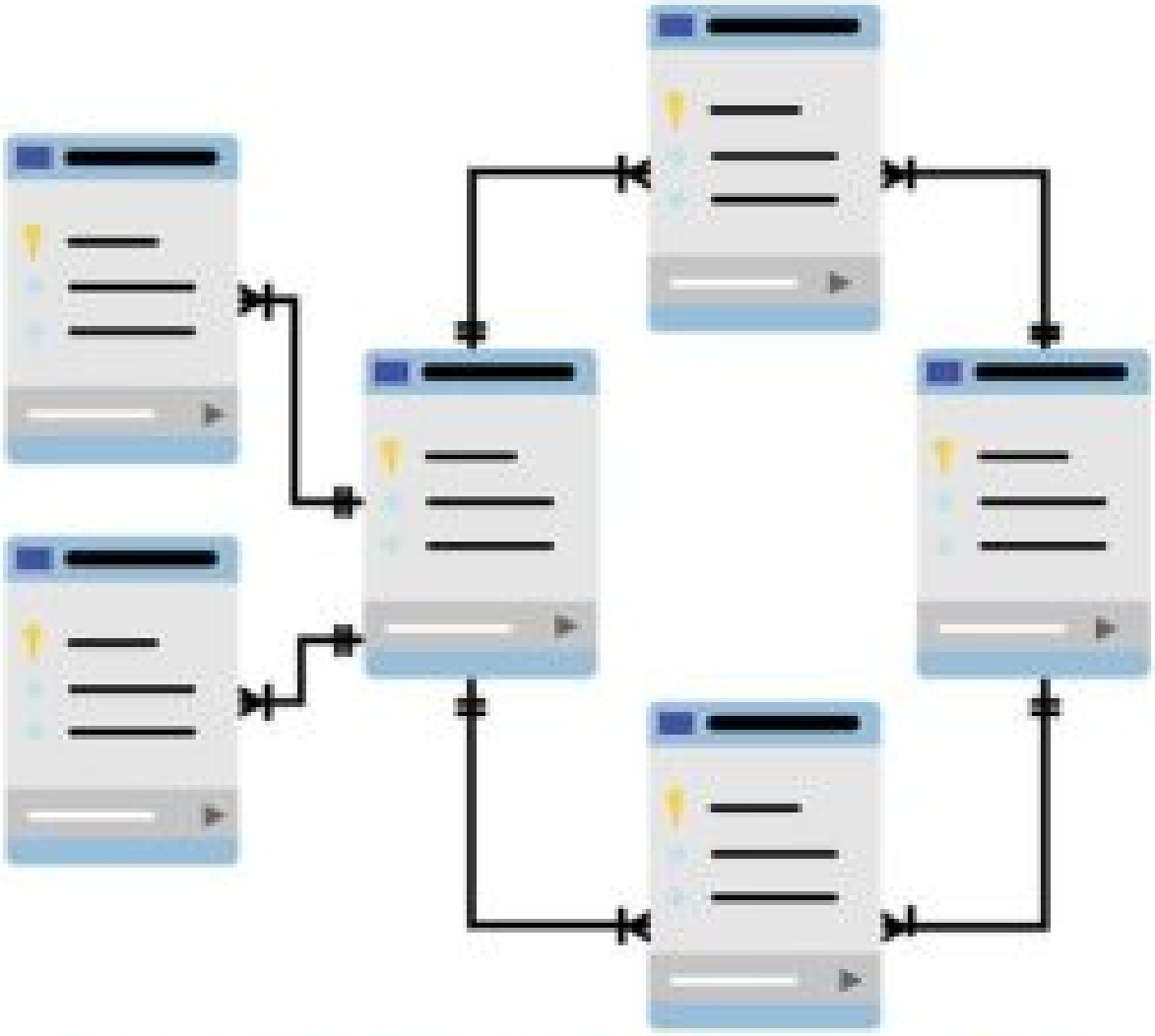


RELATIONAL DATABASES AND MICROSOFT ACCESS 365

Version 4.0

Relational Databases and Microsoft Access 365

Relational Databases and Microsoft Access 365

RON MCFADYEN

Contents

PREFACE ix

Acknowledgements x

1. RELATIONAL DATABASES AND MS ACCESS

1.1 Relational Databases 3

1.2 Microsoft Access 8

1.2.1 Modifying Rows 13

1.2.2 Adding New Rows 14

1.2.3 Deleting Rows 15

1.2.4 Table Design View 16

2. CREATING TABLES

2.1 Using Design View To Create Tables 23

2.1.1: Data Types 25

2.1.2: Properties 28

2.1.3: Primary Keys 33

3. CREATING FORMS

3.1: Using the Form Wizard 39

3.2: Modifying the Form 41

3.2.1: Adding a Button 42

3.2.2: Adding a Label 44

3.2.3: Adding a Calculated Field 45

4. MICROSOFT ACCESS QUERIES

4.1: Simple Query 49

4.2: Projection Query 53

4.3: Selection Query 55

4.4: Sorting the Result 57

4.5: And 59

4.6: Or 60

4.7: Joins 62

5. RELATIONSHIPS AND THE RELATIONSHIPS TOOL

5.1: Integrity 67
5.2: Relationships 68
5.2.1: One-To-Many 69
5.2.2: One-To-One 71
5.2.3: Many-To-Many 72

6. MICROSOFT ACCESS QUERIES – ADVANCED

6.1: Logical Expressions 77
6.1.1: And 78
6.1.2: Or 79
6.1.3: Not 80
6.2: Query Operators 82
6.2.1: Like 83
6.2.2: In 85
6.3: Query Properties 87
6.3.1: Top Values 88
6.3.2: Unique Values 90
6.4: Totals Query 93
6.5: Parameter Query 99
6.6: Crosstab Query 101
6.7: Action Queries 103
6.7.1: Make Table Query 104
6.7.2: Append Query 105
6.7.3: Delete Query 106
6.7.4: Update Query 107
6.8: Inner and Outer Joins 109
6.8.1: Inner Join 111
6.8.2: Outer Join 114
6.8.3: Cartesian Product 117
6.8.4: Self-Join 118
6.8.5: Anti-Join 120
6.8.6: Non-Equi Join 122
6.9: SQL Select Statement 124
6.10: SQL Union and Union All 127

7. ENTITY RELATIONSHIP MODELLING

7.1: Introduction 131

7.2: Entities 135
7.2.1: Weak Entities 137
7.3: Attributes 140
7.3.1: Atomic Attributes 141
7.3.2: Composite Attributes 142
7.3.3: Single-Valued Attributes 143
7.3.4: Multi-Valued Attributes 144
7.3.5: Derived Attributes 146
7.3.6: Key Attributes 147
7.3.7: Partial Key 149
7.3.8: Surrogate Key 152
7.3.9: Non-Key Attributes 153
7.3.10: Nulls 154
7.3.11: Domains 155
7.4: Relationships 156
7.4.1: Degree 157
7.4.2: Participation 158
7.4.3: Cardinality 159
7.4.4: Recursive Relationships 162
7.4.5: Identifying Relationships 164

8. MAPPING AN ERD TO A RELATIONAL DATABASE

8.1: Mapping Rules 171
8.1.1: Entity Types 172
8.1.2: Relationship Types 173
8.1.3: Attributes 174
8.2: Examples 175

9. DATA DEFINITION LANGUAGE (DDL)

9.1: Running DDL in MS Access 181
9.2: Example 182
9.2.1: DDL Commands 183
9.2.2: Creating the Database 184

10. NORMALIZATION

10.1: Functional Dependencies 189
10.1.1: Keys and Non-Keys 195
10.1.2: Anomalies 196
10.1.3: Partial Functional Dependencies 198

10.1.4: Transitive Functional Dependencies 200
10.2: Normal Forms 202
10.2.1: First Normal Form(1NF) 203
10.2.2: Boyce-Codd Normal Form (BCNF) 206
10.3: Summary 214

APPENDIX A

Forms Involving Multiple Tables 223

APPENDIX B

B.1: Drawing Supertypes and Subtypes on the Red 231
B.2: Supertypes, Subtypes and Relationships 233
B.3: Supertypes, Subtypes and Attributes 234
B.3.1: Discriminator Attributes 235
B.4: Mapping Supertypes and Subtypes to a Relational Database 236
B.4.1: Relations For All Entity Types 238
B.4.2: Relations for Bottom-Most Entity Types 242
B.4.3: One Relation Representing the Whole Hierarchy 244

PREFACE

This text is a free introductory text that introduces MS Access and relational database design. The motivation is to support an introductory database system course which, to the student, is either a service course providing an introduction to database concepts, or, as a prerequisite for more advanced study in the field.

Various texts have been used with some success but were felt lacking for various reasons such as: (1) being workbook style with extensive tutorial lessons, (2) being too focused on a technology, (3) having design material that did not fit well with more advanced courses, and (4) being so expensive that some students opted not to purchase.

Our second-year course has no prerequisites and is taken by students from various disciplines. However, most students are registered in either a Computer Science major program or the Computer Science minor. Students who enroll in the course obtain: (1) a working knowledge of a personal database system (MS Access), (2) knowledge of SQL (primarily the Select statement), and (3) awareness of concepts and techniques necessary to database design.

Following this course, students can take third- and fourth-year courses in the database subject area. The coverage of Entity Relationship Modelling in those courses is based on the Chen notation – as is usual for academic texts. To be consistent with those higher-level courses the same approach is used here.

It is our opinion that many students find normalization theory a difficult topic. Many presentations on normal forms are more complicated than necessary (e.g., some texts will give more than one definition of some normal forms). Our approach has been largely motivated by writings of Chris Date. We have attempted to give a suitable introduction to normalization theory for the beginning database student and to relate that material to other topics such as entity relationship diagrams.

This version includes two appendices that cover:

- creating forms that display data in a parent/child format where two tables are related via a one-to-many relationship, and
- entity-relationship modeling for supertypes and subtypes.

Acknowledgements

The website for this book is http://www.acs.uwinnipeg.ca/rmcfadyen/CreativeCommons/

This project is made possible with funding by the Government of Ontario and through eCampusOntario's support of the Virtual Learning Strategy. To learn more about the Virtual Learning Strategy visit: https://vls.ecampusontario.ca.

1. RELATIONAL DATABASES AND MS ACCESS

A *database* is an organized collection of data. A database may be on paper, or, held in computer files such as spreadsheets or more formally in a software system known as a computerized database management system (for example: DB2, db4o, IMS, MS Access, MS SQL Server, mySQL, Oracle, Sybase, Total, Versant). In this book we focus on Relational databases and one specific relational database system: Microsoft Access available with Microsoft 365.

There are many different commercial relational database systems and what you learn here will assist you in using those others. Because MS Access is a workstation/personal system it is a convenient system for beginners.

1.1 Relational Databases

Relational Databases were introduced by E. F. Codd in 1969[1]; Codd's 1970 paper[2] is considered one of the great papers in Computer Science.

We begin with a very small example: a database with one relation, the list of employees shown in figure 1.1. You should notice this looks just like a two-dimensional table of rows and columns. The name of the table is Employees, each column of the table has its own title, and each row has the same structure. Each row has a value for employee number, first name, last name, and gender. As tables of data appear in so many places (newspaper articles, textbooks, web pages, etc.) it is very likely you have seen and used this representation for data previously.

Employee ID	First Name	Last Name	Gender
123	Joe	Smith	Male
333	Jim	Jones	Male
456	April	Smith	Female
842	Jenny	Jones	Female
777	Tom	Lee	Male

Figure 1.1: A list of employees

Let us assume the Employees table in figure 1.1 has one row for each employee who works for some hypothetical company. Data kept for each employee comprises their employee identification number, their first and last names, and their gender. Information structured in tables is very concise; at a glance we can obtain useful information.

According to the database design methodology in *Information Modeling and Relational Databases*[3], a database designer must be able to express structured information as *verbalizations*. A verbalization that fits the information in one row of the Employees table is:

- *Employee with ID ... has a first name ..., a last name ..., and is of ... gender*

In verbalizations like this the ellipses are placeholders: we can use values from a single row to create complete statements that explain the meaning of a row. For example,

- Employee with ID 123 has a first name Joe, a last name Smith, and is of Male gender
- Employee with ID 333 has a first name Jim, a last name Jones, and is of Male gender

A similar approach to organizing knowledge about data appears in the literature on literacy. In the Journal of Reading several articles by Kirsch and Mosenthal discuss the organization of information and its conceptualization as document sentences. In *Building Documents by Combining Simple Lists*[4], Kirsch and Mosenthal present an example based on information from The World Almanac and Book of Facts: 1980 (Newspaper Enterprise Association, p. 427). That data is reproduced in figure 1.2.

Magazines	Circulation
TV Guide	19,547,763
Reader's Digest	18,094,192
National Geographic	10, 249,748
Better Homes & Gardens	8,007,202
Family Circle	7,611,578
Woman 's Day	7,535,855
McCall's	6,502,880

Figure 1.2: Circulation of leading U.S. magazines

A major point the authors make is that such information can be re-conceptualized as a series of simple document sentences formed from a basic *document sentence*. This document sentence expresses an understanding of the tabular data in natural language.

The document sentence for figure 1.2 is:

- *Magazine X has a circulation of Y.*

Kirsch and Mosenthal use variables (X and Y) to stand for data that comes from a table. Taking values from a row, we plug values for X and Y into the document sentence to obtain sentence instantiations:

- TV Guide has a circulation of 19,547,763.
- Reader's Digest has a circulation of 18,094,192.
- National Geographic has a circulation of 10,249,748.
- Better Homes & Gardens has a circulation of 8,007,202.
- Family Circle has a circulation of 7,611,578.
- Woman's Day has a circulation of 7,535,855.
- McCall's has a circulation of 6,502,880.

Document sentences and verbalization sentences are essentially the same. Both sentences use natural language to express in words the meaning of tabular data. Whether one is designing databases or reading structured information, it can be useful for understanding to re-formulate data as statements in natural language.

Let us be a bit formal for a moment. Commercial *relational* database systems are systems where data is organized into *relations*. Figure 1.3 shows the general structure of a relation. We say a relation comprises a set of *tuples* where each tuple has the same number of *attribute values,* where each attribute value is taken from some corresponding *domain*, and where a domain represents a set of valid values for an attribute.

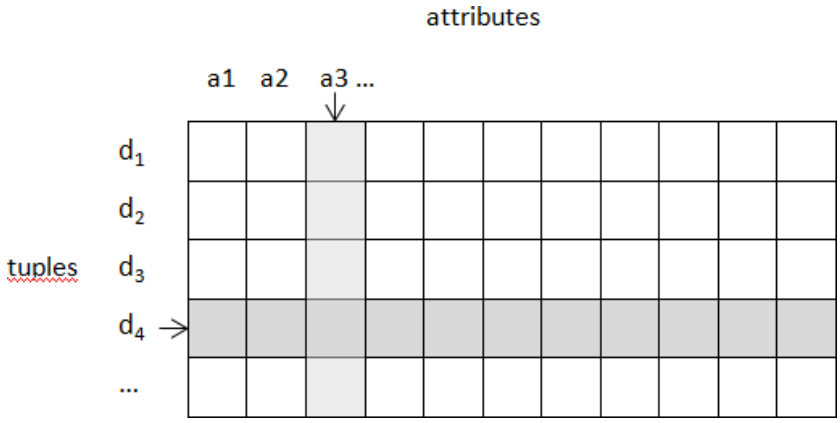


Figure 1.3: General structure of a relation

The Employees table in figure 1.1 can be considered a relation of 5 tuples where each tuple has 4 values drawn from each of the employee identifier, first name, last name, and gender domains. Similarly, we can say the lists comprising the Circulation of leading U.S. Magazines in figure 1.2 can be considered a relation with 7 tuples each having 2 attribute values.

Relations are typically implemented in commercial databases as tabular structures comprising rows and a fixed number of columns. Everybody is familiar with tables as they are commonplace in textbooks, papers, magazines, etc. This simplicity of representation is one reason why relational databases have been very successful as repositories for important data.

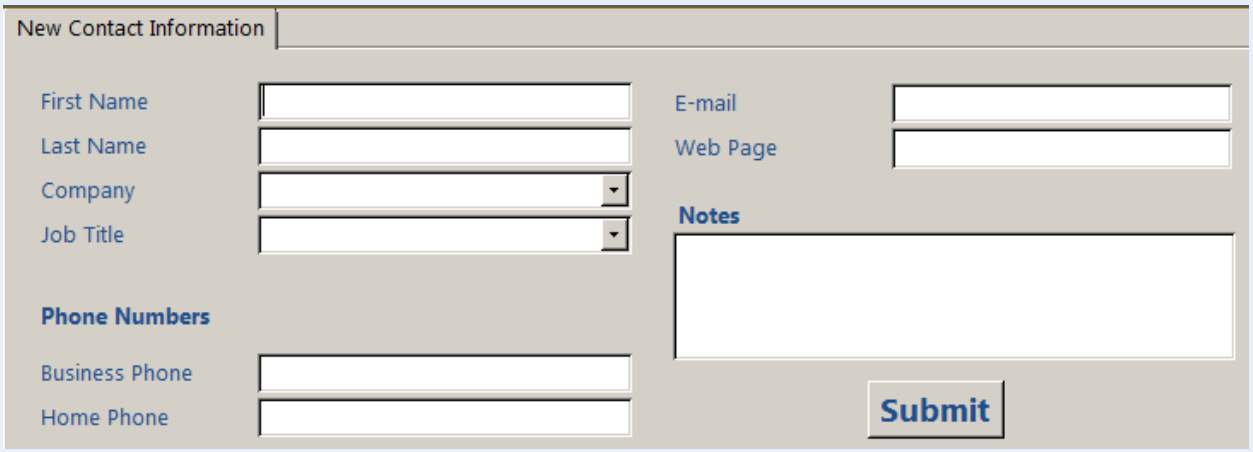
Exercises

To design a database, a database engineer needs to find good representations of how an organization uses data. Good sources include input forms, reports, web pages, etc. A challenge for database designers is to find these sources and interpret them.

1) Consider the following table of product information sold by ABC Foods. Verbalize the information presented.

Product ID	Product Name	Unit Price	Units In–Stock
1	Black Tea	$2.00	44
2	Green Tea	$3.00	33
3	Vegetarian Lasagne	$10.00	20
4	Cajun Seasoning	$11.00	29
5	Cranberry Sauce	$21.00	0

2) Suppose the following input form is used to enter contact information. Verbalize the information that is being collected:



3) Consider the following report that the Human Resources department of ABC Foods must produce. Verbalize the information in that report.

Employee ID	First Name	Last Name	Department
1	John	Smith	Receiving
2	Lee	Daniels	Sales
3	April	Turner	Sales
4	Thomas	Trump	Marketing
5	Lee	Smith	Marketing

1 Derivability, Redundancy, and Consistency of Relations Stored in Large Data Banks; IBM Research Report, 1969.
2 A Relational Model of Data for Large Shared Data Banks; CACM 13, No. 6, June 1970.
3 Information modeling and relational databases, 2nd edition; Terry Halpin and Tony Morgan; Morgan Kaufmann Publishers; ISBN -13 978-0-12-373568-3.
4 Building documents by combining simple lists; Irwin S. Kirsch and Peter B. Mosenthal; Journal of Reading, Vol. 33, No. 2, pp. 132-134.

1.2 Microsoft Access

MS Access is a relational database system for workstations that run the Microsoft Windows operating system. MS Access is typically used by individuals for data they use personally, but in some situations a single MS Access database may be used by a group of people or small department.

MS Access databases are stored in a single file that has a file suffix of ".accdb" or ".mdb". Databases created using MS Access 2007 and later have a file suffix ".accdb", and databases created using MS Access 2003 or earlier have a file suffix ".mdb". We will be using databases where the files have names ending in ".accdb". You need to use MS Access 2007 or later to open these databases. We have used Access available in Microsoft 365.

Our first sample database is in a file named Library.accdb; this database is available from the website associated with this text.

To use this database, you must first download the file containing the database, and then either:

- Navigate to its location in File Explorer, and open the database by double-clicking the file name.

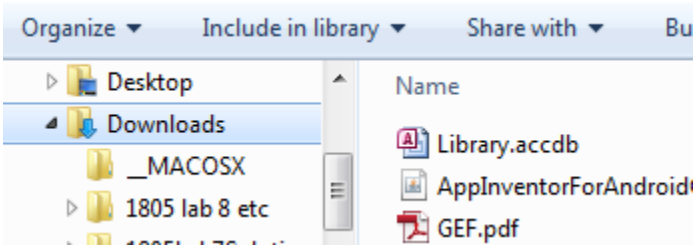
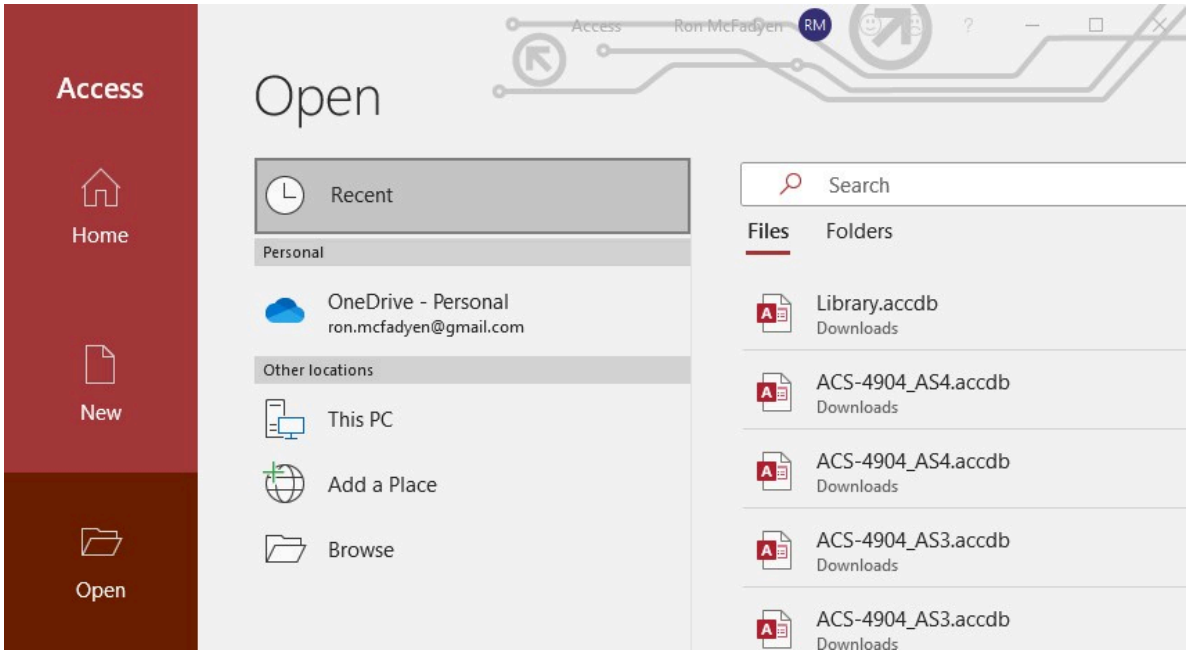


Figure 1.4: Double-click the database file to open the sample database

- Start Access and then browse to the folder holding the database, select it, and open it.

Select Open in Access:



Then browse to the location of the database file and open the file:

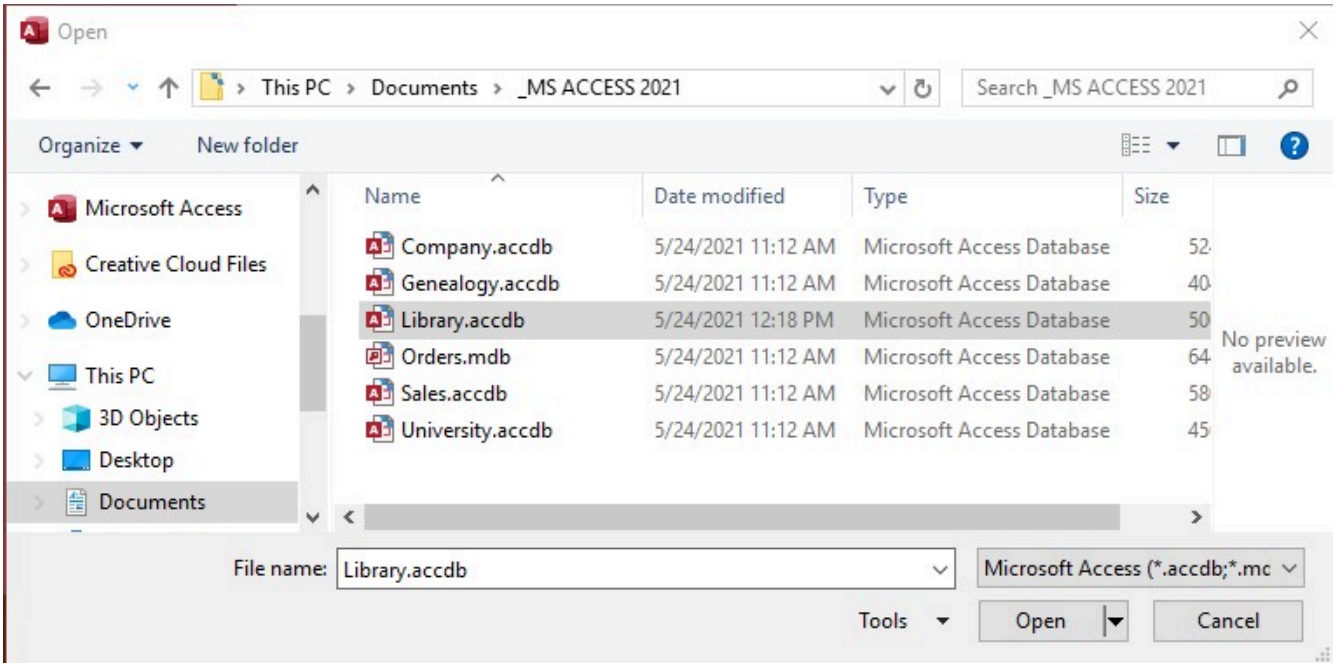


Figure 1.5: With Access started, browse and open the database

When you open this database, you see a list of objects (figure 1.6) in the database; you will see three tables: Book, Loan, Member:

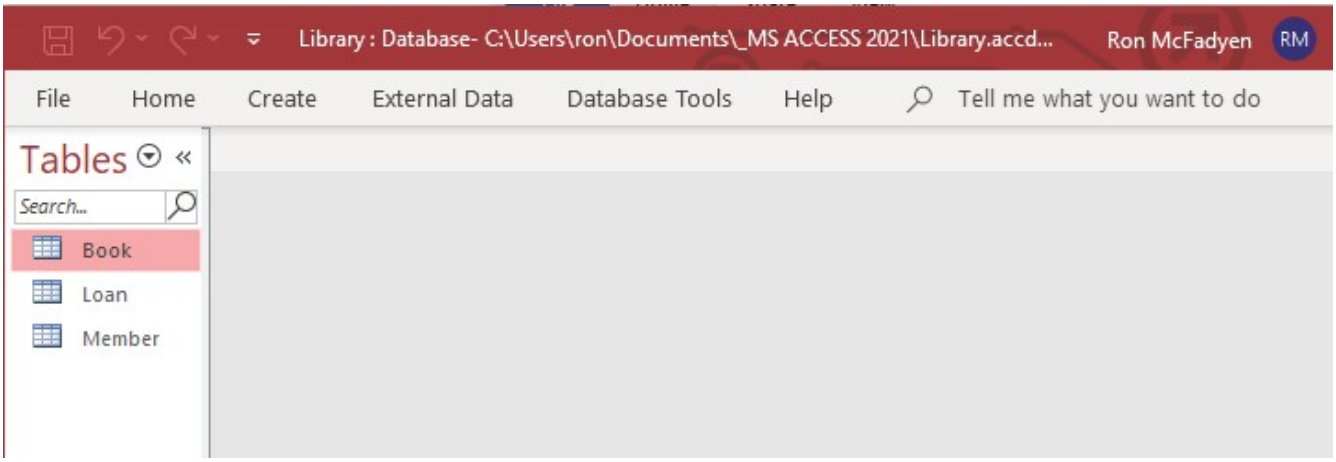


Figure 1.6: The Home tab in MS Access shows you the table names

Double-click a table name and MS Access opens the table in *Datasheet View*; you can see the contents of Book in figure 1.7. The datasheet view for a table is easily obtained, but it is not a particularly user-friendly way to view and manage data in a table. We will learn other ways of handling data with MS Access Forms. The Book table has three fields (i.e., attributes): callNo, title, author. When we view a table we see data organized into rows and columns. The data in one row corresponds to one book; if there are 11 books, then we have a table of 11 rows.

