




**After reading this chapter and completing the exercises, you will be able to:**

- Define authentication and then implement with SQL Server, MySQL, and Oracle
- Define authorization and then implement with SQL Server, MySQL, and Oracle
- Manage users based on security best practices using SQL Server, MySQL, and Oracle
- Identify and apply password best practices using SQL Server, MySQL, and Oracle
- Define and create roles using SQL Server, MySQL, and Oracle
- Define, grant, deny, and revoke privileges using SQL Server, MySQL, and Oracle



## Security In Your World

Tony, the database administrator for Haphazard, Inc., was desperate to redeem himself for the major loss that he had caused during the poorly executed MySQL deployment earlier this year. Hoping to regain trust and prove to his superiors that he was a security-conscious database administrator, he decided to create the most secure database environment possible.

Working many hours of overtime through the evenings and weekends, he mapped out each employee's database usage and created a plan that implemented a high level of granularity for object privileges and access control. Feeling confident that a layered approach to security was the best strategy, he opted against Windows authentication and created separate login accounts for users to access the database. Roles were developed based on privileges and the appropriate users were assigned to these roles.

Just as the final touches were being placed on the database, he received a message from his managers requesting a meeting. Excited to present his new security strategy to his managers, he eagerly accepted. Much to Tony's surprise, the meeting was set up to discuss his "early" retirement from Haphazard, Inc. Management felt that he hadn't displayed any progress with the new database and, in fact, they had been receiving a number of complaints from users regarding limited database access and excessive required logins.

To Tony's dismay, management informed him that he had been replaced. Infuriated with this decision, Tony returned to his desk to pack his things. Along with his boxes of desk toys, Tony took with him a few of the other employees' login credentials. Having opted out of Windows Authentication mode, Tony knew that his database account was separate from his Windows account and that even the most experienced database administrator (DBA) would have a difficult time removing it from the database. This meant that he only needed to acquire basic user-level Active Directory credentials to access the company's Windows system from his home computer. With this access, he would be able to log in to his own privileged database account (providing that it wasn't removed right away), giving him full reign over the Haphazard database once again. As he walked out of the building with his belongings in hand, he smiled, knowing that he would get the last laugh after all.

An endless number of disastrous scenarios come to mind when considering the implications of even one small mistake within user account management. User accounts, and the privileges assigned to them, should be carefully planned and monitored throughout any organization.

*(continued)*



As was discussed in previous chapters, databases often contain business-critical data, such as credit card numbers, customer account information, and company financial statistics. A loss or theft of this data can be devastating to any organization.

Thus far, we have examined steps to securely install a database and have explored ways to protect data traveling to and from the environment. The next, and arguably most important, step toward creating a safe and reliable database environment is to avoid unauthorized access. This chapter explores the steps that administrators can take to ensure proper authentication and authorization of those users attempting to access a database.

## 6

## Authentication

It takes a great deal of time and effort to ensure that access to a database remains secure. Several layers of security must be put into place to ensure that data is only granted to those authorized individuals and applications. There are two main steps to controlling access to data: authentication and authorization. Authorization is discussed later in this chapter. **Authentication** is the process of confirming the identity of those individuals or applications that request access to a secure environment. Confirming the identity of a person or an application is done by verifying that the login and credentials match those that have been created within that same environment.

A **login** is an object that is mapped to a user account within each database and is associated to users by the security identifier or SID. Logins differ from user accounts in that a login is required for authentication into the environment and a user account is used to control activities performed within that environment. This is a very specific distinction because there are default logins created during the installation of a database that must be managed correctly. Consequently, there are default user accounts that must be managed correctly as well. User accounts are discussed later in this chapter.

A **credential** is a piece of information that is used to verify identity, such as a person's username and password, an application's secure ID, or a host's network name and address. The types of credentials that are used to verify the identity of a user or an application depend on the requirements and authentication processes of a particular system or environment. Although usernames and passwords might be sufficient for some environments, additional requirements, such as host addresses and security identification numbers, might be required for others.

Authentication can be verified a few times and at different levels during a single attempt at logging in to a system before permission is granted to a user. For example, SQL Server checks the authentication of a user at the server level to establish a connection to the server and at the database level to establish access to the database. Third-party applications can also be used within a database environment to further add security to the authentication of users because they often take additional steps, such as password encryption, to keep a network environment secure. This section reviews the three most common levels of authentication found in a database environment: the operating system level, the database level, and third-party support. Combined, these levels make for a secure environment, but using one alone can bring great advantages and disadvantages to the security of the environment.

## Operating System Authentication

Credentials that are authenticated primarily through the operating system must have an account residing on that particular operating system and the operating system account credentials must be used to access the system. In some cases, an operating system login alone can be used to authenticate users to the database. This means that if the user has an operating system account residing on the system and these credentials are used, no other logins are necessary to access the database. Authentication through the operating system allows users to conveniently connect to the database without specifying a database username or password, skipping database login prompts altogether. This type of authentication also provides DBAs with the advantage of centralized account administration because all accounts are located in one place and each individual requires only one set of credentials to be managed.



With operating system authentication, workstations and user accounts must be monitored much more diligently. If an unauthorized individual obtains user credentials for accessing the operating system or gains access to an unattended workstation, they inherit access to the database as well. It does not take much knowledge or skill to access the database from the operating system.

## Database Authentication

Credentials that are checked against the database require that the user attempting to access the system has a local database account therein. In this situation, the user may be required to access a few different systems prior to reaching the database. This means that users must keep different account credentials for different systems. This often leads to poor password practices, such as writing passwords down and choosing weak passwords, for memory's sake. Administration of this type of environment is also more difficult. Administrators must not only keep track of more than one account for each individual, but often these accounts are located in separate areas, making audit tracking a very difficult feat. On the other hand, separate accounts for separate systems can create a more segmented and secure environment. Those intruders who gain unauthorized access to a user's operating system password or machine will also need to obtain the database credentials to access it.

## Network or Third-Party Authentication

Authentication for a database can also be conducted using third-party applications and at the network level of the environment. Third-party applications and network account authentication systems can be used for remote and physical environments. These users are not required to have an account created on the operating system or the database; however, they are required to have a network account or be recognized by the third-party application. For example, a type of external authentication is a smart card. Smart cards require the user to insert a PIN for authentication; this PIN is unrelated to the OS and the database. Secure protocols such as Kerberos are often used for network and third-party authentication as well. These protocols require greater technical experience and know-how, yet they add a much-needed layer of security to any database environment.

Kerberos is an authentication protocol that was built by MIT to provide secure means for authentication using symmetric-key cryptology to verify the identity of a client to a server and a server to a client. Once a client and server have used Kerberos to prove their identity, they can also encrypt all of their communications to ensure privacy throughout the transaction.



The Kerberos process begins when a user or application attempts to log in to an environment using an application or a Web form using the Kerberos protocol. When a user attempts to access the environment, a request for a ticket to get a ticket, also known as a ticket-granting ticket (TGT) is sent to a server known as the Key Distribution Center (KDC). The KDC sends the user back an encrypted response with the original password hash and the TGT embedded within a message. If the message is sent back to the KDC by the user decrypted and with the original TGT, then a service called the Ticket-granting service (TGS) located within the KDC assigns the user a service ticket. A **service ticket** is a unique key that is used to validate a person's identification (similar to a driver's license or smart card ID), for the purpose of gaining access into a secured environment. The assigned service ticket is sent to the network or database the user wants to access, and if appropriately verified, the user obtains access. It is important to note that it is only necessary for the user to log in once when using Kerberos; each resource that is requested after the original login will check the requestor's ticket as a way to authenticate that user.

