第5章

SQL

SQL是使用数据库时最常用的语言。它似乎充满了魔力，因其用途广泛而令使用者在“数据的海洋”中有一种“通行无阻”的感受。利用SQL，人们可以进行数据库的定义，进行数据库数据的操纵，还可以对数据库中的数据进行查询，更可以利用SQL不断挖掘、发现数据的价值，从最基本的数据操纵中，将平常的“数据”转变为对未来有指导意义的“洞见”信息，彰显数据的魅力。

本章介绍SQL的特征，以及通用的语法结构。

5-1

5.1 SQL概述

SQL作为一种融数据库查询和程序设计功能于一体的语言，在实践中成为用于存取数据以及查询、更新和管理控制关系数据库系统的专门语言。从最早的版本发展至今，有许多数据库产品都支持SQL，它已经很明显地确立了作为标准关系数据库语言的地位。

5.1.1 SQL的特点

SQL(Structured Query Language)是一种结构化查询语言，同时也是高级的非过程化编程语言。除了数据查询，SQL还具有很多其他功能，如定义数据结构，维护数据库中的数据，以及定义安全性约束等。它具有如下特点。

1. 语言功能的一体化

SQL集数据操纵、数据定义和数据控制功能于一体，语言风格统一，可以独立完成数据库生命周期的全部活动。其中：数据操纵语言(DML)用于对数据库中的数据进行插入、删除、修改等数据维护操作和进行查询、统计、分组、排序等数据处理操作；数据定义语言(DDL)用于定义关系数据库模式(外模式和内模式)；数据控制语言(DCL)用于实现对基本表和视图的授权，以及实现对完整性规则的描述、事务控制等操作。

2. 非过程化

SQL 是一种高度非过程化的语言。在采用 SQL 进行数据操作时，只要提出“做什么”，无须指明“怎么做”，其他工作由系统完成。因为用户无须了解存取路径的结构，存取路径的选择，以及相应操作语句的操作过程，所以大大减轻了用户负担，并有利于提高数据独立性。

3. 采用面向集合的操作方式

SQL 采用面向集合的操作方式，用户只要使用一条操作命令，其操作对象和操作结果都可以是行的集合。无论是查询操作，还是插入、删除、更新操作的对象，都可实现面向行集合的操作方式。

4. 一种语法结构和两种使用方式

SQL 具有一种语法结构和两种使用方式。既是自含式语言，又是嵌入式语言。①自含式 SQL：能够独立地进行联机交互，用户只需在终端键盘上直接输入 SQL 命令就可以对数据库进行操作；②嵌入式 SQL：能够嵌入高级语言的程序中，用来实现对数据库的操作。由于在自含式 SQL 和嵌入式 SQL 不同的使用方式中，SQL 的语法结构基本上一致，因此为程序员设计应用程序提供了很大的方便。

5. 语言结构简洁

尽管 SQL 功能极强，且有两种使用方式，但由于设计构思巧妙，语言结构简洁明了，完成数据操纵、数据定义和数据控制功能只用 9 个动词，易学、易用。

- 数据操纵：Select，Insert，Update，Delete；
- 数据定义：Create，Alter，Drop；
- 数据控制：Grant，Revoke。

6. 支持三级模式结构

SQL 支持关系数据库三级模式结构。其中：视图和部分基本表对应的是外模式，全体表结构对应的是模式，数据库的存储文件和它们的索引文件构成关系数据库的内模式。

5.1.2　SQL 的功能

SQL 具有丰富的功能，按功能分类，可将其分为如下几类。

(1) 数据定义：用来定义关系数据库的模式、外模式和内模式，以实现对基本表、

视图以及索引文件的定义，也可以实现模式的修改和删除等操作。

（2）数据操纵：提供了数据查询和数据维护两类功能。

- 数据查询：实现对数据库中的数据查询、统计、分组、排序等操作；
- 数据维护：实现数据的插入、删除、更新等数据维护等操作。

（3）数据控制：数据控制包括对基本表和视图的授权，完整性规则定义和更新的描述，以及事务控制等。

（4）系统存储过程：系统存储过程是 DBMS 专门创建的存储过程，用于用户方便地从系统表中查询信息，或者完成与更新数据库表相关的管理任务，或其他系统管理任务。

5.2 数据定义

数据定义的 SQL 语句（详见表 5-1）不仅可以实现数据库的模式定义，也可实现对基本表、视图以及索引文件的定义，以及对定义的基本表、视图以及索引文件进行修改和删除。

