第5章

数据库安全性与完整性

视频

5.1 数据库安全性概述

5.1.1 安全控制模型

数据库已在各种信息系统中得到广泛的应用，数据在信息系统中的价值越来越重要，数据库系统的安全性成为一个越来越值得关注的问题。

数据库的安全性是指在信息系统的不同层次保护数据库，防止未授权的数据访问，避免数据的泄漏、不合法的修改或对数据的破坏。安全性问题不是数据库系统所独有的，它来自各个方面，其中既有数据库本身的安全机制，如用户认证、存取权限、视图隔离、跟踪与审查、数据加密、数据完整性控制、数据访问的并发控制、数据库的备份和恢复等方面，也涉及计算机硬件系统、网络系统、操作系统、组件、Web 服务、客户端应用程序、网络浏览器等。只是在数据库系统中大量数据集中存放，而且为许多最终用户直接共享，从而使安全性问题更为突出，每一个方面产生的安全问题都可能导致出现数据库数据泄露、意外修改、丢失等后果。

在一般计算机系统中，安全措施往往是一级一级层层设置的，其安全模型如图 5.1 所示。

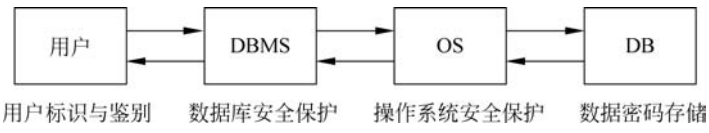

图 5.1 计算机系统的安全模型

在这个安全模型中，用户要求进入计算机时，系统首先根据输入的用户标识进行用户身份鉴定，只有合法用户才准许进入计算机系统。对已进入系统的用户，DBMS 还要设置很

多访问限制，例如 DBMS 级访问控制主要有自由存取控制（Discretionary Access Control，DAC）和强制存取控制方法（Mandatory Access Control，MAC），并只允许用户进行合法操作。操作系统一般也有自己的保护措施，它主要是基于用户访问权限的访问控制。数据最后还可以以密码形式存储到数据库中。

5.1.2　安全层次简介

在安全问题上，DBMS 应与操作系统达到某种意向，理清关系，分工协作，以加强 DBMS 的安全性。数据库系统安全保护措施是否有效是数据库系统的主要指标之一，主要包括以下几个层次的安全措施。

1. 数据库系统层次

数据库管理系统的安全保护主要表现在对数据库的存取权限控制上。同时，数据库本身的完整性问题也直接关系到数据库数据是否安全可靠。

2. 操作系统层次

操作系统的安全保护主要表现在标识、鉴别、审核用户以及隔离用户进程。

3. 网络层次

网络层次的安全性包括保密性、安全协议设计、接入控制等。

4. 物理层次

物理层次的安全性主要指物理节点保护、硬件保护等方面所采取的相应措施。

5. 人员层次

在人员层次上，主要采取用户分类、角色设定、授权等措施来防止操作人员对系统的非法访问。

5.1.3　安全标准简介

信息技术和网络空间给社会各个方面都注入了新的活力。人们在享受信息化带来的众多好处的同时，也面临着日益突出的信息安全与保密的问题。越来越多的人开始关注信息安全评估，而评估工作必须依据一定的安全标准。在一系列的安全标准中，最有影响的当推美国可信计算机系统评价标准（Trusted Computer System Evaluation Criteria，TCSEC）和信息技术安全性评估标准（Common Criteria for Information Technology Security Evaluation，CC）。

1. TCSEC

TCSEC 是指 1985 年美国国防部正式颁布的《DoD 可信计算机系统评估准则》。在 TCSEC 中，美国国防部按处理信息的等级和应采用的响应措施，将计算机安全从高到低分为：A、B、C、D 四类八个级别，共 27 条评估准则。随着安全等级的提高，系统的可信度随之增加，风险逐渐减少，如表 5.1 所示。

表 5.1 TCSEC 的等级划分

等级划分	等级名称	保护等级
D 类	最低保护等级	D 级：无保护级
C 类	自主保护级	C1 级：自主安全保护级
		C2 级：控制访问保护级
B 类	强制保护级	B1 级：标记安全保护级
		B2 级：结构化保护级
		B3 级：安全区域保护级
A 类	验证保护级别	A1 级：验证设计级
		超 A1 级

2. CC

国际《信息技术安全性评估通用准则》(简称《通用准则》,CC)是北美和欧盟联合开发一个统一的国际互认的安全标准的结果,是在美国、加拿大、欧洲等国家和地区分别自行推出的评估标准及具体实践的基础上,通过相互间的总结和互补发展起来的。目前 CC 已经基本取代了 TCSEC,成为评估信息产品安全性的主要标准。

CC 提出了目前国际上公认的表述信息技术安全性的结构,即把对信息产品的安全要求分为安全功能要求和安全保证要求。安全功能要求用以规范产品和系统的安全行为,安全保证要求解决如何正确有效地实施这些功能。安全功能要求和安全保证要求都以“类-子类-组件”的结构表述,组件是安全要求的最小构件块。

CC 的文本由三部分组成,三个部分相互依存,缺一不可。

第一部分是简介和一般模型,介绍 CC 中的有关术语、基本概念和一般模型,以及与评估有关的一些框架。

第二部分是安全功能要求,列出了一系列类、子类和组件,由 11 大类、66 个子类和 135 个组件构成。

第三部分是安全保证要求,列出了一系列保证类、子类和组件,包括 7 大类、26 个子类和 74 个组件。根据系统对安全保证要求的支持情况提出了评估保证级(Evaluation Assurance Level,EAL),EAL1～EAL7 共分为 7 级,按保证程度逐渐增高,如表 5.2 所示。

表 5.2 CC 评估保证级的划分

评估保证级	定　义	TCSEC 安全级别(近似相当)
EAL1	功能测试	
EAL2	结构测试	C1
EAL3	系统地测试和检查	C2
EAL4	系统地设计、测试和复查	B1
EAL5	半形式化设计和测试	B2
EAL6	半形式化验证的设计和测试	B3
EAL7	形式化验证的设计和测试	A1

