第5章 关系数据库标准语言SQL

结构化查询语言（Structured Query Language，简称SQL）是关系数据库系统为用户提供的对关系模式进行定义、对关系实例进行操纵的一种语言，是为在关系数据库系统中建立、存储、修改、检索和管理其数据化信息提供的语言工具。目前，SQL已经成为一种工业标准化的数据库查询语言，广泛应用于商用关系数据库管理系统，受到用户的普遍好评。

- 本章讲解的数据库表都以广告监测信息系统的数据库表为例，该系统已经在需求分析和概念设计中作了一定的描述。具体的库表结构，读者可以参考逻辑设计中的广告数据库表卡片。

5.1 SQL语言概述

5.1.1 SQL的发展过程

20世纪70年代中期，IBM公司在研制System R关系数据库系统的过程中，开发了世界上最早的SQL语言。1979年，Oracle公司最先提出了商用的SQL语言，后来在其它几种数据库系统中得以实现。

由于SQL广泛地被多种RDBMS支持和使用，为避免SQL语言的不兼容，以便基于SQL语言的程序容易移植，于是在权威机构多年的工作和努力下，制定了不断完善的SQL标准。 主要版本有：SQL-86 、SQL-89、SQL-92。 SQL-92，也称SQL2，这是目前绝大多数商用RDBMS支持的版本。

本章将参照ANSI SQL-92标准，以微软SQL Server2000关系数据库系统为背景，来介绍SQL的语言概貌。由于不同的关系数据库系统只是遵循SQL-92的大部分特性，有时为了提高系统性能，还提供了针对各自系统的、特定的、非标准的SQL语句，因此读者在使用实际的RDBMS时，一定要参考相关的技术手册。

5.1.2 SQL语言的特点

SQL语言是基于关系模型的数据库查询语言，但它不仅仅具有查询功能，还具有数据定义、数据的增加、删除、修改功能和安全以及事务的控制功能。SQL语言是一种非结构化的程序语言，具体应用中不需要用户考虑如何做，只需要写出做什么的语句即可。

综合起来它具有以下四个特点：

- 功能一体化
- 使用方式灵活
- 高度非过程化
- 简洁易学

① 功能一体化

非关系模型的数据语言一般分为模式DDL(数据描述语言)、子模式DDL、内模式DDL以及DML(数据操纵语言)。它们各自完成模式、子模式、内模式定义和数据存取、处理功能。而SQL语言能完成关系模式定义、数据录入以及数据库的建立、查询、更新、维护、重构、安全性控制等一系列操作，它具有DDL、DML、DCL(数据控制语言)为一体的特点，用SQL语言就可以实现数据库生命期内的全部活动。

另外，由于关系模型中实体以及实体之间的联系均由关系来表示，这种数据结构的单一性带来了数据库操纵符的统一性。由于信息仅仅以一种方式表示，因此所有操作都只需要一种操作符。

② 使用方式灵活

SQL语言没有任何屏幕处理或用户输入输出的能力，它只提供访问数据库的标准方法，因此在实际使用中必须和其他工具或语言配合使用才能完成具体的应用任务。SQL语言有两种使用方式：一种是联机交互方式（如SQL Server2000的查询分析器以及Delphi的Database Explore）；另一种是嵌入到某种高级语言（如C、FORTRAN、PASCAL等）的程序中，以实现对数据库的操作。前一种方式下，SQL语言为自含式语言，借助第三方工具的交互式界面可以独立使用；后一种方式下SQL语言作为嵌入式语言，它依附于主语言。前一种方式适用于非计算机专业人员，后一种方式适用于程序员。这两种方式给了用户灵活选择的余地，提供了极大的方便。尽管方式不同，但是SQL语言的语法结构是基本一致的。

③ 高度非过程化

保证数据库过程化的语言要求用户在程序设计中不仅要指明程序做什么，而且需要程序员按照一定的算法编写出怎样做的程序。而对于SQL语言来说，只要求用户提出目的即做什么，而不用指出如何去实现目的。在SQL语言的两种使用方式中均是如此，用户不必了解存取路径，存取路径的选择以及SQL语句操作的过程都交给RDBMS来完成。

④ 简洁易学

尽管SQL语言功能强大，又有两种使用方法，但由于巧妙的设计，语言十分简洁，因此易于学习和使用。SQL完成核心功能一共只用了9个动词（其中标准SQL是6个）。表5.1列出了表示SQL功能的动词。另外，SQL语言的语法非常简单，接近英语的口语。而且，目前几乎所有的RDBMS都支持SQL语言，所以使用标准SQL语言的程序可以方便地从一种RDBMS移植到另一种RDBMS上。

表5.1 SQL语言功能的动词

SQL功能	动词
数据库查询	SELECT
数据定义	CREATE，DROP，ALTER
数据操纵	INSERT，UPDATE，DELETE
数据控制	GRANT，REVOKE

5.1.3 SQL数据库的体系结构

SQL语言支持关系数据库三级模式结构，如图5.1所示。其中外模式对应于视图(View)和部分基本表(Base Table)，模式对应于基本表，内模式对应于存储文件。

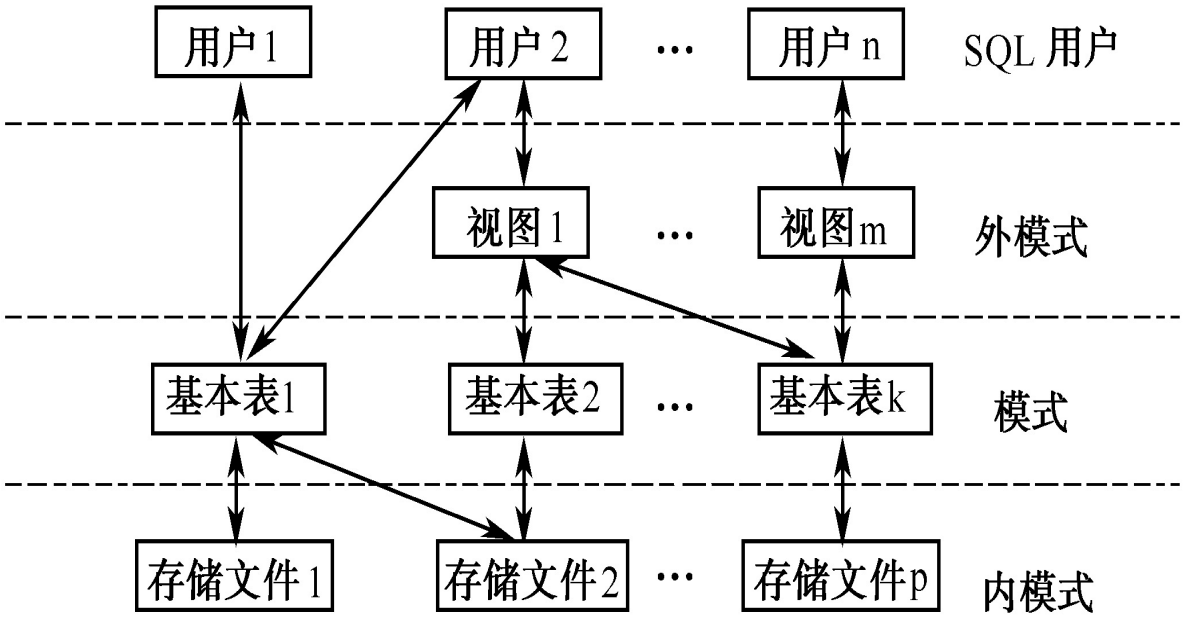


图5.1 SQL语言对关系数据库的支持

用户可以用SQL语言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。基本表是本身独立存在的表；视图是从基本表或其他视图导出的表。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存放视图对应的数据，这些数据仍存放在导出视图的基本表中，因此视图是一个虚表。

一个基本表可以跨一个或多个存储文件，而一个存储文件也可以存放一个或多个基本表。一个基本表可以有多个索引，索引也存放在存储文件中。存储文件的逻辑结构组成了关系数据库的内模式；存储文件的物理结构是任意的，对用户透明。

