数据库系统原理与设计

（第3版）

现代学习理念的四大支柱是：

- 学会认知
- 学会做事
- 学会合作
- 学会生存

——《学会学习》一书的作者：方州

数据库系统原理与设计

（第 3 版）

第9章 数据库安全性与完整性

第9章　数据库安全性与完整性

- 数据库的安全性：保护数据库以防止不合法使用所造成的数据泄密、更改或破坏。
- 数据库的完整性：防止数据库中存在不符合语义的数据，其防范对象是不合语义的、不正确的数据。
- 主要教学目标如下：
  - 要求熟练掌握DBMS安全性保护的基本原理与方法，并能熟练运用SQL中的GRANT和REVOKE语句进行授权；
  - 要求熟练掌握DBMS完整性约束的保证措施，并能熟练运用SQL中的DDL语句定义完整性约束条件。
  - 熟练掌握如何使用触发器实现复杂的安全性保护和完整性约束。

目　录

9.1　数据库安全性

- 安全性问题不是数据库系统所独有的，所有计算机系统都有这个问题。
- 数据库系统中大量数据集中存放，且为许多最终用户直接共享，安全性问题更为突出。

- **9.1.1 数据库安全概述**
- **9.1.2 SQL Server安全机制**

9.1.1 数据库安全概述

- 数据库安全保护目标是确保只有授权用户才能访问数据库，未被授权的人员则无法接近数据。
- 安全措施是指计算机系统中用户直接或通过应用程序访问数据库所要经过的安全认证过程。
- 数据库安全认证过程如图9-1所示

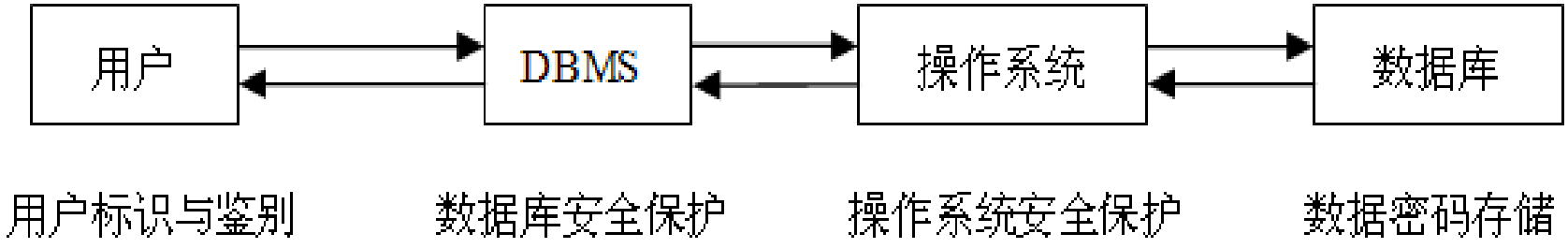


图 9-1　数据库安全认证过程

9.1.1 数据库安全概述

■ 用户标识与鉴别(identification & authentication)

- 当用户访问数据库时，要先将其用户名(user name)与密码(password)提交给数据库管理系统进行认证；
- 只有在确定其身份合法后，才能进入数据库进行数据存取操作。

