Sybase SQL Server™
Security Features User's Guide

Sybase SQL Server Release 11.0.x
Document ID: 36052-01-1100-02
Last Revised: December 15, 1995

Principal author: Barbara Grace

Document ID: 36052-01-1100

This publication pertains to Sybase SQL Server Release 11.0.x of the Sybase database management software and to any subsequent release until otherwise indicated in new editions or technical notes. Information in this document is subject to change without notice. The software described herein is furnished under a license agreement, and it may be used or copied only in accordance with the terms of that agreement.

Document Orders

To order additional documents, U.S. and Canadian customers should call Customer Fulfillment at (800) 685-8225, fax (617) 229-9845.

Customers in other countries with a U.S. license agreement may contact Customer Fulfillment via the above fax number. All other international customers should contact their Sybase subsidiary or local distributor.

Upgrades are provided only at regularly scheduled software release dates.

Copyright © 1989–1995 by Sybase, Inc. All rights reserved.

No part of this publication may be reproduced, transmitted, or translated in any form or by any means, electronic, mechanical, manual, optical, or otherwise, without the prior written permission of Sybase, Inc.

Sybase Trademarks

APT-FORMS, Data Workbench, DBA Companion, Deft, GainExposure, Gain *Momentum*, Navigation Server, PowerBuilder, Powersoft, Replication Server, SA Companion, SQL Advantage, SQL Debug, SQL Monitor, SQL SMART, SQL Solutions, SQR, SYBASE, the Sybase logo, Transact-SQL, and VQL are registered trademarks of Sybase, Inc. Adaptable Windowing Environment, ADA Workbench, AnswerBase, Application Manager, APT-Build, APT-Edit, APT-Execute, APT-Library, APT-Translator, APT Workbench, Backup Server, Bit-Wise, Client-Library, Client / Server Architecture for the Online Enterprise, Client / Server for the Real World, Client Services, Configurator, Connection Manager, Database Analyzer, DBA Companion Application Manager, DBA Companion Resource Manager, DB-Library, Deft Analyst, Deft Designer, Deft Educational, Deft Professional, Deft Trial, Developers Workbench, DirectCONNECT, Easy SQR, Embedded SQL, EMS, Enterprise Builder, Enterprise Client / Server, Enterprise CONNECT, Enterprise Manager, Enterprise SQL Server Manager, Enterprise Work Architecture, Enterprise Work Designer, Enterprise Work Modeler, EWA, Gain Interplay, Gateway Manager, InfoMaker, Interactive Quality Accelerator, Intermedia Server, IQ Accelerator, Maintenance Express, MAP, MDI, MDI Access Server, MDI Database Gateway, MethodSet, Movedb, Navigation Server Manager, Net-Gateway, Net-Library, New Media Studio, OmniCONNECT, OmniSQL Access Module, OmniSQL Gateway, OmniSQL Server, OmniSQL Toolkit, Open Client, Open Client / Server, Open Client / Server Interfaces, Open Gateway, Open Server, Open Solutions, PC APT-Execute,

PC DB-Net, PC Net Library, Powersoft Portfolio, Replication Agent, Replication Driver, Replication Server Manager, Report-Execute, Report Workbench, Resource Manager, RW-DisplayLib, RW-Library, SAFE, SDF, Secure SQL Server, Secure SQL Toolset, SKILS, SQL Anywhere, SQL Code Checker, SQL Edit, SQL Edit / TPU, SQL Server, SQL Server / CFT, SQL Server / DBM, SQL Server Manager, SQL Server Monitor, SQL Station, SQL Toolset, SQR Developers Kit, SQR Execute, SQR Toolkit, SQR Workbench, Sybase Client / Server Interfaces, Sybase Gateways, Sybase Intermedia, Sybase Interplay, Sybase IQ, Sybase MPP, Sybase SQL Desktop, Sybase SQL Lifecycle, Sybase SQL Workgroup, Sybase Synergy Program, Sybase Virtual Server Architecture, Sybase User Workbench, SyBooks, System 10, System 11, the System XI logo, Tabular Data Stream, The Enterprise Client / Server Company, The Online Information Center, Warehouse WORKS, Watcom SQL, WebSights, WorkGroup SQL Server, XA-Library, and XA-Server are trademarks of Sybase, Inc.

All other company and product names used herein may be trademarks or registered trademarks of their respective companies.

Restricted Rights

Use, duplication, or disclosure by the government is subject to the restrictions set forth in subparagraph (c)(1)(ii) of DFARS 52.227-7013 for the DOD and as set forth in FAR 52.227-19(a)-(d) for civilian agencies.

Sybase, Inc., 6475 Christie Avenue, Emeryville, CA 94608.

Table of Contents

Preface

1. Introduction to Security in SQL Server

2. Logging into SQL Server

List of Tables

Preface

This manual describes the security features available with Sybase SQL Server™ release 11.0. It provides instructions and guidelines for using the security features provided with SQL Server. Security is discussed from the perspective of the non-administrative user.

Audience

This manual is intended for all users of SQL Server, but it particularly addresses non-administrative users who own database objects but do not have special roles such as that of System Administrator or System Security Officer. If you are an administrator who is interested in setting up a secure operating environment for SQL Server, read the *Security Administration Guide*, which describes how to administer the security features provided by SQL Server.

This user guide assumes that readers have some general understanding of SQL Server and can understand simple SQL statements.

How to Use This Book

This manual contains these chapters:

- Chapter 1, "Introduction to Security in SQL Server," provides an overview of the security features that are available with SQL Server and directs you to the chapter where each feature is discussed in detail.
- Chapter 2, "Logging into SQL Server," discusses SQL Server login accounts, password management, and how to log in.
- Chapter 3, "Roles in SQL Server," describes the administrative and ownership roles that exist in SQL Server.
- Chapter 4, "Granting Permissions on Objects and Commands," describes the discretionary access control functions provided by SQL Server. The chapter describes how to use the **grant** and **revoke** commands to specify whether users or groups can access particular database objects and to specify which commands or database operations are available to individual users.
- The Glossary provides definitions of terms that might help you as you read the manual.

Related Documents

SQL Server relational database management system documentation is designed to satisfy both the inexperienced user's preference for simplicity and the experienced user's desire for convenience and comprehensiveness. The user's guide and the reference manuals address the various needs of end users, database and security administrators, application developers, and programmers.

