SQL Data Definition

Database Systems

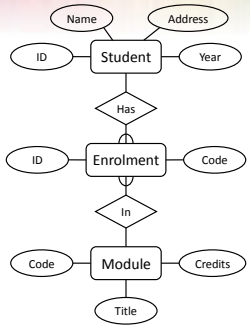
Michael Pound

This Lecture

- SQL
  - The SQL language
  - SQL, the relational model, and E/R diagrams
  - CREATE TABLE
    - Columns
    - Primary Keys
    - Foreign Keys
- Further Reading
  - Database Systems, Connolly & Begg, Chapter 7.3
  - The Manga Guide to Databases, Chapter 4

Last Lecture

- Entity Relationship Diagrams
  - Entities
  - Attributes
  - Relationships
- Example
  - Students take many Modules
  - Modules will be taken by many Students



SQL

- Originally 'Sequel' - Structured English query Language, part of an IBM project in the 70's
- Sequel was already taken, so it became SQL - Structured Query Language
- ANSI Standards and a number of revisions
  - SQL-89
  - SQL-92 (SQL2)
  - SQL-99 (SQL3)
  - ...
  - SQL:2008 (SQL 2008)
- Most modern DBMS use a variety of SQL
  - Few (if any) are true to the standard

SQL

- SQL is a language based on the relational model
  - Actual implementation is provided by a DBMS
- SQL is everywhere
  - Most companies use it for data storage
  - All of us use it dozens of times per day
  - You will be expected to know it as a software developer
- SQL provides
  - A Data Definition Language (DDL)
  - A Data Manipulation Language (DML)
  - A Data Control Language (DCL)

Database Management Systems

- A DBMS is a software system responsible for allowing users access to data
- A DBMS will usually
  - Allow the user to access data using SQL
  - Allow connections from other programming languages
  - Provide additional functionality like concurrency
- There are many DBMSs, some popular ones include:
  - Oracle
  - DB2
  - Microsoft SQL Server
  - Ingres
  - PostgreSQL
  - MySQL
  - Microsoft Access (with SQL Server as storage engine)

MySQL

- During this module we will use MySQL as our DBMS
  - Free to use
  - Source code available under General Public License
  - Extremely popular and widely used
  - Easy to set up on the school servers
  - In most cases is as functional as commercial DBMSs
- The school also has Access, Oracle and PostgreSQL installed.

SQL Case

- SQL statements will be written in **`BOLD COURIER FONT`**
- SQL keywords are not case-sensitive, but we'll write SQL keywords in upper case for emphasis
- Table names, column names etc. are case sensitive
- For example:

```
SELECT * FROM Students
    WHERE Name = "James";
```

Important: MySQL in Windows is not case sensitive. Do not be complacent during the coursework.

SQL Strings

- Strings in SQL are surrounded by single quotes:
  - `'I AM A STRING'`
- Single quotes within a string are doubled or escaped using \
  - `'I''M A STRING'`
  - `'I\'M A STRING'`
- `''` is an empty string
- In MySQL, double quotes also work (this isn't the ANSI standard)

Non-Procedural Programming

- SQL is a declarative (non-procedural) language
  - Procedural – tell the computer what to do using specific successive instructions
  - Non-procedural – describe the required result (not the way to compute it)
- Example: Given a database with tables
  - Student with attributes ID, Name, Address
  - Module with attributes Code, Title
  - Enrolment with attributes ID, Code
- Get a list of students who take the module 'Database Systems'

Procedural Programming

```
Set M to be the first Module Record              /* Find module code for    */
Code = ''                                        /* 'Database Systems'      */
While (M is not null) and (Code = '')
   If (M.Title = 'Database Systems') Then
      Code = M.Code
   Set M to be the next Module Record
Set NAMES to be empty                            /* A list of student names */
Set S to be the first Student Record
While S is not null                              /* For each student...     */
   Set E to be the first Enrolment Record
    While E is not null                          /* For each enrolment...   */
      If (E.ID = S.ID) And                       /* If this student is      */
         (E.Code = Code) Then                    /* enrolled in DB Systems  */
         NAMES = NAMES + S.NAME                  /* add them to the list    */
      Set E to be the next Enrolment Record
   Set S to be the next Student Record
Return NAMES
```

