第5章　数据库保护

在1.1.2节已经讲到，数据库系统中的数据是由DBMS统一管理和控制的，为了适应数据共享的环境，DBMS必须提供数据的安全性、完整性、并发控制和数据库恢复等数据保护能力，以保证数据库中数据的安全可靠和正确有效。

5.1　安　全　性

数据库的安全性是指保护数据库，防止因用户非法使用数据库造成数据泄露、更改或破坏。

数据库的一大特点是数据可以共享，但数据共享必然带来数据库的安全性问题。数据库中放置了组织、企业、个人的大量数据，其中许多数据可能是非常关键、机密或者涉及个人隐私，例如，军事秘密、国家机密、新产品实验数据、市场需求分析、市场营销策略、销售计划、客户档案、医疗档案、银行储蓄数据等，数据拥有者往往只允许一部分人访问这些数据。如果DBMS不能严格地保证数据库中数据的安全性，就会严重制约数据库的应用。

因此，数据库系统中的数据共享不能是无条件的共享，而必须是在DBMS统一的、严格的控制之下，只允许有合法使用权限的用户访问允许他存取的数据。数据库系统的安全保护措施是否有效是数据库系统主要的性能指标之一。

当然，与数据安全性密切相关的是数据的保密问题，即合法用户合法地访问到机密数据后能否对这些数据保密。这在很大程度上是法律、政策、伦理、道德方面的问题，而不属于技术上的问题，不属于本书讨论的范围。事实上，一些国家已成立了专门机构对数据的安全保密制定了法律道德准则和政策法规。

5.1.1　安全性控制的一般方法

用户非法使用数据库可以有很多种情况，例如，用户编写一段合法的程序绕过DBMS及其授权机制，通过操作系统直接存取、修改或备份数据库中的数据；编写应用程序执行非授权操作；通过多次合法查询数据库从中推导出一些保密数据，如某数据库应用系统禁止查询某人的工资，但允许查询任意一组人的平均工资，用户甲想了解张三的工资，于是他首先查询包括张三在内的一组人的平均工资，然后用自己替换张三后查询这组人的平均工资，从而推导出张三的工资；等等。这些破坏安全性的行为可能是无意的，也可能是故意的，甚至可能是恶意的。安全性控制就是要尽可能地杜绝所有可能的数据库非法访问，不管它们是有意的还是无意的。

实际上，安全性问题并不是数据库系统所独有的，所有计算机化的系统中都存在这个问题，只是由于数据库系统中存放了大量数据，并为许多用户直接共享，使安全性问题更为突出而已。所以，在计算机系统中，安全措施一般是一级一级层层设置的，例如，图5-1就是一种很常用的安全模型。

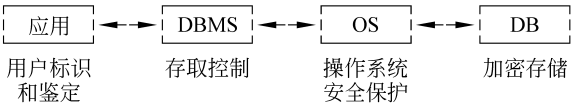

图 5-1　安全模型

图 5-1 的安全模型中，在用户要求进入计算机系统时，系统首先根据输入的用户标识进行用户身份鉴定，只有合法的用户才准许进入计算机系统。对已进入系统的用户，DBMS 还要进行存取控制，只允许用户执行合法操作。操作系统一级也会有自己的安全保护措施。数据最后还可以以加密的形式存储到数据库中。操作系统一级的安全保护措施可参考操作系统的有关书籍，这里不再详述。另外，对于强力逼迫透露口令、盗窃物理存储设备等行为而采取的保安措施，例如，出入机房登记、加锁等，也不在讨论之列。这里只讨论与数据库有关的用户标识和鉴定、存取控制和数据加密等安全性措施。

1. 用户标识和鉴定

用户标识和鉴定是系统提供的最外层安全保护措施。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，由系统将用户提供的身份标识与系统内部记录的合法用户标识进行核对，通过鉴定后才提供机器使用权。用户标识和鉴定的方法有很多种，而且在一个系统中往往是多种方法并举，以获得更强的安全性。

