第 11 章 用户及权限管理

用户和权限管理是数据库管理中的重要方面，不合适的数据库访问权限设置可能会引起严重的安全问题。本章对 Oracle 和 SQL Server 的用户和权限管理方面的概念做出对比，并说明数据库安全访问常用设置的操作方法。

本章主要内容包括：

- Oracle 与 SQL Server 的用户和权限相关概念
- 用户管理
- 密码管理
- Oracle 的权限管理
- SQL Server 的权限管理
- 角色
- SQL Server 安全管理的几个易混淆问题

11.1 Oracle 与 SQL Server 的用户和权限相关概念

作为两个典型的大型数据库产品，Oracle 和 SQL Server 对用户和权限管理都做了精巧的设计。两者在用户和权限管理方面的一些概念虽名称相同，含义却大不一样，本节对几个常用的概念通过对比做出详细说明。

11.1.1 用户

Oracle 数据库和服务器合二为一，Oracle 中只有数据库用户，英文为 user，不存在服务器登录帐号。

SQL Server 的服务器和数据库是两个层次的概念，SQL Server 用户也分为两种，一是服务器登录帐号，二是数据库用户，英文分别是 login 和 user。一个人要操作 SQL Server 数据库，首先要为其创建服务器登录帐号，使他可以登录服务器，在要操作的数据库上还需创建与此登录帐号对应的数据库用户。

Oracle 用户要访问数据库，需要对其赋予相应权限。同样，SQL Server 登录帐号要管理服务器，要对其赋予相应权限。登录到服务器后，若操作数据库，则要对其在这个数据库中对应的用户赋予相应权限。

Oracle 用户分为操作系统验证和数据库验证，操作系统验证用户是把操作系统用户映射为数据库用户，而数据库验证，则是在数据库中创建的、与操作系统无关的用户。与此相同，SQL Server 服务器登录帐号也分为 Windows 验证及 SQL Server 验证两种。Windows 验证登录帐号是把 Windows 的操作系统用户添加为 SQL Server 服务器登录帐号，而 SQL Server 验证的登录帐号与 Windows 无关，是在服务器上创建的另外一种独立帐号。

11.1.2 角色

角色是权限的集合体，目的是方便权限管理，一个角色可以包含另一个角色。

Oracle 的角色分为数据库预定义角色和用户定义角色，常用的预定义角色包括 dba、connect、resource。

SQL Server 的角色分为服务器角色和数据库角色，两者都有预定义角色和用户定义角色，

预定义角色分别称为固定服务器角色和固定数据库角色。常用的固定服务器角色包括 public 和 sysadmin，常用的固定数据库角色包括 public 和 dbo。对于服务器角色，SQL Server 2012 之前版本只有固定服务器角色，从 2012 版本开始，用户也可以自定义服务器角色。

11.1.3 模式和架构

模式是 Oracle 的称呼，架构是 SQL Server 的称呼，两者对应的英文单词都是 schema。

Oracle 数据库中的模式与用户是一一对应的，例如创建了 scott 用户，会自动创建 scott 模式。称 scott 为用户，强调的是其用户属性及其被赋予的权限，称 scott 为模式，则强调的是其拥有的数据库对象，即模式是数据库对象的集合。

SQL Server 2005 之前的版本，schema 一般也翻译为模式，与 Oracle 的模式用法及含义基本相同，即模式与数据库用户是一一对应的。从 2005 版本开始，SQL Server 重新定义了 schema 概念，专门表示数据库对象的集合，而与用户无关，其中文翻译也改称为架构。

11.1.4 SQL Server 中的主体和安全对象

主体和安全对象是 SQL Server 的概念，在 Oracle 中不存在，主体(principals)被赋予权限(permissions)以访问特定安全对象(securables)。

主体是能够使用 SQL Server 资源的实体，也可以说是能够授予 SQL Server 访问权限的实体或者拥有固定权限的固定服务器角色与固定数据库角色。根据影响范围，主体可以分为以下三个不同的类型：

表 11-1 主体类型

类型	主体
Windows	Windows 域用户或用户组
	Windows 本地用户或用户组
SQL Server	固定服务器角色
	服务器角色
	服务器登录帐号
Database	数据库用户
	固定数据库角色
	数据库角色
	应用角色

安全对象是可以被授权访问的资源，安全对象分属于服务器、数据库、架构三个层次。不同的范围包含不同的安全对象，如下表所示：

表 11-2 安全对象

范围	安全对象
服务器	端点
	登录帐号
	数据库
数据库	用户
	角色
	应用程序角色
	架构
	证书
	……

架构	类型	
	XML 架构集合	
	对象	表
		视图
		同义词
		统计信息
		过程
		函数
		约束
		聚合
		队列

11.1.5 权限概念

用户要执行数据库操作，必须对其赋予合适的权限。

Oracle 把权限分为系统权限和对象权限。系统权限不针对某个特殊的数据库对象，而是指执行特定类型 SQL 命令的权限。例如，当用户拥有 create session 权限时，可连接到数据库，当用户拥有 create table 权限时，可创建表。对象权限指访问数据库内各种对象，如表、视图、存储过程的权限。

2005 之前版本，SQL Server 权限分为语句权限及对象权限，对应于 Oracle 的系统权限和对象权限。从 2005 版本开始，SQL Server 引入了架构概念，把权限划分为服务器、数据库、架构、对象及列等层次，对高层次的安全对象赋予权限，则自动拥有其属下的低层次权限，如对用户赋予了某个架构的 select 权限，则此用户对这个架构内的所有表自动拥有 select 权限。

Oracle 和 SQL Server 使用 grant 和 revoke 命令分别执行赋予权限和撤消权限的操作，SQL Server 还增加了 deny 命令，使得权限管理更加灵活。

在 SQL Server 中，deny 命令拒绝通过 grant 命令授予用户的权限，也拒绝通过角色成员继承的角色权限。revoke 命令在 grant 命令之后执行可以撤消通过 grant 命令授予用户的权限，revoke 命令并不撤消通过角色成员继承的权限，revoke 命令在 deny 命令之后执行可以撤消通过 deny 命令拒绝的权限。

11.2 用户管理

与 SQL Server 相比，Oracle 用户有较多属性，创建用户的语法也较复杂。

11.2.1 创建用户

Oracle 数据库中创建用户可指定以下用户属性：名称、密码、默认表空间、默认临时表空间、表空间配额(quota)、概要文件，这些属性中只有名称和口令是必要的，其他属性可选。下面对这几个属性做简单介绍。

默认表空间：用户创建表或索引等对象时，若未指定表空间，则这些对象存放于用户的默认表空间。若创建用户时未指定默认表空间，则其创建的对象存储于数据库的默认表空间。

默认临时表空间：用户执行排序或散裂操作时，若所操作数据量超过 PGA 的工作区大小，则会在用户的默认临时表空间中存放排序过程中的临时数据。若创建用户时未指定默认临时表空间，则此用户使用数据库的默认临时表空间存放临时数据。

表空间配额：用户使用表空间的最大空间限制。若未指定某个表空间的配额，则默认为0。若某个表空间配额设置为 unlimited，则此用户可以无限制地使用这个表空间。若用户被赋予了 unlimited tablespace 系统权限，则这个用户可以无限制地使用数据库中的所有表空间。

概要文件：描述对用户使用资源的限制及密码策略，如连接到数据库的用户 15 分钟内未发出任何请求则断开连接。若未指定概要文件，则默认为 default，default 概要文件默认对不限制资源使用，对密码策略只做了基本设置。

具备 create user 系统权限的用户可以执行 create user 命令创建用户，并根据实际需要指定其各种属性。

下面命令创建数据库验证用户 law，密码亦为 law：

```
SQL> create user law identified by law
  2  default tablespace users
  3  temporary tablespace temp
  4  quota 100m on users
  5  quota unlimited on test
  6  profile default
  7  /
```

此命令创建 law 用户，其默认表空间为 users，默认临时表空间为 temp，users 表空间的配额为 100MB，test 表空间配额无限制，概要文件为 default。

用户创建后，因为还未对其赋予相应权限，所以这个用户对数据库还不能进行任何操作。

SQL Server 创建用户包括创建服务器登录帐号和数据库用户两部分任务，登录帐号针对服务器，数据库用户针对指定数据库。在数据库中创建用户的实质是使得登录帐号可以访问这个数据库，在数据库中可执行的操作决定于对此用户赋予的权限。

先创建测试数据库 testDB，然后在 testDB 内创建 sch 架构：

```
1> create database testDB
2> go
1> use testDB
2> go
已将数据库上下文更改为 "testDB"。
1> create schema sch
2> go
```

创建服务器登录帐号，使用 create login 命令。下面命令创建名称为“law”的登录帐号，其口令为“law1law1”，当此帐号登录到服务器时，默认连接到 testDB 数据库。

```
1> create login law with password='law1law1',default_database=testDB
2> go
```

以上命令新建的 law 帐号虽然默认会授予 connect sql 权限(连接至服务器的权限)，但还不能连接至其默认数据库，因其默认数据库 testDB 内还未建立与之对应的数据库用户。

接下来，可以在 testDB 数据库中创建对应于服务器登录帐号 law 的数据库用户 law_user，并把其默认架构设置为 sch：

```
1> use testDB
2> go
1> create user law_user for login law with default_schema=sch
2> go
```

以上新建的 law_user 会默认授予对 testDB 数据库的 connect 权限，即数据库用户创建之后，即可以连接至数据库。

下面示例创建 Windows 验证的登录帐号，并把其默认数据库设置为 testDB：

```
1> create login [LAW\dbuser] from windows with default_database=testDB
2> go
```

这里的 LAW\dbuser 分别是安装 SQL Server 的计算机名称及 Windows 帐号名称。对其创建数据库用户的命令与 SQL Server 验证登录帐号相同。