■ 数据库安全保护

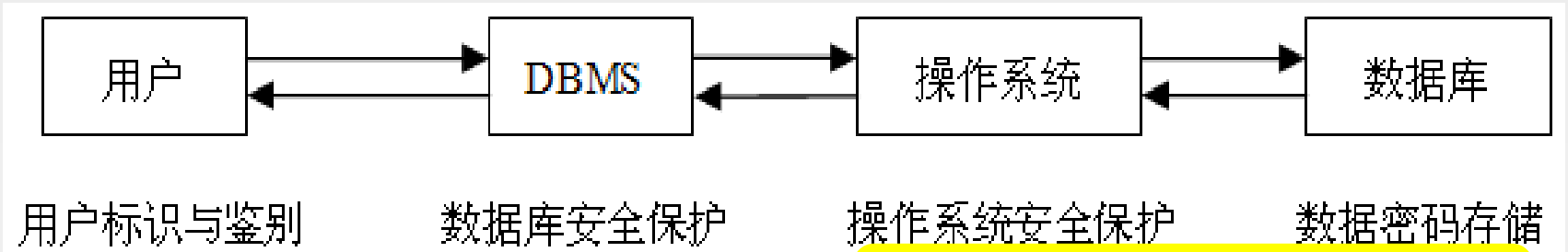


图 9-1　数据库

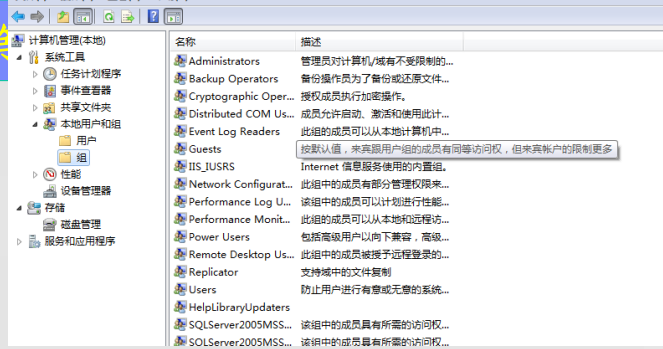
三个方面展开说明

9.1.1 数据库安全概述

- **存取控制**：决定用户对数据库中的哪些对象进行操作，进行何种操作。存取控制机制主要包括两部分：
  - **定义用户权限**及将用户权限登记到数据字典中；
  - **合法权限检查**：当用户发出操作请求后，DBMS查找数据字典并根据安全规则进行合法权限检查，若操作请求超出了定义的权限，系统将拒绝执行此操作。
- **视图**：通过为不同的用户定义不同的视图，达到限制用户访问范围的目的。
  - 视图机制能隐藏用户无权存取的数据，从而自动地对数据库提供一定程度的安全保护；
  - 视图的主要功能在于提供数据库的**逻辑独立性**，其安全性保护不太精细，往往不能达到应用系统的要求；
  - 在实际应用中，通常将视图与存取控制机制结合起来使用，如先通过视图屏蔽一部分保密数据，然后进一步定义存取权限。

9.1.1 数据库安全概述

- **审计**：是一种监视措施，用于跟踪并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**审计日志**(audit log)中；
  - **审计日志**的内容一般包括：
    - ✓ 操作类型(如修改、查询、删除)；
    - ✓ 操作终端标识与操作者标识；
    - ✓ 操作日期和时间；
    - ✓ 操作所涉及到的相关数据(如基本表、视图、记录、属性)；
    - ✓ 数据库的前映像(即修改前的值)和后映像(即修改后的值)。
  - 利用这些信息，可找出非法存取数据库的人、时间和内容等；
  - 数据库管理系统往往将**审计**作为可选特征，允许操作者打开或关闭审计功能。



9.1.1 数据库安全概述

- **操作系统安全保护**
  - 通过操作系统提供的安全措施来保证数据库的安全性
- **数据密码存储**
  - **访问控制和存取控制**可将用户的**应用系统访问范围**最小化和**数据对象操作权限**最低化，但对一些敏感数据进行“**加密存储**”也是系统提供的安全策略；

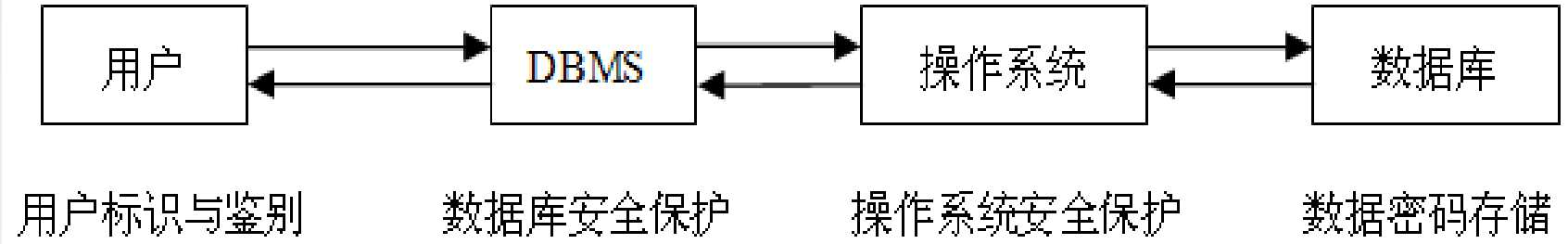


图 9-1 数据库安全认证过程

然后数据以密文的方式存储和传输。

9.1.2　SQL Server安全机制

- SQL支持受控的存取保护：
  - 在自主存取控制中，用户对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有不同的权限；
  - 用户可将其拥有的存取权限转授给其他用户。
- 自主存取控制通过SQL的GRANT和REVOKE语句实现。
- 用户权限：是指用户可以在哪些数据对象上进行哪些类型的操作。它由两个要素组成：数据对象和操作类型。
  - 定义存取权限称为授权(authorization)；
  - **授权粒度**可以精细到字段级，也可以粗到关系级；
  - 授权粒度越细，授权子系统就越灵活，但是系统的开销也会相应地增大。