标识和鉴定一个用户最常用的方法是用一个用户名或用户标识号来标明用户身份，系统鉴别此用户是不是合法用户。若是合法用户，则可进入下一步的核实；若不是合法用户，则不能使用计算机。

为了进一步核实用户，在用户输入了合法用户名或用户标识号后，系统常常要求用户输入口令(password)，然后系统核对口令以鉴别用户身份。为保密起见，用户在终端上输入的口令是不显示在屏幕上的。

通过用户名和口令来鉴定用户的方法简单易行，但用户名和口令容易被人窃取，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，在鉴别用户身份时，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。用户可以约定比较简单的计算过程或函数，以便计算起来方便；也可以约定比较复杂的计算过程或函数，以便安全性更好。

用户标识和鉴定可以重复多次。

2. 存取控制

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机制实现，其中自主存取控制和强制存取控制是两类最常见的存取控制机制。

1) 自主存取控制

在数据库系统中，为了保证用户只能访问他有权存取的数据，必须预先对每个用户定义存取权限。对于通过鉴定获得上机权的用户(即合法用户)，系统根据他的存取权限定义对

他的各种操作请求进行控制，确保他只执行合法操作。

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要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，定义存取权限称为授权(authorization)。这些授权定义经过编译后存放在数据字典中。对于获得上机权后又进一步发出存取数据库操作的用户，DBMS 查询数据字典，根据其存取权限对操作的合法性进行检查，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。这就是自主存取控制。

在层次数据库和网状数据库中，用户只能对数据进行操作，存取控制的数据对象也仅限于数据本身。而关系系统中，DBA 可以把建立、修改基本表的权限授予用户，用户获得此权限后可以建立和修改基本表、索引、视图。因此，关系系统中存取控制的数据对象不仅有数据本身，如表、属性列等，还有模式、外模式、内模式等数据字典中的内容，如表 5-1 所示。

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

授权编译程序和合法权检查机制一起组成了安全性子系统。表 5-2 就是一个授权表的例子。

表 5-2　一个授权表的例子(一)

用户名	数据对象名	允许的操作类型
张明	关系 Student	SELECT
李青	关系 Student	ALL
李青	关系 Course	ALL
李青	关系 SC	UPDATE
王楠	关系 SC	SELECT
王楠	关系 SC	INSERT
⋮	⋮	⋮

衡量授权机制是否灵活的一个重要指标是授权粒度，即可以定义的数据对象的范围。授权定义中数据对象的粒度越细，即可以定义的数据对象的范围越小，授权子系统就越灵活。

在关系系统中，实体以及实体间的联系都用单一的数据结构即表来表示，表由行和列组成。所以在关系数据库中，授权的数据对象粒度包括表、属性列、行(记录)。

表 5-2 就是一个授权粒度很粗的表，它只能对整个关系授权，如用户张明拥有对关系 Student 的 SELECT 权；用户李青拥有对关系 Student 和 Course 的一切权限，以及对关系 SC 的 UPDATE 权；用户王楠可以查询关系 SC 以及向关系 SC 中插入新记录。

表 5-3 中的授权表则精细到可以对属性列授权，用户李青拥有对关系 Student 和 Course 的一切权限，但只能查询 SC 关系和修改 SC 关系的 Grade 属性；王楠只能查询 SC 关系的 Sno 属性和 Cno 属性。

表 5-3　一个授权表的例子(二)

用户名	数据对象名	允许的操作类型
张明	关系 Student	SELECT
李青	关系 Student	ALL
李青	关系 Course	ALL
李青	关系 SC	SELECT
李青	列 SC.Grade	UPDATE
王楠	列 SC.Sno	SELECT
王楠	列 SC.Cno	SELECT
⋮	⋮	⋮

表 5-2 和表 5-3 中的授权定义均独立于数据值，用户能否执行某个操作与数据内容无关。而表 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记录授权。例如，张明只能查询信息系(IS)学生的数据。提供与数据值有关的授权，要求系统必须能支持存取谓词。

