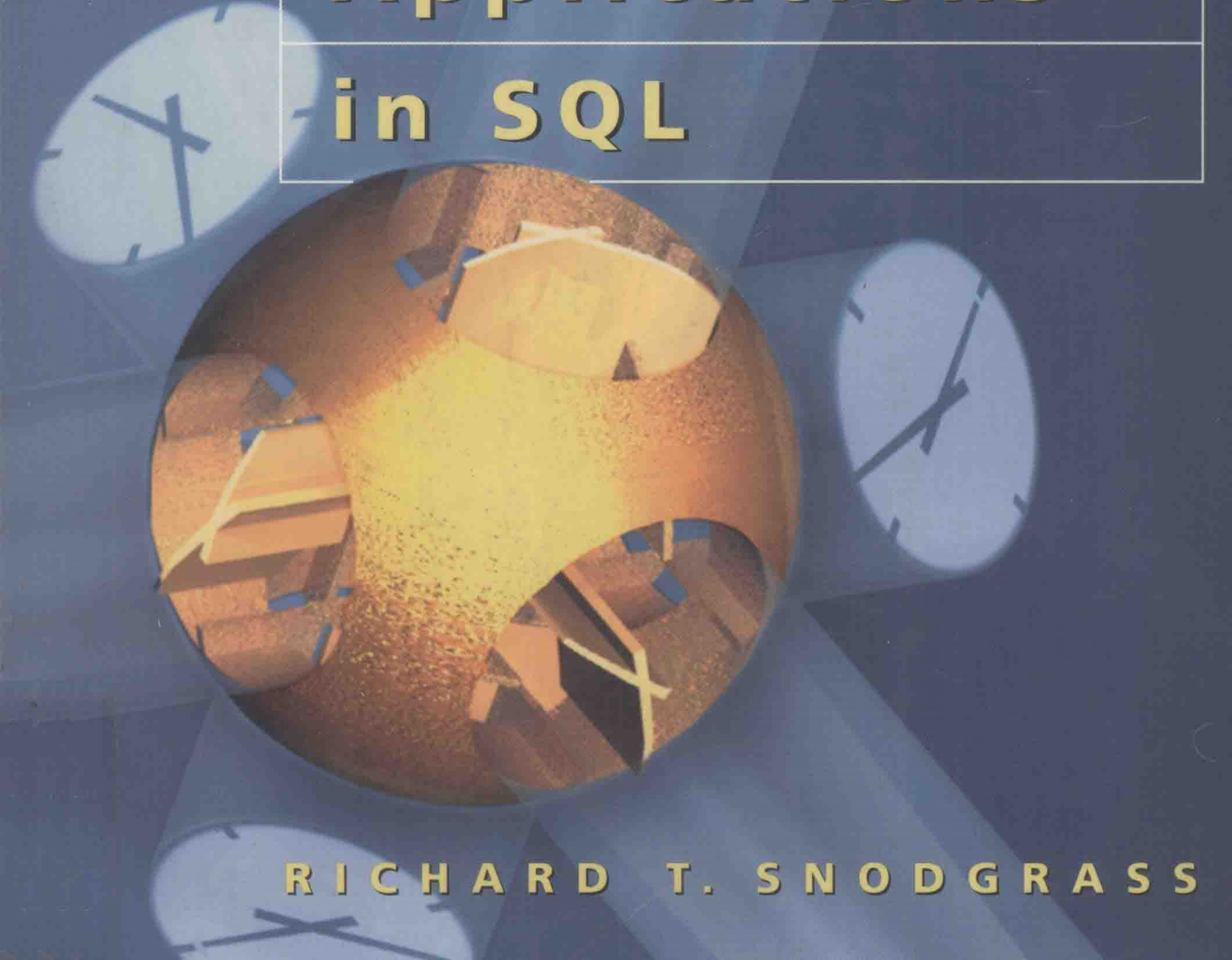
Developing Time-Oriented Database Applications in SQL



