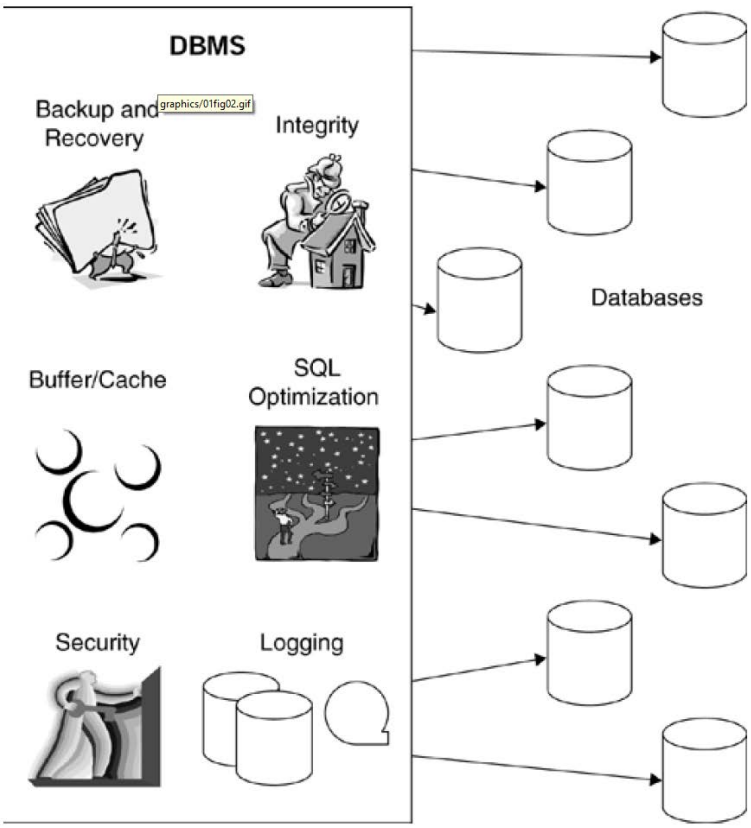
Database Technology

This is not a trivial matter; people sometimes think they know more than they actually do. For example, what is a database? I'll bet most readers believe they know the answer to that question. However, some (perhaps many) of you would be wrong. Oracle is not a database; it is a database management system. You can use Oracle to create a database, but Oracle, 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). And what is a database management system? A DBMS is software that enables end users or application programmers to share and manage data. It provides a systematic method of creating, updating, retrieving, and storing in formation in a database. A DBMS is also generally responsible for data integrity, data security, data access control and optimization, automated rollback, restart, and recovery. Figure 1-2 shows the relationship between a DBMS and a database.

Figure 1-2. Relationship of DBMS to database



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

You might think of a database as a file folder, and a DBMS as the file cabinet holding the labeled folders. You implement and access database instances using the capabilities of the DBMS. Your payroll application uses the payroll database, which may be implemented using a DBMS such as DB2, Oracle9i, or SQL Server.

Why is this important? If we don't 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 in use, and there may be many in the organization. For example, a large organization might use DB2 on the mainframe, Oracle and Informix on several different UNIX platforms, and SQL Server on Windows 2000. Older legacy systems might use IMS databases, and then there is that one crazy application out there that uses a fringe DBMS like Adabas or Ingres.

[The DBA group, therefore, must have expertise in each of these different management systems and platforms. Furthermore, the DBA must be capable of determining which DBMS and platform is best 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 impartially and implement decisions based on the best fit of application, DBMS, and platform.