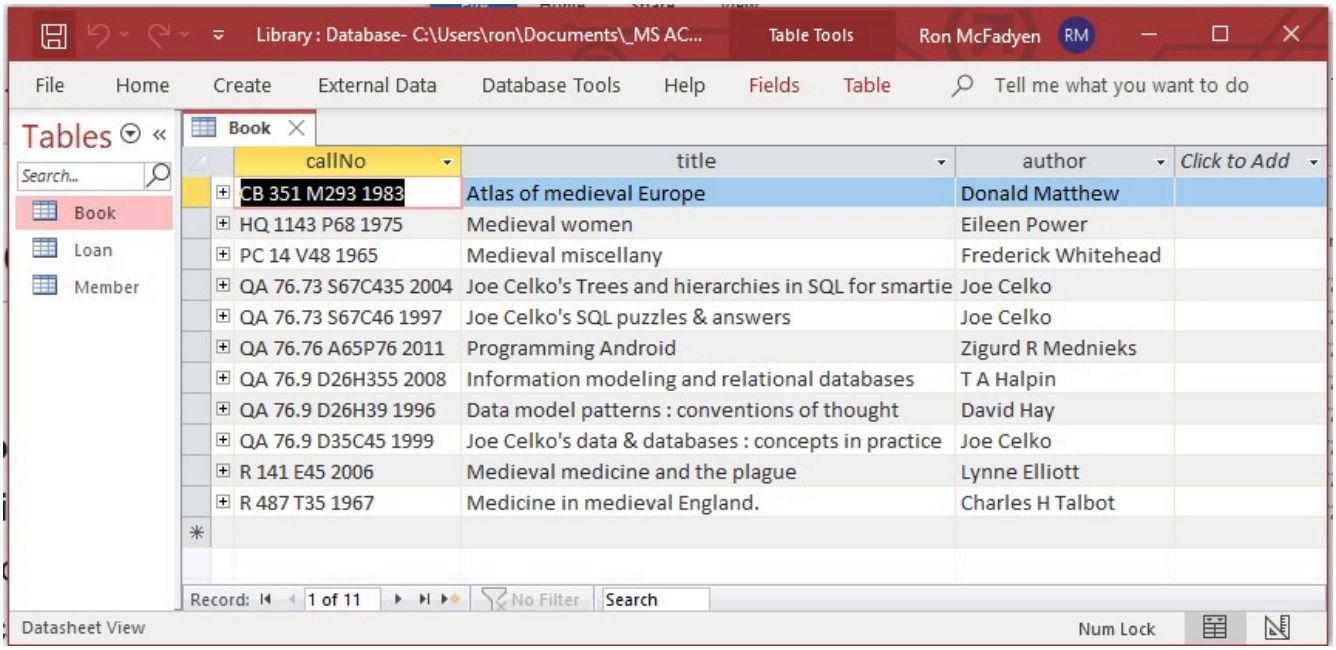


Figure 1.7: Datasheet View of a table

The Book table contains one row for each book in the library. We can verbalize the content of a row as:

- *The book identified by call number … is titled … and is authored by …*

Substituting actual values from rows we can make explicit statements such as:

- The book identified by call number *PC 14 V48 1965* is titled *Medieval miscellany* and is authored by *Frederick Whitehead*
- The book identified by call number *QA 76.76 A65P76 2011* is titled *Programming Android* and is authored by *Zigurd R Mednieks*

Knowing that books are identified by their call number and since the above statements use the conjunction 'and', the above verbalization can be expressed in an elementary form as:

- *The book identified by call number … is titled …*
- *The book identified by call number … is authored by …*

Each of these expressions is considered elementary because each states one fact about a specific book. We cannot make these statements any simpler.

Of course, we can now substitute values from the table and obtain:

- The book identified by call number *PC 14 V48 1965* is titled *Medieval miscellany*
- The book identified by call number *PC 14 V48 1965* is authored by *Frederick Whitehead*
- The book identified by call number *QA 76.76 A65P76 2011* is titled *Programming Android*
- The book identified by call number *QA 76.76 A65P76 2011* is authored by *Zigurd R Mednieks*

At this point, expressing verbalizations this way may seem trivial and unnecessary, but they do serve a purpose – they make it clear that the title and the author's name serve only to describe a book, and that the call number identifies the book. An aim of a database designer is to understand data requirements in terms of these elementary forms.

We will have more to say about this in a later chapter.

Up to this point we have seen how to:

- open an MS Access database;
- recognize the database comprises a number of tables;
- open a table to see it displayed as a collection of rows and columns;
- verbalize the information in a table.

Next, we will examine the basic features of MS Access that allow us to change, insert, and delete data.

Exercises

Recall that an elementary verbalization is one where the verbalization cannot be simplified in any further way. Simpler statements would result in a loss of information.

1) Rewrite the verbalization for the *Employees* table using elementary verbalizations.

2) Is the verbalization given for *Circulation of Leading U.S. Magazines* in elementary form?

3) What elementary verbalizations apply to the Loan table in the Library database?

4) What verbalizations apply to the Member table in the Library database?

5) View the data in the Loan table. Each row in the table corresponds to a member borrowing a book. Notice how the call number field contains values that appear in the Book table and how the id field contains values that appear in the Member table. All rows have a value for the date borrowed field. Why would some of the date returned fields appear to have no value at all?

The web site for these notes has a number of databases. Download the University database and examine its contents. This database contains information about departments and courses in a fictional university. Typically, a university is organized into faculties which comprise departments and those departments offer courses. For instance, many universities have a Faculty of Science which itself may contain departments such as Mathematics, Statistics, and Physics. Each of these departments will offer courses for students to take: Introduction to Calculus, Introduction to Statistics, Discrete Mathematics, etc.

1.2.1 Modifying Rows

With the Book table open in MS Access and with the cursor positioned in a row, try modifying the data recorded for that book. If you position the mouse cursor you can change the value recorded for the book's call number, title or author. Try doing this – remember you can always download this database again if you wish to get back to what you started with. As you begin modifying a value (e.g., adding an 's' to make the last name Matthews) an editing symbol appears to the left of the row:

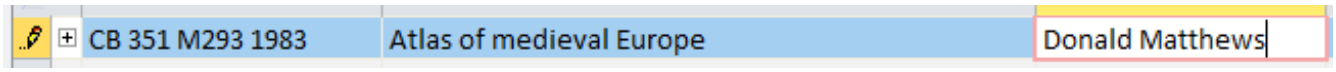


Figure 1.8: Editing a row

If you recognize that you are making a mistake you can undo your editing action by pressing the Escape key.

To make your change permanent you must move the cursor to another row for the update to be completed – when you do this, you will note the editing symbol disappears.

In some situations, you will find MS Access provides a formal Undo capability. Consider the following figure that shows an Undo icon in the upper left corner that appeared after Matthew was changed to Matthews and the cursor moved to the next row:

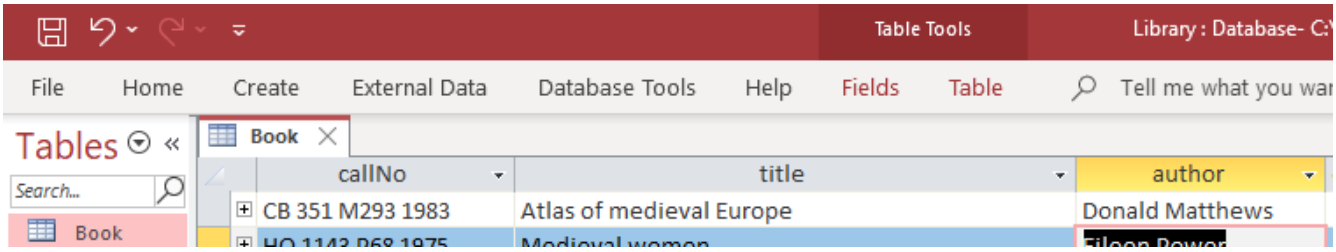


Figure 1.9: The Undo icon – click it and the last action is undone

1.2.2 Adding New Rows

Try adding a new book to the Book table. You can add a new book by first clicking on the New Record button shown near the bottom of the window:

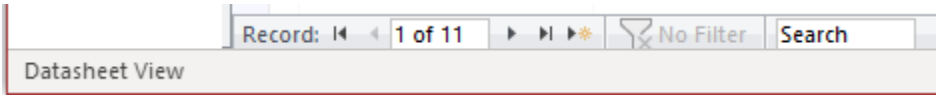


Figure 1.10: To add new record

To complete the action of adding a new book you must type values for callNo, title, and author. As a first example use a call number that does not appear for any other book. As we will soon see the Book table is designed in such a way that each book must have a different call number. Your addition will be successful if your book is given a call number that no other book has. When you add a new row, you must move the cursor out of the row for the addition to be completed.

As a second example try to add a new book, but this time, use a call number that already appears in the table. In this case MS Access will reject your new record. Try this and you will see a response similar to:

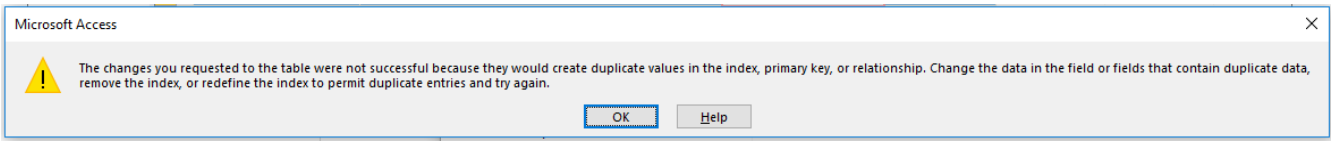


Figure 1.11: MS Access message for duplicate primary key value

The important part of this message for us is the part that refers to *duplicate values* or *duplicate data*. When we try to add a row with the same call number as some other row MS Access refers to the duplicated call number value. Note that you can press the Escape key to remove the new row from the table display. Soon we discuss table design where you will see that the call number field is designed to be the *primary key* of the Book table.

Adding a row to a table is also referred to as inserting or appending a row.

1.2.3 Deleting Rows

You can remove a book from the table by highlighting a row (click in the cell just to the left of a call number) and then press the Delete key on the keyboard:

Figure 1.12: Delete a record: select, press delete

When you press the Delete key MS Access will respond in one of two ways depending on whether or not there is an existing reference to the row you are trying to delete:

A reference to the book does not appear in the Loan table:

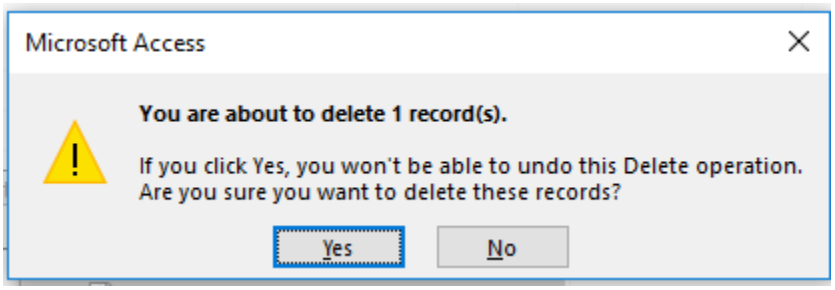


Figure 1.13: Prompt message

A reference to the book does appear in the Loan table:

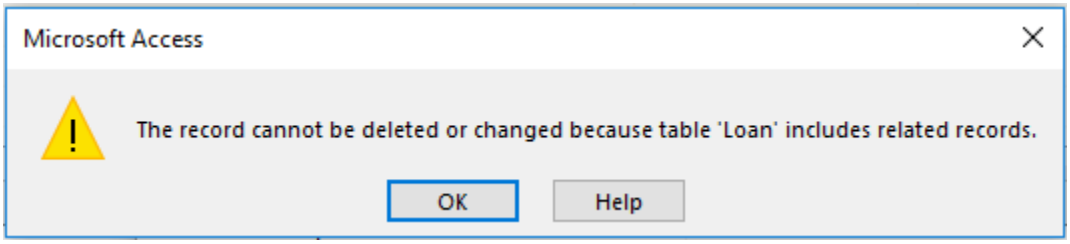


Figure 1.14: Prompt message

When you view the Loan table you can see the books that library members have taken out and whether a book has been returned. Rows in the Loan table have references to rows in the Book table and to rows in the Member table. The default action in MS Access is that a deletion is disallowed if there is some row in a table that has a reference to it. So, we cannot delete a book if there is a Loan row referencing it.

We have briefly shown how to modify, add and delete data in tables. Next, we introduce the design perspective for tables.

1.2.4 Table Design View

Up to this point we have been opening tables in Datasheet View where we can view and change data in rows of a table. When in Datasheet View we can switch from datasheet view to *Design* View by right-clicking on Book and choosing Design View (see figure 1.13). When the Design View icon is clicked, the display changes to reveal design information (see figure 1.14).

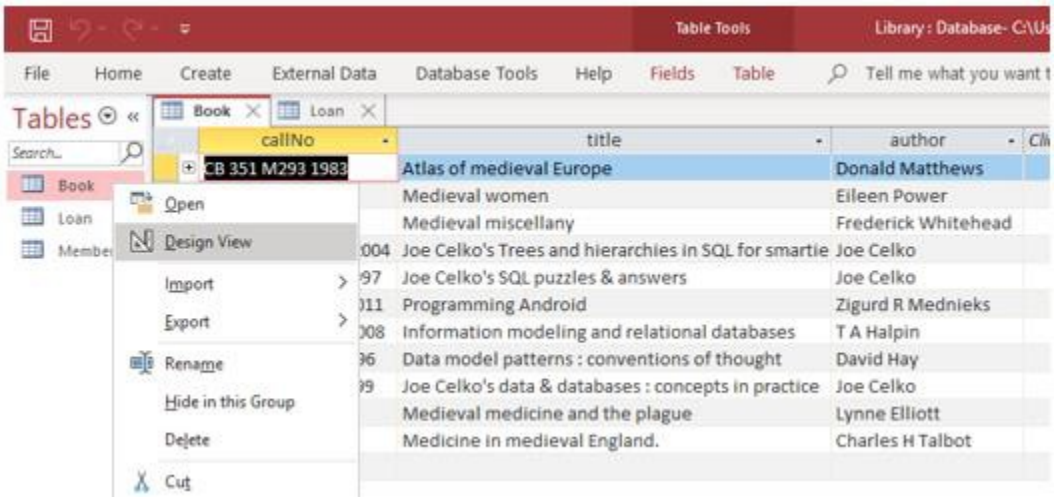


Figure 1.15: Click on the Design View Icon to Switch to Design View

When you click the Design View icon you will see the display change as shown in figure 1.14 – you will see the field names listed along with their *datatype*, and according to the field where the cursor is located you see other *properties* for that field. Datatypes vary somewhat from one database system to another, but of course there are many similarities too. Properties are other characteristics that you can define for a field such as the maximum length of values stored for the field.

Generally, we want data in a database to be reasonable and correct. We can use datatypes and properties to achieve certain types of correctness. Consider the following integrity rules as rules we would like to enforce:

- Call numbers, titles, and authors are alphanumeric. Any text you can type on the keyboard is acceptable.
- Each call number must be unique (there can be no duplicates)
- Each book must have a title
- A value for call number must be no more than 50 characters long
- A value for title must be no more than 255 characters long
- A value for author must be no more than 255 characters long
- The author field can be left out (it can be *null*).

Now we discuss how these integrity rules are obtained in Table Design View.

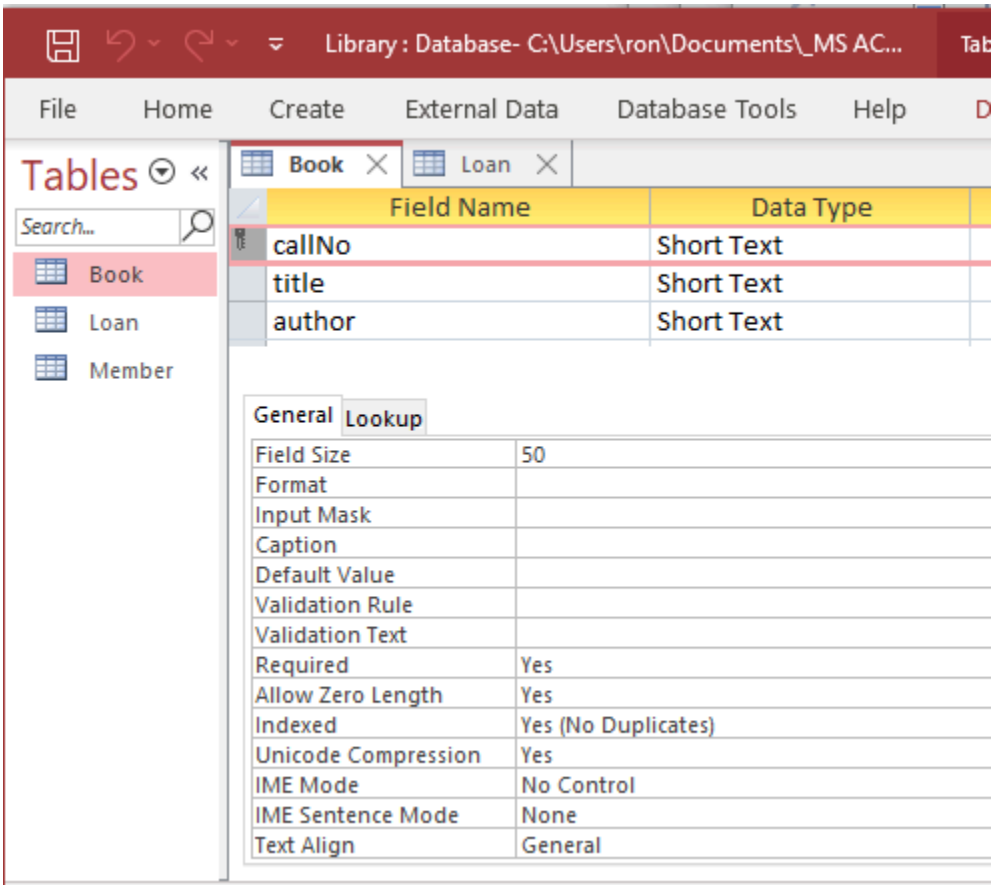


Figure 1.16: Table Design View

In figure 1.14 the cursor is located on the callNo field; some properties of callNo are circled and discussed below:

- Beside the callNo field you can see the *key* icon. This means the callNo field is the *primary key*. A primary key is a unique identifier – every row in the table must have a unique value in that field. Every table should have a *PK* specified and there can be only one PK for a table. When a field is defined as the PK then a value must be provided in each and every row.
- The callNo field has a datatype of *Short Text* and a field size of 50. Any value you can type on the keyboard is acceptable but the overall length, number of characters, is restricted to at most 50.
- The callNo field is *indexed* and in this case no duplicates are allowed. The index constructed by MS Access is similar in purpose to the index at the back of any book: the index allows MS Access to quickly locate a specified row. However, this index is different from that at the back of a book because it allows only one entry per indexed value (*No Duplicates* is specified for the *Indexed* property). Each call number is unique.

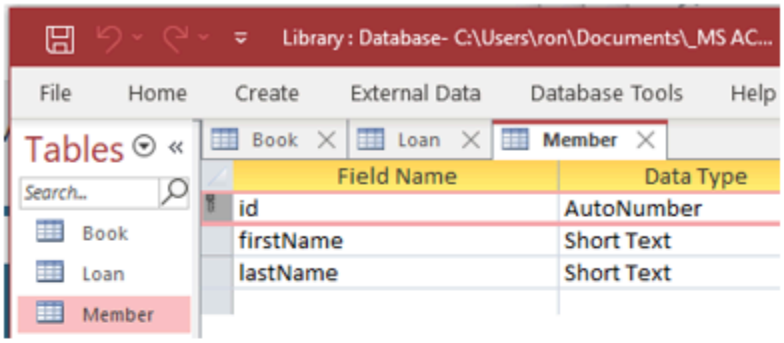
As you move the cursor up and down you should note the following for this sample table: For title:

- The title field has a datatype of *Short Text* with a field size of 255. A text field can comprise any combination of letters, digits, and punctuation. Any value entered by a user cannot exceed 255 characters in length.
- A value *is required*. When entering data for some book, the user cannot omit the title.

- There is no *index* on title. For author:
- The author field has a datatype of *Short Text* with a field size of 255. A text field can comprise any combination of letters, digits, and punctuation. Any value entered by a user cannot exceed 255 characters in length.
- A value is *not required*. When entering data for some book, the user can omit the author.
- There is no *index* on author.

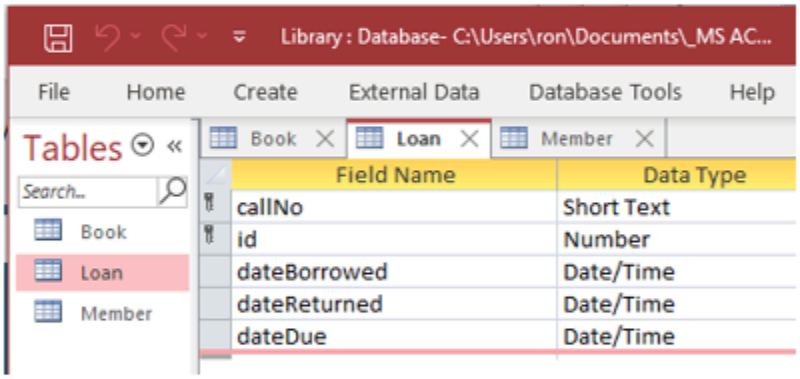
Now, open the Member table and then the Loan table in design view. Examine the properties of each field. For reference see figure 1.15.

Member Table



Field	Datatype
id	Autonumber: MS Access will generate a unique number for each row. The user cannot enter id values.
firstName	Short Text
lastName	Short Text

Loan Table



Field	Datatype
callNo	ShortText
id	Number: numeric values must be entered by the user. This datatype is a proper match to the Autonumber datatype.

The callNo and id fields both have the key symbol beside them. This means that both together form the primary key, and so combinations of these two field values must be unique: there can be several rows with the same call number and several rows with the same user id, but any pair of call number and id values can only appear in a single row. This type of primary key is called a *composite* primary key.

dateBorrowed	Date/Time: the user must enter a date
dateReturned	Date/Time
dateDue	Date/Time

Figure 1.17: The fields of the Member and Loan tables.

Later we will examine datatypes and properties in much more detail.

Exercises

1) Use Design View to add fields to the Member table as indicated:

- gender: Short Text field of length 6 to accommodate the values *male* and *female*. Make this a required field that is not indexed.
- birthDate: A Date/Time field; required; not indexed.
    - Switch back to Datasheet View (You must reply **yes** to the system prompts to save your changes). You should notice there are no values for gender nor birthDate.

2) Now enter values you deem appropriate in the gender and birthdate fields for each member. Close the table and reopen it. You will see the values you entered are still there.

3) When new members join the library, information about them must be entered into the Member table. Each member is given an id value automatically. Add new members to the library and note how MS Access will not let you enter id values; instead, MS Access generates those values for you – id values are generated sequentially. Close the table and then reopen the table to confirm your additions worked.

4) In exercise 3 you added a new member and in exercise 4 you added fields to the Loan table. Consider that the person you added now borrows a book and so a row must be entered into the Loan table. Enter such a row.

5) Typically, a library assesses a fine the user must pay if they keep a book out past the due date. As well the library needs to track the amount, if any, the member has paid. In this exercise we add two fields to the Loan table so we can keep track of fines that are assessed and the amount the member has paid.

- Open the Loan table in design view and add two new fields named *fineAssessed* and *finePaid*. These fields must have a data-type of Currency.
- Save the Loan table and then view the rows of the table. There are no amounts for these fields.
- Choose some row(s) in the Loan table and enter values for the fineAssessed and finePaid fields. Note the values you enter will appear as dollars and cents.

6) After successfully entering data for exercises 3, 4, and 5 you are aware of a member and a book for which there are references in the Loan table.

- View the Member table and try to delete that member, and then view the Book table and try to delete that book. These deletion attempts are unsuccessful because of the references to the Loan table.
- Now open the Loan table and find the loan record you entered in exercise 5. If you delete this row in Loan, then you will find that you are able to delete the member that was referenced by that row (provided you did not enter more loans for this person). These actions mirror the way in which data would typically be deleted from a database: if you want to delete a row you must first delete (or modify appropriately) any rows that reference it.

2. CREATING TABLES

The typical MS Access database comprises several kinds of database objects such as indexes and tables. Each table represents a kind of entity (persons, places, things, events, etc.), or relationship between entities. For instance, if we are keeping track of departments and courses at our University then we should have two tables:

- Department: to keep information about departments
- Course: to keep information about courses.

For each department suppose we need to know things such as: department code, department name, location of the department (an office number), phone number for the department, and the name of the department's chair. Suppose departments can be identified by their department code (e.g., ACS) and by their department name (e.g.

Applied Computer Science); both of these fields are assigned by the University and each will be unique across departments. We will choose to use the department code as the primary key; that is, we choose to use department code as the primary identifier for departments. We show Department with some sample data:

	deptName	**deptLocn**	**deptPhone**	**chairName**
ENGL	English	3D05	786-9999	April Jones
MATH	Mathematics	2R33	786-0033	Peter Smith
ACS	Applied Computer Science	3D07	786-0300	Simon Lee
PHIL	Philosophy	3C11	786-3322	Judy Chan
BIOL	Biology	288	786-9843	James Dunn

Figure 2.1: Department table

Suppose the Course table keeps track of courses offered by the University and includes the fields: course number, title, short description and credit hours. At the University, what is a course number? The ways of identifying courses varies from one institution to another, but a common way is to give the department code followed by the course number, such as "ENGL-2221"; "ENGL-2221" comprises two parts: a department code and a course number (the dash is just there for formatting purposes). We will use this convention and so we must include department code as a field in the Course table, and the combination of department code and course number serve as a unique identifier (i.e., together they form a *composite* primary key). We show this structure with sample data:

deptCode	courseNo	title	description	creditHours
ACS	1453	Introduction to Computers	This course will introduce students to the basic concepts of computers: types of computers, hardware, software, and types of application systems.	3
ACS	1803	Introduction to Information Systems	This course examines applications of information technology to businesses and other organizations.	3
ENGL	2221	The Age of Chaucer	This course examines a selection of medieval poetry and drama with emphasis upon Chaucer's Canterbury Tales.	6
PHIL	2219	Philosophy of Art	Through reading key theorists in the history of esthetics, this course examines some of the fundamental problems in the philosophy of art, including those of the definition and purpose of art, the nature of beauty, the sources of genius and originality, the problem of forgery, and the possible connection between art and the moral good.	3
BIOL	4451	Forest Ecosystems Field Course	This is an intensive three-week field course designed to give students a comprehensive overview of forest ecology field skills.	2
BIOL	4931	Immunology	Immunology is the study of the defence system which the body has evolved to protect itself from external threats such as viruses and internal threats such as tumour cells.	3

Figure 2.2: Course table

2.1 Using Design View To Create Tables

In this section we will step through the process of creating a table. From the web page for these notes download and open the MyUniversity Database.

