SEVENTH EDITION

Database System Concepts

Abraham Silberschatz
Henry F. Korth
S. Sudarshan

Mc Graw Hill Education

DATABASE
SYSTEM CONCEPTS

SEVENTH EDITION

Abraham Silberschatz
Yale University

Henry F. Korth
Lehigh University

S. Sudarshan
Indian Institute of Technology, Bombay

Mc Graw Hill Education

DATABASE SYSTEM CONCEPTS, SEVENTH EDITION

This book is printed on acid-free paper.

1 2 3 4 5 6 7 8 9 LCR 21 20 19

ISBN 978-0-07-802215-9 (bound edition)
MHID 0-07-802215-0 (bound edition)
ISBN 978-1-260-51504-6 (loose-leaf edition)
MHID 1-260-51504-4 (loose-leaf edition)

Portfolio Manager: *Thomas Scaife Ph.D.*
Product Developers: *Tina Bower & Megan Platt*
Marketing Manager: *Shannon O'Donnell*
Content Project Managers: *Laura Bies & Sandra Schnee*
Buyer: *Susan K. Culbertson*
Design: *Egzon Shaqiri*
Content Licensing Specialists: *Shawntel Schmitt & Lorraine Buczek*
Cover Image: © *Pavel Nesvadba/Shutterstock*
Compositor: *Aptara®, Inc.*

Library of Congress Cataloging-in-Publication Data

Names: Silberschatz, Abraham, author. | Korth, Henry F., author. | Sudarshan, S., author.
Title: Database system concepts/Abraham Silberschatz, Yale University, Henry F. Korth, Lehigh University, S. Sudarshan, Indian Institute of Technology, Bombay.
Description: Seventh edition. | New York, NY: McGraw-Hill, [2020] | Includes bibliographical references.
Identifiers: LCCN 2018060474 | ISBN 9780078022159 (alk. paper) | ISBN 0078022150 (alk. paper)
Subjects: LCSH: Database management.
Classification: LCC QA76.9.D3 S5637 2020 | DDC 005.74—dc23 LC record available at https://lccn.loc.gov/2018060474

mheducation.com/highered

To meine schatzi, Valerie
her parents and my dear friends, Steve and Mary Anne
and in memory of my parents, Joseph and Vera

Avi Silberschatz

To my wife, Joan
my children, Abigail and Joseph
my mother, Frances
and in memory of my father, Henry

Hank Korth

To my wife, Sita
my children, Madhur and Advaith
and my mother, Indira

S. Sudarshan

About the Authors

Abraham (Avi) Silberschatz is the Sidney J. Weinberg Professor of Computer Science at Yale University. Prior to coming to Yale in 2003, he was the vice president of the Information Sciences Research Center at Bell Labs. He previously held an endowed professorship at the University of Texas at Austin, where he taught until 1993. Silberschatz is a fellow of the ACM, a fellow of the IEEE, and a member of the Connecticut Academy of Science and Engineering. He received the 2002 IEEE Taylor L. Booth Education Award, the 1998 ACM Karl V. Karlstrom Outstanding Educator Award, and the 1997 ACM SIGMOD Contribution Award. Silberschatz was awarded the Bell Laboratories President's Award three times, in 1998, 1999 and 2004. His writings have appeared in numerous journals, conferences, workshops, and book chapters. He has obtained over 48 patents and over 24 grants. He is an author of the textbook *Operating System Concepts*.

Henry F. (Hank) Korth is a Professor of Computer Science and Engineering and co-director of the Computer Science and Business program at Lehigh University. Prior to joining Lehigh, he was director of Database Principles Research at Bell Labs, a vice president of Panasonic Technologies, an associate professor at the University of Texas at Austin, and a research staff member at IBM Research. Korth is a fellow of the ACM and of the IEEE and a winner of the 10-Year Award at the VLDB Conference. His numerous research publications span a wide range of aspects of database systems, including transaction management in parallel and distributed systems, real-time systems, query processing, and the influence on these areas from modern computing architectures. Most recently, his research has addressed issues in the application of blockchains in enterprise databases.

S. Sudarshan is currently the Subrao M. Nilekani Chair Professor at the Indian Institute of Technology, Bombay. He received his Ph.D. at the University of Wisconsin in 1992, and he was a member of the technical staff at Bell Labs before joining IIT Bombay. Sudarshan is a fellow of the ACM. His research spans several areas of database systems, with a focus on query processing and query optimization. His paper on keyword search in databases published in 2002 won the IEEE ICDE Most Influential Paper Award in 2012, and his work on main-memory databases received the Bell Laboratories President's Award in 1999. His current research areas include testing and grading of SQL queries, optimization of database applications by rewriting of imperative code, and query optimization for parallel databases. He has published over 100 papers and obtained 15 patents.

Contents

Chapter 4 Intermediate SQL

Chapter 5 Advanced SQL

PART TWO ■ DATABASE DESIGN

Chapter 6 Database Design Using the E-R Model

Chapter 7 Relational Database Design

PART THREE ■ APPLICATION DESIGN AND DEVELOPMENT

Chapter 8 Complex Data Types

Chapter 9 Application Development

PART FOUR ■ BIG DATA ANALYTICS

Chapter 10 Big Data

Chapter 11 Data Analytics

PART FIVE ■ STORAGE MANAGEMENT AND INDEXING

Chapter 12 Physical Storage Systems

Chapter 13 Data Storage Structures

Chapter 14 Indexing

PART SIX ■ QUERY PROCESSING AND OPTIMIZATION

Chapter 15 Query Processing

Chapter 16 Query Optimization

PART SEVEN ■ TRANSACTION MANAGEMENT

Chapter 17 Transactions

Chapter 18 Concurrency Control

Chapter 19 Recovery System

PART EIGHT ■ PARALLEL AND DISTRIBUTED DATABASES

Chapter 20 Database-System Architectures

Chapter 21 Parallel and Distributed Storage

Chapter 22 Parallel and Distributed Query Processing

Chapter 23 Parallel and Distributed Transaction Processing

PART NINE ■ ADVANCED TOPICS

Chapter 24 Advanced Indexing Techniques

Chapter 25 Advanced Application Development

Chapter 26 Blockchain Databases

PART TEN ■ APPENDIX A

PART ELEVEN ■ ONLINE CHAPTERS

Preface

Database management has evolved from a specialized computer application to a central component of virtually all enterprises, and, as a result, knowledge about database systems has become an essential part of an education in computer science. In this text, we present the fundamental concepts of database management. These concepts include aspects of database design, database languages, and database-system implementation.

