Inside Relational Databases
with Examples in Access
Contents

Preface  xiii

Chapter 1 • Introduction  1
  Who are we?  1
  What is a database?  2
  Databases vs. Database Management Systems  3
  Relational Database Management Systems  3
  Why this book?  4
  Who should read this book?  5
  Organization of the book  6
  Some ground rules  7
  Downloading files from the website  8
  Acknowledgements  8
  We don’t have problems …  9
  Outroduction  9

Part 1 – A simple, single-table database  11

Chapter 2 • Introduction to Part 1  13
  Tables  13
  Queries/Views  14
  Forms  14
  Reports  15
Chapter 3 • Tables  17
  Rows & columns – records & fields  18
  Building a table  22
  Types of data  23
  Meaningful operations  24
  Excluding certain errors  26
  Making storage more efficient  26
  Making data recall more rapid  28
  Field size  28
  General notes on table design  29

Chapter 4 • Queries/Views  36
  Queries usually find subsets of the data  36
  Queries, answer tables and base tables finally defined properly
    and closure mentioned briefly  37
  Summarizing data  42
  Other useful queries  42
  Graphical querying tools  43
  SQL and Views  44

Chapter 5 • Forms  45
  Multiple forms per table  48
  Text boxes can be made read only  49
  Text boxes don't have to present data from just one field  49
  It isn't necessary for each field in a table to appear on the form  51
  Controlling data entry  51
  Use of forms can be controlled  51
  Forms can be web pages  51
  Summary  52

Chapter 6 • Reports  54

Chapter 7 • Summary of Part 1  56
Part 2 – A multi-table database  59

Chapter 8 • Introduction to Part 2  61

Chapter 9 • Serious problems with single tables  62
  Redundant data  63
  Typographical errors  63
  Modifying data  64
  Summary  65

Chapter 10 • Multiple tables cure serious problems  67
  Redundant data  69
  Typographical errors  72
  Modifying data  72

Chapter 11 • Making multiple tables work together  73
  Databases are designed to model the real world  74

Chapter 12 • Getting the data into the correct tables  75
  Not normalization (and not ER modeling either)  77
  Object identification  78

Chapter 13 • Relationships in the real world  81
  One-to-many  81
  One-to-one  82
  Many-to-many  82
  None  82
  Mapping real world relationships to tables  83

Chapter 14 • How are relationships modeled?  84
  Primary keys  86
  Foreign keys  91
  Summary so far  92
  Joins  93
General lessons about joins      106

Chapter 15 • Revisiting the big four – the synergy begins      112
  Closure      112
  Tables      115
  Queries (and a bit on forms)      116
  Forms      123
  Reports      124

Chapter 16 • Integrity      127
  Data integrity – is it worth the effort?      127
  Types of data integrity error (and some cures)      128
  Declarative and procedural referential integrity      134
  Nulls in foreign keys      139
  These options in context      142
  Other integrity issues      143
  Integrity – where should you set it?      143

Chapter 17 • Summary of Part 2      146

Part 3 – Database Design & Architecture      147

Chapter 18 • Database design      149
  Designing databases – user, logical and physical models      149
  The Logical model – overview      151
  More about the logical model      152
  CASE tools      154
  Summary so far      158
  The final big advantage of CASE tools      158
  More about the differences between the Logical and Physical models      160
  Reality check      162
  Normalization can help      162
Contents

Reverse engineering      163
Methodologies      164
Summary of design models      164

Chapter 19 ● The seven layers of wisdom      165
The seven layers of wisdom      165

Chapter 20 ● Database architecture 168
Default Architecture in Access 168
Access – PC front end – data on file server 168
Client-server (or two-tier) architecture 171
Three-tier architecture (also known as multi-tier) 173
Web-based applications 174
Choosing a database architecture 176
What comes next 177

Part 4 – Related database topics 179

Chapter 21 ● What exactly is a relational database? 181
Do multiple tables a relational database make? 181

Chapter 22 Triggers and stored procedures 183
Triggers 183
Stored procedures 187
Summary – triggers and stored procedures 189