- Click on the Create tab – use the mouse and click it to see the options available (*Table*, *Table Design*, *SharePoint Lists*, etc.):

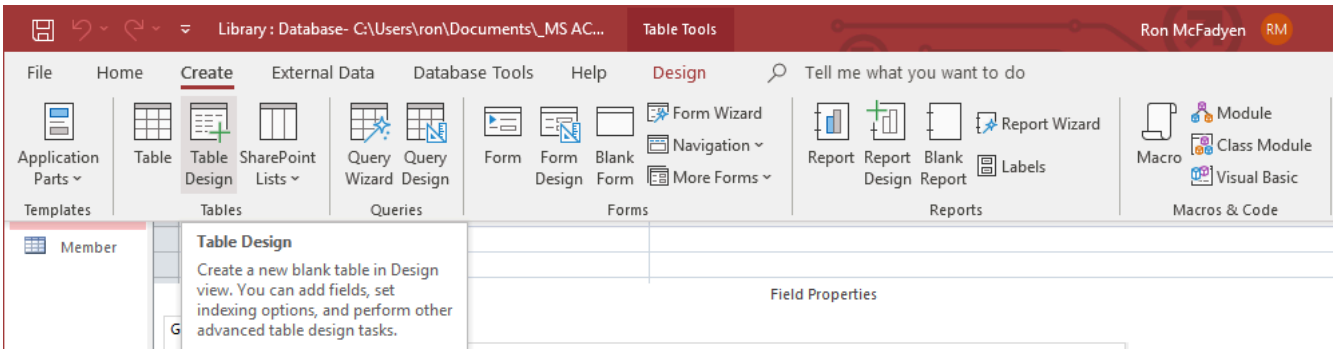


Figure 2.3: Create a table using Table Design

- Click the *Table Design* option to begin a process to create a new table, the Department table. You are presented with a form where you can enter field definitions for a table. A field definition comprises field name, data type, and description. You can also set the table's primary key. The Table Design form is shown below:

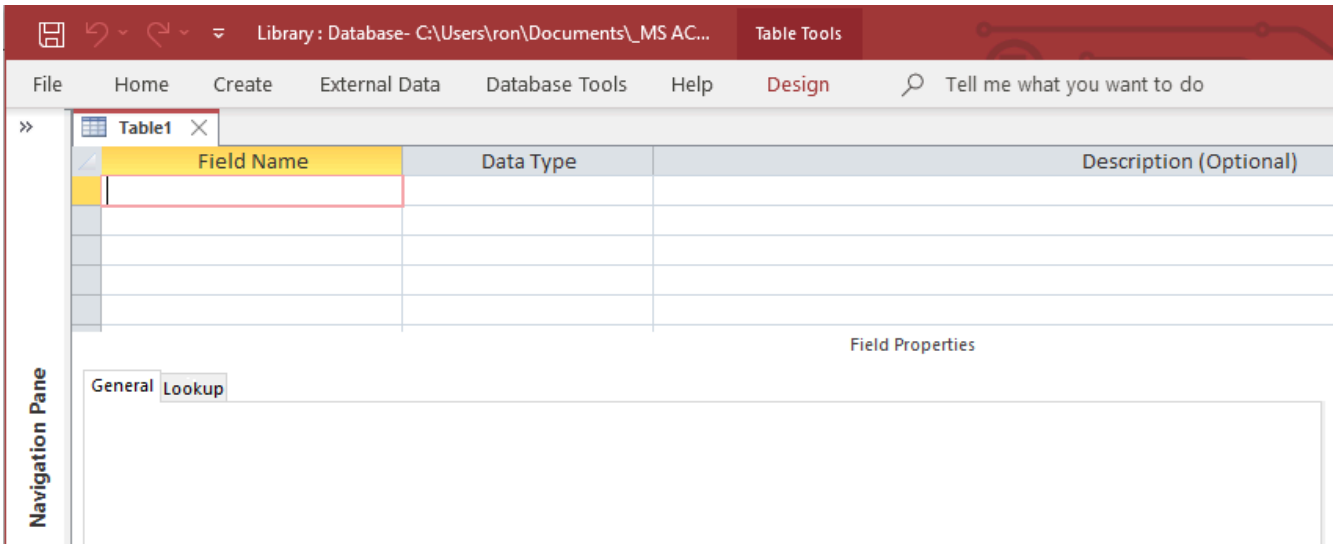


Figure 2.4: Table Design View

- Begin by entering each field name and choosing Short Text as the data type. In the description column you may enter a longer description of the field's contents – these are treated as comments

that may be useful for someone who is learning about your tables. Once you have done this you should have a form that looks like:

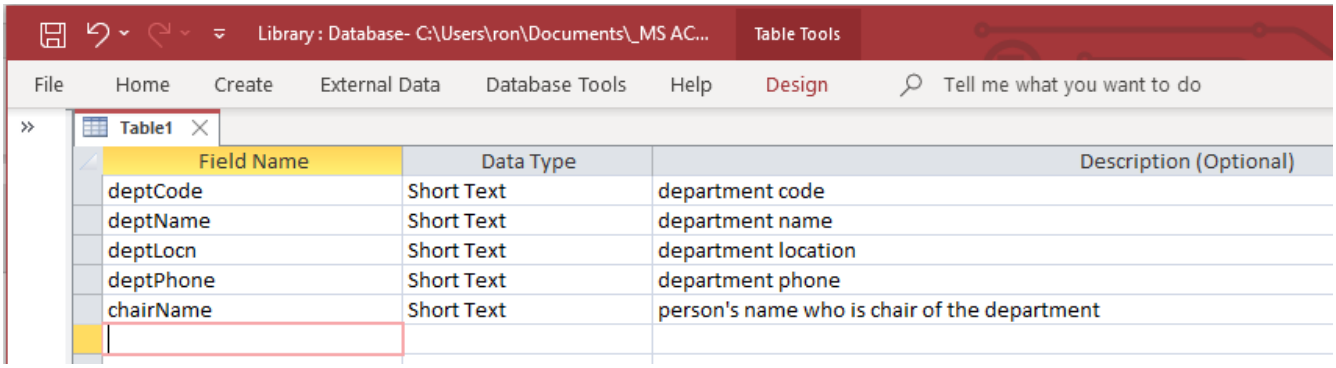


Figure 2.5: Department Table

- Next, we set the primary key for the table to be the deptCode field. Right-Click the mouse in the spot just to the left of deptCode and then click the Primary Key. Access uses an icon to show the deptCode as the PK:

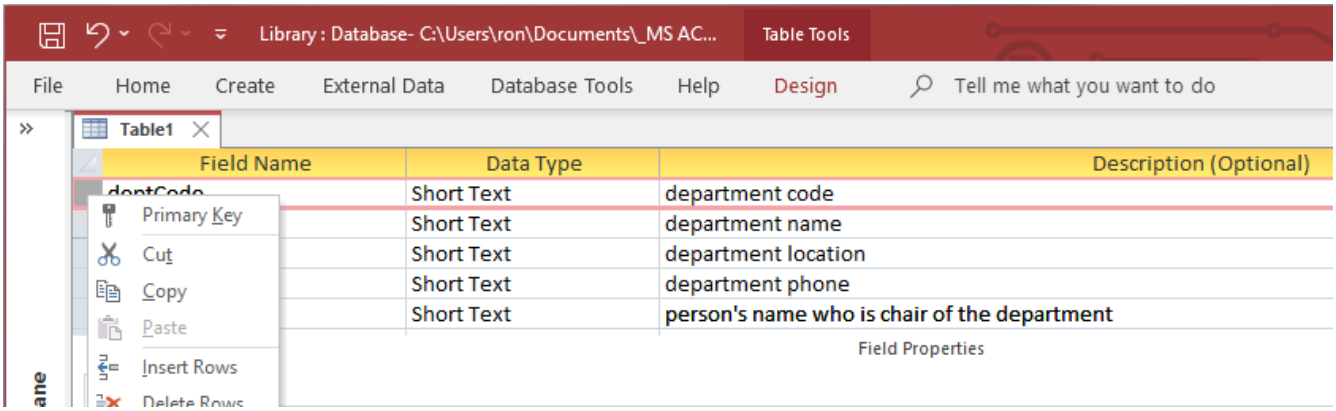


Figure 2.6: Setting the Primary Key

- At this point you should save your work by clicking the Save icon in the upper left- hand corner of the form – you will be prompted to give the table a name – name it Department.

You should still be in Design View for the Department table. Note that you can press the F1 function key to get help pertinent to the location of the mouse cursor. If your cursor is positioned on a Field Name and you press F1 you will see a window pop open that displays suggestions from MS Access regarding how you should name fields. Try this. Before going any further, try pressing F1 in other locations too, such as Data Type and Description. We recommend that you read some of the information available to become more familiar with MS Access.

2.1.1: Data Types

MS Access provides several data types – we will discuss Short Text, Long Text, Number, Large Number, Date/Time, Currency, AutoNumber, Yes/No, Calculated, and Lookup Wizard.

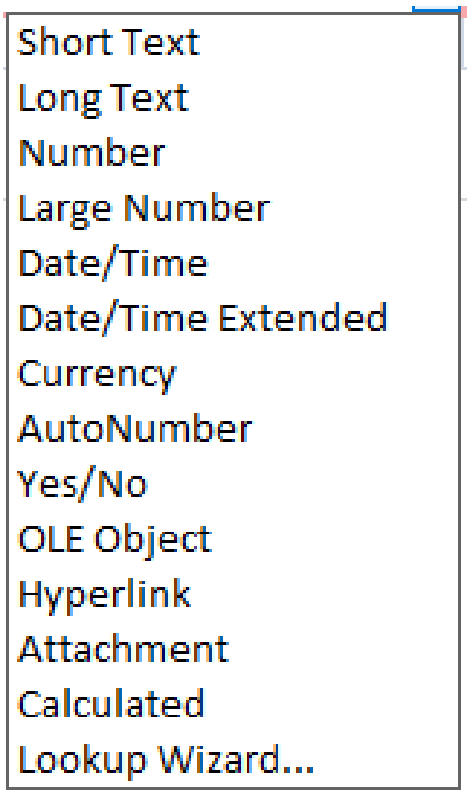


Figure 2.7: MS Access Data Types

Short Text

If you specify that a field has the **Short Text** data type, then Access will permit any characters to be placed in that field in a row of the table. This is a common choice when the data will not be used in calculations. The Text data type provides for values that have fewer than 256 characters. If you know that a maximum length less than 255 would be appropriate, then you could use the Field Size property (discussed in the next section) to limit the maximum length of a text string.

Long Text

A designer selects **Long Text** if the field will have character data, but the length might be longer than 255. Long text allows for a maximum length of 63,999 characters. Consider the description field of the Course table: could these be longer than 255 characters?

Number

If a field is used for storing values that are used in numerical calculations and if the values will small enough, then Number is appropriate. Number values occupy 4 bytes of memory and have a range of -231 to 231-1.

Large Number

If a field is used for storing values that are used in numerical calculations and if the values can be huge, then Large Number is appropriate. Large number values occupy 8 bytes of memory and have a range of -263 to 263-1.

Date/Time

If a field contains date and or time values, then the Date/Time data type should be chosen. The Format property (discussed later) allows you to control how these values will appear to the user.

Currency

If a field will contain monetary values, then the Currency data type should be chosen. This data type provides for numeric calculations that are accurate to 15 digits to the left of the decimal and 4 digits to the right of the decimal.

AutoNumber

If you choose AutoNumber, then MS Access will generate a value for you when a row is inserted into the table. You can, via the New Values property, arrange the numbers to be generated sequentially or randomly. Often control numbers for things like orders, invoices, registrations, etc. are numeric and we can leave it to the system to generate a next value for us.

Yes/No

This data type restricts possible values to yes or no.

Calculated

A calculated field is one that has a derived value determined by some function.

Lookup Wizard

Sometimes you need to restrict values to a list of known values (e.g., a list of genders: Male, Female), or to values appearing as primary key values elsewhere in the database. Consider the creditHours field – a suitable list could be (1, 2, 3, 6). The Lookup Wizard is a suitable data type for these situations; when selected the system steps you through a series of windows where you can make the appropriate choices.

Exercises

These exercises refer to the Library database.

1) Consider the Book table. Add a field, *paperback*, that can be used to indicate if a book is a paperback. Choose the YES/NO datatype. Save the design and switch to datasheet view. Now you will

see how to enter such values – MS Access provides a box that is to be checked, or not. You can select (a 'Yes') using the mouse or by using the space bar. You should experiment with this.

2) Consider the Member table. The *id* field was defined with the AutoNumber datatype. Experiment by adding new members and you will note that id values increase by 1. Now try deleting the last two members that you added. If you add those members back in, what id values do they get? Are id values reused?

3) Consider the Member table. Previously you added a *gender* field. Open the Member table in design view and change the datatype for gender to be Lookup Wizard. The wizard will automatically present 3 successive popup windows where you will:

- specify that you are providing the lookup values;
- enter the values (Male and Female);
- specify that values are to be limited to your list.
    - Save the table and enter datasheet view so you can test out the datatype you have just created. You will notice the user sees a drop-down list containing Male and Female, and so the user cannot enter/select an inappropriate value.

2.1.2: Properties

Each field must have a data type as discussed above. According to the data type, MS Access will present to you a set of field properties that you can tailor for your table. We will discuss the following: Field Size, Format, Input Mask, Caption, Default Value, Validation Rule & Validation Text, Required, Indexed, Show Date Picker, and New Values.

Field Size

Consider a field like deptCode. Suppose the University uses 3-character and 4-character values for department codes. Because of this it is reasonable to set the Field Size to 4, in order to limit the possibility that an end-user accidentally types a longer string of characters and thereby enters incorrect data. In this way we can limit the kinds of errors users make when they enter data and thus improve the overall quality of our database.

Data integrity is a serious issue for databases. Setting Field Size for Text data and Number data is a common thing to do. Often organizations will limit the data they collect for fields such as last name and first name (for example, 30 characters). If the data type is Number, then values selected for Field Size are values such as Byte, Integer, Long Integer, etc.

These kinds of values are associated with increasing number of memory locations used per value. A selection of Byte restricts storage to 1 byte of memory (8 bits), and since the largest positive integer that can be stored in a byte is 255, the values stored in the field are forced to be in the range from 0 to 255. Further information is readily available if you use the F1 function key on Field Size for a Number data type.

Format

The Format property is used to customize the way text, number, dates, and times are displayed to the end user. For instance, selecting Medium Date causes values like January 14, 2013 to be displayed as 14-Jan-13; selecting Long Date results in the display January- 14-13. See figure 2.8 for examples. If you have Text data such as department code, then you could force the display to be in capital letters by specifying > as the format code. An interesting Format specification is *@;None*. If this is used and if there is no value at all to display, then the word *None* will be displayed to the user. Another example: suppose the field is for the Canadian SIN. You may have seen these displayed to users with hyphens between the 3rd and 4th digits and the 6th and 7th digits. If the SIN is a Text field of length 9 it can be displayed this way by using a Format specification of @@@-@@@-@@@.

Value in field	Format Property	Displayed as
barack obama	>	BARACK OBAMA
January 14, 2013	Medium Date	14-Jan-13
January 14, 2013	Long Date	January-14-13
	@;None	None
786456789	@@@-@@@-@@@	786-456-789
	@@@-@@@-@@@;*None*	None

Figure 2.8: Format Examples

Input Mask

The Input Mask property is used to force the user to add data according to some pattern. This is another nice feature to help improve the overall quality of data added to a database. When the cursor is in the Input Mask area a 'builder button' appears. When you click this button you will see a list of popular controls. If you were to choose the mask for phone number you will see the control *!(999) 000-0000* appear. As a result of this choice, the user must enter a 7-digit phone number with an optional 3-digit area code).

Caption

If there is no caption, then the heading used in displays of data is the field name. Sometimes the field name is not what you want your users to see. For example, instead of the heading *deptCode* above a list of department codes, you may prefer to use the words *Department Code*. To accomplish this just enter such a heading in the caption property for the field.

Default Value

If some value for a field is very common, then you should consider setting a default value. For example, if most courses are 3-credit hour courses, then the value 3 can be set as the default for all new courses.

Validation Rule & Validation Text

If a field has a validation rule, then the rule is tested whenever the user enters data. If the test fails the user is prompted with a message containing the validation text. A simple use of this could enforce the credit hours to be less than 10 by entering the rule *<10* and the validation text *Please enter a value between 0 and 9*. Again, this is a nice feature to improve overall data quality.

Required

Consider the deptName field of the Department table. If a user enters data for a new department then it is unreasonable for the deptName field to not have a value. To ensure there will be a value we make the field required – i.e., we choose *Yes* for the Required property.

Indexed

MS Access automatically creates an index (unique – no duplicates) on a field that is the primary key. A unique index is a special internal data structure that Access builds to facilitate two things:

- to ensure fast access to rows of data when the user specifies a value for such a field in a query, and
- to ensure in the case of *no duplicates* that no two rows of the indexed table could have the same value for that field.

The index data structure is very similar to the index you see at the back of books. An index comprises several entries where each entry has a value (a term used in the book) and a reference (a page number in a book) – in the case of *duplicates allowed* there can be several references (several pages where the term appears).

You may choose to have an index on any field. If a field could have duplicate values, then you must choose an index that allows duplicates.

Show Date Picker

If the data type is Date/Time, then this selection enables the user to select a date using a *picker* – a convenient tool for data entry.

New Values

If the data type is AutoNumber, you can use New Values to specify whether the next value for the field will be the next highest integer, or if it will be a random integer.

Exercises

In the next two exercises you are working with your University database:

1) Consider the Department table. In design view, set the deptCode field to have a length of 4 and use > as the display format. Set the length of the deptPhone field to be 10 and choose the Phone Number input mask. Save the table and switch to datasheet view. Use figure 2.1 as a guide and enter data into the Department table.

2) Create a Course table with attributes for department code, course number, title, short description, and credit hours. The credit hours field should be numeric with no decimal places, and the other fields are Text fields. Set the deptCode field to be Short Text with a length of 4 so that it matches the properties of deptCode in Department. Later it will be important that the deptCode field in both Department and Course are defined the same. Use figure 2.2 as a guide and enter data into the Course table.

The following exercises relate to the Library database:

3) Consider the firstName and lastName fields in the Member table of the Library database. Modify the caption for these fields to be First Name and Last Name respectively. Save the table and reopen in datasheet view. You will see these captions at the top of their respective columns.

4) The Loan table has fields that are defined with the Date/Time datatype. Experiment with different formats for these dates.

5) Consider the id field in the Member table of the Library database. In design view change the increment property of the id field to be *random* instead of *increment*. This is a non-reversible action (but you can download the database later to get a fresh copy). Now add some new members. What can you say about the id values that are assigned?

6) Validation rules and validation text are important features to assist database users.

- Consider the Loan table and its date fields. MS Access has many built in functions one of which is Date(), which always returns today's date. So, to ensure that someone always enters a due date later than today, in the properties section for the date due field enter the following:

    - Validation rule:>= Date()
    - Validation text:Enter a future date.

- In this situation we are entering a field-level validation rule. These rules are useful when we can state a requirement independent of other fields. Test the effect of this validation rule by switching to datasheet view and entering valid and invalid values for the due date.
- To ensure the date borrowed value is less than or equal to the date returned value we construct a validation rule that involves two fields. MS Access will not let you enter this rule at the field level; instead, such a rule must be specified at the table level. To enter a table level rule, you can click *Design* and then *Property Sheet* as indicated in figure 2.9a.

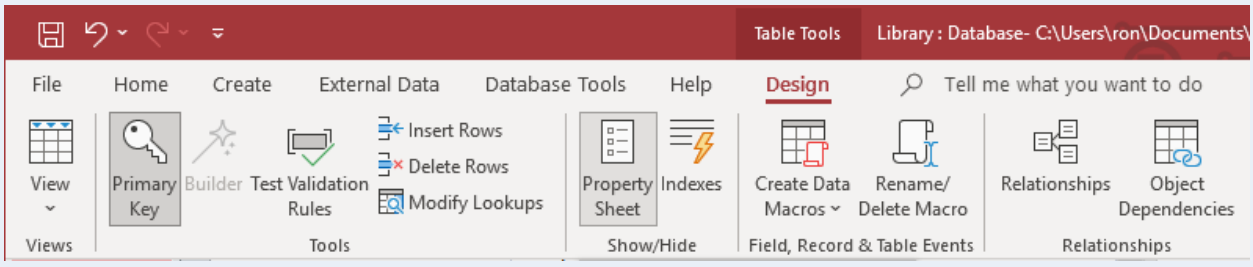


Figure 2.9a: Table level properties

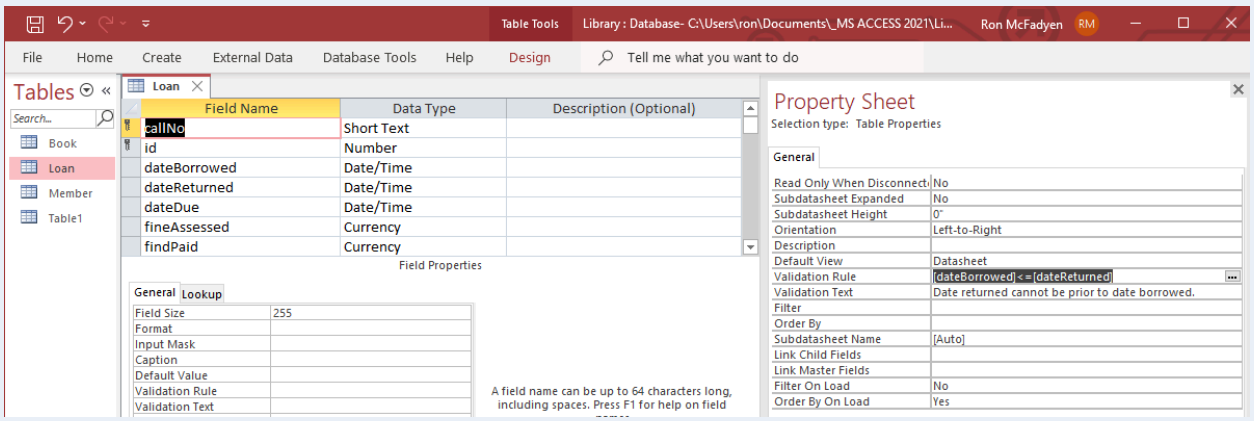


Figure 2.9b: Enter a validation rule

- And, as shown in figure 2.9b enter the properties:
    - Validation rule:[dateBorrowed]<=[dateReturned]
    - Validation text:Date returned cannot be prior to date borrowed.

- The square braces, [ ], that appear in the expression are required. These inform MS Access of references to fields in the table.
- Enter this rule and verify that it prevents a user from entering improper dates.

2.1.3: Primary Keys

This section assumes you have created the Department and Course tables in the MyUniversity database. Every table should have a primary key, but this is just a rule-of- thumb that most database designers follow. In our database:

- The Department table has deptCode as its primary key.
- The Course table has a composite primary key – a key formed using two attributes: deptCode and courseNo.

To set a primary key the table must be open in Design View. You must first select the field (or combination of fields) and then click the Primary Key icon. This is straightforward for the Department table, but not for the Course table because its' primary key comprises two fields. Because the PK involves more than one field we say this primary key is *composite*.

Exercises

1) Set the primary key for the Department table. With the Department table in Design View, select the deptCode field and then right-click and choose Primary Key. When done successfully you will see the deptCode field with a key icon beside it:

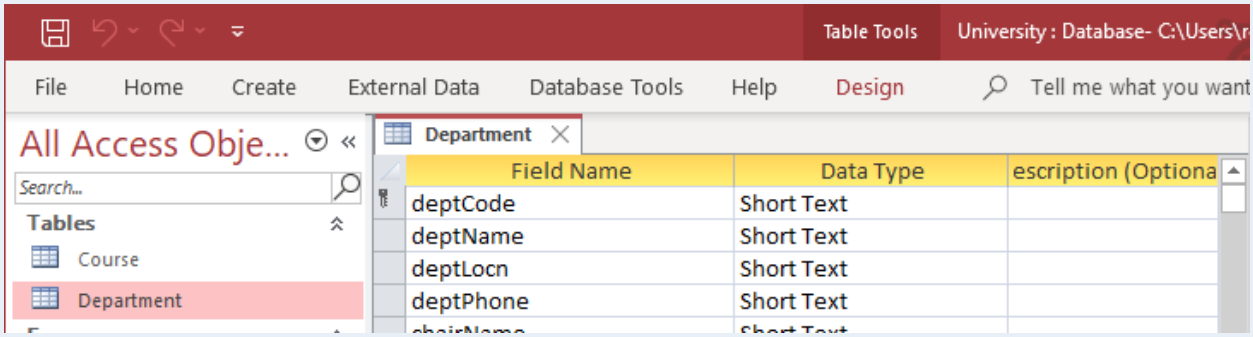


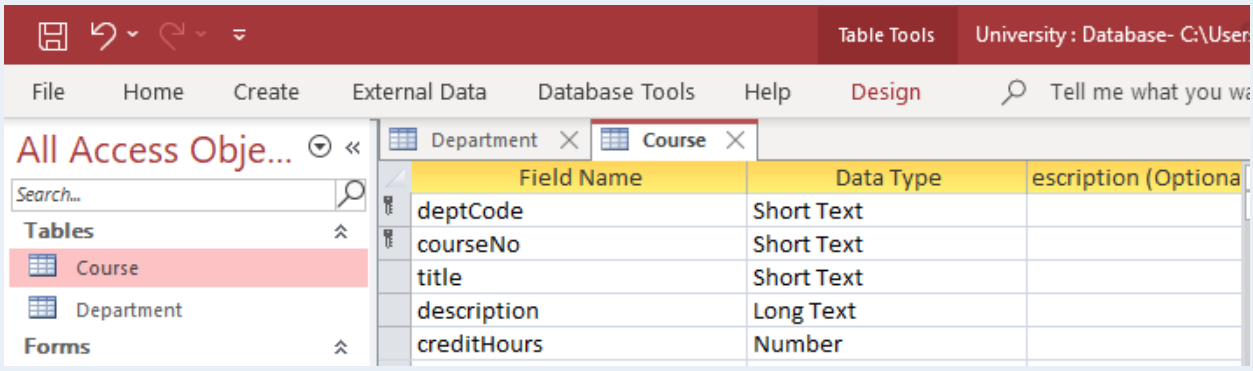
Figure 2.10: Setting the PK for Department

- If MS Access rejects your primary key, then you must examine the values you previously entered for deptCode – there must be some duplicated value. If this happens you must view the table in Datasheet View and find the duplicated value and make necessary changes.
- Once MS Access has accepted your primary key you should open the table in datasheet view and experiment: How does MS Access respond if you try to create a new row with an existing

primary key value?