Now, let's review this process again, this time using a simple example that will help you better understand what happens throughout the Kerberos process.

Let's assume that there is a secret group of individuals known as the World Bank Database Society (WBDS). Members of this society have control over the world's banking system, so obtaining access to the WBDS would provide an individual with the tools needed to attain wealth or financial security.

To acquire access to the WBDS, an applicant must first be given the secret password (a service ticket), which they can only obtain through a trusted source (the KDC), or a partner of the WBDS. So, in order to receive the secret password and subsequently use it to request membership into the WBDS, the applicant must know where to find a trusted source (KDC), obtain access to the trusted source (authentication), and request the password information needed for requesting membership into the society (TGT).

Once deemed fit for membership (authenticated through the KDC), the trusted source gives the person a photo ID that includes the encrypted version of the password, the location of the secret society, and the applicant's identification information (service ticket). The WBDS can't send the applicant the real password because it would jeopardize its secrecy, so the applicant uses a photo ID (service ticket) as a way to locate the secret society and subsequently request membership. If the society finds that an applicant has been approved by the trusted source and that the photo ID contains the correct password, decrypted, then the applicant is offered membership or access into the World Bank Database.

Network authentication can also use other security protocols to validate a person's identity. Another common security technique is known as public key infrastructure (PKI). PKI utilizes encrypted keys similar to the way in which Kerberos issues TGTs, yet it verifies a requester's identity by assigning a digital certificate to a user for authentication into secure environments. A **digital certificate** is a password-protected and encrypted file that holds the identity of a user or object. A digital certificate is essentially a digital document that is issued by a trusted central authority. Individuals and companies can obtain digital certificates for a fee through a trusted authority such as VeriSign. A digital certificate can also be assigned after an identity verification process similar to that of public key infrastructure, or PKI. It is important that network or third-party external authentication is carefully considered and that the technology used for authentication is researched thoroughly prior to its implementation because although

network authentication uses sophisticated cryptology to authenticate users, it poses a greater risk to a network due to its exposure to the public network.

Third-party or external authentication is not recommended to be used alone, yet it can be combined with OS and server authentication to create an ideal security scenario for any environment.

## Database Vendor–Specific Authentication Components

There are a few important differences in the way that authentication is handled between different database vendors. This section explores the differences regarding authentication modes between SQL Server, Oracle, and MySQL.

**SQL Server Authentication Information** SQL Server supports two types of authentication modes, Windows Authentication mode and Mixed Mode Authentication. **Windows Authentication mode** is a form of authentication that only allows Windows authentication to be used for accessing the database; users logging in to the database must have a Windows login to access it. Windows authentication is also known as trusted authentication because of the security enforced through Windows. This is the default mode during installation and it is the recommended authentication mode for SQL Server. It is much more secure than the alternative. It uses the Kerberos protocol and requires strong passwords and password expiration (discussed later in the chapter).

Once a user is able to log in using his or her Windows account, user verification is confirmed by Windows and no further credentials are needed to access the database server. SQL Server authentication is disabled when Windows authentication is chosen. **Mixed Mode Authentication** is a form of authentication that allows both Windows authentication and SQL Server authentication to be used to obtain access into the database; therefore, the database will accept both Windows and Server logins. Yet, to access a SQL server using Mixed Mode Authentication, a user must supply two separate sets of credentials. As mentioned previously, this multiaccount approach poses a difficulty for administration and security of database users. This mode is also known as an “untrusted” connection because it is not as secure as Windows Authentication mode and because protocols such as Kerberos cannot be used. Mixed Mode Authentication is most appropriate for environments with older operating systems and mixed operating system environments.

**MySQL Authentication Information** MySQL uses an authentication protocol for access to the server and it identifies users slightly differently than SQL Server and Oracle do. A MySQL user’s identity is verified using three pieces of information:

- The host name for which the server is running
- The user-supplied MySQL username
- The user-supplied password

To access the database, the identity credentials must match those credentials stored in the database. The username is separate from login names that are managed by the operating system, and management of a username is separate from a login name. There is no way to synchronize this management, so they remain separate and distinct from one another. The password is also separate; there is no relationship between the MySQL password and the one used for the operating system. Host names can be given as an IP address, and the host



name does not have to be provided if the host that is trying to authenticate is on the same machine as the server. These three components are stored in three user table scope columns (host, user, and password). The connection is only permitted if the host and user columns in some user table row match the client host name and username and the client supplies the password that is specified in that row.

**Oracle Authentication Information** Oracle supports many options for authenticating users, applications, and machines, and it provides customization options to support almost any environment. Database servers, database links, and environment passwords can all be used as credentials for authentication within Oracle 11g, and several additional applications are available to be purchased to further enhance the security of the database infrastructure.

One of the more notable services that can be purchased and added to Oracle as a way to further enhance authentication is *Advanced Security*. Advanced Security is a comprehensive security application. It offers encryption of both information transmitting across the network and stored within the database and provides strong and proxy authentication strategies that support and integrate with the industry-standard authentication methods (e.g., Kerberos, PKI, and SSL).

In addition to the applications that can be added onto the environment, Oracle offers a number of applications that are built to mediate the communication between the database environment and the external Web resource. These are known as middleware applications. Middleware applications are designed to monitor external requests that are sent to obtain access to the database, and the database environment's responses to these requests. Middleware applications that offer strong authentication and authorization include: Server Certificate Authority, Oracle Identity Federation, and Oracle Operating System Authentication Services.

The Oracle database management system itself includes many strategies that help ensure proper authentication of the database environment as well. Database linking is an example of one feature that enhances authentication support. Database links can be created in Oracle, allowing users to access data from remote servers without additional credentials. A database link is a link made between two databases that, when created, results in one logical data storage unit. Links are created to apply common policies and to create associations between databases. Database links can be both private and public. Private linked databases can only be accessed by the user who created the link itself; public linked databases are available to all users who have access to the databases. Database links allow two physically separate databases to be logically combined for data maintenance, storage, and retrieval. When a public database link is made between two physically separate databases, users view and access them together as one logical data storage unit.

Two authentication methods are supported between database links: *current user* and *connect to user*. A current user authentication method for a database link requires the current user's credentials (CURRENT\_USER), and all users must have an account in all of the linked databases to be accessed. If the database link is created with *connect to user* authentication, the user account must exist in the remote database only.

## Password Policies

Passwords are the key to opening a user account. Ask any experienced database administrator—he or she would not deny that most intrusions into a system originate from a cracked or stolen user password. An endless amount of money, time, and resources are wasted due to password

vulnerability. Password policy implementation should be the first defense for any organization desiring to lower the risk of compromised user passwords. Password policies can be enforced within the database server application as well as within the workplace's written user agreements. That is, policies can be defined within the database system, or they can be written as a formal agreement of acceptable computer use between management and employees. Database server password policies that are configured within the database are enforced throughout the database management system. They are much more effective than written policy, because users cannot violate them, yet written policies can result in negative consequences (e.g., demotions, suspensions) and raise awareness of the need for security. Therefore, both written and server-defined policies should be used within any database environment for maximum effectiveness.

