Database System Implementation

Hector Garcia-Molina
Jeffrey D. Ullman
Jennifer Widom

Department of Computer Science
Stanford University

An Alan R Apt Book

Prentice Hall
Upper Saddle River, New Jersey 07458

Preface

This book was designed for CS245, the second course in the database sequence at Stanford. Here, the first database course, CS145, covers database design and programming, for which the book *A First Course in Database Systems* by Jeff Ullman and Jennifer Widom, Prentice-Hall, 1997, was written. The CS245 course then covers implementation of a DBMS, notably storage structures, query processing, and transaction management.

Use of the Book

We're on a quarter system at Stanford, so the principal course using this book — CS245 — is only ten weeks long. In the Winter of 1999, Hector Garcia-Molina used a "beta" version of this book, and covered the following parts: Sections 2.1-2.4, all of Chapters 3 and 4, Sections 5.1 and 5.2, Sections 6.1-6.7, Sections 7.1–7.4, all of Chapter 8, Chapter 9 except for Section 9.8, Sections 10.1-10.3, Section 11.1, and Section 11.5.

The balance of Chapters 6 and 7 (query optimization) is covered in an advanced course, CS346, where students implement their own DBMS. Other portions of the book that are not covered in CS245 may appear in another advanced course, CS347, which talks about distributed databases and advanced transaction processing.

Schools that are on the semester system have the opportunity to combine the use of this book with its predecessor: *A First Course in Database Systems.* We recommend using that book in the first semester, coupled with a database-application programming project. The second semester could cover most or all of the content of this book. An advantage to splitting the study of databases into two courses is that students not planning to specialize in DBMS construction can take only the first course and be able to use databases in whatever branch of Computer Science they enter.

Prerequisites

The course on which the book is based is rarely taken before the senior year, so we expect the reader to have a fairly broad background in the traditional areas of Computer Science. We assume that the reader has learned something about database programming, especially SQL. It is helpful to know about relational algebra and to have some familiarity with basic data structures. Likewise, some knowledge of file systems and operating systems is useful.

Exercises

The book contains extensive exercises, with some for almost every section. We indicate harder exercises or parts of exercises with an exclamation point. The hardest exercises have a double exclamation point.

Some of the exercises or parts are marked with a star. For these exercises, we shall endeavor to maintain solutions accessible through the book's Web page. These solutions are publicly available and should be used for self-testing. Note that in a few cases, one exercise *B* asks for modification or adaptation of your solution to another exercise *A*. If certain parts of *A* have Web-published solutions, then you should expect the corresponding parts of *B* to have solutions as well.

Support on the World-Wide Web

The book's home page is

`http://www-db.stanford.edu/~ullman/dbsi.html`

Here you will find solutions to starred exercises, errata as we learn of them, and backup materials. We hope to make available the notes for each offering of CS245 and relevant portions of other database courses, as we teach them, including homeworks, exams, and solutions.

Acknowledgements

Thanks go to Brad Adelberg, Karen Butler, Ed Chang, Surajit Chaudhuri, Rada Chirkova, Tom Dienstbier, Xavier Faz, Tracy Fujieda, Luis Gravano, Ben Holzman, Fabien Modoux, Peter Mork, Ken Ross, Mema Roussopolous, and Jonathan Ullman for assistance gathering material and/or discovering errors in earlier drafts of this work. Remaining errors are ours, of course.

H. G.-M.
J. D. U.
J. W.
Stanford, CA

Table of Contents

1 Introduction to DBMS Implementation **1**
1.1 Introducing: The Megatron 2000 Database System 2
1.1.1 Megatron 2000 Implementation Details 2
1.1.2 How Megatron 2000 Executes Queries 4
1.1.3 What's Wrong With Megatron 2000? 5
1.2 Overview of a Database Management System 6
1.2.1 Data-Definition Language Commands 6
1.2.2 Overview of Query Processing 8
1.2.3 Main-Memory Buffers and the Buffer Manager 8
1.2.4 Transaction Processing 9
1.2.5 The Query Processor 10
1.3 Outline of This Book 11
1.3.1 Prerequisites 11
1.3.2 Storage-Management Overview 12
1.3.3 Query-Processing Overview 13
1.3.4 Transaction-Processing Overview 13
1.3.5 Information Integration Overview 13
1.4 Review of Database Models and Languages 14
1.4.1 Relational Model Review 14
1.4.2 SQL Review 15
1.4.3 Relational and Object-Oriented Data 18
1.5 Summary of Chapter 1 19
1.6 References for Chapter 1 20