2) Set the composite primary key for the Course table. To do this you first select one field, and then while holding the *Control* key down select the other field. With both fields selected and the *Control* key down, right-click and select Primary Key:

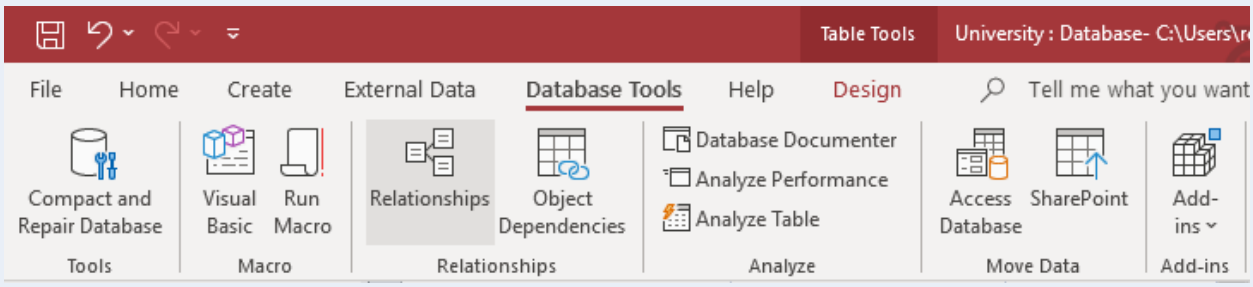
- Select the deptCode (click to the left of the deptCode field).
- To select the next field to be part of the PK: while holding the Control key down, click the courseNo field
- Now, still with the Control key down, right-click and you will see options
- Choose Primary Key. You will now see the key image beside both fields as in:



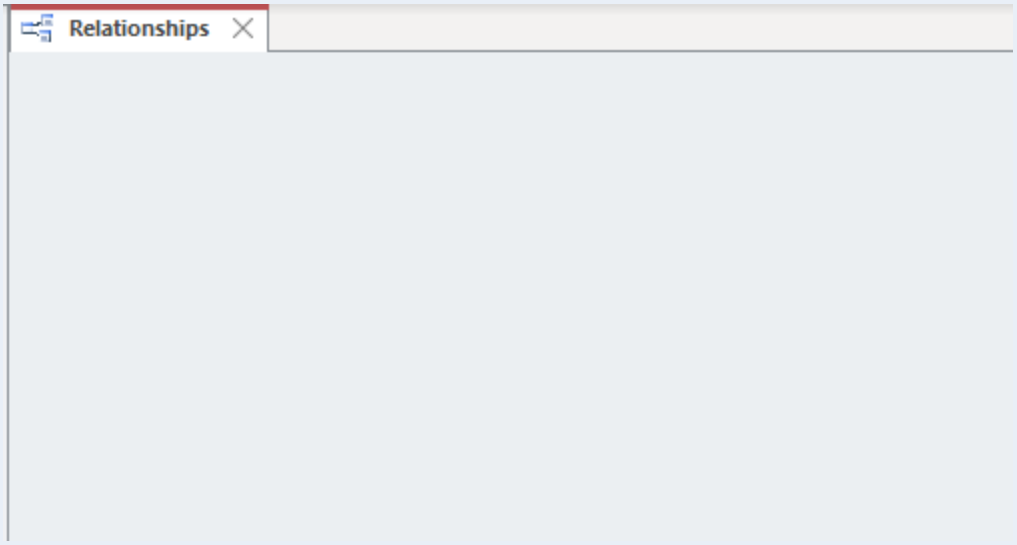
- If MS Access rejects your primary key, then you must examine the values you previously entered for deptCode and courseNo – there must be some duplicated value (two or more rows have the same pair of values for deptCode and courseNo). If this happens open the table in Datasheet View and examine the rows to find duplicate values of the combination {deptCode, courseNo}.
- Once MS Access has accepted your primary key you should open the table in datasheet view and experiment: How does MS Access respond if you try to create a new row with an existing primary key value?

3) (Advanced) Later on we discuss relationships between tables. Perhaps you are willing to try this now. The Department and Course tables are related to one another through the deptCode field. It is reasonable for us to expect that a deptCode value in a row of the Course table also appears in a row of the Department table. That is, if we are recording a course for the mathematics department then we expect the database to have a corresponding row in the Department table. To ensure this is the case we create a formal relationship between these two tables using the Relationships Tool:

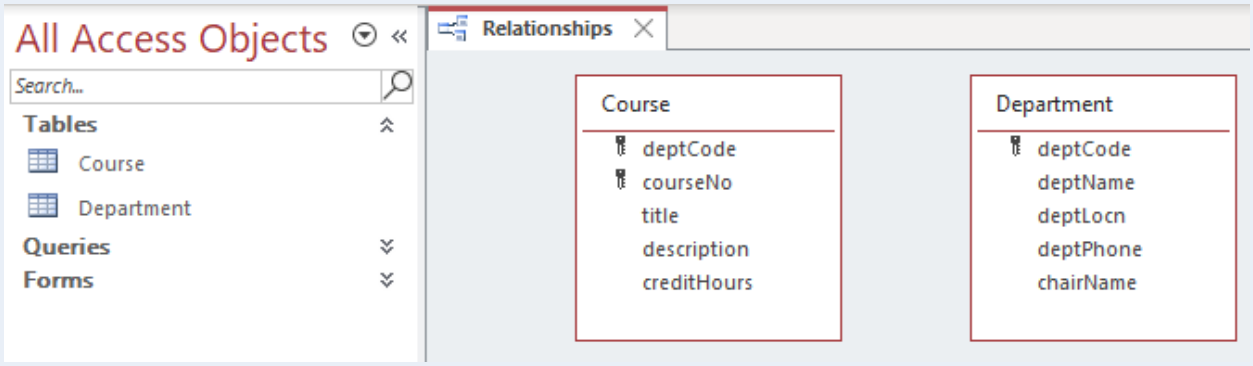
- First, click Database Tools. Then click Relationships:



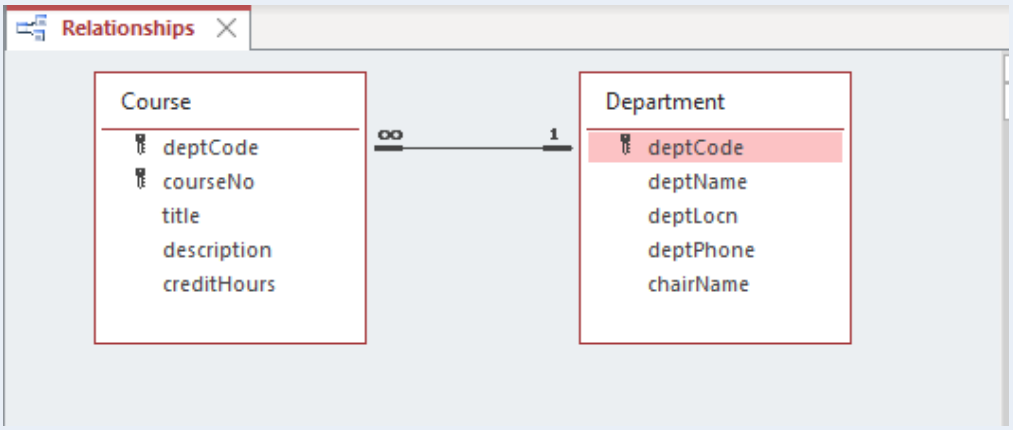
- The Relationships Tool opens and you see a blank relationships diagram:



- Drag the Department and Course tables from the Tables List to the diagram:



- With both tables showing on the diagram, you must select the PK of Department, drag it to the Course table, and release the mouse button above the deptCode field of Course. If you follow the directions on the screen you will be able to select *enforce referential integrity* (RI) and then you end up with the following:



- If RI is enforced, then it becomes impossible to have a row in Course without a matching row in Department.

You can compare your MyUniversity database to the University database provided on the web page for these notes.

3. CREATING FORMS

For each table you should create a basic form that can be used to manage data for the table. Once forms have been created your users will have a more user-friendly way of entering and managing data in your database (datasheet view is not considered user-friendly).

3.1: Using the Form Wizard

There are many ways to create forms. We discuss the simplest approach here. Click the *Create* tab and then select the *Form Wizard*:

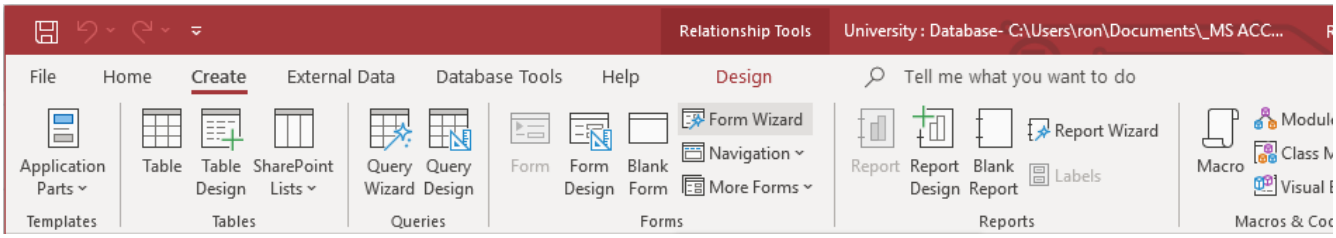


Figure 3.1: Use Form Wizard

The Form Wizard steps you through a sequence of choices where you choose the table for the form, the fields to appear on the form, the layout for the form, the style, and the title to appear at the top of the form. At this point you should create two forms in your University database, one for the Department table and one for the Course table. As the wizard steps you through each case, you should select/enter:

1. *All fields* of the table to appear on the form (try clicking the  button when it shows)
2. A *Columnar* layout
3. A *style* of your choice
4. A *title* of your choice

The last thing you do with the wizard is choose one of: *Open the form and view data*, or, to *Modify the form's design* – choose to *open the form to view data*. Note that data appears according to information you provided for data types and properties.

A major difference now is that the user will see just one row at a time. Notice the navigation buttons at the bottom of the form where the user can click to navigate through the rows of the table or to add a new row:

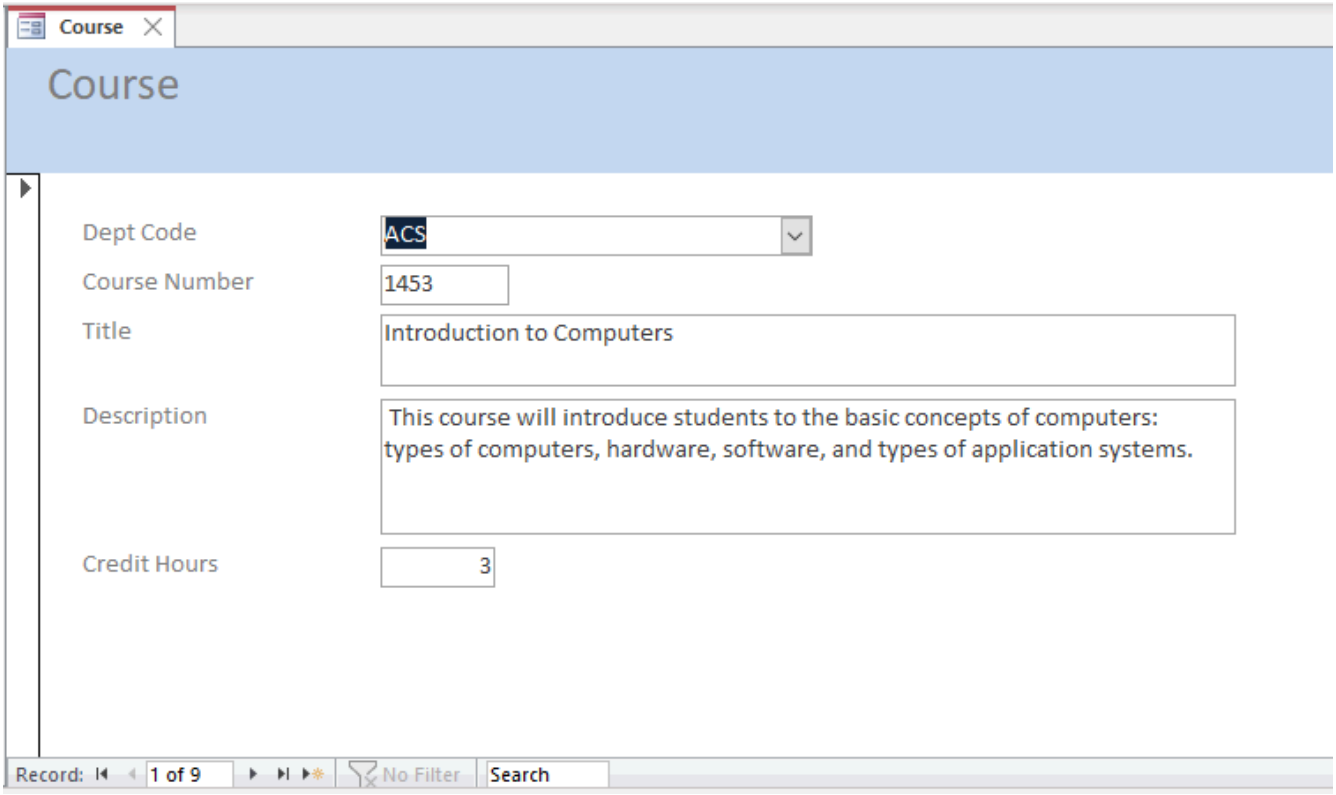


Figure 3.2: Navigation buttons on a form

3.2: Modifying the Form

You can make forms easier to use by adding buttons for common operations of, say,

- adding a new row
- deleting the currently displayed row
- closing the form

3.2.1: Adding a Button

To add buttons to an existing form you must open the form in Design View. At this point make sure that *wizard* capabilities are available:

- if necessary, select *Use Control Wizards*: (figure 3.3a) to turn wizard capabilities *on*
- you may need to click a *more* button to get a complete display as shown.

To add a button you click the *Button* button (figure 3.3b):

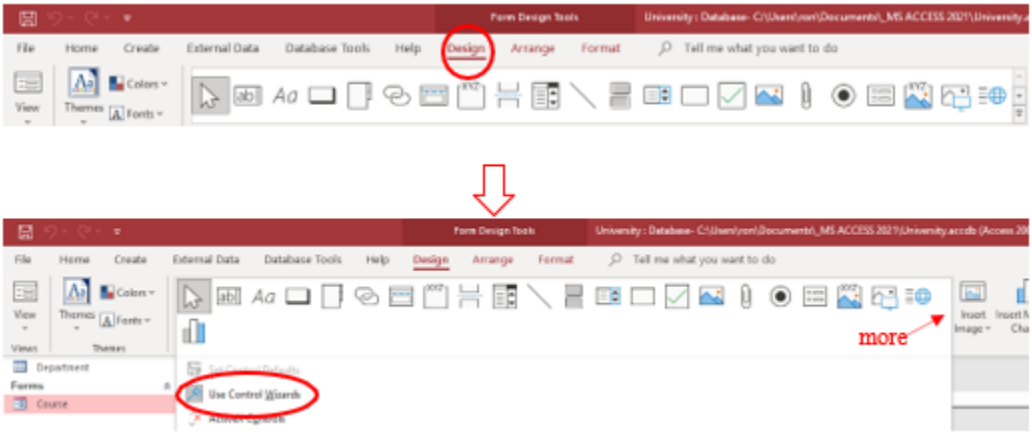


Figure 3.3a: Select Control Wizards

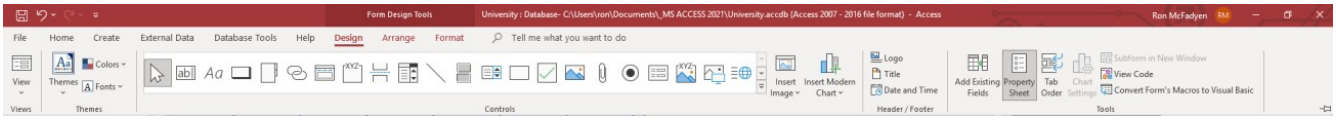


Figure 3.3b: Add a button control

Next, you click a location on the form where you would like the button to be placed. Click in the area labeled *Form Footer* and space will be added to the form's design to accommodate the button.

Because *Use Control Wizards* is on, the system will take you through a series of prompts where you specify the nature of the button. You need to add three buttons to each form. To improve your productivity, you may want the tool bar always visible. In the Tool Bar to the far right (figure 3.3b) you will see a pin … click that and the Tool Bar will stay pinned to your display. Consider selecting/ entering the following at the pertinent prompts:

Button	Category	Action	Text/Picture
Add button	Record operations	Add new record	New Row
Delete button	Record operations	Delete a record	Delete Row
Close button	Form operations	Close form	Close

Figure 3.4: Button categories

A Course form in Design View with three buttons in the Form Footer section:

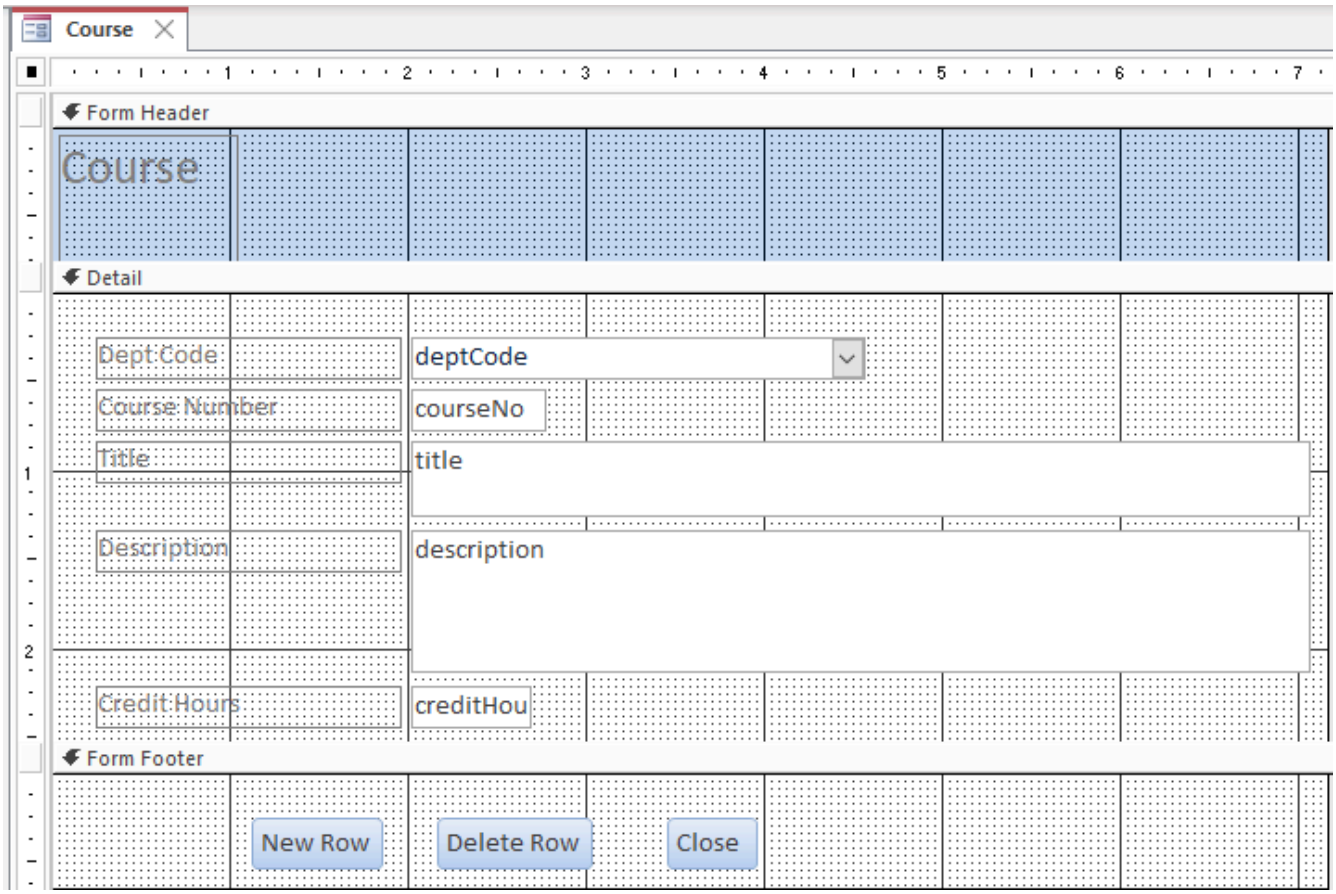


Figure 3.5: Course form with three buttons in form footer

At any time after creating a button you can switch to *Form View* to test your design. If some button is not working as you like then just switch back to *Design View*, delete the button and try again.

3.2.2: Adding a Label

A label is a control that holds text for display purposes only. By default, MS Access adds a label containing the table name in the Form Header area of the form.

To add a label you must click the Label control, then click (and drag for sizing) where you want the label placed. You can then type the content for the label and adjust its properties for formatting (e.g., font size, colour, etc.).

3.2.3: Adding a Calculated Field

A calculated field is one that involves a calculation using existing fields; for instance, to multiply a quantity and a unit price to get an extended price.

To add a control where a calculated value will be displayed you must click the Text Box control, then click (and drag for sizing) where you wish the control placed. You will see two controls placed in the form: a label and a text box. For the label one could enter *Extended price*, and in the text box you would enter a formula (e.g., for an extended price a formula would be: *=[quantity]*[unitprice]*).

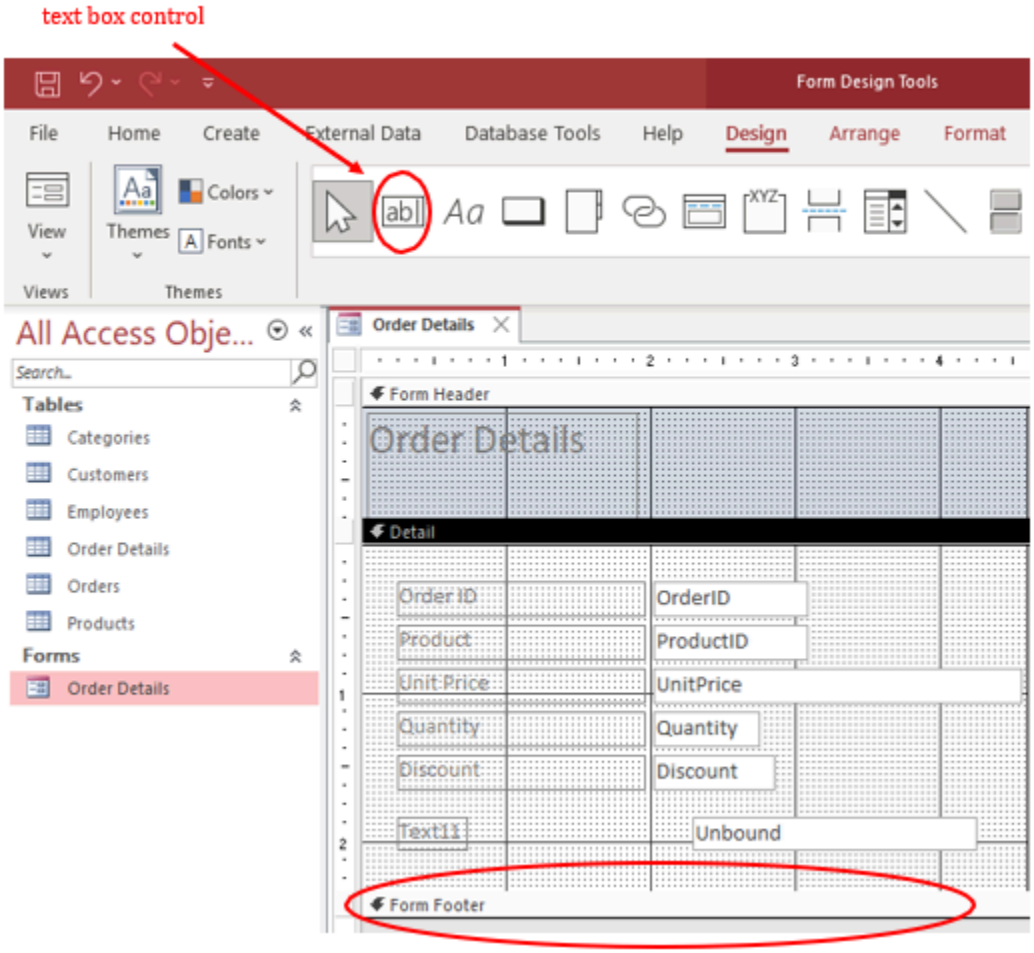


Figure 3.6: A TextBox control on a form

Adjust the size and location of the controls as necessary. To do this can be a little tricky. To move a control, you must select the control, and then click (and drag) the large dot in the control's upper left corner:

Figure 3.7: Move a control

To resize a control you must position the mouse so you can see a resizing indicator, and then click (and drag) to get the size you need:

Figure 3.8: Resize a control

Exercises

1) Open the Orders database (see databases for these notes) and create a form for OrderDetails. You will be able to incorporate the calculated field discussed above. Open your form in Form View and view the data to verify your calculated field displays properly.

- Note that you can modify the properties of fields on a form. When you are in Design View for this form you can right-click a field and select properties – the first property on the Format tab is *Format* and, for this calculated field, you could choose *Currency*.

2) Open the Library database and create forms for each of Book, Loan and Member.

4. MICROSOFT ACCESS QUERIES

Queries are used for multiple purposes in a database environment. They can be used directly to

- restrict the information a user can see,
- as the basis of a MS Access form,
- as the basis of a MS Access report.

In this chapter we use the Library database for examples. With MS Access you can create a query in multiple ways; we will examine the use of the Query Designer. To create a query, click the Create tab and then click the Query Design icon:

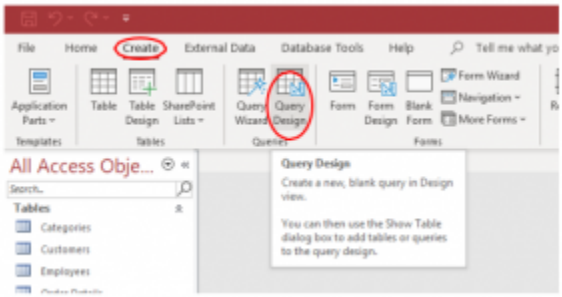


Figure 4.1: Create a query

As a result, MS Access opens a Query By Example (QBE) window:

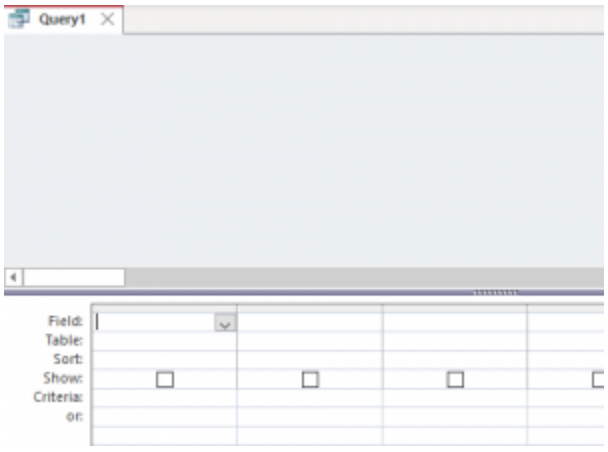


Figure 4.2: QBE window

This window comprises two areas: **Relationships** and **Grid**. The Relationships area will show each table that needs to be accessed and the relationships to be used with those tables.

The **Grid** area is used to specify:

- fields and tables,
- sort fields,
- fields to be included in the results display,
- criteria fields must meet for a row to be included in the query result,
- calculations,
- grouping of rows for displaying summary information

We use the Library database; we start with simple examples and work our way to more complex situations.

4.1: Simple Query

The simplest query is one that displays a complete table – all rows and columns. Suppose we want to list all books in the library. The process of creating the query is as follows:

- Click on the Create tab if necessary and then click on the Query Design icon; then:
- drag the Book table from the Tables panel into the Relationships area, **or**,
- right-click in the Relationships area and choose the Show Table. Double-click Book.