CC 的附录部分主要介绍保护轮廓(Protection Profile,PP)和安全目标(Security Target,ST)的基本内容。

这三部分的有机结合具体体现在保护轮廓和安全目标中，CC 提出的安全功能要求和安全保证要求都可以在具体的保护轮廓和安全目标中进一步细化和扩展，这种开放式的结构更适应信息安全技术的发展。CC 的具体应用也是通过保护轮廓和安全目标这两种结构来实现的。

3. 我国的信息安全评估标准

我国在信息系统安全的研究与应用方面与其他先进国家相比有一定的差距，但近年来，国内的研究人员已经在安全操作系统、安全数据库、安全网关、防火墙、入侵检测系统等方面做了许多工作，1999 年发布的国家强制性标准《计算机信息系统安全保护等级划分准则》(GB17859—1999)为安全产品的研制提供了技术支持，也为安全系统的建设和管理提供了技术指导。CC V2.1 版于 1999 年被 ISO 采用为国际标准，2001 年被我国采用为国家标准。

5.2　Oracle 的安全机制


视频

5.2.1　用户管理

在 Oracle 中，最外层的安全措施就是让用户标识自己的名字，然后由系统进行核实。Oracle 允许用户重复标识三次，如果三次未通过，系统就自动退出。

在 Oracle 数据库系统中可以通过设置用户的安全参数维护安全性。为了防止非授权用户对数据库进行存取，在创建用户时必须使用安全参数对用户进行限制。用户的安全参数包括：用户名、口令、用户默认表空间、用户临时表空间、用户空间存取限制和用户资源存取限制。

1. 创建用户

在 Oracle 19c 中，对用户分为两类，即公有用户(Common User)和本地用户(Local User)。这样划分的目的是为了 Oracle 云平台的创建，同时两类用户的保存内存不同，其中公有用户保存在数据库容器(Container Database，CDB)中，而本地用户保存在可拔插数据库(Pluggable Database，PDB)中。一个 CDB 下会包含多个 PDB。如果是 CDB 用户，必须使用“C##”或“c##”开头；如果是 PDB 用户，则不需要使用“C##”或“c##”开头。本章主要介绍 CDB 用户。

Oracle 数据库使用 CREATE USER 命令来创建一个新的数据库用户，但是创建者必须具有 CREATE USER 系统权限。在建立用户时应该为其指定一个口令，该口令加密后存储在数据库数据字典中。当用户与数据库建立连接时，Oracle 验证用户提供的口令与存储在数据字典中的口令是否一致。

使用 SQL 命令创建用户的语法如下。

语法：

```
CREATE USER 用户名
IDENTIFIED BY 口令
[DEFAULT TABLESPACE 表空间名]
[TEMPORARY TABLESPACE 表空间名]
[ QUOTA n K | M | UNLIMITED ON tablespace_name ]
```

```
[PROFILE profile_name]
[PASSWORD EXPIRE]
[ACCOUNT {LOCK | UNLOCK}]
```

说明：

(1) 使用 IDENTIFIED BY 子句为用户设置口令，这时用户将通过数据库来进行身份认证。值得注意的是，口令区分大小写。

(2) 使用 DEFAULT TABLESPACE 子句为用户指定默认表空间。如果没有指定默认表空间，Oracle 会把 SYSTEM 表空间作为用户的默认表空间。

(3) TEMPORARY TABLESPACE 子句为用户指定临时表空间，若没有指定，TEMP 为该用户的临时表空间。

(4) QUOTA 用于指定用户在特定表空间的配额，即用户在该表空间中可以分配的最大空间。默认情况下，新建用户在任何表空间都不具有任何配额。

(5) PROFILE 用于为用户指定概要配置文件，默认值为 DEFAULT，采用系统默认的概要配置文件。

(6) PASSWORD EXPIRE 子句用于设置用户口令的初始状态为过期。当用户使用 SQL/PLUS 第一次登录数据库时，强制用户重置口令。

(7) ACCOUNT LOCK 子句用于设置用户账户的初始状态为锁定，默认为 ACCOUNT UNLOCK。当一个账号被锁定并且用户试图连接到该数据库时，会显示错误提示。

例题 5.1 创建一个 c##test 用户，口令为 c##test。该用户口令没有到期，账号也没有被锁住，默认表空间为 users，在该表空间的配额为 20MB，临时表空间为 temp。

解析：

```
CREATE USER c##test
IDENTIFIED BY c##test
DEFAULT TABLESPACE users
TEMPORARY TABLESPACE temp
QUOTA 20M ON users
ACCOUNT UNLOCK;
```

说明：当建立用户后，必须给用户授权，用户才能连接到数据库，并对数据库中的对象进行操作。只有拥有 CREATE SESSION 权限的用户才能连接到数据库。可用下列语句对 test 用户授权。

```
GRANT CREATE SESSION TO c##test;
```

2. 修改用户

建立用户时指定的所有特性都可以使用 ALTER USER 命令加以修改。使用此命令可修改用户的默认表空间、临时表空间、口令、口令期限以及加锁设置，但是不能更改用户名。执行该语句必须具有 ALTER USER 的系统权限。

修改用户的语法如下：

```
ALTER USER 用户名
IDENTIFIED  BY 口令
[DEFAULT TABLESPACE 表空间名]
```

```
[TEMPORARY TABLESPACE 表空间名]
[PASSWORD EXPIRE]
[ACCOUNT {LOCK | UNLOCK}]
```

例题 5.2　将 c##test 用户的口令修改为 Oracle19c，并且将其口令设置为到期。

解析：

```
ALTER USER c##test
IDENTIFIED BY Oracle19c
PASSWORD EXPIRE;
```

例题 5.3　修改 c##test 用户的默认表空间和账户的状态，将默认表空间改为 system，账户的状态设置为锁定状态。

解析：

```
ALTER USER c##test
DEFAULT TABLESPACE system
ACCOUNT LOCK;
```

说明：修改用户的默认表空间只影响将来建立的对象，以前建立的对象仍然存放在原来的表空间上，将来建立的对象存放到新的默认表空间。

3. 删除用户

使用 DROP USER 命令可以从数据库中删除一个用户。当一个用户被删除时，其所拥有对象也随之被删除。

删除用户的语法如下：

```
DROP USER 用户名;
```

假如用户拥有对象，必须指定 CASCADE 关键字才能删除用户，否则返回一个错误。假如指定了 CASCADE 关键字，Oracle 先删除该用户所拥有的所有对象，然后删除该用户。如果其他数据库对象（如存储过程、函数等）引用了该用户的数据库对象，则这些数据库对象将被标识为失效（INVALID）。