SQL用户可以是终端用户，也可以是应用程序。SQL用户交互方式或者嵌入式方式与数据库进行连接操作。

5.2 SQL 语言基础

几乎所有的SQL语句都牵涉到数据类型、运算以及一些常用函数的问题，因此在讲解SQL语句之前，我们简单介绍一下SQL语言的基础知识。这些基础对不同的关系数据库产品相互之间会有一些差异，本节的基础知识以及本章后续的SQL语句的主要应用背景是SQL Server2000。考虑到Sybase的SQL语言和SQL Server比较相近，因此在介绍过程中，只是适当地指出SQL Server与Oracle的相关SQL语句的差别。

5.2.1 SQL语言的数据类型及其运算

SQL定义的常用数据类型有数字、字符串、布尔、时间和大对象等数据类型，具体如表5.2所示。

对于每一种数据类型，都有类型转换、赋值、比较和其他操作四类，下面对这些操作做一简单的介绍。

1. 类型转换

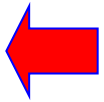
数据转换用CAST标量算符，数据转换的语法如下：

- **CAST（源数据 AS 目标数据）**

该语法表示把源数据类型值转换为目标数据类型值。这里要特别注意的是并非所有的数据类型之间都可以相互转换；针对不同的数据类型，其目标数据类型也不一样。下面对各种数据类型的转换做一简单介绍。

类别	数据类型	说明
数字	INT	整数类型，一般长度为4个字节
	SMALLINT	小整数类型，一般长度为2个字节
	NUMERIC	数值型类型，格式NUMERIC（N，M），N表示总位数，M表示小数点右边的位数。
	DECIMAL	数值型类型，格式和含义同NUMERIC，区别是它的精度可以大于给定的精度。
	FLOAT	浮点数值类型，格式FLOAT（N）精度可大于给定精度。由于实际应用系统的小数位数固定，如保留2位或4位，所以为了不产生累计误差，一般用NUMERIC和DECIMAL
布尔型	BIT	布尔类型，取值TRUE和FALSE。Oracle没有该类型，一般用CHAR代替

类别	数据类型	说明
字符串	CHAR	字符串类型，格式CHAR（N），N表示字符串长度
	VARCHAR	长字符串类型，格式VARCHAR（N），N表示最大变长字符串长度，实际占用空间为实际的字符串长度加1
时间	DATETIME	长时间类型，可以按照指定格式显示和输入。一般格式是年 - 月 - 日 时：分：秒
	SMALL - DATETIME	短时间类型，可以按照指定格式显示和输入。一般格式是年 - 月 - 日。ORACLE只有DATETIME一种时间类型。
大对象	IMAGE	图象类型，可把图象等大对象以二进制流形式存入到数据库表，在ORACLE中数据类型一般是BLOB。
	TEXT	文本类型，可把大的文本对象以字符流形式存入数据库表，在ORACLE中一般用LONG。



① 数字类型的CAST运算

一般数字类型的数据可以转换为更长类型的数字类型数据，同时近似数据类型和精确数据类型之间可以相互转换。另外，数字类型可以转换为字符串类型的数据。下面举例如下：

CAST（25 AS INTEGER）

CAST（1.47E-5 AS DECIMAL（9，5））；得到数字0.00001。

CAST（25 AS CHAR（2））；得到字符串‘25’。

CAST（1.47E-5 AS CHAR（8））；得到字符串‘.0000147’。

② 布尔型

布尔类型可以转换为字符串类型的数据。当布尔值为TRUE，且字符串的长度至少是4位时，转换的结构是‘TRUE’；当布尔值为FALSE，且字符串的长度至少是5位时，转换的结构是‘FALSE’。

③ 字符串类型

字符串类型的数据可以转换为不同类型的字符串数据；当字符串本身是数字字符形式时，可以转换为相应的数字型数据，否则将出错；字符串数据也可以转换为时间类型数据，前提是格式必须符合，否则将出错；当字符串符合“TRUE”、“FALSE”等格式时，还可以转换为布尔类型的数据，下面举例如下：

CAST（ ‘-25’ AS SMALLINT）；得到数字-25。

CAST（ ‘050.00’ AS DECIMAL（3，1））；得到数字50.0。

CAST（ ‘2004-01-01’ AS DATETIME）；得到日期2004年01月01日

.

CAST（ ‘FALSE’ AS BIT）；得到布尔值FALSE。

④ 时间类型

时间类型的数据可以转换为其他类型的时间数据,多数情况下是转换为字符串类型的数据。时间类型的数据转换为字符串时，一般按照默认的时间格式进行转换。下面举例如下：

CAST（DATETIME ‘2004-01-05 10:20:20’ AS SMALLDATETIME）

；得到日期2004-01-05。

CAST（DATETIME ‘2004-01-05’ AS CHAR（10））

□；得到字符串 ‘2004-01-05’。

2. 赋值

SQL赋值时，原则上要求源数据和目标数据的数据类型一致。赋值语义相对简单，跟一般的编程语言的赋值含义没有本质差别，因此在这里不再进行详细说明。

3. 比较

SQL语言提供了一般标量比较运算符：

=、<>、<、<=、>、>=

来进行运算。所有这些都是读者所熟悉的，在其他计算机语言中也有类似的算符。在SQL中比较的结果是一个布尔值：TRUE、FALSE、UNKNOWN，其中UNKOWN表示比较的一方的值是NULL。下面针对不同的数据类型予以简单说明。

① 数字类型的比较运算

数字是按照通常的方式进行比较的，具体示例如下：

- NULL = ？ ；得到布尔值UNKNOWN。
- 97 = 105.2 ；得到布尔值FALSE。
- 9944 > 944 ；得到布尔值TRUE。
- 97〈〉105.2 ；得到布尔值TRUE。

另外，对于数字型SQL还提供了三个量词：ALL，SOME，ANY，可以与比较符一起用于真值与一个子查询返回的真值集合的比较。如果集合是一个空集，或者比较符对于集合中的任何一个值都返回TRUE，那么ALL返回TRUE；如果比较符对于集合中的至少一个值返回结果FALSE，则ALL返回FALSE。SOME和ANY是同义词，它们在比较运算符对于集合中的至少一个值返回TRUE时，返回TRUE；如果集合是一个空集，或者比较运算符对于集合中的任何一个值都返回FALSE时，则返回FLASE。

② 布尔型

在SQL中，布尔值TRUE>FALSE，因此布尔型数据的比较运算较简单。对于布尔型，SQL语言也提供了三个量词：ALL，SOME和ANY，具体含义同数字型，这里不再介绍。

③ 字符串类型

对于CHAR、VARCHAR等字符串的运算，SQL语言提供了所有常用的标量运算符；而对于IMAGE和TEXT之类的大对象（其实也是一种大的字符串数据类型），则只提供了=和<>两个算符。字符串的比较，是从左到右进行的，比较的是每个字符的10进制数值，下面举例说明：

'hello'='HELLO'　　；得到布尔值FALSE。

- 'hello'<'zero'　　；得到布尔值TRUE。
- 'hello'>'hez'　　；得到布尔值FALSE。

SQL对字符串的数据类型也提供了三个量词：ALL，SOME和ANY,具体的含义同数字型，在这里不再说明。

④ 时间类型

时间类型的数据在数据库中实际上是以数字的形式存贮的，只是按照时间的格式给年、月、日、时、分、秒以一定的位数，因此时间类型的比较其实就是数字类型的比较。具体示例如下：

DATE ‘2004-01-01'=DATE ‘2004-01-08‘ ；得到布尔值FALSE。

DATE ‘2004-01-01'>DATE ‘2004-03-08‘ ；得到布尔值FALSE。

DATE ‘2004-05-01'<>DATE ‘2004-04-08' ；得到布尔值TRUE。

4. 其他操作

常用数据类型的其他操作主要有：

数字类型的：加、减、乘、除，分别记为+、-、*、/，这些运算和普通的数字运算完全一致。

- 日期类型的：加、减、乘、除，分别记为+、-、*、/，这些运算需要特定的约定才能合法，下面予以说明：

- **日期+间隔=日期**　　**日期-间隔=日期**
- **日期-日期=间隔**　　**时间+间隔=时间**
- **时间-间隔=时间**　　**时间-时间=间隔**
- **间隔*数字=间隔**　　**间隔/数字=间隔**

字符串最重要的其他操作是连接运算，在SQL Server中，直接用+即可，而在Oracle中，则需要用连接符 || 来完成，示例如下：

- ‘hello'+‘bob!'=‘hello bob!'

