SQL 1: Create and Drop

Databases and Interfaces

Matthew Pike

This Lecture

- Introduction to SQL
  - What is SQL?
  - How to:
    - CREATE tables
    - DROP (delete) tables

What is SQL?

SQL

- SQL is a language based on the relational model
  - An international (ANSI) standard language
- DBMS implements an interface between SQL and our data tables
  - SQLite and MySQL are examples of a DBMS
- Important to note – not all DBMS implementations are equal
  - There are varying degrees of support for SQL

Database Management Systems (DBMS)

- 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
  - SQLite

Provided Languages

- Data Definition Language (DDL)
  - Specify database format
- Data Manipulation Language (DML)
  - Specify and retrieve database contents
- Data Control Language (DCL)
  - Specify access controls (privileges)
- Which are often all one piece of software
  - E.g. SQLite, MySQL, Postgres

Relations, Entities and Tables

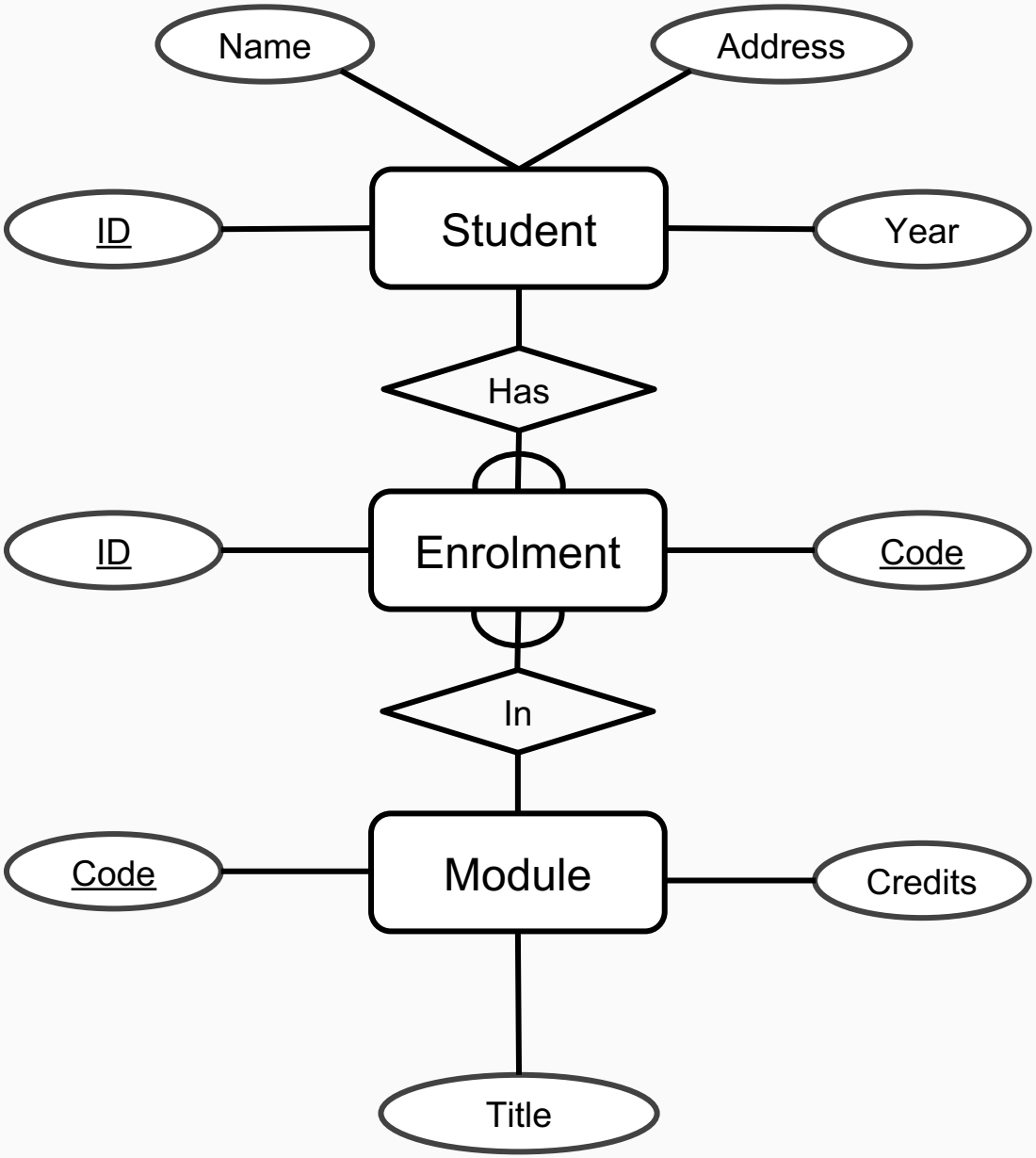
- The terminology changes from the Relational Model through to SQL, but usually means the same thing

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

CREATE

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
  - Note the notation for primary keys in ER



Create Table Definitions

```
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
);
```