Chapter 23 ● Transactions, logs, backup, locking and concurrency 190
Transactions 190
Logs 191
Locking 197
Concurrency 199
Row locking and page locking 199
### Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>WHERE</td>
<td>262</td>
</tr>
<tr>
<td>Conditions</td>
<td>263</td>
</tr>
<tr>
<td>ORDER BY</td>
<td>267</td>
</tr>
<tr>
<td>Wildcards</td>
<td>270</td>
</tr>
<tr>
<td>Sub-queries</td>
<td>271</td>
</tr>
<tr>
<td>Built-in functions</td>
<td>272</td>
</tr>
<tr>
<td>GROUP BY – collecting information</td>
<td>276</td>
</tr>
<tr>
<td>GROUP BY…HAVING – collecting specific information</td>
<td>282</td>
</tr>
<tr>
<td>Working with multiple tables</td>
<td>285</td>
</tr>
<tr>
<td>Inner (Natural) joins</td>
<td>290</td>
</tr>
<tr>
<td>Outer joins</td>
<td>291</td>
</tr>
<tr>
<td>UNION</td>
<td>293</td>
</tr>
<tr>
<td>SELECT summary</td>
<td>296</td>
</tr>
<tr>
<td>INSERT</td>
<td>297</td>
</tr>
<tr>
<td>UPDATE</td>
<td>300</td>
</tr>
<tr>
<td>DELETE</td>
<td>302</td>
</tr>
<tr>
<td>A question (and a free SQL diagnostic tool)</td>
<td>303</td>
</tr>
<tr>
<td>Summary</td>
<td>306</td>
</tr>
<tr>
<td>Chapter 30 ● Domains</td>
<td>307</td>
</tr>
<tr>
<td>Chapter 31 ● What does null mean?</td>
<td>309</td>
</tr>
<tr>
<td>Chapter 32 ● Primary keys</td>
<td>313</td>
</tr>
<tr>
<td>Candidate keys</td>
<td>315</td>
</tr>
<tr>
<td>Part 5 – Speeding up your database</td>
<td>317</td>
</tr>
<tr>
<td>Chapter 33 ● Hardware considerations</td>
<td>319</td>
</tr>
<tr>
<td>CPUs</td>
<td>320</td>
</tr>
<tr>
<td>Memory</td>
<td>320</td>
</tr>
<tr>
<td>Disks</td>
<td>322</td>
</tr>
</tbody>
</table>
Data volume vs. disk capacity 322
Don’t put all your eggs in one basket 323

Chapter 34 • Indexing 324
   Indexing techniques 324
   Applying indexes – which fields/columns should be indexed? 333
   Intelligent use of indexes 337

Chapter 35 • More on optimization 338
   Query optimization 338
   Update statistics 339
   Query analysis 340
   Writing good SQL code 342

Chapter 36 • Denormalization 344
   Mirroring tables 345
   Splitting tables 346
   Redundant data 348
   Repeating groups (breaking 1NF) 349
   Derived columns 351
   Summary 352

Appendix 1 • GUIs, macros and control languages 353
   Creating a very simple user interface 353
   Other languages – SQL 362

Index 365
Preface

Bill and I first wrote *Inside Relational Databases* to help people who were new to building databases. We could find lots of books that told people how to use their database engine of choice (Access, SQL Server, MySQL, whatever) but very few that described the underlying way in which relational databases work. So we wrote one. We guessed that many of the readers would be using Access, so we used Access to illustrate the relational model and called the book “Inside Relational Databases – with examples in Access”. However, we did try very hard to make it clear that the book wasn’t about how to drive Access; it was about the relational model that underpins all relational databases. We were simply using Access for illustrative purposes. To our enormous relief this was understood by those people kind enough to buy it and the book sold well.

For the second edition we expanded the book to include information about client-server databases and we illustrated that book with images from several database engines so we dropped ‘with examples in Access’ from the title.

That second edition was, happily, also well received, so eventually we turned our thoughts to another edition. It seemed to us very important that we continue to focus the book on the relational model and not turn it into a “Teach yourself about database X” book. On the other hand, we were nagged by the feeling that it is very convenient for the reader if we illustrate the relational model using their favorite product – such as Access. Eventually we realized that all we had to do was to produce several versions of the same book, each based on a different database engine.