表 5-1 数据定义的 SQL 语句

对　象	创　建	删　除	修　改
数据库	CREATE DATABASE	DROP DATABASE	
表	CREATE TABLE	DROP TABLE	ALTER TABLE
视图	CREATE VIEW	DROP VIEW	ALTER VIEW
索引	CREATE INDEX	DROP INDEX	

5.2.1 定义数据库

SQL 定义数据库的语句。

语句格式：

```
CREATE DATABASE < database_name >
```

功能：创建一个新数据库。

说明：< database_name >是所要定义的数据库的名字。

例 5-1 创建一个新数据库，命名为 MY_database。

SQL 命令如下：

```
CREATE DATABASE MY_database
```

在 GaussDB(for MySQL)管理控制平台中,执行 SQL 命令,操作结果如图 5-1 所示。

图 5-1　创建数据库(MY_database)

5.2.2　定义及维护数据库表

定义数据库表是数据库操作中最基本的操作。一个数据库是由多个数据库表构成的,当我们定义了数据库所有的表的结构之后,事实上就完成了数据库结构的定义。

1. SQL 定义表的语句

语句格式:

```
CREATE TABLE < table_name >
   ([< column1_name, data_type, column_Length >] [default] not null|null
     [,< column2_name > type [[default] not null|null]…
     [,UNIQUE(column_name [,column_name1]…)]
     [,PRIMARY KEY(column_name [,column_name]…)]
     [,FOREIGN KEY (column_name [,column_name]…)
     REFERENCES < Reference_ table_name >(column_name [,column_name]…)]
     [,CHECK (condition)] )
```

功能:创建一个数据库表。

几点说明:

(1) < table_name >:所要定义的数据库表的名字;

(2) < column_name,data_type,column_Length >:组成该表的各个属性(字段)的名称、类型和长度。有关数据类型及长度详见 4.1 节;

(3) not null|null:涉及相应属性字段的完整性约束条件;

(4) 在表级约束有如下 6 种约束:

① DEFAULT：默认值约束；
② UNIQUE：唯一性约束；
③ PRIMARY KEY：主键约束；
④ FOREIGN KEY：外键约束；
⑤ REFERENCES：参照完整性约束；
⑥ CHECK：检查约束。

例 5-2 设计一个数据库表，其结构定义如表 5-2 所示。

表 5-2 School 表结构

字 段 名	字 段 别 名	字 段 类 型	字段长度(字节)	索 引	备 注
School_id	学院编号	char	1	有(无重复)	主键
School_name	学院名称	char	10	—	—
School_dean	院长姓名	char	6	—	—
School_tel	电话	char	13	—	—
School_addr	地址	char	10	—	—

在已有的数据库(MY_database)中，创建一个数据库表(MY_school)。
SQL 命令如下：

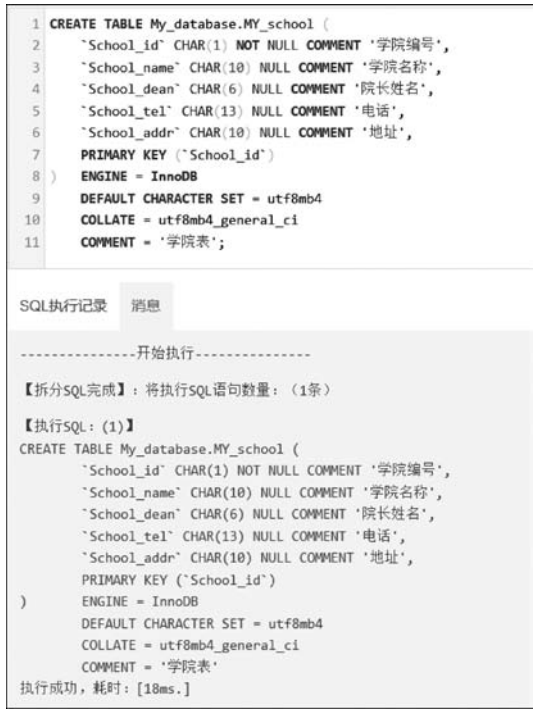
```
CREATE TABLE MY_database.MY_school (
    `School_id` CHAR(1) NOT NULL COMMENT '学院编号',
    `School_name` CHAR(10) NULL COMMENT '学院名称',
    `School_dean` CHAR(6) NULL COMMENT '院长姓名',
    `School_tel` CHAR(13) NULL COMMENT '电话',
    `School_addr` CHAR(10) NULL COMMENT '地址',
    PRIMARY KEY (`School_id`)
)   ENGINE = InnoDB
    DEFAULT CHARACTER SET = utf8mb4
    COLLATE = utf8mb4_general_ci
    COMMENT = '学院表';
```

