第5章　数据库的安全性

数据或信息是现代信息社会的五大“经济要素”(人、财、物、信息、技术)之一,是与财、物同等重要(有时甚至更重要)的资产。企业数据库中的数据对于企业是至关重要的,尤其是一些敏感性的数据,必须加以保护,以防止故意的破坏或改变、未经授权的存取和非故意的损害。其中,非故意的损害属于数据完整性和一致性保护问题,而故意的破坏或改变、未经授权的存取则属于数据库安全保护问题,本章将进行专门讨论。

5.1　数据库安全性概述

数据库的安全性(security)是指保护数据库,防止不合法的使用,以免数据的泄露、更改或破坏。

数据库的安全性和完整性这两个概念听起来有些相似,有时容易混淆,但两者是完全不同的。

(1) 安全性。保护数据,防止非法用户故意造成的破坏,确保合法用户做其想做的事情。

(2) 完整性。保护数据,防止合法用户无意中造成的破坏,确保用户所做的事情是正确的。

两者的不同关键在于“合法”与“非法”、“故意”与“无意”。

为了保护数据库,防止故意的破坏,可以在从低到高的5个级别上设置各种安全措施。

(1) 物理控制。计算机系统的机房和设备应加以保护,如通过加锁或专门监护等措施来防止系统场地被非法进入,进行物理破坏。

(2) 法律保护。通过立法、规章制度防止授权用户以非法的形式将其访问数据库的权限转授给非法者。

(3) 操作系统支持。无论数据库系统多么安全,操作系统的安全弱点均可能成为入侵数据库的手段,应防止未经授权的用户从OS处着手访问数据库。

(4) 网络管理。由于大多数DBS都允许用户通过网络进行远程访问,因此网络软件内部的安全性是很重要的。

(5) DBMS 实现。DBMS 安全机制的职责是检查用户的身份是否合法及使用数据库的权限是否正确。

实现数据库系统安全,要具体考虑很多方面的问题,诸如:

(1) 法律、道德伦理及社会问题。例如,请求者对其请求的数据的权限是否合法。

(2) 政策问题。例如,拥有系统的组织单位如何授予使用者对数据的存取权限。

(3) 可操作性问题。有关的安全性政策、策略与方案如何落实到系统中实现。例如,若使用口令或密码,如何防止密码本身的泄露;若可以授权,如何防止被授权者再授权给不应被授权的人。

(4) 设施有效性问题。例如,系统所在地的控制保护、软硬件设备管理的安全特性等是否合适。

这些问题不属于本书讨论的范畴,我们只考虑数据库系统本身。要实现数据库安全,DBMS 必须提供下列支持:

(1) 安全策略说明:安全性说明语言,如支持授权的 SQL。

(2) 安全策略管理:安全约束目录的存储结构、存取控制方法和维护机制,如自主存取控制方法和强制存取控制方法。

(3) 安全性检查:执行"授权"及其检验,判定"他能做他想做的事情吗?"

(4) 用户识别:标识和确认用户,确定"他就是他说的那个人吗?"

现代 DBMS 一般采用自主(discretionary)和强制(mandatory)两种存取控制方法来解决安全性问题。在自主存取控制方法中,每个用户对各个数据对象被授予不同的存取权力(authority)或特权(privilege),哪些用户对哪些数据对象有哪些存取权限都按存取控制方案执行,但并不完全固定。而在强制存取控制方法中,所有的数据对象被标定一个密级,所有的用户也被授予一个许可证级别。对于任一数据对象,凡具有相应许可证级别的用户都可存取,否则不能。

5.2 数据库安全性控制

在一般计算机系统中,安全措施是一级级设置的,其安全控制模型如图 5-1 所示。

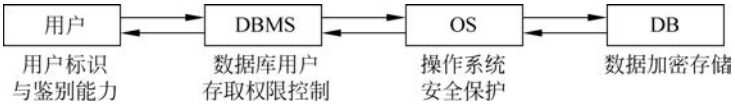

图 5-1 计算机系统的安全控制模型

(1) 当用户进入计算机系统时,系统首先根据输入的用户标识(如用户名)进行身份鉴定,只有合法的用户才准许进入系统。

(2) 对已进入计算机系统的用户,DBMS 还要进行存取控制,只允许用户在所授予的权限之内进行合法的操作。

(3) DBMS 是建立在操作系统之上的,安全的操作系统是数据库安全的前提。操作系统应能保证数据库中的数据必须由 DBMS 访问,而不允许用户越过 DBMS,直接通过操作系统或其他方式访问。

DBMS 与操作系统在安全上的关系,可用一个现实生活中与安全有关的实例来形象地

说明。2005年，某市发生了一起特大虫草盗窃案。盗贼通过租用店铺，从店铺的沙发下秘密地挖掘了一条39m长的地道，通往街对面的一家虫草行库房，盗走了价值千万元的虫草。虫草行库房周围的物理防护坚固，但盗贼绕过了这些防护，从库房地面这个薄弱环节盗走了虫草。

(4) 数据最后通过加密的方式存储到数据库中，即便非法者得到了已加密的数据，也无法识别数据内容。

对于操作系统这一级的安全措施不进行讨论，我们只讨论与数据库有关的用户标识与鉴别、存取控制、视图、数据加密和审计等安全技术。

5.2.1　用户标识与鉴别

实现数据库的安全性包含两方面的工作：一是用户的标识与确认，即用什么来标识一个用户，又怎样去识别他；二是授权及验证，即每个用户对各种数据对象的存取权限的表示和检查。这里只讨论第一个方面。

