Creating, Populating and Using Database Tables

Up until now we have been querying existing tables. However, somebody has to create the tables in the first place.

Creating Tables…

We may create new database tables by using a CREATE TABLE statement. This statement requires the following information:

- The name of the new table (which should differ from any existing table in the current database).
- The name and type of each of the table's attributes. Although the names of the attributes should all differ, several attributes may share the same type.

There are a number of types that may be used in Access including:

• TEXT	this indicates that the attribute value must be a character string
• INT	this indicates that the attribute value must be an integer
• DOUBLE	this indicates that the attribute value must be a real number
• DATE	this indicates that the attribute value must be a date

Here is an example CREATE TABLE statement.

```
CREATE TABLE Modules

(ModuleID  text,

ModuleLeader text,

ModuleLevel int

);
```

Here we are creating a new table called **Modules** which has three attributes: **ModuleID** which must be a text value; **ModuleLeade**r which must be a text value; and **ModuleLevel** which must be an integer.

Exercise: Create a new table called Students that has the following attributes:

- Surname
- FirstName
- KUID
- Degree
- Year

One may indicate which attributes must always have a value when creating a new table. This is done by adding the expression NOT NULL to an attribute declaration. For example, in the following CREATE TABLE statement the ModuleID attribute must always have a value for every record.

Here is an example CREATE TABLE statement.

```
CREATE TABLE Modules

(ModuleID  text not null,

ModuleLeader text,

ModuleLevel int

);
```

Populating Tables…

Once we have created a new table we need to populate it with new records. This is done in either of two ways:

- By using an INSERT INTO statement to insert a record into a table
- By using an INSERT INTO…SELECT statement to insert records from an existing table into a new table

Here is an example of the INSERT INTO statement.

```
insert into Modules (ModuleID, ModuleLeader, ModuleLevel)

values ('BB4301', 'Dan Russell', 4);
```

After this statement is run the Modules table will have one record with the values listed in the tuple. To add more records you need to repeat this statement with the required values. Note that if you try to add a value of the incorrect type you may receive an error message or an unexpected result. Therefore you should always be guided by the table specification as defined in the CREATE statement. In Access one may view the design specification of a table by right clicking on a table and selecting DESIGN VIEW. For the Modules table you will see the following:

Field Name	Data Type
ModuleID	Text
ModuleLeader	Text
ModuleLevel	Number

Also note that if you are adding a complete record (one which has a value for every attribute) then you may use the following INSERT statement.

```
insert into Modules values ('BB5105', 'Barry Avery', 5);
```

It is important that you present the values in the correct order in the input tuple. That is, the module ID first, followed by the module leader and then the module level.

We may want to add an incomplete record to the table (one for which not every attribute has a value). To do this one should specify in the INSERT INTO statement the attributes to which the values are associated. For example,

```
insert into Modules(ModuleID, ModuleLevel)

values ('BB4321', 4);
```

Here the new record has two values: the ModuleID and the ModuleLevel.

Exercise:

Insert 10 records into the Students table.

Use the three different approaches to add the records.

A second way to populate a table with records is by selecting values from an existing table and inserting them into the new table. Assume that we have a new table called ModuleDetails which has three attributes: ModuleID, ModuleLevel and Department. We may populate it with the ModuleIDs and ModuleLevels of the modules in the Modules table by using the following statement.

```
insert into ModuleDetails(ModuleID, ModuleLevel)

select ModuleID, ModuleLevel from Modules;
```

Note that the INSERT INTO…SELECT statement uses a SELECT statement to select the attributes of the Modules table to add to the ModuleDetails table. This select statement may also include a WHERE clause if one only wants to select those records that satisfy a particular condition.

Exercise:

Create a StudentDetails table which has 3 attribute: KUID, Surname and Age.

Populate the table with the KUIDs and Surnames of the students in the Students table.

Updating Tables…

We may update existing tables by using an UPDATE statement. For example, we may want to add a department value to each record in the ModuleDetails table. Here is an example.

```
Update ModuleDetails

Set Department = "AFI"

Where ModuleID="BB4301";
```

This update statement sets the department attribute to the value “AFI” for the module with the ID “BB4301”.

Exercise:

Update the StudentDetails table with the age of each student.

You may also use update statements to update existing values. For example, if we wanted to change the department of the module with module ID “BB5105” to “A,F and I” we may use the following statement.

```
Update ModuleDetails

Set Department = "A,F and I"

Where ModuleID="BB5105";
```

Here the department name for module “BB5105” is changed from “AFI to “A,F and I”.

Exercise:

Use an UPDATE statement to change the age of one of the students in the Students table

Deleting Records…

We may want to delete records from an existing table. This is done by using a DELETE statement. For example, the following statement will remove all the level 4 records from the ModuleDetails table.

```
Delete * from ModuleDetails

Where ModuleLevel = 4;
```

Note that we use the where clause to select the records to be deleted. In this case, any record with a ModuleLevel of 4 will be deleted.

Exercise:

Delete all the students who are 19 years old from the Students table.

Deleting Tables…

Sometimes we want to delete a whole table. This may be done by using a DROP statement.

For example, if we want to delete the ModuleDetails table we may do this by using the following statement.

```
Drop table ModuleDetails;
```

Exercise:

Delete the StudentDetails table.