- A table is associated to a database
- You supply
  - A name for the table
  - A name and definition / type for each column
  - A list of constraints (e.g. Keys)

Column Definitions

```
col-name col-def

    [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

CREATE Example (Incomplete)

```
CREATE TABLE Student (
    sID INTEGER ...,
    sName VARCHAR(50)...,
    sAddress VARCHAR(255),
    sYear INTEGER ...
);
```

- We have specified the table name (Student)
- And four columns and their types
  - ID
  - Name
  - Address
  - Year of Study

Types

- There are many types in SQL, but most are variations of the standard types:
  - Numeric
    - TINYINT, SMALLINT, INT, MEDIUMINT, BIGINT
    - FLOAT, REAL, DOUBLE, DECIMAL
  - Dates and Times
    - DATE, TIME, YEAR
  - Strings
    - CHAR, VARCHAR
- **Important -** Not all data types are supported by every relational database vendors.

Types in SQLite

- SQLite flexible and forgiving with regard to datatypes
  - Table columns can be created that have no specified datatype at all! (not recommended)
  - Other DBMS are “Rigidly-Typed” and enforce strict data-typing
- More info - https://www.sqlite.org/datatype3.html
- SQLite 3 defines 5 “affinities” which each column’s data type will be assigned:
  - TEXT
  - NUMERIC
  - INTEGER
  - REAL
  - BLOB

Example of Types

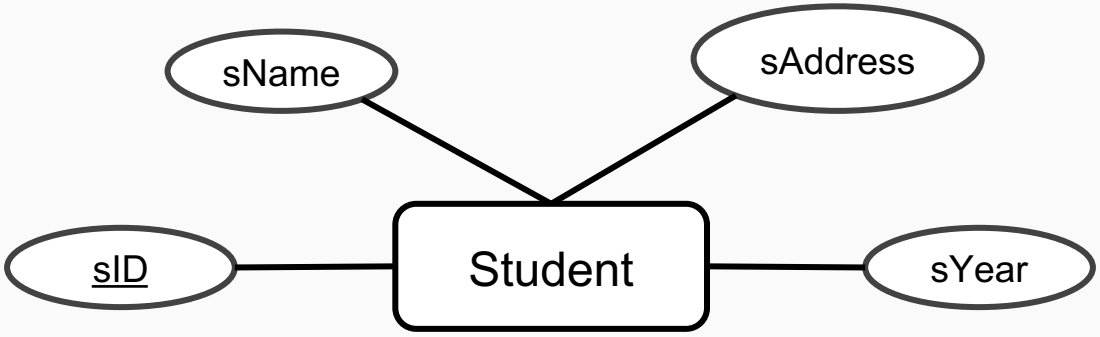
Type	**Description**	**Example**
INTEGER or INT	32 bit integer	25 or -25 or 50 or -50 etc
CHAR (m)	String of fixed length m (Length not enforced in SQLite)	CHAR(1) – ‘A’ CHAR(2) – ‘AB’
VARCHAR (m) TEXT	String of maximum length m (Length not enforced in SQLite)	”Hello World”
REAL	A double precision number	3.14159
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,
- e.g.:
  - `col-name INT DEFAULT 0`

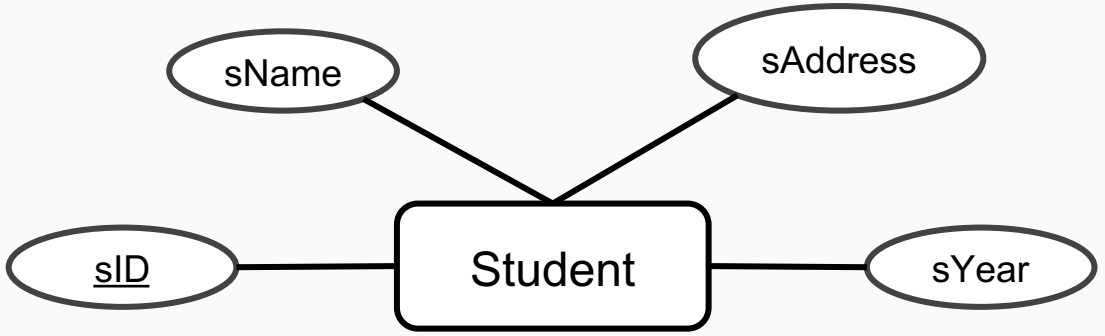
Worked Example

Write the SQL statement to create a table for **Student** with the **attributes** listed below, where the **sID** number and the Student name **cannot be null** and, if not otherwise specified, **students are in Year 1**.



Example

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



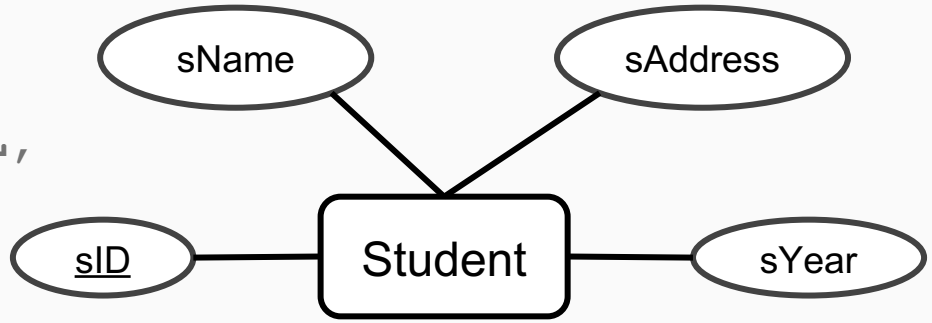
AUTOINCREMENT

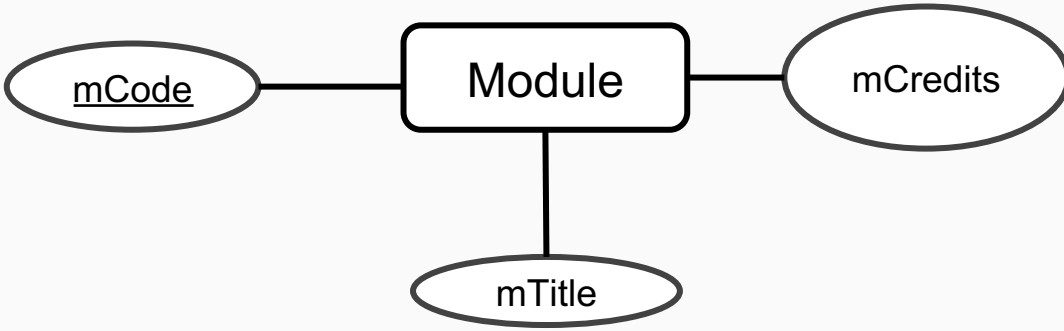
- If you specify a column as **`AUTOINCREMENT`**, a value (usually max(col) + 1) is automatically inserted when data is added. This is useful for Primary Keys
- For example:
  - **`col-name INTEGER AUTOINCREMENT;`**
- When it comes to inserting values, you should use NULL, 0 or nothing to ensure you don't override the automatic value
- **`AUTOINCREMENT`** AUTOINC is common in other DBMS' but is not recommended in SQLite
  - More Info – https://sqlite.org/autoinc.html

Example

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

CREATE TABLE Module (
    ...
);
```



