DATABASE ADMINISTRATION

The Complete Guide to DBA Practices and Procedures

SECOND EDITION

CRAIG S. MULLINS

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

Database Administration

The Complete Guide to DBA Practices and Procedures

Second Edition

Craig S. Mullins

Addison-Wesley

Upper Saddle River, NJ • Boston • Indianapolis • San Francisco
New York • Toronto • Montreal • London • Munich • Paris • Madrid
Capetown • Sydney • Tokyo • Singapore • Mexico City

The publisher offers excellent discounts on this book when ordered in quantity for bulk purchases or special sales, which may include electronic versions and/or custom covers and content particular to your business, training goals, marketing focus, and branding interests. For more information, please contact:

U.S. Corporate and Government Sales
(800) 382-3419
corpsales@pearsontechgroup.com

For sales outside the United States, please contact:

International Sales
international@pearsoned.com

Visit us on the Web: informit.com/aw

Library of Congress Cataloging-in-Publication Data

Database administration : the complete guide to DBA practices and procedures / Craig S. Mullins.—2 [edition].
pages cm
Includes bibliographical references and index.
ISBN 978-0-321-82294-9 (alk. paper)—ISBN 0-321-82294-3 (alk. paper)
1. Database management. I. Title.
QA76.9.D3M838 2013
005.74—dc23
2012029466

ISBN-13: 978-0-321-82294-9
ISBN-10: 0-321-82294-3
Text printed in the United States on recycled paper at Edwards Brothers Malloy in Ann Arbor, Michigan.
First printing October, 2012

Editor-in-Chief
Mark Taub

Acquisitions Editor
Greg Doench

Development Editor
Susan Brown Zahn

Managing Editor
John Fuller

Production Editor
Caroline Senay

Copy Editor
Barbara Wood

Indexer
Richard Evans

Proofreader
Diane Freed

Technical Reviewers
William Arledge
Kevin Kline

Editorial Assistant
Michelle Housley

Cover Designer
Chuti Prasertsith

Compositor
Rob Mauhar
The CIP Group

To my wife, Beth, for her unending love,
constant support, and beautiful smile.

This page intentionally left blank

Contents

Preface

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.

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 of 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.craigsmullins.com. The author can be contacted at craig@craigsmullins.com or in care of the publisher.

2

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 process of 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.

The DBA group should be empowered to make the DBMS decisions for the organization.

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, select a product from a tier-1 vendor.

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 2.1 *Tier-1 DBMS Vendors*

DBMS Vendor	DBMS Product
IBM Corporation New Orchard Road Armonk, NY 10504 Phone: (914) 499-1900	DB2
Oracle Corporation 500 Oracle Parkway Redwood Shores, CA 94065 Phone: (650) 506-7000	Oracle
Microsoft Corporation One Microsoft Way Redmond, WA 98052 Phone: (425) 882-8080	SQL Server

Table 2.2 *Tier-2 DBMS Vendors*

DBMS Vendor	DBMS Product
IBM Corporation New Orchard Road Armonk, NY 10504 Phone: (914) 499-1900	Informix Dynamic Server
Sybase Inc. (an SAP Company) 6475 Christie Avenue Emeryville, CA 94608 Phone: (510) 922-3500	Adaptive Server Enterprise
Teradata Corporation 10000 Innovation Drive Dayton, OH 45342 Phone: (937) 242-4030	Teradata
MySQL (a subsidiary of Oracle Corporation) Phone: (208) 338-8100	MySQL

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]

Choosing any of the lower-tier candidates involves incurring additional risk.

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 are constantly updated to show new and improved performance measurements.

- *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

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.

