第3章 chapter 3

数据库SQL语言原理

SQL(Structured Query Language,结构化查询语言)是一种面向元数据的第4代语言,这里的元数据,是指数据库或数据仓库中的表名、字段名、索引名、视图名等,它们是组织数据的数据。虽然SQL字面含义是“查询语言”,但其功能却包括数据定义、查询、更新、视图和控制等许多内容。因此,它是一种通用的、功能极强的关系数据库编程语言,利用它可以在数据库服务器上编制出许多精彩的程序。正因为如此,它早已成为关系数据库的标准编程语言。关系数据库的功能、特点、优点,基本上都表现在SQL上。

SQL语言易懂易学易用,这是因为,数据库的难点不是数据库编程语言,而是数据库设计。即使如此,数据库程序员仍然要熟练地掌握SQL语言的编程原理与编程技术,因为它是数据库的基础。

3.1 SQL语言特点

与其他语言不同,第4代语言SQL具有如下特点。

1. 非过程化的第4代编程语言

SQL是非过程化的编程语言,它不要求用户指定对数据的存放方法,也不需要用户了解具体的数据存放方式与存储过路径,而是以记录集合作为操作对象,所有的SQL语句接受集合作为输入,返回集合作为输出,这种集合允许一条SQL语句的输出作为另一条SQL语句的输入,所以SQL语句可以嵌套,这使它具有极大的灵活性和强大的功能。在多数情况下,其他语言需要一大段程序才能实现的功能,而SQL只需要一个语句就可以实现。

非过程化的编程语言属于第四代语言,第四代语言的操作特点是只要提出“做什么”?而不需说明“怎么做”。由于“怎么做”全由系统自动完成,这不仅减轻了用户负担,而且提高了数据的独立性,使程序与数据彻底分离。

2. 一种语言多种使用方式

SOL语言是“自含式”语言,也是“嵌入式”语言。作为自含式语言,SQL不仅可以作为程序语言进行编程使用,而且可以作为命令使用,独立地用于联机交互操作,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。

SQL还可以作为嵌入式语言，嵌入到高级语言的程序中去，如C、C++、PowerBuilder、VB、VC、Delphi、ASP等。在两种不同的使用方式下，SQL的语法结构基本一致。SQL以统一的语法结构提供两种不同的使用环境，为应用程序的研制与开发带来了很大的灵活性与方便性。

SOL语言还可以通过ODBC、JDBC和ADO，来实现高级语言(C++、Java等)与数据库的互连，使得在高级语言中可以操纵数据库的查询与更新。

3. 结构简洁易学易用

SQL语言是面向元数据的语言，它集数据查询(data query)、数据定义(data definition)、数据更新(data update)和数据控制(data control)等数据库必需的基本功能为一身，充分体现了关系数据库的本质特点和巨大优势。

SQL语言不但功能极强，而且设计构思非常巧妙，语言结构简洁明快，语句非常接近英语语句，特别易学易用。

4. 应用领域前途无量

(1) 由于各个数据库厂家积极推出各自支持SQL的软件或接口软件，使得各种类型的计算机和DBS都采用SQL，作为共同的数据存取语言和标准接口。这样，数据库之间的交互操作就有了共同基础，不同数据库就有可能连接成一个统一的整体。

(2) SQL成为国际标准，使得SQL的影响已经超出了数据库领域本身，在计算机技术应用的其他领域，也受到了重视和采用。不少软件产品将SQL的数据查询功能与多媒体图形功能、软件工程工具、软件开发工具和人工智能程序结合起来，使得SQL在这些数据库以外的领域中，显示出了相当大的潜力。