9.1.2　SQL Server安全机制

- 使用GRANT和REVOKE语句，向用户授予或收回对数据的操作权限。对数据库模式的授权则由DBA在创建用户时实现。
- SQL Server安全管理机制是架构在认证和权限两大机制下：
  - 认证是指用户必须要有一个登录账号和密码来登录SQL Server，只有登录后才有访问使用SQL Server的入门资格，也只能处理SQL Server特定的管理工作；
  - 而数据库内的所有对象的访问权限必须通过权限设置来决定登录者是否拥有某一对象的访问权限。
  - 在数据库内可创建多个用户，然后针对具体对象将对象的创建、读取、修改、插入、删除等权限授予特定的数据库用户。
- 利用角色，SQL Server管理者可以将某类用户设置为某一角色，这样只对角色进行权限设置便可以实现对该类所有用户权限的设置，大大减少了管理员的工作量。

9.1.2　SQL Server安全机制

■ 登录账户管理

● 创建登录账号：

[EXECUTE] sp_addlogin [*@loginame*=] '*login*'
[, [*@passwd*=] '*password*'] [, [*@defdb*=] '*database*']

- *@loginame='login'*：*login*指定登录名称。
- *@passwd='password'*：*password*指定登录密码，若不指定则默认为NULL。
- *@defdb='database'*：*database*指定登录后用户访问的数据库，若不指定则默认为master数据库。

执行系统存储过程sp_addlogin，除了指定登陆名称之外，其余选项均为可选项。执行sp_addlogin时，必须具有相应的权限。只有syssdmin和securityasmin固定服务器角色的成员才能执行该命令。

9.1.2　SQL Server安全机制

■ 登录账户管理

● 创建登录账号：

[例9.1] 创建登录账号为login1、密码为p666666的登录账号；
创建登录账号为login2、密码为p888888的登录账号。

```
sp_addlogin  'login1', 'p666666'
sp_addlogin  'login2', 'p888888'
```

[例9.2] 创建登录账号login3，密码为p123456，默认的数据库为ScoreDB。

```
sp_addlogin  login3, 'p123456', 'ScoreDB'
```

9.1.2　SQL Server安全机制

■ 登录账户管理

● 修改登录账号属性：

sp_password [[@old=] 'old_password',]
[@new=] 'new_password' [, [@loginame=] 'login']

[例9.3] 将login3的密码修改为p654321。

sp_password 'p123456', 'p654321', 'login3'

9.1.2　SQL Server安全机制

■ 登录账户管理

- **修改登录账号属性：**

  sp_password [[*@old*=] '*old_password*',]
  [*@new*=] '*new_password*' [, [*@loginame*=] '*login*']

- **修改默认的数据库：**

  sp_defaultdb [*@loginame*=] '*login*', [*@defdb*=] '*database*'

[例9.4] 将login3访问的数据库修改为BookDB。

```
sp_defaultdb 'login3', 'BookDB'
```

9.1.2　SQL Server安全机制

■ 登录账户管理

- **修改登录账号属性：**

  sp_password [[@old=] 'old_password',]
  [@new=] 'new_password' [, [@loginame=] 'login']

- **修改默认的数据库：**

  sp_defaultdb [@loginame=] 'login', [@defdb=] 'database'

- **删除登录账号：**

  sp_droplogin [@loginame=] 'login'

[例9.5] 删除登录账号login3。

```
sp_droplogin 'login3'
```

9.1.2　SQL Server安全机制

■ 用户管理

● 添加用户：

sp_adduser [*@loginame*=] '*login*'
[, [*@name_in_db*=] '*user*']