创建登录帐号默认会验证是否符合密码策略，若忽略密码复杂度及过期检查，可以使用下面语法形式关闭 check_expiration 和 check_policy 属性：

```
1> create login login4 with password='login4',check_expiration=off, check_policy=off
2> go
```

11.2.2 修改用户属性

Oracle 修改用户属性只要把创建用户的 create user 命令改为 alter user 命令，设置其他属性的语法不变。

下面命令修改 law 用户的各种属性：

```
SQL> alter user law identified by tian
  2  default tablespace test
  3  temporary tablespace temp1
  4  quota 200m on users
  5  account lock
  6  /
```

上面第 5 行的作用是把用户锁住，即不能登录数据库。

SQL Server 修改登录帐号属性只要把 create login 命令改为 alter login 命令，设置其他属性的语法不变，与此类似，修改数据库用户属性使用 alter user 命令，各属性的设置方式与创建数据库用户相同。

下面几个示例分别修改登录帐号的不同属性。

禁用登录帐号 law：

```
1> alter login law disable
2> go
```

再次登录服务器，则出现错误：

```
C:\>sqlcmd -U law -P lawlaw
消息 18470，级别 14，状态 1，服务器 APPLE，第 1 行
用户 'law' 登录失败。原因：该帐户被禁用。
```

修改 law 帐号的名称为 law2，口令为 12345678，默认数据库为 law：

```
1> alter login law with name=law2, password='12345678',default_database=law
2> go
```

下面两个示例分别修改数据库用户名称和默认架构：

```
1> alter user law_user with name=lawuser
2> go
1> alter user lawuser with default_schema=sch
2> go
```

11.2.3 删除用户

Oracle 使用 drop user 命令删除用户，如果此用户对应的模式下包含表等数据库对象，则执行 drop user 命令时要附加 cascade 关键字。

下面命令删除 law 用户：

```
SQL> drop user law;
用户已删除。
```

若删除 scott 用户，因为 scott 模式下包含表，不附加 cascade 关键字会报错：

```
SQL> drop user scott;
drop user scott
*
第 1 行出现错误:
ORA-01922: 必须指定 CASCADE 以删除 'SCOTT'

SQL> drop user scott cascade;
用户已删除。
```

SQL Server 删除用户包括删除登录帐号和数据库用户两种操作。删除登录帐号使用 drop login 命令，删除数据库用户使用 drop user 命令。被删除的登录帐号和数据库用户不能拥有安全对象。

下面命令删除登录帐号 law：

```
1> drop login law
2> go
```

下面命令删除数据库用户 lawuser：

```
1> drop user lawuser
2> go
```

11.2.4 用户信息查询

Oracle 使用 dba_users 数据字典视图查询用户属性信息。

查询所有用户名称：

```
SQL> select username from dba_users;

USERNAME
------------------------------------------
AUDSYS
GSMUSER
SPATIAL_WFS_ADMIN_USR
SPATIAL_CSW_ADMIN_USR
APEX_PUBLIC_USER
……
```

查询 scott 用户的默认表空间、默认临时表空间及概要文件：

```
SQL> select username,
  2         default_tablespace,
  3         temporary_tablespace,
  4         profile
  5  from dba_users
  6  where username='SCOTT'
  7  /

USERNAME   DEFAULT_TABLESPACE   TEMPORARY_TABLESPACE PROFILE
---------- -------------------- -------------------- ---------------
SCOTT      USERS                TEMP                 DEFAULT
```

用户在所有表空间的配额可以通过查询 dba_ts_quotas 中的 max_bytes 列得到，其中的 bytes 列表示用户当前在相应表空间上已经使用的空间大小。

查询 scott 用户的表空间配额及当前使用空间的大小：

```
SQL> select tablespace_name,
  2         max_bytes/1024/1024 as "quota_size(MB)",
  3         bytes/1024/1024     as "current_size(MB)"
  4  from dba_ts_quotas
  5  where username='SCOTT'
  6  /

TABLESPACE_NAME quota_size(MB) current_size(MB)
--------------- -------------- ----------------
USERS                        5            .3125
```

SQL Server 的用户包括登录帐号和数据库用户两类。

使用 system_user 系统函数查询当前登录帐号名称：

```
1> print system_user
2> go
law_x240\Administrator
```

使用 user 函数查看当前数据库用户：

```
1> print user
2> go
dbo
```

使用 sys.server_principals 目录视图查询所有服务器登录帐号信息：

```
1> select name,type_desc,default_database_name
2> from sys.server_principals
3> where type in('S','U')
4> go
name                                type_desc            default_database_name
----------------------------------- -------------------- ------------------------
sa                                  SQL_LOGIN            master
##MS_PolicyEventProcessingLogin##   SQL_LOGIN            master
##MS_PolicyTsqlExecutionLogin##     SQL_LOGIN            master
law_x240\Administrator              WINDOWS_LOGIN        master
NT SERVICE\SQLWriter                WINDOWS_LOGIN        master
NT SERVICE\Winmgmt                  WINDOWS_LOGIN        master
NT Service\MSSQLSERVER              WINDOWS_LOGIN        master
NT AUTHORITY\SYSTEM                 WINDOWS_LOGIN        master
NT SERVICE\SQLSERVERAGENT           WINDOWS_LOGIN        master
NT SERVICE\SQLTELEMETRY             WINDOWS_LOGIN        master
law2                                SQL_LOGIN            law
iml                                 SQL_LOGIN            master

(12 行受影响)
```

其中 type_desc 列是登录帐号的类型，SQL_LOGIN 表示 SQL Server 验证方式的登录帐号，WINDOWS_LOGIN 表示 Windows 验证方式的登录帐号。where 条件中的 type 列是用字符表示的服务器登录帐号类型，常用的有 S、U、G，S 表示 SQL Server 验证，U 表示 Windows 验证，G 表示使用 Windows 验证的 Windows 用户组。

数据库用户信息可以查询 sys.database_principals 得到：

```
1> use law
2> go
已将数据库上下文更改为 'law'。
1> select name,type_desc,default_schema_name
```

```
2> from sys.database_principals
3> go
name                 type_desc            default_schema_name
-------------------- -------------------- --------------------
public               DATABASE_ROLE        NULL
dbo                  WINDOWS_USER         dbo
guest                SQL_USER             guest
INFORMATION_SCHEMA   SQL_USER             NULL
sys                  SQL_USER             NULL
db_owner             DATABASE_ROLE        NULL
db_accessadmin       DATABASE_ROLE        NULL
db_securityadmin     DATABASE_ROLE        NULL
db_ddladmin          DATABASE_ROLE        NULL
db_backupoperator    DATABASE_ROLE        NULL
db_datareader        DATABASE_ROLE        NULL
db_datawriter        DATABASE_ROLE        NULL
db_denydatareader    DATABASE_ROLE        NULL
db_denydatawriter    DATABASE_ROLE        NULL

(14 行受影响)
```

11.2.4 几个预置特殊用户简介

Oracle 数据库中的 sys 和 system 帐号用于系统管理。sys 用户是服务器实例中权限最大的用户，system 用户是数据库中权限最大的用户，sys 用户除了拥有 system 在数据库内的所有权限外，还可以执行启动和关闭数据库、备份恢复等任务。

在 SQL Server 中，sa 和 dbo 分别对应于 Oracle 的 sys 和 system 帐号，用于执行系统管理任务。

sa 帐号是一个 SQL Server 验证的登录帐号，被赋予了 sysadmin 固定服务器角色，其权限类似于 Oracle 的 sys 用户，是服务器上权限最大的帐号，为了提高安全性，应该使用手工创建的服务器管理帐号，禁用 sa 帐号或对其设置强密码。

禁用 sa 帐号，可以使用下面命令：

```
1> alter login sa disable
2> go
```

重新启用，则执行下面命令：

```
1> alter login sa enable
2> go
```

创建数据库时，默认会包含 guest 与 dbo 用户。

在数据库中没有对应用户的服务器登录帐号访问某个数据库时，会继承该数据库中 guest 用户被授予的权限。master 及 tempdb 数据库中的 guest 用户不能禁用。model 数据库的 guest 用户默认是禁用的，所以新建用户数据库中的 guest 用户默认也是禁用的。为了提高安全性，如果没有特殊需要，应该禁用 guest 帐号。数据库中的 guest 用户不能删除，但是可以通过下面命令撤消其 connect 权限，从而在数据库中达到禁用的目的：

```
1> revoke connect from guest
2> go
```

每个数据库都有 dbo 用户，dbo 用户可以对数据库执行任何操作。固定服务器角色 sysadmin 中的成员都被自动映射为数据库中的 dbo 用户，由 sysadmin 角色中的成员创建的对象默认属于 dbo 架构。SQL Server 的 dbo 用户类似于于 Oracle 中的 system 用户。可以把数据库用户加入 db_owner 角色而使其具有与 dbo 用户相同的权限。SQL Server 数据库中的

db_owner 角色类似于 Oracle 数据库中的 DBA 角色。

11.3 密码管理

为了防止用户密码被破解，Oracle 和 SQL Server 对密码策略管理都有完备的规则。对于非操作系统验证的用户，Oracle 使用概要文件设置密码策略，其设置独立于操作系统，SQL Server 通过设置用户属性使用密码策略，其密码策略继承自 Windows 系统的安全设置。