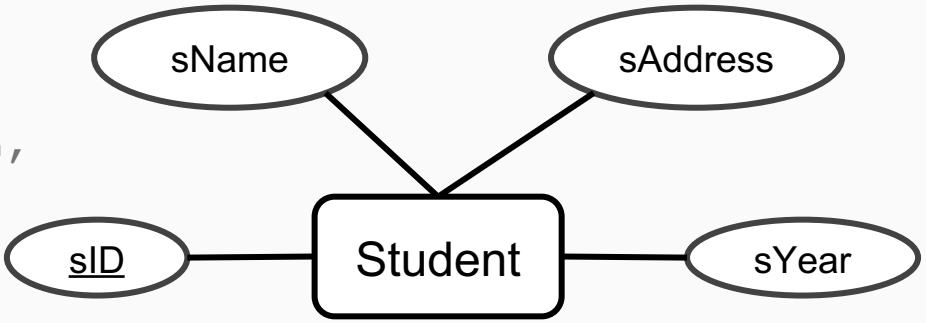


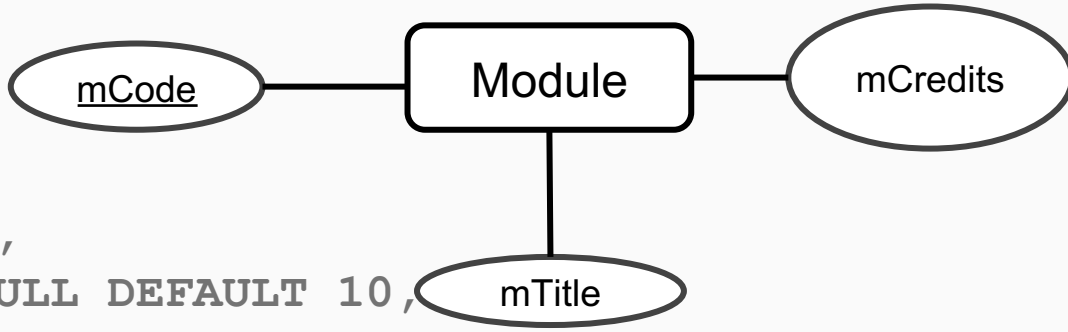
Tips:

- Every module has a 6 characters code (e.g. AE1DBI)
- Every module usually gives 10 credits

Example

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






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

Helpful SQLite3 Special commands (dot-commands)

- The Command-Line Interface (CLI) provides special commands to alter the format of the output from the DB
- For a listing of the available dot commands, you can enter "`.help`"
- More info - https://sqlite.org/cli.html

