第6章 数据库的管理

数据库管理的主要目的是，防止不合法用户对数据库进行非法操作，实现数据库的平安性；防止不合法数据进入数据库，实现数据库的完整性；防止并发操作产生的事务不一致性，进行并发控制；防止计算机系统硬件故障、软件错误、操作操作等所造成的数据丧失，采取必要的数据备份措施，并能从错误状态恢复到正确状态。

本章从数据库管理系统的角度讲述数据库管理的原理和方法，主要介绍数据库的平安性、完整性、并发控制和恢复技术，并以 SQL Server 为例进行具体说明。

6.1 数据库的平安性

6.1.1 数据库平安性概述

数据库的平安性是指保护数据库以防止非法用户访问数据库，造成数据泄露、更改或破坏。在数据库系统中大量数据集中存放，并为许多用户直接共享，数据库的平安性相对于其它系统尤其重要。实现数据库的平安性是数据库管理系统的重要指标之一。

数据库的平安性不是孤立的。在网络环境下，数据库的平安性与三个层次相关：网络系统层、操作系统层、数据库管理系统层。这三层共同构筑起数据库的平安体系，它们与数据库的平安性逐步紧密，重要性逐层加强，从外到内保证数据库的平安性。在规划和设计数据库的平安性时，要综合每一层的平安性，使三层之间相互支持和配合，提高整个系统的平安性。在此只讨论数据库管理系统对数据库进行平安管理的问题，网络系统层的平安性、操作系统层的平安性不做介绍。

影响数据库平安性的因素很多，不仅有软硬件因素，还有环境和人的因素；不仅涉及技术问题，还涉及管理问题、政策法律问题等等。其内容包括计算机平安理论、策略、技术，计算机平安管理、评价、监督，计算机平安犯罪、侦察、法律等等。概括起来，计算机系统的平安性问题可分为三大类，即：技术平安类、管理平安类和政策法律类。此处只在技术层面介绍数据库的平安性。

为了准确地测定和评估计算机系统的平安性能指标，标准和指导计算机系统的生产，逐步建立和开展了一套“可信计算机系统评测标准”。其中最重要的是 1985 年美国国防部(DoD)公布的《DoD 可信计算机系统评估标准》(Trusted Computer System Evaluation Criteria,简称 TCSEC，称桔皮书)和 1991 年美国国家计算机平安中心(NCSC)公布的《可信计算机系统评估标准关于可信数据库系统的解释》(TrustedDatabase Interpretatution,简称 TDI，称紫皮书)。TDI 将 TCSEC 扩展到数据库管理系统，定义了数据库管理系统的设计与实现中需要满足和用以进行平安性级别评估的标准。

TCSEC/TDI 将系统划分为 DCBA 四组，D;C1;C2;B1;B2;B3;A1 从低到高七个等级。较高平安等级提供的平安保护要包含较低等级的所有保护要求，同时提供更多更完善的保护能力。七个平安等级的根本要求如下：

D 级：提供最小保护(MinimalProtection)。可以将不符合更高标准的系统归于 D 级。如：DOS 就是操作系统中平安标准为 D 的典型例子，它具有操作系统的根本功能，如文件系统，进程调度等等，但在平安性方面几乎没有什么专门的机制来保障。

C1 级：提供自主平安保护(DiscretionarySecurityProtection)。实现用户和数据的别离，进行自主存取控制(Discretionary Access Control，简记为 DAC)，保护和限制用户权限的传播。

C2 级：提供受控的存取保护(ControlledAccessProtection)。将 C1 级的 DAC 进一步细化，以个人身份注册负责，并实施审计和隔离，是平安产品的最低档次。

B1 级：标记平安保护(LabeledSecurityProtection)。对系统的数据加以标记，并对标记的主体和客体实施强制存取控制(Mandatory Access Control,简记为 MAC)。B1 级能够较好地满足大型企业或一般政府部门对于数据的平安需求，这一级别的产品才认为是真正意义上的平安产品。满足此级别的产品多冠以“平安”(Security)或“可信的”

(Trusted)字样，作为区别于普通产品的平安产品出售。在数据库管理系统方面有 Oracle 公司的 Trusted Oracle 7，Sybase 公司的 Secure SQL Server version 11.0.6，Informix 公司的 Incorporated INFORMIX-OnLine/Secure5.0。

B2 级：结构化保护(Structural Protection)。建立形式化的平安策略模型并对系统内的所有主体和客体实施 DAC 和 MAC。到达 B2 级的系统非常稀少，在数据库方面暂时没有此级别的产品。

B3 级：平安域保护(Security Domains)。该级的可信任运算根底(Trusted Computing Base，TCB)必须满足访问监控器的要求，审计跟踪能力更强，并提供系统恢复过程。

A1 级：验证设计(Verified Design)。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平安保护真正实现。

从分级标准可以看出，支持自主存取控制〔DAC〕的 DBMS 属于 C1 级，支持审计功能的 DBMS 属于 C2 级，支持强制存取控制〔MAC〕的 DBMS 那么可以到达 B1 级。B2 以上的系统标准更多地还处于理论研究阶段，产品化以至商品化的程度都不高，其应用也多限于一些特殊的部门如军队等。下面以 B1 级标准中的用户标识和鉴别、存取控制〔DAC 和 MAC〕、审计等功能进行介绍。

6.1.2 用户标识和鉴别

用户标识(Identification)和鉴别(Authentication 是数据库系统提供的最外层平安保护措施。其方法是由系统提供一定的方式让用户标识自己的身份，每次用户要求进入系统时，通过鉴别后才提供系统使用权。

用户标识的鉴别方法有多种途径，可以委托操作系统进行鉴别，可以委托专门的全局验证效劳器进行鉴别。一般数据库管理系统提供了用户标识和鉴别机制。

用一个用户名或者用户标识号来标明用户身份。系统内部记录着所有合法用户的标识，系统鉴别此用户是否合法，假设是，那么进入口令的核实；假设不是，那么不能使用系统。

口令(Password)，为了进一步鉴别用户，系统常常要求用户输入口令。为保密起见，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。