2 Data Storage **21**
2.1 The Memory Hierarchy 22
2.1.1 Cache 22
2.1.2 Main Memory 23
2.1.3 Virtual Memory 24
2.1.4 Secondary Storage 25
2.1.5 Tertiary Storage 27
2.1.6 Volatile and Nonvolatile Storage 28
2.1.7 Exercises for Section 2.1 29

2.2 Disks 30
2.2.1 Mechanics of Disks 30
2.2.2 The Disk Controller 32
2.2.3 Disk Storage Characteristics 32
2.2.4 Disk Access Characteristics 34
2.2.5 Writing Blocks 38
2.2.6 Modifying Blocks 39
2.2.7 Exercises for Section 2.2 39
2.3 Using Secondary Storage Effectively 40
2.3.1 The I/O Model of Computation 41
2.3.2 Sorting Data in Secondary Storage 42
2.3.3 Merge-Sort 43
2.3.4 Two-Phase, Multiway Merge-Sort 44
2.3.5 Extension of Multiway Merging to Larger Relations . . . 47
2.3.6 Exercises for Section 2.3 48
2.4 Improving the Access Time of Secondary Storage 49
2.4.1 Organizing Data by Cylinders 51
2.4.2 Using Multiple Disks 52
2.4.3 Mirroring Disks 53
2.4.4 Disk Scheduling and the Elevator Algorithm 54
2.4.5 Prefetching and Large-Scale Buffering 58
2.4.6 Summary of Strategies and Tradeoffs 59
2.4.7 Exercises for Section 2.4 61
2.5 Disk Failures 63
2.5.1 Intermittent Failures 63
2.5.2 Checksums 64
2.5.3 Stable Storage 65
2.5.4 Error-Handling Capabilities of Stable Storage 66
2.5.5 Exercises for Section 2.5 67
2.6 Recovery from Disk Crashes 67
2.6.1 The Failure Model for Disks 67
2.6.2 Mirroring as a Redundancy Technique 68
2.6.3 Parity Blocks 69
2.6.4 An Improvement: RAID 5 73
2.6.5 Coping With Multiple Disk Crashes 73
2.6.6 Exercises for Section 2.6 77
2.7 Summary of Chapter 2 80
2.8 References for Chapter 2 82

3 Representing Data Elements **83**
3.1 Data Elements and Fields 83
3.1.1 Representing Relational Database Elements 84
3.1.2 Representing Objects 85
3.1.3 Representing Data Elements 86
3.2 Records 90

3.2.1 Building Fixed-Length Records 91
3.2.2 Record Headers 93
3.2.3 Packing Fixed-Length Records into Blocks 94
3.2.4 Exercises for Section 3.2 95
3.3 Representing Block and Record Addresses 96
3.3.1 Client-Server Systems 97
3.3.2 Logical and Structured Addresses 98
3.3.3 Pointer Swizzling 99
3.3.4 Returning Blocks to Disk 104
3.3.5 Pinned Records and Blocks 105
3.3.6 Exercises for Section 3.3 105
3.4 Variable-Length Data and Records 108
3.4.1 Records With Variable-Length Fields 108
3.4.2 Records With Repeating Fields 109
3.4.3 Variable-Format Records 111
3.4.4 Records That Do Not Fit in a Block 112
3.4.5 BLOBS 114
3.4.6 Exercises for Section 3.4 115
3.5 Record Modifications 116
3.5.1 Insertion 116
3.5.2 Deletion 118
3.5.3 Update 119
3.5.4 Exercises for Section 3.5 119
3.6 Summary of Chapter 3 120
3.7 References for Chapter 3 122

