DATABASE SECURITY

GENERAL DATABASE SECURITY CONCEPTS

Modern databases must meet different goals. They must be reliable, provide for quick access to information, and provide advanced features for data storage and analysis.

Many organizations rely on databases to serve as the “back end” for purchased applications or custom-developed applications. The “front end” of these systems are generally client applications or web user interfaces.

Architecturally, relational databases function in a client-server manner running. They cannot directly access the database files, as can be done with “desktop” database systems, such as Microsoft Access.

DATABASES CAN BE USED IN VARIOUS CAPACITIES,

Application support

Relational databases are the most commonly used method for storing data. Through the use of modern databases, users and developers can rely on security, scalability, and recoverability features.

Secure storage of sensitive information

Relational databases offer one of the most secure methods of centrally storing important data. Organizations must take precautions against data breaches, unauthorized access, and data corruption.

DATABASES CAN BE USED IN VARIOUS CAPACITIES,

Online transaction processing (OLTP)

OLTP services are often the most common functions of databases in many organizations. These systems are responsible for receiving and storing information that is accessed by client applications and other servers.

Data warehousing

Data warehousing is the act of gathering, compiling, and analyzing massive volumes of data from multiple sources to assist commercial decision-making processes is known as data warehousing. The data warehouse acts as a central store for data, giving decision-makers access to real-time data analysis from a single source of truth.

UNDERSTANDING DATABASE SECURITY LAYERS

Since relational databases can support a wide array of different types of applications and usage patents, they generally utilize security at multiple layers.

Each layer of security is designed for a specific purpose and can be used to provide authorization rules. In order to get access to your most trusted information, users must have appropriate permissions at one or more of these layers.

SERVER-LEVEL SECURITY

A database application is only as secure as the server it is running on. Therefore, it's important to start considering security settings at the level of the physical server or servers on which your databases will be hosted. One of the first steps you should take in order to secure a server is to determine which users and applications should have access to it.

Modern database platforms are generally accessible over a network, and most database administration tasks can be performed remotely. It's also very important to physically protect databases in order to prevent unauthorized users from accessing database files and data backups. If an unauthorized user can get physical access to your servers, it's much more difficult to protect against further breaches.

NETWORK-LEVEL SECURITY

Databases work with their respective operating system platforms to serve users with the data they need.

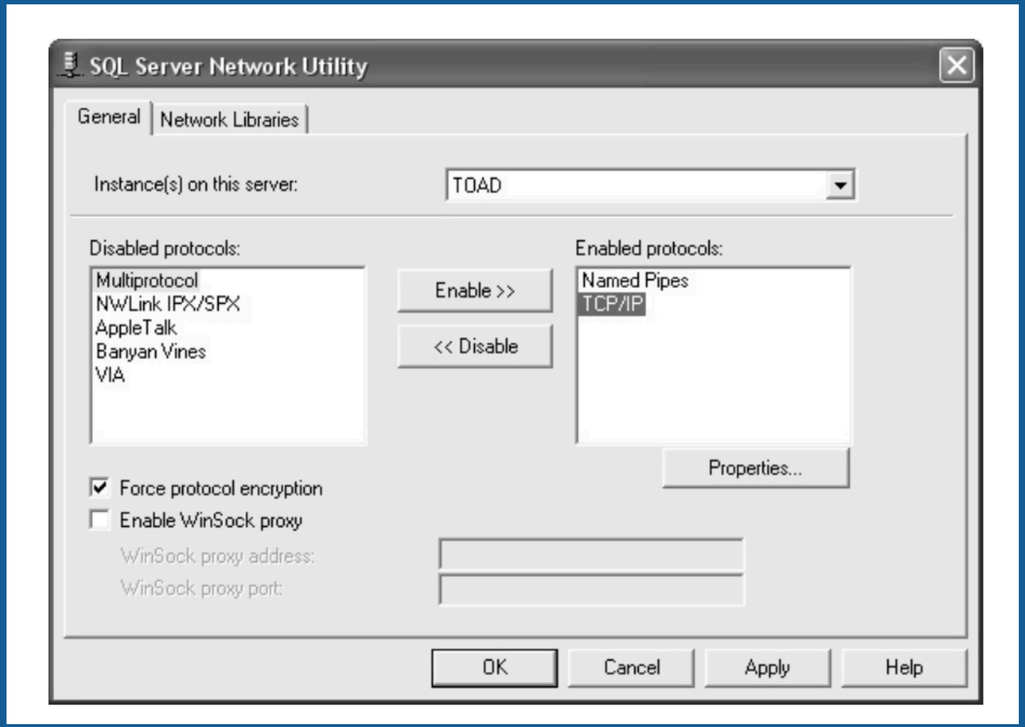
Some standard “best practices” for securing databases include limiting the networks and/or network addresses that have direct access to the computer.

For example, you might implement routing rules and packet filtering to ensure that only specific users on your internal network will even be able to communicate with a server.

You can apply data encryption tools and technologies across multiple devices. Data encryption enables more secure communications.

NETWORK-LEVEL SECURITY

As an example, Microsoft's SQL Server database platform uses a default TCP port of 1433 for communications between clients and the database. If you know for certain that there is no need for users on certain subnets of your network to be able to access this server directly, it would be advisable to block network access to this TCP port.



DATA ENCRYPTION

Massive quantities of sensitive information are managed and stored online in the cloud or on connected servers.

This is because it’s virtually impossible to conduct business or go through personal life day to day without your sensitive data being transmitted and stored by the networked computer systems of various organizations.

Data encryption algorithms scramble plaintext so that only the person with the decryption key can read it. This process provides data security for personal information that users receive, send, and store on mobile devices, including those connected to the IoT.

DATA ENCRYPTION

There are two main kinds of data encryption: symmetric encryption and asymmetric encryption.

In symmetric encryption, a single, private password both encrypts and decrypts data.

Asymmetric encryption, sometimes referred to as public-key encryption or public-key cryptography, uses two keys for encryption and decryption. A shared, public key encrypts the data. A private, unshared key that must remain protected decrypts the data.