- Useful dot-commands:
  - `.import FILE TABLE`
    - Import data from FILE into TABLE
  - `.read FILE`
    - Read input from FILE
  - `.schema`
    - Show the CREATE statements
  - `.tables`
    - List names of tables

Constraints

- `CONSTRAINT`
  - `name`
  - `type`
  - `details`

- Example 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` will typically add `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, …)
```

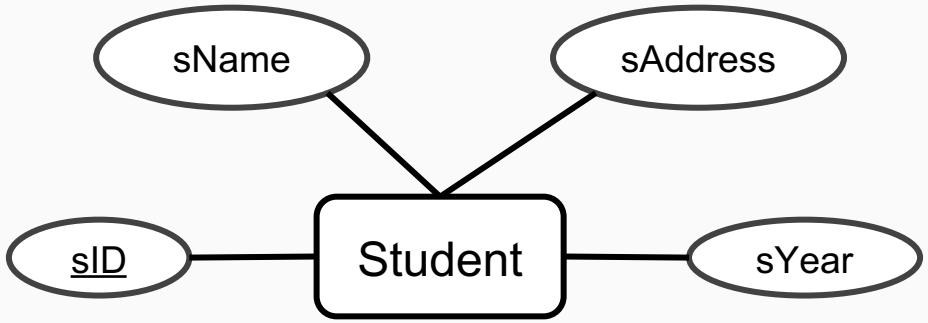
Unique Constraints / CKs

- 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 (CK)

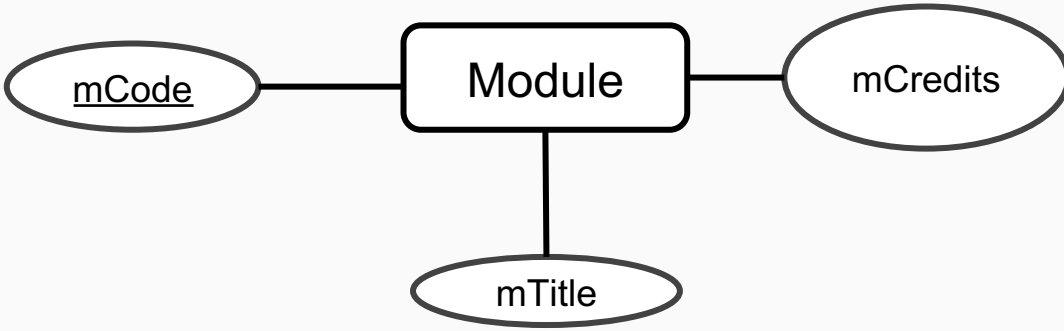
```
CONSTRAINT name
    UNIQUE
    (col1, col2, …)
```

Example

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

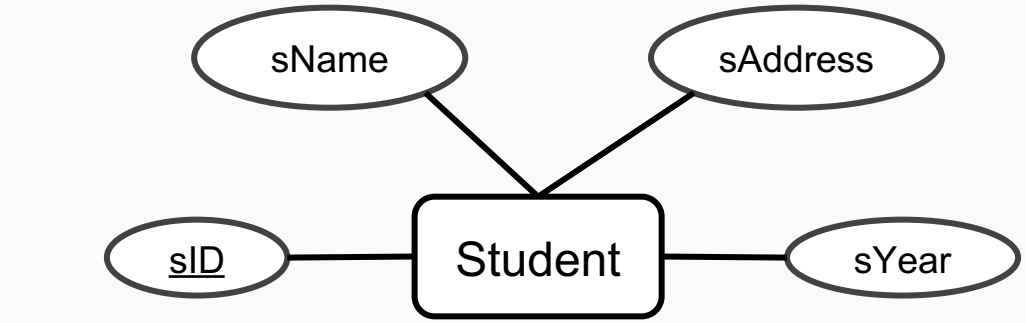


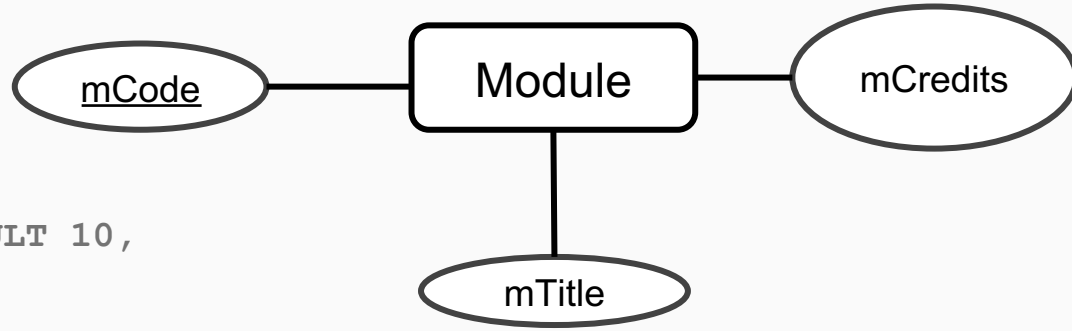

