/Theory/In/Practice

O'REILLY®

Database In Depth

Relational Theory for Practitioners

C. J. Date

Database

Relational databases are ubiquitous in today's world. They're everywhere, from large e-commerce sites such as Amazon.com to small MP3 players that you can hold in the palm of your hand. Few technology professionals can afford to be without a good grounding in the fundamentals of relational database technology, yet many today who work with databases have had no formal training in relational theory.

In *Database in Depth*, author and well-known database authority Chris Date lays out the fundamentals of the relational model. This model was first introduced to the world in 1969 by E. F. Codd in his seminal paper "Derivability, Redundancy, and Consistency of Relations Stored in Large Data Banks," and it forms the basis for all database products in wide use today.

In this book, Chris dispels many commonly held misconceptions about the relational model, explaining that:

_The term "relational" has nothing to do with relating two tables on a common set of columns.

_Relations are multidimensional. They are not flat. They are not two-dimensional. Don't let the term "table" mislead you!

_Nulls are most certainly not values, even though the SQL standard calls them so.

_Attributes of a relation can contain values of arbitrary complexity, including such things as arrays, XML documents, and even other relations.

_Base relations do not necessarily have to be physically stored.

_SQL is not a set-oriented language, but rather is bag-oriented.

If you work with databases, you cannot afford to be without this book. Chris writes for the practitioner–for you–explaining clearly the fundamental principles that you need to know in order to be successful at what you do. Don't let a lack of formal education in database theory hold you back. Instead, let Chris's clear explanation of relational concepts, set theory, the difference between model and implementation, relational algebra, normalization, and much more set you apart and well above the competition when it comes to getting work done with a relational database.

C. J. Date was one of the first to recognize the genius in E. F. Codd's vision. He became a colleague of Codd's early on, worked closely with Codd during the formative years of the relational model, and has been a strong influence in the development of the database technology that you use day-in and day-out. This is your chance to learn from the master. Don't pass it by.

O'REILLY® www.oreilly.com

US $29.95 CAN $41.95
ISBN-10: 0-596-10012-4
ISBN-13: 978-0-596-10012-4

9 780596 100124 52995

Database in Depth

Database in Depth

Relational Theory for Practitioners

C.J. Date

O'REILLY®

Beijing • Cambridge • Farnham • Köln • Paris • Sebastopol • Taipei • Tokyo

Database in Depth
Relational Theory for Practitioners
by C.J. Date

Published by O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472.

Editor:	Jonathan Gennick
Production Editor:	Genevieve d'Entremont
Cover Designer:	MendeDesign, *www.mendedesign.com*
Interior Designer:	Marcia Friedman
Creative Director:	Michele Wetherbee
Printing History:	May 2005: First Edition.

This book uses RepKover™, a durable and flexible lay-flat binding.
ISBN: 0-596-10012-4
[M] [10/05]

Dedicated to the memory of E. F. Codd

Those who are enamored of practice without theory are like a pilot who goes into a ship without rudder or compass and never has any certainty where he is going. Practice should always be based upon a sound knowledge of theory.

Leonardo da Vinci (1452 – 1519)

The trouble with people is not that they don't know but that they know so much that ain't so.

Josh Billings (1818 – 1885)

TABLE OF CONTENTS

Foreword

I remember clearly the first time I didn't buy a book by Chris Date. That's right, I said "didn't buy." It was late 1991 or early 1992. I was in a small bookstore in the Saginaw, Michigan mall, looking for a book on SQL. I found one, too, by some guy named C. J. Date, whom I'd never heard of. The book looked good, the writing was clear, half of me wanted to buy it, but…oh, the price! I stood there holding a book in my hands that was about 1/4-inch thick, had less than 200 pages, yet it cost $30—a lot for me back in those days. I struggled with myself for a bit, then, reluctantly, I put the book back on the shelf and walked away.

What a mistake! A career was in the balance—I just didn't know it at the time.

