TURING 图灵原版计算机科学系列

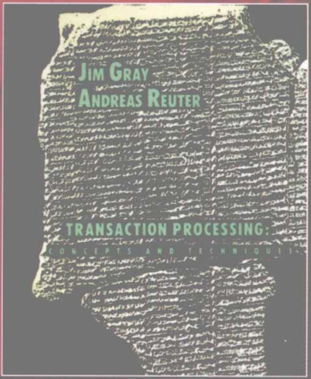
ELSEVIER 爱思唯尔

Transaction Processing
Concepts and Techniques

1998年图灵奖得主著作
事务处理圣经

事务处理
概念与技术
(英文版)

[美] Jim Gray
Andreas Reuter 著



人民邮电出版社
POSTS & TELECOM PRESS

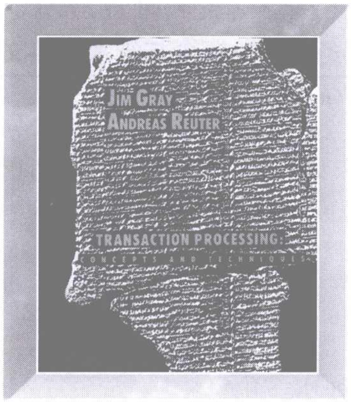
TURING 图灵原版计算机科学系列

Transaction Processing
Concepts and Techniques

事务处理
概念与技术

（英文版）

[美] Jim Gray
Andreas Reuter 著



人 民 邮 电 出 版 社
北 京

图书在版编目（CIP）数据

事务处理：概念与技术＝Transaction Processing: Concepts and Techniques: First Edition: 英文/（美）格雷（Gray, J.），（美）路透（Reuter, A.）著. —北京：人民邮电出版社，2009.5

（图灵原版计算机科学系列）

ISBN 978-7-115-19586-9

I. 事… II. ①格… ②路… III. 事务计算程序－英文 IV. TP319

中国版本图书馆CIP数据核字（2008）第211006号

内 容 提 要

本书从系统的角度全面阐述事务处理的概念和技术，其中涉及终端上的表示管理、通信子系统、操作系统、数据库、程序设计语言的运行时系统以及应用开发环境等。本书重点放在事务处理的基本概念上，主要阐述事务概念是如何用于解决分布式系统问题的，以及利用这些概念如何能够在有限的资金和风险范围内建立高性能、高可用性的应用系统。全书重点讲述了事务处理基础、容错基础知识、面向事务的计算、并发控制、恢复、事务型文件系统、系统概览等7个主题，介绍了事务的ACID特性、并发的理论和实践、事务管理和恢复技术等方面的内容，最后还介绍了一个非常重要的资源管理器的实现。

本书主要面向计算机及相关专业的高年级本科生和研究生，适合作为事务处理导论、数据库系统、分布式系统、操作系统等课程的辅助教材，需要了解事务处理系统的开发人员也可将其作为基本参考书。

图灵原版计算机科学系列

事务处理：概念与技术（英文版）

- ◆ 著　[美] Jim Gray　Andreas Reuter
  责任编辑　杨海玲
- ◆ 人民邮电出版社出版发行　北京市崇文区夕照寺街14号
  邮编　100061　电子函件　315@ptpress.com.cn
  网址　http://www.ptpress.com.cn
  北京鑫正大印刷有限公司印刷
- ◆ 开本：800 × 1000　1/16
  印张：69
  字数：1 327 千字　2009年 5 月第 1 版
  印数：1- 2 000 册　2009年 5 月北京第 1 次印刷

著作权合同登记号　图字：01-2008-5836号

ISBN 978-7-115-19586-9/TP

定价：138.00元

读者服务热线：（010）88593802　印装质量热线：（010）67129223

反盗版热线：（010）67171154

版权声明

Elsevier (Singapore) Pte Ltd.
3 Killiney Road
08-01 Winsland House I
Singapore 239519
Tel: (65)6349-0200
Fax: (65)6733-1817

First Published 2009
2009年初版

Foreword

Bruce Lindsay
IBM Almaden Research Center
San Jose, California

Commercial, governmental, scientific, and cultural activities are becoming increasingly dependent on computer-based information resources. As increasing amounts and varieties of information are captured and maintained by computer systems, the techniques for exploiting, managing, and protecting this information become critical to the well-being, and indeed the very survival, of modern industrialized societies.