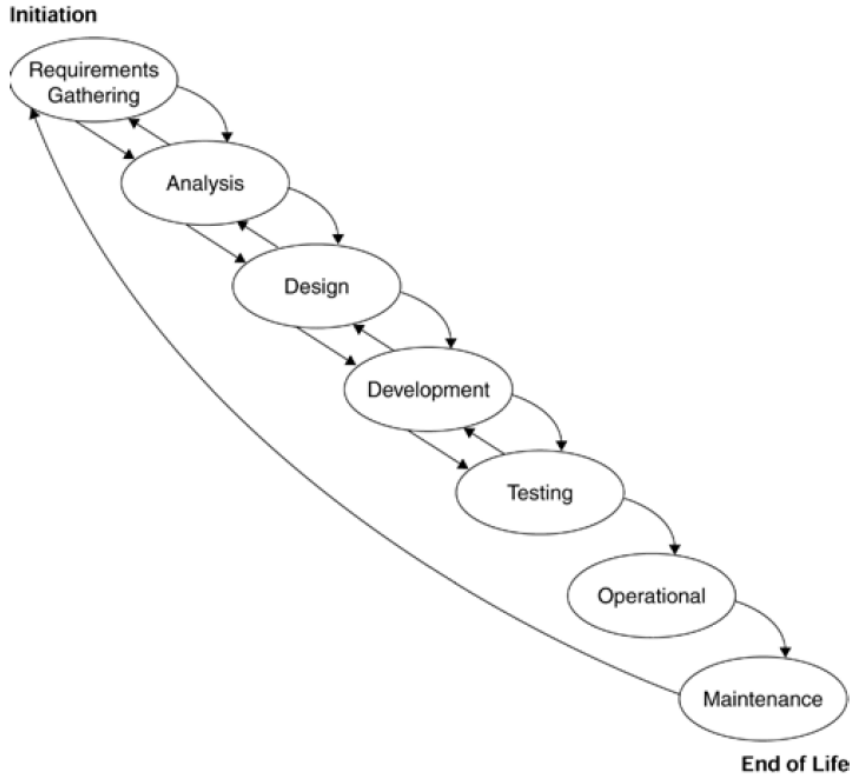
The Management Discipline of Database Administration

Database administration is rarely approached as a management discipline. The term discipline implies a plan, 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, such as understaffing, overcommitment to supporting new (and even legacy) application development projects, lack of repeatable processes, or lack of budget. The reactive DBA functions more like a firefighter than an administrator; he attempts to resolve problems only after problems occur. The reactive DBA is focused on resolving the biggest problem confronting him.

In contrast, the proactive DBA implements practices and procedures to avoid problems before they occur. A proactive database administrator develops and implements a strategic blueprint for deploying databases within the organization. This plan should address all phases of the application development life cycle. A data specialist, usually the DBA, should be involved during each phase of the cycle, as shown in Figure 1-3. During the initiation and requirements gathering phase, 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.

Figure 1-3. The application development life cycle



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 SQL Server. Sample data must be populated into the physical database to allow 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 ensures 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 the production environment.

While the application is operational, the DBA performs a host of duties including assuring availability, performance monitoring, tuning, backup and recovery, and authorization management. However, no application or database remains static for long. Because business needs change over time, the IT systems that support the business will also change. When maintenance is requested, the DBA becomes engaged in the entire process once again, from requirements gathering to implementation.

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 life of the application? Does the business have any privacy policies that impose special rules for handling the data.

Privacy Policies and Data

Bankruptcy of e-business toy seller Toysmart.com provides a good lesson in the impact of privacy policies on corporate data. In May 2000, Toysmart filed for bankruptcy and announced its intention to sell its database of customer information. Toysmart's customer list was estimated to contain information on 250,000 former customers, including their names, phone numbers, street and e-mail addresses, and product preferences. However, Toysmart's own privacy policy, previously posted on its Web site, promised that it would not disclose the personal information of its customers to third parties.

The FTC and a group of state attorneys general blocked the sale on the grounds that the sale would violate Toysmart's privacy policy. They argued that Toysmart's customers conducted business with Toysmart.com under the conditions of the privacy policy. The court ruling further stipulated that the company had to provide an affidavit describing how the list was destroyed.

This is just one example of how privacy policies can impact database administrators 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.

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 implemented. Furthermore, ad hoc database access is a requirement for many organizations.

Additionally, the DBA is in charge of setting up an ad hoc query environment that includes evaluating and implementing query and reporting tools, 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 hectic. The DBA maintains production and test environments, monitors active application development projects, attends strategy and design meetings, selects and evaluates new products, and connects legacy systems to the Web. And, of course: Joe in Accounting, he just resubmitted that query from hell that's bringing the system to a halt. Can you do something about that? All of this can occur within a single 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 do not expect any private time: A DBA must be prepared for interruptions 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." A DBA is rarely approached with a question like "I've got some really bad SQL here. Can you help me fix it?" Instead, the DBA is forced to investigate problems where the underlying assumption is that the DBMS or perhaps the DBA is at fault, when the most common cause of relational performance problems is inefficiently coded applications.