SQL语言提供了常用的布尔值算符AND、OR、NOT和IS，其中AND、OR、NOT运算和其他语言的编程一致，而IS的用法是

IS 表达式 或者 IS NOT 表达式

分别表示表达式=TRUE或者表达式=FALSE。因此，在实际的应用中，IS很少使用，一般用= 代替。

5.2.2 SQL语言的函数介绍

SQL语言的函数很多，而且不同的关系数据库产品的函数也会有些差别。常用的基于SQL server2000的字符串函数、数学函数和日期函数如表5.3所示。对于其他的数据库产品，可能会有一些差别。

数学函数	
定义	说明
Abs(numeric_expr)	求绝对值
Ceiling(numeric_expr)	大于或等于指定植的最小整数
Floor(numeric_expr)	小于或等于指定植的最大整数
Power(numeric_expr, power)	返回numeric_expr的power次幂
Round(numeric_expr, int_expr)	把数字表达式四舍五入到int_expr指定的精度
Sqrt(float_expr)	指定值的平方根

字符串函数	
定义	说明
Datalength(char_expr)	返回char_expr所包含的字符个数，不包括尾部空格
Substring(char_expr ，start，length)	返回char_expr中从start开始的length个字符的字符串
Upper(char_expr)	把字符串char_expr转换为大写
Lower(char_expr)	把字符串char_expr转换为小写
Space(int_expr)	生成int_expr个空格的字符串
Ltrim(char_expr)	删除打头的空格
Rtrim(char_expr)	删除打尾的空格
Ascii(char_expr)	返回字符串char_expr中首字符的ASCII码值
Char(int_expr)	把ASCII代码转换为字符
Str(float_expr[，length[，decimal]]	数字型转换为字符型
Charindex(char_expr，expression)	返回expression在char_expr中的起始位置，否则为零

日期函数	
定义	说明
Getdate	返回当前系统的日期和时间
Datepart(datapart， date_expr)	以整数形式返回date_expr中的指定部分
Datediff(datepart ， date_expr1，date_expr2)	以datepart指定的方式，返回date_expr2和date_expr1之间的差值
Dateadd(datepart ， number ，date_expr)	返回以datepart指定的方式表示的date_expr加上number以后的日期
Convert(datatype[(length)], expression,format)	把expression表达的日期按照format格式转换为datatype所指定的字符串，其中length为可选项，表示datatype的数据长度

5.2.3 SQL语言的语句结构

不管是操作人员直接键入的还是嵌入到宿主语言的某个程序中，SQL语句都是从动词开始的，紧跟其后的是动词“应该做什么”的确切信息。语句的末尾必须有一个结束符，SQL标准的语句为空白。不同的SQL产品使用的语句结束符也是不同的，如Oracle SQL为分号‘；’。如果动词是对某个表的动作（即动词要求使用某个表），则该表的名称必须出现在语句中。

SQL语言和其他计算机语言一样，保留一些字作为本系统专用。SQL对它的保留字规定有确切的含义和用法，用户只能按照SQL的规范使用其保留字，不能用它们做为表名、列名等。一些常用的SQL标准规定的保留字有：SELECT、FROM、WHERE、CREATE、TABLE、DROP、INSERT、UPDATE、DELETE等。

5.2.4 SQL语言的命令分类

SQL语言的命令可以分为以下四类：

- ① 数据定义语言DDL。

DDL命令用来创建数据库中的各种对象，包括模式、表、视图、索引等.

② 数据操纵语言DML

DML命令分为数据查询和数据更新两类。数据查询语言用来对已经存在于数据库中的数据按照指定的条件进行检索。数据更新又可以分为插入、删除、修改三种操作。

- ③ 数据控制语言DCL

DCL命令用来完成授予或收回访问数据库的某种特权，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动作。

④ 嵌入式SQL的使用

这一部分内容涉及到SQL语句嵌入在宿主语言程序中的使用规则。

- 限于篇幅，本章只讲述数据定义、数据操纵和数据控制语言，对于嵌入式SQL，读者请参考相关书籍。

5.2.5 SQL语法说明

在本章的SQL语句中使用了BNF标记的下列通用变型：

- ① < >
- 尖括号中是语法元素的名称。
- ② [ ]

方括号中是选择语法，可以在构造SQL语句时决定是否省略这样的语法。

③ { }

大括号中的是强制性语法组合。在构造SQL语句时必须选择包括所有的组合。

④ |

竖条将语法元素组分开，在构造SQL语句时，必须选择一个元素。

上面的四种符号(尖括号、方括号、大括号和竖条)都不是语法的一部分，不要将其包含在SQL语句中。

5.3 SQL语言的数据定义

在具体的RDBMS中，维护数据库对象一般都有两种方法，一种是用SQL语言来定义和维护数据库对象；另一种是使用RDBMS或第三方提供的图形化工具。对于初学者来说，也许用图形化的工具更加方便。但是，随着对SQL语言的熟练掌握，读者就会发现编码比使用工具更容易创建和调试数据库对象。

- SQL 的数据定义语言是用来定义和管理 SQL 数据库中的所有对象，包括数据库、表、视图、索引和存储过程等。每个对象类通常都包含 CREATE、ALTER 和 DROP 语句，如 CREATE TABLE、ALTER TABLE 和 DROP TABLE。

5.3.1 数据库的创建和删除

数据库由包含数据的表集合和其他对象（如视图、索引、存储过程等）组成，目的是为执行与数据有关的活动提供支持。存储在数据库中的数据通常与特定的主题或过程（如生产数据或者信息发布数据）相关。一般大中型的RDBMS都能够支持许多数据库，每个数据库可以存储来自其他数据库的相关或不相关数据。在创建任何数据库对象之前，必须首先创建数据库。

在SQL Server中，数据库对象至少对应两个文件：MDF文件和LDF文件，其中MDF文件用于存储数据库的所有对象；而LDF文件则存储数据库的操作日志。而在Oracle中，一般每个数据库由多个设备文件（DAT文件）组成，在这些设备文件之上可以创建数据库的其他对象。

1. 数据库的创建

SQL语言用CREATE DATABASE来创建数据库，其一般格式为：

CREATE DATABASE <数据库名>

[ON] ([NAME = 逻辑文件名 ,]

FILENAME = 操作系统文件名

[, SIZE = 初始文件大小]

[, MAXSIZE = {最大文件大小 | UNLIMITED }]

[, FILEGROWTH = 文件每次增量])

[LOG ON] ([NAME = 逻辑文件名 ,]

FILENAME = 操作系统文件名

[, SIZE = 初始文件大小]

[, MAXSIZE = {最大文件大小 | UNLIMITED }]

[, FILEGROWTH = 文件每次增量])

其中数据库名为要创建的数据库的名字，数据库名称在服务器中必须唯一，并且符合标识符的规则；ON用来指定存储数据库数据部分的磁盘文件（数据文件），LOG ON用来指定存储数据库日志部分的磁盘文件；逻辑文件名指在创建数据库后执行的SQL语句中引用文件的名称；操作系统文件名指实际物理存在的带路径的文件名；SIZE指定所定义的文件的大小；MAXSIZE指定所定义的文件可以增长到的最大；FILEGROWTH指定所定义的文件的增长增量，大小设置不能超过文件的MAXSIZE，若FILEGROWTH用UNLIMITED，则指定所定义的文件将增长到磁盘变满为止。另外，对于可选的参数（［］中的参数），如果不指定，则RDBMS自动选用缺省参数。

【例5.1】针对广告监测信息系统，我们建立相关的GGJC数据库，具体的创建语句如下：

```
CREATE DATABASE GGJC
ON (NAME = 'GGJC_DAT',
    FILENAME='C:\PROGRAM FILES\MICROSOFT SQL SERVER\MSSQL
                \DATA\GGJC_DATA.MDF ',
    SIZE = 100MB,
    MAXSIZE = 500MB,
    FILEGROWTH = 10MB )
LOG ON (NAME = 'GGJC_LOG',
    FILENAME='C:\PROGRAM FILES\MICROSOFT SQL SERVER\MSSQL
                \DATA\GGJC_LOG.LDF ',
    SIZE = 20MB,
    MAXSIZE = 100MB,
    FILEGROWTH = 5MB )
```