如何识别一个用户，常用的方法有三种：

(1) 用户的个人特征识别，如用户的声音、指纹、签名等。

(2) 用户的特有物品识别，如用户的磁卡、钥匙等。

(3) 用户的自定义识别，如用户设置的口令、密码和预定的问答等。

1. 用户的个人特征识别

使用每个人所具有的个人特征，如声音、指纹、签名等来识别用户是当前最有效的方法。但是有两个问题必须解决：

(1) 专门设备：要能准确地记录、存储和存取这些个人特征。

(2) 识别算法：要能较准确地识别出每个人的声音、指纹或签名。这里的关键问题是要让“合法者被拒绝”和“非法者被授权”的误判率达到应用环境可接受的程度。百分之百正确，即误判率为零几乎是不可能的。

另外，其实现代价也不得不考虑，这不仅包括经济上的代价，还包括识别算法执行的时间、空间代价。它影响整个安全子系统的代价/性能比。

2. 用户的特有物品识别

让每一用户持有一个其特有的物件，如磁卡、钥匙等。识别时，将其插入一个“阅读器”，它就会读取其面上的磁条中的信息。该方法是目前一些安全系统中较常用的一种方法，但用在数据库系统中要考虑如下问题：

(1) 需要专门的阅读装置。

(2) 要求有从阅读器中抽取信息及与DBMS接口的软件。

该方法的优点是比个人特征识别更简单、有效，代价/性能比更好；缺点是容易忘记带磁卡或钥匙等，也可能丢失甚至被别人窃取。

3. 用户的自定义识别

使用只有用户自己知道的定义内容来识别用户是最常用的一种方法，一般用口令或密码，有时用只有用户自己能给出正确答案的一组问题，有的还可以用两者的组合。

使用这类方法要注意：

(1) 标识的有效性。口令、密码或问题答案要尽可能准确地标识每一个用户。

(2) 内容的简易性。口令或密码要长短适中,问答过程不要太烦琐。

(3) 本身的安全性。为了防止口令、密码或问题答案的泄露或失窃,应经常地改变。

实现这种方法需要专门的软件来进行用户名或用户 ID 及其口令的登记、维护与检验等。但它不需要专门的硬件设备,较之以上的方法这是其优点。其主要的缺点是口令、密码或问题答案容易被人窃取,因此还可以用更复杂的方法。例如每个用户都预先约定好一个计算过程或函数,鉴别用户身份时,系统提供一个随机数,用户根据自己预先约定的计算过程或函数进行计算,系统根据用户的计算结果是否正确进一步鉴定用户身份。

例如,让用户记住一个表达式,如 T=XY+2Y。系统告诉用户 X=1,Y=2,如果用户回答是 T=6,则证明该用户的身份是合法的。当然,这是一个简单的例子,在实际使用中,还可以设计复杂的表达式,以使安全性更好。

5.2.2 存取控制

数据库安全性所关心的主要是 DBMS 的存取控制策略。数据库安全最重要的一点就是确保只授权给有资格的用户访问数据库的权限,同时令所有未被授权的人员无法接近数据,这主要通过数据库系统的存取控制策略来实现。

存取控制策略主要包括如下两部分:

(1) 定义用户权限,并将用户权限登记到数据字典中。用户对某一数据对象的操作权利称为权限。某个用户应该具有何种权限是管理问题和政策问题,而不是技术问题。DBMS 的功能就是保证这些决定的执行。为此 DBMS 必须提供适当的语言来定义用户权限,这些定义经过编译后存放在数据字典中,被称作安全规则或授权规则。

(2) 合法权限检查。每当用户发出存取数据库的操作请求后,DBMS 就会查找数据字典,根据安全规则进行合法权限检查。若用户的操作请求超出了定义的权限,系统将拒绝执行此操作。

定义用户权限和合法权限检查策略一起组成了 DBMS 的安全子系统。

当前,大多数 DBMS 所采取的存取控制策略主要有两种:自主存取控制和强制存取控制,其中自主存取控制的使用更为普遍。

1. 自主存取控制

在自主存取控制方法中,用户对于不同的数据库对象有不同的存取权限,不同的用户对同一对象也有不同的权限,而且用户还可将其拥有的存取权限转授给其他用户。因此,自主存取控制非常灵活。

赋予用户使用数据库的方式称为授权(Authorization)。权限有两种:操作数据的权限和修改数据库结构的权限。

操作数据的权限有 4 个:

(1) 读(SELECT)权限:允许用户读数据,但不能修改数据。

(2) 插入(INSERT)权限:允许用户插入新的数据,但不能修改数据。

(3) 修改(UPDATE)权限:允许用户修改数据,但不能删除数据。

(4) 删除(DELETE)权限:允许用户删除数据。

根据需要,可以授予用户上述权限中的一个或多个,也可以不授予上述任何一个权限。

修改数据库结构的权限也有 4 个:

