**Accolades for *Database Administration***

“I’ve forgotten how many times I’ve recommended this book to people. It’s well written, to the point, and covers the topics that you need to know to become an effective DBA.”

—Scott Ambler, Thought Leader, Agile Data Method

“This is a well-written, well-organized guide to the practice of database administration. Unlike other books on general database theory or relational database theory, this book focuses more directly on the theory and reality of database administration as practiced by database professionals today, and does so without catering too much to any specific product implementation. As such, *Database Administration* is very well suited to anyone interested in surveying the job of a DBA or those in similar but more specific roles such as data modeler or database performance analyst.”

—Sal Ricciardi, Program Manager, Microsoft

“One of Craig’s hallmarks is his ability to write in a clear, easy-to-read fashion. The main purpose of any technical book is to transfer information from writer to reader, and Craig has done an excellent job. He wants the reader to learn—and it shows.”

—Chris Foot, Manager, Remote DBA Experts and Oracle ACE

“A complete and comprehensive listing of tasks and responsibilities for DBAs, ranging from creating the database environment to data warehouse administration, and everything in between.”

—Mike Tarrani, Computer Consultant

“I think every business manager and every IT manager should have a copy of this book.”

—Dan Hotka, Independent Consultant and Oracle ACE

“This book by Craig Mullins is wonderfully insightful and truly important. Mullins describes the role and duties of data administrators and database administrators in modern organizations with remarkable insight and clarity.”

—Michael Tozer, Author and former U.S. Navy officer
This page intentionally left blank
Database Administration
Second Edition
This page intentionally left blank
To my wife, Beth, for her unending love, constant support, and beautiful smile.
Database, Data, and System Administration 15

Data Administration 15
Database Administration 19
System Administration 20

DBA Tasks 20

Database Design 21
Performance Monitoring and Tuning 22
Ensuring Availability 24
Database Security and Authorization 24
Governance and Regulatory Compliance 26
Backup and Recovery 26
Ensuring Data Integrity 27

DBMS Release Migration 29

Jack-of-All-Trades 29

The Types of DBAs 31

System DBA 31
Database Architect 32
Database Analyst 33
Data Modeler 33
Application DBA 34
Task-Oriented DBA 36
Performance Analyst 36
Data Warehouse Administrator 36

Staffing Considerations 37

How Many DBAs? 37

DBA Reporting Structures 40

Multiplatform DBA Issues 42

Production versus Test 44

The Impact of Newer Technology on DBA 46

Procedural DBAs: Managing Database Logic 46
Contents

The Internet: From DBA to eDBA  50
The Personal DBA and the Cloud  53
NoSQL, Big Data, and the DBA  55
New Technology Impacts on DBA  56

DBA Certification  56
The Rest of the Book  58
Review  58
  Bonus Question  59

Chapter 2  Creating the Database Environment  61

Defining the Organization’s DBMS Strategy  61
  Choosing a DBMS  63
  DBMS Architectures  68
  DBMS Clustering  71
  DBMS Proliferation  73
  Hardware Issues  73
  Cloud Database Systems  74

Installing the DBMS  75
  DBMS Installation Basics  75
  Hardware Requirements  76
  Storage Requirements  76
  Memory Requirements  78
  Configuring the DBMS  80
  Connecting the DBMS to Supporting Infrastructure Software  81
  Installation Verification  81
  DBMS Environments  82

Upgrading DBMS Versions and Releases  82
  Features and Complexity  87
  Complexity of the DBMS Environment  87
  Reputation of the DBMS Vendor  89
  Support Policies of the DBMS  89
Contents

Organization Style  89
DBA Staff Skill Set  90
Platform Support  90
Supporting Software  91
Fallback Planning  92
Migration Verification  92
The DBMS Upgrade Strategy  92
Database Standards and Procedures  92
Database Naming Conventions  93
Other Database Standards and Procedures  96
DBMS Education  103
Summary  104
Review  104
Bonus Question  105
Suggested Reading  105

Chapter 3 Data Modeling and Normalization  107
Data Modeling Concepts  108
Entity-Relationship Diagramming  110
The Components of a Data Model  113
Entities  113
Attributes  115
Keys  120
Relationships  122
Discovering Entities, Attributes, and Relationships  124
Conceptual, Logical, and Physical Data Models  125
What Is Normalization?  128
The Normal Forms  128
First Normal Form  129
Second Normal Form  129
Third Normal Form  132
Chapter 4 Database Design  141

From Logical Model to Physical Database  141
   Transform Entities to Tables  142
   Transform Attributes to Columns  142
   Build Referential Constraints for All Relationships  146
   Build Physical Data Structures  147

Database Performance Design  150
   Designing Indexes  150
   Hashing  158
   Clustering  159
   Interleaving Data  160

Denormalization  160
   When to Denormalize  161
   Prejoined Tables  164
   Report Tables  164
   Mirror Tables  165
   Split Tables  165
   Combined Tables  168
   Redundant Data  168
   Repeating Groups  169
   Derivable Data  170
   Hierarchies  171
Chapter 5 Application Design  185

Database Application Development and SQL  186
  SQL  186
  Set-at-a-Time Processing and Relational Closure  189
  Embedding SQL in a Program  191
  SQL Middleware and APIs  192
  Application Infrastructure  193
  Object Orientation and SQL  199
  Types of SQL  200
  SQL Coding for Performance  202
  Querying XML Data  203
  Defining Transactions  205
    Transaction Guidelines  207
    Unit of Work  207
    Transaction Processing Systems  207
    Application Servers  209
  Locking  210
    Types of Locks  212
    Lock Time-outs  213
Chapter 6 Design Reviews 227

What Is a Design Review? 227

Rules of Engagement 228

Design Review Participants 229

Knowledge and Skills Required 232

Types of Design Reviews 232

Conceptual Design Review 233

Logical Design Review 235

Physical Design Review 236

Organizational Design Review 237

SQL and Application Code Design Review 238

Pre-Implementation Design Review 239

Post-Implementation Design Review 239

Design Review Output 239

Additional Considerations 240

Dealing with Remote Staff 240

Mentorship and Knowledge Transfer 240

Summary 241

Review 241

Suggested Reading 242
Chapter 7  Database Change Management  243
Change Management Requirements  244
   The Change Management Perspective of the DBA  246
Types of Changes  247
   DBMS Software  248
   Hardware Configuration  248
   Logical and Physical Design  248
   Applications  249
   Physical Database Structures  250
Impact of Change on Database Structures  250
   The Limitations of ALTER  252
Database Change Scenarios  254
Comparing Database Structures  257
Requesting Database Changes  258
Standardized Change Requests  259
Communication  260
   Coordinating Database and Application Changes  260
Compliance  261
   DBA Scripts and Change Management  262
Summary  262
Review  263
Suggested Reading  263

Chapter 8  Data Availability  265
Defining Availability  267
   Increased Availability Requirements  268
Cost of Downtime  271
   How Much Availability Is Enough?  273
Availability Problems  274
   Loss of the Data Center  274
   Network Problems  275
Contents

Loss of the Server Hardware  276
Disk-Related Outages  278
Operating System Failure  279
DBMS Software Failure  279
Application Problems  279
Security and Authorization Problems  280
Corruption of Data  280
Loss of Database Objects  281
Loss of Data  282
Data Replication and Propagation Failures  283
Severe Performance Problems  283
Recovery Issues  284
DBA Mistakes  284
Outages: Planned and Unplanned  286

Ensuring Availability  287
Perform Routine Maintenance While Systems Remain Operational  288
Automate DBA Functions  290
Exploit High-Availability Features  291
Exploit Clustering Technology  292
Database Architecture and NoSQL  296

Summary  296
Review  297
Suggested Reading  298

Chapter 9  Performance Management  299
Defining Performance  299
A Basic Database Performance Road Map  302
Monitoring versus Management  304
Reactive versus Proactive  306
Contents

Preproduction Performance Estimation  307
Historical Trending  308
Service-Level Management  308
Types of Performance Tuning  311
  System Tuning  311
  Database Tuning  312
  Application Tuning  312
Performance Tuning Tools  313
DBMS Performance Basics  315
Summary  316
Review  316
  Bonus Question  317
Suggested Reading  317

Chapter 10  System Performance  319
The Larger Environment  320
  Interaction with the Operating System  320
Allied Agents  321
Hardware Configuration  322
Components of the DBMS  324
DBMS Installation and Configuration Issues  327
  Types of Configuration  327
Memory Usage  328
Data Cache Details  332
“Open” Database Objects  336
Database Logs  336
Locking and Contention  341
The System Catalog  342
Other Configuration Options  343
General Advice  344
System Monitoring  345
Summary  346
Review  346
  Bonus Question  347
Suggested Reading  347

Chapter 11  Database Performance  349
Techniques for Optimizing Databases  349
  Partitioning  350
  Raw Partition versus File System  351
  Indexing  352
  Denormalization  355
  Clustering  356
  Interleaving Data  360
  Free Space  360
  Compression  361
  File Placement and Allocation  362
  Page Size (Block Size)  364
Database Reorganization  365
  Determining When to Reorganize  369
  Automation  371
Summary  371
Review  371
Suggested Reading  372

Chapter 12  Application Performance  373
Designing Applications for Relational Access  373
Relational Optimization  374
  CPU and I/O Costs  376
  Database Statistics  376
  Query Analysis  378
  Joins  379
  Access Path Choices  381
Chapter 14  Database Security  449
Data Breaches  449
Database Security Basics  451
   Database Users  455
Granting and Revoking Authority  456
   Types of Privileges  457
   Granting to PUBLIC  460
   Revoking Privileges  461
   Label-Based Access Control  463
   Security Reporting  465
Authorization Roles and Groups  466
   Roles  466
   Groups  467
Other Database Security Mechanisms  468
   Using Views for Security  468
   Using Stored Procedures for Security  470
Encryption  470
   Data at Rest Encryption  472
   Data in Transit Encryption  472
   Encryption Techniques  472
SQL Injection  473
   SQL Injection Prevention  475
Auditing  477
External Security  478
   Job Scheduling and Security  479
   Non-DBMS DBA Security  480
DBMS Fixpacks and Maintenance  480
Summary  481
Review  481
Suggested Reading  482
Chapter 15  Regulatory Compliance and Database Administration  483

A Collaborative Approach to Compliance  486
   Why Should DBAs Care about Compliance?  487
Metadata Management, Data Quality, and Data Governance  488
   Metadata  488
   Data Quality  489
   Data Governance  489
Database Auditing and Data Access Tracking  490
   Database Auditing Techniques  493
   Privileged User Auditing  495
Data Masking and Obfuscation  496
   Data Masking Techniques  497
Database Archiving for Long-Term Data Retention  498
   The Life Cycle of Data  499
   Database Archiving  500
   Components of a Database Archiving Solution  505
   The Impact of e-Discovery on DBA  506
Closer Tracking of Traditional DBA Tasks  507
   Database Change Management  508
   Database Backup and Recovery  508
Summary  511
Review  511
Suggested Reading  512

Chapter 16  Database Backup and Recovery  515

The Importance of Backup and Recovery  515
Preparing for Problems  516
Backup  517
   Full versus Incremental Backups  521
   Database Objects and Backups  523
Contents

DBMS Control 524
Concurrent Access Issues 525
Backup Consistency 527
Log Archiving and Backup 529
Determining Your Backup Schedule 531
DBMS Instance Backup 533
Designing the DBMS Environment for Recovery 533
Alternate Approaches to Database Backup 534
Document Your Backup Strategy 536
Database Object Definition Backups 536

Recovery 537
Determining Recovery Options 538
General Steps for Database Object Recovery 540
Types of Recovery 541
Index Recovery 550
Testing Your Recovery Plan 551
Recovering a Dropped Database Object 552
Recovering Broken Blocks and Pages 553
Populating Test Databases 553

Alternatives to Backup and Recovery 554
Standby Databases 554
Replication 555
Disk Mirroring 556

Summary 557
Review 557
Suggested Reading 558

Chapter 17 Disaster Planning 559
The Need for Planning 559
Risk and Recovery 561
Contents

General Disaster Recovery Guidelines  563
  The Remote Site  564
  The Written Plan  564
  Personnel  569

Backing Up the Database for Disaster Recovery  569
  Tape Backups  570
  Storage Management Backups  572
  Other Approaches  573
  Some Guidelines  573

Disaster Prevention  575
  Disaster and Contingency Planning Web Sites  576

Summary  576
Review  576
Suggested Reading  577

Chapter 18  Data and Storage Management  579

Storage Management Basics  579
Files and Data Sets  583
  File Placement on Disk  584
  Raw Partitions versus File Systems  586
  Temporary Database Files  587

Space Management  587
  Data Page Layouts  588
  Index Page Layouts  592
  Transaction Logs  594

Fragmentation and Storage  595

Storage Options  596
  RAID  597
  JBOD  604
  Storage Area Networks  604
Network-Attached Storage 605
Tiered Storage 606
Planning for the Future 608
Capacity Planning 608
Summary 609
Review 609
Suggested Reading 610

Chapter 19 Data Movement and Distribution 613
Loading and Unloading Data 614
The LOAD Utility 614
The UNLOAD Utility 618
Maintaining Application Test Beds 621
EXPORT and IMPORT 622
Bulk Data Movement 623
ETL Software 623
Replication and Propagation 623
Messaging Software 624
Other Methods 625
Distributed Databases 626
Setting Up a Distributed Environment 627
Data Distribution Standards 629
Accessing Distributed Data 630
Two-Phase COMMIT 631
Distributed Performance Problems 632
Summary 633
Review 634
Bonus Question 634
Suggested Reading 635
Chapter 20  Data Warehouse Administration  637

What Is a Data Warehouse?  637
  Analytical versus Transaction Processing  638
Administering the Data Warehouse  640
  Too Much Focus on Technology?  641
Data Warehouse Design  641
Data Movement  644
Data Cleansing  645
Data Warehouse Scalability  649
Data Warehouse Performance  650
Data Freshness  654
Data Content  654
Data Usage  655
  Financial Chargeback  655
Backup and Recovery  656
  Don’t Operate in a Vacuum!  657
Summary  658
Review  658
Suggested Reading  659

Chapter 21  Database Connectivity  661

Multitier, Distributed Computing  661
  A Historical Look  661
  Business Issues  663
What Is Client/Server Computing?  663
  Types of Client/Server Applications  667
Network Traffic  670
  Database Gateways  671
  Database Drivers  672
Connection Pooling  674
Chapter 22  Metadata Management  685
What Is Metadata?  685
From Data to Knowledge and Beyond  686
Metadata Strategy  687
Data Warehousing and Metadata  688
Types of Metadata  689
Repositories and Data Dictionaries  691
Repository Benefits  693
Repository Challenges  693
Data Dictionaries  695
Summary  696
Review  696
Suggested Reading  697

Chapter 23  DBA Tools  699
Types and Benefits of DBA Tools  699
Data Modeling and Design  700
Database Change Management  701
Table Editors  707
Performance Management  708
Backup and Recovery  714
Database Utilities  715
Data Protection, Governance, Risk, and Compliance Tools  716
Data Warehousing, Analytics, and Business Intelligence  721
Programming and Development Tools  724
Miscellaneous Tools  726
Examine Native DBA Tools  728
Evaluating DBA Tool Vendors  729
Homegrown DBA Tools  732
Summary  733
Review  733

Chapter 24  DBA Rules of Thumb  735
Write Down Everything  735
Keep Everything  736
Automate!  737
Share Your Knowledge  739
Analyze, Simplify, and Focus  741
Don't Panic!  742
Measure Twice, Cut Once  743
Understand the Business, Not Just the Technology  743
Don't Become a Hermit  745
Use All of the Resources at Your Disposal  745
Keep Up-to-Date  746
Invest in Yourself  747
Summary  748
Final Exam  748

Appendix A  Database Fundamentals  753
What Is a Database?  753
Why Use a DBMS?  754
Advantages of Using a DBMS  755
Summary  759
Appendix B  The DBMS Vendors  761
   The Big Three  762
   The Second Tier  763
   Other Significant Players  763
   Open-Source DBMS Offerings  764
   Nonrelational DBMS Vendors  765
   NoSQL DBMS Vendors  765
   Object-Oriented DBMS Vendors  766
   PC-Based DBMS Vendors  766

Appendix C  DBA Tool Vendors  769
   The Major Vendors  769
   Other DBA Tool Vendors  770
   Data Modeling Tool Vendors  771
   Repository Vendors  772
   Data Movement and Business Intelligence Vendors  773

Appendix D  DBA Web Resources  775
   Usenet Newsgroups  775
   Mailing Lists  776
   Web Sites, Blogs, and Portals  778
      Vendor Web Sites  778
      Magazine Web Sites  778
      Consultant Web Sites  779
      Blogs  780
      Database Portals  781
      Other Web Sites  782

Appendix E  Sample DBA Job Posting  785
   Job Posting  785
      Database Administrator (DBA)  785
The need for database administration is as strong as, or stronger than, it was when I originally wrote the first edition of this book in 2002. Relational database management systems are still at the core of most serious production systems, and they still need to be managed. And this is still the job of database administrators. Whether you use Oracle, Microsoft SQL Server, DB2, Informix, Sybase, MySQL, Teradata, PostgreSQL, Ingres, or any combination of these popular DBMS products, you will benefit from the information in this book.

But a decade is forever in the world of information technology. And even though some basic things stay the same (e.g., databases require administration), many things change. The second edition of this book incorporates the many changes that impact database administration that have occurred in the industry over the past decade. What made the book unique remains. It is still the industry’s only non-product-based description of database administration techniques and practices. The book defines the job of database administrator and outlines what is required of a database
administrator, or DBA, in clear, easy-to-understand language. The book can be used

- As a text for learning the discipline of database administration
- As the basis for setting up a DBA group
- To augment a DBMS-specific manual or textbook
- To help explain to upper-level management what a DBA is, and why the position is required

But what is new? One of the significant improvements added to this edition is coverage of regulatory compliance. The number of governmental and industry regulations has exploded over the course of the past decade, and many of these regulations dictate changes in the way that data is managed, handled, and processed. Although the most visible governmental regulation is undoubtedly the Sarbanes-Oxley Act (aka the U.S. Public Company Accounting Reform and Investor Protection Act of 2002), there are many others, including HIPAA (the Health Insurance Portability and Accountability Act) and GLB (the Gramm-Leach-Bliley Act) to name a couple. The most visible industry regulation is PCI DSS (Payment Card Industry Data Security Standard). All of these regulations, and many others, impose an additional administrative burden on data. This edition of the book provides an entire chapter devoted to this topic, including the impact of regulatory compliance on data management tasks such as metadata management, data quality, database auditing, data masking, database archiving, and more traditional DBA tasks such as database change management and database recovery.

Database security is another rapidly evolving area that required a significant upgrade from the first edition. Fresh coverage is offered on new security functionality and requirements, including label-based access control, encryption, and preventing SQL injection attacks.

The book adds coverage of technology that was not widely adopted ten years ago, such as XML, and where appropriate it discusses nascent technology that DBAs should be aware of, including NoSQL and cloud computing. It also covers newer DBMS functionality, such as temporal database support and INSTEAD-OF triggers.
Finally, the entire book was reviewed and revised to ensure that each topic addressed up-to-date technology and requirements. Care was taken to ensure that the example DBMS features used to highlight specific technologies are accurate and up-to-date. For example, consider the descriptions of DB2 HADR, SQL Server 2012 AlwaysOn, and Oracle Transparent Data Encryption.

With the second edition of this book you now have a timely, accurate, and updated guide to implementing and maintaining heterogeneous database administration. You can use it to learn what is required to be a successful database administrator. And you can use it on the job in conjunction with the vendors’ manuals or product-specific books for your particular DBMS products.

**How to Use This Book**

This book can be used as both a tutorial and a reference. The book is organized to proceed chronologically through DBA tasks that are likely to be encountered. Therefore, if you read the book sequentially from Chapter 1 through Chapter 24, you will get a comprehensive sequential overview of the DBA job. Alternatively, you can read any chapter independently because each chapter deals with a single topic. References to other chapters are clearly made if other material in the book would aid the reader’s understanding.
This page intentionally left blank
Acknowledgments

Writing is a rewarding task, but it also requires a lot of time—researching, writing, reviewing, editing, and rewriting over and over again until you get it just right. But no one can write a technical book in a vacuum. I had many knowledgeable and helpful people to assist me along the way.

First of all, I’d like to thank the many industry experts who reviewed the original book proposal. The following folks provided many useful suggestions and thoughts on my original outline that helped me to create a much better book: Michael Blaha, Keith W. Hare, Michael J. Hernandez, Robert S. Seiner, and David L. Wells. Additionally, I’d like to thank everyone who took the time to listen to my ideas for this book before I began writing. This list of folks is too numerous to include, and I’m sure I’d miss someone—but you know who you are.

I would like to thank the many folks who have reviewed and commented on the text of this book. For the second edition of the book, Bill Arledge and Kevin Kline provided their expertise to the review process and offered many helpful corrections and suggestions that improved the quality of the book. And let’s not forget the reviewers of the first edition: Dan Hotka, Chris Foot, Chuck Kosin, David L. Wells, and Anne Marie Smith...
pored over each chapter of various incarnations of the manuscript, and this book is much better thanks to their expert contributions. Special thanks go to data modeling and administration gurus William J. Lewis and Robert S. Seiner, who took extra time to review and make suggestions on Chapter 3. I’d also like to thank my brother, Scott Mullins, who offered his guidance on application design and development by reviewing Chapter 5.

My appreciation goes to Mary Barnard, who did a wonderful job editing the first edition of this book; and Greg Doench, who did a similarly fantastic job with the second edition. Kudos to both Mary and Greg for making my book much more readable.

Additionally, thanks to the many understanding and patient folks at Addison-Wesley who worked with me to make each edition of the book come to fruition. This list includes Michelle Housley, Patrick Peterson, Stacie Parillo, Barbara Wood, and Mary O’Brien who were particularly helpful throughout the process of coordinating the production of the book.

Thank you, too, to my wonderful wife, Beth, whose understanding and support made it possible for me to write this book. Indeed, thanks go out to all my family and friends for being supportive and helpful along the way.

And finally, a thank-you to all the people with whom I have worked professionally at SoftwareOnZ, NEON Enterprise Software, Embarcadero Technologies, BMC Software, Gartner Group, PLATINUM Technology, Inc., Duquesne Light Company, Mellon Bank, USX Corporation, and ASSET, Inc. This book is a better one due to the many outstanding individuals with whom I have had the honor to work.
About the Author

Craig S. Mullins is President and Principal Consultant for Mullins Consulting, Inc., a consulting practice specializing in data management and database management systems. Craig has extensive experience in the field of database management, having worked as an application developer, a DBA, and an instructor with multiple database management systems, including DB2, Oracle, and SQL Server. Craig has worked in multiple industries, including manufacturing, banking, commercial software development, education, research, utilities, and consulting. Additionally, Craig worked as a Research Director with Gartner Group, covering the field of database administration. He is the author of *DB2 Developer's Guide*, the industry-leading book on DB2 for z/OS, currently in its sixth edition.