Which is why you have in your hand a copy of “Inside Relational Databases – with examples in Access”. If this isn’t your database engine of choice, scan the internet and see if we have come up with one that matches your requirements. If not, email me Mark@Penguinsoff.co.uk. We’re not proud. We’ll do a version for any relational database engine if enough people want it.
Should we tell you the whole story?

Of course, there is an inevitable tension in trying to work like this. For example, in Chapter 16 we talk about referential integrity. There are essentially six different flavors of referential integrity but Access only supports four of them (they are the most important ones however, so you aren’t missing out on too much). The problem is this. Should we tell you about the other two? If we do, as an Access user you have every right to be annoyed that we are telling you about a feature you can’t use. On the other hand, the six different types that we describe are part of the relational world and this book is about that world – we are not trying to teach you how to use Access, we are simply using Access to illustrate the relational model. Ultimately we decided to risk your ire and to describe all of the features of the relational model as we see it, even if Access doesn’t support all of them. One advantage of this approach is that if you need to use a different database engine you will almost certainly find the extra information useful.

Incidentally, this is not meant to imply that Access is somehow lacking as a relational database engine. The reason we chose it for the first book is that it is such a good example of a relational database tool. We are putting exactly the same warning in all the versions of the book that we write – there are no engines that support all aspects of the relational model.

Other changes

We have also taken this opportunity to restructure the book significantly. I (Mark) continue to teach database design and practice, both to undergraduates and in the commercial world. Without doubt the most popular topic in the commercial world is how to make databases run faster (no great surprise there) so we have added an entire section of brand new material – more than 10% of the entire book – on that topic. The section on designing databases has been reorganized and expanded and we also re-read the entire book (several times) and brought it all up to date.
Chapter 1

Introduction

Chapter 1 of Sir Henry Birkin’s autobiography “Full Throttle”, published in 1932, ends with the following: “I can waste no more time on this matter; for the end is reached of what I now confess to have been 24 pages of deceit. I have disguised under the designation of Chapter One what was really nothing more than an introduction; but I know quite well, that had I been honest and called it an introduction, nobody would have read it.”

If Henry Birkin wasn’t my hero for racing two-ton motor cars on appalling road surfaces at speeds well in excess of 100 m.p.h., he would be my hero for sheer literary nerve.

I’ll be less disingenuous. This chapter is an introduction. It defines the very basic terms that you may need like ‘database’, ‘relational’, ‘DBMS’ and ‘RDBMS’. It tells you why the book was written, at whom it is aimed, and how it is organized.

If you know all of this already, or it sounds tedious and you really want to get down to the nitty gritty, please don’t bother reading this chapter; dive straight into the book at Chapter 2 or wherever you fancy. As far as we’re concerned, anyone who has paid us the considerable compliment of actually exchanging money for our words is entitled to read them as they please.

Who are we?

This book has two authors: Mark Whitehorn and Bill Marklyn.

Bill worked for Microsoft as the Development Manager for the first three versions of Access (1.0, 1.1 and 2.0). I (Mark) work as a database consultant, teach database theory and practice at two Universities and have written the UK Personal Computer World’s database column for more than twelve years.

We met (at a database conference, not unreasonably, given our interests) in the
summer before Access 1.0 was launched and found that we shared similar views on how databases should be designed and built. We wrote the original version of this book in 1997 and have worked together on book projects ever since.

I (Mark) penned most of the words and whenever the pronoun ‘I’ appears in the text, I accept full responsibility. So what is Bill doing there on the front cover? Well, writing down the words in a book is only one component. Books also need ideas and enthusiasm. I may have written the words, but Bill, more than anyone else, fired my interest in writing this book, provided his own inexhaustible enthusiasm and many of the ideas. I couldn’t have written it without him, and he would probably not have found the time to write it himself, so it is truly a joint venture.

What is a database?

A database is simply a collection of data. Nowadays the term tends to be used about computerized systems, but the old cards which were used to classify and locate books in a library are a good example of a non-computerized database. The difference computers have made to databases is deceptively simple: computers make access to the data faster. That means, on a trivial level, that instead of hunting through 50,000 pieces of cardboard in a dusty room for three days, a computer will do the same job in under a second. However, there is more to this speed than meets the eye.