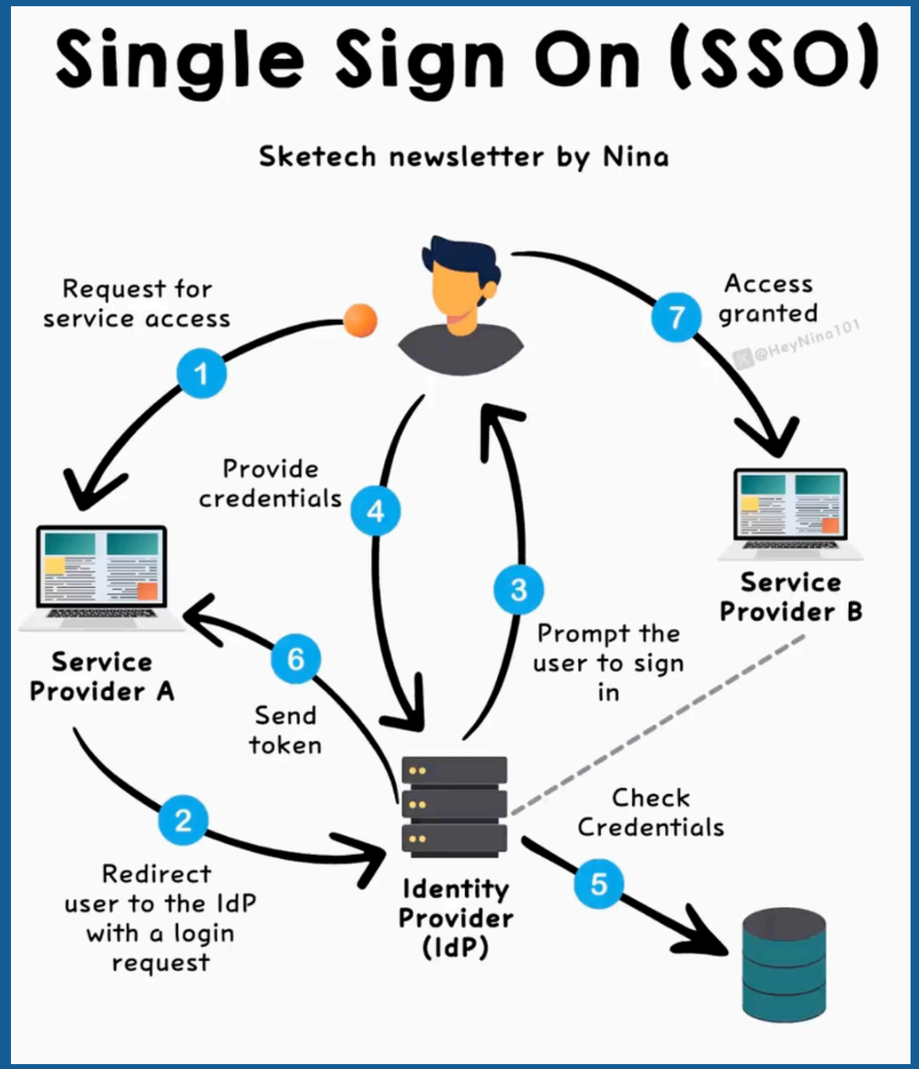
OPERATING SYSTEM SECURITY

Network configuration settings, file system permissions, authentication mechanisms, and operating system encryption features can all play a role in ensuring that databases remain secure.

The operating system is responsible for managing hardware resources, executing programs, and providing security mechanisms to ensure safe and authorized use of a computer system.

In environments that use a centralized directory services infrastructure, it’s important for systems administrators to keep permissions settings up to date and to ensure that unnecessary accounts are deactivated as soon as possible.

MANAGING DATABASE LOGINS



This first level of database security can be based on a standard username and password combination.

Or, for improved manageability and single sign-on purposes, the database systems can be integrated with an organization's existing authentication system.

Among the many benefits of this method is the ability to centrally administer user accounts. When a user account is disabled at the level of the organization's directory service, no further steps need to be taken to prevent the user from accessing database systems.

MANAGING DATABASE LOGINS

In addition, organizations are increasingly turning to biometric-based authentication, as well as smart-card and token-based authentication.

Another important consideration to keep in mind is that most relational database platforms allow operating system administrators to have many implicit permissions on the database.

For example, systems administrators can start and stop the services and can move or delete database files.