Craig is a frequent contributor to computer industry publications, having authored hundreds of articles in the past several years. His articles have appeared in popular industry magazines and Web sites, including *Database Programming & Design, Data Management Review, DBMS, DB2 Update, Oracle Update, SQL Server Update*, and many others. Craig writes several regular columns, including a monthly column called “The DBA Corner” for *Database Trends and Applications* magazine, a quarterly column called
“The Database Report” for The Data Administration Newsletter (www.tdan.com), and a regular column on DB2 and mainframe data management called “z/Data Perspectives” for zJournal Magazine. Craig is also a regular blogger, managing and authoring two popular data-related blogs: The DB2 Portal (http://db2portal.blogspot.com) focusing on DB2 for z/OS and mainframe “stuff,” and Data and Technology Today (http://datatechnologytoday.wordpress.com), which focuses on data and database management issues, DBA news and thoughts, metadata management, and data architecture, as well as data-related topics in the realm of IT and software. Craig is also the publisher and editor of The Database Site (www.thedatabasesite.com).

Craig regularly presents technical topics at database industry conferences and events. He has spoken to thousands of technicians about database management and administration issues at such conferences as Database and Client/Server World, SHARE, GUIDE, DAMA Symposium, Enterprise Data World, IBM Information On Demand Conference, the DB2 Technical Conference, the International DB2 Users Group (IDUG), and Oracle Open World. He has also spoken at regional database user groups across North America, Europe, Asia, and Australia.

Craig graduated cum laude from the University of Pittsburgh with a double major in computer science and economics and a minor in mathematics. Craig has been appointed as an Information Management Champion by IBM for his work in the field of DB2 database administration, development, and management.

Readers can obtain information about this book, including corrections, future editions, and additional writings on database administration by the author, at the author's Web site at www.craigsullins.com. The author can be contacted at craig@craigsullins.com or in care of the publisher.
Creating the Database Environment

One of the primary tasks associated with the job of DBA is the process of choosing and installing a DBMS. Unfortunately, many business executives and IT professionals without database management background assume that once the DBMS is installed, the bulk of the work is done. The truth is, choosing and installing the DBMS is hardly the most difficult part of a DBA’s job. Establishing a usable database environment requires a great deal of skill, knowledge, and consideration. This chapter will outline the principles involved in establishing a usable database environment.

Defining the Organization’s DBMS Strategy

Choosing a suitable DBMS for enterprise database management is not as difficult as it used to be. The number of major DBMS vendors has dwindled due to industry consolidation and domination of the sector by a few very large players.

Yet, large and medium-size organizations typically run multiple DBMS products, from as few as two to as many as ten. For example, it is not uncommon for a large company to use IMS or IDMS and DB2 on the mainframe,
Oracle and MySQL on several different UNIX servers, Microsoft SQL Server on Windows servers, as well as pockets of other DBMS products such as Sybase, Ingres, Adabas, and PostgreSQL on various platforms, not to mention single-user PC DBMS products such as Microsoft Access, Paradox, and FileMaker. Who chose to install all these DBMSs and why?

Unfortunately, often the answer is that not much thought and planning went into the decision-making process. Sometimes the decision to purchase and install a new DBMS is driven by a business need or a new application. This is reasonable if your organization has no DBMS and must purchase one for the first time. This is rarely the case, though. Regardless of whether a DBMS exists on-site, a new DBMS is often viewed as a requirement for a new application. Sometimes a new DBMS product is purchased and installed without first examining if the application could be successfully implemented using an existing DBMS. Or, more likely, the DBAs know the application can be implemented using an existing DBMS but lack the organizational power or support to reject a new DBMS proposal.

There are other reasons for the existence of multiple DBMS platforms in a single organization. Perhaps the company purchased a commercial off-the-shelf application package that does not run on any of the current DBMS platforms. Sometimes the decision to buy a new DBMS is driven by the desire to support the latest and greatest technology. For example, many mainframe shops moving from a hierarchic (IMS) or CODASYL (IDMS) database model to the relational model deployed DB2, resulting in an additional DBMS to learn and support. Then, when client/server computing became popular, additional DBMSs were implemented on UNIX, Linux, and Windows servers.

Once a DBMS is installed, removal can be difficult because of incompatibilities among the different DBMSs and the necessity of converting application code. Furthermore, when a new DBMS is installed, old applications and databases are usually not migrated to it. The old DBMS remains and must continue to be supported. This complicates the DBA’s job.

So what should be done? Well, the DBA group should be empowered to make the DBMS decisions for the organization. No business unit should be allowed to purchase a DBMS without the permission of the DBA group. This is a difficult provision to implement and even more difficult to enforce. Business politics often work against the DBA group because it frequently possesses less organizational power than other business executives.
Choosing a DBMS

The DBA group should set a policy regarding the DBMS products to be supported within the organization. Whenever possible, the policy should minimize the number of different DBMS products. For a shop with multiple operating systems and multiple types of hardware, choose a default DBMS for the platform. Discourage deviation from the default unless a compelling business case exists—a business case that passes the technical inspection of the DBA group.

Most of the major DBMS products have similar features, and if the feature or functionality does not exist today, it probably will within 18 to 24 months. So, exercise caution before deciding to choose a DBMS based solely on its ability to support a specific feature.

When choosing a DBMS, it is wise to select a product from a tier-1 vendor as listed in Table 2.1. Tier 1 represents the largest vendors having the most heavily implemented and supported products on the market. You cannot go wrong with DB2 or Oracle. Both are popular and support just about any type of database. Another major player is Microsoft SQL Server, but only for Windows platforms. DB2 and Oracle run on multiple platforms ranging from mainframe to UNIX, as well as Windows and even handheld devices. Choosing a DBMS other than these three should be done only under specific circumstances.

After the big three come MySQL, Sybase, Teradata, and Informix. Table 2.2 lists these tier-2 DBMS vendors. All of these offerings are quality DBMS

<table>
<thead>
<tr>
<th>DBMS Vendor</th>
<th>DBMS Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM Corporation</td>
<td>DB2</td>
</tr>
<tr>
<td>New Orchard Road</td>
<td></td>
</tr>
<tr>
<td>Armonk, NY 10504</td>
<td></td>
</tr>
<tr>
<td>Phone: (914) 499-1900</td>
<td></td>
</tr>
<tr>
<td>Oracle Corporation</td>
<td>Oracle</td>
</tr>
<tr>
<td>500 Oracle Parkway</td>
<td></td>
</tr>
<tr>
<td>Redwood Shores, CA 94065</td>
<td></td>
</tr>
<tr>
<td>Phone: (650) 506-7000</td>
<td></td>
</tr>
<tr>
<td>Microsoft Corporation</td>
<td>SQL Server</td>
</tr>
<tr>
<td>One Microsoft Way</td>
<td></td>
</tr>
<tr>
<td>Redmond, WA 98052</td>
<td></td>
</tr>
<tr>
<td>Phone: (425) 882-8080</td>
<td></td>
</tr>
</tbody>
</table>
products, but their installed base is smaller, their products are engineered and marketed for niche purposes, or the companies are smaller with fewer resources than the Big Three (IBM, Oracle, and Microsoft), so there is some risk in choosing a DBMS from tier 2 instead of tier 1. However, there may be solid reasons for deploying a tier-2 solution, such as the high performance offered by Informix or the data warehousing and analytics capabilities of Teradata.

Of course, there are other DBMS products on the market, many of which are fine products and worthy of consideration for specialty processing, certain predefined needs, and niche roles. If your company is heavily into the open-source software movement, PostgreSQL, EnterpriseDB, or MySQL might be viable options. If an object DBMS is important for a specific project, you might consider ObjectDesign or Versant. And there are a variety of NoSQL DBMS offerings available, too, such as Hadoop, Cassandra, and MongoDB.\(^1\)

However, for the bulk of your data management needs, a DBMS from a tier-1, or perhaps tier-2, DBMS vendor will deliver sufficient functionality with minimal risk. A myriad of DBMS products are available, each with

---

1. If you prefer commercial software over open source, there are commercial offerings of some of the NoSQL products. For example, DataStax is based on Cassandra.
certain features that make them worthy of consideration on a case-by-case basis. Choosing any of the lower-tier candidates—even such major names as Software AG’s Adabas and Actian’s Ingres—involves incurring additional risk. Refer to Appendix B for a list of DBMS vendors.

I do not want it to sound as if the selection of a DBMS is a no-brainer. You will need a strategy and a plan for selecting the appropriate DBMS for your specific situation. When choosing a DBMS, be sure to consider each of these factors:

- **Operating system support.** Does the DBMS support the operating systems in use at your organization, including the versions that you are currently using and plan on using?

- **Type of organization.** Take into consideration the corporate philosophy when you choose a DBMS. Some organizations are very conservative and like to keep a tight rein on their environments; these organizations tend to gravitate toward traditional mainframe environments. Government operations, financial institutions, and insurance and health companies usually tend to be conservative. More-liberal organizations are often willing to consider alternative architectures. It is not uncommon for manufacturing companies, dot-coms, and universities to be less conservative. Finally, some companies just do not trust Windows as a mission-critical environment and prefer to use UNIX; this rules out some database vendors (Microsoft SQL Server, in particular).

- **Benchmarks.** What performance benchmarks are available from the DBMS vendor and other users of the DBMS? The Transaction Processing Performance Council (TPC) publishes official database performance benchmarks that can be used as a guideline for the basic overall performance of many different types of database processing. (Refer to the sidebar “The Transaction Processing Performance Council” for more details.) In general, performance benchmarks can be useful as a broad indicator of database performance but should not be the only determinant when selecting a DBMS. Many of the TPC benchmarks are run against database implementations that are not representative of most production database systems and therefore are not indicative of the actual performance of a particular DBMS. In addition, benchmarks are constantly updated to show new
and improved performance measurements for each of the major DBMS products, rendering the benchmark “winners” obsolete very quickly.

- **Scalability.** Does the DBMS support the number of users and database sizes you intend to implement? How are large databases built, supported, and maintained—easily or with a lot of pain? Are there independent users who can confirm the DBMS vendor’s scalability claims?

- **Availability of supporting software tools.** Are the supporting tools you require available for the DBMS? These items may include query and analysis tools, data warehousing support tools, database administration tools, backup and recovery tools, performance-monitoring tools, and improved performance measurements for each of the major DBMS products, rendering the benchmark “winners” obsolete very quickly.

---

### The Transaction Processing Performance Council (TPC)

The Transaction Processing Performance Council is an independent, not-for-profit organization that manages and administers performance benchmark tests. Its mission is to define transaction processing and database benchmarks to provide the industry with objective, verifiable performance data. TPC benchmarks measure and evaluate computer functions and operations.

The definition of *transaction* espoused by the TPC is a business one. A typical TPC transaction includes the database updates for things such as inventory control (goods), airline reservations (services), and banking (money).

The benchmarks produced by the TPC measure performance in terms of how many transactions a given system and database can perform per unit of time, for example, number of transactions per second. The TPC defines three benchmarks:

- **TPC-C**, for planned production workload in a transaction environment
- **TPC-H**, a decision support benchmark consisting of a suite of business-oriented ad hoc queries and concurrent data modifications
- **TPC-E**, an updated OLTP workload (based on financial transaction processing)

Additional information and in-depth definitions of these benchmarks can be found at the TPC Web site at www.tpc.org (see Figure 2.1).
tools, capacity-planning tools, database utilities, and support for various programming languages.

- **Technicians.** Is there a sufficient supply of skilled database professionals for the DBMS? Consider your needs in terms of DBAs, technical support personnel (system programmers and administrators, operations analysts, etc.), and application programmers.

- **Cost of ownership.** What is the total cost of ownership of the DBMS? DBMS vendors charge wildly varying prices for their technology. Total cost of ownership should be calculated as a combination of the license cost of the DBMS; the license cost of any required supporting software; the cost of database professionals to program, support, and administer the DBMS; and the cost of the computing resources required to operate the DBMS.

---

**Figure 2.1** The TPC Web site

---
• **Release schedule.** How often does the DBMS vendor release a new version? Some vendors have rapid release cycles, with new releases coming out every 12 to 18 months. This can be good or bad, depending on your approach. If you want cutting-edge features, a rapid release cycle is good. However, if your shop is more conservative, a DBMS that changes frequently can be difficult to support. A rapid release cycle will cause conservative organizations either to upgrade more frequently than they would like or to live with outdated DBMS software that is unlikely to have the same level of support as the latest releases.

• **Reference customers.** Will the DBMS vendor supply current user references? Can you find other users on your own who might provide more impartial answers? Speak with current users to elicit issues and concerns you may have overlooked. How is support? Does the vendor respond well to problems? Do things generally work as advertised? Are there a lot of bug fixes that must be applied continuously? What is the quality of new releases? These questions can be answered only by the folks in the trenches.

When choosing a DBMS, be sure to take into account the complexity of the products. DBMS software is very complex and is getting more complex with each new release. Functionality that used to be supported only with add-on software or independent programs is increasingly being added as features of the DBMS, as shown in Figure 2.2. You will need to plan for and support all the features of the DBMS. Even if there is no current requirement for certain features, once you implement the DBMS the programmers and developers will find a reason to use just about anything the vendor threw into it. It is better to plan and be prepared than to allow features to be used without a plan for supporting them.

**DBMS Architectures**

The supporting architecture for the DBMS environment is very critical to the success of the database applications. One wrong choice or poorly implemented component of the overall architecture can cause poor performance, downtime, or unstable applications.

When mainframes dominated enterprise computing, DBMS architecture was a simpler concern. Everything ran on the mainframe, and that
was that. However, today the IT infrastructure is distributed and heterogeneous. The overall architecture—even for a mainframe DBMS—will probably consist of multiple platforms and interoperating system software. A team consisting of business and IT experts, rather than a single person or group, should make the final architecture decision. Business experts should include representatives from various departments, as well as from accounting and legal for software contract issues. Database administration representatives (DA, DBA, and SA), as well as members of the networking group, operating system experts, operations control personnel, programming experts, and any other interested parties, should be included in this team.

Furthermore, be sure that the DBMS you select is appropriate for the nature and type of processing you plan to implement. Four levels of DBMS architecture are available: enterprise, departmental, personal, and mobile.

An enterprise DBMS is designed for scalability and high performance. An enterprise DBMS must be capable of supporting very large databases, a large number of concurrent users, and multiple types of applications. The enterprise DBMS runs on a large-scale machine, typically a mainframe or a high-end server running UNIX, Linux, or Windows Server. Furthermore, an enterprise DBMS offers all the “bells and whistles” available from the DBMS vendor. Multiprocessor support, support for parallel queries, and other advanced DBMS features are core components of an enterprise DBMS.
A *departmental DBMS*, sometimes referred to as a workgroup DBMS, serves the middle ground. The departmental DBMS supports small to medium-size workgroups within an organization; typically, it runs on a UNIX, Linux, or Windows server. The dividing line between a departmental database server and an enterprise database server is quite gray. Hardware and software upgrades can allow a departmental DBMS to tackle tasks that previously could be performed only by an enterprise DBMS. The steadily falling cost of departmental hardware and software components further contributes to lowering the total cost of operation and enabling a workgroup environment to scale up to serve the enterprise.

A *personal DBMS* is designed for a single user, typically on a low- to medium-powered PC platform. Microsoft Access, SQLite, and FileMaker² are examples of personal database software. Of course, the major DBMS vendors also market personal versions of their higher-powered solutions, such as Oracle Database Personal Edition and DB2 Personal Edition. Sometimes the low cost of a personal DBMS results in a misguided attempt to choose a personal DBMS for a departmental or enterprise solution. However, do not be lured by the low cost. A personal DBMS product is suitable only for very small-scale projects and should never be deployed for multiuser applications.

Finally, the *mobile DBMS* is a specialized version of a departmental or enterprise DBMS. It is designed for remote users who are not usually connected to the network. The mobile DBMS enables local database access and modification on a laptop or handheld device. Furthermore, the mobile DBMS provides a mechanism for synchronizing remote database changes to a centralized enterprise or departmental database server.

A DBMS designed for one type of processing may be ill suited for other uses. For example, a personal DBMS is not designed for multiple users, and an enterprise DBMS is generally too complex for single users. Be sure to understand the differences among enterprise, departmental, personal, and mobile DBMS software, and choose the appropriate DBMS for your specific data-processing needs. You may need to choose multiple DBMS types—that is, a DBMS for each level—with usage determined by the needs of each development project.

If your organization requires DBMS solutions at different levels, favor the selection of a group of DBMS solutions from the same vendor whenever

---

2. FileMaker is offered in a professional, multiuser version, too.
possible. Doing so will minimize differences in access, development, and administration. For example, favor Oracle Database Personal Edition for your single-user DBMS needs if your organization uses Oracle as the enterprise DBMS of choice.

**DBMS Clustering**

*Clustering* is the use of multiple “independent” computing systems working together as a single, highly available system. A modern DBMS offers clustering support to enhance availability and scalability. The two predominant architectures for clustering are *shared-disk* and *shared-nothing*. These names do a good job of describing the nature of the architecture—at least at a high level.

*Shared-nothing* clustering is depicted in Figure 2.3. In a shared-nothing architecture, each system has its own private resources (memory, disks, etc.). The clustered processors communicate by passing messages through a network that interconnects the computers. In addition, requests from clients are automatically routed to the system that owns the resource. Only one of the clustered systems can “own” and access a particular resource at a time. In the event a failure occurs, resource ownership can be dynamically transferred to another system in the cluster. The main advantage of shared-nothing clustering is scalability. In theory, a shared-nothing multiprocessor

---

*Figure 2.3  Shared-nothing architecture*
can scale up to thousands of processors because they do not interfere with one another—nothing is shared.

In a **shared-disk** environment, all the connected systems share the same disk devices, as shown in Figure 2.4. Each processor still has its own private memory, but all the processors can directly address all the disks. Typically, shared-disk clustering does not scale as well for smaller machines as shared-nothing clustering. Shared-disk clustering is better suited to large-enterprise processing in a mainframe environment. Mainframes—very large processors—are capable of processing enormous volumes of work. Great benefits can be obtained with only a few clustered mainframes, while many PC and midrange processors would need to be clustered to achieve similar benefits.

Shared-disk clustering is usually preferable for applications and services requiring only modest shared access to data and for applications or workloads that are very difficult to partition. Applications with heavy data update requirements are probably better implemented as shared-nothing. Table 2.3 compares the capabilities of shared-disk and shared-nothing architectures.

![Figure 2.4 Shared-disk architecture](image)

**Table 2.3** Comparison of Shared-Disk and Shared-Nothing Architectures

<table>
<thead>
<tr>
<th>Shared-Disk</th>
<th>Shared-Nothing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quick adaptability to changing workloads</td>
<td>Can exploit simpler, cheaper hardware</td>
</tr>
<tr>
<td>High availability</td>
<td>Almost unlimited scalability</td>
</tr>
<tr>
<td>Performs best in a heavy read environment</td>
<td>Works well in a high-volume, read-write environment</td>
</tr>
<tr>
<td>Data need not be partitioned</td>
<td>Data is partitioned across the cluster</td>
</tr>
</tbody>
</table>
The major DBMS vendors provide support for different types of clustering with different capabilities and requirements. For example, DB2 for z/OS provides shared-disk clustering with its Data Sharing and Parallel Sysplex capabilities; DB2 on non-mainframe platforms uses shared-nothing clustering. Oracle’s Real Application Clusters provide shared-disk clustering.

For most users, the primary benefit of clustering is the enhanced availability that accrues by combining processors. In some cases, clustering can help an enterprise to achieve five-nines (99.999 percent) availability. Additionally, clustering can be used for load balancing and failover.

**DBMS Proliferation**

As a rule of thumb, create a policy (or at least some simple guidelines) that must be followed before a new DBMS can be brought into the organization. Failure to do so can cause a proliferation of different DBMS products that will be difficult to support. It can also cause confusion regarding which DBMS to use for which development effort.

As mentioned earlier, there is a plethora of DBMS vendors, each touting its benefits. As a DBA, you will be bombarded with marketing and sales efforts that attempt to convince you that you need another DBMS. Try to resist unless a very compelling reason is given and a short-term return on investment (ROI) can be demonstrated. Even when confronted with valid reasons and good ROI, be sure to double-check the arguments and ROI calculations. Sometimes the reasons specified are outdated and the ROI figures do not take everything into account—such as the additional cost of administration.

Remember, every DBMS requires database administration support. Moreover, each DBMS uses different methods to perform similar tasks. The fewer DBMS products installed, the less complicated database administration becomes, and the better your chances become of providing effective data management resources for your organization.

**Hardware Issues**

When establishing a database environment for application development, selecting the DBMS is only part of the equation. The hardware and operating system on which the DBMS will run will greatly impact the reliability, availability, and scalability (RAS) of the database environment. For example, a mainframe platform such as an IBM zEC12 running z/OS will probably
provide higher RAS than a midrange IBM xSeries machine running AIX, which in turn will probably exceed a Dell server running Windows. That is not to say everything should run on a mainframe; other issues such as cost, experience, manageability, and the needs of the applications to be developed must be considered. The bottom line is that you must be sure to factor hardware platform and operating system constraints into the DBMS selection criteria.

Cloud Database Systems

Cloud computing (see the sidebar) is increasing in usage, especially at small to medium-size businesses. A cloud implementation can be more cost-effective than building an entire local computing infrastructure that requires management and support.

A cloud database system delivers DBMS services over the Internet. The trade-off essentially comes down to trusting a cloud provider to store and manage your data in return for minimizing database administration and maintenance cost and effort. Using cloud database systems can enable organizations, especially smaller ones without the resources to invest in an enterprise computing infrastructure, to focus on their business instead of their computing environment.

By consolidating data sources in the cloud, it is possible to improve collaboration among partners, branch offices, remote workers, and mobile devices, because the data becomes accessible as a service. There is no need to install, set up, patch, or manage the DBMS software because the cloud

---

Cloud Computing Overview

At a high level, cloud computing is the delivery of computing as a service. Cloud computing applications rely on a network (typically the Internet) to provide users with shared resources, software, and data. With cloud computing, computer systems and applications are supposed to function like a utility provider (such as the electricity grid).

The term cloud is used as a metaphor for the Internet. It is based on the tendency to draw network access as an abstract “cloud” in infrastructure diagrams. An example of this can be seen in Figure 1.11 in Chapter 1 of this book.

From a DBMS perspective, cloud computing moves the data and its management away from your local computing environment and delivers it as a service over the Internet.
provider manages and cares for these administrative tasks. Of course, the downside is that your data is now stored and controlled by an external agent—the cloud provider. Another inherent risk of cloud computing is the possibility of nefarious agents posing as legitimate customers.

