Principles of Transaction Processing

Principles of Transaction Processing

Second Edition

Philip A. Bernstein

Eric Newcomer

ELSEVIER

AMSTERDAM • BOSTON • HEIDELBERG • LONDON
NEW YORK • OXFORD • PARIS • SAN DIEGO
SAN FRANCISCO • SINGAPORE • SYDNEY • TOKYO
Morgan Kaufmann Publishers is an imprint of Elsevier

MORGAN KAUFMANN

Morgan Kaufmann Publishers is an imprint of Elsevier

30 Corporate Drive, Suite 400, Burlington, MA 01803, USA

Library of Congress Cataloging-in-Publication Data
Bernstein, Philip A.
Principles of transaction processing/Philip A. Bernstein, Eric Newcomer.—2nd ed.
p. cm.—(The Morgan Kaufmann series in data management systems)
Includes bibliographical references and index.
ISBN 978-1-55860-623-4 (pbk.)
1. Transaction systems (Computer systems) I. Newcomer, Eric. II. Title.
QA76.545.B47 2009
005.74′5–dc22

2009003605

Printed in the United States of America
09 10 11 12 13   5 4 3 2 1

For Jim Gray, wherever he may be

Contents

Preface

WHY READ THIS BOOK?

Transaction processing has been an important software technology for 40 years. Large enterprises in transportation, finance, retail, telecommunications, manufacturing, government, and the military are utterly dependent on transaction processing applications for electronic reservations, banking, stock exchanges, order processing, music and video services, shipment tracking, government services, telephone switching, inventory control, and command and control. Many large hardware and software vendors receive much of their revenue from components of transaction processing systems, such as IBM, HP, Oracle, Microsoft, Dell, Red Hat, and EMC. The market for transaction processing products and services is many tens of billions of dollars per year. As consumers, we all use this technology every day to withdraw cash, buy gas, rent movies, and make purchases on the Internet.

How exactly do these transaction processing systems work? This question was once of interest only to computer professionals in the commercial data processing field. Now, given the widespread use of transaction processing in today's economy, it is of interest to a much broader engineering audience. Despite this interest, there is little written for a system professional to get a readable, technically solid introduction to this complex technology. This book fills the gap.

The software environment of most large-scale transaction processing systems is based on transactional middleware, which helps knit together many software components. These components include front-end applications to drive web browsers and other devices, middle-tier applications to route requests to the server that can run them, and server applications that execute business logic. Examples of transactional middleware include IBM's CICS; Microsoft's .NET Enterprise Services; and Java Enterprise Edition products, such as IBM WebSphere Application Server, Oracle's WebLogic Server, and Red Hat's JBoss Application Server. The first half of this book focuses on transactional middleware technology.

For many software engineers, transactional middleware is obscure technology—strange software glue that seems to be needed beyond operating systems, database systems, communication systems, and application programming languages. This book demystifies transactional middleware by explaining how it contributes to the performance, security, scalability, availability, manageability, and ease-of-use of transaction processing systems. The first half of the book explains transactional middleware outside and in—the features it offers to application programmers and how it is constructed to offer these features.

The transaction abstraction itself is largely implemented by database systems. They ensure that each transaction executes in its entirety, is isolated from interference by other transactions, and generates results that will survive hardware and software failures. This behavior is implemented by locking, logging, communication protocols, and replication. These technologies are the subject of the second half of this book.

This book is an introduction to transaction processing, intended to meet the needs of a broad audience, including:

- Application programmers with an interest in building transaction processing applications
- Database administrators who manage database systems used for transaction processing
- Application analysts who design applications for deployment on transaction processing systems
- Product developers in related areas, such as database systems, operating systems, and communications

- Marketing and technical support engineers for both systems and application products
- Computer science undergraduates and graduate students looking for an introduction to this topic

Our focus is on the principles of transaction processing, not on a prescription for how to build a transaction processing application—"how come?" not "how to." We include examples from many products, to illustrate how the principles have been applied and where ideas originated. But we do not dwell heavily on any one product. We present technology that is practical and used in products and pay only modest attention to good ideas that are not commonly used in practice.