例题 5.4　删除 c##test 用户。

解析：

```
DROP USER c##test;
```

说明：如果我们已经在 c##test 用户下创建了相应的对象，如表、视图，那么我们在使用上述命令对用户进行删除时将出现错误，此时语句应改为：DROP USER c##test CASCADE；但是，一个连接到 Oracle 服务器的用户是不能被删除的。

4. 查询用户信息

可以通过查询数据字典视图或动态性能视图来获取用户信息。

（1）ALL_USERS：包含数据库所有用户的用户名、用户 ID 和用户创建时间。

（2）DBA_USERS：包含数据库所有用户的详细信息。

（3）USER_USERS：包含当前用户的详细信息。

（4）V$SESSION：包含用户会话信息。

(5) V$OPEN_CURSOR：包含用户执行的 SQL 语句信息。

普通用户只能查询 USER_USERS 数据字典，只有拥有 DBA 权限的用户才能查询 DBA_USERS 数据字典。

例题 5.5 查询当前用户的详细信息。

解析：

```
SELECT USERNAME,
DEFAULT_TABLESPACE,
TEMPORARY_TABLESPACE,
ACCOUNT_STATUS, EXPIRY_DATE
FROM USER_USERS;
```

程序运行效果如图 5.2 所示。

```
USERNAME
DEFAULT_TABLESPACE              TEMPORARY_TABLESPACE
------------------------------ ------------------------------
ACCOUNT_STATUS                   EXPIRY_DATE
-------------------------------- -------------
SYSTEM
SYSTEM                          TEMP
OPEN                             21-8月 -20
```

图 5.2 当前用户的详细信息

例题 5.6 查询数据库中所有用户名、默认表空间和账户的状态。

解析：

```
SELECT USERNAME,
DEFAULT_TABLESPACE,
ACCOUNT_STATUS
FROM DBA_USERS;
```

5.2.2 权限管理

创建了用户，并不意味着用户就可以对数据库随心所欲地进行操作。创建用户账号也只是意味着用户具有了连接、操作数据库的资格，用户对数据库进行的任何操作，都需要具有相应的操作权限。

权限是在数据库中执行一种操作的权力。在 Oracle 数据库中，根据系统管理方式的不同，可以将权限分为两类，即系统权限和对象权限。

1. 系统权限

系统权限是指在系统级控制数据库的存取和使用的机制，系统权限决定了用户是否可以连接到数据库以及在数据库中可以进行哪些操作。可以将系统权限授予用户、角色、PUBLIC 用户组。由于系统权限有较大的数据库操作能力，因此应该只将系统权限授予值得信赖的用户。

系统权限可划分成下列三类。

第一类：允许在系统范围内操作的权限。如 CREATE SESSION、CREATE TABLESPACE

等与用户无关的权限。

第二类：允许在用户自己的账号内管理对象的权限。如 CREATE TABLE 等建立、修改、删除指定对象的权限。

第三类：允许在任何用户账号内管理对象的权限。如 CREATE ANY TABLE 等带 ANY 的权限，允许用户在任何用户账号下建表。

常见的系统权限如表 5.3 所示。

表 5.3　Oracle 常用的系统权限

系 统 权 限	描　　述
CREATE SESSION	创建会话
CREATE SEQUENCE	创建序列
CREATE SYNONYM	创建同名对象
CREATE TABLE	在用户模式中创建表
CREATE ANY TABLE	在任何模式中创建表
DROP TABLE	在用户模式中删除表
DROP ANY TABLE	在任何模式中删除表
CREATE PROCEDURE	创建存储过程
EXECUTE ANY PROCEDURE	执行任何模式的存储过程
CREATE USER	创建用户
DROP USER	删除用户
CREATE VIEW	创建视图

1）系统权限的授权

使用 GRANT 命令可以将系统权限授予给一个用户、角色或 PUBLIC。给用户授予系统权限应该根据用户身份的不同进行。如数据库管理员用户应该具有创建表空间、修改数据库结构、修改用户权限、对数据库任何模式中的对象进行管理的权限；而数据库开发人员具有在自己模式下创建表、视图、索引、同义词、数据库链接等权限。

语法格式如下：

```
GRANT {系统权限 | 角色} [,{系统权限 | 角色}]…
TO {用户 | 角色 | PUBLIC} [,{用户 | 角色 | PUBLIC}]…
[WITH ADMIN OPTION]
```

说明：

（1）PUBLIC 是创建数据库时自动创建的一个特殊的用户组，数据库中所有的用户都属于该用户组。如果将某个权限授予 PUBLIC 用户组，则数据库中所有用户都具有该权限。

（2）WITH ADMIN OPTION 表示允许得到权限的用户进一步将这些权限或角色授予给其他的用户或角色。

例题 5.7　先创建用户 c##user1，再为用户 c##user1 授予 CREATE SESSION 系统权限，保证用户 c##user1 成功登录。

解析：

创建用户 c##user1，结果如图 5.3 所示。

```
SQL> CREATE USER c##user1
  2  IDENTIFIED BY c##user1
  3  DEFAULT TABLESPACE users
  4  TEMPORARY TABLESPACE temp
  5  QUOTA 20M ON users
  6  ACCOUNT UNLOCK;

用户已创建。
```

图 5.3　用户 c##user1 的创建

此时登录时，系统会拒绝并给出如下提示信息，如图 5.4 所示。

```
SQL> conn c##user1
输入口令:
ERROR:
ORA-01045: 用户 C##USER1 没有 CREATE SESSION 权限; 登录被拒绝


警告: 您不再连接到 ORACLE。
```

图 5.4　用户 c##user1 登录失败

通过授权解决用户登录失败的问题，程序如下：

```
GRANT CREATE SESSION
TO c##user1;
```

为用户 c##user1 授予权限后，再次登录的结果如图 5.5 所示。

```
SQL> CONN c##user1
输入口令:
已连接。
```

图 5.5　用户 c##user1 登录成功

例题 5.8　为用户 c##user1 授予 CREATE TABLE 系统权限。

解析：

在例题 5.7 中，用户 c##user1 已经创建成功，并且能够成功登录数据库。在用户 c##user1 中创建一张基本表，结果如图 5.6 所示。

当前用户 c##user1 不具有创建表的权限，所以创建失败了。在 SYSTEM 用户下，给用户 c##user1 授权，程序如下：

```
GRANT CREATE TABLE
TO c##user1;
```

