第3章 关系数据库的定义与完整性的实现

第2章介绍了怎样设计数据库，在数据库设计好以后，可利用SSMS对数据库、表结构进行定义。本章将介绍如何利用SQL语言来定义数据库、表结构及其完整性约束。

SQL是Structured Query Language(结构化查询语言)的缩写。虽然它叫结构化查询语言，查询操作是它的一个主要的操作，但是它不仅仅只有查询功能。实际上，它有4大类功能：数据定义功能、数据查询功能、数据操纵功能和数据控制功能。本章的数据库的定义和表的定义是利用它的数据定义功能来实现。

3.1 SQL语言

SQL语言是用户操作关系数据库的通用语言。本节介绍SQL语言的特点、主要功能以及提供的主要数据类型。

3.1.1 SQL的特点

数据库系统的主要功能是通过数据库支持的数据语言来实现。SQL语言具有如下的特点。

1. 一体化

SQL则集数据查询DSL、数据定义语言DDL、数据操纵DML、数据控制语言DCL的功能于一体，语言风格统一，可以独立完成数据库生命周期中的全部活动，包括：

(1) 定义关系模式，插入数据，建立数据库；

(2) 对数据库中的数据进行查询和更新；

(3) 数据库重构和维护；

(4) 数据库安全性、完整性控制等一系列操作要求。

这就为数据库应用系统的开发提供了良好的环境。特别是用户在数据库系统投入运行后，还可根据需要随时地、逐步地修改模式，并不影响数据库的运行，从而使系统具有良好的可扩展性。

2. 高度非过程化

在使用SQL进行数据操作时，用户无须指明“怎么做”，只要描述清楚要“做什么”，因此不需要了解存取路径。存取路径的选择以及SQL的操作过程由系统自动完成。这不但大

大减轻了用户负担，而且有利于提高数据独立性。

3. 以同一种语法结构提供多种使用方式

SQL 既是独立的语言，又是嵌入式语言。

作为独立的语言，它能够独立地用于联机交互的使用方式，用户可以在终端键盘上直接键入 SQL 命令对数据库进行操作；作为嵌入式语言，SQL 语句能够嵌入到高级语言程序中，供程序员设计程序时使用。而在两种不同的使用方式下，SQL 的语法结构基本上是一致的。这种以统一的语法结构提供多种不同使用方式的做法，提供了极大的灵活性与方便性。

4. 语言简洁，易学易用

SQL 功能极强，但由于设计巧妙，语言十分简洁，完成核心功能只用 9 个动词即可完成。SQL 接近英语口语，因此容易学习，容易使用。

3.1.2 SQL 的主要功能

SQL 语言按其功能可以分为 4 大部分：数据查询(Data Query)、数据操纵(Data Manipulation)、数据定义(Data Definition)和数据控制(Data Control)。表 3.1 列出了实现这四部分功能的动词。

表 3.1 SQL 核心动词

SQL 功能	所使用动词
数据定义	CREATE、DROP、ALTER
数据查询	SELECT
数据操纵	INSERT、UPDATE、DELETE
数据控制	GRANT、REVOKE

1. 数据定义功能

通过 DDL(Data Definition Language)语言来实现。可用来支持定义或建立数据库对象(如表、索引、序列、视图等)，定义关系数据库的模式、外模式、内模式。常用 DDL 语句为不同形式的 CREATE、ALTER、DROP 命令。

2. 数据查询功能

数据查询功能通过 DQL(Data Query Language)语言来实现，通过数据查询语言实现用的各种查询要求。

3. 数据操纵功能

数据操纵功能通过 DML(Data Manipulation Language)语言来实现，DML 包括数据查询和数据更新两种语句，数据查询指对数据库中的数据进行查询、统计、排序、分组、检索等操作。数据更新指对数据的更新、删除、修改等操作。

4. 数据控制功能

数据库的数据控制功能指数据的安全性和完整性。通过数据控制语句 DCL(Data Control Language)来实现。

本章将介绍利用数据定义功能来定义关系数据库、关系模式,以及在定义关系模式时如何实现数据库的完整性约束。

3.1.3　SQL Server 提供的主要数据类型

数据类型是指在特定的列使用什么样数据的类型。如果一个列的名字为 Last_Name,它是用来容纳人名的,所以这个特定列就应该采用 varchar (variable-length character,变长度的字符型)数据类型。我们在定义表结构的时候,必然要指明每个列的数据类型。

每个数据库管理系统所支持的数据类型并不完全相同,下面介绍 Microsoft SQL Server 支持的常用数据类型。

1. 整数数据类型

整数数据类型是最常用的数据类型之一,按照要存储数据的大小有 INT、SMALLINT、TINYINT、BIGINT 类型,表 3.2 列出了这些类型的说明及存储空间。