RICHARD T. SNODGRASS

Developing Time-Oriented Database Applications in SQL

RICHARD T. SNODGRASS

Morgan Kaufmann Publishers
San Francisco, California

Senior Editor	Diane D. Cerra
Director of Production & Manufacturing	Yonie Overton
Production Editor	Edward Wade
Editorial Coordinator	Belinda Breyer
Cover Design	Ross Carron Design
Cover Illustration	Michael Bloomenfeld, AEC
Text Design	Mark Ong, Side By Side Studios
Technical Illustration	Cherie Plumlee
Composition	Ed Sznyter, Babel Press
Copyeditor	Ken DellaPenta
Proofreader	Jennifer McClain
Indexer	Steve Rath
Printer	Courier Corporation

Chapter opener art based on illustrations from the following sources: Chapters 1, 12, and 13 from Brackin, A. J., *Clocks: Chronicling Time, Encyclopedia of Discovery and Invention*, Lucent Books, 1991; Chapters 2, 3, 4, 5, and 6 from Woodward, P., *My Own Right Time: An Exploration of Clockwork Design*, Oxford University Press, 1995; Chapters 7, 8, and 9 from Britton, F. J., *The Escapements: Their Action, Construction, and Proportion*, Geo. K. Hazlitt & Co., reprinted by Arlington Book Co, 1984; and Chapters 10 and 11 from Headrick, M. V., *Clock and Watch Escapement Mechanics*, self-published, 1997.

Morgan Kaufmann Publishers
Editorial and Sales Office
340 Pine Street, Sixth Floor
San Francisco, CA 94104-3205
USA

Telephone	415-392-2665
Facsimile	415-982-2665
Email	*mkp@mkp.com*
WWW	*http://www.mkp.com*

