LAB 2: From data model to database

Getting started

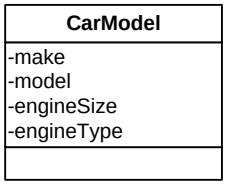
In this lab you will **implement and test** a database to represent part of the GCUCars data model you discussed in the tutorial. An example solution for the full data model will be provided.

Start up Microsoft Access and create a new database called ***gcucars.mdb***.

Create a new Word document called ***comu114_lab2.doc***. You will use this document to record all the SQL statements you use in this lab.

Task 1 : Implement a table in your database to represent the *CarModel* entity

You will start by creating a table to represent the *CarModel* entity. The attributes of *CarModel* are shown in the class diagram below:



Before you actually create the table you will need to decide on the following:

- table **name** (following the usual convention this should be *CarModels*)
- **fields**
- **data types** for fields
- **primary key** for the table (a separate *id* field is often the best choice)
- any additional **constraints**, for example NOT NULL

1. Create a new query in Design View, and close the Show Tables dialog without adding any tables. Save the query as *Create CarModels Table*. Now switch to **SQL View**.

2. Enter an SQL Create Table statement which will implement the table according to the decisions you have made. You can see examples of Create Table statements in chapter 2 of your notes. You should make sure your table will have a **primary key**.

   You should use the **Access SQL data types** reference on page 3 to help you choose data types for the fields in your table.

Your SQL statement should start like this:

```
CREATE TABLE CarModels (
```

3. Save the query again. If you have made any mistakes in the **syntax** of your SQL you will get an error message.

   *A **syntax error** tells you that you have typed something which is not valid SQL: for example, you might have missed out a bracket or a comma, or you may have misspelled one of your data types.*
   *You may find the default font size in SQL view rather small when looking for incorrect syntax.You can change the font size to whatever you want by clicking the **Office Button** and selecting **Access Options** at the foot of the menu. The Access Options window has a tab marked **Object Designers** and you can set the Query Design font and size there.*

4. Once you have saved the query then **run it**. If it works, you will not get any obvious feedback.

5. Select **Tables** in the database window, and check that your new table is there. Open your table and look at it in Datasheet View and Design View. Check that the correct fields are in the table. There will be no data in the table yet.

   If you are not happy with the table, you can right-click on the table in the database window and select Delete from the pop-up menu to remove it. You can then modify the query and run it again.

6. Finally, record your SQL in your Word document (*comu114_lab2.doc*). Type the heading *TASK1*, and copy and paste in the SQL statement from Access.

Access SQL data types quick reference

AccessSQL Data Type	Comments
Text(size)	Can specify max number of characters of **text**, up to 255
Memo	Use for **text** with more than 255 characters
Counter	SQL name for **AutoNumber**
Integer	
Long	Use for **foreign key fields** which match to **Counter** primary keys
Number	Use for **decimals**. You **can't** specify precision or decimal places in SQL, you must to go into Table Design View after table is created and set properties Field Size (choose Decimal), Precision and Scale (i.e. decimal places).
Double	Use for **real numbers**
Money	SQL name for **Currency**
YesNo	SQL name for **Yes/No**, stored values are -1 or 0
DateTime	SQL name for **Date/Time**

Notes:

- This is **not** a complete list of Access data types
- Some data types have alternative names, e.g. *Number* can also be *Numeric*
- see **http://office.microsoft.com/en-us/access/HP010322291033.aspx** for a complete reference

Task 2: Entering data into the table

You should now test that you can actually store data successfully in this table.

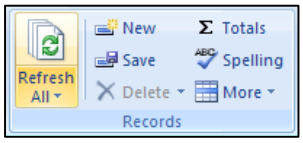
1. Create a new query in Design View and save the query as *Insert CarModel*. Switch to **SQL View**.

2. Enter an SQL Insert statement to store a row in the *CarModels* table representing the following model:

   Vauxhall Astra with 1.8 litre petrol engine

   You can see examples of Insert statements in chapter 2 of your notes.

3. Open the table in Datasheet View and check that the data has been stored successfully. If the table is open you will need to click **Refresh All** on the **Home** ribbon tab to see the new data.