表 3.2　整型数据类型

整型数据类型	说　明	存储空间/B
bigint	存储从 -2^{63}(−9 223 372 036 854 775 808)到 $2^{63}-1$ (9 223 372 036 854 775 807)范围的整数	8
int	存储从 -2^{31}(−2 147 483 648)到 $2^{31}-1$(2 147 483 647)范围的整数	4
smallint	存储从 -2^{15}(−32 768)到 $2^{15}-1$(32 767)范围的整数	2
tinyint	存储从 0 到 255 之间的整数。	1

2. 浮点数据类型

浮点数据类型用于存储十进制小数。浮点数值的数据在 SQL Server 中采用上舍入(Round up 或称为只入不舍)方式进行存储。所谓上舍入,是指当(且仅当)要舍入的数是一个非零数时,对其保留数字部分的最低有效位上的数值加 1 ,并进行必要的进位。若一个数是上舍入数,其绝对值不会减少。如:对 3.141 592 653 589 79 分别进行 2 位和 12 位舍入,结果为 3.15 和 3.141 592 653 590。表 3.3 给出了浮点数据类型。

表 3.3　浮点数据类型

浮点数据类型	说　明	存储空间/B
float[(n)]	存储从 −1.79E + 308 至 −2.23E−308、0 以及 2.23E−308 至 1.79E+308 范围的浮点数。*n* 有两个值,如果指定的 *n* 在 1～24 之间,则使用 24,占用 4 字节空间;如果指定的 *n* 在 25～53 之间,则使用 53,占用 8 字节空间。若省略 *n*,则默认为 53	4 或 8
real	存储从 −3.40E + 38 到 3.40E + 38 范围的浮点型数	4

续表

浮点数据类型	说　明	存储空间/B
numeric(p,s)或 decimal(p,s)	定点精度和小数位数。使用最大精度时，有效值从 $-10^{38}+1$ 到 $10^{38}-1$。其中，p 为精度，指定小数点左边和右边可以存储的十进制数字的最大个数。s 为小数位数，指定小数点右边可以存储的十进制数字的最大个数，$0 \leqslant s \leqslant p$。$s$ 的默认值为 0	最多 17

3. 字符数据类型

字符数据类型是使用最多的数据类型。它可以用来存储各种字母、数字符号、特殊符号。一般情况下，使用字符类型数据时需在其前后加上单引号。表 3.4 给出了字符数据类型。

表 3.4　字符数据类型

字符串类型	说　明	存 储 空 间
char(n)	固定长度的普通编码字符串类型，n 表示字符串的最大长度，取值范围为 1～8000	n 个字节。当实际字符串所需空间小于 n 时，系统自动在后边补空格
varchar(n)	可变长度的字符串类型，n 表示字符串的最大长度，取值范围为 1～8000	字符数＋2 字节额外开销
nchar(n)	固定长度的统一编码字符串类型，n 表示字符串的最大长度，取值范围为 1～4000	$2n$ 字节。当实际字符串所需空间小于 $2n$ 时，系统自动在后边补空格
nvarchar(n)	可变长度的统一编码字符串类型，n 表示字符串的最大长度，取值范围为 1～4000	2×字符数＋2 字节额外开销

4. 日期和时间数据类型

在 SQL Server 中，日期时间类型是将日期和时间合起来存储的，它没有单独存储的日期和时间。表 3.5 列出了 SQL Server 所支持的日期时间类型。

表 3.5　日期时间类型

日期时间类型	说　明	存储空间/B
smalldatetime	存储 1900 年 1 月 1 日到 2079 年 6 月 6 日的日期只能精确到分钟	4
datetime	定义一个采用 24 小时制并带有秒的小数部分的日期和时间，时间范围是 00:00:00 到 23:59:59.997。默认格式为：YYYY-MM-DD hh:mm:ss.nnn，n 为数字，表示秒的小数部分(精确到 0.00333 秒)	8

3.2　关系数据库的定义

3.2.1　数据库的创建

在 SQL Server 2008 中创建数据库的方法主要有两种：一是在 SQL Server Management Studio 窗口中使用现有命令和功能，通过方便的图形化向导创建(第 1 章已经介绍过)；二

是通过编写 T-SQL 语句创建。

虽然使用图形化向导创建数据库可以方便应用程序对数据的直接调用。但是，有些情况下，不能使用图形化方式创建数据库。比如，在设计一个应用程序时，开发人员会直接使用 T-SQL 在程序代码中创建数据库及其他数据库对象，而不用在制作应用程序安装包时再放置数据库或让用户自行创建。