Suppose I asked you to find me the names and addresses of all male hospital patients in the country who are over 60 years old, have a history of diabetes in the family and have two children. Given a local paper-based system, the question is unanswerable in any meaningful way. By the time you have traveled around the country and searched all the available records, most of said patients would be dead. With centralized, computerized patient records, this question should be answerable in minutes or at worst a few hours. So computerizing databases hasn’t simply speeded them up, it has opened up whole new ways of looking at data that simply weren’t possible before.

And databases are becoming ever more pervasive. If you book a seat on a plane or train then someone, somewhere is using a database. A bank account is nothing more than a complex database; credit card purchases, your appointment with the doctor – all are likely to be entered into a database.
Databases vs. Database Management Systems

One important distinction which should be made early on in this book is the difference between a database and the software which is used to control and manipulate that database. A database is a collection of data – perhaps a list of your customers, their addresses, fax numbers and so on. In order to keep the data in your database under control, you need software known as a DBMS (Database Management System). The DBMS is to a database what a word processor is to a letter. The former is the controlling software, the latter the data that it manipulates. Examples of DBMSs include Access, SQL Server, MySQL, Oracle, DB2 – the list is not endless but certainly long. DBMSs are also referred to as database engines.

Relational Database Management Systems

There are several fundamentally different ways in which data can be handled or modeled – Hierarchical, Network and Relational are three such models. Without doubt the most widely used is the relational model, the brainchild of Dr Edgar Codd.

Codd is often described as ‘the Father of the Relational Database’ (with obligatory upper-case letters) and this is perfectly fair since he came up with the original idea which he announced to the world in 1970 with a paper entitled "A Relational Model of Data for Large Shared Data Banks". This is often quoted as Codd’s first paper on the relational model but in fact he published one in 1969 called “Derivability, Redundancy, and Consistency of Relations Stored in Large Data Banks”. However this paper was an IBM research report and carried a limited distribution notice, so it wasn’t seen by many people at the time.

Codd wrote with enormous precision, for example:

Note that a view is theoretically updatable if there exists a time-independent algorithm for unambiguously determining a single series of changes to the base relations that will have as their effect precisely the requested changes in the view.

Since he is the original source, the very bedrock, from which this material comes, it was vital that he wrote in such a way as to leave no doubt whatsoever as to the meaning. Indeed, his ability to communicate with such precision to both mathematicians and database people undoubtedly helped to promote the relational model. However, a side effect is that his material can be a little difficult to follow on first reading.

Chris Date was at one time a co-worker of Codd’s and has, in my opinion,
done a wonderful job of explaining, popularizing and generally advancing the use and understanding of the relational model. These two guys rank alongside Henry Birkin in my hall of heroes.

So, why do you need to know anything about Codd and Date? Well, the world of databases is crowded with people who will try to impress you with their knowledge (which makes it exactly the same as every other branch of life). If you don’t know these two names you can be seriously out-bluffed. For example, in 1985 Codd published a set of rules which defined the concept of a relational DBMS at that time. These rules have subsequently become enshrined in database lore and are constantly referred to in conversation – for example, “You can’t do that, it contravenes Codd’s 7th rule!” Experience suggests that very few people have actually gone back to the source material (Computerworld, 14-21 October 1985) and read them, so simply knowing who Ted Codd was and that he had a set of rules somewhere should be enough information to allow you to argue on even terms. However, if you want to know more about these rules, read Chapter 24 which lists and defines them all in (hopefully) understandable terms.

As an aside, there is nothing to stop you from playing the same game. If you find that you are being out-bluffed, you can try “Well, of course, when Chris contacted me about this last week he said that ... [fill in appropriate supporting statement].”

Very sadly, you can no longer substitute ‘Ted’ for ‘Chris’ because Edgar Codd died on the 18th. April 2003. Out of all the tributes paid to him, for me the most touching came from Sharon Weinberg, initially a colleague at IBM, later his second wife. “For a while, we had work stations side by side. I’d see him staring at his screen, thinking. I’d worry, and say, ‘Breathe, Ted, breathe!’ He’d work like a demon, you could not break his focus.”

Any DBMSs you use are likely to be based on the relational model. Such DBMSs are, perfectly sensibly, known as RDBMSs (Relational Database Management Systems).

**Why this book?**

One of the main aims of this book is to demystify the relational database model. It is an excellent way of handling information and we want to make the relational model accessible to more people. This book attempts to introduce the concepts and ideas behind the relational model without the usual jargon.