Transaction processing technology is the key to the coherent management and reliable exploitation of computer-based information resources.

Transaction processing encompasses techniques for managing both the stored information itself and the application programs which interpret and manipulate that information. From database recovery and concurrency control to transaction monitors that initiate and control the execution of applications, transaction processing technology provides mechanisms and facilities needed to protect and manage the critical information resources which underlie many (if not most) commercial, scientific, and cultural activities.

In order for the increasingly vast quantity of computer-based information to be useful, it must accurately reflect the real world and be available to the application programs that interpret and exploit the information. In general, exploiting stored information involves accessing and modifying related data items which *collectively* describe or model the status and evolution of real-world phenomena and activities. Because multiple data items must usually be accessed or updated together to correctly reflect the real world, great care must be taken to keep related data items mutually consistent. Any interruption of updates to related items, or the interleaving of updates and accesses, can make the data inconsistent.

The key to maintaining data consistency is to identify the sequences of accesses and updates that represent the interrogation and modification of related data items. Such sequences are called *transactions*. Transaction processing technology ensures that each transaction is either executed to completion or not at all, and that concurrently executed transactions behave as though each transaction executes in isolation. The significance of this technology is magnified by the fact that these guarantees are upheld despite failures of computer components, distribution of data across multiple computers, and overlapped or parallel execution of different transactions.

Twenty-five years of intense effort in commercial and university laboratories has led to transaction processing technologies capable of all-or-nothing execution and isolation of concurrent transactions. This book presents, for the first time, a comprehensive description of the techniques and methods used by transaction processing systems to control and protect the valuable information resources they manage. The authors describe, in detail, the state-of-the-art techniques used in the best-of-breed commercial and experimental transaction processing systems. They concentrate on proven techniques that are effective and efficient. Detailed explanations of *why* various problems must be solved and *how* they can be addressed make this book useful to both the serious student and to system developers.

The authors, Dr. James Gray and Professor Andreas Reuter, have between them five decades of direct experience with the implementations of both commercial and experimental transaction processing systems. They have both made significant contributions to the art of transaction processing and are famous for their scientific publications and their ability to explain the fruits of their research. This book is a distillation of their deep understanding of transaction processing issues and their hard-won appreciation of the most effective techniques for implementing transaction processing methods. The authors' ability to distinguish between fundamental concepts and speculative approaches gives the reader a firm and practical basis for understanding the issues and techniques of transaction processing systems.

This book covers all aspects of transaction processing technology. The introductory chapters give the reader a basic understanding of transaction concepts and the computing environment in which transactions must execute, including important assumptions about the component failures which the transaction processing system must tolerate. The explanation of the role of transaction processing monitors, which control the activation and execution of application programs and the facilities they provide, sets the stage for the presentation of concurrency control and recovery techniques. The discussion of transaction isolation covers concurrency control issues from the hardware level to the isolation semantics for records and indexes. The important and complex technology of transaction recovery in the face of failures is covered in great detail. From record management to distributed commit protocols, the recovery techniques needed to ensure all-or-nothing execution and data persistence are presented and explained. The transaction recovery and isolation techniques are then applied to the design and implementation of record-oriented storage and associative indexes. Students and developers of database systems will find much useful information in these chapters. The book concludes with a survey of transaction processing systems from both the commercial and the academic worlds.

Throughout the book, the authors provide in-depth discussions of the underlying issues, and detailed descriptions of proven techniques. The concepts are illustrated with numerous carefully designed figures. In addition, the techniques are accompanied by code fragments that increase the reader's appreciation of the implementation issues.

Transaction Processing: Concepts and Techniques is both comprehensive in its coverage of transaction processing technology and detailed in its descriptions of the issues and algorithms. In-depth presentations of the techniques are enlightening to the student and a resource for the professional. The importance of transaction processing technology to the information management needs of industrialized societies makes it essential that this technology be well understood and widely applied. This book will serve as a guide and a reference for the many individuals who will apply and extend transaction processing concepts and techniques in the years ahead.