An example of a cloud database platform is Microsoft SQL Azure. It is built on SQL Server technologies and is a component of the Windows Azure platform.

## Installing the DBMS

Once the DBMS has been chosen, you will need to install it. Installing a DBMS is not as simple as popping a CD into a drive and letting the software install itself (or, for you mainframe folks, just using IEBGENER to copy it from a tape). A DBMS is a complex piece of software that requires up-front planning for installation to be successful. You will need to understand the DBMS requirements and prepare the environment for the new DBMS.

### DBMS Installation Basics

The very first thing to do when you install a DBMS for the first time is to understand the prerequisites. Every DBMS comes with an installation manual or guide containing a list of the operating requirements that must be met for the DBMS to function properly. Examples of prerequisites include ensuring that an appropriate version of the operating system is being used, verifying that there is sufficient memory to support the DBMS, and ensuring that any related software to be used with the DBMS is the proper version and maintenance level.

Once the basics are covered, read the installation guide from cover to cover. Make sure that you understand the process before you even begin to install the DBMS. Quite a few preparations need to be made before installing a DBMS, and reading about them before you start will ensure a successful installation. Review how the installation program or routine for the DBMS operates, and follow the explicit instructions in the installation guide provided with the DBMS software. You additionally might want to work closely with the DBMS vendor during an initial installation to ensure that your plans are sound. In some cases, working with a local, experienced vendor or consultant can be beneficial to avoid installation and configuration errors.
Chapter 2  Creating the Database Environment

The remainder of this section will discuss some of the common preparations that are required before a DBMS can be installed. If the DBMS is already operational and you are planning to migrate to a new DBMS release, refer to the section “Upgrading DBMS Versions and Releases.”

Hardware Requirements

Every DBMS has a basic CPU requirement, meaning a CPU version and minimum processor speed required for the DBMS to operate. Additionally, some DBMSs specify hardware models that are required or unsupported. Usually the CPU criterion will suffice for an Intel environment, but in a mainframe or enterprise server environment the machine model can make a difference with regard to the DBMS features supported. For example, certain machines have built-in firmware that can be exploited by the DBMS if the firmware is available.

Furthermore, each DBMS offers different “flavors” of its software for specific needs. (I use “flavor” as opposed to “version” or “release,” which specify different iterations of the same DBMS.) Different flavors of the DBMS (at the same release level) are available for specific environments such as parallel processing, pervasive computing (such as handheld devices), data warehousing, and/or mobile computing. Be sure to choose the correct DBMS for your needs and to match your hardware to the requirements of the DBMS.

Storage Requirements

A DBMS requires disk storage to run. And not just for the obvious reason—to create databases that store data. A DBMS will use disk storage for the indexes to be defined on the databases as well as for the following items:

- The system catalog or data dictionary used by the DBMS to manage and track databases and related information. The more database objects you plan to create, the larger the amount of storage required by the system catalog.
- Any other system databases required by the DBMS, for example, to support distributed connections or management tools.
- Log files that record all changes made to every database. These include active logs, archive logs, rollback segments, and any other type of change log required by the DBMS.
• Start-up or control files that must be accessed by the DBMS when it is started or initialized.
• Work files used by the DBMS to sort data or for other processing needs.
• Default databases used by the DBMS for system structures or as a default catchall for new database objects as they are created.
• Temporary database structures used by the DBMS (or by applications accessing databases) for transient data that is not required to be persistent but needs reserved storage during operations (such as rebuilding clustered indexes on Microsoft SQL Server).
• System dump and error-processing files.
• DBA databases used for administration, monitoring, and tuning—for example, DBA databases used for testing new releases, migration scripts, and so on.

Be sure to factor in every storage requirement of the DBMS and reserve the appropriate storage. Also, be aware that the DBMS will use many of these databases and file structures concurrently. Therefore, it is a good idea to plan on using multiple storage devices even if you will not fill them to capacity. Proper database and file placement will enable the DBMS to operate more efficiently because concurrent activities will not be constrained by the physical disk as data is accessed.

Disk storage is not the only requirement of a DBMS. Tape or optical discs (such as DVDs and CDs) are also required for tasks such as database backups and log off-loading. When the active log file fills up, the log records must be off-loaded to an archive log either on disk or on tape, as shown in Figure 2.5. Depending on the DBMS being used and the features that have been activated, this process may be automatic or manual. The archive log files must be retained for recovery purposes, and even if originally stored on disk, they must eventually be migrated to an external storage mechanism for safekeeping.

Plan on maintaining multiple tape or CD/DVD drives to enable the DBMS to run concurrent multiple processes that require external storage, such as concurrent database backups. Database outages can occur if you single-thread your database backup jobs using a single drive.
Memory Requirements

Relational DBMSs, as well as their databases and applications, love memory. A DBMS requires memory for basic functionality and will use it for most internal processes such as maintaining the system global area and performing many DBMS tasks.

A DBMS requires a significant amount of memory to cache data in memory structures in order to avoid I/O. Reading data from a disk storage device is always more expensive and slower than moving the data around in memory. Figure 2.6 shows how the DBMS uses a memory structure called a buffer pool or data cache to reduce physical I/O requests. By caching data that is read into a buffer pool, the DBMS can avoid I/O for subsequent requests for the same data, as long as it remains in the buffer pool. In general, the larger the buffer pool, the longer the data can remain in memory and the better overall database processing will perform.

Besides data, the DBMS will cache other structures in memory. Most DBMSs set aside memory to store program structures required by the DBMS.
to process database requests. The program cache stores things like “compiled” SQL statements, database authorizations, and database structure blocks that are used by programs as they are executed. When these structures are cached, database processing can be optimized because additional I/O requests to access them from a physical storage device are avoided.

Memory is typically required by the DBMS to support other features such as handling lock requests, facilitating distributed data requests, sorting data, optimizing processes, and processing SQL.

Ensure that the DBMS has a more-than-adequate supply of memory at its disposal. This will help to optimize database processing and minimize potential problems.

3. In DB2, the area used for caching program structures in memory is referred to as the EDM pool. In SQL Server it is called the SQL cache, and in Oracle two structures are used, the PGA and the shared pool in the SGA.
Configuring the DBMS

Configuring the system parameters of the DBMS controls the manner in which the DBMS functions and the resources made available to it.\(^4\) Each DBMS allows its system parameters to be modified in different ways, but the installation process usually sets the DBMS system parameters by means of radio buttons, menus, or panel selections. During the installation process, the input provided to the installation script will be used to establish the initial settings of the system parameters.

Each DBMS also provides a method to change the system parameters once the DBMS is operational. Sometimes you can use DBMS commands to set the system’s parameters; sometimes you must edit a file that contains the current system parameter settings. If you must edit a file, be very careful: An erroneous system parameter setting can be fatal to the operational status of the DBMS.

What do the system parameters control? Well, for example, system parameters control DBA authorization to the DBMS and the number of active database logs; system parameters set the amount of memory used for data and program caching and turn DBMS features on or off. Although every DBMS has system parameters that control its functionality, each DBMS has a different method of setting and changing the values. And, indeed, each DBMS has different specifications that can be set using system parameters.

Beware of simply using default system parameters when installing the database system software. Although using defaults can save time and make for an easier installation, it can also result in subsequent problems. Most DBMSs are poorly served, in the long run, by default settings and, in some cases, can experience worsening performance over time because resources were not preallocated during installation or setup.

Be sure to understand fully the parameters used by your DBMS. Failure to do so can result in an incorrectly configured database environment, which can cause performance problems, data integrity problems, or even DBMS failure.

---

\(^4\) In DB2, system parameters are set by assembling the DSNZPARM member. SQL Server uses the SP_CONFIGURE system procedure to set system parameters, and Oracle parameters are controlled using INIT.ORA.
Connecting the DBMS to Supporting Infrastructure Software

Part of the DBMS installation process is the connection of the DBMS to other system software components that must interact with the DBMS. Typical infrastructure software that may need to be configured to work with the DBMS includes networks, transaction processing monitors, message queues, other types of middleware, programming languages, systems management software, operations and job control software, Web servers, and application servers.

Each piece of supporting infrastructure software will have different requirements for interfacing with the DBMS. Typical configuration procedures can include installing DLL files, creating new parameter files to establish connections, and possibly revisiting the installation procedures for the supporting software to install components required to interact with the DBMS.

Installation Verification

After installing the DBMS, you should run a battery of tests to verify that the DBMS has been properly installed and configured. Most DBMS vendors supply sample programs and installation verification procedures for this purpose. Additionally, you can ensure proper installation by testing the standard interfaces to the DBMS. One standard interface supported by most DBMSs is an interactive SQL interface where you can submit SQL statements directly to the DBMS.5

Create a set of SQL code that comprises SELECT, INSERT, UPDATE, and DELETE statements issued against sample databases. Running such a script after installation helps you to verify that the DBMS is installed correctly and operating as expected.

Furthermore, be sure to verify that all required connections to supporting software are operational and functioning properly. If the DBMS vendor does not supply sample programs, you may need to create and run simple test programs for each environment to ensure that the supporting software connections are functioning correctly with the DBMS.

5. In DB2, the SQL interface is referred to as SPUFI. IBM also provides Data Studio for GUI-based SQL creation and submission. SQL Server calls the interface ISQL, and when using Oracle you can choose to submit SQL using SQL*Plus or the SQL Worksheet in Oracle Enterprise Manager.
**DBMS Environments**

Generally, installing a DBMS involves more than simply installing one instance or subsystem. To support database development, the DBA needs to create multiple DBMS environments to support, for example, testing, quality assurance, integration, and production work. Of course, it is possible to support multiple environments in a single DBMS instance, but it is not prudent. Multiple DBMS installations are preferable to support multiple development environments for a single database. This minimizes migration issues and won't require complex database naming conventions to support. Furthermore, segregating database instances makes testing, tuning, and monitoring easier.

**Upgrading DBMS Versions and Releases**

Change is a fact of life, and each of the major DBMS products changes quite rapidly. A typical release cycle for DBMS software is 18 to 24 months for major releases, with constant bug fixes and maintenance updates delivered between major releases. Indeed, keeping DBMS software up-to-date can be a full-time job.

The DBA must develop an approach to upgrading DBMS software that conforms to the organization's needs and minimizes business disruptions due to outages and database unavailability.

You may have noticed that I use the terms *version* and *release* somewhat interchangeably. That is fine for a broad discussion of DBMS upgrades, but a more precise definition is warranted. For a better discussion of the differences between a version and a release, please refer to the sidebar.

A DBMS version upgrade can be thought of as a special case of a new installation. All the procedures required of a new installation apply to an upgrade: You must plan for appropriate resources, reconsider all system parameters, and ensure that all supporting software is appropriately connected. However, another serious issue must be planned for: existing users and applications. An upgrade needs to be planned to cause as little disruption to the existing users as possible. Furthermore, any additional software that works with the DBMS (such as purchased applications, DBA tools, utilities, and so on) must be verified to be compatible with the new DBMS version. Therefore, upgrading can be a tricky and difficult task.
In a complex, heterogeneous, distributed database environment, a coherent upgrade strategy is essential. Truthfully, even organizations with only a single DBMS should approach DBMS upgrades cautiously and plan accordingly. Failure to plan a DBMS upgrade can result in improper and inefficient adoption of new features, performance degradation of new and existing applications, and downtime.

Upgrading to a new DBMS release offers both rewards and risks. The following are some of the benefits of moving to a new release:

- Developers can avail themselves of new features and functionality delivered only in the new release. If development requires a new feature, or can simply benefit from a new feature, program development time can be reduced or made more cost-effective.

- For purchased applications, the application vendor may require a specific DBMS version or release for specific versions of its application to enable specific functionality within the application.

- New DBMS releases usually deliver enhanced performance and availability features that can optimize existing applications.

### Version or Release?

Vendors typically make a distinction between a version and a release of a software product. A new version of software is a major concern, with many changes and new features. A release is typically minor, with fewer changes and not as many new features.

For example, moving from Version 10g of Oracle Database to Version 11g would be a major change—a version change. However, an in-between point such as Oracle Database 11g Release 2 would be considered a release—consisting of a smaller number of changes. Usually DBMS vendors increase prices for versions, but not necessarily for releases (but that is not a hard-and-fast rule).

Usually significant functionality is added for version upgrades, less so for point releases. Nevertheless, upgrading from one point release to another can have just as many potential pitfalls as a version upgrade. It depends on the nature of the new features provided in each specific release.

The issues and concerns discussed in this chapter pertain to both types of DBMS upgrades: to a new release and to a new version.
Sometimes a new DBMS release is required to scale applications to support additional users or larger amounts of data.

- DBMS vendors often provide better support and respond to problems faster for a new release of their software. DBMS vendors are loath to allow bad publicity about bugs in a new and heavily promoted version of their products.

- Cost savings may accrue by upgrading to a new DBMS release. Some vendors charge additionally when a company uses multiple versions of a DBMS, such as the new version in a test environment and the old in production. When both are migrated to the same version, the price tag for the DBMS sometimes can be reduced.

- Production migration to a new DBMS release will align the test and production database environments, thereby providing a consistent environment for development and implementation. If a new release is running in the test environment for too long, database administration and application development tasks become more difficult because the test databases will operate differently from the production databases.

However, an effective DBMS upgrade strategy must balance the benefits against the risks of upgrading to arrive at the best timeline for migrating to a new DBMS version or release. The risks of upgrading to a new DBMS release include the following:

- An upgrade to the DBMS usually involves some level of disruption to business operations. At a minimum, databases will not be available while the DBMS is being upgraded. This can result in downtime and lost business opportunities if the DBMS upgrade occurs during normal business hours (or if there is no planned downtime). Clustered database implementations may permit some database availability while individual database clusters are migrated to the new DBMS version.

- Other disruptions can occur, such as having to convert database structures or discovering that previously supported features were removed from the new release (thereby causing application errors). Delays to application implementation timelines are another possibility.
• The cost of an upgrade can be a significant barrier to DBMS release migration. First, the cost of the new version or release must be budgeted for (price increases for a new DBMS version can amount to as much as 10 to 25 percent). The upgrade cost must also factor in the costs of planning, installing, testing, and deploying not just the DBMS but also any applications that use databases. Finally, be sure to include the cost of any new resources (such as memory, storage, additional CPUs) required to use the new features delivered by the new DBMS version. 6

• DBMS vendors usually tout the performance gains that can be achieved with a new release. However, when SQL optimization techniques change, it is possible that a new DBMS release will generate SQL access paths that perform worse than before. DBAs must implement a rigorous testing process to ensure that new access paths are helping, not harming, application performance. When performance suffers, application code may need to be changed—a very costly and time-consuming endeavor. A rigorous test process should be able to catch most of the access path changes in the test environment.

• New DBMS releases may cause features and syntax that are being used in existing applications to be deprecated. 7 When this occurs, the applications must be modified before migration to the new release can proceed.

• To take advantage of improvements implemented in a new DBMS release, the DBA may have to apply some invasive changes. For example, if the new version increases the maximum size for a database object, the DBA may have to drop and recreate that object to take advantage of the new maximum. This will be the case when the DBMS adds internal control structures to facilitate such changes.

• Supporting software products may lack immediate support for a new DBMS release. Supporting software includes the operating

---

6. Be careful, too, to examine the specifications for any new DBMS version or release. Sometimes features and functionality are removed from the DBMS, which might result in having to spend additional money to replace the lost functionality. For example, IBM removed its formerly free database utilities from DB2 between Versions 6 and 7 and bundled them for sale.

7. When a feature is deprecated it is no longer supported in the software.
system, transaction processors, message queues, purchased applications, DBA tools, development tools, and query and reporting software.

After weighing the benefits of upgrading against the risks of a new DBMS release, the DBA group must create an upgrade plan that works for the organization. Sometimes the decision will be to upgrade immediately upon availability, but often there is a lag between the general availability of a new release and its widespread adoption.

When the risks of a new release outweigh the benefits, some organizations may decide to skip an interim release if doing so does not impact a future upgrade. For example, a good number of Oracle customers migrated directly from Oracle7 to Oracle8i, skipping Oracle8. If the DBMS vendor does not allow users to bypass a version or release, it is still possible to "skip" a release by waiting to implement that release until the next release is available. For example, consider the following scenario:

1. ABC Corporation is using DB Version 8 from DBCorp.
2. DBCorp announces Version 9 of DB.
3. ABC Corporation analyzes the features and risks and determines not to upgrade immediately.
4. DBCorp later announces DB Version 10 and that no direct migration path will be provided from Version 8 to Version 10.
5. ABC Corporation decides that DB Version 10 provides many useful features and wants to upgrade its current Version 8 implementation of DB. However, it has no compelling reason to first implement and use Version 9.
6. To fulfill its requirements, ABC Corporation first upgrades Version 8 to Version 9 and then immediately upgrades Version 9 to Version 10.

Although a multiple-release upgrade takes more time, it allows customers to effectively control when and how they will migrate to new releases of a DBMS instead of being held hostage by the DBMS vendor. When attempting a multiple-release upgrade of this type, be sure to fully understand the features and functionality added by the DBMS vendor for each interim release. In the case of the hypothetical ABC Corporation, the DBAs would
need to research and prepare for the new features of not just Version 10 but also Version 9.

An appropriate DBMS upgrade strategy depends on many things. The following sections outline the issues that must be factored into an effective DBMS release upgrade strategy.

**Features and Complexity**

Perhaps the biggest factor in determining when and how to upgrade to a new DBMS release is the functionality supported by the new release. Tightly coupled to functionality is the inherent complexity involved in supporting and administering new features.

It is more difficult to delay an upgrade if application developers are clamoring for new DBMS features. If DBMS functionality can minimize the cost and effort of application development, the DBA group will feel pressure to migrate swiftly to the new release. An additional factor that will coerce rapid adoption of a new release is when DBMS problems are fixed in the new release (instead of through regular maintenance fixes).

Regardless of a new release’s “bells and whistles,” certain administration and implementation details must be addressed before upgrading. The DBA group must ensure that standards are modified to include the new features, educate developers and users as to how new features work and should be used, and prepare the infrastructure to support the new DBMS functionality.

The types of changes required to support the new functionality must be factored into the upgrade strategy. When the DBMS vendor makes changes to internal structures, data page layouts, or address spaces, the risks of upgrading are greater. Additional testing is warranted in these situations to ensure that database utilities, DBA tools, and data extraction and movement tools still work with the revised internal structures.

**Complexity of the DBMS Environment**

The more complex your database environment is, the more difficult it will be to upgrade to a new DBMS release. The first complexity issue is the size of the environment. The greater the number of database servers, instances, applications, and users, the greater the complexity. Additional concerns include the types of applications being supported. A DBMS upgrade is easier
to implement if only simple, batch-oriented applications are involved. As the complexity and availability requirements of the applications increase, the difficulty of upgrading also increases.

Location of the database servers also affects the release upgrade strategy. Effectively planning and deploying a DBMS upgrade across multiple database servers at various locations supporting different lines of business is difficult. It is likely that an upgrade strategy will involve periods of supporting multiple versions of the DBMS at different locations and for different applications. Supporting different versions in production should be avoided, but that is not always possible.

Finally, the complexity of the applications that access your databases must be considered. The more complex your applications are, the more difficult it will be to ensure their continuing uninterrupted functionality when the DBMS is modified. Complexity issues include the following:

- Usage of stored procedures and user-defined functions.
- Complexity of the SQL—the more tables involved in the SQL and the more complex the SQL features, the more difficult it becomes to ensure that access path changes do not impact performance.
- Client/server processing—network usage and usage of multiple tiers complicates testing the new DBMS release.
- Applications that are designed, coded, and generated by a framework or an IDE (for example, Hibernate) may have additional components that need to be tested with a new DBMS release.
- Integration with other infrastructure software such as message queues and transaction processors can complicate migration because new versions of these products may be required to support the new DBMS release.
- The language used by the programs might also impact DBMS release migration due to different support for compiler versions, changes to APIs (application programming interfaces), or new ways of embedding SQL into application programs.
Reputation of the DBMS Vendor

DBMS vendors have different reputations for technical support, fixing bugs, and responding to problems, which is why customer references are so important when choosing a database.

The better the reputation of the vendor, the greater the likelihood of organizations rapidly adopting a new release. If the DBMS vendor is good at responding to problems and supporting its customers as they migrate to new releases, those customers will more actively engage in migration activities.

Support Policies of the DBMS

As new releases are introduced, DBMS vendors will retire older releases and no longer support them. The length of time that the DBMS vendor will support an old release must be factored into the DBMS release migration strategy. You should never run a DBMS release in production that is no longer supported by the vendor. If problems occur, the DBMS vendor will not be able to resolve them for you.

Sometimes a DBMS vendor will provide support for a retired release on a special basis and at an increased maintenance charge. If you absolutely must continue using a retired DBMS release (for business or application issues), be sure to investigate the DBMS vendor's policies regarding support for retired releases of its software.

Organization Style

Every organization displays characteristics that reveal its style when it comes to adopting new products and technologies. Industry analysts at Gartner, Inc., have ranked organizations into three distinct groups labeled types A, B, and C. A type-A enterprise is technology driven and, as such, is more likely to risk using new and unproven technologies to try to gain a competitive advantage. A type-B organization is less willing to take risks but will adopt new technologies once others have shaken out the bugs. Finally, a type-C enterprise, very conscious of cost and averse to risk, will lag behind the majority when it comes to migrating to new technology.

Only type-A organizations should plan on moving aggressively to new DBMS releases immediately upon availability and only if the new features...
Chapter 2  Creating the Database Environment

of the release will deliver advantages to the company. Type-C enterprises should adopt a very conservative strategy to ensure that the DBMS release is stable and well tested by type-A and type-B companies first. Type-B organizations will fall somewhere between types A and C: Almost never upgrading immediately, the type-B company will adopt the new release after the earliest users have shaken out the biggest problems, but well before type-C enterprises.

DBA Staff Skill Set

Upgrading the DBMS is easier if your DBA staff is highly skilled and/or experienced. The risk of an upgrade increases as the skills of the DBA staff decrease. If your DBAs are not highly skilled, or have never migrated a DBMS to a new release, consider augmenting your DBA staff with consultants for the upgrade. Deploying an integrated team of internal DBAs and consultants will ensure that your upgrade goes as smoothly as possible. Furthermore, the DBA staff will be better prepared to handle the future upgrades alone.

If consultants will be required, be sure to include their contracting cost in the DBMS release upgrade budget. The budget should allow you to retain the consultants until all production database environments are stable.