A fair question at this point is “Why should anyone want to know about the
relational model?”. The answer to that lies in the distinction between knowing how to use a software package to perform a specific task and knowing why you would want to perform that task in the first place.

For example, if you browse through the Access help system, you will be able to find out how to, for example, declare a primary key for a table.

- Select the table.
- Open it in Design view.
- Select the field or fields you want to define as the primary key.
  - To select one field, click the row selector for the desired field.
  - To select multiple fields, hold down the CTRL key and then click the row selector for each field.
- Click Primary Key on the toolbar.

Great. Excellent, so now you know how to do it. But … what is a primary key? Are they really important? Should every table have one? What are they used for? The help system may give some pointers, but it is essentially focused on telling you how to achieve a desired end result. We have a totally different focus (although we may illustrate processes from time to time): we tell you why you would want to do it in the first place. This book is about the relational model because understanding the model is essential if you want to create excellent, stable and robust databases.

Who should read this book?

You should read this book if:

- You have created databases but they don’t seem to work very well. Perhaps you:
  - can’t retrieve the information that you want.
  - have to type in the same information over and over again.
  - type in data and it appears to go missing.
  - ask questions and get answers that you know are wrong.
  - can use Access but you don’t know exactly what to do with it.
  - know that a relational database lets you create multiple tables in the database but you are uncertain why this is to your advantage.
  - find that there are lots of features in a database that sound interesting but you have no idea what you are supposed to do with them.
Or perhaps you hear words in connection with databases like:
- normalization
- functional dependency
- inner join
- union
- redundant data
- data dictionary
- meta-data
- ER modeling
- transaction
- concurrency
- locking

and you haven’t got the faintest idea what they mean and there is no one you can ask.

As we said above, you shouldn’t read this book if you are looking for a ‘How to use Access’ book.

If you are looking for such a book we feel honor bound to recommend “Accessible Access 2003” published by Springer-Verlag, for the simple reason that we wrote it …

Organization of the book

Within a database itself, the data is stored in tables. A simple database will contain a single table; more complex ones can contain many tables. For example, if you want to keep your address book in a database (as I do on a hand-held computer) then a single table is perfect. However, if you want to store all of the business transactions of a company, you will find that it is more efficient to store the data in multiple tables – one for the customers, another for the goods you sell and so on.

This book is divided into five parts.

Part 1 concentrates on databases which contain only single tables because this should make some of the basic principles easier to understand. It describes the components which make up a typical database:
- tables
- queries/views
- user-interface components (forms)
- reports
Part 2 explains that when most people start building more complex databases they tend to try and put all of the data into one table. The section illustrates why this is a bad idea and then outlines how you can use multiple tables to overcome the problems inherent in single-table databases. Part 2 is a seriously chunky section since much of the relational model is explained in there.

Part 3 discusses, in broad terms, where the components of your database can be located. Databases (even when you are using a client-server database engine like SQL Server) can be created on single, stand-alone machines, in which case only one person can use the database at any one time. If you want the database to have multiple, concurrent users (clearly a major advantage if the database is used within a company), parts or all of it can be moved onto a networked machine. However, allowing multiple people to manipulate the same data at the same time introduces a whole host of new topics that you need to understand if you are to build and manage effective databases.

Part 4 is very different from the other parts. The first three are lovingly hand-crafted so that each chapter builds on the information in the previous one (or, at least, that was the idea). The chapters in Part 4 can be read in any order because the information in each one is essentially unconnected to that in the others. Much of the information in any given chapter in Part 4 assumes that you do understand the information in Parts 1-3 but there is essentially no cross_requirement between chapters in Part 4. Thus, if you understand the information in Parts 1-3 and want to know about SQL, go straight to Chapter 29 and read it. If someone is bugging you with terms like ‘third normal form’ or ‘functional dependency’ then read the chapter on normalization (Chapter 25).

Part 5 is about speeding up your database and a range of techniques is discussed there.

Some ground rules

This book assumes that the reader will be interested in relational databases. There are several other models for organizing data but the relational is the most common. Thus, in the text which follows, in order to avoid continually having to prefix the word ‘database’ with the word ‘relational’, you can assume that I mean relational unless it is explicitly stated otherwise.

