1. What Is a DBA?

Every organization that manages data using a database management system (DBMS) requires a database administration (DBA) group to ensure the effective use and deployment of the company's databases. And since most modern organizations of every size use at least one DBMS, the need for database administrators (DBAs) is greater today than ever before. However, the discipline of database administration is not well understood or universally practiced in a coherent and easily replicated manner.

> The need for a database administrator is greater today than ever before.

There is a frequently repeated joke about database administration that helps to underscore both the necessity for DBA and lack of understanding of the DBA function. It goes something like this:

> The CIO of Acme Corporation hires a management consulting company to help them streamline their IT operations. The consultant, determined to understand the way Acme works, begins by interviewing the CIO. One of the questions he asks is "So, I see that you have a DBA on staff; what does he do?"
>
> The CIO says "Well, we use Oracle and I'm told that we need the DBA to make sure our Oracle databases stay online. I know that some of our critical business processes like order entry and inventory use Oracle, but I really don't know what the DBA does," says the CIO. "But please don't tell me I need another one because we can barely afford to pay the one we have!"

This is a sad, but too often true, commentary on the state of database administration in many organizations. Frequently the DBA is viewed as a guru or magician who uses tricks to make databases and systems operate efficiently. DBMS software is so complex these days that very few people understand more than just the basics (like SQL). But DBAs understand the complexities of the DBMS, making them a valuable resource. Indeed, sometimes the only source of database management and development

knowledge within the organization is the DBA.

The role of the DBA is as the guardian of the data as a corporate asset. So DBAs, in trying to protect the data, often are perceived as slow moving and risk adverse. Developers, on the other hand, are charged with building new applications and are constantly being challenged to build things fast and move on to the next project. Obviously, the difference between these two roles and expectations—with DBAs saying "change control, change management" and developers saying "deploy it now, deploy it now"—can create friction.

Another frequent criticism of the DBA staff is that they can be difficult to deal with. Sometimes viewed as prima donnas, DBAs can be curmudgeons who have vast technical knowledge but limited people skills. Just about every database programmer has a favorite DBA story. You know, those famous anecdotes that begin with "I have a problem . . ." and end with ". . . and then he told me to stop bothering him and read the manual." DBAs simply do not have a "warm and fuzzy" image. This probably has more to do with the nature and scope of the job than anything else. The DBMS spans the enterprise, effectively placing the DBA on call for the applications of the entire organization.

The fact that DBAs often must sit down and work things through on their own can be a mitigating factor in this poor reputation. Many database problems require periods of quiet reflection and analysis to resolve. So DBAs do not generally like to be disturbed. But even though many problems will require solitude, there are many other problems that require a whole team to resolve. And due to the vast knowledge most DBAs possess, their quiet time is usually less than quiet; constant interruptions to answer questions and solve problems are a daily fact of life.

DBAs should not be encouraged to be antisocial. In fact, DBAs should be trained to acquire exceptional communication skills. Data is the lifeblood of computerized applications. Application programs are developed to read and write data, analyze data, move data, perform calculations using data, modify data, and so on. Without data there would be nothing for the programs to do. The DBA is at the center of the development life cycle—ensuring that application programs have efficient, accurate access to the corporation's data. As such, DBAs frequently interface with many different types of people: technicians, programmers, end users, customers, and executives. However,

many DBAs are so caught up in the minutiae of the inner workings of the DBMS that they never develop the skills required to relate appropriately with their coworkers and customers.

> DBAs need to acquire exceptional communication skills.

But we have not yet answered the question that is the title of this chapter: What is a DBA? The short answer to that question is simple: A DBA is the information technician responsible for ensuring the ongoing operational functionality and efficiency of an organization's databases and the applications that access those databases.

> A DBA is the information technician responsible for ensuring the ongoing operational functionality and efficiency of an organization's databases and applications.

The long answer to that question requires a book to answer—this book. This text will define the management discipline of database administration and provide practical guidelines for the proper implementation of the DBA function. In order to begin to answer the question, though, this chapter will provide an introductory overview of database administration, covering the reasons to pursue a career as a DBA, high-level summaries of DBA tasks, and a look at the organizational structure of the DBA team.

Why Learn Database Administration?

As we have already mentioned, data is at the center of today's applications, and today's modern organization simply cannot operate without data. In many ways, business today *is* data.[1] Without data, businesses would not have the ability to manage finances, to conduct transactions, or to contact their customers. Databases are created to store and organize this data. The better the design and utility of the database, the better the organization will be positioned to compete for business.

Indeed, one of the largest problems faced by IT organizations is ensuring quality database administration. According to a recent study:[2]

- Good DBAs are hard to find and costly to acquire—76 percent of respondents say it takes more than three months to hire a DBA, and training a new DBA in a new environment takes months beyond that.

- The database infrastructure supporting applications is complex, chronically fragmented, and cumbersome to manage.

Both of these findings clearly indicate that database administration is a difficult job that is in high demand. Additionally, according to the Dice[3] 2010–11 Tech Salary Survey, Oracle experience is requested in more than 15,000 job postings on any given day. Demand for Oracle skills is up 57 percent year over year, and the national average salary for technology professionals with experience in Oracle Database is $90,914.

A Unique Vantage Point

The DBA is responsible for designing and maintaining an enterprise's databases, placing the DBA squarely at the center of the business. The DBA has the opportunity to learn about many facets of business and how they interrelate. The DBA can explore groundbreaking technologies as they are adopted by the organization. Exposure to new technology keeps the job stimulating—but frustrating if you are trying to figure out how a new technology works for the first time. The DBA is often working alone in these endeavors; he does not have access to additional expertise to assist when troubles arise. Therefore, a good DBA needs to enjoy challenges and be a good problem solver.

> A good DBA needs to enjoy challenges and be a good problem solver.

DBA Salaries

As a technician you can find no more challenging job in IT than DBA. Fortunately, the job of DBA also is quite rewarding. DBAs are well paid. According to a salary study conducted by Global Knowledge[4] and TechRepublic,[5] the average DBA salary is $78,468, while their managers average $87,261. These average salaries are a little lower than the numbers from the Janco[6] salary survey (conducted the previous year).

For full-time employees functioning as DBAs, the mean salary ranges in the high $80 thousands. Refer to Table 1.1 for a breakdown of the mean salary for DBAs between 2006 and 2011 (according to the Dice 2010–11 Tech Salary Survey).

Table 1.1. Mean Salary for DBAs between 2006 and 2011

Year	Mean DBA Salary
2006–7	$85,441
2007–8	$85,092
2008–9	$89,742
2009–10	$91,283
2010–11	$88,443

Keep in mind that the salary figures quoted here are for illustrative purposes only and will vary based on numerous factors. As might be expected, as the years of experience and number of people managed increases, so does the salary. Of course, DBA salaries, like all salaries, vary by region of the country as well. In the United States, DBA salaries are likely to be higher in the Northeast and on the West Coast than in other regions. Type of industry is also a factor, with pharmaceuticals paying higher than government entities (for example). When all of these factors are considered, the salary figures for DBAs are quite stable and on the higher end of IT individual contributors.

According to the 2012 Computerworld Salary Survey, the national average salary for a DBA is $95,187, but the average salary for a DBA with 15 to 20 years of experience and working on the Pacific coast is $103,597.[7] Obviously experience level and location make a significant difference.

The U.S. Bureau of Labor Statistics (BLS)[8] provides additional information about DBA employment and compensation. As of May 2010, the BLS reports that the median annual wages of database administrators were $73,490 and the mean annual wage was $75,730. The lowest 10 percent earned less than $41,570, and the highest 10 percent earned more than $115,660. The BLS also breaks down earnings statistics by geographical region, so you can use their data to determine expected salary ranges for your specific location of interest.

Perhaps more important than compensation is employability, and your prospects for employment as a DBA are quite good. According to the U.S. Bureau of Labor Statistics *Occupational Outlook Handbook*, 2010–11

Edition, “Employment is expected to grow much faster than the average, and job prospects should be excellent.” Indeed, between the years 2008 and 2018 the BLS estimates that the number of DBA jobs will increase by 20 percent.

> The job market for DBAs will grow much faster than the average.

So, DBAs are well paid, highly employable, possess challenging jobs, and are likely to be engaged in the most visible and important projects. What’s not to like? Well, DBAs are expected to know everything, not just about database technology but about anything remotely connected to it. DBAs almost never work simple eight-hour days, instead working long days with a lot of overtime, especially when performance is suffering or development projects are behind schedule. According to industry analysts, the average DBA works more than 50 hours per week, including an average of six hours on weekends. DBAs frequently have to work on weekends and holidays to maintain databases during off-peak hours.

> Database administration is a nonstop job.

The job of DBA is technically challenging and rewarding, but also potentially one that will exhaust and frustrate you. But don’t let that scare you. The positive aspects of the job far outweigh the negative.

Database Technology

The DBA is the IT professional who understands the specific details of database technology. It is important for the DBA to have a solid understanding of the fundamentals of database management. This requires much more than simple knowledge of rows, columns, and tables, or SQL.

This book assumes the reader has basic knowledge of relational database technology and DBMS fundamentals. For readers needing to review these concepts, please refer to Appendix A, “Database Fundamentals.” This is not a trivial matter. Sometimes the problem is that people think they know more than they actually do. For example, “What is a database?” I bet most people reading this believe they know the answer to that question. But some (perhaps many) of you would be wrong. SQL Server is not a database; it is a DBMS, or database management system. You can use SQL Server to create a database, but SQL Server, in and of itself, is not a database.

So what is a database? A database is an organized store of data wherein the data is accessible by named data elements (for example, fields, records, and files) (see Figure 1.1).

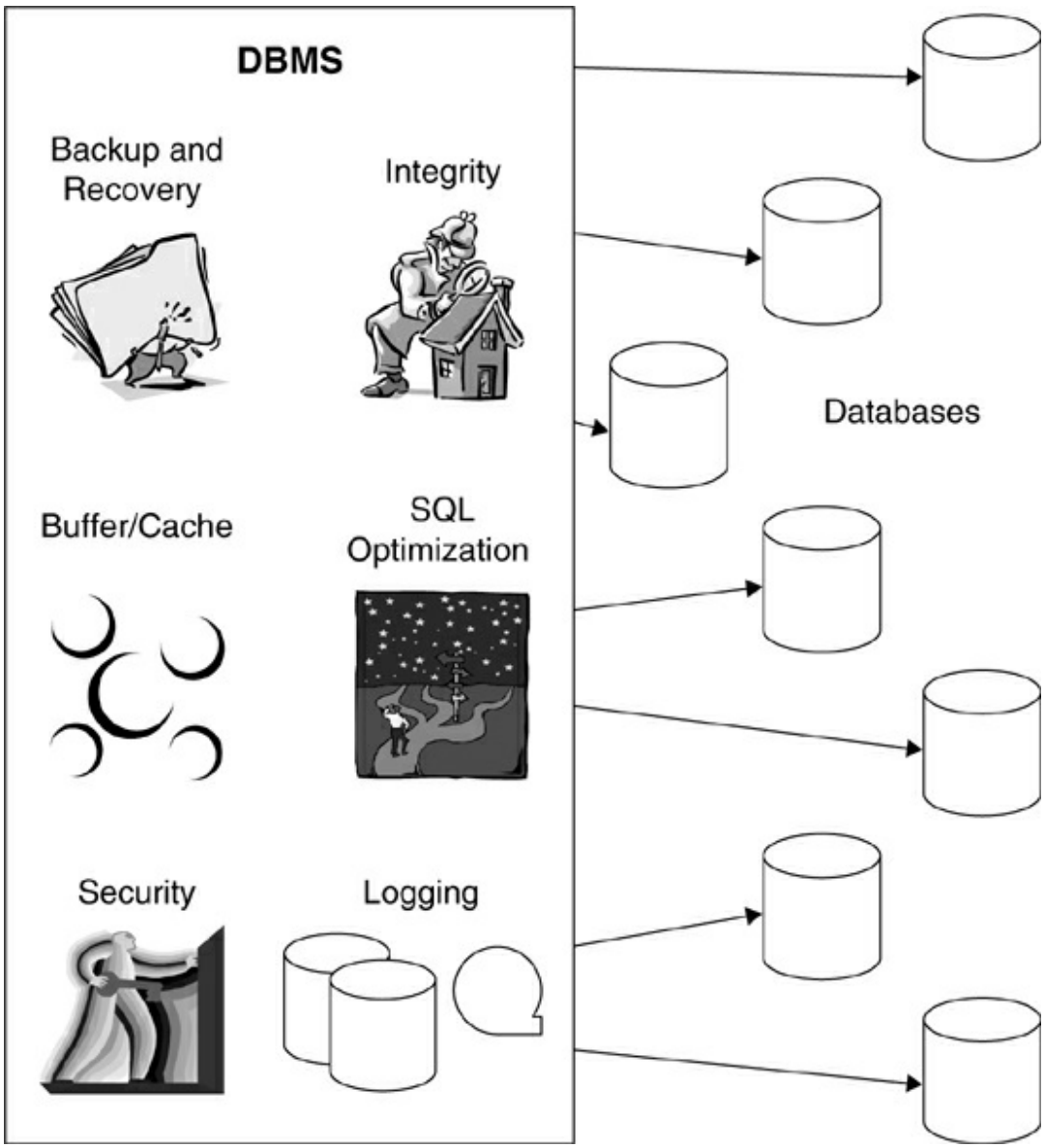

Figure 1.1. DBMS versus database

> A database is an organized store of data wherein the data is accessible by named data elements

A DBMS is software that enables end users or application programmers to share data. It provides a systematic method of creating, updating, retrieving,

and storing information in a database. DBMSs also are generally responsible for data integrity, data security, data access control and optimization, automated rollback, restart, and recovery.

In layperson's terms, you can think of a database as a file folder. You can think of the filing cabinet holding the files along with the file labels as the DBMS. A DBMS manages databases. You implement and access database instances using the capabilities of the DBMS. So, DB2 and Oracle and SQL Server are database management systems. Your payroll application uses the payroll database, which may be implemented using DB2 or Oracle or SQL Server.

Why is that important? If we do not use precise terms in the workplace, confusion can result. And confusion leads to over-budget projects, improperly developed systems, and lost productivity.