Other manuals you may find useful are:

- SQL Server installation and configuration guide, which describes the installation procedures for SQL Server and documents operating-system-specific system administration, security administration, and tuning tasks.
- *SQL Server Performance and Tuning Guide*, which explains how to tune SQL Server for maximum performance. The book includes information about database design issues that affect performance, query optimization, how to tune SQL Server for very large databases, disk and cache issues, and the effects of locking and cursors on performance.
- *SQL Server Reference Manual*, which contains detailed information on all of the commands and system procedures discussed in this manual.
- *SQL Server Reference Supplement*, which contains a list of Transact-SQL® reserved words, definitions of system tables, a description of the *pubs2* sample database, a list of SQL Server error messages, and other reference information common to all the manuals.
- *SQL Server Security Administration Guide*, which is addressed to administrators who are responsible for maintaining a secure operating environment for SQL Server. The manual explains how to use the security features provided by SQL Server to control user access to data. The manual includes information about how to add users to the server, give them controlled access to database objects and procedures, and manage remote servers.
- *SQL Server System Administration Guide*, which provides in-depth information about administering servers and databases. The manual includes instructions and guidelines for managing physical resources and user and system databases, and specifying character conversion, international language, and sort order settings.

- SQL Server utility programs manual, which documents the Sybase utility programs, such as **isql** and **bcp**, that are executed at the operating system level.
- *Transact-SQL User's Guide*, which documents Transact-SQL, Sybase 's enhanced version of the relational database language. It serves as a textbook for beginning users of the database management system.
- *What's New in Sybase SQL Server Release 11.0?*, which describes the new features in release 11.0.

Conventions Used in This Manual

Formatting SQL Statements

SQL is a free-form language: there are no rules about the number of words you can put on a line or where you must break a line. However, for readability, all examples and syntax statements in this manual are formatted so that each clause of a statement begins on a new line. Clauses that have more than one part extend to additional lines, which are indented.

SQL Syntax Conventions

The conventions for syntax statements in this manual are as follows:

Table 1: Syntax statement conventions

Key	Definition
command	Command names, command option names, utility names, utility flags, and other keywords are in **bold Helvetica** in paragraph text and **bold courier** in syntax statements.
variable	Variables, or words that stand for values that you fill in, are in *italics*.
{ }	Curly braces indicate that you choose at least one of the enclosed options. Do not include braces in your option.
[ ]	Brackets mean that choosing one or more of the enclosed options is optional. Do not include brackets in your option.

Table 1: Syntax statement conventions (continued)

Key	Definition
( )	Parentheses are to be typed as part of the command.
|	The vertical bar means you may select only one of the options shown.
,	The comma means you may choose as many of the options shown as you like, separating your choices with commas that are to be typed as part of the command.

- Syntax statements (displaying the syntax and all options for a command) are printed like this:

  sp_dropdevice [*device_name*]

  or, for a command with more options:

  ```
  select column_name
      from table_name
      where search_conditions
  ```

  In syntax statements, keywords (commands) are in non-italic font and user-supplied words (variables) are italicized.

- Examples showing the use of Transact-SQL commands are printed like this:

  ```
  select * from publishers
  ```

- Examples of output from the computer are printed like this:

  ```
  pub_id  pub_name              city       state
  ------- --------------------- --------   ----
  0736    New Age Books         Boston     MA
  0877    Binnet & Hardley      Washington DC
  1389    Algodata Infosystems  Berkeley   CA

  (3 rows affected)
  ```

Case

You can disregard case when you type keywords:

```
SELECT is the same as Select is the same as select
```

Obligatory Options {You Must Choose At Least One}

- Curly braces and vertical bars: Choose **one and only one** option.

```
{die_on_your_feet | live_on_your_knees |
live_on_your_feet}
```

- Curly braces and commas: Choose one or more options. If you choose more than one, separate your choices with commas.

  ```
  {cash, check, credit}
  ```

Optional Options [You Don't Have to Choose Any]

- One item in square brackets: You don't have to choose it.

  ```
  [anchovies]
  ```

- Square brackets and vertical bars: Choose **none or only one**.

  ```
  [beans | rice | sweet_potatoes]
  ```

- Square brackets and commas: Choose **none, one, or more than one** option. If you choose more than one, separate your choices with commas.

  ```
  [extra_cheese, avocados, sour_cream]
  ```

Ellipsis: Do it Again (and Again)...

An ellipsis (. . .) means that you can **repeat** the last unit as many times as you like. In this syntax statement, **buy** is a required keyword:

```
buy thing = price [cash | check | credit]
    [, thing = price [cash | check | credit]]...
```

You must buy at least one thing and give its price. You may choose a method of payment: one of the items enclosed in square brackets. You may also choose to buy additional things: as many of them as you like. For each thing you buy, give its name, its price, and (optionally) a method of payment.

Expressions

Several different types of expressions are used in SQL Server syntax statements.

Table 2: Types of expressions used in syntax statements

Usage	Definition
expression	Can include constants, literals, functions, column identifiers, variables, or parameters

Table 2: Types of expressions used in syntax statements (continued)

Usage	Definition
logical expression	An expression that returns TRUE, FALSE, or UNKNOWN
constant expression	An expression that always returns the same value, such as “5+3” or “ABCDE”
float_expr	Any floating-point expression or expression that implicitly converts to a floating value
integer_expr	Any integer expression, or an expression that implicitly converts to an integer value
numeric_expr	Any numeric expression that returns a single value
char_expr	An expression that returns a single character-type value
binary_expression	An expression that returns a single binary or varbinary value

If You Need Help

Help with your Sybase software is available in the form of documentation and Technical Support.

Each Sybase installation has a designated person who is authorized to contact Sybase Technical Support. If you cannot resolve your problem using the manuals, ask the designated person at your site to contact SybaseTechnical Support.

1 Introduction to Security in SQL Server

SQL Server and Security

SQL Server uses a number of security features that enable you to protect sensitive data from inappropriate access and unauthorized disclosure. SQL Server is targeted for security evaluation at the Class C2 level. The requirements for the C2 criteria are given by the Department of Defense in DOD 52.00.28-STD, *Department of Defense Trusted Computer System Evaluation Criteria* (TCSEC), also known as the "Orange Book."

SQL Server achieves security through a collection of protection mechanisms that enforce a security policy. The administrators of your system determine the local application of the security policy. Your role as a user is to be aware of the security mechanisms that are available to you and to make use of them. The purpose of this manual is to describe these security mechanisms and tell you how to use them effectively. The following sections provide an overview of the major security features provided by SQL Server.

User Identification and Authentication

Every user in SQL Server is given a login account with a unique ID. When you log into SQL Server, you must identify yourself and enter the correct password before you are permitted to access the server. This process is called **identification and authentication**. All of your activity on the server can be attributed to your server user ID, and your activities can be audited.

Your SQL Server password must be six bytes or longer. It is stored in the *master..syslogins* table in encrypted form. In addition, when you log into SQL Server from a client, you can choose client-side password encryption to encrypt your password before sending it over the network. Chapter 2, "Logging into SQL Server," describes your SQL Server login account, password management, and how to log into SQL Server.

Discretionary Access Controls

Discretionary access controls are controls that are used at the discretion of object owners. They are "discretionary" because an

object owner can choose to allow you to access an object or can disallow such access.

The SQL commands **grant** and **revoke** control SQL Server's discretionary access control system. You can give various kinds of permissions to users, groups, and roles with the **grant** command and rescind them with the **revoke** command. **grant** and **revoke** are used to give users permission to create objects within a database, to create databases, and to access specified tables, views, and columns.