(3) 近年来，随着Internet技术的迅猛发展和快速普及，人们在HTML和XML中加入SQL语句，通过WWW来访问数据库，使得SQL有了进一步的发展。

5. SQL语言由三部分语句组成

具体来说，SQL由下述三个基本部分组成。

(1) 数据定义语言(Data Definition Language，DDL)。DDL用来创建数据库中的各类对象，其中包括：

① SQL模式(数据库)的创建、撤销与更改；

② 基本表的创建、撤销与更改；

③ 索引的创建与撤销；

④ 域、触发器和自定义类型的创建与撤销。

(2) 数据操纵语言DML(Data Manipulation Language)。DML用来查询和更新数据库中的数据，其中包括：

① 数据查询。例如，单表查询、多表查询，其中包括连接查询和嵌套查询。

② 数据更新。例如，数据插入、删除和修改。

③ 查询所需的附加功能。例如，求和函数SUM、平均函数AVG、元组个数求和函数

COUNT。最大函数 MAX 和最小函数 MIN 等。

(3) 数据控制语言(Data Control Language,DCL)。数据控制语是用来授予或收回访问数据库的某些权限,控制数据操纵事务的发生时间及效果,对数据库进行监视等。由 DBMS 提供的统一数据控制功能,是数据库系统的主要特征之一。数据控制主要包括如下两个部分:

① 数据库保护。例如,数据库的安全性和完整性保护。

② 事务管理。例如,数据库故障恢复和并发事务处理。

6. SQL 的语句类型

SQL 的语句类型按照其组成和功能,可分为如下所述的三种形式。

(1) 模式语句。模式语句功能是创建、更新和撤销模式及其对象。基本语句为 CREATE TABLE、CREATE VIEW、CREATE DOMAIN、CREATE TRIGGER、CREATE TYPE 等。

(2) 数据语句。数据语句功能是完成数据库的查询与更新操作,其主要语句为:SELECT(查询)、INSERT(插入)、UPDATE(修改)和 DELETE(删除)。

(3) 事务与控制语句。该语句功能为完成数据库的授权、事务管理、控制 SQL 语句集合的运行,其基本语句为:GRANT、START TRANSATION、REVOKE、COMMIT、ROLLBACK 等。

7. SQL 语言的不足之处

任何东西都不是十全十美的。由于 SQL 是一种面向元数据的语言,所以 SQL 是一种高层次的非过程化的第四代语言,与低层次的面向过程或面向对象的第二代或第三代语言不同。正因为如此,SQL 只能面向非过程编程与操作,缺少流程控制能力,不能面向过程或面向对象编程与操作,所以 SQL 不能编写底层程序,不能实现图形或窗口界面,这就是 SQL 的缺点或不足之处。

3.2 数据库定义语句

为了实现关系数据库系统支持"模式、外模式、内模式"的三级模式体系,SQL 的数据库定义语句必须包括模式定义、表定义、视图定义和索引定义四个部分。因为 SQL 不提供模式定义之后的模式修改语句,所以这里指的数据库定义包括创建模式、删除模式和重新定义模式 3 个部分。

3.2.1 经典示例数据库

因为本章所有的例题,均建立在学生选课系统这个经典示例数据库上,所以首先来研究这个数据库:看它包括哪些基本表,有什么优点,有什么不足。

【例 3-1】 创建学生选课系统的经典示例数据库。本章所有的 SQL 程序,都是对这

个数据库的操作。该数据库只有三个基本表，它们是：

(1) 学生基本表 Student(Sno,Sname,Ssex,Sage,Sdept)；

(2) 课程基本表 Course(Cno,Cname,Cpno,Ccredit)；

(3) 选课基本表 SC(Sno,Cno,Grade)。

这三个基本表，如表 3-1 至表 3-3 所示。

表 3-1 学生基本表 Student

学号(Sno)	姓名(Sname)	性别(Ssex)	年龄(Sage)	系别(Sdept)
200215121	李勇	男	20	CS
200215122	刘晨	女	19	CS
200215123	王敏	女	18	MA
200215125	张立	男	19	IS

表 3-2 课程基本表 Course

课程编号(Cno)	课程名称(Cname)	先修课号(Cpno)	学分(Ccredit)
1	数据库	5	4
2	数学		2
3	信息系统	1	4
4	操作系统	6	3
5	数据结构	7	4
6	数据处理		2
7	C++ 语言	6	4