通过用户名和口令来鉴定用户的方法简单易行，但用户名与口令容易被人窃取，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，系统根据用户计算结果是否正确进一步鉴定用户身份。用户可以约定比拟简单的计算过程或函数，以便计算起来方便；也可以约定比拟复杂的计算过程或函数，以便平安性更好。

6.1.3 存储控制

数据库平安主要是指 DBMS 的存取控制机制。数据库平安最重要的一点就是确保只有授权用户访问数据库，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机制实现。

存取控制机制主要包括两局部：

(1)用户权限定义。用户权限是指不同的用户对于不同的数据对象允许执行的操作权限。系统必须提供适当的语言定义用户权限，这些定义经过编译后存放在数据字典中，被称作平安规那么。

(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 查找数据字典，根据平安规那么进行合法权限检查，假设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。

用户权限定义和合法权限检查一起组成了 DBMS 的平安子系统，支持自主存取控制(DAC)和强制存取控制(MAC)。这两类方法的简单定义是：

(1)自主存取控制 DAC 方法：用户对于不同的数据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。

(2)强制存取控制 MAC 方法：每一个数据对象被标以一定的密级，每一个用户也被授予某一个级别的许可证。对于任意一个对象，只有具有合法许可证的用户才可以存取。强制存取控制因此相比照拟严格。

1. 自主存取控制

自主存取控制 DAC 是在用户标识和鉴别的根底上，对用户进行授权。大型数据库管理系统几乎都支持 C1 级的自主存取控制，目前的 SQL 标准也对自主存取控制提供支持，这主要通过 SQL 的 GRANT 语句和 REVOKE 语句来实现。

用户权限是由四个要素组成：权限授出用户(Grantor)，权限接受用户(grantee)、操作数据对象(Object)、操作类型(Operate)。定义一个用户的存取权限就是要权限授出用户定义权限接受用户可以在哪些数据对象上进行哪些类型的操作。在数据库管理系统中,定义存取权限称为授权(Authorization)。

数据对象的建立者(拥有者)和超级用户(DBA)自动拥有数据对象的所有操作权限，包括权限授出的权限；接受权限用户可以是系统中标识的任何用户；数据对象不仅有表和属性列等数据本身，还有模式、外模式、内模式等数据字典中的内容，常见的数据对象有根本表(Table)、视图(View)、过程(Procedure)等;操作类型有建立(CREATE)、增加(INSERT)、修改(UPDATE/ALTER)、删除(DELETE/DROP)、检索(SELECT)等。

例如：根本表 student 的建立者 usera 使用 GRANT 将 student 表的操作权限授予不同的用户，使用 REVOKE 收回权限。

```
GRANT SELECT ON student TO user1;
GRANT INSERT,UPDATE,DELETE ON student TO user2;
GRANT SELECT,INSERT,UPDATE,DELETE ON student TO user3;
REVOKE SELECT ON student FROM user1;
REVOKE INSERT ON student FROM user2;
REVOKE ALL ON student FROM user3;
```

用户权限定义中数据对象范围越小授权就越灵活。有的系统可精细到字段级，而有的系统只能对关系授权。授权粒度越细，系统定义与检查权限的开销也会相应地增大。关系数据库中授权的数据对象横向粒度有：数据库、表、属性列等。

衡量授权精巧程度的另一个尺度是从向粒度，能否提供与数据值有关的授权。一般的授权定义是独立于数据值的，即用户能否对某类数据对象执行的操作与数据值无关，完全由数据名决定。假设授权依赖于数据对象的内容，那么称为是与数据值有关的授权。

实现与数据值有关的授权可以使用存取谓词，定义存取的条件，限制存取的记录。同时也可以使用系统变量，限制存取的时间、存取使用的终端等等。

数据库管理系统一般都支持视图数据对象,使用视图机制实现属性列的授权和与数据值有关的授权。将属性列的存取限制和数据值的存取限制定义在适宜的视图中，然后针对视图进行授权。例如：student 表的建立者 usera 建立视图 v_student 并授权 userb 查询权限,其中属性列只选择 sno(学号),sname(姓名),sclass(班级)，其它列不能存取；只选取系部编号为"0001"的记录，其它记录不能存取。

```
CREATE VIEW v_student AS
    SELECT sno,sname,sclass FROM student WHERE sdept="0001";
GRANT SELECT ON v_student TO userb;
```

自主存取控制能够通过授权机制有效地控制其他用户对敏感数据的存取。但是由于用户对数据的存取权限是“自主”的，用户可以自由地决定将数据的存取权限授予其它用户。在这种授权机制下，仍可能存在数据的“无意泄露”。如：用户 usera 将数据对象权限授予用户 userb,usera 的意图是只允许 userb 操纵其这些数据，但是 userb 可以在 usera 不知情的情况下进行数据备份并进行传播。出现这种问题的原因是，这种机制仅仅通过限制存取权限进行平安控制，而没有对数据本身进行平安标识。解决这一问题需要对所有数据进行强制存取控制。

2. 强制存取控制

强制存取控制 MAC 是指系统为保证更高程度的平安性，按照 TDI/TCSEC 标准中平安策略的要求，所采取的强制存取检查手段。它不是用户能直接感知或进行控制的。MAC 适用于那些对数据有严格而固定密级分类的部门，例如军事部门或政府部门。

在 MAC 中，DBMS 所管理的全部实体被分为主体和客体两大类。

主体是系统中的活动实体，既包括 DBMS 所管理的实际用户，也包括代表用户的各进程。客体是系统中的被动实体，是受主体操纵的，包括文件、根本表、索引、视图等等。对于主体和客体，DBMS 为它们指派一个敏感度标记(Label)。