We do not assume any special prerequisites, other than "system sophistication." We expect most readers will have some familiarity with SQL and database systems, but this background isn't necessary.

After finishing the book, you will understand how transactional middleware works and when to use it, and how transactional middleware and database systems work together to support reliable distributed transaction processing applications. You will be able to learn quickly how to use any transactional middleware product or database system to support the development and management of transaction processing applications.

WHAT'S NEW IN THIS SECOND EDITION?

The short answer is "a lot." There are several new chapters and rewritten chapters, and many new and revised sections of the rest.

Two main goals drove these changes. Our first goal was to present the new and revised transaction architectures and technologies that have appeared since we published the first edition twelve years ago. Back then, Internet-based electronic commerce was just beginning. Now, it is established as a major segment of many business-to-consumer and business-to-business markets. The growth of this segment, along with the commoditization of server hardware and operating systems, has led to major changes in transaction processing products. Web browsers are now a dominant technology for interacting with transaction processing systems. Transactional middleware has evolved from on-line transaction processing monitors to many new product categories that are designed to work well over the Internet, such as application servers, object request brokers, message-oriented middleware, and workflow systems. Object-oriented programming and service-oriented architecture have become mainstream. And database systems have become more complete transaction processing environments. These changes are all reflected in this second edition.

Our second main goal was to add coverage and depth of classical transaction processing topics, to make the book more complete. In part, this is based on the first author's experience in using the book as a textbook for a graduate computer science course for professional masters' students at the University of Washington. It is also in response to technological improvements, where formerly exotic technologies are now widely used.

Concretely, the major changes are as follows: The three chapters on transactional middleware have been entirely rewritten—two on principles and a long one on example products and standards, including details of Java Enterprise Edition and Microsoft .NET. There is a new chapter on business process management. The chapter on locking has new sections on optimistic concurrency control, B-tree locking, multigranularity locking, and nested transactions. There are new sections on the TPC-E benchmark, state management, scalability, shadow-paging, data sharing systems, consensus algorithms, log-based replication, and multimaster replication. Concepts of service-oriented architecture (SOA), REST, and Web Services are sprinkled throughout the book. There are numerous smaller additions of technical detail in many sections. Significant changes can be found in every chapter.

Supplementary material will be available on the publisher's web page for this book. Initially, it will include a selection of problems, grouped by chapter. We will add other technical material over time.

SUMMARY OF TOPICS

The enterprise that pays for a transaction processing system wants it to give fast service, be inexpensive to buy and operate, and be scalable as usage grows and new applications are added. Application programmers want to be insulated from the complexity of the many different kinds of technologies required to run a transaction processing system, such as transaction protocols, message protocols, transactional remote procedure calls, persistent queues, multithreaded processes, resource pooling, session management, and replication protocols. An application programmer's job is to understand what the business wants the transaction to do and to write a program that does it. The system software should make it possible to run that program on a system that is fast, efficient, scalable, and reliable. This is what transactional middleware does, which is the main subject of the first half of this book, Chapters 1 through 5. Today's products and standards for transactional middleware are described in Chapter 10.

Users of a transaction processing system want to think of it as a sequential processor of transactions, one that's infinitely reliable, gives them its full and undivided attention while it executes their transaction, executes the whole transaction (not just part of it), and saves the result of their transaction forever. This is a tall order and doesn't at all describe what's really going on inside the system: The system executes many transactions concurrently; it fails from time to time due to software and hardware errors, often at the worst possible moment (when it's running *your* transaction); and it has limited storage capacity. Yet, through a combination of software techniques, the system approximates the behavior that users want. Those techniques are the main subject of Chapters 6 through 9.

As computing technology evolves, transaction processing technology will evolve to support it. We discuss some major trends in Chapter 11: cloud computing, scalable distributed computing, flash storage, and streams and event processing.

Here is a summary of what you'll find in each chapter:

Chapter 1, Introduction: Gives a broad-brush overview of transaction processing application and system structure. It describes service-oriented computing, the ACID properties of transactions, the two-phase commit protocol, the industry-standard TPC performance benchmarks, high availability requirements, and the relationship of transaction processing to batch processing, real-time, and data warehousing systems.

