CREATE TABLE `Student` (
  `sID` int(11) NOT NULL,
  `sName` varchar(64) COLLATE utf8_unicode_ci NOT NULL,
  `sAddress` varchar(255) COLLATE utf8_unicode_ci DEFAULT
NULL,
  `sYear` tinyint(4) NOT NULL DEFAULT '1',
  PRIMARY KEY (`sID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci
1 row in set (0.00 sec)
```

Next Lecture

- SQL SELECT
  - WHERE Clauses
  - SELECT from multiple tables
  - JOINs
- Further reading
  - Database Systems, Connolly and Begg, Chapter 6
  - The Manga Guide to Databases, Chapter 4