Some commands can be used at any time by any user with no permission required. Others can be used only by users of certain status (for example, only by a System Administrator) and are not transferable.

The ability to assign permissions for the commands that can be granted and revoked is determined by each user's status (as System Administrator, Database Owner, or database object owner) and by whether or not a particular user has been granted a permission with the option to grant that permission to other users.

Discretionary access controls are discussed in Chapter 4, "Granting Permissions on Objects and Commands."

Division of Roles

An important feature in SQL Server is the division of **roles**. Various security-related, administrative, and operational tasks are grouped into these roles:

- System Security Officer, who manages security in SQL Server.
- System Administrator, who manages disk storage, server configuration, some aspects of login accounts, backup and recovery of databases, and so forth.
- Operator, who can back up and load databases on a server-wide basis.

These roles provide individual accountability for users performing administrative tasks. Their actions can be audited and attributed to them.

SQL Server roles are discussed in Chapter 3, "Roles in SQL Server."

Auditing

A principal element of a secure system is accountability. One means of ensuring this accountability is by auditing events on the system. Many events that occur in SQL Server can be recorded in an audit record. Each audit record can log the nature of the event, the date and time, the user responsible for it, and the success or failure of the event. Among the events that can be audited are logins and logouts, server boots, use of data access commands, attempts to access particular objects, and a particular user's actions. The **audit trail**, or log of audit records, allows the System Security Officer to reconstruct events that have occurred on the system and evaluate their impact.

Auditing is managed by the System Security Officer, and is discussed in Chapter 8, "Auditing," in the *Security Administration Guide.*

2 Logging into SQL Server

Your SQL Server Login Account

Each SQL Server user has a login account identified by a unique login name and a password. Your account is established for you by a System Security Officer. Your login account has the following characteristics:

- A login name, unique on that server.
- A password, which must be at least six bytes long.
- A default database (optional). If one is defined, you start each SQL Server session in that database without having to issue the **use** command. If none is defined, you start each session in the *master* database.
- A default language (optional). This specifies the language in which prompts and messages will be displayed to you. If one is not defined, SQL Server's default language is used.
- A full name (optional). This is your full name, which can be useful for documentation and identification purposes. If it is omitted, no full name is recorded for your account.

Group Membership

In SQL Server, groups provide a convenient way to grant and revoke permissions to more than one user at a time within a database. For example, if everyone who works in the "Sales" department needs access to certain tables, all of those users can be put into a group called "sales". The Database Owner can grant specific access permissions to that group rather than having to grant permission to each user individually.

A group is created within a database, not on the server as a whole. The Database Owner is responsible for creating groups and assigning users to them. You are always a member of the "public" group, which includes all users on SQL Server. You can also belong to one other group. You can use the **sp_helpuser** system procedure to find out what group you are a member of. The syntax for **sp_helpuser** is:

```
sp_helpuser user_name
```

Getting Information About Your SQL Server Account

You can get information about your own SQL Server login account with the **sp_displaylogin** system procedure. Type:

```
sp_displaylogin
```

SQL Server returns the following information:

- Your server user ID
- Your login name
- Your full name
- Any roles granted to you (regardless of whether they are currently active)
- Whether your account is locked
- The date when you last changed your password

Changing Your Password

You can change your password at any time with the system procedure **sp_password**. The syntax for **sp_password** is:

```
sp_password old_passwd, new_passwd
```

where *old_passwd* is the old password, and *new_passwd* is your new password.

Your password must be at least 6 bytes long and can be any printable letters, numbers, or symbols.

When you specify a password, enclose it in quotation marks if:

- It includes characters other than A-Z, a-z, 0-9, _, #, valid single-byte or multibyte alphabetic characters, or accented alphabetic characters
- It begins with 0-9

Here is how to change the password “terrible2” to “3blindmice”:

```
sp_password terrible2, "3blindmice"
go
```

Notice that the word **go** appears on a line by itself. It must not be preceded by blanks. **go** is the command terminator, which lets SQL Server know that you have finished typing and are ready for your command to be executed.

For more information about **sp_password**, see the *SQL Server Reference Manual*. When you execute a stored procedure, a return status displays at the end of execution. A return status of "0" means that execution was successful.

It is a good idea to change your password periodically. The System Security Officer can configure SQL Server to require you to change your password at preset, regular intervals. If this is the case on your server, SQL Server will notify you when it is time to change your password.

Protecting Your Password

Your password is the first line of defense against SQL Server access by unauthorized people. SQL Server passwords must be at least six bytes long and can contain any printable characters. When you are creating your own password, choose one that cannot be guessed, following these guidelines:

- Do not use personal information, such as your birthday, street address, or any other number that has anything to do with your personal life.
- Do not use names of pets or loved ones.
- Do not use words that appear in the dictionary or words spelled backwards.

The most difficult passwords to guess are ones that combine uppercase and lowercase characters or numbers and letters. Once you have selected a password, protecting it is your responsibility. Never give anyone your password, and never write it down where anyone can see it.

Remote Logins

You can execute stored procedures on a remote SQL Server if you have been granted access to the remote server and an appropriate database on that server. Remote execution of a stored procedure is a **remote procedure call** (RPC).

In order for you to get access to a remote server, the following must occur:

- The remote server must be known to your local server. This occurs when the System Security Officer executes **sp_addserver**.

- You must acquire a login on the remote server. This is accomplished when the System Administrator executes **sp_addremotelogin**.
- You must be added as a user to the appropriate database on the remote server. This is accomplished when the Database Owner executes **sp_adduser**.

These procedures are discussed in Chapter 7, "Managing Remote Servers," in the *Security Administration Guide*.

Once you can access the remote server, you can execute a remote procedure call by qualifying the stored procedure name with the name of the remote server. For example, to execute **sp_help** on the GATEWAY server, execute this command:

```
GATEWAY...sp_help
```

Or, to fully qualify the name of the procedure, include the name of the database containing the procedure and the name of its owner:

```
GATEWAY.sybsystemprocs.dbo.sp_help
```

Changing Your Password on a Remote Server

If you want to change your password, you must change it on all remote servers that you access before you change it on your local server. This is because of the way password checking is done. If you change your password on the local server first, when you issue the remote procedure call (RPC) to execute **sp_password** on the remote server, your passwords will no longer match and the command will fail.

The syntax for changing your password on the remote server is:

```
remote_server...sp_password old_passwd, new_passwd
```

For example:

```
GATEWAY...sp_password terrible2, "3blindmice"
```

Changing your password on the local server is covered in "Changing Your Password" on page 2-2.

How to Log In

There are different ways to log into SQL Server:

- Using **isql**, as discussed in "Logging In with isql" on page 2-5.