4. Record your SQL in your Word document. Type the heading *TASK2*, and copy and paste in the SQL statement from Access.

   If you have problems with storing data, you may have to go back and modify your table. For example, you may have chosen a data type for a field which cannot store the required piece of information. If you have to change your create table statement, make sure you save the query and record the modified version in your Word document.

5. Add new rows to your table to represent the following car models. You can do this by editing your insert query, or by entering data in Datasheet view.

   Note that you can only have **one** INSERT INTO statement with a single set of VALUES in an Access query. To insert multiple rows you can:

   - create, save and run the query to insert the first row
   - edit the VALUES so that the query inserts the second row, and run the query again
   - edit and run the query to insert the next row
   - ...and so on

   - *Ford Focus with 1.8 litre petrol engine*
   - *Ford Mondeo with 2.0 litre diesel engine*
   - *Toyota Avensis with 2.2 litre diesel engine*
   - *Vauxhall Vectra with 2.2 litre petrol engine*
   - *Honda Jazz with 1.4 litre petrol engine*
   - *Vauxhall Corsa with 1.2 litre petrol engine*
   - *Toyota RAV-4 with 2.2 litre diesel engine*
   - *Vauxhall Antara with 1.9 litre diesel engine*
   - *Honda CR-V with 2.0 litre petrol engine*

6. You should try to add rows to test any **constraints** you have defined in your table.

   Firstly, if you have specified that any fields should be NOT NULL, you should check that you are not able to add a row with any of those fields left empty. For example, if the *model* field is NOT NULL, you could test and record the results in your Word document as follows:

   > **Test for constraint:** Table: *CarModels*, Field: *model*, Constraint: *NOT NULL*
   > **Test data:** VALUES ('Honda', null, 2.0, 'petrol')
   > **Expected result:** Error
   > **Result:** Error – Microsoft Access cannot append all the records in the append query
   > **Test passed**

   If a test is not passed, you should find and correct the problem and re-test.

   You can then test the primary key by adding a row which has the same primary key value(s) as an existing row (this test is not necessary if you used an autonumber id field)

7. Look at the engine type information. Can you think of a useful way in which this data could be constrained? You can try to implement this constraint in your table.

   Hint: to constrain a field to allow only values from a list, you need to view the table in Design View, and select the relevant field. You can then enter a Validation Rule which should be in the form

   In ('*first value*', '*second value*', '*third value*'**)**

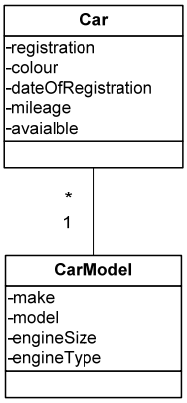
   You can specify any number of values. See your lecture notes for another example of a validation rule. Note that in Access most constraints need to be specified in table Design View after the table is created rather than in the CREATE TABLE statement.

   You should test this constraint and document the test in a similar way to the example in step 6 above.

   > **Test for constraint:** Table: *CarModels*, Field: *model*, Constraint: *validation rule (list of values)*
   > etc.

Task 3: Implement a related table

In the data model, the *Car* entity represents a specific vehicle, for example a specific Vauxhall Astra. There can be many cars of any one model - GCUCars might, for example, have five 1.8 litre Astras in their fleet.



You will now implement a table to store information about cars, and define the relationship between this and the *CarModels* table.

1. Design a table called *Cars* with the fields required to store the information about cars. As before, consider fields, data types, primary key and constraints.

2. Create, save and run new query called *Create Cars Table* which contains the SQL Create Table statement to implement this table.

