第 5 章

SQL 基 础

5.1 关 于 SQL

SQL 即结构化查询语言(Structured Query Language),最早由 Boyce 和 Chamberlin 在 1974 年提出,当时名为 Sequel 语言,在 IBM 公司研制的关系数据库管理系统原型 System R 上实现。1986 年 10 月,美国国家标准协会(American National Standard Institute, ANSI)的数据库委员会批准 SQL 作为美国的关系数据库标准语言,同年公布了 SQL 标准文本(SQL-86)。1987 年,国际标准化组织(International Organization for Standardization, ISO)也通过了这一标准。此后,ISO 不断推出新的 SQL 标准:SQL/89、SQL/92、SQL/99、SQL2003、SQL2008、SQL2011 等。

SQL 作为关系数据库的标准语言,为众多关系数据库管理系统使用。然而,每一种关系数据库管理系统都不可能支持 SQL 标准的全部概念与特性。本书仅仅介绍 SQL 的核心内容。更为丰富的 SQL 功能请参阅相关关系数据库管理系统(例如 SQL Server 2012)的书籍以及联机文档。

SQL 具有如下四大特点。

(1) 兼具数据定义、数据操纵和数据控制 3 项功能。

关系数据库之所以成为使用广泛的数据模型,其中重要的一点就在于 SQL 的优势。关系数据库之前的层次模型、网状模型所使用的数据库语言分为多种,有截然不同的语法,提供不同的功能。当时的数据库语言具体分为以下 3 类:

- 数据定义语言(Data Definition Language,DDL),用于模式、外模式等相关数据库对象的定义。
- 数据操纵语言(Data Manipulation Language, DML),用于对数据进行增、删、改、查等操作。
- 数据控制语言(Data Control Language,DCL),用于权限管理、事务管理等控制操作。

而关系数据库使用的标准语言 SQL 集数据定义、数据操纵和数据控制为一体;一种语言同时兼具 3 项功能。用户不必学习多种不同的语言句法,分别进行模式定义、外模式定义、查询与更新等。

(2) 高度非过程化。

SQL 之前的数据库语言是过程化语言。例如,查询时需要知道查询目标具体存储的数据结构是数组还是链表,需要给出具体的定位数据的方式、操作过程以及内外循环的设

定，等等。

而SQL属于第四代语言——非过程化语言。在SQL中，对于同一个查询问题的表达可以用多种不同形式的语句实现。关系数据库管理系统从SQL语句中获得的信息是根据什么条件查询什么，而具体的数据结构、查询方法、循环结构以及查询优化方案均由关系数据库管理系统内部确定，与用户的语句无关。换句话说，用户写的SQL语句仅告知关系数据库管理系统做什么，不必告知关系数据库管理系统怎么做。关系数据库管理系统的具体存储与操作过程对于用户是透明的，用户不必过问。

(3) 集合操作。

关系数据模型中的关系是元组的集合，数据的插入相当于集合的并，数据的删除相当于集合的差，数据的查询也是对元组集合、属性集合的选取。因此，SQL的操作为集合运算，SQL操作有强大的数学基础作为支撑。SQL的操作对象是元组的集合，运算结果也是元组的集合。与SQL之前的数据库操纵语言按逐条记录操作的方式相比，SQL的操作更为简便，更加贴近实际需求。

(4) 自然语言，简单易学。

SQL使用简单明了的自然语言(英语)中的词汇表达，易学易用。

5.2 本章使用的数据库模式

介绍SQL需要有具体的数据库案例。本章使用前面几章介绍过的教学信息管理数据库、图书信息管理数据库和航班信息管理数据库，基本表管理与数据更新部分主要使用教学信息管理数据库，数据查询部分主要使用图书信息管理数据库和航班信息管理数据库。

这3个数据库相应的关系模式如下。其中，主码用下画线标出，外码用斜体表示。

(1) 教学信息管理数据库。

学生(学号，姓名，性别，年龄，手机号，Email，班号)

班级(班号，专业，班主任号)

课程(课号，课名，学分，类别)

教师(职工号，姓名，性别，出生日期，职称，专业方向)

选修(学号，课号，成绩)

授课(班号，课号，职工号，学期，教室，时段)