I redeemed myself, though, by convincing my boss that the company should buy the book for me. It took them a month to get around to ordering the book, but it finally arrived, and I read it, not once, but several times. One of the best and most educational parts of the book (for me) was an appendix with Chris's critique of the then-current SQL language.

I learned a lot from Chris, from that book. I became enamored of working with databases, and fascinated by the idea of a declarative language like SQL: I could just describe the results I was after, and the database engine would do the work of deriving those results

for me. I also learned more about Chris, and about his role working with Codd from almost the very beginning of the relational era. For several years I subscribed to a magazine called *Database Programming and Design* for the sole reason that Chris wrote a column in it.

The knowledge I gained about SQL from Chris—beginning with that one thin, expensive little book that I was so reluctant to buy—has had a ripple effect down through the years until today. My introduction to databases at that time was a major turning point in my career, and Chris's clarity and rigor had a profound effect on the way in which I viewed SQL and database technology in general. Ever since, SQL has been one of my favorite things to write, to learn about, and, more recently, to write about.

I shudder to think of what my career might be today had I not gotten my hands on Chris's book. Don't make the same mistake I almost made. Don't put this book back on the shelf. Read it! Learn from someone who helped to invent and refine the relational model on which your career depends. You may not agree with everything Chris has to say, and you don't need to, but do *understand* what he says. Take the time to understand his vision of the relational model today. Take the time to understand the issues he raises with respect to how vendors (sometimes wrongly) implement that model in their products. Do these things, and you'll find yourself standing head-and-shoulders above your peers who haven't taken the time to ground themselves in the fundamentals.

If I can leave you with one thought, it's on the importance of enthusiasm for learning. It's been a great honor for me to edit this book by Chris Date, and an honor I never even remotely expected to have. I learned a lot from talking with Chris and reading his drafts. How did my work on this book come about? It was the indirect result of intense discussion of subquery optimization in SQL with several very smart people on the Oracle-L list whom I'm humbled to call colleagues. Even more indirectly, I suppose, my work on this book is a result of my reading Chris's writings for the first time, so very many years ago. Curiosity and a love for learning have taken me a lot further in my career than I ever used to dream of going, and they can do the same for you.

—Jonathan Gennick
Munising, Michigan
March 2005

Preface

After many years working in the database community in various capacities, I've come to realize there's a real need for a book for practitioners (not novices) that explains the basic principles of relational theory in a way not tainted by the quirks and peculiarities of existing products, commercial practice, or the SQL standard. I wrote this book to fill that need. My intended audience is thus experienced database practitioners or other database professionals who are honest enough to admit they don't understand the theory underlying their own field as well as they might, or should. That theory is, of course, the relational model—and while it's true that the fundamental ideas of that theory are all quite simple, it's also true that they're widely misrepresented, or underappreciated, or both. Often, in fact, they don't seem to be understood at all. For example, here are a few relational questions. How many of them can you answer?

1. What exactly is first normal form?
2. What's the connection between relations and predicates?
3. What's semantic optimization?
4. What's a join dependency?
5. Why is semidifference important?
6. Why doesn't deferred integrity checking make sense?

7. What's a relation variable?
8. What's nonloss decomposition?
9. Can a relation have an attribute whose values are relations?
10. What's the difference between SQL and the relational model?
11. Why is *The Information Principle* important?
12. How does XML fit with the relational model?

This book provides answers to these and many related questions. Overall, it's meant to help database practitioners understand relational theory in depth and make good use of that understanding in their professional day-to-day activities.

What Makes This Book Different?

I must immediately explain that very little of the technical substance of this book is new. I've said most of it before, in previous books and other publications—I've just looked around and seen that it needs to be said again. But I've tried to say it differently this time: the sequence is different, the development is different, the style and treatment are different, and the intended audience is different (more on this last point in a moment). So while parts of the material have appeared before in some form or another in a variety of places, I do regard this as a totally new book. Of course, some portions of the text are, inevitably, similar to things I've written elsewhere, because the material all comes out of the same place, as it were: namely, my own brain, and my experience in teaching this material in live seminars over many years. But there's no direct plagiarism; direct plagiarism wouldn't serve my purpose. However, I have consciously reused many of my old examples, because those examples have been very carefully tailored over the years to illustrate exactly the points I want to make, no more and no less.