This text is intended for a first course in databases at the junior or senior undergraduate, or first-year graduate, level. In addition to basic material for a first course, the text contains advanced material that can be used for course supplements, or as introductory material for an advanced course.

We assume only a familiarity with basic data structures, computer organization, and a high-level programming language such as Java, C, C++, or Python. We present concepts as intuitive descriptions, many of which are based on our running example of a university. Important theoretical results are covered, but formal proofs are omitted. In place of proofs, figures and examples are used to suggest why a result is true. Formal descriptions and proofs of theoretical results may be found in research papers and advanced texts that are referenced in the bibliographical notes.

The fundamental concepts and algorithms covered in the book are often based on those used in existing commercial or experimental database systems. Our aim is to present these concepts and algorithms in a general setting that is not tied to one particular database system, though we do provide references to specific systems where appropriate.

In this, the seventh edition of *Database System Concepts*, we have retained the overall style of the prior editions while evolving the content and organization to reflect the changes that are occurring in the way databases are designed, managed, and used. One such major change is the extensive use of "Big Data" systems. We have also taken into account trends in the teaching of database concepts and made adaptations to facilitate these trends where appropriate.

Among the notable changes in this edition are:

- Extensive coverage of Big Data systems, from the user perspective (Chapter 10), as well as from an internal perspective (Chapter 20 through Chapter 23), with extensive additions and modifications compared to the sixth edition.
- A new chapter entitled "Blockchain Databases" (Chapter 26) that introduces blockchain technology and its growing role in enterprise applications. An important focus in this chapter is the interaction between blockchain systems and database systems.
- Updates to all chapters covering database internals (Chapter 12 through Chapter 19) to reflect current-generation technology, such as solid-state disks, main-memory databases, multi-core systems, and column-stores.
- Enhanced coverage of semi-structured data management using JSON, RDF, and SPARQL (Section 8.1).
- Updated coverage of temporal data (in Section 7.10), data analytics (Chapter 11), and advanced indexing techniques such as write-optimized indices (Section 14.8 and Section 24.2).
- Reorganization and update of chapters to better support courses with a significant hands-on component (which we strongly recommend for any database course), including use of current-generation application development tools and Big Data systems such as Apache Hadoop and Spark.

These and other updates have arisen from the many comments and suggestions we have received from readers of the sixth edition, our students at Yale University, Lehigh University, and IIT Bombay, and our own observations and analyses of developments in database technology.

Content of This Book

The text is organized in eleven major parts.

- **Overview** (Chapter 1). Chapter 1 provides a general overview of the nature and purpose of database systems. We explain how the concept of a database system has developed, what the common features of database systems are, what a database system does for the user, and how a database system interfaces with operating systems. We also introduce an example database application: a university organization consisting of multiple departments, instructors, students, and courses. This application is used as a running example throughout the book. This chapter is motivational, historical, and explanatory in nature.

- **Part 1: Relational Model and SQL** (Chapter 2 through Chapter 5). Chapter 2 introduces the relational model of data, covering basic concepts such as the structure of relational databases, database schemas, keys, schema diagrams, relational query languages, relational operations, and the relational algebra. Chapter 3, Chapter 4, and Chapter 5 focus on the most influential of the user-oriented relational languages: SQL. The chapters in this part describe data manipulation: queries, updates, insertions, and deletions, assuming a schema design has been provided. Although data-definition syntax is covered in detail here, schema design issues are deferred to Part 2.
- **Part 2: Database Design** (Chapter 6 and Chapter 7). Chapter 6 provides an overview of the database-design process and a detailed description of the entity-relationship data model. The entity-relationship data model provides a high-level view of the issues in database design and of the problems encountered in capturing the semantics of realistic applications within the constraints of a data model. UML class-diagram notation is also covered in this chapter. Chapter 7 introduces relational database design. The theory of functional dependencies and normalization is covered, with emphasis on the motivation and intuitive understanding of each normal form. This chapter begins with an overview of relational design and relies on an intuitive understanding of logical implication of functional dependencies. This allows the concept of normalization to be introduced prior to full coverage of functional-dependency theory, which is presented later in the chapter. Instructors may choose to use only this initial coverage without loss of continuity. Instructors covering the entire chapter will benefit from students having a good understanding of normalization concepts to motivate them to learn some of the challenging concepts of functional-dependency theory. The chapter ends with a section on modeling of temporal data.
- **Part 3: Application Design and Development** (Chapter 8 and Chapter 9). Chapter 8 discusses several complex data types that are particularly important for application design and development, including semi-structured data, object-based data, textual data, and spatial data. Although the popularity of XML in a database context has been diminishing, we retain an introduction to XML, while adding coverage of JSON, RDF, and SPARQL. Chapter 9 discusses tools and technologies that are used to build interactive web-based and mobile database applications. This chapter includes detailed coverage on both the server side and the client side. Among the topics covered are servlets, JSP, Django, JavaScript, and web services. Also discussed are application architecture, object-relational mapping systems including Hibernate and Django, performance (including caching using memcached and Redis), and the unique challenges in ensuring web-application security.
- **Part 4: Big Data Analytics** (Chapter 10 and Chapter 11). Chapter 10 provides an overview of large-scale data-analytic applications, with a focus on how those applications place distinct demands on data management compared with the de-

mands of traditional database applications. The chapter then discusses how those demands are addressed. Among the topics covered are Big Data storage systems including distributed file systems, key-value stores and NoSQL systems, MapReduce, Apache Spark, streaming data, and graph databases. The connection of these systems and concepts with database concepts introduced earlier is emphasized. Chapter 11 discusses the structure and use of systems designed for large-scale data analysis. After first explaining the concepts of data analytics, business intelligence, and decision support, the chapter discusses the structure of a data warehouse and the process of gathering data into a warehouse. The chapter next covers usage of warehouse data in OLAP applications followed by a survey of data-mining algorithms and techniques.

