第5章 chapter 5

数据库安全保护

本章要点

随着社会信息化的不断深化，各种数据库的使用也越来越广泛。例如，一个企业管理信息系统的全部数据、国家机构的事务管理信息、国防情报机密信息、基于WEB动态发布的网上购物信息等，都集中或分布地存放在或大或小的数据库中。数据库系统中的数据是由DBMS统一进行管理和控制的。为了适应和满足数据共享的环境和要求，DBMS要保证数据库及整个系统的正常运转，防止数据意外丢失和不一致数据的产生，以及保证当数据库遭受破坏后能迅速地恢复正常，这就是数据库的安全保护。DBMS对数据库的安全保护功能是通过四方面来实现的，即安全性控制、完整性控制、并发性控制和数据库恢复。本章就将从这四方面来介绍数据库的安全保护功能，重点要求读者掌握它们的含义及实现方法，可结合SQL Server 2000或2005或2008加深对四部分内容的理解，并掌握其操作技能。

5.1 数据库的安全性

5.1.1 数据库安全性概述

数据库的安全性是指保护数据库，以防止非法使用所造成数据的泄露、更改或破坏。安全性问题有许多方面，其中包括(1)法律、社会和伦理方面，例如，请求查询信息的人是否有合法的权力；(2)物理控制方面，例如，计算机机房或终端是否应该加锁或用其他方法加以保护；(3)政策方面，确定存取原则，允许哪些用户存取哪些数据；(4)运行与技术方面，使用口令时，如何使口令保持秘密；(5)硬件控制方面，CPU是否提供任何安全性方面的功能，诸如存储保护键或特权工作方式；(6)操作系统安全性方面，在主存储器和数据文件用过以后，操作系统是否把它们的内容清除；(7)数据库系统本身安全性方面。

下面主要讨论的是数据库本身的安全性问题，即主要考虑安全保护的策略，尤其是控制访问的策略。

5.1.2 安全性控制的一般方法

安全性控制是指要尽可能地杜绝所有可能的数据库非法访问。用户非法使用数据库可以有很多种情况。例如,编写合法的程序绕过 DBMS 授权机制,通过操作系统直接存取、修改或备份有关数据。用户访问非法数据,无论他们是有意的还是无意的,都应该加以严格控制,因此,系统还要考虑数据信息的流动问题并加以控制,否则有潜在的危险性。因为数据的流动可能使无权访问的用户获得访问权力。

例如,甲用户可以访问表 T1,但无权访问表 T2,如果乙用户把表 T2 的所有记录都添加到表 T1 中之后,则由于乙用户的操作,甲用户便可获得对表 T2 中记录的访问。此外,用户可以多次利用允许的访问结果,经过逻辑推理得到无权访问的数据。

为防止这一点,访问的许可权还要结合过去访问的情况而定。可见安全性的实施是要花费一定代价并需缜密考虑的。安全保护策略就是要以最小的代价来最大程度地防止用户对数据的非法访问,通常需要层层设置安全措施。

实际上,数据库系统的安全性问题,类似于整个计算机系统一级级层层设置安全的情况,其安全控制模型一般如图 5.1 所示。

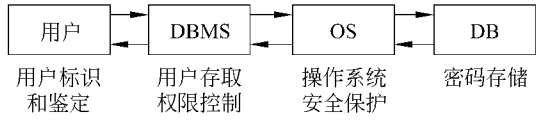

图 5.1 安全控制模型

根据图 5.1 的安全模型,当用户进入计算机系统时,系统首先根据用户输入的用户标识对用户进行身份的鉴定,只有合法的用户才准许进入系统。

对已进入系统的用户,DBMS 还要进行存取控制,只允许用户进行合法的操作。

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来访问。

数据最后可以通过密码的形式存储到数据库中。使非法者即使得到了加密数据,也无法识别它的安全效果。

下面介绍同数据库有关的用户标识和鉴定、存取控制、定义视图、数据加密和审计等几类安全性措施的内容。

1. 用户标识和鉴别

数据库系统是不允许一个未经授权的用户对数据库进行操作的。用户标识和鉴定是系统提供的最外层的安全保护措施,其方法是由系统提供一定的方式由用户标识自己的名字或身份,系统内部记录着所有合法用户的标识,每次用户要求进入系统时,都由系统进行核实,通过鉴定后才提供机器上数据库的使用权。