敏感度标记被分成假设干级别，例如绝密(Top Secret)、机密(Secret)、可信(Confidential)、公开(Public)等。主体的敏感度标记称为许可证级别(Clearance Level)，客体的敏感度标记称为密级〔Classification Level〕。MAC 机制就是通过比照主体的 Label 和客体的 Label，最终确定主体是否能够存取客体。

当某一用户或某一主体以标记 label 注册进入系统时，系统要求他对任何客体的存取必须遵循如下规那么：

(1)仅当主体的许可证级别大于或等于客体的密级时，该主体才能读取相应的客体；

(2)仅当主体的许可证级别等于客体的密级时，该主体才能写相应的客体。

在某些系统中，与第 2 条规那么有些差异：仅当主体的许可证级别小于或等于客体的密级时，该主体才能写相应的客体，即用户可以为写入的数据对象赋予高于自己的许可证级别的密级。这样一旦数据被写入，该用户自己也不能再读该数据对象了。这两种规那么的共同点在于它们均禁止了拥有高许可证级别的主体更新低密级的数据对象，从而防止了敏感数据的泄漏。

强制存取控制(MAC)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平安性。

在实现 MAC 时要首先实现 DAC，即 DAC 与 MAC 共同构成 DBMS 的平安机制。系统首先进行 DAC 检查，对通过 DAC 检查的允许存取的数据对象再由系统自动进行 MAC 检查，只有通过 MAC 检查的数据对象方可存取。

在有些系统中，为了保护数据本身的平安性，采用了数据加密技术，对高度敏感数据进行保护。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。加密的根本思想是根据一定的算法将原始数据(明文,Plain text)变换为不可直接识别的格式(密文，Cipher text)，从而使得不知道解密算法的人即使获取了密文无法获知原文。数据加密与解密是比拟费时，占用较多的系统资源，DBMS 往往都将其作为可选特征，允许 DBA 根据应用对平安性的要求，自由选择，只对高度机密的数据如：财务数据、军事数据、国家机密等数据进行加密。

6.1.5 审计管理

按照 TDI/TCSEC 标准中平安策略的要求，“审计”功能就是 DBMS 到达 C2 以上平安级别必不可少的一项指标。

因为任何系统的平安保护措施都不是完美无缺的，蓄意盗窃、破坏数据的人总是想方设法打破控制。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。DBA 可以利用审计跟踪的信息，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。

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 根据应用对平安性的要求，灵活地翻开或关闭审计功能。审计功能一般主要用于平安性要求较高的部门。

6.1.6 SQL Server 平安管理

SQLServer 平安性建立在认证和访问许可的机制上。如果一个用户访问 SQLServer 数据库中的数据，必须经过三个认证过程。第一个过程是登录认证，发生在用户连接数据库效劳器时，决定用户是否有连接到数据库效劳器的资格，验证用户连接数据库效劳器的连接权。第二个认证过程是用户认证，发生在用户访问数据库时，决定用户是否为数据库的合法用户，验证用户对数据库的访问权。第三个认证过程是权限认证，发生在用户操作数据库对象时，决定用户是否有对象操作许可，验证用户操作权。所以 SQL Server 的平安级别有三级：

第一级：数据库效劳器，使用登录认证。属于数据库效劳器级别的用户标识和鉴别。

第二级：数据库，使用用户认证。属于数据库级别的用户标识和鉴别

第三级：数据库对象，使用权限认证。属于 DAC 方法。

1. SQL Server 登录管理

(1)登录认证模式

SQL Server 提供了两种登录认证模式：

Windows 认证：由 Windows 操作系统负责登录认证。Windows 创立、管理 Windows 帐户，由 Windows 授权连接 SQL Server，并将 Windows 帐户映射为 SQL Server 登录。运行在 Windows95/98 的 SQL Server 不能采用 Windows 验证。

SQL Server 认证：由 SQL Server 本身负责登录认证。SQL Server 负责创立、管理登录，并将登录保存在数据库中。这时 SQL Server 的登录与 Windows 帐户无关。

(2)登录属性

效劳器登录属性有登录名称、密码、默认数据库、访问数据库、效劳器角色、语言、平安标识号、加密选项。

登录名称和密码：如果是 Windows 认证，登录名称必须是 Windows 帐号，其帐号、密码由 Windows 操作系统保存，但在 SQLServer 中需要指明 Windows 帐号(包括域名或组名或主机名,由 Windows 主机隶属于域或组决定，该处统一用域名表示)，不需要设置密码。如果是 SQLServer 认证，由 SQLServer 创立登录名称，同时设置其密码，其名称和密码保存在 SQL Server 数据库中。

默认数据库：是指该登录连接数据库效劳器后，其所属用户默认访问的数据库，缺省为主数据库 master。

访问数据库：是指该登录连接数据库效劳器后，其所属用户可以访问的数据库，默认数据库必须为访问数据库。

效劳器角色：是指该登录所属的效劳器角色，指明登录的效劳器权限。一般只有管理登录才赋予效劳器角色。有关“效劳器角色”，参考后面“角色管理”。

语言：是指该登录使用的语言，缺省为 SQL Server 语言设置。

平安标识号 SID：是指该登录在 SQL Server 效劳器内部使用的唯一标识号。管理员使用平安标识号确定登录，操作员一般不使用。平安标识号一般由 SQL Server 在创立登录或将 Windows 帐户映射为 SQL Server 登录时自动指定，很少人工指定。

加密选项： 是指该登录密码的是否加密，只对 SQL Server 认证有效。NULL 表示加密密码；Skip_Encryption 表示不对密码加密，Skip_Encryption_Old 表示在以前版本中加密的密码不再加密。

(3)创立登录

SQL Server 提供了企业管理器(Enterprise Manager)图形界面工具建立和维护各种对象及属性的方法，操作简单方便，这也是其优势。同时还有 T-SQL 方法、有些稍微复杂的对象还提供了建立或维护向