2. 数据库的删除

当某个数据库不再需要时，特别是测试结束，测试的数据库不再需要时，则可以使用SQL语句的DROP DATABASE删除该数据库，其一般格式为：

- **DROP DATABASE <数据库名>**

由于一旦删除数据库，则数据库中的所有对象将不再存在，因此删除数据库要特别小心。在实际应用中，由于各种原因，有些数据库会损坏或处于置疑状态（比如创建了一个数据库，然后想把其他数据库导入进来，此时很容易发生数据库的置疑），此时可以用DROP语句删除数据库。

【例5.2】针对广告监测信息系统，删除数据库GGJC的备份数据库GGJCBAK的语句如下：

- **DROP DATABASE GGJCBAK**

5.3.2 表的创建和删除

在创建了数据库之后，就可以在其上建立各种数据库表。数据库表按照使用的时效性可分为永久表和临时表。永久表在数据库的整个生命周期之内有效，而临时表顾名思义，只在某一段时间(用户创建该临时表开始到用户退出连接为止)内有效，而且只能对所创建的用户是可见的。SQL Server临时表的支持比Oracle要强些，使用和定义都比较灵活，而Oracle估计是考虑语义的严格性，在这方面不是很好用。临时表在SQL Server中用 '#'打头，后面的名字命名与普通表一致。

1. 表的创建

- SQL语言用CREATE TABLE语句来创建表，其一般格式为：

  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>])

其中表名为所要创建的表的名字，它可以有一个或者多个列属性。建表的同时通常还可以定义与该表有关的完整性约束，这些完整性约束条件被存入系统的数据字典中，当用户操作表中数据时由数据库管理系统自动检查操作是否违背这些完整性约束条件。如果完整性约束条件涉及到该表的多个属性列，则必须定义在表级上，否则既可以定义在列级上也可以定义在表级上。

定义表的每一个列的属性必须给出列名和数据类型，而对于列级的完整性约束一般包括NULL、NOT NULL、缺省值、键的约束等内容。

【例5.3】广告监测信息系统最主要的表是广告记录，它的创建语句如下：

```
CREATE TABLE ADVERTISEMENT (
    ID INT NOT NULL ,                          ；序号
    MEDIADM VARCHAR (20)  NOT NULL ,           ；媒体代码
    MEDIA VARCHAR (40)  NOT NULL ,             ；媒体名称
    PLAYTIME DATETIME NOT NULL ,               ；播出时间
    FORMDM VARCHAR (20)  NULL ,                ；播出类型代码
    FORM VARCHAR (20)  NULL ,                  ；播出类型
    CODE INT NOT NULL ,                        ；产品代码
    NAME VARCHAR (60)  NOT NULL ,              ；产品名称
    COMMODITYDM VARCHAR (20)  NOT NULL ,       ；商品代码
    TRADEMARKDM INT NOT NULL ,                 ；商标代码
    TRADEMARK VARCHAR (20)  NULL ,             ；商标
```

```
MAKERDM INT NOT NULL ,                    ；厂家代码
MAKERNAME VARCHAR (40)  NULL ,            ；厂家名称
POSITIONLXDM VARCHAR (20)  NULL ,         ；栏目类型代码
POSITION VARCHAR (40)  NULL ,             ；栏目类型
SIZE VARCHAR (20)  NULL ,          ；时长（为与报纸兼容取字符串）
BREAKDM VARCHAR (20)  NOT NULL ,          ；段位代码
BREAKMC VARCHAR (20)  NULL ,              ；段位名称
TAPENO SMALLINT NULL ,                    ；录象带号
EXPENSES DECIMAL(14, 4) NULL ,            ；费用
ISAGAINSTLAW VARCHAR (20)  NOT NULL , ；违法否（分不违法、
                                            轻度和重度违法）
AGAINSTDETIAL VARCHAR (200)  NULL ,       ；违法情况
IMAGE CHAR (8)  NULL ,                     ；图像质量
BZ VARCHAR (200)  NULL                    ；备注
)
```

其中序号ID=‘YYYYMMDDHHMISS’+‘999999’。

SQL支持NULL和NOT NULL数据。NULL表示允许取空值，一般非键属性可以作为NULL；NOT NULL表示在输入数据时，该字段必须输入数据。

另外，创建表还可以用SELECT … INTO <新表名>语句来实现，该语句我们将在数据的插入语句中予以说明。

从上面可以看出，修改表可以增加列，删除列或约束以及修改列的属性，其中删除列和删除约束的区别是删除列时需要带关键字COLUMN。在删除列时，可以带参数CASCADE或者RESTRIC，其中CASCADE表示删除该属性时同时删除引用该属性的视图和约束，而RESTRIC则表示在没有视图或约束引用该属性时才能删除该属性。每一个有关列的操作语句都可以带有多个属性列，属性列之间用‘，’分开。

- ALTER TABLE <表名>
  [ADD <新列名> <数据类型> [完整性约束条件]]
- [DROP <约束名>]
- [DROP COLUMN <列名> [CASCADE|RESTRIC]]
- [ALTER COLUMN <列名> <数据类型>]

【例5.4】在例5.3所示的ADVERTISEMENT表中，考虑到经常需要查询一段时间范围内每天的17:00—5:00的广告记录（其实相当于当天的17:00—23:59:59和第二天的00:00:00—5:00），为了统计的方便和查询速度的加快，我们增加两个字段：MYDATE和MYTIME，其中MYDATE=PLAYTIME的日期部分，MYTIME='2000-01-01'+PLAYTIME的时间部分。另外考虑到IMAGE字段没有什么用处，予以删除，同时把ISAGAINSTLAW的数据类型改为数字类型，则修改语句如下所示：

```
ALTER  TABLE  ADVERTISEMENT
ADD  MYDATE  SMALLDATETIME  NOT  NULL,
        MYTIME DATETIME NOT NULL     ；增加MYDATE和MYTIE列
ALTER  TABLE  ADVERTISEMENT
DROP  COLUMN  IMAGE                          ；删除IMAGE列
ALTER  TABLE  ADVERTISEMENT
ALTER  COLUMN ISAGAINSTLAW INT   ；修改ISAGAINSTLAW的列属性
```

3. 表的删除

当某个表不再需要时，可以使用DROP TABLE把该表删除，格式如下：

- DROP TABLE <表名> [CASCADE|RESTRIC]

其中CASCADE和RESTRIC的语义和前面的属性列的删除类似。删除表时一定要非常小心，特别是对于含有大量数据的表格的删除更加要小心，要确保万无一失。如果不是绝对有把握，则可以先备份该表的数据，然后再删除。备份表数据的方式可以使用SELECT … INTO <新表名>语句实现。

【例5.5】假设我们在数据库GGJCBAK中创建了一个调试的表CZY，现在要把它删除，则可以使用如下语句：

- DROP TABLE CZY

5.3.3 视图的创建和删除

视图是一种虚拟表，其数据是由其他表和视图中提取的，它并不包含真正的数据，也不分配任何存储空间，仅仅是一条SQL查询语句，只是查询语句返回的结果以表的形式来表示。和表一样，用户也可以在视图中插入、删除、修改、查询数据，只是有时通过视图对数据操作有一定的限制。

视图就如数据库的一个窗口，它具有如下优点：可以提高数据库对于应用程序的独立性，有利于保持数据的一致性；简化复杂的查询，可以在视图上查询数据；视图提供了一种保持数据安全性的手段，通过把视图授权给用户，可以向不同用户提供不同的视图而无需提供基本表，从而不让用户知道基本表中的某些内容，如有关财务的保密数据。

1. 视图的创建

在同一个表上可以建立多个视图，一个视图也可以建立在多个表上。视图的的建立通过CREATE VIEW 来实现，具体SQL语句的格式如下：

- CREATE VIEW <视图名> [（列名1 [，列名2] …）]
- AS 子查询

其中视图名为所要创建的视图的名字，视图名后面的参数包含了视图中各字段的名称，但也可以省略，省略时子查询的字段自动作为视图的字段。不过，当目标列中是库函数或字段表达式或者多表连接时选出几个同名字段作为视图的字段时，则在视图定义中必须给出它的字段名。一般建议列出视图的字段列表，以免引起歧义。视图定义中所罗列的字段和子查询中所罗列的顺序一一对应。

定义视图的查询子句即SELECT 语句。有关SELECT语句的详细内容见5.5节。若要从创建视图的SELECT子句所引用的对象中选择，必须具有适当的权限。视图不必是具体某个表的行和列的简单子集，可以用具有任意复杂性的 SELECT 子句，使用多个表或其他视图来创建视图。