表 5-4　一个授权表的例子(三)

用户名	数据对象名	允许的操作类型	存取谓词
张明	关系 Student	SELECT	Sname='IS'
李青	关系 Student	ALL	
李青	关系 Course	ALL	
李青	关系 SC	SELECT	
李青	列 SC.Grade	UPDATE	
王楠	列 SC.Sno	SELECT	
王楠	列 SC.Cno	SELECT	
⋮	⋮	⋮	

另外，还可以在存取谓词中引用系统变量。如终端设备号、系统时钟等就是与时间地点有关的存取权限，这样用户只能在某段时间内，某台终端上存取有关数据。例如，规定“教师只能在每年 1 月份和 7 月份的星期一至星期五上午 8 点至下午 5 点处理学生成绩数据”。

可见，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，因数据字典变大、变复杂，系统定义与检查权限的开销也会相应增大。

DBMS 一般都提供了存取控制语句进行存取权限的定义，5.1.2 节将进行介绍。

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何人，以及是否允许他传播这些权限。在这种授权机制下，仍可能存在数据的“无意泄露”。例如，甲将自己权限范围内的某些数据存取权限授权给乙，甲的意图是仅允许乙本人操纵这些数据。但甲的这种

安全性要求并不能得到保证，因为乙一旦获得了对数据的权限，就可以将数据备份，获得自身权限内的副本，并在不征得甲同意的前提下传播副本。造成这一问题的根本原因就在于，这种机制仅仅通过对数据的存取权限来进行安全控制，而数据本身并无安全性标记。要解决这一问题，就需要对系统控制下的所有主客体实施强制存取控制策略。

强制存取控制是指系统为保证更高程度的安全性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。强制存取控制适用那些对数据有严格而固定密级分类的部门，如军事部门或政府部门。

在强制存取控制中，数据库管理系统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，例如数据库用户、代表用户的各进程等。客体是系统中的被动实体，是受主体操纵的，例如文件、基本表、索引、视图等。对于主体和客体，数据库管理系统为它们每个实例（值）指派一个敏感度标记（label）。

敏感度标记被分成若干级别，例如绝密（top secret，TS）、机密（secret，S）、可信（confidential，C）、公开（public，P）等。密级的次序是 TS≥ S ≥ C ≥ P。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）。**强制存取控制机制就是通过对比主体的敏感度标记和客体的敏感度标记，最终确定主体是否能够存取客体**。

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。

3. 定义视图

进行存取权限的控制，不仅可以通过授权与收回权力来实现，还可以通过定义用户的外模式来提供一定的安全保护功能。在关系系统中，就是为不同的用户定义不同的视图，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。但视图机制更主要的功能在于提供数据独立性，其安全保护功能太不精细，往往远不能达到应用系统的要求，因此，在实际应用中通常是视图机制与授权机制配合使用，首先用视图机制屏蔽掉一部分保密数据，然后在视图上面再进一步定义存取权限。

4. 审计

用户识别和鉴定、存取控制、视图等安全性措施均为强制性机制，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可能是完美无缺的，蓄意盗窃、破坏数据的人总是想方设法打破控制。所以，当数据相当敏感，或者对数据的处理极为重要时，就必须以审计技术作为追踪手段，监测可能的不合法行为。

审计追踪使用的是一个专用文件或数据库，系统自动将用户对数据库的所有操作记录在上面，利用审计追踪的信息，就能重现导致数据库现有状况的一系列事件，以找出非法存取数据的人。

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安全性要求较高的部门。

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，以密码的形式存储和传输数据。这样企图通过不正常渠道获取数据（例如，利用系统安全措施的漏洞非法访问数据，或者在通信线路上窃取数据）只能看到一些无法辨认的二进制代码。用户正常检索数据时，首先要提供密码钥匙，由系统进行译码后，才能得到可识别的数据。

目前，不少数据库产品均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。还有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。