为用户 c##user1 授予权限后，在用户 c##user1 中再次创建基本表，结果如图 5.7 所示。

```
SQL> CREATE TABLE student
  2  (sno CHAR(8),
  3   sname VARCHAR2(20),
  4   age NUMBER);
CREATE TABLE student
*
第 1 行出现错误:
ORA-01031: 权限不足
```

图 5.6　创建基本表失败

```
SQL> CREATE TABLE student
  2  (sno CHAR(8),
  3   sname VARCHAR2(20),
  4   age NUMBER);

表已创建。
```

图 5.7　创建基本表成功

例题 5.9　为用户 c##user1 授予 CREATE VIEW 系统权限，允许用户 c##user1 将该权限再授予给其他用户。

解析：

```
GRANT CREATE VIEW
TO c##user1
WITH ADMIN OPTION;
```

2）系统权限的回收

数据库管理员或系统权限传递用户可以将用户所获得的系统权限回收。系统权限回收使用 REVOKE 命令可以从用户或角色上回收系统权限。

语法格式如下：

```
REVOKE {系统权限 | 角色} [,{系统权限 | 角色}]…
FROM {用户名 | 角色 | PUBLIC} [,{用户名 | 角色 | PUBLIC}]…
```

说明：

（1）多个管理员授予用户同一个系统权限后，其中一个管理员回收其授予该用户的系统权限时，不影响该用户从其他管理员处获得系统权限。

（2）系统权限授权语句中 WITH ADMIN OPTION 从句给了受权者将此权限再授予给另一个用户或 PUBLIC 的权利。但是当一个系统权限回收时不会有级联影响，不管在权限授予时是否带 WITH ADMIN OPTION 从句。

例题 5.10　回收用户 c##user1 的 CREATE VIEW 系统权限。

解析：

```
REVOKE CREATE VIEW
FROM c##user1;
```

2. 对象权限

对象权限是指在对象级控制数据库的存取和使用的机制，用于设置一个用户对其他用户的表、视图、序列、过程、函数、包的操作权限。对于不同类型的对象，有不同类型的对象权限。对于有些模式对象，如聚集、索引、触发器、数据库链接等没有相关的对象权限，这些权限由系统进行控制。

补充：模式(Schema)对象是具有拥有者的对象(如表 HR.TABLE1 是用户 HR 拥有的，名为 TABLE1 的表)。数据库还包含非模式对象。非模式对象与用户无关，在某些情况下，非模式对象是用户系统拥有的对象，并且可以被所有用户访问，这种非模式对象包括公共的同义词与公共的数据库链接，所有用户不需要考虑任何权限因素就能够使用这些对象。

Oracle 提供的对象权限如表 5.4 所示。

表 5.4　Oracle 提供的对象权限

对象权限	TABLE	COLUMN	VIEW	SEQUENCE	PROCEDURE/FUNCTION/PACKAGE
ALTER	√			√	
DELETE	√		√		
EXECUTE					√
INDEX	√				
INSERT	√	√	√		

续表

对象权限	TABLE	COLUMN	VIEW	SEQUENCE	PROCEDURE/FUNCTION/PACKAGE
REFERENCES	√	√			
SELECT	√		√	√	
UPDATE	√	√	√		
READ					

1) 对象权限的授权

使用 GRANT 命令可以将对象权限授予给一个用户、角色或 PUBLIC。

语法格式如下：

```
GRANT { 对象权限 [(列名 1 [,列名 2…])]
[,对象权限 [(列名 1 [,列名 2…])]]…|ALL}
ON  对象名
TO {用户名 | 角色名 | PUBLIC} [,{用户名|角色名|PUBLIC}]…
[WITH GRANT OPTION]
```

说明：WITH GRANT OPTION 表示允许得到权限的用户进一步将这些权限授予给其他的用户或角色。

例题 5.11 用户 system 将学生表(student)的 SELECT 权限和属性列 sname、age 上的 UPDATE 权限授予给用户 c##user1，并且允许用户 c##user1 再将这些对象权限授予给其他用户。

解析：

授权前，用户 c##user1 的权限测试结果如图 5.8 所示。

```
SQL> CONN c##user1
输入口令:
已连接。
SQL>  SELECT * FROM system.stduent WHERE sno='10172001';
 SELECT * FROM system.stduent WHERE sno='10172001'
                       *
第 1 行出现错误:
ORA-00942: 表或视图不存在


SQL> UPDATE system.student SET age=age+2 WHERE sno='10172001';
UPDATE system.student SET age=age+2 WHERE sno='10172001'
             *
第 1 行出现错误:
ORA-00942: 表或视图不存在
```

图 5.8 授权前用户 c##user1 的权限测试

在 system 用户下，通过给用户 c##user1 授权，解决上述问题，程序如下：

```
GRANT SELECT,UPDATE(sname,age)
ON student
TO c##user1
WITH GRANT OPTION;
```

授权后，用户 c##user1 的权限测试结果如图 5.9 所示。

说明：假如用户拥有了一个对象，他就自动地获得了该对象的所有权限。对象拥有者

```
SQL> CONN c##user1
输入口令:
已连接。
SQL> SELECT * FROM system.student WHERE sno='10172001';

SNO      SNAME      SEX        AGE DEPT
-------- ---------- ---- ---------- ----------
10172001 陈一       男           17 计算机系

SQL> UPDATE system.student SET age=age+2 WHERE sno='10172001';

已更新 1 行。

SQL>  SELECT * FROM system.student WHERE sno='10172001';

SNO      SNAME      SEX        AGE DEPT
-------- ---------- ---- ---------- ----------
10172001 陈一       男           19 计算机系
```

图 5.9　授权后用户 user1 的权限测试

可以将自己对象的操作权授予给别人。例如，用户 system 可以将 SELECT、INSERT、UPDATE、DELETE 等权限授予给其他用户。

2）对象权限的回收

通过使用 REVOKE 命令可以实现权限的回收。

语法格式如下：

```
REVOKE {对象权限  [,对象权限]… | ALL [PRIVILEGES]}
FROM {用户名 | 角色 | PUBLIC} [,{用户名 | 角色 | PUBLIC}]…
[RESTRICT | CASCADE]
```