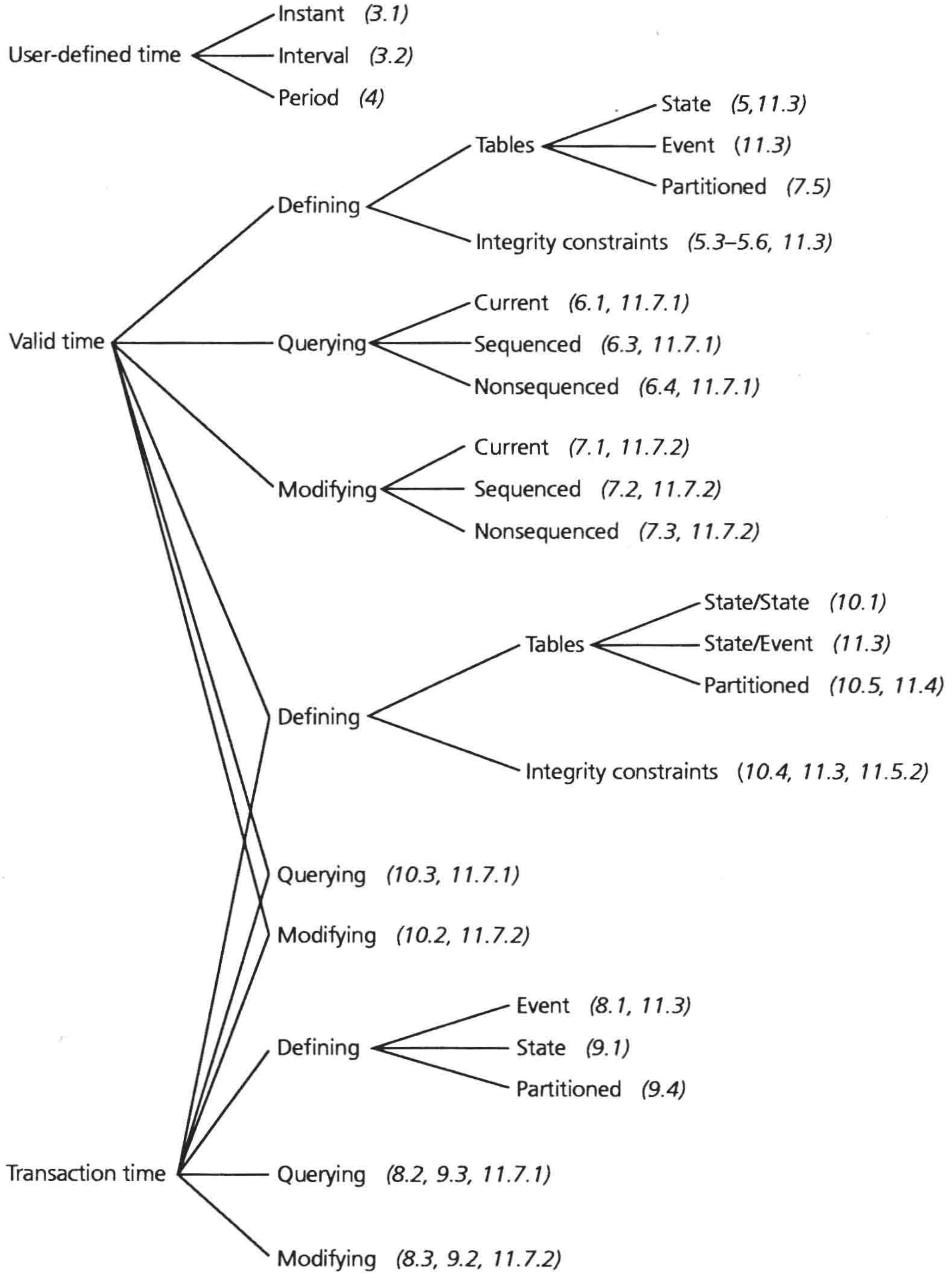
Order toll free 800-745-7323

Library of Congress Cataloging-in-Publication Data.

Snodgrass, Richard T.
Developing time-oriented database applications in SQL / Richard T. Snodgrass.
p. cm.
Includes bibliographical references.
ISBN 1-55860-436-7
1. SQL (Computer program language) 2. Database design.
I. Title.
QA76.73.S67S56 2000
005.75'6–dc21

99-14298
CIP

Concept Map



User-defined time
Instant (3.1)
Interval (3.2)
Period (4)
Valid time
Defining
Tables
State (5,11.3)
Event (11.3)
Partitioned (7.5)
Integrity constraints (5.3–5.6, 11.3)
Querying
Current (6.1, 11.7.1)
Sequenced (6.3, 11.7.1)
Nonsequenced (6.4, 11.7.1)
Modifying
Current (7.1, 11.7.2)
Sequenced (7.2, 11.7.2)
Nonsequenced (7.3, 11.7.2)
Defining
Tables
State/State (10.1)
State/Event (11.3)
Partitioned (10.5, 11.4)
Integrity constraints (10.4, 11.3, 11.5.2)
Querying (10.3, 11.7.1)
Modifying (10.2, 11.7.2)
Defining
Event (8.1, 11.3)
State (9.1)
Partitioned (9.4)
Transaction time
Querying (8.2, 9.3, 11.7.1)
Modifying (8.3, 9.2, 11.7.2)

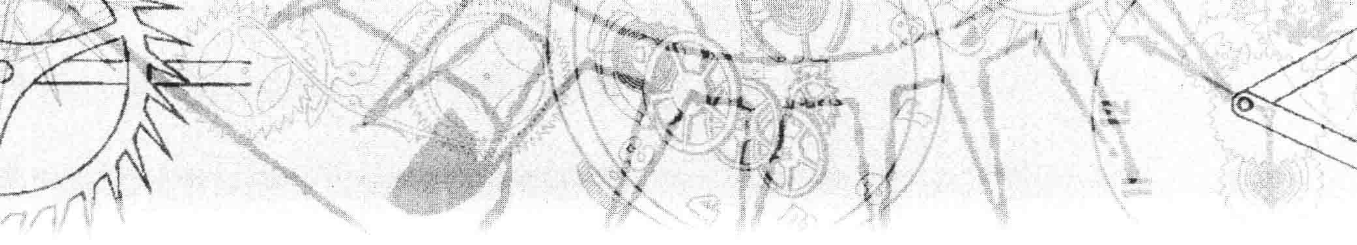
Developing Time-Oriented Database Applications in SQL

Á Merrie
Ce jour, et toujours

Mach' es wie die Sonnenuhr
Zähl' die heiteren Stunden nur

Do like the sundial:
Count only the bright hours

— *German proverb*



Foreword

by Jim Gray
Microsoft Research

Precise clocks were developed so that seafarers could find their longitude. Precise temporal data techniques were recently developed to help database designers record and reason about temporal information. It is paradoxical that we are only now coming to understand how to think about time and represent it in formal systems. After all, time is the fourth dimension; it is at the core of existence. Yet, it is only recently that we have come to understand the fundamental concepts of instants, intervals, periods, sequenced changes, valid time, transaction time, and a bitemporal view of information.