In addition to database management fundamentals, DBAs must be experts in the specific DBMS products being used, and there may be many in the organization. For example, a large organization may use DB2 on the mainframe, Oracle and Informix on several different UNIX platforms, MySQL on Linux, and SQL Server on Windows. Older legacy systems may use IMS databases, and then there is that one crazy application out there that uses a fringe DBMS like Adabas or Ingres.[9] And there is also new database technology, such as NoSQL, and column store DBMS offerings like Hadoop, as well as cloud database systems, such as Microsoft's SQL Azure and Google's BigTable.

The DBA group, therefore, must have expertise in each of these different DBMSs and platforms. Furthermore, the DBA must be capable of determining which DBMS and platform are most suited to the needs of each application. This can be a difficult job fraught with politics and conflicting opinions. The DBA group must be able to act as an impartial judge and base implementation decisions on the needs of the application compared to the capabilities and specifications of each DBMS and platform.

> DBAs must implement decisions based on the best fit of application, DBMS, and platform.

Once again, for a short introduction to DBMS concepts, refer to Appendix A.

The Management Discipline of Database Administration

Database administration is rarely approached as a management discipline. The term *discipline* implies planning and implementation according to that plan. When database administration is treated as a management discipline, the treatment of data within your organization will improve. It is the difference between being reactive and proactive.

All too frequently the DBA group is overwhelmed by requests and problems. This ensues for many reasons, including understaffing, overcommitment to supporting new (and even old) application development projects, lack of repeatable processes, lack of budget, and on and on. When operating in this manner, the database administrator is being reactive. The reactive DBA functions more like a firefighter. His attention is focused on resolving the biggest problem being brought to his attention. In other words, a reactive DBA attempts to resolve problems only after problems occur.

A proactive DBA can avoid many problems altogether by developing and implementing a strategic blueprint to follow when deploying databases within his organization. This plan should address all phases of the application development life cycle (ADLC) (see Figure 1.2). The proactive DBA implements practices and procedures to avoid problems before they occur.

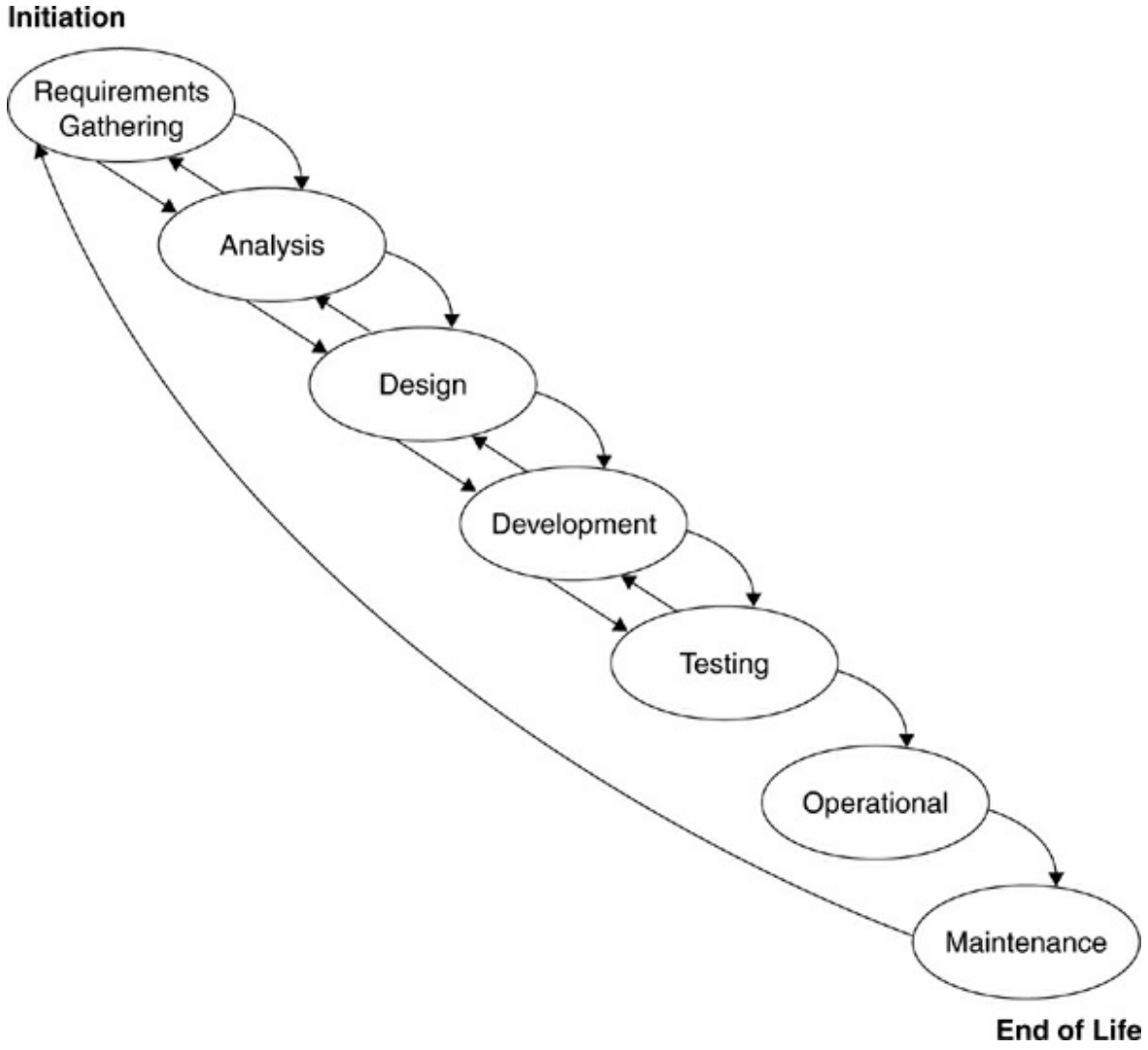

Figure 1.2. The application development life cycle (ADLC)

A proactive DBA can avoid many problems.

A data specialist, usually the DBA, should be involved during each phase of the ADLC. During the initiation and requirements-gathering phase of the project the DBA must be available to identify the data components of the project. He can help to determine if the required data already exists elsewhere in the organization or if the data is brand-new. During the analysis and design phases the rudimentary data requirements must be transformed into a conceptual and logical data model.

Before development can begin, the logical data model must be translated to a physical database design that can be implemented using a DBMS, such as

Oracle or DB2. Sample data must be populated into the physical database to assist with application testing. Furthermore, the DBA must develop and implement a process to refresh test data to enable repeatable test runs.

When the application moves from development to operational status, the DBA must ensure that the DBMS is prepared for the new workload. This preparation includes implementing appropriate security measures, measuring and modifying the storage and memory requirements for the new application, and anticipating the impact of the new workload on existing databases and applications. The DBA is also responsible for migrating the new database from the test environment to production.

While the application is operational, the DBA performs a host of duties, including performance and availability monitoring, tuning, backup and recovery, and managing authorization. But no application or database remains static for long. Because the needs of the business will change, the IT systems that support the business also will change. When maintenance is requested, the DBA becomes engaged in the entire process once again from requirements gathering all the way through to the changes becoming operational. When monitoring reveals a performance shortcoming, the DBA may suggest better-performing alternatives to the development team (if the problem is in the application code) or to the storage team (if the bottleneck is I/O-centric) or recommend other fixes such as new or improved indexes, alternative SQL code, and so forth.

Privacy Policies and Data

Data privacy is a burgeoning issue that is becoming more and more of a burden on organizations, as not only the amount of data under management increases, but the velocity of new data arriving increases.

In this networked world, in which we are thoroughly digitized, with our identities, locations, actions, purchases, associations, movements, and histories stored as so many bits and bytes, we have to ask: Who is collecting all of this? What are they doing with it? With whom are they sharing it? Most of all, individuals are asking, “How can I protect my information from being misused?” These are reasonable questions to ask; we should all want to know the answers.

There are more than 30 federal statutes and over 100 state statutes governing information privacy in the United States. The approach to

protecting privacy has been piecemeal. The European Union, on the other hand, has adopted a Data Protection Directive requiring its member countries to adopt laws that implement its terms. The directive creates rights for persons about whom information is collected, known as "data subjects." Entities that collect information must give data subjects notice explaining who is collecting the data, who will ultimately have access to it, and why the data is being collected. Data subjects also have the right to access and correct data about them.

There is even a Data Privacy Day,[10] the purpose of which is to celebrate the dignity of the individual expressed through personal information.

The recent bankruptcy of Borders Books provides an example of the impact of privacy policies on corporate data. As part of Borders ceasing operations, Barnes & Noble, a competing bookseller, acquired some of its assets, including Borders' brand trademarks and its customer list. Their first course of action, however, was to alert the customers as to their rights. The e-mail I received stated (in part):

> *It's important for you to understand however you have the absolute right to opt-out of having your customer data transferred to Barnes & Noble. If you would like to opt-out, we will ensure all your data we receive from Borders is disposed of in a secure and confidential manner.*

This is one example of how privacy policies can impact the job of database administration and corporate data experts. Of course, you may never work for a company that goes bankrupt, but your company may decide to retire applications and data due to legal regulations, business conditions, or mergers.

Finally, when the application reaches the end of its useful life, the DBA must help to determine the final status of the data used by the application. Is the data no longer required, or do other applications and processes use the data, too? Are there regulations that require the data to be stored longer than the application?[11] Does the business have any stated privacy policies that impose special rules for handling the data?

The DBA is responsible for managing the overall database environment. Often this includes installing the DBMS and setting up the IT infrastructure

to allow applications to access databases. These tasks need to be completed before any application programs can be developed. Furthermore, ad hoc database access is a requirement for many organizations.

Additionally, the DBA is in charge of setting up the ad hoc query environment, which includes query and reporting tool evaluation and implementation, establishing policies and procedures to ensure efficient ad hoc queries, and monitoring and tuning ad hoc SQL.

As you can see, a good DBA is integral to the entire application development life cycle. The DBA is "in demand" for his knowledge of data and the way in which data is managed by modern applications.

> A good DBA is integral to the entire application development life cycle.

A Day in the Life of a DBA

A day in the life of a DBA is usually quite busy. The DBA is required to maintain production and test environments while at the same time keeping an eye on active application development projects, attending strategy and design meetings, helping to select and evaluate new products, and connecting legacy systems to the Web. And Joe in Accounting, he just submitted that "query from hell" again that is bringing the system to a halt; can you do something about that? All of these things can occur within a single DBA workday.

To add to the chaos, DBAs are expected to know everything about everything. From technical and business jargon to the latest management and technology fads, the DBA is expected to be "in the know." And don't expect any private time. A DBA must always be prepared to be interrupted at any time to answer any type of question—and not just about databases, either.

When application problems occur, the database environment is frequently the first thing blamed. The database is "guilty until proven innocent" instead of the other way around. A DBA will never experience an application developer coming to him for help with a question like "I've got some really bad SQL here; can you help me fix it?" No, instead the developer will barge into the DBA's cubicle stating very loudly that "there's a problem with DB2 [or insert your favorite DBMS here]; why don't you fix it so my wonderful program can run?"

> When application problems occur, the database is "guilty until

proven innocent.”

Thus, the DBA is forced to prove that the database is not the source of the problem. He must know enough about all aspects of information technology to track down errors and exonerate that which is in his realm: the DBMS and database structures he has designed. So he must be an expert in database technology but also know about IT components with which the DBMS interacts: application programming languages, operating systems, network protocols and products, transaction processors, every type of computer hardware imaginable, and more. The need to understand such diverse topics makes the DBA a very valuable resource. It also makes the job interesting and challenging.

And the DBA job is a nonstop job. A DBA must be constantly available to deal with problems, because database applications run around the clock. Most DBAs carry a pager or mobile phone at all times so they can be reached at a moment’s notice. If there is a database problem at 2:00 A.M., the DBA must get out of bed, clear his head, and solve the problem to get the applications back up and running. Failure to do so can result in database downtime, and that can completely shut down business processes.

DBAs frequently spend weekends in front of the computer performing database maintenance and reorganizations during off-peak hours. You can’t bring mission-critical databases down during the nine-to-five day to maintain them. And frankly, more and more businesses have 24-hour days instead of the mythical eight-hour business days of yore.

If this still sounds intriguing to you, read on. And actually, it isn’t as bad as it sounds. The work is interesting, there is always something new to learn, and, as previously mentioned, the pay can be good. The only question is, Can anyone do this type of job for 20 or more years without needing a rest? And, oh, by the way, I think I hear your mobile phone buzzing, so you might want to pause here to see what is wrong.

Evaluating a DBA Job Offer

As a DBA, it is almost inevitable that you will change jobs several times during your career. When making a job change, you will obviously consider requirements such as salary, bonus, benefits, frequency of reviews, and amount of vacation time. However, you also should consider how the

company treats its DBAs. Different organizations place different value on the DBA job. It is imperative for your career development that you scout for progressive organizations that understand the complexity and ongoing learning requirements of the position.

Here are some useful questions to ask:

- Does the company offer regular training for its DBAs to learn new DBMS features and functionality? What about training for related technologies such as programming, networking, e-business, transaction management, message queuing, and the like?
- Does the company allow DBAs to regularly attend local user groups?[12] What about annual user groups at remote locations?
- Are there backup DBAs, or will you be the only one on call 24/7?
- Are there data administration and system administration organizations, or are the DBAs expected to perform all of these duties, too?
- Does the DBA group views its relationship with application development groups as a partnership? Or is the relationship more antagonistic?
- Are DBAs included in design reviews, budgeting discussions, and other high-level IT committees and functions?

The more "yes" answers you get to these questions, the more progressive the DBA environment is.

Database, Data, and System Administration

Some organizations define separate roles for the business aspects of data and the technical aspects of data. The business aspects of data are aligned with a discipline known as data administration, whereas the more technical aspects are handled by database administration. Not every organization has a data administration function. Indeed, many organizations combine data administration into the database administration role.

> Many organizations combine data administration into the database administration role.

Sometimes organizations also split up the technical aspects of data management, with the DBA being responsible for using the DBMS and

another role, known as system administration or systems programming, being responsible for installing and upgrading the DBMS.

Data Administration

Data administration (DA) separates the business aspects of data resource management from the technology used to manage data. When the DA function exists in an organization, it is more closely aligned with the actual business users of data. The DA group is responsible for understanding the business lexicon and translating it into a logical data model. Referring back to the application development life cycle, the data administrator (DA) would be involved more in the requirements-gathering, analysis, and design phases; the DBA, in the design, development, testing, and operational phases.