表 3-3 选课基本表 SC

学号(Sno)	课程编号(Cno)	成绩(Grade)
200215121	1	92
200215121	2	85
200215121	3	88
200215122	2	90
200215122	3	80

由于该示例数据库具有科学性、特殊性、实用性，所以它已成为国内外许多数据库教材中的经典示例数据库。说它具有科学性，是因为它不但反映了“学生”与“课程”之间是一个多对多联系，而且已经将这一个多对多联系化解为了两个 1 对多联系，如图 3-1 所示；说它具有特殊性，是因为它发生在高校大学生选课这种特殊环境中；说它具有实用性，是因为它非常适合大学生的需求，大学生感到非常亲切。

但是，由于历史的原因，这个经典示例数据库存在两个不足之处：

(1) 学生基本表中，有一个“年龄”列名，这个不很好，应该改为“出生日期”，或改为“身份证号”。

(2) 既然是经典示例数据库，在一定条件下，应该讲是完美无缺的。这里的“一定条件”，是指在学生选课这种小的应用环境下。如果应用环境扩大到整个学校的教学组织

结构及其教学组织管理,而且要重视与发挥系的组织管理作用,那么系就成为一个单独的实体,它有独立的业务活动与业务特性,该实体有主键"系编号",还有属性"系名称、系主任、办公地址、联系电话"等。这样,在学生基本表中,属性"系别"就要改为"系编号",表明系与学生之间是1对多联系。

先不管这些不足之处,反正是业界流行的示例数据库,不是信息系统中的正式运行的数据库。

在CASE工具PowerDesigner中,概念数据模型CDM表示实体之间的联系,物理数据模型PDM表示基本表之间的联系。若用PowerDesigner来设计该示例数据库的物理数据模型PDM,则这三个基本表及其联系如图3-1所示。图3-1是示例数据库"型",其"值"是三个表中的记录。

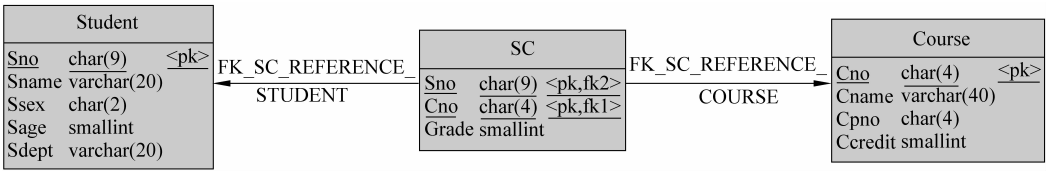

图3-1 学生选课示例数据库的物理数据模型PDM

由图3-1可见,学生基本表与选课基本表之间是1对多联系,课程基本表与选课基本表之间也是1对多联系。也就是说,每位学生可多次选课,每门课程可被多个学生选修。

图3-1在细节上有一个重大优点,它将Sname、Sdept和Cname三个字段的数据类型定义为可变字符长的数据类型varchar,这就节省了存储空间。

所谓可变字符长的数据类型varchar,就是在其规定的长度范围内,例如可变字符长的数据类型varchar(20),其长度范围是20个字符,那么在1至20个字符长度的范围内是可变的。反之,如果字符的数据类型是char(20),那么就是固定的、不可变的。

也许读者感到设计图3-1很难,其实只要你学会了PowerDesigner对话框中工具模板Palette的用法,那么用PowerDesigner设计概念数据模型CDM或物理数据模型PDM,都是一件轻松愉快的工作。有了这个PDM之后,再用SQL的创建基本表语句来定义基本表,将是一件很简单的工作。

到现在为止,已经知道:

(1)"实体、关系、基本表"其实就是同一个东西在三种不同场合下的三种不同称呼而已。

(2)表名和字段名就是元数据,它们构成了的关系的"型",或者说它们构成了基本表的"型",或者说它们构成了实体的"型"。而基本表中的记录,则构成了关系的"值",或者说它们构成了关系的元组,或者说它们构成了实体中的实例。