在 GaussDB(for MySQL)管理控制平台中，执行 SQL 命令，操作结果如图 5-2 所示。

2. SQL 修改表结构的语句

语句格式为：

```
ALTER TABLE < table_name >
    [ ADD < column_name > < type > [ REFERENCES < Reference_table_name >(column_name
[,column_name]…)] ] ]
    [ DROP REFERENCES < Reference_table_name >]
```



```
CREATE TABLE My_database.MY_school (
    `School_id` CHAR(1) NOT NULL COMMENT '学院编号',
    `School_name` CHAR(10) NULL COMMENT '学院名称',
    `School_dean` CHAR(6) NULL COMMENT '院长姓名',
    `School_tel` CHAR(13) NULL COMMENT '电话',
    `School_addr` CHAR(10) NULL COMMENT '地址',
    PRIMARY KEY (`School_id`)
)   ENGINE = InnoDB
    DEFAULT CHARACTER SET = utf8mb4
    COLLATE = utf8mb4_general_ci
    COMMENT = '学院表';
```

SQL执行记录　消息

```
----------开始执行----------
【拆分SQL完成】：将执行SQL语句数量：（1条）
【执行SQL：(1)】
CREATE TABLE My_database.MY_school (
        `School_id` CHAR(1) NOT NULL COMMENT '学院编号',
        `School_name` CHAR(10) NULL COMMENT '学院名称',
        `School_dean` CHAR(6) NULL COMMENT '院长姓名',
        `School_tel` CHAR(13) NULL COMMENT '电话',
        `School_addr` CHAR(10) NULL COMMENT '地址',
        PRIMARY KEY (`School_id`)
)       ENGINE = InnoDB
        DEFAULT CHARACTER SET = utf8mb4
        COLLATE = utf8mb4_general_ci
        COMMENT = '学院表'
执行成功，耗时：[18ms.]
```

图 5-2　创建学院表(MY_school)

[MODIFY COLUMN < column_name > < type > [REFERENCES < Reference_table_name >(column_name [,column_name]…)]]]

功能：修改表结构。

几点说明：

(1) < table_name >：要修改的数据库表；

(2) ADD 子句：增加新字段，以及新的完整性约束条件；

(3) DROP 子句：删除指定的字段及完整性约束条件；

(4) MODIFY 子句：修改指定字段，以及完整性约束条件。

例 5-3　已知 School 表的结构定义见表 5-2，请增加一个新的字段(字段名为 School_brief，字段类型为 char(50))。

SQL 命令如下：

```
ALTER TABLE MY_database.MY_school
ADD school_brief CHAR(50) NULL COMMENT '学校简介';
```

在 GaussDB(for MySQL)管理控制平台中，执行 SQL 命令，操作结果如图 5-3 所示。

图 5-3　修改表(MY_school)的结构

3. SQL 删除数据库表的语句

语句格式：

```
DROP TABLE [IF EXISTS] < database_name1 >,< database_name2 >, < database_name3 > …
```

功能：删除数据库表。

两点说明：

(1) < database_name1 >,< database_name2 >, < database_name3 > …表示要删除的表的名称，DROP TABLE 可以同时删除多个表，只要将表名依次写在后面，表名之间用逗号隔开即可。

(2) IF EXISTS 用于在删除表之前判断该表是否存在。如果不加 IF EXISTS，当数据库表不存在时将提示错误，中断 SQL 语句的执行；加上 IF EXISTS 后，当数据库表不存在时，SQL 语句可以顺利执行，但是会发出警告(warning)。

例 5-4　删除表(MY_school)，SQL 语句如下：

```
DROP TABLE MY_database.MY_school;
```

在 GaussDB(for MySQL)管理控制平台中，执行 SQL 命令，操作结果如图 5-4 所示。

5.2.3　定义视图

SQL 定义视图的语句格式如下：

图 5-4　删除表(MY_school)

```
CREATE VIEW view_name AS
    SELECT column_name(s)
    FROM table_name
    WHERE condition
```