### Database-Enforced Password Policies

Although written password policy agreements are effective, an organization should never rely solely on the confidence that users will follow written policies. Users are the weakest component within any security architecture. Mistakes happen and, far too often, users do not follow written policies as they should. As a security professional, it is best to remove user decisions and implement database server-enforced policies. Options that are available within a server for creating strong password policies are often vendor specific, yet most server applications share similar configuration settings. This section identifies the common password settings found within a database management environment. Database administrators and security professionals set the ground rules for employees defining acceptable use of equipment and technology within an organization. The following four different attributes of passwords can be enforced in almost every database server:

- *Complexity*—A policy can be created that identifies the required length and character type combination (e.g., number, letter, uppercase, lowercase, symbols) of a password. It also determines whether a user can use common or dictionary words.
- *Failed attempts*—A password that has been attempted too many times without avail can be an indication of an intruder. Therefore, it is best to lock an account that has had too many failed password attempts. The number of failed password attempts and the way in which these are handled can be identified as part of the password policy.
- *Expired passwords*—This component of a password policy specifies the length of time a user can use a password before being forced to change it. This is to minimize the damage and risk that can be done if a password is breached.
- *Password reuse*—The number of password changes that a user must make before being allowed to reuse a password is specified here.

### Written Password Policies

Written policies are often included in some type of equipment usage agreement between an organization and its employees. Even though the database management system automatically restricts users' abilities, database-enforced passwords from the previous section are included within the usage agreement as well. Too often, written policies are not consistently enforced within an organization because from a business standpoint, these policies are often not viewed as critical to meeting the business needs. For example, a common policy included within a usage agreement forbids users from sharing passwords. Imagine this scenario: The vice president of the organization is attending a meeting at a remote office and realizes that



she left a few files back on her office PC that are vital to the success of the meeting. The vice president calls her administrative assistant and asks him to log on to the computer to retrieve the important files. The vice president then provides the administrative assistant her username and password credentials to enable the assistant to obtain access to the PC to send the files. Although this is a clear violation of the policy, the vice president justifies the violation as being required and critical for overall success.

A usage agreement must be flexible enough to be consistently enforced throughout each level of an organization and yet strict enough to ensure that users abide by the policy.

Many different types of standards can be included within a written password policy, and these will vary depending on the organization, yet this section provides a list of common standards that are likely to be included in an equipment usage agreement between an IT group and its users:

- *Password discretion*—Usage agreements commonly include a policy that informs users not to tell their password to anyone in the organization. In some environments, users are even forbidden to tell their technical support group their password. In one organization, an intruder sent an e-mail to all employees claiming to be the technical support group. The e-mail directed users to reply back with a list of current passwords, explaining it as an ongoing effort toward increased quality control. Although some security-conscious users phoned the IT group to confirm the request, many provided a list without a second thought. This is one example of why users are instructed to keep their passwords private, despite who might be requesting them. Some organizations do not restrict users from communicating their password to the IT team, and instead use a variation of this policy by stating that users should never provide their password to IT through e-mail, through instant message, or by phone. This ensures that the user physically sees a person face-to-face before providing his or her password.
- *Password sharing*—Users must agree to never share their password with any other employee in the organization. It isn't uncommon for users to share their passwords with one another. This often occurs in cases where one employee has access to a resource that a coworker needs to use to complete a specific task. Using a coworker's access to the resource is often much faster and less bothersome than calling the IT group to have a new account created or access granted. Consider this scenario: Joan's supervisor is short on time and asks Joan to help her with her quarterly report by compiling specific data. Joan agrees, but quickly realizes that she does not have access to the database tables that she needs to gather some of the data. Knowing that her coworker Betty does have this access, she asks her for help. Both Betty and Joan agree that calling the IT group to obtain the appropriate access would be too cumbersome and take up too much time. Therefore, Betty walks to Joan's computer and logs in to the database using her own credentials so that Joan can gather the information that she needs.
- *Password storage*—Many equipment usage policies do not permit users to store a password on or around their desks and some demand that users never write passwords down. Although the justification for this is clear, it is important that policies be realistic in the demands that they place on users. The typical network environment user must keep track of an average of three to four different passwords (this number is drastically lowered if single sign-on strategies or Windows Authentication mode are implemented) within their organization. Users are asked not to write these passwords down. This is

not an unreasonable request alone, yet when combined with other password policies, this can be quite a great expectation. For example, if an employee has three passwords to remember and these three passwords change every 30 days, this can be difficult. Now imagine a scenario in which a nontechnical user is required to remember three different passwords that change every 30 days and that they are required to create passwords using unrecognizable words, symbols, numbers, and uppercase letters. In addition to this, that same user can only reuse a password every 10th password change, or every 300 days. This expectation is much higher! This is one case where security can add a great deal of complexity. If password storage policies are not realistic, then an organization can be setting up a system that is doomed to fail.

### Database Vendor–Specific Password Management

There are a few important differences in the way passwords are managed from one type of database to another. This section explores the differences in how passwords are managed between SQL Server, MySQL, and Oracle.

**SQL Server Password Policy** SQL Server can use the same password policy methods as available within Windows Server. Password complexity, password expiration, and enforcing password policy are the three password policy methods that are available. The following password complexity requirements are necessary for creating a new password in SQL Server:

- Passwords should be unique and cannot include common words, reserved words, or account usernames.
- Passwords should be between 8 and 30 characters, but can be as long as 128 characters.
- Passwords can include the underscore (\_), dollar sign (\$), and number sign (#) characters.
- At least one digit and one alphabetic character must be included within a password; however, a user cannot begin the password with a number.

Password policies can be reconfigured and customized for each user login using the ALTER LOGIN function.

**MySQL Password Policy** MySQL alone does not provide password enforcement, so for policy enforcement, MySQL database administrators must rely on the operating system and third-party applications for help. Passwords are stored in 45-bit encryption in the user table, allowing them to be verified and checked, but not reconstructed or viewed by outside resources. In other words, the existence of a password in the user table can be verified by an attacker and an attacker can obtain access to the user table, but there is no way for him to reconstruct and obtain the original text. MySQL passwords are case sensitive, can vary in length, and special characters can be used. One application created and built specifically for MySQL to aid in password enforcement is Securich, which is an open source security package. You can find Securich at [www.securich.com/](http://www.securich.com/).

**Oracle Password Policy** Oracle passwords are stored encrypted in the DBA\_USER table and they provide user authentication for a client/server environment. Oracle provides several built-in password protection services that go to great lengths to maintain the privacy of a password. Case sensitivity, password hashing, and password complexity checking are



only a few of the strategies used in securing passwords in Oracle. Oracle verifies the complexity of a password using a built-in program or script, (UTLPWDMG.SQL), and passwords that do not meet the default complexity criteria are not accepted. The following are the default complexity requirements verified for Oracle database accounts:

- Passwords should be unique and cannot include simple words, server names, account usernames, server names with numbers added, or account usernames with numbers appended.
- Passwords should be between 8 and 30 characters, but can be as long as 128 characters.
- A new password must differ from the previously used password by at least three letters.
- At least one digit and one alphabetic character must be included within a password, yet the password cannot begin with a number.
- Passwords can include an underscore (\_), a dollar sign (\$), and a number sign (#).
- Passwords beginning with a special character or passwords containing any character other than \_, \$, and # must be surrounded by quotation marks.

Password verification parameters can be customized by changing the PASSWORD\_VERIFY function in the UTLPWDMG.SQL script file. A **user profile**, or a set of rules that limits a user's access to database resources, can be used to set password restrictions as well. A user profile can be created using the CREATE\_PROFILE command. Table 6-1 contains Oracle's password-specific functions, along with their default installation settings.