```
CREATE TABLE Module (
    mCode CHAR(6) NOT NULL,
    mCredits TINYINT NOT NULL
                DEFAULT 10,
    mTitle VARCHAR(100) NOT NULL,
    ... ADD PRIMARY KEY
);
```



Example

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

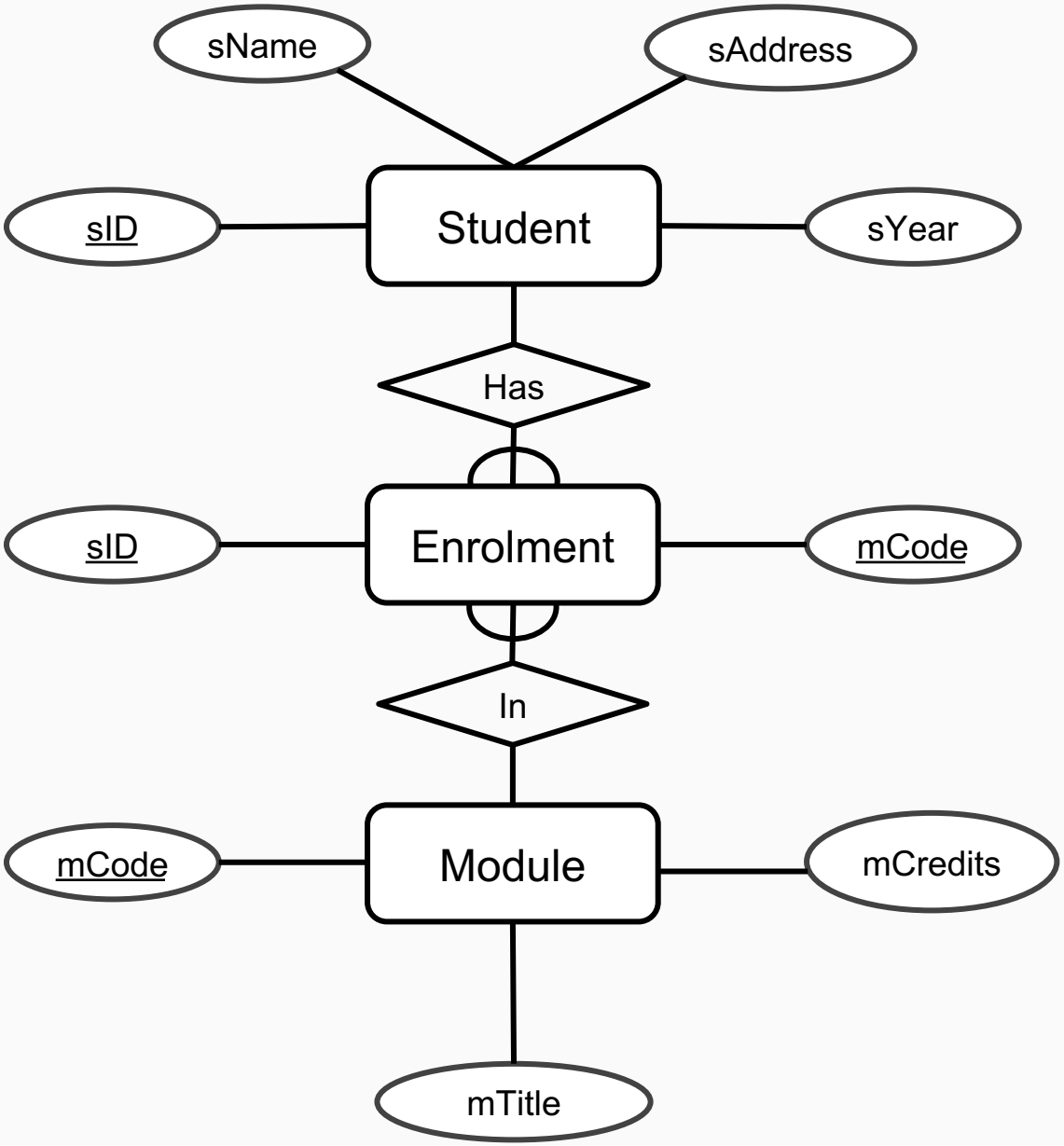