Platform Support

When a DBMS vendor unleashes a new release of its product, not all platforms and operating systems are immediately supported. The DBMS vendor usually first supports the platforms and operating systems for which it has the most licensed customers. The order in which platforms are supported for a new release is likely to differ for each DBMS vendor. For example, Linux for System z is more strategic to IBM than to Oracle, so a new DB2 release will most likely support Linux for System z very quickly, whereas this may not be true of Oracle. The issue is even thornier for UNIX platforms because of the sheer number of UNIX variants in the marketplace. The most popular variants are Oracle’s Solaris, IBM’s AIX, Hewlett-Packard’s HP-UX, and Linux, the open-source version of UNIX (the Red Hat and Suse distributions are supported more frequently and rapidly than others). Most DBMS vendors will support these UNIX platforms quickly upon general availability. Other less popular varieties of UNIX will take longer for the DBMS vendors to support.
When planning your DBMS upgrade, be sure to consider the DBMS platforms you use and try to gauge the priority of your platform to your vendor. Be sure to build some lag time into your release migration strategy to accommodate the vendor’s delivery schedule for your specific platforms.

**Supporting Software**

Carefully consider the impact of a DBMS upgrade on any supporting software. Supporting software includes purchased applications, DBA tools, reporting and analysis tools, and query tools. Each software vendor will have a different time frame for supporting and exploiting a new DBMS release. Review the sidebar to understand the difference between support and exploitation of a new DBMS release.

Some third-party tool vendors follow guidelines for supporting and exploiting new DBMS releases. Whenever possible, ask your vendors to state their policies for DBMS upgrade support. Your vendors will probably not commit to any firm date or date range to support new versions and releases—some DBMS versions are larger and more complicated and therefore take longer to fully exploit.

---

**Support versus Exploit**

Some vendors differentiate specifically between supporting and exploiting a new DBMS version or release. Software that supports a new release will continue to function the same as before the DBMS was upgraded, but with no new capabilities. Therefore, if a DBA tool, for example, supports a new version of Oracle, it can provide all the services it did for the last release, as long as none of the new features of the new version of Oracle are used. In contrast, a DBA tool that exploits a new version or release provides the requisite functionality to operate on the new features of the new DBMS release.

So, to use a concrete example, IBM added support for hashing in Version 10 of DB2. A DBA tool can support DB2 Version 10 without operating on hashes, but it must operate on hashes to exploit DB2 Version 10.

Prior to migrating to a new DBMS version or release, make sure you understand the difference between supporting and exploiting a new version, and get a schedule for both from your third-party vendors for the DBA tools you use.
Chapter 2 Creating the Database Environment

Fallback Planning

Each new DBMS version or release should come with a manual that outlines the new features of the release and describes the fallback procedures to return to a prior release of the DBMS. Be sure to review the fallback procedures provided by the DBMS vendor in its release guide. You may need to return to a previous DBMS release if the upgrade contains a bug, performance problems ensue, or other problems arise during or immediately after migration. Keep in mind that fallback is not always an option for every new DBMS release.

If fallback is possible, follow the DBMS vendor’s recommended procedures to enable it. You may need to delay the implementation of certain new features for fallback to remain an option. Understand fully the limitations imposed by the DBMS vendor on fallback, and exploit new features only when fallback is no longer an option for your organization.

Migration Verification

The DBA should implement procedures—similar to those for a new installation—to verify that the DBMS release upgrade is satisfactory. Perform the same steps as with a brand-new DBMS install, but also test a representative sampling of your in-house applications to verify that the DBMS upgrade is working correctly and performing satisfactorily.

The DBMS Upgrade Strategy

In general, design your DBMS release upgrade policy according to the guidelines discussed in the preceding sections. Each specific DBMS upgrade will be unique, but the strategies we’ve discussed will help you to achieve success more readily. A well-thought-out DBMS upgrade strategy will prepare you to support new DBMS releases with minimum impact on your organization and in a style best suited to your company.

Database Standards and Procedures

Before a newly installed DBMS can be used effectively, standards and procedures must be developed for database usage. Studies have shown that companies with high levels of standardization reduce the cost of supporting end users by as much as 35 percent or more as compared to companies with low levels of standardization.
Standards are common practices that ensure the consistency and effectiveness of the database environment, such as database naming conventions. Procedures are defined, step-by-step instructions that direct the processes required for handling specific events, such as a disaster recovery plan. Failure to implement database standards and procedures will result in a database environment that is confusing and difficult to manage.

The DBA should develop database standards and procedures as a component of corporate-wide IT standards and procedures. They should be stored together in a central location as a printed document, in an online format, or as both. Several vendors offer “canned” standards and procedures that can be purchased for specific DBMS products.

Database Naming Conventions

One of the first standards to be implemented should be a set of guidelines for the naming of database objects. Without standard database object naming conventions, it will be difficult to identify database objects correctly and to perform the proper administration tasks.

Database object naming standards should be developed in conjunction with all other IT naming standards in your organization. In all cases, database naming standards should be developed in cooperation with the data administration department (if one exists) and, wherever possible, should peacefully coexist with other IT standards, but not at the expense of impairing the database environment. For example, many organizations have shop conventions for naming files, but coordinating the database object to the operating system file may require a specific format for database filenames that does not conform to the shop standards (see Figure 2.7). Therefore, it may be necessary to make exceptions to existing shop standards for naming database files.

![Figure 2.7 Database objects map to filenames](Mullins_Book.indb)
Be sure to create and publish naming standards for all database objects that can be created within each DBMS used by your organization. A basic list of database objects supported by most DBMSs includes databases, tables, columns, views, indexes, constraints, programs, user-defined data types, user-defined functions, triggers, and stored procedures. However, this list is incomplete because each DBMS uses other database objects specific to its operation. For example, DB2 uses plans and storage groups; Oracle uses database links and clusters; SQL Server uses filegroups and rules (see the sidebar).

The database naming standard should be designed to minimize name changes across environments. For example, embedding a T into the name for “test” and a P for “production” is a bad idea. It is especially important to avoid this approach for user-visible database objects such as columns,

---

**Example Nonstandard Database Objects**

Unless you use all three of DB2, Oracle, and SQL Server, some of the database objects that are specific to only one of these database systems probably will be unfamiliar to you. Given that, this sidebar offers short definitions of the database objects mentioned in this section.

For DB2:

- A **plan** is associated with a DB2 application program and refers to packages that contain bound access path details for the SQL in that program.
- A **storage group** is a database object used to associate disk storage with DB2 tablespaces.

For Oracle:

- A **database link** is a schema object in one database that enables you to access objects in another database.
- A **cluster** is made up of a group of tables that share the same data blocks. The tables are grouped together because they share common columns and are often used together.

For SQL Server:

- Database objects and files can be grouped together in **filegroups** for allocation and administration purposes.
- A **rule** is a freestanding database constraint that can be attached to columns. Microsoft has indicated that rules will be removed from a future version of SQL Server.
tables, and views. Minimizing name changes simplifies the migration of databases from one environment to another. It is possible to make all the database object names the same by assigning each environment to a different instance or subsystem. The instance or subsystem name, rather than the database object names, will differentiate the environments.

In most cases, for objects not accessed by typical end users, provide a way to differentiate types of database objects. For example, start indexes with \textit{I} or \textit{X} and databases with \textit{D}. For tables and similar objects, though, as discussed earlier, this approach is inappropriate.

In general, do not impose unnecessary restrictions on the names of objects accessed by end users. Relational databases are supposed to be user friendly. A strict database naming convention, if not developed logically, can be antithetical to a useful and effective database environment. Some organizations impose arbitrary length limitations on database tables, such as an 8-byte limit even though the DBMS can support up to 128-byte table names. There is no practical reason to impose a limitation on the length of database table names.

Table names should be as descriptive as possible, within reason. Furthermore, the same naming conventions should be used for all “tablelike” objects, including views, synonyms, and aliases, if supported by the DBMS. Each of these objects is basically a collection of data accessible as rows and columns. Developing separate naming conventions for each is of no real value. With this approach, database objects that operate like tables will be defined similarly with a very descriptive name. The type of object can always be determined by querying the DBMS system catalog or data dictionary.

Encoding table names to make them shorter is another arbitrary naming standard that should be avoided. Table names should include a 2- or 3-byte application identification prefix, followed by an underscore and then a clear, user-friendly name. For example, a good name for the table containing employee information in a human resources system would be HR\_EMPLOYEE. You may want to drop the application identification prefix from the table name for tables used by multiple applications.

Keep in mind, too, that some database object names will, in some cases, be externalized. For instance, most DBMSs externalize constraint names when the constraint is violated. There are many types of constraints—triggers, unique constraints, referential constraints, check constraints—each of which can be named. Keeping the names consistent across environments allows the error messages to be consistent. If the DBMS delivers the same
error message in the development, test, integration, and production environments, debugging and error correction will be easier.

**Standard Abbreviations**

Although you should keep the database object names as English-like as possible, you will inevitably encounter situations that require abbreviations. Use abbreviations only when the full text is too long to be supported as an object name or when it renders the object name unwieldy or difficult to remember. Create a list of standard abbreviations and forbid the use of non-standard abbreviations. For example, if “ORG” is the standard abbreviation for “organization,” do not allow variants such as “ORZG” to be used. Using standard abbreviations will minimize mistyping and make it easier for users to remember database object names. Adhering to this practice will make it easier to understand the database objects within your environment.

**Other Database Standards and Procedures**

Although database naming standards are important, you will need to develop and maintain other types of database standards. Be sure to develop a comprehensive set of standards and procedures for each DBMS used by your organization. Although you can write your database standards from scratch, there are other potentially easier ways to build your standards library. Basic standards that can be modified to your requirements can be bought from a publisher or software vendor. Or you can gather suggested standards from the community via user groups and conferences.

Regardless of whether they are purchased, written in house, or adopted from a user group or committee, each of the following areas should be covered.

**Roles and Responsibilities**

The successful operation of a DBMS requires the coordinated management efforts of many skilled technicians and business experts. A matrix of database management and administration functions should be developed that documents each support task and who within the organization provides the support. The matrix can be created at a departmental level, a job description level, or even by individual name. A sample matrix is shown in Table 2.4. An X in the matrix indicates involvement in the process, whereas a P indicates primary responsibility.
Table 2.4  *Database Support Roles and Responsibilities*

<table>
<thead>
<tr>
<th>Task</th>
<th>DBA</th>
<th>DA</th>
<th>SA</th>
<th>Management</th>
<th>Operations</th>
<th>Applications</th>
<th>End Users</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBMS budget</td>
<td>X</td>
<td>X</td>
<td>P</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>DBMS installation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>DBMS upgrade</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Database usage policy</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Capacity planning</td>
<td>X</td>
<td>P</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Data modeling and analysis</td>
<td>X</td>
<td>P</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Metadata policy</td>
<td>X</td>
<td>P</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Governance and compliance</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>P</td>
</tr>
<tr>
<td>Database design</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Database creation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>System performance</td>
<td>X</td>
<td>P</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Database performance</td>
<td>P</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Application performance</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>P</td>
<td></td>
</tr>
<tr>
<td>Backup and recovery</td>
<td>P</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Disaster recovery</td>
<td>P</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Database security</td>
<td>P</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Stored procedures</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>P</td>
</tr>
<tr>
<td>Triggers</td>
<td>P</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>User-defined functions</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>P</td>
<td></td>
</tr>
<tr>
<td>Application design</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>P</td>
<td></td>
</tr>
<tr>
<td>Application turnover</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>P</td>
<td>X</td>
</tr>
<tr>
<td>Application design reviews</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>P</td>
</tr>
</tbody>
</table>
Chapter 2  Creating the Database Environment

Of course, you can create whatever tasks you deem necessary in your roles and responsibilities matrix. You may need additional tasks, or fewer than in this sample. For example, you may wish to differentiate between stored-procedure development, testing, and management by creating a different task category for each and breaking down the support requirements differently.

Whatever the final format of your roles and responsibilities matrix, be sure to keep it accurate and up-to-date with new DBMS features and tasks. An up-to-date matrix makes it easier to define roles within the organization and to effectively apportion database-related workload.

Communication Standards
You might also choose to develop specific standards for communication between groups and specific personnel. For example, you might want to document how and when the DBA group must communicate with the systems programming group when a new DBMS release is being installed.

Developing robust communication standards can simplify a DBA’s job during the inevitable downtime that occurs due to system, application, or even hardware errors. For example, consider adopting a standard whereby the DBA communicates only with the manager during troubleshooting and emergency remediation. This keeps the manager informed and enables the DBA to dodge the dozens of phone calls that come in from angry users, the help desk, and so on. The manager can communicate the status outward while the DBA focuses exclusively on troubleshooting and getting the systems back up and running again.

Data Administration Standards
If a DA group exists within your organization, they should develop a basic data administration standards guide to outline the scope of their job role. If a DA group does not exist, be sure to include DA standards in the DBA standards as appropriate.

The data administration standards should include the following items:

- A clear statement of the organization’s overall policy with regard to data, including its importance to the company
- Guidelines for establishing data ownership and stewardship
- Rules for data creation, data ownership, and data stewardship
• Metadata management policy
• Conceptual and logical data modeling guidelines
• The organization’s goals with regard to creating an enterprise data model
• Responsibility for creating and maintaining logical data models
• Guidelines for tool usage and instructions on how data models are to be created, stored, and maintained
• Organizational data-sharing policies
• Instructions on how to document when physical databases deviate from the logical data model
• Guidelines on communication between data administration and database administration to ensure effective database creation and usage

**Database Administration Standards**

A basic set of database administration standards should be established to ensure the ongoing success of the DBA function. The standards serve as a guide to the DBA services offered and to specific approaches to supporting the database environment. For example, standards can be developed that outline how requests are made to create a new database or make changes to existing databases, and that specify which types of database objects and DBMS features are favored and under which circumstances they are to be avoided. Standards can establish backup and recovery procedures (including disaster recovery plans) and communicate the methods used to transform a logical data model into a physical database implementation. An additional set of DBA standards that cover database performance monitoring and tuning may be useful to document procedures for overcoming performance problems.

Although the DBA standards will be most useful for the DBA staff, the application development staff will need them to learn how best to work with the DBA staff. Furthermore, any performance tuning tricks that are documented in the DBA standards should be shared with programmers. The more the application programmers understand the nuances of the DBMS and the role of the DBA, the better the working relationship between DBA and development will be—resulting in a more efficient database environment.
System Administration Standards

Once again, standards for system administration or systems programming are required only if your organization separates the SA function from the DBA function. System administration standards are needed for many of the same reasons that DBA standards are required. Standards for SA may include

- DBMS installation and testing procedures
- Upgrade policies and procedures
- Bug fix and maintenance practices
- A checklist of departments to notify for impending changes
- Interface considerations
- DBMS storage, usage, and monitoring procedures

Database Application Development Standards

The development of database applications differs from typical program development. You should document the special development considerations required when writing programs that access databases. The database application development standards should function as an adjunct to any standard application development procedures within your organization. This set of standards should include

- A description of how database access differs from flat file access
- SQL coding standards
- SQL performance tips and techniques
- Program preparation procedures and guidance on how to embed SQL in an application program
- Interpretations of SQLSTATEs and error codes
- References to other useful programming materials for teleprocessing monitors, programming languages, and general application development standards

Database Security Standards

The DBA group often applies and administers DBMS security. However, at some shops the corporate data security unit handles DBMS security. A
resource outlining the necessary standards and procedures for administering database security should contain the following information:

- Details on what authority to grant for specific types of situations; for example, if a program is being migrated to production status, what DBMS authorization must be granted before the program will operate successfully in production
- Specific documentation of any special procedures or documentation required for governance- and compliance-related requests
- A definitive list of who can approve what types of database authorization requests
- Information on any interfaces being used to connect DBMS security with operating system security products
- Policies on the use of the WITH GRANT OPTION clause of the SQL GRANT statement and how cascading REVOKEs are to be handled
- Procedures for notifying the requester that database security has been granted
- Procedures for removing security from retiring, relocating, and terminated employees

**Application Migration and Turnover Procedures**

As discussed earlier, the minimum number of environments for supporting database applications is two: test and production. Some organizations, however, create multiple environments to support, for example, different phases of the development life cycle, including

- **Unit testing**—for developing and testing individual programs
- **Integration testing**—for testing how individual programs interoperate
- **User acceptance testing**—for end user testing prior to production status
- **Quality assurance**—for shaking out program bugs
- **Education**—for training end users how to work the application system
When multiple environments exist, procedures are required for migrating database objects and programs from environment to environment. Specific guidelines are needed to accomplish migration in a manner conducive to the usage of each environment. For example, what data volume is required for each environment and how is data integrity to be assured when testing activity occurs? Should data be migrated, or just the database structures? How should existing data in the target environment be treated—should it be kept, or overlaid with new data? Comprehensive migration procedures should be developed to address these types of questions.

The migration and turnover procedures should document the information required before any database object or program can be migrated from one environment to the next. At a minimum, information will be required about the requester, when and why the objects should be migrated, and the appropriate authorization to approve the migration. To ensure the success of the migration, the DBA should document the methods used for the migration and record the verification process.

**Design Review Guidelines**

All database applications should be subjected to a design review at various stages of their development. Design reviews are important to ensure proper application design, construction, and performance. Design reviews can take many forms. Chapter 6, “Design Reviews,” offers a comprehensive discussion.

**Operational Support Standards**

*Operational support* is defined as the part of the IT organization that oversees the database environment and assures that applications are run according to schedule. Sufficient operational support must be available to administer a database environment effectively. The operational support staff is usually the first line of defense against system problems. Program failures, hardware failures, and other problems are first identified by operational support before specialists are called to resolve the problems.

Standards should be developed to ensure that the operational support staff understands the special requirements of database applications. Whenever possible, operational support personnel should be trained to resolve simple database-related problems without involving the DBA because the DBA is a more expensive resource.
DBMS Education

Organizations using DBMS technology must commit to ongoing technical education classes for DBAs, programmers, and system administrators. Provide a catalog of available courses covering all aspects of DBMS usage. At a minimum, the following courses should be made available:

- **DBMS Overview**: a one-day management-level class that covers the basics of DBMS
- **Data Modeling and Database Design**: a thorough course covering conceptual, logical, and physical database design techniques for DAs and DBAs
- **Database Administration**: in-depth technical classes for DBAs, SAs, and systems programmers
- **Introduction to SQL**: an introductory course on the basics of SQL for every DBMS user
- **Advanced SQL**: an in-depth course on complex SQL development for DBAs and programmers
- **Database Programming**: an in-depth course for application programmers and systems analysts that teaches students how to write programs that use the DBMS

Each of these courses should be available for each DBMS installed in your organization. Furthermore, provide training for any other database-related functionality and software such as proper use of database utilities, query and reporting tools, and DBA tools.

DBMS education can be delivered using a variety of methods, including instructor-led courses, computer-based training, Web-based training, and distance learning. Sources for DBMS education include DBMS vendors, ISVs, consultants (large and small, international and local), and training specialists (such as Themis and ProTech).

Finally, be sure to make the DBMS reference material available to every user. Most vendors offer their DBMS reference manuals in an online format using Adobe Acrobat files or Windows Help. Be sure that each user is given a copy of the manuals or that they are available in a central location to minimize the amount of time DBAs will have to spend answering simple questions that can be found in the DBMS documentation.
Summary

Comprehensive advance planning is required to create an effective database environment. Care must be taken to select the correct DBMS technology, implement an appropriate DBMS upgrade strategy, develop useful database standards, and ensure the ongoing availability of education for database users. By following the guidelines in this chapter, you can achieve an effective database environment for your organization.

Nevertheless, setting up the database environment is only the beginning. Once it is set up, you will need to actively manage the database environment to ensure that databases are created properly, used correctly, and managed for performance and availability. Read on to discover how the DBA can accomplish these tasks.

Review

1. Why should database standards be implemented and what are the risks associated with their lack?
2. What are the potential risks of upgrading to a new DBMS release without a plan?
3. What is the difference between a version and a release of a DBMS?
4. Name the three TPC benchmarks and describe how they differ from one another.
5. Describe the four levels of DBMS architecture in terms of the type and nature of processing to which each is best suited.
6. What are the factors to be considered when calculating total cost of ownership (TCO) for a DBMS?
7. Name five requirements that must be planned for when installing a new DBMS.
8. Describe the difference between software that supports a DBMS release and software that exploits a DBMS release.
9. How many standard abbreviations should be supported for a single term? Why?
10. What is wrong with the following SQL code for creating a relational table? (Do not approach this question from a syntax perspective; consider it, instead, in terms of database naming standards.)

```sql
CREATE TABLE tg7r5u99_p
(c1   INTEGER NOT NULL,
c2   CHAR(5) NOT NULL,
c9   DATE)
;
```

**Bonus Question**

Your DBMS vendor, MegaDataCorp, just announced the general availability of the latest and greatest version of MDC, the DBMS you use. MDC Version 9 supports several new features that your users and developers have been clamoring for over the past year. You are currently running MDC Version 7.3. Prepare a short paper discussing your plans for upgrading to MDC Version 9, and outline the potential benefits and risks of your upgrade plan.

**Suggested Reading**


This page intentionally left blank
Index

NOTES:
Page numbers ending with an italic f (e.g. 192f) indicate tables or figures.
Numbers containing a lowercase n (e.g. 301n or 432n.5) indicate a footnote.