用户标识和鉴定的方法有多种,为了获得更强的安全性,往往是多种方法并举,常用

的方法有以下几种。

(1) 用一个用户名或用户标识符来标明用户的身份,系统以此来鉴别用户的合法性。如果正确,则可进入下一步的核实,否则,用户不能使用计算机。该方法称为单用户名鉴别法。

(2) 用户标识符是用户公开的标识,它不足以成为鉴别用户身份的凭证。为了进一步核实用户身份,常采用用户名与口令(Password)相结合的方法,系统通过核对口令来判别用户身份的真伪。系统有一张用户口令表,为每个用户保持一个记录,包括用户名和口令两部分数据。用户先输入用户名,然后系统要求用户输入口令。为了保密,用户在终端上输入的口令不显示在屏幕上。系统核对口令以鉴别用户身份。该方法称为用户名与口令联合鉴别法。

(3) 通过用户名和口令来鉴定用户的方法简单易行,但该方法在使用时,由于用户名和口令的产生和使用比较简单,也容易被窃取,因此还可采用更复杂的方法。

例如,每个用户都预先约定好一个过程或者函数,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者函数进行计算,系统根据计算结果辨别用户身份的合法性。这种方法称为透明公式鉴别法。

例如,让用户记住一个表达式,如 $T=X+X*Y+3*Y$,系统告诉用户 $X=1$、$Y=2$,如果用户回答 $T=9$,则证实了该用户的身份。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。系统每次提供不同的 X、Y 值,其他人可能看到的是 X、Y 的值,但不能推算出确切的变换公式 T,从而无法得到 T 的正确值而不被系统证实。

2. 用户存取权限控制

用户存取权限指的是不同的用户对于不同的数据对象允许执行的操作权限。在数据库系统中,每个用户都只能访问他有权存取的数据并执行有权使用的操作。因此,必须预先定义用户的存取权限。对于合法的用户,系统根据其存取权限的定义对其各种操作请求进行控制,确保其合法操作。

存取权限由两个要素组成,数据对象和操作类型。定义一个用户的存取权限就是定义这个用户可以在哪些数据对象上进行哪些类型的操作。

在数据库系统中,定义用户存取权限称为授权(Authorization)。第 3 章讨论 SQL 的数据控制功能时,授权有两种:系统权限和对象权限。系统权限是由 DBA 授予某些数据库用户的,只有得到系统权限,才能成为数据库用户。对象权限可以由 DBA 授予,也可以由数据对象的创建者授予,使数据库用户具有对某些数据对象进行某些操作的权限。在系统初始化时,系统中至少有一个具有 DBA 权限的用户,DBA 可以通过 GRANT 语句将系统权限或对象权限授予其他用户。对于已授权的用户可以通过 REVOKE 语句收回所授予的权限。

这些授权定义经过编译后以一张授权表的形式存放在数据字典中。授权表主要有三个属性,用户标识、数据对象和操作类型。用户标识不但可以是用户个人,也可以是团体、程序和终端。在非关系系统中,存取控制的数据对象仅限于数据本身。而在关系系

统中，存取控制的数据对象不仅有基本表、属性列等数据本身，还有内模式、外模式、模式等数据字典中的内容。表 5.1 列出了关系系统中的存取权限。

对于授权表，一个衡量授权机制的重要指标就是授权粒度，即可以定义的数据对象的范围，在关系数据库中，授权粒度包括关系、记录或属性。

一般来说，授权定义中粒度越细，授权子系统就越灵活。

例如，表 5.2 是一个授权粒度很粗的表，它只能对整个关系授权。U1 拥有对关系 S 的一切权限；U2 拥有对关系 C 的 SELECT 权和对关系 SC 的 UPDATE 权；U3 只可以向关系 SC 中插入新记录。

表 5.1 关系系统中的存取权限

数据对象		操作类型
模式	模式	建立、修改、检索
	外模式	建立、修改、检索
	内模式	建立、修改、检索
数据	表	查询、插入、修改、删除
	属性列	查询、插入、修改、删除