Chapter 2, Transaction Processing Abstractions: Describes the main software abstractions found in transaction processing systems: transactions; processes and threads; remote procedure call; techniques for managing shared state, such as transaction context, sessions, and cookies; and scalability techniques, such as caching, resource pooling, partitioning, and replication.

Chapter 3, Transaction Processing Application Architecture: Explains the value of multitier application architecture and then delves into each tier in detail: front ends that use forms and web servers to communicate with end-user devices; request controllers that bracket transactions; and transaction servers that execute transactions. It also explains how transactional middleware and database servers structure these activities.

Chapter 4, Queued Transaction Processing: Shows how a persistent message queue adds reliability. It gives detailed walk-throughs of recovery scenarios and shows how queues drive publish-subscribe, broker-based and bus-based message-oriented middleware. It also explains the internals of queue managers, with IBM's Websphere MQ and Oracle's Stream AQ as examples.

Chapter 5, Business Process Management: Describes mechanisms to support the creation, management, and monitoring of business processes that execute as multiple related transactions. It explains how to obtain suitable atomicity, isolation, and durability of multitransaction requests. It summarizes the business process execution language (BPEL) standard and, as an example, business process mechanisms in Microsoft SQL Service Broker.

Chapter 6, Locking: Shows how and why two-phase locking works and how application programmers affect its correctness and performance. It describes lock manager implementation and deadlock handling. It then explains in detail how performance can be controlled by lock granularity, optimistic methods, batching, avoiding hot spots, avoiding phantoms, and supporting query-update workloads using lower degrees of isolation and multiversion methods. Finally, it covers B-tree locking and multigranularity locking used in SQL database systems, and nested transaction locking.

Chapter 7, System Recovery: Identifies what causes failures, and how transactions help mask their effects. It discusses checkpoint-based application recovery, using stateless servers to simplify recovery, and warm and hot standby systems that use process pairs to reduce recovery time. It then explains how database systems use logging to recover from transaction failures, system failures, and media failures. It explains the undo and redo paradigm, how and why logging algorithms work, log checkpointing, recovery algorithms, shadow paging, some fancy popular logging optimizations (including the ARIES algorithm), and archive recovery.

Chapter 8, Two-Phase Commit: Explains the two-phase commit protocol in detail. It carefully walks through recovery situations and shows where and why the user must get involved. It presents popular optimizations such as presumed abort, phase zero, and transfer of coordination. And it explains how database systems and transaction managers interoperate using the XA interface of the X/Open transaction management architecture.

Chapter 9, Replication: Describes the tradeoffs of replicating servers versus replicating resources and shows how the correctness criterion, one-copy serializability, applies to each of them. It presents the two most popular approaches to replication: primary-copy replication, where updates to a primary are propagated to secondaries; and multimaster replication, where updates are applied to any copy and then propagate to other copies. It also explains synchronization of replicated caches that connect to a shared database. It covers algorithms for electing a primary, quorum consensus, establishing the latest state, and replica recovery.

Chapter 10, Transactional Middleware Products and Standards: Describes popular products and standards for transactional middleware, such as Java Enterprise Edition, Microsoft's .NET Enterprise Services, legacy transaction processing monitors (CICS, IMS, Tuxedo, ACMS, and Pathway), and other service-oriented middleware. Component technologies include Windows Communications Foundation, Enterprise Java Beans, Java Database Connectors, Java Transaction API, and the Spring Framework, which appear in products from IBM, Oracle, Progress, and in open source software. It also describes transaction standards from OMG and X/Open, and Web Services standards from OASIS.

Chapter 11, Future Trends: Discusses major directions where transaction processing technology is headed: cloud computing platforms, composing scalable systems using distributed computing components, the use of flash storage to replace disks, and data and event streams from sensor devices as a source of transaction requests.

GUIDANCE FOR INSTRUCTORS

The first author has taught transaction processing courses several dozen times over the past 25 years. Details of his most recent offerings are on the web site of the Department of Computer Science and Engineering at the University of Washington, http://www.cs.washington.edu/education/courses/csep545/, where you'll find assignments, projects, and video-recorded lectures.

