

LAB 5: More queries and forms

Introduction

In this lab you will practice some more queries, and create some data entry forms.

Create a new Word document called **comu114_lab5.doc**. You will use this document to record all the SQL statements you use in this lab.

Task 1: Queries

In this task you will practice using SQL to query the GCUTours database.

Download the file **gcutours.mdb** from your course web site. Start up Microsoft Access and open gcutours.mdb.

Create a new query in Design View, and close the Show Tables dialog without adding any tables. Save the query as *Lab 5 Query*. Now switch to **SQL View**.

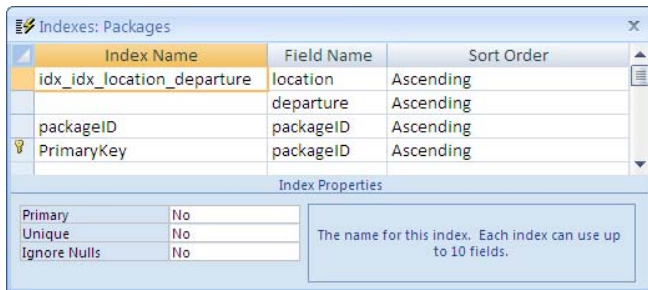
In each of the following questions you are given a description of the query you are required to create, and where appropriate, a listing of the result which your query should return.

For each question, type the **SQL statement** in *Lab 5 Query* and **run the query** to test it. You will probably sometimes need some trial-and-error before you get the correct result.

Write the heading *Task 1* in your Word document, and enter the question number in each case and copy and paste your SQL statement.

1. There may be a frequent need to look up packages by location and departure. Create (using SQL) a compound index called *idx_location_departure* on these two fields.

Check in the **Indexes** window that the index has been created. You can open the Indexes window by clicking the **Indexes** button icon on the **Table Tools > Design** ribbon tab. Note that you will need to have the Packages table open in **Design View** to see this button.



It is decided that there is no need for this index. Using SQL, drop the index *idx_location_departure*.

Again, check the Indexes window to see the effect.

2. In Lab 3 you found the package name and location for the tour with tourID = 3

| name | location |
|-------------------|----------|
| Western Adventure | USA |

using a subquery, like this:

```
SELECT name, location
FROM Packages
WHERE packageID =
    (SELECT packageID FROM Tours WHERE tourID = 3)
```

Write this query using a join instead of a subquery.

3. List for every booking the first name and last name of the user and the number of adults and children booked

| firstname | lastname | adults | children |
|-----------|----------|--------|----------|
| Marco | Polo | 2 | 2 |
| Marco | Polo | 1 | 0 |
| Marco | Polo | 2 | 2 |
| Vasco | daGama | 4 | 0 |
| Vasco | daGama | 2 | 0 |
| Ferdinand | Magellan | 2 | 3 |

4. List for every booking for which tickets have not been sent the first name and last name of the user and the number of adults and children booked

| firstname | lastname | adults | children |
|-----------|----------|--------|----------|
| Marco | Polo | 1 | 0 |
| Marco | Polo | 2 | 2 |
| Vasco | daGama | 2 | 0 |
| Ferdinand | Magellan | 2 | 3 |

5. List for every booking the tour departure date and the first name and last name of the user and the number of adults and children booked

| departureDate | firstname | lastname | adults | children |
|---------------|-----------|----------|--------|----------|
| 01/03/2008 | Marco | Polo | 2 | 2 |
| 05/06/2008 | Marco | Polo | 1 | 0 |
| 01/08/2008 | Marco | Polo | 2 | 2 |
| 01/03/2008 | Vasco | daGama | 4 | 0 |
| 02/09/2008 | Vasco | daGama | 2 | 0 |
| 01/06/2008 | Ferdinand | Magellan | 2 | 3 |

6. List for every booking the package name, the tour departure date and the first name and last name of the user and the number of adults and children booked

| name | departureDate | firstname | lastname | adults | children |
|----------------------------|---------------|-----------|----------|--------|----------|
| Western Adventure | 01/03/2008 | Marco | Polo | 2 | 2 |
| Roof of the World Explorer | 05/06/2008 | Marco | Polo | 1 | 0 |
| Amazon & Inca Adventure | 01/08/2008 | Marco | Polo | 2 | 2 |
| Roof of the World Explorer | 01/03/2008 | Vasco | daGama | 4 | 0 |
| Reef and Outback Adventure | 02/09/2008 | Vasco | daGama | 2 | 0 |
| Trans-Siberian Express | 01/06/2008 | Ferdinand | Magellan | 2 | 3 |