Feature	Default setting	Comments
FAILED_LOGIN_ATTEMPTS	10	Number of allowable failed login attempts before the account is locked
PASSWORD_LIFE_TIME	180	Number of days that the password is valid
PASSWORD_REUSE_TIME	Unlimited	Number of days that must pass before a password can be reused
PASSWORD_REUSE_MAX	Unlimited	Number of times a password can be reused
PASSWORD_LOCK_TIME	1	Number of days an account is locked due to failed attempts
PASSWORD_GRACE_TIME	7	Number of days ahead of expiration that the user is warned

**Table 6-1** Oracle password-related functions

## Authorization

Once data is authenticated, or a system has verified the identity of a user or login, a set of predefined permissions determine which databases or database objects are able to be accessed. The process for which permissions are applied to a user is known as authorization. **Authorization** is the process of ensuring that those individuals or applications that request access to an environment or an object within that environment have the permission to do so. Basically, authentication verifies the identity, whereas authorization verifies the ability of that

identity. Authorization is determined prior to a user obtaining authentication credentials. In fact, early in the planning process of a database management system, an administrator analyzes the work environment and determines each user's appropriate access to the database. User management is critical and choosing the most appropriate privileges for each user in an organization helps maintain a healthy and secure database. This section reviews user management and logins, including the privileges and roles that they are assigned.

## User Account Management

Proper administration of users is important in controlling the security and access of the database. The most common activities that a database administrator will perform on a database involve user management. At the bare minimum, a DBA must know how to properly add, remove, and assign privileges to the users in his environment. Another important thing that any security-conscious DBA should know is the user accounts and privileges that are created by default during the installation of a the database management system. This section explores the concerns related to adding, removing, and assigning privileges within an environment as well as explores the default users and user privileges.

## Default User Accounts

In virtually every type of database, default user accounts are created with predefined user access. Most of these default users created during install are the system or administration accounts, holding access to just about anything in the database. Information about these accounts, such as default passwords and usernames, rights and privileges, and account accessibility, can be easily found conducting a simple search online. Therefore, leaving these accounts untouched can provide a way for intruders to access and have free reign over your data. This section identifies the default passwords created in SQL Server, MySQL, and Oracle.

**Default Users Installed with SQL Server** During the installation of SQL Server, three main default logins and user accounts are created and configured. Two administrator accounts, *SA* and *BUILT-IN\Administration*, and one general PUBLIC account, *Guest*, are created. As you may recall from earlier in this chapter, we discussed the difference between a login and a user account. As you read this section and explore the default logins for SQL server, keep in mind that a login is mapped to a user account and is used to authenticate, whereas user accounts are used to authorize.

- *SA* is the system administrator login and it holds great power on the database. *SA* uses SQL Server Authentication, so if Mixed Mode Authentication is chosen during the installation of SQL Server, a strong password must be set to complete the installation process. If Windows Authentication mode is chosen during setup, *SA* is created, but is disabled because it will not be needed.
- The *BUILT-IN\Administration* login is mapped to the user account *dbo*, the only actual user created with the database install. The *dbo* stands for database owner, which means that it holds ownership of the database's default schema. The database owner *Guest* is another specialized user of the database and, by default, is a member of the PUBLIC role (discussed later in this chapter). The *Guest* account is not needed and should be removed whenever possible.



**Default Users Installed with MySQL** During the installation of MySQL, four default accounts are created, two root accounts and two anonymous-user accounts. Initially, no passwords are set for any of these accounts and to remain secure, passwords should be assigned immediately during installation. The root accounts allow administration of the database, so it is especially important that this account is given a password right away.

When installing MySQL on Windows, one of the root accounts is used for local connections only, whereas the other is configured for remote access. Conversely, one anonymous account is for connections from the local host, with no global permissions, whereas the other has all privileges for any test databases.

**Default Users Installed with Oracle** The number and type of default accounts created during an Oracle installation can vary greatly, as they depend on the options, features, and additions that are chosen to be installed. Three different types of accounts can be created—administrative, nonadministrative, and sample user—and each of these types has several types of user accounts. Thankfully, most of these accounts are created to expire and be locked after installation. Only three accounts are created during installation and open for use automatically after installation: SYS, SYSMAN, and SYSTEM. The SYS account is similar to the sa account found in SQL Server; it permits administrative tasks to be performed within the database. The SYSMAN account also allows database administration, but specifically for Oracle Enterprise Manager; any administrative duties in relation to this feature are done using this account. SYSTEM is the default generic database administrator account for Oracle databases. SYS and SYSTEM are automatically granted the DBA role, but SYSTEM is the only account that should be used to create additional tables and views that are used by Oracle. SYS owns the base tables and base views for the database, yet tables should never be modified by the SYS user. Default passwords are created for these three accounts; however, they are generic passwords that can be found online, so it is extremely important that you change these account passwords.

6

## Adding and Removing Users

Adding users to a database is a fairly straightforward process if the database administrator has planned appropriately and is prepared with user privileges and accessibility predefined. During the creation of a user account, security and access rights are also applied. Without proper documentation and careful consideration, a user account can expose the database to many types of security risks and violations. Always change the default password of a new user or force the change of a password prior to server entry. As a best practice, save user passwords in an encrypted file and enforce strong password policies to keep the database safe. Whenever possible, use different logins and passwords for different applications. This helps to minimize a user's access to unnecessary information and limits an intruder as well. Before providing access to the database, ensure that a user receives, reads, and agrees to the organization's policies on appropriate database usage. Policies hold great power in the security of a network.

Removing a user can be a bit more complicated. When user accounts are removed from a system, so are all of the objects for which that specific user had ownership; depending on the user's role in the company, this can be quite a disaster. Before removing any user from a database, perform a careful inventory of the user's created objects and back up the user account. It is strongly recommended, whenever possible, to disable a user account—instead of deleting it—and always document removals of database user accounts.

The most important component when adding and deleting accounts is documentation. Organizational policies should exist that include standardized steps for adding and removing users and all changes or additions to user accounts should be documented thoroughly.

## User Privileges

The smallest unit of authorization is a privilege. A **privilege** is the ability to access a specific database resource or to perform a specific action within a database. Examples of privileges include deleting a row, creating a table, or executing a procedure. Users can either be given or denied privileges, or privileges can be grouped together creating a specific role for which users can be a member. Privileges should be planned out well in advance and early in the planning stages of the database. A critical review of the database environment's user needs is necessary to properly assign privileges. Privileges should be carefully considered and follow the principle of least privilege. The **principle of least privilege** is a security standard by which each user added to a system is given the minimum set of privileges that he or she requires to conduct legitimate business within that system. User privileges can be managed by granting, denying, or revoking them within a database:

- *Granting a privilege*—The act of providing a user with permission to access or perform an action within a database
- *Denying a privilege*—The act of explicitly preventing a user from accessing or performing an action within a database
- *Revoking a privilege*—The act of taking away or reversing a previously granted or denied privilege

Privileges can be managed at different levels of the environment. The granularity at which privileges can be managed depends on the database vendor. Privileges should be allocated to individuals on a need-to-use basis only to protect the data integrity of a system. There are two ways to grant a privilege:

- *Fixed*—Fixed privileges are predefined by the server. They are often grouped together as one group to which users are assigned.
- *Single statement*—Single statement privileges are assigned individually to individual specific users of the database.