[例9.6] 将登录账号login1添加到当前数据库OrderDB中，且用户名为u1。

```
sp_adduser  login1, u1
```

[例9.7] 将登录账号login2添加到当前数据库OrderDB中，且用户名为u2。

```
sp_adduser  login2, u2
```

9.1.2　SQL Server安全机制

■ 用户管理

- **添加用户：**

  sp_adduser [@*loginame*=] '*login*'
  [, [@*name_in_db*=] '*user*']

- **删除用户：**

  sp_dropuser [@*name_in_db*=] '*user*'

[例9.8] 从当前数据库中删除用户**u1**。

```
sp_dropuser  u1
```

9.1.2　SQL Server安全机制

■ 权限的授予与收回

- GRANT和REVOKE有两种权限：命令权限和目标权限。
- 命令权限的授予与收回
  - 主要指DDL操作权限，语法分别为：

    GRANT {all | <*command_list*>} TO {public | <*username_list*>}

    REVOKE {all | <*command_list*>} FROM {public | <*username_list*>}
  - <*command_list*>可以是create database、create default、create function、create procedure、create rule、create table、create view、create index、backup database和backup log等；
  - 一次可授多种权限，授多种权限时，权限之间用逗号分隔；
  - 如果具有创建对象的create权限，则自动具有其创建对象的修改alter权限和删除drop权限；

9.1.2　SQL Server安全机制

- **all**：表示上述所有权限；
- **public**：表示所有的用户；
- ***<username_list>***：指定的用户名列表。如果将某组权限同时授予多个用户，则用户名之间用逗号分隔。

[例9.9] 将创建基本表和视图的权限授予用户u1和u2：

```
GRANT create table, create view TO u1, u2
```

[例9.10] 从用户u2收回创建视图的权限：

```
REVOKE create view FROM u2
```

9.1.2　SQL Server安全机制

- 目标权限的授予与收回
  - 主要指DML操作权限，语法分别为：

    GRANT {all | <*command_list*>} ON <*objectName*> [(<*columnName_list*>)]
    TO {public | <*username_list*>} [WITH GRANT OPTION]

    REVOKE {all | <*command_list*>} ON <*objectName*> [(<*columnName_list*>)]
    FROM {public | <*username_list*>} [CASCADE | RESTRICT]
  - <*command_list*>可以是update、select、insert、delete、excute和all
    - ✓ excute针对存储过程授予执行权限；
    - ✓ update、select、insert、delete针对基本表和视图授权；
    - ✓ all表示所有的权限。
  - 对象的创建者自动拥有该对象的插入、删除、修改和查询操作权限；存储过程的创建者自动拥有所创建过程的执行权限；

9.1.2　SQL Server安全机制

- **CASCADE**：级联收回；
- **RESTRICT**：缺省值，若转赋了权限，则不能收回；
- **WITH GRANT OPTION**：允许将指定对象上的目标权限授予其它安全用户。

■ 不允许循环授权，即不允许将得到的权限授予其祖先，如下图所示：

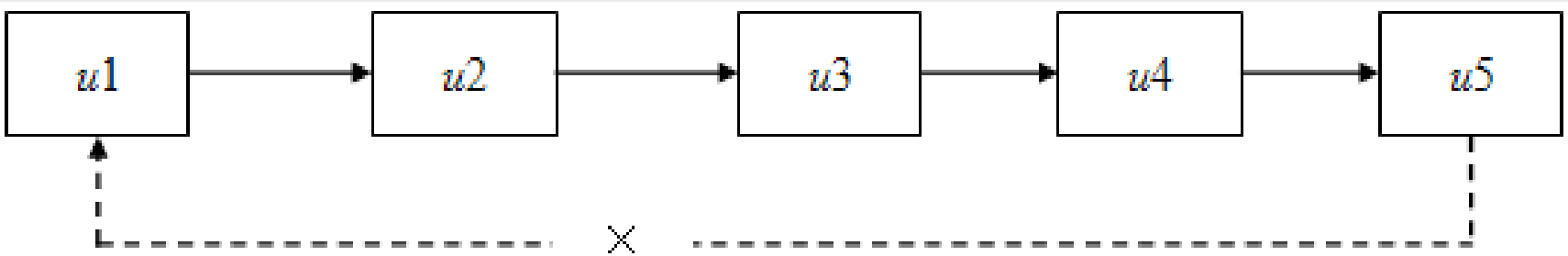


图 9-4　不允许循环授权

9.1.2　SQL Server安全机制

[例9.11] 将存储过程proSearchBySno的执行权限授予用户u1、u2和u3：

```
GRANT excute ON proSearchBySno TO u1, u2, u3
```