7. You try to list the name, location and departure date for every tour using the following query. Run this query. What happens? Why?

```
SELECT name, location, departuredate
FROM Packages, Tours
```

Modify the query to give the result below.

| name | location | departuredate |
|----------------------------|---------------|---------------|
| Western Adventure | USA | 01/03/2008 |
| Western Adventure | USA | 05/06/2008 |
| Western Adventure | USA | 02/09/2008 |
| Roof of the World Explorer | Asia | 01/03/2008 |
| Roof of the World Explorer | Asia | 05/06/2008 |
| Alpine Action | Europe | 01/03/2008 |
| Reef and Outback Adventure | Australia | 01/03/2008 |
| Reef and Outback Adventure | Australia | 02/09/2008 |
| Trans-Siberian Express | Asia | 01/03/2008 |
| Trans-Siberian Express | Asia | 01/06/2008 |
| Trans-Siberian Express | Asia | 01/08/2008 |
| Borneo Explorer | Asia | 08/03/2008 |
| Amazon & Inca Adventure | South America | 01/02/2008 |
| Amazon & Inca Adventure | South America | 01/08/2008 |
| Patagonia Trek | South America | 01/05/2008 |
| Colorado Winter Adventure | USA | 01/02/2008 |
| Colorado Winter Adventure | USA | 01/03/2008 |
| Raft the Grand Canyon | USA | 01/05/2008 |
| Raft the Grand Canyon | USA | 01/06/2008 |
| Raft the Grand Canyon | USA | 01/07/2008 |
| Raft the Grand Canyon | USA | 01/08/2008 |
| Rising Sun Explorer | Asia | 01/07/2008 |

8. List the tourID and departure date for every booking with at least one child. Write and run two versions of this query, one joining tables with WHERE and one using INNER JOIN.

| tourID | departuredate |
|--------|---------------|
| 1 | 01/03/2008 |
| 14 | 01/08/2008 |
| 10 | 01/06/2008 |

9. List the booking ID, package name and total cost for every booking. *Hint: the total cost will be the (number of adults x adult price) + (number of children x child price)*

| bookingID | name | TotalCost |
|-----------|----------------------------|-----------|
| 1 | Western Adventure | £4,996.00 |
| 2 | Roof of the World Explorer | £1,599.00 |
| 3 | Amazon & Inca Adventure | £6,996.00 |
| 4 | Roof of the World Explorer | £6,396.00 |
| 5 | Reef and Outback Adventure | £4,398.00 |
| 8 | Trans-Siberian Express | £4,795.00 |

10. List the total cost of each user's bookings. *Hint: you will need to group by username*

| username | SumTotalCost |
|----------|--------------|
| ferdy | £4,795.00 |
| mpolo | £13,591.00 |
| vdagama | £10,794.00 |

11. Find the number of bookings made for each location

| location | NoOfBookings |
|---------------|--------------|
| Asia | 3 |
| Australia | 1 |
| South America | 1 |
| USA | 1 |

12. Increase the prices of all packages by 10%.

13. Change the password for mpolo to Pa\$\$w0rd

14. Create a parameter query which you can run to change the password for any user. Your query should prompt for the username first, and then the password.

15. Delete the tour with tourID 5.

This query shouldn't work! Why not?

Edit the relationship between Tours and Bookings so that when you delete a tour you delete related bookings (you'll need to do this with the Relationships window, not with SQL). Run your query again and note changes to both tables.

Task 2: Creating forms

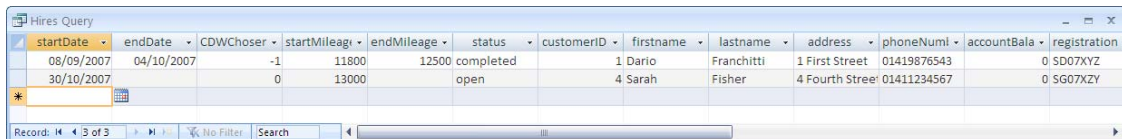
In this lab you will create forms to view and enter data in the GCUCars database.