When application problems occur, the database is "guilty until proven innocent."

Oftentimes the DBA is forced to prove that the database is not the source of the problem. The DBA must know enough about all aspects of IT to track down errors and exonerate the DBMS and database structures he has designed. So he must be an expert in database technology, but also have semi-expert knowledge of the 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 elements makes the DBA a very valuable resource. It also makes the job interesting and challenging.

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 their DBAs. Different organizations place different value on the DBA job. It is imperative to your career development that you scout for progressive organizations that understand the complexity and ongoing learning requirements for 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 queueing, and the like?
- Does the company allow DBAs to regularly attend local user groups? 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 view 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 and the technical aspects of data. The business aspects of data are aligned with 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 responsible for using the DBMS and a system administrator or systems programmer responsible for installing and upgrading the DBMS.

Data Administration

Data administration separates the business aspects of data resource management from the technology used to manage data; it is more closely aligned with the actual business users of data. The data administrator (DA) is responsible for understanding the business lexicon and translating it into a logical data model. Referring back to the ADLC, the DA would be involved more in the requirements gathering, analysis, and design phase, the DBA in the design, development, testing, and operational phases.
Another difference between a DA and a DBA is the focus of effort. The DA is responsible for the following tasks:

- Identifying and cataloging the data required by business users
- Producing conceptual and logical data models to accurately depict the relationship among data elements for business processes
- Creating 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. However, in my experience, the DA is never given an executive position, which is unfortunate. Many IT organizations state that they treat data as a corporate asset, a statement that is belied by 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: charts of account, organization charts, reporting hierarchies, building blueprints, office layouts, and bills of material. Each is tracked and protected. Professional auditors are employed to ensure that no discrepancies exist in a company's accounting of its assets. Can we say the same thing about data?
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 companywide. 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 a DA and a 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 this: Is the number 12

- A date representing December, the twelfth month of the year?
- A date representing the twelfth day of some month?
- An age?
- A shoe size?
- Or, heaven forbid, an IQ?

And so on. However, there are other, more technical aspects of metadata, too. Think about the number 12 again.

- 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 is its data type? Is it an integer or a decimal number with a 0 scale?

Metadata provides the context by 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 by which data can be understood and therefore become information.
A comprehensive metadata strategy enables an organization to understand the information assets under its control and to measure the value of those assets.

One of the biggest contributions of data administration 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 an organization.
Many DBAs dismiss data administration as mere data modeling, required only because someone needs to talk to the end users to get the database requirements. However, 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, very few organizations have implemented and staffed a DA role. The larger the organization is, the more likely that a DA function exists. However, when the DA role is undefined in an organization, the DBA must assume the mantle of data planner and modeler. Unfortunately, the DBA will usually not be able to assume all of the functions and responsibility of a DA for a number of 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 enabling 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 nontechnicians.

When DA and DBA functions coexist within the organization, the two groups must work very closely with one another. It is not necessary that both have the same manager, though it would facilitate cooperation. At any rate, it is imperative that some skills cross-pollinate 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 is more effective with some knowledge about the other.

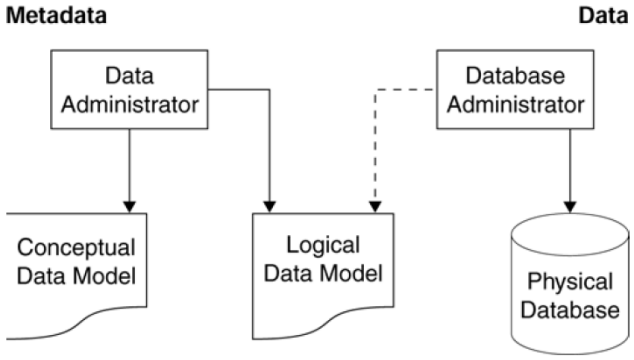
Organizations truly concerned about data quality, integrity, and reuse will invariably implement and staff the DA function.

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

Database Administration

The first duty of the DBA is to understand the data models built by the DA and 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 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 a DA should rely on a DBA for the conceptual and logical data models. Figure 1-4 depicts this relationship.

Figure 1-4. DBA vs. DA



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 ongoing support of the databases created from the physical design and management of the applications that access those databases.