表 5.2 授权表 1

用户标识	数据对象	操作类型
U1	关系 S	ALL
U2	关系 C	SELECT
U2	关系 SC	UPDATE
U3	关系 SC	INSERT
…	…	…

表 5.3 是一个授权粒度较为精细的表，它可以精确到关系的某一属性。U1 拥有对关系 S 的一切权限；U2 只能查询关系 C 的 CNO 属性和修改关系 SC 的 SCORE 属性；U3 可以删除关系 SC 中的记录。

表 5.3 授权表 2

用户标识	数据对象	操作类型	用户标识	数据对象	操作类型
U1	关系 S	ALL	U3	关系 SC	DELETE
U2	关系 C.CNO	SELECT	…	…	…
U2	关系 SC.SCORE	UPDATE			

衡量授权机制的另一个重要方面就是授权表中权限表示能力的强弱。在表 5.2 和表 5.3 的授权表中的授权只涉及关系、记录或属性的名字，而未提到具体的值。系统不必访问具体的数据本身，就可以决定执行这种控制。这种控制称为“值独立”的控制。

表 5.4 中的授权表则不但可以对属性列授权，还可以提供与数值有关的授权，即可以对关系中的一组满足存取谓词的记录授权。比如，U1 只能对非计算机系的学生进行操作。

对于提供与数据值有关的授权，系统必须能够支持存取谓词的操作。

一方面，授权粒度越细，授权子系统就越灵活，能够提供的安全性就越完善。但另一方面，如果用户比较多，数据库比较大，授权表将很大，而且每次数据库访问都要用到这张表做授权检查，这将影响数据库的性能。

表 5.4 授权表 3

用户标识	数据对象	操作类型	存取谓词
U1	关系 S	ALL	DEPT<>计算机
U2	关系 C. CNO	SELECT	
U2	关系 SC. SCORE	UPDATE	
U3	关系 SC	DELETE	
…	…	…	

所幸的是，在大部分数据库中，需要保密的数据是少数，对于大部分公开的数据，可以一次性地授权给 PUBLIC，而不必对每个用户逐个授权。对于表 5.4 中与数据值有关的授权，可以通过另外一种数据库安全措施，即定义视图来实现安全保护。

3. 视图机制

为不同的用户定义不同的视图，可以限制各个用户的访问范围。通过视图机制把要保密的数据对无权存取这些数据的用户隐藏起来，从而自动地对数据提供一定程度的安全保护。

例如，U1 只能对非计算机系的学生进行操作，一种方法是通过授权机制对 U1 授权，如表 5.4 所示，另一种简单的方法就是定义一个非计算机系学生的视图。但视图机制的安全保护功能不够全面，往往不能达到应用系统的要求，其主要功能在于提供数据库的逻辑独立性。在实际应用中，通常将视图机制与授权机制结合起来使用，首先用视图机制屏蔽一部分保密数据，然后在视图上面再进一步定义存取权限。

4. 数据加密

前面介绍的几种数据库安全措施，都是防止从数据库系统窃取保密数据，而不能防止通过不正常渠道非法访问数据，例如，偷取存储数据的磁盘，或在通信线路上窃取数据，为了防止这些窃密活动，比较好的办法是对数据加密。数据加密(Data Encryption)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(术语为明文，Plain text)加密成为不可直接识别的格式(术语为密文，Cipher text)，数据以密码的形式存储和传输。

加密方法有两种，一种是替换方法，该方法使用密钥(Encryption Key)将明文中的每一个字符转换为密文中的一个字符。另一种是转换方法，该方法将明文中的字符按不同的顺序重新排列。通常将这两种方法结合起来使用，这样就可以达到相当高的安全程度。例如，美国 1977 年制订的官方加密标准，数据加密标准(Data Encryption Standard，DES)就是使用这种算法的。

数据加密后，对于不知道解密算法的人，即使利用系统安全措施的漏洞非法访问数据，也只能看到一些无法辨认的二进制代码。合法的用户在检索数据时，首先提供密码钥匙，由系统进行译码后，才能得到可识别的数据。

目前不少数据库产品提供了数据加密例行程序，用户可根据要求自行进行加密处理，还有一些未提供加密程序的产品也提供了相应的接口，允许用户用其他厂商的加密程序对数据加密。实际上，有些系统也支持用户自己设计加、解密程序，只是这样对用户提出了更高的要求。