Buying books would be great if we could also buy the time to read them.

ARTHUR SCHOPENHAUER: PARERGA UND PARALIPOMENA

Preface

Why We Wrote this Book

The purpose of this book is to give you an understanding of how large, distributed, heterogeneous computer systems can be made to work reliably. In contrast to the often complex methods of distributed computing, it presents a distributed system application development approach that can be used by mere mortals. Why then doesn't the title use a term like *distributed systems*, *high reliability*, *interoperability*, or *client-server*? Why use something as prosaic as *transaction processing*, a term that for many people denotes old-fashioned, batch-oriented, mainframe-minded data processing?

The point is—and that's what makes the book so long—that the design, implementation, and operation of large application systems, with thousands of terminals, employing hundreds of computers, providing service with absolutely no downtime, cannot be done from a single perspective. An integrated (and integrating) perspective and methodology is needed to approach the distributed systems problem. Our goal is to demonstrate that transactions provide this integrative conceptual framework, and that distributed transaction-oriented operating systems are the enabling technology. The client-server paradigm provides a good way of structuring the system and of developing applications, but it still needs transactions to control the client-server interactions. In a nutshell: without transactions, distributed systems cannot be made to work for typical real-life applications.

This is not an outrageous claim; rather it is a lesson many people—system implementors, system owners, and application developers—have learned the hard way. Of course, the concepts for building large systems have been evolving for a long time. In fact, some of the key ideas were developed way back when batch processing was in full swing, but they are far from being obsolete. Transaction processing concepts were conceived to master the complexity in single-processor online applications. If anything, these concepts are even more critical now for the successful implementation of massively distributed systems that work and fail in much more complex ways. This book shows how transaction concepts apply to distributed systems and how they allow us to build high-performance, high-availability applications with finite budgets and risks. We've tried to provide a sense of this development by discussing some of the "lessons from history" with which we're familiar. Many of these demonstrate the problems that transactions help to avoid, as well as where they hide the complexity of distributed systems.

There are many books on database systems, both conventional and distributed; on operating systems; on computer communications; on application development—you name it. The partitioning of interests manifest in these terms has become deeply rooted in the syllabi of computer science education all over the world. Education and expertise are organized and compartmentalized. The available books typically take their readers through the ideas in the technical literature over the past decades in an enumerative style. Such presentations offer many options and alternatives, but rarely give a sense of which are the good ideas and which are the not-so-good ones, and why. More specifically, were you ever to design or build a real system, these algorithm overviews would rarely tell you how or where to start.

Our intent is to help you solve real problems. The book is focused in that we present only one or two viable approaches to problems, together with explanations; but there are many other proposals which we do not mention, and the presentation is not encyclopedic. However, the presentation is broad in the sense that it presents transaction processing from a systems perspective. To make large systems work, one must adopt a genuine end-to-end attitude, starting at the point where a request comes in, going through all the system layers and components, and not letting go until the final result is safely delivered. This necessarily involves presentation management in the terminals, the communication subsystem, the operating system, the database, the programming language run-time systems, and the application development environment. Designing a system with that sort of integration in mind requires an altogether different set of criteria than designing an algorithm within the narrow confines of its functional specification. This holistic approach is not one that we've found in other presentations of distributed systems and databases. However, since the beginning of this project in 1986, we've been convinced that such an approach is necessary.

Topic Selection and Organization

Since the end-to-end perspective forced us to cover lots of ground, we focused on the basic ideas of transaction processing: simple TP-systems structuring issues, simple transaction models, simple locking, simple logging, simple recovery, and so on. When we looked at the texts and reference books available, we noticed that they are vague on the basics. For example, we haven't found an actual implementation of B-trees in any textbook. Given that B-trees are *the* access path structure in databases, file systems, information retrieval systems, and who knows where, that is really basic. This presentation is like a compiler course textbook or like Tanenbaum's operating system book. It is full of code fragments showing the basic algorithms and data structures.