Which one of these techniques is used is dependent on the need of the database. As you will see in the next section, grouping together privileges can be very handy and efficient when assigning permissions on a system. Such grouping saves a great deal of time and resources.

**Assigning Privileges in SQL Server** Three levels of permissions can be granted within SQL Server: server-level, database-level, and object-level. Server-level privileges allow users access to certain tasks at the server level. These are fixed, which means that they are predefined and grouped by the SQL server to fit the needs of specific user accounts. Fixed, server-level privilege groups cannot be altered or removed and are only meant for specific accounts that exist on the server; it is important that administrators become familiar with these privileges to avoid redundancy within the server. For example, sysadmin is a default account that is created in SQL Server during installation. This account is assigned a fixed group of privileges meant to meet the needs of the system administrator. Changes to the sysadmin account cannot be altered. Database-level privileges are granted in one of two ways, fixed and single statement. Privileges granted at the database level can be fixed, and



similar to fixed server privileges, they are grouped together and predefined by SQL Server to fit the needs of a specific user account. Privileges can also be single statements, which are granted manually to specific users. Object-level permissions are granted using the GRANT, REVOKE, and DENY statements. You can grant object permissions to individual users or roles (roles are discussed in the next section).

To grant a permission:

```
GRANT privileges ON object TO user
```

To grant single statement permission using Enterprise Manager:

1. Open the Enterprise Manager interface.
2. Choose the plus sign to expand the server that contains the database in which you want to grant the single statement permissions.
3. Open the properties box for the database.
4. Click the Permissions tab.
5. Place a check mark in the permission that you would like to grant.
6. Click OK.

6

**Assigning Privileges in MySQL** MySQL uses tables and columns for storing and organizing privileges. Columns that hold privileges always end with the `_priv` name; tables in which these columns can be found vary, and are dependant on the level of privilege. Five levels of privileges can be granted within MySQL:

- Global privileges apply to all databases and are stored in the `mysql.user` table. Global privileges should be avoided when possible, as one mistake can cause a great security breach on the database.

To grant global privileges:

```
GRANT privilege_list ON *.* *
```

- Database privileges apply to all objects of a specific database. These are stored in the `mysql.db` and `mysql.host` tables.

To grant database privileges to all users for all tables on a specific database:

```
GRANT ALL ON db_name. *
```

- Table object privileges apply to all columns in a given table. These are stored in the `mysql.tables_priv` table.

To grant table privileges for all columns on a specific table of a database:

```
GRANT ALL ON db_name.table_name
```

- Column object privileges apply to one or more columns in a given table of a specific database. These are stored in the `mysql.columns_priv` table.

To grant SELECT privileges on two columns (`col1` and `col2`) to a user on the table located in the database `user_db`:

```
GRANT SELECT (col1.col2) ON user_db.table_name  
TO 'user'@'hostaddress';
```

- Routine privileges apply to stored routines (functions and procedures), such as CREATE ROUTINE, ALTER ROUTINE, EXECUTE, and GRANT. These can be granted at the global and database levels. These are stored in the mysql.procs\_priv table.



Where global privileges provide access for all objects of a database, object privileges are more specific to certain database objects. Object privileges are more time and resource intensive, but they are more secure.

For MySQL, the GRANT command is used to provide access to a privilege; it will also create a new user at the same time you are giving that user privileges. For example, if a statement is created to GRANT a nonexistent user a privilege, that user will be created and the privilege will be granted at the same time. The ability to grant privileges is limited to those that are already owned by the person trying to grant privileges. In other words, no user can grant a privilege that he or she does not already hold.

MySQL uses tables to hold various administrative tasks and privileges. These tables are called grant tables. Table 6-2 displays an overview of these tables as well as their privileges.

Table name	Privilege
user	Contains global privileges and specifies which users can access MySQL Server and from what servers they can access it
db	Specifies which users can access the MySQL database
host	For those not listed in db, provides information on which host names can access the database
tables_priv	Identifies which users can access which tables in a database
column_priv	Identifies which users can access which columns of a table
Procs_priv	Identifies which users are permitted to execute individual stored procedures

**Table 6-2** Grant tables for privilege administration

**Assigning Privileges in Oracle** Two levels of privileges can be granted in Oracle: system-level and object-level. Only administrators or those who have been given permission by an administrator can grant system privileges to others, whereas object privileges can be granted to a user by the schema owner of that object. A user account created within Oracle holds no privileges, not even enough to connect to the database. Privileges are granted based on need, which is the recommended approach for all database design. Privileges can also be granted to PUBLIC. A privilege granted to PUBLIC grants the privilege to all database users on the database. It is recommended never to use PUBLIC when assigning privileges; granular approaches are the most secure and reliable. One wrong PUBLIC privilege can cause a great security breach on the database.

To grant permissions in Oracle using PL/SQL:

```
grant privileges on object to user;
```



To grant permissions in Oracle using the Enterprise Manager security tool:

1. Open Enterprise Manager and select the database.
2. Click the Users link and select the user for whom you want to grant permissions.
3. Click the Object Privileges link and choose the privileges that you want to grant.
4. Click OK.

## Roles

Related privileges can be combined to create a role, used to centrally manage a group of objects or users within a database. A role is a set of related privileges that are combined to provide a centralized unit from which to manage similar users or objects of a database. Consider an enterprise telemarketing company, CallsRus, that recently hired 2000 salespeople in their sales department to fill the position of *cold call salesperson*. Based on the job duties of the *cold call salesperson*, the DBA determines that he will need to grant these individuals privileges to read and update tables for when they speak to customers, and he will need to deny them privileges to delete tables so that they cannot delete customers from the company database. The DBA is aware that individually assigning these privileges would take quite a bit of time and resources and that if individually assigned, any future management of these privileges will be very cumbersome. So, the DBA creates a role named `CLDCLL_SALES_ROLE` that includes the set of privileges that meets the needs of the salespeople and then assigns each salesperson as a member of this role. This is less time and resource intensive and ensures that any future adjustments to the database requirements of the salespeople only require an adjustment of the `CLDCLL_SALES_ROLE`.

Roles can be created for users, objects, and applications alike and they offer many advantages to database administration by saving time and resources and by providing a central location for administration. A user can be assigned many roles and a single role can be assigned to many users. Roles can also be assigned other roles and inherit their privileges, which works well for application roles. For example, suppose the company CallsRus decides to split the sales department into two separate groups, cold call salespeople and call back salespeople. Suppose the call back salespeople need to be able to delete tables from the database for those customers who call back to cancel their orders. These salespeople cannot be assigned to the `CLDCLL_SALES_ROLE` because the privilege to delete tables is denied in that role. Further, the `CLDCLL_SALES_ROLE` cannot be adjusted to allow the deletions because this would not follow the standard of least privilege, as the cold call salespeople would then have access that they do not legitimately need. CallsRus uses the same application for both sales groups. So, the DBA creates a new role with the privilege to delete tables titled `CALBAK_SALES_ROLE` and adds all of the call back salespeople to this group. Now, the database application that is used by the sales team must be available and given access to the database as well. The applications require the same privileges to the database as both `CLDCLL_SALES_ROLE` and `CALBAK_SALES_ROLE` to provide the services that both groups require. In this case, there are two choices. Create a new role and reassign all of the privileges that both groups hold individually, or create the new role `SALES_DEPTAPP_ROLE` and assign the two already existing roles to the new role. The most efficient choice is the latter, which allows for centralized administration and requires less time and resources. For example, if changes need to be made to the call back salespeople's privileges, only one adjustment needs to be made and this is to the

CALBAK\_SALES\_ROLE. Because the SALES\_DEPTAPP\_ROLE is assigned to CALBAK\_SALES\_ROLE, privileges are inherited, so any change made in one will automatically be assigned in the other. In addition to the advantages already mentioned, an administrator can password-protect a role and develop applications to enable a role only when the correct credentials are supplied. Overall, assigning roles to users, objects, applications, and other roles provides better access management and, therefore, a more security-conscious database environment.