- **Part 5: Storage Management and Indexing** (Chapter 12 through Chapter 14). Chapter 12 deals with storage devices and how the properties of those devices influence database physical organization and performance. Chapter 13 deals with data-storage structures, including file organization and buffer management. A variety of data-access techniques are presented in Chapter 14. Multilevel index-based access is described, culminating in detailed coverage of B^+-trees. The chapter then covers index structures for applications where the B^+-tree structure is less appropriate, including write-optimized indices such as LSM trees and buffer trees, bitmap indices, and the indexing of spatial data using k-d trees, quadtrees and R-trees.
- **Part 6: Query Processing and Optimization** (Chapter 15 and Chapter 16). Chapter 15 and Chapter 16 address query-evaluation algorithms and query optimization. Chapter 15 focuses on algorithms for the implementation of database operations, particularly the wide range of join algorithms, which are designed to work on very large data that may not fit in main-memory. Query processing techniques for main-memory databases are also covered in this chapter. Chapter 16 covers query optimization, starting by showing how query plans can be transformed to other equivalent plans by using transformation rules. The chapter then describes how to generate estimates of query execution costs, and how to efficiently find query execution plans with the lowest cost.
- **Part 7: Transaction Management** (Chapter 17 through Chapter 19). Chapter 17 focuses on the fundamentals of a transaction-processing system: atomicity, consistency, isolation, and durability. It provides an overview of the methods used to ensure these properties, including log-based recovery and concurrency control using locking, timestamp-based techniques, and snapshot isolation. Courses requiring only a survey of the transaction concept can use Chapter 17 on its own without the other chapters in this part; those chapters provide significantly greater depth. Chapter 18 focuses on concurrency control and presents several techniques for ensuring serializability, including locking, timestamping, and optimistic (validation) techniques. Multiversion concurrency control techniques, including the widely used snapshot isolation technique, and an extension of the technique that

guarantees serializability, are also covered. This chapter also includes discussion of weak levels of consistency, concurrency on index structures, concurrency in main-memory database systems, long-duration transactions, operation-level concurrency, and real-time transaction processing. Chapter 19 covers the primary techniques for ensuring correct transaction execution despite system crashes and storage failures. These techniques include logs, checkpoints, and database dumps, as well as high availability using remote backup systems. Recovery with early lock release, and the widely used ARIES algorithm are also presented. This chapter includes discussion of recovery in main-memory database systems and the use of NVRAM.

- **Part 8: Parallel and Distributed Databases** (Chapter 20 through Chapter 23). Chapter 20 covers computer-system architecture, and describes the influence of the underlying computer system on the database system. We discuss centralized systems, client–server systems, parallel and distributed architectures, and cloud-based systems in this chapter. The remaining three chapters in this part address distinct aspects of parallel and distributed databases, with Chapter 21 covering storage and indexing, Chapter 22 covering query processing, and Chapter 23 covering transaction management. Chapter 21 includes discussion of partitioning and data skew, replication, parallel indexing, distributed file systems (including the Hadoop file system), and parallel key-value stores. Chapter 22 includes discussion of parallelism both among multiple queries and within a single query. It covers parallel and distributed sort and join, MapReduce, pipelining, the Volcano exchange-operator model, thread-level parallelism, streaming data, and the optimization of geographically distributed queries. Chapter 23 includes discussion of traditional distributed consensus such as two-phase commit and more sophisticated solutions including Paxos and Raft. It covers a variety of algorithms for distributed concurrency control, including replica management and weaker degrees of consistency. The trade-offs implied by the CAP theorem are discussed along with the means of detecting inconsistency using version vectors and Merkle trees.
- **Part 9: Advanced Topics** (Chapter 24 through Chapter 26). Chapter 24 expands upon the coverage of indexing in Chapter 14 with detailed coverage of the LSM tree and its variants, bitmap indices, spatial indexing, and dynamic hashing techniques. Chapter 25 expands upon the coverage of Chapter 9 with a discussion of performance tuning, benchmarking, testing, and migration from legacy systems, standardization, and distributed directory systems. Chapter 26 looks at blockchain technology from a database perspective. It describes blockchain data structures and the use of cryptographic hash functions and public-key encryption to ensure the blockchain properties of anonymity, irrefutability, and tamper resistance. It describes and compares the distributed consensus algorithms used to ensure decentralization, including proof-of-work, proof-of-stake, and Byzantine consensus. Much of the chapter focuses on the features that make blockchain an important database concept, including the role of permisssioned blockchains, the encoding

of business logic and agreements in smart contracts, and interoperability across blockchains. Techniques for achieving database-scale transaction-processing performance are discussed. A final section surveys current and contemplated enterprise blockchain applications.

- **Part 10: Appendix**. Appendix A presents details of our university schema, including the full schema, DDL, and all the tables.
- **Part 11: Online Chapters** (Chapter 27 through Chapter 32) available online at db-book.com. We provide six chapters that cover material that is of historical nature or is advanced; these chapters are available only online. Chapter 27 covers "pure" query languages: the tuple and domain relational calculus and Datalog, which has a syntax modeled after the Prolog language. Chapter 28 covers advanced topics in relational database design, including the theory of multivalued dependencies and fourth normal form, as well as higher normal forms. Chapter 29 covers object-based databases and more complex data types such as array, and multiset types, as well as tables that are not in 1NF. Chapter 30 expands on the coverage in Chapter 8 of XML. Chapter 31 covers information retrieval, which deals with querying of unstructured textual data. Chapter 32 provides an overview of the PostgreSQL database system, and is targeted at courses focusing on database internals. The chapter is likely to be particularly useful for supporting student projects that work with the open-source code base of the PostgreSQL database.

At the end of each chapter we provide references in a section titled *Further Reading*. This section is intentionally abbreviated and provides references that allow students to continue their study of the material covered in the chapter or to learn about new developments in the area covered by the chapter. On occasion, the further reading section includes original source papers that have become classics of which everyone should be aware. Detailed bibliographical notes for each chapter are available online, and provide references for readers who wish to go into further depth on any of the topics covered in the chapter.

The Seventh Edition

The production of this seventh edition has been guided by the many comments and suggestions we received concerning the earlier editions, by our own observations while teaching at Yale University, Lehigh University, and IIT Bombay, and by our analysis of the directions in which database technology is evolving.

We provided a list of the major new features of this edition earlier in this preface; these include coverage of extensive coverage of Big Data, updates to all chapters to reflect current generation hardware technology, semi-structured data management, advanced indexing techniques, and a new chapter on blockchain databases. Beyond these major changes, we revised the material in each chapter, bringing the older material

up-to-date, adding discussions on recent developments in database technology, and improving descriptions of topics that students found difficult to understand. We have also added new exercises and updated references.

