

Visual Basic and Databases

2. Introduction to Databases

Review and Preview

- In the last chapter, we looked at a database in very general terms. We learned that the heart of any Visual Basic database application is the Jet database engine. In this chapter, we provide more details into the structure of databases and how they are created. We will use a sample database to illustrate the concepts presented.

Database Structure and Terminology

- In simplest terms, a **database** is a collection of information. This collection is stored in one or more well-defined **tables**, or matrices.
- The **rows** in a database table are used to describe similar items. The rows are referred to as database **records**. In general, no two rows in a database table will be alike.
- The **columns** in a database table provide characteristics of the records. These characteristics are called database **fields**. Each field contains one specific piece of information. In defining a database field, you specify the data type, assign a length, and describe other attributes. Some field types include Binary, Boolean, Counter, Double, Single, Long, Integer, etc.
- Here is a simple database example:

Field				
ID No	Name	Date of Birth	Height	Weight
1	Bob Jones	01/04/58	72	170
2	Mary Rodgers	11/22/61	65	125
3	Sue Williams	06/11/57	68	130

Record

Table

In this database **table**, each **record** represents a single individual. The **fields** (descriptors of the individuals) include an identification number (ID No), Name, Date of Birth, Height, and Weight.

- Most databases use **indexes** to allow faster access to the information in the database. Indexes are sorted lists that point to a particular row in a table. We can create an index for any field we might want to perform a search on. The neat thing about an index is that the Jet database engine (included with Visual Basic) handles all the details. We simply flag a field as an index and the Jet engine does the work.
- A database using a single table is called a **flat database**. Early database software worked only with flat databases. And, for simple applications, flat databases may be adequate. For large amounts of data, however, flat databases are cumbersome and become very large, very quickly.

Relational Databases

- Most databases are made up of many tables stored in a single file. Each table contains a logical grouping of information with its own records and fields. When using multiple tables within a database, the tables must have some common fields to allow cross-referencing of the tables. The referral of one table to another via a common field is called a **relation**. Such groupings of tables are called **relational databases**.
- Relational databases allow us to store vast amounts of data with far simpler maintenance and smaller storage requirements than the equivalent flat database. As an example, say we had a flat database listing products stocked by a grocery store with several fields describing each product's manufacturer (manufacturer name, address, phone, ...). If you have 1,000 products made by the same manufacturer, there is much repetition of information in the flat database. And, if the manufacturer changed their phone number, you would have to make that change in 1,000 places! In a relational database, you could use two tables, one for products, one for manufacturers. In the product table, you would simply have a manufacturer ID that would correspond with an ID in the manufacturer table (a **relation**), which would have that manufacturer's information. Then, if the phone number changed, you would only have to change one field in the manufacturer table - quite a savings in work! When you break down database tables into simpler tables, the process is known as **database normalization**.
- Relations among tables in a relational database are established using **keys**. A **primary key** is a field that uniquely identifies a record so it can be referenced from a related table. A **foreign key** is a field that holds identification values to relate records stored in other tables.
- When one record in one table is linked to only one record in another table, we say there is a **one-to-one** relation. When one record in one table links to many records in another table, we say there is a **one-to-many** relation. And, when many records in one table are linked to many records in another table, we say there is a **many-to-many** relation.
- In the first few chapters in this course, we will use a sample database that comes with Visual Basic. This relational database (**BIBLIO.MDB**) is found in the main Visual Basic directory - you will become very familiar with this database. It is a database of books about computer programming (and databases). Let's look at its relational structure to illustrate the many new concepts being introduced.

Sample Relational Database

- The books (BIBLIO.MDB) database is made up of four tables:

Authors (6,246 records)

