Introduction

It has been estimated that more than 80 percent of all computer programming is database-related. This is certainly easy to believe. After all, a database can be a powerful tool for doing exactly what computer programs do best: store, manipulate, and display data.

Even many programs that seem at first glance to have little to do with traditional business-oriented data use databases to make processing easier. In fact, looking back on more than 20 years of software development experience, I'm hard pressed to think of a single non-trivial application that I've worked on that didn't use some kind of database.

Not only do databases play a role in many applications, but they also often play a critical role. If the data is not properly stored, it may become corrupted and the program will be unable to use it meaningfully. If the data is not properly organized, the program may be unable to find what it needs in a reasonable amount of time.

Unless the database stores its data safely and effectively, the application will be useless no matter how well-designed the rest of the system may be. The database is like the foundation of a building: without a strong foundation, even the best crafted building will fail, sometimes spectacularly (the Leaning Tower of Pisa notwithstanding).

With such a large majority of applications relying so heavily on databases, you would expect everyone involved with application development to have a solid, formal foundation in database design and construction. Everyone including database designers, application architects, programmers, database administrators, and project managers should ideally understand what makes a good database design. Even an application's key customers and users could benefit from understanding how databases work.

Sadly that is usually not the case. Many IT professionals have learned what they know about databases through rumor, trial-and-error, and painful experience. Over the years, some develop an intuitive feel for what makes a good database design but they may still not understand the reasons why a design is good or bad, and they may leave behind a trail of rickety, poorly constructed programs built on shaky database foundations.

This book provides the tools you need to design a database. It explains how to determine what should go in a database and how a database should be organized to ensure data integrity and a reasonable level of performance. It explains techniques for designing a database that is strong enough to store data safely and consistently, flexible enough to allow the application to retrieve the data it needs quickly and reliably, and adaptable enough to accommodate a realistic amount of change.

With the ideas and techniques described in this book, you will be able to build a strong foundation for database applications.

Who This Book Is For

This book is intended for IT professionals and students who want to learn how to design, analyze, and understand databases. The material will benefit those who want a better high-level understanding of databases such as proposal managers, architects, project managers, and even customers. The material will also benefit those who will actually design, build, and work with databases such as database designers, database administrators, and programmers. In many projects, these roles overlap so the same person may be responsible for working on the proposal, managing part of the project, and designing and creating the database.

This book is aimed at IT professionals and students of all experience levels. It does not assume that you have any previous experience with databases or programs that use them. It doesn't even assume that you have experience with computers. All you really need is a willingness and desire to learn.

What This Book Covers

This book explains database design. It tells how to plan a database's structure so the database will be robust, resistant to errors, and flexible enough to accommodate a reasonable amount of future change. It explains how to discover database requirements, build data models to study data needs, and refine those models to improve the database's effectiveness.

The book solidifies these concepts by working through a detailed example that designs a realistic database. Later chapters explain how to actually build databases using two common database products: Access 2007 and MySQL.

The book finishes by describing some of the topics you need to understand to keep a database running effectively such as database maintenance and security.

What You Need to Use This Book

This book explains database design. It tells how to determine what should go in a database and how the database should be structured to give the best results.

This book does not focus on actually *creating* the database. The details of database construction are different for different database tools so, to remain as generally useful as possible, this book doesn't concentrate on any particular database system. You can apply the techniques described here equally to whatever database tool you use, whether it's Access, SQL Server, Oracle, MySQL, or some other database product.

NOTE

Most database products include free editions that you can use for smaller projects. For example, SQL Server Express Edition, Oracle Express Edition, and MySQL Community Server are all free.

To remain database neutral, the book does not assume you are using a particular database so you don't need any particular software or hardware. To work through the Exercises, all you really need is a pencil and some paper. You are welcome to type solutions into your computer if you like but you may actually find working with pencil and paper easier than using a graphical design tool to draw pictures, at least until you are comfortable with database design and are ready to pick a computerized design tool.