(3)二维表可以表示"实体、关系、基本表"这三个东西,概念数据模型CDM或物理数据模型PDM,也可以表示"实体、关系、基本表"这三个东西,而且表示得更好。

3.2.2 模式定义语句

1. 模式定义语句

在SQL中,一个关系数据库模式简称为SQL模式或简称模式,其意义在于表示为“基本表”、“视图”等的集合。

SQL模式由CREATE语句定义,该语句的一般格式如下:

```
CREATE SCHEMA<模式名>AUTHORIZATION<用户名>
    <CREATE DOMAIN>子句<CREATE TABLE>子句<CREATE VIEW>子句;
```

例如,教学数据库的SQL模式定义如下:

```
CREATE SCHEMA JIAOXUE AUTHORIZATION JOHN;
```

上述语句表明,模式名称为JIAOXUE,拥有者为JOHN。

在定义时如果没有指定<模式名>,那么<模式名>就是<用户名>。

要创建模式,该用户必须拥有DBA权限,或者获得了DAB授予的CREATE SCHEMA权限。

2. 定义模式的作用

定义了模式,实际上是定义了一个模式命名的存储空间,以后在该空间中可以进一步定义该模式包含的基本表、视图、索引等数据库对象,在此空间中的全体对象构成了对应的SQL数据库。

在不同关系数据库平台上,完成数据库模式定义的方法可能有所不同,有些需要用户书写定义语句;有些则通过可视化界面由用户直接填写数据库名称,而不必书写定义语句;还有一些两者兼有:既提供用户书写定义语句的方式,又提供通过可视化界面由用户直接填写数据库名称的方式。

3. 模式的删除与再定义

在SQL语言中,模式定义语句的使用,反映对数据概念的把握,在学习中应当予以足够重视。当一个SQL模式不需要时,可以用DROP语句予以撤销。

DROP语句的使用格式如下:

```
DROP SCHEMA <模式名>;
```

撤销之后,还可以重新定义,这就达到了修改模式的目的。

3.2.3 表定义语句

表是具体存储数据的地方,在数据库设计者的头脑中,数据库中的表按其功能不同,应该划分为基本表、代码表、查询表和临时表四种,其中最重要的是基本表。SQL不管这些不同的概念,它将这四种不同性质的表,提供统一的表定义语句。

存放原始数据的表,称为基本表。每一个信息系统,都有多个信息源,信息源产生的数据,称为原始数据。这些原始数据,都要录入到基本表中。有了基本表中的原始数据,信息系统的其他数据,都可以派生出来。所以基本表的定义、删除及修改特别重要。

1. 基本表创建

SQL 使用 CREATE TABLE 语句创建基本表,该语句的一般格式如下:

```
CREATE TABLE[模式名]<基本表名>
    (<列名><数据类型><列级完整性约束条件>
    [,<列名><数据类型><列级完整性约束条件>]
      ⋮
    [,<表级完整性约束条件>]);
```

其中<基本表名>是所要定义的基本表名称,[]内的内容是可选项。

创建基本表的同时,通常还需要定义与该表相关的完整性约束条件,这些完整性约束(实体完整性约束、参照完整性约束和用户定义完整性约束)被存储在 DBMS 的数据字典中,当用户操作基本表中数据时,由 DBMS 自动检查该操作是否违反预先设定的这些完整性约束条件。