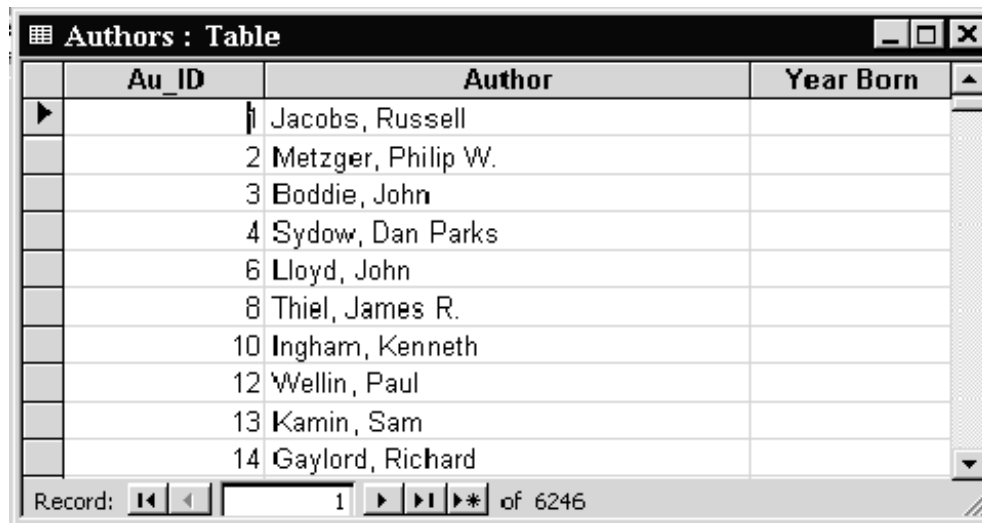
Publishers (727 records)

Titles (8,569 records)

Title Author (16,056 records)

As you look at each table, pay attention to how the tables are logical groupings of information. Examine the record and field structures. In particular, note each field with an 'ID' in the name acts as a key to relate one table to another.

- The **Authors** table contains information about the authors of the books in the database. The table has three (3) fields: Au_ID, Name, and Year Born:



Au_ID	Author	Year Born
1	Jacobs, Russell	
2	Metzger, Philip W.	
3	Boddie, John	
4	Sydow, Dan Parks	
6	Lloyd, John	
8	Thiel, James R.	
10	Ingham, Kenneth	
12	Wellin, Paul	
13	Kamin, Sam	
14	Gaylord, Richard	

There are 6,246 different authors in the database.

- The **Publishers** table contains information about the publishers in the book database. The table has ten (10) fields: PubID, Name, Company Name, Address, City, State, Zip, Telephone, Fax, and Comments:

	PubID	Name	Company Name	Address
▶	524	A K PETERS	A K PETERS LTD	
	518	A SYSTEM PUBNS	A SYSTEM PUBNS	
	499	A-R EDITIONS	A-R EDITIONS	
	116	AA BALKEMA	AA BALKEMA	
	242	AARP	AMER ASSN OF RETIRED PERSONS	
	97	ABACUS	ABACUS SOFTWARE	
	616	ABC TELETRAINING	ABC TELETRAINING	
	214	ABC-CLIO	ABC-CLIO	
	102	ABLEX	ABLEX PUB CORP	
	229	Ablex Pub	Ablex Pub	

Record: 1 of 727

There are 727 different publishers in the database.

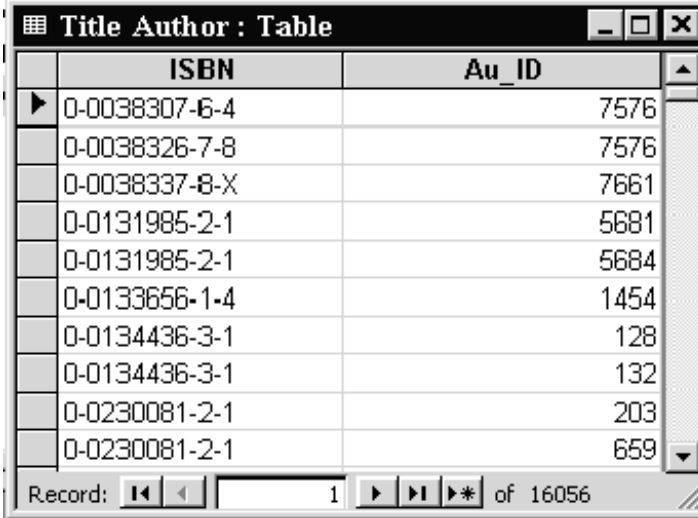
- The **Titles** table contains information about each book title in the database. The table has eight (8) fields: Title, Year Published, ISBN, PubID, Description, Notes, Subject, and Comments:

	Title	Year Published	ISBN	PubID
▶	1-2-3 Database Techniques	1990	0-8802234-6-4	45
	1-2-3 For Windows Hyperguide/Book and Disk	1993	1-5676127-1-7	192
	1-2-3 Power MacRos/Book and Disk	1992	0-8802280-4-0	45
	1-2-3 Power Tools (Bantam Power Tools Series)	1991	0-5533496-6-X	135
	1-2-3 Release 2.2 PC Tutor/Book and Disk	1990	0-8802262-5-0	45
	1-2-3 Secrets/Book and Disk	1993	1-8780587-3-8	15
	10 Minute Guide to Access	1994	1-5676123-0-X	192
	10 Minute Guide to Access (Best Selling)	1994	1-5676145-0-7	192
	10 Minute Guide to Access for Windows 95	1995	0-7897055-5-9	45
	10 Minute Guide to Act! for Windows	1995	1-5676153-9-2	192

Record: 1 of 8569

There are 8,569 distinct book titles in the database.