(1) 索引(INDEX)权限：允许用户创建和删除索引。

(2) 资源(RESOURCE)权限：允许用户创建新的关系。

(3) 修改(ALTERNATION)权限：允许用户在关系结构中加入或删除属性。

(4) 撤销(DROP)权限：允许用户撤销关系。

自主存取控制方式是通过授权和取消来实现的。下面介绍自主存取控制的权限类型，包括角色(role)权限、数据库对象权限及各自的授权和取消方法。

1) 权限类型

自主存取控制的权限类型分为两种，分别是角色权限和数据库对象权限。

(1) 角色权限：给角色授权并为用户分配角色，则用户的权限为其角色权限之和。角色权限由 DBA 授予。

(2) 数据库对象权限：不同的数据库对象可提供给用户不同的操作。该权限由 DBA 或该对象的拥有者(Owner)授予用户。

2) 角色的授权与取消

授权命令的语法如下：

```
GRANT  角色类型[,角色类型]  TO  用户  [IDENTIFIED  BY  口令]
角色类型::=Connect|Resource|DBA
```

其中，Connect 表示该用户可连接到 DBMS；Resource 表示用户可访问数据库资源；DBA 表示该用户为数据库管理员；IDENTIFIED　BY 用于为用户设置一个初始口令。

取消命令的语法如下：

```
REVOKE  角色管理[,角色管理]  FROM  用户
```

3) 数据库对象的授权与取消

授权命令的语法如下：

```
GRANT  权限  ON  表名  TO  用户[,用户]
    [WITH  GRANT  OPTION]
<权限>::=ALL PRIVILEGES|SELECT|INSERT|DELETE|UPDATE[(列名[,列名])]
```

其中，WITH GRANT OPTION 表示得到授权的用户，可将其获得的权限转授给其他用户；ALL PRIVILEGES 表示所有的操作权限。

取消命令的语法如下：

```
REVOKE  权限  ON  表名  FROM  用户[,用户]
```

说明：数据库对象除了表之外，还有视图等其他对象，但由于表的授权最具典型意义，且表的授权也最复杂，因此，此处只以表的授权为例来说明数据库对象的授权语法。其他对象的授权语法类似，只是在权限上不同。

2. 强制存取控制

自主存取控制能够通过授权机制有效地控制对敏感数据的存取。但它存在一个漏洞，一些别有用心的用户可以欺骗授权用户，采用一定的手段来获取敏感数据。例如，领导 Manager 是客户单 Customer 关系的物主，他将“读”权限授予员工 A，且 A 不能再将该权限转授他人，其目的是让 A 审查客户信息，看有无错误。现在 A 自己另外创建一个新关系 A_customer，然后将自 Customer 读取的数据写入(即复制到)A_customer。这样，A 是 A_

customer 的物主,他可以做任何事情,包括再将其权限转授给任何别的用户。

存在这种漏洞的根源在于:自主存取控制机制仅以授权将用户(主体)与被存取数据对象(客体)关联,通过控制权限来实现安全要求,对用户和数据对象本身未作任何安全性标注。强制存取控制就能处理自主存取控制的这种漏洞。

强制存取控制方法的基本思想在于为每个数据对象(文件、记录或字段等)赋予一定的密级,级别从高到低为:绝密级(Top Secret,TS)、机密级(Secret,S)、可信级(Confidential,C)、公开级(Public,P)。每个用户也具有相应的级别,称为许可证级别。密级和许可证级别都是严格有序的,如 TS > S > C > P。

在系统运行时,采用如下两条简单规则:

(1) 用户 i 只能查看比它级别低或同级的数据;

(2) 用户 i 只能修改和它同级的数据。

强制存取控制是对数据本身进行密级标记。无论数据如何复制,标记与数据是一个不可分的整体。只有符合密级标记要求的用户才可以操纵数据,从而提供了更高级别的安全性。

强制存取控制的优点是系统能执行“信息流控制”。在前面介绍的授权方法中,允许有权查看保密数据的用户把这种数据复制到非保密的文件中,造成无权用户也可以接触保密的数据。而强制存取控制可以避免这种非法的信息流动。

注意:这种方法在通用数据库系统中不十分有用,只是在某些专用系统中才有用,例如军事部门或政府部门。

5.2.3 视图机制

视图可以作为一种安全机制。通过视图,用户只能查看和修改他们所能看到的数据,其他数据库或关系既不可见也不可以访问。如果某一用户想要访问视图的结果集,必须被授予访问权限。

例如,假定李平老师具有检索和增、删、改“数据库”课程成绩信息的所有权限,学生王莎只能检索该科所有同学成绩的信息。那么,可以先建立“数据库”课程成绩的视图 score_db,然后在视图上进一步定义存取权限。具体步骤如下:

(1) 建立视图 score_db,代码如下:

```
CREATE VIEW score_db
AS
SELECT *
FROM score
WHERE cnam = '数据库';
```

(2) 为用户授予操作视图的权限,代码如下:

```
GRANT SELECT
ON score_db
TO 王莎;

GRANT ALL  PRIVILEGES
ON score_db
TO 李平;
```

5.2.4　安全级别及审计跟踪

前面讲的用户标识与鉴别、存取控制仅是安全性标准的一个重要方面(安全策略方面),不是全部。为了使 DBMS 达到一定的安全级别,还需要在其他方面提供相应的支持。例如按照 TCSEC/TDI 标准中安全策略的要求,审计功能就是 DBMS 达到 C2 以上安全级别必不可少的一项指标。

1. 安全级别[**]

美国国防部根据军用计算机系统的安全需要,于 1985 年制定了《可信计算机系统评估标准》(Trusted Computer System Evaluation Criteria,TCSEC)。1991 年,美国国家安全局的国家计算机安全中心发布了 TCSEC 的可信数据库系统解释(Trusted Database Interpretation,TDI),形成了最早的信息安全及数据库安全评估体系。TCSEC/TDI 将系统安全性分为 4 等 7 级,依次是 D——最小保护、C(包括 C1、C2)——自主保护、B(包括 B1、B2、B3)——强制保护、A(包括 A1)——验证保护,按系统可靠或可信程度逐渐增高。下面分别进行简单介绍。