(2) 图书信息管理数据库。

图书馆(图书馆号，名称，地址，电话)

图书(ISBN，书名，类别，语言，定价，开本，千字数，页数，印数，出版日期，印刷日期，出版社号)

出版社(出版社号，名称，国家，城市，地址，邮编，网址)

作者(作者编号，姓名，性别，出生日期，国籍)

收藏(*ISBN*，图书馆号，收藏日期)

编著(*ISBN*，作者编号，类别、排名)

(3) 航班信息管理数据库。

航线(航线号,出发地,到达地,飞行距离)

航班(航班号,日期,起飞时间,到达时间,机长号,机型,航线号)

乘客(编号,姓名,性别,出生日期)

职工(职工号、姓名、性别、年龄、工龄、职务)

飞机类型(机型,名称,通道数,载客人数,制造商)

工作(航班号,职工号)

乘坐(乘客号,航班号,座位号)

5.3 基本表管理

数据库、基本表、视图、触发器以及约束等均称为数据库对象。SQL 中数据库对象的建立使用 CREATE 语句,相应的数据库对象的删除使用 DROP 语句,而数据库对象结构的更改使用 ALTER 语句。所谓基本表管理就是基本表的建立(CREATE TABLE)、删除(DROP TABLE)以及表结构的更改(ALTER TABLE)。

要实现一个数据库应用系统,通常需要先完成建立数据库、在数据库中建表、向表中添加数据等步骤,然后才能进行数据的查询。建立数据库和删除数据库的句法请参看数据库管理系统方面的书籍以及联机文档。本节使用教学信息管理数据库,假定数据库已经建立,直接介绍基本表的建立、修改表结构以及删除表的操作。

建立基本表时需要定义表中各个列的数据类型。在介绍基本表的建立之前,首先讲解 SQL 中常见的数据类型。

5.3.1 SQL 的数据类型

不同的数据库管理系统提供的数据类型略有不同,在此介绍 SQL Server 中常用的类型,其他数据库管理系统提供的基本数据类型与此类似。

(1) int:整数,占 4 字节。

(2) real:实数,占 4 字节。

(3) char(n):定长字符串,n 表示字符串固定的长度。

(4) varchar(n):变长字符串,n 表示字符串最大的长度。

(5) nchar(n):固定长度的 Unicode 字符串(例如汉字串),n 为字符个数,每个字符占 2 字节。

(6) nvarchar(n):可变长度的 Unicode 字符串(例如汉字串),n 为字符个数,每个字符占 2 字节。

(7) bit:二进制位,取值为 0 或 1,通常用来取代布尔类型。

(8) date:日期类型,占 3 字节。数据格式可以设置为 YYYY-MM-DD,其中,YYYY 为年,MM 为月,DD 为日。

(9) money:货币类型,占 8 字节。

5.3.2 基本表的建立

完整的建表语句的句法相当繁杂，在此通过一些实用的例子介绍建表语句的核心内容，详尽的语法格式请参看数据库管理系统方面的书籍或联机文档。

建表时，除了给出各个列的数据类型之外，还需要指定各种数据约束。这些约束包括主码约束、外码约束、唯一性约束（UNIQUE 约束）、CHECK 约束以及默认值。其中，每一个表的定义中必须包含主码约束。

数据库管理系统提供 3 种完整性——实体完整性、参照完整性和用户定义完整性保障。主码约束用来保障实体完整性，外码约束用来保障参照完整性，其他约束用来保障用户定义的完整性。

在表定义中，列的数据约束定义在列名、数据类型之后，而针对表的数据约束放在所有列定义之后。

下面通过例子逐一介绍建表时如何表达这些约束。

例 5.1 根据 5.2 节给出的关系模式，给出教师表的建表语句。

```
CREATE 教师
(  职工号 CHAR(4)  primary key,                                    //主码约束
   姓名 VARCHAR(20) NOT NULL,                                      //非空约束
   性别 NCHAR(1) NOT NULL CHECK(性别 IN ('男','女')),              //取值约束
   出生日期 DATE,
   职称 NCHAR(5) CHECK(职称 IN ('讲师','副教授','教授')),
   专业方向 VARCHAR(20)
)
```