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

System Administration

Some organizations, usually the larger ones, also have a system administrator (SA) or systems programming role that impacts DBMS implementation and operations. The SA is responsible for the installation and setup of the DBMS. The 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.

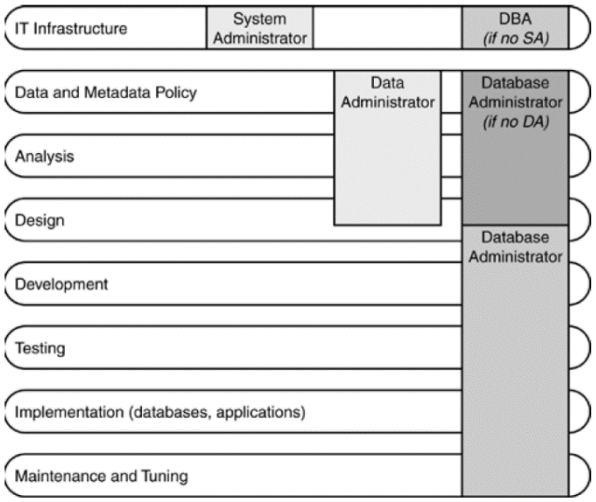
The system administrator ensures that the IT infrastructure is operational for database development by setting up the DBMS appropriately, applying ongoing maintenance from the DBMS vendor, and coordinating migration to new DBMS releases and versions.

Furthermore, the SA 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 queueing 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.

As with data administration, there must be cross-training of skills between the SA and DBA. The SA will never understand the physical database like the 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.

If no system administration group exists, or if its focus is not on the DBMS, the DBA assumes responsibility for DBMS-related system administration and programming. Figure 1-5 delineates the responsibilities of the DA, the DBA, and the SA.

Figure 1-5. DA, DBA, and SA responsibilities



Best Practices Every SQL Server DBA Must Know

My Assumptions About You

- You may be a DBA Administrator or DBA Developer.
- You may be a part-time or full-time DBA.
- You probably have less than one year's experience as a SQL Server DBA, but you are familiar with SQL Server basics.
- If you are an experienced DBA, then you probably are already familiar with most of this content. On the other hand, maybe you may pick up a tip or two, or may be reminded about something you need to do, but have forgotten about.

What We are Going to Learn Today

A brief explanation why following best practices are beneficial to you as a DBA.

- Common best practices all DBAs should follow.
- Our focus today is on what to do, not how to do it.
- These best practices are just the basics, there is a lot more to do and learn.
- These best practices are based on many of the common mistakes I see novice DBAs make.
- There are always exceptions to every rule, and not every recommendation discussed here may fit your environment.

Benefits of Focusing on Best Practices

By focusing on SQL Server best practices basics, it helps you as a DBA to:

- Optimize SQL Server performance
- Maximize SQL Server availability
- Be proactive, reducing the amount of time you spend being in "crisis mode"

Everything Counts

- While many of the best practices I discuss today might seem small in scope, the accumulative effect of following each and every recommendation can be huge.
- By following best practices consistently, SQL Server performance and availability can be boosted substantially.

Best Practices You Should be Following

- Don't Shrink Files
- Create Index Rebuilding/Reorganize Job
- Create Data Corruption Detection Job
- Set Up Alerts for Critical Errors
- Implement a Backup/Restore Strategy
- Create a Disaster Recovery Plan
- Document Everything

- Test Everything
- Installing & Upgrading SQL Server
- General Server Configuration
- Security Basics
- SQL Server Property Settings
- Memory Configuration
- User Data and Log File Management
- Tempdb Management
- Database Property Settings
- Configuring Jobs—General Guidelines

Installing & Upgrading SQL Server

- Generally, when installing a new SQL Server instance:
- Use the newest hardware drivers.
- Use the newest OS version with latest SP.
- Use the newest SQL Server version with latest SP & Hot Fixes
- Test, and once stable, be wary of making changes.
- Generally, when upgrading an existing SQL Server instance:
- Don't upgrade unless you have a good reason to upgrade. If your instance is working well, don't mess with it.
- For example, upgrade if you need new features, or have problems with an old installation, or need to upgrade hardware.
- It is always safer to upgrade to a new server with a fresh installation of the OS and SQL Server than to upgrade in place. This allows you to test more effectively, and also gives you a "back out" option.