Let me come back to that point about the intended audience for this book being different. As already indicated, I've published several previous books in the field of database technology. So how is this one different? In particular, does it compete with any of those existing books?

In my view, the answer to the latter question is no. I have two books from Addison-Wesley that might look at first sight as if they could be competitors to this one:

- *An Introduction to Database Systems*, Eighth Edition (2004)
- *Databases, Types, and the Relational Model: The Third Manifesto*, Third Edition (coauthored with Hugh Darwen, to appear 2006)

However, the first of these, though I call it an "introduction," actually covers the whole of the database field, not just the relational model. It's meant primarily as a college text, and it doesn't assume any prior database knowledge or experience on the part of its readers; also, the style is much more formal than that of the present book, as befits a textbook.

The second is an extensive reworking of an earlier book by Hugh Darwen and myself called *Foundation for Future Database Systems: The Third Manifesto*, Second Edition (Addison-Wesley, 2000). This one is an advanced (graduate-level?) text, and it's even more formal—not to say terse—than the first book. Although there's obviously some overlap in subject matter, therefore, I don't really see any of these three books as competing with the other two.

Another significant point of difference is that the present book is mainly meant for self-study (though there are portions you might want to discuss with your friends and colleagues and coworkers). There are exercises, too, to help reinforce the material; there's no obligation to do those exercises, of course, but I think it's a good idea to have a go at some of them at least. Answers, often giving more information about the subject at hand, can be found online at *http://oreilly.com/catalog/databaseid*.

While I'm on the topic of possible competition, I should mention a couple of other books of mine (the first from Addison-Wesley again, the other from Morgan Kaufmann):

- *The Database Relational Model: A Retrospective Review and Analysis* (2001)
- *Temporal Data and the Relational Model* (coauthored with Hugh Darwen and Nikos A. Lorentzos, 2003)

In my opinion, the first of these complements the present book, in that it reviews and analyzes, in a fairly informal style, the series of papers by Ted Codd that first introduced the relational model to the world at large. And the second is concerned, as its title indicates, not with relational theory as such but with a specific application of that theory. While the first chapter of that book does contain an overview of the relational model and thus *might* be considered to compete slightly with the present book, I don't really think it does.

The net of all of the above is this: although I've written on most of these topics before in a variety of places, and sometimes in unavoidably similar terms, I don't think any other publication by myself—or anyone else, so far as I know—brings them together and covers them in a way that's even close to the way the present book does.

Further Preliminaries

I need to take care of several further preliminaries. As I've already said, I'll be using some of the same examples here as in other books and articles of mine; in particular, the running example is the famous (or infamous) suppliers-and-parts database. I apologize for dragging this old warhorse out yet one more time, but the remark I made earlier about examples having been very carefully designed to illustrate exactly the points I want to make applies to this particular example in spades.

Second, my own understanding of the relational model has evolved over the years, and continues to do so. This book represents my very latest thinking on the subject; thus, if you detect any discrepancies between this book and the ones already mentioned (and

there are a few), the treatment in this book should be taken as superseding that in those earlier ones. Though I hasten to add that such discrepancies are mostly of a fairly minor nature; what's more, I've taken care always to relate new terms and concepts to earlier ones, whenever I felt it was necessary to do so.

Third, I am, of course, going to talk about theory—but it's an article of faith with me that *theory is practical*. I mention this point explicitly because so many people seem to believe the exact opposite: namely, that if something's theoretical, it can't be practical. But the truth is that theory—at least, the theory I'm talking about here, which is relational theory, of course—is most definitely very practical indeed. The purpose of that theory is *not* just theory for its own sake; the purpose of that theory is to allow us to build systems that are 100 percent practical. Every detail of the theory is there for solid practical reasons. Indeed, much of that theory is not only practical, it's fundamental, straightforward, simple, useful, and it can be *fun* (as I hope to demonstrate in the course of this book).