(1) D 级:最低安全级别。保留 D 级是为了将一切不符合更高标准的系统统统归于 D 级,如 DOS 就是操作系统中安全标准为 D 级的典型例子。

(2) C1 级:实现数据所有权与使用权的分离,进行自主存取控制,保护或限制用户权限的传播。

(3) C2 级:提供受控的存取保护,即将 C1 级的自主存取控制进一步细化,通过身份注册、审计和资源隔离来支持"责任"说明。

(4) B1 级:标记安全保护,即对每一客体和主体分别标以一定的密级和安全证等级,实施强制存取控制以及审计等安全机制。

(5) B2 级:建立安全策略的形式化模型,并能识别和消除隐通道。

(6) B3 级:提供审计和系统恢复过程,且指定安全管理员(通常是 DBA)。

(7) A1 级:验证设计,提供 B3 级保护的同时给出系统的形式化设计说明和验证,以确信各安全保护真正实现。即安全机制是可靠的,且对安全机制能实现的指定安全策略给出数学证明。

2. 审计跟踪

任何系统的安全保护措施都不是完美无缺的,蓄意盗取、破坏数据的人总会想方设法打破控制。审计功能把用户对数据库的所有操作自动记录下来放入"审计日志"(audit log)中,称为审计跟踪。DBA 可以利用审计跟踪信息,重现导致数据库现有状况的一系列事件,找出非法存取数据的人、时间和内容等,为分析攻击者线索提供依据。一般地,将审计跟踪和数据库日志记录结合起来,会达到更好的安全审计效果。

DBMS 的审计主要分为语句审计、权限审计、模式对象审计和资源审计。语句审计是指监视一个或多个特定用户或者所有用户提交的数据库操作语句(即 SQL 语句);权限审计是指监视一个或多个特定用户或所有用户使用的系统权限;模式对象审计是指监视一个模式中在一个或多个对象上发生的行为;资源审计是指监视分配给每个用户的系统资源。

审计机制应该至少记录用户标识和认证、客体的存取、授权用户进行并影响系统安全的

操作以及其他安全相关事件。对于每个记录的事件,审计记录中需要包括事件时间、时间类型、用户、事件数据和事件的成功/失败情况。对于用户标识和认证事件,必须记录事件源的终端 ID 和源地址等;对于访问和改变对象的事件,则需要记录对象的名称。

审计通常是很费时间和空间的,所以 DBMS 往往都将其作为可选特征,允许 DBA 根据应用对安全性的要求,灵活地打开或关闭审计功能。审计功能一般主要用于对安全性要求较高的部门。

5.2.5 数据加密

对于高度敏感性数据,例如财务数据、军事数据、国家机密,除以上安全性措施外,还可以采用数据加密技术。

数据加密是防止数据库中数据在存储和传输中失密的有效手段。加密的基本思想是根据一定的算法将原始数据(称为明文)变换为不可直接识别的格式(称为密文),从而使得不知道解密算法的人无法获知数据的内容。

加密方法主要有两种:一种是替换方法,另一种是转换方法。

(1) 替换加密法。这种方法是制定一种规则,将明文中的每个或每组字符替换成密文中的一个或一组字符。其缺点是使用得多了,窃密者可以从多次搜集的密文中发现其中的规律,破解加密方法。

(2) 转换加密法。这种方法不隐藏原来明文的字符,而是将字符重新排序。比如,加密方首先选择一个用数字表示的密钥,写成一行,然后把明文逐行写在数字下,再按照密钥中数字指示的顺序将原文重新抄写,就形成密文。例如:

- 密钥:6852491703。
- 明文:张三偷走了李四的钱包。
- 密文:李的了三走钱张四包偷。

单独使用这两种方法的任意一种都是不安全的,但是将这两种方法合起来就能提供相当高的安全程度。采用这种结合算法的例子是美国 1977 年制定的官方加密标准——数据加密标准(Data Encryption Standard,DES)。有关 DES 的密钥加密技术及密钥管理问题在这里不再讨论。

目前有些数据库产品提供了数据加密例行程序,可根据用户的要求自动对存储和传输的数据进行加密处理。另一些数据库产品虽然本身未提供加密程序,但提供了接口,允许用户用其他厂商的加密程序对数据加密。

由于数据加密与解密也是比较费时的操作,而且数据加密与解密程序会占用大量系统资源,因此数据加密功能通常也作为可选特征,允许用户自由选择,只对高度机密的数据加密。

5.3 Oracle 的安全设置

为了防止非法用户对数据库进行操作,保证数据库安全运行,Oracle 定义了一整套丰富的、完整的权限机制,只有通过权限认证的用户才可以对相应的数据库对象进行存取,而非授权用户则被禁止存取数据。下面将初步介绍 Oracle 对用户、权限和角色的管理。

5.3.1　用户管理

在 Oracle 数据库中，为了防止非授权数据库用户对数据库进行存取，DBA 可以创建登录用户、修改用户账号信息和删除用户账号。

1. 创建普通用户

在 Oracle 数据库中，DBA 可以创建普通登录用户。创建用户账号主要通过 CREATE USER 语句实现，CREATE USER 语句的基本语法如下：

```
CREATE  USER 用户名
IDENTIFIED  BY 口令
[DEFAULT  TABLESPACE 表空间名]
[QUOTA  nK|M  |UNLIMITED ON 表空间名]
[PASSWORD  EXPIRE]
[ACCOUNT  LOCK|UNLOCK];
```