- Using Data Workbench® or APT Workbench™, as discussed in "Logging In via Data Workbench or APT Workbench" on page 2-6.
- Through an application written using Open Client Client-Library™ or Open Client DB-Library™. This is discussed in "Handling Client/Server Connection Security with Open Client" on page 2-6.

➤ ***Note***

In addition to using these methods for invoking SQL Server, you can also use utilities such as **bcp** and **defncopy** directly from the operating system. You can use **bcp** to import and export data to and from SQL Server and use **defncopy** to copy the definitions of objects such as procedures and defaults to and from SQL Server. For information about how to use these utilities and for more information about **isql**, see the SQL Server utility programs manual for your platform.

Logging In with *isql*

You can log into SQL Server directly from the operating system with the **isql** utility program.

To use **isql**, type this at your operating system prompt:

```
isql
```

You will then see this prompt:

```
Password:
```

Type your password at the prompt and press the Return key. The password is not shown on the screen as you type. Note that login names and passwords are case sensitive. This is what you will see:

```
1>
```

At this point, you can start issuing Transact-SQL commands.

This login example assumes that your SQL Server login name is the same as your operating system name. For detailed information about specifying your server login name and other parameters for **isql**, see the SQL Server utility programs manual.

Logging Out of *isql*

You can log out of **isql** at any time by typing:

```
quit
```

or

```
exit
```

Either of these commands returns you to your operating system prompt.

Logging In via Data Workbench or APT Workbench

You can also connect to SQL Server through Data Workbench or APT Workbench.

To start Data Workbench, type:

```
dwb
```

at the operating system prompt and press Return. A login window opens, which prompts you for your SQL Server password. For more information about Data Workbench, see the *Data Workbench User's Guide*.

APT-Edit™ is the forms editor in APT Workbench. It is used to create the forms seen by application users. In addition, you can write, compile, test, and modify APT-SQL routines from within APT-Edit. To start APT-Edit, type:

```
apt
```

and then press Return. A login window opens, prompting you for your password. See the *APT Workbench User's Guide* for more information about APT-Edit and APT Workbench.

Logging Out of APT-Edit and Data Workbench

To exit either APT-Edit or Data Workbench, return to the main window, select **/** and then **exit**.

Handling Client/Server Connection Security with Open Client

The Open Client Client-Library and Open Client DB-Library libraries include routines that handle encryption of passwords between the client and the server.

Login Security in DB-Library

If you are using DB-Library to write your application, the following routines handle login security:

- **DBSETLENCRYPT** specifies whether or not network password encryption is to be used when logging into SQL Server (release 10.0 or later).
- **dbsechandle** installs user-defined functions to handle encrypted passwords. This is most useful for gateway applications.

These routines are discussed in detail in the *Open Client DB-Library/C Reference Manual.*

Login Security in Client-Library

Client-Library automatically handles password encryption between client and server, unless the application is a gateway. For gateway applications, you must handle password encryption explicitly, passing the server's encryption key on to the client and then returning the encrypted password back to the server.

For more information about these topics, see the *Open Client Client-Library/C Reference Manual.*

3 Roles in SQL Server

Introduction

In SQL Server, users can be granted special operational and administrative roles. Roles provide individual accountability for users performing system administration and security-related tasks. Roles are granted to individual server login accounts. Actions performed by these users can be audited and attributed to them.

These users gain their special status by being granted their roles with the **sp_role** system procedure. These roles are:

- System Administrator
- System Security Officer
- Operator

In addition, there are two kinds of object owners who gain special status because of the objects they own. These ownership types are:

- Database Owner
- Database object owner

All of these are discussed in the following sections.

System and Security Administration Roles

The System Administrator, System Security Officer, and Operator roles are essential for managing SQL Server.

The System Administrator

A **System Administrator** performs administrative tasks unrelated to specific applications. The System Administrator is not necessarily one individual; the role can be granted to any number of individual login accounts. In a large organization, the System Administrator's role may be carried out by several people or groups.

System Administrator Tasks

System Administrator tasks include:

- Installing SQL Server

- Managing disk storage
- Granting permissions to SQL Server users
- Transferring bulk data between SQL Server and other software programs
- Modifying, dropping, and locking server login accounts
- Monitoring SQL Server's automatic recovery procedure
- Diagnosing system problems and reporting them as appropriate
- Fine-tuning SQL Server by changing the configurable system parameters
- Creating user databases and granting ownership of them
- Granting and revoking the System Administrator role
- Setting up groups (which are convenient in granting and revoking permissions)

System Administrator Permissions

SQL Server does no discretionary access control (DAC) permission checking when a System Administrator accesses objects. A System Administrator takes on the identity of Database Owner in any database he or she enters, including *master*.

There are several commands and system procedures that only a System Administrator can issue and on which permissions cannot be transferred to other users. For detailed information on System Administrator permissions, see Chapter 5, "Roles in SQL Server," in the *Security Administration Guide*.

System Security Officer

The **System Security Officer** is responsible for security-sensitive tasks in SQL Server, such as:

- Creating server login accounts
- Granting and revoking the System Security Officer and Operator roles
- Changing the password of any account
- Setting the system-wide password expiration interval
- Managing the audit system

The System Security Officer can access any database but has no special privileges on objects within databases, with the exception of the *sybsecurity* database. Only System Security Officers can access the *sybsecurity* database and its tables. There are also several system procedures that only a System Security Officer can execute and on which permissions cannot be transferred to other users.

The Operator

An **Operator** is a user who can back up and load databases on a server-wide basis. The Operator role allows a single user to use the **dump database, dump transaction**, **load database**, and **load transaction** commands to back up and restore all databases on a server without having to be the owner of each database. These operations can be performed in a single database by the Database Owner and the System Administrator.

Data Ownership Roles

There are two types of owners recognized by SQL Server:

- Database Owners
- Database object owners

Database Owner

The **Database Owner** is the creator of a database or someone to whom database ownership has been transferred by the System Administrator. The System Administrator grants users the authority to create databases with the **grant** command.

A Database Owner logs into SQL Server using his or her assigned login name and password. In other databases, that owner is known by his or her database user name; in his or her own database, SQL Server recognizes the user as “dbo”.

Database Owner Tasks

The owner of a database may:

- Run the system procedure **sp_adduser** to allow other SQL Server users access to the database

- Use the **grant** command to give other users permission to create objects and execute commands within the database
- Create and drop user messages within the database

The Database Owner can access any object inside the database that he or she owns.

Database Object Owner

Database objects are tables, indexes, views, defaults, triggers, rules, constraints, and procedures. A user who creates a database object is its owner. The Database Owner must first grant the user permission to create the particular type of object.

The database object owner creates an object using the appropriate **create** statement, and then grants permission to other users.

The creator of a database object is automatically granted all permissions on it. System Administrators also have all permissions on the object, as long as they qualify the object name with the owner's name. The owner of an object must explicitly grant permissions to other users before they can access the object. Even the Database Owner cannot use an object directly unless the object owner grants him or her the appropriate permission. However, the Database Owner can always use the **setuser** command to impersonate any other user in the database, including the object owner.