```
CREATE TABLE Module (
    mCode CHAR(6) NOT NULL,
    mCredits TINYINT NOT NULL DEFAULT 10,
    mTitle VARCHAR(100) NOT NULL,
    CONSTRAINT mod_pk
        PRIMARY KEY (mCode)
);
```

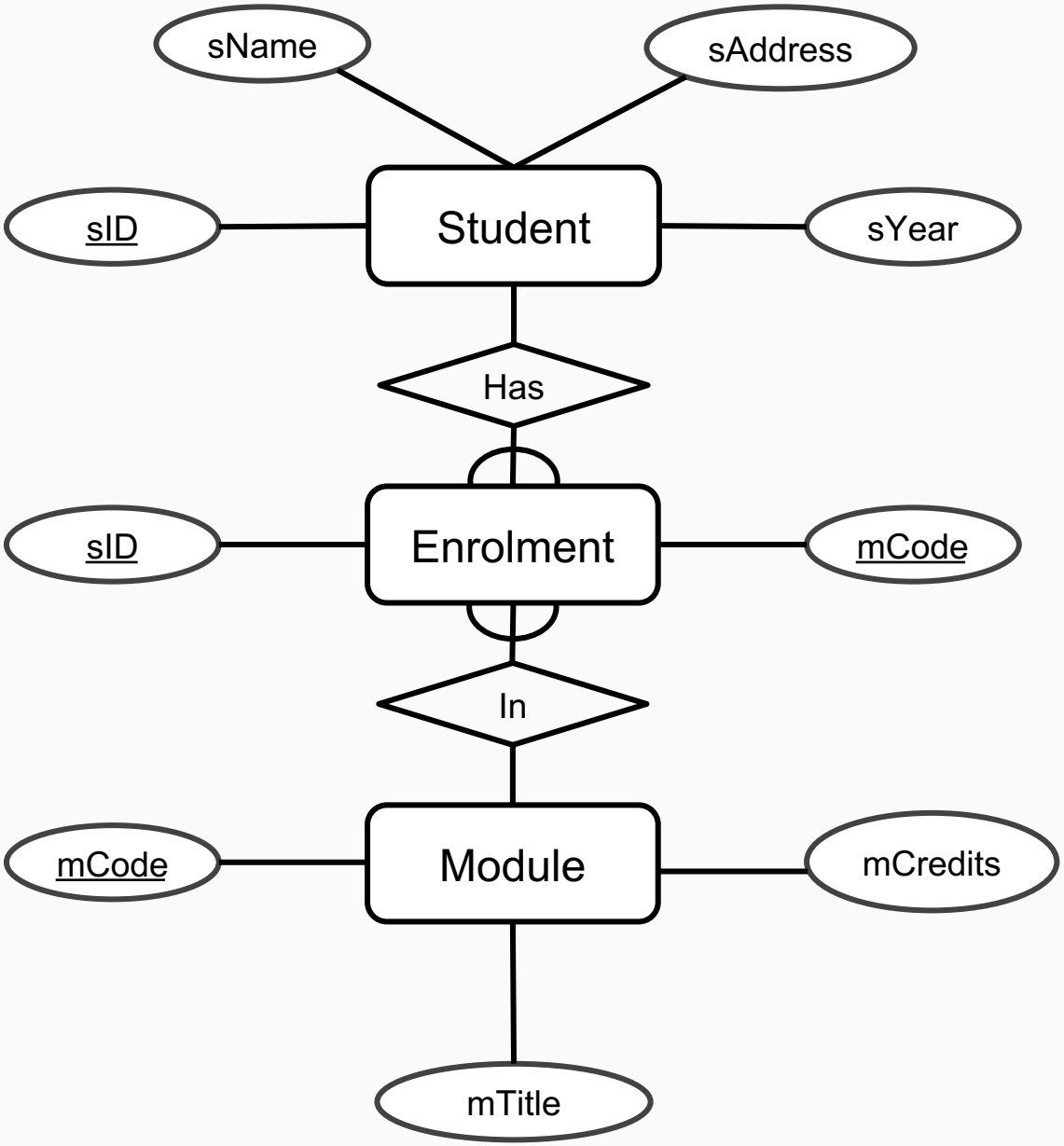
Relationships

- 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 | SET
    DEFAULT
```

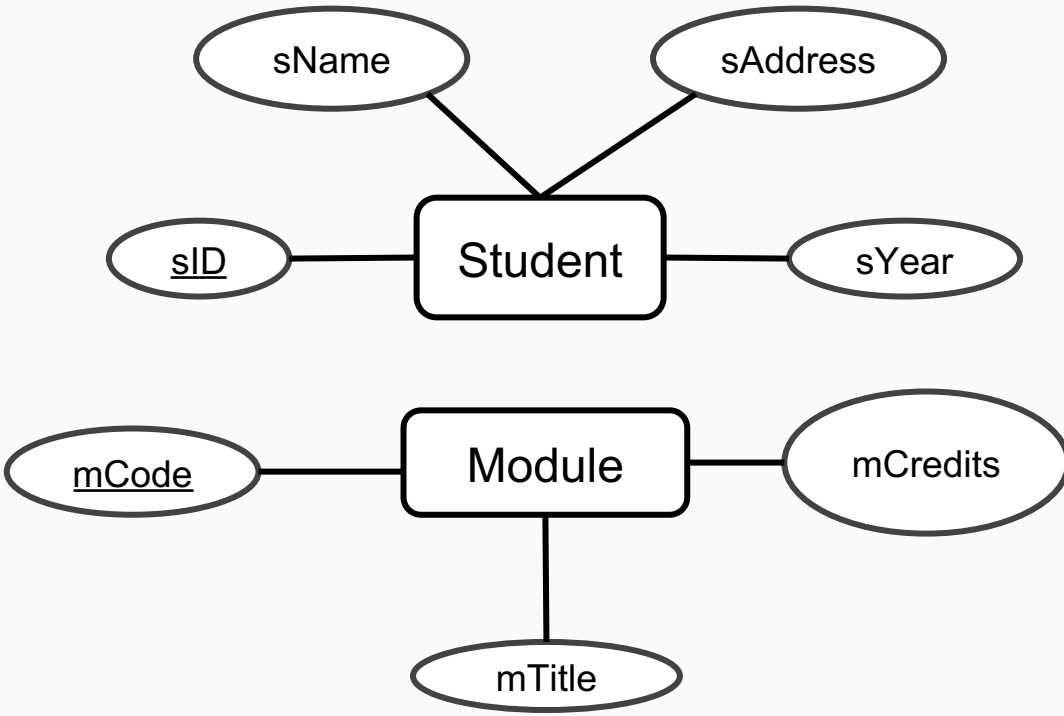
Set Default (Column Definition)