The book is pragmatic, covering basic transaction issues in considerable detail. Writing the book has convinced us that this is a good approach, but the presentation and style may seem foreign. Our motive is to document this more pragmatic perspective. There is no theory of structuring complex systems; rather, the key decisions depend on educated judgment and adherence to good engineering principles—pragmatic criteria. We believe that these principles, derived from the basic concept of transaction-oriented processing, will be important for many years to come.

Looking at the table of contents, you will find a forbidding number of chapters (16), but they are organized into seven subject areas, which can be read more or less independently, and in different order.

The first topic is an overview of transaction processing in general (Chapter 1). It presents a global system view. It introduces the basic transaction properties: *atomicity* (all-or-nothing), *consistency* (a correct transformation of state), *isolation* (without concurrency anomalies), and *durability* (committed changes survive any system failures)—ACID, for short. The nontechnical reader can end there and still gain a broad understanding of the field.

Chapter 2 is meant for readers vaguely familiar with the basic terminology of computer science. The chapter introduces the most important terms and concepts of hardware, software, protocol standards, and so on—all the terminology needed for the technical discussions later in the book.

Chapter 3 explains why systems fail and gives design advice on how to avoid such failures. It reviews hardware and software fault-tolerance concepts and techniques (fail-fast, redundancy, modularization, and repair). If you want to learn how to build a module with a mean-time-to-failure of 10,000 years, using normal, faulty, off-the-shelf components, this is where to look. Chapter 3 explains the significance of transactions in building highly available software.

Chapters 4 through 6 present the theory and use of transactions. Chapter 4 gives a detailed discussion of what it means to structure applications as transactions. Particular attention is paid to types of computations not well-supported by current flat transactions. These applications require extension and generalization of the transaction concept. Chapter 5 explains what transaction-oriented computation means for the operating system and other low-level components by describing the role of the TP monitor. It also explains how a transaction program interacts with the system services. Chapter 6 is for programmers. It contains lots of control blocks and code fragments to explain how transactional remote procedure calls work, how request scheduling is done, and other such subtleties. Readers not interested in bit-level events should skim the first half of that chapter and start reading at Section 6.4 about transactional queues.

Chapters 7 and 8 present the theory and practice of concurrency. Transaction processing systems hide all aspects of parallel execution from the application, thereby giving the *isolation* of the ACID properties. Chapter 7 presents the theory behind these techniques, and Chapter 8 demonstrates how to implement this theory with locking.

Chapters 9 through 12 present transaction management and recovery, that is, everything related to making transactions atomic and durable. Chapter 9 explains how logging and archiving are done. Chapter 10 explains how to write a transactional resource manger like a database system or a queue manager. It explains how the resource manger joins the transaction, how it writes log records, gets locks, and participates in transaction commit or rollback. A simple resource manger (the one-bit resource manger) is used to demonstrate the techniques. Chapter 11 takes the transaction manager perspective; it must always reliably know the state of a transaction and which resource managers are working on the transaction. The implementation of a simple transaction manager is laid out in Chapter 11—again, something to be skipped by those who do not need to know all the secrets. Chapter 12 is a catalog of advanced concepts and techniques used by transaction managers.

Chapters 13 through 15 deal with yet another self-contained topic: the implementation of a very important resource manager, a transactional file system. Starting from the bare metal (disks), space management is sketched, and the role of the buffer manager in the system is explained in some detail. Next comes the abstraction of varying-length tuples from fixed-length pages, and all the file organizations that support tuple-oriented access. Finally, Chapter 15 talks about associative access, focusing primarily on B-trees and how to implement them in a highly parallel environment. All this is done in a way that provides the ACID properties: the resulting files, tuples, and access paths are transactional objects.

Chapter 16 gives an overview of many commercially available systems in the transaction processing arena. We have tried to highlight the features of each system. This is not a competitive comparison; rather it is a positive description of the strengths of each system.

At the end, there is an extended glossary of transaction processing terminology. Having such a large glossary is a bit unusual in a textbook like this. However, as the motto of Chapter 5 indicates, terminology in the field of transaction processing is all but well-established. So the glossary has to serve two purposes: First, if you are uncertain about a term, you can look it up and find our interpretation. Second, by being used this way, the glossary might actually help to promote a more homogeneous terminology in the field.