用密码存储数据，在存入时需加密，在查询时需解密，这个过程会占用较多的系统资源，降低了数据库的性能。因此数据加密功能通常允许用户自由选择，只对那些保密要求特别高的数据，才值得采用此方法。

5. 审计

前面介绍的各种数据库安全性措施，都可将用户操作限制在规定的安全范围内。但实际上任何系统的安全性措施都不是绝对可靠的，窃密者总有办法打破这些控制。对于某些高度敏感的保密数据，必须以审计(Audit)作为预防手段。审计功能是一种监视措施，用来跟踪记录有关数据的访问活动。

审计追踪把用户对数据库的所有操作自动记录下来，存放在一个特殊文件中，即审计日志(Audit Log)。记录的内容一般包括操作类型，如修改、查询等；操作终端标识与操作者标识；操作日期和时间；操作所涉及的相关数据，如基本表、视图、记录、属性等；数据的前象和后象。利用这些信息，可以重现导致数据库现有状况的已发生的一系列事件，以进一步找出非法存取数据的人、时间和内容等。

使用审计功能会大大增加系统的开销，所以 DBMS 通常将其作为可选特征，提供相应的操作语句可灵活地打开或关闭审计功能。

例如，可使用如下 SQL 语句打开对表 S 的审计功能，对表 S 的每次成功的增加、删除、修改操作都作审计追踪。

```
AUDIT INSERT,DELETE,UPDATE ON S WHENEVER SUCCESSFUL
```

要关闭对表 S 的审计功能可以使用如下语句：

```
NO AUDIT ALL ON S
```

5.1.3 安全性控制的其他方法

1. 强制存取控制

有些数据库系统的数据要求很高的保密性，通常具有静态的严格的分层结构，强制存取控制(MAC)能实现这种高保密性要求。这种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，级别从高到低分为绝密级、机密级、秘密级和公用级。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，如绝密＞机密＞秘密＞公用。

在系统运行时，采用如下两条简单规则：①用户 U 只能查看比它级别低或同级的数据；②用户 U 只能修改和它同级的数据。根据第②条规则，用户 U 显然不能修改比它级别高的数据，也不能修改比它级别低的数据，这样主要是为了防止具有较高级别的用户

将该级别的数据复制到较低级别的文件中。

强制存取控制是一种独立于值的控制方法。它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，允许凡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，使无权用户也可接触保密数据。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的应用并不广泛，只是在某些专用系统中才有用。

2. 统计数据库的安全性

有一类数据库称为“统计数据库”，例如，民意调查数据库，它包含大量的记录，但其目的只是向公众提供统计、汇总信息，而不是提供单个记录的内容。也就是说所查询的仅仅是某些记录的统计值，如求记录数、和、平均值等。在统计数据库中，虽然不允许用户查询单个记录的信息，但是用户可以通过处理足够多的汇总信息来分析出单个记录的信息，这就给统计数据库的安全性带来了严重的威胁。

设有一个职工关系，包含工资信息。一般的用户只能查询统计数据，而不能查看个别的记录。有一个姓刘的用户欲窃取姓张的工资数目。他可通过下面两步实现。

(1) 用 SELECT 命令查找自己和其他 $n-1$ 个人(例如 30 岁的女职工)的工资总额 A。

(2) 用 SELECT 命令查找姓张用户和上述同样的 $n-1$ 个人的工资总额 B。

随后，刘可以很方便地通过下列式子得到张的工资数是：

$$B-A+\text{“姓刘用户自己的工资数”}$$

这样，用户刘就窃取到了张的工资数目。统计数据库应防止上述问题的发生。上述问题产生的原因是两个查询包含了许多相同的信息(即两个查询的“交”)。系统应对用户查询得到的记录数加以控制。

在统计数据库中，对查询应作下列限制：①一个查询查到的记录个数至少是 n；②两个查询查到的记录的“交”数目至多是 m。

系统可以调整 n 和 m 的值，使得用户很难在统计数据库中获取其他个别记录的信息，但要做到完全杜绝是不可能的。应限制用户计算和、个数、平均值的能力。如果一个破坏者只知道他自己的数据，那么他至少要花 $1+(n-2)/m$ 次查询才有可能获取其他个别记录的信息。因而，系统应限制用户查询的次数在 $1+(n-2)/m$ 次以内。但是这个方法还是不能防止两个破坏者联手查询导致数据的泄露。