说明：(1) 用户名：在 Oracle 19c 中，用户名必须使用“C##”或“c##”开头。

(2) 口令：注意口令区分字母大小写。

(3) DEFAULT TABLESPACE：为用户指定默认的表空间。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。在 Oracle 19c 中，默认的表空间有 5 个，分别为 SYSTEM、SYSAUX、UNDOTBS1、TEMP 和 USERS。

(4) QUOTA：设置用户使用表空间的最大值。如果设置成 UNLIMITED，表示对表空间的使用没有限制。

(5) PASSWORD EXPIRE：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。

(6) ACCOUNT：用于设置锁定状态。如果设置成 LOCK，则用户不能访问数据库；如果设置成 UNLOCK，用户可以访问数据库。默认为 UNLOCK。

【例 5-1】 使用 SYSDBA 身份连接数据库。创建用户账号 c##tempuser，其口令为 oracle，并且设置口令立即过期的方式。

以 SYSDBA 身份连接数据库，代码如下：

```
SQL> CONN sys/root@orcl as SYSDBA;

已连接。
```

创建用户，代码如下：

```
SQL> CREATE USER c##tempuser IDENTIFIED BY oracle PASSWORD EXPIRE;

用户已创建。
```

建立用户后，必须为用户授予 CREATE SESSION 权限才能连接到数据库。授权代码如下：

```
SQL> GRANT CREATE SESSION TO c##tempuser;

授权成功。
```

连接数据库，代码如下：

```
SQL> CONN c##tempuser@orcl;
```

```
输入口令:
ERROR:
ORA-28001: 口令已经失效

更改 c##tempuser 的口令
新口令:
重新键入新口令:
口令已更改。
已连接。
```

2. 修改用户账号

在创建用户账号后,允许对其进行修改。修改用户账号的 ALTER USER 语句的基本语法如下:

```
ALTER  USER 用户名
IDENTIFIED  BY 口令
[DEFAULT  TABLESPACE 表空间名]
[QUOTA  nK|M  |UNLIMITED ON 表空间名]
[PASSWORD  EXPIRE]
[ACCOUNT  LOCK|UNLOCK];
```

【例 5-2】 修改用户账号 c##tempuser 的口令为 password。

修改口令的代码如下:

```
SQL> ALTER USER c##tempuser IDENTIFIED BY password;

口令已更改。
```

3. 删除用户账号

用户被删除后,该用户所创建的所有模式对象都将被删除。删除用户账号的语句如下:

```
DROP  USER 用户名  [CASCADE];
```

说明:如果用户已经创建了模式对象(如表、视图、索引等),在删除用户时必须增加 CASCADE 选项,表示在删除用户时,连同该用户创建的模式对象也全部删除。

【例 5-3】 删除用户账号 c##tempuser。

(1) 为 c##tempuser 用户授予建立会话连接、建表、在表空间中创建对象的权限。

连接数据库,代码如下:

```
SQL> CONN sys@orcl as SYSDBA;
输入口令:
已连接。
```

为用户授权,代码如下:

```
SQL> GRANT CREATE SESSION,CREATE TABLE,UNLIMITED TABLESPACE
  2  TO c##tempuser;

授权成功。
```

(2) 以用户账号 c##tempuser 连接数据库,并创建表。

连接数据库,代码如下:

```
SQL> CONN c##tempuser/password@orcl;
已连接。
```

创建表，代码如下：

```
SQL> CREATE TABLE student(
  2  sno CHAR(6),
  3  sname VARCHAR2(8)
  4  );

表已创建。
```

(3) 再以 SYSDBA 身份连接数据库，删除 c##tempuser 用户账号。

连接数据库，代码如下：

```
SQL> CONN sys/root@orcl as SYSDBA;
已连接。
```

删除用户账号，代码如下：

```
SQL> DROP USER c##tempuser;
DROP USER c##tempuser
*
第 1 行出现错误:
ORA-01922: 必须指定 CASCADE 以删除 'c##tempuser'
```

重新删除账户，代码如下：

```
SQL> DROP USER c##tempuser CASCADE;

用户已删除。
```

5.3.2　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。用户对数据进行任何操作，都需要具有相应的操作权限。

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机制。系统级控制决定是否可以连接数据库、在数据库中可以进行哪些系统操作等。总之，系统权限是对用户设置的，用户必须具有相应的系统权限，才可以连接数据库并进行相应的操作。例如，用户为了连接数据库，必须具有的权限 CONNECT 就是一个系统权限。

Oracle 提供了多种系统权限，每一种系统权限分别能使用户进行某种或某一类特定的数据库操作。表 5-1 列出了一些常见的 Oracle 系统权限。

表 5-1　系统权限

名　　称	功　　能
CREATE SESSION	允许用户创建会话连接数据库的权限
UNLIMITED TABLESPACE	允许用户在表空间中创建对象而不受表空间限制的权限
CREATE|ALTER|DROP ANY TABLE	允许在任何用户模式中创建、修改、删除基本表的权限
CREATE|ALTER|DROP ANY INDEX	允许在任何模式下创建、修改、删除索引的权限

