Preface

This is a coder's book. It's intended to help developers build applications that make use of Transact-SQL. It's not about database administration or design. It's not about end-user or GUI application development. It's not even about server or database performance tuning. It's about developing the best Transact-SQL code possible, regardless of the application.
When I began writing this book, I had these design goals in mind:

- Be very generous with code samples—don't just tell readers how to do something, show them.
- Include complete code samples within the chapter texts so that the book can be read through without requiring a computer or CD-ROM.
- Use modern coding techniques, with specific emphases on ANSI compliance and current version features and enhancements.
- Construct chapters so that they're self-contained—so that they rely as little as possible on objects created in other chapters.
- Provide real-world code samples that have intrinsic value apart from thebook.
- Avoid rehashing what's already covered extensively in the SQL Server Books Online.
- Highlight aspects of Transact-SQL that differentiate it from other SQL dialects; don't just write another ANSI SQL book.
- Avoid excessive screenshots and other types of filler mechanisms often seen in computer books.
- Proceed from the simple to the complex within each chapter and throughout the book.
- Provide an easygoing, relaxed commentary with a de-emphasis on formality. Be the reader's indulgent, amiable tutor. Attempt to communicate in writing the way that people speak.

You'll have to judge for yourself whether these goals have been met, but my hope is that, regardless of the degree of success, the effort will at least be evident.

About the Sample Databases

This book uses SQL Server's Northwind and pubs sample databases extensively. You'll nearly always be able to determine which database a particular example uses from the surrounding commentary or from the code itself. The pubs database is used more often than Northwind, so, when it's not otherwise specified or when in doubt, use pubs.
Usually, modifications to these databases are made within transactions so that they can be reversed; however, for safety's sake, you should probably drop and recreate them after each chapter in which they're modified. The scripts to rebuild them (instnwnd.sql and instpubs.sql) can be found in the \Install subdirectory under the root SQL Server folder.

Results Abridged

If I have a pet peeve about computer books, it's the shameless use of space-filling devices to lengthen them—the dirty little secret of the computer publishing industry. Many technical books these days overflow with gratuitous helpings of screenshots, charts, diagrams, outlines, sidebars, icons, line art, etc. There are people who assign more value to a book that's heavy, and many authors and publishers have been all too happy to accommodate them. They seem to take the old saying that "a picture is worth a thousand words" literally—in some cases turning out books that are little more than picture books.
I think there's a point at which comprehensiveness gives way to corpulence, a time when exhaustiveness becomes exhausting. In this book, I've tried to strike a balance between being thorough and being space-efficient. To that end, I've often truncated or clipped query result sets, especially those too wide to fit on a page and those of excessive length (I always point this out). On occasion I also list them using reduced font sizes. I don't include screenshots unless doing so benefits the discussion at hand materially (only one chapter contains *any* screenshots). This is in keeping with my design goal of being complete without being overwrought. Nearly 600SQL scripts are used in this book, and they are all included in the chapters that reference them. Hopefully none of the abridgements will detract from the book's overall usefulness or value.

On Formality

Another of my pet peeves is formality for the sake of formality. An artist once observed that "it's harder to draw a good curved line than a straight one." What he meant was that it's in some ways more difficult to do something well for which there is no exact or stringent standard than to do something that's governed by explicit rules and stuffy precedents. All you have to do to draw a straight line is pick up a straightedge. The rules that govern formal writing, particularly that of the academic variety, make writing certain kinds of books easier because they convert much of the subjective nature of writing into something more objective. They're like training wheels on the would-be author's bicycle. Writing goes from being a creative process to a mechanical one. Cross all the T's, dot all the I's, and you're halfway there. Obviously, this relieves the author of many of the decisions that shape creative writing. It also turns otherwise good pieces of work into dreary, textbook-like dissertations that are about as interesting as the telephone book White Pages.
So, I reject the notion that formal writing is better writing, that it is a higher standard and is the ideal for which all technical writers should strive. Instead, I come from the Mark Twain school of thought—I "eschew surplusage"—and I believe that, so long as common methods of speech do not become overly banal (a subjective distinction, I freely admit), the ultimate goal of the technical writer should be to write the way that readers speak. It is the way people—even technical people—are most accustomed to communicating and the way they are the most able to learn and share ideas. I did not invent this way of thinking; it's simply the way most of my favorite authors—Mark Twain, Dean Koontz, Joe Celko, Ernest Hemingway, Robert Heinlein, Andrew Miller, Oscar Wilde, P.J. O'Rourke, Patricia O'Connor—write. Though it is far more difficult to structure and write a narrative that flows naturally and reads easily, it's worth the effort if the ideas the writer seeks to convey are understood as they were intended.
So, throughout this book, you'll see a number of the rules and pseudo rules of formal writing stretched, skirted, bent, and sometimes outright broken. This is intentional. Sometimes I split infinitives, begin sentences with conjunctions, and end them with prepositions.[1] Sometimes *record* is used interchangeably with *row;* sometimes *field* takes the place of *column;* and I never, ever treat *data* as a plural word. I saw some software recently that displayed a message to the effect "the data are being loaded," and I literally laughed out loud. The distinction between the plural *data* and its obscure singular form *datum* is not maintained in spoken language and hasn't really ever been (except, perhaps, in ancient Rome). It has also been deprecated by numerous writing guides [2] and many authors[3] You will have to look very hard for an author who treats *data*as a plural word (I can think of only one off the top of my head, the irascible Ted Codd). The tendency for technical communication to become self-important or ostentatious has always bafed me: why stoop to pretension? Why trade the uid conveyance of ideas between people for nonsense that confuses some and reads like petty one-upmanship to others?

[1] According to Patricia T. O'Connor's excellent book, *Words Fail Me* (Harcourt Brace & Company, 1999), a number of these rules are not really rules at all. The commonly cited prohibitions against split infinitives, beginning sentences with conjunctions, using contractions, and ending sentences with prepositions are all pseudo rules—they are not, nor have ever been, true English grammatical rules. They originate from dubious attmepts to force Latin grammar on the English language and have been broken and regularly ignored by writers since the 1300s.

[2] See, for example, *The Microsoft Manual of Style for Technical Publications* (Microsoft Press, 1995), p.48.

[3] See, for example, *Joe Celko's Data and Databases: Concepts in Practice* (Morgan-Kaufmann Publishers, 1999), p.3, where Joe refers to data in the singular as he does throughout the book.

Acknowledgments

I'd like to thank my wife, who not only makes it possible for me to write books but also makes it worthwhile. The book you see before you is as much hers as it is mine. I'd like to thank Neil Coy, who made a real programmer of me many years ago. Under Neil's tutelage, I learned software craftsmanship from a master. Joe Celko, the dean of the SQL language, has been a good friend and a valuable source of information throughout this project. Kudos to John Sarapata and Thomas Holaday for helping me come up with a title for the book (I'll keep *Sybase for Dummies* in mind for future use, John). Thanks to the book's technical reviewers, particularly Wayne Snyder, Gianluca Hotz, Paul Olivieri, and Ron Talmage. Heartfelt thanks to John Gmuender, Joe Gallagher, Mike Massing, and Danny Thorpe for their equanimity and for keeping me sane through the recent storm. Congratulations and genuine appreciation to the superb team at Addison-Wesley—Michael Slaughter, Marisa Meltzer, J. Carter Shanklin, and others too numerous to list. Special thanks to Nancy Cara-Sager, a friend, technical reviewer, and copyeditor who's been with me through several books and a couple of publishers now. Her tireless attention to detail has saved me from embarrassing myself more times than I can count.