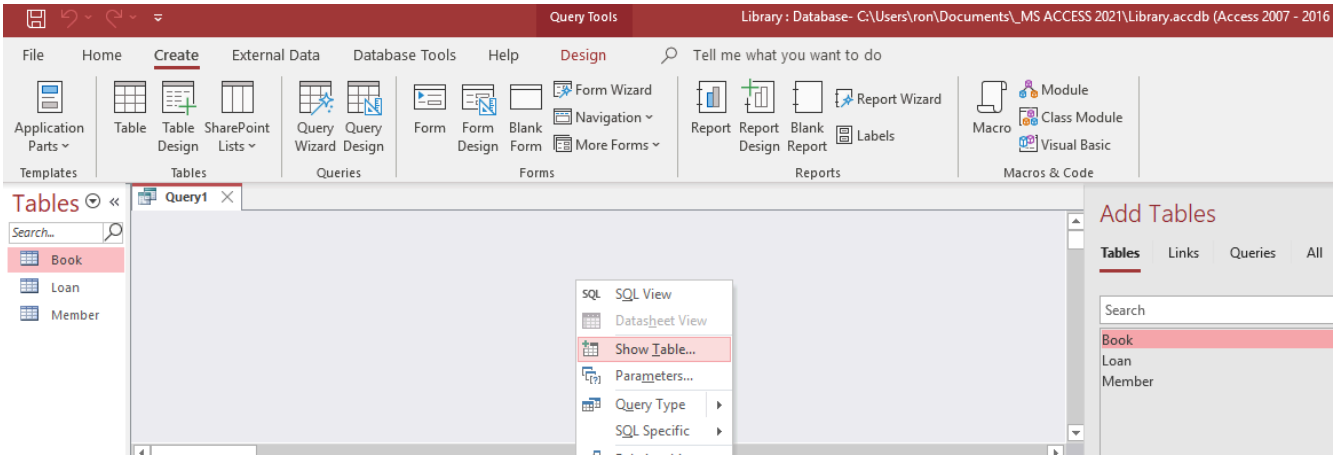


Figure 4.3: Select a table for the query

MS Access displays the Book table and its fields in the Relationships area. The first in this list is an * which stands for *all attributes* – double-click the *. Note the contents of the Grid area:

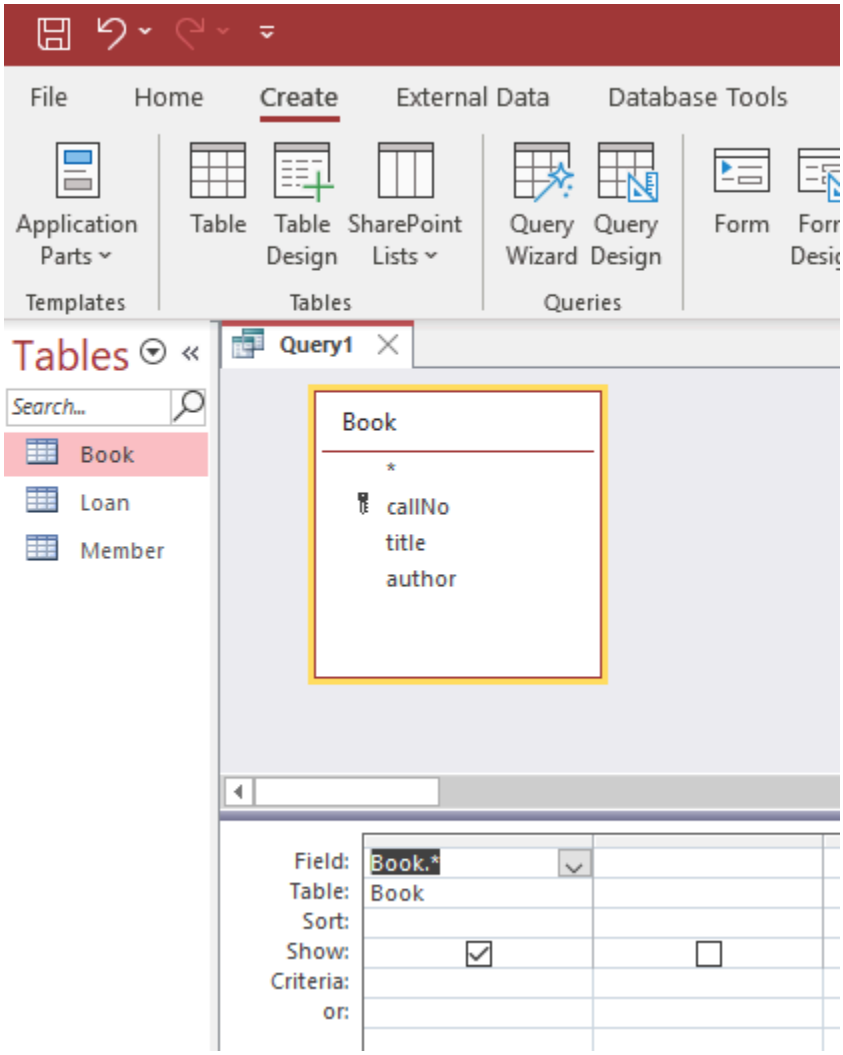


Figure 4.4: Choosing all fields of the table

You can run a query when using Query Design view. If the Run button is not visible, click the "Design" tab. Now, to run the query, click the "Run" button in the "Results" button group:

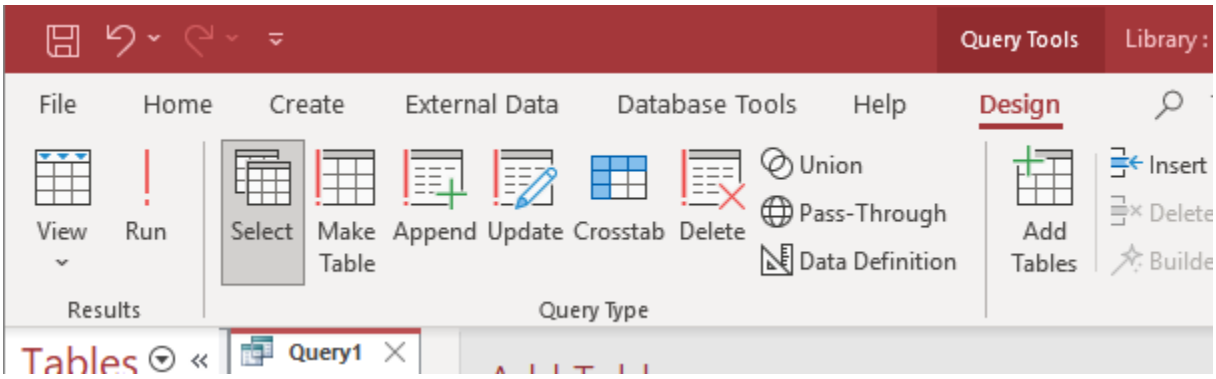


Figure 4.5: Run a query

There are other views of a query. If you click the drop-down just below *View*:

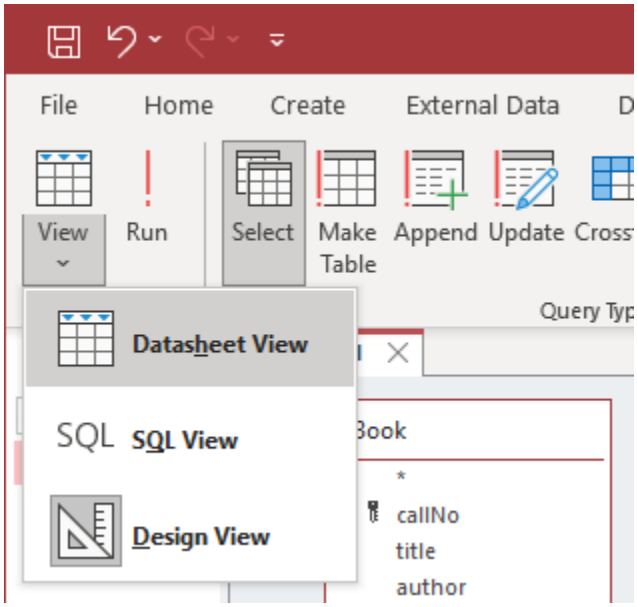


Figure 4.6: Several views for a query

You can see all the ways of viewing a query, including:

- Datasheet View
- Design View
- SQL View.

You can run a query by choosing Datasheet View. When developing a query, one often alternates between Datasheet View and Design View to get the query working as required. When you run a query, MS Access will retrieve the information requested. Our results of running the query are:

callNo	title	author
CB 351 M293 1983	Atlas of medieval Europe	Donald Matthews
HQ 1143 P68 1975	Medieval women	Eileen Power
PC 14 V48 1965	Medieval miscellany	Frederick Whitehead
QA 76.73 S67C435 2004	Joe Celko's Trees and hierarchies in SQL for smartie	Joe Celko
QA 76.73 S67C46 1997	Joe Celko's SQL puzzles & answers	Joe Celko
QA 76.76 A65P76 2011	Programming Android	Zigurd R Mednieks
QA 76.9 D26H355 2008	Information modeling and relational databases	T A Halpin
QA 76.9 D26H39 1996	Data model patterns : conventions of thought	David Hay
QA 76.9 D35C45 1999	Joe Celko's data & databases : concepts in practice	Joe Celko
R 141 E45 2006	Medieval medicine and the plague	Lynne Elliott
R 487 T35 1967	Medicine in medieval England.	Charles H Talbot

Figure 4.7: Query results

You can save the query: right-click the name and then give the query a new name:

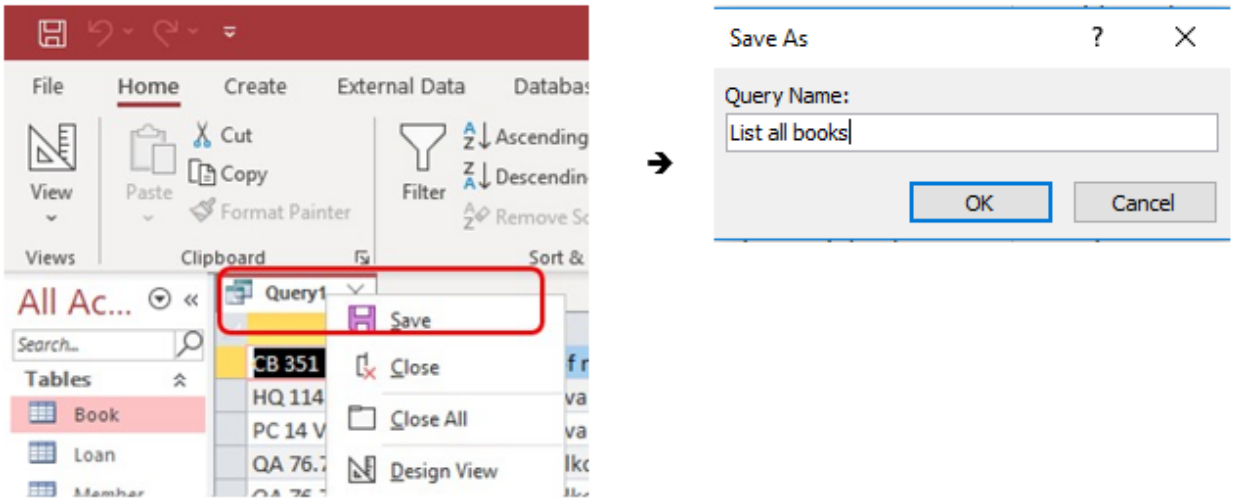


Figure 4.8a: Save a query and give it a name "List all books"

Now, provided you have selected 'All Access Objects' you will see the query as a database object. See figure 4.8b.

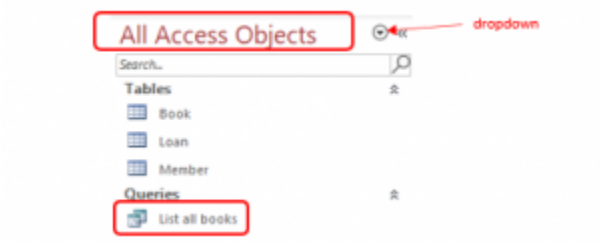


Figure 4.8b: The drop-down determines what objects you see

The query can be run any time by an end user. The results of the query are not stored or saved – only the definition of the query. Whenever a user runs the query the current contents of the Book table are accessed.

4.2: Projection Query

Next, we build a query that displays a subset of the columns of a table. Suppose we need to produce a listing of call numbers and titles. Proceed as in the previous example so that the Book table is shown in the Relationships area. Now, double-click the callNo and title fields:

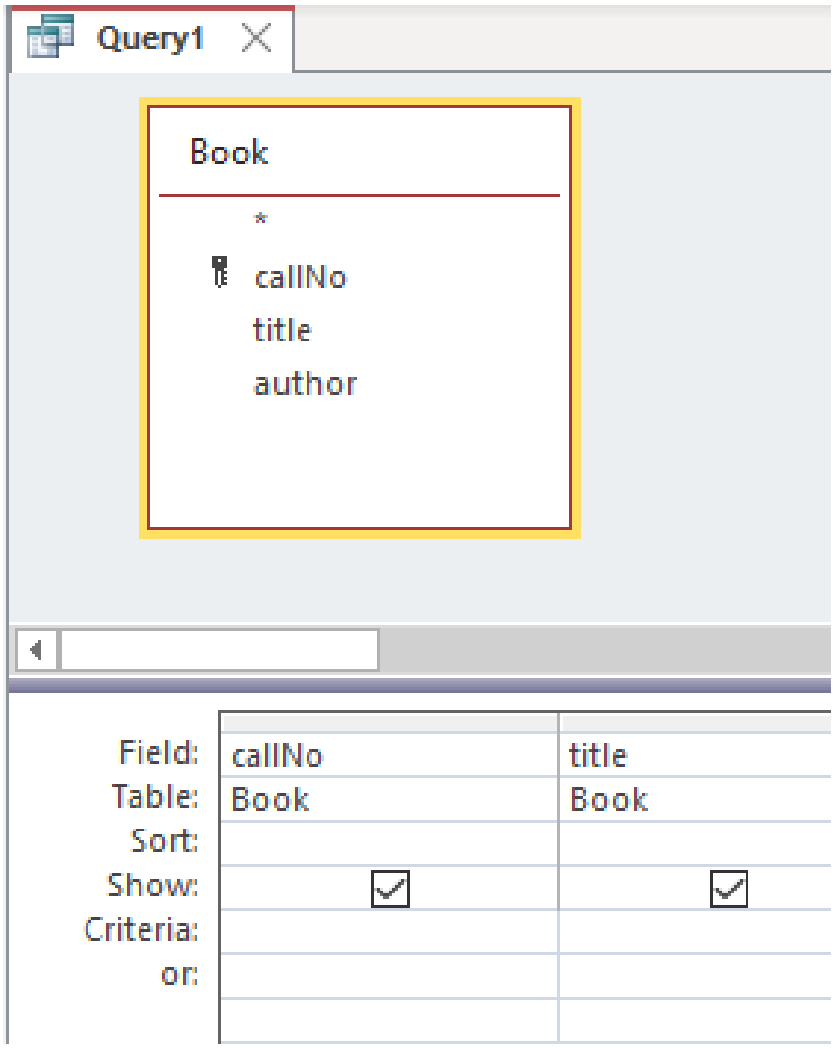


Figure 4.9: Projection query with specific fields

The definition of the query is now complete. The grid area indicates the fields involved, and both fields will be displayed because each has a check mark on the *Show* line. Only fields checked on the Show line are displayed in the results. Running this query produces 2 columns:

callNo	title
CB 351 M293 1983	Atlas of medieval Europe
HQ 1143 P68 1975	Medieval women
PC 14 V48 1965	Medieval miscellany
QA 76.73 S67C435 2004	Joe Celko's Trees and hierarchies in SQL for smartie
QA 76.73 S67C46 1997	Joe Celko's SQL puzzles & answers
QA 76.76 A65P76 2011	Programming Android
QA 76.9 D26H355 2008	Information modeling and relational databases
QA 76.9 D26H39 1996	Data model patterns : conventions of thought
QA 76.9 D35C45 1999	Joe Celko's data & databases : concepts in practice
R 141 E45 2006	Medieval medicine and the plague
R 487 T35 1967	Medicine in medieval England.

Save the query. A projection query, because it displays a subset of the fields in the table, is said to produce a vertical slice of the table.

4.3: Selection Query

Suppose we want a list of paperbacks. That is, we want to list information about books where the paperback field has a value *Yes*. Requirements like this are placed on the criteria line of the pertinent field(s).

To develop this query we need to select the Book table and then add its' fields to the grid.

For the paperback field we also enter the value *Yes* on the criteria line:

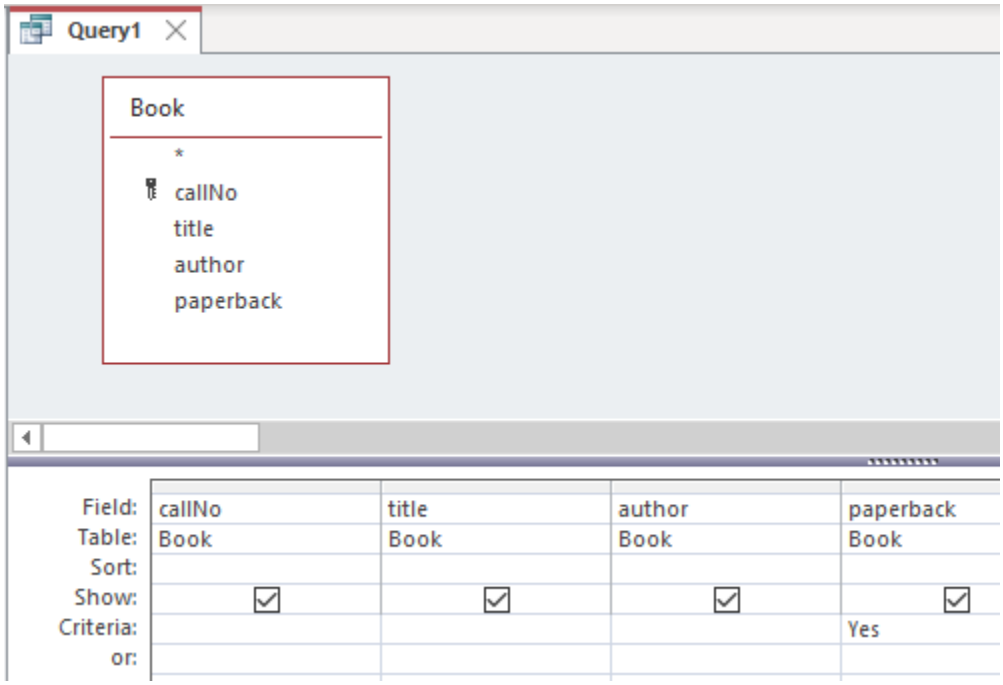


Figure 4.10: Query with selection criteria

When we run the query we get results listing paperbacks:

callNo	title	author	paperback
CB 351 M293 1983	Atlas of medieval Europe	Donald Matthews	☑
PC 14 V48 1965	Medieval miscellany	Frederick Whitehead	☑
QA 76.73 S67C46 1997	Joe Celko's SQL puzzles & answers	Joe Celko	☑
QA 76.76 A65P76 2011	Programming Android	Zigurd R Mednieks	☑
QA 76.9 D26H355 2008	Information modeling and relational databases	T A Halpin	☑
QA 76.9 D26H39 1996	Data model patterns : conventions of thought	David Hay	☑

Figure 4.11: Query results

When a query runs, the query processor accesses the underlying table(s) and displays results

where the data meets the criteria specified. For a query accessing a single table consider that the query processor is performing these actions:

For each row in the table:

- retrieve the row from the database
- test the row to see if it meets the criteria specified
    - if the row meets the criteria then display the fields marked for show

Save your query as paperbacksQuery. This is a selection query that selects, according to criteria, specific rows for display; this type of query produces a horizontal subset of a table.

Most queries are a combination of selection and projection. It is typically the case that queries will select a subset of fields for display and criteria will be needed to constrain the rows retrieved.

4.4: Sorting the Result

Sometimes an end user wants to see data in a particular order. Let us extend the previous example so books are listed in alphabetic order by title and, since they are all paperbacks, we will not display the paperback field.

Create another query similar to the last one. Now, place the cursor in the Sort line beneath the title field: click and select *ascending* from the choices. Then click the Show check box for paperback to turn Show off.

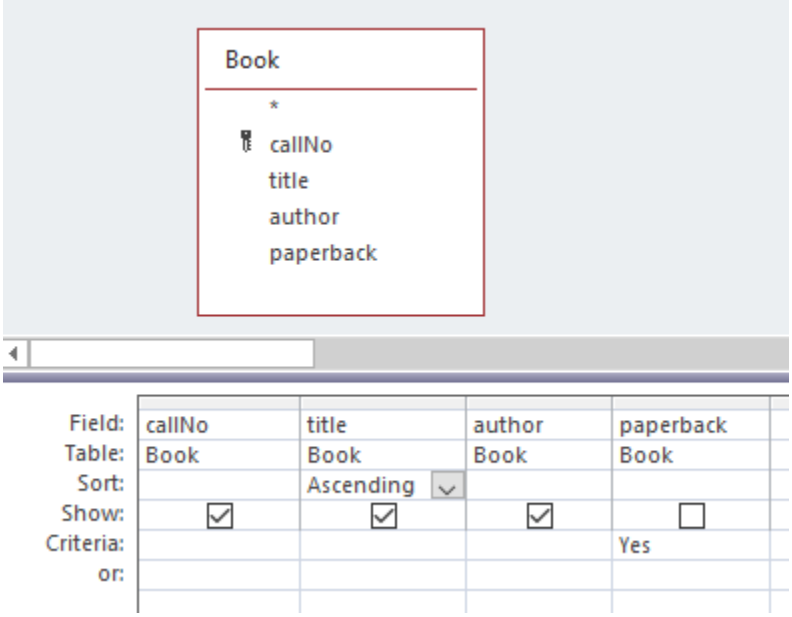


Figure 4.12: Query results can be sorted

Save this query and run it. Notice how the results are sequenced by title.

Exercises

1) List the titles of books in descending order.

2) List the titles of books written by Joe Celko.

3) List all members of the library.

4) List the members in sequence by last name.

5) List the members sequenced by last name and then by first name. (If members have the same last name they appear on consecutive lines, and those lines are in sequence by first name.)

6) Which of the above are a) simple queries, b) selection queries, c) projection queries, d) both selection and projection queries?

4.5: And

Suppose we want to list Celko's books on SQL. In this case there are two criteria a book must meet:

- Criteria 1: the author's name must end with "Celko"
- Criteria 2: "SQL" must appear in the title.

Criteria 1 must be true *and* Criteria 2 must be true – we say the two criteria are *and*ed. When using QBE we must place these two criteria on the same criteria line in order that MS Access finds rows that match both criteria.

In this example we are looking for titles that have the text SQL anywhere within the title. MS Access provides a way for us to define such a pattern. The character * when used in a text string is a wildcard character that matches any number (zero or more) of characters. For Criteria 1 we use the *Like* operator, and we need two wildcards so we specify the pattern that title must match: *Like "*SQL*"*.

For Criteria 2 we specify the pattern that author must match: *Like "*Celko"*.

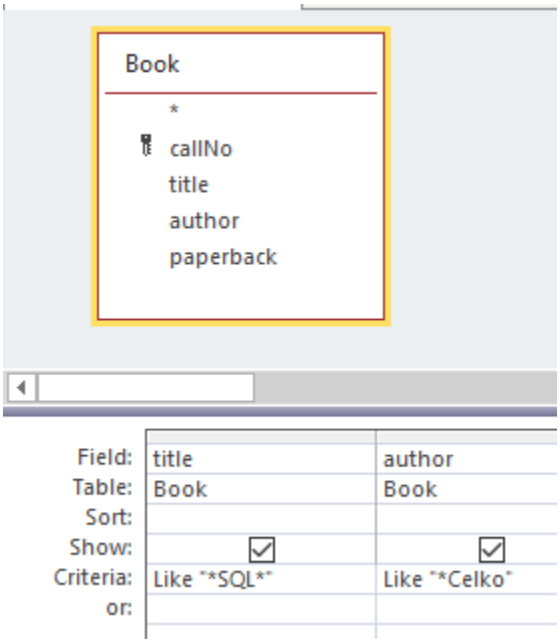


Figure 4.13: Criteria on one criteria line are ANDed

4.6: Or

Instead of books with titles containing “SQL” and authored by Celko, suppose the end user wants a list of books with “SQL” in the title or where Celko is the author. In this situation we place the criteria on separate lines. MS Access *OR*s the criteria; a row is selected for the result set if either, or both, of the criteria are true for a row.

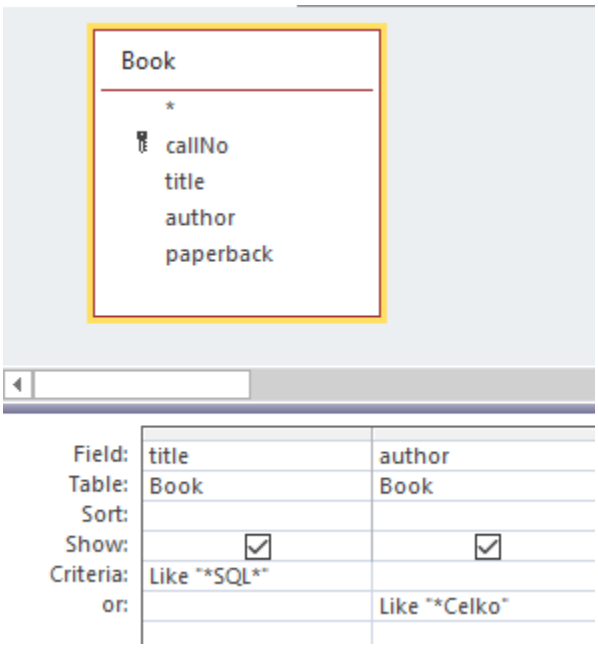


Figure 4.14: Criteria on different criteria lines are ORed

Exercises

1) List the titles of books where the authour name ends with “Celko”.

2) List the titles of books where the authour name ends with “Celko” **and** the text “data” appears in the title.

3) List the titles of books where the authour name ends with “Celko” **or** the text “data” appears in the title.

4) List titles of books where the title contains the word “medieval”.

5) List the titles of books where the title contains the words “medicine” **and** “medieval”.

6) List the titles of books where the title contains the words “medicine” **or** “medieval”.

4.7: Joins

If a query must be answered using data that appears in more than one table then the query requires a database *join*.

Suppose we wish to produce a list of member names and the call numbers of books they have borrowed. Important points about this query:

- The Loan table has the loan information we need.
- The Member table has the member names we need.

Before we compose the query consider how you would produce the results if you were to do this manually. If you had two listings showing the rows of each table in front of you on your desk you could proceed as follows going through the Loan table listing row by row starting with the first row:

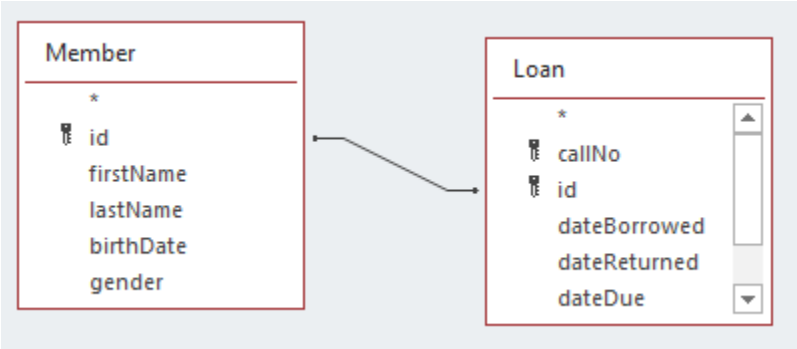
1) For the current row in Loan:

- Write down the call number of this loan.
- Let N stand for the value of the member`s id for this loan.
- Now look at the Member listing row by row starting with the first row: Examine the row to determine if the row is for member N, and if it is, write down the member`s name beside the call number.

2) If there are more rows in Loan, advance to the next row and go back to step 1

In the above algorithm we have determined a member id at step 1.b) and we next look for a matching member id in step 1.c). For a human the process is simple but tedious. We could say we are trying to go from a row in Loan to a row in Member based on rows having the same value for member id. In database terminology we say we are *joining* Loan to Member based on a common value of member id. A tedious but well-defined task is something a computer can excel at, and fortunately we can get the database system to do the job of *joining* rows, based on values of a common attribute, for us. Construct the query in the following way:

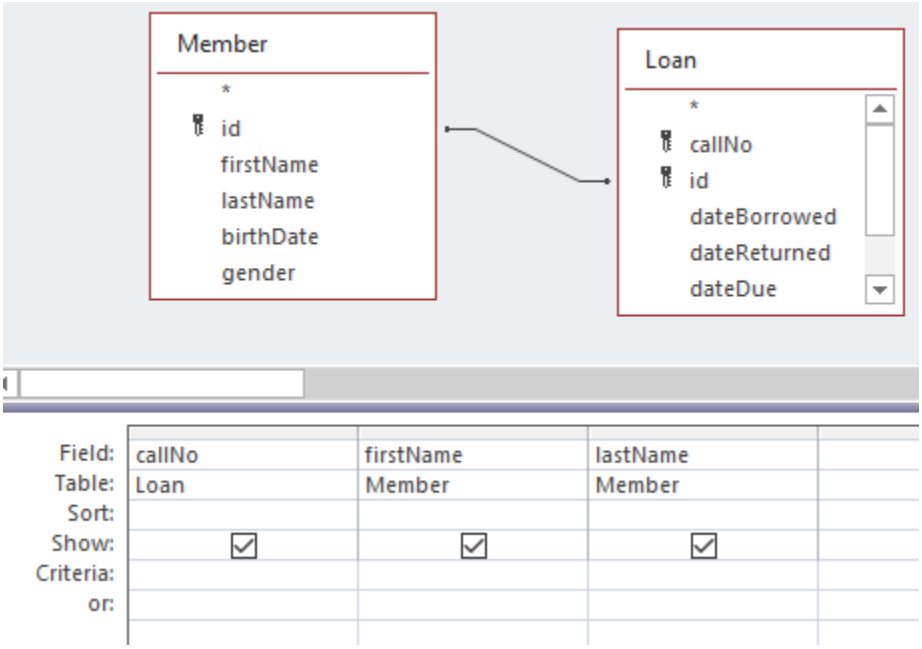
- Create a new query
- Drag the Member and Loan tables into the Relationships Area:



- Note the line connecting the two tables. This is called a *relationships line* which causes MS Access

to join pairs of rows – a row in Member is joined to a row in Loan where the two rows have the same value for id.

