

## 操作数据：SQL

### 本部分内容

- SQL 基础
- 中级 SQL
- 高级 SQL

### SQL 基础

#### 本章内容

#### SQL 介绍

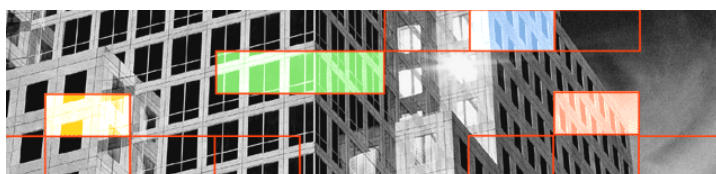
#### 使用 SELECT 语句从表中取数据

#### 创建新表


#### 字段属性


#### 向表中添加数据


#### 删除和修改表





Microsoft  
**SQL Server 2000**  
**Enterprise Edition**

 安装 SQL Server 2000 组件 (C)

 浏览安装/升级帮助 (B)

 安装 SQL Server 2000 的先决条件 (P)

 阅读发布说明 (R)

 访问我们的 Web 站点 (W)

退出 (X)

为了建立交互站点，你需要使用数据库来存储来自访问者的信息。例如，你要建立一个职业介绍服务的站点，你就需要存储诸如个人简历，所感兴趣的工作等等这样的信息。创建动态网页也需要使用数据库，如果你想显示符合来访者要求的最好的工作，你就需要从数据库中取出这份工作的信息。你将会发现，在许多情况下需要使用数据库。

在这一章里，你将学会怎样使用“结构化查询语言”（SQL）来操作数据库。SQL 语言是数据库的标准语言。在 Active Sever Pages 中，无论何时你要访问一个数据库，你就要使用 SQL 语言。因此，掌握好 SQL 对 ASP 编程是非常重要的。

**注意：**

你可以把“SQL”读作“sequel”，也可以按单个字母的读音读作 S-Q-L。两种发音都是正确的，每种发音各有大量的支持者。在本书里，认为“SQL”读作“sequel”。

通过这一章的学习，你将理解怎样用 SQL 实现数据库查询，你将学会怎样使用这种查询从数据表中取出信息，最后，你将学会怎样设计和建立自己的数据库。

**注意：**

通过下面几章对 SQL 的介绍，你将对 SQL 有足够的了解，从而可以有效地使用 Active Sever Pages。但是，SQL 是一种复杂的语言，本书不可能包括它的全部细节。要全面掌握 SQL 语言，你需要学习在 Microsoft SQL Sever 中使用 SQL。你可以到附近的书店去买一本 Microsoft SQL Sever 6.5。

## SQL 介绍：

本书假设你是在 SQL 操作 Microsoft SQL Sever 的数据库。你也可以用 SQL 操作许多其它类型的数据库。SQL 是操作数据库的标准语言。（事实上，关于 SQL 语言有一个专门的 ANSI 标准）

**注意：**

不要在你的站点上试图用 Microsoft Access 代替 Microsoft SQL Sever。SQL Sever 可以同时服务于许多用户，如果你希望你的站点有较高的访问率，MS Access 是不能胜任的。

在学习 SQL 的细节之前，你需要理解它的两大特点。一个特点容易掌握，另一个掌握起来有点困难。

第一个特点是所有 SQL 数据库中的数据都存储在表中。一个表由行和列组成。例如，下面这个简单的表包括 name 和 e-mail address：

Name	Email Address
Bill Gates	billg@microsoft.com
president Clinton	president@whitehouse.com
Stephen Walther	swalther@somewhere.com

这个表有两列（列也称为字段，域）：Name 和 Email Address。有三行，每一行包含一组数据。一行中的数据组合在一起称为一条记录。

无论何时你向表中添加新数据，你就添加了一条新记录。一个数据表可以有几十个记录，也可以有几千甚至几十亿个记录。虽然你也许永远不需要存储十亿个 Email 地址，但知道你能这样做总是好的，也许有一天你会有这样的需要。

你的数据库很有可能包含几十个表，所有存储在你数据库中的信息都被存储在这些表中。当你考虑怎样把信息存储在数据库中时，你应该考虑怎样把它们存储在表中。

SQL 的第二个特点有些难于掌握。这种语言被设计为不允许你按照某种特定的顺序来取出记录，因为这样做会降低 SQL Sever 取记录的效率。使用 SQL，你只能按查询条件来读取记录。