Richard Snodgrass and his colleagues have explored temporal data concepts over the last two decades. They now have a fairly complete solution to the problems. Indeed the concepts are now being added to the SQL language standard. This book summarizes their work and presents it in a very accessible and useful way.

Temporal databases, viewed from this modern perspective, are surprisingly simple and powerful. The book gives examples of 85-line SQL programs that collapse to 3-line programs when the new concepts are applied. It introduces the concepts using concrete examples and conventional SQL. I found this mix of theory and practice very instructive and very easy to follow.

The book explains that temporal databases can be designed in two steps. First, the static database can be designed. Then, in a second pass, each table and constraint is given its temporal attributes. This makes design much more tractable. This approach is made all the more attractive by the fact that the temporal SQL language extensions are just modifiers to standard queries and updates—this very elegant approach makes temporal issues orthogonal to the other language issues.

I highly recommend this book to anyone interested in temporal data—either designing and building databases that record information over time, or just understanding the concepts that underlie representing temporal information. This book does an excellent job of organizing and summarizing this important area.

Foreword

by Jim Melton
Oracle Corporation

It's about time—time that a book like this was written and time that the SQL community got the benefits of the careful analysis and thought put into the subject.

Rick Snodgrass is one of the relatively few researchers in the field of temporal databases and has proved himself to be one of the more important of those few, in part because he insists on applying the theoretical knowledge gained from his research to practical applications and to real products.

Snodgrass proposed in 1992 that temporal extensions to SQL be developed by the temporal database community. In response to this proposal, a virtual committee was formed to design extensions to the 1992 edition of the SQL standard (ANSI X3.135.-1992 and ISO/IEC 9075:1992); those extensions, known as TSQL2, were developed during 1993 by this committee meeting only via email. In late 1993, Snodgrass first presented this work to the group responsible for the American National Standard for Database Language SQL, ANSI Technical Committee X3H2 (now known as NCITS H2).

In response to Snodgrass's presentation, X3H2 proposed to the International Organization for Standardization (ISO) that the project to extend the standard for SQL be enhanced by adding a subproject for temporal extensions to the language. This proposal was accepted in 1994, and an initial document for ISO/IEC 9075-7, known as SQL/Temporal, was started. Over the next two years, a series of proposals from Snodgrass and others were considered by the ISO group responsible for SQL (ISO/IEC JTC1/SC21/WG3, later ISO/IEC JTC1/SC32/WG3), but progress was slowed considerably by the need to focus on what has recently been published as SQL:1999. Work will undoubtedly resume on progressing SQL/Temporal in 1999 for publication early in the next millennium, and Snodgrass will no doubt play a significant role in its standardization.

The book you hold has been a long time in the making, not only because the subject matter can seem overwhelmingly complex if not presented carefully, but also because of the great number of examples that Snodgrass has taken from real application systems and translated into standard SQL and its proposed extensions. (Of course, not all of the examples can be used in all SQL products today; some of them are directed toward specific vendors' systems, while others depend on future extensions to the language.) The result of that care and extensive use of examples is great clarity and focus, yielding ready comprehension to readers willing to give the book the attention it deserves. I recommend this book very highly to all SQL practitioners, especially those with an interest in the temporal semantics of data.

Preface

This is how it goes.

We develop a database application, and initially the project proceeds smoothly enough. There are alternatives to weigh during the schema design, problems to contend with while writing the SQL code, and constant reconfiguration and interaction with other programs and legacy data, but all in all the project is under control. Then we decide that one of the tables needs another DATE column, recording when the row was valid. (After all, we added a birth date column a few weeks ago, with no surprises.) So we rework the part of the application that maintains that table, noticing that the code is getting more complicated. During testing, we discover that the primary key no longer is sufficient. We add the DATE column to the primary key, acknowledging that this is only a stopgap measure, and hope that the input data will be well formed, as there isn't time to write code that checks those constraints properly. In the back of our mind is the lingering doubt that perhaps referential integrity checking isn't working quite right either.