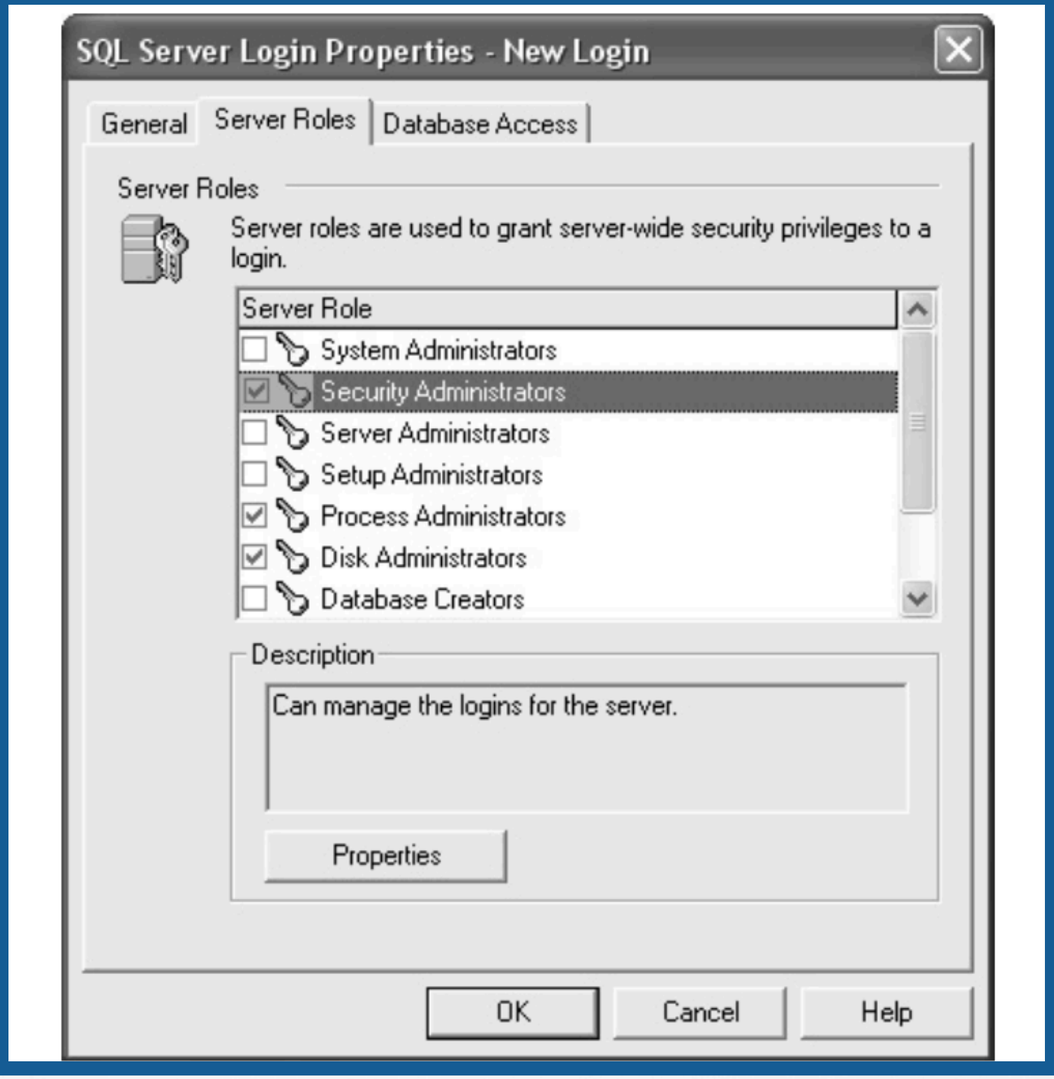
Additionally, some database platforms automatically grant to the systems administrator a database login that allows full permissions. Although this is probably desirable in some cases, it’s something that must be kept in mind when trying to enforce overall security.

UNDERSTANDING DATABASE-LEVEL SECURITY

Database Administration Security

One important task related to working with a relational database is maintenance of the server itself. Important tasks include creating databases, removing unneeded databases, managing disk space allocation, monitoring performance, and performing backup and recovery operations.

Database platforms allow the default systems administrator account to delegate permissions to other users, allowing them to perform these important operations.

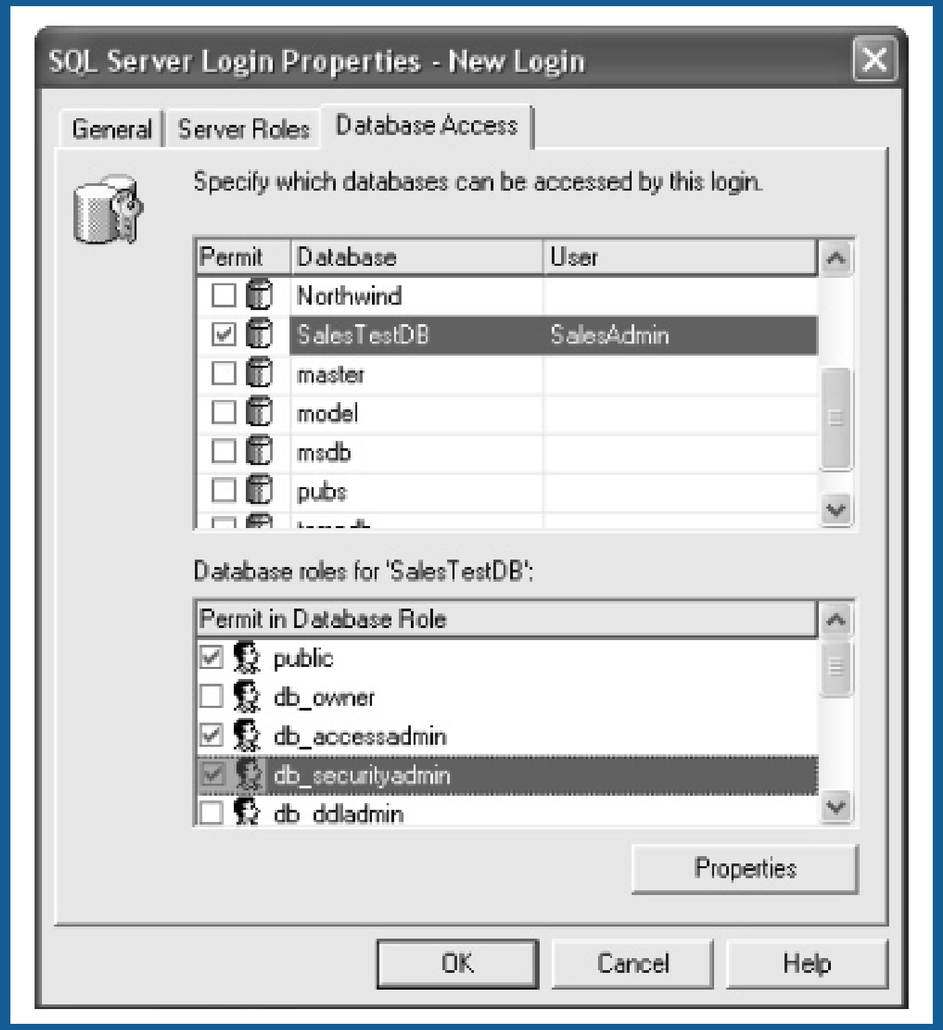


GRANTING DATABASE ADMINISTRATION PERMISSIONS TO A USER ACCOUNT

GRANTING DATABASE ADMINISTRATION PERMISSIONS TO A USER ACCOUNT

Microsoft’s SQL Server platform provides built-in server-level roles, including Database Creators, Disk Administrators, Server Administrators, Security Administrators, and many others.