(In fact, we really don't have to look any further than the relational model itself to find the most striking possible illustration of the foregoing thesis. Indeed, it really shouldn't be necessary to defend the notion that theory is practical, in a context such as ours: namely, a multibillion dollar industry totally founded on one great theoretical idea. But I suppose the cynic's position would be "Yes, but what has theory done for me lately?" In other words, those of us who do think theory is important must continually justify ourselves to our critics—which is another reason why I think a book like this one is needed.)

And another point: the standard "relational" language is SQL, of course, and I assume you're reasonably familiar with that language, as well as with basic database concepts in general. As you'll soon see, however, I'm pretty critical of SQL in what follows. The sad fact is that SQL fails in all too many ways to support the relational model properly; it suffers from numerous sins of both omission and commission (which is why I said it was "relational," in quotation marks, when I first mentioned it). But this state of affairs is precisely one of the reasons why you need to know the relational model! Because SQL's support for the model is so deficient, it gives you rope to hang yourself; so you need to know the theory in order *not* to hang yourself—that is, you need to know the theory in order to enforce for yourself the various disciplines that SQL really ought to enforce on your behalf but doesn't. A good example is duplicate rows: SQL allows duplicate rows, but the relational model doesn't. So you need to know *why* the model doesn't allow them in order to understand why it's important not to "take advantage" of this particular SQL "feature." As one reviewer of my original proposal for this book, Stephane Faroult, wrote: "When you have a bit of practice, you realize there's no way to avoid having to know the theory."

Talking of SQL, by the way, please note that I use the term *SQL* throughout the book to mean the standard version of that language exclusively, not some product-specific dialect (barring explicit statements to the contrary, of course). In particular, I follow the standard in pronouncing the name "ess cue ell," not "sequel" (though this latter pronunciation is common in the field), and therefore say things like *an* SQL table, not *a* SQL table.

Finally, I'd like to mention that a live one-day seminar is available based on the material in this book. See *http://www.dbdebunk.com* or *http://www.thethirdmanifesto.com* for further details.

Conventions Used in This Book

The following typographical conventions are used in this book:

Italic
: Used for emphasis and indicates new terms. Italic also indicates when a variable, such as *x*, is used in place of something else during a discussion in the main text of the book.

`Constant width`
: Used for code examples.

`Constant width italic`
: Marks the occurrence of a variable or user-supplied element in a code example.

Using Code Examples

This book is here to help you get your job done. In general, you may use the code in this book in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing a CD-ROM of examples from O'Reilly books *does* require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation *does* require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: "*Database in Depth: Relational Theory for Practitioners*, by C. J. Date. Copyright 2005 O'Reilly Media, Inc., 0-596-10012-4."

If you feel your use of code examples falls outside fair use or the permission given above, feel free to contact us at *permissions@oreilly.com*.

Safari Enabled

When you see a Safari® enabled icon on the cover of your favorite technology book, that means the book is available online through the O'Reilly Network Safari Bookshelf.

Safari offers a solution that's better than e-Books. It's a virtual library that lets you easily search thousands of top tech books, cut and paste code samples, download chapters, and find quick answers when you need the most accurate, current information. Try it free at *http://safari.oreilly.com*.

Comments and Questions

We've done our best to make this book as error-free as we can, but you might find mistakes. If so, please notify us by writing to:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
(800) 998-9938 (in the United States or Canada)
(707) 829-0515 (international or local)
(707) 829-0104 (FAX)