这个例子包含了“职工号”列的主码约束、“姓名”“性别”列的非空约束和“性别”“职称”列的 CHECK 约束。

1. 主码约束

每一个基本表都要利用 primary key 定义一个主码，该主码便是在数据库设计时从候选码中选出的与业务操作相适应的码。主码一旦定义，则意味着其取值一定非空且唯一，即其可以确定表中唯一的行，主码不包含冗余的属性。

本例中的主码由单一属性列组成，可以将 primary key 作为列约束置于列名“职工号”以及数据类型之后。通常主码的实施是通过建立唯一性索引实现的。主码上建立簇式索引，可以保证表中的行在物理上按照主码的顺序排列。

主码可以由一个属性列构成，也可以由多个属性列构成。当多个属性列构成主码的时候，主码中缺少任何一个属性列都不能唯一确定表中的一行。对于多个属性列构成的主码，主码约束必须作为表约束单独定义，置于列定义之后，参看下面的例 5.2。

2. 非空约束

建表的时候除了定义主码约束之外，最为常见的数据约束就是非空约束，例如“姓名”列和“性别”列。在列定义中用 NOT NULL 代表非空，表示该列必须赋值，不能为空值。列定义中不加非空说明时，默认为可空列。主码中的列默认为非空。

所谓空值指的是未知的或者说不确定的值。例如，一个商品没有给出价格，价格为空

值，绝不能说该商品价格为 0。因此空值不是数字 0、空格、空串等。数据库系统对于空值会有特殊的处理，而空值在参与运算时也有其特殊的处理，例如空值参与比较结果为“未知”。

3. CHECK 约束

教师表的“性别”列、“职称”列使用了 CHECK 约束，设定该列的取值范围，其表达方法同查询语句中 WHERE 条件子句的表达相同。有关 WHERE 条件子句请参看 5.5.1 节。本例中 CHECK(性别 IN ('男','女'))表示性别的取值必须为汉字“男”或“女”。插入(或更新)数据时，如果为其他值，系统拒绝插入(或更新)数据。

除了列的 CHECK 约束之外，还可以设定表级 CHECK 约束，即基本表中多列的取值或列之间的关联的约束。一个基本表可以有多个 CHECK 约束。表级 CHECK 约束较为复杂，有关内容在第 6 章中加以介绍。

4. 外码约束

外码约束表达的是基本表之间的关联，反映的是实体集之间联系的类型，可以是一对一、多对一、多对多等。

外码约束的意义在于，当外码非空的时候，其取值对应被参照的基本表中唯一的主码值。例如，增加选课记录('10070012','1002',85)，那么，学生表中必须有学号为 10070012 的学生，课程表中必须有课号为 1002 的课程，否则无法插入此选课记录。

在 SQL 中，允许外码为空值。如果具体表的外码要求非空，需要额外增加非空约束的定义。

单列构成的外码可以放在列定义中，也可以单独定义。特殊情况下，多列构成外码，则必须作为表约束单独定义。此外，并不是每一张基本表都有外码约束。

外码约束的维护需要使用置空、级联、默认 3 种机制，相关内容参看 6.3.1 节。

下面看一个包含组合主码以及外码的例子。

例 5.2 根据 5.2 节中关系模式的定义，给出选课表的建表语句。选课表表达了选课这个多对多联系，主码为组合码(学号，课号)。“学号”为一个外码，参照学生表的主码“学号”；“课号”为另一个外码，参照课程表的主码“课号”。

```
CREATE TABLE 选课
(  学号 CHAR(8) NOT NULL REFERENCES 学生(学号),          //外码约束在列定义中
   课号 CHAR(4) NOT NULL REFERENCES 课程(课号),          //外码约束在列定义中
   成绩 SMALLINT,
   PRIMARY KEY(学号,课号)                                //单独定义主码约束
)
```

组合主码必须单独定义，PRIMARY KEY(学号，课号)作为表约束置于所有列定义之后。外码通常由单列构成，上面为列定义方式，还可以用下面的表约束方式单独定义：

```
CREATE TABLE 选课
(  学号 CHAR(8) NOT NULL,
   课号 CHAR(4) NOT NULL,
   成绩 SMALLINT,
```

```
    PRIMARY KEY(学号,课号),                              // 单独定义主码约束
    FOREIGN KEY (学号) REFERENCES 学生(学号),            // 单独定义外码约束
    FOREIGN KEY (课号) REFERENCES 课程(课号)             //单独定义外码约束
)
```