使用 T-SQL 创建数据库的语法格式如下：

```
CREATE DATABASE database_name
    [ ON
        [ PRIMARY ] [ <filespec> [, …n ]
        [, <filegroup> [, …n ] ]
    [ LOG ON { <filespec> [, …n ] } ]
    ]
    [ COLLATE collation_name ]
    [ WITH <external_access_option> ]
]
```

上述语法中用到了很多种括号，它们本身不是 SQL 语句的部分。比如＜ ＞、[ ]，下面简单介绍一下这些符号的含义，在后面的语法介绍中也要用到这些符号：

尖括号(＜ ＞)中的内容必须要写出来。

方括号([ ])中的内容表示是可选的(即可出现 0 次或 1 次)，比如，[列级完整性约束定义]代表可以有也可以没有列级完整性约束定义。

花括号({ })与省略号(…)一起，表示其中的内容可以出现 0 次或多次。

竖线(|)表示在多个短语中选择一个，比如 term1|term2|term3，表示在 3 个选项中任选一项。竖线也能用在方括号中，表示可以选择有竖线分隔的选项中的一个，但整个句子又是可选的(也就是可以没有选项出现)。

其参数说明如下：

(1) database_name——新数据库的名称。

(2) ON——指定数据库文件或文件组的明确定义。

(3) PRIMARY——指明主数据库文件或主文件组。一个数据库只能有一个主文件，如果没有指定 PRIMARY，那么 CREATE DATABASE 语句中列出的第一个文件将成为主文件。

(4) ＜filegroup＞ ——控制文件组属性。其语法格式为

```
<filegroup> ::= FILEGROUP filegroup_name <filespec> [, …n]
```

其中＜filespec＞为控制文件属性。其格式如下：

```
<filespec> ::=
{
     (
    NAME = logical_file_name,
    FILENAME = 'os_file_name'
        [, SIZE = size [ KB | MB | GB | TB ] ]
        [, MAXSIZE = { max_size [ KB | MB | GB | TB ] | UNLIMITED } ]
        [, FILEGROWTH = growth_increment [ KB | MB | GB | TB | % ] ]
```

```
    ) [,…n ]
}
```

其中有逻辑文件名(NAME)、物理文件名(FILENAME)、初始大小(SIZE,默认单位为MB)、可增大到的最大容量(MAXSIZE)、自动增长(FILEGROWTH)。每个文件之间以逗号分隔。

LOG ON:明确指定存储数据库日志的磁盘文件(日志文件)。LOG ON 后跟以逗号分隔的用于定义日志文件的<filespec>项列表。如果没有指定 LOG ON,将自动创建一个日志文件,其大小为该数据库的所有数据文件大小总和的 25%或 512 KB,取两者之中的较大者。

例 3-1 创建一个名为 Students 的用户数据库,其主文件初始大小为 3MB,文件增长率为 10%,日志文件大小为 1MB,文件增长率为 10%,其中文件均存储在 D 盘根目录下。

代码如下:

```
CREATE DATABASE STUDENTS
ON
( NAME = Students_Data,
    FILENAME = 'D:\Students_Data.mdf',
    SIZE = 3MB,
    MAXSIZE = UNLIMITED,
    FILEGROWTH = 10 % )
LOG ON
( NAME = Students_Log,
    FILENAME = 'D:\Students_Log.ldf',
    SIZE = 1MB,
    MAXSIZE = UNLIMITED,
    FILEGROWTH = 10 % )
```

在 SQL 查询窗口中输入上述代码并执行即可创建指定的数据库。

例 3-2 通过指定多个数据和事务日志文件创建数据库 test。该数据库具有两个 10MB 的数据文件和两个 10MB 的事务日志文件。主文件是列表中的第一个文件,并使用 PRIMARY 关键字显式指定。事务日志文件在 LOG ON 关键字后指定。

```
CREATE DATABASE TEST
ON
PRIMARY
(NAME = test_data,
FILENAME = 'D:\test_dat.mdf',
SIZE = 10,
MAXSIZE = 100,
FILEGROWTH = 5),
(NAME = test_data1,
FILENAME = 'D:\test_dat1.ndf',
SIZE = 10,
MAXSIZE = 100,
FILEGROWTH = 10)
LOG ON
(NAME = test_log,
```

```
FILENAME = 'D:\test_log.ldf',
SIZE = 10MB,
MAXSIZE = 50MB,
FILEGROWTH = 5MB),
(NAME = test_log1,
FILENAME = 'E:\test_log1.ldf',
SIZE = 10MB,
MAXSIZE = 50MB,
FILEGROWTH = 5MB)
```