No two roles can be assigned the same name. Although this section discusses a hierarchical structure, the roles are not actually contained within each other, so two roles with the same name can cause a database to fail.

**Defining Roles in SQL Server** Within SQL Server, roles are defined at either the server or database level. Server roles grant rights to manipulate the server environment. These rights are granted to login accounts. Database roles grant access to database objects and grant these rights to user accounts. Five types of roles are available within SQL Server: fixed server roles, fixed database roles, user-defined roles, application roles, and public roles.

Fixed server roles include a set of predefined privileges that are combined by SQL Server to create a role for a specific user account. As with fixed privileges, these cannot be changed or deleted, yet users can be added to them. These roles provide server-level privileges. For example, the sysadmin account is assigned a fixed role that includes privileges that are predefined by SQL Server to fit the system administrator user account. Table 6-3 displays a list of predefined fixed server roles for SQL Server; it is important that an administrator familiarize herself with these roles for proper permission granting and management of the server.

User account	User permissions
sysadmin	A system administration account that holds the rights to perform any action at the server level
securityadmin	A system administration account that holds the right to manage and configure the server's security settings (e.g., passwords, logins, auditing, and read error logs)
serveradmin	A system administration account that holds the right to change server configuration settings
setupadmin	A setup administration account that holds the right to manage linked servers, replication, and stored procedures
processadmin	A process administrator account that holds the right to manage the processes running in SQL Server
dbcreator	Database creator accounts that can create, alter, and resize databases
diskadmin	A disk administration account that holds the right to manage disk files

**Table 6-3** Fixed server roles for SQL Server

Like fixed server roles, fixed database roles are predefined by SQL Server to fit the needs of a user account. These roles cannot be altered, yet users can be added to them. Whereas fixed server roles give users access to the server, those who are included within a fixed database role have privileges that are specific to the database. Table 6-4 provides a list of fixed database roles



within SQL Server; administrators should familiarize themselves with these roles to avoid redundancy when creating roles.

User account	User rights
db_owner	Members of the db_owner role hold the rights to perform any action at the server level
db_accessadmin	Members of the db_accessadmin role can add or remove database groups and users
db_datareader	Members of the db_datareader role can see all data from all user tables and have SELECT permission
db_datawriter	Members of the db_datawriter role can add, change, or delete data from all user tables and have INSERT, UPDATE, and DELETE permissions
db_ddladmin	Members of the db_ddladmin role can make any database definition language commands
db_securityadmin	Members of the db_securityadmin role can manage roles and object permissions
db_backupoperator	Members of the db_backupoperator role hold the right to back up the database and force checkpoints
db_denydatareader	Members of the db_denydatareader role are unable to read any data, but they can perform other actions, such as INSERT
db_denydatawriter	Members of the db_denydatawriter role cannot change the data in the database

**Table 6-4** Fixed database roles for SQL Server

User-defined roles are built at the database level and are created to control the access of objects within the database. Although the built-in database roles handle permissions for common database management tasks, a best practice is to group users who have access to perform and require the same database functions. The example provided in the previous section for *CallsRus* displays an example of user-defined database roles and the reason for which they would be created. User-defined roles can be nested to simplify the process.

Application roles are SQL Server roles that are created to support the security requirements of applications. They do not represent a set of users, but are intended to grant permission to software. An application role contains a password, which users must provide to gain access to the privileges of the role, providing more complex security management.

The PUBLIC role is a special role in which every SQL Server database user is a member and cannot be removed. Every SQL Server database has a role named PUBLIC and it provides a way for all users to be assigned a privilege at the same time and in the same manner. PUBLIC should be used with great caution, keeping in mind that any privilege added to the PUBLIC role applies to everyone in the database.

**Defining Roles in MySQL** Although MySQL offers a sophisticated privilege and access system, roles are not included as an ability in MySQL Server alone. Roles can be created in MySQL with the help of scripting and third-party applications. As discussed earlier in the chapter, Securich is a third party that offers security and password support. Securich also provides support for creating roles in MySQL. Again, Securich can be found at [www.securich.com](http://www.securich.com).

**Defining Roles in Oracle** Oracle comes with several predefined, built-in roles that are available to administrators and users alike. These built-in roles are automatically defined for Oracle databases when you run the standard scripts that are included during database creation and if additional options or features are installed, more roles may be created as well. Just as with Oracle privileges, roles provide privileges at two levels: system and object. Roles can be granted to other roles to further manage the access of users and applications. Table 6-5 provides a list of common default Oracle roles, yet note that there are 33 roles for the Oracle database alone.

Role	Information
DBA	Holds access to all areas of the database; this role is provided for compatibility with previous releases of Oracle Database and it is recommended that administrators create their own security-based roles
JAVA_ADMIN	Provides administrative permissions to update policy tables for Oracle Database Java applications
SCHEDULER_ADMIN	Allows the grantee to execute the procedures of the DBMS_SCHEDULER package; it includes all of the job scheduler system privileges and is included in the DBA role
WM_ADMIN_ROLE	Provides all Workspace Manager permissions and includes the grant option; by default, the DBA is granted the WM_ADMIN_ROLE role
XDB_WEBSERVICES	Allows the grantee to access Oracle Database Web services over HTTPS
XDB_WEBSERVICES_OVER_HTTP	Allows the grantee to access Oracle Database Web services over HTTP
MGMT_USER	Provides administrative privileges to perform various activities with Oracle Enterprise Manager
OEM_MONITOR	Provides privileges needed by the Management Agent component of Oracle Enterprise Manager to monitor and manage the database

**Table 6-5** Common predefined Oracle roles

It is also possible to grant roles to PUBLIC, yet this is not recommended and would defeat the reason for needing to create a role altogether. Oracle provides several dictionary views to help you find out what roles are assigned and to whom they have been assigned. Table 6-6 displays a list of the most common views used for locating roles in Oracle.

View	Contained
DBA_ROLES	All of the roles within the database
DBA_ROLE_PRIVS	All of the roles that are assigned to a user
ROLE_ROLE_PRIVS	All of the roles that are granted to other roles
ROLE_SYS_PRIVS	All system privileges that have been assigned to a role
ROLE_TAB_PRIVS	All object privileges assigned to a role
SESSION_ROLES	A list of roles that are enabled in the current session

**Table 6-6** Locating roles in Oracle

Copyright 2011 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s). Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.



## Inference

**Inference** is a way that unauthorized users can obtain sensitive information by making assumptions based on the database's reactions or query responses to nonsensitive queries. Based on the knowledge obtained about the database, the unauthorized users can use their own reasoning and deduction (e.g., SQL syntax, data structure) to draw conclusions about the database that enable them to acquire knowledge or further develop their understanding of the data within it. Inference poses a great security threat to any database management system because it is difficult to predict, detect, and eliminate. Inference can be achieved by internal users, which are those who have some legitimate level of access and authority to the database, or by external users, which are unauthorized outsiders attempting to capture data from Web applications and forms. This section explores the primary means by which internal users obtain unauthorized information using inferences and discusses a few ways in which security professionals and database administrators can detect and minimize the potential for inference. External users and inference are discussed in later chapters.