Chapter 15, "Microsoft Access," explains how to build databases using the Microsoft Access 2007 database product. If you want to follow along with the examples in that chapter and work through the Exercises, you need to have Microsoft Access 2007 installed (although other versions of Access will also work with a few differences). You can use any operating system that will run Microsoft Access 2007.

Similarly **Chapter 16**, "MySQL," explains how to build databases using the MySQL Community Server database product. If you want to follow this chapter's examples and work through them, you will need to install MySQL Community Server. You can use any operating system that will run MySQL.

To experiment with the SQL database language described in **Chapter 17**, "Introduction to SQL," and **Chapter 18**, "Building Databases with SQL Scripts," you need any database product that supports SQL (that includes pretty much all relational databases) running on any operating system.

How This Book Is Structured

The chapters in this book are divided into five parts plus appendixes. The chapters in each part are described here. If you have previous experience with databases, you can use these descriptions to decide which chapters to skim and which to read in detail.

Part I: Introduction to Databases and Database Design

The chapters in this part of the book provide background that is necessary to understand the chapters that follow. You can skim some of this material if it is familiar to you but don't take it too lightly. If you understand the fundamental concepts underlying database design, it will be easier to understand the point behind important design concepts presented later.

Chapter 1, "Goals of Effective Database Design," explains the reasons why people and organizations use databases. It explains a database's purpose and conditions that it must satisfy to be useful. This chapter also describes the basic ACID (Atomicity, Consistency, Isolation, Durability) and CRUD (Create, Read, Update, Delete) features that any good database should have. It explains in high-level general terms what makes a good database and what makes a bad database.

Chapter 2, "Database Types," explains some of the different types of databases that you might decide to use. These include flat files, spreadsheets, hierarchical databases (XML), object databases, and relational databases. The relational database is one of the most powerful and most commonly used forms of database so it is the focus of this book, but it is important to realize that there are alternatives that may be more appropriate under certain circumstances. This chapter gives some tips on deciding which kind of database might be best for a particular project.

Chapter 3, "Relational Database Fundamentals," explains basic relational database concepts such as tables, rows, and columns. It explains the common usage of relational database terms in addition to the more technical terms that are sometimes used by database theorists. It describes different kinds of constraints that databases use to guarantee that the data is stored safely and consistently.

Part II: Database Design Process and Techniques

The chapters in this part of the book discuss the main pieces of database design. They explain how to understand what should be in the database, develop an initial design, separate important pieces of the database to improve flexibility, and refine and tune the design to provide the most stable and useful design possible.

Chapter 4, "Understanding User Needs," explains how to learn about the users' needs and gather user requirements. It tells how to study the users' current operations, existing databases (if any), and desired improvements. It describes common questions that you can ask to learn about users' operations, desires, and needs, and how to build the results into requirements documents and specifications. This chapter explains what use cases are and tells how to use them and the requirements to guide database design and to measure success.

Chapter 5, "Translating User Needs into Data Models," introduces data modeling. It explains how to translate the user's conceptual model and the requirements into other more precise models that define the database design rigorously. This chapter describes several database modeling techniques including user-interface models, semantic object models, entity-relationship diagrams, and relational models.

Chapter 6, "Extracting Business Rules," explains how a database can handle business rules. It explains what business rules are, how they differ from database structure requirements, and how you can identify business rules. This chapter explains the benefits of separating business rules from the database structure and tells how to achieve that separation.

Chapter 7, "Normalizing Data," explains one of the biggest tools in database design: normalization. Normalization techniques allow you to restructure a database to increase its flexibility and make it more robust. This chapter explains the various forms of normalization, emphasizing the stages that are most common and important: first, second, and third normal forms (1NF, 2NF, and 3NF). It explains how each of these kinds of normalization helps prevent errors and tells why it is sometimes better to leave a database slightly less normalized to improve performance.

Chapter 8, "Designing Databases to Support Software Applications," explains how databases fit into the larger context of application design and lifecycle. This chapter explains how later development depends on the underlying database design. It discusses multi-tier architectures that can help decouple the application and database design so there can be at least some changes to either without requiring changes to the other.