3. Define, using SQL, the relationship between the *Cars* and *CarModels* tables. You will need to have a field (or fields) in *Cars* to match the primary key you defined in *CarModels*. You will also need to define the foreign key. You can do these tasks either:

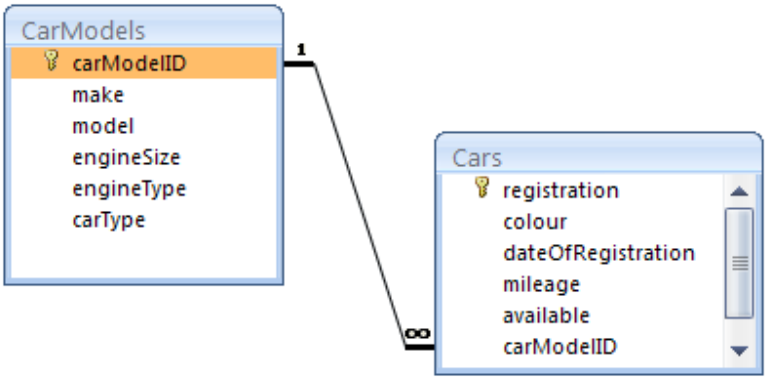
   as part of the Create Table statement
   or
   using Alter Table after the table has been created.

   Relevant examples can again be found in chapter 2 of your notes. Should the foreign key field(s) be NOT NULL? Why?

4. Record all SQL statements you use in your Word document under the heading TASK 3.

Now you need to test your new table.

5. Open the **Relationships** window and check that the relationship between *CarModels* and *Cars* is correctly defined. When you show these tables you should see that a **one-to-many** relationship linking the appropriate fields has already been created – it should look like this:



6. Store a row in the *Cars* table representing the following car which is available for hire. You can use SQL or Datasheet View.

   SD08XYZ, a red Astra, registered on 1/3/08, mileage 12500

7. Test that data which violate any constraints cannot be stored. **Record all test data** and the relevant constraints in your Word document.

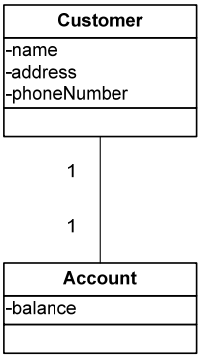
8. Test to check that the foreign key works as you intended:
   - Try to store a record with the foreign key left empty.
   - Try to store a record with a value in the foreign key which doesn't match a value in the primary key of *CarModels*
   - Record the results of these tests.

9. Finally, store rows representing the following cars:

   SE08XXZ, a green Focus, registered on 1/4/08, mileage 14000
   SF08XXY, a grey Focus, registered on 2/4/08, mileage 11800
   SG08XZY, a blue Astra, registered on 3/4/08, mileage 13000
   SH08ZXY, a silver Vectra, registered on 4/4/08, mileage 14600
   SJ08ZZZ, a silver Vectra, registered on 4/4/08, mileage 14900

Task 4: Implementing a one-to-one relationship

There is a **one-to-one** relationship between the *Customer* and *Account* entities in the data model.



In this task you should design, implement and test a table or tables to store the following information about the following customers:

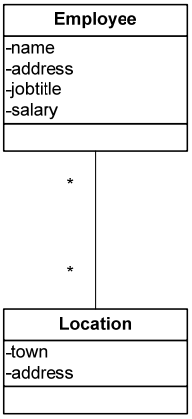
Dario Franchitti, 1 First Street, 0141 987 6543
Danica Patrick, 2 Second Street, 0141 876 5432
Marino Franchitti, 3 Third Street, 0141 765 4321
Sarah Fisher, 4 Fourth Street, 0141 123 4567

These customers all have an account with balance of zero, except Marino Franchitti, who has a balance of £156.68

Use SQL for table creation, and record SQL statements and additional test data in your Word document under the heading TASK 4.

Task 5: Implementing a many-to-many relationship

There is a many-to-many relationship between the *Employee* and *Location* entities in the data model.



In this task you should design, implement and test a table or tables to store the following information about the following employees and locations:

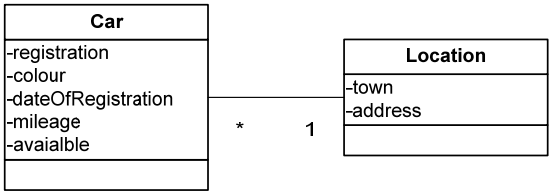
Locations:
1 First Avenue, Glasgow
2 Second Avenue, Paisley
3 Third Avenue, Irvine
4 Fourth Avenue, Kilmarnock
5 Fifth Avenue, Ayr

Employees:
Asif Rashid is an area manager, based at Glasgow and Paisley, earning £25000.
Jacklyn Elder is an area manager, based at Irvine, Kilmarnock and Ayr, earning £25000
Carol Koster is a CSR, based at Paisley, earning £15000
Martin Dale is a CSR, based at Kilmarnock and Ayr, earning £16000
Lennox Ward is a service engineer, based at Irvine, earning £21000
Jane Lee is a service engineer, based at Glasgow and Paisley, earning £22000

Use SQL for table creation, and record SQL statements and additional test data in your Word document under the heading TASK 5.