4 Index Structures 123
4.1 Indexes on Sequential Files 124
4.1.1 Sequential Files 124
4.1.2 Dense Indexes 125
4.1.3 Sparse Indexes 128
4.1.4 Multiple Levels of Index 129
4.1.5 Indexes With Duplicate Search Keys 131
4.1.6 Managing Indexes During Data Modifications 133
4.1.7 Exercises for Section 4.1 140
4.2 Secondary Indexes 142
4.2.1 Design of Secondary Indexes 142
4.2.2 Applications of Secondary Indexes 144
4.2.3 Indirection in Secondary Indexes 145
4.2.4 Document Retrieval and Inverted Indexes 148
4.2.5 Exercises for Section 4.2 151
4.3 B-Trees 154
4.3.1 The Structure of B-trees 154
4.3.2 Applications of B-trees 157
4.3.3 Lookup in B-Trees 159

4.3.4 Range Queries 160
4.3.5 Insertion Into B-Trees 161
4.3.6 Deletion From B-Trees 163
4.3.7 Efficiency of B-Trees 166
4.3.8 Exercises for Section 4.3 167
4.4 Hash Tables 170
4.4.1 Secondary-Storage Hash Tables 171
4.4.2 Insertion Into a Hash Table 172
4.4.3 Hash-Table Deletion 172
4.4.4 Efficiency of Hash Table Indexes 173
4.4.5 Extensible Hash Tables 174
4.4.6 Insertion Into Extensible Hash Tables 175
4.4.7 Linear Hash Tables 177
4.4.8 Insertion Into Linear Hash Tables 180
4.4.9 Exercises for Section 4.4 182
4.5 Summary of Chapter 4 184
4.6 References for Chapter 4 185

5 Multidimensional Indexes 187
5.1 Applications Needing Multiple Dimensions 188
5.1.1 Geographic Information Systems 188
5.1.2 Data Cubes 189
5.1.3 Multidimensional Queries in SQL 190
5.1.4 Executing Range Queries Using Conventional Indexes . . 192
5.1.5 Executing Nearest-Neighbor Queries Using Conventional Indexes 193
5.1.6 Other Limitations of Conventional Indexes 195
5.1.7 Overview of Multidimensional Index Structures 195
5.1.8 Exercises for Section 5.1 196
5.2 Hash-Like Structures for Multidimensional Data 197
5.2.1 Grid Files 198
5.2.2 Lookup in a Grid File 198
5.2.3 Insertion Into Grid Files 199
5.2.4 Performance of Grid Files 201
5.2.5 Partitioned Hash Functions 204
5.2.6 Comparison of Grid Files and Partitioned Hashing 205
5.2.7 Exercises for Section 5.2 206
5.3 Tree-Like Structures for Multidimensional Data 209
5.3.1 Multiple-Key Indexes 209
5.3.2 Performance of Multiple-Key Indexes 211
5.3.3 *kd*-Trees 212
5.3.4 Operations on *kd*-Trees 213
5.3.5 Adapting *kd*-Trees to Secondary Storage 216
5.3.6 Quad Trees 217
5.3.7 R-Trees 219

5.3.8 Operations on R-trees 219
5.3.9 Exercises for Section 5.3 222
5.4 Bitmap Indexes 225
5.4.1 Motivation for Bitmap Indexes 225
5.4.2 Compressed Bitmaps 227
5.4.3 Operating on Run-Length-Encoded Bit-Vectors 229
5.4.4 Managing Bitmap Indexes 230
5.4.5 Exercises for Section 5.4 232
5.5 Summary of Chapter 5 233
5.6 References for Chapter 5 234