Basics of Transaction Processing	1. Book Overview 2. Basic Terminology
Basics of Fault Tolerance	3. Faults and Fault Tolerance
Transaction-Oriented Computing	4. Transaction Models 5. Transactional Programming Styles 6. How a TP Monitor Works
Concurrency Control	7. Concurrency Theory (serializability) 8. Concurrency Control (locking)
Recovery	9. Logging 10. Transactional Resource Managers 11. Transaction Manager 12. Advanced TM techniques
A Sample Resource Manager	13. Implementing File and Pages 14. Implementing Record and Fields 15. Implementing Access Paths
System Surveys	16. Overviews of Popular TP Systems

Chapters 1 through 15 contain historical notes that explain how things were developed, where certain ideas appeared first, and so on. In contrast to many other areas in computer science, transaction processing developed primarily in industrial research and development labs. Some ideas conceived and implemented in commercial products were rediscovered years later and published as scientific results. In the historical notes we try give credit to both contributions, to the best of our knowledge.

Most chapters also have exercises ranging from short refreshers to term projects. The answers to most of them are provided at the end of each chapter. Following Donald Knuth, each exercise has a qualifier of the form [section, level]. The exercise applies to the material in the section indicated. The level is an indicator of how difficult the exercise is:

[10]	One minute (just to check whether you have followed the text)
[20]	15–20 minutes for full answer
[25]	One hour for full written answer
[30]	Short (programming) project: less than one full day of work
[40]	Significant (programming) project: two weeks of elapsed time

In addition, we have used the ratings *[project]* for anything that is likely to be higher than [40], and *[discussion]* for exercises that ask readers to go out and explore a certain subject on their own.

The original plan for the book was to include SQL implementation and application design along with transactions. As the work progressed, we realized that there was not space or time for these topics. So, as it stands, this book is about how to implement transactions. Ceri and Pelagatti's excellent book [1984] covers the high-level database issues (SQL, normalization, optimization, and also some transaction management issues).

Learning with this Book

The book is intended for advanced undergraduates, graduate students, and for professional programmers who either want to understand what their transaction processing system is doing to them (e.g., a CICS/DB2 user) or who need a basic reference text. It assumes that you have a reading knowledge of SQL and C.

The content of this book has no exact counterpart among computer science classes at a university. The compartmentalization of subjects is inherent in the structure of the curriculum, and as yet there is no standard class on transaction processing. However, the book has already been used in a variety of undergraduate and graduate courses during our various stages of draft manuscript. We feel the approach we have taken is appropriate within the existing structure of computer science, but we hope that exposure to our approach will help to eliminate compartmentalized thinking. Here are some suggestions for emphasizing different aspects of the coverage:

Just getting the idea: Chapter 1, Sections 2.7, 4.1–4.2, 5.1–5.3, 5.5–5.7, and Chapter 16.

An introduction to transaction processing: Chapters 1 and 2, Sections 3.1–3.6, Chapters 4 and 5, Sections 6.4–6.5, 7.1–7.6, Chapters 9, 10, 16.

Database systems: Chapters 1, 4, 7, 10, 13, 14, 15.

Distributed systems: Chapters 1, 2, 3, 4, 7, 8, 10, 11, 12, 16.

Operating systems: Chapters 1, 2, 3, 5, 6, 7, 8, 10, 11, 13, 16.

Advanced coverage: Read everything. In advanced courses, Chapters 1 and 3 can be skipped or done away with quickly. In addition, one could use the books by Tanenbaum on operating systems and computer networks, by Ceri and Pelagatti on distributed databases, by Oszu and Valduriez on principles of distributed databases, and one of Date's books on databases in general.

Many other combinations are conceivable. Those who are already familiar with the subject can skip Chapters 1 and 2; we recommend browsing them, just to grasp the terminology used throughout the book.

Chapter 3 can be skipped, but we recommend you read it. Transactions may seem all too obvious at the first glance. However, Chapter 3 carefully explains why transactions are the right exception-handling model and are a key to building highly available systems. The techniques described therein help provide a better understanding of what fault tolerance at the system level means, and they can be put to use almost immediately for anyone building applications.