11.3.1 密码策略管理

在密码复杂度方面，Oracle 使用 verify_function_11G、ora12c_verify_function 及 ora12c_strong_verify_function 设置密码复杂度，这是三个用于管理密码复杂度的函数，三个函数要求的密码复杂度逐步增强，ora12c_strong_verify_function 满足美国国防部安全技术实现指南(Department of Defense Database Security Technical Implementation Guide)的要求。

verify_function_11G 要求如下：密码长度在 8 到 30 个字符之间，不能过于简单，如不能是 oracle，abcde，welcome1，user1234 等。密码不能与用户名相同，不能是用户名中各字符的反序，不能是用户名附加 1-100 之间的数字。密码不能与服务器名称相同，也不能是服务器名称附加 1-100 之间的数字。密码中至少包含一个字母和一个数字，但不能包含双引号(即“)。修改密码时，新密码与原密码至少有 3 个不同字符。

ora12c_verify_function 要求密码长度在 8 到 256 个字符之间，其他要求与 verify_function_11G 相同。

ora12c_strong_verify_function 要求如下：密码长度在 9 到 256 个字符之间。密码要包含至少两个大写字母，两个小写字母，两个数字，两个非字母、非数字的特殊字符，但不能包含双引号(即“)。修改密码时，新密码与原密码至少有 4 个不同字符。

使用这几个密码复杂度限制函数之前，要先执行下面 SQL 脚本文件将其创建出来：

```
SQL> @%ORACLE_HOME%/rdbms/admin/utlpwdmg.sql
```

除了密码复杂度以外，还可以在概要文件中使用 password_life_time 等参数规定密码的有效期等属性。

下面命令中创建 pwd_12c 概要文件，对密码复杂度加入了 ora12c_verify_function 函数限制，错误密码最多可以输入 4 次，若达到了允许的最大密码错误次数，则把用户锁住 30 天，密码的有效期为 90 天，从第 91 天开始，3 天内还可以登录数据库，但每次登录时会给出密码将要过期的提示，3 天后，必须修改密码才能登录数据库。

```
SQL> create profile pwd_12c limit
  2  password_verify_function ora12c_verify_function
  3  failed_login_attempts 4
  4  password_lock_time 30
  5  password_life_time 90
  6  password_grace_time 3
  7  /
```

也可以修改现有概要文件，下面命令在 pwd_pro 概要文件中加入了密码复杂度限制：

```
alter profile pwd_pro limit password_verify_function ora12c_verify_function
```

概要文件设置完成后，可以把 pwd_12c 概要文件赋予指定用户，如下面命令将概要文件 pwd_12c 赋予 scott 用户：

```
SQL> alter user scott profile pwd_12c;
```

若要求用户第一次登录数据库时强制修改密码，可以在创建用户时附加 expire 子句：

```
SQL> create user law identified by law
  2  password expire
  3  /
```

若要求用户在下次登录时强制修改密码，则可以执行如下 alter user 命令：

```
SQL> alter user law password expire;
```

此用户下次连接数据库时，会要求其先修改密码：

```
SQL> conn law/law
ERROR:
ORA-28001: the password has expired
更改 law 的口令
新口令:
```

SQL Server 创建或修改 SQL Server 验证的登录帐号时，可继承 Windows 的密码策略，在三个方面管理口令：强制实施密码策略，强制密码过期，下次登录必须更改密码，相应效果与 Oracle 的密码管理策略相似。如下图所示为在 SSMS 中创建登录帐号对话框：

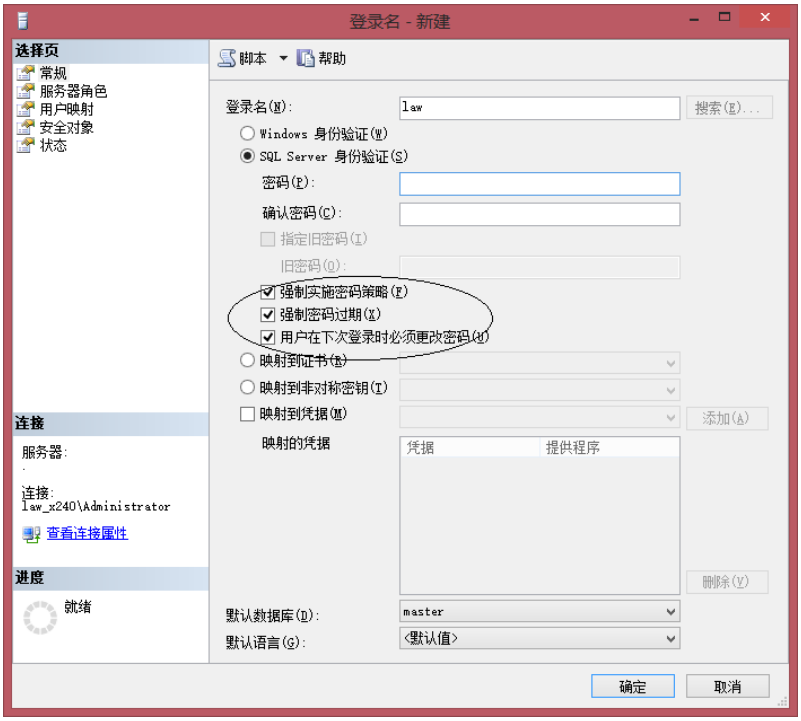

图11-1 用SSMS设置登录帐号密码策略

选择强制实施密码策略时，执行以下规则：密码长度最小值继承 Windows 密码策略中的设置值，最大长度为 128 个字符，若 Windows 密码策略的密码长度最小值设定为 0，则 SQL Server 登录帐号的密码可以为空。SQL Server 的密码复杂性要求也继承 Windows 密码策略中的设置：密码不包含登录帐号名，密码应包含下面四种字符中的三种：大写字母，小写字母，数字，非字母非数字字符。若 Windows 密码策略的复杂性要求处于禁用状态，则 SQL Server 的复杂性要求亦会失效。

当选中“强制实施密码策略”时，“强制密码过期”和“用户在下次登录时必须更改密码”也会自动选中，但“强制密码过期”可手工取消。选中“强制密码过期”时，密码有效

时间继承 Windows 密码策略中的密码最长使用期限。取消“强制实施密码策略”时，另外两个也会取消，要选择“用户在下次登录时必须更改密码”，则必须先选择另外两者。

以上设置也可以使用 SQL 命令实现，如下面命令把三个选项全部开启：

```
1> alter login law with password='iMliaiwu0' must_change,
2> check_policy=on,
3> check_expiration=on
4> go
```

check_policy 对应强制实施密码策略，check_expiration 对应强制密码过期，这两个选项使用 on/off 开启或关闭，must_change 对应下次登录时必须修改密码。需要注意的是，附加 must_change 选项，其功能即开启，不能设置其值为 on/off。

Windows 系统的密码策略可以通过“本地安全策略”设置，可以在“控制面板”的“管理工具”中打开，如下图所示。

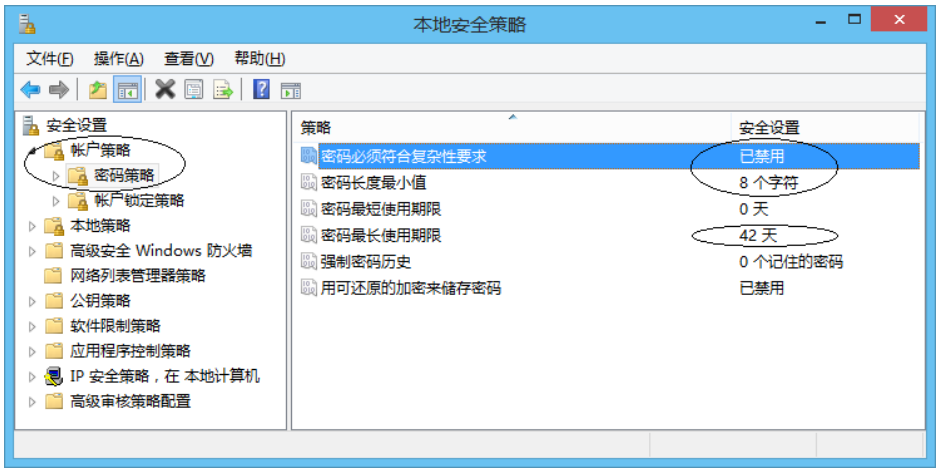

图11-2 Windows本地安全策略设置工具

11.3.2 修改密码

Oracle 使用 password 或 alter user 命令修改用户密码，SQL Server 使用 alter login 命令或 SSMS 图形界面工具修改登录帐号密码。

Oracle 用户修改自己的密码可以直接执行 password 命令，为防止泄漏，键入的新旧密码都不予显示，如 scott 用户修改其密码：

```
SQL> password
更改 SCOTT 的口令
旧口令:
新口令:
重新键入新口令:
口令已更改
```

也可以执行 alter user 命令修改自己的密码，需要使用 replace 关键字指出旧密码：

```
SQL> alter user scott identified by Newpasswd2 replace Tiger123;
```

如果以 system 用户修改普通用户密码，则不需要给出旧密码：

```
SQL> conn system/oracle
已连接。
SQL> alter user scott identified by Newpasswd123;
```

SQL Server 使用 alter login 命令修改密码：

```
1> alter login law with password='Law123456'
2> go
```

11.4 Oracle 的权限管理

建立用户后，要让其执行数据库特定操作，还要对其赋予合适的权限。具备 grant any privilege 系统权限的用户可以授予或撤销其他用户的权限。

11.4.1 系统权限和对象权限

在 Oracle 和其它一些 DBMS 中，通常把权限分为：

- 系统权限
- 对象权限

系统权限不针对某个特殊的数据库对象，而是指执行特定类型 SQL 命令的权限。例如，当用户拥有 create session 权限时，可以连接到数据库，当用户拥有 create table 权限时，可以创建表，这些都是普通的系统权限。