请注意用于 FILENAME 选项中各文件的扩展名：mdf 用于主数据文件，ndf 用于辅助数据文件，ldf 用于事务日志文件。

3.2.2　数据库的删除

当确定数据库不再使用时，可以删除数据库。删除数据库的语法为

```
Drop database <数据库名>
```

删除数据库时，会将数据库中的所有数据及数据对象一起删除，因此，做该操作时应非常谨慎。

3.3　SQL 表结构的定义

我们知道，在关系数据库中，实体和实体之间的联系都是通过关系（二维表）来进行表示的。因此，表结构是关系数据库中非常重要的数据对象。在第 2 章的数据库设计好了以后，就可以创建数据库的表了。关系数据库的表是二维表，包含行和列，创建表就是定义表所包含的各列的结构，其中包括列的名称、数据类型、约束等。列的名称是人们给列取的名字，便于记忆，一般来说，最好取有意义的名字，比如“学号”或 Sno，而不取无意义的名字，比如 X；列的数据类型说明了列的可取值范围；列的约束更进一步限制了列的取值范围，包括是否取空值、主码约束、外码约束、列取值范围约束等。

3.3.1　基本表的创建

定义基本表使用 SQL 语言数据定义功能中的 CREATE TABLE 语句实现，其一般格式为

```
CREATE TABLE <表名>(
    <列名> <数据类型> [列级完整性约束定义]
{, <列名> <数据类型> [列级完整性约束定义] … }
[, 表级完整性约束定义] )
```

提示：默认情况下，SQL 语言不区分大小写。

其中，＜表名＞是所要定义的基本表的名字，同样，这个名字最好能表达表的应用语义，比如，“学生表”或 Student。

＜列名＞是表中所包含的属性列的名字，“数据类型”指明列的数据类型。一般来说，一个表会包含多个列，因此也就包含多个列定义。

在定义表的同时还可以定义与表有关的完整性约束条件，定义完整性约束时可以在定义列的同时定义，也可以将完整性约束作为独立的项定义。在列定义同时定义的约束称为列级完整性约束，作为表中独立的一项定义的完整性约束称为表级完整性约束。大部分完整性约束都既可以在“列级完整性约束定义”处定义，也可以在“表级完整性约束定义”处定义；但涉及多个列的约束必须在“表级完整性约束定义”处定义。

具体的完整性约束的内容在将 3.4 节中详细阐述。

本节以第 2 章数据库逻辑设计阶段学生数据库中 Student 表的创建为例，说明 SQL 创建数据表的基本方法。STUDENT 表的结构如表 3.6 所示。

表 3.6 Student 表的结构

列 名	数据类型	长 度	能否为空	字段说明
SNO	CHAR	10	否	学号
SNAME	CHAR	10	是	姓名
SSEX	CHAR	2	是	性别
SAGE	INT	4	是	年龄
SDEPT	CHAR	10	是	系

例 3-3 利用 T-SQL 命令创建 Student 表，表的结构如表 3.6 所示。

代码如下：

```
CREATE TABLE Student
(SNO CHAR(10) NOT NULL,
 SNAME CHAR(10),
 SSEX CHAR(2),
 SAGE INT,
 SDEPT CHAR(10)
     )
```

3.3.2 修改表结构

在定义完表之后，如果需求有变化，比如需要添加列、删除列或修改列定义则可以使用 ALTER TABLE 语句实现。ALTER TABLE 语句可以实现添加列、删除列或修改列定义的功能，也可以实现添加和删除约束的功能。

不同的数据库管理系统对 ALTER TABLE 语句的格式可以稍有不同，这里给出 SQL Server 支持的 ALTER TABLE 语句格式。

ALTER TABLE 语句的部分语法格式如下：

```
ALTER TABLE <表名>
  [ ALTER COLUMN <列名> <新数据类型>]          -- 修改列
| [ ADD <列名> <数据类型>                      -- 添加新列
| [ DROP COLUMN <列名> ]                      -- 删除列
| [ADD [CONSTRAINT <约束名>] 约束定义]        -- 添加约束
```

```
| [DROP [CONSTRAINT ]<约束名>]                    -- 删除约束
```

例 3-4　为 Student 表添加“学生宿舍”列，此列的定义为：Room char(8)，允许空。

```
ALTER TABLE Student
    ADD  Room char(8) NULL
```

注：新增加的列只能为空，或默认，不能为 NOT NULL。

例 3-5　将新添加的 Room 列的类型改为 char(6)。

```
ALTER TABLE Student
    ALTER COLUMN Room char(6)
```