6 Query Execution **237**
6.1 An Algebra for Queries 240
6.1.1 Union, Intersection, and Difference 241
6.1.2 The Selection Operator 242
6.1.3 The Projection Operator 244
6.1.4 The Product of Relations 245
6.1.5 Joins 246
6.1.6 Duplicate Elimination 248
6.1.7 Grouping and Aggregation 248
6.1.8 The Sorting Operator 251
6.1.9 Expression Trees 252
6.1.10 Exercises for Section 6.1 254
6.2 Introduction to Physical-Query-Plan Operators 257
6.2.1 Scanning Tables 257
6.2.2 Sorting While Scanning Tables 258
6.2.3 The Model of Computation for Physical Operators 258
6.2.4 Parameters for Measuring Costs 259
6.2.5 I/O Cost for Scan Operators 260
6.2.6 Iterators for Implementation of Physical Operators 261
6.3 One-Pass Algorithms for Database Operations 264
6.3.1 One-Pass Algorithms for Tuple-at-a-Time Operations . . 266
6.3.2 One-Pass Algorithms for Unary, Full-Relation Operations 267
6.3.3 One-Pass Algorithms for Binary Operations 270
6.3.4 Exercises for Section 6.3 273
6.4 Nested-Loop Joins 274
6.4.1 Tuple-Based Nested-Loop Join 275
6.4.2 An Iterator for Tuple-Based Nested-Loop Join 275
6.4.3 A Block-Based Nested-Loop Join Algorithm 275
6.4.4 Analysis of Nested-Loop Join 278
6.4.5 Summary of Algorithms so Far 278
6.4.6 Exercises for Section 6.4 278
6.5 Two-Pass Algorithms Based on Sorting 279
6.5.1 Duplicate Elimination Using Sorting 280
6.5.2 Grouping and Aggregation Using Sorting 282

6.5.3 A Sort-Based Union Algorithm 283
6.5.4 Sort-Based Algorithms for Intersection and Difference . . 284
6.5.5 A Simple Sort-Based Join Algorithm 286
6.5.6 Analysis of Simple Sort-Join 287
6.5.7 A More Efficient Sort-Based Join 288
6.5.8 Summary of Sort-Based Algorithms 289
6.5.9 Exercises for Section 6.5 289
6.6 Two-Pass Algorithms Based on Hashing 291
6.6.1 Partitioning Relations by Hashing 292
6.6.2 A Hash-Based Algorithm for Duplicate Elimination . . . 293
6.6.3 A Hash-Based Algorithm for Grouping and Aggregation . 293
6.6.4 Hash-Based Algorithms for Union, Intersection, and Difference 294
6.6.5 The Hash-Join Algorithm 294
6.6.6 Saving Some Disk I/O's 295
6.6.7 Summary of Hash-Based Algorithms 297
6.6.8 Exercises for Section 6.6 298
6.7 Index-Based Algorithms 299
6.7.1 Clustering and Nonclustering Indexes 299
6.7.2 Index-Based Selection 300
6.7.3 Joining by Using an Index 303
6.7.4 Joins Using a Sorted Index 304
6.7.5 Exercises for Section 6.7 306
6.8 Buffer Management 307
6.8.1 Buffer Management Architecture 307
6.8.2 Buffer Management Strategies 308
6.8.3 The Relationship Between Physical Operator Selection and Buffer Management 310
6.8.4 Exercises for Section 6.8 312
6.9 Algorithms Using More Than Two Passes 3 1 3
6.9.1 Multipass Sort-Based Algorithms 313
6.9.2 Performance of Multipass, Sort-Based Algorithms 314
6.9.3 Multipass Hash-Based Algorithms 315
6.9.4 Performance of Multipass Hash-Based Algorithms 315
6.9.5 Exercises for Section 6.9 316
6.10 Parallel Algorithms for Relational Operations 317
6.10.1 Models of Parallelism 317
6.10.2 Tuple-at-a-Time Operations in Parallel 320
640.3 Parallel Algorithms for Full-Relation Operations 321
6.10.4 Performance of Parallel Algorithms 322
6.10.5 Exercises for Section 6.10 324
6.11 Summary of Chapter 6 . . . , ... 325
6.12 References for Chapter 6 327