24/24 availability, 270–271
80/20 (Pareto) rule, 302
99.999% availability, 273–274
100 Year Archive Requirements Survey, 503n

A
Abbreviations, standards for, 96
Absolute positioning, 383–385
Access (Microsoft), 767
Access paths, 187
Accessibility, DBA to coworkers, 745
Accessing data. See Data access.
Accounts. See Logins.
ACID (atomicity, consistency, isolation, and durability) properties, 205–206
Acquire/release specification, 218
Actian Corporation, 764

Active databases, 426
Active metadata sources, 690
Actuate, 773
Adabas (Software AG), 764
Adaptive Ltd., 772
Adaptive Server Enterprise, 64
AD/Cycle (IBM), 695
Adding objects. See ALTER statements.
Adelphia, 485
Adjectives as
  attributes, 116, 124
  entities, 115
Adjust tables, data page layouts, 592
ADLC (application development life cycle), 9–10, 12
ADO.NET, 194–195
Advanced analytics, 269–270
AES (Advanced Encryption Standard), 472
Agile Modeling, 783
"Airline magazine" syndrome, 266
ALL privileges, 458–459
Allen Systems Group, 770, 772
Allied agents, 321–322
Allocation pages, data page layouts, 589
Allocation units, data page layouts, 589
ALLOW UPDATES parameter, 345
ALTER statements
  changing database structures, 250–252, 252–253
  changing management, 701–703
  limitations, 252–253
  purpose of, 250–252
ALTER TABLE statements, 436–437
AlwaysOn features, 285
Analytical processing versus transaction processing, 638–640
Analytics
  advanced, 269–270
  availability requirements, 268–270
  benefits of, 269
  DBA rule of thumb, 741–742
  tools for, 721–724
ANSI Web site, 782
APIs, SQL, 192–193
Applets, 196–197
Application DBA, 34–35
Application development life cycle (ADLC), 9–10, 12
Application Security, 770
Application servers, 209–210, 664
Application time, 179–180
Applications. See also Database applications.
  availability problems, 279–280
  backing up, 516–517
  code, design review, 238
  complexity, upgrading the DBMS, 88
  criticality of, ranking, 562–563
  DBA staffing requirements, 38–39
  development standards, 100
  infrastructure design, 193–194
  integration, 624–625
  Java, 196–197
  performance, 312, 711–713. See also Relational optimization; SQL tuning.
Approach (Lotus), 767
Aquafold, 770
Archiving database logs, 77–78, 339, 529. See also Backup.
Archiving databases
  100 Year Archive Requirements Survey, 503n
  data life cycle, 499–500
  definition, 500
  e-discovery, effects on DBA, 506–507
  hardware independence, 503–505
  hardware obsolescence, 504
  overview, 500–505
  versus purging databases, 501
  requirements, 503–505
  software independence, 503–505
  system components, 505–506
Archiving databases, data retention
  overview, 498
  scope, determining, 501–503
Associative entities, 127
Atomicity
  definition, 758
  transactions, 205–206
Atomicity, consistency, isolation, and durability (ACID) properties, 205–206
Attributes
  adjectives as, 116, 124
  data types, 116
  definition, 115
  discovering, 124–125
  domains, 116
  missing values, 119–120
  naming conventions, 116–119
  nouns as, 124
  nulls, 119–120
  prepositional phrases as, 124
  purpose, 115–116
  transforming to columns, 142–143
  values, 119
Auditing
  security, 477–478
  tools for, 717–719
Auditing databases
  common questions, 495
  comprehensive methods, 494
  data access tracking, 490–493
  guidelines, 492–493
log-based auditing, 493–495
network sniffing, 494–495
noninvasive methods, 494
overview, 490–493
parsning database logs, 493–495
privileged users, 495–496
regulatory requirements, 491
selective methods, 494
tapping requests, 494–495
techniques for, 493–495
trace-based auditing, 493–495
Authentication, 452
Authority. See also Privileges; Security.
LBAC (label-based access control), 463–465
Authority, granting privileges
centralized administration, 457
database object privileges, 459
DCL (Data Control Language), 456–457
decentralized administration, 457
overview, 456–457
procedure privileges, 460
program privileges, 460
to PUBLIC authority, 460–461
system privileges, 459–460
table privileges, 458–459
types of privileges, 457–458. See also specific types.
Authority, revoking privileges
cascading REVOKEs, 462, 468
chronology and REVOKEs, 462–463
overview, 461
Authorization. See also Privileges; Security.
availability problems, 280
database administration tasks, 24–25
database administrator, 467
database maintenance, 467
for groups, 468
operations control, 467
for roles, 466, 468
security administrator, 467, 468
system administrator, 467
Automatic summary tables, 652–653
Automation
change management, 245
DBA functions, for availability, 290–291
DBA rule of thumb, 737–739
Autonomy, distributed databases, 626
Availability. See also Downtime.
"airline magazine" syndrome, 266
change management, 246
components of, 267–268
database administration tasks, 24
DBA staffing requirements, 38
definition, 267
versus downtime, 273
driving factors, 266–267
"fast food" mentality, 266
Internet databases, 676–677
Internet time, 266
manageability, 267
overview, 265–267
versus performance, 267
recoverability, 267
reliability, 267
response time, 266
serviceability, 268
Availability, ensuring
automating DBA functions, 290–291
clustering technology, 292–295
database architecture, 296
DB2 Data Sharing, 295
high-availability features, 291
load balancing, 293
nondisruptive utilities, 288–289
NoSQL, 296
online database reorganization, 288–289
performing routine maintenance, 288–289
recommended strategy, 287
Availability, problems
application problems, 279–280
authorization problems, 280
cluster failover techniques, 276
data corruption, 280–281
data replication and propagation failures, 283
DBA mistakes, 284, 286
DBMS software failure, 279
disk outages, 278
human error, 284, 286
loss of data, 282–283
loss of database objects, 281–282
loss of entire database, 277
loss of the data center, 274–275
maintenance outages, 286–287
network problems, 275
Availability, problems (continued)
operating system failure, 279
planned outages, 286–287
recovery issues, 284
SAN failure, 278
security problems, 280
server hardware failure, 276
server performance, 283–284
standby systems, 276, 277
system memory failure, 276
unplanned outages, 286–287
Availability, requirements
24/24, 270–271
across time zones, 270–271
advanced analytics, 269–270
analytics, 268–270
business intelligence, 268–270
data warehousing, 270
decision support, 268–270ive nines, 273–274, 292
full time, 270–271
IT complexity, 271
maintenance window, 268
MTBF (mean time between failure), 273–274
overview, 268
Availability, tools. See also Standby databases.
AlwaysOn features, 285
Database Definition on Demand, 289–290
DB2 HADR (high-availability disaster recovery), 285
RAC (Real Application Clusters), 294
REORG, 288–289
Bachmann E/R method, 112
Background processes, 326
Backup. See also Archiving; Disaster planning;
Recovery.
application failure, 516–517
concurrent access issues, 525–527
COPY utility, 525
data movement, 535
data warehouse, 656–657
database administration tasks, 26–27
database failure types, 516–517
database logs, 529
database objects, 523–524
DBMS control, 524–525
differential, 521
documenting your strategy, 536
DSN1COPY utility, 552
frequency, determining, 518
full image copy, 521–523
heterogeneous database migration, 534–535
hot versus cold, 527
image copies, guidelines, 519–520
importance of, 515–516
incremental, 521–523
indexes, 524
instance failure, 516–517, 533, 550
media failure, 517, 550
object definitions, 536–537
overview, 517–520
regulatory compliance, 508
release upgrades, 534
scheduling, 531–533
SQL Server transaction logs, 530
subsystem failure, 533, 550
tools for, 714–715
transaction failure, 516–517, 550
Backup, alternatives to
disk mirroring, 556–557
exporting data, 534–535
logical backups, 534–535
redundant data, 555–556
replication, 555–556
snapshot replication, 555–556
standby databases, 277, 554–555
storage management software, 535–536, 547
symmetric replication, 555–556
unloading data, 534–535
Backup, consistency
creating a recovery point, 528–529
definition, 29
overview, 527–528
quiesce point, 528
QUIESCE utility, 528
Backup files, data integrity, 411
BACKUP LOG command, 550
Batch processing, 221–222
Benchmarks, 65–66. See also Performance.
Big Data movement, 55–66
Big Three DBMS vendors, 762
BIND command, 477
Index

857

Binding check constraints, 424
Bitemporal support, 179
Bitmap indexes, 155–156
Bitmaps, data page layouts, 589
Blind men and an elephant, 108
Block size, optimizing database performance, 364–365
Blocks, recovering, 553
Blogs, 780–781
BMC Software, 769
Booch, Grady, 113
Bradmark Technologies, 770
Bridge architecture drivers, 673
b-tree index, 154–155
Buffer pools, 78–79
Bulk data movement, 623–625. See also Distributed databases.
Bulk-logged recovery, 340, 540
Bunker Hill Corporation, 770
Business intelligence
availability, 268–270
tool vendors, 773
tools for, 721–724
Business Intelligence Network, 783
Business logic, 664–666
Business metadata, 689
Business service interruption, risk of, 561–563
Business time, 179–180
Business-critical applications, 562

C
CA Technologies, Inc., 770, 772
Cache
database log, 330
Internet structure, 330
memory usage, displaying, 413–414
procedure, 329–330, 335
program, 79
sort, 330
system performance, 328–330
Cache, data
definition, 78–79
system performance, 329–330, 332–335
Callable routines, 192
Candidate keys, indexes, 152
Candle Corporation, 770
Capacity planning, tools for, 313
Cardett Associates, 773
Cardinality, 112
Careers in DBA
DBAjobs.com, 782
demand for DBAs, 4
evaluating a job offer, 14–15
salaries, 4–6
sample job posting, 785–791
skill and knowledge requirements, 788–790
typical responsibilities, 786–788
workload, 6
Carnival blogs, 781
Cartesian products, 402
Cascading DROPS, 251, 701–703
Cascading REVOKEs, 462, 468
Cassandra, 56
Catalog query and analysis tools, 705–707
CDB Software, 770
CD/DVD storage, DBMS requirements, 77
Centralized data management model, 668
Centralized processing, 666–667
Certegy Check Services, Inc., 496
Certification
and job performance, 57
overview, 56–58
sources of, 58
Change management
automation, 245
availability, 246
checklists, 260
coordinating databases with applications, 260–261
DBA scripts, 262
free space, changing, 255–256
impact analysis, 245
indexes, recreating, 257n
intelligence, 245
overview, 243–244
perspective of the DBA, 246–247
planning analysis, 245
proactivity, 245
quick, efficient delivery, 246
reasons for change, 244
regulatory compliance, 261–262, 508
reliable, predictable processes, 246
requesting changes, 258–260
sample scenarios, 254–257
Change management (continued)
standardized procedures, 245
success factors, 244–246
tools for, 254, 701–703
Change management, database structures
adding columns, 255, 256. See also ALTER statements.
ALTER statements, 250–252, 252–253
cascading DROPs, 251
changing objects, 250–252
comparing structures, 257–258
CREATE statements, 250–252
creating objects, 250–252
in database change management, 250–253
DROP statements, 250–252
dropping objects, 250–252
overview, 250–252
removing objects. See DROP statements.
Change management, types of change
applications, 249–250
DBMS software, 248
hardware configuration, 248
logical design, 248–249
overview, 247
physical database structures, 250
physical design, 248–249
Change requests, 258–260
Check conditions, 420
Check constraints
benefits of, 420–421
binding, 424
check conditions, 420
constraint names, 420
definition, 28, 419
examples, 421–423
nulls, 423–426
versus referential integrity, 441–442
relational nulls, 423–426
rules, 424
semantic data integrity, 419–426
CHECK utility (DB2), 411
CHECKALLOC option, 413
CHECKCATALOG option, 413
CHECKDB option, 413
CHECKFILEGROUP option, 413
Checkpoint/restart, tools for, 725
CHECKTABLE option, 412
Chen E/R method, 112
Child tables, referential integrity, 433–434
CKPT (checkpoint) process, 326
Client computers, 665
Client-based drivers, 673
Client/server computing. See also Database connectivity; Network traffic.
application servers, 664
applications, types of, 667–670
business logic, 664–666
centralized data management model, 668
centralized processing, 666–667
client computers, 665
cooperative processing, 667
database management systems, 664–666
database servers, 664
decentralized user presentation model, 668
definition, 663, 665
distributed data management model, 668–669
distributed processing, 666–667
distributed user presentation model, 667–668
distributing tasks across a network, 668
fat clients, 670
file servers, 664
multitier implementation, 669–670
network traffic, 670–674
performance problems, 670–674
presentation logic, 664–666
print servers, 664
recommended hardware, 666
server computers, 665–666
software layers, 664–666
thin clients, 670
Cloud computing, effect on DBAs, 53–55
Cloud database systems, 74–75
Cluster failover techniques, 276
Cluster ratios, 369
Clustering. See also Interleaving data.
definition, 71, 94
indexes, 159–160
optimizing database performance, 356–358
shared-disk, 72
shared-nothing, 71–72
standards, 94
technology for availability, 292–295
types of, 71–72
COBIT, 509–510
CODASYL (Conference on Data Systems Languages), 754–755
Codd, E. F., 128
Code
  design review, 238
  memory usage, displaying, 413
Code generators
  creating SQL, 191–192
  SQL tuning, 405
Cogit, 770
Cold backup, 527
Column-oriented data models, 756
Columns
  adding, 251, 255, 256
  constraints, 144
  deleting, 251
  fixed-length, 144
  identity property, 145
  nullability, specifying, 144
  ordering, 146
  transforming attributes to, 142–143
  unique identifiers for, 145
  variable length, 144
COM, SQL, 193
Combined tables, optimizing database performance, 356
COMMIT statements
  batch processing, 221–222
  saving transaction changes, 205
  SQL tuning, 404–405
  two-phase, 631
COMMITTED READ isolation, 216–217
Communications, standards, 98
Compliance, tools for, 716–721
Comprehensive auditing methods, 494
Compression
  data warehouse, 644
  database design, 149–150
  disaster planning, backup, 575
  optimizing database performance, 361–362
  performance tuning, 314
  tools for, 726–727
Computer Associates, 773
Computer Associates International, 771
Compuware Corporation, 770
Conceptual data modeling, 125–128
Conceptual design review, 233–235
Concurrency, unloading data, 619
Concurrency control, purpose of, 758
Conference on Data Systems Languages (CODASYL), 754–755
config.ora file, 325
Configuring the DBMS. See also Installing the DBMS; Upgrading the DBMS. default parameters, 80
  performance tuning. See System performance, DBMS installation and configuration.
  system parameters, 80
Confio Software, 770
Connection pooling, 674
Connectivity. See Database connectivity.
Consistency, transactions, 206
Consistency checking, 412–413
Constraint names, 420
Constraints
  check, 28
  columns, 144
  data integrity, 28
  enforcing while loading data, 615
  referential, 28
  unique, 28
Consultants, Web sites, 779–780
Consumption sources, 330–331
Contention
  performance factor, 301
  performance monitoring and tuning, 23
  system performance, 341–342
Contingency planning. See Disaster planning.
Control files, 325
Converting data types while loading data, 616
Cooperative processing, 667
COPY utility, 525
Copying data. See also Loading data; Unloading data.
  bulk data movement, 623–625
  EXPORT utility, 622–623
  IMPORT utility, 622–623
  to multiple databases. See Distributed databases.
CoSort/IRI, 773
Cost-based optimization versus rule-based, 344
Costs of
  CPU, relational optimization, 376
  of data breaches, 450
Costs of (continued)
  I/O, relational optimization, 376
  ownership, 67
  performance, across the ADLC, 307
  poor data quality, 488, 489
  regulatory compliance, 485
  regulatory non-compliance, 488
  upgrading the DBMS, 84, 85
CouchDB databases, 56, 766
CPU parallelism, 391
CREATE statements, 250–252
CREATE TABLE statements, 436–437
Creating objects. See CREATE statements.
Critical applications, 562–563
Criticality of data, ranking, 562–563
Cursor, SQL, 190
Cursor stability, 216–217

D
DA (data administration), 15–18, 19
DAMA (Data Management Association), Web site, 783
Darwin Professional Underwriters, 450
Data. See also Backup; Disaster planning;
  Recovery.
    abstraction levels, DBMS, 757
    breaches, 449–450
    cleansing, 645–649
    compression, 644
    content, 654
    corruption, 280–281
    definition, 686
    dictionaries, 695–696
    encryption. See Encryption.
    freshness, 654
    independence, 757
    latency, 574, 654
    length, semantic data integrity, 417–418
    moving. See Moving data.
    placement, distributed databases, 629
    privacy, policies and statutes, 11
    profiling, 489, 719–720
    protection, tools for, 716–721
    quality, 488–489, 648. See also Data integrity.
    rate of growth, 581
    record layouts, data page layouts, 590
    replication and propagation failures, 283
    rows, data page layouts, 588–589
    security, DBMS, 758
    stewardship, 688
    usage, 655
Data access
  DBMS, 758–759
  to noncurrent versions, 177–180
  tracking, 490–493
Data administration (DA), 15–18, 19
Data Administration Newsletter, 782
Data administration standards, 98–99
Data and Technology Today blog, 781
Data Control Language (DCL), 456–457
Data Definition Language (DDL), 177, 250–252
Data Dictionary. See System catalog.
Data Encryption Standard (DES), 472
Data files, 325
Data governance. See also Regulatory compliance.
  IT Governance Institute, 509
  overview, 489–490
  tools for, 716–721
Data Governance blog, 781
Data integrity. See also Data, quality.
  backup consistency, 29
  constraints, 28
  data cleansing, 646
  data warehouse, 646
  database administration tasks, 27–29
  DBMS, 758–759
  index consistency, 29
  pointer consistency, 29
  problems, loading data, 615
  types of, 409–410
Data integrity, database structure
  backup files, 411
  consistency checking, 412–413
  database checking, 413
  headers, 411
  memory usage, 413–414
  page header corruption, 411
  problem management, 411–414
  types of problems, 410–411
  utilities for, 411–414
Data integrity, semantic
  check constraints, 419–426
  data length, 417–418
data types, 417–418
default values, 419
DQS (Data Quality Services), 415
entity integrity, 416–417
example, 28
overview, 414–415
primary key constraints, 416–417
triggers, 426–433
UDT (user-defined data types), 418–419
unique constraints, 417
unique entity identification, 416–417
Data life cycle, 499–500
Data Management Association (DAMA), Web site, 783
Data mart, 638
Data masking and obfuscation. See also Encryption.
definition, 496–497
encryption, 497
nulling out, 498
number and date variance, 497
shuffling, 497
substitution, 497
table-to-table synchronization, 498
techniques for, 497–498
tools for, 720
Data mining, 639
Data modeling
concepts, 108–113
conceptual, 125–128
DA (data administration), 17–18
DBA tasks, 33
definition, 107
enterprise data model, 109
E/R (entity relationship diagram), 110–113
homonyms, 118
importance of, 107
issues, 135–136
logical, 125–128
physical, 125–128
rules for, 110
synonyms, 118
tool vendors, 771–772
tools for, 700–701
types of, 125–128
Data modeling, components
attributes, 115–119
entities, 113–115
keys, 120–122
relationships, 122–123
Data models
CODASYL (Conference on Data Systems Languages), 754–755
column-oriented, 756
DBMS, 754–755, 756
definition, 754
denormalization, 163
hierarchical, 754–755
network, 754–755
NoSQL system, 756
object-oriented, 754–755
operations, 754
relational, 754–755
relations, 755
structure, 754
Data page layouts
allocation pages, 589
allocation units, 589
bitmaps, 589
data record layouts, 590
data rows, 588–589
header information, 592
index key values, 592
index page layouts, 592–594
offset and adjust tables, 592
offset tables, 588–590
overview, 588–589
page header, 588–589
page pointer, 592
row data, 590
row header, 590
row length, 592
sample, 589
space page map, 589
table size, calculating, 591–592
transaction logs, 594–595
Data resource management (DRM), 40–42
Data retention
DBA source materials, rule of thumb, 736–737
disaster planning backup, 571
Data sets. See Files and data sets.
Data spaces, database design, 148
Data types
attributes, 116
converting, while loading data, 616
semantic data integrity, 417–418
Data warehouse
administrators, 36–37
analytical versus transaction processing, 638–640
availability, 270
data mining, 639
definition, 637–638
design, 641–644
dimensions, 639
DSS (decision support systems), 639
facts, 639
Information Center, 640
metadata, 688
OLAP (online analytical processing), 639
OLAP versus OLTP, 640
tools for, 721–724
Data warehouse, administering
backup and recovery, 656–657
data cleansing, 645–649
data compression, 644
data content, 654
data freshness, 654
data integrity problems, 646
data latency, 654
data movement, 644–645
data quality issues, 648
data usage, 655
data warehouse design, 641–644
denormalization, 643
financial chargeback, 655–656
focus on technology, 641
identifying unused data, 655
meeting business requirements, 657
metadata, 654
operational problems, 648–649
overview, 640–641
purging data, 655
scalability, 649
size issues, 649
snowflake schema, 643
standardizing default values, 647
star schema, 641–643
Data warehouse, performance
automatic summary tables, 652–653
data management, 650
extract performance, 650
indexes, 651
materialized query tables, 652–653
materialized views, 653
monitoring, 652
perspectives on, 650
query performance, 650
server performance, 650
The Data Warehousing Information Center, 783
The Data Warehousing Institute, 783
Database administration. See also DBA (database administrator).
importance of, 3–4
management discipline, 9–14
Database administration tasks. See also specific tasks.
availability, 24
backup and recovery, 26–27
data integrity, 27–29
database design, 21–22
governance and regulatory compliance, 26
jack-of-all-trades, 29–31
performance monitoring and tuning, 22–23
security and authorization, 24–25
Database administrator (DBA). See DBA (database administrator).
Database applications, designing. See also Design review; SQL (Structured Query Language); Transactions.
ADO.NET, 194–195
application infrastructure, 193–194
hardware environment, 193–194
issues, 186
J2EE (Java 2 Enterprise Edition), 195–196, 198
Java program types, 196–197
.NET framework, 194–195, 198
overview, 185–186
Ruby on Rails, 198
software environment, 193–194
Database architects, DBAs as, 32–33
Database connectivity. See also Client/server computing; Internet; Network traffic; Web services.
application servers, 664
business issues, 662
client/server computing, 663–666
database servers, 664
downsizing, 662
file servers, 664
history of, 661–662
print servers, 664
rightsizing, 662
upsizing, 662
Database Definition on Demand, 289–290
Database design. See also Indexes; Views.
  compression, 149–150
  data spaces, 148
  database administration tasks, 21–22
  domains, transforming to data types, 143–144
  for e-business, 677–680
  entities, transforming to tables, 142
  filegroups, 149
  logical model to physical database, 141–150
  overview, 141–142
  physical data structure, 147–150
  primary keys, 144
  raw files, 149
  referential constraints, 146–147
  referential integrity, 146–147
  row size, specifying, 148
  storage requirements, 148
  tablespaces, 148
  temporal requirements, 177–180
Database design, columns
  constraints, 144
  fixed-length, 144
  identity property, 145
  nullability, specifying, 144
  ordering, 146
  transforming attributes to, 142–143
  unique identifiers for, 145
  variable length, 144
Database drivers, 672–674
Database environments. See also specific environments.
  education, 101
  integration testing, 101
  multiplatform issues, 42–43
  production versus test, 44–46
  quality assurance testing, 101
  unit testing, 101
  user acceptance testing, 101
Database files, Oracle, 325
Database gateways, 671–672
Database ID. See Users, names.
Database logs
  active, 529
  archiving, 77–78, 339, 529
  backing up, 529, 530
  bulk-logged recovery, 340
  configuring, 338–339, 340
  DBMS, 758
  definition, 336
  disabling, 341
  disabling while loading data, 617
  disaster planning, backup, 570–571
  filling up, 339
  full recovery, 340
  log archival process, 529
  log off-loading, 339
  log-based auditing, 493–495
  "out of space" conditions, 339–341
  placement for optimizing performance, 363
  during recovery, 338, 340
  recovery models, 340
  selecting candidates for, 339–341
  simple recovery, 340
  system checkpoints, 337
  system performance, 336–341
  transaction logs, 336
  types of information on, 337
  write-ahead, 337
Database management system (DBMS). See DBMS (database management system).
Database performance. See also Optimizing database performance; Reorganizing databases; SQL tuning.
  80/20 (Pareto) rule, 302
  versus availability, 267
  common problems, 302–304
  contention, 301
  cost, across the ADLC, 307
  definition, 23, 300–302
  diagnosing, 302–304
  estimating, 307–308
  guidelines, 315–316
  historical trends, 308
  main factors, 301–302
  overview, 299–302
  resources, 301
  SLM (service-level management), 308–311
  throughput, 301
  tools for, 711
  tracker tables, 308
  tuning SQL, 303–304
  workload, 301
Database performance, managing
analysis, 305
components of, 304–306
definition, 304–306
versus monitoring, 304–306
overview, 304–306
reactive versus proactive, 306
Database performance, monitoring
contention, 23
database administration tasks, 22–23
factors affecting, 22–23
resources, 22
throughput, 22
tools for, 313
workload, 22
Database performance, tuning
application, 312
caching, 314
capacity planning, 313
compression, 314
contention, 23
database, 312
database administration tasks, 22–23
estimation, 313
factors affecting, 22–23
monitoring, 313
reorganizing databases, 314
resources, 22
sorting, 314
SQL analysis and tuning, 313
system, 311
throughput, 22
tools for, 313–315
workload, 22
Database servers
definition, 664
hosting, 675
location, upgrading, 88
Database Site portal, 781
Database Trends and Applications, 779
Database views. See Views.
Database wire drivers, 674
Database writer (DBWR) process, 326
Database-coupled application logic, 46–50
Databases
architecture for availability, 296
change management. See Change management.
checking for data integrity, 413
comparison, tools for, 703–704
DBA staffing requirements, 37
versus DBMS, 7–8
definition, 7, 753–754
dropping, 250–252
links, 94
logic, managing, 46
maintenance, authorization, 467
management systems, 664–666
object privileges, 459
objects, backing up, 523–524
Oracle, 325
structures, comparing, 257–258
tools for. See Tools.
users, security, 455–456
DataBee, 771
Datexanamic, 771
DB2 (IBM)
blogs, 780
IDUG (International DB2 User Group), 740, 783
nonstandard database objects, 94
vendor contact, 63
Web site, 778
DB2 Catalog. See System catalog.
DB2 Data Sharing, 295
DB2 EDM pools, 335
DB2 HADR (high-availability disaster recovery), 285
DBA (database administrator). See also Database administration.
authorization, 467
versus DA, 15–18, 19, 21
a day in the life of, 12–14
demand for, 4
description, 1–3
job scope, defining, 42–43
jobs. See Careers in DBA.
multiplatform issues, 42–43
reporting structures, 40–42
responsibilities, 12, 786–788
versus SA, 21
skill and knowledge requirements, 788–790
staffing, 37–40
standards and procedures, 98–99
tools for. See Tools, for DBAs.
typical responsibilities, 786–788
workload, 6, 12–14
DBA (database administrator), rules of thumb
accessibility to coworkers, 745
analysis, 741–742
automation, 737–739
being prepared, 743
calm in the face of adversity, 742–743
documenting your work, 735–736
effective use of resources, 745–746
focus, 741–742
investing in professional advancement, 747–748
retaining source materials, 736–737
sharing knowledge, 739–741
simplification, 741–742
technical education, 746–747
Twitter, as a resource, 741
understanding your business, 743–745
user group associations, 740
DBA (database administrator), types of
application, 34–35
data modeler, 33
data warehouse administrator, 36–37
database analysts, 33
database architects, 32–33
performance analysts, 36
system, 31–32
task-oriented, 36
technical focus versus business, 31–32