Another differentiating factor between DA and DBA is the focus of their efforts. The DA is responsible for issues such as

- Identifying and cataloging the data required by business users
- Production of conceptual and logical data models to accurately depict the relationship among data elements for business processes
- Production of an enterprise data model that incorporates all of the data used by all of the organization's business processes
- Setting data policies for the organization
- Identifying data owners and stewards
- Setting standards for control and usage of data

In short, the DA can be thought of as the Chief Data Officer of the corporation. The DA position has nothing specifically to do with technology, though. However, in my experience, the DA is never given an executive position. That is actually too bad. Many IT organizations state that they treat data as a corporate asset, but the falsehood of that statement is revealed when you review their actions. Responsibility for data policy is often relegated to technicians who fail to concentrate on the nontechnical, business aspects of data management. Technicians do a good job of ensuring availability, performance, and recoverability but are not usually capable of ensuring data quality and setting corporate policies.

> The data administrator can be thought of as the Chief Data Officer of the corporation.

In fact, data is rarely treated as a true corporate asset. Think about the assets that every company has in common: capital, human resources, facilities, and materials. Each of these assets is modeled: chart of accounts, organization charts and reporting hierarchies, building blueprints and office layouts, and bills of materials. Each is tracked and protected. Professional auditors are employed to ensure that no discrepancies exist in our accounting of the assets. Can we say the same thing about data in most organizations?

A mature DA organization is responsible for planning and guiding the data usage requirements throughout the organization. This role encompasses how data is documented, shared, and implemented. A large responsibility of the DA staff is to ensure that data elements are documented properly, usually in a data dictionary or repository. This is another key differentiation between DA and DBA. The DA focuses on the repository, whereas the DBA focuses on the physical databases and DBMS.

Furthermore, the DA deals with metadata, as opposed to the DBA, who deals with data. Metadata is often described as “data about data”; more accurately, metadata is the description of the data and data interfaces required by the business. Data administration is responsible for the business’s metadata strategy.

Examples of metadata include the definition of a data element, business names for a data element, any abbreviations used for that element, and the data type and length of the element. Data without metadata is difficult to use. For example, the number 12 is data, but what kind of data? In other words, what does that 12 mean? Without metadata we have no idea. Consider:

- Is it a date representing December, the twelfth month of the year?
- Or is it a date representing the twelfth day of some month?
- Could it be an age?
- A shoe size?
- Or, heaven forbid, an IQ?
- And so on.

But there are other, more technical aspects of metadata, too. Think about our number 12 again. Consider:

- Is 12 a large number or a small one?
- What is its domain (that is, what is the universe of possible values of

which 12 is but a single value)?

- What about the data type; is it an integer or a decimal number with a 0 scale?

Metadata provides the context in which data can be understood and therefore become information. In many organizations metadata is not methodically captured and cataloged; instead, it exists mostly in the minds of the business users. Where it has been captured in systems it is spread throughout multiple programs in file definitions, documentation in various states of accuracy, or in long-lost program specifications. Some of it, of course, is in the system catalog of the DBMS.

> Metadata provides the context in which data can be understood and therefore become information.

A comprehensive metadata strategy will enable an organization to understand the information assets under its control and to measure the value of those assets. Additional coverage of metadata is provided in Chapter 22, "Metadata Management."

One of the biggest contributions of DA to the corporate data asset is the creation of data models. A conceptual data model outlines data requirements at a very high level. A logical data model provides in-depth details of data types, lengths, relationships, and cardinality. The DA uses normalization techniques to deliver sound data models that accurately depict the data requirements of the organization.

Many DBAs dismiss data administration as mere data modeling, required only because someone needs to talk to those end users to get the database requirements. But a true DA function is much more than mere data modeling. It is a business-oriented management discipline responsible for the data asset of the organization.

Why spend so much time talking about data administration in a book about database administration? Well, few organizations have implemented and staffed a DA role. The larger the organization, the more likely it is that a DA function exists. However, when the DA role is undefined in the organization, the DBA must assume the mantle of data planner and modeler. The DBA usually will not be able to assume all of the functions and responsibility of DA as summarized in this section for the following reasons.

- The DBA has many other technical duties to perform that will consume most of his time.
- The manager of the DBA group typically does not have an executive position that enables him to dictate policy.
- The DBA generally does not have the skills to communicate effectively with business users and build consensus.
- Frankly, most DBAs are happier dealing with technical issues and technicians than with business issues and non-technicians.

When DA and DBA functions exist within the organization, the two groups must work very closely with one another. It is not necessary that both have the same manager, though that could make it easier to facilitate cooperation between DA and DBA. At any rate, it is imperative that there be some degree of cross-pollination of skills between the two groups. The DA will never understand the physical database like a DBA, and the DBA will never understand the business issues of data like a DA, but each job function would be more effective with some knowledge about the other.

> Organizations truly concerned about data quality, integrity, and reuse will invariably implement and staff the DA function.

In short, organizations that are truly concerned about data quality, integrity, and reuse will invariably implement and staff the data administration function.

Database Administration

Database administration is the focus of this entire book, so I will not spend a lot of time defining it in this short section. The rest of the book will accomplish that nicely. This section will quickly outline the functions performed by the DBA group when the DA function exists. As illustrated in Figure 1.3, at a high level, the DBA manages data and the DA manages metadata. But let’s dig a little deeper.

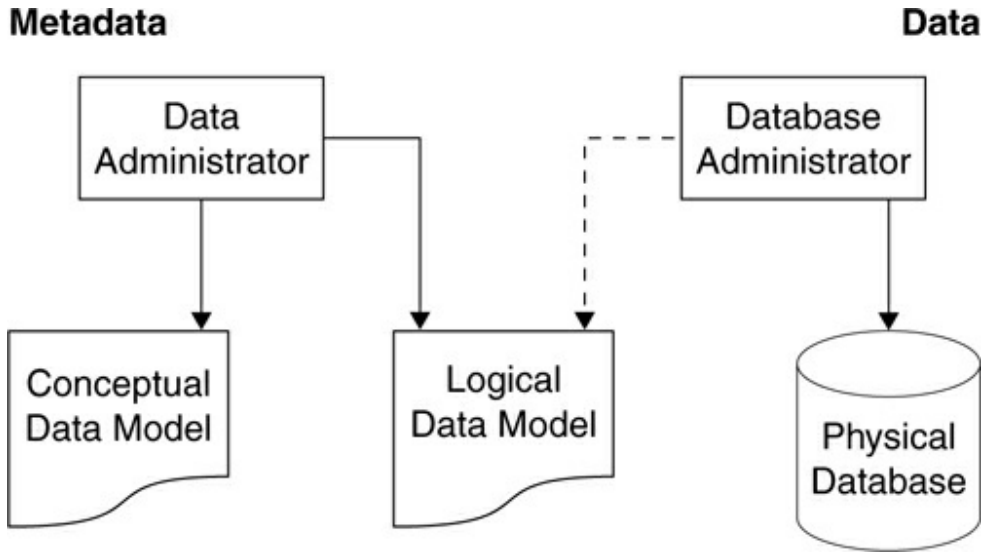

Figure 1.3. DBA versus DA

The first DBA duty is to understand the data models built by DA and to be able to communicate the model to the application developers and other appropriate technicians. The logical data model is the map the DBA will use to create physical databases. The DBA will transform the logical data model into an efficient physical database design. It is essential that the DBA incorporate his knowledge of the DBMS being used to create an efficient and appropriate physical database design from the logical model. The DBA should not rely on the DA for the final physical model any more than the DA should rely on the DBA for the conceptual and logical data models.

The DBA is the conduit for communication between the DA team and the technicians and application programming staff. Of course, the bulk of the DBA's job is the ongoing support of the databases created from the physical design and the management of the applications that access those databases. An overview of these duties is provided in the upcoming "DBA Tasks" section of this chapter.

> The DBA is the conduit for communication between the DA team and the technicians and application programming staff.

System Administration

Some organizations, once again the larger ones, also have a system administration (SA) or systems programming role that impacts DBMS implementation and operations. When the SA role exists separately from the

DBA role, it is responsible for the installation and setup of the DBMS. The system administrator (SA) typically has no responsibility for database design and support. Instead, the DBA is responsible for the databases and the SA is responsible for DBMS installation, modification, and support. If this distinction is not clear to you, please review Appendix A for a clear definition of database terminology.

Furthermore, the SA role ensures that the IT infrastructure is implemented such that the DBMS is configured to work with other enabling system software. The SA may need to work with other technicians to configure transaction processors, message queuing software, networking protocols, and operating system parameters to enable the DBMS to operate effectively. The SA ensures that the IT infrastructure is operational for database development by setting up the DBMS appropriately, applying ongoing maintenance from the DBMS vendor, and coordinating migration to new DBMS releases and versions.

> The system administrator ensures that the IT infrastructure is operational for database development by setting up the DBMS appropriately, applying ongoing maintenance from the DBMS vendor, and coordinating migration to new DBMS releases and versions.

As with DA, there must be cross-training of skills between the SA and DBA, too. The SA will never understand the physical database like a DBA, but the DBA is unlikely to understand the installation and in-depth technical relationships of system software like the SA. Each job function will be more effective with some knowledge of the other, though.

When an independent SA group does not exist, or when no DBMS-focused SA exists, the DBA assumes responsibility for system administration and programming. The graphic in Figure 1.4 provides a quick delineation of the DA, DBA, and SA tasks.

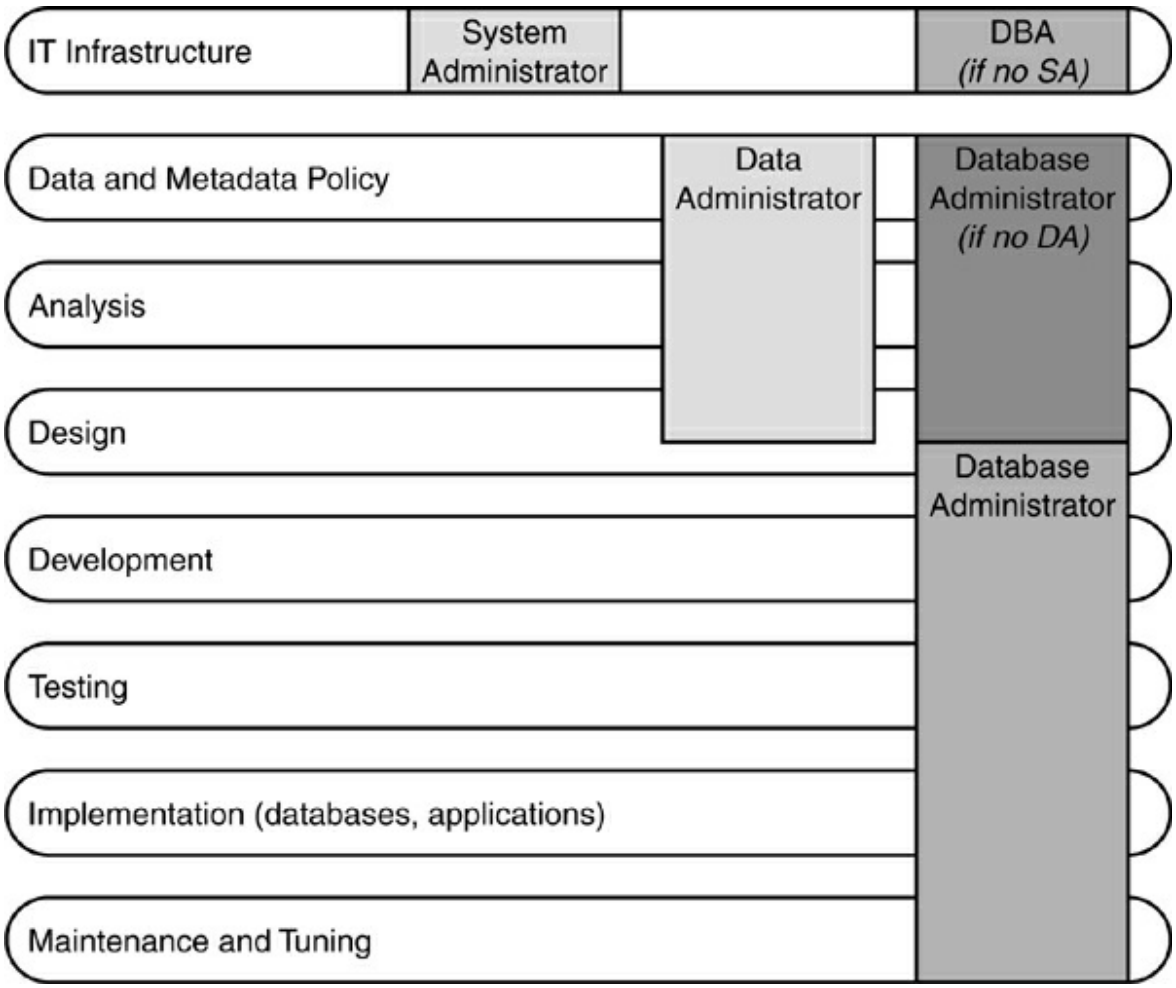

Figure 1.4. DA, DBA, and SA responsibilities

DBA Tasks

A DBA must be capable of performing many tasks to ensure that the organization's data and databases are useful, usable, available, and correct. These tasks include design, performance monitoring and tuning, ensuring availability, authorizing security, backup and recovery, ensuring data integrity, and, really, anything that interfaces with the company's databases. Let's examine each of these topics.

Database Design

The first task that most people think of when they think about DBAs is the ability to create well-designed databases. To properly design and create relational databases DBAs must understand and adhere to sound relational

design practices. They must understand both relational theory and the specific implementation of the RDBMS being used to create the database. Database design requires a sound understanding of conceptual and logical data modeling techniques. The ability to create and interpret entity-relationship diagrams is essential to designing a relational database.

Furthermore, the DBA must be able to transform a logical data model into a physical database implementation. The DBA must ensure that the database design and implementation will enable a useful database for the applications and clients that will use it.

> The DBA must be able to transform a logical data model into a physical database.

Indeed, database design is a significant skill for the DBA to possess. However, the job of the DBA often is disproportionately associated with database design. Although designing optimal databases is important, it is a relatively small portion of the DBA's job. A DBA will most likely spend more time administering and tuning databases than in originally designing and building databases.

By no means, though, should you interpret this to mean that database design is not important. A poor relational design can result in poor performance, a database that does not meet the needs of the organization, and potentially inaccurate data.