如果完整性约束涉及该基本表的多个列,则必须将其定义在表级。否则,可以定义在列级。

【例 3-2】 用 CREATE TABLE 语句,同时创建示例数据库的三个基本表:

```
CREATE TABLE Student
    (Sno CHAR(9) PRIMARY KEY,                    /*列级完整性约束,定义主键*/
    Sname VARCHAR(20) NOT NULL,                  /*列级完整性约束,名字非空*/
    Sage SMALLINT CHECK(Sage BETWEEN 12 AND 55),
                                          /*列级完整性约束,年龄在12到55之间*/
    Ssex CHAR(2),
    Sdept VARCHAR(20)
    );
CREATE TABLE Course
    (Cno CHAR(4) PRIMARY KEY,                    /*列级完整性约束,定义主键*/
    Cname VARCHAR(40) NOT NULL,                  /*列级完整性约束,名字非空*/
    Cpno CHAR(4),                                /*列级完整性约束,定义选修课程号*/
    Ccredit SMALLINT
    FOREIGN KEY(Cpno)REFERENCE Course (Cno),   /*表级完整性约束,定义外键Cpno*/
    );
/*上述表级完整性约束,定义外键Cpno,表明参照表与被参照表可以是同一个表Course*/
CREATE TABLE SC
    (Sno CHAR(9) NOT NULL,                       /*列级完整性约束,键字段必须非空*/
    Cno CHAR(4) NOT NULL,                        /*列级完整性约束,键字段必须非空*/
    Grade SMALLINT,
    PRIMARY KEY(Sno,Cno),                        /*表级完整性约束,定义组合主键*/
```

```
FOREIGN KEY (Sno) REFERENCE Student (Sno),    /*表级完整性约束,定义外键 Sno*/
FOREIGN KEY (Cno) REFERENCE Course (Cno)      /*表级完整性约束,定义外键 Cno*/
);
```

由上述语句可以知道：

(1) 基本表的定义，就是说明属性列名称和属性列类型，并且可以指明主键和多个外键。如用“PRIMARY KEY(Sno)”定义表 Student 的主键，用保留字 FOREIGN KEY 指明外键，用保留字 REFERENCE 指出外键来自的表名，即主表(被参照表)。

(2) 在 SQL 中，域的概念是通过定义属性列的数据类型来实现的。常用的数据类型，如 3-4 所示。在 PowerDesigner 中，各种数据类型全部列出，任你点击挑选。

表 3-4 常用的数据类型分类

数据类型分类	数据类型表示方法
整数数据类型	INT、SMALLINT
浮点数据类型	REAL、FLOAT
二进制数据类型	BINARY、VARBINARY
逻辑数据类型	BIT
字符数据类型	CHAR、VARCHAR
文本和图形数据类型	TEXT、IMAGE
日期和时间数据类型	DATE、DATETIME、SMALLDATETIME
货币数据类型	MONEY、SMALLMONEY

(3) 在基本表定义时，可以使用 CHECK 语句，说明各列中值应当满足的条件，例如：基本表 Student 定义中，要求学生年龄应当在 12～55 岁之间。

(4) 还可以用 UNIQUE 语句定义唯一性属性列。

(5) 建立基本表时，系统会同时在数据库的数据字典中增加该基本表有关的定义。数据库管理系统，就是靠数据字典中的定义，来管理基本表的。

2. 基本表更新

SQL 使用 ALTER TABLE 语句进行基本表结构更新。基本表结构变动包括增加新属性列、删除原有属性列、修改数据类型、补充定义主键和删除主键等。在进行基本表更新时，如果是新增属性列，需要新属性列一律为空值；如果是修改原有属性列，则要注意是否可能破坏已有数据。

1) 增加属性列

增加新的属性列使用“ALTER…ADD…”语句，基本格式如下：

```
ALTER TABLE<基本表名>
ADD<新列名><数据类型>[完整性约束条件];
```

其中，＜基本表名＞是要更新的基本表名称，ADD 子句用于增加新列和新的完整性约束条件。

【例 3-3】 在基本表 Student 中添加一个新的地址属性 ADDRESS。

```
ALTER TABLE Student
ADD ADDRESS VARCHAR(30);
```