例 3-6　删除 Student 表中新添加的 Room 列。

```
ALTER TABLE Student
    DROP COLUMN Room
```

3.3.3　删除表

当确定不再需要某个表时，可以将其删除。删除表时会将与表有关的所有对象一起删掉，包括表中的数据。

删除表的语句格式为

```
DROP TABLE <表名> [RESTRICT|CASCADE]
```

例 3-7　删除 Test 表的语句为

```
DROP TABLE Test
```

3.4　完整性约束

数据的完整性是指数据库中存储的数据的正确性和相容性。正确性是指数据要符合具体的语义，相容性是指数据的关系要正确。例如，人的性别只能是“男”或“女”，学生选课必须是课程表中已开的课程才行。

数据完整性约束是为了防止数据库中存在不符合语义的数据，为了维护数据的完整性，数据库管理系统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就是数据完整性约束。主要包括 3 大类：

- 实体完整性。
- 参照完整性。
- 用户定义完整性。

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。当用户定义好了数据完整性，后续执行对数据的增加、删除、修改操作时，数据库管理系统都会自动检查用户定义的完整性约束，只有符合约束条件的操作才会被执行。下面从 DBMS 定义完整性约束、完整性检查、违约处理这 3 方面来学习 3 大类完整性。

3.4.1 实体完整性

实体完整性保证关系中的每个元组都是可识别的和唯一的。

在关系数据库中，用主码来保证实体完整性。要求关系数据库中的表都必须有主码，而且对主码的取值有要求：

- 主码的各个属性不能为空值。
- 任意两个元组的主码值不能相同。

因此，可以通过定义主码来保证实体完整性。

如果在列级完整性约束定义主码（仅用于单列主码），则语法格式为

```
<列名> 数据类型 PRIMARY KEY
```

例：

```
SNO char(7) PRIMARY KEY
```

如果在定义完列时，作为表级完整性定义主码（用于单列或多列主码），则语法格式为

```
PRIMARY KEY (<列名>,[,…n])
```

例 3-8 为 Student 表添加主码约束：Sno。

```
PRIMARY KEY(Sno)
```

为 SC 表添加主码，主码为（Sno，Cno）

```
PRIMARY KEY(Sno,Cno)
```

如果在表建好了以后，为表添加主码约束的语法格式为

```
ALTER TABLE 表名
      ADD [CONSTRAINT <约束名>]
      PRIMARY KEY (<列名>,[,…n])
```

例 3-9 对 Student 表添加主码约束。

```
ALTER TABLE Student
  ADD  PRIMARY KEY(Sno)
```

或

```
ALTER TABLE Student
    ADD CONSTRAINT PK_Course PRIMARY KEY(Sno)
```

注意：每个表只能由一个 PRIMARY KEY 约束。

如果表中定义了主码，当我们在做如下操作的时候，数据库管理系统会自动检查我们的数据是否符合实体完整性：

- 插入元组。
- 修改主码属性的值。

如果数据不符合实体完整性，一般来说，数据库管理系统会拒绝刚刚所做的操作。

3.4.2　参照完整性

参照完整性也称为引用完整性。现实实践中的实体之前往往存在着某种联系，在关系模型中，实体与实体之间的联系都是用关系来表示的，这样就自然存在着关系与关系之间的引用。因此，参照完整性就是用来描述实体之间的联系的。

例如，学生实体和班级实体用可以用下面的关系模式表示，其中主码用下划线标识：

学生(学号，姓名，性别，班号，年龄)
班级(班号，所属专业，班主任，人数)

这两个关系模式之间存在着联系，即学生关系中的班号参照了班级关系中的班号。学生关系中的班号的值如果为空，则表示该学生没有分到任何班级；如不为空的话一定要是班级关系中确实存在的班号值。也就是说，学生关系中的班号的取值参照了班级关系中的班号的取值。这种限制一个关系中的某列的取值受另一个关系中某列的取值范围的约束的特点就称为参照完整性。

与实体间的联系类似，不仅实体之间存在着引用关系，同一个关系的内部属性之间也可以存在引用关系。

例如，职工关系模式

职工(职工号，姓名，性别，主任职工号)

职工号为主码，事实上，某个职工的主任也应该是该企业的一名职工，因此，主任职工号一定该关系模式中的职工号属性的取值一个。

进一步定义外码。

定义：设 F 是关系 R 的一个或一组属性，如果 F 与关系 S 的主码相对应，则称 F 是关系 R 的外码，并称关系 R 为参照关系，关系 S 为被参照关系。关系 R 和 S 不一定是不同的关系。