- Select the call number, first name, and last name fields by double-clicking them to obtain:



- Run the query and you see the results:

callNo	firstName	lastName
HQ 1143 P68 1975	John	Smith
QA 76.73 S67C46 1997	David	Martin
R 141 E45 2006	David	Martin
R 141 E45 2006	Betty	Freeman

Exercises

In each of the following exercises the necessary data is in more than one table and so it is necessary to specify a join.

1) For each loan show the title of the book and the date it was borrowed. Note that the title is in the Book table and the date borrowed is in the Loan table.

2) Modify the previous query to produce a listing that is in order by title and then by date.

3) Produce a list that shows for each loan the book title, the name of the member who borrowed the book, and the dates the book was borrowed and then returned. Note that 3 tables are needed for this query:

- Book joins to Loan
- Loan joins to Member

4) Produce a list of members and the books they have taken out on loan. Include the member's last name, first name, and titles of the books. The information to be displayed is in 2 tables, but it is necessary to specify 3 tables for this query:

- Member joins to Loan
- Book joins to loan

5) Modify the previous query to produce a listing that is in order by last name and then by first name.

6) For member id 2, list the person`s name and the titles borrowed.

7) Produce a list of book titles and member names for those books that are due back May 18, 2014.

8) Produce a list of book titles and member names for those books that have not been returned. In this case you must give the criteria for dateReturned as *null*. Null is a special keyword that represents no value.

5. RELATIONSHIPS AND THE RELATIONSHIPS TOOL

The Relationships Tool is used to define relationships between tables based on common fields. Relationships defined using the Relationships Tool are important as they help ensure integrity of data, and they provide us with default join criteria for queries involving more than one table. In this section we will use the University and the Library databases in our examples.

Consider the University database we have been using that contains a Department table and a Course table. These two tables have the deptCode field in common:

- In the Department table, deptCode is the primary key and is used to identify a specific department.
- In the Course table, the deptCode field is a part of the primary key and indicates the department to which the course belongs.

To ensure that a row in Course is related to an existing row in Department we can use the Relationships Tool to define a relationship between these two tables based on this common field. Using a diagram, we can illustrate this connection between these two tables:

Figure 5.1: Showing a relationship between two tables

In this situation we say that deptCode in Course is a *foreign key* referencing the deptCode field in Department.

Now, consider the Library database:

- The Loan table has a callNo field and so does the Book table; the callNo field identifies a specific book.
- The Loan table has an id field and so does the Member table; the id field identifies an individual member.

In the Library database we can establish a relationship between Loan and Book based on the callNo field, and a second relationship between Loan and Member based on the id field. Using a diagram, we can illustrate these two relationships:

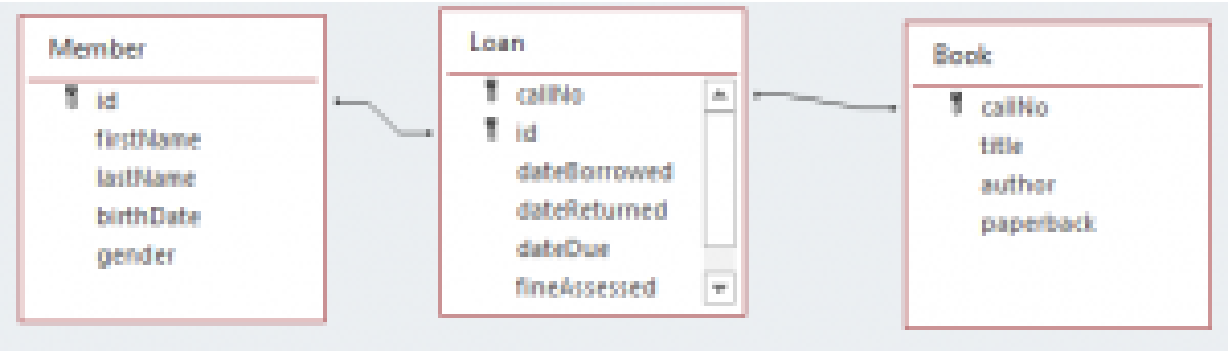


Figure 5.2: Showing relationships involving three tables

The Loan table has two foreign keys, callNo and id:

- The callNo field in Loan is a *foreign key* referencing the primary key (callNo) in Book.
- The id field in Loan is a *foreign key* referencing the primary key (id) in Member.

5.1: Integrity

Primary Key

Recall that a table's PK is a field (possibly composite) that has unique values – each row has a PK value different from any other row in the table. Such a field is a unique identifier

- if a query were designed to retrieve a row of that table based on a value of the PK, then at most one row of the table will be retrieved.

Foreign Key

A foreign key is a field (or combination of fields) in a table B that is associated with a PK in a table A through a relationship (A and B can be the same table).

Entity Integrity

When we define a PK for a table, we are enforcing entity integrity. Entity integrity means that each row in the table is identifiable through its primary key. MS Access requires a value for a PK in a newly added row, and MS Access enforces uniqueness of those values.

Referential Integrity

Suppose we have two tables, A and B, where a relationship is defined between the primary key of table A and a foreign key in table B. We say referential integrity (RI) exists for this relationship if for each row in B either:

- the FK has no value at all (i.e., it is null), or
- the FK has a value that exists as a PK value in some row of A.

5.2: Relationships

Two tables can be related through one-to-one, one-to-many, or many-to-many relationships. If you open the Relationships Tool for the University database, you will see the following diagram showing two tables and one relationship:

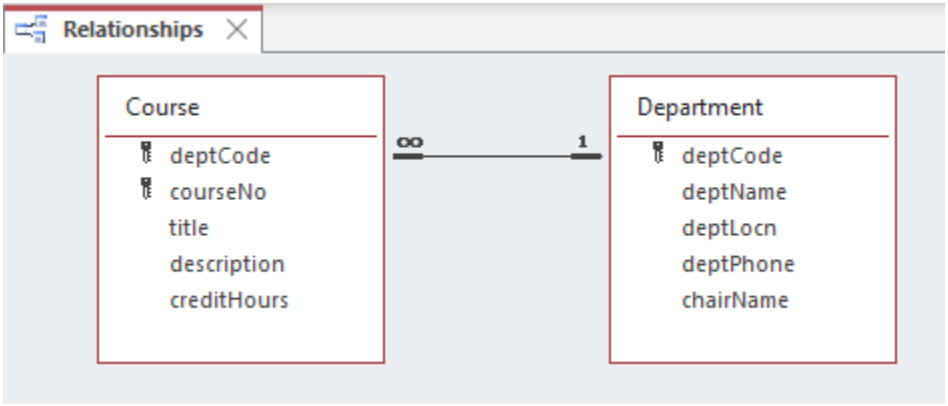


Figure 5.3: Relationship: department offers courses

There are two labels on the line which inform us the relationship is one-to-many for which there are two rules that are in place:

- for each department there will be zero or more courses for that department, and,
- each course is for exactly one department.

To create a relationship in MS Access you must:

- open the Relationships Tool
- add the pertinent tables to the diagram if they are not there already
- click, hold, and drag a field (normally this is the PK) of one table to the related field (to become an FK) in the other table.

You will be asked whether or not Referential Integrity is to be enforced. As a general rule- of-thumb, you should select Yes – there must be some exceptional circumstance that makes you select No.

Once relationships are established using the Relationships Tool they are used by MS Access when you create queries – the relationships are used as the default for table joins.

5.2.1: One-To-Many

If you drag the PK field of one table to the other table, and if the FK does not have unique values, then you are creating a one-to-many relationship. MS Access will know and will display that fact for you. For each row in the referenced table there can be several related rows in the other table; that is, for a PK value there can be many rows in the other table with that value stored in the FK.

Example

Department and Course are related through the deptCode field. You can go through the exercise of creating the relationship between these two tables, but first you must remove the current relationship:

- delete the existing relationships line (click the line, press delete, and follow through with the dialog to delete the relationship).

Now, click and drag the deptCode field in Department and drop it on top of the deptCode field in Course. On releasing the mouse MS Access will present the following dialogue box:

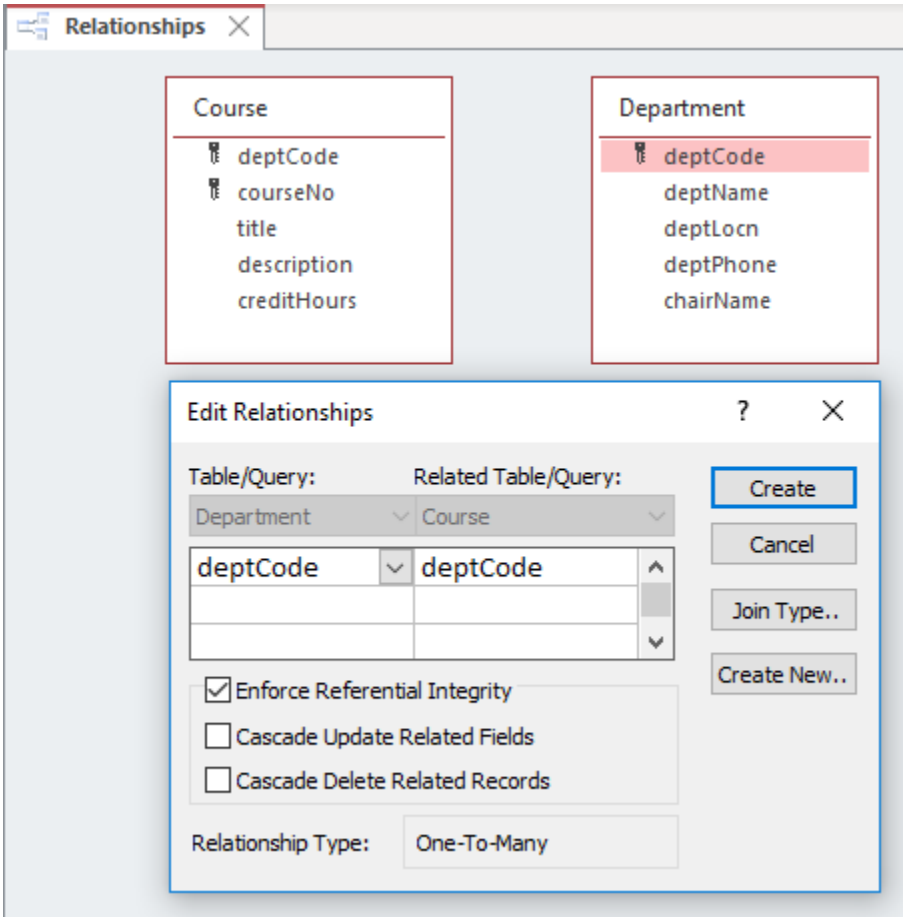


Figure 5.4: Defining a one-to-many relationship

At this point MS Access is requesting the user to confirm the proper fields are being related, and for the user to make a choice regarding Referential Integrity and on some 'Cascade' options – we do not discuss cascading in these notes. You should choose *Enforce Referential Integrity* in almost all cases as this helps reduce the chance of corrupting data.

For the above, when the user clicks Create, MS Access shows the relationships line with *1* on the one side and an *infinity* symbol on the many side of the relationship:

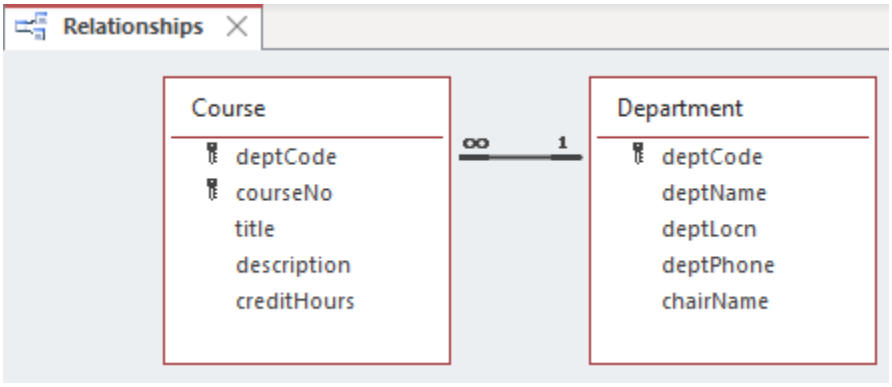


Figure 5.5: One-to-many relationship: department offers courses

5.2.2: One-To-One

If you drag a PK field of one table to another table, and if the FK has unique values (a unique index exists for it) then you are creating a one-to-one relationship. For each row in the first table there can be at most one related row in the other table; a row in the referenced table has a PK value that equals the FK value in at most one row of the referencing table.

5.2.3: Many-To-Many

If you create a relationship in MS access where both fields you associated (via the click, hold, and drag sequence) do not have unique values (i.e., neither have unique indexes) then MS Access creates an 'indeterminant' relationship. In this situation a row in one table, A, may be related to multiple rows in the other table, B, and where a row in table B may be related to multiple rows in the table A.

This is not done very often and corresponds to a many-to-many relationship. Most database designers would avoid this in their database designs. If a database designer is faced with two tables, A and B, that are related via a many-to-many relationship, the designer would likely introduce a third table, say C, where A and C will be related via a one-to-many relationship and similarly, B and C will be related via a one-to-many relationship.

Later in these notes we discuss database design. We will see how many-to-many relationships can be decomposed into two one-to-many relationships.

Exercises

For these exercises use the Company database which does not have any relationships defined. The first few rows of Employee and Department are:

empId	firstName	lastName	supervisor	dept
1	Tanya	Dickson		
2	Heidi	Herring	1	1
3	Hiroko	Hawkins	1	2
4	Emmanuel	Watkins	1	3
5	Oliver	Holt	2	1
6	Raphael	Delaney	3	2
7	Basia	Franks	2	1
8	Bruno	Pena	2	1

deptId	department	manager	phone
1	Marketing	2	(204) 999-4444
2	Human Resources	3	(204) 999-3333
3	Sales	4	(204) 999-2222

1) Consider the Employee and Department tables. Note the Employee table has a field dept which indicates the department where the employee works. The relationship can be stated:

- Each department has zero or more employees, and,
- Each employee works in at most one department.

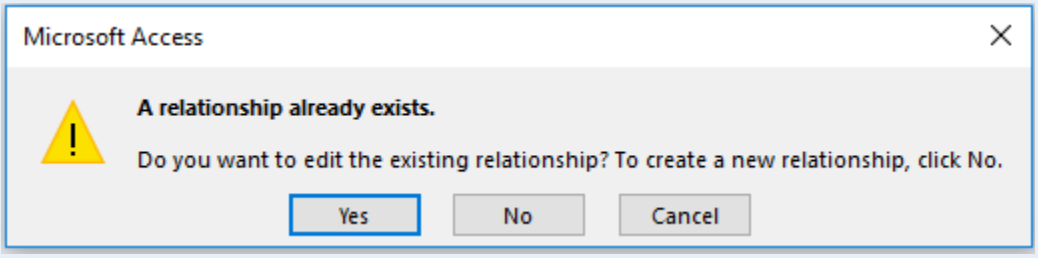
Create a one-to-many "works in" relationship between Employee and Department.

2) The Department table has a field *manager* which indicates the employee who is the head of the department. The relationship is stated:

- each department has one employee who manages that department, and,
- an employee may manage at most one department.

There is a unique index defined for the manager field and so you can create a one-to- one relationship "has manager" between Department and Employee.

If you do this exercise after exercise 1 has been completed, then you need to read the dialog boxes carefully when you create this second relationship between these tables. You must respond *No* to the following dialogue window.



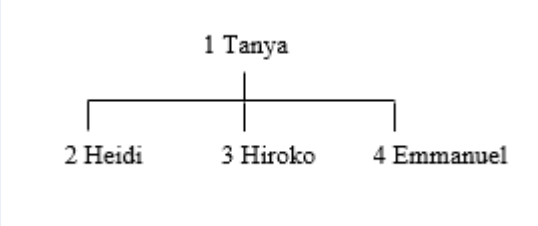
Note how MS Access represents two relationships between two tables.

3) Consider the empId and the supervisor fields of Employee. Most employees report to someone – someone who is their supervisor. Only employee 1 does not report to anyone else. The *supervises* relationship can be stated:

- an employee may supervise many other employees, and,
- an employee reports to at most one other employee.
    - Create the supervises relationship.

If you are doing this exercise after exercise 2 then your relationship diagram has 2 copies of the Employee table. If you are not doing this after exercise 2, then you must add Employee to the diagram twice so there are 2 copies of Employee on the diagram. Drag the PK, empId, from one copy of Employee to the supervisor field of the other copy. Note how MS Access draws this diagram.

- The supervisor field is an implementation of a hierarchical reporting structure for our company. Use a piece of paper and draw the reporting structure for the company (for the data given at the start of these exercises). We have started this exercise showing the reporting structure for the first 4 employees:



Exercises 4-6 depend on the relationships diagram from the above exercises. When developing a query, you will see that MS Access will include relationships when you include tables in the relationships area of a query. Do consider them closely to ensure they are the relationships and joins you need.

4) Create a query to list for each department, the name of the department and the name of its manager.

5) Create a query to list for each department, the name of the department and the names of its employees (the people who work in the department). Sequence your results by department name.

6) Create a query to list for each department, the name of the department head and the names of the department's employees. Your query must list on each row of the result set the department name, the head's last name, and the last name of each employee. Sequence your results by department name, and within department by employee last name.

7) Create a query that lists each supervisor and the employees he/she is supervising. Your query must list, on each row of the result set, the last name of the supervisor and the last name of the supervised employee. Sequence the results by supervisor and within supervisor by employee. Hint: begin the query by dragging the Employee table onto the relationships diagram twice, and then drag empId to supervisor.

6. MICROSOFT ACCESS QUERIES – ADVANCED

Previously in Chapter 4 we saw how to construct queries using simple logical expressions where either all the criteria were ANDed, or where the criteria were all ORed. Now we'll examine more complex situations.

6.1: Logical Expressions

Sometimes we need to retrieve data based on multiple criteria which are expressed as logical expressions involving the logical operators *and*, *or*, and *not*. For example, a student using the University database might want to know which courses offered by the Chemistry and Physics departments are not full courses (that is, they are not 6 credit hour courses). The criteria can be restated with emphasis on logical operators:

- a course is a Chemistry course **or** a course is a Physics course,

  and

- the course has any value for credit hours but not 6.

Such criteria involves *and*, *or*, and *not*. Stating the requirements in natural language may seem easy, but stating these properly in the forms-based Query By Example design window requires specialized knowledge.

MS Access provides a way for us to specify the above using the *Criteria* and *Or* lines in the Grid. We will consider each of the operators And, Or, and Not.

6.1.1: And

If one specifies multiple criteria on one line in the Grid area, these criteria are ANDed. For a row to contribute to the result of the query the row must satisfy all the criteria.

Example

Suppose we want a list of all ACS 3 credit hour courses. We need to obtain the rows in Course where the logical expression

(deptCode=”ACS”) **AND** (creditHours=3)

is true. We code this in QBE as:

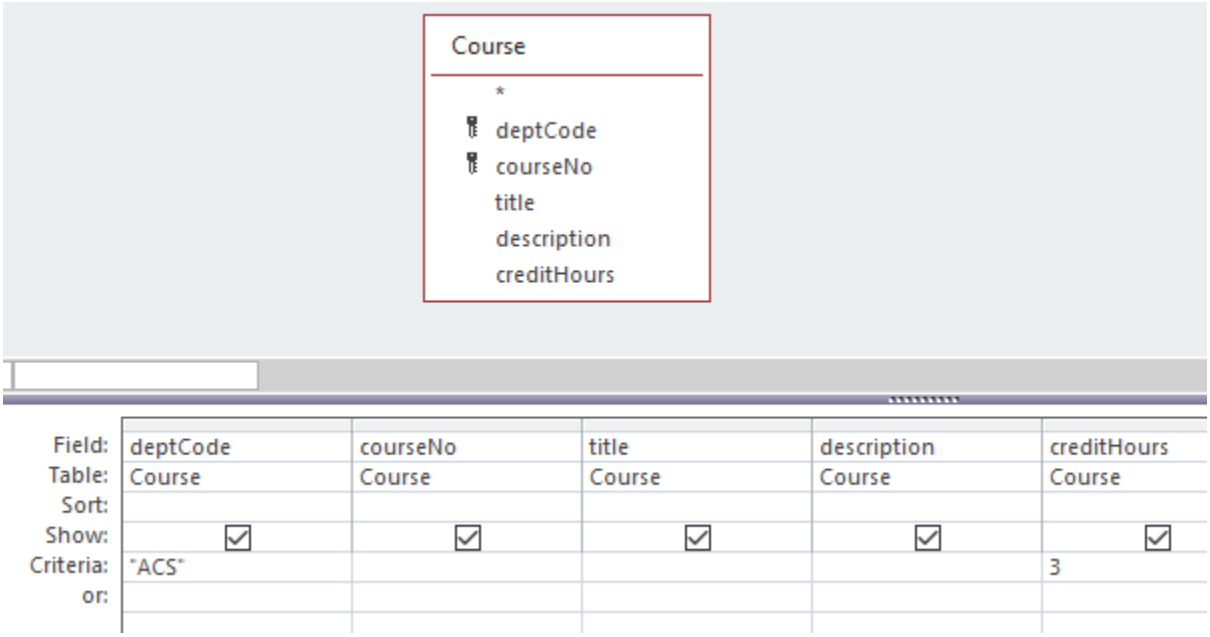


Figure 6.1: Two expressions that are ANDed

6.1.2: Or

If one specifies multiple criteria on both the Criteria and Or lines the criteria on each line is ANDed, and those evaluations are then ORed. If for some row either one or both of the sub-expressions evaluate to true, then the row will be selected for display.

Example

Suppose we need a list of all ACS courses that are 3 or 6 credit hour courses. Logically we can express this as:

(deptCode=”ACS” **AND** creditHours=3)

OR

(deptCode=”ACS” **AND** creditHours=6)

We code this in QBE as:

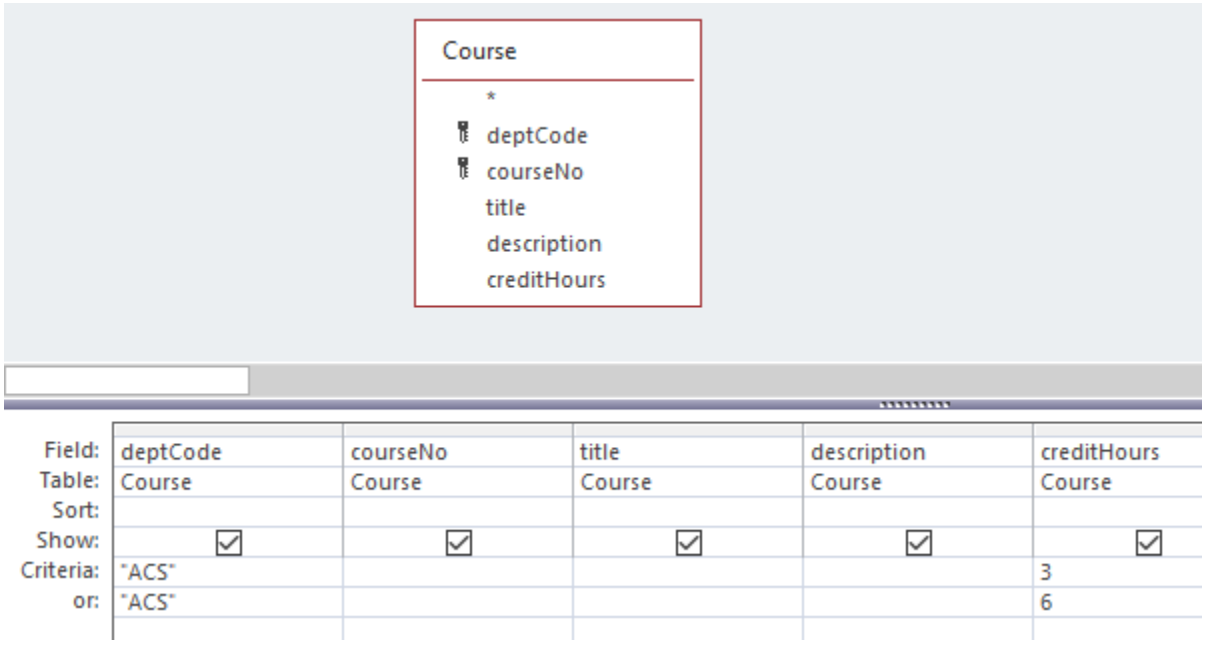


Figure 6.2: Two expressions that are ORed

6.1.3: Not

The NOT logical operator negates a logical expression.

Example

To get a list of 3-credit hour courses we would use a criteria of 3, but to list courses that are not 3 credit hours one could use the criteria: NOT 3, which, written in long form is:

NOT (creditHours = 3)

Coding this in QBE we have:

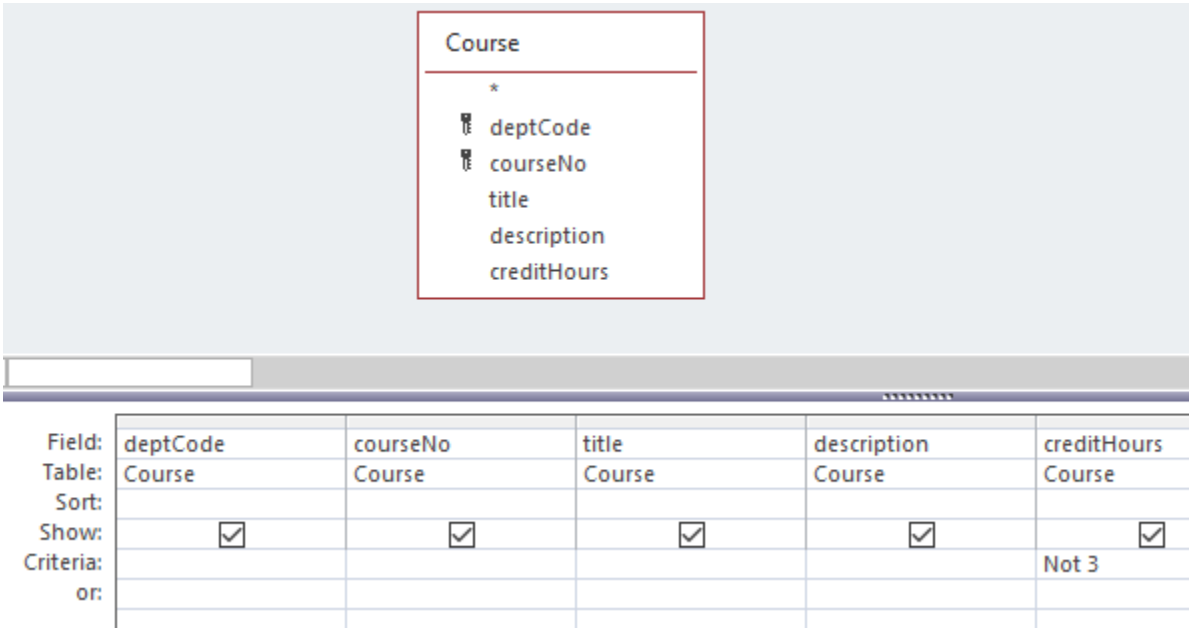


Figure 6.3: Using NOT

Exercises

Write queries to:

1) List all courses in Mathematics and Statistics.