导(Wizard)。本书只介绍 T-SQL 方法。

SQL Server 提供了 sp_addlogin 和 sp_grantlogin 存储过程用来创立登录(login)。其中 sp_addlogin 创立 SQL Server 认证登录，sp_grantlogin 映射 Windows 认证的 Windows 帐号为登录。

sp_addlogin 语法如下：

```
sp_addlogin
    [@loginname  =] 'login'
    [@passwd     =] 'password'
  [,[@defdb      =] 'database']
  [,[@deflanguage=] 'language']
  [,[@sid        =] 'sid']
  [,[@encryptopt =] 'encryption']
```

其中

```
  login     :登录名称
  password  :登录密码，其中登录名称和密码不能省略。
  database  :默认数据库，缺省为 master
  language  :登录语言，缺省为 SQL Server 设置语言
  sid       :平安标识符，一般省略，由 SQL Server 自动生成
  encryption:密码加密选项，可设置 NULL|Skip_Encryption|Skip_Encryption_Old。
```

sp_grantlogin 语法如下：

```
sp_grantlogin [@loginame =] 'login'
```

其中 login 是要添加的 Windows 用户名称。Windows 用户必须用 Windows 域名限定，格式为"域\用户"。语法中，[]标识可选项。

(4)维护登录

sp_helplogins:查询登录名 LoginName、SID 平安标识号、默认数据库 DefDBName、登录语言 DefLangName、访问的数据库 DBName、映射的用户名 UserName。

sp_droplogin:删除 SQLServer 认证和 Windows 认证的登录，但不能删除系统管理员 sa 登录，不能删除正在连接效劳器的登录。Windows 认证的登录删除不影响其在 Windows 中的帐户信息和行为。

sp_denylogin:阻止 Windows 认证的帐户连接到 SQL Server 效劳器。只能用于 Windows 认证登录，但不影响其在 Windows 中的帐户信息和行为。

sp_password:更改登录口令。

sp_addsrvrolemember:增加登录的效劳器角色。

(5)默认登录

在安装 SQL Server 数据库效劳器时，SQL Server 自动创立了默认的登录：

BUILTIN\Administrators:Windows 认证的 Administrator 组的所有帐户。默认效劳器角色为 System Administrators。

域名\Administrator:Windows 认证的 Administrator 登录。只有在安装效劳器时指明为 Windows 认证时才创立，默认效劳器角色为 System Administrators。

sa:SQL Server 认证的系统管理员登录，不一定要求是 Windows 管理员。默认效劳器角色为 System Administrators。

2. SQL Server 用户管理

创立了效劳器登录后，只能连接到 SQLServer 数据库效劳器，不能访问任何数据库，只有创立了数据库的用户(user)，成为数据库的合法用户后，才能访问数据库。数据库的用户只能来自于效劳器的登录，而且是可以访问该数据库的登录(在登录中设置访问数据库)。一个效劳器登录可以映射为多个数据库中的用户，但在一个数据库中只能映射为一个用户。

(1)用户属性

用户属性包括用户所属的登录、用户名、数据库角色。

所属的登录:是指该数据库用户所属的效劳器登录。一个数据库用户必须隶属于一个效劳器登录，一个登录在一个数据库中只能有一个用户。

用户名:是指该数据库用户的名称

用户 ID:是指该用户对应的内部标识符，SQL Server 自动产生。

数据库角色:是指该用户拥有的数据库角色，决定该用户操作数据库对象的权限。

(2)增加用户

SQL Server 提供了 sp_adduser 存储过程用来创立用户(user)。语法如下：

sp_adduser

[@loginame =] 'login'

[,[@name_in_db=] 'user']

[,[@grpname =] 'group']

其中：

login：用户所属于的登录，不能省略。

user ：指定的用户名，如果缺省，与登录同名。

group：指定用户所属的数据库角色。如果缺省，默认为 public 数据库角色。

(3)维护用户

sp_helpuser:查询数据库用户名 UserName、数据库用户的数据库角色名 GroupName、所属登录 LoginName、用户平安标识号 SID(UserID)等信息。

sp_dropuser:删除数据库用户，不能删除数据库的 dbo 用户，不能删除 master 和 tempdb 的 guest 用户，不能删除拥有对象的用户。

(4)默认用户

在建立数据库时，SQL Server 自动建立了两个用户：

dbo：数据库的拥有者用户，隶属于 sa 登录，拥有 public 和 db_owner 数据库角色，具有该数据库的所有特权。

guest：客户访问用户，没有隶属的登录。拥有 public 数据库角色。除了 master 和 tempdb 两个数据库的 guest 用户不能删除外，其它数据库的 guest 用户可以删除。

3. SQL Server 权限管理

(1)权限分类

SQL Server 使用权限许可来实现存储控制，SQL Server 权限按等级分为：

系统权限:是指在数据库效劳器级别上对整个效劳器和数据库进行管理的权限,如 shutdown、create database、backup database 等等，效劳器权限以效劳器角色的方式授予管理登录，一般不授予其它登录。效劳器角色 sysadmin 具有全部的系统权限。

对象权限:是指在特定数据库级别上对数据库对象的操作权限,如对某数据库中表的 SELECT、INSERT、UPDATE、DELETE、DRI(DeclarativeReferentialIntegrity:公布的引用完整性)；对存储过程和函数的 EXECUTE 权限等。

语句权限: 用户在数据库中创立表、视图、用户等一类特殊活动的权限为语句权限，语句权限有 BACKUP DATABASE、BACKUP LOG、CREATE DATABASE、CREATE DEFAULT、CREATE FUNCTION、CREATE PROCEDURE、CREATE RULE、CREATE TABLE、CREATE VIEW。