另外一类特殊的系统权限，使用了 any 关键字，用于设置一整类对象的权限，拥有 create any table 权限时，这个用户可以把表创建到其他模式，如 law 用户可以创建 scott.t 表，从而把表创建于非本用户模式。拥有 select any table 权限时，可以查询其他用户的表或视图(不包括数据字典视图)。拥有 update any table，insert any table，delete any table 等权限时，可以对任何其他用户的表执行 update、insert 及 delete 操作。拥有 select any dictionary 权限时，可以查询任何数据字典视图，如 dba_tables、v$instance 等。

还有一个特殊的系统权限为 unlimited tablespace，用户被授予这个权限，可以使用所有表空间，且空间大小不限制。但这个权限不能赋予除 public 以外的其他角色。

对象权限指访问数据库内各种对象，如表、视图、存储过程等的权限。

常用的对象权限如下表所示：

表 11-3 常用对象权限

对象权限名称	表	视图	序列	过程
alter	√		√	
delete	√	√		
execute				√
index	√			
insert	√	√		
references	√			
select	√	√	√	
update	√	√		

11.4.2 所有的系统权限和对象权限

通过查询 system_privilege_map 可以显示所有系统权限，下面的示例显示了系统权限的总数及前 5 个系统权限的名称。

```
SQL> select count(*) from system_privilege_map;

  COUNT(*)
----------
       236
```

```
SQL> select name from system_privilege_map
  2  where rownum<6
  3  /

NAME
----------------------------------------
ALTER ANY CUBE BUILD PROCESS
SELECT ANY CUBE BUILD PROCESS
ALTER ANY MEASURE FOLDER
SELECT ANY MEASURE FOLDER
EXEMPT DDL REDACTION POLICY
```

与系统权限相似，要查询所有对象权限名称可以使用 table_privilege_map 数据字典视图：

```
SQL> select * from table_privilege_map;

 PRIVILEGE NAME
---------- ----------------------------------------
         0 ALTER
         1 AUDIT
         2 COMMENT
         3 DELETE
         4 GRANT
         5 INDEX
         6 INSERT
         7 LOCK
         8 RENAME
         9 SELECT
        10 UPDATE
        11 REFERENCES
        12 EXECUTE
        16 CREATE
        17 READ
        18 WRITE
        20 ENQUEUE
        21 DEQUEUE
        22 UNDER
        23 ON COMMIT REFRESH
        24 QUERY REWRITE
        26 DEBUG
        27 FLASHBACK
        28 MERGE VIEW
        29 USE
        30 FLASHBACK ARCHIVE

已选择 26 行。
```

11.4.3 授予用户权限

授予用户系统权限和对象权限都是使用 grant 命令。

授予系统权限的完整语法为：

grant *system_priv* [, *system_priv*, …] to { *user* | *role* } [, { *user* | *role* , … } [with admin option]

上面语句中的 system_priv 指系统权限的名称，可以用一个语句授予多个系统权限，权限之间用逗号隔开。

user 及 role 指被授予权限的用户或角色名称。

如果授予系统权限时，附加了 with admin option 选项，则用户可以把得到的权限授予其

它用户或角色。

授权示例如下(如果数据库中无 scott 或 law 用户，请重新创建或用其它用户代替)：

```
SQL> grant create session,create table
  2  to scott,law
  3  /
```

在 Oracle 数据库中，用户可以直接访问其对象，但要访问其它用户的对象，则必须具有相应的对象权限。

默认情况下，授予用户对表的访问权限时，列的所有访问权限也会授予用户，如果只允许用户访问某些特定列，则要授予列权限。要注意的是，只能对 insert，update，references 三种权限限制到列，如果要对 select 操作限制到列，可以通过创建包含指定列的视图，然后只授予用户对视图的 select 权限，从而间接实现对其 sleect 操作限制到列。

授予或撤消对象权限时，可以使用 all 选项。执行 grant all on *xxx* 命令后，会将 xxx 的所有对象权限授予用户，而执行 revoke all on *xxx* 命令，则可以撤消 xxx 的所有对象权限限。如果不同用户的表之间存在 references 关系，使用 revoke all 撤消主表上的所有对象权限时，要附加 cascade constraints 选项。

授予对象权限的完整语法如下：

grant { *object_priv* [*column_list*] [, *object_priv* [*column_list*], …] | *all* }

on [*schema*.] *object* to { *user* | *role* } [, *user* | *role* , …] [with grant option]

授予用户对象权限时，如果附加了 with grant option 选项，则此用户可以把得到的权限授予其它用户，这与授予系统权限时附加的 with admin option 选项的作用相似。注意，对象权限可以级联撤消，而系统权限不能。下面是几个授予对象权限的示例。

授予 law 用户改变 emp 表结构的权限：

```
SQL> grant alter on scott.emp to law;
```

授予 law 用户在 emp 表上创建索引的权限：

```
SQL> grant index on scott.emp to law;
```

授予 law 用户引用 dept 表的权限，即可以把外键指向 dept 表的列：

```
SQL> grant references on scott.dept to law;
```

授予 law 用户修改 emp 表的 sal 列的权限：

```
SQL> grant update(sal) on scott.emp to law;
```

表的属主可以把其拥有的对象上的权限赋予其它用户：

```
SQL> conn scott/tiger
已连接。
SQL> grant insert on dept to law;
```

11.4.4 撤消用户权限

撤消系统权限及对象权限的语法相似，都使用 revoke 命令，下面以撤消系统权限为例说明 revoke 的用法。

撤消系统权限的语法如下：

revoke *system_priv* [, *system_priv*, …] from { *user* | *role* } [, { *user* | *role*, … }]

要注意的是，系统权限不能级联撤消，也就是说，如果用户 A 把权限 P 授予用户 B 时，附带有 with admin option 选项，用户 B 又把权限 P 授予了 C，则 A 从 B 撤消权限 P 时，C

的权限 P 不会被级联撤消。

下面示例演示了撤消 law 用户 create session 权限的过程，从实验结果可以看出，如果 public 角色拥有 create session 权限，即使从 law 用户撤消 create session 权限，这个用户依然可以连接到数据库，除非 public 角色的相应权限也被撤消。

撤消 law 用户的 create session 及 create table 权限：

```
SQL> revoke create session,create table from law;
```

以 law 用户连接数据库，并未发生错误：

```
SQL> conn law/law
已连接。
```

撤消 public 角色的 create session 权限：

```
SQL> revoke create session from public;
撤消成功。
```

重新以 law 用户连接数据库，这时发生了错误：

```
SQL> conn law/law
ERROR:
ORA-01045: user LAW lacks CREATE SESSION privilege; logon denied
警告: 您不再连接到 ORACLE。
```

11.4.5 查询用户的权限信息

查询数据库中某个用户或角色的系统权限信息，可以使用 dba_sys_privs 数据字典视图，而使用 user_sys_privs 可以查询当前用户拥有的系统权限。

dba_sys_privs 的结构如下：

```
SQL> desc dba_sys_privs
 名称                      是否为空? 类型
 ------------------------- -------- ----------------
 GRANTEE                            VARCHAR2(128)
 PRIVILEGE                          VARCHAR2(40)
 ADMIN_OPTION                       VARCHAR2(3)
 COMMON                             VARCHAR2(3)
```

如查询 scott 用户的系统权限：

```
SQL> select grantee,privilege
  2  from dba_sys_privs
  3  where grantee='SCOTT'
  4  /

GRANTEE      PRIVILEGE
------------ ----------------------
SCOTT        UNLIMITED TABLESPACE
```

上面查询结果中，继承自其所属角色的系统权限并未显示出来，如果把继承的角色中的系统权限也显示出来，可以先查询用户所属的角色：

```
SQL> select granted_role from dba_role_privs
  2  where grantee='SCOTT'
  3  /

GRANTED_ROLE
------------
```

```
RESOURCE
CONNECT
```

除了这些角色以外，数据库中的所有用户都属于 public 角色。

查询角色中的系统权限也是使用 dba_sys_privs，结合上面两个查询，scott 用户的所有系统权限，可以使用下面查询：

```
SQL> select privilege
  2  from dba_sys_privs
  3  where grantee='SCOTT'
  4  or grantee='PUBLIC'
  5  or
  6  grantee in
  7  (select granted_role from dba_role_privs where grantee='SCOTT')
  8  /

PRIVILEGE
--------------------
UNLIMITED TABLESPACE
CREATE SEQUENCE
CREATE TRIGGER
SET CONTAINER
CREATE CLUSTER
CREATE PROCEDURE
CREATE TYPE
CREATE SESSION
CREATE OPERATOR
CREATE TABLE
CREATE INDEXTYPE

已选择 13 行。
```

查询数据库中某个用户的对象权限使用 dba_tab_privs，与查询系统权限类似，下面命令查询用户的对象权限信息：

```
SQL> select grantee,owner,table_name, privilege
  2  from dba_tab_privs
  3  where grantee='LAW'
  4  or
  5  grantee in
  6  (select granted_role from dba_role_privs where grantee='LAW')
  7  /

GRANTEE      OWNER      TABLE_NAME PRIVILEGE
------------ ---------- ---------- --------------------
LAW          SCOTT      EMP        SELECT
```

11.5 SQL Server 的权限管理

SQL Server 的权限种类和管理方式与 Oracle 有很大区别，最大的不同是根据被管理对象的层次，权限也分为高低不同的层次，特别是增加了架构概念。

11.5.1 架构的概念及其管理

从 SQL Server 2005 版本开始，数据库用户不再等同于架构。架构是与数据库用户无关的命名空间，也可以说，架构是数据库对象的容器。

架构有以下几个特点：

- 架构的所有权可以转移。
- 对象可以在不同的架构之间移动。
- 单个架构可以包含多个数据库用户拥有的对象。
- 多个数据库用户可以共享一个默认架构。
- 可以删除数据库用户，而不删除相应架构中的对象。