在所有视图定义中，SELECT 语句必须是单个表的语句或带有可选聚合的多表 JOIN。对于视图定义中的 SELECT 子句有如下几个限制：

①不能包含 COMPUTE 或 COMPUTE BY 子句。

②不能包含 ORDER BY 子句。

③不能包含 INTO 关键字。

④不能引用临时表或表变量。

□ 在定义视图的SELECT语句中可以使用函数，也可以使用多个由 UNION分隔的 SELECT 语句。

【例5.6】在广告记录表中，假设部分用户只需要频繁查询某个媒体（假设媒体为西湖明珠电视台，代码为010108）下面的广告信息，内容只包括播出时间、产品、时长、栏目等信息，则我们可以建立如下视图：

CREATE VIEW v_ADVERTISEMENT_MZDS (PLAYTIME,PRODUCT,SIZE,POSITION)

AS

SELECT PLAYTIME,PRODUCT,SIZE,POSITION

FROM ADVERTISEMENT

WHERE (MEDIADM='010108')

其中视图名v_ADVERTISEMENT_MZDS中的v_表示对象为视图，后面的_MZDS表示该视图是明珠电视的广告记录。这是取名技巧，便于阅读。

假设广告监测中心经常需要关心各个媒体的所有违法广告(一般违法和严重违法)的播出时间、产品、商标、厂家、厂家地址等信息，则可建立如下视图：

```
CREATE VIEW v_ADVERTISEMENT_WF (MEDIADM，MEDIA，
      PLAYTIME,□PRODUCT, TRADEMA
□AS
□SELECT MEDIADM，MEDIA，PLA
            A.TRADEMARK，A.MAKE
□FROM ADVERTISEMENT A，MAKER B
□WHERE (A.MAKERDM=B.MAKERDM)
□        AND (A.ISAGAINSTLAW LIKE '%违法')
```

³§¼ÒMAKER	
³§¼Ò´úÂëMAKERDM	int
³§¼ÒÃû³ÆMAKERNAME	varchar(40)
Í¨Ñ¶µØÖ·TXDZ	varchar(40)
Öú¼Ç·ûMNEMONICF	varchar(20)
±¸×¢BZ	varchar(200)

2. 视图的修改

更改一个先前创建的视图（用CREATE VIEW 创建），不影响相关的存储过程，也不更改权限。修改视图用ALTER VIEW语句，其格式基本同视图的创建，具体如下：

- ALTER VIEW <视图名> [（列名1 [，列名2]…）]
- AS 子查询

其实视图的修改相当于先删除原来的视图，然后再按照新的子查询来创建新的视图。使用视图的修改语句的好处在于不需要重新设置视图的权限，也无需重新创建关联的存储过程（如果使用删除方法，则关联的存储过程也会被删除）.

【例5.7】假设在例5.6创建的视图中，用户还需要查看每一条广告的费用情况，此时可以把原来的视图做如下修改：

```
ALTER VIEW v_ADVERTISEMENT_MZDS (PLAYTIME，
                PRODUCT，SIZE，EXPENSES，POSITION)
AS
SELECT  PLAYTIME，PRODUCT，SIZE，EXPENSES，POSITION
FROM  ADVERTISEMENT  WHERE (MEDIADM='010108')
```

3. 视图的删除

视图的删除仅仅是将视图的定义删除，而对基本表及其数据没有任何影响。但要注意的是，如果基本表被删除，则由该表导出的视图也将被删除。删除视图用DROP VIEW来实现，具体格式如下：

- DROP VIEW <视图名>

【例5.8】假设要删除例5.6创建的视图v_ADVERTISEMENT_MZDS,则删除视图语句如下：

- DROP VIEW v_ADVERTISEMENT_MZDS

5.3.4 索引的创建和删除

对于一个基本表，可以根据需要建立若干索引来提供多种存取途径。正如书籍索引有助于读者更快地找到信息一样，表中的索引也有助于更快地检索到数据。特别是对于海量数据的查询，设计巧妙的索引将会大大加快其速度。一般说来，只有表或视图的所有者才能为表创建索引。索引可以随时创建，而不管表中是否有数据。

1. 索引的创建

正如视图一样，我们也可以在同一个表上建立多个索引。索引的建立通过CREATE INDEX来实现，具体SQL语句的格式如下：

CREATE [UNIQUE] [CLUSTERED|NONCLUSTERED] INDEX <索引名>

ON {表名|视图名}（列名 [ASC|DECS] [，列名 [ASC|DECS]] …）

□ [其他参数]

其中索引名为所要创建的索引的名字。关键字INDEX前面的可选参数UNIQUE表示每一个索引值只有唯一的一条记录与之对应；关键字CLUSTERED表示该索引为簇索引，表示表中记录的物理排序与索引排序相同，并且簇索引的最低一级（叶级）包含实际的数据行。一个表或视图只允许同时有一个簇索引，必须先为表或视图创建唯一簇索引，然后才能为其定义其他索引。如果没有指定UNIQUE，则创建非唯一索引；如果没有指定 CLUSTERED，则创建非簇索引。 ASC或者未指定，则索引按该列名升序排列；如果是DESC，则按降序排列。

这里需要注意的是索引名的命名问题。一般索引名的格式是：

表名_IC(NC)U(NU)_字段组合

有时使用字段组合索引名会比较长，此时可以在索引名中列出第一个和最后一个字段，中间加下划线组成。这样命名的好处是可以从索引名中知道该索引建立在哪张表上，是簇索引（C）还是非簇索引（NC），是唯一索引（U）还是非唯一索引（NU）。

CREATE INDEX ADVERTISEMENT_INCU_MEDIADMPLAYTIME

ON ADVERTISEMENT (MEDIADM, PLAYTIME)

（目的：对于大量的统计，都是针对媒体和广告的播出时间）

CREATE INDEX ADVERTISEMENT_INCU_MEDIADMMYDATEMYTIME

ON ADVERTISEMENT (MEDIADM， MYDATE，MYTIME)

（目的：有部分报表要求统计某段时间内17:00 - 5:00的广告情况，该索引可以加快这部分报表的查询速度）

2. 索引的删除

如果一个索引不需要了，就可以删除它。删除索引用DROP INDEX 来实现，具体格式如下：

DROP INDEX <索引名>

【例5.10】假设要删除例5.9创建的索引ADVERTISEMENT_INCU_MEDIADMMYDATEMYTIME，则删除索引语句如下：

DROP INDEX ADVERTISEMENT_INCU_MEDIADMMYDATEMYTIME

索引的重构是先删除然后再重新建立。

5.3.5 存储过程的创建和删除

存储过程（ stored procedure）是独立存在于表之外的数据库对象，可以由用户像函数一样调用，也可以由另外的存储过程调用。使用存储过程具有多方面的优点，具体如下：

- ①快速执行

在第一次执行之后，存储过程就驻留在内存中，而省去了重新分析、优化和编译过程，加快运行速度。

- ②减少网络通信量

存储过程可以由多至几百条的SQL语句组成，但执行它，仅用一条语句即可，所以只有少量的SQL在网络上传输。

③模块化设计

利用存储过程可以做到化整为零，各个击破。

④有限的、基于函数的表访问

通过授权，可以只允许用户通过存储过程对表进行访问，从而提高系统的安全性。

⑤减少操作出错

因为只有少量的信息在网上传递，因此可以减少出错的机会.