(2)权限的授出和收回及拒绝

不同权限的管理方式略有差异：系统权限只能通过效劳器角色整体地进行授出和收回，不能对具体权限进行授出和收回。对象权限和语句权限可以单独授出、收回、拒绝。授出是授予权限给指定用户；收回是从用户权限中删除权限；拒绝是显式地拒绝用户使用某种权限，以防止用户通过数据库角色继承某些权限。

语句权限的授出：

```
GRANT  {ALL|statement[,...]} TO user[,...]
```

其中 ALL 为所有语句权限，statement 为语句权限，可以是多个语句权限，用逗号分隔；user 为数据库用户，可以一次授予多个用户，用逗号分隔。语法中，{}标识多项选择一。

例如：**GRANT CREATE DATABASE, CREATE TABLE TO Mary, John**

语句权限的收回：

```
REVOKE {ALL|statement[,...]} FROM user[,...]
```

语句权限的拒绝：

```
DENY   {ALL|statement[,...]} FROM user[,...]
```

对象权限的授出：

```
GRANT {ALL [PRIVILEGES]|permission[,...]}
    {(column [,...]) ON {table|view}
     | ON {table|view}
     | ON {stored_procedure|extended_procedure}
     | ON user_defined_function }
    TO user[,...]
    [WITH GRANT OPTION]
```

其中:

ALL:对象的全部权限，PRIVILEGES 为 SQL-92 的标准关键字。

permission:对象的具体权限，可以指定多个。

(column[,...]) ON {table|view}:指定表或视图的列权限，有 SELECT 和 UPDATE。

ON {table|view}:表或视图的权限，有 SELECT、INSERT、UPDATE、DELETE、DRI。

ON {stored_procedure|extended_procedure}:存储过程的权限，只有 EXECUTE。

ON {user_defined_function}}:用户定义的函数权限，只有 SELECT。

TO user[,...]:接受权限的用户，可以指定多个。

WITH GRANT OPTION:接收权限的用户可以将该权限授予其它用户。

例如：**GRANT INSERT, UPDATE, DELETE ON authors TO Mary, John, Tom**

对象权限的收回：

```
REVOKE [GRANT OPTION FOR]
    {ALL [PRIVILEGES]|permission[,...]}
    {(column [,...]) ON {table|view }
     | ON {table|view}
     | ON {stored_procedure|extended_procedure}
     | ON user_defined_function }
    FROM user [,...]
    [ CASCADE ]
```

其中：

GRANT OPTION FOR:收回授出时赋予的 WITH GRANT OPTION 选项作用。

CASCADE:收回通过 WITH GRANT OPTION 权限授出的其它用户权限。

对象权限的拒绝：

```
DENY{ALL [PRIVILEGES]|permission[,...]}
    {(column [,...]) ON {table|view }
     | ON {table|view}
     | ON {stored_procedure|extended_procedure}
     | ON user_defined_function }
    FROM user [,...]
```

[CASCADE]

其中：

CASCADE:拒绝通过 WITH GRANT OPTION 权限授出的其它用户权限。

另外，SQL Server 提供了查询对象权限的存储过程 sp_helprotect。

4. SQL Server 角色管理

(1)角色分类

角色是权限的集合。SQL Server 将局部权限赋予角色，然后将角色赋予用户，简化了权限管理，也便于用户分组。SQL Server 角色分为效劳器角色和数据库角色。

效劳器角色：系统权限的集合。在效劳器级别上可以将效劳器角色赋予登录。效劳器角色是系统创立的，不允许增加和删除，效劳器角色的权限也不允许修改。效劳器角色列表如下：

效劳器角色名	说明
sysadmin	在 SQL Server 中进行任何活动。
serveradmin	配置效劳器范围的设置。
setupadmin	添加和删除链接效劳器。
securityadmin	管理效劳器登录。
processadmin	管理在 SQL Server 实例中运行的进程。
dbcreator	创立和改变数据库。
diskadmin	管理磁盘文件。
bulkadmin	执行 BULK INSERT 语句。

数据库角色：对象权限的集合，在数据库级别上可以将数据库角色赋予用户。数据库建立时，创立了标准的数据库角色。标准数据库角色列表如下：

数据库标准角色	说明
db_owner	进行所有数据库角色的活动。
db_accessadmin	在数据库中添加或删除数据库用户。
db_datareader	查看数据库中所有用户表的全部数据。
db_datawriter	添加、更改或删除数据库中所有用户表的数据。
db_ddladmin	添加、修改或删除数据库中的对象。
db_securityadmin	管理数据库角色和成员及语句权限。
db_backupoperator	有备份数据库的权限。
db_denydatareader	拒绝选择数据库数据的权限。
db_denydatawriter	拒绝更改数据库数据的权限。
public	特殊的数据库角色,默认权限。

标准数据库角色不能删除；除 public 角色外，不能修改标准角色的对象权限。可以增加非标准角色(自定义角色)，并修改自定义角色的权限。public 角色是所有用户都必须属于的角色。

(2)角色维护

SQL Server 角色的增加、删除等维护以及将角色授予用户或从用户中收回角色， SQL Server 有以下存储过程进行角色维护

增加角色:sp_addrole [@rolename=]‘role’

删除角色:sp_addrole [@rolename=]‘role’

查看角色:sp_helprole

另外对于自定义角色和 public 角色权限的增加和减少，与给用户对象权限的授予与收回相同，使用 grant、revoke 命令，在此不在赘述。

建立好数据库的角色后，可以将用户添加到角色中，也就是授予用户该角色所拥有的权限。同理，可以将角色中的用户删除，也就是收回用户该角色所拥有的权限。

增加角色的用户:sp_addrolemember [@rolename=]’role’,[@membername=]’user’

删除角色的用户:sp_droprolemember [@rolename=]’role’,[@membername=]’user’