Performance Monitoring and Tuning

The second role most closely associated with the DBA is performance monitoring and tuning. But what is meant by the term *database performance*? Think, for a moment, of database performance using the familiar concepts of supply and demand. Users demand information from the database. The DBMS supplies information to those requesting it. The rate at which the DBMS supplies the demand for information can be termed *database performance*. But it is not really that simple. Five factors influence database performance: workload, throughput, resources, optimization, and contention.

The *workload* that is requested of the DBMS defines the demand. It is a combination of online transactions, batch jobs, ad hoc queries, data warehousing and analytical queries, and commands directed through the system at any given time. Workload can fluctuate drastically from day to day,

hour to hour, minute to minute, and even second to second. Sometimes workload can be predicted (such as heavy month-end processing of payroll, or very light access after 7:30 p.m., when most users have left for the day), but at other times it is unpredictable. The overall workload has a major impact on database performance.

Throughput defines the overall capability of the computer hardware and software to process. It is a composite of I/O speed, CPU speed, parallel capabilities of the machine, and the efficiency of the operating system and system software. The hardware and software tools at the disposal of the system are known as the *resources* of the system. Examples include the database kernel, disk space, cache controllers, and microcode.

The fourth defining element of database performance is *optimization*. All types of systems can be optimized, but relational queries are unique in that optimization is primarily accomplished internal to the DBMS. However, there are many other factors that need to be optimized (SQL formulation, database parameters, programming efficiently, and so on) to enable the database optimizer to create the most efficient access paths.

When the demand (workload) for a particular resource is high, *contention* can result. Contention is the condition in which two or more components of the workload are attempting to use a single resource in a conflicting way (for example, dual updates to the same piece of data). As contention increases, throughput decreases.

Therefore, database performance can be defined as the optimization of resource usage to increase throughput and minimize contention, enabling the largest possible workload to be processed.

Whenever performance problems are encountered by an application that uses a database, the DBA is usually the first one called to resolve the problem. Of course, the DBA cannot manage database performance in a vacuum. Applications regularly communicate with other applications, systems, and components of the IT infrastructure. An effective performance monitoring and tuning strategy requires not just DBMS expertise but knowledge outside the scope of database administration. Many performance management tasks must be shared between the DBA and other technicians. In other words, handling performance problems is truly an enterprise-wide endeavor.

> Database performance can be defined as the optimization of resource usage to increase throughput and minimize contention, enabling the largest possible workload to be processed.

The DBA must be vigilant in monitoring system, database, and application performance. As much as possible this should be accomplished using automated software and scripts. Polling system tables and building alerts based on thresholds can be used to proactively identify problems. Alerts can be set up to e-mail the DBA when performance metrics are not within accepted boundaries.

Many tasks and abilities are required of DBAs to ensure efficient access to databases. Some of these abilities include building appropriate indexes, specifying large enough buffers and caches, aligning the database implementation with the IT infrastructure, ongoing monitoring of databases and applications, database reorganization, and adapting to business changes —more users, more data, additional processing, and changing requirements and regulations.

Ensuring Availability

Availability of data and databases is often closely aligned with performance, but it is actually a separate concern. Of course, if the DBMS is offline, performance will be horrible because no data can be accessed. But ensuring database availability is a multifaceted process.

> Ensuring database availability is a multi-faceted process.

The first component of availability is keeping the DBMS up and running. Vigilant monitoring and automated alerts can be used to warn of DBMS outages and call for corrective action.

Individual databases also must be maintained such that the data contained therein is available whenever applications and clients require it. Doing so requires the DBA to design the database so that it can be maintained with minimal disruptions, but also to help design applications to minimize conflicts when concurrent access is required.

An additional component of availability is minimizing the amount of downtime required to perform administrative tasks. The faster the DBA can

perform administrative tasks that require databases to be offline, the more available the data becomes. Increasingly, DBMS vendors and independent software vendors (ISVs) are providing nondisruptive utilities that can be performed on databases while applications read and write from them. But these usually require more skill and up-front planning to implement.

The DBA must understand all of these aspects of availability and ensure that each application is receiving the correct level of availability for its needs.

Database Security and Authorization

Once the database is designed and implemented, programmers and users will need to access and modify the data in the database. But only authorized programmers and users should have access to prevent security breaches and improper data modification. It is the responsibility of the DBA to ensure that data is available only to authorized users.

Typically, though not always (see the sidebar "Centralization of Security"), the DBA works with the internal security features of the DBMS in the form of SQL GRANT and REVOKE statements, as well as any group authorization features of the DBMS. Security must be administered for many actions required by the database environment:

- Creating database objects, including databases, tables, views, and program structures
- Altering the structure of database objects
- Accessing the system catalog
- Reading and modifying data in tables
- Creating and accessing user-defined functions and data types
- Running stored procedures
- Starting and stopping databases and associated database objects
- Setting and modifying DBMS parameters and specifications
- Running database utilities such as LOAD, RECOVER, and REORG

Database security can be enforced in other ways as well. For example, views can be created to block sensitive columns or rows from being viewed by end users and programmers. And the DBA also frequently interfaces with external security methods when they impact database security.

The DBA must understand and be capable of implementing any aspect of

security that impacts access to databases. One area that should be of particular interest given the data breaches in the news these days is SQL injection attacks and how to prevent them.

> The DBA must understand the aspects of security that impact access to databases.

Centralization of Security

Some organizations have taken steps to assemble all security and authorization tasks, policies, and procedures in a centralized IT security group. Such an undertaking is not trivial and, as such, it is more common in larger organizations than it is in smaller ones.

Heavily regulated industries may opt to centralize security operations. Furthermore, organizations with mainframe computers tend to consider centralization more often than those without mainframes.

Successfully transferring database security responsibility from the database administration group to a centralized security group requires training the security personnel in database security techniques. Additionally, many of these shops use software that removes security operations from the DBMS and mimics the same functionality in a more traditional security package (such as RACF or ACF2 on the mainframe).

Even taking these issues into consideration, most organizations still rely on the DBA to administer database security.

Governance and Regulatory Compliance

Assuring compliance with industry and governmental regulations is an additional task required of database administration, at least in terms of implementing proper controls. The DBA must work with management, auditors, and business experts to understand the regulations that apply to their industry and the manner in which data is treated.

Certain aspects of regulatory compliance address standard DBA operating procedures. For example, regulations may contain language enforcing specific security and authorization procedures, auditing requirements, data backup specifications, and change management procedures. To ensure compliance, however, stricter documentation may be required, or perhaps a

higher degree of diligence or automation (such as deeper audit tracing).

Other aspects of regulatory compliance may require the DBA to adopt different techniques, tactics, and skills. For example, data retention regulations may require data to be maintained long after it is needed to be stored in a production database, requiring database archiving skills. Or certain data may need to be protected from view, requiring data masking or, in some cases, encryption to be set up.

DBAs should not be tasked with understanding regulations in any depth, nor should they be setting the standards by which the organization complies with the regulations. However, DBAs will get involved in helping to set the proper controls and procedures for compliance projects, specifically and especially with regard to the treatment of data.

> DBAs set the proper technological controls and procedures for compliance with regard to the treatment of data.

Backup and Recovery

The DBA must be prepared to recover data in the event of a problem. "Problem" can mean anything from a system glitch or program error to a natural disaster that shuts down an organization. The majority of recoveries today occur as a result of application software error and human error. Hardware failures are not as prevalent as they used to be. In fact, analyst estimates indicate that 80 percent of application errors are due to software failures and human error. The DBA must be prepared to recover data to a usable point, no matter what the cause, and to do so as quickly as possible.

> The majority of recoveries today occur as a result of application software error and human error.

The first type of data recovery that usually comes to mind is a *recover to current*, usually in the face of a major shutdown. The end result of the recovery is that the database is brought back to its current state at the time of the failure. Applications are completely unavailable until the recovery is complete.

Another type of traditional recovery is a *point-in-time recovery*. Point-in-time recovery usually is performed to deal with an application-level problem. Conventional techniques to perform a point-in-time recovery will remove the

effects of *all* transactions since a specified point in time. This sometimes can cause problems if there were some valid transactions during that time frame that still need to be applied.

Transaction recovery is a third type of recovery that addresses the shortcomings of the traditional types of recovery: downtime and loss of good data. Thus, transaction recovery is an application recovery whereby the effects of specific transactions during a specified time frame are removed from the database. Therefore, transaction recovery is sometimes referred to as application recovery.

Most technicians think about recovery in order to resolve disasters such as hardware failures. Although hardware failures still occur, and technicians need to be prepared to recover from such failures, most recoveries today are required because of human error or errors in programs.

The DBA must be prepared to deal with all of these types of recovery. This involves developing a backup strategy to ensure that data is not lost in the event of an error in software, hardware, or a manual process. The strategy must be applicable to database processing, so it must include image copies of database files as well as a backup/recovery plan for database logs. It needs to account for any non-database file activity that can impact database applications as well.

> The DBA must be prepared to deal with all types of recovery.

Ensuring Data Integrity

A database must be designed to store the correct data in the correct way without that data becoming damaged or corrupted. To ensure this process, the DBA implements integrity rules using features of the DBMS. Three aspects of integrity are relevant to our discussion of databases: physical, semantic, and internal.

> Three aspects of integrity are relevant to our discussion of databases: physical, semantic, and internal.

Physical issues can be handled using DBMS features such as domains and data types. The DBA chooses the appropriate data type for each column of each table. This action ensures that only data of that type is stored in the database. That is, the DBMS enforces the integrity of the data with respect to

its type. A column defined as “integer” can contain only integers. Attempts to store nonnumeric or noninteger values in a column defined as integer will fail. DBAs can also use constraints to further delineate the type of data that can be stored in database columns. Most relational DBMS products provide the following types of constraints:

- *Referential constraints* are used to specify the columns that define any relationships between tables. Referential constraints are used to implement referential integrity, which ensures that all intended references from data in one column (or set of columns) of a table are valid with respect to data in another column of the same or a different table.
- *Unique constraints* ensure that the values for a column or a set of columns occur only once in a table.
- *Check constraints* are used to place more complex integrity rules on a column or set of columns in a table. Check constraints are typically defined using SQL and can be used to define the data values that are permissible for a column or set of columns.

Semantic integrity is more difficult to control and less easily defined. DBAs must be prepared to enforce policies and practices to ensure that data stored in their databases is accurate, appropriate, and usable. An example of a semantic issue is the quality of the data in the database. Simply storing any data that meets the physical integrity definitions specified to the database is not enough. Procedures and practices need to be in place to ensure data quality. For example, a customer database that contains a wrong address or phone number for 25 percent of the customers stored therein is an example of a database with poor quality. There is no systematic, physical method of ensuring data accuracy. Data quality is encouraged through proper application code, sound business practices, and specific data policies. Redundancy is another semantic issue. If data elements are stored redundantly throughout the database, the DBA should document this fact and work to ensure that procedures are in place to keep redundant data synchronized and accurate.

The final aspect of integrity is an internal DBMS issue. The DBMS relies upon internal structures and code to maintain links, pointers, and identifiers. In most cases the DBMS will do a good job of maintaining these structures, but the DBA needs to be aware of their existence and how to cope when the

DBMS fails. Internal DBMS integrity is essential in the following areas:

- *Index consistency.* An index is really nothing but an ordered list of pointers to data in database tables. If for some reason the index gets out of sync with the data, indexed access can fail to return the proper data. The DBA has tools at his disposal to check for and remedy these types of errors.
- *Pointer consistency.* Sometimes large multimedia objects are not stored in the same physical files as other data. Therefore, the DBMS requires pointer structures to keep the multimedia data synchronized to the base table data. Once again, these pointers may get out of sync if proper administration procedures are not followed.
- *Backup consistency.* Some DBMS products occasionally take improper backup copies that effectively cannot be used for recovery. It is essential to identify these scenarios and take corrective actions.

Overall, ensuring integrity is an essential DBA skill.

Ensuring integrity is an essential DBA skill.

DBMS Release Migration

The DBA is also responsible for managing the migration from release to release of the DBMS. DBMS products change quite frequently—new versions are usually released every year or so. The task of keeping the DBMS running and up-to-date is an ongoing effort that will consume many DBA cycles. Whatever approach is taken must conform to the needs of the organization, while reducing outages and minimizing the need to change applications.

The task of keeping the DBMS running and up-to-date is an ongoing effort that will consume many DBA cycles.

Jack-of-All-Trades

Databases are at the center of modern applications. If the DBMS fails, applications fail, and if applications fail, the entire business can come to a halt. If databases and applications fail often enough, the entire business can fail. Database administration therefore is critical to the ongoing success of modern business.

Furthermore, databases interact with almost every component of the IT infrastructure. The IT infrastructure of today consists of things such as

- Programming languages and environments such as COBOL, Microsoft Visual Studio, C/C++/C#, Java, and PHP
- Software frameworks such as .NET and J2EE
- Database and process design tools such as ERwin and Rational Rose
- Transaction processing systems such as CICS and Tuxedo
- Application servers such as WebSphere, JBoss, Oracle Application Server, and EAServer
- Message queuing software such as MQSeries and MSMQ
- Networking software and protocols such as SNA, VTAM, and TCP/IP
- Networking hardware such as bridges, routers, hubs, and cabling
- Multiple operating systems such as Windows, z/OS and MVS, UNIX and Linux, and perhaps others
- Data storage hardware and software such as enterprise storage servers, Microsoft SMS, IBM DFHSM, SANs, and NAS
- Operating system security packages such as RACF, ACF2, and Kerberos
- Other types of storage hardware, for example, tape machines, silos, and solid-state (memory-based) storage
- Non-DBMS data set and file storage techniques such as VSAM and b-tree
- NoSQL products such as Hadoop and MongoDB
- Database administration tools and how they interface with other systems management solutions
- Systems management tools and frameworks such as HP OpenView and CA Unicenter
- Operational control software such as batch scheduling software and job entry subsystems
- Software distribution solutions for implementing new versions of system software across the network
- The Internet and Web-enabled databases and applications

- Client/server development techniques (multitier, fat server/thin client, thin server/fat client, etc.)
- Object-oriented and component-based development technologies and techniques such as CORBA, COM, OLE/DB, ADO, and EJB
- Pervasive computing technology devices such as tablets and smartphones

Although it is impossible to become expert in all of these technologies, the DBA should have some knowledge of each of these areas and how they interrelate. Even more importantly, the DBA should have the phone numbers of experts to contact in case any of the associated software and hardware causes database access or performance problems.