下面我们用几个示例说明对架构的常见管理操作，包括创建、删除架构，转移架构中的指定对象以及有关架构系统信息的查询。

删除 sch 架构：

```
1> drop schema sch
2> go
```

删除架构之前要先删除其中的对象。

创建架构，名称为 sch：

```
1> create schema sch
2> go
```

设置 sch 为数据库用户 user1 的默认架构：

```
1> alter user user1 with default_schema=sch
2> go
```

上述命令其实是在修改用户的默认架构属性。

把表 t 创建至 sch 架构：

```
1> create table sch.t(a int)
2> go
```

再创建架构 sch1：

```
1> create schema sch1
2> go
```

把 sch 中的 t 表转移至 sch1 架构：

```
1> alter schema sch1 transfer sch.t
2> go
```

查询数据库中的所有架构及其属主：

```
1> select p.name as db_user, s.name as sch_name
2> from sys.database_principals p,sys.schemas s
3> where s.principal_id=p.principal_id
4> go
db_user                  sch_name
------------------------ ------------------------
dbo                      dbo
dbo                      sch
guest                    guest
INFORMATION_SCHEMA       INFORMATION_SCHEMA
sys                      sys
db_owner                 db_owner
db_accessadmin           db_accessadmin
db_securityadmin         db_securityadmin
db_ddladmin              db_ddladmin
db_backupoperator        db_backupoperator
```

```
db_datareader              db_datareader
db_datawriter              db_datawriter
db_denydatareader          db_denydatareader
db_denydatawriter          db_denydatawriter
```

查询 sch1 架构中包含的表：

```
1> select name from sys.objects
2> where schema_id=schema_id('sch1')
3> go
name
--------------------
t
```

查询包含 t 表的架构：

```
1> select name, schema_name(schema_id)
2> from sys.tables
3> where name='t'
4> go
name
-------------------- --------------------
t                    sch1
```

11.5.2 主要权限列表

SQL Server 中的几种主要权限可以列表如下：

表 11-4 SQL Server 主要权限列表

权限名称	描述
alter	可对授权范围内的任何安全对象执行 alter、create 及 drop 命令。
connect	可连接到 SQL Server 资源，如数据库。若安全对象为服务器，则此权限使用的名称为 control sql。
control	拥有对安全对象及其内部更低层次对象的所有权限。若安全对象为服务器，则此权限使用的名称为 control server。
create	能够创建指定类型的安全对象，如 create table 权限可分别创建表。
impersonate	使被授权者能够以另外一个主体身份执行特定命令，如执行 execute as 命令就需要用户具备 impersonate 权限。
take ownership	使被授权者可以使用 alter authorization 命令成为指定安全对象的属主(owner)，注意，授予 take ownership 命令本身并未改变安全对象的属主。
view	使被授权者可以查看指定对象的系统定义信息(system metadata)。
select	对表或视图或授权范围内的表或视图进行 select 操作。
insert	对表或视图或授权范围内的表或视图进行 insert 操作。
delete	对表或视图或授权范围内的表或视图进行 delete 操作。
update	对表或视图或授权范围内的表或视图进行 update 操作。
references	引用指定表或授权范围内表的权限，即创建外键时允许指向此表的列。
execute	对存储过程及函数，或授权范围内的所有存储过程及函数的执行权限。

使用系统表函数 sys.fn_builtin_permissions()可以返回 SQL Server 的不同层次安全对象的所有权限。使用这个函数时，可选参数值为：default | null | empty_string(三者功能相同)或一个指定的安全对象层次，前者返回所有类型安全对象的所有权限，而后者返回指定层次安全对象的所有权限。常用的主要是 object、schema、database、server。

如查询适用于所有安全对象层次的所有权限：

```
1> select class_desc, permission_name
2> from sys.fn_builtin_permissions(null)
3> go
class_desc                                permission_name
----------------------------------------- ----------------------------------------
DATABASE                                  CREATE TABLE
DATABASE                                  CREATE VIEW
DATABASE                                  CREATE PROCEDURE
DATABASE                                  CREATE FUNCTION
...
DATABASE                                  EXECUTE ANY EXTERNAL SCRIPT
DATABASE                                  CONTROL
OBJECT                                    SELECT
OBJECT                                    UPDATE
...
OBJECT                                    TAKE OWNERSHIP
OBJECT                                    CONTROL
TYPE                                      REFERENCES
TYPE                                      EXECUTE
TYPE                                      VIEW DEFINITION
TYPE                                      TAKE OWNERSHIP
TYPE                                      CONTROL
SCHEMA                                    SELECT
SCHEMA                                    INSERT
SCHEMA                                    UPDATE
SCHEMA                                    DELETE
...
SERVER                                    IMPERSONATE ANY LOGIN
SERVER                                    SELECT ALL USER SECURABLES
...
SERVER ROLE                               TAKE OWNERSHIP
SERVER ROLE                               CONTROL

(230 行受影响)
```

把系统函数 sys.fn_builtin_permissions()的参数替换为 database、schema、object 等值，可以查询相应层次下的所有权限。

各种层次的权限个数统计如下：

```
1> select class_desc, count(permission_name) as count
2> from sys.fn_builtin_permissions(null)
3> group by class_desc
4> order by class_desc
5> go
class_desc           count
-------------------- -----------
APPLICATION ROLE               3
ASSEMBLY                       5
ASYMMETRIC KEY                 5
AVAILABILITY GROUP             4
CERTIFICATE                    5
CONTRACT                       5
DATABASE                      75
ENDPOINT                       5
FULLTEXT CATALOG               5
FULLTEXT STOPLIST              5
LOGIN                          4
MESSAGE TYPE                   5
OBJECT                        12
```

```
REMOTE SERVICE BINDI          4
ROLE                          4
ROUTE                         4
SCHEMA                       12
SEARCH PROPERTY LIST          5
SERVER                       34
SERVER ROLE                   4
SERVICE                       5
SYMMETRIC KEY                 5
TYPE                          5
USER                          4
XML SCHEMA COLLECTIO          6

(25 行受影响)
```

11.5.3 权限管理的三个命令

主体(principals)被赋予权限(permissions)以访问特定安全对象(securables)。

与权限相关的三个命令为：

- grant：赋予主体权限。
- deny：除了撤消显式赋予的权限外，也撤消通过组或角色成员身份继承的权限。
- revoke：在 grant 命令之后执行，撤消主体被赋予的相关权限，在 deny 命令之后执行，则去除 deny 命令的效果。

Oracle 只支持其中的 grant 和 revoke 命令。

对不同层次的主体及安全对象，这三个命令的用法类似，其简化的语法形式分别为：

grant *permission* [on [*class*::] *securable*] to *principal* with grant option

revoke *permission* [on [*class*::] *securable*] from *principal*

deny *permission* [on [*class*::] *securable*] to *principal*

各个参数的含义如下：

- permission：用逗号隔开的若干权限名称。
- class：安全对象类型。
- securable：安全对象名称。
- principal：接受权限的主体(服务器登录帐号或数据库用户、角色等)名称。
- with grant option：主体被授予的权限可以再由这个主体授予其它主体。

执行这些操作时，要注意以下几点：

- 执行授权操作的主体一般是相关安全对象的属主或对安全对象拥有 control 权限。
- 安全对象是服务器或属于服务器层次时，授权之前要先连接到 master 数据库。
- 安全对象是服务器层次或数据库层次时，[on [*class*::] *securable*]可以省略。
- 安全对象是架构或表时，执行授权操作要先连接到架构或表所在的数据库。
- 安全对象是表、视图、存储过程等时，schema 关键字可以省略。
- 安全对象是表、视图、存储过程等时，若其所属架构不是执行授权命令用户的默认架构，则要在安全对象之前指定其所属架构名称。

11.5.4 服务器层次的权限管理

服务器层次的权限管理指服务器层次安全对象的权限管理，位于权限层次结构的顶端。服务器层次的安全对象是在实例范围内唯一的对象，包括服务器本身，以及服务器内部的登录帐号、端点及数据库，其权限只能授予服务器层次的主体，即 SQL 登录帐号或 Windows 登录帐号，而不能授予数据库用户或数据库角色。

服务器层次的权限允许用户执行诸如创建数据库、创建登录帐号、创建链接服务器或关

闭实例、使用 SQL Profiler 等任务。

所有的服务器层次权限可以查询如下：

```
1> select permission_name from sys.fn_builtin_permissions('server')
2> order by permission_name
3> go
permission_name
------------------------------------
ADMINISTER BULK OPERATIONS
ALTER ANY AVAILABILITY GROUP
ALTER ANY CONNECTION
ALTER ANY CREDENTIAL
ALTER ANY DATABASE
ALTER ANY ENDPOINT
ALTER ANY EVENT NOTIFICATION
ALTER ANY EVENT SESSION
ALTER ANY LINKED SERVER
ALTER ANY LOGIN
ALTER ANY SERVER AUDIT
ALTER ANY SERVER ROLE
ALTER RESOURCES
ALTER SERVER STATE
ALTER SETTINGS
ALTER TRACE
AUTHENTICATE SERVER
CONNECT ANY DATABASE
CONNECT SQL
CONTROL SERVER
CREATE ANY DATABASE
CREATE AVAILABILITY GROUP
CREATE DDL EVENT NOTIFICATION
CREATE ENDPOINT
CREATE SERVER ROLE
CREATE TRACE EVENT NOTIFICATION
EXTERNAL ACCESS ASSEMBLY
IMPERSONATE ANY LOGIN
SELECT ALL USER SECURABLES
SHUTDOWN
UNSAFE ASSEMBLY
VIEW ANY DATABASE
VIEW ANY DEFINITION
VIEW SERVER STATE

(34 行受影响)
```