- If you have defined a **`DEFAULT`** value you can use it with referential integrity
- When relations are updated, referential integrity might be violated
- This usually occurs when a referenced tuple is updated or deleted

- There are a number of options when this occurs:
  - **RESTRICT** – stop the user from doing it
  - **CASCADE** – let the changes flow on
  - **SET NULL** – make referencing values null
  - **SET DEFAULT** – make referencing values the default for their column

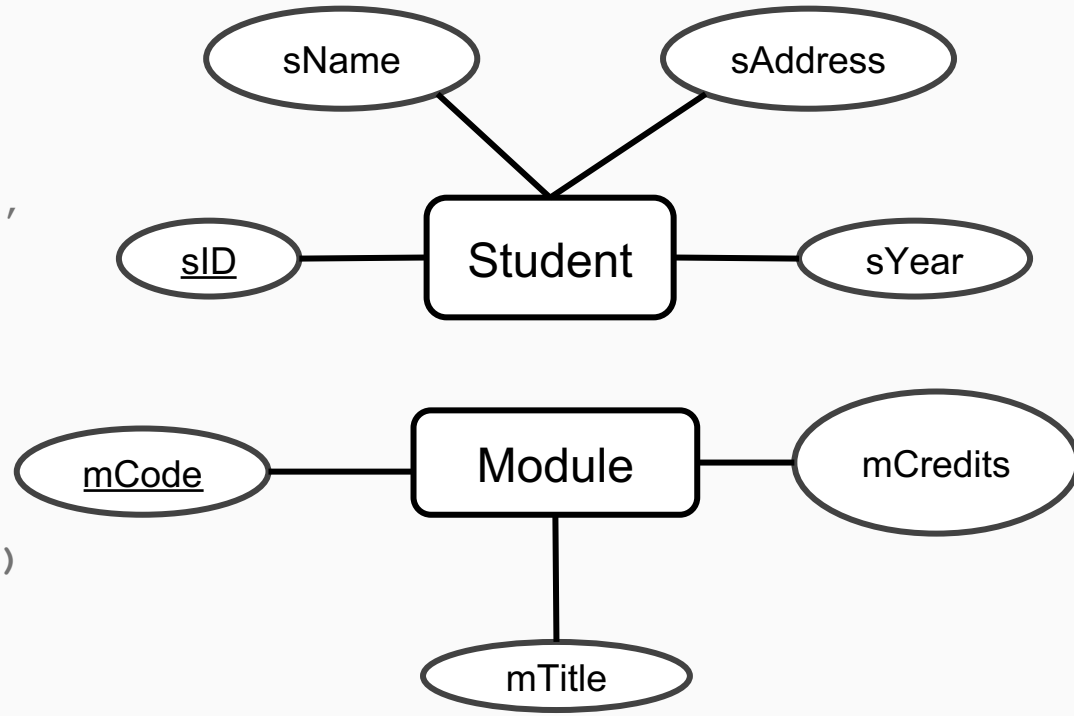
Example

```
CREATE TABLE Enrolment (
     sID INT NOT NULL,
     mCode CHAR(6) NOT NULL,
     ... ADD PRIMARY KEY
     ... AND 2 FOREIGN KEYS
);
```