Hires Form

1. Download the file **gcucars.mdb** from your course web site. Start up Microsoft Access and open gcucars.mdb.
2. Create a new query in Design View, and close the Show Tables dialog without adding any tables. Save the query as *Hires Query*.
3. Now switch to **SQL View**, and enter the following SQL and **save the query**.

```
SELECT startDate, endDate, CDWChosen, startMileage,
endMileage, status, Hires.customerID, firstname, lastname,
address, phoneNumber, accountBalance, registration
FROM Hires
INNER JOIN Customers
ON Hires.customerID = Customers.customerID
```

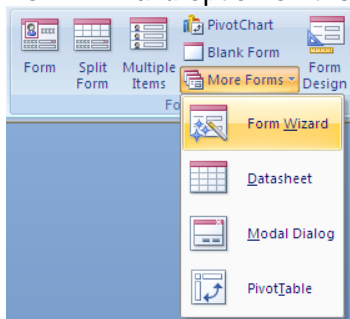
This query retrieves data from **two tables**: *Hires* and *Customers*. Run the query to test it. You should get a result like this:



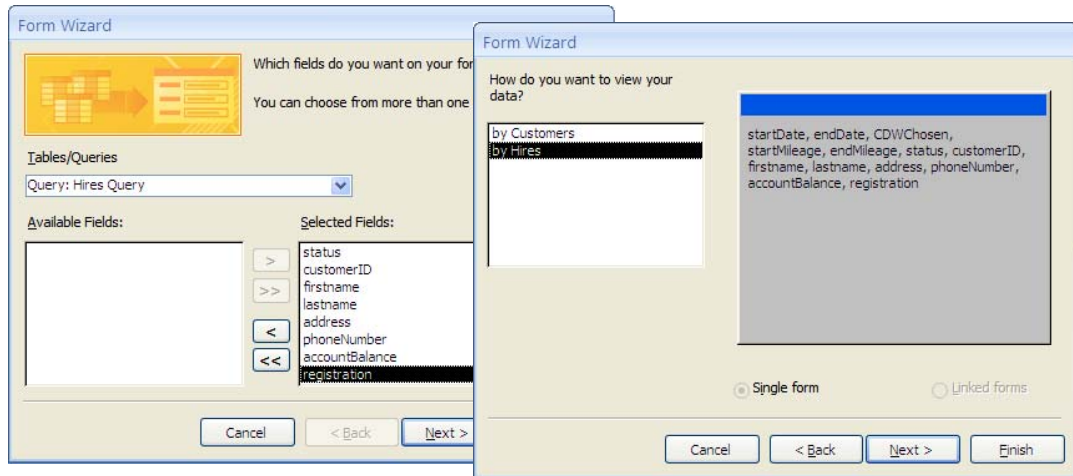
| startDate | endDate | CDWChoser | startMileagi | endMileage | status | customerID | firstname | lastname | address | phoneNuml | accountBala | registration |
|------------|------------|-----------|--------------|------------|-----------|------------|-----------|------------|----------------|-------------|-------------|--------------|
| 08/09/2007 | 04/10/2007 | -1 | 11800 | 12500 | completed | 1 | Dario | Franchitti | 1 First Street | 01419876543 | 0 | SD07XYZ |
| 30/10/2007 | | 0 | 13000 | | open | 4 | Sarah | Fisher | 4 Fourth Stree | 01411234567 | 0 | SG07XZY |
| | | | | | | | | | | | | |

Note that there is a **blank row** at the end of the data sheet to allow a new row to be entered. This shows that this query is **updateable**. Inserting or modifying data in the query will modify the underlying tables.