在学生关系中，班号属性的与班级关系中的主码班号相对应，因此，学生关系中的班号为外码，引用了班级关系中班号。这里班级关系时被参照关系，学生是参照关系。

显然，外码与相对应的主码应该有相同的数据类型，但是不一定要相同的名字，例如职工关系模式的主任职工号和职工号。但在实际应用中为了便于识别，当外码与相应的主码属于不同的关系时，一般给它们取相同的名字。

因此，可以通过外码来保证参照完整性。外码的取值，一般应符合如下要求：

- 或者为空值。
- 或者等于其所参照的关系中的某个元组的主码。

参照完整性的定义就是通过定义外码来实现。

一般情况下，外码都是单列的，它可以定义在列级完整性约束处，也可以定义在表级完整性约束处。定义外码的语法格式为

```
[ FOREIGN KEY (<本表列名>)] REFERENCES <外表名>(<外表主码列名>)
[ON DELETE {CASCADE|NO ATION|SET NULL}]
[ON UPDATE {CASCADE|NO ATION|SET NULL}]
```

如果是在列级完整性约束处定义外码，则可以省略“FOREIGN KEY（<本表列名>）”部分，如果是在表级完整性约束处定义外码，则不能省略。

其中{CASCADE|NO ATION|SET NULL}为级联引用完整性，说明当某个主码值被删除/更新时（这个主码值在被参照关系中）如何处理对应的外部码值（这些外部码值在参照关系中）。

- CASCADE 方式：连带将所有对应的外码值一起删除/更新（删除外码值，实际上就是将所在的元组删除）。
- NO ACTION 方式：仅当没有任何对应的外码值时，才可以删除/更新这个主码值，否则系统拒绝执行此操作。
- SET NULL 方式：将所有对应的外码值设为空值。

例 3-10 选课关系（SC）中的学号（Sno）参照学生关系（Student）中的学号（Sno）。

```
FOREIGN KEY(Sno)REFERENCES Student(Sno)ON DELETE CASCADE
```

例 3-11 选课关系（SC）中的课程号（Cno）参照课程关系（Course）中的课程号（Cno）。

```
FOREIGN KEY(Cno)REFERENCES Course(Cno)
```

若用户在对参照表进行插入或者修改操作时违反了参照完整性，则数据库管理系统会拒绝用户所做的操作。

当用户在对被参照表进行修改或者删除操作时违反了参照完整性，数据库管理系统按照外码定义时说明的方法来处理外码值，默认情况下为拒绝。

3.4.3 用户定义完整性

用户定义完整性也称为域完整性或语义完整性。任何关系数据管理系统都应该支持实体完整性和参照完整性。除此之外，不同的数据库应用系统根据应用环境的不同，往往还需要一些特殊的约束条件，用户定义完整性就是针对某一具体应用领域定义的数据库约束条件。它反映某一具体应用所涉及的数据必须满足应用语义的要求。

用户定义的完整性实际上就是指明关系中属性的取值范围，也就是属性的域，即限制关系中的属性的取值类型及取值范围，防止属性的值与应用语义矛盾。例如，学生的性别取{男、女}，学生的成绩在 0～100 之间。

用户定义完整性可以通过以下约束来保证：NOT NULL 约束、CHECK 约束、DEFAULT 约束和 UNIQUE 约束。

1. NOT NULL 约束

限制列取值非空，它只能作为列级完整性约束定义，不能作为表级完整性约束定义，定义非空约束的语法格式为

```
<列名> <数据类型> NOT NULL
```

例 3-12 限制学生姓名列不能取空值。

```
Sname CHAR(10) NOT NULL
```

2. CHECK 约束

CHECK 约束用于限制输入一列或多列的值的范围，通过逻辑表达式来判断数据的有效性，也就是一个列输入内容必须满足 CHECK 约束的条件；否则，数据无法正常输入，从而强制数据的域完整性。定义 CHECK 约束的语法格式为

```
[ CONSTRAINT 约束名] CHECK(逻辑表达式)
```

CHECK 约束可以作为列级完整性约束定义，也可以作为表级完整性约束定义，语法格式相同。但是，当表达式涉及多列时，只能作为表级完整性约束来定义。

例 3-13　在 Student 表中（如表 3.6 所示）要求 SSEX 这一列的值要求只能取“男”或“女”，如果用户输入其他值，系统均提示输入无效。

```
CHECK(SSEX = '男' OR SSEX = '女')
```

如果是在已建好的表中增加 CHECK 约束，则语法格式为

```
ALTER TABLE 表名
  ADD [ CONSTRAINT 约束名]
  CHECK(逻辑表达式)
```