续表

名　　称	功　　能
CREATE|ALTER|DROP ANY PROCEDURE	允许在任何用户模式中创建、修改、删除存储过程的权限
EXECUTE ANY PROCEDURE	允许执行任何用户模式中的存储过程的权限
CREATE|ALTER|DROP ANY TRIGGER	在任何用户模式中创建、修改、删除触发器的权限
CREATEE|ALTER|DROP ANY ROLE	在任何用户模式中创建、修改、删除角色的权限
GRANT ANY ROLE	允许用户将数据库中任何角色授予其他用户的权限
GRANT ALL PRIVILEGE	将数据库任何权限授予任何用户。该权限仅包含所有的系统权限,并不包括对象权限

(1) 使用 GRANT 语句向用户授予系统权限的基本语法如下:

```
GRANT 系统权限 TO 用户名 [WITH ADMIN OPTION];
```

说明: WITH ADMIN 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用户。

(2) 使用 REVOKE 语句撤销系统权限的基本语法如下:

```
REVOKE  系统权限  FROM  用户名;
```

【例 5-4】 分析下面各 SQL 语句。

创建用户,代码如下:

```
SQL> CREATE USER c## user1 IDENTIFIED BY user1;

用户已创建。
```

连接数据库,代码如下:

```
SQL> CONN c## user1/user1@orcl;
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。
```

重新连接数据库,代码如下:

```
SQL> CONN sys/root@orcl as SYSDBA;
已连接。
```

向用户授权,代码如下:

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

授权成功。
```

再次连接数据库,代码如下:

```
SQL> CONN c## user1/user1@orcl;
已连接。
```

创建表,代码如下:

```
SQL> CREATE TABLE test_date(
  2  id NUMBER(10),
  3  name VARCHAR2(20)
  4  );
```

```
CREATE TABLE test_date(
 *
第 1 行出现错误:
ORA-01031: 权限不足
```

重新连接数据库,代码如下:

```
SQL> CONN sys/root@orcl as SYSDBA;
已连接。
```

向用户授权,代码如下:

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

授权成功。
```

再次连接数据库,代码如下:

```
SQL> CONN c##user1/user1@orcl
已连接。
```

创建表,代码如下:

```
SQL> CREATE TABLE test_date(
  2  id NUMBER(10),
  3  name VARCHAR2(20)
  4  );

表已创建。
```

插入数据,代码如下:

```
SQL> INSERT INTO test_date VALUES(1,'Mary');
INSERT INTO test_date VALUES(1,'Mary')
            *
第 1 行出现错误:
ORA-01950: 对表空间 'USERS' 无权限
```

重新连接数据库,代码如下:

```
SQL> CONN sys/root@orcl as SYSDBA;
已连接。
```

向用户授权,代码如下:

```
SQL> GRANT UNLIMITED TABLESPACE TO c##user1;

授权成功。
```

再次连接数据库,代码如下:

```
SQL> CONN c##user1/user1@orcl;
已连接。
```

向表中插入数据,代码如下:

```
SQL> INSERT INTO test_date VALUES(1,'Mary');

已创建 1 行。
```

2. 对象权限

对象权限是指在模式对象上控制存取和使用的机制。例如,希望向 ORCL 模式的

“DEPT”表插入行时,用户必须具有完成该操作的权限。

系统权限会控制对 Oracle 数据库中各种系统级功能的访问,而对象权限可以用来控制对指定数据库对象的访问。任何数据库用户都可以被授予这些权限,以便他们对模式中的对象进行访问。

相对于数量众多的各种 Oracle 系统权限,对象权限相对较少,并且容易理解。最常使用的对象权限如表 5-2 所示。

表 5-2 对象权限

对 象	操 作
表	SELECT、INSERT、UPDATE、DELETE、REFERANCES
视图	SELECT、INSERT、UPDATE、DELETE
存储过程	EXECUTE
列	SELECT、UPDATE

(1) 使用 GRANT 语句向用户授予对象权限的基本语法如下:

```
GRANT  对象权限 ON 对象名 TO 用户名 [WITH  GRANT  OPTION];
```

说明:WITH GRANT OPTION 表示被授予对象权限的用户可以将其获取的权限授予其他用户。

(2) 使用 REVOKE 语句撤销用户对象权限的基本语法如下:

```
REVOKE  对象权限  ON 对象名 FROM 用户名;
```

【例 5-5】 分析下面各 SQL 语句。

以 SYSDBA 身份连接数据库,代码如下:

```
SQL> CONN sys/root@orcl as SYSDBA;
已连接。
```

创建用户 c##user2,代码如下:

```
SQL> CREATE USER c##user2 IDENTIFIED BY user2;

用户已创建。
```

创建用户 c##user3,代码如下:

```
SQL> CREATE USER c##user3 IDENTIFIED BY user3;

用户已创建。
```

向用户授予系统权限,代码如下:

```
SQL> GRANT CREATE SESSION TO c##user2,c##user3;

授权成功。
```

向用户 c##user2 授予对象权限,代码如下:

```
SQL> GRANT SELECT ON sys.dept TO c##user2 WITH GRANT OPTION;

授权成功。
```

以 c##user2 身份连接数据库,代码如下:

```
SQL> CONN c##user2/user2@orcl;
已连接。
```

查看信息，代码如下：

```
SQL> SELECT * FROM sys.dept;

DEPTNO    DNAME         LOC
------  ----------  ---------
    10    ACCOUNTING    NEW YORK
    20    RESEARCH      DALLAS
    30    SALES         CHICAGO
    40    OPERATIONS    BOSTON
```