[例9.12] 将对班级表Class的查询、插入权限授予用户u1，且用户u1可以转授其所获得的权限给其它用户：

```
GRANT select, insert ON Class TO u1 WITH GRANT OPTION
```

[例9.13] 将对学生表Student的性别、出生日期的查询和修改权限授予用户u3、u4和u5，且不可以转授权限：

```
GRANT select, update ON Student(sex, birthday) TO u3, u4, u5
```

- 如果是对列授予权限，命令项可以包括select或update或两者组合；
- 若使用了select *，则必须对表的所有列赋予select权限。

9.1.2　SQL Server安全机制

[例9.14] 将基本表Score的若干权限分别授予用户u1、u2、u3、u4、u5和u6。

- 将表Score的所有权限授予用户u1，且可以转授权限

```
GRANT all ON Score TO u1 WITH GRANT OPTION
```

- 用户u1将表Score的所有权限授予用户u2，且可以转授权限

```
GRANT all ON Score TO u2 WITH GRANT OPTION
```

- 用户u2将表Score的查询和插入权限授予用户u5，且不可转授权限

```
GRANT select, insert ON Score TO u5
```

- 用户u2将表Score的所有权限授予用户u4，且可以转授权限

```
GRANT all ON Score TO u4 WITH GRANT OPTION
```

- 用户u4将表Score的查询和删除权限授予用户u6，且可以转授权限

```
GRANT select, delete ON Score TO u6 WITH GRANT OPTION
```

9.1.2　SQL Server安全机制

■ 通过上述的授权操作，用户u1、u2、u3、u4、u5和u6分别得到的权限如下图所示：

授权用户	被授权用户	数据库对象	允许的操作	能否转授权限
DBA	u1, u2, u3	过程 proSearchBySno	执行权限	不能
DBA	u3, u4, u5	基本表 Student	select, update	不能
DBA	u1	基本表 Class	select, insert	能
DBA	u1	基本表 Score	所有权限	能
u1	u2	基本表 Score	所有权限	能
u2	u5	基本表 Score	select, insert	不能
u2	u4	基本表 Score	所有权限	能
u4	u6	基本表 Score	select, delete	能

图 9-5　用户权限定义

9.1.2　SQL Server安全机制

■ 数据库角色

- 数据库角色是指被命名的一组与数据库操作相关的权限；
- 角色是权限的集合，可以为一组具有相同权限的用户创建一个角色；
- 角色简化了授权操作。

9.1.2　SQL Server安全机制

- 角色的创建、删除、授权、转授和收回语句的语法如下：
  - 角色的创建，使用系统存储过程sp_addrole创建角色：

```
sp_addrole  [@rolename=] 'role'
```

[例9.17] 建立角色r1和r2。

```
sp_addrole  'r1'
sp_addrole  'r2'
```

  - 只有固定服务器角色sysadmin、db_securityadmin及db_owner的成员才能执行该系统存储过程。

9.1.2　SQL Server安全机制

- 角色的创建、删除、授权、转授和收回语句的语法如下：
  - 角色的创建，使用系统存储过程sp_addrole创建角色：

    sp_addrole [@*rolename*=] '*role*'
  - 删除数据库角色

    sp_droprole [@*rolename*=] '*role*'

[例9.18] 删除数据库角色r2。

```
sp_droprole 'r2'
```

9.1.2　SQL Server安全机制

- 角色的创建、删除、授权、转授和收回语句的语法如下：
  - 角色的创建，使用系统存储过程sp_addrole创建角色：

    sp_addrole [@*rolename*=] '*role*'
  - 删除数据库角色

    sp_droprole [@*rolename*=] '*role*'
  - 增加数据库角色成员

    sp_addrolemember [@*rolename*=] '*role*',
    [@*membername*=] '*security_account*'

[例9.19] 将用户u2添加到数据库角色r1中。

```
sp_addrolemember 'r1', 'u2'
```

  - 只有固定服务器角色sysadmin及db_owner的成员才能执行该系统存储过程。

9.1.2　SQL Server安全机制

- 角色的创建、删除、授权、转授和收回语句的语法如下：
  - 角色的创建，使用系统存储过程sp_addrole创建角色：

    sp_addrole [@rolename=] 'role'
  - 删除数据库角色

    sp_droprole [@rolename=] 'role'
  - 增加数据库角色成员

    sp_addrolemember [@rolename=] 'role',
    [@membername=] 'security_account'
  - 删除数据库角色成员

    sp_droprolemember [@rolename=] 'role',
    [@membername=] 'security_account'