例 3-14　在 Student 表中为学生年龄 Sage 增加约束，限制其取值在[10,40]之间

```
ALTER TABLE Student
    ADD CONSTRAINT CHK_Sage CHECK(Sage >= 10 and Sage <= 40)
```

3. DEFAULT 约束

用于提供列的默认值。若在表中某列定义了 DEFAULT 约束，用户在插入新数据行时，如果该列没有指定数据，那么系统将默认值赋给该列。只有在向表中插入数据时系统才检查 DEFAULT 约束。其语法格式为

```
<列名> <数据类型> DEFAULT 默认值
```

例 3-15　为 Student 表中为学生所在系 Sdept 增加默认值约束，默认值为“计算机系”。

```
Sdept CHAR(20) DEFAULT '计算机系'
```

如果是在已建好的表中增加 DEFAULT 约束，则语法格式为

```
ALTER TABLE 表名
  ADD [ CONSTRAINT 约束名]
   DEFAULT 默认值 FOR 列名
```

在例 3-15 中，如 Student 已经建好，则 DEFAULT 约束应该写为

```
ALTER TABLE 表名
  ADD CONSTRAINT DF_SDEPT
   DEFAULT '计算机系' FOR Sdept
```

4. UNIQUE 约束

UNIQUE 约束用于限制列中不能有重复值。这个约束用在事实上具有唯一性的属性列上，比如每个人的身份证号码，手机号码，电子邮件等均不能有重复值。定义 UNIQUE 约束时需要注意如下事项：

- UNIQUE 约束的列允许有一个空值；
- 在一个表中可以定义多个 UNIQUE 约束；
- 可以在多个列上定义一个 UNIQUE 约束，表示这些列组合起来不能有重复值。

它可以作为列级完整性约束定义，其语法格式为

```
<列名> <数据类型> UNIQUE
```

也可以表级完整性约束定义，但是 UNIQUE 约束涉及多列时，只能作为表级完整性来定义。其语法格式为

```
UNIQUE(列名[,…n])
```

例 3-16 为 Student 表的学生姓名 Sname 列添加 UNIQUE 约束。

```
Sname char(7) UNIQUE
```

或

```
UNIQUE(Sname)
```

上述这些约束都可以在定义表的时候同时定义，对第 2 章设计的学生选课数据库中的 3 张表：学生表、课程表、选课表进行定义。这 3 张表的结构如表 3.7～表 3.9 所示。

表 3.7 Student 表

列名	说明	数据类型	约束说明
Sno	学号	字符串，长度为 10	主键
Sname	姓名	字符串，长度为 8	取值唯一
Ssex	性别	字符串，长度为 1	取“男”或“女”
Sage	年龄	整数	取值范围为(15,45)
Sdept	所在系	字符串，长度为 15	默认值“计算机系”

表 3.8 Course 表

列名	说明	数据类型	约束说明
Cno	课程号	字符串，长度为 6	主码
Cname	课程名	字符串，长度为 20	非空值
Pcno	先行课程号	字符串，长度为 6	外码，参照本表中的 Cno
Credits	学分	整数	取值大于零

表 3.9　SC 表结构

列　名	说　明	数据类型	约束说明
Sno	学号	字符串,长度为 10	外码,参照 Students 的主码
Cno	课程号	字符串,长度为 6	外码,参照 Courses 的主码
Grade	成绩	整数	取值范围为[0,100]

创建满足约束条件的上述 3 张表的 SQL 语句如下(为了说明问题,这里将有些约束定义在列级,有些定义在表级):

```
CREATE TABLE Student (
  Sno char (7) PRIMARY KEY,
  Sname char (10) UNIQUE,
  Ssex char (2) CHECK (Ssex = '男' OR Ssex = '女'),
  Sage tinyint CHECK (Sage > = 15 AND Sage < = 45),
  Sdept char (20) DEFAULT '计算机系'
)
CREATE TABLE Course (
  Cno char(6) NOT NULL,
  Cname char(20) NOT NULL,
  Pcno char(6),
  Ccredit int CHECK (Ccredit > 0),
  PRIMARY KEY(Cno),
FOREIGN KEY (Pcno) REFERENCES Course(Cno),
)
CREATE TABLE SC(
Sno char(10) REFERENCES Student(Sno),
Cno char(6) REFERENCES Course(Cno),
Grade int CHECK(grade < = 100 and grade > = 0),
PRIMARY KEY(Sno,Cno)
      )
```