功能：创建视图。

两点说明：

(1) view_name：指定视图的名称。该名称在数据库中必须是唯一的，不能与其他数据库表或视图重名；

(2) SELECT …FROM …WHERE …：指定创建视图的 SELECT 语句，可用于查询多个数据库表或源视图。

例 5-5　已知表(My_school)，创建单表视图(v_school)。

SQL 语句如下：

```
CREATE VIEW v_school
AS
SELECT school_id,school_name FROM MY_database.MY_school
```

在 GaussDB(for MySQL)管理控制平台中，执行 SQL 命令，操作结果如图 5-5 所示。

5.2.4　定义触发器

SQL 定义触发器的语句格式如下：

```
CREATE < Trigger_name > < BEFORE | AFTER >
< INSERT | UPDATE | DELETE >
ON < table_name > FOR EACH Row < Trigger body >
```

图 5-5　创建视图(v_school)

功能：创建触发器。

几点说明：

(1) < Trigger_name >：指定要创建的触发器名称；

(2) < BEFORE | AFTER >：触发器是在动作之前触发还是之后触发；

(3) < INSERT | UPDATE | DELETE >：要进行什么操作；

(4) EACH Row < Trigger body >：触发器触发检验的条件。

例 5-6　已知表(MY_class)和表(MY_student)，创建 INSERT 触发器(tri_studentInsert)，当向表(MY_student)插入学生数据时，则更新表(MY_class)的班级人数(student_num)字段。

SQL 语句如下：

```
DELIMITER $
CREATE trigger tri_studentInsert
AFTER INSERT
on MY_student for each row
begin
declare c int;
set c = (select count( * ) from MY_student where class_id = new.class_id);
update MY_class set student_sum = c + 1 where class_id = new.class_id;
end $
DELIMITER ;
```

在 GaussDB(for MySQL)管理控制平台中，执行 SQL 命令，操作结果如图 5-6 所示。

```
DELIMITER $
CREATE trigger tri_studentInsert
AFTER INSERT
on MY_student for each row
begin
declare c int;
set c = (select count(*) from MY_student where class_id=new.class_id);
update MY_class set student_sum = c+1 where class_id = new.class_id;
end$
DELIMITER ;
```

SQL执行记录　消息

```
----------开始执行----------
【拆分SQL完成】：将执行SQL语句数量：（1条）
【执行SQL：(1)】
CREATE trigger tri_studentInsert
AFTER INSERT
on MY_student for each row
begin
declare c int;
set c = (select count(*) from MY_student where class_id=new.class_id);
update MY_class set student_sum = c+1 where class_id = new.class_id;
end
执行成功，耗时：[9ms.]
```

图 5-6　创建 INSERT 触发器(tri_studentInsert)

5.3　数据操纵

5-2

数据操纵命令用于对表中的数据进行插入、删除、更新和查询等。数据操纵的 SQL 命令如表 5-3 所示。

表 5-3　数据操纵的 SQL 命令

数据操纵	命　令
插入	INSERT
更新	UPDATE
删除	DELETE

5.3.1　数据库表的数据插入

SQL 数据库表数据插入的语句格式如下：

```
INSERT
INTO < table_name >  (< column1_name >
[,< column2_name > … )]
VALUES (< value1 > [,< value2 >]  … )
```

功能：插入单个记录。

两点说明：

(1) INTO：指定要插入数据的表名及字段，字段的顺序可与表定义中的顺序不一致。没有指定字段则表示要插入的是一条完整的记录，且字段属性与表定义中的顺序一致；指定部分字段表示插入的记录在其余字段上取空值。

(2) VALUES：提供的值必须与 INTO 子句匹配(值的个数及类型)。

例 5-7 已知表(MY_school)的结构如表 5-4 所示，请插入学院“媒体与设计”的信息。

表 5-4 MY_school 表结构

学院编号	学院名称	院长姓名	电话	地址
A	计算机科学	沈存放	010-86782098	A-JSJ
B	电子信息与电气工程	张延俊	010-85764325	B-DZXDQG
C	生命科学	于博远	010-86907865	C-SMKJ
D	化学化工	杨晓宾	010-86878228	D-HXHG
E	数学科学	赵石磊	010-81243989	E-SXKX
F	物理与天文	曹朝阳	001-80758493	F-WLTW
H	媒体与设计	王佳佳	010-81794522	H-MTSJ