DBA Direct, 779
DBAjobs.com, 782
dBase, 767
DBCC utility, options, 412–414
DBE Software, 771
DBI Software, 771
dbMaestro, 771

DBMS (database management system)
architectures, 68–71
atomicity, 758
availability problems, 279
buying. See Vendors, DBMS.
clustering. See Clustering.
concurrency control, 758. See also Locking.
data abstraction levels, 757
data access, 758–759
data independence, 757
data integrity, 758–759
data models, 754–755, 756
data security, 758

versus database, 7–8
database logging, 758
definition, 8, 753–754
departmental architecture, 70
durability, 758
time period, 758
enterprise architecture, 69
mobile architecture, 70
organizational strategy. See Strategies for DBMS.
personal architecture, 70
proliferation, 73
upgrading, 87–88
vendors. See Vendors, DBMS.

DBWR (database writer) process, 326
DCL (Data Control Language), 456–457
DDL (Data Definition Language), 177, 250–252
Deadlock detection, 341
Deadlocks, 214–215, 342
Debugging, tools for, 726
Decentralized user presentation model, 668
Decision support, availability, 268–270
Decision support systems (DSS), 639
Defragmenting indexes, 413
DELETE privileges, 458–459
DELETE rule, 435–436
DELETE statements
modifying temporal data, 180
in triggers, 429
DELETE trigger, 438–441
Deleting objects. See also DROP statements.
columns, 251
purging data, data warehouses, 655
purging databases, versus archiving, 501
rows, 435–436

Denormalization
benefits, evaluating, 175
combined tables, 168
data warehouse, 643
derivable data, 170–171
description, 160–161
evaluating the need for, 161–162, 174–175
hierarchies, 171–173
identifying candidates for, 162–163
Internet databases, 680
issues, 161–162
logical data models, 163
mirror tables, 165
optimizing database performance, 355–356
Denormalization (continued)
overview, 161–163
physical implementation requirements, 173
prejoined tables, 164
redundant data, 168–169
repeating groups, 169–170
report tables, 164–165
speed tables, 172–173
split tables, 165–166
splitting text columns, 166–168
types of, 174
Density, relational optimization, 377
Departmental DBMS architectures, 70
Deprecated features, 85n.7
Derivable data, 170–171, 356
Derived data, storing versus calculating, 170–171
DES (Data Encryption Standard), 472
Design review
guidelines, 102, 228–229
output, 239–240
overview, 227–228
purpose of, 228
Design review, participants
knowledge and skills required, 232
leader of, 229–230
mediator, 230–231
mentorship and knowledge transfer, 240–241
recommended personnel, 231
remote staff, 232
scribe, 230
Design review, types of
in the ADLC, 234
code, 238
conceptual, 233–235
logical, 235
organizational, 237
overview, 233
physical, 236
post-implementation, 239
pre-implementation, 239
Designing
applications. See Database applications, designing.
databases. See Database design.
Determinant, 135
Devices, naming conventions, 364
Diagramming entity relationships. See E/R (entity relationship diagram).
Differential backup, 521
Dimensions, data warehouse, 639
Direct index lookup, 383
Dirty read, 216–217
DISABLE option, 477
Disabling
database logs, 341
passwords, 453
Disaster, definition, 559–560
Disaster planning. See also Backup; Recovery.
business-critical applications, 562
critical applications, 562–563
criticality of data, ranking, 562–563
lengthy outages, 568
need for, 559–563
noncritical applications, 563
prevention, 575–576
required applications, 563
very critical applications, 562
Web sites about, 576
Disaster planning, backup
compression, 575
data latency, 574
data retention, 571
database logs, 570–571
encryption, 575
important files and data, 574–575
indexes, 570
order of recovery, 574
over a WAN (wide-area network), 573
post-recovery image copies, 575
remote mirroring, 573
standby databases, 573
storage management software, 572–573
on tape, 570–571
Disaster planning, recovery
off-site locations, 564
personnel, 569
plan content, 566
recovery site, choosing, 564
rehearsing, 567–569
team members, 569
testing your plan, 567–569, 574
written plans, 564–566
Disaster planning, risk
assessing, 561–563
business service interruption, 561–563
categories of, 561
financial loss, 561–563
legal responsibilities, 561–563
Discovering attributes and entities, 124–125
Disk drives
MTBF (mean time between failures), 580
overview, 580
Disk storage
DBMS requirements, 76–78
SSDs (solid state devices) versus traditional disks, 323–324
system performance, 322–324
Disks. See also Storage management.
allocation, optimizing database performance, 364
fragmentation, 595
JBOD (just a bunch of disks), 604
m mirroring, as a backup/recovery alternative, 556–557
outages, 278
performance improvement, 584–585
raw partitions versus file systems, 586–587
SCSI (small computer system interface), 605
short-stroking, 584–585
size terminology, 582
storage management option, 596
striping, 597
usage spikes, 580
DISTINCT clause, 387
Distributed data
accessing, 630–631
distributed request, 631
distributed unit of work, 631
DRDA (Distributed Relational Database Architecture), 629–630
placement for optimum performance, 363–364
RDA (Remote Database Access), 629–630
remote requests, 630–631
remote unit of work, 630–631
standards, 629–630
two-phase COMMIT, 631
Distributed data management model, 668–669
Distributed databases. See also Bulk data movement; Copying data; Moving data.
autonomy, 626
characteristics of, 626
data placement, 629
definition, 626
environment, setting up, 627–629
federated multidatabase schemes, 627
isolation, 626
performance problems, 632–633
system performance, 344
transparency, 626
unfederated multidatabase schemes, 627
usage guidelines, 629
Distributed processing, 666–667
Distributed request, 631
Distributed unit of work, 631
Distributed user presentation model, 667–668
Distributing tasks across a network, 668
Document Type Definition (DTD), 204
Documentation
DBA activities, 735–736
online standards manuals, 727–728
Domains
attributes, 116
transforming to data types, 143–144
Downsizing, and database connectivity, 662
Downtime. See also Availability.
versus availability, 273
cost of, 271–273
DBA staffing requirements, 38
negative publicity, 272
DQS (Data Quality Services) (Microsoft), 415
DRDA (Distributed Relational Database Architecture), 629–630
Drivers
JDBC, 673–674
ODBC, 192, 673
DRM (data resource management), 40–42
DROP statements
cascading DROPs, 251, 701–703
in database change management, 250–252
Dropped database objects, recovering, 552–553
Dropping
database objects, 250–252
tables, 250–252
DSNICOPY utility, 552
DSNZPARM parameter, 80n.4
DSS (decision support systems), 639
DTD (Document Type Definition), 204
Duplicate values, relational optimization, 377
Durability
DBMS, 758
transactions, 206
Dynamic SQL, 201
E
Ebbers, Bernard, 485
E-business. See also Internet.
effects on DBAs, 50–51
infrastructure, 52
E-discovery, effects on DBA, 506–507
EDM pools, 335
Education
database environment, 101
recommended courses, 103
E-Government Act, 484–485
80/20 (Pareto) rule, 302
Elephant and blind men, 108
Embarcadero Technologies, 770–771
Embedded SQL, 191–192, 201
ENABLE option, 477
Encoding scheme, specifying, 620
Encryption. See also Data masking and obfuscation.
data at rest, 472
data in transit, 472
data masking and obfuscation, 497
disaster planning, backup, 575
overview, 470, 472
techniques for, 472
transparent, 473
wallets, 473
End-to-end performance, tools for, 713–714
Enron Corporation, 485
Enterprise data model, 109
Enterprise DBMS architectures, 69
Entities. See also Relationships.
adjectives as, 115
associative, 127
definition, 113
discovering, 124–125
instances, 115
naming conventions, 113
nouns as, 115, 124
transforming to tables, 142
Entity integrity, 416–417
Entity occurrences, 115
Environments, system. See Database environments.
Epsilon, data breach, 449
E/R (entity relationship diagram)
Bachmann method, 112
cardinality, 112
Chen method, 112
definition, 110
diagramming methods, 111–113
example, 111
Information Engineering method, 112
Martin method, 112
Ross method, 112
Rumbaugh method, 112
UML (Unified Modeling Language), 113
Error correction coding, 599
Estimating
memory requirements, 331–332
performance, 307–308
tools for, 313
ETL (extract, transfer, load), 623–625, 721–723
ETL tool vendors, 773
E-vailability, 676–677
EXCEPT clause, 388
Exclusive locks, 213
EXECUTE privileges, 460
EXPLAIN command, 394–398, 712
Exploiting versus supporting, 91
EXPORT utility, 622–623. See also UNLOAD utility.
Exporting data
backup/recovery alternative, 534–535
EXPORT utility, 622–623
EXtensible Markup Language (XML), 204
External security, 478–480
Extract performance, 650
F
Fabian Pascal's site, 779
Facts, data warehouse, 639
Fallback planning, 92
"Fast food" mentality, 266
Fat clients, 670
Fault tolerance, 601–602
Federal Rules of Civil Procedure (FRCP), 506–507
Federated multidatabase schemes, 627
Fiber channel, storage management option, 605
File extents, reorganizing databases, 366
File servers, database connectivity, 664
File systems versus raw partitions, 586–587
Filegroups
database design, 149
definition, 94
standards, 94
Index

FileMaker, 767
Files and data sets. See also Storage management.  
optimal placement, 584–586  
overview, 583–584  
placement and allocation, optimizing database  
performance, 362–364  
temporary database files, 587  
Fill factor. See Free space.  
Financial chargeback, 655–656  
Financial loss, risk of, 561–563  
Financial Modernization Act of 1999, 484–485  
Firing triggers, 428–429  
FISMA (Federal Information Security Management  
Act), 485  
Five-nines availability, 273–274, 292  
Fixed-length columns, 144  
Fixpacks and maintenance, 480–481  
Floating-point data  
loading, 616  
unloading data, 620  
Focus, DBA rule of thumb, 741–742  
FORCEPLAN option, 398–399  
Forcing access path choices, 398–399  
Foreign key perspective, 434–435  
Foreign key values, 434–436  
Foreign keys, indexes, 151  
Forrester Research, 450  
Fragmentation  

disks, 595  
indexes, 595  
reorganizing databases, 366  
FRCP (Federal Rules of Civil Procedure), 506–507  
Free space  
changing, 255–256  
optimizing database performance, 360–361  
FREEPAGE parameter, 360  
Full image copy, 521–523  
Full recovery, 340, 540  
Full-time availability, 270–271  

G  
Gerstner, Lou, 581  
Giga Research Group, 582  
GLB (Gramm-Leach-Bliley) Act, 484–485, 491  
Governance. See Data governance.  
Grandite, 772  
GRANT statements, 456–457  
Granularity  
locks, 210–211, 219–220  
triggers, 431–432  
GROUP BY clause, 388  
Groups, authorization, 468  
"Guilty until proven innocent," 13

H  
Hadoop databases, 766  
Hardware  
configuration for system performance, 322–324  
environment, designing, 193–194  
issues, strategies for, 73–74  
requirements, installing the DBMS, 76  
Hash function, 389, 390  
Hash joins, 379  
Hashed access, 389–390  
Hashing, randomizing, 158  
HBase databases, 56, 766  
Header information, data page layouts, 592  
Headers, data integrity, 411  
Health Net Federal Services, 450  
Heterogeneous database migration, 534–535  
Hibernate, ORM library, 200  
Hierarchical data models, 754–755  
Hierarchies, 171–173  
HIPAA (Health Insurance Portability and  
Accountability Act), 484–485, 491  
Hit Software, 771  
Homonyms, in data modeling, 118  
Horizontal restriction, 469  
Hostile databases, 678–679  
Hosting database servers, 675  
Hot backup, 527  
Human error, availability problems, 284, 286  
100 Year Archive Requirements Survey, 503n  
Hybrid joins, 379

I  
IBM Corporation  
DB2 Web site, 778  
DBMS vendor, 63–64, 762, 765  
rate of data growth, 581  
tool vendor, 772, 773  
IBM Data Management, 779  
IDC Corporation, 581
Index

IDE (integrated development environment), 191–192
Identity property, columns, 145
Identity values, system performance, 344
Idera, 771
IDMS (Cullinet), 765
IDUG (International DB2 User Group), 740, 783
IIUG (International Informix Users Group), 740, 783
ILM (information life cycle management), 606
Image copies
  backup guidelines, 519–520
  backups, unloading data from, 619
  disaster planning, post-recovery, 575
Impact analysis, change management, 245
IMPORT utility, 622–623. See also LOAD utility.
Importing data, 622–623
IMS (IBM), 765
Incremental backup, 521–523
Independent Oracle Users Group (IOUG), 740
Index covering, 386–387
Indexed access, 382–389
Indexes
  absence of, 151
  avoiding, 354
  avoiding sorts, 387–388
  backup, 524
  based on workload, 152
  bitmap, 155–156
  b-tree, 154–155. See also Partitioned index;
    Reverse key index.
  candidate keys, 152
  clustering, 159–160
  consistency, 29
  costs of, 153–154
  data warehouse, 651
  defragmenting, 413
  designing, 150–154
  disaster planning, backup, 570
  file placement, 584
  foreign keys, 151
  fragmentation, 595
  indexing by object, 152
  index-only access, 152
  key values, 592
  leaf pages, 155
  locking, 212
  nodes, 155
  optimal number per table, 353
  optimizing database performance, 352–355
  ordered, 157
  overloading, 355
  page layouts, 592–594
  partitioned, 157. See also b-tree index.
  primary keys, 151
  recovery, 550–551
  recreating, 257n
  reorganizing, 369–370
  reverse key, 156–157. See also b-tree index.
  screening, 386
  size, calculating, 592–594
  sorting, 152
  table scans, 151
  unused, dropping, 153
Indexing by object, 152
Index-only access, 152, 386–387
Informatica, 773
Information, definition, 687
Information Builders, 773
Information Center, 640
Information Engineering, E/R method, 112
Information life cycle management (ILM), 606
Information Management, 779
Informix, 763
  IIUG (International Informix Users Group), 740, 783
  vendor contact, 64
  Web site, 778
InfoTel Corporation, 771
Ingres, 763–764
INIT.ORA, 80n.4
Inner table, 379
INSERT privileges, 458–459
INSERT rule, 434–436
INSERT statements
  modifying temporal data, 180
  recording in the transaction log, 341
  in triggers, 429
INSERT trigger, 438–441
Installing the DBMS. See also Configuring the DBMS; Upgrading the DBMS.
  connecting to infrastructure software, 81
  hardware requirements, 76
  memory requirements, 78–79
in multiple environments, 82
performance tuning. See System performance,
DBMS installation and configuration.
prerequisites, 75–76
storage requirements, 76–78
verifying the install, 81
Instance failure, backup, 516–517, 533, 550
Instances
entities, 115
Oracle databases, 325
INSTEAD OF trigger, 432
Integrated development environment (IDE),
191–192
Integrated metadata sources, 690
Integration testing, 101
Integrity. See Data integrity; RI (referential integrity).
Intelligence, change management, 245
Intent locks, 213
Interconnected databases, 676–680
Interleaving data. See also Clustering.
optimizing database performance, 360
performance design, 160
International DB2 User Group (IDUG), 740, 783
International Informix Users Group (IIUG), 740, 783
International issues, Internet databases, 679
International Oracle Users Group, 783
International Sybase User Group (ISUG), 740, 783
Internet, and e-business
effect on DBAs, 50–52
infrastructure, 52
Internet, database connectivity. See also Web services.
availability, 676–677
denormalization, 680
designing for e-business, 677–680
effect on DBA duties, 676–680
e-availability, 676–677
hostile databases, 678–679
hosting database servers, 675
interconnected databases, 676–680
international issues, 679
Internet time, 677
key design, 679
normalization, 679
RAD (rapid application development), 677
Internet resources
blogs, 780–781
database portals, 781–782
industry standards, 782
jobs, 782
mailing lists, 776–778
Usenet newsgroups, 775–776
Internet resources, Web sites
consultants, 779–780
magazines, 778–779
Mullins, Craig, 780
user group associations, 740
vendors, 778
Internet time, 266, 677
INTERSECT clause, 388
Invasive performance tools, 710
I/O parallelism, 390–391
IOUG (Independent Oracle Users Group), 740
ISO Web site, 782
Isolation
distributed databases, 626
levels, 216–218
transactions, 206
ISQL, 81n.5
ISUG (International Sybase User Group), 740, 783
IT complexity, and availability, 271
IT Governance Institute, 509
J
J2EE (Java 2 Enterprise Edition), 195–196, 198
Jack-of-all-trades, 29–31
Jacobson, Ivar, 113
Java
applets, 196–197
applications, 196–197
choosing a program type, 196–197
Hibernate, ORM library, 200
LINQ (Language Integration Query), 200
NHibernate, ORM library, 200
program types, 196–197
servlets, 196–197
JBOD (just a bunch of disks), 604
JDBC (Java Database Connectivity)
bridge architecture drivers, 673
client-based drivers, 673
database wire drivers, 674
drivers, 673–674
JDBC (Java Database Connectivity) (continued)
  network protocol architecture drivers, 673–674
  overview, 192–193
  Pure Java drivers, 673–674
  Type 1 drivers, 673
  Type 2 drivers, 673
  Type 3 drivers, 673–674
  Type 4 drivers, 673
Job scheduling, security, 479
Jobs in DBA. See Careers in DBA.
Join order, 381
Joining tables
  hash join, 379
  hybrid join, 379
  inner table, 379
  join order, 381
  merge-scan join, 379–380
  nested-loop join, 379–380
  outer table, 379
  qualifying rows, 380
  relational optimization, 379–381
Joins, SQL, 189

K
Kernel memory usage, displaying, 413
Keys
  candidate, 115, 121
  description, 120
  designing for Internet connectivity, 679
  foreign, 121–122
  primary, 121
Knowledge, definition, 687
Kozlowski, Dennis, 485