但是不能从 public 角色中删除用户，也不能将角色中的 dbo 用户。

6.2 数据库的完整性

6.2.1 数据库完整性概述

数据库的完整性是指数据的正确性和相容性。与数据库的平安性不同，数据库的完整性是为了防止错误数据的输入，其防范对象是不合语义的数据，而平安性防范对象是非法用户和非法操作。维护数据库的完整性是数据库管理系统的根本要求。

为了维护数据库的完整性，数据库管理系统(DBMS)必须提供一种机制来检查数据库中的数据是否满足语义约束条件。这些加在数据库数据之上的语义约束条件称为数据库的完整性约束条件。DBMS 检查数据是否满足完整性约束条件的机制称为完整性检查。

6.2.2 完整性约束条件

完整性约束条件作用对象可以是关系、元组、列。其中列约束主要是列的数据类型、取值范围、精度、是否为空等等；元组约束是元组之间列的约束关系；关系约束是指关系中元组之间以及关系和关系之间的约束。

完整性约束条件涉及的这三类对象，其状态可以是静态的，也可以是动态的。所谓静态约束是指数据库每一确定状态时的数据对象所应满足的约束条件，它是反映数据库状态合理性的约束，这是最重要的一类完整性约束。动态约束是指数据库从一种状态转变为另一种状态时新、旧值之间所应满足的约束条件，它是反映数据库状态变迁的约束。

综合上述两个方面，可以将完整性约束条件分为六类。

1. 静态列约束

静态列约束是对一个列的取值域的说明，这是最常用也最容易实现的一类完整性约束，主要有以下几个方面：

(1) 对数据类型的约束，包括数据的类型、长度、单位、精度等

例：name 类型为字符型，长度为 8。货物重量单位为公斤(kg)，类型为数值型，长度为 24 位，精度为小数点后 4 位。

(2) 对数据格式的约束

例：出生日期的格式为‘YYYY-MM-DD’。学生编号的格式共八位，前两位为入学年份，中间两位是院系编号，后面四位是顺序编号。

(3) 对取值范围或取值集合的约束

例：学生成绩的取值范围位 0～100，性别的取值集合为[男，女]。

(4) 对空值的约束

空值表示未定义或未知的值，与零值和空格不同，可以设置列不能为空值，例：学生学号不能为空值，而学生成绩可以为空值。

2. 静态元组约束

一个元组是由假设干个列值组成的，静态元组约束就是规定元组的各个列之间的约束关系。例如，定货关系中包含发货量、定货量，规定发货量不得大于定货量。

3. 静态关系约束

在一个关系的各个元组之间或者假设干关系之间常常存在各种联系或约束。常见的静态关系约束有:

(1) 实体完整性约束

(2) 参照完整性约束。

(3) 函数依赖约束。大局部函数依赖约束都在关系模式中定义。

(4) 统计约束。即字段值与关系中多个元组的统计值之间的约束关系。

其中，实体完整性约束和参照完整性约束是关系模型的两个极其重要的约束，称为关系的两个不变性。

4. 动态列约束

动态列约束是修改列定义或列值时应满足的约束条件，包括以下两方面：

(1) 修改列定义时的约束。例如，将允许空值的列改为不允许空值时，如果该列目前已存在空值，那么拒绝这种修改。

(2) 修改列值时的约束。修改列值有时需要参照其旧值，并且新旧值之间需要满足某种约束条件。例如，职工工资调整不得低于其原来工资，学生年龄只能增长等等。

5. 动态元组约束

动态元组约束是指修改元组的值时，元组中各个字段间需要满足某种约束条件。例如职工工资调整时新工资不得低于原工资+工龄*1.5，等等。

6. 动态关系约束

动态关系约束是加在关系变化前后状态上的限制条件，例如事务一致性、原子性等约束条件。

6.2.3 完整性控制

DBMS 的完整性控制机制应具有三个方面的功能：

⑴ 定义功能，提供定义完整性约束条件的机制。

⑵ 检查功能，检查用户发出的操作请求是否违背了完整性约束条件。

⑶保护功能，如果发现用户的操作请求使数据违背了完整性约束条件，那么采取一定的动作来保证数据的完整性。

完整性约束条件包括有六大类，约束条件可能非常简单，也可能极为复杂。一个完善的完整性控制机制应该允许用户定义所有这六类完整性约束条件。

检查是否违背完整性约束的时机通常是在一条语句执行完后立即检查，我们称这类约束为立即执行约束〔Immediate Constraints〕。有时完整性检查需要延迟到整个事务执行结束后再进行，检查正确方可提交，我们称这类约束为延迟执行约束〔DeferredConstraints〕。如银行数据库中“借贷总金额平衡”就是延时执行约束。从帐号 A 转帐到帐号 B 是一个事务，从帐号 A 转出后帐不平衡，必须等到转入帐号 B 后才能平衡，这时才能进行完整性检查。如果违背了完整性约束条件，系统将拒绝操作，对于延迟执行约束，系统将拒绝整个事务，把数据库恢复到事务前的状态。

关系系统中，最重要的完整性约束是实体完整性和参照完整性，其他完整性约束条件那么可以归入用户定义的完整性。

实体完整性：根本关系的所有主属性都不能取空值，以便唯一地标识实体。

参照完整性：假设属性或属性组 F 是根本关系 R 的外码，它与根本关系 S 的主码 K 相对应，那么对于 R 中的每个元组在 F 上的值必须为空值或等于 S 中某个元组的主码值。其中 S 和 R 可以是同一关系。参照完整性定义了外码和主码之间的引用规那么。

用户定义完整性：除了实体完整性和参照完整性外，针对某一具体的关系数据库，如果需要一些特殊的约束条件，用户可以自行定义其约束条件。

目前 DBMS 系统中，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。对于违反实体完整性和用户定义完整性的操作，一般拒绝执行，而对于违反参照完整性的操作，不是简单的拒绝，而是根据语义执行一些附加操作，以保证数据库的正确性。