When a database object is owned by someone other than the Database Owner, a user other than the owner (including a System Administrator) must qualify the name of that object with the object owner's name—*ownername.objectname*—to access the object. If an object or a procedure needs to be accessed by a large number of users, particularly in ad hoc queries, having these objects owned by "dbo" greatly simplifies access.

4 Granting Permissions on Objects and Commands

This chapter discusses **discretionary access controls**, which are used at the discretion of an object's owner. They are "discretionary" because an object owner can choose to allow or disallow access to the object.

Discretionary access permissions are controlled mainly through the use of the **grant** and **revoke** commands. This chapter provides database object owners with the information they need to grant and revoke privileges on their objects to other users.

This chapter discusses:

- Types of permissions assigned to users and groups
- How to grant and revoke permissions
- How to use views and stored procedures as security mechanisms

Assigning Permissions to Users

The SQL commands **grant** and **revoke** control SQL Server's command and object protection system. You can give various kinds of permissions to users, groups, and roles with the **grant** command and rescind them with the **revoke** command. **grant** and **revoke** are used to give users permission to:

- Create databases
- Create objects within a database
- Access tables, views, and columns
- Execute stored procedures

Some commands can be used at any time by any user, with no permission required. Others can be used only by users of certain status (for example, only by a System Administrator) and are not transferable.

The ability to assign permissions for the commands that can be granted and revoked is determined by each user's status (as System Administrator, Database Owner, or database object owner) and by whether or not a particular user has been granted a permission with the option to grant that permission to other users.

The Database Owner does not automatically receive permissions on objects owned by other users. But a Database Owner or System

Administrator can always acquire any permission by assuming the identity of the object owner with the **setuser** command and then writing the appropriate **grant** or **revoke** statements.

You can use **grant** and **revoke** to assign two kinds of permissions: **object access permissions** and **object creation permissions**. These are discussed in "Object Access Permissions" on page 4-2 and in "Object Creation Permissions" on page 4-6.

This chapter focuses on the aspects of the permissions system that are of interest to non-administrative users, including the owners of database objects. For more information about permissions, see Chapter 6, "Managing User Permissions" in the *SQL Server Security Administration Guide*, and "grant" and "revoke" in the *SQL Server Reference Manual*.

Object Access Permissions

Object access permissions regulate the use of certain commands that access certain database objects. For example, you must explicitly be granted permission to use the **select** command on the *authors* table. Object access permissions are granted and revoked by the owner of the object, who can grant them to other users.

Table 4-1 lists the types of object access permissions and the objects to which they apply:

Table 4-1: Permissions and the objects to which they apply

Permission	Object
select	table, view, column
update	table, view, column
insert	table, view
delete	table, view
references	table, column
execute	stored procedure

grant and *revoke* Syntax: Object Access Permissions

Here is the full syntax for granting and revoking object access permissions:

```
grant {all | permission_list}
   on {table_name [(column_list)] |
      view_name [(column_list)] |
      stored_procedure_name}
   to {public | name_list | role_name}
   [with grant option]

revoke [grant option for]
   {all | permission_list}
   on {table_name [(column_list)] |
      view_name [(column_list)] |
      stored_procedure_name}
   from {public | name_list | role_name}
   [cascade]
```

You can include more than one command in *permission_list*. Separate the commands with commas.

The parameters of the **grant** and **revoke** commands are as follows:

- If you use the keyword **all** in the statements for object access permissions, every one of the permissions applicable to the object is granted or revoked.
- The contents of *permission_list* varies according to the type of object on which you are granting permissions:
  - When you use the **grant** or **revoke** command to assign permissions on a table or view, *permission_list* can consist of any combination of **select**, **insert**, **delete**, **references**, and **update**.
  - When you grant permissions on columns in a table, *permission_list* can include **select** or **update** or both. To use **select** *, you must have **select** permission on all columns in the table.
  - When you are granting permissions on stored procedures, *permission_list* can include **execute** only.
- The **on** clause specifies the object on which you are granting or revoking permission. You can grant or revoke privileges on tables, views, and stored procedures for only one object at a time. *table_name*, *view_name*, or *stored_procedure_name* is the name of the table, view, or stored procedure. You can grant privileges for more than one column at a time, but all the columns must be in the same table or view. *column_list* is a list of columns separated by commas. For example, to specify the *price* and *total_sales* columns of the *titles* table, enter:

  ```
  on titles(price, total_sales)
  ```

- Use **with grant option** to allow a specified user(s) to grant permissions to other users.

- The **grant option for** clause of the **revoke** command revokes grant permission from the specified user or users. If the user has granted permissions to other users, you must use the **cascade** option as well to revoke permissions from those users.
- The keyword **public** refers to the "public" group, which includes all users of SQL Server. **public** means slightly different things for **grant** and **revoke**:
  - For **grant**, **public** includes you, the object owner. Therefore, if you have revoked permissions from yourself on your object, and later you **grant** permissions to **public**, you regain the permissions along with the rest of "public".
  - For **revoke** on object access permissions, **public** excludes the owner. If a user has **with grant option** permission on an object, that permission is not revoked when permissions on the object are revoked from **public**.
- The *name_list* includes the names of:
  - Groups
  - Users
  - A combination of users and groups with each name separated from the next by a comma
- The *role_name* is the name of a SQL Server role. This allows you to grant permissions to all users who have been granted a specific role. The roles are **sa_role** (System Administrator), **sso_role** (System Security Officer), and **oper_role** (Operator).
- The **with grant option** clause in a **grant** statement specifies that the user(s) specified in *name_list* can grant the specified object access permission(s) to other users. If a user has **with grant option** permission on an object, that permission is not revoked when permissions on the object are revoked from **public** or from a group to which the user belongs.
- The **cascade** option in a **revoke** statement removes the specified object access permissions from the user(s) specified in *name_list* and also from any users they granted those permissions to.

Special Requirements for Compliance to SQL92 Standards

When you have used the **set** command to turn **ansi_permissions** to "on", additional permissions are required for **update** and **delete** statements. The following table summarizes the required permissions.

Table 4-2: ANSI permissions for update and delete

	Permissions Required: *set ansi_permissions* off	Permissions Required: *set ansi_permissions* on
update	**update** permission on columns where values are being set	**update** permission on columns where values are being set and **select** permission on all columns appearing in **where** clause **select** permission on all columns on the right side of the **set** clause
delete	**delete** permission on the table	**delete** permission on the table from which rows are being deleted and **select** permission on all columns appearing in the **where** clause

If **ansi permissions** is set to "on" and you attempt to **update** or **delete** without having the additional required **select** permissions, the transaction is rolled back and you receive an error message. If this occurs, the column owner must grant you **select** permission on all relevant columns.