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

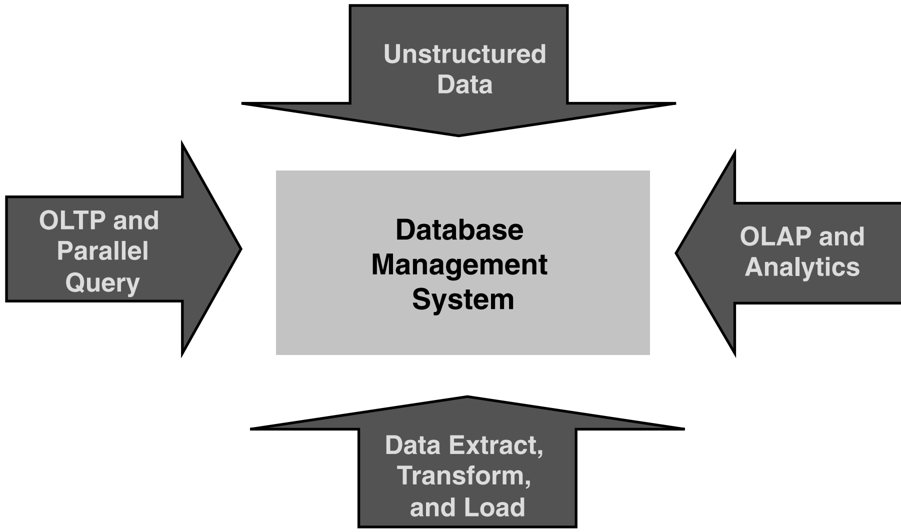

Figure 2.2 *Convergence of features and functionality in DBMS software*

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.

Four levels of DBMS architecture are available: enterprise, departmental, personal, and mobile.

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 FileMaker[2] 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

A modern DBMS offers clustering support to enhance availability and scalability.

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.

The main advantage of shared-nothing clustering is scalability.

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

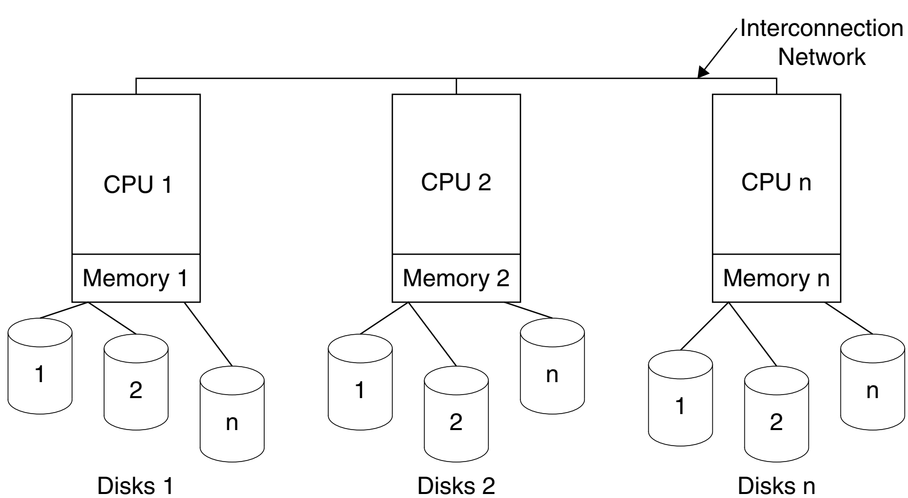

Figure 2.3 *Shared-nothing architecture*

can scale up to thousands of processors because they do not interfere with one another—nothing is shared.

Shared-disk clustering is better suited to large-enterprise processing in a mainframe environment.

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.

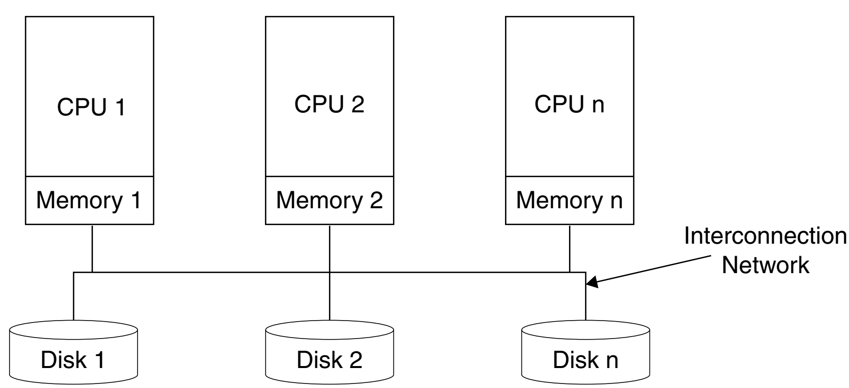

Figure 2.4 *Shared-disk architecture*

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

Shared-Disk	**Shared-Nothing**
Quick adaptability to changing workloads	Can exploit simpler, cheaper hardware
High availability	Almost unlimited scalability
Performs best in a heavy read environment	Works well in a high-volume, read-write environment
Data need not be partitioned	Data is partitioned across the cluster

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

A proliferation of different DBMS products can be difficult to support.