For instructors who previously used the sixth edition, we list the more significant changes below:

- Relational algebra has been moved into Chapter 2, to help students better understand relational operations that form the basis of query languages such as SQL. Deeper coverage of relational algebra also aids in understanding the algebraic operators needed for discussion later of query processing and optimization. The two variants of the relational calculus are now in an online chapter, since we believe they are now of value only to more theoretically oriented courses, and can be omitted by most database courses.
- The SQL chapters now include more details of database-system specific SQL variations, to aid students carrying out practical assignments. Connections between SQL and the multiset relational algebra are also covered in more detail. Chapter 4 now covers all the material concerning joins, whereas previously natural join was in the preceding chapter. Coverage of sequences used to generate unique key values, and coverage of row-level security have also been added to this chapter. Recent extensions to the JDBC API that are particularly useful are now covered in Chapter 5; coverage of OLAP has been moved from this chapter to Chapter 11.
- Chapter 6 has been modified to cover E-R diagrams along with E-R concepts, instead of first covering the concepts and then introducing E-R diagrams as was done in earlier editions. We believe this will help students better comprehend the E-R model.
- Chapter 7 now has improved coverage of temporal data modeling, including SQL:2011 temporal database features.
- Chapter 8 is a new chapter that covers complex data types, including semi-structured data, such as XML, JSON, RDF, and SPARQL, object-based data, textual data, and spatial data. Object-based databases, XML, and information retrieval on textual data were covered in detail in the sixth edition; these topics have been abbreviated and covered in Chapter 8, while the original chapters from the sixth edition have now been made available online.
- Chapter 9 has been significantly updated to reflect modern application development tools and techniques, including extended coverage of JavaScript and JavaScript libraries for building dynamic web interfaces, application development in Python using the Django framework, coverage of web services, and disconnection operations using HTML5. Object-relation mapping using Django has been added, as also discussion of techniques for developing high-performance applications that can handle large transaction loads.

- Chapter 10 is a new chapter on Big Data, covering Big Data concepts and tools from a user perspective. Big Data storage systems, the MapReduce paradigm, Apache Hadoop and Apache Spark, and streaming and graph databases are covered in this chapter. The goal is to enable readers to use Big Data systems, with only a summary coverage of what happens behind the scenes. Big Data internals are covered in detail in later chapters.
- The chapter on storage and file structure has been split into two chapters. Chapter 12 which covers storage has been updated with new technology, including expanded coverage of flash memory, column-oriented storage, and storage organization in main-memory databases. Chapter 13, which covers data storage structures has been expanded, and now covers details such as free-space maps, partitioning, and most importantly column-oriented storage.
- Chapter 14 on indexing now covers write-optimized index structures including the LSM tree and its variants, and the buffer tree, which are seeing increasing usage. Spatial indices are now covered briefly in this chapter. More detailed coverage of LSM trees and spatial indices is provided in Chapter 24, which covers advanced indexing techniques. Bitmap indices are now covered in brief in Chapter 14, while more detailed coverage has been moved to Chapter 24. Dynamic hashing techniques have been moved into Chapter 24, since they are of limited practical importance today.
- Chapter 15 on query processing has significantly expanded coverage of pipelining in query processing, new material on query processing in main-memory, including query compilation, as well as brief coverage of spatial joins. Chapter 16 on query optimization has more examples of equivalence rules for operators such as outer joins and aggregates, has updated material on statistics for cost estimation, an improved presentation of the join-order optimization algorithm. Techniques for decorrelating nested subqueries using semijoin and antijoin operations have also been added.
- Chapter 18 on concurrency control has new material on concurrency control in main-memory. Chapter 19 on recovery now gives more importance to high availability using remote backup systems.
- Our coverage of parallel and distributed databases has been completely revamped. Because of the evolution of these two areas into a continuum from low-level parallelism to geographically distributed systems, we now present these topics together.
    - Chapter 20 on database architectures has been significantly updated from the earlier edition, including new material on practical interconnection networks like the tree-like (or fat-tree) architecture, and significantly expanded and updated material on shared-memory architectures and cache coherency. There is an entirely new section on cloud-based services, covering virtual machines and containers, platform-as-a-service, software-as-a-service, and elasticity.

- Chapter 21 covers parallel and distributed storage; while a few parts of this chapter were present in the sixth edition, such as partitioning techniques, everything else in this chapter is new.
- Chapter 22 covers parallel and distributed query processing. Again only a few sections of this chapter, such as parallel algorithms for sorting, join, and a few other relational operations, were present in the sixth edition, almost everything else in this chapter is new.
- Chapter 23 covers parallel and distributed transaction processing. A few parts of this chapter, such as the sections on 2PC, persistent messaging, and concurrency control in distributed databases, are new but almost everything else in this chapter is new.

As in the sixth edition, we facilitate the following of our running example by listing the database schema and the sample relation instances for our university database together in Appendix A as well as where they are used in the various regular chapters. In addition, we provide, on our web site db-book.com, SQL data-definition statements for the entire example, along with SQL statements to create our example relation instances. This encourages students to run example queries directly on a database system and to experiment with modifying those queries. All topics not listed above are updated from the sixth edition, though their overall organization is relatively unchanged.

End of Chapter Material

Each chapter has a list of review terms, in addition to a summary, which can help readers review key topics covered in the chapter.

As in the sixth edition, the exercises are divided into two sets: **practice exercises** and **exercises**. The solutions for the practice exercises are publicly available on the web site of the book. Students are encouraged to solve the practice exercises on their own and later use the solutions on the web site to check their own solutions. Solutions to the other exercises are available only to instructors (see "Instructor's Note," below, for information on how to get the solutions).

Many chapters have a tools section at the end of the chapter that provides information on software tools related to the topic of the chapter; some of these tools can be used for laboratory exercises. SQL DDL and sample data for the university database and other relations used in the exercises are available on the web site of the book and can be used for laboratory exercises.

Instructor's Note

It is possible to design courses by using various subsets of the chapters. Some of the chapters can also be covered in an order different from their order in the book. We outline some of the possibilities here:

- Chapter 5 (Advanced SQL). This chapter can be skipped or deferred to later without loss of continuity. We expect most courses will cover at least Section 5.1.1 early, as JDBC is likely to be a useful tool in student projects.
- Chapter 6 (E-R Model). This chapter can be covered ahead of Chapter 3, Chapter 4, and Chapter 5 if you so desire, since Chapter 6 does not have any dependency on SQL. However, for courses with a programming emphasis, a richer variety of laboratory exercises is possible after studying SQL, and we recommend that SQL be covered before database design for such courses.
- Chapter 15 (Query Processing) and Chapter 16 (Query Optimization). These chapters can be omitted from an introductory course without affecting coverage of any other chapter.
- Part 7 (Transaction Management). Our coverage consists of an overview (Chapter 17) followed by chapters with details. You might choose to use Chapter 17 while omitting Chapter 18 and Chapter 19, if you defer these latter chapters to an advanced course.
- Part 8 (Parallel and Distributed Databases). Our coverage consists of an overview (Chapter 20), followed by chapters on the topics of storage, query processing, and transactions. You might choose to use Chapter 20 while omitting Chapter 21 through Chapter 23 if you defer these latter chapters to an advanced course.
- Part 11 (Online chapters). Chapter 27 (Formal-Relational Query Languages). This chapter can be covered immediately after Chapter 2, ahead of SQL. Alternatively, this chapter may be omitted from an introductory course. The five other online chapters (Advanced Relational Database Design, Object-Based Databases, XML, Information Retrieval, and PostgreSQL) can be used as self-study material or omitted from an introductory course.