The majority of database users will not require server-level permissions. Instead, they’ll need permissions that are assigned at the level of the database.



DATABASE ROLES AND PERMISSIONS

Having a valid server login only allows a user the permission to connect to a server. In order to actually access a database, the user's login must be authorized to use

The general process begins with specifying to which database(s) a login may connect. Then, permissions must be assigned within the database. The details here do vary between types of relational database platforms, but the overall concepts are the same.

DATABASE ROLES AND PERMISSIONS

Generally,database administrators will create “groups” or “roles,” and each of these will contain users. Specific permissions are assigned to the roles.

This process is quite similar to the best practices that are suggested for most modern network operating systems.

Additionally, some relational database platforms allow groups to be nested, thereby allowing you to create a hierarchy of permissions.

DATABASE ROLES AND PERMISSIONS

Through the use of roles, database administrators can easily control which users have which permissions.

Note, however, that it is very important to properly design security based on the needs of database users.

Again, the principal of providing the least required permissions should be kept in mind.

OBJECT-LEVEL SECURITY

Relational databases support many different types of objects. Tables are the fundamental unit of data storage. Each table is generally designed to refer to some type of entity (such as an Employee, a Customer, or an Order).

Columns within these tables store details about each of these items (FirstName or CustomerNumber are common examples).

OBJECT-LEVEL SECURITY

Permissions are granted to execute one or more of the most commonly used SQL commands. These commands are SELECT,INSERT, UPDATE,DELETE

The ANSI Standard SQL language provides for the ability to use three commands for administering permissions to tables and other database object
GRANT, REVOKE, DENY

OBJECT-LEVEL SECURITY

- **GRANT** Specifies that a particular user or role will have access to perform a specific action.

- **REVOKE** Removes any current permissions settings for the specified users or roles.

- **DENY** Prevents a user or role from performing a specific action.

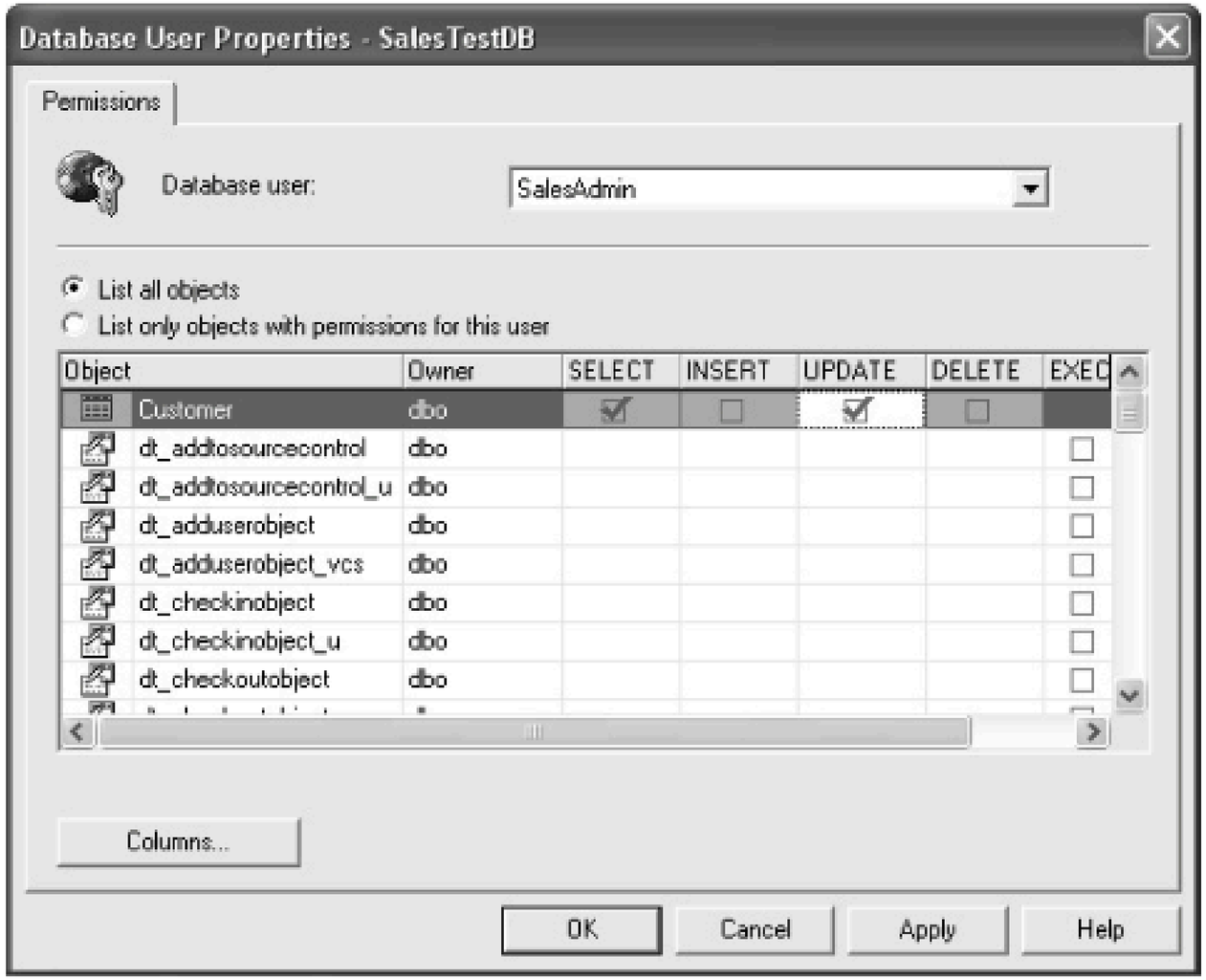
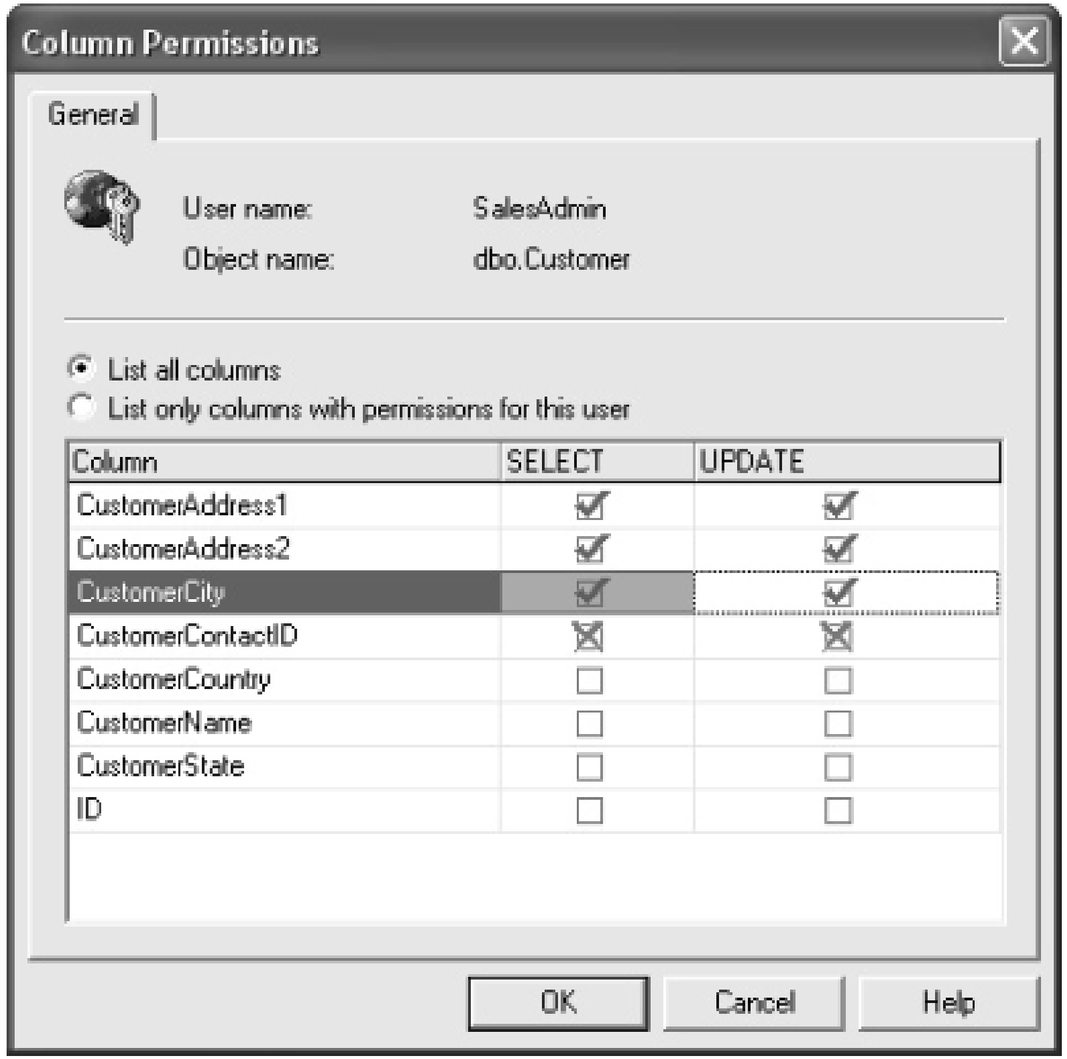