[例9.20] 在数据库角色r1中删除用户u2。

```
sp_droprolemember 'r1', 'u2'
```

➢ 只有固定服务器角色sysadmin及db_owner的成员才能执行该系统存储过程

目　录

9.2　数据库完整性

- 为维护数据库的完整性，数据库管理系统提供如下功能：
  - **完整性约束条件定义机制**
    - 完整性约束条件也称为完整性规则，是数据库中的数据必须满足的语义约束条件；
    - 由SQL的DDL实现，作为模式的一部分存入数据库中。
  - **完整性检查方法**
    - 检查数据是否满足已定义的完整性约束条件称为完整性检查；
    - 一般在insert、delete、update执行后开始检查，或事务提交时进行检查。
  - **违约处理措施**
    - 若发现用户操作违背了完整性约束条件，应采取一定的措施，如拒绝操作等。
- 商用DBMS都支持完整性控制。定义数据库模式时,都可以很明确地对完整性约束加以说明。

9.2　数据库完整性

- 9.2.1 数据库完整性概述
- 9.2.2 SQL Server完整性
- 9.2.3 使用规则和触发器实现完整性

9.2.1 数据库完整性概述

- 用户可以为完整性约束命名，命名格式如下：

```
[ CONSTRAINT <constraintName> ]
        PRIMARY KEY (<constraintExpr>)
[ CONSTRAINT <constraintName> ]
        FOREIGN KEY (<constraintExpr>)
                REFERENCE <refTable>(<constraintExpr>)
[ CONSTRAINT <constraintName> ]
        CHECK (<constraintExpr>)
```

- 用户命名有两点好处：
  - 一是便于理解约束的含义；
  - 二是修改约束方便，不必查询数据字典。
- 使用ALTER TABLE语句修改基本表中的完整性约束。
  - 要修改约束，必须先删除约束，然后加入新的约束。

9.2.2 SQL Server完整性：实体完整性

- 实体完整性要求基本表的主码值唯一且不允许为空值。
- 在SQL中：
  - 实体完整性定义使用CREATE TABLE语句中的PRIMARY KEY短语实现；
  - 或使用ALTER TABLE语句中的ADD PRIMARY KEY短语实现；
  - 有关CREATE TABLE、ALTER TABLE语句的语法详见第7章7.1.2节（P216～219）。
  - 对单属性构成的主码可定义为列约束, 也可定义为元组约束;
  - 对多个属性构成的主码，只能定义为元组约束(也有的书将其划归关系约束，即表约束)。

9.2.2　SQL Server完整性：实体完整性

■ 实体完整性定义

[例9.23] 在班级表Class中将classNo定义为主码。

```
CREATE  TABLE  Class  (
    classNo      char(6)                    NOT  NULL,   --班级号，列约束
    className    varchar(30)  UNIQUE        NOT  NULL,   --班级名，列约束
    institute    varchar(30)                NOT  NULL,   --所属学院，列约束
    grade        smallint   DEFAULT  0      NOT  NULL,   --年级，列约束
    classNum     tinyint                    NULL,        --班级人数，列约束
    CONSTRAINT  ClassPK  PRIMARY  KEY (classNo)          -- 元组约束
)
```

- 本例定义classNo为主码，使用CONSTRAINT短语为该约束命名为ClassPK；
- 该主码定义为元组约束(也有将其划归关系约束的)。

9.2.2　SQL Server完整性：参照完整性

- 参照完整性为若干个基本表中的相应元组建立联系。
- 在SQL中：
  - 参照完整性定义使用CREATE TABLE语句中的FOREIGN KEY和REFERENCES短语来实现；
  - 或通过使用ALTER TABLE语句中的ADD FOREIGN KEY短语来实现；
  - FOREIGN KEY指出定义哪些列为外码；
  - REFERENCES短语指明这些外码参照哪些关系；
  - 给出FOREIGN KEY定义的关系称为参照关系；
  - 由REFERENCES指明的基本表称为被参照关系。