说明：

（1）ALL 用于回收授予给用户的所有对象权限。

（2）可选项[RESTRICT | CASCADE]中，CASCADE 表示回收权限时要引起级联回收。即从用户 A 回收权限时，要把用户 A 转授出去的同样的权限同时回收。RESTRICT 表示，当不存在级联连锁回收时，才能回收权限，否则系统拒绝回收。

（3）当使用 WITH GRANT OPTION 从句授予对象权限时，一个对象权限回收时存在级联影响。

例题 5.12　用户 system 从用户 c##user1 中回收学生表(student)上 SELECT 权限。

解析：

回收权限前，测试用户 c##user1 的权限结果如图 5.10 所示。

```
SQL> CONN c##user1
输入口令:
已连接。
SQL> SELECT * FROM system.student WHERE sno='10172002';

SNO      SNAME      SEX        AGE DEPT
-------- ---------- ---- ---------- ----------
10172002 姚二       女           20 计算机系
```

图 5.10　回收权限前测试用户 c##user1 的权限

在用户 system 下回收用户 c##user1 对学生表 student 的 SELECT 权限，程序如下：

```
REVOKE SELECT
```

```
ON student
FROM c##user1;
```

回收权限后,测试用户 c##user1 的权限结果如图 5.11 所示。

```
SQL> CONN c##user1
输入口令:
已连接。
SQL> SELECT * FROM system.student WHERE sno='10172002';
SELECT * FROM system.student WHERE sno='10172002'
                    *
第 1 行出现错误:
ORA-01031: 权限不足
```

图 5.11 回收权限后测试用户 c##user1 的权限

说明:多个管理员授予用户同一个对象权限后,其中一个管理员回收其授予该用户的对象权限时,不影响该用户从其他管理员处获得的对象权限。如果一个用户获得的对象权限具有传递性(授权时使用了 WITH GRANT OPTION 子句),并且给其他用户授权,那么该用户的对象权限被回收后,其他用户的对象权限也被回收。

3. 查询各种权限

可以通过数据字典视图查询数据库相应权限信息。对象权限有关的数据字典视图如表 5.5 所示。

表 5.5 对象权限有关的数据字典视图

数据字典视图	描 述
DBA_TAB_PRIVS	包含数据库所有对象的授权信息
ALL_TAB_PRIVS	包含数据库所有用户和 PUBLIC 用户组的对象授权信息
USER_TAB_PRIVS	包含当前用户对象的授权信息
DBA_COL_PRIVS	包含数据库中所有字段已授予的对象权限信息
ALL_COL_PRIVS	包含所有字段已授予的对象权限信息,该用户或 PUBLIC 是被授予者
USER_COL_PRIVS	包含当前用户所有字段已授予的对象权限信息
DBA_SYS_PRIVS	包含授予用户或角色的系统权限信息
USER_SYS_PRIVS	包含授予当前用户的系统权限信息

例题 5.13 查询当前用户 system 所具有的权限。

解析:

```
SELECT username,privilege,admin_option FROM user_sys_privs;
```

程序运行结果如图 5.12 所示。

USERNAME	PRIVILEGE	ADM
SYSTEM	CREATE TABLE	NO
SYSTEM	SELECT ANY TABLE	NO
SYSTEM	DEQUEUE ANY QUEUE	YES
SYSTEM	GLOBAL QUERY REWRITE	NO
SYSTEM	ENQUEUE ANY QUEUE	YES
SYSTEM	CREATE MATERIALIZED VIEW	NO
SYSTEM	UNLIMITED TABLESPACE	NO
SYSTEM	MANAGE ANY QUEUE	YES

图 5.12 用户 system 所具有的权限

5.2.3　角色管理

数据库的用户通常有几十个、几百个，甚至成千上万个。如果管理员为每个用户授予或者撤销相应的系统权限和对象权限，则工作量是非常庞大的。为简化权限管理，Oracle 提供了角色的概念。

角色是具有名称的一组相关权限的集合，即将不同的权限集合在一起就形成了角色。可以使用角色为用户授权，同样也可以撤销角色。由于角色集合了多种权限，所以当为用户授予角色时，相当于为用户授予了多种权限。这样就避免了向用户逐一授权，从而简化了用户权限的管理。

Oracle 中的角色可以分为预定义角色和自定义角色两类。当数据库创建时，会自动为数据库预定义一些角色，这些角色主要用来限制数据库管理系统权限。此外，用户也可以根据自己的需求，将一些权限集中到一起，建立用户自定义的角色。

1. 预定义角色

预定义角色是在数据库安装后，系统自动创建的一些常用的角色。预定义角色的细节可以从 DBA_SYS_PRIVS 数据字典视图中查询到。表 5.6 列出了几个常见的预定义角色。

表 5.6　Oracle 常用的预定义角色

角　色　名	描　　述
CONNECT	连接到数据库的权限，建立数据库链路、序列生成器、同义词、表、视图以及修改会话的权限
RESOURCE	建立表、序列生成器，以及建立过程、函数、包、数据类型、触发器的权限
DBA	带 WITH ADMIN OPTION 选项的所有系统权限可以被授予给数据库中其他用户或角色，DBA 角色拥有最高级别的权限
EXP_FULL_DATABASE	使用 EXPORT 工具执行数据库完全卸出和增量卸出的权限
IMP_FULL_DATABASE	使用 IMPORT 工具执行数据库完全装入的权限，这是一个功能非常强大的角色

2. 用户自定义角色

1）创建角色

使用 CREATE ROLE 命令可以创建角色，角色是属于整个数据库的，而不属于任何用户。当创建一个角色时，该角色没有相关的权限，系统管理员必须将合适的权限授予角色。此时，角色才是一组权限的集合。

语法格式如下：

```
CREATE ROLE 角色名 [NOT IDENTIFIED | IDENTIFIED {BY 口令}]
```

说明：在 Oracle 19c 中的 CDB 下创建角色时，角色名称必须以“C##”或“c##”开头，否则会出现错误提示消息“公共用户名或角色名无效”。

例题 5.14　建立一个带口令 Oracle19c 的角色 c##student_role。

解析：

```
CREATE ROLE c##student_role IDENTIFIED BY Oracle19c;
```

2）修改角色

语法格式如下：

```
ALTER ROLE role_name [ NOT IDENTIFIED ][ IDENTIFIED BY password ];
```