Model course syllabi, based on the text, can be found on the web site of the book.

Web Site and Teaching Supplements

A web site for the book is available at the URL: db-book.com. The web site contains:

- Slides covering all the chapters of the book.
- Answers to the practice exercises.
- The six online chapters.
- Laboratory material, including SQL DDL and sample data for the university schema and other relations used in exercises, and instructions for setting up and using various database systems and tools.
- An up-to-date errata list.

The following additional material is available only to faculty:

- An instructor's manual containing solutions to all exercises in the book.
- A question bank containing extra exercises.

For more information about how to get a copy of the instructor's manual and the question bank, please send an email message to sem@mheducation.com. In the United States, you may call 800-338-3987. The McGraw-Hill web site for this book is www.mhhe.com/silberschatz.

Contacting Us

We have endeavored to eliminate typos, bugs, and the like from the text. But, as in new releases of software, bugs almost surely remain; an up-to-date errata list is accessible from the book's web site. We would appreciate it if you would notify us of any errors or omissions in the book that are not on the current list of errata.

We would be glad to receive suggestions on improvements to the book. We also welcome any contributions to the book web site that could be of use to other readers, such as programming exercises, project suggestions, online labs and tutorials, and teaching tips.

Email should be addressed to db-book-authors@cs.yale.edu. Any other correspondence should be sent to Avi Silberschatz, Department of Computer Science, Yale University, 51 Prospect Street, P.O. Box 208285, New Haven, CT 06520-8285 USA.

Acknowledgments

Many people have helped us with this seventh edition, as well as with the previous six editions from which it is derived, and we are indebted to all of them.

Seventh Edition

- Ioannis Alagiannis and Renata Borovica-Gajic for writing Chapter 32 on the PostgreSQL database, which is available online. The chapter is a complete rewrite of the PostgreSQL chapter in the 6th edition, which was authored by Anastasia Ailamaki, Sailesh Krishnamurthy, Spiros Papadimitriou, Bianca Schroeder, Karl Schnaitter, and Gavin Sherry.
- Judi Paige for her help in generating figures, presentation slides, and with handling the copy-editing material.
- Mark Wogahn for making sure that the software to produce the book, including LaTeX macros and fonts, worked properly.

- Sriram Srinivasan for discussions and feedback that have immensely benefited the chapters on parallel and distributed databases.
- N. L. Sarda for his insightful feedback on the sixth edition, and on some sections of the seventh edition.
- Bikash Chandra and Venkatesh Emani for their help with updates to the application development chapter, including creation of sample code.
- Students at IIT Bombay, particularly Ashish Mithole, for their feedback on draft versions of the chapters on parallel and distributed databases.
- Students at Yale, Lehigh, and IIT Bombay, for their comments on the sixth edition.
- Jeffrey Anthony, partner and CTO, Synaptic; and Lehigh students Corey Caplan (now co-founder, Leavitt Innovations); Gregory Cheng; Timothy LaRowe; and Aaron Rotem for comments and suggestions that have benefited the new blockchain chapter.

Previous Editions

- Hakan Jakobsson (Oracle), for writing the chapter on the Oracle database system in the sixth edition; Sriram Padmanabhan (IBM), for writing the chapter describing the IBM DB2 database system in the sixth edition; and Sameet Agarwal, José A. Blakeley, Thierry D'Hers, Gerald Hinson, Dirk Myers, Vaqar Pirzada, Bill Ramos, Balaji Rathakrishnan, Michael Rys, Florian Waas, and Michael Zwilling for writing the chapter describing the Microsoft SQL Server database system in the sixth edition; and in particular José Blakeley, who sadly is no longer amongst us, for coordinating and editing the chapter; and César Galindo-Legaria, Goetz Graefe, Kalen Delaney, and Thomas Casey for their contributions to the previous edition of the Microsoft SQL Server chapter. These chapters, however, are not part of the seventh edition.
- Anastasia Ailamaki, Sailesh Krishnamurthy, Spiros Papadimitriou, Bianca Schroeder, Karl Schnaitter, and Gavin Sherry for writing the chapter on PostgreSQL in the sixth edition.
- Daniel Abadi for reviewing the table of contents of the fifth edition and helping with the new organization.
- Steve Dolins, University of Florida; Rolando Fernanez, George Washington University; Frantisek Franek, McMaster University; Latifur Khan, University of Texas at Dallas; Sanjay Madria, Missouri University of Science and Technology; Aris Ouksel, University of Illinois; and Richard Snodgrass, University of Waterloo; who served as reviewers of the book and whose comments helped us greatly in formulating the sixth edition.