所有提供加密机制的系统必然也提供相应的解密程序。这些解密程序本身也必须具有一定的安全性保护措施，否则数据加密的优点也就遗失殆尽了。

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。

5.1.2 SQL 中的安全性控制

SQL 提供了数据库存取控制功能，可用来定义不同用户对于不同数据对象所允许执行的操作，并控制各用户只能存取他有权存取的数据。不同的用户对不同的数据应具有何种操作权力，是由 DBA 和表的建立者（即表的属主）根据具体情况决定的，SQL 则为 DBA 和表的属主定义与回收这种权力提供了安全控制语句。

1. 授权

SQL 用 GRANT 语句向用户授予操作权限，GRANT 语句的一般格式：

```
GRANT <权限>[,<权限>]…
      [ON <对象类型><对象名>]
      TO <用户>[,<用户>]…
      [WITH GRANT OPTION];
```

其语义为将对指定操作对象的指定操作权限授予指定的用户。

不同类型的操作对象有不同的操作权限，常见的操作权限如表 5-5 所示。

表 5-5　不同对象类型允许的操作权限

对象	对象类型	操 作 权 限
属性列	TABLE	SELECT，INSERT，UPDATE，DELETE，ALL PRIVILEGES
视图	TABLE	SELECT，INSERT，UPDATE，DELETE，ALL PRIVILEGES
基本表	TABLE	SELECT，INSERT，UPDATE，DELETE，ALTER，INDEX，ALL PRIVILEGES
数据库	DATABASE	CREATETAB

对属性列和视图的操作权限有 5 类：查询(SELECT)、插入(INSERT)、修改(UPDATE)、删除(DELETE)以及这 4 种权限的总和(ALL PRIVILEGES)。对基本表的操作权限有 7 类：查询(SELECT)、插入(INSERT)、修改(UPDATE)、删除(DELETE)、修改表(ALTER)和建立索引(INDEX)以及这 6 种权限的总和(ALL PRIVILEGES)。对数据库可以有建立表(CREATETAB)的权限，该权限属于 DBA，可由 DBA 授予普通用户，普通用户拥有此权限后可以建立基本表，基本表的属主拥有对该表的一切操作权限。

接受权限的用户可以是一个或多个具体用户，也可以是 PUBLIC 即全体用户。

如果指定了 WITH GRANT OPTION 子句，则获得某种权限的用户还可以把这种权限再授予别的用户。如果没有指定 WITH GRANT OPTION 子句，则获得某种权限的用户只能使用该权限，但不能传播该权限。

例 5.1 把查询 Student 表权限授给用户 U1。

```
GRANT SELECT ON TABLE Student TO U1;
```

例 5.2 把对 Student 表和 Course 表的全部权限授予用户 U2 和 U3。

```
GRANT ALL PRIVILEGES ON TABLE Student, Course TO U2, U3;
```

例 5.3 把对表 SC 的查询权限授予所有用户。

```
GRANT SELECT ON TABLE SC TO PUBLIC;
```

例 5.4 把查询 Student 表和修改学生学号的权限授给用户 U4。

这里实际上要授予 U4 用户的是对基本表 Student 的 SELECT 权限和对属性列 Sno 的 UPDATE 权限。授予关于属性列的权限时必须明确指出相应属性列名。完成本授权操作的 SQL 语句：

```
GRANT UPDATE(Sno), SELECT ON TABLE Student TO U4;
```

例 5.5 把对表 SC 的 INSERT 权限授予 U5 用户，并允许他再将此权限授予其他用户。

```
GRANT INSERT ON TABLE SC TO U5 WITH GRANT OPTION;
```

执行此 SQL 语句后，U5 不仅拥有了对表 SC 的 INSERT 权限，还可以传播此权限，即由 U5 用户发上述 GRANT 命令给其他用户。例如，U5 可以将此权限授予 U6：

```
GRANT INSERT ON TABLE SC TO U6 WITH GRANT OPTION;
```