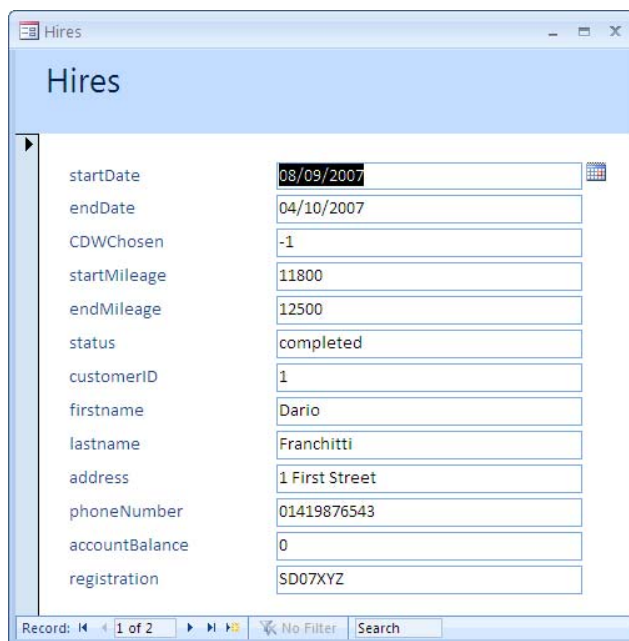
4. Create a new form as you did in lab 1, except that here you should select the **Form Wizard** option on the **Create** ribbon tab dialog.



5. Select *Hires Query* as the source of data and select all fields in the first step of the wizard. Choose to view data **by Hires** in the next step.



6. In the next steps, choose **Columnar** layout, **Access 2007** style and name the form *Hires*. The form should now look something like this:



| | |
|----------------|----------------|
| startDate | 08/09/2007 |
| endDate | 04/10/2007 |
| CDWChosen | -1 |
| startMileage | 11800 |
| endMileage | 12500 |
| status | completed |
| customerID | 1 |
| firstname | Dario |
| lastname | Franchitti |
| address | 1 First Street |
| phoneNumber | 01419876543 |
| accountBalance | 0 |
| registration | SD07XYZ |

You can use the navigation buttons to browse the hires.

What are the possible values of *CDWChosen*? What is the data type of this field.

- Use the button to create a new hire. Enter the following data, in order, using the TAB key to move between fields:

startDate: **5/11/07**
endDate: **NULL (leave blank)**
CDWChosen: **-1**
startMileage: **14900**
endMileage: **NULL**
status: **open**
customerID: **4**

Press the TAB key after entering the *customerID*. **What happens?**

Now enter the following data:

registration: **SJ07ZZZ**

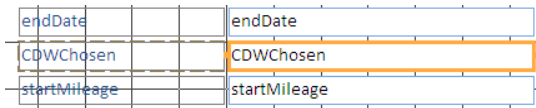
Press the ENTER key to store the record. Look at the *Hire* table to check that the new data has been stored.

Can you create a new Customer with this form?

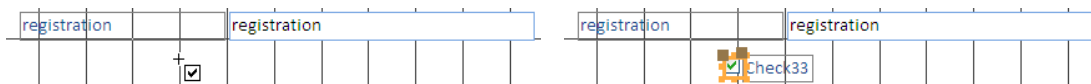
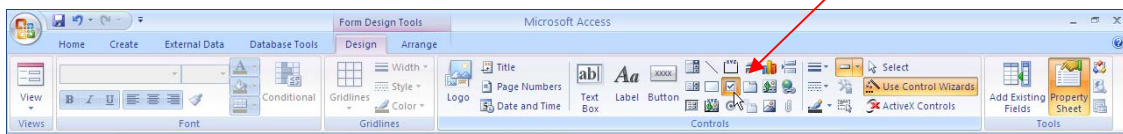
A text box is not a very good way of displaying the *CDWChosen* field. Let's change this to a **check box**.

- Switch to form **Design View**.

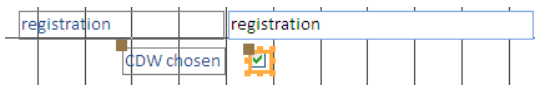
- Select and delete the *CDWChosen* box.



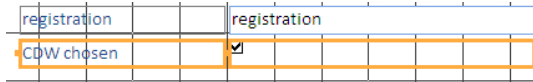
- Select **Check Box** in the **Form Design Tools > Design** ribbon bar. Move the cursor to below the fields on the form and click to place a **checkbox control** there.



- Move the label so that it is just below the registration label, and drag the check box to the right of its label. Edit the label text to read *CDW chosen*.