5. UNIQUE 约束

UNIQUE 约束即**唯一性**约束,它与主码约束的唯一区别是不要求非空。例如,学生表中“手机号”列可以使用唯一性约束,保证每一个学生手机号不同,但不要求每一位学生都保存手机号。

例 5.3 根据 5.2 节的关系模式,给出学生表的建表语句。

```
CREATE 学生
(  学号 CHAR(8) primary key,                          //主码约束
   姓名 VARCHAR(20) NOT NULL,
   性别 NCHAR(1) NOT NULL CHECK(性别 IN ('男','女')),
   年龄 SMALLINT CHECK(年龄>=13 AND 年龄<=30 ),
   手机号 CHAR(11) UNIQUE,                            //唯一性约束
   EMAIL VARCHAR(20),
   班号 CHAR(6) REFERENCES 班级(班号)                 //外码约束
)
```

对于基本表中主码之外的其他候选码(例如,学生表中除了学号还有身份证号,那么身份证号是另一个候选码),不能用主码约束,可以使用唯一性约束加上非空约束实现。

此例中“班号”为外码,表达了学生与班级之间的多对一联系。

6. 默认值

例 5.4 根据 5.2 节的关系模式,给出课程表的建表语句。

```
CREATE 课程
(  课号 CHAR(4) primary key,                                    //主码约束
   课名 VARCHAR(20) NOT NULL,
   学分 SMALLINT NOT NULL),
   类别 VARCHAR(10) CHECK (类别 IN ('必修','选修','专业必修') DEFAULT '必修'
                                                                //默认值
)
```

此例包含了默认值的定义。当课程类别没有具体的输入数值的时候,自动填入“必修”。

默认值属于一种特定的约束类型。默认值可以保证数据非空且具有合理性,当然也可以简化大量重复数据的输入。建表语句中用 DEFAULT 定义默认值,当插入数据时如果没有给定该列的具体值,则该列填入默认值。

注意:这里没有给出教学信息管理数据库所有基本表的建立语句。实际建立数据库表的时候,要注意相关基本表的建立次序。由于外码要参照另外一个表的主码,建表时,务必先建立主码所在的表,后建立外码所在的表。例如,班级表的外码“班主任号”参照教师表的“职工号”,因此,要先建立教师表,后建立班级表。同理,要先建立班级表,再建立学生表。选课表要在学生表与课程表建立之后才可建立。

关于约束的命名、定义、修改与删除参看 6.3.1 节。

另外，断言、触发器也是数据库管理系统维护数据约束的手段，有关内容在 6.3.2 节和 6.3.3 节中有详细的说明。

5.3.3 修改基本表的结构

初始建表的时候，表的定义或许不够完善。SQL 提供了 ALTER TABLE 语句用于修改表的结构，例如增加列、删除列，添加非空约束、默认值和 CHECK 约束，修改列的数据类型，等等。ALTER TABLE 语句完整的句法功能强大，但非常复杂，有兴趣的读者可以参阅数据库管理系统方面的书籍以及联机文档或者《数据库原理实践(SQL Server 2012)》[3]。

下面通过一些例子说明 ALTER TABLE 语句的使用。

例 5.5 为教师表增加电话列，同时指定数据类型、默认值和 CHECK 约束。

```
ALTER TABLE 教师 ADD 电话 CHAR(8) DEFAULT 'no list;'
    CHECK (电话 LIKE '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]' )
```

这里的默认值虽然不具体，只是“no list”，但是可以避免空值参与运算。

例 5.6 为授课表增加“周序”列，指定数据类型为 CHAR(6)，且取值必须非空。

```
ALTER TABLE 授课 ADD 周序 CHAR(6) NOT NULL
```

例 5.7 修改授课表中“周几”列的数据类型为 NCHAR(3)。

```
ALTER TABLE 授课 ALTER COLUMN 周几 NCHAR(3) NOT NULL
```