6

### Examples of Inference

Inference is possible in any way that a user can apply logic to a database's response to a query. This section explores two primary examples and means by which inference takes place within a database environment: those that use logic and those that use statistics. These examples are meant to provide an understanding of the sheer complexity of these likely security hazards as well as display the struggles that security professionals and database administrators must manage every day in their attempts to maintain a secure environment.

**Logic, Relationship, and Constraint Inference** Database management systems are built in a way that enables the efficient storage and retrieval of data. Tables that are well organized, logical, and object oriented are very efficient, yet it is these same characteristics that make them so vulnerable to inference. Once a user obtains enough experience using a particular database, he can use his own logic and his own understanding about object relationships or constraint-driven responses to make inferences about data within a database.

Consider a high-class hotel in Hollywood, California. Each time a person registers for a night, a record is added to a table in the hotel's database that includes his or her customer ID, room number, last name, first name, and profile level. The customer ID acts as the primary key in this table and can be used as a reference to locate other information held in the database related to a particular person, such as date of arrival, home address, and payment information. Table 6-7 represents what the table might look like for the 40th floor of the hotel.

CustID	Room	LName	FName	Profile
120209	4000	Jones	Michael	Low
120210	4001	Lopez	Jennifer	High
120211	4002	Franks	Peter	Low

**Table 6-7** Guest table view

Copyright 2011 Cengage Learning. All Rights Reserved. May not be copied, scanned, or duplicated, in whole or in part. Due to electronic rights, some third party content may be suppressed from the eBook and/or eChapter(s). Editorial review has deemed that any suppressed content does not materially affect the overall learning experience. Cengage Learning reserves the right to remove additional content at any time if subsequent rights restrictions require it.

For this particular hotel, the DBA has created a special security rule or constraint to ensure that only hotel managers can view information about rooms that are occupied by high-profile guests. Let's assume that the hotel desk attendant gets a call from a potential guest inquiring about room availability on the 40th floor of the hotel. She queries the database and obtains the view shown in Table 6-8.

CustID	Room	LName	FName	Profile
120209	4000	Jones	Michael	Low
120211	4002	Franks	Peter	Low

**Table 6-8** Secured available room view

Assuming that room 4001 is available, the desk attendant tries to book the room for the inquiring caller. However, the attempt fails due to another rule that is set up to ensure that no room is booked by two people at the same time. Without much thought, the desk attendant can now infer that the room is occupied by a high-profile client. Further exploration using the customer ID can potentially help the desk attendant acquire more unauthorized information about the guest in room 4001. This is one example of how even in a multilevel, secured database, rules, relationships, logic, and constraints all give clues that can lead to database inference.

**Statistical Inference** Statistical database inferences present another great problem in database security. Statistical queries are often overlooked as potential security breaches because they do not return any actual data from the database. These queries are used to analyze the data in a database for auditing users and finding trends. For example, a statistical database can be used to identify the average amount of user activity over a given time period or to identify the number of queries per user in a specific day. These queries seem harmless and in many ways benefit an organization, yet they can be easily manipulated to retrieve sensitive information such as employee salary and company sales.

For example, let's assume that Sally works in data entry for Haphazard, Inc. and has found out that another data entry worker, Shawn, who has less seniority than herself, has recently been given a raise. Frustrated, and considering leaving the company, she wants to find out how much Shawn makes in comparison with herself. Sally, like most typical database users, does not have the security privileges to query her coworker's private records, but she does hold the rights to perform basic query operations. She discovers that if she can obtain the average of her salary and Shawn's salary combined and apply basic arithmetic, she can figure out exactly how much Shawn makes. This example shows that the statistical database has potential to be compromised, and an unauthorized individual can retrieve sensitive company stats by obtaining collective or aggregate data and applying basic mathematical formulas to it. Like all other types of inferences, those drawn from statistical databases can be done from inside the company as well as from outside the company. Finding the salary for someone outside the company would only require a few more queries and comparisons, but the information is certainly attainable.

## Minimizing Inference

As with all other approaches to security, there is no guaranteed assurance in securing against inference. As was mentioned earlier in this section, inference is difficult to predict, identify, and eliminate. There isn't an automatic scan that will help here, yet there are a few techniques



that can have a great impact on limiting a person's ability to infer. This section identifies a few primary strategies to help minimize inference capabilities within database storage and management environment.

**Polyinstantiation** Polyinstantiation is a strategy that allows the database to contain multiple instances of a record, all pointing to the same primary key, but which contains and displays different values to users of different security classifications. For example, review our Hollywood hotel example from earlier. Polyinstantiation allows two records to exist for the hotel room 4001, both of which point to the same customer ID, as shown in Table 6-9.

CustID	Room	LName	FName	Profile
120209	4000	Jones	Michael	Low
120210	4001	Lopez	Jennifer	High
120210	4001	Smith	Paul	Low
120211	4002	Franks	Peter	Low

Table 6-9 Polyinstantiation view

This strategy enables two separate records to exist and be retrieved and manipulated separately. Those who attempt to view, modify, or delete the record with the correct privileges will have access to the correct record of the guest residing in room 4001, whereas those who attempt to view, modify, or delete the record without the correct privileges will only have access to the “fake” record of the guest in room 4001. If the security classifications are set up correctly, the unauthorized user isn't aware that they have access to a false record and the sensitive record remains safe and secure. Jennifer Lopez can rest easy, knowing that her true identity will not be discovered.

Polyinstantiation is an excellent strategy for keeping sensitive information hidden, yet like all security strategies, it has its downfalls. Many have criticized this technique for being morally wrong because it involves the falsification of records, yet some state that this falsification is greatly justified. Another issue with this strategy is that it leaves a database inconsistent, which can cause great confusion if not properly documented and managed. For example, think of the consequences of confusing the real files with the false ones.

Finally, the redundancy required for this strategy will consume a much greater amount of resources, especially if your security classifications are very granular. All in all, this is a great technique if you have the time and resources to make it work and you feel morally okay with the decision.

**Other Ways to Minimize** Polyinstantiation is an effective technique for keeping sensitive information hidden, yet a number of efforts can be made are less disruptive to the database environment. This section provides a few examples of ways in which inference can be minimized with or without polyinstantiation. These examples do not by any means represent an exhaustive list. If there is anything that we learned about security thus far, it's that it should be multilayered. This list represents a few tasks that focus on inference and that can be included in one of the many layers of security within the environment:

- *Log, monitor, and alert of events*—Activity logging is an important task for any security administrator. Almost every type of activity can be monitored actively within a database environment. Even queries can be monitored in real time. Both statistical and logical inference can be identified by monitoring and analyzing unusual queries and by setting a baseline and threshold alert for unusual user activity. Alerts can be sent directly to an administrator's cell phone for instant action; in addition, a great amount of insight can be found by capturing and analyzing the database activity logs.
- *Limit user capability*—It should be somewhat apparent by now that limiting a user's capabilities within a database is extremely important. This is especially true for minimizing statistical inference. For example, only allowing aggregate operators and limiting a user's query size can be great strides in minimizing inference attempts.
- *Limit query responses*—Limiting query responses can be just as helpful as limiting a user's capabilities. If you feel as though polyinstantiation isn't morally correct, limiting a query response can be almost as effective. For example, when it comes to the extremely sensitive data such as an employee's income, return classes and ranges instead of exact numbers. This helps to maintain morality of the database while minimizing inference.