保证数据库安全性的另一个方法是“数据污染”，也就是在回答查询时，提供一些偏离正确值的数据，以免数据泄露。当然，这个偏离要在不破坏统计数据的前提下进行。此时，系统应该在准确性和安全性之间作出权衡。当安全性遭到威胁时，只能降低准确性的标准。

5.1.4 SQL Server 安全性概述

SQL Server 安全系统的构架建立在用户和用户组的基础上。Windows(这里的 Windows 代表 Windows NT 4.0、Windows 2000 及以上版本，下同)中的用户和本地组

及全局组可以映射到 SQL Server 中的安全账户，也可以创建独立 Windows 账户的安全账户。

SQL Server 提供了三种安全管理模式，即标准模式、集成模式和混合模式，数据库设计者和数据库管理员可以根据实际情况进行选择。

1. 两个安全性阶段

在 SQL Server 中工作时，用户要经过两个安全性阶段：身份验证和授权(权限验证)。授权阶段使用登录账户标识用户并只验证用户连接 SQL Server 实例的能力。如果身份验证成功，那么用户即可连接到 SQL Server 实例。然后用户需具有访问服务器上数据库的权限，为此需授予每个数据库中映射到用户登录的账户访问权限。权限验证阶段控制用户在 SQL Server 数据库中所允许进行的活动。

每个用户都必须通过登录账户建立自己的连接能力(身份验证)，以获得对 SQL Server 实例的访问权限。然后，该登录必须映射到用于控制在数据库中所执行的活动(权限验证)的 SQL Server 用户账户。如果数据库中没有用户账户，则即使用户能够连接到 SQL Server 实例，也无法访问该数据库。

2. 用户权限

登录创建在 Windows 中，而非 SQL Server。该登录随后被授予连接到 SQL Server 实例的权限。该登录在 SQL Server 内被授予访问权限。

当用户连接到 SQL Server 实例后，他们可以执行的活动由授予以下账户的权限来确定：①用户的安全账户；②用户的安全账户所属 Windows 组或角色层次结构；③用户若要进行任何涉及更改数据库定义或访问数据的活动，则必须有相应的权限。

管理权限包括授予或废除执行以下活动的用户权限：①处理数据和执行过程(对象权限)；②创建数据库或数据库中的项目(语句权限)；③利用授予预定义角色的权限(暗示性权限)。

3. 视图安全机制

SQL Server 可限制用户使用的数据，可以将视图作为安全机制。用户可以访问某些数据，并进行查询和修改操作，但是表或数据库的其余部分是不可见的，也不能进行访问。对 SQL Server 来说，无论在基础表(一个或多个)上的权限集合有多大，都必须授予、拒绝或废除访问视图中数据子集的权限。

4. 加密方法

SQL Server 支持加密或可以加密的内容包括①SQL Server 中存储的登录和应用程序角色密码；②作为网络数据包而在客户端和服务器端之间发送的数据；③SQL Server 中定义内容包括存储过程、用户定义函数、视图、触发器、默认值、规则等。

例如，SQL Server 可使用安全套接字层(SSL)加密在应用程序计算机和数据库计算机上的 SQL Server 实例之间传输的所有数据。

5. 审核活动

SQL Server 提供审核功能，用以跟踪和记录每个 SQL Server 实例上已发生的活动（如成功和失败的记录）。SQL Server 还提供管理审核记录的接口，即 SQL 事件探查器。只有 sysadmin 固定安全角色的成员才能启用或修改审核，而且审核的每次修改都是可审核的事件。

5.2 完整性控制

5.2.1 数据库完整性概述

数据库的完整性是指保护数据库中数据的正确性、有效性和相容性，防止错误的数据进入数据库造成无效操作。

有关完整性的含义在第 1 章中已作了简要介绍。比如年龄属于数值型数据，只能含有 0,1,…,9，不能包括字母或特殊符号；月份只能用 1～12 之间的正整数表示；表示同一事实的两个数据应相同，否则就不相容，例如，一个人不能有两个学号。