Non-Procedural (SQL)

```
SELECT Name FROM Student, Enrolment
  WHERE
  (Student.ID = Enrolment.ID)
  AND
  (Enrolment.Code =
    (SELECT Code FROM Module WHERE
       Title = 'Database Systems'));
```

NoSQL

- SQL is by no means perfect
  - Edgar Codd hated it – It's actually a pretty poor implementation of the relational model
  - Implementations vary wildly. For example, while Oracle and MySQL both use SQL, there are commands that won't work on both systems.
  - It's extremely easy to trigger vast joins or delete large numbers of rows by mistake
- NoSQL is a term used to describe database systems that attempt to avoid SQL and the relational model

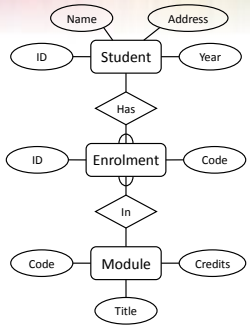
Relations, Entities and Tables

- The terminology changes from the Relational Model through to SQL, but usually means the same thing

Relations	E/R Diagrams	SQL
Relation	Entity	Table
Tuple	Instance	Row
Attribute	Attribute	Column or Field
Foreign Key	M:1 Relationship	Foreign Key
Primary Key	Attribute	Primary Key

Implementing E/R Diagrams

- Given an E/R design
  - The entities become SQL tables
  - Attributes of an entity become columns in the corresponding table
  - We can approximate the domains of the attributes by assigning types to each column
  - Relationships may be represented by foreign keys



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>
);
```

- You supply
  - A name for the table
  - A name and definition for each column
  - A list of constraints (e.g. Keys)

Column Definitions

```
<col-name> <type>
  [NULL | NOT NULL]
  [DEFAULT default_value]
  [NOT NULL | NULL]
  [AUTO_INCREMENT]
  [UNIQUE [KEY] |
  [PRIMARY] KEY]
```

([] *optional*, | *or*)

- Each column has a name and a type
- Most of the rest of the column definition is optional
- There's more you can add, like storage and index instructions

Types

- There are many types in MySQL, but most are variations of the standard types
- Numeric Types
  - TINYINT, SMALLINT, INT, MEDIUMINT, BIGINT
  - FLOAT, REAL, DOUBLE, DECIMAL
- Dates and Times
  - DATE, TIME, YEAR
- Strings
  - CHAR, VARCHAR
- Others
  - ENUM, BLOB

Types

- We will use a small subset of the possible types:

Type	Description	Example
TINYINT	8 bit integer	-128 to 127
INT	32 bit integer	2147483648 to 2147483647
CHAR (m)	String of fixed length m	"Hello World "
VARCHAR (m)	String of maximum length m	"Hello World"
REAL	A double precision number	3.14159
ENUM	A set of specific strings	('Cat', 'Dog', 'Mouse')
DATE	A Day, Month and Year	'1981-12-16' or '81-12-16'

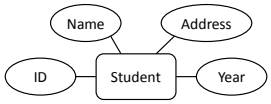
Column Definitions

- Columns can be specified as **NULL** or **NOT NULL**
- **NOT NULL** columns cannot have missing values
- **NULL** is the default if you do not specify either
- Columns can be given a default value
- You just use the keyword DEFAULT followed by the value, eg:

```
col-name INT DEFAULT 0,
```

Example

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



AUTO_INCREMENT

- If you specify a column as **AUTO_INCREMENT**, a value (usually max(col) + 1) is automatically inserted when data is added. This is useful for Primary Keys
- For example:

```
col-name INT AUTO_INCREMENT,
```

- When it comes to inserting values, you should use NULL, 0 or nothing to ensure you don't override the automatic value

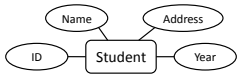
Note: The table auto_increment value isn't recalculated during deletes. You might want to reset it using:

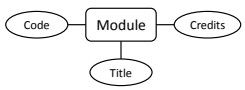
```
ALTER TABLE <name> AUTO_INCREMENT=1;
```

Example

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

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





Constraints