Examples: Object Access Permissions

The following statement gives Mary and the "sales" group permission to **insert** into and **delete** from the *titles* table:

```
grant insert, delete
on titles
to mary, sales
```

The following statement gives Harold permission to use the stored procedure **makelist**:

```
grant execute
on makelist
to harold
```

The following statement grants permission to execute the stored procedure *sa_only_proc* to users who have been granted the System Administrator role:

```
grant execute
on sa_only_proc
to sa_role
```

The following statement gives Aubrey permission to **select**, **update**, and **delete** from the *authors* table and to grant the same permissions to other users:

```
grant select, update, delete
on authors
to aubrey
with grant option
```

Both of the following statements revoke permission for all users except the table owner to **update** the *price* and *total_sales* columns of the *titles* table:

```
revoke update
on titles (price, total_sales)
from public
```

```
revoke update(price, total_sales)
on titles
from public
```

The following statement revokes permission from Clare to update the *authors* table and simultaneously revokes that permission from all users to whom she had granted that permission:

```
revoke update
on authors
from clare
cascade
```

The following statement revokes permission from Operators to execute the stored procedure *new_sproc:*

```
revoke execute
on new_sproc
from oper_role
```

Object Creation Permissions

Object creation permissions regulate the use of commands that create objects. These permissions can be granted only by a System Administrator or a Database Owner.

The commands to which the object creation permissions apply are:

create database
create default
create procedure
create rule
create table
create view

Each database has its own independent protection system. In other words, being granted permission to use a certain command in one database has no effect in other databases.

grant and *revoke* Syntax: Object Creation Permissions

The syntax for granting and revoking these permissions is:

```
grant {all | command_list}
   to {public | name_list | role_name}

revoke {all | command_list}
   from {public | name_list | role_name}
```

The parameters for these commands are used as follows:

- The *command_list* is a list of the object creation permissions that you are granting or revoking. If more than one command is listed, separate them with commas. The command list can include **create database**, **create default**, **create procedure**, **create rule**, **create table**, and **create view**. **create database** permission can be granted only by a System Administrator, and only from within the *master* database.
- **all** can be used only by a System Administrator or the Database Owner. When used by a System Administrator in the *master* database, **grant all** assigns all **create** permissions, including **create database**. If the System Administrator executes **grant all** from another database, all **create** permissions are granted except **create database**. When a Database Owner who is not a System Administrator uses **grant all**, SQL Server grants all **create** permissions except **create database** and prints an informational message.
- The keyword **public** refers to the group "public", which includes all users of SQL Server. **public** means slightly different things for **grant** and **revoke**:
  - For **grant**, **public** includes the Database Owner. Therefore, if you have revoked permissions from yourself for a command, and

later you **grant** permissions to **public**, you regain the permissions along with the rest of "public".

- For **revoke**, **public** excludes the Database Owner.

- The *name_list* is a list of the names of:
  - Groups
  - Users
  - A combination of users and groups with each name separated from the next by a comma
- The *role_name* is the name of a SQL Server role. This allows you to grant permissions to all users who have been granted a specific role. The roles are **sa_role** (System Administrator), **sso_role** (System Security Officer), and **oper_role** (Operator).

Examples: Object Creation Permissions

The following examples show how to **grant** and **revoke** object creation permissions.

This example grants permission to Mary, Jane, and Bob to create tables and views in the current database:

```
grant create table, create view
to mary, jane, bob
```

This example grants permission to the "admin" group to create stored procedures:

```
grant create procedure
to admin
```

This example revokes permission from Mary to create tables and rules in the current database:

```
revoke create table, create rule
from mary
```

grant and Roles

You can use the **grant** command to grant permission on objects to all users who have been granted a specified role. This allows you to restrict use of an object to users who are System Administrators, System Security Officers, or Operators.

However, **grant** permission does not prevent users who do **not** have the specified role from being granted execute permission on a stored

procedure. If you want to ensure, for example, that only System Administrators can successfully execute a stored procedure, you can use the **proc_role** system function within the stored procedure itself. It checks to see whether the invoking user has the correct role to execute the procedure. See Chapter 5, "Roles in SQL Server," in the *Security Administration Guide* for more information.

Permissions granted to roles override permissions granted to users or groups. For example, assume John has been granted the System Security Officer role, and **sso_role** has been granted permission on the *sales* table. If John's individual permission on *sales* is revoked, he is still able to access *sales* when he has **sso_role** active because his role permissions override his individual permissions.

Combining *grant* and *revoke* Statements

grant and **revoke** statements are order-sensitive: in case of a conflict, the most recently issued statement supersedes all others.

There are two basic styles of setting up permissions in a database or on a database object. The most straightforward style is to assign specific permissions to specific users.

However, if most users are going to be granted most privileges, it's easier to assign all permissions to all users and then revoke specific permissions from specific users.

For example, a Database Owner can grant all permissions on the *titles* table to all users by issuing the following statement:

```
grant all
on titles
to public
```

Then the Database Owner can issue a series of **revoke** statements, for example:

```
revoke update
on titles (price, advance)
from public

revoke delete
on titles
from mary, sales, john
```

Conflicting *grant* and *revoke* Statements

As mentioned in the previous section, **grant** and **revoke** statements are sensitive to the order in which they are issued. For example, if Jose's group has been granted **select** permission on the *titles* table and then Jose's permission to **select** the *advance* column has been revoked, Jose can **select** all the columns except *advance*, while the other users in his group can still **select** all the columns.

A **grant** or **revoke** statement that applies to a group or role changes any conflicting permissions that have been assigned to any member of that group or role. For example, if the owner of the *titles* table has granted different permissions to various members of the *sales* group, and wants to standardize, he or she might issue the following statements:

```
revoke all on titles from sales
grant select on titles(title, title_id, type,
        pub_id)
    to sales
```

Similarly, a **grant** or **revoke** statement issued to **public** will change, for all users, all previously issued permissions that conflict with the new regime.

The same **grant** and **revoke** statements issued in different orders can create entirely different situations. For example, the following set of statements leaves Jose, who belongs to the **public** group, without any **select** permission on *titles*:

```
grant select on titles(title_id, title) to jose
revoke select on titles from public
```

In contrast, the same statements issued in the opposite order result in only Jose having **select** permission and only on the *title_id* and *title* columns:

```
revoke select on titles from public
grant select on titles(title_id, title) to jose
```

Remember that when you use the keyword **public** with **grant**, you are including yourself. With **revoke** on object creation permissions, you are included in **public** unless you are the Database Owner. With **revoke** on object access permissions, you are included in **public** unless you are the object owner. You may wish to deny yourself permission to use your own table, while giving yourself permission to access a view built on it. To do this you must issue **grant** and **revoke** statements explicitly setting your permissions. (You can always change your mind and reinstitute the permission with a **grant** statement.)