The Types of DBAs

There are DBAs who focus on logical design and DBAs who focus on physical design; DBAs who specialize in building systems and DBAs who specialize in maintaining and tuning systems; specialty DBAs and general-purpose DBAs. Truly, the job of DBA encompasses many roles.

Some organizations choose to split DBA responsibilities into separate jobs. Of course, this occurs most frequently in larger organizations, because smaller organizations often cannot afford the luxury of having multiple, specialty DBAs.

Still other companies simply hire DBAs to perform all of the tasks required to design, create, document, tune, and maintain the organization's data, databases, and database management systems. Let's look at some of the more common types of DBA.

System DBA

A *system DBA* focuses on technical rather than business issues, primarily in the system administration area. Typical tasks center on the physical installation and performance of the DBMS software, including

A *system DBA* focuses on technical rather than business issues.

- Installing new DBMS versions and applying maintenance fixes supplied by the DBMS vendor

- Setting and tuning system parameters
- Tuning the operating system, network, and transaction processors to work with the DBMS
- Ensuring appropriate storage for the DBMS
- Enabling the DBMS to work with storage devices and storage management software
- Interfacing with any other technologies required by database applications
- Installing DBA tools and utilities

System DBAs rarely get involved with actual implementation of databases and applications. They may get involved in application tuning efforts when operating system parameters or complex DBMS parameters need to be altered.

Indeed, the job of system DBA usually exists only if the organization does not have an official system administration or systems programming department.

Database Architect

Some organizations designate a separate job, referred to as a *database architect*, for designing and implementing new databases. Database architects are involved in new design and development work only; they do not get involved in maintenance, administration, and tuning efforts for established databases and applications. The database architect designs new databases for new applications or perhaps a new database for an existing application.

> The database architect is involved in new design and development work only.

The rationale for creating a separate position is that the skills required for designing new databases are different from the skills required to keep an existing database implementation up and running. A database architect is more likely to have data administration and modeling expertise than a general-purpose DBA, because DA skills are more useful for developing an initial database design.

Typical tasks performed by the database architect include

- Creation of a logical data model (if no DA or data modeler position

exists)

- Translation of logical data models into physical database designs
- Implementing efficient databases, including physical characteristics, index design, and mapping database objects to physical storage devices
- Analysis of data access and modification requirements to ensure efficient SQL and to ensure that the database design is optimal
- Creation of backup and recovery strategies for new databases

Most organizations do not staff a separate database architect position, instead requiring DBAs to work on both new and established database projects.

Database Analyst

Another common staff position is the *database analyst*. There is really no common definition for database analyst. Sometimes junior DBAs are referred to as database analysts. Sometimes a database analyst performs a role similar to the database architect. Sometimes the DA is referred to as the database analyst, or perhaps the data analyst. And sometimes *database analyst* is just another term used by some organizations instead of *database administrator*.

Data Modeler

When the DA role is not defined or staffed, there may be a data modeler role defined. A *data modeler* is usually responsible for a subset of the DA's responsibilities. Data modeling tasks include

- The collection of data requirements for development projects
- Analysis of the data requirements
- Design of project-based conceptual and logical data models
- Creation of a corporate data model and keeping the corporate data model up-to-date
- Working with the DBAs to ensure they have a sound understanding of the data models

Application DBA

In direct contrast to the system DBA is the *application DBA*. Application DBAs focus on database design and the ongoing support and administration of databases for a specific application or applications. The application DBA

is likely to be an expert at writing and debugging complex SQL and understands the best ways to incorporate database requests into application programs. The application DBA also must be capable of performing database change management, performance tuning, and most of the other roles of the DBA. The difference is the focus of the application DBA—not on the overall DBMS implementation and database environment, but on a specific subset of applications (see Figure 1.5).

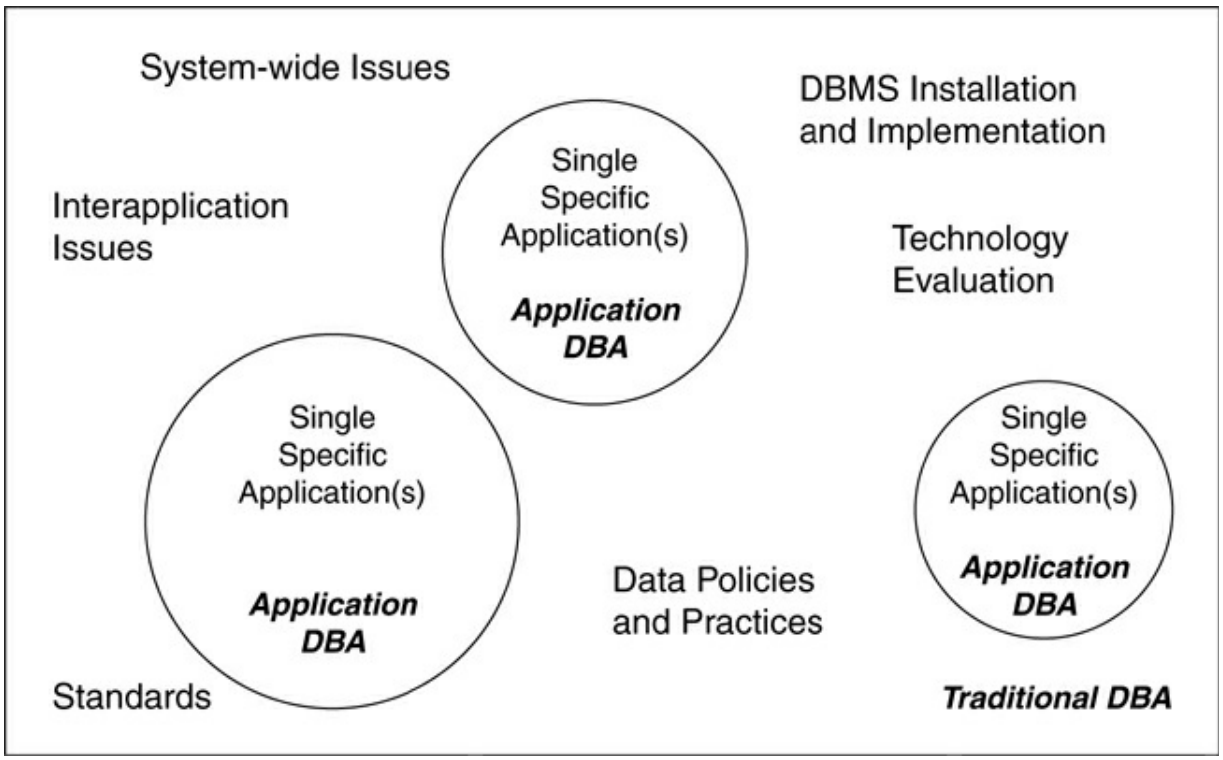

Figure 1.5. Focus of the application DBA

> The application DBA focuses on database design and the ongoing support and administration of databases for a specific application or applications.

Not every organization staffs application DBAs. However, when application DBAs exist, general-purpose DBAs are still required to support the overall database environment and infrastructure. When application DBAs do not exist within an organization, general-purpose DBAs are likely to be assigned to support specific applications while also maintaining the organization's database environment.

There are pros and cons to staffing application DBAs. The arguments in

favor of application DBAs include the following:

- Application DBAs can better focus on an individual application, which can result in better service to the developers of that application.
- The application DBA is more often viewed as an integral component of the development team and therefore is better informed about new development plans and changes to plans.
- Because application DBAs consistently work on a specific set of applications, they can acquire a better overall understanding of how each application works, enabling them to better support the needs of the application developers.
- With a more comprehensive understanding of the application, an application DBA will have a better understanding of how the application impacts the overall business. This knowledge will likely result in the execution of DBA tasks to better support the organization.

But all is not favorable for application DBAs. There are cons to implementing an application DBA role, including these:

- Application DBAs can lose sight of the overall data needs of the organization because of their narrow focus on a single application.
- The application DBA can become isolated. Lack of communication with a centralized DBA group (if one exists) can result in diminished sharing of skills.
- When application DBAs implement useful procedures, it takes more effort to share these procedures among the other DBAs.
- Due to the application-centric nature of application DBAs, they can lose sight of new features and functionality being delivered by the DBMS group.

In general, when staffing application DBAs, be sure to also staff a centralized DBA group. The application DBAs should have primary responsibility for specific applications but should also be viewed as part of the centralized DBA group. Doing so will encourage the pros of implementing application DBAs while discouraging the cons.

> When staffing application DBAs, be sure to also staff a centralized DBA group.

Task-Oriented DBA

Larger organizations sometimes create very specialized DBAs who focus on a single specific DBA task. But task-oriented DBAs are quite rare outside of very large IT shops. One example of a task-oriented DBA is a backup and recovery DBA who devotes his entire day to ensuring the recoverability of the organization's databases.

Most organizations cannot afford this level of specialization, but when possible, task-oriented DBAs can ensure that very important DBA tasks are tackled by very knowledgeable specialists.

Performance Analyst

Performance analysts are a specific type of task-oriented DBA. The performance analyst, more common than other task-oriented DBAs, focuses solely on the performance of database applications.

A performance analyst must understand the details and nuances of SQL coding for performance, as well as have the ability to design databases for performance. Performance analysts have knowledge of the DBMS being used at a very detailed technical level so that they can make appropriate changes to DBMS and system parameters when required.

But the performance analyst should not be a system DBA. The performance analyst must be able to speak to application developers in their language in order to help them facilitate the appropriate program changes for performance.

The performance analyst is usually the most skilled, senior member of the DBA staff. It is very likely that a senior DBA grows into this role due to his experience and the respect he has gained in past tuning endeavors.

> The performance analyst is usually the most skilled, senior member of the DBA staff.

Data Warehouse Administrator

Organizations that implement data warehouses for performing in-depth data analysis often staff DBAs specifically to monitor and support the data warehouse environment. *Data warehouse administrators* must be capable DBAs, but with a thorough understanding of the differences between a database that supports online transaction processing (OLTP) and a data

warehouse. Common data warehouse administration tasks and requirements include

- Experience with business intelligence, data analytics, and query and reporting tools
- Database design for read-only access
- Data warehousing design issues such as star schema
- Data warehousing technologies such as online analytical processing, or OLAP (including ROLAP, MOLAP, and HOLAP)
- Data transformation and conversion skills
- An understanding of data quality issues
- Experience with data formats for loading and unloading of data
- Middleware implementation and administration

Staffing Considerations

Staffing the DBA organization is not a simple matter. There are several nontrivial considerations that must be addressed, including the size of the DBA staff and the reporting structure for the DBAs.

How Many DBAs?

One the most difficult things to determine is the optimal number of DBAs required to keep an organization's databases online and operating efficiently. Many organizations try to operate with the minimal number of DBAs on staff, the idea being that having fewer staff members lowers cost. But that assumption may not be true. An overworked DBA staff may make mistakes that cause downtime and operational problems far in excess of the cost of the salary of an additional DBA.

But determining the optimal number of DBAs is not a precise science. It depends on many factors, including

- *Number of databases.* The more databases that need to be supported, the more complex the job of database administration becomes. Each database needs to be designed, implemented, monitored for availability and performance, backed up, and administered. There is a limit to the number of databases that an individual DBA can control.
- *Number of users.* As additional users are brought online as clients of

the applications that access databases, it becomes more difficult to ensure optimal database performance. Additionally, as the number of users increases, the potential for increase in the volume of problems and calls increases, further complicating the DBA's job.

- *Number of applications.* A single database can be used by numerous applications. Indeed, one of the primary benefits of the DBMS is to enable data to be shared across an organization. As more applications are brought online, additional pressure is exerted on the database in terms of performance, availability, and resources required, and more DBAs may be required to support the same number of databases.
- *Service-level agreements (SLAs).* The more restrictive the SLA, the more difficult it becomes for the DBA to deliver the service. For example, an SLA requiring subsecond response time for transactions is more difficult to support than an SLA requiring 3-second response time.
- *Availability requirements.* When databases have an allowable period of scheduled downtime, database administration becomes easier because some DBA tasks either require an outage or are easier when an outage can be taken. Considerations such as supporting e-business transactions and the Web drive the need for 24/7 database availability.
- *Impact of downtime.* The greater the financial impact of a database being unavailable, the more difficult DBA becomes because pressure will be applied to assure greater database availability.
- *Performance requirements.* As the requirements for database access become more performance oriented and faster and more frequent access is dictated, DBA becomes more complicated.
- *Type of applications.* Organizations implement all kinds of applications. The types of applications that must be supported have an impact on the need for DBA services. The DBMS and database needs of a mission-critical application differ from those of a non-mission-critical application. Mission-critical applications are more likely to require constant monitoring and more vigilance to ensure availability. Likewise, OLTP application will have different characteristics and administration requirements from OLAP applications. OLTP transactions are likely to be of shorter duration than OLAP queries; OLTP applications perform both read and write operations, whereas

OLAP applications are usually read only. Each has administration challenges that impose different DBA procedures and needs.

- *Volatility.* The frequency of database change requests is an important factor in the need for additional DBAs. A static database environment requiring few changes will not require the same level of DBA effort as a volatile, frequently changing database environment. Unfortunately, the level of volatility for most databases and applications tends to change dramatically over time. It is difficult to ascertain how volatile an overall database environment will be over its lifetime.
- *DBA staff experience.* The skill of the existing DBA staff will impact whether or not additional DBAs are required. A highly skilled DBA staff will be able to accomplish more than a novice team. Skills more than experience dictate DBA staffing level requirements. A highly motivated DBA with two years of experience might easily outperform a ten-year veteran who is burned out and unmotivated.
- *Programming staff experience.* The less skilled application developers are in database and SQL programming, the more involved DBAs will need to be in the development process, performing tasks such as complex SQL composition, analysis, debugging, tuning, and ensuring connectivity. As the experience of the programming staff increases, the complexity of DBA decreases.
- *End user experience.* When end users access databases directly with ad hoc SQL, their skill level has a direct impact on the complexity of DBA.
- *DBA tools.* DBMS vendors and a number of ISVs offer tools that automate DBA tasks and make administering databases easier. The more tools that are available and the degree to which they are integrated can make DBA tasks less complex. Industry analysts have estimated that without DBA tools up to twice the number of DBAs can be required.