L
Lay, Ken, 485
LBAC (label-based access control), 463–465
Leader, design review, 229–230
Leaf distance, reorganizing databases, 370
Leaf pages, 155
Legal responsibilities, risks, 561–563
LGWR (log writer) process, 326
Life cycle
  ADLC (application development life cycle), 9–10, 12
  data, 499–500
LIKE logical operator, 403–404
LIMIT parameter, 620–621
LINQ (Language Integration Query), 200
List servers, 777
Listservs. See Mailing lists.
Load balancing, 293
LOAD parameters, 620
LOAD utility, 614–618, 621–622. See also IMPORT utility.
Loading data. See also Unloading data.
  for application test beds, 621–622
  converting data types, 616
  data integrity problems, 615
  describing the input file, 615
  disabling logging, 617
  efficiency, 617–618
  enforcing constraints, 615
  firing triggers, 615
  floating-point data, 616
  LOAD utility, 614–618, 621–622
  nulls, 616
Lock duration
  acquire/release specification, 218
  definition, 215
  isolation level, 216–218
  SKIP LOCKED DATA parameter, 219
  skipping locked rows, 219
Lock escalation, 219–220
Lock suspensions, 341
Locking
  COMMITTED READ isolation, 216–217
  cursor stability, 216–217
  deadlocks, 214–215
  description, 210
  dirty read, 216–217
  exclusive locks, 213
  granularity, 210–211, 219–220
  index entries, 212
  intent locks, 213
  levels. See Granularity.
  minimizing problems, 220
  passwords, 453
  phantoms, 218
  read locks, 212–213
  REPEATABLE READ isolation, 217
  SERIALIZABLE isolation, 218
  shared locks, 212–213
<table>
<thead>
<tr>
<th>Page</th>
<th>Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>341–342</td>
<td>system performance, 873</td>
</tr>
<tr>
<td>213–214</td>
<td>time outs, 873</td>
</tr>
<tr>
<td>212–213</td>
<td>types of locks, 873</td>
</tr>
<tr>
<td>216–217</td>
<td>UNCOMMITTED READ isolation, 873</td>
</tr>
<tr>
<td>213</td>
<td>update locks, 873</td>
</tr>
<tr>
<td>212–213</td>
<td>write locks, 873</td>
</tr>
<tr>
<td>326</td>
<td>Log writer (LGWR) process, 873</td>
</tr>
<tr>
<td>534–535</td>
<td>Logical backups, 873</td>
</tr>
<tr>
<td>757</td>
<td>Logical data independence, 873</td>
</tr>
<tr>
<td>125–128</td>
<td>Logical data modeling, 873</td>
</tr>
<tr>
<td>235</td>
<td>Logical design review, 873</td>
</tr>
<tr>
<td>141–150</td>
<td>Logical models, converting to a physical database, 873</td>
</tr>
<tr>
<td>470</td>
<td>Logic-oriented security, 873</td>
</tr>
<tr>
<td>453, 455</td>
<td>administration, rules of thumb, 873</td>
</tr>
<tr>
<td>455–456</td>
<td>definition, 873</td>
</tr>
<tr>
<td>455</td>
<td>limiting, 873</td>
</tr>
<tr>
<td>452–453</td>
<td>required information, 873</td>
</tr>
<tr>
<td>282–283</td>
<td>Loss of data, 873</td>
</tr>
<tr>
<td>274–275</td>
<td>data center, 873</td>
</tr>
<tr>
<td>281–282</td>
<td>database objects, 873</td>
</tr>
<tr>
<td>277</td>
<td>entire database, 873</td>
</tr>
<tr>
<td>767</td>
<td>Lotus, 873</td>
</tr>
<tr>
<td>295n</td>
<td>LPARS (Logical PARtitions), 873</td>
</tr>
</tbody>
</table>

**M**

<table>
<thead>
<tr>
<th>Page</th>
<th>Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>777–779</td>
<td>Magazines, Web sites, 873</td>
</tr>
<tr>
<td>776–778</td>
<td>Mailing lists, 873</td>
</tr>
<tr>
<td>596</td>
<td>Main-memory database management systems (MMDBMSs), 873</td>
</tr>
<tr>
<td>563</td>
<td>Memory, requirements</td>
</tr>
<tr>
<td>413–414</td>
<td>MEMUSAGE option, 873</td>
</tr>
<tr>
<td>379–380</td>
<td>Merge-scan joins, 873</td>
</tr>
<tr>
<td>624–625</td>
<td>Message queuing software, 873</td>
</tr>
<tr>
<td>624–625</td>
<td>Messaging software, 873</td>
</tr>
<tr>
<td>690</td>
<td>Metadata</td>
</tr>
<tr>
<td>689</td>
<td>business metadata, 873</td>
</tr>
<tr>
<td>686</td>
<td>data, definition, 873</td>
</tr>
<tr>
<td>695–696</td>
<td>data dictionaries, 873</td>
</tr>
<tr>
<td>688</td>
<td>data stewardship, 873</td>
</tr>
<tr>
<td>654</td>
<td>data warehouse, 873</td>
</tr>
<tr>
<td>688</td>
<td>data warehousing, 873</td>
</tr>
<tr>
<td>16, 488</td>
<td>definition, 873</td>
</tr>
<tr>
<td>16–17</td>
<td>examples, 873</td>
</tr>
<tr>
<td>687</td>
<td>information, definition, 873</td>
</tr>
<tr>
<td>690</td>
<td>integrated sources, 873</td>
</tr>
<tr>
<td>687</td>
<td>knowledge, definition, 873</td>
</tr>
<tr>
<td>690</td>
<td>nonsubvertible sources, 873</td>
</tr>
<tr>
<td>685–686</td>
<td>overview, 873</td>
</tr>
<tr>
<td>691–695</td>
<td>repositories, 873</td>
</tr>
<tr>
<td>690</td>
<td>sources for, 873</td>
</tr>
<tr>
<td>687–688</td>
<td>strategy for, 873</td>
</tr>
<tr>
<td>689–691</td>
<td>system catalog, 873</td>
</tr>
<tr>
<td>689</td>
<td>technology metadata, 873</td>
</tr>
<tr>
<td>689–691</td>
<td>types of, 873</td>
</tr>
</tbody>
</table>
Microsoft Corporation
  DBMS vendor, 63, 762, 767
tool vendor, 772
Middleware, 192–193
Migration. See Upgrading the DBMS.
Mirror tables, 356
Mirroring, 597
MMDBMSs (main-memory database management systems), 596
Mobile DBMS architectures, 70
Mobile platforms, effect on DBAs, 53–55
ModelRight, 772
mongoDB, 56, 766
Monitoring
  data warehouse performance, 652
  SP_MONITOR procedure, 345–346
  system-level performance, 345–346
Monitoring, database performance
  contention, 23
  database administration tasks, 22–23
  factors affecting, 22–23
  optimization, 23
  resources, 22
  throughput, 22
tools for, 313
workload, 22
Monotonic page splits, 358–359
Moving data. See also EXPORT utility; IMPORT utility; Loading data; Unloading data.
  backup, 535
  bulk movement, 623–625
data warehouses, 644–645
  EXPORT utility, 622–623
  IMPORT utility, 622–623
to multiple databases. See Distributed databases.
tool vendors, 773
MTBF (mean time between failures)
  availability, 273–274
disk drives, 580
Mullins, Craig, Web site, 780
Multi-index access, 387
Multiple platforms, strategies for, 61–62
Multitier implementation, 669–670
MySQL, 64, 765
databases, 93–96
devices, 364
entities, 113
NAS (network-attached storage), 605, 606
Nested triggers, 343–344, 429
Nested-loop joins, 379–380
.NET framework, 194–195, 198
NetIQ, 771
Network data models, 754–755
Network protocol architecture drivers, 673–674
Network traffic
  availability problems, 275
collection pooling, 674
database drivers, 672–674
database gateways, 671–672
performance problems, 670–671
sniffing, 494–495
Newsgroups, 775–776
NHibernate, ORM library, 200
99.999% availability, 273–274
NIST Web site, 782
Nodes, 155
Noncritical applications, 563
Noninvasive auditing methods, 494
Nonmatching index scan, 385
Nonstandard database objects, 94
Nonsubvertible metadata sources, 690
Normal forms
  1NF (first normal form), 129
  2NF (second normal form), 129, 131, 132
  3NF (third normal form), 132–133
  4NF (fourth normal form), 134
  5NF (fifth normal form), 134
  atomic, 129
  BCNF (Boyce Codd normal form), 134
  examples, 130–131
Normal page splits, 358–359
Normalization
  definition, 128
goals of, 128
  Internet databases, 679
  mapping logical to physical, 135
Normalized data model, 133–134
NoSQL
  for availability, 296
data models, 756
  DBMS vendors, 765–766
effect on DBAs, 55–56

N
Naming conventions
  attributes, 116–119
Nouns as
    attributes, 124
    entities, 115, 124
Nullability, specifying, 144
Nulling out, data masking, 498
Nulls
    attributes, 119–120
    check constraints, 423–426
    loading data, 616
Number and date variance, data masking, 497

O
Obfuscation. See Data masking and obfuscation.
Object definitions, backup, 536–537
Object Management Group, 783
Object migration, tools for, 704–705
Object orientation
    data models, 754–755
    DBMS vendors, 766
    ORM (object-relational mapping), 200
    relational databases, 199–200
    SQL, 199–200
    SQL (Structured Query Language), 199–200
Object Store (Progress Software), 766
Object-relational mapping (ORM), 200
Obsessive-Compulsive Data Quality blog, 781
ODBC (Object Database Connectivity)
    callable routines, 192
    definition, 192
    drivers, 192, 673
    overview, 192–193
Off-loading logs, 339, 529
Offset tables, 588–590, 592
Off-site disaster recovery, 547
Off-site locations, disaster planning, 564
OLAP (online analytical processing), 639, 640
The OLAP Council, 783
OLAP tool vendors, 773
OLE DB, 193
100 Year Archive Requirements Survey, 503n
Online database reorganization, 288–289
Online resources. See Internet resources.
Ontos, 766
Open database objects, system performance, 336
The Open Group, 783
Open-source software
    belief system, 764–765
    choosing, 64
DBMS vendors, 764
    definition, 764
    vendors, 764–765
Operating system failure, availability problems, 279
Operating system support, strategies for, 65
Operational support, standards, 102
Operations, data models, 754
Operations control, authorization, 467
Optimizer, 374
Optimizing database performance. See also Database performance; Relational optimization;
    Reorganizing databases.
        cost-based versus rule-based, 344
        database performance, managing, 305
        performance monitoring and tuning, 23
        split tables, 356
Optimizing database performance, techniques for
    block size, 364–365
    clustering, 356–358
    combined tables, 356
    compression, 361–362
    database log placement, 363
    denormalizing tables, 355–356
    derivable data, 356
    disk allocation, 364
    distributed data placement, 363–364
    file placement and allocation, 362–364
    free space, 360–361
    indexing, 352–355
    interleaving data, 360. See also Clustering.
    mapping physical files to database tables, 350–352
    mirror tables, 356
    monotonic page splits, 358–359
    normal page splits, 358–359
    overview, 349–350
    page size, 364–365
    page splitting, 358–359
    parallelism, 351
    partitioning, 350–352
    physical denormalization, 356
    prejoined tables, 355
    raw partition versus file system, 351–352
    redundant data, 356
    repeating groups, 356
    report tables, 355
    speed tables, 356
    split tables, 356
OR logical operator, 403
Oracle Corporation
  DBMS vendor, 63, 762
  nonstandard database objects, 94
  vendor contact, 63, 772
Oracle Corporation (Hyperion), 773
Oracle Magazine, 779
Oracle program
  blogs, 780
  IOUG (Independent Oracle Users Group), 740
  Web site, 778
Oracle program, architecture
  background processes, 326
  CKPT (checkpoint) process, 326
  config.ora file, 325
  control files, 325
  data files, 325
  database files, 325
  databases, 325
  DBWR (database writer) process, 326
  file categories, 325
  instances, 325
  LGWR (log writer) process, 326
  Oracle processes, 326
  overview, 325–327
  parameter files, 325
  PGA (program global area), 326
  physical structures, 325
  PMON (process monitor) process, 326
  processes, 326
  RECO (recover) process, 326
  redo log buffer, 326
  redo log files, 325
  server processes, 326
  SGA (system global area), 326
  SMON (system monitor) process, 326
  sort area, 326
  user processes, 326
Oracle transportable tablespaces, 625
ORDER BY clause, 388
Ordered indexes, 157
Ordering columns, 146
Organization type, effect on DBMS strategy, 65
Organizational design review, 237
ORM (object-relational mapping), 200
"Out of space" log conditions, 339–341
Outer table, 379
Overloading indexes, 355

P
Page header
  corruption, 411
  data page layouts, 588–589
Page pointer, data page layouts, 592
Page size, optimizing database performance, 364–365
Page splits
  optimizing database performance, 358–359
  reorganizing databases, 366
Pages, recovering, 553
Paradox, 767
Parallel access, 390–391
Parallelism
  CPU, 391
  I/O, 390–391
  optimizing database performance, 351
  system, 391
Parameter files, Oracle, 325
Parent tables, 433–435
Pareto (80/20) rule, 302
Parity bits, 597
Parsing database logs, 493–495
Partial recovery, 542–543
Partition scans, 381–382
Partitioned index, 157. See also b-tree index.
Partitioning, 350–352
Passwords. See also Security.
  changing, 453
  creating, 454
  definition, 452
  disabling, 453
  embedding in code, 479
  guidelines for, 454
  limiting, 455
  locking, 453
Patches
  critical updates, 480–481
  for Oracle databases, 480–481
  security alerts, 480–481
PCI (Payment Card Industry) DSS (Data Security Standard), 485, 491
PCTFREE parameter, 360
Performance. See also Applications, performance;
  Database performance; System performance.
  benchmarks, 65–66
  DBA staffing requirements, 38
  gains from upgrading the DBMS, 85
management, tools for, 708–714
monitoring, tools for, 709–710
RAID levels, 603
Performance analysts, DBAs as, 36
Performance problems
client/server computing, 670–674
distributed databases, 632–633
network traffic, 670–671
Permissions. See Privileges.
Personal computing, effect on DBAs, 53–55
Personal DBMS architectures, 70
Personnel, disaster planning, 569
PGA (program global area), 326
Phantoms, 218
Physical data
  dependence, 376
  independence, 757
  modeling, 125–128
Physical denormalization, 356
Physical design review, 236
PII (Personally Identifiable Information), 497
PIT (point-in-time) recovery, 542–543, 545
Planning
  change management, 245
  for disaster. See Disaster planning.
  outages, 286–287
  SQL, 201
  storage capacity, 608–609
Plans (DB2), 94
PLAN_TABLE, 395–398
PMON (process monitor) process, 326
Poet, 766
Pointer consistency, 29
Pointers for very large objects, 410–411
Point-in-time recovery, 27
Ponemon Institute, 450
PostgreSQL, 765
Post-implementation design review, 239
Powersoft, 772
Pre-implementation design review, 239
Prejoined tables, 355
Prepositional phrases, as attributes, 124
Presentation logic, 664–666
Primary key constraints, 416–417
Primary key perspective, 435–436
Primary keys
  database design, 144
  indexes, 151
Print servers, database connectivity, 664
Privacy. See Data, privacy.
Privacy Rights Clearinghouse, 449–450
Privileged users, auditing, 495–496
Privileges. See also Authority; Authorization;
  Security.
  monitoring and reporting, 465
  types of, 457–458
Privileges, granting
  centralized administration, 457
  database object privileges, 459
  DCL (Data Control Language), 456–457
  decentralized administration, 457
  overview, 456–457
  procedure privileges, 460
  program privileges, 460
  to PUBLIC authority, 460–461
  system privileges, 459–460
  table privileges, 458–459
Privileges, revoking
  cascading REVOKEs, 462, 468
  chronology and REVOKEs, 462–463
  overview, 461
Proactive performance, 306
Proactivity, change management, 245
Procedural DBAs
  duties of, 49
  effect on DBAs, 46–50
  managing database logic, 46
  procedural database objects, 48, 49
  role of, 49–50
  stored procedures, 47, 48–50
  triggers, 47, 48–50
  UDFs (user-defined functions), 47, 48–50
Procedure privileges, 460
Procedures (programmatic). See Triggers.
Procedures (standards). See Standards and
  procedures.
Process monitor (PMON) process, 326
Processes, Oracle, 326
Professional advancement, DBA rule of thumb, 747–748
Professional Association for SQL Server, 740, 783
Professional certification. See Certification.
Profiling data, 489
Program global area (PGA), 326
Program privileges, 460
Programming and development, tools for, 724–726
Progress Software, 766
Propagation, 623–624, 722
PUBLIC authority, 460–461
Pure Java drivers, 673–674
Purging data, data warehouse, 655
Purging databases, versus archiving, 501
The Pythian Group, 779, 781

Q
Qualifying rows, 380
Quality assurance testing, 101
Quality of data. See Data, quality; Data integrity.
Queries
   analysis, 378–379
   performance, 650
   tools for, 723–724
   XML data, 203–205
Query rewrite, 392–393
Quest Software, 771
Quiesce point, 528
QUIESCE utility, 528