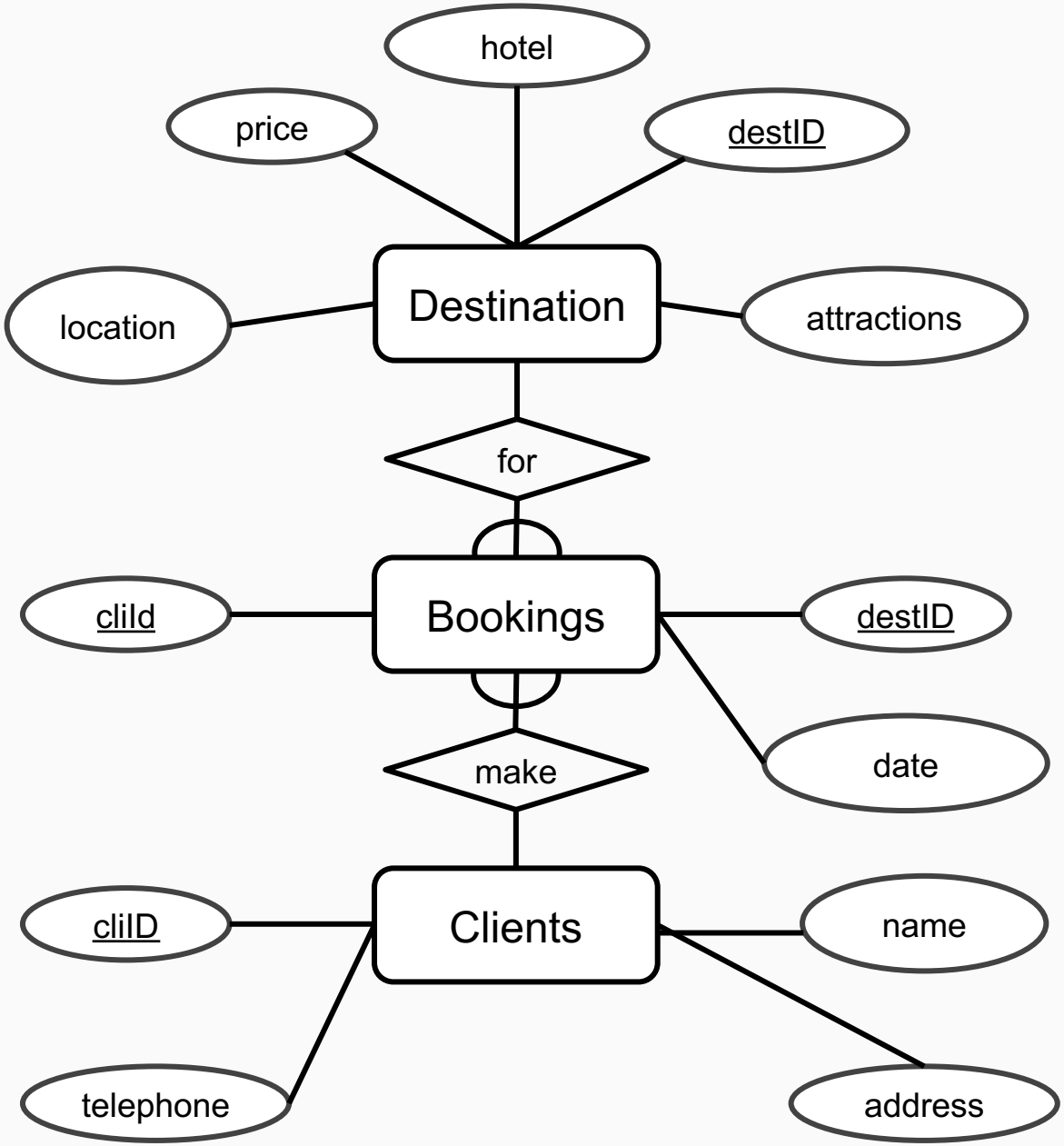
Example

```
CREATE TABLE Enrolment (
     sID INTEGER 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
          ON DELETE CASCADE,
     CONSTRAINT en_fk2
          FOREIGN KEY (mCode)
          REFERENCES Module (mCode)
          ON UPDATE CASCADE
          ON DELETE RESTRICT
);
```



Exercise

- Create table in SQLite from the E/R diagram on the right by identifying the:
  - Name of the tables
  - The columns (inc. data types and attributes) for each table
  - Each table's constraints



Solutions (1)

```
CREATE TABLE Clients(
    cliID INTEGER PRIMARY KEY,
    cliName VARCHAR(255) NOT NULL,
    cliAddress VARCHAR(255),
    cliTel INTEGER
);

CREATE TABLE Destination(
    destID INTEGER PRIMARY KEY,
    destLocation VARCHAR(255),
    destPrice REAL,
    destHotel VARCHAR(255),
    destAttractions VARCHAR(255)
);
```

Solutions (2)

```
CREATE TABLE Bookings(
    cliID INTEGER NOT NULL,
    destID INTEGER NOT NULL,
    bookDate DATE,
    CONSTRAINT book_pk
        PRIMARY KEY(cliID,destID, bookDate),
    CONSTRAINT book_fk1
        FOREIGN KEY (cliID)
            REFERENCES Clients (cliID)
        ON UPDATE CASCADE
        ON DELETE CASCADE,
    CONSTRAINT book_fk2
        FOREIGN KEY (destID)
            REFERENCES Destination (destID)
        ON UPDATE CASCADE
        ON DELETE CASCADE
);
```

DROP

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
    - Assume it is not possible when performing the operation

Deleting Tables

- 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)

Takeaways

1. SQL - Structured Query Language
2. We use SQLite as our DBMS
3. CREATE
    a. Database and Tables
    b. Data types / column definition
    c. Constraints (Primary and Foreign keys)
4. DROP
    a. Removes tables from the DB

Questions?