- The **Title Author** table contains information relating book titles to authors within the database. It has just two fields: ISBN (International Standard Book Number, a number used by bookstores and libraries to reference books) and Au_ID:



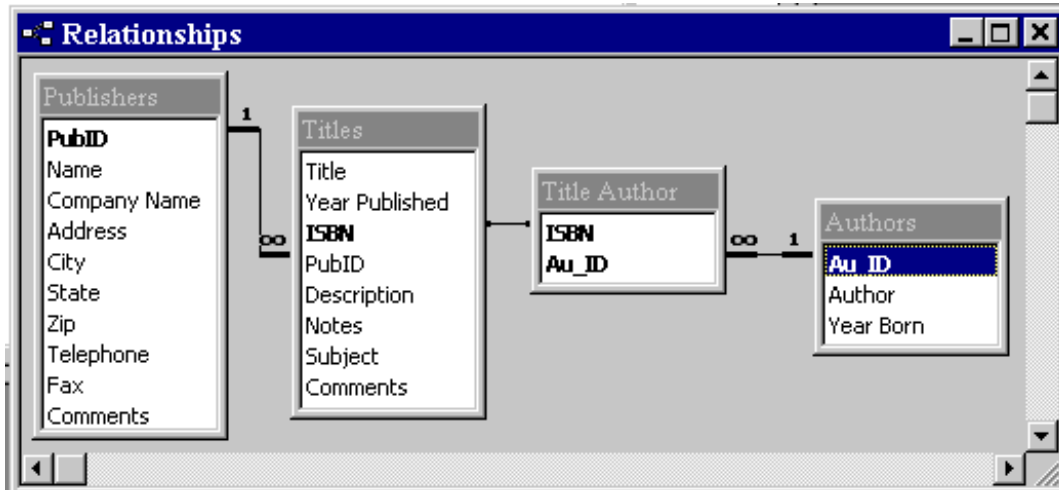
ISBN	Au_ID
0-0038307-6-4	7576
0-0038326-7-8	7576
0-0038337-8-X	7661
0-0131985-2-1	5681
0-0131985-2-1	5684
0-0133656-1-4	1454
0-0134436-3-1	128
0-0134436-3-1	132
0-0230081-2-1	203
0-0230081-2-1	659

There are 16,056 entries in this table. You may wonder - if there are 8,569 titles in the database, how can there be nearly twice as many entries in this table. The answer is that many books have more than one author and this table lists all the authors for each title.

- There is obviously a lot of information in the books database! This example, though, is very useful and shows the kind of database we can work with using Visual Basic. It is a well-designed database we can learn from. We will discuss database design in a later chapter, so much of what is discussed here will be very useful information later on. You may be wondering – where did these views of the database tables come from? They were obtained using Microsoft Access. In a couple of more chapters, you will be able to obtain such views using Visual Basic without writing a single line of code!

Sample Database Structure

- Let's examine the books database a little closer. To help, we'll use this block diagram (obtained using Access) that illustrates the database structure:



This diagram shows each table as a separate window listing the corresponding fields. Relations between tables are illustrated via linear links.

- Look at the books database tables. Note each table is a logical grouping of information. Book publishers are in a single table (**Publishers**), book titles are in a single table (**Titles**), and book authors are in a single table (**Authors**). A well-designed database has such well-defined tables. Well-defined tables make database management a far simpler task.
- Note each table has two types of information: **source** data and **relational** data. Source data is actual information, such as names, phone numbers, and addresses. Relational data are references to data in other tables via **keys**, such as **PubID**, **ISBN**, and **Au_ID**.
- A **primary** key defines a unique record. **PubID** in the **Publishers** table, **ISBN** in the **Titles** Table, and **Au_ID** in the **Authors** table are primary keys. They identify a unique entry in their respective table.
- A **foreign** key is a piece of relational information in one table that links to information in another table. In the **Titles** table, **PubID** is a foreign key. Using a PubID from this table in conjunction with the PubID primary key in the **Publishers** table will provide us with complete information about a particular publisher. In the **Title Author** table, ISBN and Au_ID are foreign keys.