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

Factor hardware platform and operating system constraints into the DBMS selection criteria.

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.

Read the installation guide from cover to cover.

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.

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.

Choose the correct DBMS for your needs and match your hardware to the requirements of the DBMS.

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.

Factor in every storage requirement of the DBMS and reserve the appropriate storage.

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.

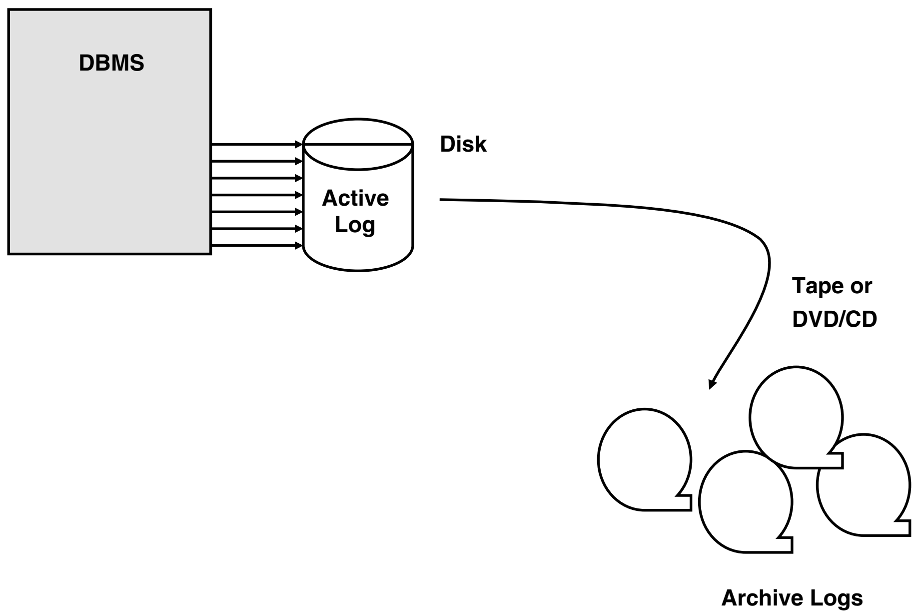

Figure 2.5 *Log off-loading*

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

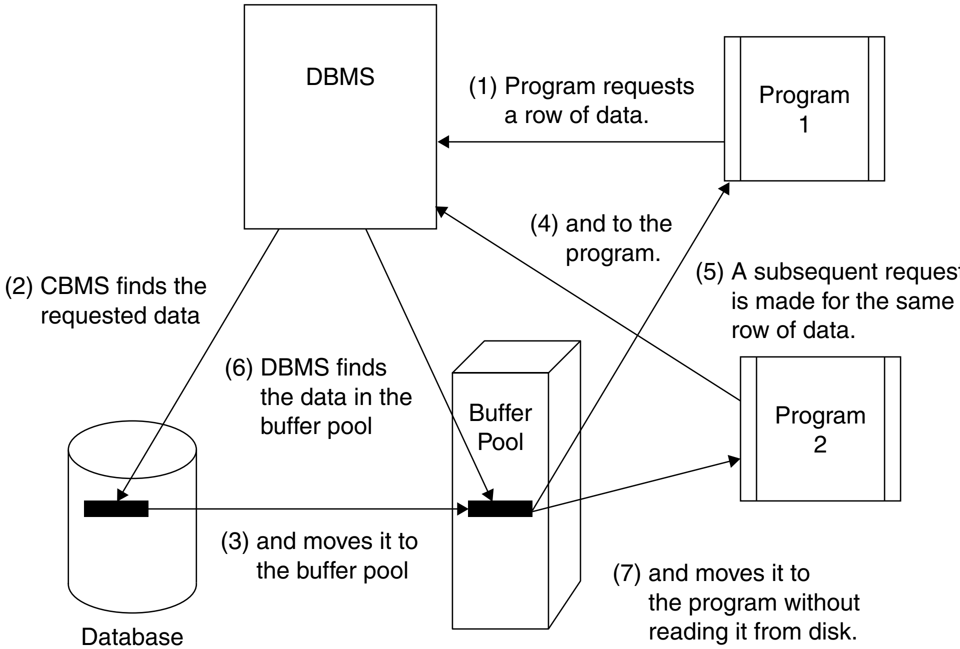

Figure 2.6 *Buffer pool (or data cache)*

to process database requests.[3] 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.

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.

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.

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.

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.

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.

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.

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.

An effective DBMS upgrade strategy must balance the benefits against the risks of upgrading.

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.

A multiple-release upgrade allows customers to effectively control when and how they will migrate to new releases of a DBMS.

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.

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 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

The risk of an upgrade increases as the skills of the DBA staff decrease.

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.

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.

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 to verify that the DBMS release upgrade is satisfactory.

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

Standards and procedures must be developed for database usage.

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.

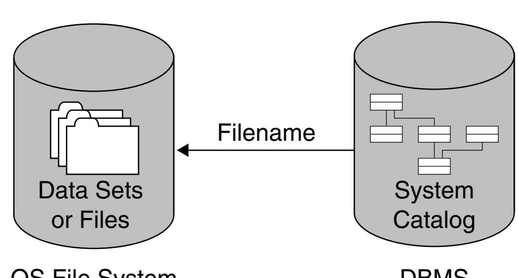

Figure 2.7 *Database objects map to filenames*

Be sure to establish naming conventions for all database objects.

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).

Minimize name changes across environments.