向用户 c##user3 授予对象权限，代码如下：

```
SQL> GRANT SELECT ON sys.dept TO c##user3;

授权成功。
```

以 c##user3 身份连接数据库，代码如下：

```
SQL> CONN c##user3/user3@orcl;
已连接。
```

查看信息，代码如下：

```
SQL> SELECT * FROM sys.dept;
DEPTNO    DNAME         LOC
------  ----------  ---------
    10    ACCOUNTING    NEW YORK
    20    RESEARCH      DALLAS
    30    SALES         CHICAGO
    40    OPERATIONS    BOSTON
```

以 SYSDBA 身份连接数据库，代码如下：

```
SQL> CONN sys/root@orcl as SYSDBA;
已连接。
```

撤销 c##user2 的对象权限，代码如下：

```
SQL> REVOKE SELECT ON sys.dept FROM c##user2;

撤销成功。
```

以 c##user2 身份连接数据库，代码如下：

```
SQL> CONN c##user2/user2@orcl;
已连接。
```

查看修改后的信息，代码如下：

```
SQL> SELECT * FROM sys.dept;
SELECT * FROM sys.dept
                  *
第 1 行出现错误:
ORA-00942: 表或视图不存在
```

5.3.3 角色管理

从前面的介绍中可以看出,Oracle 的权限设置是非常复杂的,权限的类型也非常多,这就为 DBA 有效地管理数据库权限带来了困难。另外,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,则这个工作量是非常庞大的。为了简化权限管理,Oracle 提供了角色的概念。

角色是具有名称的一组相关权限的组合,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。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。最常使用的对象权限如表 5-3 所示。

表 5-3 Oracle 常用的预定义角色

角 色 名	角色权限功能
CONNECT	连接数据库,建立数据库链路、序列生成器、表、视图以及修改会话的权限
RESOURCE	建立表、序列生成器、存储过程、触发器的权限
DBA	拥有最高级别的权限

2. 自定义角色

1) 创建用户自定义角色

用户可以自己创建角色,即自定义角色。创建自定义角色的语法如下:

```
CREATE  ROLE  角色名;
```

说明:角色名必须以“C##”或“c##”开头。

【例 5-6】 分析下面各 SQL 语句。

以 SYSDBA 身份连接数据库,代码如下:

```
SQL> CONN sys/root@orcl as SYSDBA;
已连接。
```

创建自定义角色 c##general_user,代码如下:

```
SQL> CREATE ROLE c##general_user;

角色已创建。
```

为角色 c##general_user 授予对象权限,代码如下:

```
SQL> GRANT SELECT,INSERT,DELETE ON sys.dept TO c##general_user;

授权成功。
```

将角色 c##general_user 授予 c##user2,代码如下:

```
SQL> GRANT c## #general_user TO c## #user2;

授权成功。
```

以 c## #user2 身份连接数据库，代码如下：

```
SQL> CONN c## #user2/user2@orcl;
已连接。
```

查看修改后的信息，代码如下：

```
SQL> SELECT * FROM sys.dept;

DEPTNO  DNAME          LOC
------  -----------  ---------
    10  ACCOUNTING     NEW YORK
    20  RESEARCH       DALLS
    30  SALES          CHICAGO
    40  OERATIONS      BOSTON
```

再次以 SYSDBA 身份连接数据库，代码如下：

```
SQL> CONN sys/root@orcl as SYSDBA;
已连接。
```

撤销 c## #general_user 的对象权限，代码如下：

```
SQL> REVOKE SELECT ON sys.dept FROM c## #general_user;

撤销成功。
```

再次以 c## #user2 身份连接数据库，代码如下：

```
SQL> CONN c## #user2/user2@orcl;
已连接。
```

查看修改后的信息，代码如下：

```
SQL> SELECT * FROM sys.dept;
SELECT * FROM sys.dept
              *
第 1 行出现错误:
ORA-01031: 权限不足
```

2）删除角色

当不需要某个角色时，可以使用语句 DROP ROLE 进行删除。角色被删除后，使用该角色的用户的所有权限将丢失。删除角色的语法如下：

```
DROP ROLE 角色名;
```

【例 5-7】 接续例 5-6，分析下面各 SQL 语句。

以 SYSDBA 身份连接数据库，代码如下：

```
SQL> CONN sys/root@orcl as SYSDBA;
已连接。
```

删除角色 c## #general_user，代码如下：

```
SQL> DROP ROLE c## #general_user;
```

角色已删除。

以 c##user2 身份连接数据库,代码如下:

```
SQL> CONN c##user2/user2@orcl;
已连接。
```

插入数据,代码如下:

```
SQL> INSERT INTO sys.dept VALUES(70,'SALE2','CHINA');
INSERT INTO sys.dept VALUES(70,'SALE2','CHINA')
                    *
第 1 行出现错误:
ORA-00942: 表或视图不存在
```

5.4 小　　结

数据库的安全指的是保护数据,防止非法使用造成的数据泄露、更改和破坏。数据库的安全管理涉及用户的访问权限问题,可通过设置用户标识、用户的存取控制权限、定义视图、审计、数据加密技术等来保证数据不被非法使用。

实现数据库系统安全性的技术和方法有多种,最重要的是存取控制技术、视图技术和审计技术。自主存取控制功能一般是通过 SQL 的 GRANT 语句和 REVOKE 语句来实现的,对数据库模式的授权则由 DBA 在创建用户时通过 CREATE USER 语句实现。数据库角色是一组权限的集合。使用角色来管理数据库权限可以简化授权的过程。在 SQL 中用 CREATE ROLE 语句创建角色,用 GRANT 语句给角色授权。