当考虑如何从表中取出记录时，自然会想到按记录的位置读取它们。例如，也许你会尝试通过一个循环，逐个记录地扫描，来选出特定的记录。在使用 SQL 时，你必须训练自己，不要有这种思路。

假如你想选出所有的名字是“Bill Gates”的记录，如果使用传统的编程语言，你也许会构造一个循环，逐个查看表中的记录，看名字域是否是“Bill Gates”。

这种选择记录的方法是可行的，但是效率不高。使用 SQL，你只要说，“选择所有名字域等于 Bill Gates 的记录”，SQL 就会为你选出所有符合条件的记录。SQL 会确定实现查询的最佳方法。

建设你想取出表中的前十个记录。使用传统的编程语言，你可以做一个循环，取出前十个记录后结束循环。但使用标准的 SQL 查询，这是不可能实现的。从 SQL 的角度来说，在一个表中不存在前十个记录这种概念。

开始时，当你知道你不能用 SQL 实现某些你感觉应该能实现的功能，你会受到挫折。你也许会以头撞墙甚至想写恶毒的信件给 SQL 的设计者们。但后来你会认识到，SQL 的这个特点不仅不是个限制，反而是其长处。因为 SQL 不根据位置来读取记录，它读取记录可以很快。

综上所述，SQL 有两个特点：所有数据存储在表中，从 SQL 的角度来说，表中的记录没有顺序。在下一节，你将学会怎样用 SQL 从表中选择特殊的记录。

## 使用 SQL 从表中取记录。

SQL 的主要功能之一是实现数据库查询。如果你熟悉 Internet 引擎，那么你已经熟悉查询了。你使用查询来取得满足特定条件的信息。例如，如果你想找到有 ASP 信息的全部站点，你可以连接到 Yahoo!并执行一个对 Active Sever Pages 的搜索。在你输入这个查询后，你会收到一个列表，表中包括所有其描述中包含搜索表达式的站点。

多数 Internet 引擎允许逻辑查询。在逻辑查询中，你可以包括特殊的运算符如 AND、OR 和 NOT，你使用这些运算符来选择特定的记录。例如，你可以用 AND 来限制查询结果。如果你执行一个对 Active Sever Pages AND SQL 的搜索。你将得到其描述中同时包含 Active Sever Pages 和 SQL 的记录。当你需要限制查询结果时，你可以使用 AND。

如果你需要扩展查询的结果，你可以使用逻辑操作符 OR。例如，如果你执行一个搜索，搜索所有的其描述中包含 Active Sever Pages OR SQL 的站点，你收到的列表中将包括所有其描述中同时包含两个表达式或其中任何一个表达式的站点。

如果你想从搜索结果中排除特定的站点，你可以使用 NOT。例如，查询“Active Sever Pages ” AND NOT “SQL” 将返回一个列表，列表中的站点包含 Active Sever Pages，但不包含 SQL。当必须排除特定的记录时，你可以使用 NOT。

用 SQL 执行的查询与用 Internet 搜索引擎执行的搜索非常相似。当你执行一个 SQL 查询时，通过使用包括逻辑运算符的查询条件，你可以得到一个记录列表。此时查询结果是来自一个或多个表。

SQL 查询的句法非常简单。假设有一个名为 email\_table 的表，包含名字和地址两个字段，要得到 Bill Gates 的 e\_mail 地址，你可以使用下面的查询：

```
SELECT email from email_table WHERE name="Bill Gates"
```

当这个查询执行时，就从名为 email\_table 的表中读取 Bill Gates 的 e\_mail 地址。这个简单的语句包括三部分：

■ SELECT 语句的第一部分指名要选取的列。在此例中，只有 email 列被选取。当执行 时，

只显示 email 列的值 billg@microsoft.com。

■ SELECT 语句的第二部份指明要从哪个（些）表中查询数据。在此例中，要查询的表名为 email\_table。

■ 最后，SELECT 语句的 WHERE 子句指明要选择满足什么条件的记录。在此例中，查询条件为只有 name 列的值为 Bill Gates 的记录才被选取。

Bill Gates 很有可能拥有不止一个 email 地址。如果表中包含 Bill Gates 的多个 email 地址。用上述的 SELECT 语句可以读取他所有的 email 地址。SELECT 语句从表中取出所有 name 字段值为 Bill Gates 的记录的 email 字段的值。