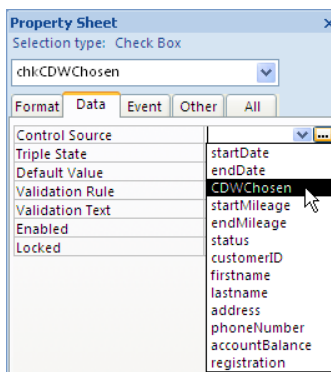
- With the check box label selected, press the UP ARROW key until the new control becomes part of the tabular layout of the other controls. This may take one or more key presses.



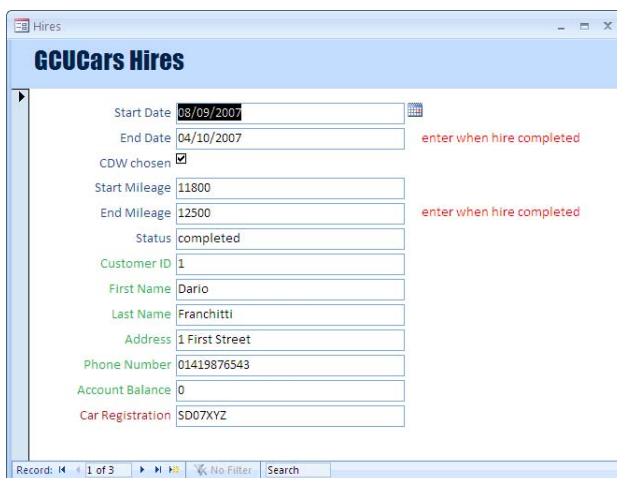
- Keep pressing the UP ARROW to make the checkbox move up the layout until it is immediately below *endDate*, in the position where the original CDWChosen box was.

Note that Access laid out the form controls as a **Stacked layout**. You can remove this layout using the **Form Design Tools > Arrange > Layout** ribbon bar, and then move each control individually.

- Select the check box and use the **Property Sheet** to set the **Control Source** to *CDWChosen* and **name** to *chkCDWChosen*.



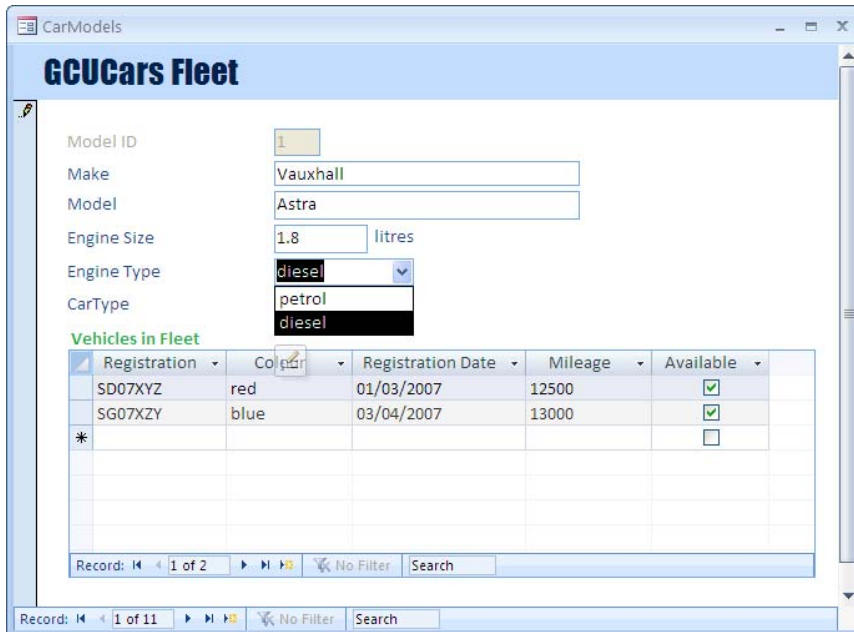
In Design View, you can experiment with the design, for example by repositioning controls, editing labels, adding new labels, changing text box formats, and so on. For example, your completed form might look like this:



Cars Form

Create a **master-detail form** to view data on **car models** and **cars**. The master form should show car models, and the details form should show all the cars of the currently displayed model.