在完整性约束定义好了以后,当用户在对数据库中的数据做增加、删除、修改操作时,数据库管理系统会自动检查数据是否符合完整性约束,若不符合,则拒绝所做的操作或者按照完整性定义时说明的方法来处理。

本章小结

本章首先介绍了 SQL 的特点、功能以及所支持的数据类型。SQL 的功能包括数据定义功能、数据查询功能、数据操纵功能和数据控制功能。

本章重点介绍了利用 SQL 对数据库的创建、删除;基本表的创建、修改和删除;详细介绍了关系数据库的 3 大类完整性:实体完整性、参照完整性、用户定义完整性的实现方法。实体完整性通过定义主码来保证;参照完整性利用外码来保证;而用户定义完整性通过 NOT NULL 约束、UNIQUE 约束、CHECK 约束、DEFAULT 约束等约束来保证。

习题 3

一、选择题

1. SQL 语言是(　　)的语言,容易学习。

A. 过程化　　B. 非过程化　　C. 格式化　　D. 导航式

2. SQL 语言的数据操纵语句包括 SELECT、INSERT、UPDATE、DELETE 等。其中最重要的,也是使用最频繁的语句是(　　)。

A. SELECT　　B. INSERT　　C. UPDATE　　D. DELETE

3. 在视图上不能完成的操作是(　　)。

A. 更新视图　　B. 查询

C. 在视图上定义新的表　　D. 在视图上定义新的视图

4. SQL 语言集数据查询、数据操纵、数据定义和数据控制功能于一体,其中,CREATE、DROP、ALTER 语句可实现哪种功能?(　　)

A. 数据查询　　B. 数据操纵　　C. 数据定义　　D. 数据控制

5. SQL 语言中,删除一个视图的命令是(　　)。

A. DELETE　　B. DROP　　C. CLEAR　　D. REMOVE

6. 在 SQL 语言中的视图 VIEW 是数据库的(　　)。

A. 外模式　　B. 模式　　C. 内模式　　D. 存储模式

7. 下列的 SQL 语句中,(　　)不是数据定义语句。

A. CREATE TABLE　　B. DROP VIEW

C. CREATE VIEW　　D. GRANT

8. 若要撤销数据库中已经存在的表 S,可用(　　)。

A. DELETE TABLE S　　B. DELETE S

C. DROP TABLE S　　D. DROP S

9. 若要在基本表 S 中增加一列 CN(课程名),可用(　　)。

A. ADD TABLE S CN CHAR(8)

B. ADD TABLE S ALTER CN CHAR(8)

C. ALTER TABLE S ADD CN CHAR(8)

D. ALTER TABLE S ADD CN CHAR(8)

10. 学生关系模式 S(S#,Sname,Sex,Age),S 的属性分别表示学生的学号、姓名、性别、年龄。要在表 S 中删除一个属性"年龄",可选用的 SQL 语句是(　　)。

A. DELETE Age from S　　B. ALTER TABLE S DROP Age

C. UPDATE S Age　　D. ALTER TABLE S 'Age'

二、简答题

1. 试述 SQL 语言的特点。

2. 试述 SQL 的定义功能。

3. RDBMS 在实现参照完整性时需要考虑哪些方面的问题？可以采取哪些策略？

三、设有一个图书馆数据库，用 SQL 语句建立其中的 3 个表：图书表、读者表和借阅表。3 个表的结构如下：

图书表

列　名	说　明	数据类型	约束说明
书号	图书唯一的编号	字符串，长度为 20	主键
书名	图书的名称	字符串，长度为 50	非空值
作者	图书的编著者名	字符串，长度为 30	空值
出版社	图书的出版社	字符串，长度为 30	空值
单价	出版社确定的图书的单价	浮点型，float	[0，100]

读者表

列　名	说　明	数据类型	约束说明
读者号	读者唯一的编号	字符串，长度为 10	主键
姓名	读者姓名	字符串，长度为 8	非空值
性别	读者的性别	字符串，长度为 2	只能取男或者女
电话	读者性别	字符串，长度为 8	唯一
部门	读者办公电话	字符串，长度为 30	默认值为会计系

借阅表

列　名	说　明	数据类型	约束说明
读者号	读者唯一的编号	字符串，长度为 10	主键，外码
书号	图书唯一的编号	字符串，长度为 20	主键，外码
借出日期	借出图书的日期	Datatime 类型，为'yymmdd'	非空值
归还日期	归还图书的日期	Datatime 类型，为'yymmdd'	空值

1. 在图书表中增加“存放位置”列。
2. 为读者表的姓名列增加唯一约束。
3. 将读者表中电话的数据类型为 char(11)。
4. 将图书表中的“存放位置”列删除。