第3章 SQL

SQL(Structed Query Language,结构化查询语言)是操作关系数据库的通用语言。虽然SQL名为查询语言,但不只支持数据库查询操作,其功能还包括数据定义、数据操纵、数据控制等。

SQL成为国际标准语言以来,各数据库厂商纷纷推出各自的SQL软件或与SQL的接口软件。虽然各个数据库厂商附带的SQL软件产品对SQL的支持很相似,但也存在一定的差异。本章主要介绍Microsoft SQL Server支持的SQL。

3.1 SQL概述

3.1.1 SQL的产生与发展

IBM研究人员在20世纪70年代研究出SQL原型,命名为SEQUEL(Structured English Query Language),并在IBM公司研制的关系数据库管理系统原型System R上应用成功。由于SQL简单易学、功能强大,被数据库厂商广泛采用,SQL也因此发展迅速。1986年10月,美国国家标准局(American National Standard Institute,ANSI)采用SQL作为关系数据库管理系统的标准语言,其标准随后被国际标准化组织(International Organization for Standardization,ISO)采纳为国际标准。

在不断的发展中,SQL经历了SQL/86、SQL/89、SQL/92、SQL/99、SQL/2003、SQL/2008、SQL/2011等版本。SQL标准的内容越来越多,也越来越复杂,现在的SQL标准已经包括SQL框架、SQL调用接口、SQL永久存储模块、SQL宿主语言绑定、SQL外部数据管理、XML相关规范等内容。

3.1.2 SQL的特点

SQL是一个综合、功能强大又简单易学的语言,从数据库定义到数据库维护都提供了相应功能。其主要特点如下。

1. 一体化

不论使用SQL完成何种功能,其语法结构统一,这为数据库应用系统的开发提供了良好的使用环境。用户还可以在数据库系统投入使用后,根据需要修改模式而不影响数据库运行,从而使数据库系统具有良好的可扩展性。

2. 高度非过程化

用户只需要使用SQL语句提出“做什么”,而不需要知道“怎么做”。中间的执行过程由

数据库管理系统自动完成。这不但减轻了用户负担,而且提高了数据独立性。

3. 语言简洁

SQL 使用为数不多的命令,就能完成所有功能。SQL 的语法简单,接近英语语法,简单易学。

4. 多种使用方式

SQL 可以直接以命令形式使用,也可以嵌套在多种程序开发语言中使用。现在很多高级语言(例如 Java,C++,C#)均提供了使用 SQL 的模块,可以方便地在程序开发语言中使用 SQL 操作数据库数据。

3.1.3 SQL 功能概述

SQL 的功能主要包括:数据定义、数据查询、数据操纵、数据控制。各功能对应的命令如表 3-1 所示。

表 3-1 SQL 包含的主要命令

SQL 功能	对 应 命 令
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、DENY

数据定义功能用于定义、修改、删除数据库中的对象,这些对象包括表、视图、索引等。数据查询功能用于从数据库中获取满足查询条件的数据。数据操纵功能包括添加、修改、删除数据等功能。数据控制用于管理数据库用户的操作权限,保证数据库的完整性与安全性。

3.2 数 据 定 义

数据库中存在多种数据对象,SQL Server 就包含数据库、表、视图、索引、触发器、存储过程、函数等对象。

3.2.1 数据库定义及维护

从存储的角度看,SQL Server 数据库中的所有数据、对象和事务日志均以文件的形式保存。根据作用不同,这些文件可分为数据文件与事务日志文件。数据文件可根据数据存储需要进行组织,除了必需的一个主数据文件,还可以包括一个或多个次数据文件。

主数据文件用于存储数据库的系统表,数据库对象启动信息和数据库数据。所有数据库只能有一个主数据文件,其文件扩展名为 mdf。

次数据文件用于存储主数据文件中未存储的数据和数据对象。一个数据可有一个或多个次数据文件,其文件扩展名为 ndf。

事务日志文件用于记录对数据库的操作情况。对数据库执行的插入、删除、更新等操作都会记录在文件中。必要时可以根据日志文件恢复数据库。每个数据库至少有一个事务日志文件,其文件扩展名为 ldf。

一旦建立数据库文件，则在 SQL Server 资源管理器目录菜单中就会显示该数据库对象及其包含的数据表、视图等子项。在 SQL Sever 2012 环境中，已建好的数据库 supermarket 如图 3-1 所示。

创建数据库时，SQL Server 首先将系统数据库 model 的内容复制到新数据库，然后使用空页填充新数据库剩余部分。model 数据库中的对象均被复制到新数据库中，其数据库选项也被新数据库继承。在 SQL Server 2012 中，可以使用 SQL Server Management Studio 管理平台和 SQL 语句来建立数据库对象。下面介绍使用 SQL 语句建立数据库对象。

1. 数据库定义

数据库使用 CREATE DATABASE 语句实现，其一般格式如下。

图 3-1 数据库 supermarket 示意图

```
CREATE DATABASE database_name
[ ON
   (NAME = logical_file_name,
[, FILENAME = {'os_file_name'|'filestream_path'} ]
[, SIZE = size [ KB | MB | GB | TB] ] ]
[, MAXSIZE = { maxsize [ KB | MB | GB | TB ] | UNLIMITED  }]
[,FILEGROWTH = growth_increment[KB | MB | GB | TB |  % ]
)  ]
[ LOG ON
(NAME = logical_file_name,
[, FILENAME = {'os_file_name'|'filestream_path'} ]
[, SIZE = size [ KB | MB | GB | TB] ] ]
[, MAXSIZE = { maxsize [ KB | MB | GB | TB ] | UNLIMITED  }]
[,FILEGROWTH = growth_increment[KB | MB | GB | TB |  % ]
)  ]
```

其中，database_name 表示新建数据库的名称，其命名遵循 SQL Server 标识符命名标准。如果在新建数据库时没有指定日志文件逻辑名，则 database_name 后加“-log”作为日志文件的逻辑名和物理名。

ON 后的语句是用来定义数据库数据文件的列表，LOG ON 后的语句为定义日志文件的列表。两个列表均包含文件逻辑名称 NAME，物理文件名称 FILENAME，文件初始大小 SIZE，最大文件大小 MAXSIZE，文件自动增量 FILEGROWTH。表示文件大小的单位默认为 MB。方括号([ ])中的内容表示可选，如果不选，则系统会使用默认值建立数据库。

下面举例说明。

【例 3-1】 创建一个只设置名称的数据库，数据库名称为 dbtest。

```
CREATE DATABASE  dbtest
```

在 SQL Server 2012 中，执行该命令后，会建立逻辑名称为 dbtest，初始大小为 5MB、增量 1MB 且增长无限制的数据文件，其物理文件名为 dbtest. mdf。也会建立逻辑名称为 dbtest_log，初始大小为 2MB、增量 10%且最大为 2 097 152MB 的日志文件，其物理文件名为 dbtest_log. ldf。

【例 3-2】 创建一个包含数据文件和日志文件的数据库 sjkDB。已创建路径 E:\teaching。命名数据文件的逻辑名称为 sjkDB_data，物理文件名为 sjkDB_data. mdf，存放在上述已建

立路径下，初始大小为 6MB，最大 60MB，自动增长增量为 2MB。命名数据库逻辑文件的逻辑名称为 sjkDB_log，物理文件名为 sjkDB_data.ldf，存放在上述已建立路径下，初始大小为 3MB，最大 30MB，自动增长增量为 1MB。

创建此数据库的 SQL 代码如下。

```
CREATE DATABASE sjkDB
ON
(NAME = sjkDB_data,
  FILENAME = 'E:\teaching\sjkDB_data.mdf',
  SIZE = 6,
  MAXSIZE = 60,
  FILEGROWTH = 2)
LOG ON
(NAME = sjkDB_log,
  FILENAME = 'E:\teaching\sjkDB_log.ldf',
  SIZE = 3,
  MAXSIZE = 30,
  FILEGROWTH = 1
  )
```

2. 数据库维护

数据库建立好后，可以使用 ALTER DATABASE 语句对其进行维护。下面简单举例说明，更多应用可以参考 SQL Server 2012 联机丛书。

【例 3-3】 修改数据库 sjkDB 中数据文件的初始大小，将其初始大小改为 9MB，最大为 120MB。

```
ALTER DATABASE sjkDB
MODIFY FILE
(NAME = sjkDB_data,
  SIZE = 9,
  MAXSIZE = 120
)
```

【例 3-4】 为数据库 sjkDB 添加新的日志文件，逻辑名称为 sjkDBlog1，存储路径为 E:\teaching，物理文件名为 sjkDBlog1.ldf，初始大小 3MB，增量 1MB，最大 20MB。

```
ALTER DATABASE sjkDB
ADD LOG FILE
(NAME = sjkDBlog1,
  FILENAME = 'E:\teaching\sjkDBlog1.ldf',
  SIZE = 3,
  MAXSIZE = 20,
  FILEGROWTH = 1
  )
```

【例 3-5】 将数据库 test 更名为 test_1。

```
ALTER DATABASE test
modify name = test_1
```

【例 3-6】 使用 DROP DATABASE 语句删除数据库 dbtest。

```
DROP DATABASE dbtest
```


3.2.2 表定义及维护

表是关系数据库中的基本对象，关系数据库的数据均存储在表中。在关系数据库中，每个关系都对应一个表，一个数据库包含一个或多个表。其他的对象，如视图、索引等均依附于表存在。

3 数据定义——基本表定义、删除

1. 表定义

SQL 使用 CREATE TABLE 语句定义基本表，其基本格式如下。

```
CREATE TABLE < table_name >
( < column_name > < data_type > [column_constraint]
[,< column_name > < data_type > [column_constraint]]
...
[,table constraint]
)
```

其中，table_name 是要定义的表名，column_name 是列名，每个表可以包含一列或多列。在定义表的时候可以定义列级完整性约束(column_constraint)。如果完整性约束涉及该表的多个列，则必须定义为表级完整性约束(table constraint)。

例 3-7～例 3-11 展示了定义一个校园超市数据库各表的较完整的 SQL 语句。该数据库的应用情景假定校园超市主要顾客是学生，其会员用户也假定为学生。该数据库包含学生表、商品表、销售表、商品种类和供应商表。本章后续例子也基于此数据库操作。

【例 3-7】 校园超市的顾客群体主要是学生，如下 SQL 代码实现了“学生”表的建立，字段含义依次是学号、姓名、出生年份、性别、学校、专业、微信号。

```
CREATE TABLE  Student (
   SNO                          varchar(20)   primary key,
   SName                        varchar(20),
   BirthYear                    int,
   Ssex                         varchar(2),
   College                      varchar(100),
   Major                        varchar(100),
   WeiXin                       varchar(100)
)
```

【例 3-8】 建立校园超市的“商品”表，字段含义依次是商品编号、供应商编号、商品种类编号、商品名、商品进价、售价、库存量、生产日期、保质期(月)。

```
CREATE TABLE Goods (
GoodsNO                         varchar(20)      primary key,
SupplierNO                      varchar(20),
CategoryNO                      varchar(20),
GoodsName                       varchar(100),
Barcode                         varchar(100),
InPrice                         decimal(18,2),
```

```
SalePrice                    decimal(18,2),
Number                       int,
ProductTime                  smalldatetime,
QGPeriod                     tinyint,
foreign key (CategoryNO)     references Category (CategoryNO),
foreign key (SupplierNO)     references Supplier (SupplierNO)
)
```

【例 3-9】 建立校园超市的“商品种类”表，字段含义依次为商品种类编号、商品名、商品描述。

```
CREATE TABLE Category (
   CategoryNO                varchar(20)    primary key,
   CategoryName              varchar(100),
   Description               varchar(500)
  )
```

【例 3-10】 建立校园超市的“供应商”表，字段含义依次是供应商编号、供应商名、供应商地址、联系电话。

```
CREATE  TABLE Supplier (
   SupplierNO                varchar(20)   primary key,
   SupplierName              varchar(100),
   Address                   varchar(200),
   Telephone                 varchar(20)
)
```

【例 3-11】 建立校园超市的“销售”表，字段含义依次是商品编码、学号、销售时间、销售数量。

```
CREATE TABLE SaleBill (
   GoodsNO                   varchar(20),
   SNO                       varchar(20),
   HappenTime                datetime,
   Number                    int,
 primary key (GoodsNO, SNO),
foreign key (GoodsNO)  references Goods (GoodsNO),
foreign key (SNO)  references Student (SNO)
)
```

2. 数据类型

在定义表结构时，需要指明每个列的数据类型。每种数据库产品支持的数据类型并不相同，与标准 SQL 也存在差异。

数据的类型决定了数据表中存储数据的存储空间和对这些数据能进行的运算，存储空间决定了数据的存储范围和精度。在 SQL Server 中，凡是具有值的数据对象，如表中的列、变量、函数的参数等，均应该给其定义数据类型。

SQL Server 定义了丰富的基本数据类型，包括字符数据类型、日期时间数据类型、数值数据类型和逻辑数据类型等。

表 3-2 列出了 SQL Server 的常用数据类型。

表 3-2　SQL Server 常用数据类型

数 据 类 型	说　　明
char(n)	固定长度字符串类型,n 表示字符串最大长度,n 字节
nchar(n)	固定长度字符串类型,Unicode 编码,n 字节
varchar(n)	可变长度字符串类型,n 表示字符串最大长度,2n 字节
nvarchar(n)	可变长度字符串类型,Unicode 编码,2×实际字符数字节
text	最多 $2^{31}-1$ 字符,每个字符一个字节
date	0001-1-1～9999-12-31,3 字节
time(n)	小时：分钟：秒.小数秒,n(0～7)指定小数秒位数。3～5 字节
datetime	1753-1-1～9999-12-31,精确到 3.33ms,8 字节
smalldatetime	1900-1-1～2079-6-6,精确到分钟,4 字节
int	-2^{31}～$2^{31}-1$ 的整数,4 字节
smallint	-2^{15}～$2^{15}-1$ 的整数,2 字节
tinyint	0～255 的整数,1 字节
bigint	-2^{63}～$2^{63}-1$ 的整数,8 字节
float(n)	-1.79×10^{308}～1.79×10^{38},n 为 1～24 时,显示 7 位数字的小数,4 字节；n 为 25～53 时,显示 15 位数字的小数,8 字节
decimal(p,q)	$-10^{38}+1$～$10^{38}-1$ 的数值,p 为数字个数,q 为小数位数。最多 17 字节
numeric(p,q)	$-10^{38}+1$～$10^{38}-1$ 的数值,p 为数字个数,q 为小数位数。最多 17 字节
money	−922 337 203 685 477.5808～922 337 203 685 477.5807,8 字节
smallmoney	−214 748.3648～214 748.3647,4 字节