SQL 语句如下：

```
INSERT INTO MY_school(School_id,School_name,School_dean,School_tel,School_addr)
VALUES('H','媒体与设计','王佳佳','010-81794522','H-MTSJ');
```

在 GaussDB(for MySQL)管理控制平台中，执行 SQL 命令，操作结果如图 5-7 所示。

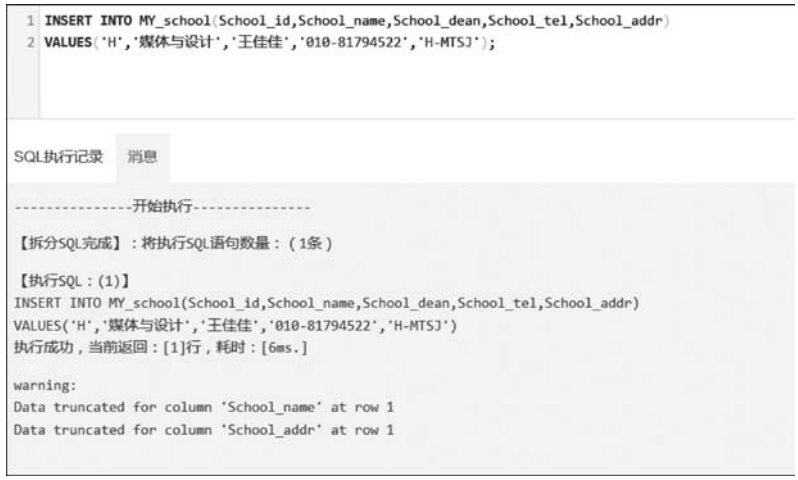


图 5-7 表(MY_school)的数据插入

5.3.2　数据库表的数据修改

SQL 数据库表数据修改的语句格式如下：

```
UPDATE < table_name >
SET < column_name1 > = < new_value 1 >
    [,< column_name2 > = < new_value 2 >] …
[WHERE column_name = some_value]
```

功能：更新指定表中满足 WHERE 子句条件字段的对应的数据。

几点说明：

(1) SET：指定修改方式、要修改的字段、修改后的取值；

(2) WHERE：指定要修改的字段，若默认表示要修改表中的所有字段；

(3) DBMS 在执行修改语句时，会检查修改操作是否破坏表中已定义的完整性规则。

例 5-8　已知表(MY_school)，修改“媒体与设计”学院的院长姓名改为“刘国栋”。

SQL 语句如下：

```
UPDATE MY_school SET School_dean = '刘国栋' WHERE School_id = 'H';
```

在 GaussDB(for MySQL)管理控制平台中，执行 SQL 命令，操作结果如图 5-8 所示。

图 5-8　修改数据库表(MY_school)中数据

5.3.3 数据库表的数据删除

SQL 数据库表数据删除语句的格式如下：

```
DELETE FROM < table_name >
[ WHERE < condition >]
```

功能：删除指定表中满足 WHERE 子句条件的记录。

两点说明：

(1) WHERE：指定要删除的记录应满足的条件，若默认表示要删除表中的所有记录；

(2) DBMS 在执行删除语句时会检查所删除记录是否破坏表中已定义的完整性规则。

例 5-9 已知表(MY_school)，删除“媒体与设计”学院这条数据。

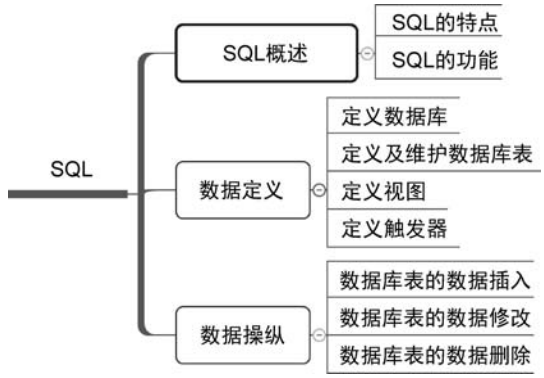
SQL 语句如下：

```
DELETE FROM MY_school
WHERE School_id = 'H';
```

在 GaussDB(for MySQL)管理控制平台中，执行 SQL 命令，操作结果如图 5-9 所示。

图 5-9 删除表(MY_school)中的数据

知识点树



思考题

（1）简述 SQL 的特点。
（2）简述 SQL 的功能。
（3）试述 SQL 语句能完成哪些操作。
（4）试述 SQL 有几类。
（5）试述 SQL 能定义哪些数据库对象。