2) List all courses in Mathematics and Statistics where the credit hours are greater than 1.

3) Lists the titles of courses offered by the Chemistry and Physics departments that are not full courses (that is, they are not 6 credit hour courses).

- Once this query is coded, close and reopen your query. You may see that MS Access converts your “Not 6” expression to another alternate form, “< > 6”, where the combination < > stands for *not equal to*.

4) List all 3 and 6 credit hour courses that are not ACS courses.

6.2: Query Operators

We present two more operators: LIKE, which is used for pattern matching of text values, and IN, which is used to test for inclusion in a set.

6.2.1: Like

Sometimes we need to get information based on partial information. Consider someone using the University database and wanting to find courses where the course description contains the word “computer”. To find courses matching this criterion we can use the *Like* operator where we specify an appropriate pattern. These patterns are defined using one or more *wildcard* characters. By default, our MS Access databases use the ANSI-89 standard for special wildcard characters.

Note: At some point you may want to investigate the more recent ANSI-92 standard for wildcards. You can change the standard your database is using by examining and changing the MS Access Options for Object Designers/Query Design.

The ANSI-89 wildcard characters are:

Wildcard Character	Matching criteria	Example
*	Matches any number of characters	Like “1*” matches all text strings that start with “1”
?	Matches any single character	Like “a?c” matches “aac”, “abc”, “acc”, etc. but does not match longer strings such as “aacc” or “xabc”
#	Matches any single numeric character	Like “b#b” would match “b2b” and “b7b” but not “bam”
[]	Matches any single character within the brackets	Like “j[ai]m” matches “jim” and “jam” but not “jaim”
!	Used with [] when you do not want to match any of the enclosed characters	Like “b[!ao]b” matches “bim” and “bub” but not “bam” or “bob”
–	Used with [] to specify a range of matching characters (given in ascending sequence)	Like “b[0-9]b” would match to “b2b” but not to “bam” Like “b[a-c]b” would match “bab”, “bbb”, and “bcb”

Figure 6.4: ANSI-89 wildcard characters

Example

To list courses where the description begins with “This course” you need a pattern where you specify that a text value begins with “This course” which can be followed by anything else: *“This course*”*.

And so, in QBE you enter the criteria for title: *Like “This course*”*:

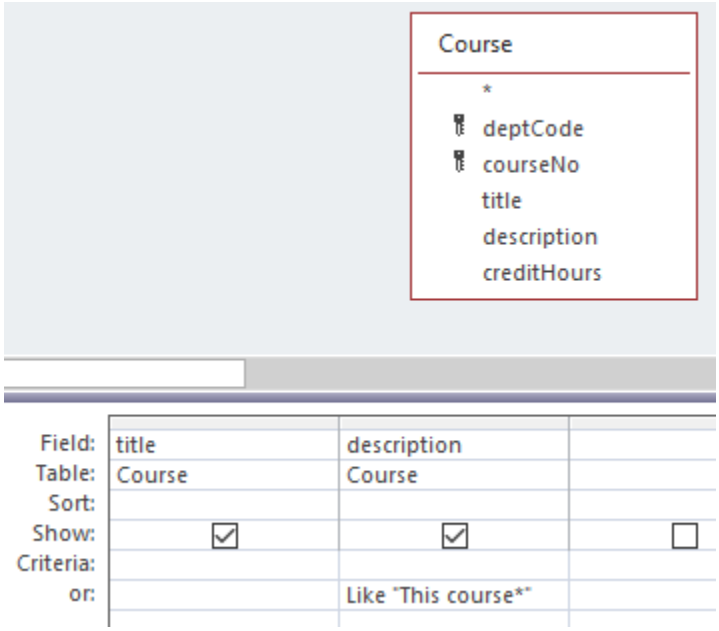


Figure 6.5: Using LIKE

6.2.2: In

The IN operator can be used if you need to determine if a field value is in a specific list of values. The list of values is a comma-separated list enclosed in parentheses; for example (1, 3, 6)

Example

To list those courses offered by the Physics, Statistics and Mathematics departments you need a list of values: (“PHYS”, “STAT”, “MATH”)

Using QBE we code IN (“PHYS”, “STAT”, “MATH”) in the criteria line:

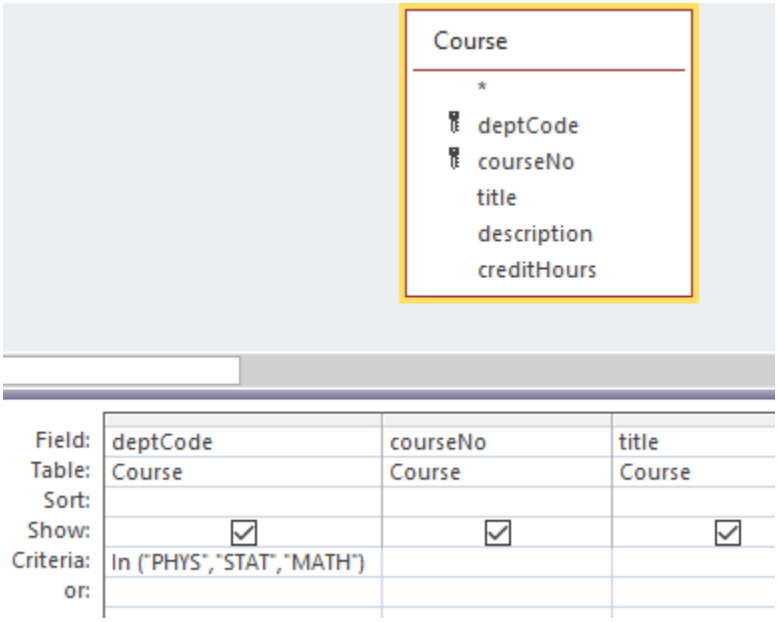


Figure 6.6: Using IN

Note: using IN is equivalent to using three simple logical expressions that are ORed, and is a convenient way of expression if there are several values in the list:

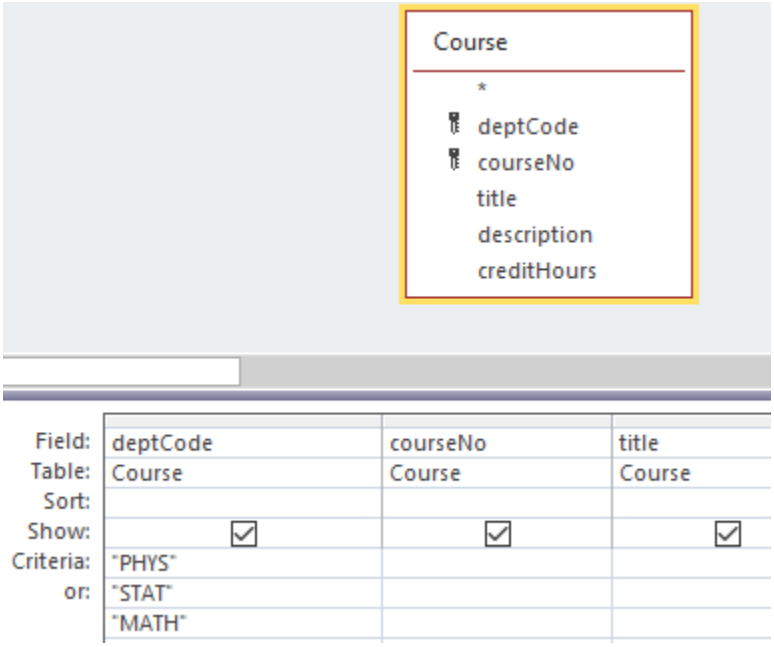


Figure 6.7: IN vs OR

Exercises

Develop queries to:

1) List courses offered by *Physics* or *Applied Computer Science* where the course description contains the word *computer*.

2) List courses where the course description contains the word *computer* but where the course is not offered by the *Applied Computer Science* department.

3) List courses where the credit hours are 1, 3, 6 or 9.

4) List courses where the credit hours are not 1, not 3, not 6, and not 9.

6.3: Query Properties

In the upper-right area of a query in Query Design View you will see a button labeled *Property Sheet*. Click this and you will see properties for a field in the Grid, or, for the query itself, depending on where the cursor is located. Click the mouse in an open area in the Relationships Diagram and you will see properties for the query. Two query properties we discuss are *Top Values* and *Unique Values*.

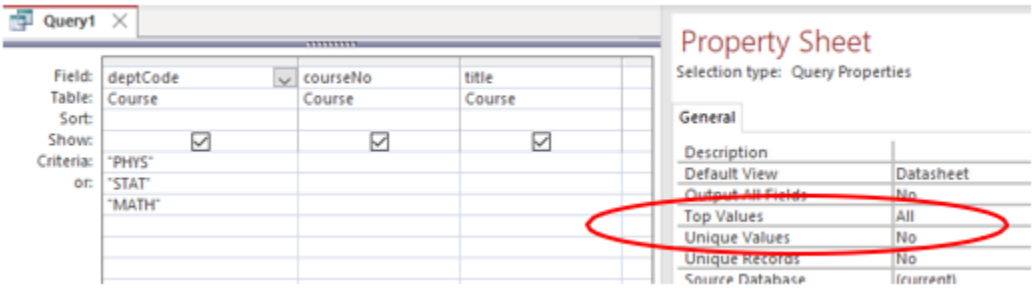


Figure 6.8: Query properties

6.3.1: Top Values

You can change the selection for Top Values. The default is ALL which results in all rows displayed when the query is run, but you can use this property to limit the rows displayed. As indicated below you can type an explicit number of rows such as 5, or a specific percentage of rows to be displayed.

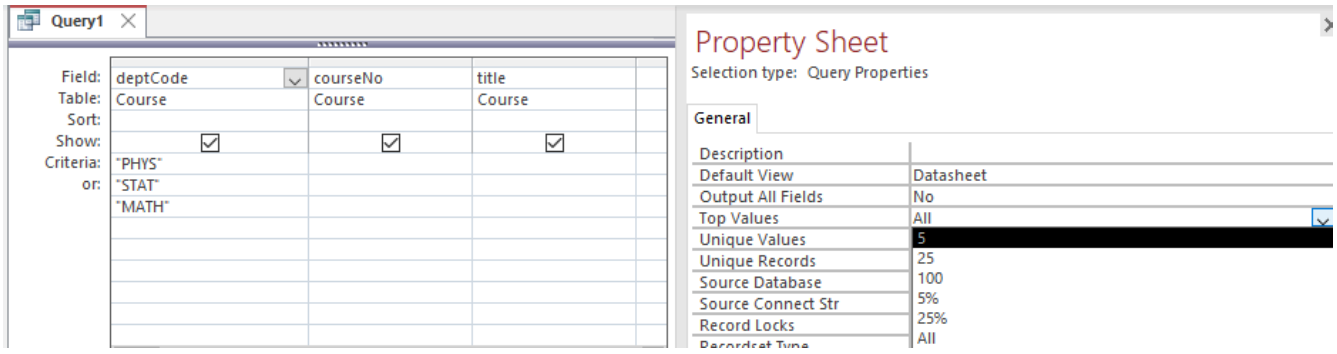


Figure 6.9: Setting the Top Values property

Example

Consider the Library database where the Member table has one row per member. Sample are shown below:

id	firstName	lastName	birthDate	gender
1	John	Smith	5/15/1999	male
2	David	Martin	6/8/2000	male
3	Betty	Freeman	9/18/1997	female
4	John	Martin	9/11/2000	male

Figure 6.10: Sample library members

Suppose we wanted to know who is the youngest member. One way to find out is to sort the members by birthdate and then pick either the first or last row according to how you ordered them (descending or ascending). Consider the following where the members are sorted in descending order by birthdate and then we list the first row by specifying *Top Values = 1*:

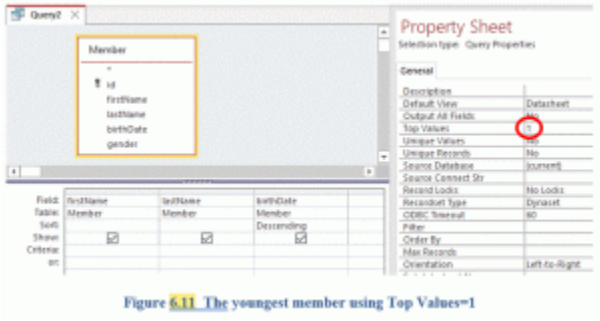


Figure 6.11: The youngest member using Top Values=1

For the sample rows the query produces the result:

firstName	lastName	birthDate
John	Martin	9/11/2000

Figure 6.12: Query returns one result row

6.3.2: Unique Values

If the Unique Values property is set to Yes, then MS Access will eliminate duplicates rows from the result.

Example

Suppose a librarian wants a list of authors from the Library database. If we use a query to list the authors but we do not set *Unique Values* to *Yes*, then the result could show an author several times, once for each of his/her books. The following result set shows Jeo Celko listed 3 times:

author
Donald Matthews
Eileen Power
Frederick Whitehead
Joe Celko
Joe Celko
Zigurd R Mednieks
T A Halpin
David Hay
Joe Celko
Lynne Elliott
Charles H Talbot

Figure 6.13: Query with duplicates

We can eliminate such duplicates by specifying *Unique Values = Yes* as in:

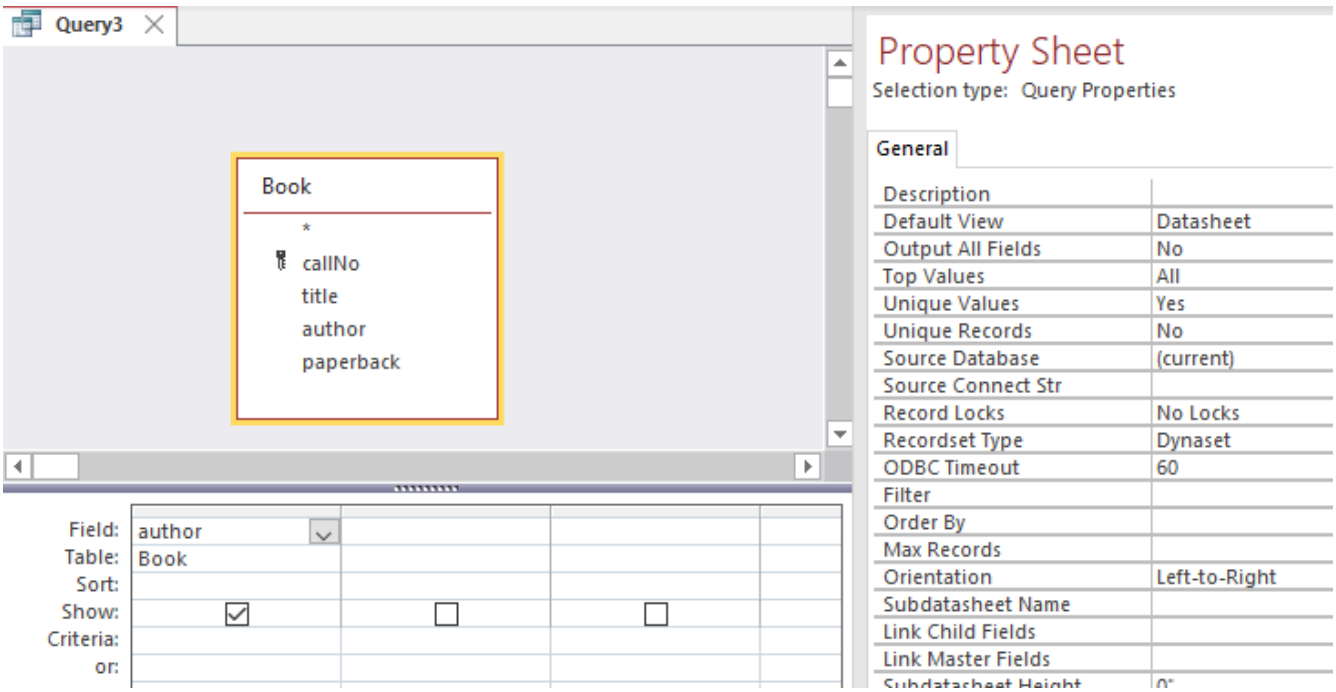


Figure 6.14: Setting Unique Value to yes

Instead of 11 names this query would only list the 9 different author names.

Exercises

1) Consider the Library database.

- Which member is the oldest?
- Which book was the first one to be taken out on loan?
- Which books have been taken out on loan? Any book listed should be listed only once – no duplicates

2) Consider the University database.

- Create a query to list the department codes (with no duplicates) of departments that offer 6 credit hour courses.
- Modify your query to list the department names too.

3) Consider the Company database and its Employee table.

- The empId field is assigned values sequentially starting at 1. What is the last empId value that was used? (What is the empId for the last employee added to the table?)
- Write a query to determine the name of the oldest employee.
- Write a query to list all of the employee last names. If at least two employees have the same last name then this list will be shorter that a list of employees.

6.4: Totals Query

A Totals query allows you to summarize information in the database. When you summarize data from one or more tables then either:

- you are producing summary data for the whole table, or
- you are producing summary data for specific groups.

For instance, you may want to know

- how many courses there are
- the average of the credit hours
- the number of courses in each department

To create a Totals query you begin by creating a simple query that retrieves all the attributes that will be needed in the summarization, and then click the Totals icon found in the upper-right hand corner of the MS Access window:

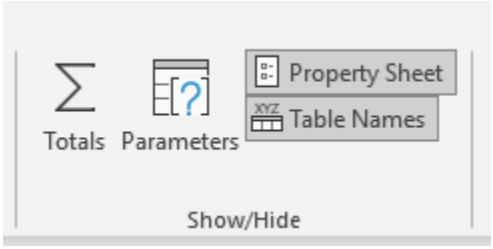


Figure 6.15: Totals icon

When you click the Totals icon the Grid changes to include a *Total* line. For each field in the grid you must choose from the drop-down:

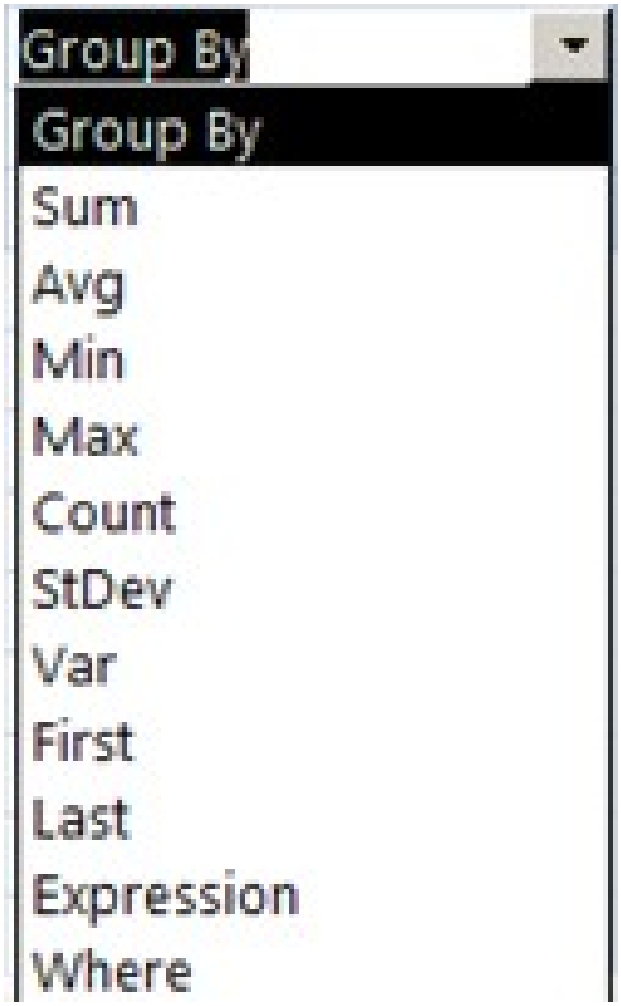


Figure 6.16: Choices for the Total line

For each field in the grid you choose one of:

- "Group By": if the field is used for grouping
- An aggregate function: if the field is to be summarized using that function. We will consider the standard set including sum, average, minimum, maximum, count.
- "Where": if the field has criteria to be used for selecting rows. Only rows satisfying the criteria contribute to the grouping and display of results.

Example

The simplest type of totalling query displays an aggregate over an entire set of rows. For example, to sum the credit hours over all courses in the University database one can use:

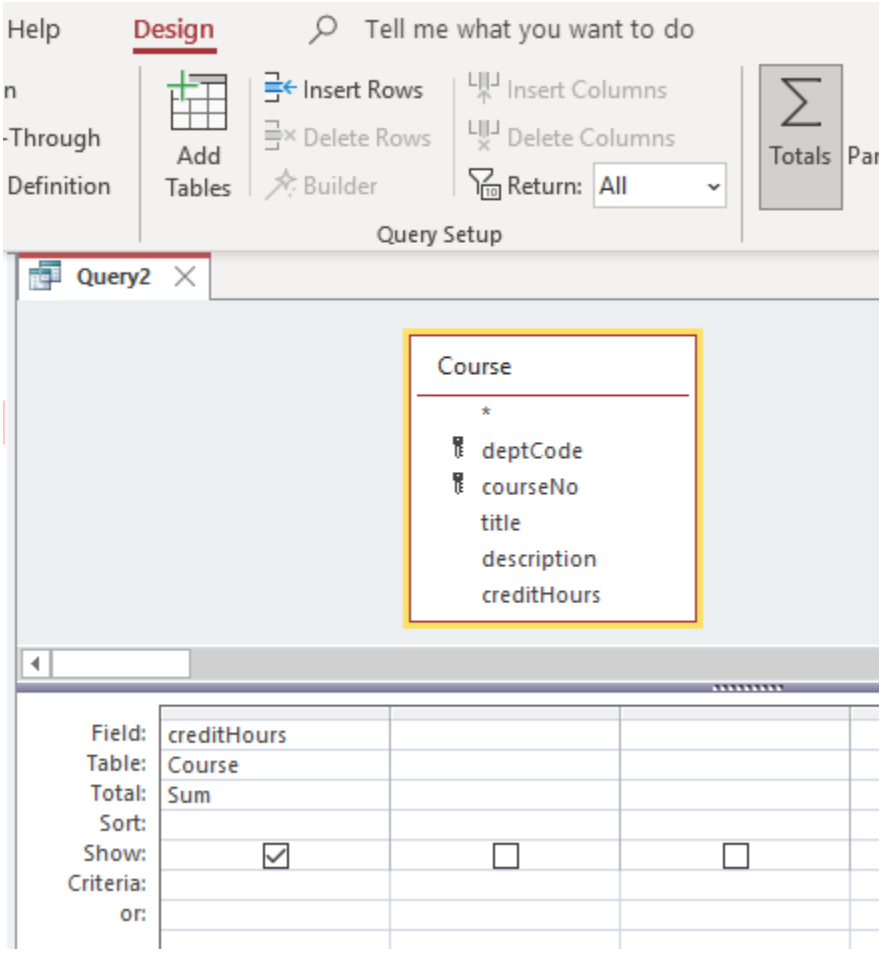


Figure 6.17: Determining the total for one field over all rows

This query summarizes the entire table. The result of this query is one line displaying a sum.

Example

Typically, the use of the Totalling feature is more complicated. Consider the University database and that we need to obtain a count of the number of courses offered by each department. We begin with a query that lists the department code and any other field of the Course table (courseNo is a good choice because it can never be null – nulls are passed over when the counting of field values is performed). The query below lists the fields we need:

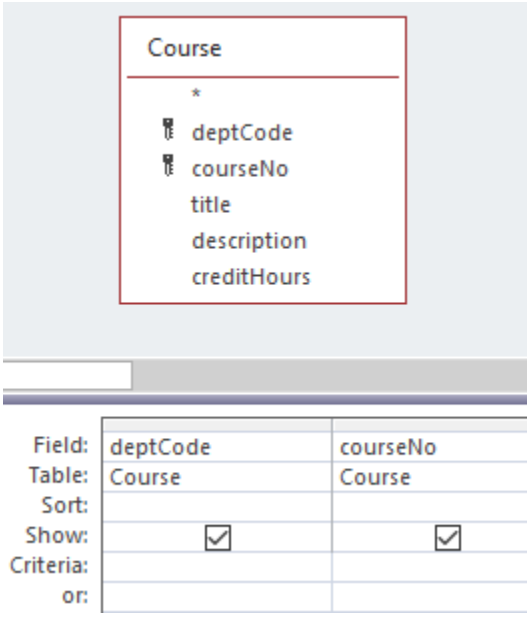


Figure 6.18: Step 1: the fields needed

In the upper right-hand corner of Design View for queries you must click the *Totals* icon. When you click the Totals icon a new line (*Total* line) is added to the grid:

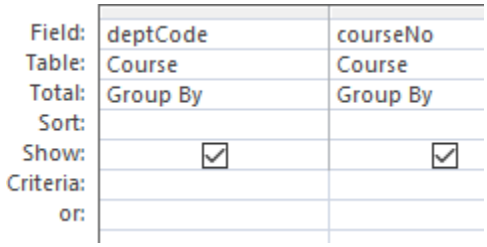


Figure 6.19: Step 2: Total line is added to the grid

By default, MS Access sets each field up for grouping. To count the number of courses in each department you must click in the Total area for courseNo and change from *Group By* to *Count*:

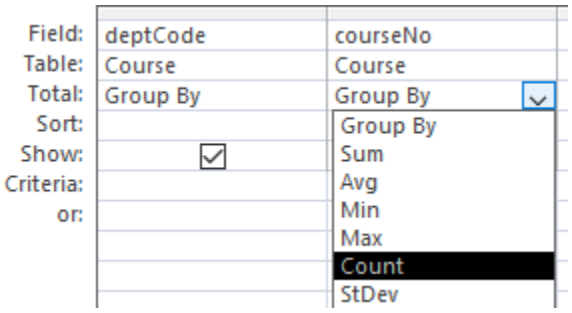


Figure 6.20: Step 3: Choose the appropriate aggregate for the group

Now you have a query that will show the value of each department code along with a count of the number of courses for the department. This query produces one row per department.

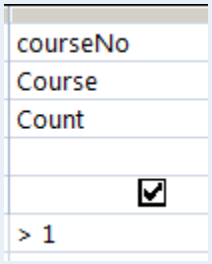
Note the choices for aggregate functions – the first 5 are part of the SQL standard: *SUM*, *AVG*, *MIN*, *MAX*, *COUNT* and perform a sum, average, minimum, maximum or count over the values found within a group. When a *Totalling* query is executed, the following actions are performed by MS Access:

1. Rows are retrieved from the underlying table(s): Recall that when *Where* is specified in the *Total* line, then there is a criteria that must evaluate to true for a row to be part of this result.
2. The retrieved rows are organized into groups where the rows forming a group have the same value for the grouping field(s).
3. For each group aggregates are evaluated.
4. A group can be eliminated from the results: If in the same column of the grid where Group by or an aggregate function is specified, there is some criteria given in the criteria line, then if the criteria evaluates to false, the pertinent group is excluded.

Exercises

Write queries for:

1) Consider the last example where the number of courses per department is listed. The sample database is small and so many departments have just 1 course. Modify the query to list results only for departments where there is more than 1 course. For this you must include a criteria >1 for the field where COUNT is specified:



2) Consider the University database:

- For each department list the department code and the largest value for credit hours.
- For each department list the department code, department name, and the number of courses.

3) Consider the Library database.

- List the number of books that have SQL in the title.

- List the number of members by gender.
- What is the total for fines?

4) Consider the Company database

- List the number of employees in each department.
- List departments that have more than 25 employees.
- For each employee who is a supervisor, list the supervisor name and the number of employees they supervise.
- Suppose the Employee table has a salary field holding an employee's salary. What is the average salary?

6.5: Parameter Query

If you need a query but the criteria will not be known until runtime you use a Parameter Query. A parameter query is one where, on the *Criteria* line for some field, one types square braces [ ] and inside the [ ] one types a prompt for the user who runs the query. When a user runs a parameterized query, MS Access will show the user the prompt and waits for the user to respond with a value for the parameter – MS Access replaces parameters with the user-supplied values just before it executes the query.