9.2.2　SQL Server完整性：自定义完整性

- 用户自定义完整性就是定义某一具体应用中数据必须满足的语义要求，由RDBMS提供，而不必由应用程序承担。
- 用户自定义完整性包括属性上的约束和元组上的约束两种。
- 属性上的约束——列约束
  - 属性上的约束包括：数据类型、列值非空、列值唯一、设置默认值、满足CHECK (<*predicate*>)定义等；
  - 属性上的约束：当向基本表中插入或修改属性值时，系统检查是否满足约束条件，若不满足，则拒绝相应的操作。

9.2.2 SQL Server完整性：自定义完整性

[例9.27] 创建学生表Stud，属性及要求为：学号studNo为5位字符, 且第1位为字母D, M或U, 其他4位为数字, 主码, 不允许为空值；姓名studName为12位字符, 不允许为空值, 且值必须唯一；性别sex为2位字符, 允许为空值, 但值只能取'男' /'女' ；年龄age为整型, 允许为空值, 缺省值为16, 取值范围(0, 60)；民族nation为变长20位字符, 允许为空值, 缺省值为'汉族' 。

```
CREATE TABLE Stud (
  --学号, 列约束: 不允许为空值; 第1位为字母D, M或U, 其他4位为数字, 约束名为sNoCK
  studNo      char(5)                          NOT NULL
     CONSTRAINT sNoCK CHECK (studNo LIKE '[D,M,U][0-9][0-9][0-9][0-9]'),
  --姓名, 列约束: 不允许为空值; 取值必须唯一
  studName  char(12)  UNIQUE       NOT NULL,
  --性别, 列约束: 允许为空值, 仅取男或女两个值
  sex           char(2)                          NULL  CHECK ( sex IN ( '男', '女') ),
  --年龄, 列约束: 允许为空值, 默认值为16; 取值范围(0, 60), 约束名为ageCK
  age           tinyint   DEFAULT 16    NULL
     CONSTRAINT ageCK CHECK ( age > 0 AND age < 60 ),
  nation varchar(20) DEFAULT '汉族' NULL,  --民族, 允许空值, 默认值为汉族
  CONSTRAINT StudPK PRIMARY KEY (studNo)   -- 定义主码，元组约束
)
```

9.2.2 SQL Server完整性：修改完整性约束

■ 删除约束：

ALTER TABLE <*tableName*>

DROP CONSTRAINT <*constraintName*>

■ 添加约束：

ALTER TABLE <*tableName*>

ADD CONSTRAINT <*constraintName*>

< CHECK | UNIQUE | PRIMARY KEY | FOREIGN KEY >

(<*constraintExpr*>)

■ 其中

- <*tableName*>为欲修改约束所在的基本表的名称；
- <*constraintName*>为欲修改的约束的名称。

9.2.2 SQL Server完整性：修改完整性约束

[例9.30]在例9.27的基础上，修改基本表Stud中的约束条件，要求学号改为在15001～25999之间，年龄由(0, 60)之间修改为[15, 50]之间。

- 首先，删除已经存在的约束：

```
ALTER TABLE Stud
    DROP CONSTRAINT sNoCK
ALTER TABLE Stud
    DROP CONSTRAINT ageCK
```

- 然后，添加修改后的约束：

```
ALTER TABLE Stud
    ADD CONSTRAINT sNoCK
        CHECK ( studNo BETWEEN '15001' AND '25999' )
ALTER TABLE Stud
    ADD CONSTRAINT ageCK
        CHECK ( age >= 15 AND age <= 50 )
```

- 对于复杂的完整性约束要求，必须通过触发器来实现。

9.3　数据库应用与安全设计

- 9.3.1　数据库安全性控制
- 9.3.2　数据库完整性控制
- 9.3.3　存储过程设计

9.3.1 数据库安全性控制

- 《网上书店系统》的权限分析
  - 游客：可以浏览图书信息。
  - 会员：可浏览图书信息、在线订书、修改自己的个人信息和订单，但不能修改图书信息及查看其它会员的订单。
  - 职员：可增加新书、修改图书信息，查看和修改所有订单；可查看除会员登录密码外的所有信息，但不能修改会员信息。
  - 系统管理员即DBA，负责所有数据的全局访问并保证数据库系统的正常运行。
- 《网上书店系统》的角色创建
- 《网上书店系统》的角色授权
- 《网上书店系统》的视图定义
- 《网上书店系统》的Web安全考虑