Task 6: Altering a table

You have already implemented tables to represent the *Location* and *Car* entities. However, you have not implemented the one-to-many relationship between them.



In this task you should use Alter Table statements to modify the tables to implement the relationship (actually, you should only need to modify one table). You should then test your modifications as follows:

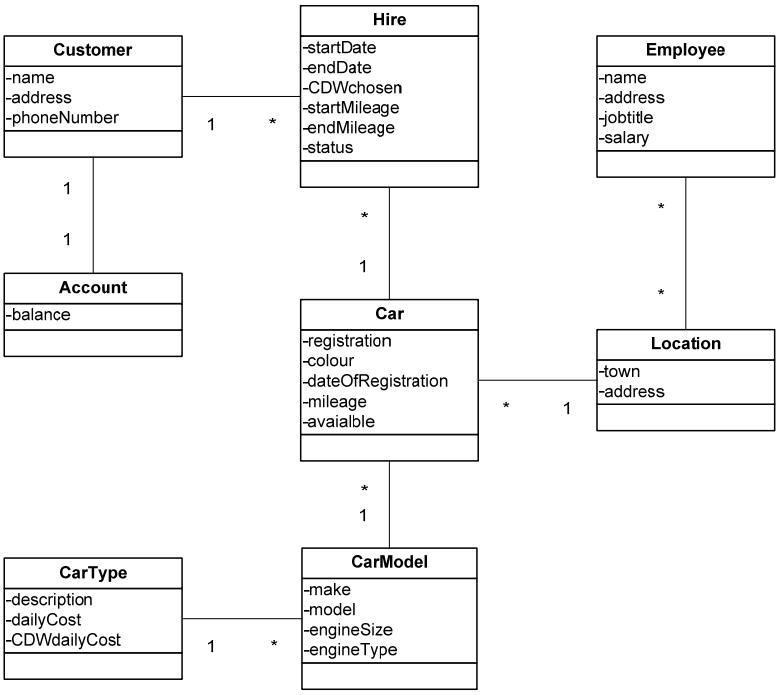
Specify the locations for cars:
SD08XYZ at Glasgow
SE08XX, at Paisley
SF08XXY at Paisley
SG08XZY at Paisley
SH08ZXY at Kilmarnock

Try to specify a non-existent location for SJ08ZZZ – this should be ***unsuccessful***
(To test this you will need to try to store a row in Cars with no matching row in Locations)

Use SQL for table modification, and record SQL statements and additional test data in your Word document under the heading TASK 6

TASK 7: Completing the database

If you have completed tasks 1 to 6, you should be able to complete the database so that it represents the whole data model. The entities you have not dealt with yet are *CarType* and *Hire*.



In this task you should design, implement, modify and test tables as needed to complete the database. You should be able to store the following information:

> *Vauxhall Corsa and Honda Jazz are in the **Economy** car type, daily cost £38, CDW £12*
> *Vauxhall Astra and Ford Focus are in the **Compact** car type, daily cost £45, CDW £14*
> *Ford Mondeo, Toyota Avensis and Vauxhall Vectra are in the **Intermediate** car type, daily cost £49, CDW £15*
> *Toyota RAV-4, Vauxhall Antara and Honda CR-V are in the **SUV** car type, daily cost £58, CDW £19*
>
> *Dario Franchitti hired SD08XYZ on 27/9/08 until 4/10/08 with CDW, start mileage 11800, end mileage 12500*
> *Sarah Fisher hired SG08XZY on 30/10/08 with no CDW, start mileage 13000, and this hire is not completed*

Use SQL for table creation and modification, and record SQL statements and additional test data in your Word document under the heading TASK 7.