下面详细讨论实现参照完整性要考虑的几个问题。

1. 外码能否为空值问题

例如：学生-班级关系中，学生关系 student 和班级关系 class,其中 class 关系的主码为班级号 cno,student 关系的主码为学生号 sno,外码为班级号 cno,称 class 为被参照关系,student 为参照关系。student 关系中某一元组的 cno 列值假设为空值，表示这个学生还没有分配到任何班级。这个和应用环境的语意是相符的，因此 student 的 cno 列可以取空值。

在客户-订单关系中，包含客户关系 client 和订单关系 order，client 关系为被参照关系，其主码为客户编号 cno。order 为参照关系，主码为订单编号 ono，外码为订单客户编号 cno。假设 order 中元组的 cno 为空值的话，那么说明存在没有客户的订单。这与实际的应用环境是不相符合的，因此 order 的 cno 列不能取空值。

因此在实现参照完整性时，系统除了提供定义外码的机制外，还应提供定义外码是否允许空值的机制。

2. 在被参照关系中删除元组的问题

当删除被参照关系的某个元组时，而参照关系存在假设干元组，且其外码值与被参照关系中删除元组的主码值相同。如：员工和部门关系，部门 dept 关系中，部门编号 dno 是主码；员工 employ 关系中，员工编号 eno 是主码，员工所在部门 dno 是外码，对应 dept 中的主键。删除 dept 部门编号 dno=9999 的部门，而 employ 中存在 dno=9999 的 8 名员工。这时可有三种不同的删除策略：

(1)级联删除(CASCADES)

将参照关系外码值与被参照关系中要删除元组主码值相同的元组一起删除。如上例中，删除关系 dept 中 dno=9999 的部门，同时删除 employ 中 8 名 dno=9999 的员工。

(2)受限删除〔RESTRICTED〕

仅当参照关系中没有任何元组的外码值与被参照关系中要删除元组的主码值相同时，系统才执行删

除操作，否那么拒绝此删除操作。

如上例中，删除关系 dept 中 dno=9999 的部门时，检查 employ 中是否有 dno=9999 的员工，如果有那么不能删除，只有先删除 employ 中 dno=9999 的 8 名员工，然后才能删除关系 dept 中 dno=9999 的部门。

(3)置空值删除〔NULLIFIES〕

删除被参照关系的元组，并将参照关系中相应元组的外码值置空值。如上例中，删除关系 dept 中 dno=9999 的部门时，检查 employ 中是否有 dno=9999 的员工，如果有将 employ 中 dno=9999 员工的 dno 设置为 NULL。

3. 在参照关系中插入元组时的问题

当参照关系插入某个元组，而被参照关系不存在相应的元组。如向关系 employ 中插入部门编号 dno=9999,员工编号 eno=1968 的元组(1968，9999)，而关系 dept 中没有 dno=9999 的部门。这时可有以下两种插入策略：

(1)受限插入

仅当被参照关系中存在相应的元组，其主码值与参照关系插入元组的外码值相同时，系统才允许插入，否那么拒绝插入。在上例中，系统拒绝插入 employ 元组(1968，9999)，因为被参照关系 dept 中没有 dno=9999 的元组。

(2)递归插入

首先向被参照关系中插入相应的元组，其主码值等于参照关系插入元组的外码值，然后向参照关系插入元组。在上例中，系统先在 dept 中插入 dno=9999 的元组，然后在关系 employ 中插入元组(1968，9999)。

4. 修改关系中主码的问题

(1)不允许修改主码

在有些 RDBMS 中，不允许修改关系主码。如上例中不能修改 dept 关系中的部门编号 dno。如果要修改，只能先删除，然后再增加。

(2)允许修改主码

在有些 RDBMS 中，允许修改关系主码，但必须保证主码的唯一性和非空，否那么拒绝修改。当修改的关系是被参照关系时，还必须检查参照关系。如上例中修改 dept 中的 dno=9999 为 dno=8888,检查 employ 关系中是否有 dno=9999 的元组，如果存在，那么与删除策略相同。当修改的关系是参照关系时，要检查被参照关系。如学生-选课关系中，选课关系 elective 中的学生号 sno 和课程号 cno 联合为主码，sno 也是外码，对应 student 中的学生编号 sno。如果选课 elective 关系中的(90211111,20,90)改为〔90212222，20，90〕，检查 student 关系中是否存在 sno=90212222 的主码值，否那么与插入策略相同。

从上面的讨论我们看到 DBMS 在实现参照完整性时，除了要提供定义主码、外码的机制外，还需要提供不同的策略供用户选择。选择哪种策略，都要根据应用环境的要求确定。

6.2.4 SQL Server 的完整性

SQL Server 提供了比拟完善的完整性约束机制，不仅有实体完整性和参照完整性，还提供了多种自定义完整性的方法。

1. SQL Server 的实体完整性

实体完整性约束就是定义主键，并设置主键不为空(NOT NULL)。

定义主键可以使用 CREATE TABLE 语句，在建立表时定义；如果创立表时没有设置主键，可以使用 ALTER TABLE 语句增加主键,在增加主键时,如果原有数据中设置主键的列不符合主键约束条件(NOT NULL 和唯一性)，拒绝执行，要先对数据进行处理。创立表时定义主键例如：

```
CREATE TABLE student(
    sno         CHAR(8)         NOT NULL,
    sname       VARCHAR2(10),
    sex         CHAR(2),
    birthday    DATE,
    CONSTRAINT  sno_pk  PRIMARY KEY (sno));
```

首先定义 sno 不为空 NOT NULL，使用关键字 PRIMARY KEY 定义 sno 为主键，其约束名为 sno_pk, SQL Server 根据主键自动建立索引，索引名为 sno_pk。当主键由一个字段组成时，可以直接在字段后面定义主键，称为列约束，如：

```
sno  CHAR(8) NOT NULL  PRIMARY KEY,
```