Figure 12-4 Setting object-level permission in a SQL Server database



Permissions can also be granted at a more granular level. In the case of specifying permissions on tables, database administrators can define permissions at the column level,

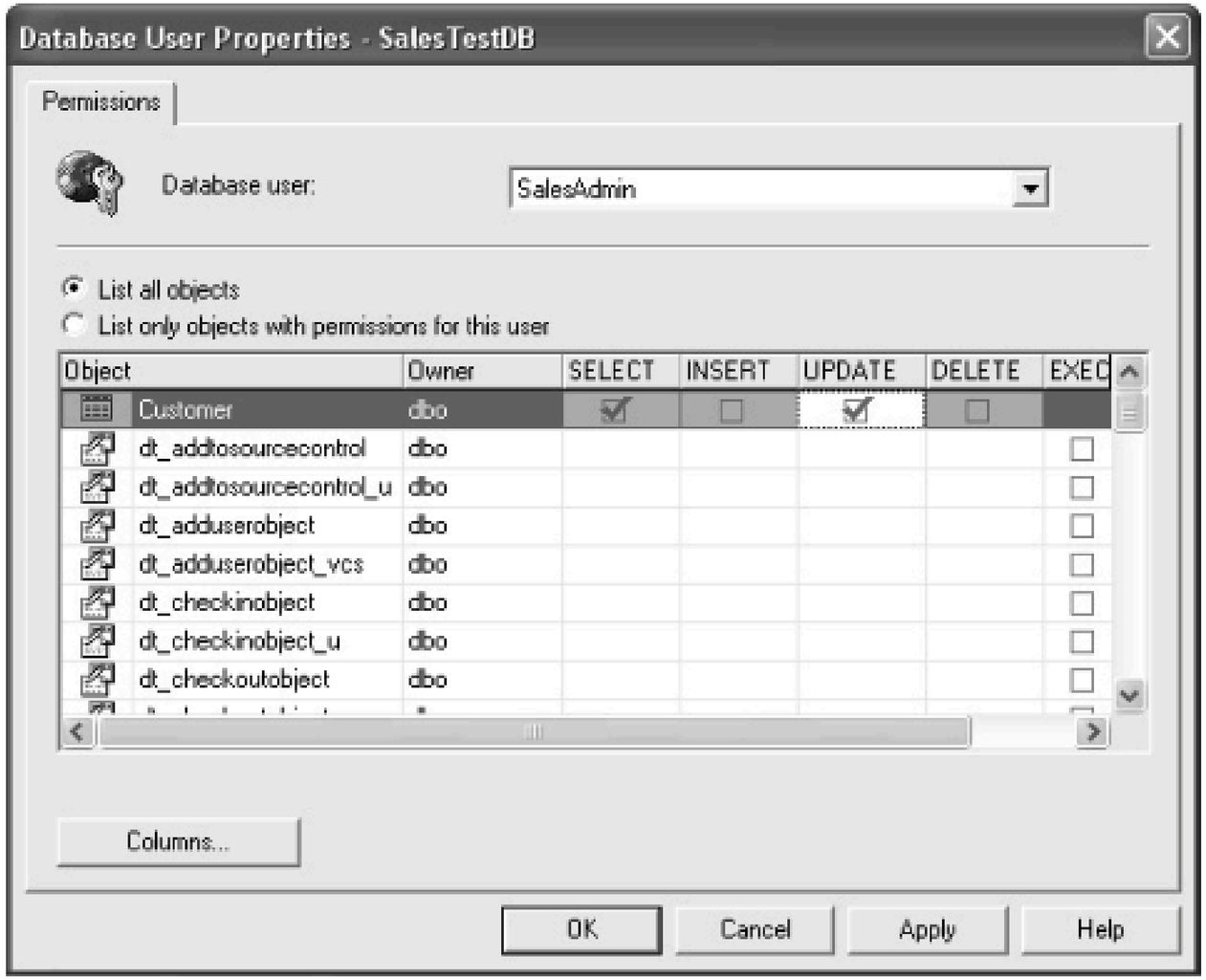


Figure 12-4 Setting object-level permission in a SQL Server database

USING OTHER DATABASE OBJECTS FOR SECURITY

The simplest usage of databases, you will store data in many tables. And, each of these tables might have millions of rows of data.

It doesn't take much imagination to see how this can lead to a lot of management effort.

Fortunately, relational databases offer many other types of objects that can be used to better manage data and control access to information.

The basic 3 database objects for security will be

VIEWS

A view is essentially a virtual table in a database that can be used to present specific data from one or more underlying tables, often with restricted access to certain columns or rows. Views are commonly used to enhance security by limiting access to sensitive data and simplifying the management of complex queries.

Views are an effective way to enhance security by allowing database administrators to control:

- Which columns or fields are accessible to specific users.
- Which rows are visible to users based on certain criteria (e.g., restricting access to data based on user roles, departments, etc.).
- Which complex operations can be performed (e.g., calculation or aggregation) while restricting direct access to the base data.

VIEWS

Furthermore, views can query other views, thereby creating a chain of objects based on business rules. When portions of the logic change, only some of the views may be affected. And, if business or technical requirements change, you can make corresponding changes in the view.

Views are generally used to return sets of data to users. Database developers can allow users to modify data through the use of views, but there are many important limitations to this method. That’s where another type of database object can be helpful.

STORED PROCEDURES FOR DATABASE OBJECTS SECURITY

Databases offer developers the ability to create and reuse SQL code through the use of objects called stored procedures.

They provide a layer of abstraction between the user and the underlying database structure, which can be particularly useful for enhancing database security.

```
-- Create a stored procedure named "GetCustomersByCountry"
CREATE PROCEDURE GetCustomersByCountry
    @Country VARCHAR(50)
AS
BEGIN
    SELECT CustomerName, ContactName
    FROM Customers
    WHERE Country = @Country;
END;

-- Execute the stored procedure with parameter "Sri lanka"
EXEC GetCustomersByCountry @Country = 'Sri lanka';
```

HOW STORED PROCEDURE IMPROVES DATABASE SECURITY

- Restrict Direct Data Access and limits permission
- Prevention from SQL injection
- Enhance Data Integrity
- Reduce Risk of Human Error
- Improved Audit and Logging
- Reduced Network Traffic
- Data Masking

TRIGGERS

Triggers are designed to automatically be “fired” whenever specification actions take place within a database. For example, you might create a trigger on the Sales Order table that will automatically create a corresponding row in the Invoice table.

A trigger is a database object that automatically executes a predefined action in response to certain events (such as INSERT, UPDATE, or DELETE) on a table or view.

SECURITY BENEFITS OF USING TRIGGER

- Enforcing Data Integrity
- Tracking Data Modifications
- Preventing Unauthorized Access or Changes
- Preventing SQL Injection Attacks
- Preventing Cascading Deletes and Updates
- Data Encryption and Masking

DATABASE BACKUP AND RECOVERY

Overall database security strategy should be providing backup and recovery.

In the event that data is incorrectly modified or destroyed altogether, the only real method to recover information is from backups.

Most often, it seems that systems administrators perform backups to protect information in the case of server hardware failures.

Since all relational database systems provide some method for performing database backups while a server is still running, there isn't much of an excuse for not implementing backups.

Data can be lost due to accidental human errors, flawed application logic, defects in the database or operating system platform, and, of course, malicious users who are able to circumvent security measures.

There are many constraints in the real world that can affect the implementation of backup processes.

DATABASE BACKUP AND RECOVERY

First, resources such as storage space, network bandwidth, processing time, and local disk I/O bandwidth are almost always limited.

Additionally, human resources—especially knowledgeable and experienced database administrators—may be difficult to find.

And, performance requirements, user load, and other factors can prevent you from taking all the time you need to implement an ideal backup solution.