习　题　五

一、选择题

1. 对用户访问数据库的权限加以限定是为了保护数据库的(　　)。

A. 安全性　　B. 完整性　　C. 一致性　　D. 并发性

2. 数据库的(　　)是指数据的正确性和相容性。

A. 完整性　　B. 安全性　　C. 并发控制　　D. 系统恢复

3. 在数据库系统中,定义用户可以对哪些数据对象进行何种操作被称为(　　)。

A. 审计　　B. 授权　　C. 定义　　D. 视图

4. 某高校 5 个系的学生信息存放在同一个基本表中,采取(　　)的措施可使各系的管理员只能读取本系学生的信息。

A. 建立各系的列级视图,并将对该视图的读权限赋予该系的管理员

B. 建立各系的行级视图,并将对该视图的读权限赋予该系的管理员

C. 将学生信息表部分列的读权限赋予各系的管理员

D. 将修改学生信息表的权限赋予各系的管理员

5. 关于 SQL 对象的操作权限,描述正确的是(　　)。

A. 权限的种类分为 INSERT、DELETE 和 UPDATE 三种

B. 权限只能用于实表,不能应用于视图

C. 使用 REVOKE 语句可撤销权限

D. 使用 COMMIT 语句可赋予权限

6. 若将 Workers 表的插入权限赋予用户 c##user1，并允许其将该权限授予他人，那么对应的 SQL 语句为“GRANT(　①　)TABLE Workers TO user1 (　②　);”。

① A. INSERT　　B. INSERT ON　　C. UPDATE　　D. UPDATE ON

② A. FOR ALL　　B. PUBLIC

C. WITH CHECK OPTION　　D. WITH GRANT OPTION

二、填空题

1. 对数据库_____性的保护就是指要采取措施，防止库中数据被非法访问、修改，甚至恶意破坏。

2. 安全性控制的一般方法有_____、_____、_____、_____和_____5 种。

3. Oracle 数据库中将权限分为两类，即_____。_____是指在系统级控制数据库的存取和使用机制，_____是指在模式对象上控制存取和使用的机制。

4. _____是具有名称的一组相关权限的组合。

5. 授予权限和撤销权限的命令依次是_____和_____。

三、操作题

1. 对 ORCL 数据库下的表 DEPT：

```
DEPT(deptno,dname,loc)
```

请用 SQL 的 GRANT 和 REVOKE 语句(加上视图机制)完成以下授权定义或存取控制功能：

(1) 使用 SYSDBA 身份连接数据库，并创建用户账号 c##test_user，其口令为 oracle。

(2) 向用户 c##test_user 授予连接数据库系统的权限。

(3) 向用户 c##test_user 授予对象“SYS.DEPT”的 SELECT 权限。

(4) 向用户 c##test_user 授予对象“SYS.DEPT”的 INSERT、DELETE 权限，使其对 LOC 字段具有更新权限。

(5) 用户 c##test_user 具有对 SYS.DEPT 表的所有权限(读、插、改、删)，并具有给其他用户授权的权限。

(6) 撤销用户 c##test_user 对 SYS.DEPT 表的所有权限。

(7) 用户 c##test_user 只有查看“10”号部门的权限，不能查看其他部门信息。

(8) 建立角色 c##ROLE1，使其具有连接数据库、创建表的权限。

(9) 将 c##ROLE1 角色的权限授予用户 c##test_user。

(10) 删除角色 c##ROLE1。

2. 阅读下列说明，回答问题(1)～(5)。

某工厂仓库管理数据库的部分关系模式如下所示：

```
仓库(仓库号,面积,负责人,电话)
原材料(编号,名称,数量,储备量,仓库号)
```

要求一种原材料只能存放在同一仓库中。仓库和原材料的关系实例分别如表 5-4 和

表 5-5 所示。

表 5-4　仓库关系

仓　库　号	面积/m^2	负　责　人	电　　话
01	500	李劲松	87654120
02	300	陈东明	87654122
03	300	郑爽	87654123
04	400	刘春来	87654125

表 5-5　原材料关系

编　　号	名　　称	数量/kg	储备量/kg	仓　库　号
1001	小麦	100	50	01
1002	大豆	20	10	02
2001	玉米	50	30	01
2002	花生	30	50	02
3001	菜油	60	20	03

(1) 根据上述说明,用 SQL 定义原材料和仓库的关系模式如下:

```
CREATE  TABLE  仓库(仓库号  Char(4),  面积  Int,  负责人  Char(8),
  电话  Char(8), _____________________________________);      //主键定义
CREATE  TABLE  原材料(编号  Char(4)  _____________,      //主键定义
    名称  CHAR(6),数量  INT,储备量  INT,
    仓库号 _____________________ ,
    _____________________________);                    //外键定义
```

(2) 将下面的 SQL 语句补充完整,完成“查询存放原材料数量最多的仓库号”的功能。

```
SELECT 仓库号
FROM_____________________________;
```

(3) 将下面的 SQL 语句补充完整,完成“01 号仓库所存储的原材料信息只能由管理员李劲松来维护,而采购员李强能够查询所有原材料的库存信息”的功能。

```
CREATE  VIEW  raws_in_wh01
AS
  SELECT_____________________  FROM  原材料  WHERE  仓库号 = '01';

Grant_____________________  ON  _____________  TO  李劲松;
Grant_____________________  ON  _____________  TO  李强;
```