同样，U6 还可以将此权限授予 U7：

```
GRANT INSERT ON TABLE SC TO U7;
```

因为 U6 未给 U7 传播的权限，因此 U7 不能再传播此权限。

例 5.6 DBA 把在数据库 S_C 中建立表的权限授予用户 U8。

```
GRANT CREATETAB ON DATABASE S_C TO U8;
```

由上面的例子可以看到，GRANT 语句可以一次向一个用户授权，如例 5.1 所示，这是最简单的一种授权操作；也可以一次向多个用户授权，如例 5.2、例 5.3 等所示；还可以一次

传播多个同类对象的权限，如例 5.2 所示；甚至一次可以完成对基本表和属性列这些不同对象的授权，如例 5.4 所示；但授予关于 DATABASE 的权限必须与授予关于 TABLE 的权限分开，因为它们使用不同的对象类型关键字。

2. 收回权限

授予的权限可以由 DBA 或其他授权者用 REVOKE 语句收回，REVOKE 语句的一般格式：

```
REVOKE <权限>[,<权限>]…
    [ON <对象类型><对象名>]
    FROM <用户>[,<用户>]…;
```

例 5.7 把用户 U4 修改学生学号的权限收回。

```
REVOKE UPDATE(Sno) ON TABLE Student FROM U4;
```

例 5.8 收回所有用户对表 SC 的查询权限。

```
REVOKE SELECT ON TABLE SC FROM PUBLIC;
```

例 5.9 把用户 U5 对 SC 表的 INSERT 权限收回。

```
REVOKE INSERT ON TABLE SC FROM U5;
```

在例 5.5 中，U5 又将对 SC 表的 INSERT 权限授予了 U6，而 U6 又将其授予了 U7，执行此 REVOKE 语句后，DBMS 在收回 U5 对 SC 表的 INSERT 权限的同时，还会自动收回 U6 和 U7 对 SC 表的 INSERT 权限，即收回权限的操作会级联。但如果 U6 或 U7 还从其他用户处获得对 SC 表的 INSERT 权限，则他们仍具有此权限，系统只收回直接或间接从 U5 处获得的权限。

可见，SQL 提供了非常灵活的授权机制。用户对自己建立的基本表和视图拥有全部的操作权限，并且可以用 GRANT 语句把其中某些权限授予其他用户。被授权的用户如果有继续授权的许可，还可以把获得的权限再授予其他用户。DBA 拥有对数据库中所有对象的所有权限，并可以根据应用的需要将不同的权限授予不同的用户。而所有授予出去的权力在必要时又都可以用 REVOKE 语句收回。

3. 基于视图的存取控制

视图机制与授权机制相配合，可以间接地实现支持存取谓词的用户权限定义。例如，假定王萍老师只能检索计算机系学生的信息，系主任张凌具有检索、增加、删除和修改计算机系学生信息的所有权限。这就要求系统能支持“存取谓词”的用户权限定义。在不直接支持存取谓词的系统中，可以先建立计算机系学生的视图 CS_Student，然后在视图上进一步定义存取权限。

例 5.10 建立计算机系学生的视图，把对该视图的 SELECT 权限授予王萍，把该视图上的所有操作权限授予张凌。

```
CREATE VIEW CS_Student          /* 先建立视图 CS_Student */
```

```
AS
SELECT *
FROM Student
WHERE Sdept='CS';

GRANT SELECT                    /*王萍老师只能检索计算机系学生的信息*/
ON CS_Student
TO 王萍;
GRANT ALL PRIVILEGES   /*系主任具有检索、增加、删除和修改计算机系学生信息的所有权限*/
ON CS_Student
TO 张凌;
```

4. 审计

许多 DBMS 都提供了比较灵活的审计功能，例如是否使用审计，对哪些表进行审计，对哪些操作进行审计等，都可以由用户自由选择。用户可以通过 AUDIT 语句设置审计功能，NOAUDIT 语句取消审计功能。DBMS 会将审计设置以及审计内容均存放在数据字典中。用户在设置审计时，可以详细指定对哪些 SQL 操作进行审计。例如，如果想对修改 SC 表结构或数据的操作进行审计，可使用如下语句：

```
AUDIT ALTER,UPDATE ON SC;
```