Database Recovery Techniques

DATABASE BACK UP AND RECOVERY

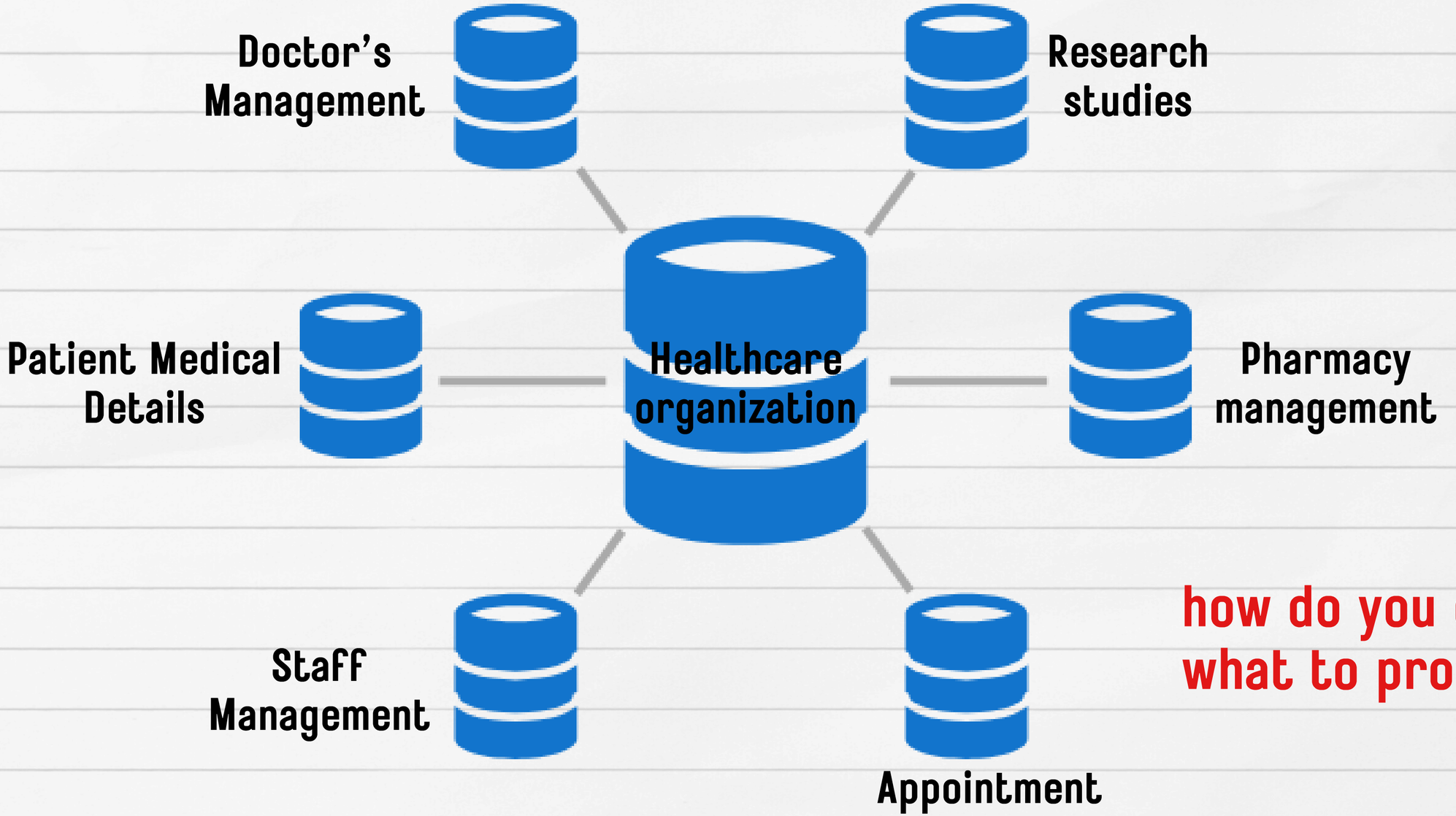
sales database

- → Customer Purchase
- → Inventory level
- → Financial Transaction

The company's sales database is mission-critical because it directly impacts the business's daily operations, revenue, and decision-making.

If this data is lost or corrupted, it could cause severe financial losses, reputational damage, and business disruptions. For this reason, the sales database should have high-level protection like regular backups, encryption, and strict access controls.

DATABASE BACKUP- CONSTRAINTS



how do you decide what to protect?

DATABASE BACKUP – CATEGORIZATION

Resource	Importance	Notes
OLTP server	Critical	Information can't be easily re-created, and data loss will lead to inaccurate or misleading reports.
E-mail server	High	Recovering lost messages and user mailboxes is very difficult.
Decision-support server (data warehouse)	Medium	Information can be regenerated from other sources.
Intranet web server	Medium	Content is important, but is replicated among multiple machines as part of development processes.

Table 12-1 A Sample Categorization of Data Based on Importance

DATA PROTECTION WORKSHEET REQUIREMENT

Business Requirement	Description	Data Category	Data Sensitivity	Protection Requirements	Compliance Requirements	Retention Period	Access Control Requirements	Backup Frequency	Encryption Requirements	Incident Response Plan
Patient Data Protection	Protect all patient data (medical records, personal details, etc.) due to sensitivity and legal requirements	Medical records, patient contact details	High (Sensitive personal and health data)	Data must be encrypted, access should be restricted to authorized personnel only	HIPAA (Health Insurance Portability and Accountability Act), GDPR (General Data Protection Regulation)	Data should be retained for 7 years after treatment	Access control based on roles (e.g., only doctors and authorized staff)	Daily backups, real-time replication for critical data	Data must be encrypted at rest and in transit using strong encryption standards (e.g., AES-256)	Incident response plan should be in place for data breaches, including notification procedures as per HIPAA

Determining Recovery Requirements

It’s important to keep in mind that the purpose of data protection is not to create backups. The real purpose is to provide the ability to recover information, in case it is lost. To that end, a good practice is to begin designing a backup solution based on your recovery requirements

You should take into account the cost of downtime, the value of the data, and the amount of acceptable data loss in a worst-case scenario.

“If we lose data due to failure or corruption, how long will it take to get it back?” In some cases, the answer will be based on the technical limitations of the hardware you select.

Determining Recovery Requirements

A company manages online transactions for an e-commerce platform. We need a back up of 10GB of transaction data every night to an external hard drive.

After a server crash, you need to recover the transaction database, but the recovery time takes about three hours, causing a significant delay in processing orders and customer service.