Security Basics

- Don't give users more permissions than they need to perform their job. (Critical. Sounds simple, often hard.)
- Don't use the SA account for anything. Assign it a complex password, and keep it handy just in case. Use a domain account that is a member of the sysadmin role.
- Don't allow an application to use the SA or a sysadmin account to access SQL Server.
- Use Windows Authentication security whenever possible. (Applicable for in-house development).
- Don't give vendors sysadmin access to your servers.
- Log off or lock your SQL Server (or workstation) when done.
- General Server Configuration
- Ideally, SQL Server instances should run on a stand-alone server (physical or virtual) with no other apps running on it.
- Avoid multiple instances unless you have a really good reason to use them. Consider virtualization instead.
- Unnecessary SQL Server services should be uninstalled or turned off.
- Ideally, don't run antivirus/antispyware software locally.
- If your organization's policy requires running antivirus/antispyware software locally, exclude MDF, NDF, LDF, BAK, and TRN files.

SQL Server Property Settings

- Don't change any of the default SQL Server instance-wide configuration property settings unless you thoroughly understand the implication of making the change. Examples of Server Property settings include:
- Memory
- Processors
- Security
- Connections
- Database Settings
- Advanced
- Permissions

Memory Configuration

- Ideally, use 64-bit hardware and the 64-bit version of the OS and SQL Server.

- Generally speaking, if using 64-bit memory, turn on "Lock Pages in Memory," and let the instance dynamically manage its own memory (especially 2008).
- If using the 32-bit version of SQL Server, and if using 4 GB or more of RAM, ensure that /3GB switch and AWE memory are correctly configured. Correct settings depend on available RAM.

Data and Log File Management

- Remove physical file fragmentation before creating new MDF or LDF files.
- When creating new MDFs and LDFs, pre-size them to minimize autogrowth events.
- MDF files should be located on their own disks.
- LDF files should be located on their own disks.
- BAK and TRN backup files should be located on their own disks.

Instant File Initialization

- Enable instant file initialization, which prevents MDF files from being zeroed out when they are grown, which allows MDF files to be created quickly. LDF files are not affected.
- Speeds up CREATE DATABASE, ALTER DATABASE, RESTORE DATABASE, Autogrowth.
- Requires SQL Server 2005/2008, and Windows Server 2003/2008 (or higher version).
- Instant file initialization is turned on if the SQL Server (MSSQLSERVER) service account has been granted the SE_MANAGE_VOLUME_NAME permission by adding the account to the Perform Volume Maintenance Tasks security policy. Members of the local Windows Administrator group automatically have this right.

Tempdb Management

- Pre-size tempdb so autogrowth doesn't have to happen often (8MB is default, which is very low).
- Set autogrowth to avoid many growth spurts, use a fixed amount that minimizes autogrowth use. (10% is default, which causes lots of autogrowth).
- If tempdb is very active, locate it on its own disks.
- If very active, consider dividing the tempdb into multiple physical files so that the number of files is % to 4 the number of CPU cores, up to 8 files. Each physical file must be the same size.

Database Property Settings

- Don't change database property settings unless you have a very good reason.
- Auto Create Statistics: On
- Auto Update Statistics: On
- Auto Shrink: Off
- Autogrowth: Leave on. Use mainly for catching mistakes. File growth should be managed manually. Use fixed amount that minimizes autogrowth occurrences.
- Recovery Mode: Set to full for all production databases so transaction log backups can be made.
- Page Verify: Use Checksum (2005/2008), don't turn off.
- Compatibility Level: Should be set to match current server version, unless there are compatibility problems.
- Configuring Jobs—General
- If your server doesn't have any jobs, then there is a problem, as all servers need jobs.
- Try to schedule jobs so they don't interfere with production.
- Try to prevent jobs from overlapping.
- Set alerts on jobs so you are notified if they fail.
- Check jobs daily to verify that they have run correctly (not hung, not run abnormally long, etc).
- If you use the Maintenance Plan Wizard, be careful to use it properly. If misused, it can create maintenance jobs that hurt performance.