We sketched the few dependencies that exist among the chapters while introducing the subject areas. If you read the entire book, you will notice some redundancy among the chapters. This feature allows chapters to be used independently.

To enhance course use of this book, instructors might consider using software available from Transarc Corporation through the Encina University Program. The Encina products are designed to enable distributed, standards-based online transaction processing in open computing environments. Many of the topics in this book could be explored through course projects using the Encina family of modular products: for example, distributed transaction management, transactional remote procedure calls, advanced lock and recovery models, shared logging systems, a record-oriented resource manager, and a transaction monitor. Contact information about the Encina family of products and academic site licences can be found on the copyright page of this book.

If you have time enough, and if you are thoroughly interested in transaction processing, get copies of papers by Bjork and Davies [1972; 1973; 1978]. These early articles started the whole field, and it is instructive to read them from our current perspective. One appreciates that transactions evolved not so much as a natural abstraction, but as a radical attempt to cut out complexity that otherwise proved to be unmanageable. When reading these seminal papers, you will also find that the vision outlined there has not yet become reality—not by a wide margin.

Concluding Remarks

As the acknowledgments indicate, drafts of this book were reviewed and class-tested for two years (1990–1991). Not only has this improved the presentation and accuracy of the text, it also changed the overall design of the book. New chapters were added, others were dropped, and the book doubled in size. The review process and heated debates also changed the way the material is organized. One of us started top-down, whereas the other one wrote bottom-up. In the end, we both reversed our approaches.

Apart from that, errors remain errors, and they must be blamed on us with no excuse. We will be very grateful, though, if you let us know the ones you find. Please send such comments to the authors in care of the publisher, or to the electronic mailbox Gray@microsoft.com.

Acknowledgments

This book benefited from the detailed criticism and advice of many people. We gratefully acknowledge their guidance. We are *especially* grateful to Frank Bamberger, Phil Bernstein, Noel Herbst, Bruce Lindsay, Dave Lomet, Dan Siewiorek, Nandit Soparkar, Kent Treiber, and Laurel Wentworth for detailed advice on improving the emphasis or focus of the book. More than anything else, the criticism of Betty Salzberg and her classes shaped the presentation of this book. Thank you all.

Phil Bernstein at Digital
Frank Bamberger at Citibank
Edward Brajinski at Digital
Mike Carey at Wisconsin
Stefano Ceri at Milano
Edward Cheng at Digital
Joe Coffee at Northeastern
Mike Cox at Princeton
Flaviu Cristian at IBM
Gary Cuhna at Northeastern
Walker Cunningham at Morgan Kaufmann
Allyn Dimock at Northeastern
Robert Drum at Northeastern
Dan Duchamp at Columbia
Amanda Carlson at HP
Jeff Eppinger at Transarc
Georgios Evangelidis at Northeastern
Scott Fitzgerald at Northeastern
Hector Garcia-Molina at Princeton
Amal Gebrael at Northeastern
Yujia Haung at Northeastern
Ricky Ho at G.O. Graphics
Walter Hursch at Northeastern
Meichun Hsu at Digital
Pat Helland at HAL
Noel Herbst at Tandem
Pete Homan at Tandem
Olaf Kruger at Northeastern
T.J. Jafar at Northeastern
Jim Johnson at Digital
Johannes Klein at Digital
Angelica Kokkinaki at Northeastern
Jacek Kruszelnicki at Northeastern
Olaf Kruger at Northeastern
Lori Ann Lashway at Polaroid
Don Langenhorst at Northeastern
Bruce Lindsay at IBM
Dave Lomet at Digital
Randell MacBlane at USL
Dorthy Minior at GSSI
Elliot Moss at U. Massachusetts
Ron Obermarck at Digital
Jeffry Picciotto at Mitre
Franco Putzolu at Tandem
Xinyang Qian at Northeastern
Ron Regan at Northeastern
Dennis Roberson at Digital
Eleana Rosenzweig at Northeastern
Daphne Ryan at Northeastern
Betty Salzberg at Northeastern
Tom Sawyer at Tom Sawyer
Fabio Schreiber at Milan
Linda Seiter at Northeastern
Mark Sherman at Transarc
Dan Siewiorek at CMU
Eric Skov at Northeastern
Donald Slutz at Tandem
Mark Smith at U. Massachusetts
Narjit Sindupal at ATT
Nandit Soparkar at U. Texas
Christos Stamelos at Northeastern