Chapter 9, "Common Design Patterns," explains some common patterns that are useful in many applications. Some of these techniques include implementing various kinds of relationships among objects, storing hierarchical and network data, recording temporal data, and logging and locking.

Chapter 10, "Common Design Pitfalls," explains some common design mistakes that occur in database development. It describes problems that can arise from insufficient planning, incorrect normalization, and obsession with ID fields and performance.

Part III: A Detailed Case Study

If you follow all of the examples and exercises in the earlier chapters, by this point you will have seen all of the major steps for producing a good database design. However, it's often useful to see all of the steps in a complicated process put together in a continuous sequence. The chapters in this part of the book walk through a detailed case study following all of the phases of database design for the fictitious Pampered Pet database.

Chapter 11, "User Needs and Requirements," walks through the steps required to analyze the users' problem, define requirements, and create use cases. It describes interviews with fictitious customers that are used to identify the application's needs and translate them into database requirements.

Chapter 12, "Building a Data Model," translates the requirements gathered in the previous chapter into a series of data models that precisely define the database's structure. This chapter builds user-interface models, entity-relationship diagrams, semantic object models, and relational models to refine the database's initial design. The final relational models match the structure of a relational database fairly closely so they are easy to implement.

Chapter 13, "Extracting Business Rules," identifies the business rules embedded in the relational model constructed in the previous chapter. It shows how to extract those rules in order to separate them logically from the database's structure. This makes the database more robust in the face of future changes to the business rules.

Chapter 14, "Normalization and Refinement," refines the relational model developed in the previous chapter by normalizing it. It walks through several versions of the database that are in different normal forms. It then selects the degree of normalization that provides a reasonable tradeoff between robust design and acceptable performance.

Part IV: Implementing Databases (with examples in Access and MySQL)

Though this book focuses on abstract database concepts that do not depend on a particular database product, it's also worth spending at least some time on more concrete implementation issues. The chapters in this part of the book describe some of those issues and explain how to build databases with two different database products: Access 2007 and MySQL.

Chapter 15, "Microsoft Access," explains how to build a database with Microsoft Access 2007. This chapter doesn't cover everything there is to know about Access, it just explains enough to get started and to use Access to build non-trivial databases. You can use other versions of Access to work through this chapter, although the locations of menus, buttons, and other Access features are different in different versions.

Chapter 16, "MySQL," explains how to build a database with MySQL. This chapter tells where to download a free version of MySQL. It explains how to use the MySQL Command Line Client as well as some useful graphical tools including MySQL Query Browser and MySQL Workbench.

Part V: Advanced Topics

Although this book does not assume you have previous database experience, that doesn't mean it cannot cover some more advanced subjects. The chapters in this part of the book explain some more sophisticated topics that are important but not central to database design.

Chapter 17, "Introduction to SQL," provides an introduction to SQL (Structured Query Language). It explains how to use SQL commands to add, insert, update, and delete data. By using SQL, you can help insulate a program from the idiosyncrasies of the particular database product that it uses to store data.

Chapter 18, "Building Databases with SQL Scripts," explains how to use SQL scripts to build a database. It explains the advantages of this technique, such as the ability to create scripts to initialize a database before performing tests. It also explains some of the restrictions on this method, such as the fact that the user must create and delete tables in specific orders to satisfy table relationships.

Chapter 19, "Database Maintenance," describes some of the database maintenance issues that are part of any database application. Though performing and restoring backups, compressing tables, rebuilding indexes, and populating data warehouses are strictly not database design tasks, they are essential to any working application.

Chapter 20, "Database Security," explains database security issues. It explains the kinds of security that some database products provide. It also explains some additional techniques that can enhance database security such as using database views to appropriately restrict the users' access to data.

Appendixes

The book's appendixes provide additional reference material to supplement the earlier chapters.

Appendix A, "Exercise Solutions," gives solutions to Exercises so you can check your progress as you work through the book.