例 5.8 删除授课表中的“周几”列。

```
ALTER TABLE 授课 DROP COLUMN 周几
```

5.3.4 删除基本表

基本表不再使用或者以前的设计有错误时，需要删除基本表。DROP TABLE 语句用来删除基本表。注意，由于外码参照另一表的主码，因此应先删除有外码定义的表。一个 DROP TABLE 语句可以删除多个表，表名之间用逗号分隔，如例 5.9 所示。

例 5.9 删除选课表与授课表。

```
DROP TABLE 选课,授课
```

5.4 数 据 更 新

创建基本表后需要向表中插入数据，在基本表的使用过程中需要对表中的数据进行删除和修改，这几种操作简称为对数据的增删改操作。对基本表中数据的增删改称为数据的更新。本节的例子使用教学信息管理数据库说明如何对数据进行增删改操作。

5.4.1 数据约束与数据更新

数据约束用来保证数据库中的数据的合理性与完整性。具体来说，数据库管理系统

要通过主码约束维护实体完整性，通过外码约束维护参照完整性，通过CHECK约束、默认值、非空约束等来维护用户定义完整性。因此，当需要对数据更新操作加以控制的时候，要在建表时定义相应的数据约束。

1. 主码约束与数据更新

前面讲过，建立基本表的时候必须定义主码约束。数据库管理系统（如SQL Server）在数据更新时自动检测主码约束。插入数据的时候，码中的属性不能为空值，码不能与表中已经存在的行的码值重复，否则插入失败。例如，插入新的学生记录，学号不能为空，也不能与前面已经插入的学生记录的学号相同。修改数据的时候，不能将码中的属性设置为空值，不能改为表中其他行已经使用的值。通常，主码不能修改，如果以前录入有误，应先删除记录，再用正确的主码值重新插入记录。

2. 外码约束与数据更新

对于存在外码的表，在建立基本表的时候定义相应的外码约束。数据库管理系统（如SQL Server）在数据更新时自动检测外码约束。可以将外码所在的表称为外码表，其所参照的表称为主码表。插入外码表数据时要求非空的外码值在主码表有对应的主码值，否则插入失败。例如，插入选课记录中课号为0903，而课程表中没有这样一个课号的课程与之对应，则系统拒绝插入。修改外码表数据时要求修改后非空的外码值在主码表有对应的主码值，否则修改失败。例如，课程表中的课号取值有0981、0982、0983、0984、0985、0986和0987，在修改选课表中的记录时，不能将选课记录中的课号修改为课程表中不存在的课号，如0988。删除主码表的数据时，要求外码表没有参照此主码的记录，否则删除失败。例如，存在选修0123号课程的选修记录，不能删除课程表中课号为0123的课程记录。

注意：外码约束用于保障参照完整性，这里参照完整性的维护策略为默认策略，没有使用级联策略与置空策略，相关的内容可以参看6.3节。

3. CHECK约束与数据更新

表达用户要求的CHECK约束也要在建立基本表的时候定义。数据库管理系统在数据更新时自动检测CHECK约束。插入数据时，若不满足CHECK约束，插入失败。例如，教师表输入数据时，职称的取值不是CHECK约束限定的“讲师”“副教授”“教授”之一，则新的行不能插入表中。同理，修改数据时也检测CHECK约束，不满足CHECK约束则修改失败。

注意：删除数据时，不检测CHECK约束，因此，试图用CHECK约束取代外码约束是行不通的。

4. 非空约束与数据更新

插入和修改记录时，非空列的取值不得为空，否则操作被数据库管理系统禁止。

5. 默认值与数据插入

插入记录的时候，如果含有默认值的列没有提供新的值，数据库管理系统将自动填入默认值。假设学生表有“国籍”列，默认值设置为“中国”，那么仅仅外籍学生的记录才需要给定具体的国籍，大多数中国学生的记录不必提供国籍信息，系统会自动填入“中国”。

5.4.2 INSERT 语句

SQL 中使用 INSERT 语句插入数据行到基本表中。INSERT 语句的基本句法如下：

```
INSERT 表名 | 视图名 [ ( 列名列表 ) ]
    VALUES ( DEFAULT | NULL |表达式 [ ,…n ] )
    | SELECT 语句
```