7 The Query Compiler 329
7.1 Parsing 330
7.1.1 Syntax Analysis and Parse Trees 330
7.1.2 A Grammar for a Simple Subset of SQL 331
7.1.3 The Preprocessor 336
7.1.4 Exercises for Section 7.1 337
7.2 Algebraic Laws for Improving Query Plans 337
7.2.1 Commutative and Associative Laws 338
7.2.2 Laws Involving Selection 340
7.2.3 Pushing Selections 343
7.2.4 Laws Involving Projection 345
7.2.5 Laws About Joins and Products 348
7.2.6 Laws Involving Duplicate Elimination 348
7.2.7 Laws Involving Grouping and Aggregation 349
7.2.8 Exercises for Section 7.2 351
7.3 From Parse Trees to Logical Query Plans 354
7.3.1 Conversion to Relational Algebra 354
7.3.2 Removing Subqueries From Conditions 355
7.3.3 Improving the Logical Query Plan 362
7.3.4 Grouping Associative/Commutative Operators 364
7.3.5 Exercises for Section 7.3 365
7.4 Estimating the Cost of Operations 366
7.4.1 Estimating Sizes of Intermediate Relations 367
7.4.2 Estimating the Size of a Projection 368
7.4.3 Estimating the Size of a Selection 369
7.4.4 Estimating the Size of a Join 371
7.4.5 Natural Joins With Multiple Join Attributes 374
7.4.6 Joins of Many Relations 375
7.4.7 Estimating Sizes for Other Operations 378
7.4.8 Exercises for Section 7.4 379
7.5 Introduction to Cost-Based Plan Selection 380
7.5.1 Obtaining Estimates for Size Parameters 381
7.5.2 Incremental Computation of Statistics 384
7.5.3 Heuristics for Reducing the Cost of Logical Query Plans . 385
7.5.4 Approaches to Enumerating Physical Plans 388
7.5.5 Exercises for Section 7.5 391
7.6 Choosing an Order for Joins 393
7.6.1 Significance of Left and Right Join Arguments 393
7.6.2 Join Trees 394
7.6.3 Left-Deep Join Trees 395
7.6.4 Dynamic Programming to Select a Join Order and Grouping 398
7.6.5 Dynamic Programming With More Detailed Cost Functions 402
7.6.6 A Greedy Algorithm for Selecting a Join Order 403
7.6.7 Exercises for Section 7.6 404
7.7 Completing the Physical-Query-Plan Selection 406

7.7.1 Choosing a Selection Method 406
7.7.2 Choosing a Join Method 409
7.7.3 Pipelining Versus Materialization 409
7.7.4 Pipelining Unary Operations 410
7.7.5 Pipelining Binary Operations 411
7.7.6 Notation for Physical Query Plans 414
7.7.7 Ordering of Physical Operations 417
7.7.8 Exercises for Section 7.7 418
7.8 Summary of Chapter 7 419
7.9 References for Chapter 7 421

8 Coping With System Failures 423
8.1 Issues and Models for Resilient Operation 424
8.1.1 Failure Modes 424
8.1.2 More About Transactions 426
8.1.3 Correct Execution of Transactions 427
8.1.4 The Primitive Operations of Transactions 429
8.1.5 Exercises for Section 8.1 432
8.2 Undo Logging 432
8.2.1 Log Records 433
8.2.2 The Undo-Logging Rules 434
8.2.3 Recovery Using Undo Logging 436
8.2.4 Checkpointing 439
8.2.5 Nonquiescent Checkpointing 440
8.2.6 Exercises for Section 8.2 444
8.3 Redo Logging 445
8.3.1 The Redo-Logging Rule 446
8.3.2 Recovery With Redo Logging 447
8.3.3 Checkpointing a Redo Log 448
8.3.4 Recovery With a Checkpointed Redo Log 450
8.3.5 Exercises for Section 8.3 451
8.4 Undo/Redo Logging 451
8.4.1 The Undo/Redo Rules 452
8.4.2 Recovery With Undo/Redo Logging 453
8.4.3 Checkpointing an Undo/Redo Log 454
8.4.4 Exercises for Section 8.4 456
8.5 Protecting Against Media Failures 457
8.5.1 The Archive 458
8.5.2 Nonquiescent Archiving 459
8.5.3 Recovery Using an Archive and Log 461
8.5.4 Exercises for Section 8.5 462
8.6 Summary of Chapter 8 462
8.7 References for Chapter 8 464