We soon realize that we need another DATE column to record when the row was no longer valid. In doing so we encounter a raft of off-by-one bugs, in which some less-than comparisons should have been '<=', and other places where we need to add "+ `'1' DAY`". We think we've found all the code locations that need to be changed, but we're not sure. And we now know for a fact that the primary and foreign keys are wrong, but we don't know how to even approach that mess.

The code to modify the database is becoming increasingly convoluted. Each modification has to at least consider changing the DATE columns, but it isn't at all clear how to approach such changes in a systematic fashion. And even the most trivial queries, such as "Who was Aaron's manager when he worked on the Capital account?", which before we could code in our sleep, now become painful to even contemplate writing in SQL.

Around this time, users start complaining that reports aren't consistent, that copies of the end-of-the-year summary have different numbers in them. Looking into this anomaly, we finally figure out that the reports were run at different times, and the data had been changed in the meantime. We then realize that there is no way to correlate the end-of-the-year report with the cash flow report, unless they are run at the same time. Users are adopting an irreverent view of these reports: if you wait a few days, maybe the numbers will fix themselves.

To address the inconsistencies in the reports, someone suggests a quick fix: add another DATE column. The development group responds with astonishment and chagrin. How can we possibly get the code working with *another* DATE column, when we all know how much work resulted from adding the previous column? In fact, some in the group despair of ever getting the code as is, with just two DATE columns, working correctly—there are just too many arbitrary decisions, each layered on other equally ill-motivated quick fixes.

Looking back on the history of the development process, everyone has a vague idea that the problems started when that pesky DATE column was first added. How could one column flummox the whole system? And why do some columns, such as the birth date column, slide in smoothly, and other DATE columns cause no end of problems?

A PARADIGM SHIFT

Thomas Kuhn, in his insightful and highly influential book, *The Structure of Scientific Revolutions* [64], argued that science does not proceed in a linear, monotonic accumulation of knowledge, but rather exhibits intellectually jarring discontinuities, as radical ideas become the established world view, replacing the now-discredited prior conceptual foundation.

Two decades of research into temporal databases have unequivocally shown that a *time-varying table*, containing certain kinds of DATE columns, is a completely different animal than its cousin, the table without such columns. Effectively designing, querying, and modifying time-varying tables requires a different set of approaches and techniques than the traditional ones taught in database courses and training seminars. Developers are naturally unaware of these research results (and researchers are often clueless as to the realities of real-world applications development). As such, developers often reinvent concepts and techniques with little knowledge of the elegant conceptual framework that has evolved and recently consolidated, and researchers continue to conceal this framework with overly formal prose, never bothering to make the connection with existing tools at hand.

This book is an attempt to recast the insights from some 1600 papers in the research literature into terms usable by those brave SQL application coders working in the trenches. These concepts are integrated with the state-of-the-art approaches utilized by forward-thinking developers, as showcased in the case studies that form the bulk of the book. The result is, to use Kuhn's phrase, a *paradigm shift* in how we think about time-varying data. This shift impacts how such tables are specified, how they are maintained, and how they are queried.

PREREQUISITES

I assume you are comfortable with the SQL query language. This book is not a primer on that language, though I do cover the temporal data types and temporal constructs of SQL-92 in depth. There are many excellent books that serve as introductions to SQL.

It helps if you have implemented an application involving time-varying data, if only to realize firsthand how difficult and confusing such a project can be, and thus to appreciate the degree to which the approach presented here helps clear out the undergrowth and achieve an elegant and unfettered design. One chapter assumes familiarity with the entity-relationship model; the rest of the book focuses solely on the relational model.

The conceptual tools introduced here are in a specific and fundamental way extensions of existing strategies, so everything you've learned until now (well, almost everything) will be useful in this brave new world. The hardest part, for which I'll provide careful guidance, is to jettison the notion that this DATE column "is just another column." Operating under the old assumptions unhappily doesn't work, as project after project after project has shown. Paradigm shifts are always scary, but the benefits are there for those willing to make the jump.

WHAT TO READ

The best way to understand the principles of time-varying applications and their expression in SQL is to work through a series of tangible examples. By examining the design issues that arise and the kinds of constraints, queries, and modifications that we wish to express in implementing these specific applications, you will gain an appreciation of the abstract principles at play. For this reason, the bulk of this book is comprised of case studies.