说明：新添加属性列时不允许出现 NOT NULL。基本表在增加一列后，原有元组在新增加的列上的值都定义为空值(NULL)。

2) 删除属性列

删除已有属性列使用“ALTER…DROP…”语句，其基本格式如下：

```
ALTER TABLE<基本表名>DROP<属性列名>;
```

【例 3-4】 在 Student 中删除属性列 Sage。

```
ALTER TABLE Student DROP Sage;
```

3) 修改属性列

修改已有属性的类型及宽度，使用“ALTER…MODIFY…”语句，其基本格式如下：

```
ALTER TABLE<基本表名>MODIFY<属性列名><类型>;
```

【例 3-5】 在 Student 中将 Sno 的长度修改为“6”。

```
ALTER TABLE Student MODIFY SNO CHAR(6);
```

4) 补充定义主键

在 SQL 中，并不要求每个表都定义主键，可以在需要情况下随时定义，这称为主键的补充定义。

补充定义主键的语句格式如下：

```
ALTER TABLE<表名>ADD PRIMARY KEY(<列名表>);
```

需要指出，被定义为主键的属性列，应当是非空和满足唯一性要求的。

【例 3-6】 设有全体男生的表 Smale，其结构与 Student 表相同，补充定义 Smale 的主键 SQL 的语句如下：

```
ALTER TEBLE Smale ADD PRIMARY KEY(Sno);
```

5) 删除主键

如果一个表无子孙(无从表)，那么该表可以不定义主键，所以可以从一个表中删除主键。

删除主键的 SQL 语句格式如下：

```
ALTER TABLE<表名>DROP PRIMARY KEY(<列名表>);
```

【例 3-7】 删除 Student 表中主键 Sno 的 SQL 语句如下：

```
ALTER TEBLE Student DROP PRIMARY KEY(Sno);
```

3. 删除基本表

SQL使用“DROP TABLE”语句撤销基本表,该语句的一般格式如下:

```
DROP TABLE<基本表名>;
```

【例3-8】 撤销基本表Student,但要求只有在没有视图或约束引用Student的属性列时才能撤销,否则拒绝撤销,则其实现语句为:

```
DROP TABLE Student;
```

删除基本表要慎之又慎。因为基本表被删除后,表中的记录、表上建立的索引、视图、存储过程和触发器,全部被删除了。对于“1对多”联系的参照关系表,若要删除被参照表(主表),则必须先删除参照表(子表),之后才能删除被参照表,否则,子表中的记录就成了“没妈的孩子是根草”。

3.2.4 索引定义语句

在定义基本表时还要定义索引。索引的作用是将数据库物理结构与逻辑结构联系在一起,以提高查询的速度,该功能仅限于查询时发挥作用。

1. 创建索引

SQL使用CREATE INDEX语句创建索引,该语句的一般格式如下:

```
CREATE[UNIQUE][CLUSTEREDED]
INDEX<索引名>ON <基本表名>(<列名>[<排列方式>][,<列名>[<排列方式>]]…);
```

其中:

(1) ＜基本表名＞是要为其创建索引的基本表的名称。

(2) 索引可以创建在该表的一列或者多列上,各列名之间用逗号分隔。每个＜列名＞后面还可以用＜排序方式＞来指定索引值是按照升序(ASC)或者按照降序(DESC)的方式排列,默认值为ASC。

(3) UNIQUE表明,此索引的每一索引值,只对应唯一的一个元组。

(4) CLUSTERED表示要创建的索引是聚簇索引。所谓聚簇索引是指索引项的顺序与表中元组的物理顺序一致的索引组织。

(5) 建立索引时,系统会同时在数据库的数据字典中增加该索引有关的定义。数据库管理系统,就是靠数据字典中的定义来管理索引的。

【例3-9】 在Student(Sno)上建立个按升序排列的索引S_XSNO。

```
CREATE UNIQUE INDEX S_XSNO ON Student(Sno);
```

【例3-10】 在SC上建立按SC(Sno,Cno)升序排列名为SC_XSC的索引。

```
CREATE INDEX SC_XSC ON SC(Sno,Cno);
```