使用 ALTER TABLE 增加主键定义例如：在创立课程关系中 course 中已经定义了课程编号 cno CHAR(4) NOT NULL 和课程名称 cname VARCHAR(20),没有定义主码，下面操作增加主键，约束名为 cno_pk:

```
ALTER TABLE course ADD CONSTRAINT cno_pk PRIMARY KEY(cno);
```

2. SQL Server 的参照完整性

参照完整性就是定义好被参照关系及其主码后，在参照关系中定义外键。定义语法：

```
CONSTRAINT constraint_name FOREIN KEY (column [,...])
    REFERENCES ref_table(ref_column [,...])
    [ ON DELETE {CASCADE | NO ACTION }]
    [ ON UPDATE {CASCADE | NO ACTION }]
```

其中:

constraint_name:限制名。

FOREIN KEY(column [,...]):外键,如果是单个字段，可作为列约束，省略限制名。

REFERENCES ref_table(ref_column [,...]):被参照关系及字段。

ON DELETE {CASCADE | NO ACTION }:定义删除行为，CASCADE 为级联删除，NO ACTION 为不允许删除。

ON UPDATE {CASCADE | NO ACTION }:定义更新行为，CASCADE 为级联更新，NO ACTION 为不允许更新。

例如，以 student、course 为参照关系，建立 elective 关系，同时定义 elective 关系的外码(sno，cno)，分别对应 student 中的主码 sno 和 course 中的主码 cno。使用 CREATE TABLE 语句，在表建立时定义外键，如果表已经创立了，可以使用 ALTER TABLE 语句增加或修改。创立表时定义外键例如：

```
CREATE TABLE sourse(
    sno   CHAR(8)   NOT NULL,
    cno   CHAR(4)   NOT NULL,
    grade NUMBER(6) ,
    CONSTRAINT sno_cno_pk  PRIMARY KEY(sno,cno),
    CONSTRAINT sno_fk  FOREIN KEY (sno)
       REFERENCES student(sno)  ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT cno_fk  FOREIN KEY (cno)
       REFERENCES course(cno) ON DELETE NO ACTION ON UPDATE NO ACTION);
```

首先定义了 sourse 表的实体完整性，sno 和 cno 联合作为主键。然后定义了 sno 为外码，约束名为 sno_pk，参照 student 表中的 sno 属性,删除策略和更新策略是 CASCADE 级联删除。定义了 cno 为外码，约束名为 cno_pk，参照 course 表中的 cno 属性，删除策略和更新策略是 NO ACTION 禁止删除和更新。

3. SQL Server 的用户定义完整性

(1)NOT NULL 约束

NOT NULL 约束应用在单一的数据列上，保护该列必须要有数据值。缺省状况下，SQL Server 允许任何列都可以有 NULL 值。主键必须有 NOT NULL 约束。设置 NOT NULL 约束可以使用 CREATE TABLE 语句，在表建立时一起设置，如：

```
CREATE TABLE student(
    sno         CHAR(8)       NOT NULL,
    sname       VARCHAR(20));
```

如果创立表时没有 NOTNULL 约束，可以使用 ALTERTABLE 语句修改。在修改时，如果原有数据中有 NULL 值，将拒绝执行，要先对数据进行处理。如增加 student 表中 name 列的 NOT NULL 约束。

```
ALTER TABLE student MODIFY (name VARCHAR2(10) NOT NULL);
```

(2)CHECK 约束

CHECK 约束设置一个特殊的布尔条件，只有使布尔条件为 TRUE 的数据才接受。CHECK 约束用于增强表中数据的简单商业规那么。用户使用 CHECK 约束保证数据规那么的一致性。如果用户的商业规那么需要复杂的数据检查，那么可以使用触发器(TRIGGER)。CHECK 约束不保护 LOB 类型的数据列。单一数据列可以有多个 CHECK 约束保护，一个 CHECK 约束可以保护多个数据列。当 CHECK 约束保护多个数据列时，必须使用表约束语法。可用 CREATE TABLE 语句在定义表时设置 CHECK 约束，如：

```
CREATE TABLE student(
    sno         CHAR(8)    NOT NULL,
    sex         CHAR(2),
    age         int,
    CONSTRAINT age_ck CHECK(age>20));
```

如果 CHECK 只对一列进行约束，可以作为列约束直接写在列后面：

```
    age int CHECK(age>20),
```

ALTER TABLE 语句可以增加或修改 CHECK 约束。如在 student 表中增加性别 sex 约束：

```
ALTER TABLE student ADD CONSTRAINT sex_ck CHECK(sex in ('男','女'));
```

(3) UNIQUE 约束

唯一性 UNIQUE 约束使数据列中任何两行的数据都不相同或为 NULL。唯一性约束与主码不同的是，唯一性约束可以为 NULL(是指没有 NOT NULL 约束的情况下)，一个表可以有多个唯一性约束，而主码只能有一个。可以使用 CREATE TABLE 语句，在创立表时设置 UNIQUE 约束。如将学生表中身份证 sid 设置为 UNIQUE 约束：

```
CREATE TABLE student(
    sno         CHAR(8)    NOT NULL,
    sid         CHAR(20),
    cname       VARCHAR(20),
    CONSTRAINT sid_unique UNIQUE(sid));
```

UNIQUE 由单列组成，可以作为列约束直接写在列的后面：

```
    sid         CHAR(20) UNIQUE,
```

ALTER TABLE 语句可以增加或修改 UNIQUE 约束。在修改时，如果原有数据中不符合唯一性，那么拒绝执行，要先对数据进行处理。如在课程表 course 中增加课程名 cname 的 UNIQUE 约束：

```
ALTER TABLE course ADD CONSTRAINT cname_unique UNIQUE(cname);
```

约束名为 cname_unique。由于定义 UNIQUE 唯一性约束后，自动建立索引，所以一般指定约束名。