CS230 Lecture 18
Validation and Referential Integrity

Tuesday, April 27, 2004

Validation Rules in Queries

Nothing that we have done so far ensures that user's only ever enter valid data into tables. The database designer needs to specify the validation rules that distinguishes the data that is valid from the data that is not.

Validation rules can indicate:

- A required field
- The entry is within a certain range of values
- Legal values
- The default value of a field

Problem 1

Let us go back to the customer table. Let us make the address, city, state, amount paid and current due be fields that are not required (could be left blank), while the customer number, name, zip and driver number should be required fields.

You should also specify a range for the amount paid and the current due. Both of these should be positive amounts. You should therefore type >=0 in the validation rule box. In the validation text box you type the message that appears to the user when they enter a negative number. Test your range now by adding a record to your table and trying to type in a negative amount.

Let us add another field to the customer table to store the customer type. The type must be EDU, MAN or SER. We must therefore specify that these are the only possible legal values in this field. We will again do this in the validation rule box. Type =”EDU” OR “MAN” OR “SER”. Test that the rule works correctly by trying to enter some incorrect values.

Referential Integrity

Last week we learnt that tables should have primary keys. We also saw how we can join two tables together. Joining two tables requires that a field from one table match a field in another table.

A foreign key is a field in one table whose values are required to match the primary key for another table.

Question: Do we have a foreign key in our customer and driver tables?

The property that ensures that the value in a foreign key must match another table’s primary key is called referential integrity.

Before we specify referential integrity, we must specify a relationship between the two tables. Access will then prohibit any updates to the database that would violate the referential integrity.

The type of relationship between two tables specified by the relationship command is referred to as a one-to-many relationship. This means that one record in the first table matches many records in the second table, but each record in the second table is only related to one record in the first table.

Problem 2

Let us create a relationship between the driver number in both tables.

From the Tools menu select Relationships and add both tables to the relationship. Drag the driver number from one table to the driver number on the other table. In the box that appears check the Enforce Referential Integrity box.

Let us try to add and delete records and observe how referential integrity works.

Subdatasheets

Once you have a relationship between two tables, it is possible to view the fields in which the primary key appears as a foreign key.

We know this by the + sign that has appeared to the left of the driver number. Click on the + sign to see what I mean.

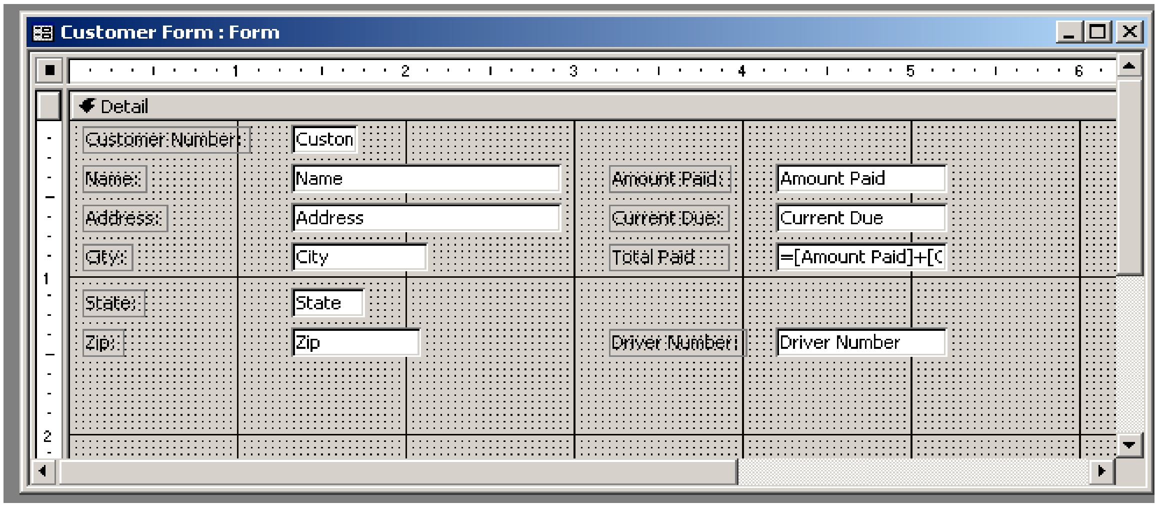
Complex Forms

The form that we created last week was created using the AutoForm option. This created a standard form where all the fields were listed on the left hand side. As you can imagine this is not an entirely useful or pleasing form.

The easiest way to create our own custom forms is to use the form wizard to create a basic form for us, then we can modify it as we see fit.