I am all too aware of the fact (having been corrected many times) that ‘datum’ is the singular form of ‘data’, and so I should write ‘every datum’ rather than ‘every piece of data’. The trouble is that using ‘data’ as both the singular and plural forms is now so widespread that to do otherwise smacks of pedantry
and obscures rather than clarifies. I’ve been swayed by the common usage argument and have used just ‘data’.

In fact, now is a good time to point out that this entire book fails to use exact terminology. I know that a table isn’t a table, it is a ‘relation’. Or to be precise, a table isn’t exactly a relation, it’s... and so on. I happily acknowledge that I shouldn’t talk about the number of rows (or, slightly more accurately, the number of ‘tuples’); what I really mean is the cardinality. Relational databases have their origins in the precise world of mathematics and that particular world has a very precise language. The trouble is that I also live and work in the real world and in it real people talk about tables, rows and columns. So, at the expense of a small degree of accuracy and the gain (I hope) of a great deal of clarity, I have elected to use less formal terminology whenever possible. Apologies are proffered in advance to those who are terminally offended.

**Downloading files from the website**

Where appropriate, the databases etc. which are used to illustrate this book are available for download from www.penguinsoft.co.uk.

The files are in Access 2000 format (which is compatible with Access 2000 and Access 2003): simply look in the folder with the appropriate name. Access 2003 was used for the screen shots.

Each Access file is tied to its chapter by name so, for example, the Access file associated with Chapter 2 is called CHAP2.MDB. Occasionally a chapter warranted more than one database file, so sometimes you will find names like CHAP25A.MDB and CHAP25B.MDB.

**Acknowledgements**

Very grateful thanks go to Mary Whitehorn, a talented writer in her own right and not unrelated to one of the authors. She put in so much work, both proof-reading and actually writing sections, that she easily qualified as another author. Only her innate modesty prevents her name appearing on the cover.

I (Mark) also acknowledge a huge debt to Professor John Parker. He was my PhD supervisor many years ago and is currently the Director of the Cambridge Botanic Garden. Whatever communication skills I have acquired came directly from sitting at the feet of a master of the art.
We don’t have problems...

If you find a bug (sorry, bookware anomaly) anywhere in this book, I would be delighted if you would tell me by visiting www.penguinsoft.co.uk, where all known problems (and fixes) will also be posted.

Outroduction

So, that’s the end of the introduction. I’d love to know how many people actually read it; perhaps Sir Henry is right. Either way, I hope that you get as much pleasure from the elegance of the relational model as I have done and I will be delighted if this book illuminates even a small section of it for you.
Part 1

A simple, single-table database
Chapter 2

Introduction to Part 1

The first database you ever build is likely to do something relatively simple, such as keeping a list of your customers or friends. Happily, even a simple database like this can be used to introduce the four most important components of a database, namely:

- Tables
- Queries/Views
- Forms
- Reports

It is difficult to over-stress the importance of these four components and we will start with a quick look at each.

Tables

Tables are the basic structures in which data is stored within a database. Think of a table as the container in which the data sits and the other three components as devices which manipulate the data contained in the table. A table that contains information about your employees might look like the one shown below. The name of the table (EMPLOYEES) is shown in upper-case.

<table>
<thead>
<tr>
<th>EmployeeNo</th>
<th>FirstName</th>
<th>LastName</th>
<th>DateOfBirth</th>
<th>DateEmployed</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Manny</td>
<td>Tomanny</td>
<td>12 Apr 1966</td>
<td>01 May 1999</td>
</tr>
<tr>
<td>2</td>
<td>Rosanne</td>
<td>Kolumn</td>
<td>21 Mar 1977</td>
<td>01 Jan 2000</td>
</tr>
<tr>
<td>3</td>
<td>Cas</td>
<td>Kade</td>
<td>01 May 1977</td>
<td>01 Apr 2002</td>
</tr>
<tr>
<td>4</td>
<td>Norma</td>
<td>Lyzation</td>
<td>03 Apr 1966</td>
<td>01 Apr 2002</td>
</tr>
<tr>
<td>5</td>
<td>Juan</td>
<td>Tomani</td>
<td>12 Apr 1966</td>
<td>01 Apr 2002</td>
</tr>
</tbody>
</table>
If you do download the sample files (see Chapter 1) you’ll find this EMPLOYEES table in CHAP2.MDB.