3 数据定义——基本表修改

3. 表维护

建立表后,根据需求变化进行表的维护,主要包括对表结构的修改和完整性约束的修改,本节先阐述表的修改与删除。SQL 使用 ALTER TABLE 语句修改表,其基本格式如下。

```
ALTER TABLE < table_name >
[ADD  < column_name >< data_type >[constaint]]                /* 增加列 */
[DROP COLUMN < column_name >]                                 /* 删除列 */
[ALTER COLUMN < column_name >< data_type >[constaint]]        /* 修改列 */
```

SQL 使用 DROP TABLE 删除表,其基本格式如下。

```
DROP TABLE < table_name > [,< table_name >][…]                /* 删除表 */
```

【例 3-12】 将例 3-9 的“商品种类”表添加一列,用以存放描述商品大类的数据,例如牙刷属于日用品类。

```
ALTER TABLE Category
ADD Cat_CategoryNO  varchar(20)
```

【例 3-13】 将例 3-8 中 Goods 表的 Barcode 列删除。

```
ALTER TABLE  Goods
DROP COLUMN Barcode
```

【例 3-14】 将例 3-10 的 Supplier 表 SupplierName 列的数据类型修改为 nvarchar(200),且不允许为空。

```
ALTER TABLE  Supplier
ALTER COLUMN SupplierName  nvarchar(200)  not null
```

对于已经建立好的表，如果要对某列添加非空约束，也使用上述语句形式完成。

【例 3-15】 假设 sjkDB 数据库有表 sjktable，使用 DROP TABLE 删除表的语句如下。

```
DROP TABLE sjktable
```

3.2.3 完整性定义及维护

数据库完整性是指数据的正确性与相容性。前者要求数据符合现实语义、反映实际情况，后者要求在不同的关系中的相关数据符合逻辑。本节介绍使用 SQL 来定义及维护数据库完整性。

1. 完整性约束定义

如前所述，完整性约束包括实体完整性、参照完整性与用户自定义完整性。使用 SQL 定义的完整性约束的作用范围可以是列级约束、元组约束和关系约束。列级约束是指某列的约束，例如该列的取值范围；元组约束是指元组中各字段之间联系的约束，例如最低工资要小于最高工资等；关系约束是指关系之间联系的约束，例如供货商数据表里没有的商家就不能提供货品。

使用 SQL 定义完整性可以在定义表的时候进行。也可以使用 ALTER TABLE 语句定义完整性，其 SQL 代码如下。

```
ALTER TABLE < table_name >
[ADD  [< constraint > < constraint_name >] < constraint >]     /* 增加约束 */
```

下面举例说明常见完整性约束定义。先定义两张表，然后添加约束。其表结构如表 3-3 和表 3-4 所示。其后为分别定义表的 SQL 语句。

表 3-3 员工表

列　　名	数 据 类 型	约　　束
员工编号	char(7)	非空、主码
姓名	nchar(5)	非空
入职日期	smalldatetime	非空
转正日期	smalldatetime	
手机号码	char(11)	每一位都为数字，11 位，不重复
薪级编号	char(3)	外码

表 3-4 薪资表

列　　名	数 据 类 型	约　　束
薪级编号	char(3)	非空、主码
基础薪资	numeric(8,2)	默认 1200
薪级名称	nchar(10)	取值不重复
应发薪资	numeric(8,2)	非空
实发薪资	numeric(8,2)	非空 小于应发薪资

```
CREATE TABLE 员工表 (
   员工编号        char(7)  not null,
   姓名            nchar(5) ,
   入职日期        smalldatetime not null,
   转正日期        smalldatetime,
   手机号码        char(11),
   薪级编号        char(3)
)
CREATE TABLE 薪资表 (
   薪级编号        char(3)  not null,
   薪级名称        nchar(10) ,
   基础薪资        numeric(8,2) not null,
   应发薪资        numeric(8,2) not null,
   实发薪资        numeric(8,2) not null
 )
```

1）实体完整性

实体完整性要求表中每条记录必须唯一，也不能为空。为保证实体完整性，每个表需指定一个属性或属性组合作为它的主码。主码能够保证数据表中没有重复记录。一个表只能设置一个主码。也可以通过索引、UNIQUE 约束等实现实体完整性。

【例 3-16】 为员工表添加主码。

```
ALTER  TABLE 员工表
ADD CONSTRAINT pk_yg PRIMARY KEY(员工编号)
```

上述语句可以省略 CONSTRAINT pk_yg 语句，只不过约束名就会由数据库系统自动生成，在删除约束的时候需要先查看该约束的名称。定义员工编号为主码以后，该列值就不允许重复或者为空了。还可以在多个列上定义主码，形如 PRIMARY KEY（薪级名称，应发薪资）。

【例 3-17】 为薪资表的薪级名称列添加 UNIQUE 约束。

```
ALTER  TABLE 薪资表
ADD CONSTRAINT U_xinzname  UNIQUE(薪级名称)
```

UNIQUE 约束限制薪级名称列不能出现重复的值，在一个表中可以定义多个 UNIQUE 约束。还可以在多个列上定义 UNIQUE 约束，形如 UNIQUE(薪级名称，应发薪资)。

2）用户自定义完整性

用户自定义完整性指用户根据业务逻辑定义约束规则，防止数据库出现不符合逻辑的数据。用户自定义完整性通过 DEFAULT、CHECK 等实现，也可以使用存储过程、触发器等实现。

【例 3-18】 为薪资表的基础薪资列定义 DEFAULT 约束。

```
ALTER  TABLE 薪资表
ADD CONSTRAINT DF_jichu  DEFAULT 1200 FOR 基础薪资
```

一个列只能有一个 DEFAULT 约束，该约束也只能定义在一个列上。在插入数据时，数据库系统会检查 DEFAULT 约束，不输入新值的情况下，系统会用默认值填充。

【例 3-19】 对薪资表的实发薪资列添加 CHECK 约束，使其值小于应发薪资列。

```
ALTER  TABLE 薪资表
ADD CONSTRAINT CK_shifa  CHECK (实发薪资<应发薪资)
```

数据库系统在插入数据和更新数据时会检查 CHECK 约束，不满足条件会拒绝执行。CHECK 约束也可以对多列的取值关系做约束。

【例 3-20】 对员工表的手机号码列添加 CHECK 约束，使其值符合手机号码规定。

```
ALTER  TABLE 员工表
ADD CONSTRAINT CK_phone
CHECK (手机号码 LIKE'[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]')
```

上述代码使用到的 LIKE 运算符常用于模糊查询，具体用法参照 3.3.1 节。

如果表原来的数据不满足新添加的约束，只对新插入的数据实现约束，则可以在语句中加入 WITH NOCHECK 语句。

【例 3-21】 假设新进员工的薪水要求实发薪资<应发薪资，对原来员工的数据不做要求。则可以添加 WITH NOCHECK 语句来实现这一设定。

```
ALTER  TABLE 薪资表
WITH  NOCHECK
ADD CONSTRAINT CK_nock  CHECK (实发薪资<应发薪资)
```

3）参照完整性

参照完整性属于表间规则，定义了数据库中一个表中的主码与另一个表外码之间的关系，保证两个表的相容性。若主码与外码来自同一个表，则称为自参照完整性。只要依赖于某主码的外码值存在，该表中该主码的值就不能任意修改与删除，除非设置了级联删除与修改。

SQL Server 2012 使用 FOREIGN KEY、触发器等来实现参照完整性。这里介绍使用 FOREIGN KEY 实现参照完整性。

【例 3-22】 为员工表的薪级编码列添加外码约束，引用薪资表的薪级编号。

```
ALTER  TABLE 员工表
ADD CONSTRAINT FK_xinji
FOREIGN KEY(薪级编号) REFERENCES 薪资表(薪级编号)
```

外码约束是参照完整性的具体执行方式，在此例中，参照表是员工表，被参照表是薪资表。在 SQL Server 中，前者被称为外码表，后者被称为主码表，主码表被引用的属性必须是候选码或主码。

对于参照表员工表，如果插入的数据或者修改的数据会破坏参照完整性，则系统会拒绝执行。对于被参照表薪资表，如果删除元组或修改主码值会破坏参照完整性的话，可以有三种违约处理方式，分别是拒绝、级联删除（修改）和设置为空值，其中，拒绝是默认策略。

在例 3-22 中，存在的可能破坏参照完整性的情况有如下 4 种。

（1）在员工表中添加一个元组，其薪级编号属性值在薪级表中无相等的薪级编号属性值。

（2）修改员工表中的薪级编号属性值，修改后的值在薪级表中无相等的薪级编号属性值。

（3）删除薪级表中的某一元组，但其薪级编号属性值与员工表的某一元组薪级编号属性值相等。

（4）修改薪级表中某一元组的薪级编号属性值，但其薪级编号属性值与员工表的某一元组薪级编号属性值相等。

【例 3-23】 为员工表的薪级编码列添加外码约束，引用薪资表的薪级编号。定义该完整性约束可以级联删除或修改。

```
ALTER  TABLE 员工表
ADD CONSTRAINT FK_xinji
FOREIGN KEY(薪级编号) REFERENCES 薪资表(薪级编号)
ON DELETE CASCADE                    /* 级联删除 */
ON UPDATE CASCADE                    /* 级联修改 */
```

定义为级联删除后，如果删除薪级表中的某一个元组，则会删除员工表中与该元组的薪级编号属性值相等的元组。定义为级联修改后，修改薪级表中的某一元组薪级编号属性值，则员工表中的薪级编号属性值会做相应修改。这样就保证了数据库完整性。

上述约束也可以在定义表的时候一并定义，SQL 语句如下。

```
CREATE TABLE 员工表 (
   员工编号        char(7)  primary key,
   姓名            nchar(5)  not null,
   入职日期        smalldatetime  not null,
   转正日期        smalldatetime,
   手机号码        char(11) unique  CHECK (手机号码 LIKE
  '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'),
   薪级编号        char(3) ,
  FOREIGN KEY(薪级编号) REFERENCES 薪资表(薪级编号)
  ON DELETE CASCADE
  ON UPDATE CASCADE
)
CREATE TABLE 薪资表 (
   薪级编号        char(3)  primary key,
   薪级名称        nchar(10)  unique,
   基础薪资        numeric(8,2) default 1200,
   应发薪资        numeric(8,2) not null,
   实发薪资        numeric(8,2) ,
   CONSTRAINT CH1 CHECK (实发薪资<应发薪资)  /* 约束取名为 CH1 */
   )
```

2. 完整性约束维护

对完整性约束的维护主要是修改和删除约束。修改约束可以先删除约束，再添加同名约束。删除约束的语句如下。

```
ALTER TABLE < table_name >
[DROP < constraint_name >]              /* 删除约束 */
```

【例 3-24】 删除员工表外码约束 FK_xinji。

```
ALTER  TABLE 员工表
DROP CONSTRAINT FK_xinji
```

3.2.4 索引定义及维护

通常在数据库中会存储大量的数据，索引是提高查询速度的重要手段。索引与图书目录类似，查找书本内容，可以在目录中直接查看该内容在书本中的页数，而不需要查阅整本书。数据库索引包含来自基本表的属性组成的索引关键字和对应各行数据的存储位置。如果对学生表建立 Sno 列上的索引，属性 Sno 就是索引关键字，其后存储的是该 Sno 值对应元组的存储地址，如图 3-2 所示，箭头表示指针。

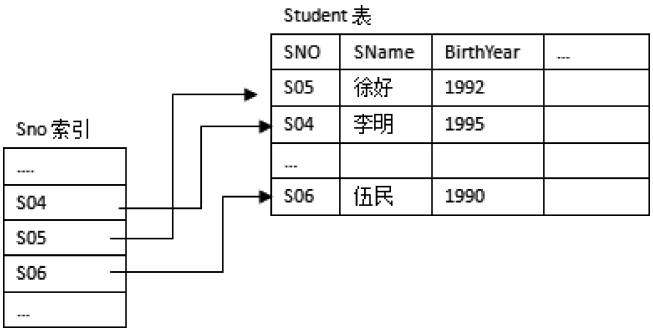

图 3-2 索引表与数据对应关系示意图

索引虽然会加快查询速度，但索引表本身会占用用户数据库空间，在对数据进行插入、更新、删除时，维护索引也会增加时间成本。因此，是否建立索引需要综合考虑。

根据不同观察的角度，索引有聚集索引、非聚集索引、唯一索引等种类。

（1）**聚集索引**是指数据表中的数据按照索引关键字顺序存储。建立聚集索引时，数据库系统会将数据表中的数据按照索引关键字的顺序在磁盘上重新存储。如果在 Student 表上建立聚集索引，则数据表的数据顺序与索引关键字顺序一致。SQL Server 为表设置主码后，就会建立一个主码上的聚集索引。因为一个表的数据只能按照一种物理顺序存储，所以一个表上只能有一个聚集索引。

（2）**非聚集索引**则不要求数据表的数据按照索引关键字顺序排序，表的物理顺序与索引关键字顺序不同。一个表上可以有多个非聚集索引。

（3）**唯一索引**的索引关键字不允许重复。如果在 Student 表的 SName 字段上建立了唯一索引，则 SName 的值不允许重复。

聚集索引与非聚集索引都可以是唯一索引。

SQL 使用 CREATE INDEX 语句建立索引，基本格式如下。

```
CREATE
   [UNIQUE][CLUSTERED|NONCLUSTERED]          /* 定义索引类型 */
   INDEX  index_name                         /* 定义索引名称 */
   ON table_name(column[ASC|DESC][,…n])      /* 定义索引属性列及次序，默认为升序 */
```

【例 3-25】 假设已建立不加约束的如表 3-4 所示的薪资表。为属性薪级名称建立唯一非聚集升序索引。

```
CREATE
   UNIQUE NONCLUSTERED
   INDEX  index_xinz
   ON 薪资表(薪级名称 ASC)
```

【例 3-26】 按应发薪资升序和实发薪资降序建立唯一索引。

```
CREATE
UNIQUE
INDEX  index_yfsf
ON 薪资表(应发薪资 ASC,实发薪资 DESC)
```

可以使用系统存储过程 Sp_helpindex 查看所建立的索引,查看薪资表索引语句为:

```
Sp_helpindex 薪资表
```

结果如图 3-3 所示。

	index_name	index_description	index_keys
1	PK__薪资表__58A53C9411A7D634	clustered, unique, primary key located on PRIMARY	薪级编号
2	UQ__薪资表__5A4820B8F8D9BA45	nonclustered, unique, unique key located on PRI...	薪级名称

图 3-3 薪资表索引示意图

使用该存储过程查看建立好约束的员工表上的索引会发现,主码约束其实是一种索引,在数据库系统中描述为“clustered, unique, primary key…”,UNIQUE 约束默认为非聚集唯一索引,描述为“nonclustered, unique, unique key…”。

索引一经建立,就由数据库系统维护,无须用户参与。但是在建立索引前,应根据需要设计索引,常见准则如下。

(1) 避免在经常更新的表上建立过多索引,如果建立聚集索引,应设置较短的索引长度。

(2) 对经常用于查询中的谓词和连接条件的列建立非聚集索引。

(3) 在经常用作查询过滤的列建立索引。

(4) 在查询中经常进行 GROUP BY、ORDER BY 的列上建立索引。

(5) 在不同值较少的列上不必要建立索引,如性别字段。

(6) 对于经常存取的列避免建立索引。

(7) 在经常存取的多个列上建立复合索引,但要注意复合索引的建立顺序要按照使用的频度来确定。

(8) 考虑对计算列建立索引。

在维护数据库时,随着需求的变化,可能会删除一些索引,减少维护的开销。

SQL Server 2012 删除索引的基本格式如下。

```
DROP  INDEX  table_name.index_name
```

或者

```
DROP  INDEX  index_name  ON  table_name
```

【例 3-27】 删除薪资表上的所有索引。

```
DROP  INDEX 薪资表.index_yfsf, 薪资表.index_xinz
```

或者

```
DROP  INDEX index_yfsf on 薪资表, index_xinz  on 薪资表
```

3.3 数 据 查 询

数据存储到数据库中以后,使用最多的操作就是数据查询。SQL 使用 SELECT 语句进行数据查询,该语句结构简单、使用灵活、功能丰富。其一般格式为:

```
SELECT  [ALL |DISTINCT] < Target Column | Expression >
[, < Target Column | Expression >] …
FROM < TABLE_name | VIEW_name > [, < TABLE_name | VIEW_name > … ]
|  (< SELECT … >)  [AS]  < Alias Name >
[WHERE  < Conditional Expression > ]
[GROUP BY  < COLUMN_name > [, COLUMN_name … ]  HAVING
< Conditional Expression >]]
[ORDER BY < COLUMN_name >[ASC |DESC]];
```

SELECT 语句是根据 WHERE 子句的条件表达式从 FROM 子句指定的对象中筛选出满足条件的元组,再按 SELECT 子句指定的列名、表达式选出属性值形成结果集。FROM 子句指定的对象可以是表、视图或派生表。GROUP BY 子句会根据其后的属性或属性组对查询结果进行分组,HAVING 子句用于筛选满足条件的组予以显示。ORDER BY 子句对查询结果按照其后的属性或属性组进行升序或降序排序。

本节的查询均基于 3.2.2 节定义的校园超市数据库。其设计功能还不能达到实际需要,数据不完全符合实际情况,仅作示例用。

本节的数据部分取自重庆某超市数据库。示例数据如图 3-4～图 3-8 所示。各表字段含义见 3.2.2 节数据表定义。

	GoodsNO	SupplierNO	CategoryNO	GoodsName	InPrice	SalePrice	Number	ProductTime	QGPeriod
1	GN0001	Sup001	CN001	麦氏威尔冰咖啡	5.79	7.80	20	2016-02-08 00:00:00	18
2	GN0002	Sup002	CN001	捷荣三合一咖啡	12.30	17.30	15	2017-10-08 00:00:00	18
3	GN0003	Sup002	CN001	力神咖啡	1.81	2.70	30	2018-05-06 00:00:00	18
4	GN0004	Sup001	CN001	麦氏威尔小三合一咖啡	8.12	10.80	20	2017-05-06 00:00:00	18
5	GN0005	Sup003	CN001	雀巢香滑咖啡饮料	1.99	2.70	3	2018-01-01 00:00:00	18
6	GN0006	Sup003	CN001	雀巢听装咖啡	84.21	113.70	6	2018-05-06 00:00:00	18
7	GN0007	Sup004	CN002	夏士莲丝质柔顺洗发水	25.85	35.70	30	2018-03-08 00:00:00	36
8	GN0008	Sup005	CN002	飞逸清新爽洁洗发水	20.47	30.00	50	2018-03-09 00:00:00	36
9	GN0009	Sup005	CN002	力士柔亮洗发水（中/干）	22.65	32.30	20	2017-12-08 00:00:00	36
10	GN0010	Sup005	CN002	风影去屑洗发水(清爽)	22.98	34.20	6	2017-10-07 00:00:00	36

图 3-4　Goods 表数据

	SNO	SName	BirthYear	Ssex	college	Major	WeiXin
1	S01	李明	2001	男	CS	IT	wx001
2	S02	徐好	2000	女	CS	IT	wx002
3	S03	伍民	1998	男	CS	MIS	wx003
4	S04	闵红	1999	女	ACC	AC	wx004
5	S05	张小红	1999	女	ACC	AC	wx005
6	S06	张舒	2001	男	CS	MIS	wx006
7	S07	王民为	1999	男	CS	MIS	wx007
8	S08	李士任	2001	男	ACC	AC	wx008

图 3-5　Student 表数据

	CategoryNO	CategoryName	Description
1	CN001	咖啡	速溶咖啡、咖啡粉、罐装咖啡
2	CN002	洗发水	袋装、瓶装洗发水
3	CN003	方便面	袋装、碗装方便面

图 3-6　Category 表数据

	GoodsNO	SNO	HappenTime	Number
1	GN0001	S01	2018-06-09 00:00:00	3
2	GN0001	S02	2018-05-03 00:00:00	1
3	GN0001	S03	2018-04-07 00:00:00	1
4	GN0001	S06	2018-06-12 00:00:00	2
5	GN0002	S02	2018-05-08 00:00:00	2
6	GN0002	S05	2018-06-26 00:00:00	3
7	GN0002	S06	2018-06-16 00:00:00	2
8	GN0003	S01	2018-07-10 00:00:00	2
9	GN0003	S02	2018-07-08 00:00:00	2
10	GN0003	S05	2018-06-01 00:00:00	2
11	GN0003	S06	2018-07-01 00:00:00	2
12	GN0005	S05	2018-06-11 00:00:00	1
13	GN0006	S03	2018-05-07 00:00:00	1
14	GN0007	S01	2018-06-09 00:00:00	1
15	GN0007	S04	2018-06-08 00:00:00	1
16	GN0007	S05	2018-06-09 00:00:00	1
17	GN0008	S02	2018-06-04 00:00:00	1
18	GN0008	S06	2018-06-26 00:00:00	1

图 3-7　SaleBill 表数据

	SupplierNO	SupplierName	Address	Telephone
1	Sup001	卡夫食品(中国)有限公司广州分公司	广州佛山	12348768900
2	Sup002	东莞市南城久润食品贸易部	广州东莞	13248768901
3	Sup003	重庆飞鹤食品贸易公司	重庆解放碑	12648768901
4	Sup004	重庆南山日化品贸易公司	重庆南坪	11648768903
5	Sup005	重庆缙云日化品贸易公司	重庆北碚	19648768903

图 3-8　Supplier 表数据

3.3.1　单表查询

3 单表查询——列操作

单表查询是指 From 子句后面的数据表只有一张的查询。下面分别从列筛选和行筛选角度叙述。

1. 选择表中的列

选择列即关系代数中的投影运算。SELECT 子句可以查询指定列、表达式。

1）查询指定列

查询指定列可以是部分列，也可以是全部列，列的显示顺序由 SELECT 子句目标列顺序决定。

【例 3-28】 查询全体学生姓名、学号、专业。

```
SELECT SName,SNO,Major FROM Student
```

查询结果如图 3-9 所示，列排序遵照目标列顺序，与 Student 表中列顺序不同。

【例 3-29】 查询全体学生的详细信息。

```
SELECT SNO,SName,BirthYear,Ssex,college,Major,WeiXin    FROM Student
```

查询结果如图 3-10 所示，如果列的显示顺序与表中的列顺序一致，可以将目标列用 * 代替。

```
SELECT    *    FROM    Student
```

	SName	SNO	Major
1	李明	S01	IT
2	徐好	S02	MIS
3	伍民	S03	MIS
4	闵红	S04	AC
5	张小红	S05	AC
6	张舒	S06	MIS
7	王民为	S07	MIS
8	李士任	S08	AC

图 3-9　例 3-28 查询结果

	SNO	SName	BirthYear	Ssex	college	Major	WeiXin
1	S01	李明	2001	男	CS	IT	wx001
2	S02	徐好	2000	女	CS	IT	wx002
3	S03	伍民	1998	男	CS	MIS	wx003
4	S04	闵红	1999	女	ACC	AC	wx004
5	S05	张小红	1999	女	ACC	AC	wx005
6	S06	张舒	2001	男	CS	MIS	wx006
7	S07	王民为	1999	男	CS	MIS	wx007
8	S08	李士任	2001	男	ACC	AC	wx008

图 3-10　例 3-29 查询结果

查询结果同例 3-29。

2）查询表达式的值

SELECT 子句中的表达式可以是包含列的计算表达式，也可以是常量或函数。

【例 3-30】 查询全体学生的学号、姓名、年龄。

```
SELECT SNO, SName,2020 - BirthYear FROM Student
```

查询结果如图 3-11 所示，Student 表中只记录了学生出生年份，用当前年份 2020 减去出生年份就是学生年龄。

Student 表中学生出生年固定，当前时间却在不断变化，可以使用系统函数 GETDATE() 读取当前时间，再用函数 YEAR() 读取年份，就可以固定表达式。SQL 语句如下，结果同图 3-11。

```
SELECT SNO, SName,YEAR(GETDATE()) - BirthYear FROM Student
```

表达式的计算值被记录在结果集中，但没有列名，如图 3-11 中显示为“无列名”，可以使用 AS 子句为其添加别名记录其语义，AS 也可以省略。

```
SELECT SNO, SName,YEAR(GETDATE()) - BirthYear AS Age FROM Student
```

	SNO	SName	(无列名)
1	S01	李明	19
2	S02	徐好	20
3	S03	伍民	22
4	S04	闵红	21
5	S05	张小红	21
6	S06	张舒	19
7	S07	王民为	21
8	S08	李士任	19

图 3-11　例 3-30 查询结果

查询结果如图 3-12 所示。

也可以使用常量来表示其语义。结果如图 3-13 所示，SQL 语句如下。

```
SELECT SNO, SName,'Age',YEAR(GETDATE()) - BirthYear AS Age FROM Student
```

	SNO	SName	Age
1	S01	李明	19
2	S02	徐好	20
3	S03	伍民	22
4	S04	闵红	21
5	S05	张小红	21
6	S06	张舒	19
7	S07	王民为	21
8	S08	李士任	19

图 3-12　添加别名示意图

	SNO	SName	(无列名)	Age
1	S01	李明	Age	19
2	S02	徐好	Age	20
3	S03	伍民	Age	22
4	S04	闵红	Age	21
5	S05	张小红	Age	21
6	S06	张舒	Age	19
7	S07	王民为	Age	21
8	S08	李士任	Age	19

图 3-13　添加常量示意图

3）去掉重复列

【例 3-31】 查询购买了商品的学生学号。

```
SELECT SNO FROM SaleBill
```

查询部分结果如图 3-14 所示，包含重复的行。使用 DISTINCT 关键字可以去掉结果集中重复的行。例 3-31 可以改写为：

```
SELECT DISTINCT SNO FROM SaleBill
```

查询结果如图 3-15 所示。

3 单表查询——行操作(一)

	SNO
1	S01
2	S02
3	S03
4	S06
5	S02
6	S05
7	S06

图 3-14　例 3-31 查询结果

	SNO
1	S01
2	S02
3	S03
4	S04
5	S05
6	S06

图 3-15　去掉重复值

3 单表查询——行操作(二)

2. 选择表中的元组

在前面选择列的例子中，都是查询表的全部元组。SQL 可以使用 WHERE 子句对元组进行筛选。WHERE 使用查询条件筛选元组，常用查询条件运算符如表 3-5 所示。

表 3-5　常用查询条件运算符

查 询 条 件	谓　　词
比较	=，>，<，>=，<=，!=，<>，！>，！<；NOT＋上述比较运算符
确定范围	BETWEEN AND，NOT BETWEEN AND
确定集合	IN，NOT IN
字符匹配	LIKE，NOT LIKE
空值	IS NULL，IS NOT NULL
多重条件(逻辑运算)	AND，OR，NOT

1）比较大小

比较大小的谓词包括＝(等于)、>(大于)、<(小于)、>＝(大于等于)、<＝(小于等于)、!＝(不等于)、<>(不等于)、!>(不大于)、!<(不小于)。

【例 3-32】 查询管理信息系统专业学生名单。

```
SELECT * FROM Student WHERE Major = 'MIS'
```

查询结果如图 3-16 所示。

	SNO	SName	BirthYear	Ssex	college	Major	WeiXin
1	S03	伍民	1998	男	CS	MIS	wx003
2	S06	张舒	2001	男	CS	MIS	wx006
3	S07	王民为	1999	男	CS	MIS	wx007

图 3-16　例 3-32 查询结果

【例 3-33】 查询年龄不大于 20 的学生名单。

```
SELECT * FROM Student WHERE YEAR(GETDATE()) - BirthYear!> 20
```

查询结果如图 3-17 所示。学生的年龄由表达式 YEAR(GETDATE())-BirthYear 求出。

	SNO	SName	BirthYear	Ssex	college	Major	WeiXin
1	S01	李明	2001	男	CS	IT	wx001
2	S02	徐妤	2000	女	CS	IT	wx002
3	S06	张舒	2001	男	CS	MIS	wx006
4	S08	李士任	2001	男	ACC	AC	wx008

图 3-17　例 3-33 查询结果

2）确定范围

谓词 BETWEEN AND 可以确定取值范围，BETWEEN 后跟范围下限，AND 后跟上限。NOT BETWEEN AND 确定取值范围以外的值。

【例 3-34】 查询现货存量为 3～10 的商品信息。

```
SELECT * FROM Goods WHERE Number BETWEEN 3 AND 10
```

查询结果如图 3-18 所示。

	GoodsNO	SupplierNO	CategoryNO	GoodsName	InPrice	SalePrice	Number	ProductTime	QGPeriod
1	GN0005	Sup003	CN001	雀巢香滑咖啡饮料	1.99	2.70	3	2018-01-01 00:00:00	18
2	GN0006	Sup003	CN001	雀巢听装咖啡	84.21	113.70	6	2018-05-06 00:00:00	18
3	GN0010	Sup005	CN002	风影去屑洗发水(清爽)	22.98	34.20	6	2017-10-07 00:00:00	36

图 3-18　例 3-34 查询结果

【例 3-35】 查询 2017 年生产的商品信息。

```
SELECT * FROM GOODS
WHERE  ProductTime  BETWEEN  '2017－1－1  'AND'2017－12－31'
```

查询结果如图 3-19 所示，日期也是有序数据类型，可以基于范围查询。

【例 3-36】 查询姓名在“李明”和“闵红”之间的学生信息。

```
SELECT * FROM Student WHERE SName BETWEEN'李明'AND'闵红'
```

	GoodsNO	SupplierNO	CategoryNO	GoodsName	InPrice	SalePrice	Number	ProductTime	QGPeriod
1	GN0002	Sup002	CN001	捷荣三合一咖啡	12.30	17.30	15	2017-10-08 00:00:00	18
2	GN0004	Sup001	CN001	麦氏威尔小三合一咖啡	8.12	10.80	20	2017-05-06 00:00:00	18
3	GN0009	Sup005	CN002	力士柔亮洗发水（中/干）	22.65	32.30	20	2017-12-08 00:00:00	36
4	GN0010	Sup005	CN002	风影去屑洗发水(清爽)	22.98	34.20	6	2017-10-07 00:00:00	36

图 3-19　例 3-35 查询结果

查询结果如图 3-20 所示。中文字符串按字符拼音字母先后排序，如果拼音第一个字母相同，则比较第二个字母，以此类推。

	SNO	SName	BirthYear	Ssex	college	Major	WeiXin
1	S01	李明	2001	男	CS	IT	wx001
2	S04	闵红	1999	女	ACC	AC	wx004
3	S08	李士任	2001	男	ACC	AC	wx008

图 3-20　例 3-36 查询结果

3）确定集合

谓词 IN 用来查找属性值属于指定集合的元组。NOT IN 运算符的含义相反，用来查找属性值不属于指定集合的元组。

【例 3-37】 查询商品编号分别为 GN0001、GN0002 的销售信息。

```
SELECT * FROM SaleBill WHERE GoodsNO IN ('GN0001','GN0002')
```

查询结果如图 3-21 所示。

	GoodsNO	SNO	HappenTime	Number
1	GN0001	S01	2018-06-09 00:00:00	3
2	GN0001	S02	2018-05-03 00:00:00	1
3	GN0001	S03	2018-04-07 00:00:00	1
4	GN0001	S06	2018-06-12 00:00:00	2
5	GN0002	S02	2018-05-08 00:00:00	2
6	GN0002	S05	2018-06-26 00:00:00	3
7	GN0002	S06	2018-06-16 00:00:00	2

图 3-21　例 3-37 查询结果

【例 3-38】 查询不是 MIS 专业的学生信息。

```
SELECT * FROM Student WHERE Major NOT IN ('MIS')
```

运行结果如图 3-22 所示，等价于

```
SELECT * FROM Student WHERE Major!= 'MIS'
```

	SNO	SName	BirthYear	Ssex	college	Major	WeiXin
1	S01	李明	2001	男	CS	IT	wx001
2	S02	徐好	2000	女	CS	IT	wx002
3	S04	闵红	1999	女	ACC	AC	wx004
4	S05	张小红	1999	女	ACC	AC	wx005
5	S08	李士任	2001	男	ACC	AC	wx008

图 3-22　例 3-38 查询结果

4）字符匹配

在字符查询条件不确定时，可以使用 LIKE 运算符进行模糊查询。LIKE 运算符通过匹配部分字符达到查询目的，其一般格式如下。

```
[NOT] LIKE  '<匹配串>'  [ESCAPE '<转义字符>']
```

匹配串可以是完整的字符串，也可以是含有通配符的字符串。通配符包括如下四种。

(1) _(下画线)：匹配任一字符。

(2) %(百分号)：匹配任一长度字符串，可以是 0 个，也可以是多个。

(3) [ ]：数据表列值匹配[ ]中任一字符成功，该 LIKE 运算符结果均为 TRUE。如果[ ]中的字符是连续的，可以使用“-”代表中间部分。例如 a、b、c、d，记为[a-d]。

(4) [^]：表示不匹配[]中的任意字符。例如不匹配 a～d 的字符，记为[^a-d]。

【例 3-39】 查询商品名称中包含“咖啡”的商品信息。

```
SELECT *  FROM  Goods  WHERE  GoodsName  LIKE  '%咖啡%'
```

查询结果如图 3-23 所示，不管字符串“咖啡”在元组 GoodsName 列的开头、结尾还是中间，该元组都会被筛选出来。

	GoodsNO	SupplierNO	CategoryNO	GoodsName	InPrice	SalePrice	Number	ProductTime	QGPeriod
1	GN0001	Sup001	CN001	麦氏威尔冰咖啡	5.79	7.80	20	2016-02-08 00:00:00	18
2	GN0002	Sup002	CN001	捷荣三合一咖啡	12.30	17.30	15	2017-10-08 00:00:00	18
3	GN0003	Sup002	CN001	力神咖啡	1.81	2.70	30	2018-05-06 00:00:00	18
4	GN0004	Sup001	CN001	麦氏威尔小三合一咖啡	8.12	10.80	20	2017-05-06 00:00:00	18
5	GN0005	Sup003	CN001	雀巢香滑咖啡饮料	1.99	2.70	3	2018-01-01 00:00:00	18
6	GN0006	Sup003	CN001	雀巢听装咖啡	84.21	113.70	6	2018-05-06 00:00:00	18

图 3-23　例 3-39 查询结果

【例 3-40】 查询学生姓名第二个字为“民”的学生信息。

```
SELECT * FROM Student WHERE SName LIKE'_民%'
```

查询结果如图 3-24 所示，如果去掉后面的%，则查询姓名为两个字，第二个字为“民”的学生信息。

	SNO	SName	BirthYear	Ssex	college	Major	WeiXin
1	S03	伍民	1996	男	CS	MIS	wx003
2	S07	王民为	1997	男	CS	MIS	wx007

图 3-24　例 3-40 查询结果

【例 3-41】 查询商品编号最后一位不是 1、4、7 的商品信息。

```
SELECT *  FROM Goods WHERE GoodsNO NOT  LIKE'%[147]'
```

等同于

```
SELECT  *  FROM Goods WHERE GoodsNO  LIKE'%[^147]'
```

查询结果如图 3-25 所示。

如果查询的字符串含有通配符，为了与通配符区分开，需要使用 ESCAPE 关键字对通

	GoodsNO	SupplierNO	CategoryNO	GoodsName	InPrice	SalePrice	Number	ProductTime	QGPeriod
1	GN0002	Sup002	CN001	捷荣三合一咖啡	12.30	17.30	15	2017-10-08 00:00:00	18
2	GN0003	Sup002	CN001	力神咖啡	1.81	2.70	30	2018-05-06 00:00:00	18
3	GN0005	Sup003	CN001	雀巢香滑咖啡饮料	1.99	2.70	3	2018-01-01 00:00:00	18
4	GN0006	Sup003	CN001	雀巢听装咖啡	84.21	113.70	6	2018-05-06 00:00:00	18
5	GN0008	Sup005	CN002	飞逸清新爽洁洗发水	20.47	30.00	50	2018-03-09 00:00:00	36
6	GN0009	Sup005	CN002	力士柔亮洗发水（中/干）	22.65	32.30	20	2017-12-08 00:00:00	36
7	GN0010	Sup005	CN002	风影去屑洗发水(清爽)	22.98	34.20	6	2017-10-07 00:00:00	36

图 3-25　例 3-41 查询结果

配符进行转义，告诉数据库系统该字符不是通配符，而是字符本身。ESCAPE 关键字后所跟的一个字符为转义字符，转义字符后所跟字符不再为通配符，而是代表其本来的含义。

例如，要查找包含 5%的元组，则其 WHERE 子句部分可以写为：

```
WHERE  column_name  LIKE  '%5a%%'  ESCAPE  'a'
```

其中，字符“a”即为转义字符，表明其后的“%”不是通配符，而是百分号。查询包含“[ ]”元组的 WHERE 子句部分可以写为：

```
WHERE  column_name  LIKE  '%![%!]%'  ESCAPE'!'
```

5）空值查询

空值(NULL)在数据库中表示不确定值，即在字符集中没有确定值与之对应。未对某元组的某个列输入值，就会形成空值(NULL)。涉及空值的判断，不能用前述运算符，只能使用 IS 或 NOT　IS 来判断。

查询还没有输入供应商编号的商品信息可以用如下语句：

```
SELECT * FROM Goods WHERE  SupplierNO  IS  NULL
```

6）多重条件查询

使用运算符 AND 和 OR 可以连接多个查询条件。多个运算符的执行顺序是从左至右，AND 的运算级别高于 OR，用户可以使用小括号改变优先级。AND 连接的条件只有所有子表达式为 TRUE 时，整个表达式的结果才为 TRUE。OR 连接的条件只有所有的子表达式为 FALSE 时，整个表达式的结果才为 FALSE。

【例 3-42】 查询 AC 专业的学生和 MIS 专业男生的信息。

```
SELECT * FROM Student
WHERE Major = 'AC'OR Major = 'MIS'AND Ssex = '男'
```

查询结果如图 3-26 所示，如果用括号改变上述代码执行顺序：

```
SELECT * FROM Student
WHERE (Major = 'AC'OR Major = 'MIS')AND Ssex = '男'
```

3 排序、聚合函数

则其语义变为：查询 AC 专业和 MIS 专业的男生信息，即查询两个专业的男生信息。括号改变了优先级别。查询结果如图 3-27 所示。

3. 对查询结果排序

查询结果可以按照 ORDER BY 子句指定升序(ASC)或降序(DESC)排列，默认为升序。

	SNO	SName	BirthYear	Ssex	college	Major	WeiXin
1	S03	伍民	1998	男	CS	MIS	wx003
2	S04	闵红	1999	女	ACC	AC	wx004
3	S05	张小红	1999	女	ACC	AC	wx005
4	S06	张舒	2001	男	CS	MIS	wx006
5	S07	王民为	1999	男	CS	MIS	wx007
6	S08	李士任	2001	男	ACC	AC	wx008

图 3-26　例 3-42 查询结果

	SNO	SName	BirthYear	Ssex	college	Major	WeiXin
1	S03	伍民	1998	男	CS	MIS	wx003
2	S06	张舒	2001	男	CS	MIS	wx006
3	S07	王民为	1999	男	CS	MIS	wx007
4	S08	李士任	2001	男	ACC	AC	wx008

图 3-27　查询 AC 专业与 MIS 专业男生信息的结果

【例 3-43】 查询学生信息，按出生年升序排列。

```
SELECT  *  FROM Student ORDER BY BirthYear
```

查询结果如图 3-28 所示。ORDER BY 子句后也可以跟多个字段。先按第一个字段的顺序排列，如果第一个字段的排序结果相同，则按第二个字段顺序排列，以此类推。

	SNO	SName	BirthYear	Ssex	college	Major	WeiXin
1	S03	伍民	1998	男	CS	MIS	wx003
2	S04	闵红	1999	女	ACC	AC	wx004
3	S05	张小红	1999	女	ACC	AC	wx005
4	S07	王民为	1999	男	CS	MIS	wx007
5	S02	徐好	2000	女	CS	IT	wx002
6	S01	李明	2001	男	CS	IT	wx001
7	S08	李士任	2001	男	ACC	AC	wx008
8	S06	张舒	2001	男	CS	MIS	wx006

图 3-28　例 3-43 查询结果

【例 3-44】 查询商品名含“咖啡”的商品的商品编号、商品名、现货存量和生产时间。按现货存量升序、生产日期降序排列。

```
SELECT GoodsNO,GoodsName,Number,ProductTime
FROM Goods WHERE GoodsName LIKE  '%咖啡%'
ORDER BY NUMBER  ASC , ProductTime  DESC
```

查询结果如图 3-29 所示。对于第 4、5 条记录，Number 值均为 20，则按生产日期降序排列。

	GoodsNO	GoodsName	Number	ProductTime
1	GN0005	雀巢香滑咖啡饮料	3	2018-01-01 00:00:00
2	GN0006	雀巢听装咖啡	6	2018-05-06 00:00:00
3	GN0002	捷荣三合一咖啡	15	2017-10-08 00:00:00
4	GN0004	麦氏威尔小三合一咖啡	20	2017-05-06 00:00:00
5	GN0001	麦氏威尔冰咖啡	20	2018-02-08 00:00:00
6	GN0003	力神咖啡	30	2018-05-06 00:00:00

图 3-29　例 3-44 查询结果

ORDER BY 子句后也可以跟表达式、函数等。

【例 3-45】 查询商品表的商品编号、商品名称、现货存量、生产日期、保质期剩余天数，按保质期剩余天数升序排列。

```
SELECT GoodsNO,GoodsName,Number,ProductTime,
QGPeriod * 30 - DATEDIFF ( day ,ProductTime ,GETDATE( ) ) 保质期剩余天数
FROM Goods ORDER BY
QGPeriod * 30 - DATEDIFF ( day ,ProductTime ,GETDATE( ) )
```

查询结果如图 3-30 所示，剩余天数为负数表明已经过期。函数 DATEDIFF 计算生产日期 ProductTime 与当前日期的相隔天数，函数 GETDATE 提取系统当前日期。

	GoodsNO	GoodsName	Number	ProductTime	保质期剩余天数
1	GN0001	麦氏威尔冰咖啡	20	2016-02-08 00:00:00	-337
2	GN0004	麦氏威尔小三合一咖啡	20	2017-05-06 00:00:00	116
3	GN0002	捷荣三合一咖啡	15	2017-10-08 00:00:00	271
4	GN0005	雀巢香滑咖啡饮料	3	2018-01-01 00:00:00	356
5	GN0011	我的[5-6]啊	6	2018-02-01 00:00:00	447
6	GN0006	雀巢听装咖啡	6	2018-05-06 00:00:00	481
7	GN0003	力神咖啡	30	2018-05-06 00:00:00	481
8	GN0010	风影去屑洗发水(清爽)	6	2017-10-07 00:00:00	810
9	GN0009	力士柔亮洗发水（中/干）	20	2017-12-08 00:00:00	872
10	GN0007	夏士莲丝质柔顺洗发水	30	2018-03-08 00:00:00	962
11	GN0008	飞逸清新爽洁洗发水	50	2018-03-09 00:00:00	963

图 3-30　例 3-45 查询结果

4. 聚合函数

SQL 使用聚合函数提供了一些统计功能，常见聚合函数及功能如表 3-6 所示。

表 3-6　常见聚合函数及功能

聚合函数名及参数	功　能
COUNT(* |<列名>)	统计元组个数
COUNT([DISTINCT|ALL] <列名>)	统计一列中值的个数
SUM([DISTINCT|ALL] <列名>)	计算一列值的总和(此列必须为数值型)
AVG([DISTINCT|ALL] <列名>)	计算一列值的平均值(此列必须为数值型)
MAX([DISTINCT|ALL] <列名>)	求一列中的最大值
MIN([DISTINCT|ALL] <列名>)	求一列中的最小值

上述函数除了 COUNT(*)外，其余函数均忽略 NULL 值。聚合函数计算时默认为 ALL，如果指定为 DISTINCT，则会忽略重复值。聚合函数是对确认的组或结果集进行计算。

【例 3-46】 查询商品个数。

```
SELECT COUNT( * ) FROM Goods
```

【例 3-47】 查询售出商品种类。

```
SELECT COUNT(DISTINCT GoodsNO) FROM SaleBill
```

这里的 DISTINCT 是必需的，因为 GoodsNO 列有重复值。如果去掉，则统计的是该表

有多少条元组。

【例 3-48】 统计销售表中最多、最少和平均销售量。

```
SELECT MAX(Number)最大销售量 , MIN(Number) 最小销售量,
AVG(Number)平均销售量 FROM SaleBill
```

查询结果如图 3-31 所示。

5. 分组统计

3 分组——GROUP BY

如果查询每个学生购买了几种商品，则需要分别对每个学生的购买记录进行统计。SQL 使用 GROUP BY 子句对元组分组。如果使用 GROUP BY 子句进行分组，数据表中只有出现在 GROUP BY 子句后的列才能放在 SELECT 后面的目标列中，否则 SQL Server 会提示出错信息，“因为该列没有包含在聚合函数或 GROUP BY 子句中”。

分组查询可以先对数据使用 WHERE 进行选择，再使用 GROUP BY 分组查询，一般情况下，可以提高查询效率。

【例 3-49】 统计每个学生购买的商品种类。

```
SELECT SNO,COUNT( * ) AS 商品种类 FROM SaleBill GROUP BY SNO
```

查询结果如图 3-32 所示。销售表(SaleBill)没有考虑某个学生不同时间购买同一种商品的情况。在本例中，数据库系统先分组，再统计。

	最大销售量	最小销售量	平均销售量
1	3	1	1

图 3-31　例 3-48 查询结果

	SNO	商品种类
1	S01	3
2	S02	4
3	S03	2
4	S04	1
5	S05	4
6	S06	4

图 3-32　例 3-49 查询结果

【例 3-50】 统计每个学生购买的商品种类，列出购买 3 种或 3 种以上商品学生的学号，购买商品种类。

```
SELECT SNO,COUNT( * ) AS 商品种类 FROM SaleBill
GROUP BY SNO
HAVING COUNT( * ) > = 3
```

查询结果如图 3-33 所示。HAVING 对组进行选择，后面可以跟列名、聚合函数作为条件表达式。WHERE 对元组进行选择，因此聚合函数不能出现在 WHERE 子句里作为条件表达式。

【例 3-51】 统计学生表中每年出生的男、女生人数，按出生年降序、人数升序排列。

```
SELECT BirthYear,Ssex ,COUNT( * )  FROM Student
GROUP BY BirthYear,Ssex
ORDER BY BirthYear DESC,COUNT( * )
```

查询结果如图 3-34 所示。可以用多个字段作为分组依据，分组以后可以按组进行排序。

	SNO	商品种类
1	SO1	3
2	SO2	4
3	SO5	4
4	SO6	4

图 3-33　例 3-50 查询结果

	BirthYear	Ssex	(无列名)
1	1999	男	3
2	1998	女	1
3	1997	男	1
4	1997	女	2
5	1996	男	1

图 3-34　例 3-51 查询结果

3.3.2　多表连接查询

前面的单表查询只涉及一个表的数据，更多的时候需要从多个表中查询数据。涉及两个或两个以上表的查询，需要先连接后查询。连接包括内连接和外连接。

1. 内连接

3 多表查询——内连接

内连接是一种常见的查询方式。内连接包括非等值连接、等值连接。等值连接的连接字段如果一样，去掉重复的列，就是自然连接。如果连接的是两个相同的表，就是自连接。

在 SQL 中，实现内连接有两种方式，一种是采用 WHERE 子句将连接字段的条件表达式表达出来。例如，将商品表与商品种类表连接起来的语句如下。

```
SELECT * FROM Goods,Category where Goods.CategoryNO = Category.CategoryNO
```

连接结果如图 3-35 所示。商品表的字段 CategoryNO 与商品类别表的字段 CategoryNO 语义相同、数据类型相同(相容)，被用作连接字段。

	GoodsNO	SupplierNO	CategoryNO	GoodsName	InPrice	SalePrice	Number	ProductTime	QGPeriod	CategoryNO	CategoryName	Description
1	GN0001	Sup001	CN001	麦氏威尔冰咖啡	5.79	7.80	20	2016-02-08 00:00:00	18	CN001	咖啡	速溶咖啡、咖啡粉、罐装咖啡
2	GN0002	Sup002	CN001	捕荣三合一咖啡	12.30	17.30	15	2017-10-08 00:00:00	18	CN001	咖啡	速溶咖啡、咖啡粉、罐装咖啡
3	GN0003	Sup002	CN001	力神咖啡	1.81	2.70	30	2018-05-06 00:00:00	18	CN001	咖啡	速溶咖啡、咖啡粉、罐装咖啡
4	GN0004	Sup001	CN001	麦氏威尔小三合一咖啡	8.12	10.80	20	2017-05-06 00:00:00	18	CN001	咖啡	速溶咖啡、咖啡粉、罐装咖啡
5	GN0005	Sup003	CN001	雀巢香滑咖啡饮料	1.99	2.70	3	2018-01-01 00:00:00	18	CN001	咖啡	速溶咖啡、咖啡粉、罐装咖啡
6	GN0006	Sup003	CN001	雀巢听装咖啡	84.21	113.70	6	2018-05-06 00:00:00	18	CN001	咖啡	速溶咖啡、咖啡粉、罐装咖啡
7	GN0007	Sup004	CN002	夏士莲丝质柔顺洗发水	25.85	35.70	30	2018-03-08 00:00:00	36	CN002	洗发水	袋装、瓶装洗发水
8	GN0008	Sup005	CN002	飞逸清新爽洁洗发水	20.47	30.00	50	2018-03-09 00:00:00	36	CN002	洗发水	袋装、瓶装洗发水
9	GN0009	Sup005	CN002	力士柔亮洗发水（中/干）	22.65	32.30	20	2017-12-08 00:00:00	36	CN002	洗发水	袋装、瓶装洗发水
10	GN0010	Sup005	CN002	风影去屑洗发水(清爽)	22.98	34.20	6	2017-10-07 00:00:00	36	CN002	洗发水	袋装、瓶装洗发水

图 3-35　使用 WHERE 连接

另一种连接方式是采用 JOIN…ON 子句连接。本节介绍后一种方式，其一般格式为：

```
FROM <TABLE1_name> [INNER] JOIN <TABLE2_name> ON [<TABLE1_name>.]<COLUMN_name>
<comparisonoperator>[<TABLE2_name>.]
<COLUMN_name> [JOIN…]
```

INNER 关键字表示内连接，可以省略，即 JOIN 连接默认为内连接。关键字 ON 后的连接字段 COLUMN_name 如果在各表中是唯一的，则表名前缀(表 1. 或表 2.)可以省略，否则必须加表名予以区分。连接字段在语法上必须是可以比较的数据类型。在语义上必须符合逻辑，否则比较毫无意义。

1) 等值连接

比较运算符为等号的连接称为等值连接，不为等号时为非等值连接。连接查询中常用等值连接查询。

【例 3-52】 查询学生购物情况。

```
SELECT * FROM Student JOIN SaleBill ON Student.SNO = SaleBill.SNO
```

查询结果前 6 条元组如图 3-36 所示。关系数据库管理系统执行该连接操作的可能过程是：首先定位表 Student 的第一条元组，然后从头扫描 SaleBill 表，如果某元组的 SNO 值与 Student 表第一条元组的 SNO 值相等，则该元组与 Student 表第一条元组连接起来形成结果集的一条元组，直至扫描 SaleBill 表结束。然后定位到 Student 表的第二条元组，再从头扫描 SaleBill 表，进行同样的处理，直至扫描 Student 表结束。

	SNO	SName	BirthYear	Ssex	college	Major	WeiXin	GoodsNO	SNO	HappenTime	Number
1	S01	李明	2001	男	CS	IT	wx001	GN0001	S01	2018-06-09 00:00:00.000	3
2	S02	徐好	2000	女	CS	IT	wx002	GN0001	S02	2018-05-03 00:00:00.000	1
3	S03	伍民	1998	男	CS	MIS	wx003	GN0001	S03	2018-04-07 00:00:00.000	1
4	S06	张舒	2001	男	CS	MIS	wx006	GN0001	S06	2018-06-12 00:00:00.000	2
5	S02	徐好	2000	女	CS	IT	wx002	GN0002	S02	2018-05-08 00:00:00.000	2
6	S05	张小红	1999	女	ACC	AC	wx005	GN0002	S05	2018-06-26 00:00:00.000	3

图 3-36　例 3-52 查询结果

结果集中有两个 SNO 字段，如果去掉重复字段，则为自然连接，SQL 语句如下。

```
SELECT Student.SNO,SName,BirthYear,Ssex,college,Major,
WeiXin,GoodsNO,HappenTime,Number
FROM Student JOIN SaleBill ON Student.SNO = SaleBill.SNO
```

【例 3-53】 查询 MIS 专业学生的购物情况。

```
SELECT  *  FROM Student  JOIN SaleBill ON Student.SNO = SaleBill.SNO WHERE Major = 'MIS'
```

查询结果前 6 条元组如图 3-37 所示。在查询的时候可以把多表连接的结果集看成一个单表来操作，在其后添加 WHERE 子句、GROUP BY 子句等。为简化代码，可以为连接表指定别名，一旦指定别名后，查询语句中相应的表都要用该别名替代。在后面介绍的自连接中，必须使用别名区分相同的表。

	SNO	SName	BirthYear	Ssex	college	Major	WeiXin	GoodsNO	SNO	HappenTime	Number
1	S03	伍民	1998	男	CS	MIS	wx003	GN0001	S03	2018-04-07 00:00:00.000	1
2	S06	张舒	2001	男	CS	MIS	wx006	GN0001	S06	2018-06-12 00:00:00.000	2
3	S06	张舒	2001	男	CS	MIS	wx006	GN0002	S06	2018-06-16 00:00:00.000	2
4	S06	张舒	2001	男	CS	MIS	wx006	GN0003	S06	2018-07-01 00:00:00.000	2
5	S03	伍民	1998	男	CS	MIS	wx003	GN0006	S03	2018-05-07 00:00:00.000	1
6	S06	张舒	2001	男	CS	MIS	wx006	GN0008	S06	2018-06-26 00:00:00.000	1

图 3-37　例 3-53 查询结果

【例 3-54】 查询“CS”学校各学生的消费金额。

```
SELECT college, SNAME, SUM(SA.Number * SalePrice) 消费金额
FROM Student S JOIN SaleBill  SA ON S.SNO = SA.SNO
 JOIN Goods G ON G.GoodsNO = SA.GoodsNO
WHERE college = 'CS'GROUP BY college,SNAME
```

查询结果如图 3-38 所示。在例子中，使用 JOIN 连接了三张表，Student、SaleBill 和 Goods，连接更多的表只需要在现有基础上加 JOIN…ON 就行了。SUM 函数的参数为一个表达式：SA.Number * SalePrice。先用 WHERE college＝'CS'

	college	SNAME	消费金额
1	CS	李明	64.50
2	CS	伍民	121.50
3	CS	徐好	77.80
4	CS	张舒	85.60

图 3-38　例 3-54 查询结果

子句对数据进行选择再分组。也可以先分组再用 HAVING 筛选，SQL 语句如下。

```
SELECT college,SNAME,SUM(SA.Number * SalePrice)消费金额
FROM Student S JOIN SaleBill  SA ON S.SNO = SA.SNO JOIN Goods G
ON G.GoodsNO = SA.GoodsNO
GROUP BY college,SNAME
HAVING college = 'CS'
```

2）自连接

自连接将同一张表进行连接。在连接时必须使用别名使同一张表在逻辑上成为两张表。

【例 3-55】 查询与商品“麦氏威尔冰咖啡”同一类别的商品的商品编号、商品名。

```
SELECT G2.GoodsNO, G2.GoodsName FROM Goods  JOIN Goods G2
ON Goods.CategoryNO = G2.CategoryNO
WHERE Goods.GoodsName = '麦氏威尔冰咖啡'
AND G2.GoodsName!= '麦氏威尔冰咖啡'
```

	GoodsNO	GoodsName
1	GN0002	捷荣三合一咖啡
2	GN0003	力神咖啡
3	GN0004	麦氏威尔小三合一咖啡
4	GN0005	雀巢香滑咖啡饮料
5	GN0006	雀巢听装咖啡

图 3-39　例 3-55 查询结果

查询结果如图 3-39 所示。在该例语句中，只对后一个表名指定了别名即可完成自连接，为了书写方便，也可以为第一个表名取上别名。同类别商品具有相同的类别编号，所以用 CategoryNO 作为连接字段，即同类别编号的商品的元组都会连接一次。商品名为“麦氏威尔冰咖啡”的元组会与同类别商品元组连接，所以子句 WHERE Goods.GoodsName＝'麦氏威尔冰咖啡'就把该类别商品筛选出来，同时查询“麦氏威尔冰咖啡”同类别商品，后一个子句 G2.GoodsName！＝'麦氏威尔冰咖啡'将该元组排除在外。

2. 外连接

3 自身连接、外连接

内连接是将满足连接条件的元组连接起来形成结果集元组，但有时用户需要将不满足连接条件的元组也显示在结果集中，例如查看哪些商品没有人买。这时就需要使用外连接来完成此类查询。外连接包括全外连接，左外连接和右外连接。

1）全外连接

全外连接是将参与连接的表中不满足连接条件的元组均显示出来，无对应连接元组值使用 NULL 填充。假设有两个表 A 与 A2，其数据如图 3-40 和图 3-41 所示。全外连接使用 FULL［OUTER］JOIN…ON 语句连接。如果将 A 与 A2 全外连接，则 SQL 语句如下。

```
SELECT * FROM A FULL  JOIN A2 ON A.SNO = A2.SNO
```

	SNO	SNAME
1	01	A1
2	02	A2
3	03	A3
4	04	A4

图 3-40　A 表数据

	SNO	SNAME
1	03	B3
2	04	B4
3	05	B5
4	06	B6

图 3-41　A2 表数据

查询结果如图 3-42 所示。两表中 SNO 相等值只有 “01”“02”，它们所在的元组对应连接，其余的在对方表无对应值的元组也在结果集中显示，均用 NULL 填充。

2）左外连接

左外连接使用 LEFT［OUTER］JOIN…ON 语句连接。左边表的元组不管满不满足连接条件均显示，右边表不满足连接条件的不显示。如果将 A 与 A2 左外连接的话，其 SQL 语句如下。

```
SELECT * FROM A LEFT  JOIN A2 ON A.SNO = A2.SNO
```

查询结果如图 3-43 所示。

	SNO	SNAME	SNO	SNAME
1	01	A1	NULL	NULL
2	02	A2	NULL	NULL
3	03	A3	03	B3
4	04	A4	04	B4
5	NULL	NULL	05	B5
6	NULL	NULL	06	B6

图 3-42　全外连接查询结果

	SNO	SNAME	SNO	SNAME
1	01	A1	NULL	NULL
2	02	A2	NULL	NULL
3	03	A3	03	B3
4	04	A4	04	B4

图 3-43　左外连接查询结果

3）右外连接

右外连接使用 RIGHT［OUTER］JOIN…ON 语句连接。右边表的元组不管满不满足连接条件均显示，左边表不满足连接条件的不显示。如果将 A 与 A2 右外连接的话，其 SQL 语句如下。

```
SELECT * FROM A  RIGHT  JOIN A2 ON A.SNO = A2.SNO
```

查询结果如图 3-44 所示。改变表在语句中的位置，左外连接与右外连接的查询结果可以一样。查询语句

```
SELECT * FROM A  LEFT  JOIN A2 ON A.SNO = A2.SNO
```

的查询结果可以用语句

```
SELECT * FROM A2  RIGHT JOIN A ON A.SNO = A2.SNO
```

得到。

【例 3-56】 查询没人购买的商品，列出商品名与现货存量。

```
SELECT GoodsName,G.Number FROM Goods G LEFT
JOIN SaleBill GA ON GA.GoodsNO = G.GoodsNO
WHERE GA.SNO IS NULL
```

查询结果如图 3-45 所示。

	SNO	SNAME	SNO	SNAME
1	03	A3	03	B3
2	04	A4	04	B4
3	NULL	NULL	05	B5
4	NULL	NULL	06	B6

图 3-44　右外连接查询结果

	GoodsName	Number
1	麦氏威尔小三合一咖啡	20
2	力士柔亮洗发水（中/干）	20
3	风影去屑洗发水(清爽)	6

图 3-45　例 3-56 查询结果

3.3.3 子查询

在查询中,有一些 SQL 查询语句会用到其他 SQL 查询语句的结果。此时使用子查询是一个很好的方法。子查询就是嵌套在另一个查询语句中的查询语句,因此,子查询也叫嵌套查询。包含子查询的查询通常称为外层查询,通常是从查询语句的第一个“SELECT”开始的查询语句,嵌套在其中的子查询称为内层查询。

3 嵌套查询(一)

1. 不相关子查询

不相关子查询是指内层查询条件不依赖于外层查询。即单独执行内层语句也会得到明确结果集。

3 嵌套查询(二)

【例 3-57】 查询与商品“麦氏威尔冰咖啡”同一类别的商品的商品编号、商品名。

在前面的例子中,使用自连接完成,本节中使用子查询分如下步骤完成。

(1) 查询商品“麦氏威尔冰咖啡”的商品类别编号。

3 嵌套查询(三)

```
SELECT CategoryNO  from Goods WHERE GoodsName = '麦氏威尔冰咖啡'
```

查询结果为“CN001”。

(2) 查询种类编号为“CN001”的商品名字。

```
SELECT GoodsName FROM Goods WHERE CategoryNO = 'CN001'
```

(3) 排除“麦氏威尔冰咖啡”商品。

```
SELECT GoodsName FROM Goods WHERE CategoryNO = 'CN001'
AND GoodsName != '麦氏威尔冰咖啡'
```

“CN001”是第一步查询的结果,第二步需要的结果可以用第一步的查询语句替代,并用小括号将该查询语句括起来。

```
SELECT GoodsName FROM Goods WHERE CategoryNO =
 (SELECT CategoryNO  from Goods WHERE
 GoodsName = '麦氏威尔冰咖啡')
AND GoodsName != '麦氏威尔冰咖啡'
```

查询结果同例 3-55。语句中的“=”也可用“IN”代替。只要子查询的结果是确定的,就可以用与其数据类型相符的运算符进行比较。

【例 3-58】 查询进价大于平均进价的商品名称和进价。

(1) 查询商品平均进价。

```
SELECT AVG(InPrice) FROM Goods
```

查询结果为“20.617000”。

(2) 查询进价大于 20.617 的商品名称和进价。

```
SELECT GoodsName,InPrice FROM Goods
WHERE InPrice > 20.617
```

第二步的 20.617 用第一步的子查询代码替代:

```
SELECT GoodsName,InPrice FROM Goods
```

```
WHERE InPrice >
(SELECT AVG(InPrice) FROM Goods)
```

查询结果如图 3-46 所示。

【例 3-59】 查询购买了“东莞市南城久润食品贸易部”经销的商品的学生学号和姓名。

(1) 查询“东莞市南城久润食品贸易部”的供货商编号。

```
SELECT  SupplierNO   FROM Supplier
WHERE  SupplierName = '东莞市南城久润食品贸易部'
```

查询结果为“Sup002”。

(2) 查询供货商编号为“Sup002”的供货商经销的商品编号。

```
SELECT GoodsNO FROM Goods WHERE SupplierNO = 'Sup002'
```

查询结果为两个元组,商品编号列值分别为“GN0002”和“GN0003”。

(3) 查询购买了商品编号为“GN0002”或“GN0003”的商品的学生学号。

```
SELECT  DISTINCT SNO FROM SaleBill
WHERE GoodsNO IN ('GN0002','GN0003')
```

查询结果为 4 个元组,学号列值分别为“S01”“S02”“S05”和“S06”。

(4) 根据学号查询学生姓名。

```
SELECT SNO,SName FROM STUDENT
WHERE SNO IN('S01','S02','S03','S04')
```

查询结果如图 3-47 所示。

	GoodsName	InPrice
1	雀巢听装咖啡	84.21
2	夏士莲丝质柔顺洗发水	25.85
3	力士柔亮洗发水(中/干)	22.65
4	风影去屑洗发水(清爽)	22.98

图 3-46　例 3-58 查询结果

	SNO	SName
1	S01	李明
2	S02	徐好
3	S03	伍民
4	S04	闵红

图 3-47　例 3-59 查询结果

将子查询结果用相应的子查询语句替代:

```
SELECT SNO,SName FROM STUDENT  WHERE SNO
IN
    (SELECT  DISTINCT SNO FROM SaleBill
    WHERE GoodsNO
    IN
        (SELECT GoodsNO FROM Goods
        WHERE SupplierNO =
            (SELECT SupplierNO FROM Supplier
            WHERE
            SupplierName = '东莞市南城久润食品贸易部')))
```

由本例可见,复杂的查询结果可以由简单的查询组合得到。这也是 SQL 结构化的优点之一。

2. 相关子查询

如果子查询内层查询的查询条件依赖于外层查询，则被称为相关子查询。

1）不带 EXISTS 谓词的子查询

【例 3-60】 查询超过同种类商品平均进价的商品信息。

```
SELECT * FROM Goods WHERE  InPrice >
        (SELECT AVG(InPrice) FROM Goods G
        WHERE CategoryNO = Goods.CategoryNO)
```

查询结果如图 3-48 所示。对于每个外层查询的元组，内层根据其类别号计算出该类别商品的平均进价，再用外层查询当前定位的元组的进价与平均进价比较。内层查询的商品类别号由外层当前定位的元组的类别号决定，因此是相关查询。

	GoodsNO	SupplierNO	CategoryNO	GoodsName	InPrice	SalePrice	Number	ProductTime	QGPeriod
1	GN0006	Sup003	CN001	雀巢听装咖啡	84.21	113.70	6	2018-05-06 00:00:00	18
2	GN0007	Sup004	CN002	夏士莲丝质柔顺洗发水	25.85	35.70	30	2018-03-08 00:00:00	36

图 3-48　例 3-60 查询结果

2）带 EXISTS 谓词的子查询

带有 EXISTS 谓词的子查询不返回任何数据，如果子查询结果不为空，则返回真值“TRUE”，否则返回假值“FALSE”。NOT　EXISTS 则相反。因为只关心返回真值或假值，不关心具体数据，所以带 EXISTS 谓词的子查询往往用“*”代替目标列。

【例 3-61】 查询购买了商品的学生信息。

```
SELECT * FROM Student WHERE EXISTS
        (SELECT * FROM SaleBill WHERE SNO = Student.SNO)
```

查询结果如图 3-49 所示。

	SNO	SName	BirthYear	Ssex	college	Major	WeiXin
1	S01	李明	2001	男	CS	IT	wx001
2	S02	徐好	2000	女	CS	IT	wx002
3	S03	伍民	1998	男	CS	MIS	wx003
4	S04	闵红	1999	女	ACC	AC	wx004
5	S05	张小红	1999	女	ACC	AC	wx005
6	S06	张舒	2001	男	CS	MIS	wx006

图 3-49　例 3-61 查询结果

本例中，外层查询先定位 Student 表中的第一个元组，此时取出的 SNO 列值为“S01”，内层查询根据“S01”在 SaleBill 中查询，如果不为空，则返回真值，则 Student 表中第一个元组被放入结果集，如果为空，返回假值，则第一个元组不被放入结果集。然后外层定位 Student 表中第二个元组，重复这一过程，直到 Student 表查询完毕。

【例 3-62】 查询至少购买了学生 S02 购买的全部商品的学生学号。

本例中，实际上是查找这样的学生 S，凡是 S02 号学生购买过的商品，他（她）都应该购买了。在 SQL 语句中，一次只能处理一个元组。本例中，处理某学生一个元组时，不能知道该学生的其他元组是否包含在 S02 学生的购买商品集合里。因此，可以将该查询转换为如下语义：不存在这样的商品，学生 S02 购买了，而学生 S 没有购买。

```
SELECT DISTINCT SNO FROM SaleBill S1 WHERE
 S1.SNO!= 'S02'AND NOT EXISTS                                    --1层查询
        (SELECT * FROM SaleBill S2 WHERE S2.SNO = 'S02'AND       --2层查询
        NOT EXISTS
            (SELECT * FROM SaleBill S3
            WHERE S3.SNO = S1.SNO AND
            S3.GoodsNO = S2.GoodsNO                              --3层查询
            ) )
```

查询结果为“S06”。

为解释执行过程，分别考查不满足条件的学号“S01”和满足条件的学号“S06”。同时将最外层查询标记为 1 层，其次为 2 层，最里层为 3 层。

(1) 对于学号“S01”

此时，1 层查询定位到学号为“S01”的元组，如图 3-50 所示。至于这个元组的选择条件是否为真，取决于 2 层查询的结果。因为 1 层查询的运算谓词为“NOT EXITS”，因此 2 层查询结果集为空，逻辑值为 FALSE 时，1 层元组才会被选中。

	GoodsNO	SNO	HappenTime	Number
1	GN0001	S02	2018-05-03 00:00:00.000	1
2	GN0002	S02	2018-05-08 00:00:00.000	2
3	GN0003	S02	2018-07-08 00:00:00.000	2
4	GN0008	S02	2018-06-04 00:00:00.000	1

图 3-50　例 3-62 第 2 层查询过程示意图

在定位在第一条记录时，第 3 层查询条件为 SNO＝'S01'，GoodsNO＝'GN0001'。查询结果为有一个记录，因为是 NOT EXISTS，所以对于第 2 层的第一个元组，查询条件为 FALSE，不筛选。

在定位在第二条记录时，第 3 层查询条件为 SNO＝'S01'，GoodsNO＝'GN0002'，查询结果为空。因为第二层子查询条件谓词为 NOT EXISTS，所以对于第二个元组，查询条件为 True，被筛选进第 2 层结果集。此时，因为第 2 层查询结果不为空，所以对于第 1 层的 NOT EXISTS 子句返回的逻辑值为 FALSE，则第 1 层中 SNO＝'S01'的记录不被筛选，即 S01 号学生没有购买 S02 号学生购买的所有商品。

(2) 对于学号“S06”

在第 2 层查询定位第一条记录时，第 3 层查询条件为 SNO＝'S06'，GoodsNO＝'GN0001'。查询结果为有一条记录，因为是 NOT EXISTS，所以对于第 2 层的第一个元组，查询条件返回逻辑值为 FALSE，不筛选。

在第 2 层查询定位第二、三、四条记录时，第 3 层查询条件分别为 SNO＝'S06'，GoodsNO＝'GN0002'；SNO＝'S06'，GoodsNO＝'GN0003'；SNO＝'S06'，GoodsNO＝'GN0008'。查询结果均有一条记录，因为是 NOT EXISTS，所以对于第 2 层的第二、三、四条元组，查询条件返回逻辑值为 FALSE，均不筛选。

第 2 层查询完毕，无被筛选出元组，结果集为空。所以对于第 1 层的 NOT EXISTS 子查询，返回逻辑值为 TURE，即 S06 学号学生购买了 S02 号学生购买的所有商品。

从查询过程来看，第 2 层子查询的结果集是 S02 号学生购买商品元组，第 3 层查询某学号的学生购买商品的元组。如果第 2 层子查询的元组在第 3 层子查询均能查找到，很明显

被判断的学号学生就购买了 S02 号学生购买的所有商品。

3.3.4 集合查询

SQL 还提供了集合查询操作，主要包括并操作 UNION、交操作 INTERSECT 和差操作 EXCEPT。参加集合操作的列数需相等，对应列的数据类型需相同。

3 集合、派生表查询

【例 3-63】 查询 MIS 专业或出生年晚于 1999 年的学生信息。

```
SELECT * FROM Student WHERE Major = 'MIS'
UNION
SELECT * FROM STUDENT WHERE BirthYear > 1999
```

查询结果如图 3-51 所示。

	SNO	SName	BirthYear	Ssex	college	Major	WeiXin
1	S01	李明	2001	男	CS	IT	wx001
2	S02	徐好	2000	女	CS	IT	wx002
3	S03	伍民	1998	男	CS	MIS	wx003
4	S06	张舒	2001	男	CS	MIS	wx006
5	S07	王民为	1999	男	CS	MIS	wx007
6	S08	李士任	2001	男	ACC	AC	wx008

图 3-51 例 3-63 查询结果

【例 3-64】 查询 MIS 专业，出生年晚于 1991 年的学生信息。

```
SELECT * FROM Student WHERE Major = 'MIS'
       INTERSECT
SELECT * FROM STUDENT WHERE BirthYear > 1991
```

【例 3-65】 查询至少购买了学生 S02 购买的全部商品的学生学号。

```
SELECT DISTINCT SNO FROM SaleBill
WHERE SNO!= 'S02'AND  NOT EXISTS
  (
       (SELECT GoodsNO FROM SaleBill  WHERE SNO = 'S02')          -- 子查询 A
             EXCEPT
       (SELECT GoodsNO FROM SaleBill S WHERE
       S.SNO = SaleBill.SNO) )                                    -- 子查询 B
```

查询结果为 S06。

子查询中使用了差操作 EXCEPT，将子查询 A 的查询结果集中，去掉子查询 B 中有的元组。当外层查询中指定 SNO 为 S01 时，子查询 A 查询结果集为{ 'GN0001 ', 'GN0002 ', 'GN0003 ', 'GN0008 '}，子查询 B 查询结果集为{ 'GN0001 ', 'GN0003 ', 'GN0007 '}，求差结果为{ 'GN0002', 'GN0008 '}，则外层 NOT EXISTS 查询条件返回假值，S01 号元组不被选择。对其余学号元组操作过程相同。

3.3.5 基于派生表查询

当子查询出现在 FROM 子句中时，子查询的查询结果形成一个临时派生表，这个表也可以作为查询对象。

【例 3-66】 查询各类别商品的商品种类名和平均售价。

```
SELECT  C.CategoryName , AVG_CA.AVGSALEPRICE
FROM Category C  JOIN
    (SELECT CategoryNO,AVG(SalePrice) FROM Goods
    GROUP BY CategoryNO)
    AS  AVG_CA(CategoryNO,AVGSALEPRICE)
ON C.CategoryNO = AVG_CA.CategoryNO
```

	CategoryName	AVGSALEPRICE
1	咖啡	25.833333
2	洗发水	33.050000

图 3-52 例 3-66 查询结果

查询结果如图 3-52 所示。

本例中,子查询 SELECT CategoryNO,AVG(SalePrice) FROM Goods GROUP BY CategoryNO 查询结果为生产派生表 AVG_CA,子句 AS AVG_CA(CategoryNO,AVGSALEPRICE)为派生表指定别名。如果没有聚合函数,派生表别名后可以不跟列名。AS 关键字可以省略。

【例 3-67】 查询购买了 GN0002 商品的学生信息。

```
SELECT * FROM Student S JOIN
 (SELECT SNO,GoodsNO FROM SaleBill WHERE GoodsNO = 'GN0002') SA_SNO ON S.SNO = SA_SNO.SNO
```

查询结果如图 3-53 所示。

	SNO	SName	BirthYear	Ssex	college	Major	WeiXin	SNO	GoodsNO
1	S02	徐好	2000	女	CS	IT	wx002	S02	GN0002
2	S05	张小红	1999	女	ACC	AC	wx005	S05	GN0002
3	S06	张舒	2001	男	CS	MIS	wx006	S06	GN0002

图 3-53 例 3-67 查询结果

3.3.6 使用 TOP 选择结果集元组

查询 SQL 使用 TOP 谓词选择前 n 条记录。一般格式为:

```
TOP  n  [percent] [ WITH TIES]
```

其中,n 为非负数,表示前 n 条元组; percent 表示前 n%条元组; WITH TIES 表示包括并列结果,如果使用了 WITH TIES,则必须使用 ORDER BY 对结果集进行排序。

【例 3-68】 查询销售额前三的商品与销售额。

```
SELECT TOP 3  G.GoodsNO ,SUM(SA.Number * G.SalePrice)  GOODSUM
 FROM Goods G  JOIN  SaleBill  SA
ON SA.GoodsNO = G.GoodsNO
 GROUP BY G.GoodsNO
ORDER BY GOODSUM DESC
```

查询结果如图 3-54 所示。

【例 3-69】 查询年龄最大的 3 名学生的信息。

```
SELECT TOP 3  WITH TIES  *  FROM Student
 ORDER  BY  BirthYear
```

查询结果如图 3-55 所示。第 2～4 名出生年是一样的,所

	GoodsNO	GOODSUM
1	GN0002	121.10
2	GN0006	113.70
3	GN0007	107.10

图 3-54 例 3-68 查询结果

以结果集有 4 条元组。这里的 ORDER BY 子句不能省略，否则会报错。

	SNO	SName	BirthYear	Ssex	college	Major	WeiXin
1	S03	伍民	1998	男	CS	MIS	wx003
2	S04	闵红	1999	女	ACC	AC	wx004
3	S05	张小红	1999	女	ACC	AC	wx005
4	S07	王民为	1999	男	CS	MIS	wx007

图 3-55　例 3-69 查询结果

3.4　数 据 更 改


3 数据插入

除了广泛使用的数据查询外，用户有时也需要对数据表中的数据进行更改。数据更改包括插入数据、修改数据和删除表中数据。

3.4.1　插入数据

SQL 使用 INSERT 语句插入数据，通常有两种形式，一种是插入一个元组，另一种是插入子查询结果。

1. 插入元组

SQL 插入元组的格式为：

```
INSERT INTO < TABLE_name >
[(< COLUMN_name1 >[, COLUMN_name2] … )]
VALUES
(< CONSTANT1 >[,< CONSTANT2 >] … )
```

其中，如果插入全部列值，则列名可以省略。插入常量的顺序应该与对应的列名顺序一致，否则会出现字段语义错误，或者数据类型不匹配，系统会报错。如果 INTO 子句中只指定了部分列名，没有出现的列允许取空值，则新元组在没有出现的列上插入空值。

【例 3-70】 将学生程浩的信息插入 Student 表中。

```
INSERT INTO Student
VALUES ('S09','程浩',1999,'男','CS','IT','wx009')
```

或

```
INSERT INTO Student(SNO,SName,BirthYear,Ssex,college,Major,WeiXin)
VALUES  ('S09','程浩',1999,'男','CS','IT','wx009')
```

因为是插入全部列，所以第一种写法省略了列名。如果只插入学号、姓名、性别和学校的值，其余默认为空值，则可以写为

```
INSERT INTO Student(SNO,SName,Ssex,college)
VALUES  ('S09','程浩','男','CS')
```

2. 插入子查询结果

INSERT 子句后可以跟子查询语句，将子查询结果集插入相应的表里。其语句格式为：

```
INSERT INTO < TABLE_name >
```

```
( < COLUMN_name1 > [,< COLUMN_name2 > … ])
SELECT …
```

执行该语句需要先建立表,然后将子查询结果集插入。

其语句格式也可以为:

```
SELECT < COLUMN_name1 >[,< COLUMN_name2 > … ]
 I NTO  < NEW_TABLE_name >
FROM …
```

不需要建立表,在执行语句的时候同时建立与查询字段同数据类型的表,该表必须是新表。

【例 3-71】 将例 3-56 的查询结果插入数据库。

使用第一种方法:先建立表 SubGoods,其中一列存放商品名,一列存放存量。

```
CREATE TABLE SubGoods(
GoodName varchar(100),
Number int
)
```

然后在例 3-56 的语句前添加"INSERT INTO SubGoods":

```
INSERT INTO SubGoods
SELECT GoodsName,G.Number FROM Goods G
 LEFT JOIN SaleBill GA
ON GA.GoodsNO = G.GoodsNO  WHERE GA.SNO IS NULL
```

使用 SELECT * FROM SubGood 查看表中数据,其查询结果如图 3-45 所示。

使用第二种方法:

```
SELECT GoodsName,G.Number into  SubGoods1
 FROM Goods G LEFT JOIN SaleBill GA
ON GA.GoodsNO = G.GoodsNO  WHERE GA.SNO IS NULL
```

其所建新表结构如图 3-56 所示,查询结果如图 3-45 所示。

列名	数据类型	允许 Null 值
GoodsName	varchar(100)	☑
Number	int	☑

图 3-56 SubGoods1 表结构

3.4.2 修改数据

3 修改数据

SQL 可以根据需要使用 UPDATE 语句对数据表数据进行更新,其一般格式为:

```
UPDATE < TABLE_name >
SET < COLUMN_name =  Expression >[, … ]
 [WHERE < UPDATE_condition >]
```

其功能是修改指定表中满足 UPDATE_condition 的元组,使用 SET 子句的 Expression 值替代相应 COLUMN_name 原有值。如果没有 WHERE 子句,则对全表所有对应列进行修改。

1. 无条件更新

【例 3-72】 将货物保有量均加 2。

```
UPDATE Goods SET Number = Number + 2
```

2. 有条件更新

【例 3-73】 将过期商品现货存量清零。

```
UPDATE Goods
SET Number = 0
WHERE DATEDIFF(DAY,ProductTime,GETDATE()) - QGPeriod * 30 > 0
```

【例 3-74】 将“重庆缙云日化品贸易公司”的商品加价 10%。

使用连接方式的 SQL 语句：

```
UPDATE Goods
SET SalePrice = SalePrice * 1.1
 FROM Supplier S JOIN Goods G ON G.SupplierNO = S.SupplierNO
WHERE SupplierName = '重庆缙云日化品贸易公司'
```

使用子查询方式：

```
UPDATE Goods
SET SalePrice = SalePrice * 1.1
WHERE SupplierNO IN
        (SELECT SupplierNO FROM Supplier
         WHERE SupplierName = '重庆缙云日化品贸易公司')
```

3 删除数据

3.4.3 删除数据

使用 SQL 删除数据的一般格式为：

```
DELECT [FROM]  < TABLE_name >
[WHERE < DELETE_condition >]
```

DELETE 删除满足条件的元组，FROM 关键字可以省略。没有 WHERE 子句则删除表中全部元组。DELETE 与 DROP 的不同之处在于，前者删除表中的数据，后者删除表的结构。使用 DROP 后，数据库中不再存在删除对象。

1. 无条件删除

无条件删除是删除表中全部数据。

【例 3-75】 删除例 3-71 SubGoods 表中的数据。

```
DELETE SubGoods
```

2. 有条件删除

【例 3-76】 将重庆缙云日化品贸易公司的商品下架，即删除该供应商在商品表中的元组。

1）使用连接方式

```
DELETE FROM Goods
```

```
FROM Supplier S JOIN Goods G ON G.SupplierNO = S.SupplierNO
WHERE SupplierName = '重庆缙云日化品贸易公司'
```

2）使用子查询方式

```
DELETE  FROM Goods
WHERE SupplierNO IN
        (SELECT SupplierNO FROM Supplier
        WHERE SupplierName = '重庆缙云日化品贸易公司')
```

3.5 视　　图

视图是数据库中的常用对象之一，它的内容是数据库部分数据或以聚合等方式重构的数据。在数据库中只存放视图的定义，不存放视图对应的数据，这些数据仍在原数据表中，只有执行视图查询的时候，这些数据才出现在结果集中。因为不存储数据，所以视图与基本表不同，是一个虚表。同时，因为数据存在基本表中，如果基本表的数据发生变化，视图查询的结果集会随之改变。

视图的数据来源可以是一个表，也可以是多个表。定义好的视图可以和基本表一样被查询、被删除。

3.5.1 定义视图


3 定义视图

1. 定义视图

SQL 使用 CREATE VIEW 语句定义视图，其一般格式为：

```
CREATE VIEW < VIEW_name >
[(< COLUMN_name >[,< COLUMN_name >][ … ])]
AS  < SELECT … >
[WITH CHECK OPTION]
```

其中，AS 后的子查询可以是任意具体的数据库系统支持的 SELECT 语句。语句 WITH CHECK OPTION 表示通过视图进行更新操作时要保证更新的数据满足子查询的条件表达式。

组成视图的列名要么省略，要么全部指定。如果省略，则视图的列名就由子查询中的列名组成。在下列情况下，必须指定视图列名。

（1）子查询的某个目标列是聚合函数或列表达式。

（2）多表连接时出现同名列作为视图的列。

（3）需要在视图中指定新列名替代子查询列名。

【例 3-77】 建立咖啡类商品的视图。

```
CREATE VIEW Coffee
AS
SELECT GoodsNO, GoodsName, InPrice, SalePrice, ProductTime
 FROM Goods G JOIN Category C ON G.CategoryNO = C.CategoryNO
 WHERE CategoryName = '咖啡'
```

本例中省略了视图列名，则以子查询目标列为列名。同时子查询以 Goods 表和 Category 表作为数据源。

【例 3-78】 建立 MIS 专业学生的视图，并要求通过视图完成修改与插入操作时视图仍只有 MIS 专业学生。

```
CREATE VIEW MIS_student
 AS
 SELECT * FROM Student WHERE Major = 'MIS'
WITH CHECK OPTION
```

本例使用 WITH CHECK OPTION 语句对以后通过视图进行插入、修改的数据进行限制，均要求满足 Major＝'MIS'条件。

视图可以定义在已经定义的视图上，也可以建立在表与视图的连接上。

【例 3-79】 建立购买了咖啡类商品的学生视图。

```
CREATE VIEW Buy_coffee
 AS
SELECT * FROM Student  WHERE EXISTS
      (
      SELECT * FROM SaleBill S JOIN Coffee  C ON C.GoodsNO = S.GoodsNO
      WHERE S.SNO = Student.SNO )
```

本例基于视图 Coffee 建立了新的视图。

定义基本表时，为了减少数据冗余，表中只存放基本数据，在基本数据上的聚合运算、列表达式运算等一般不予存储。可以定义视图存储这些运算结果，便于使用。

【例 3-80】 建立保存商品编号与销售额的视图。

```
CREATE VIEW SumSale(GoodsNO, SumSale)
AS
SELECT G.GoodsNO,  SUM(SalePrice * S.Number)  SumSale
FROM SaleBill S JOIN Goods G ON S.GoodsNO = G.GoodsNO
group by G.GoodsNO
```

本例中，商品销售额由该商品售价与数量相乘再累加获得，因为不是原数据表列，所以在视图中必须使用新的列名，这里命名为“SumSale”。

视图子查询也可以用 TOP、ORDER BY 谓词。

【例 3-81】 建立销售额前 5 的商品视图。

```
CREATE VIEW Top5SumSale(GoodsNO, SumSale)
AS
SELECT TOP 5  G.GoodsNO,  SUM(SalePrice * S.Number)  SumSale
FROM SaleBill S JOIN Goods G ON S.GoodsNO = G.GoodsNO
group by G.GoodsNO
ORDER BY  SumSale  DESC
```

2. 删除视图

SQL 使用 DROP VIEW 语句删除视图，一般格式为：

```
DROP VIEW < VIEW_name >
```

【例 3-82】 删除视图 Coffee。

```
DROP VIEW Coffee
```

需要注意的是，因为视图 Buy_coffee 部分建立在 Coffee 上，所以删除 Coffee 后，查询 Buy_coffee 不会成功。对于建立在基本表上的视图也一样，如果基本表结构发生变化甚至被删除，则查询视图也会报错。

3.5.2 查询视图

3 视图的操作和作用

定义好视图，就可以像查询基本表一样查询视图了。

【例 3-83】 查询 MIS 专业购买了咖啡类商品的学生信息。

```
SELECT * FROM Buy_coffee WHERE Major = 'MIS'
```

查询结果如图 3-57 所示。数据库系统执行该查询时，先检查涉及的对象是否存在(视图 Buy_coffee)，然后取出视图的定义，将定义中的子查询与用户的查询结合起来(Major＝'MIS')，转换为对基本表的等价查询语句。

```
  SELECT * FROM Student  WHERE Major = 'MIS'
AND EXISTS(
        SELECT * FROM SaleBill S JOIN Coffee  C
          ON C. GoodsNO = S. GoodsNO
        WHERE S. SNO = Student. SNO )
```

	SNO	SName	BirthYear	Ssex	college	Major	WeiXin
1	S03	伍民	1998	男	CS	MIS	wx003
2	S06	张舒	2001	男	CS	MIS	wx006

图 3-57 例 3-83 查询结果

【例 3-84】 查询销售额前 5 商品的供应商编号。

```
SELECT SupplierNO FROM Top 5 SumSale T
 JOIN Goods G ON G. GoodsNO = T. GoodsNO
```

【例 3-85】 查询销售额大于 100 的供应商编号。

```
SELECT * FROM SumSale WHERE SumSale > 100
```

查询结果如图 3-58 所示。这里视图列名与视图相同，这是允许的。在视图中，列名 SumSale 表示的是表达式 SalePrice * S. Number，所以在转换查询时，应该使用 HAVING 谓词。

```
SELECT G. GoodsNO,  SUM(SalePrice * S. Number)  SumSale
FROM SaleBill S  JOIN Goods G ON S. GoodsNO = G. GoodsNO
 group by G. GoodsNO
 HAVING  SUM(SalePrice * S. Number)> 100
```

	GoodsNO	SumSale
1	GN0002	121.10
2	GN0006	113.70
3	GN0007	107.10

图 3-58 例 3-85 查询结果

HAVING 子句 SUM(SalePrice * S. Number)> 100 中的表达式 SUM(SalePrice * S. Number)也不能换成表达式别名 SumSale。

3.5.3 更新视图

更新视图是通过视图来插入、删除、修改数据。由于视图不存储数据，通过视图更新数据最终要转换为对基本表的更新。

【例 3-86】 在 Buy_coffee 视图中插入一个新的学生信息，其中学号为 S09，姓名为 程伟，出生年份为 1993，其余为空。

```
INSERT INTO Buy_coffee (SNO,SName,BirthYear)
 VALUES('S09','程伟',1993)
```

执行时 SQL 转换为

```
INSERT INTO Student (SNO,SName,BirthYear)
VALUES('S09','程伟',1993)
```

S09 学生是新加入的，没有购买过咖啡，并不属于视图 Buy_coffee 包含的数据。为防止这种通过视图更新不属于视图范围的数据，可以在定义视图时添加 WITH CHECK OPTION 子句。

【例 3-87】 将视图 MIS_student 中姓名为“李明”的学生微信更改为“LiMing”。

```
UPDATE MIS_student SET WeiXin = 'LiMing'
 WHERE SName = '李明'
```

会发现系统给出提示：(0 行受影响)。原因是视图 MIS_student 在定义时添加了 WITH CHECK OPTION 子句。学生“李明”不是 MIS 专业的，不满足定义视图时的条件“Major＝'MIS'”。如果更改 MIS 专业学生“徐好”的信息，则会成功。

【例 3-88】 将视图 MIS_student 中姓名为“闵红”的学生元组删除。

```
DELETE FROM MIS_student  WHERE SName = '闵红'
```

同例 3-87 原因一样，删除闵红学生的元组也不会成功，因为闵红不是 MIS 专业学生。

对于不能唯一地转换为对应基本表更新的视图更新，数据库系统会拒绝执行。例如对于语句

```
DELETE FROM Coffee  WHERE GoodsName = '力神咖啡'
```

系统会给出提示“视图或函数 'Coffee' 不可更新，因为修改会影响多个基表”，因为视图 Coffee 连接两个表。要想将 GN0002 商品的销售额改为 120，执行语句

```
UPDATE SumSale SET SumSale = 120 WHERE GoodsNO = 'GN0002'
```

也不会成功，系统会给出提示“对视图或函数 'SumSale' 的更新或插入失败，因其包含派生域或常量域。”，因为 SumSale 字段是各商品销售额的总和，数据库系统无法修改各商品销售额使其总和为 120。

3.5.4 视图的作用

合理使用视图会带来许多好处。

1. 简化数据查询

利用视图，用户可以把经常使用的连接查询、聚合查询等比较复杂的查询定义为视图，这样执行相同查询时，不必重新编写复杂的语句。此时，视图向用户隐藏了复杂的操作，降低了用户操作数据的要求。

2. 使用户多角度看待同一数据

对同一数据，不同用户可以根据需要提取基本表各属性，或者对各属性列进行分组、聚合运算等操作，从而组成新的逻辑对象，提高了数据库应用的灵活性。

3. 提供一定程度的逻辑独立性

当用户对数据库进行增加新的关系或添加新的字段等数据库重构行为时，会影响应用程序的运行。使用视图构造数据库重构之前的逻辑关系，可以保持用户应用程序不变，从而保持了数据逻辑独立性。

4. 提供数据库安全性

可以在设计数据库时对不同用户定义不同的视图，使各级用户只能看到权限范围内的数据。例如，校园超市数据库中的商品表的进价等机密数据就不能被一般员工查询，可以在商品表上建立一个不含进价字段的视图，让一般用户通过视图访问数据表中的数据，而不授予直接访问基本表的权限，这样就在一定程度上提高了数据库安全性。

小　　结

本章介绍了 SQL 的发展历程及其在定义、查询数据库对象中的应用。数据库对象包括基本表、视图、索引、约束等。SQL 使用 CREATE 命令定义对象，使用 ALTER 命令维护基本表及其约束，使用 DROP 命令删除各类对象。

SQL 的强大功能还体现在数据查询上。使用 SELECT 查询语句，可以完成单表查询、多表连接查询、子查询、派生表查询等查询。使用 INSERT 命令插入数据，还可以通过子查询插入数据。使用 DELETE 命令删除数据，UPDATE 更新数据。也可以通过视图简化查询、更新数据。

习　　题

一、简答题

1. 日期类型的输入格式有哪些？

2. 字符串“国庆节快乐”分别用 CHAR(n)、NCHAR(n)类型字段来存储，其中的 n 最小定义为多大合适？

3. CHAR(n)和 VARCHAR(n)的区别是什么？其中 n 表示什么？

4. UNIQUE 的作用是什么？对数据进行什么操作时，检查 UNIQUE 约束？

5. 当操作违反参照完整性约束条件时，一般是怎样处理的？

6. 建立索引需要考虑什么准则？

7. 一个表可以建立多个聚集索引吗？

8. 索引提高查询速度的原理是什么？有哪些类型？

9. 聚集索引一定是唯一索引，这种说法正确吗？反之呢？

10. 索引能提高查询速度，能否在表的每个列都建立索引？

11. 试述数据库完整性约束的种类与作用。

12. 唯一约束与主码约束的区别是什么？

13. 简述视图与基本表的区别和联系。试述视图的优点。

14. 所有的视图是否都可以更新？为什么？

15. 建立视图需要考虑什么准则？

二、综合题

利用本章定义的表 Category、Goods、SaleBill、Student 和 Supplier，编写 SQL 语句完成下列操作。

1. 查询 IT 专业的学生信息。

2. 查询 1992 年后出生的 MIS 专业学生信息。

3. 查询每个供应商供应的商品种类数，列出供应商编号、商品种类数。

4. 统计各供应商的各类别商品的销售额。

5. 分别统计男女生购买的商品类别及其数量。

6. 分别统计男女生购买金额前 3 的商品信息。

7. 查询价格最高的前 3 种商品，列出商品名、库存数量。

8. 查询利润率最高的前 3 种商品，列出商品名、库存数量。

9. 查询购买咖啡数量最多的学生信息，按购买数量降序排列。

10. 查询没有购买过咖啡的学生信息。

11. 建立 Supplier 表上供应商名称的唯一非聚集索引。

12. 查询购买了商品编号为“GN0001”“ GN0002”商品的学生信息。

13. 建立视图，存储将在 30 天内过期的商品信息。

14. 通过 13 题视图，将 30 天内过期的商品售价修改为 8 折。

15. 统计各校的销售额，并插入一个新表。

16. 查询贡献利润最高的学校、专业。

17. 删除销售额排名后两位的供应商信息。

18. 将利润率大于 30%的商品降价 5%。

实　　验

实验 3-1　数据库、数据表定义

一、实验目的

掌握数据库定义及删除。

掌握基本表的定义、修改、删除。

掌握添加、删除约束。

二、实验平台

操作系统：Windows XP/7/8/10。

数据库管理系统：SQL Server 2012。

三、实验内容

1. 使用 SQL 语句创建数据库 students，数据文件初始大小 6MB，增量 1MB，最大 100MB；日志文件初始大小 3MB，增量 10%，最大 80MB，存放 E 盘。

2. 创建表文件 Student、Course、Sc，表结构如表 3-7～表 3-9 所示。

表 3-7　Student 表结构

列　名	含　义	数据类型	约　束
Sno	学号	CHAR(7)	主码
Sname	姓名	NCHAR(5)	非空
Sex	性别	NCHAR(1)	
Sage	年龄	TINYINT	
Sdept	所在系	NVARCHAR(20)	

表 3-8　Course 表结构

列　名	含　义	数据类型	约　束
Cno	课程号	CHAR(6)	主码
Cname	课程名	NVARCHAR(20)	非空
Ccredit	学分	TINYINT	
Semester	学期	TINYINT	

表 3-9　Sc 表结构

列　名	含　义	数据类型	约　束
Sno	学号	CHAR(7)	主码
Cno	课程号	CHAR(6)	主码
Grade	成绩	TINYINT	

3. 为表 Student 添加地址列 Address，数据类型为 NVARCHAR(50)。

4. 将地址列数据类型修改为 NVARCHAR(30)。

5. 删除地址列。

6. 为 Sc 表中的 Sno 添加外码约束，引用 Student 表的 Sno；为 Sc 表添加外码约束，引用 Course 表的 Cno。

7. 为 Student 表中的 Sname 列添加唯一约束，使其值不重复。

8. 为 Sc 表中的 Grade 列添加 CHECK 约束，使其值为 0～100。

9. 为 Student 表中的 Sage 列添加 DEFAULT 约束，使其默认值为 19。

10. 删除第 9 题中的 DEFAULT 约束。

实验 3-2　数据查询

一、实验目的

掌握单表查询。

掌握多表连接查询。

掌握子查询、集合查询。

掌握派生表查询。

掌握聚合函数使用方法。

二、实验平台

操作系统：Windows XP/7/8/10。

数据库管理系统：SQL Server 2012。

三、实验内容

在数据库 supermarket 上完成下列操作。

1. 查询商品种类信息。

2. 查询 IT 专业所有学生信息。

3. 查询 MIS 专业年龄小于 20 岁的学生信息。并为 MIS 列取别名为“信息管理系统”。

4. 查询利润率大于 30%的商品编号与商品名。

5. 查询广州佛山供应的商品信息。

6. 查询购买了商品种类为咖啡的 MIS 专业的学生信息。

7. 查询购买了商品种类为咖啡的各专业的学生人数。

8. 查询购买各商品种类的各专业的学生人数。

9. 查询从未购买过商品的学生信息。

10. 查询与商品编号 GN0005 相同产地的商品编号、商品名。

11. 使用派生表查询各供应商的存货量。

12. 查询售价大于该种类商品售价均值的商品号、商品名。

13. 分别用子查询与连接查询查询购买了商品编号为“GN0003”和“GN0007”的学生学号与姓名。

14. 查询各校销售额。

15. 查询购买额前三的校名、专业名。

16. 使用集合查询方式查询生产日期早于 2018-1-1 或库存量小于 30 的商品信息。

实验 3-3　索引与视图

一、实验目的

掌握索引建立、修改与删除。

掌握建立视图、修改视图、删除视图。

掌握使用视图查询、更新数据。

二、实验平台

操作系统：Windows XP/7/8/10。

数据库管理系统：SQL Server 2012。

三、实验内容

在数据库 supermarket 上完成下列操作。

1. 为表 Supplier 的字段 SupplierName 创建一个非聚集、唯一索引。

2. 使用系统存储过程 Sp_helpindex 查看表 Supplier 的索引情况，如果已有主码，能否为其再建立一个聚集索引？为什么？

3. 删除第 1 题中所建立的索引。

4. 写出创建满足下述要求的视图的 SQL 语句。

(1) 统计每个学生的消费金额。

(2) 统计每个供货商提供的商品种类(一个商品编号代表一种)。

(3) 统计各商品种类的销售数量及平均售价。

(4) 建立 Sup001 供货商的商品信息视图,并要求通过视图完成修改与插入操作时视图仍只有 Sup001 供货商的商品。

5. 利用上述视图,完成如下任务。

(1) 统计每个 MIS 专业学生的消费金额。

(2) 查询售价低于该商品种类售价平均价的商品名和售价。

(3) 利用第 4 题(4)中的视图插入供货商 Sup002 的商品信息,结果如何? 为什么?

(4) 利用第 4 题(4)中的视图删除 GN0004 的商品信息,结果如何? 为什么?

(5) 查询供货种类大于等于 2 的供货商的名称及数量。

实验 3-4　数据更新

一、实验目的

掌握插入数据、删除数据、修改数据。

掌握使用子查询插入数据、更新数据。

二、实验平台

操作系统: Windows XP/7/8/10。

数据库管理系统: SQL Server 2012。

三、实验内容

在数据库 supermarket 上完成下列操作。

1. 添加新品"GN0011　Sup002　CN001　乐至三合一咖啡　12.30　17.30　100　2018-11-12　18"。

2. 先建立一张新表,使用子查询将各月的销售额插入该表,存储月份及销售额。

3. 使用子查询将各学生的购买额插入新表,由系统自建新表,存储学生学号、姓名、销售额。

4. 将所有商品存量增加 2。

5. 将保质期还有 30 天的商品价格打 8 折。

6. 分别使用子查询方式与连接方式将广州地区供货商的商品加价 10%。

7. 将销售额后两位的商品下架。

8. 删除销售额最小的供应商信息。