Getting Information about Permissions

These system procedures provide information about permissions:

- **sp_helprotect** reports on permissions on database objects or users.
- **sp_column_privileges** reports on permissions on specific columns in a table.
- **sp_table_privileges** reports on permissions on a specific table.

sp_helprotect

The system procedure **sp_helprotect** reports on permissions by database object or by user, and (optionally) by user for a specified object. Any user can execute this procedure. Its syntax is:

```
sp_helprotect name [, name_in_db [, "grant"]]
```

- The first parameter, *name*, is either the name of the table, view, or stored procedure; or the name of a user, group, or role in the current database.
- If you specify the second parameter, *name_in_db*, only that user's permissions on the specified object are reported. If *name* is not an object, **sp_helprotect** checks whether *name* is a user, group, or role. If so, the permissions for the user, group, or role are listed.
- If you specify the third parameter, the keyword **grant**, and *name* is not an object, **sp_helprotect** displays all permissions granted by **with grant option.**

For example, suppose you issue the following series of **grant** and **revoke** statements:

```
grant select on titles to judy
grant update on titles to judy
revoke update on titles(contract) from judy
grant select on publishers to judy
    with grant option
```

To determine the permissions Judy now has on each column in the *titles* table, type:

```
sp_helprotect titles, judy
```

The following display results:

```
grantor grantee type   action   object     column      grantable
------- ------ -----   ------   ------     ------      -------
dbo     judy   Grant   Select   titles     All         FALSE
dbo     judy   Grant   Update   titles     advance     FALSE
dbo     judy   Grant   Update   titles     notes       FALSE
dbo     judy   Grant   Update   titles     price       FALSE
dbo     judy   Grant   Update   titles     pub_id      FALSE
dbo     judy   Grant   Update   titles     pubdate     FALSE
dbo     judy   Grant   Update   titles     title       FALSE
dbo     judy   Grant   Update   titles     title_id    FALSE
dbo     judy   Grant   Update   titles     total_sales FALSE
dbo     judy   Grant   Update   titles     type        FALSE
```

The first row of the display shows that the Database Owner ("dbo") gave Judy permission to select all columns of the *titles* table. The rest of the lines indicate that she can update only the columns listed in the display. Judy's permissions are not grantable: she cannot give **select** or **update** permissions to any other user.

To see Judy's permissions on the *publishers* table, type:

```
sp_helprotect publishers, judy
```

In the display below, the *grantable* column indicates TRUE, meaning that Judy can grant the permission to other users.

```
grantor grantee type   action   object     column      grantable
------- ------ -----   ------   ------     ------      -------
dbo     judy   Grant   Select   publishers all         TRUE
```

sp_column_privileges

sp_column_privileges is a catalog stored procedure that returns information about permissions on columns in a table. The syntax is:

```
sp_column_privileges table_name [, table_owner
   [, table_qualifier [, column_name]]]
```

- *table_name* is the name of the table. This is required.
- *table_owner* can be used to specify the name of the table owner, if it is not "dbo" or the user executing **sp_column_privileges**.
- *table_qualifier* is the name of the current database.
- *column_name* is the name of the column on which you want to see permissions information.
- Use **null** for parameters that you do not wish to specify.

For example, this statement:

```
sp_column_privileges publishers, null, null, pub_id
```

returns information about the *pub_id* column of the *publishers* table. See the *SQL Server Reference Manual* for more specific information about the output of **sp_column_privileges**.

sp_table_privileges

sp_table_privileges is a catalog stored procedure that returns permissions information about a specified table. The syntax is:

```
sp_table_privileges table_name [, table_owner
   [, table_qualifier]]
```

- *table_name* is the name of the table. It is required.
- *table_owner* can be used to specify the name of the table owner, if it is not "dbo" or the user executing **sp_column_privileges**.
- *table_qualifier* is the name of the current database.
- Use **null** for parameters that you do not wish to specify.

For example, this statement:

```
sp_table_privileges titles
```

returns information about all permissions granted on the *titles* table. See the *SQL Server Reference Manual* for more specific information about the output of **sp_table_privileges**.

Permissions on Views and Stored Procedures

Views and stored procedures can serve as security mechanisms. A user can be granted permission on a view or on a stored procedure even if he or she has no permissions on objects that the view or procedure references. For this to be possible, the view or stored procedure and its underlying objects must be owned by the same user. Otherwise, the person using the stored procedure or view must be granted object access permissions on the underlying objects as well as on the view or stored procedure.

SQL Server makes permission checks, as required, when the view or procedure is used. When you create the view or procedure, SQL Server makes no permission checks on the underlying objects. One exception to this occurs when you try to create a procedure that accesses objects in another database. If you do not have the required permissions for the database objects in the other database, SQL Server gives you an error message, and the procedure creation fails.

Views As Security Mechanisms

Through a view, users can query and modify only the data defined by the view. The rest of the database is neither visible nor accessible. Data in an underlying table that is not included in the view is hidden from users who are authorized to access the view but not the underlying table.

Permission to access the view must be explicitly granted or revoked, regardless of the set of permissions in force on the view's underlying tables. By defining different views and selectively granting permissions on them, a user (or any combination of users) can be restricted to different subsets of data. The following examples illustrate the use of views for security purposes:

- Access can be restricted to a subset of the rows of a base table (a value-dependent subset). For example, you might define a view that contains only the rows for business and psychology books, in order to keep information about other types of books hidden from some users.
- Access can be restricted to a subset of the columns of a base table (a value-independent subset). For example, you might define a view that contains all the rows of the *titles* table, but omits the *price* and *advance* columns, since this information is sensitive.
- Access can be restricted to a row-and-column subset of a base table.
- Access can be restricted to the rows that qualify for a join of more than one base table. For example, you might define a view that joins the *titles*, *authors*, and *titleauthor* tables in order to display the names of the authors and the books they have written. This view would hide personal data about authors and financial information about the books.
- Access can be restricted to a statistical summary of data in a base table. For example, you might define a view that contains only the average price of each type of book.
- Access can be restricted to a subset of another view, or of some combination of views and base tables.

As an example, you want to prevent some users from accessing the columns in the *titles* table that have to do with money and sales. You could create a view of the *titles* table that omits those columns, revoke permission from "public" to access the table, and then give all users permission on the view but only the Sales department permission on the table. Here's how:

```
grant all on bookview to public

revoke all on titles from public

grant all on titles to sales
```

You can also set up these privilege conditions without using a view by typing:

```
grant all on titles to public

revoke select, update on titles (price, advance,
   total_sales)
from public

grant select, update on titles (price, advance,
   total_sales)
to sales
```

One possible problem with the second scheme is that users not in the *sales* group who enter the command:

```
select * from titles
```

might be surprised to see the message that includes the phrase:

```
permission denied
```

SQL Server expands the asterisk into a list of all the columns in the *titles* table, and because permission on some of these columns has been revoked from non-sales users, SQL Server returns an error message. The message lists the columns for which the user does not have access.

In order to see all the columns for which they do have permission, the non-sales users would have to name them explicitly. For this reason, creating a view and granting the appropriate permissions on it is a better solution.