Appendix B, "Sample Database Designs," includes the designs for a variety of common database situations. These designs store information about such topics as books, movies, documents, customer orders, employee timekeeping, rentals, students, teams, and vehicle fleets.

The Glossary provides definitions for useful database and software development terms. The Glossary includes terms defined and used in this book in addition to other useful terms that you may encounter while reading other database material. This appendix can be a useful reference when you encounter an unfamiliar term on the Web or in database articles.

How to Use This Book

Because this book is aimed at readers of all experience levels, you may find some of the material familiar if you have previous experience with databases. In that case, you may want to skim chapters covering material that you already thoroughly understand.

If you are familiar with relational databases, you may want to skim **Chapter 1**, "Goals of Effective Database Design," **Chapter 2**, "Database Types," and **Chapter 3**, "Relational Database Fundamentals."

If you have previously helped write project proposals, you may understand some of the questions you need to ask users to properly understand their needs. In that case, you may want to skim **Chapter 4**, "Understanding User Needs."

If you have built databases before, you may understand at least some of the data normalization concepts explained in **Chapter 7**, "Normalizing Data." This is a complex topic, however, so I would recommend that you not skip this chapter unless you have a really thorough understanding of data normalization.

If you have extensive experience with using the SQL database language, you may want to skim **Chapter 17**, "Introduction to SQL." (Many developers who have used but not designed databases fall into this category.)

In any case, I strongly recommend that you at least skim the material in every chapter to see if there are any new concepts you can pick up along the way. Look at the Exercises at the end of a chapter before you decide that you can safely skip that chapter. If you don't know how to outline the solutions to the Exercises, you should consider looking at the chapter more closely.

Different people learn best in different ways. Some learn best by listening to lecturers, others by reading, and others by doing. Everyone learns better by combining learning styles. You will get the most from this book if you read the material and then work through the Exercises. It's easy to think to yourself, "Yeah, that makes sense" and believe you understand the material but working through several of the Exercises will help solidify the material in your mind. It may also help you see new ways that you can apply the concepts covered in the chapter.

NOTE

Normally, when I read a new technical book, I work through every example myself, modifying the problems to see what happens if I try different things not covered by the author. I work through as many questions and exercises as I can until I reach the point where more examples don't teach me anything new. Then I move on. It's one thing to read about a concept in the chapter; it's another to try to apply it to data that is meaningful to you.

After you have mastered the ideas in the book, you can use it for a reference. When you are starting a new project, you may want to refer to **Chapter 4**, "Understanding User Needs," to refresh your memory about the kinds of questions you should ask users to really discover their true needs.

Visit the book's Web site to download supplementary material such as checklists of questions to ask users and quick summaries of key techniques. This material is included in the book but it is also available for easy download on the book's Web site.

Also visit the book's Web site to look for updates and addendums. If readers find typographical errors or places where a little additional explanation may help, I'll post updates on the Web site.

Finally, if you get stuck on a really tricky concept and need a little help, email me at **RodStephens@vb-helper.com** and I'll try to help you out.

Note to Instructors

Database programming is boring. Not for you and me who have discovered the ecstatic joy of database design, the thrill of normalization, and the slightly risqué elation brought by slightly de-normalizing a database to achieve optimum performance. But let's face it, to a beginner database design and development can be a bit dull.

There's little you can do about the basic concepts but you can do practically anything with the data. At some point it's useful to explain how to design a simple inventory system but that doesn't mean you can't use other examples designed to catch students' attention. Data that relates to the students' personal experiences or that is just plain outrageous keeps them awake and alert (and most of us know that it's easier to teach students who are awake).

The examples in this book are intended to demonstrate the topic at hand but not all of them are strictly business-oriented. I've tried to make them cover a wide variety of topics from serious to silly. To keep your students interested and alert, you should add new examples from your personal experiences and from your students' interests.