```
CONSTRAINT
 <name>
 <type>
 <details>
```

- MySQL Constraints
  - **PRIMARY KEY**
  - **UNIQUE**
  - **FOREIGN KEY**
  - **INDEX**
- Each constraint is given a name. If you don't specify a name, one will be generated
- Constraints which refer to single columns can be included in their definition

Primary Keys

- A primary key for each table is defined through a constraint
- **`PRIMARY KEY`** also automatically adds **`UNIQUE`** and **`NOT NULL`** to the relevant column definition
- The details for the Primary Key constraint are the set of relevant columns

```
CONSTRAINT <name>
  PRIMARY KEY
  (col1, col2, …)
```

Unique Constraints

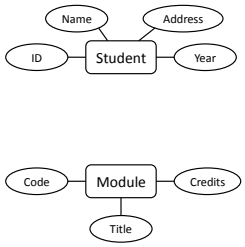
- As well as a single primary key, any set of columns can be specified as **`UNIQUE`**
- This has the effect of making candidate keys in the table
- The details for a unique constraint are a list of columns which make up the candidate key

```
CONSTRAINT <name>
  UNIQUE
  (col1, col2, …)
```

Example

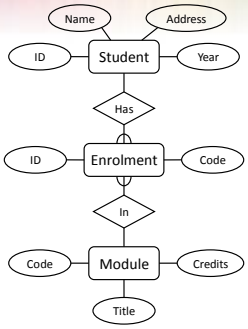
```
CREATE TABLE Student (
  sID INT AUTO_INCREMENT
   PRIMARY KEY,
  sName VARCHAR(50) NOT NULL,
  sAddress VARCHAR(255),
  sYear INT DEFAULT 1
);

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



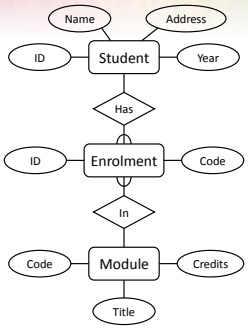
Relationships

- Relationships are represented in SQL using Foreign Keys
  - 1:1 are usually not used, or can be treated as a special case of M:1
  - M:1 are represented as a foreign key from the M-side to the 1
  - M:M are split into two M:1 relationships



Relationships

- The Enrolment table
  - Will have columns for the student ID and module code attributes
  - Will have a foreign key to Student for the 'has' relationship
  - Will have a foreign key to Module for the 'in' relationship



Foreign Keys

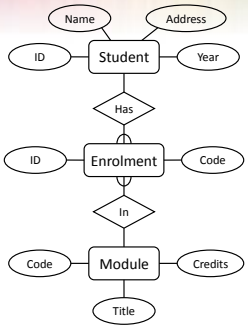
- Foreign Keys are also defined as constraints
- You need to provide
  - The columns which make up the foreign key
  - The referenced table
  - The columns which are referenced by the foreign key
- You can optionally provide reference options

```
CONSTRAINT <name>
 FOREIGN KEY
  (col1, col2, ...)
 REFERENCES
  table-name
  (col1, col2, ...)
 ON UPDATE ref_opt
 ON DELETE ref_opt

ref_opt: RESTRICT |
  CASCADE | SET NULL
```

Example

```
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
);
```



Storage Engines

- In MySQL you can specify the engine used to store files onto disk
- The type of storage engine will have a large effect on the operation of the database
- The engine should be specified when a table is created
- Some available storage engines are:
  - **MyISAM** – The default, very fast. Ignores all foreign key constraints
  - **InnoDB** – Offers transactions and foreign keys
  - **Memory** – Stored in RAM (extremely fast)
  - **Blackhole** – Deletes everything you put in it!

InnoDB

- We will use InnoDB for all tables during this module, for example:

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

Note: All tables in a relationship must be InnoDB for FK constraints to work

This Lecture in Exams

Give the SQL statement(s) required to create a table called Books with the following columns

- bID, an integer that will be the Primary Key
- bTitle, a string of maximum length 64
- bPrice, a double precision value
- gCode, an integer that will be a foreign key to a gCode column in another table Genres

Next Lecture

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