下面几个示例演示授予 login1 登录帐号服务器层次的不同权限。

为了完成本节及后面的实验，先创建测试数据库 law 及测试帐号 login1、login2：

```
1> create database law
2> go
1> create login login1 with password='login1login1',default_database=law
2> create login login2 with password='login2login2'
3> go
```

新建的服务器登录帐号默认会赋予 connect sql 权限，即连接到服务器的权限，但若真的能连接到服务器，还要在其默认数据库内有与其对应的数据库用户，否则会报错如下：

```
C:\Windows\system32>sqlcmd -U login1 -P login1login1 -d law
Sqlcmd: 错误: Microsoft ODBC Driver 11 for SQL Server: 用户 'login1' 登录失败。。
```

```
Sqlcmd: 错误: Microsoft ODBC Driver 11 for SQL Server: 无法打开登录所请求的数据库 "law"。登录失败。。
```

在 law 数据库中创建测试用户 user1，对应 login1 登录帐号：

```
1> use law
2> go
已将数据库上下文更改为 'law'。
1> create user user1 for login login1 with default_schema=sch
2> go
```

授予 login1 帐号连接到服务器的权限：

```
1> grant connect sql to login1
2> go
```

也可以使用下面语法形式，在授权时指明安全对象类型及名称：

```
1> grant connect sql on server::mssqlserver to login1
2> go
```

login1 帐号现在可以登录服务器了，登录后会默认连接至 law 数据库。

授予 login1 帐号创建数据库的权限：

```
1> grant create any database to login1
2> go
```

授予 login1 帐号关闭服务器的权限：

```
1> grant shutdown to login1
2> go
```

授予 login1 帐号控制服务器的权限：

```
1> grant control server to login1
2> go
```

授予此权限后，login1 帐号和 sa 帐号的权限系统，可以在服务器上执行任何操作。

授予 login1 帐号对 login2 帐号的 alter 权限：

```
1> grant alter on login::login2 to login1
2> go
```

拒绝 login1 帐号控制服务器的权限：

```
1> deny control server to login1
2> go
```

撤消 login1 帐号控制服务器的权限：

```
1> revoke control server from login1
2> go
```

使用 sys.server_permissions 可以查询服务器登录帐号的权限信息。

查询 login1 帐号拥有的权限信息：

```
1> select suser_name(grantee_principal_id) as login_name,
2>        permission_name as permission,
3>        class_desc as class,
4>        state_desc as state
5> from sys.server_permissions
6> where suser_name(grantee_principal_id)='login1'
```

```
7> go
login_name           permission           class                state
-------------------- -------------------- -------------------- ------------------
login1               CONNECT SQL          SERVER               GRANT
login1               CREATE ANY DATABASE  SERVER               GRANT
login1               SHUTDOWN             SERVER               GRANT
login1               ALTER                SERVER_PRINCIPAL     GRANT
```

11.5.6 数据库层次的权限管理

与前面内容类似，可以使用sys.fn_builtin_permissions()函数查询数据库层次的所有权限：

```
1> select permission_name from sys.fn_builtin_permissions('database')
2> order by permission_name
3> go
permission_name
----------------------------------------
ALTER
ALTER ANY APPLICATION ROLE
ALTER ANY ASSEMBLY
ALTER ANY ASYMMETRIC KEY
ALTER ANY CERTIFICATE
ALTER ANY COLUMN ENCRYPTION KEY
ALTER ANY COLUMN MASTER KEY
ALTER ANY CONTRACT
...
CREATE TABLE
CREATE TYPE
CREATE VIEW
CREATE XML SCHEMA COLLECTION
DELETE
EXECUTE
EXECUTE ANY EXTERNAL SCRIPT
INSERT
KILL DATABASE CONNECTION
REFERENCES
SELECT
SHOWPLAN
SUBSCRIBE QUERY NOTIFICATIONS
TAKE OWNERSHIP
UNMASK
UPDATE
VIEW ANY COLUMN ENCRYPTION KEY DEFINITIO
VIEW ANY COLUMN MASTER KEY DEFINITION
VIEW DATABASE STATE
VIEW DEFINITION
(75 行受影响)
```

下面是几个以 law 数据库和 user1 用户为例，赋予数据库层次权限的示例。执行下面命令之前，要先切换至 law 数据库。

授予 user1 用户查询数据库中所有表的权限：

```
1> use law
2> go
已将数据库上下文更改为 'law'。
1> grant select to user1
2> go
```

也可以在权限名称后面指明对象类型及名称，两种方式是等效的：

```
1> grant select on database::law to user1
2> go
```

授予 user1 用户使用 alter database 命令修改数据库属性的权限：

```
1> grant alter to user1
2> go
```

授予 user1 用户创建表及视图的权限：

```
1> grant create table, create view to user1
2> go
```

授予 user1 用户修改数据库内各种架构的权限：

```
1> grant alter any schema to user1
2> go
```

撤消 user1 用户创建表的权限：

```
1> revoke create table from user1
2> go
```

查询用户的数据库层次权限可以使用 sys.database_permissions，如下面命令所示：

```
1> use law
2> go
已将数据库上下文更改为 'law'。
1> select user_name(grantee_principal_id) as grantee,
2>         class_desc as class,
3>         object_name(major_id) as object_name,
4>         permission_name,
5>         state
6> from sys.database_permissions
7> where user_name(grantee_principal_id)='user1'
8> and class_desc='database'
9> go
grantee            class              object_name        permission_name    state
------------------ ------------------ ------------------ ------------------ -----
user1              DATABASE           NULL               ALTER              G
user1              DATABASE           NULL               ALTER ANY SCHEMA   G
user1              DATABASE           NULL               CONNECT            G
user1              DATABASE           NULL               CREATE VIEW        G
user1              DATABASE           NULL               SELECT             G
```

以上查询结果中的 connect 权限是在创建用户时默认授予的。

11.5.7 架构层次的权限管理

授予数据库用户架构层次的权限，可使用户能够对架构内的相关对象执行指定操作。所有针对架构的权限可以查询如下：

```
1> select permission_name
2> from sys.fn_builtin_permissions('schema')
3> go
permission_name
--------------------------------------------------
SELECT
INSERT
UPDATE
DELETE
```

```
REFERENCES
EXECUTE
CREATE SEQUENCE
VIEW CHANGE TRACKING
VIEW DEFINITION
ALTER
TAKE OWNERSHIP
CONTROL

(12 行受影响)
```

架构层次的各种权限与数据库层次权限相似，只不过，这里的权限是指架构内各种安全对象的权限。如 select，insert，update，delete 是对架构内的表或视图执行相应 DML 命令的权限，alter 是对架构内的各种安全对象执行 create、drop、alter 命令的权限，control 是对架构的控制权限，即对架构可以进行任意操作。

架构层次的权限是针对架构的操作权限，授予或撤消架构层次的权限时，要指明权限名称以及架构的名称，架构层次的权限名称用以下形式表示：

permission1[, permission2, ...] on schema::*schema_name*

其中，permission1、permission2 用于指定多个权限名称，schema 关键字用于限定对象类型，schema_name 用于指定架构名称。

我们用几个简单示例来说明如何授予和撤消架构层次的权限。

授予 user1 用户对 sch 架构的 alter 及 update 权限：

```
1> grant alter,update on schema::sch to user1
2> go
```

授予 user1 用户对 sch1 架构的 select 权限：

```
1> grant select on schema::sch1 to user1
2> go
```

查询用户的架构层次权限，依然使用 sys.database_permissions 目录视图：

```
1> select user_name(grantee_principal_id) as grantee,
2>         class_desc as class,
3>         schema_name(major_id) as schema_name,
4>         permission_name as permission,
5>         state
6> from sys.database_permissions
7> where user_name(grantee_principal_id)='user1'
8> and class_desc='schema'
9> go
grantee      class        schema_name  permission   state
------------ ------------ ------------ ------------ -----
user1        SCHEMA       sch          ALTER        G
user1        SCHEMA       sch          UPDATE       G
```

下面命令撤消架构层次的权限：

```
1> revoke alter,update on schema::sch from user1
2> go
```

11.5.8 对象权限

对象权限是对架构内的各种数据库对象执行各种操作的权限。

所有的对象权限可以查询如下：

```
1> select permission_name
2> from sys.fn_builtin_permissions('object')
3> go
permission_name
--------------------------------------------
SELECT
UPDATE
REFERENCES
INSERT
DELETE
EXECUTE
RECEIVE
VIEW CHANGE TRACKING
VIEW DEFINITION
ALTER
TAKE OWNERSHIP
CONTROL

(12 行受影响)
```

授予或撤消对象权限时，如果对象所在的架构不是执行授权命令用户的默认架构，则需要指明架构的名称：

```
1> grant select, insert on sch1.t to user1
2> go
```

也可以附加安全对象的类型(即 object)，而采用以下语法形式：

```
1> grant select, insert on object::sch1.t to user1
2> go
```

表的权限可以限制到列级，如下面命令授予 user1 用户对 t 表 a 列的 update 权限：

```
1> grant update(a) on sch1.t to user1
2> go
```

要查询数据库中的对象权限信息，可以执行下面查询：

```
1> select user_name(grantee_principal_id) as grantee,
2>         class_desc as class,
3>         object_name(major_id) as object_name,
4>         permission_name as permission,
5>         state
6> from sys.database_permissions
7> where user_name(grantee_principal_id)='user1'
8> and class_desc='object_or_column'
9> go
grantee          class             object_name      permission       state
---------------- ----------------- ---------------- ---------------- -----
user1            OBJECT_OR_COLUMN  t                INSERT           G
user1            OBJECT_OR_COLUMN  t                SELECT           G
user1            OBJECT_OR_COLUMN  t                UPDATE           G
```

revoke 及 deny 命令的使用方法与其他权限相似，这里不再赘述。

11.5.9 查询当前数据库用户具备的权限信息

可以使用系统函数 sys.fn_my_permissions()查询当前用户对指定安全对象的权限信息，使用时，需要指定安全对象名称(第 1 个参数)以及安全对象类型(第 2 个参数)，如下面示例