Determining how many DBAs are needed is not a precise science.

The previous list of complexity issues notwithstanding, it is very difficult to combine all of these factors into a formula that will dictate the optimum number of DBAs to employ. Though the research is somewhat dated,

industry analysts at the META Group[13] created a loose formula for calculating DBA level of effort (LOE). The formula is not a rigorous one but arrives at a DBA LOE by applying weights to six factors: system complexity, application immaturity, end user sophistication, software functionality, system availability, and staff sophistication. By measuring each of these items as much as possible to indicate high or low rates, you plug values into the formula and arrive at a number that is translated into an estimate for the number of DBAs required.

> Creating a formula that can dictate the optimal number of DBAs to employ is difficult to achieve.

DBA Reporting Structures

To whom should the DBA group report within the organization? Different companies have taken different approaches to the DBA reporting structure, but a few reporting hierarchies are quite common. There is no one correct answer, but some reporting structures work better than others. Let's review some of the possibilities.

One of the best structures is to create a data resource management (DRM) group that consists of all the data and information specialists of the organization—DA, DBA, data analysts, performance analysts, and so on. This group usually reports directly to the CIO but might report through a systems programming unit, the data center, or technical support. Figure 1.6 depicts a typical reporting structure.

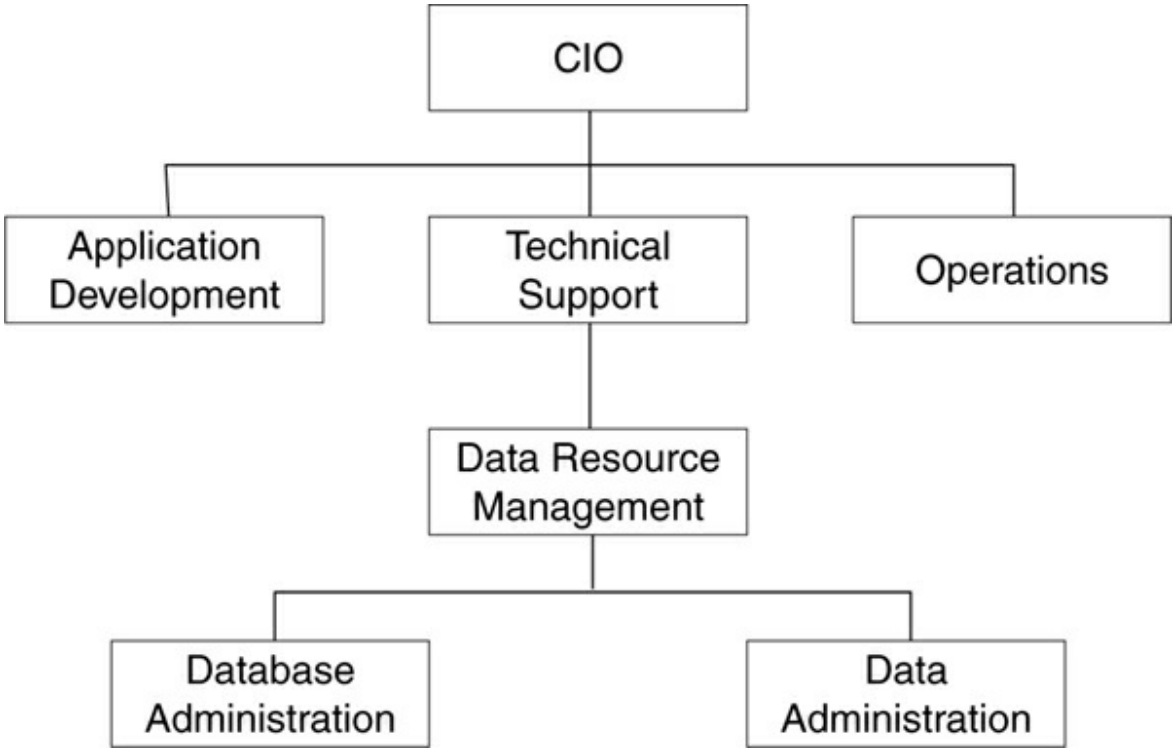

Figure 1.6. Typical DBA reporting structure

When an organization staffs application DBAs, they will be spread out in application groups, typically with a direct reporting line to the business programming managers. Each application development team has a dedicated application DBA resource, as shown in Figure 1.7.

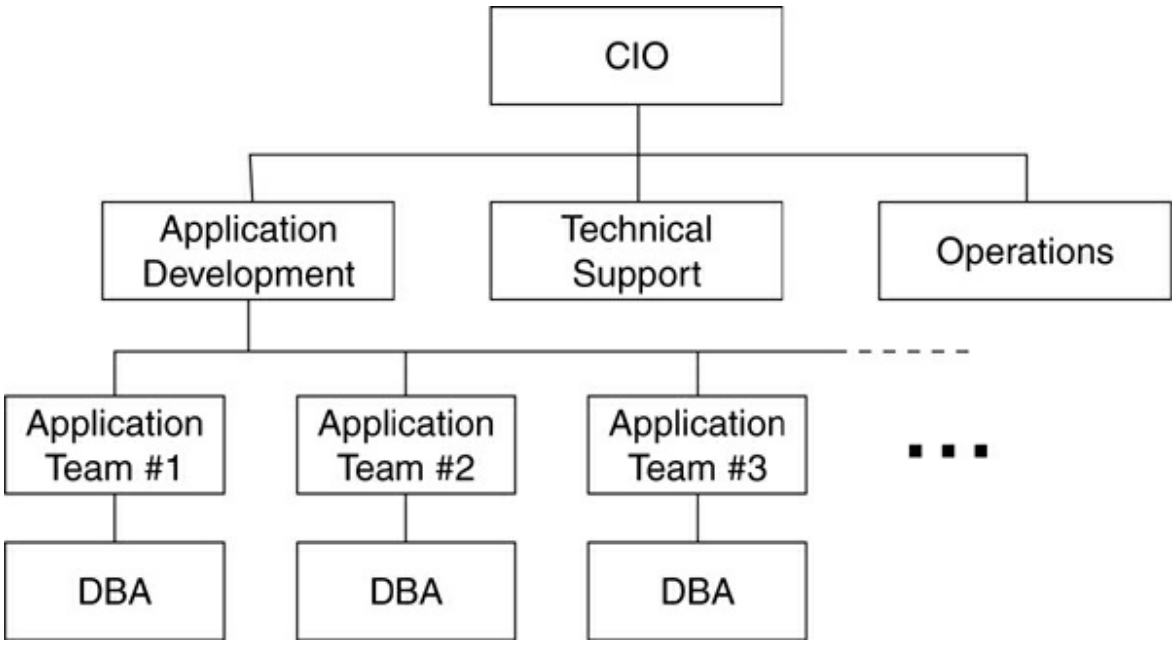

Figure 1.7. Application DBA reporting structure

There are problems with both of these reporting structures, though. The first problem is that DRM should be placed higher in the IT reporting hierarchy. It is a good idea to have the DRM group report directly to the CIO. When an organization understands the importance of data to the health of the organization, placing DRM at this level is encouraged.

Furthermore, when application DBAs exist, they should not report to the application programming manager only. A secondary line of reporting to the DRM group will ensure that DBA skills are shared and communicated throughout the organization. Figure 1.8 delineates the recommended reporting structure for the data resource management group.

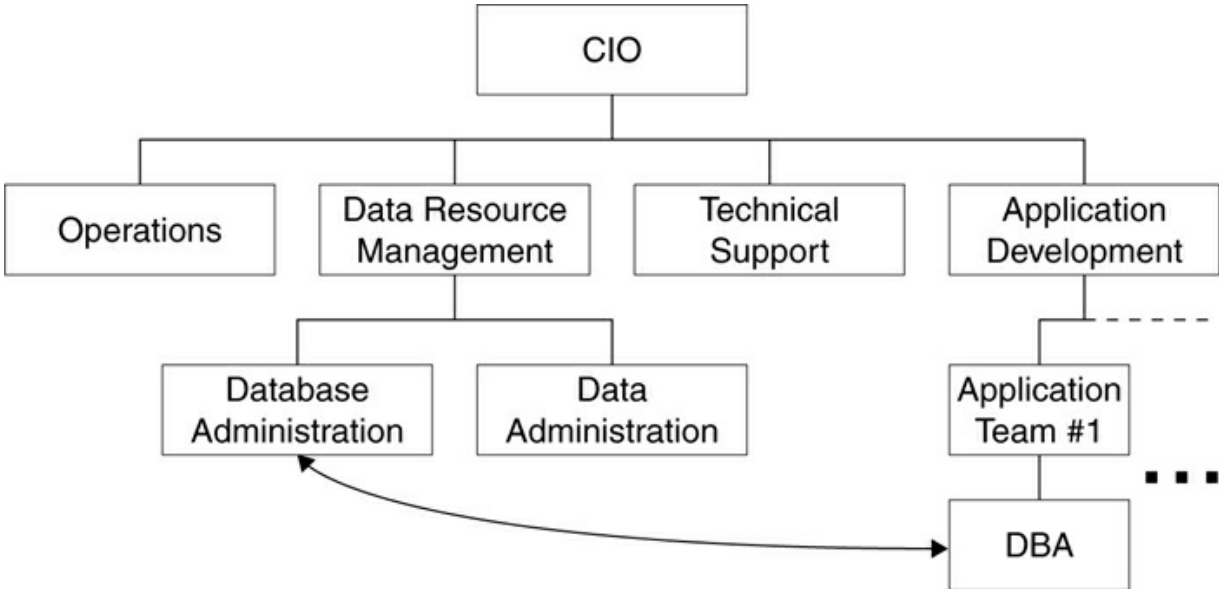

Figure 1.8. Recommended DBA reporting structure

Multiplatform DBA Issues

Managing a multiplatform environment complicates the job of database administration. A whole batch of different problems and issues arise that need to be addressed. The first task is to define the scope of each DBA's job. Does a single DBA administer all of the different DBMSs, or does each DBA focus on supporting only one DBMS?

> Managing a multiplatform environment complicates the job of database administration.

This is a particularly thorny issue. On the one hand, the functionality of a DBMS is strikingly similar regardless of platform and vendor. A DBMS is designed to store, retrieve, and protect data. Programmers, programs, and end users all interact with the DBMS to access and modify data. Administration issues are similar—design, creation, optimization, and so on—though each DBMS implements these items differently. So, the case can be made that a DBA should support multiple DBMSs and databases, regardless of platform or vendor.

On the other hand, each DBMS offers different features, functionality, and technology. Keeping all of the differences and nuances straight is a monumental task. Wouldn't it be better to develop platform-expert DBAs? That way, your Oracle DBAs can focus on learning all there is to know about Oracle, your DB2 DBAs can focus on DB2, and so on.

Every organization will have to make this determination based on its particular mix of DBMSs, features, and DBA talent. If your organization uses one DBMS predominantly, with limited use of others, it may make sense for each DBA to support all of them, regardless of platform or vendor. Sparse usage of a DBMS usually means fewer problems and potentially less usage of its more sophisticated features. By tasking your DBAs to be multi-DBMS and multiplatform, you can ensure that the most skilled DBAs in your shop are available for all database administration issues. If your organization uses many different DBMSs, it is probably wise to create specialist DBAs for the heavily used platforms and perhaps share administration duties for the less frequently used platforms among other DBAs.

When DBA duties are shared, be sure to carefully document the skills and knowledge level of each DBA for each DBMS being supported. Take care to set up an effective and fair on-call rotation that does not unduly burden any particular DBA or group of DBAs. Furthermore, use the organizational structure to promote sharing of database standards and procedures across all supported DBMS environments.

Keep in mind, too, that when multiple DBMSs and platforms are supported, you should consider implementing DBA tools, performance monitors, and scripts that can address multiple platforms. For this reason, DBA tools from third-party vendors are usually better for heterogeneous environments than similar tools offered by the DBMS vendors.

When your organization supports multiple DBMSs, the DBA group should

develop guidelines for which DBMS should be used in which situations. These guidelines should not be hard-and-fast rules but instead should provide guidance for the types of applications and databases best supported by each DBMS. Forcing applications into a given DBMS environment is not a good practice. The guidelines should be used simply to assure best fit of application to DBMS. These guidelines should take into account

- Features of each DBMS
- Features and characteristics of the operating system
- Networking capabilities of the DBMS and operating system combination
- DBMS skills of the application developers
- Programming language support
- Any other organizational issues and requirements

Production versus Test

At least two separate environments must be created and supported for a quality database implementation: production and test (or development). New development and maintenance work is performed in the test environment; the operational functioning applications are run in the production environment. It is necessary to completely separate the test environment from the production environment to ensure the integrity and performance of operational work. Failure to separate test and production will cause development activities to impair the day-to-day business of your organization. The last thing you want is for errant program code in the early stages of development to access or modify production data. Test access to production data can cause production performance problems. And, of course, test programs modifying production data can create invalid data.

> Separating the test and production environments ensures the integrity and performance of operational work.

The test environment need not be exactly the same as the production environment. The production environment will contain all of the data required to support the operational applications. The test environment, however, can contain only a subset of the data that is required to facilitate acceptable application testing. Furthermore, the test DBMS implementation

usually will not be set up with the same amount of resources as the production environment. For example, less memory will be allocated to buffering and caches, data set allocations will be smaller and on fewer devices, and the DBMS software may be a later version in test than in production (to shake out any bugs in the DBMS code itself before it is trusted to run in production).

The test and production environments should be put together similarly, though. Access to the same system software should be provided in test as is provided to production because the programming staff will need to create their applications in the same type of environment in which they will eventually run.

Another difference in the test environment is the number of databases. Multiple copies of databases may need to be created to support concurrent development by multiple programmers. The DBA must plan and create this environment in such a way as to allow the programming staff to control the contents of the test databases. During the development process programs that modify data in the database may need to be run multiple times. The programmer must be able to ensure, however, that the data at the beginning of each test run is the same. Failure to do so can render the results of the test invalid. Therefore, the DBA must assist the programming staff in the creation of database load and unload jobs to set up test databases for testing runs. Prior to a test run, the database must be loaded with the test data. After the test run, the programmer can examine the output from the program and the contents of the database to determine if the program logic is correct. If not, he can repeat the process, loading to reset the data in the database and retesting. Automated procedures can be put in place to unload the databases impacted by the program and compare the results to the load files.