⑥增强一致性

如果用户仅通过自己的存储过程访问表，则可以使库表数据的访问出现高度的一致性。

⑦保证数据完整性

在一个存储过程中对某些表进行各种处理，可以保证这些表的数据完整性。

其中过程名为所要创建的存储过程的名字，存储名后面跟的是存储过程的参数，每个参数以@开始。在 CREATE PROCEDURE 语句中可以声明一个或多个参数，用户必须在执行过程时提供每个所声明参数的值（除非定义了该参数的缺省值）。OUTPUT表示该参数为返回参数，使用 OUTPUT 参数可将信息返回给调用过程。AS表示存储过程要进行的操作，后面SQL语句可以包含任意数目的合法语句。

- CREATE PROCEDURE <过程名>
- [{ @参数1 类型 } [= 缺省值] [OUTPUT]]
- { , {@参数2 类型 } [= 缺省值] [OUTPUT] }
- [, …]
- AS SQL语句

【例5.11】在广告监测信

Ã½ÌåMEDIA	
Ã½Ìå´úÂëMEDIADM	varchar(20)
Ã½ÌåÃû³ÆMEDIA	varchar(40)
µÈ¼¶LEVELDJ	int
Ò¶½ÚµãISLEAF	bit
É϶¼´úÂëPARENTDM	varchar(20)
ÏµÍ³Òå·ñSYSTEMDYF	bit
Öú¼Ç·ûMNEMONICF	varchar(20)
±¸×¢BZ	varchar(200)
fgrq	varchar(20)

具体电视台的代码，则我们可

CREATE PROCEDURE P_GETME

@MEDIADM VARCHAR(20) OU

AS

SELECT @MEDIADM=MEDIADM

FROM MEDIA

WHERE MEDIA=@MEDIA

RETURN 0

而调用上面的存储过程的SQL语句如下：

DECLARE @PPP VARCHAR(20)

EXEC P_GETMEDIADM ‘杭州电视台’ ，@PPP OUTPUT

SELECT @PPP

此时显示结果为：

010107

2. 存储过程的修改

更改一个先前创建的存储过程（用 CREATE PROCEDURE 创建），用ALTER PROCEDURE语句，其格式基本同过程的创建，具体如下：

- ALTER PROCEDURE <过程名>
- [{ @参数1 类型 } [= 缺省值] [OUTPUT]]
- { , {@参数2 类型 } [= 缺省值] [OUTPUT] }
- [,…]
- AS SQL语句

存储过程的修改也相当于先删除原来的过程，然后再按照新的SQL语句来创建新的存储过程。使用过程的修改语句的好处在于不需要重新设置过程的权限，也无需重新创建关联的存储过程和其他数据库（如果使用删除方法，则关联的存储过程和对象也会被删除）。

过程中，要求增加一个返回

Ã½ÌåMEDIA	
Ã½Ìå´úÂëMEDIADM	varchar(20)
Ã½ÌåÃû³ÆMEDIA	varchar(40)
µÈ¼¶LEVELDJ	
Ò¶½ÚµãISLEAF	
É϶¼¶´úÂëPARENTDM	
ïµÍ³¶¨Òå·ñSYSTEMDYF	
Öú¼Ç·ûMNEMONICF	
±¸×¢BZ	
fgrq	

ÀàÐÍMEDIALX	
ÀàÐÍ´úÂëMEDIALXDM	char(2)
ÀàÐÍÃû³ÆMEDIALX	varchar(40)
ïµÍ³¶¨Òå·ñSYSTEMDYF	bit
Öú¼Ç·ûMNEMONICF	varchar(20)
±¸×¢BZ	varchar(200)

- SELECT @MEDIADM=MEDIADM, @MEDIALX=MEDIALX
- FROM MEDIA A, MEDIALX B
- WHERE (MEDIA=@MEDIA) AND
  SUBSTRING(A.MEDIADM,1,2)=B.MEDIALXDM)
- RETURN 0
-

而调用上面的存储过程的SQL语句如下：

DECLARE @PPP VARCHAR(20)

DECLARE @QQQ VARCHAR(20)

EXEC P_GETMEDIADM ‘杭州电视台’ ，@PPP OUTPUT，@QQQ OUTPUT

SELECT @PPP, @QQQ

此时显示结果为：

010107 电视

3. 存储过程的删除

存储过程的用DROP PROCEDURE来实现，具体格式如下：

- DROP PROCEDURE <过程名>

【例5.13】假设要删除例5.11创建的存储过程P_GETMEDIADM，则删除语句如下：

- DROP PROCEDURE P_GETMEDIADM

SQL的数据定义语句

操作对象	操作方式		
	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	
存储过程	CREATE PROCEDURE	DROP PROCEDURE	ALTER PROCEDURE

5.4 SQL语言的数据更新

数据更新是指对已经存在的数据库表或者视图进行记录的插入、删除、修改操作。SQL语言用INSERT、UPDATE、DELETE三条语句来改变数据库中的记录行。这三条语句和SELECT语句一起构成数据操纵语言。

5.4.1 数据的插入

数据的插入包括基本插入语句、用SELECT插入数据以及用SELECT在创建新表的同时插入记录三种。

1. 基本的插入语句

- 数据插入的基本语句格式为：
- INSERT INTO <表名> [（〈字段名〉[，〈字段名〉]…）]
- VALUES（〈常量〉[，〈常量〉]…）

其中字段名是将要输入值的字段名，它们与VALUES子句中的值要一一对应。如果字段名省略，则必须由VALUES子句提供表中所有字段的值，且顺序和表的定义一致。在INSERT语句中没有指定的字段将自动赋予空值NULL或字段的缺省值，如果该字段定义为非空NOT NULL，则会出错。该语句只能向表中插入一行。

【例5.14】向媒体类型表MEDIALX中插入三条记录，分别是（ ‘01’，‘电视’ ），（ ‘02’，‘报纸’ ），（ ‘03’，‘广播’ ），则相应的SQL语句如下：

- INSERT INTO MEDIALX（MEDIALXDM，MEDIALX）
  VALUES（ ‘01’，‘电视’ ）
- INSERT INTO MEDIALX（MEDIALXDM，MEDIALX）
  VALUES（ ‘02’，‘报纸’ ）
- INSERT INTO MEDIALX（MEDIALXDM，MEDIALX）
  VALUES（ ‘03’，‘广播’ ）

2. 子查询的插入语句

利用SELECT 语句插入数据的基本语句格式为：

- **INSERT INTO <表名> [(〈字段名〉[, 〈字段名〉]…)]**
- **子查询**

该语句的特点是可以利用任意复杂的查询语句得到数据，然后插入到已经存在的数据库表中。该语句的最大优点就是一次可以插入多条语句。

【例5.15】在很多时候，当天的广告和前一天的广告是大同小异。因此，在广告记录的录入过程中，为了加快录入的速度，客户要求广告记录有拷贝功能，即把前一天的广告数据拷贝到今天，同时要求相应的日期改为今天的日期。这样，操作员在录入的时候可以节省大量重复数据的录入工作，只要做适当的调整就可以了。此时相应的SQL语句如下：

```
INSERT INTO ADVERTISEMENT (ID,MEDIADM,MEDIA,PLAYTIME,
        MYDATE, MYTIME, FORMDM, FORM, CODE, NAME，
        COMMODITYDM, POSITION, POSITIONLXDM，
        TRADEMARKDM, TRADEMARK, MAKERDM，
        MAKERNAME, BREAKDM,BREAKMC,EXPENSES,
        ISAGAINSTLAW,TAPENO)
SELECT  DATESTR1+SUBSTRING(ID,15,6), MEDIADM, MEDIA,
        DATESTR+‘ ’+CONVERT(CHAR(8), PLAYTIME,108)，
        DATESTR, MYTIME, FORMDM,FORM,CODENAME，
        COMMODITYDM, POSITION, POSITIONLXDM,
        TRADEMARKDM, TRADEMARK, MAKERDM,
        MAKERNAME, SIZE, BREAKDM, BREAKMC, EXPENSES，
        ISAGAINSTLAW, TAPENO
FROM ADVERTISEMENT
WHERE (MEDIADM=STR_MEDIADM)
    AND (PLAYTIME>=DT_BEGIN)
    AND (PLAYTIME<=DT_END)
```

其中DATESTR为当天日期的字符串形式，如‘2004-03-05’，而DATESTR1格式则为YYYYMMDD形式，如‘20040305’；STR_MEDIADM表示选中的某个媒体；DT_BEGIN= DATESTR +‘17:00:00’，即当天的17:00如‘2004-03-05 17:00:00’，而DT_END表示第二天的5点，如‘2004-03-06 05:00:00’。

在选择语句中，主要对ID和PLAYTIME做了一定的计算：ID= DATESTR1+SUBSTRING(ID,9,12)，表示原ID的年、月、日由当天的年、月、日替换而保留原时分秒和序号的不变；PLAYTIME= DATESTR+‘ ’+CONVERT(CHAR(8),PLAYTIME,108) 表示原播出时间中的年、月、日部分由新的日期替换而保留原播出时间的时、分、秒部分不变。