- Judi Paige for her help in generating figures and presentation slides.
- Mark Wogahn for making sure that the software to produce the book, including LaTeX macros and fonts, worked properly.
- N. L. Sarda for feedback that helped us improve several chapters. Vikram Pudi for motivating us to replace the earlier bank schema; and Shetal Shah for feedback on several chapters.
- Students at Yale, Lehigh, and IIT Bombay, for their comments on the fifth edition, as well as on preprints of the sixth edition.
- Chen Li and Sharad Mehrotra for providing material on JDBC and security for the fifth edition.
- Marilyn Turnamian and Nandprasad Joshi provided secretarial assistance for the fifth edition, and Marilyn also prepared an early draft of the cover design for the fifth edition.
- Lyn Dupré copyedited the third edition and Sara Strandtman edited the text of the third edition.
- Nilesh Dalvi, Sumit Sanghai, Gaurav Bhalotia, Arvind Hulgeri K. V. Raghavan, Prateek Kapadia, Sara Strandtman, Greg Speegle, and Dawn Bezviner helped to prepare the instructor's manual for earlier editions.
- The idea of using ships as part of the cover concept was originally suggested to us by Bruce Stephan.
- The following people offered suggestions and comments for the fifth and earlier editions of the book. R. B. Abhyankar, Hani Abu-Salem, Jamel R. Alsabbagh, Raj Ashar, Don Batory, Phil Bernhard, Christian Breimann, Gavin M. Bierman, Janek Bogucki, Haran Boral, Paul Bourgeois, Phil Bohannon, Robert Brazile, Yuri Breitbart, Ramzi Bualuan, Michael Carey, Soumen Chakrabarti, Tom Chappell, Zhengxin Chen, Y. C. Chin, Jan Chomicki, Laurens Damen, Prasanna Dhandapani, Qin Ding, Valentin Dinu, J. Edwards, Christos Faloutsos, Homma Farian, Alan Fekete, Frantisek Franek, Shashi Gadia, Hector Garcia-Molina, Goetz Graefe, Jim Gray, Le Gruenwald, Eitan M. Gurari, William Hankley, Bruce Hillyer, Ron Hitchens, Chad Hogg, Arvind Hulgeri, Yannis Ioannidis, Zheng Jiaping, Randy M. Kaplan, Graham J. L. Kemp, Rami Khouri, Hyoung-Joo Kim, Won Kim, Henry Korth (father of Henry F.), Carol Kroll, Hae Choon Lee, Sang-Won Lee, Irwin Levinstein, Mark Llewellyn, Gary Lindstrom, Ling Liu, Dave Maier, Keith Marzullo, Marty Maskarinec, Fletcher Mattox, Sharad Mehrotra, Jim Melton, Alberto Mendelzon, Ami Motro, Bhagirath Narahari, Yiu-Kai Dennis Ng, Thanh-Duy Nguyen, Anil Nigam, Cyril Orji, Meral Ozsoyoglu, D. B. Phatak, Juan Altmayer Pizzorno, Bruce Porter, Sunil Prabhakar, Jim Peterson, K. V. Raghavan, Nahid Rahman, Rajarshi Rakshit, Krithi Ramamritham, Mike Reiter, Greg Ric-

cardi, Odinaldo Rodriguez, Mark Roth, Marek Rusinkiewicz, Michael Rys, Sunita Sarawagi, N. L. Sarda, Patrick Schmid, Nikhil Sethi, S. Seshadri, Stewart Shen, Shashi Shekhar, Amit Sheth, Max Smolens, Nandit Soparkar, Greg Speegle, Jeff Storey, Dilys Thomas, Prem Thomas, Tim Wahls, Anita Whitehall, Christopher Wilson, Marianne Winslett, Weining Zhang, and Liu Zhenming.

Personal Notes

Sudarshan would like to acknowledge his wife, Sita, for her love, patience, and support, and children Madhur and Advaith for their love and joie de vivre. Hank would like to acknowledge his wife, Joan, and his children, Abby and Joe, for their love and understanding. Avi would like to acknowledge Valerie for her love, patience, and support during the revision of this book.

A. S.
H. F. K.
S. S.

CHAPTER 1

Introduction

A **database-management system** (**DBMS**) is a collection of interrelated data and a set of programs to access those data. The collection of data, usually referred to as the **database**, contains information relevant to an enterprise. The primary goal of a DBMS is to provide a way to store and retrieve database information that is both *convenient* and *efficient*.

Database systems are designed to manage large bodies of information. Management of data involves both defining structures for storage of information and providing mechanisms for the manipulation of information. In addition, the database system must ensure the safety of the information stored, despite system crashes or attempts at unauthorized access. If data are to be shared among several users, the system must avoid possible anomalous results.

Because information is so important in most organizations, computer scientists have developed a large body of concepts and techniques for managing data. These concepts and techniques form the focus of this book. This chapter briefly introduces the principles of database systems.

1.1 Database-System Applications

The earliest database systems arose in the 1960s in response to the computerized management of commercial data. Those earlier applications were relatively simple compared to modern database applications. Modern applications include highly sophisticated, worldwide enterprises.

All database applications, old and new, share important common elements. The central aspect of the application is not a program performing some calculation, but rather the data themselves. Today, some of the most valuable corporations are valuable not because of their physical assets, but rather because of the information they own. Imagine a bank without its data on accounts and customers or a social-network site that loses the connections among its users. Such companies' value would be almost totally lost under such circumstances.

Database systems are used to manage collections of data that:

- are highly valuable,
- are relatively large, and
- are accessed by multiple users and applications, often at the same time.

The first database applications had only simple, precisely formatted, structured data. Today, database applications may include data with complex relationships and a more variable structure. As an example of an application with structured data, consider a university's records regarding courses, students, and course registration. The university keeps the same type of information about each course: course-identifier, title, department, course number, etc., and similarly for students: student-identifier, name, address, phone, etc. Course registration is a collection of pairs: one course identifier and one student identifier. Information of this sort has a standard, repeating structure and is representative of the type of database applications that go back to the 1960s. Contrast this simple university database application with a social-networking site. Users of the site post varying types of information about themselves ranging from simple items such as name or date of birth, to complex posts consisting of text, images, videos, and links to other users. There is only a limited amount of common structure among these data. Both of these applications, however, share the basic features of a database.

Modern database systems exploit commonalities in the structure of data to gain efficiency but also allow for weakly structured data and for data whose formats are highly variable. As a result, a database system is a large, complex software system whose task is to manage a large, complex collection of data.

Managing complexity is challenging, not only in the management of data but in any domain. Key to the management of complexity is the concept of *abstraction*. Abstraction allows a person to use a complex device or system without having to know the details of how that device or system is constructed. A person is able, for example, to drive a car by knowing how to operate its controls. However, the driver does not need to know how the motor was built nor how it operates. All the driver needs to know is an abstraction of what the motor does. Similarly, for a large, complex collection of data, a database system provides a simpler, abstract view of the information so that users and application programmers do not need to be aware of the underlying details of how data are stored and organized. By providing a high level of abstraction, a database system makes it possible for an enterprise to combine data of various types into a unified repository of the information needed to run the enterprise.

Here are some representative applications:

- **Enterprise Information**
  - **Sales:** For customer, product, and purchase information.

- **Accounting:** For payments, receipts, account balances, assets, and other accounting information.
- **Human resources:** For information about employees, salaries, payroll taxes, and benefits, and for generation of paychecks.