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 *I* or *X* and databases with *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.

Avoid encoding table names to make them shorter.

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

Create a list of 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 nonstandard abbreviations. For example, if "ORG" is the standard abbreviation for "organization," do not allow variants such as "ORGZ" 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*

Task	DBA	DA	SA	Management	Operations	Applications	End Users
DBMS budget	X		X	P		X	X
DBMS installation	P		X		X	X	X
DBMS upgrade	P		X	X	X	X	X
Database usage policy	P	X		X			
Capacity planning	X		P	X	X	X	
Data modeling and analysis	X	P					X
Metadata policy	X	P		X			X
Governance and compliance	X	X	X	X			P
Database design	P	X				X	
Database creation	P						
System performance	X		P				
Database performance	P		X			X	
Application performance	X		X			P	
Backup and recovery	P		X		X	X	
Disaster recovery	P		X		X		
Database security	P		X		X		
Stored procedures	X					P	
Triggers	P					X	
User-defined functions	X					P	
Application design	X					P	
Application turnover	X				X	P	X
Application design reviews	X	X	X	X	X	P	X

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

Include DA standards in the DBA standards as appropriate.

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

The DBA standards serve as a guide to specific approaches to supporting the database environment.

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

Outline necessary standards and procedures for administering database security.

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

Procedures are required for migrating database objects and programs from environment to environment.

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 assures that applications are run according to schedule.

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

Commit to ongoing technical education classes.

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.)

```
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

Blaha, Michael R. *A Manager's Guide to Database Technology.* Upper Saddle River, NJ: Prentice Hall (2001). ISBN 0-13-030418-2.

Connolly, Thomas, and Carolyn Begg. *Database Systems: A Practical Approach to Design, Implementation, and Management.* 4th ed. Harlow, England: Addison-Wesley (2004). ISBN 978-0-321-29401-2.

This page intentionally left blank

Index

NOTES:

Page numbers ending with an italic f (e.g. 192*f*) indicate tables or figures.

Numbers containing a lowercase n (e.g. 301n or 432n.5) indicate a footnote.

Q

R