3. 子查询创建新表的插入语句

利用SELECT语句创建新表的插入数据的基本语句格式为：

- 子查询 INTO <表名>
- 该语句的特点是可以利用任意复杂的查询语句得到数据，然后用这些数据来创建一个新表，新表的结构与子查询的结果完全一样（查询结果的字段名或别名就是新表的字段名，查询结果的数据类型就是新表的数据类型）。新表的名字就是表名，该语句在创建新表的同时，把查询结果数据全部插入到新表中。

子查询创建新表的插入语句在调试阶段和数据载入过程中用得比较多，此时往往用作数据的备份或者加工数据的一个中间过程；另外，在编制应用系统程序过程中，建立一些复杂的数据加工过程或者设计复杂的统计报表往往也用到该语句。

【例5.16】要求创建一个临时表，并把杭州电视台（代码为‘010107’）的2004-03-01的广告数据导入临时表中，假设临时表名为#tmp_ADVERTISEMENT，包括内容为播出时间、产品代码、产品名称、播出类型、栏目等信息。此时相应的SQL语句如下：

```
SELECT PLAYTIME，CODE，NAME，FORM，POSITION
INTO #tmp_ADVERTISEMENT
FROM  ADVERTISEMENT
WHERE（MEDIADM=‘01017’）
    AND（PLAYTIME>=‘2004-03-01 17:00:00’）
    AND（PLAYTIME<=‘2004-03-02 5:00:00’）
```

需要注意的是，广告的监测是从当天的17:00到次日的5:00。

5.4.2 数据的修改

修改数据库表中的记录用UPDATE语句，具体的SQL语句格式如下：

- UDATE <表名>
- SET <字段>=<表达式1> [，<字段>=<表达式2>] …
- [WHERE 语句]

修改语句一次可以只修改一个字段，也可以同时修改很多字段；WHERE条件语句可选，如果省略，则表示修改该表中的所有记录；否则只修改满足条件的一条或多条记录。

【例5.17】操作员在一段时间后发现，某一段时间内某个产品的广告在录入时认为是合法的，现在通过调查发现全部都是严重违法的，因此需要把所有这些广告都改成严重违法。假设时间范围从2004-02-01到2004-02-15，产品代码为353（产品名称略），则相应的SQL语句为：

- UPDATE ADVERTISEMENT
- SET ISAGAINSTLAW=‘严重违法’
- WHERE （CODE=353）
- AND （MYDATE>=‘2004-02-01'）
- AND （MYDATE<=‘2004-02-15‘）

5.4.3 数据的删除

删除数据库中的记录用DELETE语句，其语句格式为：

- DELETE FROM <表名>
- [WHERE 语句]

如果带WHERE条件语句，则该语句表示删除所有满足条件的记录；反之如果省略条件语句，则删除数据库中的所有记录，但此表的定义还在数据字典中。

- 【例5.18】操作员把在录入广告信息时，使用拷贝功能把西湖明珠电视台的2004-03-06的数据拷贝到2004-03-09中后发现，该数据没有什么用，因此要予以删除，此时我们可以设计如下的删除语句：

- DELETE FROM ADVERTISEMENT
- WHERE （ MEDIADM=‘010108’ ）
- AND （ MYDATE=‘2004-03-09' ）

5.4.4 视图数据的修改

在SELECT语句中，视图基本上可以当作实际的数据库表来处理，但对于INSERT、UPDATE和DELETE操作则受到一定的限制。对视图的修改最终要转换为对基本表的修改。在RDBMS中，并非所有的视图都是可以修改的，也就是说，有些视图的修改不能唯一地转换为对基本表的修改。若一个视图是从单个基本表中导出的，并且只是去掉了基本表的某些行和列（不包括键），那么就可以执行插入、删除、修改的操作。目前RDBMS只提供对上面这种情况下的视图的数据的修改。

整个语句的含义是：根据WHERE子句的条件表达式，从基本表或视图中找出满足条件的记录行，按SELECT子句中的目标列，选出记录行中的列值形成结果表；如果有ORDER BY子句，则结果表根据指定的列名序列按升序（ASC）或者降序（DESC）进行排列；GROUP BY子句将结果按列名列分组，每个组产生结果集中的一条记录；分组的附加条件用HAVING短语给出，只有满足内部函数表达式的记录才被输出。

- SELECT语句的一般格式为：
- SELECT [DISTINCT] 目标列
- FROM 基本表|视图列
- [WHERE 条件表达式]
- [GROUP BY 列名列 [HAVING 内部函数表达式]]
  [ORDER BY 列名[ASC|DESC][，列名[ASC|DESC]…]

5.5.2 单表查询

单表查询分为简单查询和条件查询两种，而条件查询根据条件的不同又可以分为比较条件查询、范围条件查询、枚举条件查询、字符串匹配查询和空值条件查询等，下面予以详细说明。

1. 简单查询

单表的简单查询是指从单表中选出指定列，没有任何条件的约束，格式为：

- SELECT 目标列 FROM <表名>

其中目标列可以是具体的列名，也可以用*代替。当用*代替时，表示选择表中的所有列。

【例5.19】要求从媒体类型MEDIALX表中选择出媒体类型代码和媒体类型名称。此时我们可以设计如下的查询语句：

SELECT MEDIALXDM，MEDIALX FROM MEDIALX

执行结果为：

MEDIALXDM	MEDIALX
01	电视
02	报纸
03	广播

2. 带搜索条件的查询

当SELECT语句带有WHERE子句时，表示该查询为带搜索条件查询，查询结果为满足条件语句的记录集。如果要使记录集的记录不重复，则选择参数DISTINCT。下面列出几种常见的搜索条件，其中每种搜索条件都只列出了条件的基本表达式，这些表达式都可以通过AND和OR形成复合的条件表达式。

①比较条件查询

比较查询是最为常见的查询，其基本结构为：

WHERE <表达式> <比较运算符> <表达式>

其中比较运算符有=、<>、<、 <=、 >、 >=等，其中基本条件 <表达式> <比较运算符> <表达式> 可以有多个，相互之间用AND或者OR进行连接，分别表示且和或。对于数字、时间、字符串都可以进行比较。

【例5.20】要求从媒体MEDIA表中选择出杭州地区（ ‘0101’）和温州地区（ ‘0103’）的电视媒体，内容包括媒体代码和媒体名称。此时我们可以设计如下的查询语句：

SELECT MEDIADM，MEDIA FROM MEDIA

WHERE（SUBSTRING(MEDIADM，1，4)=‘0101’）

OR（SUBSTRING(MEDIADM，1，4)=‘0103’）

执行结果为：

MEDIADM	MEDIA
010101	浙江电视台新闻综合频道
010102	浙江电视台钱江都市频道
010103	浙江电视台教育科技频道
010104	浙江电视台影视文化频道
010105	浙江电视台经济生活频道
010106	浙江电视台体育健康频道
010107	杭州电视台
010108	西湖明珠电视台
010109	杭州有线综艺频道
010110	杭州有线影视频道
010301	温州电视一台

②范围条件查询

- 范围条件的语法为：

WHERE <表达式1> [NOT] BETWEEN <表达式2> AND <表达式3>

如果不选择参数NOT，则该条件表示表达式1的值在表达

【例5.21】要求从广告记录ADVERTISEMENT表中选择出在2004-03-01到2004-03-03之间的杭州电视台（代码=‘010107’）的广告记录，则有如下的查询语句：

SELECT * FROM ADVERTISEMENT

WHERE （MEDIADM=‘010107’）

- AND （MYDATE BETWEEN ‘2004-03-01’ AND ‘2004-03-03’）

③枚举条件查询

枚举条件的语法有两种形式，分别为：

- **WHERE <表达式> [NOT] IN (<取值清单>|<子查询>)**
- **WHERE <表达式> < > [=] ANY (<取值清单>|<子查询>)**

【例5.22】例5.20的SQL查询语句也可以写成如下方式：

```
SELECT MEDIADM，MEDIA
FROM MEDIA
WHERE （SUBSTRING(MEDIADM，1，4) IN （ ‘0101’， ‘0103’ ））
```

④字符串匹配条件查询

字符串匹配条件的语法为：

- **WHERE <列名> [NOT] LIKE <字符串常数>**

例如：

NAME LIKE ‘DS_’ ；表示长度为3个字符，且前两个字符为DS的名字。

NAME LIKE ‘%DS’ ；表示任何以DS结尾的的名字。

NAME LIKE ‘DS%’ ；表示任何以DS开头的名字。

NAME LIKE ‘%DS%’ ；表示含有DS字符的任何名字。

【例5.23】例5.20的SQL查询语句也可以写成如下方式：

```
SELECT MEDIADM，MEDIA
FROM MEDIA
WHERE （ MEDIADM LIKE ‘0101%’ ）
      OR （ MEDIADM LIKE ‘0103%’ ）
```