The syllabus that has worked best for a formal university course is to use the first half of the course to cover Chapter 1 of this book followed by principles of concurrency control (Sections 6.1–6.4 of Chapter 6) and recovery (Chapter 7). These topics immerse students in challenging technical details that are best learned through structured homework assignments and are amenable to a conventional exam. This gets students to the point where they can work on a course project.

Transaction processing is a systems engineering problem, with many interacting parts. We have tried three different kinds of course projects to help students deepen their understanding of how the parts fit together: case studies of applications; building an application using commercial products (such as Microsoft .NET or Java Enterprise Edition); and building a transactional middleware system for running distributed transactions. The last of these projects has been the most effective by far, from both the students' and instructor's viewpoint. So in recent offerings, we require that all students do this project.

The project involves building a skeleton of a travel reservation system for flights, hotel rooms, and rental cars. This requires them to build a resource manager with locking and recovery, a two-phase commit protocol, and transactional middleware to move requests around. We found this was too much work for a 10-week quarter, even for graduate students who are full-time professional programmers. So we give them some of the components to start with. The software is downloadable from the course web site.

ACKNOWLEDGMENTS

This book began over 20 years ago as course notes developed by the first author. Over the years, the course has been presented to over a thousand people at Digital Equipment Corp., the Wang Institute of Graduate Studies (gone, but not forgotten), Microsoft Corp., and University of Washington. The "students," most of them practicing engineers, suggested countless ideas that have become part of this book. We thank them all for the rich detail they provided.

Many people gave generously of their time to review selected chapters. They corrected our blunders, pointed out holes, and often filled them in. We are very grateful to Brian Milnes, who reviewed the entire book in detail and added much from his experience in running large Internet TP systems. We greatly appreciate the help we received on various chapters from John Apps, Darach Ennis, Mike Keith, David Kubelka, Mark Little, Peter Niblett, Betty O'Neil, Pat O'Neil, Gera Shegalov, Tony Storey, Satish Thatte, Roger Wolter, and especially Lev Novik, who was a partner in developing the new section on multimaster replication.

It was a major challenge to include many examples of products, applications, and benchmarks, and to get them right. We could never have done it without the substantial assistance of the engineers who work on those artifacts. They reviewed several iterations, to help us think through every detail. While we take full responsibility for any errors that slipped through, we are pleased to share the credit with Keith Evans, Max Feingold, Tom Freund, Jonathan Halliday, Jeurgen Hoeller, Rajkumar Irudayaraj, Jim Johnson, Ed Lassettre, Charles Levine, Anne Thomas Manes, Miko Matsumura, Laurence Melloul, Dean Meltz, Geoff Nichols, Greg Pavlik, Mike Pizzo, Ian Robinson, Adrian Trenaman, Steve Vinoski, John Wells, and Jesse Yurkovich.

Although much of this edition is new, much is not. In writing the first edition, we benefited enormously from the help of Jim Gray and Joe Twomey. We also appreciate the help we received from Mario Bocca, Dexter Bradshaw, Ian Carrie, Ed Cobb, Gagan Chopra, Dick Dievendorff, Keith Evans, Wayne Duquaine, Terry Dwyer, Ko Fujimura, Per Gyllstrom, Vassos Hadzilacos, Brad Hammond, Pat Helland, Greg Hope, Larry Jacobs, Roger King, Walt Kohler, Barbara Klein, Dave Lomet, Susan Malaika, Michael C. Morrison, M. Tamer Ozsu, Wes Saeger, David Schorow, Randy Smerik, Alex Thomasian, Karen Watterson, and Tom Wimberg.

We thank Vassos Hadzilacos, Nat Goodman, and the Addison-Wesley Publishing Company for permission to republish excerpts from *Concurrency Control and Recovery in Database Systems*, by P. Bernstein, V. Hadzilacos, and N. Goodman, primarily in Chapter 8.

We thank our editors, Rick Adams, Diane Cerra, and Denise Penrose for their encouragement, flexibility, and good advice, as well as the production staff at Elsevier, and especially Jeff Freeland, for their efficiency and careful attention in the production of the book.

Finally, we thank our families and friends for indulging our moaning, keeping us happy, and accepting our limited companionship without complaint, while all our discretionary time was consumed by this writing. It's over … for awhile☺.

Trademarks