- How the keys are used in the database is shown via the linear links. For example, **PubID** (a primary key) in the **Publishers** table relates to the **PubID** (a foreign key) in the **Titles** table. The one (1) next to PubID in the Publishers table and the infinity symbol (∞) next to PubID in the Titles table show this is a **one-to-many** relationship. That is, there is one PubID in the Publishers table, but this value may appear many times in the Titles table.
- There is also a one-to-many relationship between **Au_ID** (primary key) in the **Authors** table and **Au_ID** (foreign key) in the **Title Author** table. The relationship between **ISBN** in the **Titles** table and **ISBN** in the **Title Author** table cannot be determined by Access (indicated by no markings on the linear link). Such indeterminate links will happen occasionally.

Virtual Database Tables

- The primary purpose of the books database (BIBLIO.MDB) is to track information about book titles. Note each table gives us a piece of information about a particular book, but to get all the information about a book, we need all four tables.
- Using the relational data in the four tables, we should be able to obtain a complete description of any book in the database. Let's look at one example. Look back at the **Titles** table and note the first book (a record) listed has this information:

Title	1-2-3 Database Techniques
Year Published	1990
ISBN	0-8802234-6-4
PubID	45
Description	29.95
Notes	650.0285536920
Subject	[Blank]
Comments	HF5548.4.L67A52 1989

Taking the **ISBN** into the **Title Author** table will provide us with an Au_ID:

Au_ID	2467, 5265, 5266
-------	------------------

The book has three authors. Using these **Au_ID** values in the **Authors** table reveals author information:

Au_ID=2467

Author	Stern, Nancy
Year Born	[Blank]

Au_ID=5265

Author	Weil, Bill
Year Born	[Blank]

Au_ID=5266

Author	Anderson, Dick
Year Born	[Blank]

A last relational move of using the **PubID** in the **Publishers** table will give us complete details about the book publisher:

Name	QUE CORP
Company Name	QUE CORP
Address	11711 N College Ave, Suite 140
City	Carmel
State	IN
Zip	46032
Telephone	[Blank]
Fax	[Blank]
Comments	[Blank]

- Once done, we know everything there is to know about this one particular book “1-2-3 Database Techniques.” What we essentially have done is formed one huge table with a single record and many, many fields. This new view of the data in the database is called a **virtual database table**. It is virtual because it doesn’t exist as a native table in the BIBLIO.MDB database – it was formed using the native four tables.

- Making a **query** of the database created a virtual table above. We asked the database to tell us everything it knew about the book “1-2-3 Database Techniques.” The database responded (well, we really did the work) with all information from its four tables. This is a very common task in database management systems and one we will be doing often in this course, **querying the database**. With each query of the database, we form a virtual table that contains the results of our query. Our queries will not be as comprehensive as the one made here (show me everything!). Usually, the query will ask for all records that meet some particular criteria. As an example, we might like to query the books database to show us all books published by a specific company. The results of this query would be returned in a virtual table.
- Database queries are made with a specific language named **SQL** (structured query language). We will study SQL in a later chapter. For now, be aware that SQL can be used to form virtual tables from a database. These tables show us information of interest from the database. And, with Visual Basic as the front-end, doing a query with SQL is simple. We form the query, pass it on to the Jet database engine (via the data control) and the engine does all the work for us, returning all records that our query requested. It’s like magic! In the first few chapters, we will doing just that – opening the books database and forming virtual tables we can view.

Creating a Database

- Before leaving this database introduction, you may be asking yourself – how are databases like the books database created? How are tables defined? How are fields defined? How are records created?
- Databases are created using commercial applications like Access, dBase, FoxPro, Oracle, and others. Each of these products has a design mode where you define a table and the fields that are part of the table. You can also enter records into the table using these applications. The books database was built with Access. In the first part of this course, we will work with existing databases and will not be concerned with creating a database.
- Later chapters discuss proper database design and creation of databases. It is possible to create databases with Visual Basic (we’ll look at how to do this in the final chapter). For now, when we need to create our own database, we will use a product shipped with Visual Basic called the **Visual Data Manager**. It is a fairly easy-to-use application that will suit our needs quite well. If you know how to use Access, you could also use that when the time for creating a database arises.

Summary

- In this chapter, we looked at our first database – the books database (BIBLIO.MDB) that is included with Visual Basic. We studied the structure of a relational database, discussing tables, records, and fields. Relationships using primary and foreign keys were illustrated.
- The concept of a virtual table was introduced. Making a query of the database forms virtual tables. In the next chapter, we begin learning how to use Visual Basic to connect to a database and process queries to form such virtual tables.

This page intentionally not left blank.