取消对 SC 表的一切审计可使用如下语句：

```
NOAUDIT ALL ON SC;
```

5. 用触发器自定义安全性措施

除了系统级的安全性措施外，DBMS 还允许用户用触发器(trigger)定义特殊的更复杂的用户级安全性措施。

触发器是用户定义在关系表上的一类由事件驱动的特殊过程。一旦定义，触发器将被保存在数据库服务器中。任何用户对表的增加、删除和修改操作均由服务器自动激活相应的触发器，在关系系统核心层进行集中控制。

触发器在 SQL 99 之后才写入 SQL 标准，但是很多关系系统很早就支持触发器，因此不同的关系系统实现的触发器语法各不相同、互不兼容。请读者在上机实验前注意阅读所用系统的使用说明。

由于用触发器定义安全性措施通常都比较简单，所以关于触发器的详细介绍将放在 5.2 节，这里只给一个例子。

例 5.11 规定只能在工作时间内更新 Student 表。

可以定义如下触发器，其中 sysdate 为系统当前时间：

```
CREATE OR REPLACE TRIGGER secure-student
BEFORE INSERT OR UPDATE OR DELETE ON Student
BEGIN
    IF (TO-CHAR(sysdate,'DY') IN ('SAT','SUN'))
```

```
    OR (TO-NUMBER(sysdate,'HH24') NOT BETWEEN 8 AND 17)
  THEN
    RAISE-APPLICATION-ERROR(-20506,
    'You may only change data during normal business hours.')
  END IF;
END;
```

这个例子定义了一个名为 secure-student 的触发器，触发器一经定义后，将存放在数据字典中。用户每次对 Student 表执行 INSERT、UPDATE 或 DELETE 操作时都会自动激活该触发器，由系统检查当时的系统时间，如果是周六或周日，或者不是 8 点至 17 点，系统会拒绝执行用户的更新操作，并提示出错信息。

5.2 完 整 性

数据库的完整性是指数据的正确性和相容性。例如，学生的年龄必须是整数，取值范围为 14～29；学生的性别只能是男或女；学生的学号一定是唯一的；学生所在系必须是学校开设的系；等等。数据库是否具备完整性关系到数据库系统能否真实地反映现实世界，因此维护数据库的完整性是非常重要的。

数据的完整性与安全性是数据库保护两个不同的方面。安全性是防止用户非法使用数据库，包括恶意破坏数据和越权存取数据。完整性则是防止合法用户使用数据库时向数据库中加入不合语义的数据。也就是说，安全性措施的防范对象是非法用户和非法操作，完整性措施的防范对象是不合语义的数据。

为维护数据库的完整性，DBMS 必须提供一种机制来检查数据库中的数据，看其是否满足语义规定的条件。这些加在数据库数据之上的语义约束条件称为数据库完整性约束条件，它们作为模式的一部分存入数据库中。而 DBMS 中检查数据是否满足完整性条件的机制称为完整性检查。

5.2.1 完整性约束条件

整个完整性控制都是围绕完整性约束条件进行的，从这个角度说，完整性约束条件是完整性控制机制的核心。

完整性约束条件作用的对象可以有列级、元组和关系 3 种粒度。其中，对列级的约束主要指对其取值类型、范围、精度、排序等的约束条件；对元组的约束是指对记录中各个字段间的联系的约束；对关系的约束是指对若干记录间、关系集合上以及关系之间的联系的约束。

完整性约束条件涉及的这 3 类对象，其状态可以是静态的，也可以是动态的。其中，对静态对象的约束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；对动态对象的约束是反映数据库状态变迁的约束。

综合上述两方面，可以将完整性约束条件分为 6 类，如图 5-2 所示。

1. 静态列级约束

静态列级约束是对一个列的取值域的说明，这是最常见、最简单，同时也最容易实现的