Lecture 14 - Implementing a Simple Database

Note that you should have read `Lecture 14 - Designing a Simple Database` before reading these notes.

There will also be some accompanying videos with this for the tutorial.

Disclaimer - There are betters ways to implement this, but it is a simple demonstration of how to do the basics in Microsoft Access

The following notes will walk you through how to implement a simple database solution for a Library. For simplicity, the data model we use will not be completely representative of what you would use in reality as this would require more complexity.

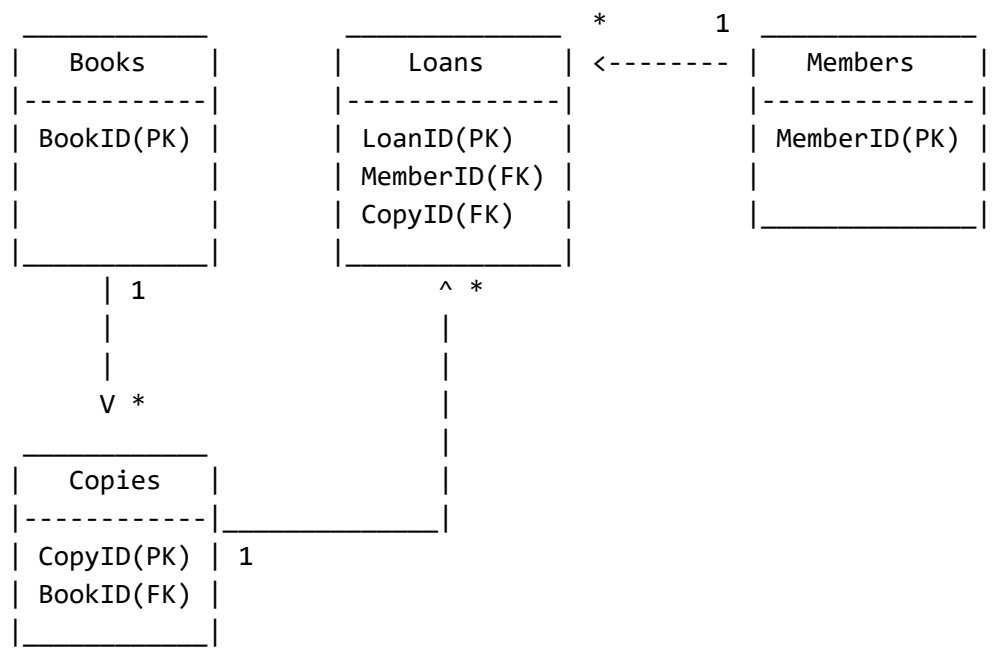
Our tool of choice for this will be **Microsoft Access** which is a very good starting point for learning databases.

1. Entity Relationship Diagram (ERD)

The following ERD represents the tables and relationships which we wish to implement.

Each table contains a primary key (PK) and in some cases one or more foreign keys (FK) which establish relationships between them.

If you do not understand this then you should read the `Lecture 14 - Designing a Simple Database` notes before proceeding.



2. Full Description of Tables

`Books`

	Field Name	Data Type
Primary Key	BookID	Auto Number
	Title	Short Text
	Author	Short Text
	Genre	Short Text
	PublicationYear	Number

`Copies`

	Field Name	Data Type
Primary Key	CopyID	Auto Number
Foreign Key	BookID	Number
	Condition	Short Text
	Borrowed	Yes/No
	Active	Yes/No

Setting the `Active` field to `No` will mark a copy as deleted, this is non-destructive and allows recovery of data and a historical record.

`Members`

	Field Name	Data Type
Primary Key	MemberID	Auto Number
	Name	Short Text
	Email	Short Text
	PhoneNumber	Short Text
	Address	Short Text
	Active	Yes/No

Setting the `Active` field to `No` will mark a copy as deleted, this is non-destructive and allows recovery of data and a historical record.

`Loans`

	Field Name	Data Type
Primary Key	LoanID	Auto Number
Foreign Key	CopyID	Number
Foreign Key	MemberID	Number
	BorrowDate	DateTime
	ReturnDate	DateTime

3. Basic Implementation Steps

1. Create the tables in Microsoft Access. Use this guide from Microsoft.

   Build tables with Table Designer

   - Certain fields should be set as required - See this guide

2. Create the relationships between the tables. Use this guide from Microsoft.

   Get started with table relationships

You can also add data from Excel.

4. A More Intuitive Interface

We would like the following forms.

1. Books - Manage the `Books` table.
   - **Add** a book
   - **Edit** a book
   - Do NOT allow **deletion**
2. Members - Manage the `Members` table.
   - **Add** a member
   - **Edit** a member
   - Do NOT allow **deletion**
3. Copies - Manage the `Copies` tables.
   - **Add** a copy
   - **Edit** a copy
   - Do NOT allow **deletion**
4. Loans - Manage the `Loans` and `Copies` tables.
   - **Add** a loan
     - when a loan is made the `Borrowed` field in the `Copies` table should be set to `Yes` for the copy of the book.
   - **Edit** a loan - needed when loan returned
     - when a loan is returned the `Borrowed` field in the `Copies` table should be set to `No` for the copy of the book.

- Do NOT allow **deletion**

4.1 Implementing the Books/Copies and Members

The following guide will prove useful for this.

Create a from an existing table

Click on the **Books** table and then click **Create** from the ribbon. If you then click **Form** Access will automatically create a form. If the relationship between **Books** and **Copies** is set up correctly, it will also create a subform that will allow you to add copies of the books.

