Database Practicals 7 & 8 1999-2000

Objectives To set up a small multi-table database, with an appropriate 'user interface', including forms for data input.

(a) The files **e:\examples\jgb\students.txt** and **e:\examples\jgb\courses.txt** list some information about an imaginary university. A printed copy of each is given on the sheet of data. Each student takes three courses, and details are given in the marklists. Your first task is to set up an appropriate database and incorporate all the information provided. We have discussed this situation in class, and you might find it useful to start with an E-R diagram.

(b) The **students** and **courses** files are in the import format that you have previously used in Practical 2. (A copy of the relevant instructions from Practical 2 is available on the CS10610 web page; you will have to interpret them appropriately in the new situation!) Set up a new database and import the data from these files. Set the (course) **Code** as the primary key of the "courses" table, and make a note of its **Data Type** (**Text**)**.** Add an extra field (say **StudentID)** of type **Autonumber** to the "students" table and make it the primary key. Adjust the field sizes to appropriate values. Make a note of the **Code** field size.

(c) Set up a "registrations" table to link your other two tables, and also hold the marks. It should have a field **StudentID,** which should be of Data type **Number** and field size **Long integer** to match the **Autonumber** key of the "students" table. It should also have a field **Code,** which should have exactly the same **Data Type** and **Field Size** as the field **Code** in the "courses" table. Make these two fields **(StudentID** and **Code)** jointly the primary key of this table. Finally there should be a field to contain the mark. Since a mark of zero means something different to a missing mark, you should delete the **Default Value** of 0.

(d) *Before you enter any data* in the registrations table, set up relationships

(i) between the "students" table and the "registrations" table, and

(ii) between the "courses" table and the "registrations" table.

In each case you should **Enforce Referential Integrity** and make the relationship 1 to many (if this fails, you should check the match between the fields you are trying to relate).

(e) Enter a few of the marks provided, to check that the relationships are correctly set. Note that you will have to do some translating, since the students must be referred to by their ID's and the courses by their codes in the "registrations" table.

(f) The following four sections describe how to set up a form to input the marks. When you have added each control, you may like to switch from Design view to Form view to see the effect. (You may remember that a similar recommendation was made for queries in practical 3.) However, you should realise that you will not be able to use the form to actually write to a new record until you have the controls to put in *both* parts of the primary key.

(g) **To create the basic form:** Select **Forms** in the database window, and choose **Create form in Design view.** Choose **Properties** from the **right click menu** associated with the black-centred square in the top left of the form window. This shows properties associated with the form as a whole, rather than with individual controls. On the **Data** tab, set the **Record Source** to your "registrations" table. (There is a droplist giving the available tables.) **Close** the properties sheet.

(h) **To add a combo box to input the *Code:*** Use the **Toolbox** to add a **Combo Box** control to your form. In the (bulleted) stages of the combo box wizard make the choices indicated.

- You want the combo box to **… look up the values in a table …**
- Choose your "courses" table to provide the values.
- The **Code** field contains the values.
- Set the width of the column.
- Store the value in the **Code** field. (Since we set the **Form Record Source** to the "registrations" table, this is the Code field *of the "registrations" table*, not the "courses" table. You should have been able to work this out from the choices offered!)
- Choose an appropriate label.

Open the **Properties Sheet** *of the Combo Box control.* Set the **Limit to List** property to **Yes.** Also inspect the values of the **Auto Expand, Enabled,** and **Locked** properties, whose default values are the values we want. Save the form, giving it an appropriate name, and inspect its behaviour in Form view.

(i) **To add a Combo Box to input the *StudentID:*** The procedure is as in the previous section, except that you should choose the **"students"** table, and the **Name** field to provide the values *(yes I really did mean Name).* You should store the value in the **StudentID** field. *Access* realises that the **Name** is inappropriate to store in the **StudentID** field (wrong data type, wrong for the relationship) and will store the value from the appropriate **(StudentID)** field of the same record in the "students" table.

(j) Add a **Text Box** (with label) control to your form to input the mark. You now have to bind the control to the appropriate field in the "registrations" table. Use the properties sheet *of the control* to choose the correct **Control Source** from the list of possibilities offered.

(k) Now use your form to input the rest of the marks. Firstly, try (mouse) selecting a Code and a Name from the combo boxes, and then typing in the mark. Secondly, try inputting from the keyboard alone, a better way of doing it! You should find the **Tab** key takes you round the controls, and on to the next record (writing the current one) when you have filled them in. Since we set *Limit to List,* the Code and Name controls will only accept values already stored in the relevant field of the relevant table. (Note that this restriction is dynamic, in the sense that adding a new Code to the "courses" table results in it becoming available to the combo box; try it!) Since we set *Auto Expand,* the control will use the first character(s) typed in to guess a completion; if the guess is incorrect, continue typing until the correct completion is uniquely identified.)

(l) Produce a form that the Registrar can use to send a student's results to the student.

(m) Produce a report showing, on a separate page for each course, the students in descending mark order, together with a count of the number of students on the course, and the average mark for the course. (This cannot be based on a single table, so you should create a suitable query to base it on before you start the Report Wizard.)

(n) Look in the **Help** for instructions on how to add a chart to your report. Add a chart (graph) to the end of your report showing the average mark for each of the courses, so that they may be compared.

(o) Produce forms for adding new students and new courses. (To decide what types of controls are required, think carefully about what the forms have to do!)

(p) Print (one sample page of) your forms, and enough pages of your report to show all the features, for your practical file.

(q) ***Before signing off your practical,*** the demonstrator will ask to see your Registrar's form and your report, and to see some of your input forms in action.