You can also send messages electronically. To be put on the mailing list or request a catalog, send email to:

info@oreilly.com

To ask technical questions or comment on the book, send email to:

bookquestions@oreilly.com

We have a web site for this book, where you can find examples and errata (previously reported errors and corrections are available for public view there). You can access this page at:

http://www.oreilly.com/catalog/databaseid

For more information about this book and others, see the O'Reilly web site:

http://www.oreilly.com

Acknowledgments

It's a pleasure to acknowledge my debt to the many people involved, directly or indirectly, in the production of this book. In particular, I'd like to mention my reviewers Stephane Faroult, Jonathan Gennick, Lex de Haan, Anthony Molinaro, Peter Robson, and Michael Wener for their helpful comments on earlier drafts of the manuscript. I'd also like to thank Nagraj Alur and Hugh Darwen for various technical discussions. Next, I'd like to thank my wife Lindy (as always) for her support throughout this and all of my other database projects over the years. Finally, I'm grateful to everyone at O'Reilly—especially Jonathan Gennick and Genevieve d'Entremont—for their encouragement, contributions, and support throughout the production of this book.

—C. J. Date, Healdsburg, California, 2005

CHAPTER ONE

Introduction

Professionals in any discipline need to know the foundations of their field. So if you're a database professional, you need to know the relational model, because that model is the foundation (or a huge part of the foundation, anyway) of the database field in particular. Now, every course in database management, be it academic or commercial, does at least pay lip service to the idea of teaching the relational model—but most of that teaching seems to be done very badly, if results are anything to go by. The relational model certainly isn't very well or very widely understood in the database community at large. Here are some possible reasons for this state of affairs:

- The model is taught in a vacuum. That is, for beginners at least, it's hard to see the relevance of the material, or it's hard to understand the problems it's meant to solve, or both.
- The instructors themselves don't fully understand or appreciate the significance of the material.
- (Most likely in practice.) The model as such isn't taught at all—the SQL language or some specific dialect of that language, such as the Oracle dialect, is taught instead.

So this book is aimed at database professionals, especially commercial database practitioners, who have had some exposure to the relational model but don't know as

much about it as they ought to. It's definitely *not* meant for beginners; however, it isn't a refresher course, either. To be more specific: I'm sure you know something about SQL, but—and I apologize if my tone is somewhat offensive here—if your knowledge of the relational model derives only from your knowledge of SQL, then I'm afraid you won't know the relational model as well as you should, and you'll probably know "some things that ain't so."

> **NOTE**
> **SQL ≠ the relational model!**

Here by way of illustration are some relational issues that SQL isn't too clear on (to put it mildly):

- What databases, relations, and tuples really are
- The difference between relations and types
- The difference between relation values and relation variables
- The relevance of predicates and propositions
- The legitimacy of relation-valued attributes
- The crucial role of integrity constraints

and so on (this isn't an exhaustive list). All of these issues, and of course many others, are addressed in this book.

I say again: if your knowledge of the relational model derives only from your knowledge of SQL, then you might know "some things that ain't so." One unfortunate consequence of this state of affairs is that you might find, in reading this book, that there are some things you have to unlearn—and unlearning something is notoriously hard to do. And a related point…I'd like to recommend, politely, that you not skip the discussion of some topic because you think you're thoroughly familiar with that topic already. For example, are you *sure* you know exactly what a key is, in relational terms? Or a join?

A Remark on Terminology

In that list of relational issues in the foregoing section, you probably noticed right away that I used the formal terms *relation, tuple,** and *attribute*. SQL doesn't use these terms, of course—it uses the more "user-friendly" terms *table, row,* and *column* instead. And I'm generally sympathetic to the idea of more user-friendly terms if they can help make the ideas more palatable. In the case at hand, however, it seems to me that, regrettably, they don't make the ideas more palatable; instead, they distort them, and in fact do the cause of genuine understanding a grave disservice. The truth is, a relation is *not* a table, a tuple

* Usually pronounced to rhyme with couple.

is *not* a row, and an attribute is *not* a column. And while it might be acceptable to pretend otherwise in informal contexts—indeed, I've done so myself, in many of my books and other writings—I would argue that it's acceptable only if we all understand that the more user-friendly terms are just an approximation to the truth and fail overall to capture the essence of what's really going on. To put it another way: if you do understand the true state of affairs, then judicious use of the user-friendly terms can be a good idea; but in order to learn and appreciate that true state of affairs in the first place, you really do need to come to grips with the more formal terms. In this book, therefore, I'll use those more formal terms most of the time—and of course I'll give precise definitions for them when we need them (mostly not in this first chapter, though, where I'm just trying to lay a certain amount of elementary groundwork).