Don't Shrink Files

- If you properly size your MDFs and LDFs, then you should never have to shrink a file.
- Don't schedule database or file shrinking operations.
- If you must shrink a database:
- Do so manually
- Rebuild the indexes after the shrink is complete
- Schedule these steps during the slow time of the day
- Benefits of not automatically shrinking files:
- Eliminates grow and shrink syndrome

- Reduces physical file fragmentation
- Reduces resources used for these operations, allowing more important tasks to use them

Create Index Rebuilding/Reorganize Job

- Indexes need to be rebuilt or reorganized regularly to minimize fragmentation and reduce wasted space.
- Consider rebuilding an index if it is heavily fragmented (>30%). In Enterprise Edition, can perform online. If Standard Edition, consider it an off-line job. This automatically updates statistics, so you don't need to do this again.
- Consider reorganizing an index if it is not heavily fragmented (>5% and <= 30%). This is an online operation and doesn't use a lot of resources. You must update statistics afterwards, as this is not automatical y done for you.
- Ideally, you should only rebuild or reorganize indexes that need it. Use sys.dm_db_index_physical_stats to identify what tables/indexes need to be rebuilt/reorganized.

Create Data Corruption Detection Job

- Ideally, run DBCC CHECKDB as frequently as practical.
- Create an appropriate job to run this (or similar) command:
- DBCC CHECKDB ('DATABASE_NAME') WITH NO_INFOMSGS, ALL_ERRORMSGS;
- Note: Consider using PHYSICAL_ONLY option for large or busy production servers to reduce run time.
- If you have a problem, you want to find it as soon as possible to reduce the risk of data loss. Don't use the DBCC CHECKDB repair option unless you fully understand its implications.

ServerCentraf. contS

Implement a Backup/Restore Strategy

- Create a job to perform full backups daily on all system and user production databases, plus log backups hourly (or similar variation).
- If a database uses the bulk or full recovery model, you must back up the transaction log to keep in from growing uncontrollably.
- Backup using RESTORE WITH VERIFYONLY to help verify backup integrity. (Does not guarantee good backups.)
- Periodically test backups to see if they can be restored.
- Set up an appropriate backup retention policy.
- Store backups securely and off-site (not on same disk array or SAN).
- If you have a limited backup window, or have limited disk space, use backup compression. Can be a big time saver.

Sample Maintenance Scripts

ServerCentrai. contS

- Set Up Alerts for Critical Errors
- Create a SQL Server Event Alert for all events with a severity of 19 [fatal] and higher.
- Have alerts sent to you or whoever is responsible for day-to-day monitoring.
- Consider a third-party alerting tool if SQL Server Alerts doesn't meet all of your needs.
- Create a Disaster Recovery Plan
- You must create a document that outlines, step-by-step, in great detail, how you will recover your SQL Servers in the case of any problem, small or large.
- You need to practice using the plan so you are familiar with it and can easily implement it.
- Keep Microsoft SQL Server's Product Support phone number handy. Paste it near your computer.
- Remember: Most "disasters" are small, such as a corrupted database. Big "disasters" occur very rarely, if ever. But you need to be prepared for both.
- Document Everything
- Yes, documentation is very boring, but it is very critical to being a successful DBA. Be sure to document:
- The installation and configuration of each instance.
- The installation and configuration of any application that uses SQL Server as its back end (as related to SQL Server).
- Troubleshooting tasks, as the same problem may reoccur, and you don't want to reinvent the wheel.
- Any time any change is made to any instance for any reason.
- Be sure that documentation is easily available to everyone who needs access to it.

Test Everything

- Before you make any change on a production SQL Server, be sure you test it first in a test environment.
- NO EXCEPTIONS!
- I mean it!
- Really!
- No kidding.
- I wouldn't lie to you.
- You don't want to loose your job.
- You'd be crazy not listening to this advice.
- Civilization as we know it may lie in your hands.
- Never forget, DBAs are the protectors of the organization's data. You took this oath when you accepted the job of DBA.

Take Homes for Today

- By focusing on best practices, you gain the following:
- Better SQL Server performance
- Higher SQL Server availability
- Being proactive helps to you prevent being in a "crisis" mode all the
- time
- The total effect of following each and every recommendation made today can be huge.
- What you learned today is only the tip of the iceberg, you will need to take time to learn many other best practices.
- When you get back to work, use this as a checklist to give your SQL Servers a quick health check.