前面说过，查询可以在查询条件中包含逻辑运算符。假如你想读取 Bill Gates 或 Clinton 总统的所有 email 地址，你可以使用下面的查询语句：

```
SELECT email FROM email_table WHERE name="Bill Gates" OR
                                         name="president Clinton"
```

此例中的查询条件比前一个复杂了一点。这个语句从表 email\_table 中选出所有 name 列为 Bill Gates 或 president Clinton 的记录。如果表中含有 Bill Gates 或 president Clinton 的多个地址，所有的地址都被读取。

SELECT 语句的结构看起来很直观。如果你请一个朋友从一个表中为你选择一组记录，你也许以非常相似的方式提出你的要求。在 SQL SELECT 语句中，你“SELECT 特定的列 FROM 一个表 WHERE 某些列满足一个特定的条件”。

下一节将介绍怎样执行 SQL 查询来选取记录。这将帮助你熟悉用 SELECT 语句从表中取数据的各种不同方法。

## 使用 ISQL 执行 SELECT 查询

当你安装 SQL Sever 时，你同时安装了一个叫作 ISQL/w 的应用程序。ISQL/w 允许你执行交互的 SQL 查询。在把查询包括到你的 ASP 网页中之前，用 ISQL/w 对其进行测试是非常有用的。

### 注意：

在这本书的第一部份，你学习了怎样安装和配置 Microsoft SQL Sever。如果没有安装 SQL Sever 或者 SQL Sever 不能运行，请参阅第三章“安装和使用 SQL Sever”。

选择任务上 SQL Sever 程序组中的 ISQL\_w 以启动该程序。程序启动时，首先会出现一个对话框，要求输入服务器信息和登录信息（见图 10.1）。在 Sever 框中，输入你的 SQL 服务器的名字。如果服务器正运行在本地计算机上，服务器名字就是你计算机的名字。在登录信息框中，输入一个登录帐号和密码或选择使用“可信连接”，然后单击 Connect 按钮。

图 10.1

### 注意：

如果你将 SQL Sever 配置为使用完整安全或混合安全，那么你可以使用可信连接。如果你使用标准安全，你则需要提供用户帐号和密码。要了解更多信息，参见第三章。

如果一切正常，在你单击连接按钮后会出现一个查询窗口，如图 10.2 所示。（如果有异常，请

参考第三章)

图 10.2

在执行查询之前，你需要选择数据库。安装 SQL Sever 时你已为自己创建了一个数据库，SQL Sever 还有许多系统数据库，如 master，model，msdb，和 tempdb。

方便的是，SQL Sever 带有一个特殊的名为 pubs 的例子数据库。库 pubs 中包含供一个虚拟的出版商使用的各个表。文档中所有的例子程序都是针对这个库来设计的。本书中的许多例子也使用这个数据库。

在查询窗口顶部的 DB 下拉框中选择数据库 pubs，这样你就选择了数据库。你所有的查询都将针对这个库中的各个表来执行。现在你可以执行你的第一个查询了。这真让人兴奋！

你的第一个查询将针对一个名为 autrors 的表，表中包含所有为某个虚拟出版商工作的作者的相关数据。单击查询窗口并输入以下的语句：

```
SELECT phone FROM authors WHERE au_name="Ringer"
```

输入完成后，单击执行查询按钮（一个绿色三角形，看起来像 VCR 播放键）。单击此按钮后，任何出现在查询窗口中的语句均会被执行。查询窗口会自动变成结果显示窗口，你可以看到查询的结果（见图 10.3）。

你看到的查询结果也许与图 10.3 所示的不同。在 SQL Sever 的不同版本中，库 pubs 中的数据会有所不同。对 SQL Sever 6.5 来说，将会找到两条记录。结果显示窗口中应显示如下内容：

```
phone
.....
801 826_0752
801 826_0752
(2 row(s) affected)
```

图 10.3

你所执行的 SELECT 语句从表 authors 中取出所有名字为 Ringer 的作者的电话号码。你通过在 WHERE 子句中使用特殊的选择条件来限制查询的结果。你也可以忽略选择条件，从表中取出所有作者的电话号码。要做到这一点，单击 Query 标签，返回到查询窗口，输入以下的 SELECT 语句：

```
SELECT Phone FROM authors
```