9 Concurrency Control 467
9.1 Serial and Serializable Schedules 468
9.1.1 Schedules 468
9.1.2 Serial Schedules 469
9.1.3 Serializable Schedules 470
9.1.4 The Effect of Transaction Semantics 471
9.1.5 A Notation for Transactions and Schedules 473
9.1.6 Exercises for Section 9.1 474
9.2 Conflict-Serializability 475
9.2.1 Conflicts 475
9.2.2 Precedence Graphs and a Test for Conflict-Serializability 476
9.2.3 Why the Precedence-Graph Test Works 479
9.2.4 Exercises for Section 9.2 481
9.3 Enforcing Serializability by Locks 483
9.3.1 Locks 483
9.3.2 The Locking Scheduler 485
9.3.3 Two-Phase Locking 486
9.3.4 Why Two-Phase Locking Works 487
9.3.5 Exercises for Section 9.3 488
9.4 Locking Systems With Several Lock Modes 490
9.4.1 Shared and Exclusive Locks 491
9.4.2 Compatibility Matrices 493
9.4.3 Upgrading Locks 494
9.4.4 Update Locks 495
9.4.5 Increment Locks 497
9.4.6 Exercises for Section 9.4 499
9.5 An Architecture for a Locking Scheduler 502
9.5.1 A Scheduler That Inserts Lock Actions 502
9.5.2 The Lock Table 504
9.5.3 Exercises for Section 9.5 507
9.6 Managing Hierarchies of Database Elements 508
9.6.1 Locks With Multiple Granularity 508
9.6.2 Warning Locks 509
9.6.3 Phantoms and Handling Insertions Correctly 512
9.6.4 Exercises for Section 9.6 514
9.7 The Tree Protocol 514
9.7.1 Motivation for Tree-Based Locking 514
9.7.2 Rules for Access to Tree-Structured Data 515
9.7.3 Why the Tree Protocol Works 516
9.7.4 Exercises for Section 9.7 520
9.8 Concurrency Control by Timestamps 521
9.8.1 Timestamps 521
9.8.2 Physically Unrealizable Behaviors 522
9.8.3 Problems With Dirty Data 523
9.8.4 The Rules for Timestamp-Based Scheduling 525

9.8.5 Multiversion Timestamps 527
9.8.6 Timestamps and Locking 528
9.8.7 Exercises for Section 9.8 530
9.9 Concurrency Control by Validation 530
9.9.1 Architecture of a Validation-Based Scheduler 531
9.9.2 The Validation Rules 532
9.9.3 Comparison of Three Concurrency-Control Mechanisms . 535
9.9.4 Exercises for Section 9.9 536
9.10 Summary of Chapter 9 536
9.11 References for Chapter 9 539

10 More About Transaction Management 541
10.1 Transactions that Read Uncommitted Data 541
10.1.1 The Dirty-Data Problem 542
10.1.2 Cascading Rollback 544
10.1.3 Managing Rollbacks 545
10.1.4 Group Commit 546
10.1.5 Logical Logging 548
10.1.6 Exercises for Section 10.1 551
10.2 View Serializability 552
10.2.1 View Equivalence 552
10.2.2 Polygraphs and the Test for View-Serializability 553
10.2.3 Testing for View-Serializability 556
10.2.4 Exercises for Section 10.2 557
10.3 Resolving Deadlocks 558
10.3.1 Deadlock Detection by Timeout 558
10.3.2 The Waits-For Graph 559
10.3.3 Deadlock Prevention by Ordering Elements 561
10.3.4 Detecting Deadlocks by Timestamps 563
10.3.5 Comparison of Deadlock-Management Methods 566
10.3.6 Exercises for Section 10.3 566
10.4 Distributed Databases 568
10.4.1 Distribution of Data 568
10.4.2 Distributed Transactions 570
10.4.3 Data Replication 570
10.4.4 Distributed Query Optimization 571
10.4.5 Exercises for Section 10.4 572
10.5 Distributed Commit 572
10.5.1 Supporting Distributed Atomicity 573
10.5.2 Two-Phase Commit 573
10.5.3 Recovery of Distributed Transactions 576
10.5.4 Exercises for Section 10.5 578
10.6 Distributed Locking 579
10.6.1 Centralized Lock Systems 579
10.6.2 A Cost Model for Distributed Locking Algorithms 579