In addition to protecting data based on a selection of rows and / or columns, views can be used for **context-sensitive protection**. For example, you can create a view that gives a data entry clerk permission to access only those rows that he or she has added or updated. To do this, you would add a column to a table in which the user ID of the user entering each row is automatically recorded with a default. You can define this default in the **create table** statement like this:

```
create table testtable
    (empid       int,
    startdate    datetime,
    username     varchar(30) default user)
```

Next, define a view that includes all the rows of the table where *uid* is the current user:

```
create view context_view
as
    select *
    from testtable
    where username = user_name()
with check option
```

The rows that are retrievable through this view depend on the identity of the person who issues the **select** command against the view. By adding the **with check option** to the view definition, you make it impossible for any data entry clerk to falsify the information in the *username* column.

Stored Procedures As Security Mechanisms

A user with permission to execute a stored procedure can do so even if he or she does not have permissions on tables or views referenced in it, if the stored procedure and its referenced objects have the same owner. For example, a user might be given permission to execute a stored procedure that updates a row-and-column subset of a specified table even though that user does not have any direct permissions on that table.

Glossary

command

An instruction that specifies an operation to be performed by the computer. Each command or SQL statement begins with a keyword, such as **insert**, which names the basic operation performed. Many SQL commands have one or more keyword phrases, or clauses, that tailor the command to meet a particular need.

context-sensitive protection

Context-sensitive protection provides certain permissions or privileges, depending on the identity of the user. This type of protection can be provided by SQL Server using a view and the **user_id** built-in function.

database

A set of related data tables and other database objects that are organized and presented to serve a specific purpose.

database object

A database object is one of the components of a database: table, view, index, procedure, trigger, column, default, constraint, or rule.

Database Owner

The user who creates a database becomes the Database Owner. A Database Owner has control over all the database objects in that database. The login name for the Database Owner is “dbo”.

dbo

In a user’s own database, SQL Server recognizes the user as “dbo”. A database owner logs into SQL Server using his or her assigned login name and password. See also **Database Owner**.

default

The option chosen by the system when no other option is specified.

default database

The database that a user gets by default when he or she logs in.

default language

1. The default language of a user is the language (such as U. S. English) that displays that user's prompts and messages. It can be set with **sp_modifylogin** or the **language** option of the **set** command.

2. The SQL Server default language is the language that is used to display prompts and messages for all users unless a user chooses a different language.

discretionary access control (DAC)

Restricts access to objects based on identity and/or group membership. The controls are discretionary in the sense that a user with a certain access permission (for example, an object owner) is capable of passing access permission on to any other user, such as with the **grant** command.

login

The name a user uses to log into SQL Server. A login is valid if SQL Server has an entry for that user in the system table *master..syslogins*.

mandatory access control (MAC)

Restricts access to objects based on the sensitivity (as represented by a label) of the information contained in the objects and the formal authorization (that is, clearance) of users to access information of such sensitivity. This security feature is available with Secure SQL Server™, but not with the standard SQL Server.

master database

Controls the user databases and the operation of SQL Server as a whole. Known as *master*, it keeps track of such things as user accounts, ongoing processes, and system error messages.

object

A passive entity that contains or receives information and that cannot change the information it contains. In SQL Server, objects can include rows, tables, databases, stored procedures, and views.

object access permissions

Permissions that regulate the use of certain commands (data modification commands plus **select** and **execute**) to specific tables, views, columns, or procedures. These permissions are granted and revoked by the owner of the object, who can grant them to other users. See also **object creation permissions**.

object creation permissions

Permissions that regulate the use of commands that create objects (for example, **create table**, **create procedure**, and **create database**). These permissions can be granted only by a System Administrator or a Database Owner. See also **object access permissions**.

permission

The authority to perform certain actions on certain database objects or to run certain commands. See also **object access permissions** and **object creation permissions**.

read access

Permission to read an object (for example, to select rows from a table).

remote procedure calls

A **stored procedure** executed on a server different from the server the user is logged into.

roles

Privileges granted to identified users to perform various administrative, operational, and security-related tasks. In SQL Server, the available roles are System Administrator, System Security Officer, and Operator.

SA

See **System Administrator**.

Secure SQL Server

A multilevel trusted database management system that is targeted for evaluation at the Class B1 criteria. It is an enhanced version of the standard SQL Server, which is targeted for evaluation at the Class C2 criteria. The requirements for both criteria are given by the Department of Defense in DOD 52.00.28-STD, *Department of Defense Trusted Computer System Evaluation Criteria* (TCSEC), also known as the "Orange Book." The Secure SQL Server adds security functions to those offered by the standard server, including **mandatory access controls**. See also **SQL Server** and **mandatory access controls**.

server user ID

The ID number by which a user is known to SQL Server.

SQL Server

The server in the Sybase client-server architecture. SQL Server manages multiple databases and multiple users, keeps track of the actual location of data on disks, maintains a mapping of logical data descriptions to physical data storage, and maintains data and procedure caches in memory. SQL Server supports security features such as **discretionary access controls** and division of roles. SQL Server is targeted to evaluate at the Class C2 criteria.

SSO

See **System Security Officer**.

statement

A statement begins with a **keyword** that names the basic operation or command to be performed.

stored procedure

A collection of SQL statements and optional control-of-flow statements stored under a name. SQL Server-supplied stored procedures are called **system procedures**.

subject

An active entity that can manipulate database objects. In SQL Server, subjects include users and processes acting on behalf of users.

System Administrator

A user authorized to handle SQL Server system administration, including installing SQL Server, creating databases, managing disk storage, and fine-tuning SQL Server by changing the configurable system parameters.

system databases

The four databases on a newly installed SQL Server: the *master* database, which controls user databases and the operation of SQL Server; the temporary database (*tempdb*), which is used for temporary tables; the system procedures database (*sybsystemprocs*), and the model database (*model*), which is used as a template to create new user databases. If auditing is installed, SQL Server also includes the *sybsecurity* database, which contains the audit trail.

system procedure

Stored procedures that SQL Server supplies as shortcuts for retrieving information from the system tables, or mechanisms for accomplishing database administration and other tasks that involve updating system catalogs.

System Security Officer

A user who controls security-related operations in SQL Server, including auditing, password management, creating server login accounts, and granting and revoking the System Security Officer and Operator roles.

trigger

A special form of **stored procedure** that goes into effect when a user gives a change command such as **insert**, **delete**, or **update** to a specified table or column. Triggers are often used to enforce referential integrity.

user id

The ID number by which a user is known in a specific database. Distinct from **server user ID.**

view

An alternative way of looking at the data in one or more tables. Usually created as a subset of columns from one or more tables.

write access

Permission to write an object (for example, to update a row or to add a row to a table).

Index

The index is divided into two sections:

- Symbols

  Indexes each of the symbols used in Sybase SQL Server documentation.

- Subjects

  Indexes subjects alphabetically.

Page numbers in **bold** are primary references.

S

T

U

V

W