使用 ALTER ROLE 命令可以修改角色的口令，但不能修改角色名。

例题 5.15 修改角色 c##student_role，使其没有口令。

解析：

```
ALTER ROLE c##student_role NOT IDENTIFIED;
```

3）授予角色权限

创建完角色后需要给角色授权，授权后的角色才是一组权限的集合。在数据库运行过程中，可以为角色增加权限，也可以回收其权限。

例题 5.16 将用户 system 中学生表(student)的 SELECT、UPDATE 和 DELETE 权限的集合授予角色 c##student_role。

解析：

```
GRANT SELECT,UPDATE,DELETE
ON student
TO c##student_role;
```

3. 给用户或角色授予角色

可以使用 GRANT 语句将角色授予用户或其他角色。

语法格式如下：

```
GRANT role_list TO user_list|role_list;
```

例题 5.17 将角色 c##student_role 授予给用户 c##user1。

解析：

角色授予前，用户 c##user1 的权限测试结果如图 5.13 所示。

```
SQL> CONN c##user1
输入口令:
已连接。
SQL> select * from system.student where dept='管理系';
select * from system.student where dept='管理系'
                    *
第 1 行出现错误:
ORA-01031: 权限不足
```

图 5.13 角色授予前用户 c##user1 的权限测试

将角色 c##student_role 授予给用户 c##user1 后，用户 c##user1 就具有了相应的权限，程序如下：

```
GRANT c##student_role TO c##user1;
```

授权后，测试结果如图 5.14 所示。

```
SQL> CONN c##user1
输入口令:
已连接。
SQL> select * from system.student where dept='管理系';

SNO      SNAME      SEX          AGE DEPT
-------- ---------- ---- ---------- ----------
10172009 张九       女           18 管理系
10172010 孙十       女           21 管理系
```

图 5.14　角色授予后用户 c##user1 的权限测试

4. 从用户或角色回收角色

可以使用 REVOKE 语句从用户或其他角色回收角色。

语法格式如下：

```
REVOKE role_list FROM user_list|role_list;
```

例题 5.18　将角色 c##student_role 从用户 c##user1 回收。

解析：

角色回收前，用户 c##user1 的权限测试结果如图 5.15 所示。

```
SQL> SHOW USER
USER 为 "C##USER1"
SQL> SELECT * FROM system.student WHERE age<20;

SNO      SNAME      SEX          AGE DEPT
-------- ---------- ---- ---------- ----------
10172001 陈一       男           19 计算机系
10172003 张三       女           19 计算机系
10172006 赵六       男           19 日语系
10172009 张九       女           18 管理系
```

图 5.15　角色回收前用户 c##user1 的权限测试

将角色 c##student_role 从用户 c##user1 中回收，程序如下：

```
REVOKE c##student_role FROM c##user1;
```

角色回收后，用户 c##user1 就失去了相应的权限，测试结果如图 5.16 所示。

```
SQL> SHOW USER
USER 为 "C##USER1"
SQL>  SELECT * FROM system.student WHERE age<20;
 SELECT * FROM system.student WHERE age<20
                      *
第 1 行出现错误:
ORA-01031: 权限不足
```

图 5.16　角色回收后用户 c##user1 的权限测试

5. 删除角色

使用 DROP ROLE 命令可以删除角色。即使此角色已经被授予给一个用户，数据库也允许用户删除该角色。

例题 5.19　从数据库中删除 c##student_role 角色。

解析：

```
DROP ROLE c##student_role;
```

6. 查询角色信息

可以通过数据字典视图或动态性能视图获取数据库角色相关信息。与角色有关的数据字典视图如表 5.7 所示。

表 5.7 与角色有关的数据字典视图

数据字典视图	描 述
DBA_ROLES	包含数据库中所有角色及其描述
DBA_ROLE_PRIVS	包含为数据库中所有用户和角色授予的角色信息
USER_ROLE_PRIVS	包含为当前用户授予的角色信息
ROLE_ROLE_PRIVS	为角色授予的角色信息
ROLE_SYS_PRIVS	为角色授予的系统权限信息
ROLE_TAB_PRIVS	为角色授予的对象权限信息
SESSION_PRIVS	当前会话所具有的系统权限信息
SESSION_ROLES	当前会话所具有的角色信息

例题 5.20 查询当前用户 system 所具有的角色。

解析：

```
SELECT * FROM user_role_privs;
```

程序运行效果如图 5.17 所示。

```
USERNAME
------------------------------------------------------------
GRANTED_ROLE
                    ADM DEL DEF OS_ COM INH
------------------------------------------------------------
SYSTEM
AQ_ADMINISTRATOR_ROLE
                    YES NO  YES NO  YES NO

SYSTEM
DBA
                    NO  NO  YES NO  YES NO
```

图 5.17 用户 system 拥有的角色

例题 5.21 查询角色 exp_full_database 所拥有的权限。

解析：

```
SELECT * FROM role_sys_privs WHERE role = 'EXP_FULL_DATABASE';
```

程序运行效果如图 5.18 所示。

5.2.4 视图机制

视图是数据库系统提供给用户以多种角度观察数据库中数据的重要机制，是从一个或几个基本表(或视图)导出的表，它与基本表不同，是一个虚表。数据库中只存放视图的定义，而不存放视图对应的数据，这些数据仍存放在原来的基本表中。

```
ROLE
--------------------------------------------------------------
PRIVILEGE                                ADM COM INH
---------------------------------------- --- --- ---
EXP_FULL_DATABASE
READ ANY FILE GROUP                      NO  YES NO

EXP_FULL_DATABASE
EXECUTE ANY TYPE                         NO  YES NO

EXP_FULL_DATABASE
SELECT ANY SEQUENCE                      NO  YES NO

EXP_FULL_DATABASE
ADMINISTER SQL MANAGEMENT OBJECT         NO  YES NO

EXP_FULL_DATABASE
ANALYZE ANY                              NO  YES NO

EXP_FULL_DATABASE
EXECUTE ANY PROCEDURE                    NO  YES NO

EXP_FULL_DATABASE
SELECT ANY TABLE                         NO  YES NO

EXP_FULL_DATABASE
CREATE TABLE                             NO  YES NO

EXP_FULL_DATABASE
CREATE SESSION                           NO  YES NO

EXP_FULL_DATABASE
ADMINISTER RESOURCE MANAGER              NO  YES NO

EXP_FULL_DATABASE
RESUMABLE                                NO  YES NO

EXP_FULL_DATABASE
EXEMPT REDACTION POLICY                  NO  YES NO

EXP_FULL_DATABASE
FLASHBACK ANY TABLE                      NO  YES NO

EXP_FULL_DATABASE
BACKUP ANY TABLE                         NO  YES NO
```