查询当前连接用户对 sch1 架构中的 t 表的操作权限：

```
C:\ Windows\system32>sqlcmd -Y 15 -U login1 -P login1login1
1> print db_name
2> go
1> print db_name()
2> go
law
1> print user
2> go
user1
1> select * from sys.fn_my_permissions('sch1.t','object')
2> go
entity_name     subentity_name  permission_name
--------------- --------------- ---------------
sch1.t                          SELECT
sch1.t                          INSERT
sch1.t                          ALTER
sch1.t          a               SELECT
sch1.t          a               UPDATE

(5 行受影响)
```

查询当前连接用户对 sch1 架构的权限：

```
1> select * from sys.fn_my_permissions('sch1','schema')
2> go
entity_name     subentity_name  permission_name
--------------- --------------- ---------------
sch1                            SELECT
sch1                            CREATE SEQUENCE
sch1                            ALTER

(3 行受影响)
```

以上查询结果中的 create sequence 和 alter 权限是因为对 user1 授予了数据库的 alter 权限间接授予的。

查询当前连接用户数据库层次的权限(数据库名称可以省略，指当前数据库)：

```
1> select entity_name, permission_name
2> from sys.fn_my_permissions('null','database')
3> go
entity_name                    permission_name
------------------------------ ------------------------------
database                       CREATE TABLE
database                       CREATE VIEW
database                       CREATE PROCEDURE
database                       CREATE FUNCTION
database                       CREATE RULE
database                       CREATE DEFAULT
database                       CREATE TYPE
database                       CREATE ASSEMBLY
...
database                       ALTER ANY CONTRACT
database                       ALTER ANY SERVICE
database                       ALTER ANY REMOTE SERVICE BINDI
database                       ALTER ANY ROUTE
database                       ALTER ANY FULLTEXT CATALOG
database                       ALTER ANY SYMMETRIC KEY
```

```
database                        ALTER ANY ASYMMETRIC KEY
database                        ALTER ANY CERTIFICATE
database                        SELECT
database                        ALTER ANY DATABASE DDL TRIGGER
database                        ALTER ANY DATABASE EVENT NOTIF
database                        ALTER ANY DATABASE AUDIT
database                        ALTER ANY DATABASE EVENT SESSI
database                        VIEW ANY COLUMN ENCRYPTION KEY
database                        VIEW ANY COLUMN MASTER KEY DEF
database                        ALTER

(52 行受影响)
```

以上权限，多数是由对数据库授予 alter 权限间接授予的，若撤销 alter 权限：

```
C:\Windows\system32>sqlcmd -d law
1> revoke alter from user1
2> go
1> exit
```

则剩余的权限如下：

```
C:\Windows\system32>sqlcmd -Y20 -Ulogin1 -Plogin1login1
1> select entity_name, permission_name
2> from sys.fn_my_permissions('null','database')
3> go
entity_name          permission_name
-------------------- --------------------
database             CONNECT
database             SELECT
database             VIEW ANY COLUMN ENCR
database             VIEW ANY COLUMN MAST

(4 行受影响)
```

11.6 角色

角色可以看作是权限的集合体，为了方便权限管理，可以把一些常用权限授予指定角色，然后再把角色授予相关用户，这些用户就继承了角色中的权限。数据库中会内置一些角色，用户也可以创建自定义角色。角色的权限管理与用户是相同的。

11.6.1 预置角色

Oracle 数据库的常用预置角色包括 public、dba、resource 及 connect。dba 角色包括了 system 用户的所有权限。

每个数据库用户都被赋予了 public 角色，管理员不能在数据库中删除这个角色，对用户不能授予或撤销 public 角色。数据字典视图 dba_roles 和 session_roles 未包含此角色的信息。

执行下面查询，可以得到 connect 和 resource 角色包含的权限：

```
SQL> select grantee,privilege
  2  from dba_sys_privs
  3  where grantee in('CONNECT','RESOURCE')
  4  order by grantee
  5  /

GRANTEE      PRIVILEGE
```

```
------------ ----------------------------------------
CONNECT      CREATE SESSION
CONNECT      SET CONTAINER
RESOURCE     CREATE TYPE
RESOURCE     CREATE TABLE
RESOURCE     CREATE CLUSTER
RESOURCE     CREATE OPERATOR
RESOURCE     CREATE INDEXTYPE
RESOURCE     CREATE SEQUENCE
RESOURCE     CREATE TRIGGER
RESOURCE     CREATE PROCEDURE

已选择 10 行。
```

若在 Oracle 11g 版本执行以上查询，resource 角色同样显示以上 8 种权限，但把 resource 角色赋予用户时，除了以上 8 种权限，其实还会隐含赋予 unlimited tablespace 权限。从 12c 版本开始，赋予用户 resource 角色时，不再包含 unlimited tablespace 权限。

SQL Server 包括服务器和数据库两个层次的角色，其预置服务器角色和预置数据库角色中的权限都不能改变，在官方文档中称为固定服务器角色和固定数据库角色。

所有固定服务器角色包括以下 8 种：

```
1> select name
2> from sys.server_principals
3> where type='R' and is_fixed_role=1
4> go
name
--------------------
sysadmin
securityadmin
serveradmin
setupadmin
processadmin
diskadmin
dbcreator
bulkadmin

(8 行受影响)
```

所有固定数据库角色包括以下 9 种：

```
1> select name
2> from sys.database_principals
3> where type='R' and is_fixed_role=1
4> go
name
--------------------
db_owner
db_accessadmin
db_securityadmin
db_ddladmin
db_backupoperator
db_datareader
db_datawriter
db_denydatareader
db_denydatawriter

(9 行受影响)
```

11.6.2 创建及删除角色

Oracle 创建和删除角色的命令分别为：

create role *role_name*

drop role *role_name*

对角色授予和撤销权限的语法形式与用户相同：

grant *priv1 [, priv2, priv3, ...]* to *role_name*

revoke *priv1 [, priv2, priv3, ...]* from *role_name*

对数据库用户授予角色与对用户授予或撤销权限的语法相同：

grant *role_name* to *user_name*

revoke *role_name* from *user_name*

下面命令创建了角色 role_conn_sel_emp，并对其赋予 connect 预置角色及 emp 表的查询权限：

```
SQL> create role role_conn_sel_emp;
角色已创建。
SQL> grant connect to role_conn_sel_emp;
授权成功。
SQL> grant select on scott.emp to role_conn_sel_emp;
授权成功。
```

创建 user1 用户，对其授予 role_conn_sel_emp 角色：

```
SQL> create user user1 identified by user1;
用户已创建。
SQL> grant role_conn_sel_emp to user1;
授权成功。
```

查询用户及其所属角色信息可以使用 dba_role_privs，如查询 user1 用户被授予的角色：

```
SQL> select grantee, granted_role
  2  from dba_role_privs
  3  where grantee='USER1'
  4  /

GRANTEE              GRANTED_ROLE
-------------------- --------------------------
USER1                ROLE_CONN_SEL_EMP
```

查询角色被授予的系统权限和对象权限的方法与普通用户相同，即使用 dba_sys_privs 和 dba_tab_privs，这里不再赘述。

SQL Server 2012 开始支持用户定义服务器角色，创建和删除服务器角色的语法分别为：

create server role *role_name*

drop server role *role_name*

SQL Server 创建和删除数据库角色的语法与 Oracle 相同，分别为：

create role *role_name*

drop role *role_name*

对两类角色授予和撤销权限的语法形式与用户相同：

grant *priv1 [, priv2, priv3, ...]* to *role_name*

revoke *priv1 [, priv2, priv3, ...]* from *role_name*

对登录帐号授予或撤销服务器角色使用下面语法形式：

alter server role *role_name* add/drop member *login_name*

对数据库用户授予或撤销数据库角色使用下面语法形式：

alter role *role_name* add/drop member *user_name*

上面命令的字面意思是把登录帐号或数据库用户加入角色，为了与 Oracle 的说法一致，我们依然称其为对登录帐号或数据库用户授予或撤销角色。

下面命令创建了服务器角色 role_connect，并把连接服务器和任何数据库的权限授予此角色：

```
1> use master
2> go
已将数据库上下文更改为 "master"。
1> create server role role_connect
2> go
1> grant connect sql, connect any database to role_connect
2> go
```

然后把此角色授予登录帐号 login1：

```
1> alter server role role_connect add member login1
2> go
```

下面命令把 sysadmin 固定服务器角色授予登录帐号 login1：

```
1> alter server role sysadmin add member login1
2> go
```

下面命令撤销列 login1 帐号的 role_connect 角色：

```
1> alter server role role_connect drop member login1
2> go
```

查询登录帐号的服务器角色信息，可以使用 sys.server_role_members。

重新把 role_connect 角色授予登录帐号 login1：

```
1> alter server role role_connect add member login1
2> go
```

查询服务器角色 role_connect 的成员帐号：

```
1> select suser_name(role_principal_id) as role_name,
2> suser_name(member_principal_id) as login_name
3> from sys.server_role_members
4> where suser_name(role_principal_id)='role_connect'
5> go
role_name            login_name
-------------------- --------------------
role_connect         login1
```

把查询条件稍加修改，可以查询登录帐号 login1 所属的角色：

```
1> select suser_name(member_principal_id) as login_name,
2> suser_name(role_principal_id) as role_name
3> from sys.server_role_members
4> where suser_name(member_principal_id)='login1'
5> go
login_name           role_name
-------------------- --------------------
login1               sysadmin
login1               role_connect
```

去除查询条件，即可以使用上述命令查询所有服务器角色及其成员帐号的信息。

SQL Server 创建数据库角色的语法与 Oracle 相同，下面命令创建数据库角色 role_del_t，并把 t 表的 delete 权限授予此角色：

```
1> use law
2> go
1> create role role_del_t
2> go
1> grant delete on t to role_del_t
2> go
```

然后把 role_del_t 角色授予 user1 用户：

```
1> alter role role_del_t add member user1
2> go
```

下面命令撤销列 user1 用户的 role_del_t 角色：

```
1> use law
2> go
已将数据库上下文更改为 "law"。
1> alter role role_del_t drop member user1
2> go
```

要查询数据库角色及其成员信息，可以使用 sys.database_role_members 目录视图.