R
RAC (Real Application Clusters), 294
RAD (rapid application development), 677
RAID (redundant array of independent disks)
   definition, 597
   disk striping, 597
   error correction coding, 599
   fault tolerance, 601–602
   mirroring, 597
   parity bits, 597
   storage type, choosing, 603–604
   striping, 597
RAID levels
   performance, 603
   RAID-0, 597–598
   RAID-0+1, 602
   RAID-1, 598
   RAID-2, 599
   RAID-3, 599
   RAID-4, 600
   RAID-5, 600–601
   RAID-6, 601
   RAID-10, 601
   RAID-50, 602
Raw files, database design, 149
Raw partitions versus file systems, 351–352, 586–587
RDA (Remote Database Access), 629–630
Read efficiency, 333–335
Read locks, 212–213
RECO (recover) process, 326
Recover to current, 26–27
RECOVER utility, 553–554
Recoverability
   availability, 267
   goals of, 509–510
Recover-to-current recovery, 541
Recovery. See also Backup; Disaster planning.
   availability problems, 284
   basic steps, 540–541
   broken blocks or pages, 553
   bulk-logged, 540
   COBIT, 509–510
   common reasons for, 548
   data warehouse, 656–657
   database administration tasks, 26–27
   database logs, 338, 340
   designing for, 533–534
   dropped database objects, 552–553
   duration of, 518, 549
   full, 540
   importance of, 515–516
   indexes, 550–551
   models, 340
   objects, 534
   optimum strategy, 547–549
   options, determining, 538–539
   overview, 537–538
   planning for, 551
   point-in-time recovery, 27
   recover to current, 26–27
   RECOVER utility, 553–554
   regulatory compliance, 508
   RMAN (Recovery Manager), 525–526
   rows, 534
   simple, 540
   SQL Server models, 540
   test databases, populating, 553–554
   testing your plan, 551
   tools for, 714–715
   transaction recovery, 27
   types of, 26–27
   UNLOAD utility, 553–554
Recovery, alternatives to
  disk mirroring, 556–557
  redundant data, 555–556
  replication, 555–556
  snapshot replication, 555–556
  standby databases, 554–555
  symmetric replication, 555–556
Recovery, types of
  matching to failure type, 549
  partial, 542–543
  PIT (point-in-time), 542–543, 545
  recover to current, 541
  to a recovery point, 543–544
  REDO, 545–547
  selecting, 548
  transaction, 544–545
  UNDO, 545–546
Recovery Manager (RMAN), 525–526
Red Gate Software, 771
Redman, Thomas C., 489
Redo log buffer, 326
Redo log files, 325
REDO recovery, 545–547
Redundant array of independent disks (RAID).
  See RAID (redundant array of independent disks).
Redundant data
  backup/recovery alternative, 555–556
  database design, 168–169
  optimizing database performance, 356
Reference customers, 68
Referential constraints, 28, 146–147, 433
Referential integrity (RI). See RI (referential integrity).
Regulatory compliance. See also Data governance; specific regulations.
  best practices, 509
  change management, 261–262
  COBIT, 509–510
  a collaborative approach to, 486–488
  costs of compliance, 485
  costs of non-compliance, 488
  DBA tasks, 26, 487–488
  importance to DBAs, 487–488
  overview, 483–485
  prosecution for non-compliance, 485
  recoverability, 509–510
REINDEX option, 413
Relational Architects, 771
Relational closure, 189–191
Relational data models, 754–755
Relational databases
  application design issues, 373–374
  object orientation, 199–200
Relational nulls, 423–426
Relational optimization
  CPU costs, 376
  database statistics, 376–378
  definition, 375
  density, 377
  design issues, 374
  duplicate values, 377
  I/O costs, 376
  joining tables, 379–381
  optimizer, 374
  physical data dependence, 376
  query analysis, 378–379
  query rewrite, 392–393
  rule-based optimization, 393–394
  view access, 391–392
  view materialization, 392
  view merging, 392
Relational optimization, access path choices
  absolute positioning, 383–385
  avoiding sorts, 387–388
  CPU parallelism, 391
  direct index lookup, 383
  forcing, 398–399
  hashed access, 389–390
  index covering, 386–387
  index screening, 386
  indexed access, 382–389
  index-only access, 386–387
  I/O parallelism, 390–391
  matching index scan, 383–385
  multi-index access, 387
  nonmatching index scan, 385
  parallel access, 390–391
  partition scans, 381–382
  relative positioning, 385
  reviewing, 394–398
  system parallelism, 391
  table scans, 381–382
  tablespace scans, 381–382
Relations, data models, 755
Relationships, 122–123
Relationships between entities
cardinality, 122–123
definition, 122
degree. See Cardinality.
description, 122
discovering, 124–125
optionality, 123
verbs as, 124
Relative positioning, 385
Release schedules, effect on DBMS strategy, 68
Release upgrades, backup, 534
Releases versus versions, 82–87
Reliability, availability, 267
Remote Database Access (RDA), 629–630
Remote mirroring, disaster planning, 573
Remote requests, 630–631
Remote unit of work, 630–631
Removing. See Deleting.
REORG, 288–289
REORG utility, 368–369
Reorganizing databases. See also Database performance; Optimizing database performance.
automation, 371
causes of disorganization, 365–369
cluster ratios, 369
determining the need for, 369
disorganized tablespace, 367–368
ensuring availability, 288–289
file extents, 366
fragmentation, 366
gathering statistics, 370
indexes, 369–370
leaf distance, 370
manually, 368–369
online, 288–289
page splits, 366
row chaining, 366
row migration, 366
tools for, 314
unclustered data, 366
utilities for, 368–369
REPAIR utility (DB2), 411–412
REPEATABLE READ isolation, 217
Repeating groups
denormalization, 169–170
optimizing database performance, 356
Replication
backup/recovery alternative, 555–556
bulk data movement, 623–624
tools, 722
Report tables, 355
Reporting, tools for, 723–724
Repositories, 691–695
Repository Manager (IBM), 695
Repository tools, vendors, 772–773
Required applications, 563
Resources
effective use of, DBA rule of thumb, 745–746
performance factor, 301
performance monitoring and tuning, 22
Response time, 266
Responsive Systems, 771
REST (representation state transfer), 681
Reverse key index, 156–157
Reviewing access path choices, 394–398
REVOKE statements, 456–457, 461
RI (referential integrity)
versus check constraints, 441–442
child tables, 433–434
DBMS support for, 438
definition, 433
deleting rows, 435–436
foreign key perspective, 434–435
foreign key values, 434–436
overview, 146–147, 433–434
parent tables, 433–435
primary key perspective, 435–436
versus program logic, 441–442
referential constraints, 433
relationships, setting up, 436–437
rules, 434–436
rules of thumb, 442–444
self-referencing constraints, 437
system-managed, 441
tools for, 705
with triggers, 438–441
user-managed, 441
Riax, 56
Richard Foote's Oracle Blog, 780
Rigas, John, 485
Rigas, Tony, 485
Rightsizing, and database connectivity, 662
Risk, disaster planning
assessing, 561–563
business service interruption, 561–563
categories of, 561
financial loss, 561–563
legal responsibilities, 561–563
Risk management
tools for, 716–721
upgrading the DBMS, 84–86
RMAN (Recovery Manager), 525–526
Rocket Software, 771
Roles and responsibilities
authorization by, 466, 468
standards, 96–98, 97
Ross E/R method, 112
Row-level triggers, 432
Rows
chaining, 366
data page layouts, 590
deleting, 435–436
headers, data page layouts, 590
length, data page layouts, 592
migration, 366
recovery, 534
size, specifying, 148
Ruby on Rails, 198
Rule-based optimization
versus cost-based, 344
relational optimization, 393–394
Rules. See also specific rules.
check constraints, 424
definition, 94
referential integrity, 434–436
standards, 94
Rumbaugh, James, 113
Rumbaugh E/R method, 112
RUNSTATS utility, 377–378
S
SA (system administration), 20, 21
Salaries of DBAs, 4–6
SAN (storage area network), 278, 604–605, 606
SAP (Business Objects), 773
Sarbanes-Oxley (SOX) Act, 483, 485, 491
SAS Institute, 773
Scalability
data warehouse, 649
effect on DBMS strategy, 66
Scribe, design review, 230
Scripts, change management, 262
SCSI (small computer system interface), 605
SearchDataManagement portal, 781
SearchOracle portal, 781
SearchSQLServer portal, 781
Secured Hash Algorithm (SHA-1), 472
Security. See also Authority; Encryption; Passwords; Privileges.
administrator authorization, 467, 468
auditing, 477–478
authentication, 452
availability problems, 280
basics, 451–455
centralizing, 26
costs of data breaches, 450
data breaches, 449–450
data theft (example), 496
database administration tasks, 24–25
database users, 455–456
external threats, 478–480
fixpacks and maintenance, 480–481
horizontal restriction, 469
job scheduling, 479
logic-oriented, 470
non-DBA, 480
options for system performance, 344
replacement tools for, 721
scope of the problem, 449–450
sensitive data sets, 478–479
standards, 100–101
with stored procedures, 470
tools for, 720–721
user names, 456
using views for, 468–470
vertical restriction, 469
Security, login
administration, rules of thumb, 453, 455
definition, 455–456
limiting, 455
required information, 452–453
Security, passwords
changing, 453
creating, 454
definition, 452
disabling, 453
embedding in code, 479
guidelines for, 454
limiting, 455
locking, 453
SEGUS Inc., 771
SEI Web site, 782
SELECT INTO statements, 341
SELECT INTO/BULKCOPY option, 341
SELECT privileges, 458–459
Selective auditing methods, 494
Self-referencing constraints, 437
Semantic data integrity. See Data integrity, semantic.
Semantic integrity, 28
Serena, 773
SERIALIZABLE isolation, 218
Server processes, 326
Servers. See also Client/server computing; SQL Server (Microsoft).
application, 664
availability problems, 276, 283–284
database, 664
definition, 665–666
file, 664
hardware failure, availability problems, 276
list, 777
performance, 283–284, 650
print, 664
transaction, 207–210
Servers, database
definition, 664
hosting, 675
location, upgrading, 88
Service level agreements (SLAs), 38
Serviceability, availability, 268
Service-level management (SLM), 308–311
Service-oriented architecture (SOA), 680
Servlets, 196–197
Set theory, SQL, 190
Set-at-a-time processing, 189–191
SGA (system global area), 326
SHA-1 (Secured Hash Algorithm), 472
SHA-256 hashing, 472
Shared locks, 212–213
Shared-disk clustering, 72, 294
Shared-nothing clustering, 71–72, 294
Sharing knowledge, DBA rule of thumb, 739–741
SHOWPLAN command, 394–398, 712
Shuffling, data masking, 497
Silos, in a fractured environment, 310–311
Simple Object Access Protocol (SOAP), 680
SIMPLE parameter, 620–621
Simple recovery, 340, 540
Simplification, DBA rule of thumb, 741–742
Size terminology, data storage, 582
Skilling, Jeff, 485
SKIP LOCKED DATA parameter, 219
Skipping
interim releases, 86–87
locked rows, 219
SLAs (service level agreements), 38
SLM (service-level management), 308–311
Small computer system interface (SCSI), 605
SMON (system monitor) process, 326
Snapshot replication, 555–556
SNIA (Storage Networking Industry Association), 503n, 783
Snowflake schema, data warehouse, 643
SOA (service-oriented architecture), 680
SOAP (Simple Object Access Protocol), 680
Software AG, 764
Software environment, designing, 193–194
SoftwareOnZ LLC, 771
Solid state devices (SSDs), 323–324, 596
SolidDB (IBM), 596
Sort area, 326
Sorting
avoiding, 387–388
indexes, 152
SQL tuning, 404
tools for, 314
SOX (Sarbanes-Oxley) Act, 483, 485, 491
Space management. See also Storage management.
monitoring usage, 587–588
tools for, 726
Space management, data page layouts
allocation pages, 589
allocation units, 589
bitmaps, 589
data record layouts, 590
data rows, 588–589
header information, 592
index key values, 592
index page layouts, 592–594
offset and adjust tables, 592
offset table, 588–589
offset tables, 590
overview, 588–589
page header, 588–589
page pointer, 592
row data, 590
row header, 590
row length, 592
sample, 589
space page map, 589
table size, calculating, 591–592
transaction logs, 594–595
Space page map, data page layouts, 589
SP_CONFIGURE procedure, 80n.4
Speed tables, optimizing database performance, 356
Split tables
  horizontally split, 166
  overview, 165–166
  vertically split, 166
Splitting text columns, 166–168
SP_MONITOR procedure, 345–346
SPUFI, 81n.5
SQData, 773
SQL (Structured Query Language)
  access paths, 187
  APIs, 192–193
  benefits of, 188
  binding, 477
  callable routines, 192
  code, design review, 238
  coding for performance, 202–203
  COM, 193
  creating with code generators, 191–192
cursor, 190
definition, 186–187
dynamic, 201
embedded, 201
embedding in programs, 191–192
JDBC (Java Database Connectivity), 192–193
joins, 189
middleware, 192–193
object orientation, 199–200
ODBC (Object Database Connectivity), 192–193
OLE DB, 193
overview, 186–188
planned, 201
query analysis, 378–379, 713
query rewrite, 392–393
querying XML data, 203–205
relational closure, 189–191
set theory, 190
set-at-a-time processing, 189–191
SQL/XML, 204
standalone, 201
standards Web site, 782–783
static, 201
subqueries, 189
syntax, 187
types of, 200–201
unplanned, 201
usage considerations, 188, 202
XQuery language, 204
SQL injection attacks
  examples, 474
  overview, 201–202, 473–475
  preventing, 475–476
  static versus dynamic SQL, 476
SQL Is Your Friend blog, 780
SQL Marklar blog, 780
SQL Rockstar blog, 780
SQL Server (Microsoft)
  filegroups, 149
  nonstandard database objects, 94
  Professional Association for SQL Server, 740
transaction logs, backup, 530
  vendor contact, 63
  Web site, 778
SQL Server Pro, 779
SQL tuning
  basic steps, 399–400
  Cartesian products, 402
code generators, 405
  COMMIT frequency, 404–405
  finding problem statements, 303–304, 406–407
  LIKE logical operator, 403–404
  OR logical operator, 403
  overview, 202–203
  rules of thumb, 400–406
  sorts, 404
  stored procedures, 405–406
tools for, 313
SQL...User Group portal, 782
SQL/XML, 204
SSDs (solid state devices), 323–324, 596
Standalone SQL, 201
Standardizing default values, data warehouse, 647
Standards and procedures
abbreviations, 96
application development, 100
clusters, definition, 94
communications, 98
data administration, 98–99
database links, definition, 94
database naming conventions, 93–96
definition, 93
design review guidelines, 102
distributed data, 629–630
filegroups, definition, 94
importance of, 93
Internet resources, 782
migration and turnover, 101–102
nonstandard database objects, 94
online manuals, 727–728
operational support, 102
plans, definition, 94
roles and responsibilities, 96–98, 97
rules, definition, 94
security, 100–101
storage groups, definition, 94
system administration, 100
Standby databases
availability problems, 276, 277
backup/recovery alternative, 554–555
versus backups, 277
DB2 HADR (high-availability disaster recovery), 285
definition, 276
disaster planning, backup, 573
Oracle, 277
Star schema, data warehouse, 641–643
Statement-level triggers, 432
Static SQL, 201
Storage. See specific media.
Storage area network (SAN), 278, 604–605, 606
Storage groups, 94
Storage management. See also Files and data sets;
Space management; specific media.
capacity planning, 608–609
cool data, 607–608
dormant data, 607–608
fragmentation, 595
goals for, 583
hot data, 607–608
integrity versus availability, 580
multitemperature data, 607–608
overview, 579–583
rate of data growth, 581–582
size terminology, 582
warm data, 607–608
Storage management, media options. See also specific media.
disk, 596
fiber channel, 605
JBOD (just a bunch of disks), 604
MMDBMSs (main-memory database management systems), 596
NAS (network-attached storage), 605, 606
overview, 596
SAN (storage area network), 604–605, 606
SCSI disks, 605
SSDs (solid state devices), 596
tape, 596
tiered storage, 606–608
Storage management, software
backup/recovery alternative, 535–536, 547
disaster planning, backup, 572–573
Storage Networking Industry Association (SNIA), 503n, 783
Storage requirements
database design, 148
installing the DBMS, 76–78
Stored procedures
procedural DBAs, 47, 48–50
as security tools, 470
SQL tuning, 405–406
Strategies for DBMS
benchmarks, TPC, 65–66
choosing a DBMS, 63–68. See also Vendors, DBMS.
cloud database systems, 74
cost of ownership, 67
DBMS architectures, 68–71
DBMS clustering, 71–73
DBMS proliferation, 73
factors affecting, 65–68
hardware issues, 73–74
multiple platforms, 61–62
operating system support, 65
organization type, 65
product complexity, 68
reference customers, 68
release schedules, 68
scalability, 66
technical support, 67
tool availability, 66
Striping, 597
Structural data integrity. See Data integrity, database structure.
Structure, data models, 754
Structured Query Language (SQL). See SQL (Structured Query Language).
Subqueries, SQL, 189
Substitution, data masking, 497
Subsystem failure, backup, 533, 550
Suppliers. See Vendors.
Support policies for old releases, 89
Supporting versus exploiting, 91
Swartz, Mark, 485
Sybase Inc.
  ISUG (International Sybase User Group), 740, 783
  vendor contact, 64, 763
  Web site, 778
Symmetric replication, 555–556
Synonyms, in data modeling, 118
Sysplex (IBM), 294–295
System administration (SA), 20, 21
System administration standards, 100
System administrators
  authorization, 467
  limiting number of, 468
System catalog, 342–343, 689–691
System catalog tables, 345
System DBAs, 31–32
System global area (SGA), 326
System memory failure, 276
System monitor (SMON) process, 326
System monitoring, 345–346
System parallelism, 391
System performance
  allied agents, 321–322
  DBMS components, 324
  disk storage and I/O, 322–324
  hardware configuration, 322–324
  operating system interaction, 320–321
  overview, 319–320
  tools for, 709–710
System performance, DBMS installation and configuration
  cache, 328–330
  configuration types, 327–328
  contention, 341–342
  data cache, 329–330, 332–335
  database log cache, 330
  database logs, 336–341
  deadlock detection, 341
  deadlocks, 342
  defaults, 344
distributed database, 344
guidelines, 344
identity values, 344
Internet structure cache, 330
lock suspensions, 341
locking, 341–342
memory, 328–332
nested trigger calls, 343–344
open database objects, 336
optimization, cost-based versus rule-based, 344
procedure cache, 329–330, 335
read efficiency, 333–335
sample options, 343–344
security options, 344
sort cache, 330
time-outs, 342
System privileges, 459–460
System time, 179–180
System-managed referential integrity, 441
T
Table editors, 707–708
Table scans, 151, 381–382
Tables
  adjust, 592
  combined, 168
  dropping, 250–252
  mirror, 165
  naming conventions, 95–96
  offset, 592
Tables (continued)
  prejoined, 164
  privileges, 458–459
  report, 164–165
  size, calculating, 591–592
  size control, 585
  speed, 172–173
  split, 165–166
  storage requirements, calculating, 590–592
  Sybase segments, 585
Tablespace
  database design, 148
  disorganized, 367–368
  scans, 381–382
Table-to-table synchronization, 498
Tamino (Software AG), 764
Tape storage. See also Storage management.
  DBMS requirements, 77
  disaster planning, backup, 570–571
  storage management option, 596
  WORM (write once, read many) technology, 596
Tapping requests, 494–495
Task-oriented DBAs, 36
TBCHECK utility (Informix), 411
TDES (Triple DES), 472
Team members, disaster planning and recovery, 569
Technical education, DBA rule of thumb, 746–747
Technical support, effect on DBMS strategy, 67
Technology, effects on DBAs
  Big Data movement, 55–56
  cloud computing, 53–55
  database-coupled application logic, 46–50
  Internet and e-business, 50–52
  managing database logic, 46–50
  mobile platforms, 53–55
  NoSQL, 55–56
  personal computing, 53–55
  procedural DBAs, 46–50
Technology metadata, 689
Technology silos, in a fractured environment, 310–311
TechTarget, 781
Temporal data support, 177–180
Temporal database systems, data integrity, 444–446
Temporal requirements, database design, 177–180
Teradata Corporation, 64, 607, 763
Teradata Magazine, 779
Test beds, loading/unloading data for, 621–622
Test databases, populating, 553–554
Testing
  disaster planning, recovery, 567–569, 574
  recovery plans, 551
  tools for, 725
Thin clients, 670
Throughput
  performance factor, 301
  performance monitoring and tuning, 22
Tibbetts, Hollis, 488
Tier-1 DBMS vendors, 63, 762
Tier-2 DBMS vendors, 64, 763
Tiered storage, 606–608
Time outs, locks, 213–214
Time zones, availability across, 270–271
Time-outs, system performance, 342
TimesTen (Oracle), 596
Tool vendors. See Vendors, tools.
Tools, for. See also specific tools.
  capacity planning, 313
  catalog query and analysis, 705–707
  checkpoint/restart, 725
  compliance, 716–721
  debugging, 726
  end-to-end performance, 713–714
  replication, 722
  repositories, 772–773
  trending, 719
  utility management, 716
Tools, for availability. See also Standby databases.
  AlwaysOn features, 285
  Database Definition on Demand, 289–290
  DB2 HADR (high-availability disaster recovery), 285
  effect on DBMS strategy, 66
  RAC (Real Application Clusters), 294
  REORG, 288–289
Tools, for DBAs
  analytics, 721–724
  application performance, 711–713
  auditing tools, 717–719
  availability, effect on DBMS strategy, 66
  backup and recovery, 714–715
  benefits of, 699–700
  business intelligence, 721–724
catalog query and analysis, 705–707
catalog visibility, 706
change management, 254, 701–703
checkpoint/restart, 725
compliance, 716–721
compression, 726–727
cost justification, 702, 731
data integrity, 411–414
data masking, 720
data modeling and design, 700–701
data profiling, 719–720
data protection, 716–721
data warehousing, 721–724
database comparison, 703–704
database performance, 711
database utilities, 715–716
DBA staffing requirements, 39
debugging, 726
database comparison, 703–704
database performance, 711
database utilities, 715–716
DBA staffing requirements, 39
debugging, 726
database comparison, 703–704
database performance, 711
database utilities, 715–716
DBA staffing requirements, 39
debugging, 726
database comparison, 703–704
database performance, 711
database utilities, 715–716
DBA staffing requirements, 39
debugging, 726
end-to-end performance, 713–714
ETL (extract, transfer, load), 721–723
governance, 716–721
homegrown, 732–733
invasive performance tools, 710
native versus third-party, 728
object migration, 704–705
online standards manuals, 727–728
performance management, 708–714
performance monitor, 709–710
programming and development, 724–726
propagation, 722
query, 723–724
reorganizing databases, 368–369
replication, 722
reporting, 723–724
RI (referential integrity), 705
risk management, 716–721
security replacement, 721
security tools, 720–721
space management, 726
system performance, 709–710
table editors, 707–708
testing, 725
trending, 719
types of, 699–700
utility management, 716
vendors, evaluating, 729–732
Tools, vendors for. See also specific vendors.
business intelligence tools, 773
data modeling tools, 771–772
data movement tools, 773
DBA tools, 729–732, 770–771
ETL tools, 773
OLAP tools, 773
repository tools, 772–773
TP (transaction processing) system, 207–209
TPC (Transaction Processing Performance Council), 66–67
Trace-based auditing, 493–495
Tracker tables, 308
Transaction failure, backup, 516–517, 550
Transaction logs. See also Database logs.
backing up, 530
data page layouts, 594–595
file placement, 585
The Transaction Processing Council, 783
Transaction processing monitor, 207–209
Transaction recovery, 27, 544–545
Transaction servers, 207–209
Transaction time, 179–180
Transactions
ACID properties, 205–206
application servers, 209–210
atomicity, 205–206
consistency, 206
definition, 205
durability, 206
example, 206–207
guidelines, 207
isolation, 206
TP (transaction processing) system, 207–209
transaction processing monitor, 207–209
transaction servers, 207–209
UOW (unit of work), 207
Transition tables, 430–431
Transition variables, 430–431
Transparency, distributed databases, 626
Transparent encryption, 473
Transportable tablespaces (Oracle), 625
Treehouse Software, 773
Trending, tools for, 719
Triggers
active databases, 426
definition, 426
DELETE, 438–441
example, 431
firing, 428–429
Triggers (continued)
firing while loading data, 615
granularity, 431–432
implementing referential integrity, 429–430
INSERT, 438–441
INSTEAD OF, 432
multiple on same table, 428–429
nested, 429
overview, 426–428
procedural DBAs, 47, 48–50
referential integrity, 438–441
row level, 432
semantic data integrity, 426–433
statement level, 432
transition tables, 430–431
transition variables, 430–431
UPDATE, 438–441
Triple DES (TDES), 472
TRUNC LOG ON CHKPT option, 340, 530
TRUNCATE TABLE statements, 341
Tuning performance. See Database performance, tuning.
Turnover, standards, 101–102
TUSC, 779
24/24 availability, 270–271
Twitter, as a resource, 741
Two-phase COMMIT, 631
Tyco, 485
Type 1 drivers, 673
Type 2 drivers, 673
Type 3 drivers, 673–674
Type 4 drivers, 673
U
UDFs (user-defined functions), 47, 48–50
UDT (user-defined data types), 418–419
UML (Unified Modeling Language), 113, 114
Unclustered data, reorganizing databases, 366
UNCOMMITTED READ isolation, 216–217
UNDO recovery, 545–546
Unfederated multidatabase schemes, 627
UNION clause, 388
Unique constraints, 28, 417
Unique entity identification, 416–417
Unique identifiers for columns, 145
Unit testing, database environment, 101
UNLOAD utility, 553–554, 618–621, 621–622. See also EXPORT utility.
Unloading data. See also Loading data.
for application test beds, 621–622
backup/recovery alternative, 534–535
concurrency, 619
encoding scheme, specifying, 620
floating-point data, 620
generating LOAD parameters, 620
from image copy backups, 619
LIMIT parameter, 620–621
limiting, 620–621
number of rows, specifying, 620–621
partial unload, 620–621
selection criteria, specifying, 620–621
SIMPLE parameter, 620–621
UNLOAD utility, 618–621, 621–622
from views, 621
WHEN clause, 620–621
Unplanned outages, 286–287
Unplanned SQL, 201
UOW (unit of work), 207
Update locks, 213
UPDATE privileges, 458–459
UPDATE rule, 434–436
UPDATE statements
modifying temporal data, 180
in triggers, 429
UPDATE STATISTICS command, 377–378
UPDATE trigger, 438–441
Upgrading the DBMS. See also Configuring the DBMS; Installing the DBMS.
application complexity, 88
benefits of, 83–84
costs, 84, 85
database server location, 88
DBA staff skill set, 90
DBMS environment complexity, 87–88
deprecated features, 85n.7
fallback planning, 92
features and complexity, 87
migration standards, 101–102
migration verification, 92
organizational style, 89–90
overview, 82–87
performance gains, 85
platform support, 90–91
risks, 84–86
skipping interim releases, 86–87
strategy for, 92
support polices for old releases, 89
supporting software, 91
supporting versus exploiting, 91
vendor reputation, 89
versions versus releases, 82–87
Upsizing, and database connectivity, 662
U.S. Public...Protection Act of 2002, 483
Usenet newsgroups, 775–776
User acceptance testing, database environment, 101
User processes, 326
User-defined data types (UDT), 418–419
User-defined functions (UDFs), 47, 48–50
User-managed referential integrity, 441
Users
  DBA staffing requirements, 38, 39
  group associations, 740
  names, 456
  privileged, auditing, 495–496
  security, 455–456
Utility management, tools for, 716
Utility programs. See Tools, specific programs.

V
V$ tables, 370
Valid time, 179–180
Variable-length columns, 144
Vendors
  evaluating, 729–732
  reputation, importance of, 89
Vendors, DBMS
  Actian Corporation, 764
  Big Three, 762
dBase, 767
FileMaker, 767
IBM Corporation, 762
Informix, 763
Ingres, 763–764
Lotus, 767
main groups, 761–762
Microsoft, 762, 767
NoSQL systems, 765–766
object-oriented systems, 766
Ontos, 766
open-source systems, 764
Oracle Corporation, 762
PC-based systems, 766–767
Poet, 766
Progress Software, 766
Software AG, 764
Sybase Inc., 763
Teradata Corporation, 763
tier 1, 63, 762
tier 2, 64, 763
Web sites for, 778
Vendors, tools. See also specific vendors.
  business intelligence tools, 773
  data modeling tools, 771–772
  data movement tools, 773
  DBA tools, 729–732, 770–771
  ETL tools, 773
  OLAP tools, 773
  repository tools, 772–773
Verifying the DBMS install, 81
Versions versus releases, 82–87
Vertical restriction, 469
Very critical applications, 562
View access, relational optimization, 391–392
View materialization, relational optimization, 392
View merging, relational optimization, 392
Views
  description, 175–176
  as security tools, 468–470
  unloading data from, 621
  uses for, 176–177
Visible Systems, 772
Vision Solutions, 773
Visual Insights, 773
Volatility, DBA staffing requirements, 39
VPD (Virtual Private Database) (Oracle), 471

W
Wallets, 473
WAN (wide-area network), disaster planning, 573
Web Farming site, 783
Web resources. See Internet resources.
Web services. See also Database connectivity; Internet.
  definition, 680
  REST (representational state transfer), 681
  SOA (service-oriented architecture), 680
  SOAP (Simple Object Access Protocol), 680
WHEN clause, 620–621
White Sands Technology, Inc., 771
Wide-area network (WAN), disaster planning, 573
Winter Corporation, 581–582
WITH GRANT OPTION, 457
Workload
  performance factor, 301
  performance monitoring and tuning, 22
WorldCom, 485
Write locks, 212–213
Write-ahead logs, 337

X
XML (eXtensible Markup Language), 204
XML data, querying, 203–205
The XML portal, 783
XQuery language, 204

Y
Yevich, Lawson & Associates, 779

Z
z/OS, data sharing, 295