- **Manufacturing:** For management of the supply chain and for tracking production of items in factories, inventories of items in warehouses and stores, and orders for items.
- **Banking and Finance**
  - **Banking:** For customer information, accounts, loans, and banking transactions.
  - **Credit card transactions:** For purchases on credit cards and generation of monthly statements.
  - **Finance:** For storing information about holdings, sales, and purchases of financial instruments such as stocks and bonds; also for storing real-time market data to enable online trading by customers and automated trading by the firm.
- **Universities:** For student information, course registrations, and grades (in addition to standard enterprise information such as human resources and accounting).
- **Airlines:** For reservations and schedule information. Airlines were among the first to use databases in a geographically distributed manner.
- **Telecommunication:** For keeping records of calls, texts, and data usage, generating monthly bills, maintaining balances on prepaid calling cards, and storing information about the communication networks.
- **Web-based services**
  - **Social-media:** For keeping records of users, connections between users (such as friend/follows information), posts made by users, rating/like information about posts, etc.
  - **Online retailers:** For keeping records of sales data and orders as for any retailer, but also for tracking a user's product views, search terms, etc., for the purpose of identifying the best items to recommend to that user.
  - **Online advertisements:** For keeping records of click history to enable targeted advertisements, product suggestions, news articles, etc. People access such databases every time they do a web search, make an online purchase, or access a social-networking site.
- **Document databases:** For maintaining collections of new articles, patents, published research papers, etc.
- **Navigation systems:** For maintaining the locations of varies places of interest along with the exact routes of roads, train systems, buses, etc.

As this list illustrates, databases form an essential part not only of every enterprise but also of a large part of a person's daily activities.

The ways in which people interact with databases has changed over time. Early databases were maintained as back-office systems with which users interacted via printed reports and paper forms for input. As database systems became more sophisticated, better languages were developed for programmers to use in interacting with the data, along with user interfaces that allowed end users within the enterprise to query and update data.

As the support for programmer interaction with databases improved, and computer hardware performance increased even as hardware costs decreased, more sophisticated applications emerged that brought database data into more direct contact not only with end users within an enterprise but also with the general public. Whereas once bank customers had to interact with a teller for every transaction, automated-teller machines (ATMs) allowed direct customer interaction. Today, virtually every enterprise employs web applications or mobile applications to allow its customers to interact directly with the enterprise's database, and, thus, with the enterprise itself.

The user, or customer, can focus on the product or service without being aware of the details of the large database that makes the interaction possible. For instance, when you read a social-media post, or access an online bookstore and browse a book or music collection, you are accessing data stored in a database. When you enter an order online, your order is stored in a database. When you access a bank web site and retrieve your bank balance and transaction information, the information is retrieved from the bank's database system. When you access a web site, information about you may be retrieved from a database to select which advertisements you should see. Almost every interaction with a smartphone results in some sort of database access. Furthermore, data about your web accesses may be stored in a database.

Thus, although user interfaces hide details of access to a database, and most people are not even aware they are dealing with a database, accessing databases forms an essential part of almost everyone's life today.

Broadly speaking, there are two modes in which databases are used.

- The first mode is to support **online transaction processing**, where a large number of users use the database, with each user retrieving relatively small amounts of data, and performing small updates. This is the primary mode of use for the vast majority of users of database applications such as those that we outlined earlier.
- The second mode is to support **data analytics**, that is, the processing of data to draw conclusions, and infer rules or decision procedures, which are then used to drive business decisions.

  For example, banks need to decide whether to give a loan to a loan applicant, online advertisers need to decide which advertisement to show to a particular user. These tasks are addressed in two steps. First, data-analysis techniques attempt to automatically discover rules and patterns from data and create *predictive models*. These models take as input attributes ("features") of individuals, and output pre-

dictions such as likelihood of paying back a loan, or clicking on an advertisement, which are then used to make the business decision.

As another example, manufacturers and retailers need to make decisions on what items to manufacture or order in what quantities; these decisions are driven significantly by techniques for analyzing past data, and predicting trends. The cost of making wrong decisions can be very high, and organizations are therefore willing to invest a lot of money to gather or purchase required data, and build systems that can use the data to make accurate predictions.

The field of *data mining* combines knowledge-discovery techniques invented by artificial intelligence researchers and statistical analysts with efficient implementation techniques that enable them to be used on extremely large databases.

1.2 Purpose of Database Systems

To understand the purpose of database systems, consider part of a university organization that, among other data, keeps information about all instructors, students, departments, and course offerings. One way to keep the information on a computer is to store it in operating-system files. To allow users to manipulate the information, the system has a number of application programs that manipulate the files, including programs to:

- Add new students, instructors, and courses.
- Register students for courses and generate class rosters.
- Assign grades to students, compute grade point averages (GPA), and generate transcripts.

Programmers develop these application programs to meet the needs of the university.

New application programs are added to the system as the need arises. For example, suppose that a university decides to create a new major. As a result, the university creates a new department and creates new permanent files (or adds information to existing files) to record information about all the instructors in the department, students in that major, course offerings, degree requirements, and so on. The university may have to write new application programs to deal with rules specific to the new major. New application programs may also have to be written to handle new rules in the university. Thus, as time goes by, the system acquires more files and more application programs.

This typical **file-processing system** is supported by a conventional operating system. The system stores permanent records in various files, and it needs different application programs to extract records from, and add records to, the appropriate files.

Keeping organizational information in a file-processing system has a number of major disadvantages:

- **Data redundancy and inconsistency**. Since different programmers create the files and application programs over a long period, the various files are likely to have different structures, and the programs may be written in several programming languages. Moreover, the same information may be duplicated in several places (files). For example, if a student has a double major (say, music and mathematics), the address and telephone number of that student may appear in a file that consists of student records of students in the Music department and in a file that consists of student records of students in the Mathematics department. This redundancy leads to higher storage and access cost. In addition, it may lead to **data inconsistency**; that is, the various copies of the same data may no longer agree. For example, a changed student address may be reflected in the Music department records but not elsewhere in the system.

- **Difficulty in accessing data**. Suppose that one of the university clerks needs to find out the names of all students who live within a particular postal-code area. The clerk asks the data-processing department to generate such a list. Because the designers of the original system did not anticipate this request, there is no application program on hand to meet it. There is, however, an application program to generate the list of *all* students. The university clerk now has two choices: either obtain the list of all students and extract the needed information manually or ask a programmer to write the necessary application program. Both alternatives are obviously unsatisfactory. Suppose that such a program is written and that, several days later, the same clerk needs to trim that list to include only those students who have taken at least 60 credit hours. As expected, a program to generate such a list does not exist. Again, the clerk has the preceding two options, neither of which is satisfactory.

  The point here is that conventional file-processing environments do not allow needed data to be retrieved in a convenient and efficient manner. More responsive data-retrieval systems are required for general use.

- **Data isolation**. Because data are scattered in various files, and files may be in different formats, writing new application programs to retrieve the appropriate data is difficult.