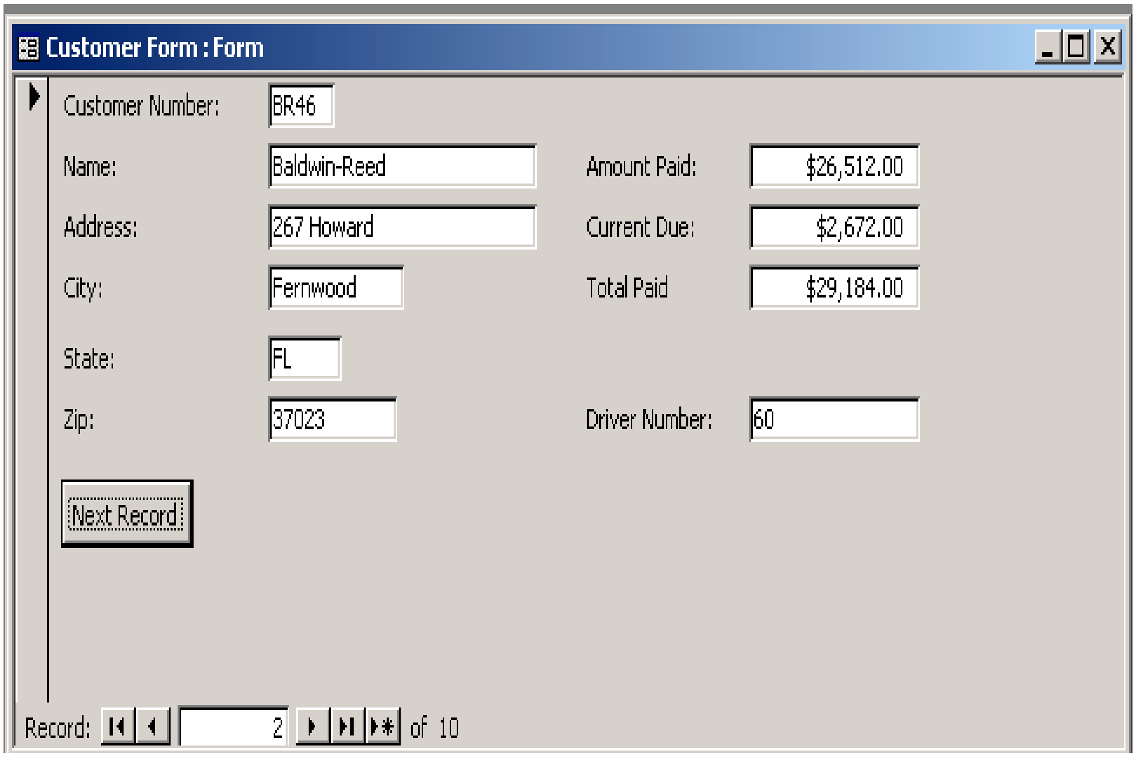
Problem 3

The goal is to create a form like the one below for the driver table.



Customer Number:	Custon		
Name:	Name	Amount Paid:	Amount Paid
Address:	Address	Current Due:	Current Due
City:	City	Total Paid	=[Amount Paid]+[C
State:	State		
Zip:	Zip	Driver Number:	Driver Number

Once you have created the basic form above, try and enhance the form by adding command buttons and a combo box for easy navigation. To create Command Buttons, select the Design View and using the Toolbox, drag a button onto the screen giving it the functionality of "Next Record." Do the same for "Previous Record", "Add Record", "Delete Record", and "Close Form."



Customer Number:	BR46		
Name:	Baldwin-Reed	Amount Paid:	$26,512.00
Address:	267 Howard	Current Due:	$2,672.00
City:	Fernwood	Total Paid	$29,184.00
State:	FL		
Zip:	37023	Driver Number:	60

Record: 2 of 10

Since we can have images in forms we are going to add a couple of fields to the driver form. Let us add one field for the driver's picture (you can just download any picture from the Internet to use) and the driver's Internet home page.

Advanced Forms

Yesterday we created an advanced form that used data from two tables and listed the picture of the driver. Now we will look at combo boxes.

Combo Boxes

A Combo box is the control that has a drop down list of valid values.

We will now modify our form so that it contains a drop down list of the customer name. The end user can then use the combo box to directly jump to the details of a specific customer.

Problem

Open up the form from yesterday and add a combo box somewhere in the form. The combo box wizard should automatically appear. Use the wizard to link the combo box to the name field in the customer table.

Switch to form view and test out your combo box.