And another point on terminology: having said that SQL tries to simplify one set of terms, I must now add that it does its best to complicate another. I refer to its use of the terms *operator, function, procedure, routine,* and *method,* all of which refer to essentially the same thing (with, perhaps, very minor differences). In this book I'll use the term *operator* throughout.

Talking of SQL, by the way, *please note that I use the term SQL to mean the standard version of that language exclusively,* not some product-specific dialect—barring explicit statements to the contrary, of course. (I did mention this point in the preface, but I know that few people actually read prefaces.) In particular, my criticisms of SQL apply to the standard version specifically. Thus, if some particular criticism happens not to apply to your own favorite product, well, good, I'm glad to hear it (and bully for you).

Principles, Not Products

It's worth taking a few moments to examine the question of why, as I claimed earlier, you as a database professional need to know the relational model. The reason is that the relational model isn't product-specific; rather, it is concerned with *principles.* What do I mean by *principles?* Well, here's a definition (from *Chambers Twentieth Century Dictionary*):

> **principle:** a source, root, origin: that which is fundamental: essential nature: theoretical basis: a fundamental truth on which others are founded or from which they spring

The point about principles is this: they *endure*. By contrast, products and technologies (and the SQL language, come to that) change all the time—but principles don't. For example, suppose you know Oracle; in fact, suppose you're an expert on Oracle. But if Oracle is *all* you know, then your knowledge is not necessarily transferable to, say, a DB2 or SQL Server environment (it might even get in the way of your making progress in that new environment). But if you know the underlying principles—in other words, if you know the relational model—then you have knowledge and skills that *will* be transferable: knowledge and skills that you'll be able to apply in *every* environment and that will never be obsolete.

In this book, therefore, we'll be concerned with principles, not products, and foundations, not fads. Of course, I realize that sometimes you do have to make compromises and trade-offs in the real world. For one example, sometimes you might have good pragmatic reasons for not designing the database in the theoretically optimal way (an issue I discuss in Chapter 7). For another, consider SQL once again. Although it's certainly possible to use SQL relationally (for the most part, at any rate), sometimes you'll find—because existing implementations are so far from perfect—that there are severe performance penalties for doing so…in which case you might more or less be forced into doing something not "truly relational" (like writing a query in some weird and unnatural way in order to get the implementation to use an index). However, I believe very firmly that you should always make such compromises and trade-offs from *a position of conceptual strength*. That is:

- You should understand what you're doing when you do have to make such a compromise.
- You should know what the theoretically correct situation is, and you should have very good reasons for departing from it.
- You should document those reasons, too, so that if they go away at some future time (for example, because a new release of the product you're using does a better job in some respect), then it might be possible to back off from the original compromise.

The following quote—which is attributed to Leonardo da Vinci (1452–1519) and is thus some 500 years old!—sums up the situation admirably:

> Those who are enamored of practice without theory are like a pilot who goes into a ship without rudder or compass and never has any certainty where he is going. *Practice should always be based on a sound knowledge of theory.*

(OK, I added the italics.)

A Review of the Original Model

You're a database professional, so you already have some familiarity with the relational model. The purpose of this section is to serve as a kickoff point for our subsequent discussions; it reviews some of the most basic aspects of that model as originally defined. Note the qualifier "as originally defined"! One widespread misconception about the relational model is that it's a totally static thing. It's not. It's like mathematics in that respect: mathematics too is not a static thing but changes over time. In fact, the relational model can itself be seen as a small branch of mathematics; as such, it evolves over time as new theorems are proved and new results discovered. What's more, those new contributions can be made by anyone who's competent to do so. Like mathematics again, the relational model, though originally invented by one man, has become a community effort and now belongs to the world.