Trying to predict how test applications will perform once they are moved to production is a difficult task. But the DBA can assist here as well. A relational DBMS typically provides a method to gather statistical information about the contents of its databases. These statistics are then used by the relational optimizer to determine how SQL will retrieve data. This topic is covered in more depth in Chapter 12, "Application Performance." But remember, there will be much less data in test databases than in production. In some cases, though, the DBA can set up scripts to read the production statistics and copy them to the test environment, thereby enabling developers

to more accurately gauge how test applications will perform in production.

Some organizations implement more than two environments, as shown in Figure 1.9. If special care is needed for complex application development projects, additional levels of isolated testing may need to occur. For example, a unit test environment may exist for individual program development, then an integration testing environment to ensure that new programs work together, or that new programs work correctly with existing programs. A quality assurance (QA) environment may need to be established to perform rigorous testing against new and modified programs before they are migrated to the production environment.

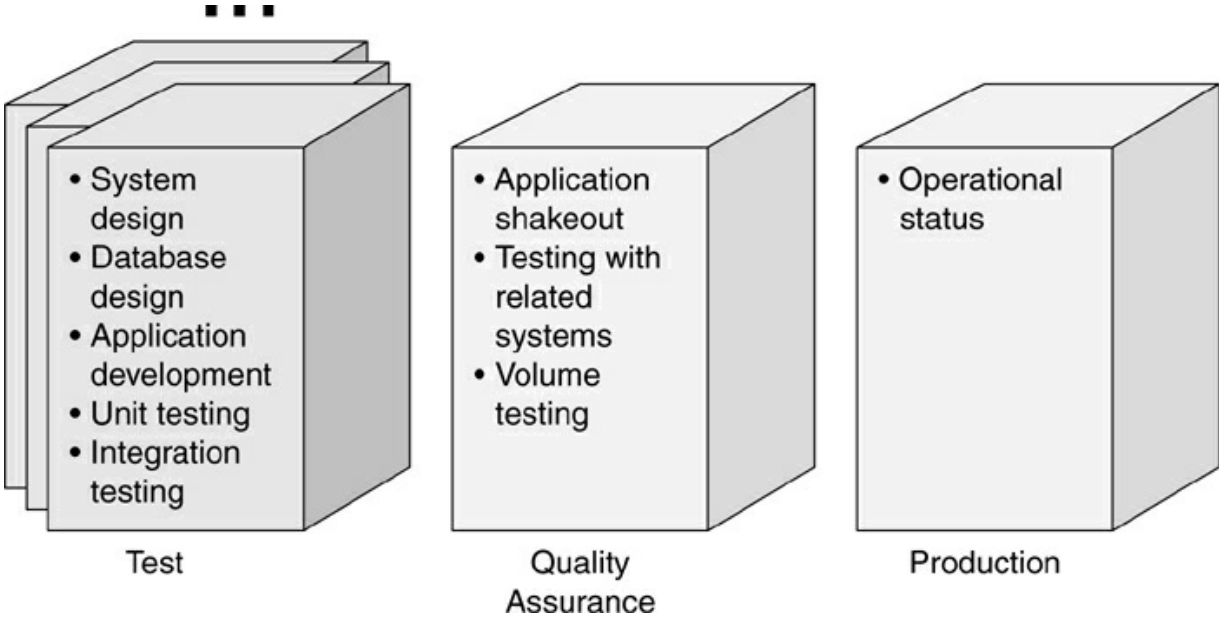

Figure 1.9. Establishing multiple database environments

> A QA environment may be needed to perform rigorous testing against new and modified programs before they are migrated.

The Impact of Newer Technology on DBA

The DBA is at the center of the action whenever new ways of doing business and new technologies are introduced to the organization. Data is at the heart of any application, and most new technology impacts data as it is adopted by application developers. Indeed, data is the lifeblood of modern business; the database houses the data; and the DBA is the expert who understands database technology—and in particular, how databases can be integrated with

other new technologies.

Let’s examine three specific newer technologies that rely on database administration, at least somewhat, to be effectively implemented: database-coupled application logic, Internet-enabled e-business development, and handheld computing.

Procedural DBAs: Managing Database Logic

Traditionally, the domain of a database management system was, appropriately enough, to store, manage, and access data. Although these core capabilities are still required of modern DBMS products, additional procedural functionality is slowly becoming not just a nice feature to have, but a necessity. Features such as triggers, user-defined functions, and stored procedures provide the ability to define business rules to the DBMS instead of in separate application programs. These features tightly couple application logic to the database server.

Since all of the most popular RDBMS products provide sometimes complex features to facilitate database-coupled procedural logic, additional work is required to manage and ensure the optimal usage of these features. This requires an expansion of the management discipline of database administration. Typically, as new features are added, the administration, design, and management of these features are assigned to the DBA by default. Without proper planning and preparation, this can lead to chaos. But first let’s quickly examine how database logic is stored in a DBMS.

Stored Procedures

Stored procedures can be thought of as programs that live in a database. The procedural logic of a stored procedure is maintained, administered, and executed through the database commands. The primary reason for using stored procedures is to move application code from a client workstation to the database server. Stored procedures typically consume less overhead in a client/server environment because one client can invoke a stored procedure that causes multiple SQL statements to be run. The alternative, the client executing multiple SQL statements directly, is less efficient because it increases network traffic which can degrade overall application performance. A stored procedure is a freestanding database object; it is not “physically” associated with any other object in the database. A stored procedure can access and/or modify data in many tables.

Triggers

Triggers are event-driven specialized procedures that are attached to database tables. The trigger code is executed by the RDBMS automatically as data changes in the database. Each trigger is attached to a single, specified table. Triggers can be thought of as an advanced form of rule or constraint written using procedural logic. A trigger cannot be directly called or executed; it is automatically executed (or "fired") by the RDBMS as the result of an INSERT, UPDATE, or DELETE SQL statement being issued on its associated table. Once a trigger is created, it is always executed when its "firing" event occurs.

User-Defined Functions

A UDF, or user-defined function, provides a result based upon a set of input values. UDFs are programs that can be executed in place of standard, built-in SQL scalar or column functions. A scalar function transforms data for each row of a result set; a column function evaluates each value for a particular column in each row of the results set and returns a single value. Once written, and defined to the RDBMS, a UDF becomes available just like any other built-in database function.

Table 1.2 summarizes the differences among stored procedures, triggers, and user-defined functions.

Table 1.2. Procedural Database Objects

Object Type	Definition	Executed	How
Stored procedures	Program logic executed on the database server	By request	Explicit
Triggers	Event-driven procedures attached to database tables	Automatic	Implicit
UDFs	Program logic extending SQL functionality	By request in SQL	Explicit

Administering Stored Procedures, Triggers, and UDFs

Once applications and developers begin to rely upon stored procedures, triggers, and UDFs, steps need to be taken to ensure they are managed properly. DBAs must grapple with the issues of quality, maintainability, efficiency, and availability. How and when will these procedural objects be

tested? The impact of a failure is enterprise-wide, not relegated to a single application. This increases the visibility and criticality of these objects. Who is responsible if they fail? The answer must be: a DBA.

The role of administering procedural database logic should fall upon someone skilled in that discipline. A new type of DBA is required to accommodate the administration of database procedural logic. This new role can be defined as a *procedural DBA*.

The procedural DBA should be responsible for those database management activities that require procedural logic support. This should include primary responsibility for ensuring that stored procedures, triggers, and user-defined functions are effectively planned, implemented, shared, and reused. The procedural DBA also should take primary responsibility for coding and testing all triggers. Stored procedures and user-defined functions, however, will most likely be coded by application programmers and reviewed for accuracy and performance by procedural DBAs.

The procedural DBA should participate in and lead the review and administration of all procedural database objects: that is, triggers, stored procedures, and UDFs. Although procedural DBAs are unlikely to be as skilled at programming as application programmers or systems analysts, they must be able to write and review program code reasonably well. The skill level required depends on what languages are supported by the DBMS for creating procedural objects, the rate and level of adoption within the organization, and whether or not an internal organization exists for creating common, reusable programs. Table 1.3 provides a reasonable level of procedural DBA involvement for each type of procedural object. Additionally, the procedural DBA should be on call for any problems that occur to database procedural objects in production.

Table 1.3. Procedural DBA Involvement by Object

Object Type	Level of Procedural DBA Involvement
Stored procedures	Not likely to actually write stored procedures; must review all code before migration to production; should communicate availability and promote reuse
Triggers	Likely to actually write, test, and debug triggers; must communicate deployment of triggers to ensure application awareness
UDFs	Not likely to actually write user-defined functions; should work closely with the development team to ensure UDF functionality and performance; must review all code before migration to production; must communicate availability and promote reuse

> The procedural DBA should participate in and lead the review and administration of all procedural database objects: that is, triggers, stored procedures, and UDFs.

The role of the procedural DBA requires communication skills as much as it requires technological acumen (see Figure 1.10). In addition to managing and optimizing database procedural objects, the procedural DBA must inform the development community of new triggers, stored procedures, and UDFs. Furthermore, the DBA must promote reuse. If the programmers do not know that these objects exist, they will never be used. Other procedural administrative functions can be allocated to the procedural DBA. Depending upon the number of DBAs and the amount of application development being done, the procedural DBA can be assigned to additional functions, such as

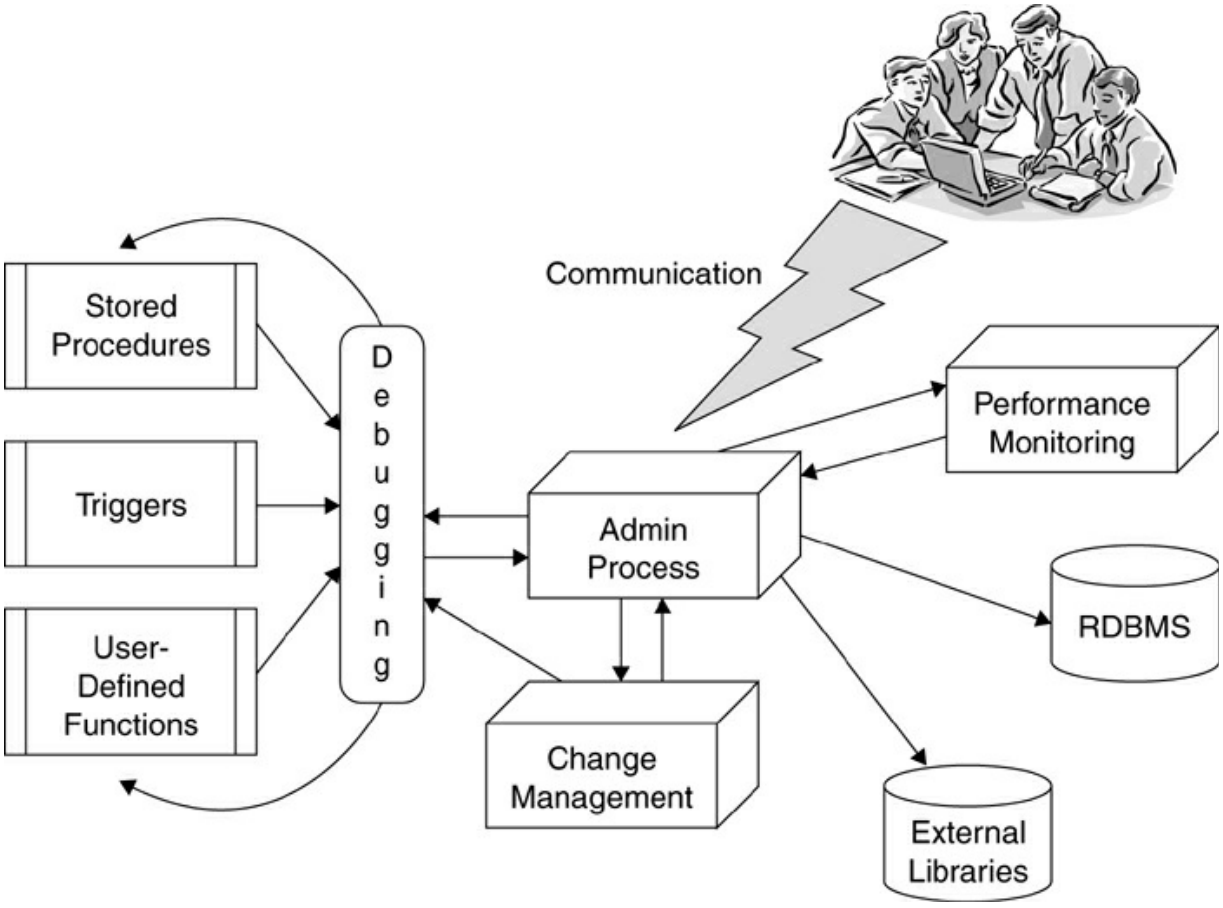

Figure 1.10. Procedural DBA duties

- Participating in application code design reviews
- Reviewing and analyzing SQL access paths (from EXPLAIN or SHOWPLAN)
- Debugging SQL
- Writing and analyzing complex SQL statements
- Rewriting queries for optimal execution

Off-loading coding-related tasks to the procedural DBA can help the other staff DBAs to concentrate on the actual physical design and implementation of databases, resulting in much better-designed databases. The procedural DBA should still report through the same management unit as the traditional DBA. Doing so enables better sharing of skills between the procedural DBAs and traditional data-focused DBAs. Of course, there will need to be a greater synergy between the procedural DBA and the application programmers. The typical job path for the procedural DBA should be from the application

programming ranks because this is where the coding skill base exists.

The Internet: From DBA to eDBA

Although at this late date the Internet can hardly be considered a trend, companies and technologists are still adapting their processes to align with e-commerce. Organizations of every type and size are using Internet technologies to speed up business processes, and database administration practices and procedures are impacted by the adoption of Internet-enabled applications and databases.

E-businesses must be able to adapt and react to constant change. When your business is online, it never closes. People expect full functionality on Web sites they visit regardless of the time. And the Web *is* worldwide. It may be two o'clock in the morning in New York City, but it is always prime time somewhere in the world. An e-business must be available and prepared to engage with customers 24 hours a day, 365 days a year (366 during leap years). Failing to do so can cause loss of business. When a Web site is down, the customer will go elsewhere to do business because the competition is just a simple mouse click away. So those who manage an e-business must be adept, proactive, and ever vigilant.