下面命令查询 role_del_t 角色中的所有成员用户(查询之前请重新把 role_del_t 角色赋予 user1 用户)：

```
1> select user_name(role_principal_id) as role_name,
2> user_name(member_principal_id) as user_name
3> from sys.database_role_members
4> where user_name(role_principal_id)='role_del_t'
5> go
role_name            user_name
-------------------- --------------------
role_del_t           user1
```

把查询条件改为 where user_name(member_principal_id)='user1'，则可以查询 user1 用户所属的角色：

```
1> select user_name(member_principal_id) as user_name,
2> user_name(role_principal_id) as role_name
3> from sys.database_role_members
4> where user_name(member_principal_id)='user1'
5> go
user_name            role_name
-------------------- --------------------
user1                role_del_t
```

去除查询条件，即可以查询所有数据库角色及其成员用户的信息。

11.7 SQL Server 安全管理的几个易混淆问题

在安全管理方面，SQL Server 的结构复杂，概念繁多，有些概念甚至官方文档也未对其详细说明，本节对几个易混淆的概念予以澄清，以免误用。

11.7.1 revoke 与 deny

deny 命令拒绝通过 grant 命令授予用户的权限以及通过角色成员继承的角色权限。

revoke 命令在 grant 命令之后执行可以撤消通过 grant 命令授予用户的权限，revoke 命令并不撤消通过角色成员继承的权限。

revoke 命令在 deny 命令之后执行可以撤销 deny 命令拒绝的权限。

下面通过实验过程验证上面结论。

创建测试登录帐号 login_rd，默认数据库为 law，在 law 数据库中创建对应用户 user_rd，并创建测试表 t：

```
C:\Windows\system32>sqlcmd
1> create login login_rd with password='login_rdlogin_rd',default_database=law
2> go
1> use law
2> go
已将数据库上下文更改为 "law"。
1> create user user_rd for login login_rd
2> go
1> create table t(a int)
2> go
```

创建测试角色，并对其授予对 t 表的查询权限：

```
1> create role role_rd
2> go
1> grant select on t to role_rd
2> go
```

为了方便验证数据库用户 user_rd 的权限变化，创建一个存储过程 check_permission 执行权限验证的任务：

```
1> create proc check_permission
2> as
3> select user_name(grantee_principal_id) as grantee,
4>         object_name(major_id) as object_name,
5>         permission_name as permission,
6>         state
7> from sys.database_permissions
8> where user_name(grantee_principal_id)='user_rd'
9> and class_desc='object_or_column'
10> go
```

做好以上准备后，可以开始验证过程了，先以管理员身份连接至 law 数据库：

```
C:\Windows\system32>sqlcmd -Y 11
1> use law
2> go
已将数据库上下文更改为 "law"。
1> print user
2> go
dbo
```

对 user_rd 用户授予 t 表的查询权限，然后再执行 revoke 命令，可以发现查询权限被撤销了：

```
1> grant select on t to user_rd
2> go
1> check_permission
```

```
2> go
grantee     object_name permission  state
----------- ----------- ----------- -----
user_rd     t           SELECT      G

(1 行受影响)
1> revoke select on t from user_rd
2> go
1> check_permission
2> go
grantee     object_name permission  state
----------- ----------- ----------- -----

(0 行受影响)
```

对 user_rd 用户 t 表的查询权限执行 deny 命令，然后再执行 revoke 命令，可以发现 deny 的效果被撤销了：

```
1> deny select on t to user_rd
2> go
1> check_permission
2> go
grantee     object_name permission  state
----------- ----------- ----------- -----
user_rd     t           SELECT      D

(1 行受影响)
1> revoke select on t from user_rd
2> go
1> check_permission
2> go
grantee     object_name permission  state
----------- ----------- ----------- -----

(0 行受影响)
```

对 user_rd 用户授予 role_rd 角色及对表 t 的 select 权限：

```
C:\Windows\system32>sqlcmd -Y11 -dlaw
1> sp_addrolemember 'role_rd', 'user_rd'
2> go
1> grant select on t to user_rd
2> go
1> check_permission
2> go
grantee     object_name permission  state
----------- ----------- ----------- -----
user_rd     t           SELECT      G

(1 行受影响)
```

对表 t 的 select 权限执行 deny 命令后，可以发现 user_rd 用户没有查询 t 表的权限了：

```
1> deny select on t to user_rd
2> go
1> check_permission
2> go
grantee     object_name permission  state
----------- ----------- ----------- -----
user_rd     t           SELECT      D
```

```
(1 行受影响)
1> exit

C:\Windows\system32>sqlcmd -Y 11 -d law -U login_rd -P login_rdlogin_rd
1> select * from dbo.t
2> go
消息 229, 级别 14, 状态 5, 服务器 LAW_X240, 第 1 行
拒绝了对对象 't' (数据库 'law', 架构 'dbo')的 SELECT 权限。
```

再次执行 revoke 命令，user_rd 用户拥有 role_rd 角色中的查询权限，可对 t 表查询：

```
C:\Windows\system32>sqlcmd -Y 11 -d law
1> revoke select on t from user_rd
2> go
1> check_permission
2> go
grantee     object_name permission  state
----------- ----------- ----------- -----

(0 行受影响)
1> exit

C:\Windows\system32>sqlcmd -Y11 -dlaw -Ulogin_rd -Plogin_rdlogin_rd
1> select * from dbo.t
2> go
a
----------

(0 行受影响)
```

11.7.2 安全对象的属主(owner)

属主对其安全对象具备所有操作权限，alter authorization 命令可改变安全对象属主。

关于安全对象的属主有以下几个结论：

- 数据库的默认属主是创建这个数据库的服务器登录帐号。
- 架构的默认属主是创建这个架构的数据库用户。
- 对于表或视图等属于架构层次内的安全对象来说，其属主默认为其所属架构的属主，而不是创建这些对象的数据库用户，如果查询 sys.objects 目录视图，这些对象对应的 principal_id 列的值为 null。
- 对于架构层次内的安全对象，如果使用 alter authorization 命令显式改变了其属主，再次查询 sys.objects 目录视图，其对应的 principal_id 列则为改变后的属主 id 编号，而不再为 null。

以下几个示例说明如何查询安全对象的属主。

查询 law 数据库的属主：

```
1> select name as db_name, suser_sname(owner_sid) as db_owner
2> from sys.databases
3> where name='law'
4> go
db_name                db_owner
---------------------- ----------------------
law                    law_x240\Administrator
```

查询 sch 架构的属主：

```
1> select name as sch_name, user_name(principal_id) as owner_name
2> from sys.schemas
3> where name='sch1'
4> go
sch_name                  owner_name
------------------------- -------------------------
sch1                      dbo
```

查询 sch1 架构中的 t 表的属主：

```
1> alter authorization on sch1.t to user1
2> go
1> select name as table_name, user_name(principal_id) as table_owner
2> from sys.tables
3> where name='t'
4> and schema_name(schema_id)='sch1'
5> go
table_name                table_owner
------------------------- -------------------------
t                         user1
```

如果对 t 表未执行过 alter authorization 命令，则上面的查询结果为空。

11.7.3 安全对象的 control 权限

安全对象的 control 权限与其属主对此安全对象的权限相同，即可以对相应安全对象执行任何操作。

11.7.4 control server 权限与 sysadmin 服务器角色

control server 权限与 sysadmin 角色成员的权限相同，都可以在任何数据库中执行任意操作，但还是有一点区别，sysadmin 角色成员在数据库中的默认架构为 dbo，而具备 control server 权限的登录帐号在数据库中不存在默认架构。

具备 control server 权限的登录帐号在数据库中创建对象时，若不指定架构，则默认会创建与登录帐号名称相同的架构作为其默认架构。

11.7.5 安全对象的 take ownership 权限

假定某个安全对象 a 的属主为 user2，另外一个 user3 用户要通过执行 alter authorization 命令成为 a 的属主，则要先执行下面两个操作：

授予 user3 对 user2 的 impersonate 权限。

被授予 user3 对 a 的 take ownership 权限。

下面以改变 sch 架构的属主为例说明 take ownership 的用法，sch 架构的属主是 user2，其对应登录帐号为 login2，user3 对应的登录帐号为 login3：

以 user2 或 dbo 连接数据库执行下面操作：

```
1> grant impersonate on user::user2 to user3
2> go
1> grant take ownership on schema::sch to user3
2> go
```

然后以 user3 连接数据库，执行 alter authorization 操作：

```
C:\>sqlcmd -U login3 -P login33
1> use law
2> go
```

```
已将数据库上下文更改为 'law'。
1> alter authorization on schema::sch to user3
2> go
```

经过上述操作后，user3 也成为 sch 的属主了。

一个用户被授予某个安全对象的 take ownership 权限后，并未改变这个安全对象的属主，而是让这个用户自身有能力决定是否成为这个安全对象的属主。