- **Integrity problems**. The data values stored in the database must satisfy certain types of **consistency constraints**. Suppose the university maintains an account for each department, and records the balance amount in each account. Suppose also that the university requires that the account balance of a department may never fall below zero. Developers enforce these constraints in the system by adding appropriate code in the various application programs. However, when new constraints are added, it is difficult to change the programs to enforce them. The problem is compounded when constraints involve several data items from different files.

- **Atomicity problems**. A computer system, like any other device, is subject to failure. In many applications, it is crucial that, if a failure occurs, the data be restored to the

consistent state that existed prior to the failure. Consider a banking system with a program to transfer $500 from account *A* to account *B*. If a system failure occurs during the execution of the program, it is possible that the $500 was removed from the balance of account *A* but was not credited to the balance of account *B*, resulting in an inconsistent database state. Clearly, it is essential to database consistency that either both the credit and debit occur, or that neither occur. That is, the funds transfer must be *atomic*—it must happen in its entirety or not at all. It is difficult to ensure atomicity in a conventional file-processing system.

- **Concurrent-access anomalies**. For the sake of overall performance of the system and faster response, many systems allow multiple users to update the data simultaneously. Indeed, today, the largest internet retailers may have millions of accesses per day to their data by shoppers. In such an environment, interaction of concurrent updates is possible and may result in inconsistent data. Consider account *A*, with a balance of $10,000. If two bank clerks debit the account balance (by say $500 and $100, respectively) of account *A* at almost exactly the same time, the result of the concurrent executions may leave the account balance in an incorrect (or inconsistent) state. Suppose that the programs executing on behalf of each withdrawal read the old balance, reduce that value by the amount being withdrawn, and write the result back. If the two programs run concurrently, they may both read the value $10,000, and write back $9500 and $9900, respectively. Depending on which one writes the value last, the balance of account *A* may contain either $9500 or $9900, rather than the correct value of $9400. To guard against this possibility, the system must maintain some form of supervision. But supervision is difficult to provide because data may be accessed by many different application programs that have not been coordinated previously.

  As another example, suppose a registration program maintains a count of students registered for a course in order to enforce limits on the number of students registered. When a student registers, the program reads the current count for the courses, verifies that the count is not already at the limit, adds one to the count, and stores the count back in the database. Suppose two students register concurrently, with the count at 39. The two program executions may both read the value 39, and both would then write back 40, leading to an incorrect increase of only 1, even though two students successfully registered for the course and the count should be 41. Furthermore, suppose the course registration limit was 40; in the above case both students would be able to register, leading to a violation of the limit of 40 students.

- **Security problems**. Not every user of the database system should be able to access all the data. For example, in a university, payroll personnel need to see only that part of the database that has financial information. They do not need access to information about academic records. But since application programs are added to the file-processing system in an ad hoc manner, enforcing such security constraints is difficult.

These difficulties, among others, prompted both the initial development of database systems and the transition of file-based applications to database systems, back in the 1960s and 1970s.

In what follows, we shall see the concepts and algorithms that enable database systems to solve the problems with file-processing systems. In most of this book, we use a university organization as a running example of a typical data-processing application.

1.3 View of Data

A database system is a collection of interrelated data and a set of programs that allow users to access and modify these data. A major purpose of a database system is to provide users with an *abstract* view of the data. That is, the system hides certain details of how the data are stored and maintained.

1.3.1 Data Models

Underlying the structure of a database is the **data model**: a collection of conceptual tools for describing data, data relationships, data semantics, and consistency constraints.

There are a number of different data models that we shall cover in the text. The data models can be classified into four different categories:

- **Relational Model**. The relational model uses a collection of tables to represent both data and the relationships among those data. Each table has multiple columns, and each column has a unique name. Tables are also known as **relations**. The relational model is an example of a record-based model. Record-based models are so named because the database is structured in fixed-format records of several types. Each table contains records of a particular type. Each record type defines a fixed number of fields, or attributes. The columns of the table correspond to the attributes of the record type. The relational data model is the most widely used data model, and a vast majority of current database systems are based on the relational model. Chapter 2 and Chapter 7 cover the relational model in detail.
- **Entity-Relationship Model**. The entity-relationship (E-R) data model uses a collection of basic objects, called *entities*, and *relationships* among these objects. An entity is a "thing" or "object" in the real world that is distinguishable from other objects. The entity-relationship model is widely used in database design. Chapter 6 explores it in detail.
- **Semi-structured Data Model**. The semi-structured data model permits the specification of data where individual data items of the same type may have different sets of attributes. This is in contrast to the data models mentioned earlier, where every data item of a particular type must have the same set of attributes. *JSON* and *Extensible Markup Language* (*XML*) are widely used semi-structured data representations. Semi-structured data models are explored in detail in Chapter 8.

- **Object-Based Data Model**. Object-oriented programming (especially in Java, C++, or C#) has become the dominant software-development methodology. This led initially to the development of a distinct object-oriented data model, but today the concept of objects is well integrated into relational databases. Standards exist to store objects in relational tables. Database systems allow procedures to be stored in the database system and executed by the database system. This can be seen as extending the relational model with notions of encapsulation, methods, and object identity. Object-based data models are summarized in Chapter 8.

A large portion of this text is focused on the relational model because it serves as the foundation for most database applications.

1.3.2 Relational Data Model

In the relational model, data are represented in the form of tables. Each table has multiple columns, and each column has a unique name. Each row of the table represents one piece of information. Figure 1.1 presents a sample relational database comprising two tables: one shows details of university instructors and the other shows details of the various university departments.

The first table, the *instructor* table, shows, for example, that an instructor named Einstein with *ID* 22222 is a member of the Physics department and has an annual salary of $95,000. The second table, *department*, shows, for example, that the Biology department is located in the Watson building and has a budget of $90,000. Of course, a real-world university would have many more departments and instructors. We use small tables in the text to illustrate concepts. A larger example for the same schema is available online.

1.3.3 Data Abstraction

For the system to be usable, it must retrieve data efficiently. The need for efficiency has led database system developers to use complex data structures to represent data in the database. Since many database-system users are not computer trained, developers hide the complexity from users through several levels of **data abstraction**, to simplify users' interactions with the system:

- **Physical level**. The lowest level of abstraction describes *how* the data are actually stored. The physical level describes complex low-level data structures in detail.
- **Logical level**. The next-higher level of abstraction describes *what* data are stored in the database, and what relationships exist among those data. The logical level thus describes the entire database in terms of a small number of relatively simple structures. Although implementation of the simple structures at the logical level may involve complex physical-level structures, the user of the logical level does not need to be aware of this complexity. This is referred to as **physical data indepen-**