⑤空值条件查询

- 空值匹配条件的语法为：
- **WHERE <列名> IS [NOT] NULL**

在SQL中，NULL的唯一含义是未知值。NULL不能用来表示无形值、默认值、不可用值。在SQL中，由NULL引起的麻烦主要是查询条件如何取值的问题。

- 【例5.24】在表厂家MAKER中查询缺少通讯地址的厂家，其查询语句如下：
- SELECT * FROM MAKER
- WHERE (TXDZ IS NULL)

【例5.25】要求列出录入厂家MAKER的操作员信息（CZY），此时可以得到查询语句：

SELECT * FROM CZY

WHERE YHM IN （SELECT DISTINCT EDITUSER

FROM MAKER）

子查询是嵌套在另一种SELECT、INSERT、UPDATE或DELETE语句中的查询语句。在SELECT语句中使用子查询，就是在SELECT语句中先用子查询查出一个表的值，主句根据这些值再去查另一个表的内容。子查询总是在括号中，作为表达式的可选部分出现在运算符的右边，并且可以有选择地跟在ANY、ALL后面，也可以用在IN、NOT IN后。子查询语法格式与SELECT语法格式相同，但不能含有ORDER BY、COMPUTE子句及INTO关键字。（这些谓词将在后面介绍）

2. 连接查询

把两个以上的表连接起来，使查询的数据从多个表中检索取得。连接查询是关系数据库中最主要的查询功能。在SELECT的FROM子句中写上所有有关的表名（可以定义简单的别名，用以指定同名字段用），就可以得到由几个表中的数据组合而成的查询结果。为了得到感兴趣的结果，一般在WHERE子句中给出连接条件。

【例5.26】例5.25的查询语句也可以用连接查询来表示：

- SELECT CZYID，YHM，XM
- FROM CZY A，MAKER B
- WHERE（A.YHM=B.EDITUSER）
- 多表查询和子查询嵌套使用可以构造出十分复杂的查询语句。

5.5.4 聚合函数

SQL语言提供了下列聚合函数：

COUNT（*）表示计算记录的总条目数

COUNT（ [DISTINCT|ALL] 列名）对一列中的值计算个数，如果选DISTINCT则计算该字段值非重复的记录条数；ALL为缺省值，如果选ALL则相当于COUNT（*）

SUM（列名）　求某列总和

AVG（列名）　求某列均值

MAX（列名）　求某列最大值

MIN（列名）　求某列最小值

【例5.27】求2004-03-08杭州电视台（代码为 ‘010107’）的所有广告的记录数、总广告时间、费用总数、平均费用(由于时长由电视和报纸合用，类型为字符串，所以对电视时间统计需用CAST函数把字符型转化为整数型)。查询语句如下：

SELECT COUNT（*），SUM（CAST（SIZE AS INTEGER）），
SUM（EXPENSES），AVG（EXPENSES）
FROM ADVERTISEMENT
WHERE （MEDIADM=‘010107’）
AND （PLAYTIME >=‘2004-03-08 17:00:00’）
AND （PLAYTIME <=‘2004-03-09 5:00’）

5.5.5 数据分组

在实际应用中，经常需要将查询结果进行分组，然后再对每个分组进行统计，SQL语句提供了**GROUP BY**子句和**HAVING**子句来实现分组统计。具体用法举例如下。

【例5.28】国家规定在单位时间内广告时间不能超出规定的时间，因此我们先统计每个电视台在2004-03-08的广告总时长的分布情况：

```
SELECT MEDIA，SUM（CAST(SIZE AS INTEGER））
FROM ADVERTISEMENT
WHERE（PLAYTIME>=‘2004-03-08 17:00:00’）
    AND（PLAYTIME <=‘2004-03-09 5:00’）
  GROUP BY MEDIA
```

假设规定在12个小时里面广告时间不允许超过1小时，超过即为违规，那么我们使用HAVING子句可以把所有违规的电视媒体罗列出来，具体如下：

- SELECT MEDIA，SUM（CAST(SIZE AS INTEGER））
- FROM ADVERTISEMENT
- WHERE （PLAYTIME >=‘2004-03-08 17:00:00’）
- AND （PLAYTIME <=‘2004-03-09 5:00’）
- GROUP BY MEDIA
- HAVING SUM（CAST(SIZE AS INTEGER））>=60*60
- 因为广告时间以秒为单位，所以1小时需要转换为秒，用60*60代替。

5.5.6 联合操作

SQL 语言的UNION关键字允许请求两个或多个结果集合的逻辑合并，列出在各结果集合中的行。其基本语法结构为：

- **SELECT 查询语句1**
- **UNION [ALL]**
- **SELECT 查询语句2**
- **UNION [ALL]**
- **[…]**
- **[ORDER BY 列名序列]**

每一个集合都必须有和第一个结果相同数量和数据类型的列，每列的列名都是从第一个结果集合中派生出来的。同时，还可以对UNION操作的结果进行排序。需要注意的是ORDER BY子句出现在最后的SELECT语句后面，但它引用的列名或表达式是在第一个SELECT列表中。

【例5.28】要求列出2004-03-08的每个电视台的广告明细记录以及广告的总时长，可以设计如下UNION查询语句：

```
SELECT MEDIA，PLAYTIME，NAME，POSITIONLX，SIZE
FROM ADVERTISEMENT
WHERE （PLAYTIME>=‘2004-03-08 17:00:00’）
    AND （PLAYTIME <=‘2004-03-09 5:00’）
UNION
SELECT MEDIA, ‘2004-03-08’, ‘ ’, ‘ ’, SUM(CAST(SIZE AS INTEGER))
FROM ADVERTISEMENT
WHERE （PLAYTIME>=‘2004-03-08 17:00:00’）
    AND （PLAYTIME <=‘2004-03-09 5:00’）
ORDER BY MEDIA，PLAYTIME
```

5.6 SQL语言的数据控制

SQL数据控制的功能包括事务管理功能和数据保护功能，即数据库的恢复、并发控制、安全性和完整性。SQL语言的数据完整性功能主要体现在CREATE TABLE和ALTER TABLE语句中，这里主要讨论SQL语言的安全控制功能。

数据库管理系统保证数据库安全的重要措施是进行存取控制，即规定不同用户对于不同数据对象所允许进行的操作，并控制各用户只能存取他有权存取的数据。不同的用户对不同的数据一般应具有不同的操作权限。

某个用户对某类数据具有何种操作权力是由DBA决定的。DBA具有数据库管理员特权，拥有系统的全部特权。首先，DBA把授权的功能告诉系统，这是由SQL的GRANT和REVOKE语句来实现的；接着，将授权的结果存入数据字典；最后，当用户提出操作请求时，根据授权情况进行检查，来决定是执行操作请求还是拒绝执行。

5.6.1 授权

授权语句用GRANT来实现，其语法格式为：

- GRANT 权力 [，权力] … [ON 对象名] TO 用户 [，用户] …;
- [WITH GRANT OPTION] ;

【例5.29】把插入、删除、修改、查询ADVERTISEMENT表的权限授予用户蒋菲（用户名为JF），把查询MEDIA表的权限授予用户张雷（用户名为ZL），并给张雷再授权的权力。可以写出如下授权语句：

- **GRANT INSERT ，DELETE，UPDATE，SELECT**
  ON ADVERTISEMENT TO JF
- **GRANT SELECT ON MEDIA TO ZL**
  WITH GRANT OPTION

5.6.2 取消特权

授予的权力可以用REVOKE语句收回，其格式为：

REVOKE 权力[，权力]…[ON 对象名] FROM 用户[，用户]…

需要注意的是，当将用户的权力收回时，系统将自动地对同一对象的传播授予权力收回。如例5.30中张雷的权力收回时，同时收回由张雷授予其他用户的对表MEDIA查询的权力。SQL的授权机制十分灵活，用户对子集建立的基本表和视图拥有全部的操作权力。它可以用GRANT语句把某些权力授予其他用户，包括“授权”的权力。必要时，再把授予的权力用REVOKE语句收回。