I've had great success in my classroom using examples that involve sports teams (particularly local rivalries), music (combining classics such as Bach, Beethoven, and Tone-Loc), the students in the class (but be sure not to put anyone on the spot), television shows and stars, comedians, and political candidates. (Be careful with politics, though, because some people can become really emotionally attached to a particular candidate, no matter how stupid that candidate is. I focus on things they do that are so stupid that even loyal followers have to admit, "Yeah, that was a mistake." Fortunately politicians make those kinds of mistakes daily so there's plenty to work with. Watch the evening comedians for material.)

For exercises, encourage students to design databases that they will find personally useful. I've had students build databases that track statistics for the players on their favorite football teams, inventory their DVD or CD collections, file and search recipe collections, store data on "Magic: The Gathering" trading cards, track role-playing game characters, record information about classic cars, and schedule athletic tournaments. (Although the tournament scheduler didn't work out too well —the scheduling algorithms were too tricky.) One student even built a small but complete inventory application for his mother's business that she actually found useful. I think he was as surprised as anyone to discover he'd learned something useful.

When students find an assignment interesting and relevant, they become emotionally invested and will apply the same level of concentration and intensity to building a database that they normally reserve for console gaming, *South Park*, and "World of Warcraft." They may spend hours crafting a database to track WoW alliances just to fulfill a five-minute assignment. They may not catch every nuance of domain/key normal form but they'll probably learn how to build a functional database.

Note to Students

If you're a student and you peeked at the previous section, "**Note to Instructors**," shame on you! If you didn't peek, do so now.

Building a useful database can be a lot of work but there's no reason it can't be interesting and useful to you when you're finished. Early in your reading, pick some sort of database that you would find useful (see the previous section for a few ideas) and think about it as you read through the text. When the book talks about creating an initial design, sketch out a design for your database. When the book explains how to normalize a database, normalize yours. As you work through the exercises, think about how they would apply to your dream database.

Don't be afraid to ask your instructor if you can use your database instead of one suggested by the book for a particular assignment. (Unless you have one of those instructors who hand out extra work to anyone who crosses their path. In that case, keep your head down.) Usually an instructor's thought process is quite simple: "I don't care what database you use as long as you learn the material." Your database may need to contain several related tables to create the complexity needed for a particular exercise but it's usually not too hard to make a database more complex.

When you're finished, you will hopefully know a lot more about database design than you do now and, if you're persistent, you might just have a database that's actually good for something. Hopefully you'll also know how to design other useful databases in the future. (And when you're finished, email me at **RodStephens@vb-helper.com** and let me know what you built!)

Conventions

To help you get the most from the text and keep track of what's happening, we've used a number of conventions throughout the book.

TRY IT OUT

The *Try It Out* is an exercise you should work through, following the text in the book.

1. They usually consist of a set of steps.
2. Each step has a number.
3. Follow the steps through with your copy of the database.

How It Works

After most *Try It Out* sections, the process you've stepped through will be explained in detail.

NOTE

Tips, hints, tricks, and asides to the current discussion are offset and placed in italics *like this*.

As for styles in the text:

- We *highlight* new terms and important words when we introduce them.
- We show keyboard strokes like this: Ctrl+A.
- We show file names, URLs, and code within the text like so: `SELECT * FROM Students`.
- We present blocks of code like this: `We use a monofont type with no highlighting for code examples`.

Source Code

As you work through the examples in this book, you may choose either to type in all the code manually or to use the source code files that accompany the book. All of the source code used in this book is available for download at `www.wrox.com`. Once at the site, simply locate the book's title (either by using the Search box or by using one of the title lists) and click the Download Code link on the book's detail page to obtain all the source code for the book.

NOTE

Because many books have similar titles, you may find it easiest to search by ISBN; this book's ISBN is 978-0-470-38549-4.

Once you download the code, just decompress it with your favorite compression tool. Alternatively, you can go to the main Wrox code download page at. `www.wrox.com/dynamic/books/download.aspx`. to see the code available for this book and all other Wrox books.

The Book's Web Site

No book can possibly cover everything there is to know about any topic and this book is no exception. I have tried to make it as complete, correct, and understandable as possible but there isn't room for everything here.