You can now do the same with the **Members** table and you will get a members form with a subform for loans.

Now go to the Design View and set the Locked property to **Yes** as we want this to be read-only. We will add and edit loans on a separate form, but seeing what loans a member currently has from one place is useful.

4.2 Implementing the Loans Form

This is more complicated and involves creating a Query. We will first create the Query and then come back to implementing the form.

5. Creating a Query

In a database context, a query is a request for data or information from a database. It's a command used to retrieve, insert, update, or delete data in a database management system (DBMS) based on specific criteria. Queries are typically written in a structured query language (SQL) such as MySQL, PostgreSQL, SQL Server, or Oracle SQL.

We will use this query to combine data in our `Books` and `Copies` table to see which books are available to Loan.

This will allow us to **only** display copies that are available to loan on the form.

5.1 Creating the Query

Click **Create** on the MS Access ribbon and then click **Query Design**. Finally, change the **View** to **SQL View**.

We are now going to use SQL (Standard Query Language) to extract only the books from the database that have the **Borrowed**** field in the table **Copies** set to **No**. An intuitive way to think of this is as a filtered list of available **Copies**.

We will also want to combine the **Books** and **Copies** table so that we can use the book **Title** and **Author**.

Ideally we want to be able to have the following information for only the records (entries) in **Copies** that have **Borrowed** set to **No**.

CopyID	Title	Author

The following SQL query will achieve this.

Note MS Access does not support comments (words in green), so remove them when typing in the query in the SQL View editor.

```
-- Select the following fields
Select CopyID, Title, Author
-- From the combined Books and Copies tables
FROM Books INNER JOIN Copies ON Books.BookID = Copies.BookID
-- Where the borrowed field is set to No
WHERE Copies.Borrowed = No;
```

After typing this in click on **Run** and you should see that a table has been generated which now has the required information.

6. Loans Form

The loans form will have the following features:

1. Select a Member - Show `MemberID` with the member name (required)
2. Select an available copy for loan (required)
3. Input `LoanDate` (this should be enforced to be in the future) (required)
4. Input `ReturnDate`
5. When data for the first 3 is entered and the record created, the `Borrowed` field for the selected copy should be set to `Yes`.
6. When a `ReturnDate` is entered, the `Borrowed` field for the selected copy should be set to `No`.

6.1 Creating the Loans form controls

Using the form wizard select `LoanID`, `LoanDate` and `ReturnDate` from the `Loans` table and click **Finish**. Now press **Ctrl + s** to save the form.

Technically, we don't need to add the `LoanID`, but I think it is fine to have on the form (as an AutoNumber type it won't be editable).

We now need to add in the functionality to add the copy of the book and the member who is loaning the book.

Switch the view of the form to **Design View** and add a **Combo Box** to the form, this will open the **Combo Box Wizard**. Select the first option and click **Next**. Now select **Queries** and then select `CopyID`, `Title` and `Author` and click **Next**.

Now select **Title** for the sort records and click **Next**. Click **Next** again and make sure **Copy ID** is selected, then click **Next**.

Now choose **Store the value in this field** and select `CopyID` from the drop down menu and click **Next**. Give the label a name, e.g. **Book to Loan** and click **Finish**.

You will now have a combo box that will only display available copies of a book.

Now repeat the process, but instead select `MemberID` and `Name` from the `Members` table and store the `MemberID` in the `MemberID` in a similar manner to the `CopyID` above.

6.2 Setting the `Borrowed` field

We will use VBA for this. You can ask ChatGPT how you would do this, but the code is provided below.

First, make sure you have the Loans form open.

You can now open the VBA editor by clicking on the **Create** tab on the Ribbon and clicking on the Visual Basic option on the right-hand side.

We are going to add an event handler. This will trigger when a record is saved.

If a **ReturnDate** has not been given then we will set the **Borrowed** field to **Yes** for the appropriate record. If it has been entered we will set it to **No**.

Add the following snippet of code to the bottom of the code window that has opened within VBA.

```
Private Sub Form_AfterUpdate()
    Dim db As DAO.Database
    Dim rs As DAO.Recordset
    Dim strSQL As String
    Dim CopyID As Long

    ' Get selected CopyID from the combo box
    CopyID = Me.CopyID.Value

    ' Check if ReturnDate is not Null for the selected CopyID
    If DCount("*", "Loans", "CopyID = " & CopyID & " AND ReturnDate Is Not Null")
> 0 Then
        ' Set up a SQL statement to update the Copies table to False
        strSQL = "UPDATE Copies SET Borrowed = False WHERE CopyID = " & CopyID
    Else
        ' Set up a SQL statement to update the Copies table to True
        strSQL = "UPDATE Copies SET Borrowed = True WHERE CopyID = " & CopyID
    End If

    ' Execute the SQL statement
    Set db = CurrentDb
    db.Execute strSQL, dbFailOnError

    ' Clean up
    Set rs = Nothing
    Set db = Nothing
End Sub
```

Now when you create a loan or add in a Return Date the Copies table will be updated appropriately.

7. Improving the Forms and Interface

You should realise that this use of forms is quite limited and somewhat unfriendly.

For example, if the database has a lot of entries, then returning a book would require the librarian to find the Loan to close it off.

```
As an exercise you should list all the possible improvements you might make to the
forms.

You might also consider what reports you could run. For example - What books were
most popular? What books had all copies out? (i.e. do we need to order more copies
of a book).
```