Example

Suppose a user in the University database needs a list of courses having a specific value for credit hours. The query below has a parameter in the criteria line for creditHours:

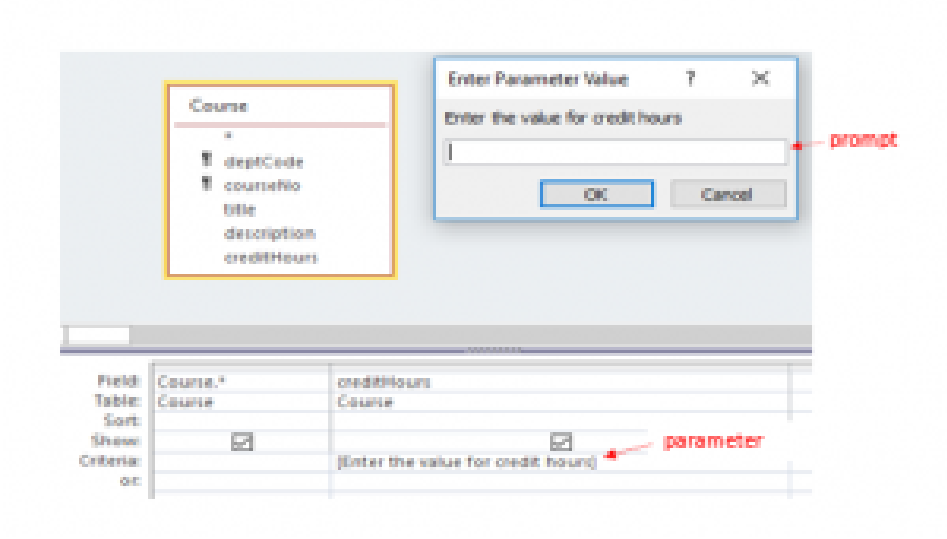


Figure 6.21: Parameter query

When the query is run, the query is temporarily suspended while the user is prompted with the message as given in the square braces [ ]. Once the user responds to the prompt, the running of the query continues with the value the user entered as the criteria value.

Exercises

1) Consider the University database:

- Create a query to list all courses in a department (for which the user supplies the department code).
- Create a query to list all course titles where the user supplies both the department code and the credit hours. Note that two separate criteria, each with their own parameter, must be specified.

2) Consider the Company database:

- Write a query to list the employees who manage a department where the department code is provided by the person running the query.
- Write a query to list all employees in some department where the department code is provided by the person running the query.
- Modify the employee data in the Company database so at least two employees have the same first and last names. Develop a query that lists all employees having a specific first name and last name that will be specified by the end user.

3) Consider the Genealogy database:

- Create a query with two parameters: a *start date* and an *end date*. The query will list all persons whose birth dates fall in the range from *start date* to *end date*.

4) Consider the Library database:

- Write a query to list books due on a specific date (a parameter).
- Write a query to list books written by a specific author (a parameter).

6.6: Crosstab Query

Standard MS Access queries produce results with column headings. Crosstab queries are queries where results are displayed with both row and column headings.

We limit our discussion to the use of the Crosstab Query Wizard for creating crosstab queries.

Example

As an example, suppose we wish to display for each department a count of the number of 3 and 6 credit hour courses. The counts are to appear in matrix format where rows are labeled with department names and are columns appear with labels 3 and 6. Below is an outline of how the results should appear:

Course	3-credit hours	6-credit hours
Chemistry	16	7
Mathematic	22	11
...	...	...

Figure 6.22: Query results to appear with row and column headings

Crosstab queries have at least three fields; one field (department) is used for row labels, another field (credit hours) is used for column labels, and one field (course number) is used with an aggregate function (Count).

We can begin by creating a Simple query that retrieves all the necessary values:

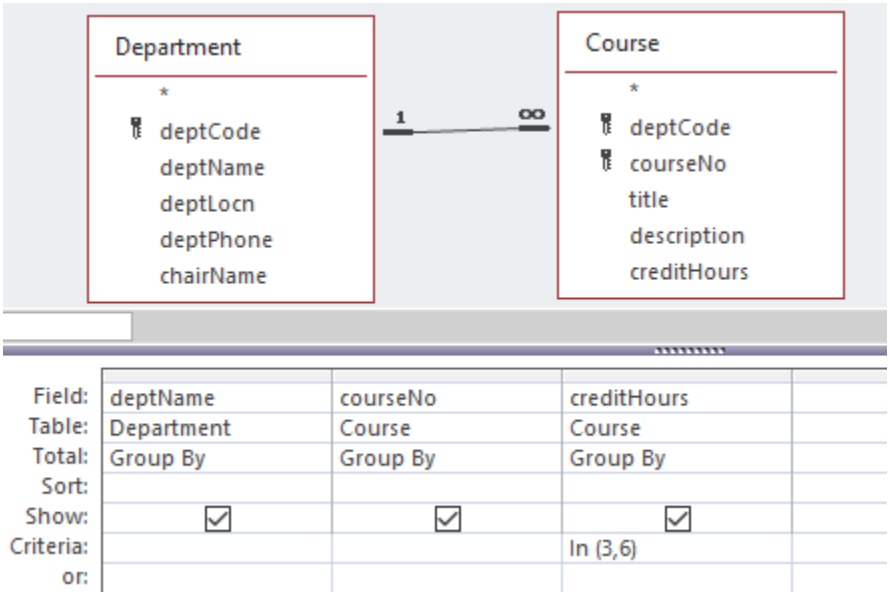
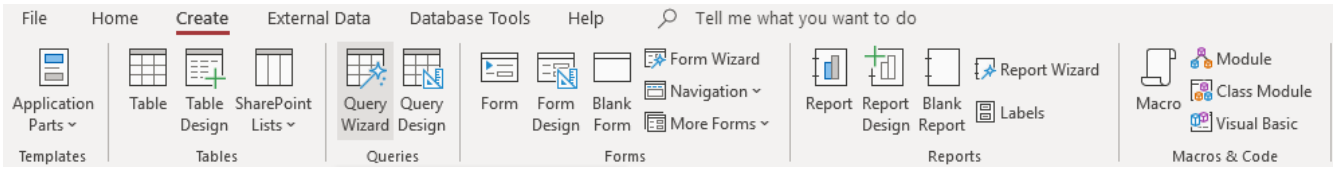


Figure 6.23: Query with required fields

Next, we save the query (say Q1) and create a new query – click on the Query Wizard:



The wizard prompts for

- type of query – choose Crosstab Query Wizard
- the table/query to use as the basis for the new crosstab query (Select Queries and then Q1 – the query previously saved)
- the field to use for row labels \ deptName
- the field to use for column labels \ creditHours
- the field (courseNo) and the aggregate function (Count) to use for summarizing data.

Running the query shows several columns: the department name (values in this column are the row labels), total over the remaining columns for the row, columns for credit hour values 3 and 6 (the column labels). A sample run:

Dept Name	Total Of courseNo	3	6
Applied Computer Science	5	4	1
English	3	2	1
Mathematics	1	1	

Figure 6.24: Standard crosstab results

Exercises

1) Create and run the query to display for each department a count of the number of 3 and 6 credit hour courses.

2) Modify the query so that credit hour values appear as row labels and department names appear as column labels.

6.7: Action Queries

Action Query is a category that MS Access uses to distinguish queries that can modify the data in the database. We discuss the query types: Make-Table, Append, Delete, and Update. To create such a query, one typically starts with a Simple Query that is subsequently changed (by clicking the pertinent button) to an Action Query type. You will notice as you experiment running action queries that MS Access gives a warning message asking you to confirm the changes the query will make to the database. The reason for this is that you cannot click some Undo button to undo such changes as you can in other Office applications. To undo a database action query, you need to design and execute a compensating action query.

When you are in Design View for some query you will see the buttons for changing the query type:

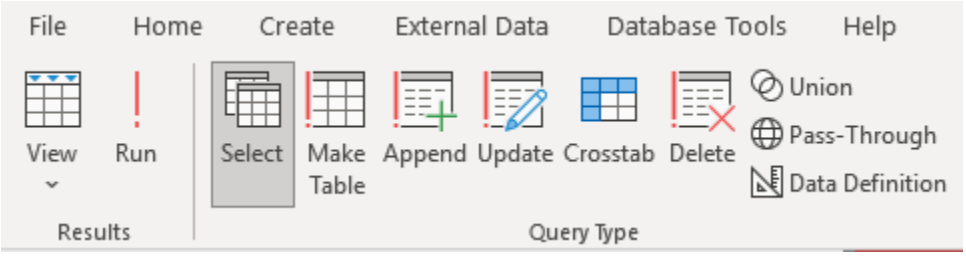


Figure 6.25: Types of queries

6.7.1: Make Table Query

Make-table queries are useful if you want to use existing data when you create a new table.

Consider the University database and suppose we need to create a table of ACS courses. We would start with a query that retrieves all ACS courses:

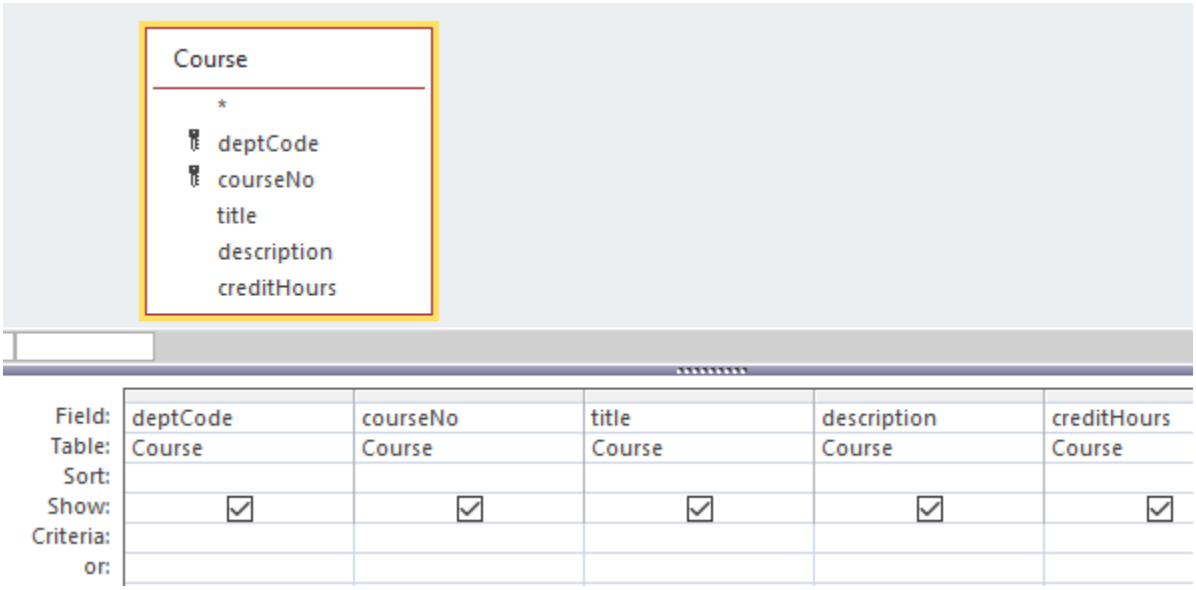


Figure 6.26 Begin with a select query

Next, we change the query to a Make-Table Query by clicking the Make Table button. When you do this, MS Access will prompt you for the name for your new table:

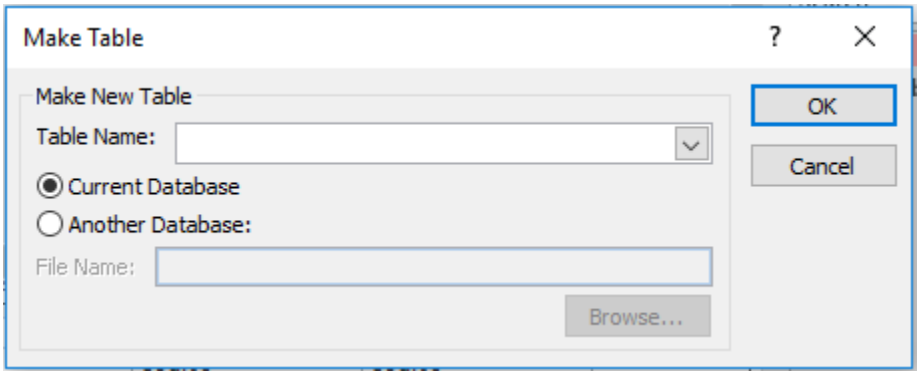


Figure 6.27: Prompt for table name for Make-Table query

The query does not run yet; you must either click the Run button or save the query and run it later. Each time you run the query MS Access will empty the table and insert rows into it.

6.7.2: Append Query

Suppose you wish to add rows to an existing table. To do that you must use an Append query. To create an Append query begin by creating a Simple query that lists the information you wish to see inserted to the table. Once you know the query retrieves the proper information click the Append button and MS Access will prompt you for the table name that should receive the new rows. After this you can run the query from the Run button, or you can save the query and run it later.

6.7.3: Delete Query

To remove entire rows from a table you use a Delete query. As with previous query types discussed, you can begin with a Simple query that retrieves the rows you wish to delete. Once the Simple query is working you can change its type to Delete and run the query (or save it and run it later). Be careful with delete queries, as they can delete many rows in a single run.

6.7.4: Update Query

The type of query used to modify existing rows in a table is the Update query. In order to create such a query, you should begin with a Simple query that retrieves the rows that are to be updated and then change the type to Update. When you change the type to Update MS Access will add a new row to the Grid area where you specify the new values for each field to be updated. The new value can be the result from a calculation.

Example

Suppose we wish to update the course Table so the credit hours are doubled for each ACS course. We begin with a Simple query to retrieve the PK field, the fields to be updated, and the fields needed for selection criteria purposes. In this case we will need a Simple query to retrieve the department code, course number, and credit hours fields:

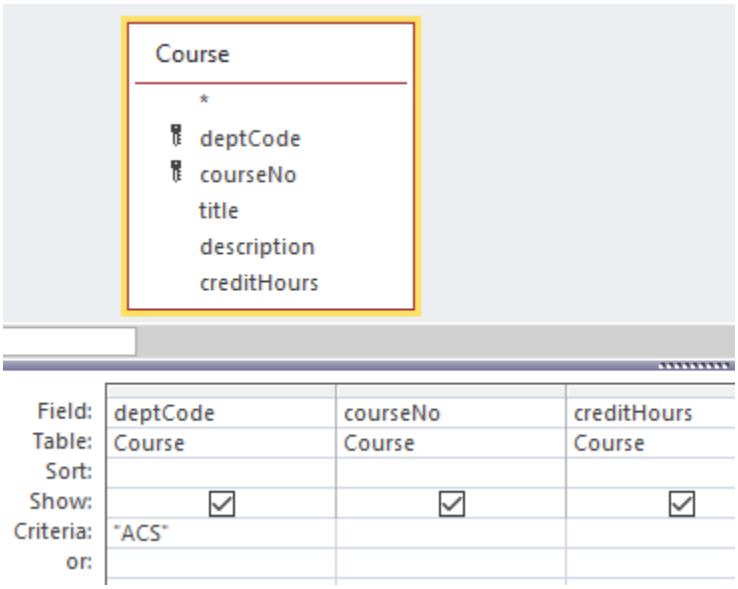


Figure 6.28: Simple select query

Next, we change the query type to Update and MS Access modifies the Grid to include an *UpdateTo* line. On that line we enter an expression that generates the new values. To double the credit hours, we need the expression [creditHours]*2, as in:

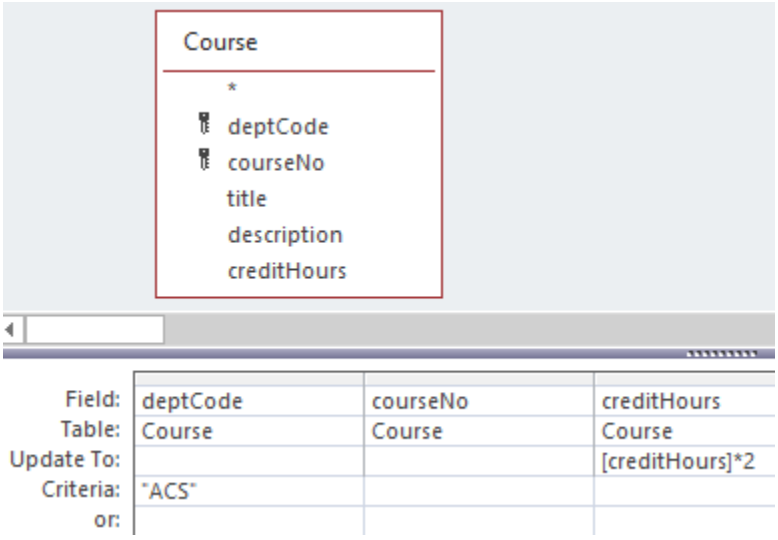


Figure 6.29: Update query with Update To line

Exercises

1) Create a table of ACS courses, but name the new table ScienceCourses.

2) Does the table ScienceCourses have a primary key? If not, create one.

3) Run a delete query on ScienceCourses to delete all non-3-credit hour courses.

4) Append all 3-credit hour MATH courses to ScienceCourses.

5) Run an update query on ScienceCourses to double the credit hours of all 3-credit hour courses.

6.8: Inner and Outer Joins

Whenever we use a query to retrieve data from two or more tables, the database query processor performs an operation called a *join*. In this section, we discuss inner joins, outer joins, and Cartesian products. Following that we discuss some interesting special cases: self-join, anti-join, non-equi joins.

If we have previously established relationships between tables, and if we have more than one table in a query, then MS Access will create *joins* based on those relationships. If necessary, we can alter, delete, or include new relationships.

MS Access creates joins where rows join if the join fields are equal in value; such joins are called *equi*-joins. If we create a query for the University database and add the Department and Course tables to the relationships area of the query we have:

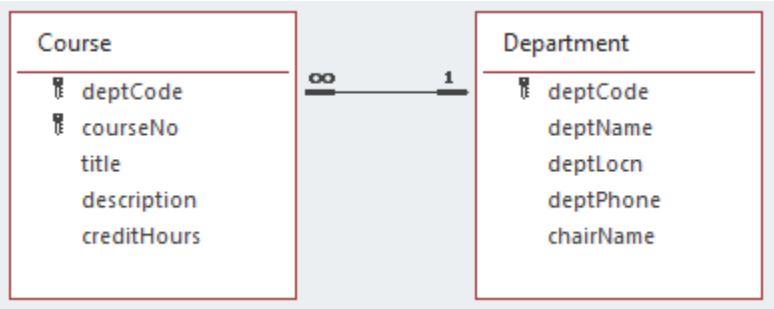


Figure 6.30: Standard equi-join

If you edit the relationship line (double-click it), you see the join properties:

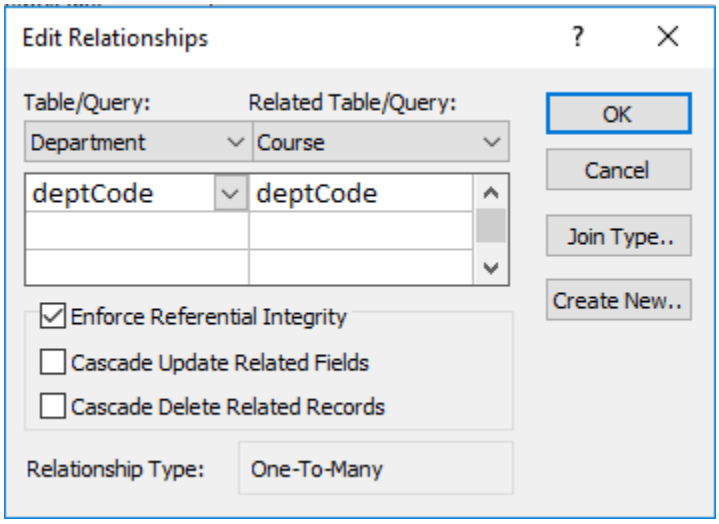


Figure 6.31: Join properties

Here, we can see the join is based on the common attribute deptCode. If you click on the Join Type button, you will get information on the type of join used. You will see (as the following diagram shows) that Access has selected the first of three options:

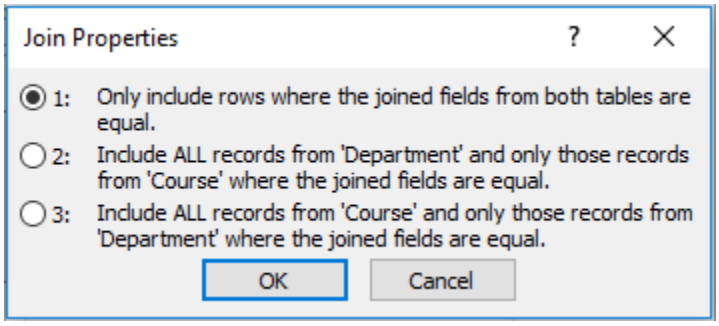


Figure 6.32 Choosing inner join or outer join

Joins can be further characterized as *inner* or *outer* joins. Option 1 is an inner join.

Options 2 and 3 are outer joins. One of these would also be called a *Left Outer Join* and the other a *Right Outer Join*. If you examine the SQL statement generated you will see which is used; Left and Right choices are related to the textual expression of the SQL statement – which table name is leftmost/rightmost in the From clause.

6.8.1: Inner Join

All of the joins we have seen up to this point have been inner joins. For a row of one table to be included in the result of an inner join, the row must match a row in the other table. Because all joins so far have also been equi-joins the matching is based on the values of the join fields of one table being equal to the values of the join fields of the other table.

Consider the inner join between Department and Course based on deptCode:

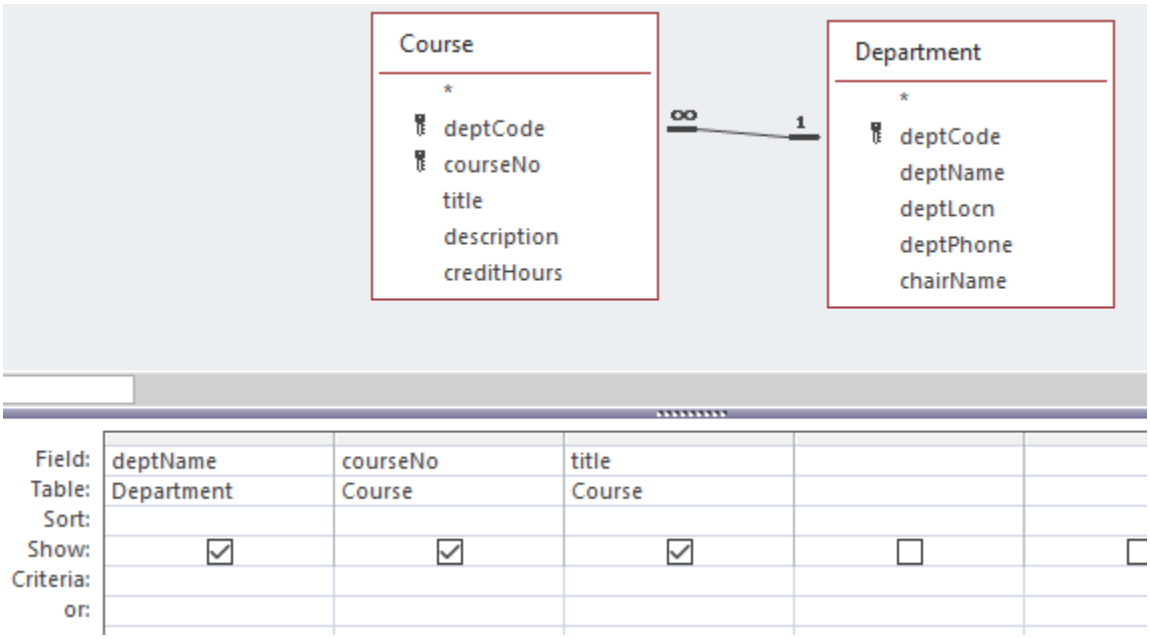


Figure 6.33: Inner join

If the tables have the contents shown below:

Course

Dept Code	Course Number	Title	Description	Credit Hours
ACS	1453	Introduction to Computers	This course will introduce students to the basic concepts of computers: types of computers, hardware, software, and types of application systems.	3
ACS	1803	Introduction to Information Systems	This course examines applications of information technology to businesses and other organizations.	3

Dept Code	Dept Name	Location	Phone	Chair
ACS	Applied Computer Science	3D07	(204) 786-0300	Simon Lee
ENG	English	3D05	(204) 786-9999	April Jones
MATH	Mathematics	2R33	(204) 786-	Peter
			0033	Smith

Figure 6.34: Table contents

then the result of running the query is:

Dept Name	Course Number	Title
Applied Computer Science	1453	Introduction to Computers
Applied Computer Science	1803	Introduction to Information Systems

Figure 6.35: Query result

In the above result, notice there is no result line for English or Mathematics; this is because for the sample data there were no rows in Course that joined to the English or Mathematics rows in Department. Both rows in Course have a value of "ACS" in the deptCode field and so they joined to the ACS row in Department.

This query demonstrates a distinguishing characteristic of the inner join: only rows that match other rows are included in the results.

Exercises

1) Consider the Library database:

- Write a query that joins Loan and Member. List the member name and date due.
- Write a query that joins Loan and Book. List the book title and date due.
- Write a query that joins all three tables and lists the member name, book title, and date due.

2) Consider the two tables A and B below.

Table A

X	Y	Z
1	3	5
2	4	6
4	9	9

Table B

X	Y	Q
1	3	5
1	4	6
2	4	7
3	4	5

- How many rows are in the result if A and B are joined based on the attribute X?
- How many rows are in the result if A and B are joined based on both attributes X and Y?

6.8.2: Outer Join

Consider the Company database. Suppose we wanted to produce a report that lists each department and its employees, and that we must include every department. The two tables would be joined based on equal values of the dept id field. We want all departments and we know that an inner join will not include a department if there are no employees for the department to join to. To get all departments included when we are joining two tables, we must use an *outer* join.

Consider the query that is started below:

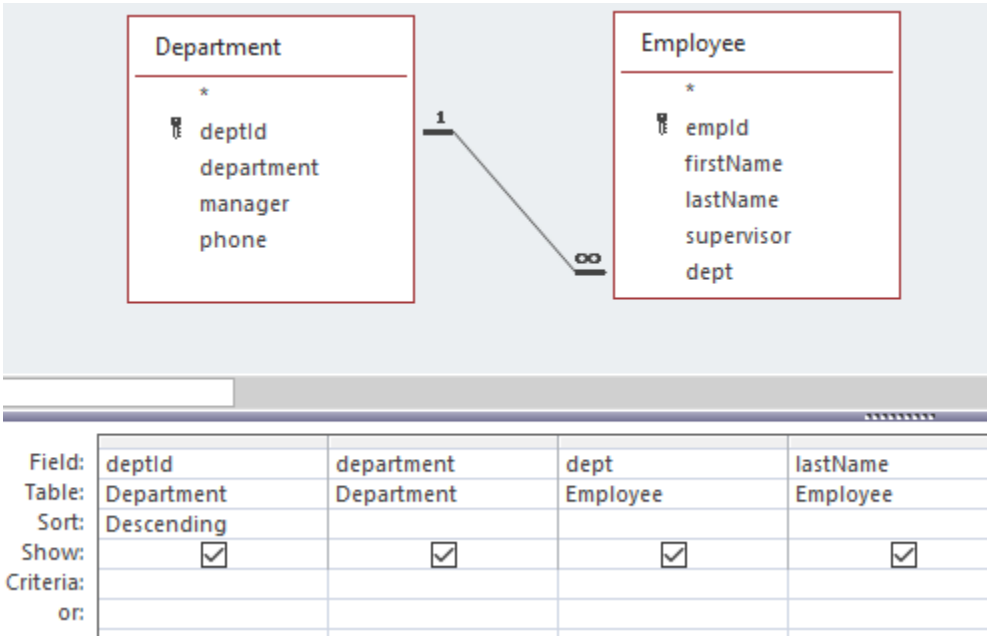


Figure 6.36: Initial query

By default the join is an inner join, but with MS Access you can get an outer join if you edit the relationship and specify either option 2 or option 3, as shown in the dialogue below: