# 第3章 关系数据库标准语言SQL

关系数据库的标准语言是结构化查询语言，简称 SQL(Structured Query Language)语言。它是一种通用的、功能强大的、集数据库的定义、操纵和控制于一体的关系数据库语言。目前几乎所有的关系数据库管理系统都支持 SQL 语言，即它作为国际化的标准语言广泛应用于关系数据库管理系统之中。本章将详细介绍 SQL 语言的查询、插入、删除、修改和控制等语句的语法及其使用，重点是查询语句中查询条件的组织和表示问题。

## 3.1 SQL 概述

SQL 语言是用于访问数据库的标准语言，也是关系数据库系统的管理与使用、数据库设计与编程等至关重要的数据库语言。为了学习好 SQL 语言，需要了解 SQL 语言的标准、组成及其特点。

### 3.1.1 SQL 简介

SQL 语言是于 1974 年由 Boyce 和 Chamberlin 提出的，IBM 公司在 1975 年至 1979 年间研制出著名的关系数据库管理系统原型 System R，并在该系统上实现了这种语言。1986 年 10 月美国国家标准局 ANSI 批准了 SQL 作为关系数据库语言的美国标准，同年发布了 SQL 标准文本(简称 SQL-86 标准)。1987 年此标准也获得了国际标准化组织 ISO 的认可，成为国际标准语言。此后 ANSI 不断修改和完善 SQL 标准，并于 1989 年发布了 SQL-89 标准，1992 年又发布了 SQL-92 标准(也称 SQL2)，1999 年发布了 SQL-99 标准(SQL3)，2003 年发布了 SQL2003 标准，2006 年发布了 SQL2006 标准，2008 年发布了 SQL2008 标准。从 SQL-99 到 SQL2008，可以看到标准修订的周期越来越短，反映了技术的需求变化非常快。

SQL 语言成为国际标准语言以后，随着数据库技术的发展不断发展。各个数据库厂商纷纷推出了自己的数据库系统软件或与 SQL 相关的接口软件。这使大多数数据库均用 SQL 作为共同的数据存取语言和标准接口，使不同数据库系统之间的相互操作有了共同的基础。SQL 语言已经成为数据库领域中的主流核心语言。

### 3.1.2 SQL 数据库结构

支持 SQL 的关系数据库管理系统同样支持关系数据库三级模式结构，如图 3.1 所示。

1）视图和部分基本表构成了关系数据库的外模式

图 3.1 中外模式对应于视图和部分基本表。视图是从一个或几个基本表导出的表。视图本身不独立存储在数据库中，即数据库中只存放视图的定义而不直接存放视图对应的数据。这些数据仍存放在与视图相关的基本表中。因此，视图可以称为虚表。

用户可用 SQL 对基本表和视图进行查询或其他操作，基本表和视图一样，都是关系。在数据查询时，SQL 对基本表和视图等同对待。

2）全体基本表构成了关系数据库的模式

基本表是本身独立存在的表，SQL 中一个关系对应一个基本表。一个或多个基本表对应一个存储文件，一个表可以带有若干索引，索引也存放在存储文件中。

3）数据库的存储文件构成了关系数据库的内模式

基本表对应的存储文件及索引文件等的逻辑结构组成了关系数据库的内模式。

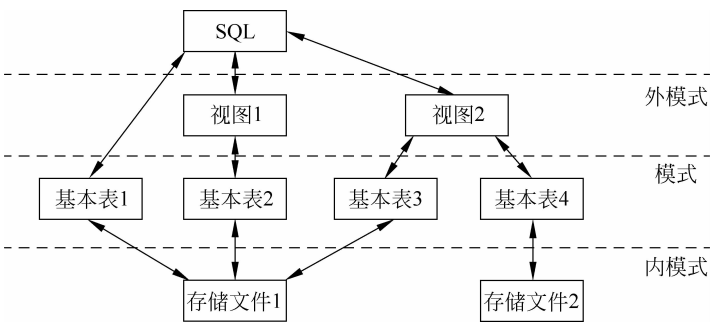


图 3.1　SQL 对关系数据库模式的支持

## 3.1.3　SQL 的组成及特点

### 1. SQL 的组成

SQL 语言是一个通用的、功能极强的关系数据库语言。SQL 的功能组成可以分为以下三类。

1）数据定义

SQL 的数据定义功能是通过 DDL（数据定义语言）来实现的，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、修改和删除等操作。

2）数据操纵

SQL 的数据操纵功能是通过 DML（数据操作语言）来实现的。DML 包括数据查询和数据更新两种数据操作语句。其中，数据查询语句是对数据库中的数据查询、统计、分组、排序、检索等操作，数据更新语句是数据的插入、删除和修改等操作。

3）数据控制

数据库的控制是指数据的安全性和完整性控制。SQL 的数据控制功能是通过 DCL（数据控制语言）来实现的。SQL 通过对数据库用户的授权和回收命令来实现数据的存取控制，以保证数据库的安全性。当然 SQL 还提供了数据完整性约束条件的定义和检查机制来

保障数据库的完整性。

### 2. SQL 的特点

SQL 之所以能够被用户和数据库行业所广泛接受，并成为国际标准，是因为它是一个综合的、功能极强的、简洁易学的数据库语言。SQL 语言具有以下特点。

1）综合统一

SQL 语言将数据定义语言 DDL、数据操纵语言 DML、数据控制语言 DCL 的功能集成于一体，语言风格统一，可独立完成数据库生命周期中的全部活动，主要包括：

- 定义关系模式，插入数据，建立数据库；
- 对数据库中的数据进行查询和更新；
- 数据库重构和维护；
- 数据库安全性、完整性控制，等等。

这些给数据库应用系统的开发提供了良好的数据环境。尤其是用户在数据库系统投入运行后，可以根据需求的变更而修改模式，但并不影响数据库的运行，使系统具有良好的可扩展性。

在关系模型中实体和实体之间的联系都使用关系来表示，这种数据结构的单一性带来了数据操作符号的统一性，数据的查找、插入、删除、更新等操作都只需要一种操作符号，从而简化了系统中的复杂多样化的数据信息的统一表示方式。

2）高度非过程化

SQL 语言进行数据操作，只要提出“做什么”的功能，而无须指明“怎么做”，也就是不必了解数据的存取路径。数据存取路径的选择以及 SQL 的操作过程由系统自动完成。这不仅大大减轻了用户负担，而且有利于提供数据独立性。

3）面向集合的操作方式

关系模型中实体和实体之间的联系都用关系表示，而对关系的操作特点是集合操作方式，即操作的对象和结果都是集合。关系数据库语言 SQL 也是采用集合操作方式，不仅操作对象是元组的集合，而且查询、插入、删除、修改等操作的结果也都是元组的集合。

4）统一的语法结构，多种使用方式

SQL 既是独立的自含式语言，又是嵌入式语言。独立自含式 SQL 能够独立进行联机交互，用户只需在终端键盘上直接输入 SQL 命令就可对数据库进行操作；而作为嵌入式 SQL 语言，能够嵌入到高级语言（例如 C/C++，Java，C＃）程序中来实现对数据库的数据存取操作。在这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特点，为使用 SQL 的程序员与用户提供了极大的灵活性与方便性。

5）语言简洁，易学易用

尽管 SQL 语言功能极强，而且又有两种使用方式，但由于设计巧妙，语言十分简洁，完成核心功能的语句只用了 9 个动词。SQL 的命令动词及其功能如表 3.1 所示。

表 3.1 SQL 的命令动词

| SQL 功 能 | 命 令 动 词 |
|---|---|
| 数据定义(数据模式定义、删除、修改) | CREATE,DROP,ALTER |
| 数据操纵(数据查询和维护) | SELECT,INSERT,UPDATE,DELETE |
| 数据控制(数据存取控制授权和回收) | GRANT,REVOKE |

## 3.2 SQL 的数据定义

SQL 的数据定义包括模式定义、表定义、索引定义、视图定义和定义数据库,如表 3.2 所示。

表 3.2 SQL 的数据定义语句

| 操 作 对 象 | 创 建 语 句 | 删 除 语 句 | 修 改 语 句 |
|---|---|---|---|
| 模式 | CREATE SCHEMA | DROP SCHEMA | |
| 基本表 | CREATE TABLE | DROP TABLE | ALTER TABLE |
| 索引 | CREATE INDEX | DROP INDEX | |
| 视图 | CREATE VIEW | DROP VIEW | |
| 数据库 | CREATE DATABASE | DROP DATABASE | ALTER DATABASE |

在 SQL 语句格式中,语法规定和约定符号说明如下。

• 语句格式约定符号

SQL 语句语法中,尖括号“< >”中的内容为实际语义;中括号“[ ]”中的内容为任选项;花括号“{}”或分隔符“|”中的内容为必选项,即必选其一;[,…] 和[,…n]表示前面的项可重复多次。

• 一般语法规定

SQL 中以英文半角逗号“,”作为数据项(包括表、视图和字段或属性列)的分隔符号,其字符串常量的分界符使用英文半角单引号“’”表示。其他的标点符号也是在英文半角下表示的。

• SQL 特殊语法规定

SQL 的关键字一般使用大写字母表示;SQL 语句的结束符为英文半角分号“;”。注意在 MS SQL Server 中可以省略分号。

另外,为了讲解 SQL 语句语法,本章中使用一个简单的学生课程数据库作为样例数据库。学生课程数据库中的三个关系模式分别为:

学生:Student(Sno,Sname,Ssex,Sbirthday,Sdept) 其属性分别表示为:学号,姓名,性别,出生日期,系名;

课程:Course(Cno,Cname,Cpno,Ccredit) 其属性分别表示为:课程号,课程名,先行课,学分;

选课成绩:SC(Sno,Cno,Grade) 其属性分别表示为:学号,课程号,成绩。

其中,关系的主关键字加下划线表示,外关键字加波浪线表示。Cpno 是 Course 表的外关键字,Sno、Cno 是 SC 表的两个主关键字。

其对应有三张表，该三张表的定义详见 3.2.3 节中的例 3.4 所示。为了对该数据库的结构充分理解，且帮助后面的 SQL 语法的学习与理解，该数据库的表结构图如图 3.2 所示。

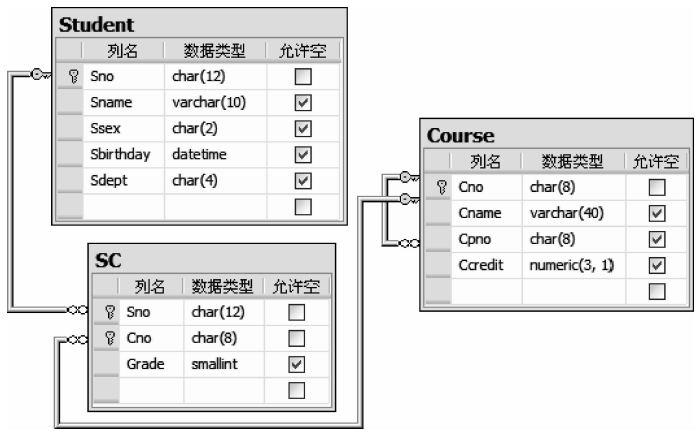


图 3.2　学生课程数据库的表结构

各个表中的数据示例分别如表 3.3～表 3.5 所示。

**表 3.3　学生表 Student 示例数据**

| Sno | Sname | Ssex | Sbirthday | Sdept |
|---|---|---|---|---|
| 070107011101 | 卜玉 | 女 | 1989-8-1 | CS |
| 070107011102 | 陈博 | 男 | 1990-5-11 | CS |
| 070107011103 | 陈亮 | 男 | 1989-1-7 | CS |
| 080171011301 | 王萧 | 男 | 1988-2-2 | CS |
| 080171011304 | 程杏红 | 女 | 1989-9-12 | CS |
| 090301011101 | 蔡华兵 | 男 | 1989-8-17 | EA |
| 090301011102 | 陈晓骏 | 女 | 1988-2-3 | EA |
| 090301011116 | 罗莹莹 | 女 | 1990-5-20 | EA |
| 090171011304 | 陈敏 | 女 | 1989-11-10 | CS |
| 090171021317 | 吴伟平 | 男 | 1990-6-2 | CS |

注：表中“CS”代表“计算机科学系”，“EA”代表“电气工程与自动化系”。

**表 3.4　课程表 Course 示例数据**

| Cno | Cname | Cpno | Ccredit |
|---|---|---|---|
| 0101001 | 数据结构 | 0107002 | 4 |
| 0101002 | 数据库系统原理 | 0101001 | 3.5 |
| 0101003 | 计算机网络 | | 4 |
| 0101011 | 操作系统 | 0101001 | 4 |
| 0101014 | 软件工程 | 0107002 | 3 |
| 0101060 | 专业英语 | | 2 |
| 0101066 | 离散数学 | | 4 |
| 0107002 | C 语言程序设计 | | 4.5 |
| 0601001 | 大学英语 | | 20 |
| 0702001 | 高等数学 | | 10 |

表 3.5 选课成绩表 SC 示例数据

| Sno | Cno | Grade |
|---|---|---|
| 070107011101 | 0101002 | 82 |
| 070107011102 | 0101002 | 65 |
| 070107011103 | 0101002 | 70 |
| 080171011301 | 0101014 | 61 |
| 080171011304 | 0101014 | 82 |
| 090171021317 | 0101001 | 86 |

## 3.2.1 模式的创建与删除

### 1. 创建模式

创建模式的 SQL 语句语法：

```
CREATE SCHEMA [<模式名>] AUTHORIZATION <用户名>
      [<表定义子句> | <视图定义子句> |<授权定义子句>]
```

若没有指定＜模式名＞，则模式名隐含为＜用户名＞。调用该命令的用户必须拥有 DBA 权限，或者被授予了 CREATE SCHEMA 的权限。

**【例 3.1】** 给用户 Steven 定义一个学生-课程模式 StudentCourse。

```
语句 1: CREATE SCHEMA StudentCourse AUTHORIZATION Steven;
语句 2: CREATE SCHEMA AUTHORIZATION Steven;
```

解答说明：语句 2 使用的是隐含模式名形式定义的。

创建模式实际上是定义了一个命名空间，在此空间中可以进一步定义该模式所包含的数据库对象，如基本表、视图、索引等。

若使用完整语法格式，即表示 CREATE SCHEMA 语句可以接受基本表定义、视图定义和授权子句，表示创建模式的同时在此模式中创建基本表、视图和定义授权。

**【例 3.2】** CREATE SCHEMA StudentCourse AUTIONRIZATION Steven。

```
CREATE TABLE Student
(   Sno          CHAR(12),
    Sname        VARCHAR(10),
    Ssex         CHAR(2),
    Sbirthday    DATETIME,
    Sdept        CHAR(4)
);
```

解答说明：该语句给用户 Steven 创建一个模式 StudentCourse，并在此模式中定义了一张表 Student。

### 2. 删除模式

删除模式的 SQL 语句语法：

```
DROP SCHEMA <模式名> <CASCADE | RESTRICT>
```

其中 CASCADE(级联)和 RESTRICT(限制)两者必选其一。级联 CASCADE 表示删除模式时将该模式中所有的数据库对象同时删除。限制 RESTRICT 表示若该模式中存在数据库对象,则拒绝该模式的删除。

**【例 3.3】** DROP SCHEMA StudentCourse CASCADE。

该语句删除模式 StudentCourse,即同时删除该模式中所有的数据库对象。

## 3.2.2　SQL 的数据类型

关系模型中的域是表示属性的特性或取值范围。在 SQL 中域的概念用数据类型来表示。定义基本表的各个属性列时需要指明其数据类型及长度。SQL 中提供了一些主要的数据类型,但不同的数据库系统支持的数据类型不完全相同。表 3.6 列出了 SQL 的主要数据类型。

**表 3.6　SQL 的主要数据类型**

| 类型表示 | | 类型说明 |
|---|---|---|
| 数值型 | SMALLINT | 短整型 |
| | INT 或 INTEGER | 长整型 |
| | NUMERIC($p$,$d$) | 定点数,由 $p$ 位数字(不包括符号和小数点)组成,小数后面有 $d$ 位数字 |
| | FLOAT($n$) | 浮点数,精度至少为 $n$ 位数字 |
| | REAL | 取决于机器精度的浮点数 |
| | Double Precision | 取决于机器精度的双精度浮点数 |
| 字符型 | CHAR($n$) | 长度为 $n$ 的定长字符串 |
| | VARCHAR($n$) | 最大长度为 $n$ 的变长字符串 |
| 日期时间型 | DATE | 日期型,格式为 YYYY-MM-DD,年月日 |
| | TIME | 时间型,格式为 HH:MM:SS,时分秒 |

关于属性列的数据类型选取需要根据实际情况来决定,一般要考虑属性的取值范围及参与什么运算。例如,对于学生的年龄属性,可使用字符型 CHAR(3),但考虑到年龄要参与算术运算,所以最好还是数值型的,因为字符型不能进行算术运算。又因为一个人的年龄在百岁左右,所以选用短整型或微整型作为年龄的数据类型。当然对于年龄而言,实际应用中通常使用日期型表示人的出生日期,用当前日期减去出生日期即可表示年龄。

## 3.2.3　基本表的创建、删除与修改

### 1. 定义基本表

创建了一个模式,也就是建立了一个数据库的命名空间,或称为表空间。在此空间中首先需要定义的数据库对象是该模式所包含的基本表。

SQL 语言使用 CREATE TABLE 语句定义基本表,其一般语法格式为:

```
CREATE TABLE <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][,…] );
```

创建基本表的同时通常定义与该表有关的完整性约束条件,这些约束条件存储在数据库的系统数据字典中,当用户操作表中数据时由数据库系统自动检查该操作是否违背了完整性约束条件。若完整性约束条件涉及该表的多个属性列,则必须定义成表级约束,否则既可定义为列级也可定义为表级。关于完整性约束条件的几点说明如下。

1) 列级完整性约束条件

列级完整性约束是针对属性列赋值的限制条件。SQL 的列级完整性约束有以下几种。

(1) NOT NULL 或 NULL 约束。NOT NULL 约束不允许字段值为空,即非空,而 NULL 约束允许字段值为空。字段值为 NULL 的含义是该属性值"未知"、"不详"或"无意义"。关系的主属性必须限定为"NOT NULL",以满足实体完整性要求。

(2) PRIMARY KEY 约束。

(3) UNIQUE 约束。唯一性约束,即不允许属性列中出现重复的取值。

(4) DEFAULT 约束。缺省值约束,即属性列的默认取值。

(5) CHECK 约束。检查约束,通过约束条件表达式设置属性列应满足的条件。

2) 表级完整性约束条件

表级完整性约束条件是指涉及基本表中多个字段列的限制条件。有以下几种表级约束:

(1) UNIQUE 约束;

(2) PRIMARY KEY 约束;

(3) FOREIGN KEY 约束。

**【例 3.4】** 创建学生-课程模式中的学生表 Student、课程表 Course、选课成绩表 SC。

```
CREATE TABLE Student
(   Sno         CHAR(12)  PRIMARY KEY,          /* 列级完整性约束条件,Sno 为主键 */
    Sname       VARCHAR(10)  UNIQUE,            /* 学生姓名 Sname 取唯一值 */
    Ssex        CHAR(2),                        /* 性别 */
    Sbirthday   DATETIME,                       /* 出生日期 */
    Sdept       CHAR(4)                         /* 所在系别 */
);
CREATE TABLE Course
(   Cno         CHAR(8)  PRIMARY KEY,           /* 列级完整性约束条件,Cno 为主键 */
    Cname       VARCHAR(40),                    /* 课程名称 */
    Cpno        CHAR(8),                        /* 先修课程 */
    Ccredit     NUMERIC(3,1),                   /* 学分 */
    FOREIGN KEY ( Cpno ) REFERENCES Course( Cno )
    /* 表级完整性约束条件,Cpno 为外键,被参照表是 Course,被参照列是 Cno */
);   /* 注: 此表的外键定义表示了同表之间的联系 */

CREATE TABLE SC
(   Sno     CHAR(12),                           /* 学生编号 */
    Cno     CHAR(8),                            /* 课程编号 */
```

```
    Grade   SMALLINT,                          /* 成绩 */
    PRIMARY KEY ( Sno,Cno ),
      /* 表级完整性约束条件,主键由两个属性列构成 */
    FOREIGN KEY ( Sno ) REFERENCES Student( Sno ),
      /* 表级完整性约束条件,Sno 为外键,被参照表是 Student,被参照列是 Sno */
    FOREIGN KEY ( Cno ) REFERENCES Course( Cno ),
      /* 表级完整性约束条件,Cno 为外键,被参照表是 Course,被参照列是 Cno */
);
```

### 2. 删除基本表

当不再需要某个基本表时,可以使用 DROP TABLE 语句删除它。其一般语法格式为:

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

其中,RESTRICT(限制)和 CASCADE(级联)两者可选,一般默认为 RESTRICT,但并不是所有的 DBMS 都支持。限制 RESTRICT 表示删除是有条件的,若该表被其他表的约束所引用(如 CHECK,FOREIGN KEY 等约束),或者存在视图、触发器、存储过程或函数等使用了该表,则拒绝删除该表。级联 CASCADE 表示删除表没有限制条件,删除表的同时删除相关的依赖对象,如视图。

**【例 3.5】** 删除学生表 Student。

```
DROP  TABLE  Student ;
```

**注**:Microsoft SQL Server 没有 RESTRICT 和 CASCADE 选项;Oracle 9i 没有 RESTRICT 选项。

### 3. 修改基本表

有时因为需求的变更导致了数据库的表结构的变化,需要使用 SQL 语句 ALTER TABLE 来修改基本表。其一般语法格式为:

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

其中,ADD 子句用于添加新列和新的完整性约束条件;DROP 子句用于删除指定的完整性约束条件;ALTER COLUMN 子句用于修改原有的列定义,包括修改列名和数据类型。

**【例 3.6】** 向课程表 Course 中增加“学时”字段列,其数据类型为短整型。

```
ALTER TABLE Course ADD Chour SMALLINT;
```

对于新添加的列,基本表中无论有无数据都一律为空值。

**【例 3.7】** 修改选课成绩表 SC,将“成绩”字段类型改为带 1 位的小数类型。

```
ALTER TABLE SC ALTER COLUMN Grade NUMERIC(3,1);
```

**【例 3.8】** 修改课程表 Course,添加课程名称必须取唯一值的约束条件。

```
ALTER TABLE Course ADD UNIQUE ( Cname );
```

**注**：如果要删除约束条件，则需要建立约束时使用命名约束。如下示例所示：

```
ALTER TABLE Course ADD CONSTRAINT UQ__Course UNIQUE ( Cname );
```

删除时则使用：

```
ALTER TABLE Course DROP UQ__Course;
```

### 3.2.4 索引的创建与删除

索引是基本表的目录。一个基本表可以根据应用环境的需要建立一个或多个索引，以提供多种存取路径，加快查找速度。基本表的存储文件和索引的存储文件一起构成了数据库系统的内模式。

#### 1. 索引的功能作用

1）使用索引加快数据查询的速度

如果基本表中的数据量非常大，则其数据文件会非常大。在查询数据时，如果不使用索引则需要将整个数据文件分块，逐个读到内存中，进行查找比较操作。而使用索引后，先将索引文件读入内存，根据索引项找到元组数据的地址，然后再根据该地址将元组数据直接读入计算机。索引文件中只含有索引项和元组地址，一般可以一次读入内存。而且索引项是经过排序了的，所以很快找到索引项及元组地址。使用索引大大减少了磁盘的 I/O 操作，从而加快查询速度。

2）使用索引保证数据的唯一性

定义索引时可以包括定义数据唯一性的要求。这样在对相关数据进行输入或更改时，系统将进行检查来确保数据的唯一性。

3）使用索引加快连接速度

在两个基本表进行连接操作时，系统需要在连接关系中对每一个被连接字段进行查询操作。显然，如果在连接文件的连接字段上建立索引，则可以大大提高连接操作速度。

#### 2. 创建索引

SQL 语言中，创建索引使用 CREATE INDEX 语句，其一般语法格式为：

```
CREATE INDEX [ UNIQUE ] [ CLUSTER ] INDEX <索引名>
ON <表名> ( <列名> [ <次序> ] [,<列名> [ <次序> ] ] … );
```

其中：

- ＜表名＞是要创建索引的基本表的名称。索引可以建立在该表的一列或多列上，各列名之间用逗号分隔。
- 每个＜列名＞后面还可以用＜次序＞来指定索引值的排列次序，次序可选 ASC(升序)或 DESC(降序)，缺省值为 ASC。
- UNIQUE 表示此索引的每一个索引值只对应唯一的数据记录。
- CLUSTER 表示要创建的索引是聚簇索引。所谓聚簇索引是指使索引项的排列顺序与基本表中数据的物理顺序一致的索引组织。