Chapter 6 of your notes will help you with this task, and your completed form should look something like this:



The screenshot shows a web application window titled "CarModels" with a sub-header "GCUCars Fleet". The form contains the following fields:

- Model ID: 1
- Make: Vauxhall
- Model: Astra
- Engine Size: 1.8 litres
- Engine Type: diesel (dropdown menu)
- CarType: petrol (dropdown menu)

Below the form is a table titled "Vehicles in Fleet" with the following columns: Registration, Colour, Registration Date, Mileage, and Available. The table contains two rows of data:

| Registration | Colour | Registration Date | Mileage | Available |
|--------------|--------|-------------------|---------|-------------------------------------|
| SD07XYZ | red | 01/03/2007 | 12500 | <input checked="" type="checkbox"/> |
| SG07XZY | blue | 03/04/2007 | 13000 | <input checked="" type="checkbox"/> |
| * | | | | <input type="checkbox"/> |

At the bottom of the application, there are navigation controls for records. The top bar shows "Record: 1 of 2" and the bottom bar shows "Record: 1 of 11".

Task 3: Using a query to help normalisation

When you normalise a database you alter existing tables and move fields to new tables until your tables satisfy the rules. This is straightforward if you normalise when you are first designing the database, or before the database has been put to use.

Sometimes, though, you need to 'fix' an existing database which might have a large amount of data in it already. You don't want to lose the data, and you don't want to manually enter lots of data in new fields or tables. Fortunately, INSERT queries can help.

1. Download the file **normalisation_demo.mdb** from your course web site and open it in Access. This database contains some of the data used in chapter 4 of your notes.
2. Open the *Consultants* table. This table is not in first normal form (why?). There are only three rows in the table, but in a real situation there could be many more.

Consultants

| consultantID | firstname | lastname | skill1 | skill2 | skill3 |
|--------------|-----------|----------|-----------|------------|-----------|
| 1001 | John | Smith | databases | Java | UML |
| 1002 | Amy | Jones | networks | web design | |
| 1003 | Jane | Lee | Windows | Linux | databases |

To get to 1NF, you need to apply the rule given in your notes to give two tables:

Consultants

| consultantID | firstname | lastname |
|--------------|-----------|----------|
| 1001 | John | Smith |
| 1002 | Amy | Jones |
| 1003 | Jane | Lee |

ConsultantSkills

| consultantID | skill |
|--------------|------------|
| 1001 | databases |
| 1001 | Java |
| 1001 | UML |
| 1002 | networks |
| 1002 | web design |
| 1003 | Windows |
| 1003 | Linux |
| 1003 | databases |



3. Create the *ConsultantSkills* table with the fields and foreign key shown.
4. Create a new query and save it as *Copy to ConsultantSkills query*.
5. Switch to SQL View, enter and save the following SQL:

```
INSERT INTO ConsultantSkills(consultantID, skill)
SELECT consultantID, skill1 FROM Consultants
WHERE skill1 IS NOT NULL
```

6. Run the query and look at the *ConsultantSkills* table (you may need to refresh the table, or close and reopen it, to see the updated data). You should see rows in the table containing the skills listed in the *skill1* field in *Consultants*.
7. Modify the query to add the data from the *skill2* field and run it. Repeat for the *skill3* field. The *ConsultantSkills* table should now contain the data shown in the figure (not necessarily in the same order). Copy and paste the SQL you use into your Word document under the heading *Task 3*.
8. Delete all the skills fields from the *Consultants* table.

You can also try the following example if you wish. This takes a table from **1NF** to **2NF**.

1. Add a field *hourlyRate* to your *ConsultantSkills* table, and enter the data shown in page 8 of your notes (chapter 4). The table is now not in 2NF, as described in the notes.
2. Create a new table *Skills* with the fields shown in page 9 of the notes (chapter 4).
3. Devise and run a query which inserts the correct data into *Skills*, and drop the *hourlyRate* field from *ConsultantSkills*. You should NOT type any data into the table manually. Copy and paste the SQL you use into your Word document under the heading *Task 3 part 2*.
4. Set up the foreign key relationship between *ConsultantSkills* and *Skills*.
5. Delete the *hourlyRate* field from *ConsultantSkills*.