Queries/Views

Queries are questions that you can ask of the data in a table. If you wanted to find all of your employees who were born after, say, 1970, you would use a query. Queries are used frequently within databases because typically the tables hold very large amounts of data and we often want to deal with, or look at, just a subset of that data.

Below is the result of a query to find the employees who were born after 1970.

<table>
<thead>
<tr>
<th>EmployeeNo</th>
<th>FirstName</th>
<th>LastName</th>
<th>DateOfBirth</th>
<th>DateEmployed</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Rosanne</td>
<td>Kolumn</td>
<td>21 Mar 1977</td>
<td>01 Jan 2000</td>
</tr>
<tr>
<td>3</td>
<td>Cas</td>
<td>Kade</td>
<td>01 May 1977</td>
<td>01 Apr 2002</td>
</tr>
</tbody>
</table>

The results of queries that we use to illustrate this book are usually shown with names; for example, in this case, it is EmployeesBornAfter1970, which is simply the name of the query. The query can be found in the appropriate .MDB file, in this case called CHAP2.MDB.

Views are very much like queries. Both are devices that are used to extract information from the database. Essentially the difference between a view and a query is simply its location. In a client-server database (such as SQL Server) any query that is stored on the workstation is a query. The same query, stored on the server (where it is accessible to lots of people) is typically called a view.

Since Access sits on your PC it becomes very difficult to say whether the query is on a server or a workstation, so Access simply uses the term Query and doesn’t bother with the term View.

In large measure you can take every sentence written about queries, substitute the word “View” for the word “Query” and it still makes perfect sense.

Forms

Users need to gain access to the data in a database, so the database needs some kind of user interface. Generally a complete user interface will consist of a number of different components called forms. A form is a device which
allows you to look at and edit the data in a table, like the one below showing information from a table of data about customers.

Some databases, such as Access, provide inbuilt tools for creating forms. Others, like Oracle, provide a separate tool (Oracle Forms which is a component of the Oracle Developer Suite) to build them. Forms can be produced as part of applications or as web pages (web forms). We’ll drill into this later but for now you can think of forms as the user-friendly front end of the database.

This form allows you to look at and edit the data in the table. In fact, you can usually go directly to a table itself and perform both of these actions but typically they are accomplished via a form. A good question at this point is “Why use a form?” and a simple (but true) answer is that forms can be made more attractive and easier to use than tables. For more compelling reasons to use forms, see Chapter 5 – Forms.

**Reports**

Reports are used to produce printed output from the table. If you want a list of all of your customers’ names and addresses, you would use a report to roll out just such a list from the printer.
This report produces a printed list of information about employees, sorted by last name.

Simple, isn’t it? And if you can get to grips with these four fundamental components – tables, queries, forms and reports – you will have acquired a very good handle on databases in general. Of course, there is more to them than just their definitions, so in the next four chapters we’ll have a look at each in greater detail.
Tables are containers for holding data which is similar in structure. If you collected information about your employees, the data about each would be similar and would therefore make the contents of a perfectly satisfactory table.

<table>
<thead>
<tr>
<th>EmployeeNo</th>
<th>FirstName</th>
<th>LastName</th>
<th>DateOfBirth</th>
<th>DateEmployed</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Manny</td>
<td>Tomanny</td>
<td>12 Apr 1966</td>
<td>01 May 2004</td>
</tr>
<tr>
<td>2</td>
<td>Rosanne</td>
<td>Kolumn</td>
<td>21 Mar 1977</td>
<td>01 Jan 2003</td>
</tr>
<tr>
<td>3</td>
<td>Cas</td>
<td>Kade</td>
<td>01 May 1977</td>
<td>01 Apr 2004</td>
</tr>
<tr>
<td>4</td>
<td>Norma</td>
<td>Lyzation</td>
<td>03 Apr 1966</td>
<td>01 Apr 2004</td>
</tr>
<tr>
<td>5</td>
<td>Juan</td>
<td>Tomani</td>
<td>12 Apr 1966</td>
<td>01 Apr 2004</td>
</tr>
</tbody>
</table>

This part is all about databases made up of a single table, so it may come as a bit of a shock to find that the CHAP3.MDB (the Access file which holds the sample files used for this chapter) has more than one table. However, this is simply because we use different tables to illustrate different points.