显然，维护数据库的完整性非常重要，数据库中的数据是否具备完整性关系到数据能否真实地反映现实世界。

数据库的完整性和安全性是数据库保护的两个不同的方面。

安全性是指保护数据库，以防止非法使用所造成数据的泄露、更改或破坏，安全性措施的防范对象是非法用户和非法操作；完整性是指防止合法用户使用数据库时向数据库中加入不符合语义的数据，完整性措施的防范对象是不合语义的数据。

但从数据库的安全保护角度来讲，安全性和完整性是密切相关的。

5.2.2 完整性规则的组成

为了实现完整性控制，数据库管理员应向 DBMS 提出一组完整性规则，用来检查数据库中的数据，看其是否满足语义约束。这些语义约束构成了数据库的完整性规则，这组规则作为 DBMS 控制数据完整性的依据。它定义了何时检查、检查什么、查出错误又怎样处理等事项。具体地说，完整性规则主要由以下三部分构成。

(1) 触发条件：规定系统什么时候使用规则检查数据；

(2) 约束条件：规定系统检查用户发出的操作请求违背了什么样的完整性约束条件；

(3) 违约响应：规定系统如果发现用户的操作请求违背了完整性约束条件，应该采取一定的动作来保证数据的完整性，即违约时要做的事情。

完整性规则从执行时间上可分为立即执行约束(Immediate Constraints)和延迟执行约束(Deferred Constraints)。

立即执行约束是指在执行用户事务过程中，某一条语句执行完成后，系统立即对此

数据进行完整性约束条件检查；延迟执行约束是指在整个事务执行结束后，系统再对约束条件进行完整性检查，结果正确后才能提交。

例如，银行数据库中“借贷总金额应平衡”的约束就应该属于延迟执行约束，从账号A转一笔钱到账号B为一个事务，从账号A转出去钱后，账就不平了，必须等转入账号B后，账才能重新平衡，这时才能进行完整性检查。

如果发现用户操作请求违背了立即执行约束，则可以拒绝该操作，以保护数据的完整性；如果发现用户操作请求违背了延迟执行约束，而又不知道是哪个事务的操作破坏了完整性，则拒绝整个事务，把数据库恢复到该事务执行前的状态。

一条完整性规则可以用一个五元组(D,O,A,C,P)来形式化地表示。其中具体说明如下。

D(data)：代表约束作用的数据对象；

O(operation)：代表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；

A(assertion)：代表数据对象必须满足的语义约束，这是规则的主体；

C(condition)：代表选择A作用的数据对象值的谓词；

P(procedure)：代表违反完整性规则时触发执行的操作过程。

例如，对于“学号不能为空”的这条完整性约束，具体说明如下。

D：代表约束作用的数据对象为SNO属性；

O：当用户插入或修改数据时需要检查该完整性规则；

A：SNO不能为空；

C：A可作用于所有记录的SNO属性；

P：拒绝执行用户请求。

关系模型的完整性包括实体完整性、参照完整性和用户定义完整性。

对于违反实体完整性和用户定义完整性规则的操作，一般是采用拒绝执行的方式进行处理的。而对于违反参照完整性的操作，并不都是简单地拒绝执行，一般在接受这个操作的同时，执行一些附加的操作，以保证数据库的状态仍然是正确的。

例如，在删除被参照关系中的元组时，应该将参照关系中所有的外码值与被参照关系中要删除元组主码值相对应的参照关系中的元组一起删除。

比如，要删除S关系中SNO='S2'的元组，而SC关系中又有两个SNO='S2'的元组。这时根据应用环境的语义，因为当一个学生毕业或退学后，他的个人记录从S关系中删除，选课记录也应随之从SC表中删除，所以应该将SC关系中所有SNO='S2'的元组同时删除。

这些完整性规则都由DBMS提供的语句进行描述，经过编译后存放在数据字典中。数据进出数据库系统，这些规则就开始起作用，用于保障数据的正确性。

这样做的优点包括(1)完整性规则的执行由系统来处理，而不是由用户处理；(2)规则集中在数据字典中，而不是散布在各应用程序之中，易于从整体上理解和修改，效率较高。

数据库系统的整个完整性控制都是围绕着完整性约束条件进行的，从这个角度来