## Chapter Summary

- Authentication is the process of verifying the identity of a user attempting to access a resource, whereas authorization is the process of verifying the user's permission to access a resource.
- A login account is used to verify the identity of a user for authentication purposes and user accounts are used to determine privileges of a user for authorization purposes.
- Credentials are used to authenticate and authorize a user or application. Credentials can be required a few times throughout the login process, and can require a different set of credentials each time to positively identify a user.
- Credentials can be required at different levels of an environment, as authentication can be verified at the operating system, database, and network layer. Third-party applications can also be used to authenticate a user.
- Operating system authentication requires that the user has an account that resides locally on the server's OS, and only this account is necessary to access the database, despite its location.
- Database authentication checks the user's credentials against an account that resides within the database. Two sets of credentials (operating system and database) are required to log in to the database when database authentication is used.
- Third-party applications can be used to verify a user's identity. These applications use security protocols such as Kerberos and PKI.
- Authentication varies from one database vendor to another. For example, SQL Server supports only two different types of authentication, whereas Oracle supports many.
- Intrusions into a system can originate from a user's password; all of the security measures in the world will not make a difference if your passwords are found out. Therefore, server-enforced password policies are vital to the security of your organization's data.



- Server password policies can be in written form or they can be defined and enforced within your servers. When combined, they offer the most effective way to maintain the privacy of your database environments.
- Database servers can enforce different types of password policies. Variables such as the complexity and the number of failed attempts of a password help to ensure accurate authentication.
- User management is one of the most important duties of an administrator. Any security-conscious DBA knows how to effectively assign privileges and understands the default users that come installed with a specific database.
- User privileges can be granted, denied, or revoked, and administrators should use these functions while following the principle of least privilege.
- For effective access management, related privileges can be combined to create roles, from which users can be added or removed. Roles allow for centralized management and security of the database.
- Inference can be used to gain unauthorized information from a database either intentionally or unintentionally, from both internal users and external intruders. Inference is very difficult to detect, predict, and control.

## Key Terms

**authentication** The process of confirming the identity of those individuals or applications that request access to a secure environment.

**authorization** The process of ensuring that those individuals or applications that request access to an environment or an object within that environment have the permission to do so.

**credential** A piece of information that is used to verify identity, such as a person's username and password, an application's secure ID, or a host's network name and address.

**database link** A link made between two databases that when created results in one logical data storage unit. Links are created in Oracle to apply common policies and to create associations between databases.

**digital certificate** A password-protected and encrypted file that holds the identity of a user or object.

**inference** A way that unauthorized users can obtain sensitive information by making assumptions based on the database's reactions or query responses to nonsensitive queries.

**Kerberos** An authentication protocol that was built by MIT to provide secure means for authentication using symmetric-key cryptology to verify the identity of a client to a server and a server to a client.

**login** An object that is mapped to a user account within each database and is associated to users by the security identifier or SID.

**Mixed Mode Authentication** A form of authentication that allows both Windows authentication and SQL Server authentication to be used. The database will accept both Windows and server logins.

**principle of least privilege** A security standard by which each user added to a system is given the minimum set of privileges that he or she requires to conduct legitimate business within that system.

**privilege** The ability to access a specific database resource or to perform a specific action within a database.

**polyinstantiation** A strategy that allows the database to contain multiple instances of a record, all pointing to the same primary key, but contain and display different values to users of different security classifications.

**role** A set of related privileges that are combined to provide a centralized unit from which to manage similar users or objects of a database.

**service ticket** A unique key that is used to validate a person's identification (similar to a driver's license), for the purpose of gaining access into a secured environment.

**user profile** A set of rules that limits a user's access to database resources, and can be used to set password restrictions as well.

**Windows Authentication mode** A form of authentication that allows only Windows authentication to be used for accessing the database; those users logging in to the database must have a Windows login to access it.

---

## Review Questions

1. Explain the difference between *authentication* and *authorization*.
2. Explain the difference between a login and a user account. Which is used for authentication?
3. List and explain the two authentication methods.
4. Define and explain the process of Kerberos.
5. Define and explain the difference between the authentication modes of at least two of the three database vendors mentioned in the chapter (SQL Server, MySQL, Oracle).
6. Explain at least two server-enforced password policies.
7. Identify two written password policies that you find to be the most important. Explain why you chose those two.
8. Identify and explain the default logins or user accounts for at least two of the database vendors discussed in the chapter.
9. Identify at least five best practices when adding and removing users.
10. Explain the principle of least privilege and how it should be applied within a database environment.
11. Identify three actions that can be applied to a database environment to manage user access.
12. Explain the circumstances in which roles should be used and identify the reasons for using roles.



## Case Projects



### Case Project 6-1: Database Password Policies

Create and document a written password policy to be given out to database users in an organization.

### Case Project 6-2: User Management Policies

Define a set of standards and policies for adding, modifying, and removing users from a database.

### Case Project 6-3: MySQL User Accounts

Using MySQL's Web site (<http://mysql.com>), identify how to create a user in MySQL. Write the steps for creating a user in MySQL.

### Case Project 6-4: Oracle, User Management, and PL/SQL

Using Oracle's Web site (<http://Oracle.com>), identify how to create a user in Oracle. Write the steps for creating a user in Oracle PLSQL.

### Case Project 6-5: SQL, Users, Server Management Studio, and Transact-SQL

Using SQL Server's Web site ([www.microsoft.com/sqlserver/2008/en/us/](http://www.microsoft.com/sqlserver/2008/en/us/)), identify how to create a Windows authenticated login using SQL Server's Server Management Studio and Transact-SQL.

### Case Project 6-6: Roles

Provide a scenario in which a role is assigned to another role to fulfill a specific need.

6

## Hands-On Projects



### Hands-On Project 6-1: Securing the Oracle Environment

You have been hired as the DBA for Haphazard, Inc. You are asked to fulfill the following needs of the Oracle Database environment.

1. Users, roles, and privileges need to be added to the database. Identify the statements that would be used for creating the following users, roles, and privileges that match the following requirements:
  - a. Create a user account NLitinger identified by the password Dubrucr90.
  - b. Grant NLitinger the SELECT and UPDATE permissions on the table called *clients*.
  - c. Create a user account HMimnaugh identified by the password EvCsvds01.
  - d. Grant HMimnaugh the select and delete permissions on the table called *clients*.

- e. Create a user account TylerM identified by the password Fwdtwet12.
  - f. Grant TylerM the select and insert permissions on the table called *clients*.
  - g. Create a role named Most\_Privileged without a password.
  - h. Grant the Most\_Privileged role the Update and Delete permissions on the *clients* table.
  - i. Add all users to the Most\_Privileged role.
2. Identify the most privileged users. (Which user has the most permissions?) Discuss the way in which the preceding steps could have been made more efficient.
3. A password policy needs to be enforced at Haphazard. Identify the statements required to create a server-enforced password policy with the following requirements:
  - a. Complexity is a necessity.
  - b. The password should be a minimum of seven characters.
  - c. Allow the user to reuse the password after a minimum of 10 password changes.
  - d. Lock accounts that have had more than three failed attempts.
  - e. Expire the password every 60 days.
4. What security suggestions would you provide to Haphazard in terms of authentication and authorization? Explain what policies you would develop as well.