Adopting the frantic pace of an e-business requires changes for those who keep the e-business operational. DBAs are extremely impacted by e-business. The need to integrate the Web with traditional IT services, such as the DBMS, places high expectations on database administrators.

A DBA who is capable of managing Web-based applications because he understands the special issues that arise because of the Internet is an eDBA. An eDBA also needs to have all of the knowledge and training of a traditional DBA, but these skills need to be adapted to suit applications and databases that are Internet enabled. When the Web is coupled with traditional applications and databases, a complex infrastructure is the result (see Figure 1.11). The eDBA must be capable of navigating this complex, heterogeneous infrastructure and providing expertise wherever databases interact within this infrastructure.

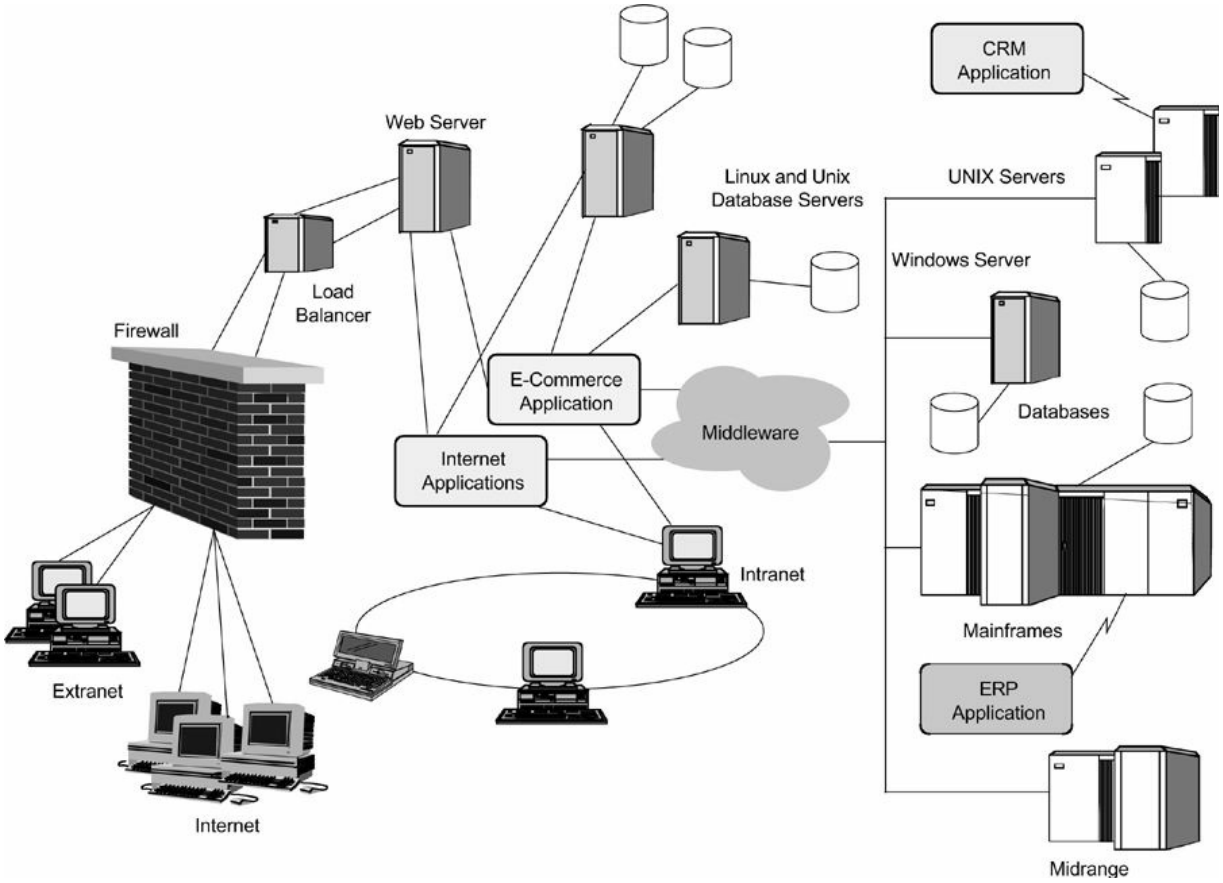

Figure 1.11. The complex infrastructure resulting from Web-to-database capabilities

> A DBA who is capable of managing Web-based applications because he understands the special issues that arise because of the Internet is an eDBA.

Indeed, there are many factors that impact database administration when the Internet is coupled with database technology. Some of these issues include

- 24/7 data availability
- Adoption of new technologies such as Java and XML
- Connectivity to the Web
- Integrating legacy data with modern Web-based applications
- Database and application architecture
- Web-based administration

- Performance engineering for the Internet
- Unpredictable workload

The Personal DBA and the Cloud

Personal devices, usually smartphones but also PDAs (personal digital assistants), are fast becoming a necessity for modern executives and businessmen. A smartphone is a handheld computing device. And sometimes it will have a database management system running on it. Why is that interesting? Does it change the way you will use your device? What will that mean to your IT department?

Popular mobile computing platforms include Symbian OS, Windows Mobile, iPhone OS, and Android.

Smartphones offer many benefits. The devices are small and therefore easily transportable. They do not interfere with a mobile worker's ability to be mobile. And because most everyone carries a mobile phone these days, boosting its capabilities to perform computing tasks is a no-brainer.

Perhaps the biggest benefit of these devices is their ability to run mobile applications. It is not uncommon for an enterprise mobile application to rely on information retrieved from a main computer server to be delivered to the mobile device pervasively, at any place and at any time it is required. Mobile applications using the cloud as the back end are becoming more and more popular.

> Enterprise mobile applications rely on information retrieved from a main computer server.

What Is Cloud Computing?

Cloud computing offers a new model for the delivery of IT resources to users. The primary defining characteristic of cloud computing is to give the illusion of on-demand access to an infinite amount of computing resources. A good example of a cloud computing service is offered by Salesforce.com, which delivers access to a CRM application over the Web.

Another aspect prevalent with cloud computing offerings is that users can rent computing power with no commitment. Instead of buying a server, you can rent the use of one and pay just for what you use. This used to be referred to as utility computing because it mimics how people

pay for utilities, such as water or electricity. It is a “pay as you go” service.

From a database perspective, there are several cloud offerings, including Amazon’s SimpleDB and Google’s App Engine Datastore. Additionally, Microsoft’s SQL Azure supports data in the cloud.

Cloud computing can allow even the smallest of organizations—or indeed, a single individual—to obtain computing resources in ways not previously possible.

The design and implementation of mobile applications is not as straightforward as desktop PC application development. It is very important for mobile application developers to consider the context in which the application will be used.

Although the benefits are significant, there are challenges to be faced as organizations incorporate personal devices into their infrastructure. The data on the device must be managed professionally to ensure integrity and reliability. Because the device is remote, sharing of data can be difficult. The business data on a smartphone must be reliably synchronized with existing enterprise systems and databases. And from a business and compliance perspective, it can be difficult to assess the risk associated with a mobile application. Mobile devices are easily misplaced and without proper security can result in a data breach.

All of the major DBMS vendors provide small-footprint versions of their flagship products to run on personal devices. For example, IBM markets DB2 Personal Edition, Oracle sells Oracle Database Lite, Microsoft provides SQL Server Compact, and Sybase offers Adaptive Server Anywhere. The general idea is to store a small amount of critical data on the mobile device in a database. The local database is later synchronized to long-term data stores on enterprise database servers. Each mobile DBMS provides technology to synchronize data back and forth from the mobile device to the enterprise server platforms.

Impact on DBAs

DBAs are not needed to manage and work on the database on each PDA, but the job of the DBA will be impacted by this development. A database the size of those stored on PDAs should not require the in-depth tuning and

administration that are required of enterprise database implementations. However, DBAs will be called upon to help design appropriately implemented databases for small-form-factor devices like PDAs. But this is not the biggest impact.

A big impact on the DBA is in the planning for and management of the data synchronization from hundreds or thousands of PDAs. When should synchronization be scheduled? How will it impact applications that use large production databases that are involved in the synchronization? How can you ensure that mobile users will synchronize their data reliably and on schedule?

Additionally, for cloud implementations, DBAs will likely be responsible for assuring reliable data availability. Designing and tuning a database implementation for cloud computing can require significant resources to manage large amounts of data and assure around-the-clock availability.

These are not minor issues. Before implementing a large battalion of mobile database users who must synchronize their data, make sure that your DBA staff is prepared for the impact on their databases. Like most anything, failure to prepare is a recipe for sure disaster. But prepare we must. The DBA staff must be ready to support the mobile workforce by understanding data synchronization technology, cloud computing, and the potential need for remote database users in your organization.

An additional step in the preparation process for supporting databases on handheld devices is to review the applications in your organization and try to determine which might be impacted first. Companies with remote workers such as a distributed sales force or delivery tracking services will most likely be the first impacted. Take some time to review the data requirements of those applications and how a large influx of remote connections might impact the current systems.

Pervasive computing and the mobile workforce are here to stay. And the DBA staff must be ready to support these mobile workers with a valid, shared data infrastructure.

NoSQL, Big Data, and the DBA

NoSQL is another trend that can impact the job performed by the DBA. NoSQL is a movement that, at its most basic, is described by its title. That is, a NoSQL DBMS does not support SQL. At a high level, NoSQL implies nonrelational, distributed, flexible, and scalable. Most NoSQL offerings are

also open source.

NoSQL grew out of the perceived need for “modern” database systems to support Web initiatives. Additionally, some common attributes of NoSQL DBMSs include the lack of a schema, simplicity of use, replication support, and an “eventually consistent” capability (instead of the typical ACID—atomicity, consistency, isolation, and durability—transaction capability). It really does not mean no SQL support. Some NoSQL offerings have begun to support SQL, leading some pundits to define NoSQL as “Not Only SQL.”

Examples of NoSQL Offerings

- Cassandra: http://cassandra.apache.org/
- CouchDB: http://couchdb.apache.org/
- HBase: http://hbase.apache.org/
- mongoDB: www.mongodb.org/
- Riak: www.basho.com/

The NoSQL movement is tied to the Big Data movement. NoSQL databases are designed to deliver low-cost storage and access to large amounts of data.

With NoSQL implementations the data is typically accessed in one way and there is little to no flexibility in terms of generating off-the-cuff, ad hoc queries.

Impact on DBAs

Another hallmark of NoSQL is that it requires very little database administration work. Of course, the DBMS must be set up and managed, and the data must be backed up. So be careful when anyone tries to tell you that any database system requires no database administration. It just isn’t true.

New Technology Impacts on DBA

As new technology is introduced to the organization, the DBA group is typically the first to examine and use it. The preceding technologies are merely examples of recent trends and technologies that require database administration for efficient and effective implementation. Most new technologies will have some impact on the role of the DBA.

DBA Certification

Professional certification is an ongoing trend in IT and is available for many different IT jobs. The availability and levels of certification have been growing at an alarming rate for database administration. Certification programs are available for most of the popular DBMS platforms, including IBM DB2, Microsoft SQL Server, and Oracle. The concept behind certification of DBAs is to certify that an individual is capable of performing database administration tasks and duties.

This is a noble goal, but the problem is that passing a test is not a viable indicator of being able to perform a complex job like DBA. Some things you just have to learn by doing. Now I am not saying that certification is useless. Indeed, taking the test and focusing on the questions you miss can help to point out areas of weakness upon which you can improve. But does anyone *really* believe that someone who passes a formal test will be as capable as someone with several years of experience as a DBA? Organizations should hire DBAs based on past experience that indicates a level of capability. Of course, someone with both experience and certification is better than someone with only one of the two.

> Certification can make you more employable.

I do recommend that professional DBAs take the time to study and pass the certification exams, not because certification will make you a better DBA, but because it will make you more employable. Some companies will hire only certified professionals. The trend toward using certification to guide hiring practices will increase because of increasing IT complexity. If you think you might change jobs at some point in your career (and who among us will not?), certification is a worthwhile pursuit.

Keep in mind that the DBA certification tests sometimes ask arcane syntax questions that are not really good indicators of a DBA's skills. Getting the syntax 100 percent accurate is what manuals and DBA tools are for. There is no reason to memorize syntax because it tends to change quite often. It is better to know where to find the syntax, parameters, and answers to your questions when you need them, that is, which manuals and textbooks contain the needed information. DBAs should possess a broad overarching knowledge of DBMS concepts and IT fundamentals, and a good knowledge

of the way in which their organization's database systems work. Memorizing every detail about SQL syntax and structure is a waste of time because it is complex and changes all the time. In other words, it is better to know off the top of your head that something can (or cannot) be done than to know the exact syntax for how to accomplish it.

> Certification tests sometimes ask arcane syntax questions that are not really good indicators of a DBA's skills.

If you decide to pursue certification, take the time to prepare for the tests. There are books and self-learning software titles available that can be quite useful. These books and programs cover the most likely test topics and provide sample questions to help you prepare. In many ways it is like preparing for a college entrance exam, such as the SATs.

And once you earn your certification, make sure you display it proudly on your résumé and your business card (if your company allows it).

Consult Table 1.4 for Web sites that contain information about professional certification for the most popular DBMS products.

Table 1.4. Sources of DBA Certification Information

DBMS	Web Site with Certification Information
Oracle, MySQL	http://education.oracle.com/
Microsoft SQL Server	www.microsoft.com/learning/en/us/default.aspx
IBM DB2, Informix	www.ibm.com/certify
Sybase	www.sybase.com/detail?id=1009636

The Rest of the Book

This first chapter has introduced you to the world of the DBA. Ideally you have garnered a respect for the complexity of the position and the qualities required of a good DBA. The remainder of the book will examine the details of the tasks, roles, and responsibilities required of the DBA. Read on to learn of the challenges faced by DBAs and how to overcome them.

Review

1. At a high level, discuss the primary job responsibilities of a DBA.

2. What is the single biggest problem faced by organizations using relational databases?
3. What is the difference between a data administrator and a database administrator?
4. What factors determine the number of DBAs required to properly support an organization's database environment?
5. How does new technology impact the job of the DBA?
6. Cite the technology influences that mandate the need for procedural DBAs.
7. What is the difference between a database architect and a system administrator?
8. What job function is most likely to be responsible for installing a new DBMS release?
9. What are the three types of integrity that DBAs must understand?
10. Is a certified DBA necessarily a qualified DBA? Why or why not?

Bonus Question

Why must the DBA be prepared to function as a jack-of-all-trades?