Kent Treiber at IBM
Mike Ubell at Digital
Tom Vancor at BGS
Vic Vyssotsky at Digital
Laurel Wentworth at Digital
Gary Warner at Northeastern
Bill Wisnaskas at Northeastern
David Wong at Northeastern
Robert Wu at Digital
Hans-Jörg Zeller at Tandem
Ying Zhou at Northeastern

Walker Cunningham, our developmental writer, carefully criticized all the chapters. His efforts made an enormous difference in the clarity of the book.

Our colleagues heavily influenced certain chapters. Betty Salzberg suggested the need for Chapter 2 (terminology). Dan Siewiorek and Vic Vyssotsky contributed heavily to the chapter on fault tolerance. Bruce Lindsay and Harald Sammer influenced our presentation of transaction concepts. Phil Bernstein, Noel Herbst, Bruce Lindsay, and Ron Obermarck caused us to rethink our presentation of transaction management. Dave Lomet and Franco Putzolu contributed heavily to the chapters on record and file management. Frank Bamberger, Elliot Moss, Don Slutz, and Nandit Sopakar gave us valuable overall criticism of the book.

Charlie Davies, the visionary who launched our field 20 years ago, deserves mention here. Charlie's work on *spheres of control* is little known. His papers are obscure, but all who came in contact with him were inspired by his vision. Workers in the field are still trying to work out the details of that vision. Chapter 4 presents his ideas in more modern terms.

Transaction processing is a field in which practice has led theory. Commercial systems often implement ideas long before the ideas appear in an academic setting. One is reminded of Galileo claiming the invention of the telescope to the doges of Venice while merchants were selling mass-produced Dutch telescopes in the streets below.

Throughout the book, we are faced with a dilemma. Does one give credit to the first to publish an idea? Or, does one give priority to the earlier development and implementation of the idea in a product? Academic tradition and U.S. patent law give priority to the first to publish. We have tried to credit both. The historical notes in each chapter recount, as best we can, the parallel commercial and academic development of the ideas. Perhaps the point is moot; implementors don't want credit for ideas, they want the cash.

One implementor, Franco Putzolu, has deeply influenced our field in general, and the authors in particular. Franco has never written a paper or a patent, but his ideas and code are at the core of System R, SQL/DS, DB2, Allbase, and NonStop SQL. These designs have been widely copied by others and are repeated here. Franco Putzolu deserves much of the credit for this book.

Bruce Spatz, our publisher, gave us excellent guidance on the focus of the book. He arranged for many reviews, arranged the classroom tests, and encouraged us when we needed it most. We also are very grateful to our project manager, Jennifer Ballentine of Professional Book Center, and Jeanne Thieme, our copyeditor, for their contributions to this book.

Andreas's students contributed to the book in many ways: They read draft versions, used it for teaching, tried the exercises, worked on the references, and so on—typical student slave labor. Moreover, the absence of their supervisor when he periodically got serious about "finishing that book *now*" meant they had to work on their own much more than one would wish.

Gabriele Ziegler, Andreas's secretary, deserves special thanks for keeping him organized during the whole endeavor—not an easy assignment. She had to pacify many people who had good reason to be angry for his not responding for days or weeks when he was again "finishing that book *now*."

Christiane Reuter almost wrote a chapter on application design, but decided against it. However, she took care of the authors during the first long writing assignment in Ripa—not a simple task given the quality of the local power and the dreadful weather. She also accepted the multiyear ordeal of late nights and lost weekends.

Tandem Computers and Digital Equipment Corporation were very generous in their support of Jim Gray's efforts on this book.

Thanks to all of you who kept us going, even kept us amused, and tolerated our obsessions and varying tempers.

Jim Gray

Andreas Reuter
University of Stuttgart

Trademarks

PART ONE

The Basics of Transaction Processing