10.6.3 Locking Replicated Elements 581
10.6.4 Primary-Copy Locking 581
10.6.5 Global Locks From Local Locks 582
10.6.6 Exercises for Section 10.6 584
10.7 Long-Duration Transactions 584
10.7.1 Problems of Long Transactions 585
10.7.2 Sagas 587
10.7.3 Compensating Transactions 588
10.7.4 Why Compensating Transactions Work 590
10.7.5 Exercises for Section 10.7 590
10.8 Summary of Chapter 10 591
10.9 References for Chapter 10 593

11 Information Integration 595
11.1 Modes of Information Integration 595
11.1.1 Problems of Information Integration 596
11.1.2 Federated Database Systems 597
11.1.3 Data Warehouses 599
11.1.4 Mediators 601
11.1.5 Exercises for Section 11.1 604
11.2 Wrappers in Mediator-Based Systems 605
11.2.1 Templates for Query Patterns 606
11.2.2 Wrapper Generators 607
11.2.3 Filters 608
11.2.4 Other Operations at the Wrapper 610
11.2.5 Exercises for Section 11.2 611
11.3 On-Line Analytic Processing 612
11.3.1 OLAP Applications 613
11.3.2 A Multidimensional View of OLAP Data 614
11.3.3 Star Schemas 615
11.3.4 Slicing and Dicing 618
11.3.5 Exercises for Section 11.3 620
11.4 Data Cubes 621
11.4.1 The Cube Operator 622
11.4.2 Cube Implementation by Materialized Views 625
11.4.3 The Lattice of Views 628
11.4.4 Exercises for Section 11.4 630
11.5 Data Mining 632
11.5.1 Data-Mining Applications 632
11.5.2 Association-Rule Mining 635
11.5.3 The A-Priori Algorithm 636
11.6 Summary of Chapter 11 639
11.7 References for Chapter 11 640

Index 643

About the Authors

Hector Garcia-Molina is the Leonard Bosack and Sandra Lerner Professor in the Computer Science and Electrical Engineering Departments at Stanford University. He has published extensively in the fields of database systems, distributed systems, and digital libraries. His research interests also include distributed computing systems, database systems, and digital libraries.

Jeffrey D. Ullman is the Stanford W. Ascherman Professor of Computer Science at Stanford University. He is the author or co-author of 15 books and 170 technical publications, including *A First Course in Database Systems* (Prentice Hall 1997) and *Elements of ML Programming* (Prentice Hall 1998). His research interests include database theory, database integration, data mining, and education using the information infrastructure. He has received numerous awards such as the Guggenheim Fellowship and election to the National Academy of Engineering. He also received the 1996 Sigmod Contribution Award and the 1998 Karl V. Karstrom Outstanding Educator Award.

Jennifer Widom is an Associate Professor in the Computer Science and Electrical Engineering Departments at Stanford University. She has served on numerous editorial boards and program committees, she has published widely in computer science conferences and journals, and is co-author of *A First Course in Database Systems* (Prentice Hall 1997). Her research interests include database systems for semistructured data and XML, data warehousing, and active database systems.

Database System Implementation Hector Garcia-Molina Jeffrey D. Ullman Jennifer Widom Department of Computer Science Stanford University. An Alan R Apt Book. Prentice Hall Upper Saddle River, New Jersey 07458. Short Description. Download Database Systems Design, Implementation, And Management Eighth Edition-Ch09 Description.Â The Information System (continued) â€¢ Applications â€“ Transform data into information that forms basis for decision making â€“ Usually produce the following: â€¢ Formal report â€¢ Tabulations â€¢ Graphic displays. â€“ Composed of following two parts: â€¢ Data â€¢ Code by which data are transformed into information Database Systems, 8th Edition. 5. Database Systems, 8th Edition. 6.