Question: Can you see any problems/limitations with the combo box as it now stands?

We can tackle some of these limitations directly by programming in Visual Basic.

Visual Basic for Applications

VBA is an event-driven programming language that can be used with Access.

Programs in visual basic consist of code, which is a collection of statements (commands), which are instructions that will cause actions to take place when the program is executed.

When working with Access, visual basic deals with events. Events are associated with user actions such as Click, DblClick, MouseDown, …

Variables: 255 characters beginning with a letter including letters, numbers, underscore (no spaces).

Assignment Statement: `Factor = 1.05`

Simple IF

```
    If Customer_Type = "EDU" Then
        Factor = 1.05
    End If
```

Note: The If statement can also have an Else and ElseIf component.

Comments: these are notes to yourself that will be ignored by visual basic. You indicate a comment by placing an apostrophe just before the comment. Everything to the right of the apostrophe will now be a comment.

Functions : these are a group of statements that calculate and return a value.

```
Function Factor (Customer_Type)
‘ Determine factor base on Customer Type
If Customer_Type = “EDU” Then
     Factor = 1.05
Else
     Factor = 1
End If
End Function
```

Subroutines: these are a group of statements that do not return a value.

```
Public Sub ShowPromotion ()
     txtPromoAmount.Visible = True
     txtPromoFactor.Visible = True
     cmdPromoQuery.Visible = True
End Sub
```

Module: a group of procedures

- Standard Module – procedures available anywhere in the DB
- Class Module – procedures available in a particular form or report

This was just a quick look at the basics in visual basic. The best way to really get a hang of it though is to use it directly.

Problem

We will use VB to modify the Add Record button so that as soon as we click on it, it will place an insertion point in the customer number box.

Question: What currently happens when we click on the Add Record button?

The first thing you need to do is go back to the Customer table and rename the field Customer Number to Customer_Number.

Question: Why do we need to do this?

Go back to the form, and in the design view, click on the text box and again rename Customer Number to Customer_Number. Once you have done this, click on the Build Event option on the toolbar.

VB opens up the code and goes directly to the subroutine for the text box. After the line that starts with DoCmd add
Customer_Number.setFocus

Close VB and test out the form.

Problem

The next thing that we are going to do is modify the combo box so that it

- matches the name of the current record
- sorts the names alphabetically
- is not included in the tabs

Another Example

Let us move onto another database example. For this one you will be working more independently but I’ll be there to help if needed.

Basically, every action we perform at the computer is an event that is sent to the Operating System. Events can be typing, mouse clicks, mouse moves, …

Let’s take a look at some Form events in Access:

- Open Event
- Load Event
- Resize Event
- UnLoad Event
- Close Event

Event Actions

```
Private Sub Form_Load()
     MsgBox “Form Is Loaded"
End Sub
```

Problem

Load the DB VBAccess1.mdb and add the following code for the Open Event associated with the form frmAlgebraicOperators

Variable Declarations

In general, variables are declared using Dim

```
Private Sub NameIt()
     Dim Name As String
     Dim Value As Integer
     Dim Found As Boolean
     Dim Total As Single
End Sub
```

Other Possible Data Types

- Byte
- Long
- Double
- Currency
- Date
- Object

- Variant

Operators

‘ Comment	‘ This is a comment
= Assignment	txtName = txtFirstName
“ String	“hello there”
& Concatenate	“hi” & “there”

Problem

Continue using VBAccess1.mdb and do the following:

For each button, write the code that will calculate and display the Perimeter and Area for both the Square and Rectangle.

Problem

Insert another page into the VBAccess1.mdb database called Compute Wage. Allow the user the ability to input Hours Worked and Hourly Wage. Then call a function that returns the amount of money earned. Add a calculate button that when pressed calls your function and displays the proper value in the text area.

```
Function AmountEarned (dblHoursWorked As Double,
dblHourlyWage As Double)

End Function
```

You will need to use an IF Then Statement

Looping

The general form for looping (doing something repeatedly) is:

```
Do While Condition
  Statements
Loop
```

More Looping

```
Do
     Statements
Loop While Condition

Do Until Condition
     Statements
Loop
```

Yet More Looping

```
For Counter = Start To End
     Statements
```

```
Next
For Each Element In Group
     Statements
Next Element
```

Example

```
Sum = 0
For Counter = 1 To 10
     Sum = Sum + Counter
Next
```

Problem

Add another page called SumIt that allows the user the ability to enter a starting value and an ending value. Calculate and place the sum and average of the numbers, from the starting value to the ending value, in two separate text box areas in the form.