图 5.18　查看角色 exp_full_database 的权限

从某种意义上讲，视图就像一个窗口，通过它可以看到数据库中自己感兴趣的数据及其变化。进行存取权限控制时，可以为不同的用户定义不同的视图，把访问数据的对象限制在一定的范围内，也就是说，通过视图机制把要保密的数据对无权存取的用户隐藏起来，从而对数据提供一定程度的安全保护。视图的创建与应用已经在第 3 章讲解过，这里就不再赘述。

5.2.5　审计

Oracle 数据库的审计功能与 SQL Server 相比更加灵活。Oracle 19c 提供了细粒度的审计，允许用户定义审计策略，可以对对象、权限、SQL 语句的类型等进行审计，审计结果被存储在 SYS 用户的数据库字典中，数据库管理员可以查询该字典，从而获取审计结果。用户还可以开启报警功能，这样管理员可以在出现安全问题时迅速接到通知。

Oracle 数据库中的审计大概可以分为以下几种类型。

1. 语句审计

按照语句类型审计 SQL 语句,而不论访问何种特定的模式对象。也可以在数据库中指定一个或多个用户,针对特定的语句审计这些用户。

2. 权限审计

审计系统权限,例如 CREATE TABLE 或 ALTER INDEX。和语句审计一样,权限审计可以指定一个或多个特定的用户作为审计的目标。

3. 模式对象审计

审计特定模式对象上运行的特定语句(例如,学生表上的 UPDATE 语句)。模式对象审计总是应用于数据库中的所有用户。

4. 细粒度的审计

根据访问对象的内容来审计表访问和权限。使用程序包 DBMS_FGA 来建立特定表上的策略。

Oracle 中的 AUDIT 语句用来设置审计功能,NOAUDIT 语句用来取消审计功能。

例题 5.22 对修改学生表(student)的表结构或数据的操作进行审计。

解析:

```
AUDIT ALTER,UPDATE
ON student;
```

例题 5.23 取消对学生表 student 修改表结构和修改表数据的审计。

解析:

```
NOAUDIT ALTER,UPDATE
ON student;
```

5.2.6 数据加密

通过数据加密可以提高存储数据的安全性,Oracle 数据库加密功能的实现由数据库平台提供的软件包来支持。Oracle 提供了特殊 DBMS_OBFUSCATION_TOOLKIT 包、DBMS_CRYPTO 包等用于数据加密/解密,同时支持数据加密算法(Data Encryption Algorithm,DES)、高级加密标准(Advanced Encryption Standard,AES)等多种加密/解密算法。

在数据加密中,密钥的存储和管理是非常重要的,它直接影响到数据加密的安全性。但是,在数据库管理系统内核层加密策略中,并没有提供密钥存储的方法,这也是在以 Oracle 提供的安全包为基础制定加密策略时最难解决的部分。在制定密钥的存储和管理方案时,要确保以下两点:

(1) 密钥存储,足够可靠,以确保能够保护数据;

(2) 要保证合法用户且只有合法用户可以获取密钥。

Oracle 数据库文件加密中,密钥仍然是以数据表形式存放在数据库中。如果密钥以明文形式存放在数据库中,那么攻击者只要进入数据库系统中,他就很容易找到破解密文的密

钥。如果密钥以密文形式存放在数据库中，那么加密密钥的密钥如何存放就成了需要解决的新问题。常用的解决方法是采用多级密钥存储管理，把用户密钥与数据密钥结合使用，提高数据库加密的安全性。任何加密策略的安全性都依赖于密钥的安全，但是 Oracle 数据库系统提供的加密方案中，并没有给出密钥存储与管理的安全方法。要想提高数据库加密系统的安全性，还需要辅助其他的外部手段来实现密钥的安全存储与管理。

5.3　数据库完整性控制

5.3.1　完整性基本含义

数据库的安全性和完整性是数据库安全保护的两个不同的方面。数据库的安全性保护数据库免受不合法用户的故意破坏，数据库的完整性保护数据库免受合法用户的无意破坏。从数据库的安全保护角度来讲，完整性和安全性是密切相关的。

数据库的完整性的基本含义是指数据库中数据的正确性、有效性和相容性，其主要目的是防止错误的数据进入数据库。正确性是指数据的合法性，例如数值型数据只能含有数字而不能含有字母。有效性是指数据是否属于所定义域的有效范围。相容性是指表示同一事实的两个数据应当一致，不一致即是不相容。

5.3.2　完整性约束条件

数据库系统是对现实系统的模拟，现实系统中存在各种各样的规章制度，以保证系统正常、有序地运行。许多规章制度可转化为对数据的约束，例如，单位人事制度中对职工的退休年龄会有规定，一个部门的主管不能在其他部门任职，职工工资只能涨不能降等。对数据库中的数据设置某些约束机制，这些添加在数据上的语义约束条件称为数据库完整性约束条件，简称“数据库的完整性”。

SQL 中使用了一系列的概念来描述完整性，包括实体完整性、参照完整性和用户定义完整性。这些完整性一般由 SQL 的 DDL 语句来实现，它们作为模式的一部分存入数据字典中。