Each case study sets the stage with a discussion of the application domain, which includes oil field records, cattle location information, and cadastral data. The relevant tables are introduced, followed by a discussion of the design, querying, and modification of these (time-varying) tables. While the applications and the people mentioned in the case studies all exist, the specific SQL examples have been tailored to bring out the issues under discussion.

The case studies were easy to locate. It seems that most database applications involve time-varying data. Indeed, applications that are inherently *not* temporal are about as prevalent as the proverbial hen's teeth. In fact, the only places you encounter nontemporal examples are in books and seminars, a phenomenon that unintentionally emphasizes the inherent complexity of time-varying applications.

To understand the fundamental concepts, you are encouraged to read *all* the chapters, even if you aren't an oil field engineer or a veterinarian. Each case study brings out a new category of temporal data, with its unique characteristics and demands. In fact, by studying other fields, you are relieved of the minutiae of your current environment. By studying a foreign language or culture, a deeper understanding of your own language or culture often follows as an additional, or even sometimes primary, benefit. After you have read the book, a productive approach to address a new set of requirements is to ask, To which case study is the application under development most closely related? Then the relevant code fragments can be customized to the problem at hand.

A few sections are marked with an asterisk to indicate advanced material. Feel free to skip these sections on a first, or even second, reading.

CASE STUDIES

Befitting the book's categorization as nonfiction, the people and their situations are as described herein. The specifics of their solutions to the problems presented by time-varying data have been adapted to better illustrate general approaches that I wish to emphasize. Most of the SQL code was written by use for the book, but it is reminiscent of that appearing in the actual applications. In the discussion, I have attempted to not oversimplify. Much of the complexity inherent in these applications is cleverly hidden in the details, and any realistic solution must ultimately confront the enterprise in all its glory and intricacy.

CD-ROM

The included CD-ROM contains the code fragments implemented in a variety of commercial systems, including IBM DB2 Universal Database (UDB), Ingres, Informix–Universal Server, Microsoft Access, Microsoft SQL Server, Sybase SQLServer, Oracle8 Server, and UniSQL. While these code fragments have been tested, the author and the publisher make no claims as to the suitability or correctness of these code fragments.

Also included are versions of some of the systems discussed in Chapter 12.

ERRORS

I would appreciate hearing about any errors that you find in the book, as well as receiving any other constructive suggestions you may have. (I'd especially like to hear of better ways to write individual code fragments.) Please email your comments to the author at *snodgrass@mkp.com*.

ACKNOWLEDGMENTS

The specific content of this book, as well as its overall composition, came about through collaboration with my dear friend and colleague Christian S. Jensen, who has assembled and directs the world's strongest temporal database research group at the University of Aalborg in Denmark. He is present on every single page, looking over my shoulder, probing, clarifying, generalizing, exemplifying. While the words in this book are mine (well, except for "exemplifying," which is definitely one of *his* words!), the ideas are jointly ours, comingled with the contributions of the many authors discussed in the "Readings" sections. Research is primarily an excuse to play with ideas, and concepts provide opportunities to interact on a deep level with others. Much of the joy I have experienced in this crazy business has directly or indirectly involved Christian, whether in the heady probing of trying to get at core truths, or in appreciating the shimmering simplicity of the resulting insights that, once uncovered, are in retrospect obvious.

Many others helped with the case studies, tested the code fragments, read drafts of this book, and offered nontechnical sustenance.

Cheryl Bach, Jim Barnett, Nigel Corbin, Brad De Groot, Jens Gadgaard, and Chris Janton were generous with their time in explaining the details of their applications, which provided the impetus for the case studies.

I thank Anindya Datta, Igor Viguier, and the Department of Information Systems at the University of Arizona for access to the Sybase DBMS. Cliff Leung from IBM, Paul Brown from Informix, Nick Kline and Goetz Graefe from Microsoft, Rafi Ahmed from Oracle, and Jack Reid from UniSQL Customer Support Services provided help with their respective DBMS products.

I appreciate comments on the manuscript from Rafi Ahmed, John Bair, Paul Brown, Jim Gray, Brad De Groot, Jeff Gross, Robert Brett Gulledge, Christian S. Jensen, and Dennis Shasha. Jim Melton went above and beyond the call of duty in providing *two* detailed reviews of the entire manuscript.