To get the most out of this book, you should visit the book's Web page. There you will find additional useful information that didn't fit in the book such as checklists and user requirement surveys that you can download and print, corrections and clarifications, example SQL scripts, forums for questions and discussion, and other supplementary material.

To visit the book's Wrox Web site, go to. `www.wrox.com` and search for the book's title or ISBN, or for the author's name Rod Stephens. This Web site includes author information, excerpts, example programs that you can download, and so forth.

NOTE

Please visit the book's Web site and look for additions and addendums. I also monitor the book's Wrox forum closely and answer questions as quickly as I can.

The book's author web site, `www.vb-helper.com/db_design.htm`, contains similar material and links to the Wrox Web site. The main VB Helper Web site also contains thousands of tips, tricks, and examples written in various versions of Visual Basic.

To keep informed of changes to this book or my other books, you can sign up for one of my newsletters at. `www.vb-helper.com/newsletter.html`. The newsletters, which are sent every week or so, include Visual Basic tips, tricks, and examples, in addition to updates on my books and other thoughts about Visual Basic development.

If you have corrections or comments, please send them to me at **RodStephens@vb-helper.com**. I will try to help you out and do my best to keep the Web sites as up-to-date as possible.

Errata

We make every effort to ensure that there are no errors in the text or in the code. However, no one is perfect, and mistakes do occur. If you find an error in one of our books, like a spelling mistake or faulty piece of code, we would be very grateful for your feedback. By sending in errata you may save another reader hours of frustration and at the same time you will be helping us provide even higher quality information.

To find the errata page for this book, go to `www.wrox.com` and locate the title using the Search box or one of the title lists. Then, on the book details page, click the Book Errata link. On this page you can view all errata that has been submitted for this book and posted by Wrox editors. A complete book list including links to each book's errata is also available at `www.wrox.com/misc-pages/booklist.shtml`.

If you don't spot "your" error on the Book Errata page, go to `www.wrox.com/contact/techsupport.shtml` and complete the form there to send us the error you have found. We'll check the information and, if appropriate, post a message to the book's errata page and fix the problem in subsequent editions of the book.

`p2p.wrox.com`

For author and peer discussion, join the P2P forums at `p2p.wrox.com`. The forums are a Web-based system for you to post messages relating to Wrox books and related technologies and interact with other readers and technology users. The forums offer a subscription feature to email you topics of interest of your choosing when new posts are made to the forums. Wrox authors, editors, other industry experts, and your fellow readers are present on these forums.

At `p2p.wrox.com` you will find a number of different forums that will help you not only as you read this book, but also as you develop your own applications. To join the forums, just follow these steps:

1. Go to `p2p.wrox.com` and click the Register link.
2. Read the terms of use and click Agree.
3. Complete the required information to join as well as any optional information you wish to provide and click Submit.
4. You will receive an e-mail with information describing how to verify your account and complete the joining process.

NOTE

You can read messages in the forums without joining P2P but in order to post your own messages, you must join.

Once you join, you can post new messages and respond to messages other users post. You can read messages at any time on the Web. If you would like to have new messages from a particular forum emailed to you, click the Subscribe to this Forum icon by the forum name in the forum listing.

For more information about how to use the Wrox P2P, be sure to read the P2P FAQs for answers to questions about how the forum software works as well as many common questions specific to P2P and Wrox books. To read the FAQs, click the FAQ link on any P2P page.

Contacting the Author

If you have questions, suggestions, comments, or just want to say "Hi," email me at **RodStephens@vb-helper.com**. I can't promise that I'll be able to help you with every problem, but I do promise to try.

Disclaimer

Many of the examples in this book were chosen for interest or humorous effect. They are not intended to disparage anyone. I mean no disrespect to police officers (or anyone else who regularly carries a gun), plumbers, politicians, jewelry store owners, street luge racers (or anyone else who wears helmets and Kevlar body armor to work), or college administrators. Or anyone else for that matter.

Well, maybe politicians.