On the other hand, you cannot put one or more carefully spotted train numbers and a list of your favorite books into the same table.
To do so would entirely miss the point that tables should contain data with similar structure.

**Rows & columns – records & fields**

Tables consist of rows (horizontal) and columns (vertical). In the sensible sample table (shown again below) each row contains the data about one employee. The table has five columns, each of which has a name.

<table>
<thead>
<tr>
<th>EMPLOYEES</th>
<th>EmployeeNo</th>
<th>FirstName</th>
<th>LastName</th>
<th>DateOfBirth</th>
<th>DateEmployed</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>Manny</td>
<td>Tomanny</td>
<td>12 Apr 1966</td>
<td>01 May 1999</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Rosanne</td>
<td>Kolumn</td>
<td>21 Mar 1977</td>
<td>01 Jan 2000</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>Cas</td>
<td>Kade</td>
<td>01 May 1977</td>
<td>01 Apr 2002</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>Norma</td>
<td>Lyzation</td>
<td>03 Apr 1966</td>
<td>01 Apr 2002</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>Juan</td>
<td>Tomani</td>
<td>12 Apr 1966</td>
<td>01 Apr 2002</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>2312234</td>
<td>Steam Train</td>
<td>Red and Black</td>
<td>3.45 to Bedford</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>The Egg-Shaped Thing</td>
<td>10c</td>
<td>Christopher Hodder-Williams</td>
<td>Hard Back</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>34223</td>
<td>Diesel</td>
<td>Black and Soot</td>
<td>2.17 to Seattle</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>The Mullenthorpe Thing</td>
<td>$2.50</td>
<td>Christopher Hood</td>
<td>Hard Back</td>
</tr>
</tbody>
</table>

Rows are also called records and columns are also called fields. In many cases the terms are used interchangeably. If I was being pedantic I’d say that the difference is that record and field are often used about the data:

“Well, I’m looking at Fred’s record on the screen now and he doesn’t have an entry in the DOB field.”

whereas row and column are often used about the table:

“That table has 50,000,000 rows and 120 columns – it’s a monster!”
However, it isn’t quite as simple as that.

In some cases it isn’t clear whether we are really talking about the table or the data. So a sentence like “If we run the query against that table we should get about 50 records.” sounds just as appropriate as “If we run the query against that table we should get about 50 rows.”

In addition the usage (in my experience) varies between communities of database engine users. For example, Access users tend to favor record and field whereas Oracle users favor row and column. In addition, row and column are usually favored by people from either camp when they are discussing the more formal aspects of databases (normalization, denormalization etc.).

And, just in case you still think this is simple, as discussed earlier there is a third set of terms that are used, for example, when Codd and Date write about the relational model.

<table>
<thead>
<tr>
<th>Data terms</th>
<th>Table structure terms</th>
<th>Very formal terms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table</td>
<td>Table</td>
<td>Relation</td>
</tr>
<tr>
<td>Record</td>
<td>Row</td>
<td>Tuple</td>
</tr>
<tr>
<td>Field</td>
<td>Column</td>
<td>Attribute</td>
</tr>
<tr>
<td>Number of records</td>
<td>Number of rows</td>
<td>Cardinality</td>
</tr>
<tr>
<td>Number of fields</td>
<td>Number of columns</td>
<td>Degree or Arity</td>
</tr>
</tbody>
</table>

In my opinion, people like Chris Date use the very formal terms for reasons of precision – a tuple isn’t exactly the same as a row, neither is a relation exactly the same as a table, although they are very nearly the same. Other people seem to use them purely for reasons of obfuscation. (Bill added “Or elitism, a very common malady”.)

You can see our problem. Which of these terms should we use in this book? Standardising on one set would provide a great deal of consistency. On the other hand, that doesn’t help you gain a feel for the way the terms are used in practice. In the end we decided simply to use whichever term seemed appropriate to us for the context in which we were writing. We think this is ultimately better but it does, absolutely 100%, guarantee that we will have been inconsistent somewhere.

**Numbers of rows and columns**

In most databases, each table can be considered to be infinitely expandable in terms of the number of rows it can contain and so no limit is set on the num-