一般一条完整性规则可以用一个五元组(D,O,A,C,P)来表示，其中：

(1) D(Data)表示约束作用的对象。

(2) O(Operation)表示触发完整性检查的数据库操作，即当用户发出什么操作请求时需要检查该完整性规则，是立即检查还是延迟检查。

(3) A(Assertion)表示数据对象必须满足的断言或语义约束，这是规则的主体。

(4) C(Condition)表示选择 A 作用的数据对象值的谓词。

(5) P(Procedure)表示违反完整性规则时触发的过程。

5.3.3　完整性控制机制

DBMS 必须提供一种机制来检查数据库中数据的完整性，看其是否满足语义规定的条

件,这种机制称为“完整性检查”。为此,数据库管理系统的完整性控制机制应具有三个方面的功能,来防止合法用户在使用数据库时,向数据库注入不合法或不合语义的数据。

(1) 定义功能:提供定义完整性约束条件的机制。

(2) 验证功能:检查用户发出的操作请求是否违背了完整性约束条件。

(3) 处理功能:如果发现用户的操作请求使数据违背了完整性约束条件,则采取一定的动作来保证数据的完整性。

目前,关系数据库系统都提供了定义和检查实体完整性、参照完整性和用户定义完整性的功能。违反实体完整性规则和用户定义完整性规则的操作,一般采用拒绝执行的方式进行处理;对于违反参照完整性的操作,并不都是拒绝执行,也可以接受这个操作,同时执行一些附加的操作,以保证数据库的状态正确。

5.4 实验

5.4.1 实验 1 用户管理

1. 实验目的

(1) 掌握数据库用户的创建方法。

(2) 掌握数据库用户的修改方法。

(3) 掌握数据库用户的删除方法。

2. 实验内容

(1) 创建一个 c##test2 用户,密码为 c##test2。默认表空间为 system,在该表空间的配额为 10MB。使用新创建的用户 c##test2 登录数据库,如果不能立即登录,出现错误提示信息,请给出理由。

(2) 创建一个 c##test3 用户,密码为 c##test3。默认表空间为 users,在该表空间的配额为 20MB,临时表空间为 temp。该用户的口令初始为过期,账户初始设置为锁定状态。

(3) 修改 c##test3 用户,将密码改为 c##tiger,默认表空间改为 system,账户状态设置为解锁状态。

(4) 创建一个 c##test4 用户,密码为 c##test4。账户初始设置为锁定状态。

(5) 删除 c##test4 用户。

3. 考核标准

本实验为必做实验,要求学生在课堂上独立完成。根据题目要求,按照实验步骤完成相应实验内容。用户管理的语句无语法错误、书写规范、运行结果正确为优秀;如果出现错误,根据错误个数以及难易程度灵活给分。

5.4.2 实验 2 权限管理

1. 实验目的

(1) 掌握权限的授予方法。

(2) 掌握权限的回收方法。

2. 实验内容

(1) 为实验1中创建的用户c##test2授予CREATE SESSION、CREATE TABLE和CREATE VIEW系统权限,并且允许用户c##test2将相关权限授予给其他用户。

(2) 由用户c##test2将CREATE TABLE系统权限授予给用户c##test3。

(3) 回收用户c##test2的CREATE VIEW系统权限。

(4) 将用户system下员工表(emp)的SELECT和INSERT权限授予给用户c##test2。

(5) 从用户c##test2处收回对员工表(emp)的INSERT权限。

3. 考核标准

本实验为必做实验,要求学生在课堂上独立完成。根据题目要求,按照实验步骤完成相应实验内容。权限管理的语句无语法错误、书写规范、运行结果正确为优秀;如果出现错误,根据错误个数以及难易程度灵活给分。

5.4.3　实验3　角色管理

1. 实验目的

(1) 掌握角色的创建和修改方法。

(2) 掌握角色的授予方法。

(3) 掌握角色的删除方法。

2. 实验内容

(1) 建立一个不带口令的角色c##emp_role。

(2) 将用户system下员工表(emp)的SELECT和UPDATE权限授予给角色c##emp_role。

(3) 将角色c##emp_role授权给用户c##test2。

(4) 从c##test2用户回收c##emp_role角色。

(5) 删除角色c##emp_role。

3. 考核标准

本实验为必做实验,要求学生在课堂上独立完成。根据题目要求,按照实验步骤完成相应实验内容。角色管理的语句无语法错误、书写规范、运行结果正确为优秀;如果出现错误,根据错误个数以及难易程度灵活给分。

5.5　本章小结

本章首先介绍了数据库安全性的含义、计算机系统的安全模型、五个层次的安全措施和安全标准。

其次,介绍了Oracle数据库的安全机制,包括用户管理、权限管理、角色管理、视图机制、审计功能和数据加密。

最后,介绍了数据库的完整性控制,包括完整性的基本含义、约束条件和控制机制。

5.6 课后习题

1. 以下不是创建用户过程中必要信息的是(　　)。

A. 用户名　　B. 用户权限　　C. 临时表空间　　D. 口令

2. SQL 的 GRANT 和 REVOKE 语句主要用来进行授权与回收授权,其主要目的是用来维护数据库的(　　)。

A. 完整性　　B. 可靠性　　C. 安全性　　D. 一致性

3. 当对用户授予系统权限时,使用(　　)从句表示允许得到权限的用户进一步将这些权限授予其他的用户。

A. WITH ADMIN OPTION　　B. WITH REVOKE OPTION

C. WITH GRANT OPTION　　D. WITH USER OPTION

4. 下列(　　)权限不是用户权限。

A. SELECT　　B. INSERT　　C. UPDATE　　D. CREATE

5. 关于角色的说法不正确的是(　　)。

A. 将角色授予用户使用 GRANT 命令

B. 角色一旦授予,不能收回

C. 角色是属于整个数据库的,而不属于任何用户

D. 删除角色,使用 DROP 命令

6. 创建一个用户 c##test_user,密码为 Oracle19c,用户的默认表空间为 users,该用户的口令没有到期,账户被锁定,完成以下操作。

(1) 修改用户 c##test_user,使其账户解锁。

(2) 将用户 system 下的课程表(course)的查询权限和删除权限授予给用户 c##test_user,用户 c##test_user 同时获得将这些权限转授给其他用户的权限。

(3) 将用户 system 下的课程表(course)的课程名称属性列的修改权限授予给用户 c##test_user。

(4) 从用户 c##test_user 处收回对课程表(course)的删除权限,若用户 c##test_user 已经把获得的删除权限转授给其他用户,则需要级联收回。

(5) 建立一个不带口令的角色 c##test_role。

(6) 将用户 system 下部门表(dept)的查询权限和更新权限授予给角色 c##test_role。

(7) 将角色 c##test_role 授权给用户 c##test_user。

(8) 从所有用户身上回收角色 c##test_role。

(9) 删除角色 c##test_role。

(10) 删除用户 c##test_user。