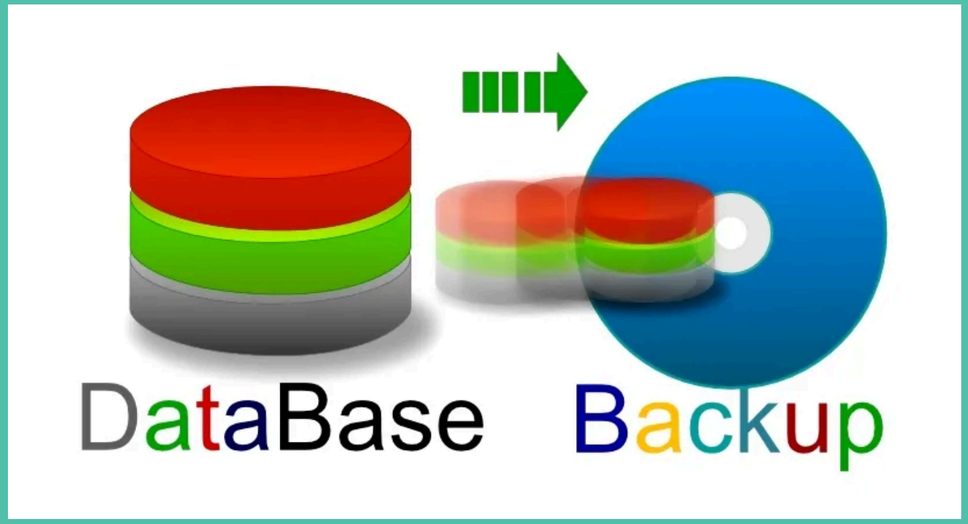
Suppose your company has a business requirement: the platform must be up and running within two hours after any data loss, to avoid affecting sales and customer trust.

Determining Recovery Requirements

Solution could be upgrading to faster backup systems, such as a cloud-based backup solution with faster restore capabilities or RAID(Redundant Array of Independent Disks) configurations that allow for quicker data recovery. Additionally, you could consider implementing real-time replication to a secondary server, reducing recovery time significantly.

TYPES OF DATABASE BACKUPS

Large databases and performance requirements can often constrain the operations that can be performed and when they can be performed.



Therefore, you'll need to make some compromises. For example, instead of backing up all of your data hourly, you might have to resort to doing full backups once per week and smaller backups on other days.

TYPES OF DATABASE BACKUPS

FULL BACKUPS

- This type of backup consists of making a complete copy of all of the data in a database.
- Generally, the process can be performed while a database is up and running.
- On modern hardware, the performance impact of full backups may be almost negligible.
- Of course, it’s recommended that database administrators test the performance impact of backups before implementing an overall schedule.
- Full backups are the basis for all other types of backups. If disk space constraints allow it, it is recommended to perform full backups frequently.

TYPES OF DATABASE BACKUPS

DIFFERENTIAL BACKUPS

- This type of backup consists of copying all of the data that has changed since the last full backup.

- Since differential backups contain only changes, the recovery process involves first restoring the latest full backup and then restoring the latest differential backup.

- Although the recovery process involves more steps (and is more time-consuming), the use of differential backups can greatly reduce the amount of disk storage space and backup time required to protect large databases.

TYPES OF DATABASE BACKUPS

TRANSACTION LOG BACKUPS

- Relational database systems are designed to support multiple concurrent updates to data.
- In order to manage contention and to ensure that all users see data that is consistent to a specific point in time, data modifications are first written to a transaction log file.
- Periodically, the transactions that have been logged are then committed to the actual database.
- Database administrators can choose to perform transaction log backups fairly frequently, since they only contain information about transactions that have occurred since the last backup.

TRANSACTION LOG BACKUPS

The major drawback to implementing transaction log backups is that, in order to recover a database, the last full (or differential) backup must be restored.

Then, the unbroken chain of sequential transaction log files must be applied. Depending on the frequency of full backups, this might take a significant amount of time.

One extremely important feature that other backup types do not: point-in-time recovery, provided that backups have been implemented properly, database administrators can roll a database back to a specific point in time

The various backup types that are available can be combined in order to provide flexible methods of backing up large or very busy databases. For example, you might choose to implement weekly full backups, daily differential backups, and hourly transaction log backups.

TRANSACTION LOG BACKUPS

Modern relational database systems allow database administrators to make backups of specific tables or portions of a database.

For example, Microsoft's SQL Server platform allows database administrators to create tables on specific physical data files.

These files can then be backed up and restored individually. Although using this method takes a lot of planning (for both backup and recovery operations), it can reduce backup times and provide for greater data protection on large, busy servers.

TRANSACTION LOG BACKUPS

- Another important consideration related to backups is where to store the database dumps that are created.

- The two main options are disk and tape. Both are commonly used solutions and have various pros and cons.

- Based on cost considerations, data volume, and performance requirements, you can choose to implement one or both of these solutions.

KEEPING YOUR SERVERS UP TO DATE

An important general security best practice that also applies to databases is keeping systems up to date. In order to ensure that known vulnerabilities and server problems are repaired.

You must apply the latest security and application patches. It's especially difficult to keep active databases up to date, since downtime, testing, and potential performance degradation can be real concerns.

However, you should always review available updates and find out if the servers you manage have problems that are potentially solved by an update. If so, plan to install the updates as soon as you can test and deploy them.

DATABASE AUDITING AND MONITORING

The process of auditing involves keeping a log of data modifications and permissions usage. Often, users that are attempting to overstep their security permissions (or users that are unauthorized altogether) can be detected and dealt with before significant damage is done.

once data has been tampered with, auditing can provide details about the extent of loss or data changes.

There's another benefit to implementing auditing: when users know that certain actions are being tracked, they might be less likely to attempt to snoop around your databases. Thus, this technique can serve as a deterrent. Unfortunately, in many environments, auditing is overlooked.

REVIEWING OF AUDIT DATA MUST

DATABASE MONITORING

By establishing a performance and usage baseline, you will be able to quickly identify any potential misuse of the system.

For example, using the Windows Performance Toolkit (WPT) that is part of Microsoft's server-side operating systems, you can track many statistics related to database usage.

You can also configure alerts that can be used to notify you when performance or other statistics are "out of bounds," based on normal activity. All of these mechanisms can be helpful in monitoring the usage of your database systems.

THANK YOU VERY MUCH!