Amad Arsalan, Scott Calvert, Wen-Ke Chen, Brad De Groot, Alvin Gendrano, Bruce Huang, Vijaykumar Immanuel, Jie Li, Wei Li, Giedrius Slivinskas, Helen Thomas, Brad Traweek, and Inés F. Vega-López provided corrections to the code fragments. Alan Brucks provided information on the year 2000 problem.

I thank the following people who created the content of the CD-ROM: HTML templates (Christopher Cooper, Jose Alvin Gendrano, Rachana Shah, and Jian Yang), data types: IBM DB2 UDB (Brad Traweek and Giedrius Slivinskas), Informix–Universal Server (Jason Cox and Anand Wagle), Ingres (Wen-Ke Chen), Microsoft Access (Inés F. Vega-López, Lincoln Turner, and Ze-Yuan Zou), Microsoft SQL Server

(Inés F. Vega-López and Giedrius Slivinskas), Sybase SQLServer (Robert Brett Gulledge and Miltos Vafiadis), Oracle8 Server (Christopher Cooper and Jose Alvin Gendrano), and UniSQL (Qing Yan); valid-time state tables: IBM DB2 UDB (Vijaykumar Immanuel and Giedrius Slivinskas), Informix–Universal Server (Jason Cox), Microsoft Access (Ahmad Arsalan and Inés F. Vega-López), Microsoft SQL Server (Inés F. Vega-López and Giedrius Slivinskas), Sybase SQLServer (Wenmin Chen), Oracle8 Server (Jose Alvin Gendrano, Bruce Huang, and Wei Li), and UniSQL (Lincoln Turner and Carlos Ugarte); temporal join and coalescing: Access (Yuji Nishimura, Dhumil Sheth, and Lincoln Turner), IBM DB2 UDB (Jie Li and Kristin Tolle), Oracle8 Server (Jose Alvin Gendrano), and Sybase SQLServer (Sameer Verkhedkar and Xianjin Yang); tracking logs: IBM DB2 UDB (Helen Thomas and Giedrius Slivinskas), Microsoft SQL Server (Inés F. Vega-López and Giedrius Slivinskas), Sybase SQLServer (Yi-Jin Shi), Oracle8 Server (Scott Calvert and Wei Li), and UniSQL (Rachana Shah); transaction-time state tables: Microsoft SQL Server (Inés F. Vega-López and Giedrius Slivinskas) and Oracle8 Server (Scott Calvert and Wei Li); bitemporal tables: Oracle8 Server (Scott Calvert and Wei Li); the capstone case: Oracle8 Server (Wei Li); TimeDB: Andreas Steiner; Tiger: Michael Böhlen; Synchrony: John Bair, Hollis Chin, and Michael Soo; and the white papers: W. L. A. Derks, Heidi Gregersen, Christian S. Jensen, Leo Mark, Janne Skyt, and Jeroen Wijnands. Jian Yang superbly assembled all the files into a coherent and consistent whole. I also appreciate the support over the years from the National Science Foundation, recently via grants IRI-9632569 and ISI-9817798, and from AT&T Corporation, DuPont Corporation, IBM Corporation, and NCR Corporation, which enabled the research that provides the foundation for this book.

My editor, Diane Cerra, was wonderful throughout the involved process of writing and producing this book. I especially appreciate her constant quest for quality. I've written for several editors and publishers, and Diane and Morgan Kaufmann are by far the most author- and book-friendly.

The design of this book is considerably more complex than that of my other books. Edward Wade worked closely with me on this design. Although I would not have thought it possible, Edward cared as much as I did about getting the design just so, putting his heart into the project. The design feels *right*, and Edward deserves most of the credit. Edward, it was truly a joy to work with you.

Finally, I thank my wife, Merrie, and my two children, Eric and Melanie, for continually emphasizing by their example that life is so much more than book writing. They provided delightful distractions—how could I resist rambling through the park, or reading a poem by Shel Silverstein, or helping out on a pinewood derby car?

One of the oft-unexpected benefits of taking photographs, whether as a hobby or as a profession, is that you *see* more vividly. The veins within a fallen leaf become all-absorbing, and shadows on a building are suddenly profound and evocative. My experience in writing this book has been similar, in that the words I have been fortunate to encounter in my reading and listening have particularly