解释：

(1) 列名列表。插入数据时给定具体数据的列名列表，列名之间用逗号进行分隔。若包含表的所有列，且列的顺序与表定义相同。可以省略列名列表。如果提供的数值与表中各列的顺序不相同，或者未包含表中所有列的值，则必须使用列名列表显式指定这些列。如果某列不在列名列表中，列定义为可空列，自动使用空值；列定义含有默认值，自动使用默认值。

(2) VALUES 子句。引入要插入的数据值的列表。对于表中的每个列或已经由列名列表指定的若干列，都必须有一个数据值。必须用圆括号将数据值列表括起来。

(3) 有关针对视图使用 INSERT 语句的说明，参看 6.1 节。

下面给出几个 INSERT 语句的例子。

例 5.10 向课程表插入课程“高等数学”的记录。

```
INSERT INTO 课程(课号,课名,学分,类别) VALUES('1001','高等数学',3,'必修')
```

此例含有课程表完整的列名列表。由于插入数据的顺序与表中列的顺序相同，因此插入语句也可以省略列名列表。下面是等价的语句：

```
INSERT INTO 课程 VALUES('1001','高等数学',3,'必修')
```

例 5.11 向授课表插入一个新记录。

```
INSERT INTO 授课(职工号,班号,课号) VALUES('2300','3008', '102101')
```

此例不含表的可空列，因而没有包含所有列，列的次序也与表的定义不同，因此必须给出列名列表。

例 5.12 插入课程“大学英语”的记录，其中包含带有默认值的列。

```
INSERT INTO 课程 VALUES('1003','大学英语',3,DEFAULT)
```

此行数据中的“类别”列自动填入默认值“必修”。

例 5.13 从课程表中选取“课号”“课名”两列填入另一个名为课程_1 的表。假定课程_1 已经建立，且仅有“课号”“课名”两列，对应的数据类型与课程表相同。

```
INSERT INTO 课程_1 SELECT 课号,课名 FROM 课程
```

此例使用查询子句，从一个表中检索出相应的记录后向另一个表中插入数据。使用旧表的数据构造新表的时候，常常使用这一方法。

5.4.3 DELETE 语句

当需要将表中的一些数据行删除时，使用 DELETE 语句。DELETE 语句的基本句

法如下：

```
DELETE 表名 | 视图名
    [ WHERE <搜索条件>]
```

解释：

(1) 删除语句从指定的表或视图删除数据行。

(2) WHERE 子句给出删除数据行的条件，用于确定哪些数据行可以删除。省略WHERE 子句时，删除表或视图中的所有行。

(3) 删除操作只有在满足数据约束的前提下才会执行。

(4) 有关针对视图使用 DELETE 语句的详细说明，参看 6.1 节。

下面给出从表中删除数据行的几个例子。

例 5.14 删除选课表中学号为 10250104 且课号为 2006 的行。

```
DELETE FROM 选课表 WHERE 学号='10250104' AND 课号='2006'
```

例 5.15 删除选课表中学号为 09080324 且成绩为空的行。

```
DELETE FROM 选课表 WHERE 学号='09080424' AND 成绩 IS NULL
```

例 5.16 删除授课表中“数据库原理”课程的所有记录。

```
DELETE FROM 授课表 WHERE 课号 IN (SELECT 课号 FROM 课程表 WHERE 课名='数据库原理')
```

例 5.17 删除授课表中的所有行。

```
DELETE FROM 授课表
```

此例没有给出 WHERE 子句，故删除表中的所有行。

5.4.4 UPDATE 语句

当需要更新表中某一数据项的具体值时，使用 UPDATE 语句。UPDATE 语句的基本句法如下：

```
UPDATE 表名|视图名
    SET 列名 =表达式 | DEFAULT | NULL[ ,…n ]
[ WHERE <搜索条件>]
```

解释：

(1) SET 子句指定要更新的列及修改后的数值。

(2) 列名指定要更改数据的具体列的名字。

(3) 表达式指返回单个值的变量、文字值、表达式或嵌套 SELECT 语句(加括号)。返回的值替换相应列的现有值，即列的新值。

(4) DEFAULT 指定用默认值替换列中的现有值。

(5) NULL 指定将该列更改为 NULL，该列应可为空列。

(6)WHERE 子句用于指定更新条件，确定要对哪些行进行修改。

修改数据的操作，只有在满足数据约束的前提下才会执行。有关针对视图使用