这个查询执行后，会取出表 authors 中的所有电话号码（没有特定的顺序）。如果表 authors 中包含一百个电话号码，会有一百个记录被取出，如果表中有十亿个电话号码，这十亿条记录都会被取出（这也许需要一些时间）。

表 authrs 的字段包括姓，名字，电话号码，地址，城市，州和邮政编码。通过在 SELECT 语句

的第一部份指定它们，你可以从表中取出任何一个字段。你可以在一个 SELECT 语句中一次取出多个字段，比如：

```
SELECT au_fname ,au_lname, phone FROM authors
```

这个 SELECT 语句执行后，将取出这三个列的所有值。下面是这个查询的结果的一个示例（为了节省纸张，只显示查询结果的一部分，其余记录用省略号代替）：

au_fname	au_lname	phone
Johnson	White	408 496_7223
Marjorie	Green	415 986_7020
Cheryl	Carson	415 548_7723
Michael	O'Leary	408 286_2428
...		

(23 row(s) affected)

在 SELECT 语句中，你需要列出多少个字段，你就可以列出多少。不要忘了把字段名用逗号隔开。你也可以用星号（\*）从一个表中取出所有的字段。这里有一个使用星号的例子：

```
SELECT * FROM authors
```

这个 SELECT 语句执行后，表中的所有字段的值都被取出。你会发现你将在 SQL 查询中频繁使用星号。

#### 技巧：

你可以使用星号来查看一个表的所有列的名字。要做到这一点，只需要在执行完 SELECT 语句后看一下查询结果的列标题。

## 操作多个表

到现在为止，你只尝试了用一句 SQL 查询从一个表中取出数据。你也可以用一个 SELECT 语句同时从多个表中取出数据，只需在 SELECT 语句的 FROM 从句中列出要从中取出数据的表名称即可：

```
SELECT au_lname ,title FROM authors, titles
```

这个 SELECT 语句执行时，同时从表 authors 和表 titles 中取出数据。从表 authors 中取出所有的作者名字，从表 titles 中取出所有的书名。在 ISQL/w 程序中执行这个查询，看一下查询结果。你会发现一些奇怪的出乎意料的情况：作者的名字并没有和它们所著的书相匹配，而是出现了作者名字和书名的所有可能的组合，这也许不是你所希望见到的。

出了什么差错？问题在于你没有指明这两个表之间的关系。你没有通过任何方式告诉 SQL 如何把表和表关联在一起。由于不知道如何关联两个表，服务器只能简单地返回取自两个表中的记录的所有可能组合。

要从两个表中选出有意义的记录组合，你需要通过建立两表中字段的关系来关联两个表。要做到这一点的途径之一是创建第三个表，专门用来描述另外两个表的字段之间的关系。

表 authors 有一个名为 au\_id 的字段，包含有每个作者的唯一标识。表 titles 有一个名为

title\_id 的字段，包含每个书名的唯一标识。如果你能在字段 au\_id 和字段 title\_id 之间建立一个关系，你就可以关联这两个表。数据库 pubs 中有一个名为 titleauthor 的表，正是用来完成这个工作。表中的每个记录包括两个字段，用来把表 titles 和表 authors 关联在一起。下面的 SELECT 语句使用了这三个表以得到正确的结果：

```
SELECT  au_name,title FROM authors,titles,titleauthor
        WHERE  authors.au_id=titleauthor.au_id
        AND    titles.title_id=titleauthor.title_id
```

当这个 SELECT 语句执行时，每个作者都将与正确的书名相匹配。表 titleauthor 指明了表 authors 和表 titles 的关系，它通过包含分别来自两个表的各一个字段实现这一点。第三个表的唯一目的是在另外两个表的字段之间建立关系。它本身不包含任何附加数据。

注意在这个例子中字段名是如何书写的。为了区别表 authors 和表 titles 中相同的字段名 au\_id，每个字段名前面都加上了表名前缀和一个句号。名为 author.au\_id 的字段属于表 authors，名为 titleauthor.au\_id 的字段属于表 titleauthor，两者不会混淆。

通过使用第三个表，你可以在两个表的字段之间建立各种类型的关系。例如，一个作者也许写了许多不同的书，或者一本书也许由许多不同的作者共同完成。当两个表的字段之间有这种“多对多”的关系时，你需要使用第三个表来指明这种关系。

但是，在许多情况下，两个表之间的关系并不复杂。比如你需要指明表 titles 和表 publishers 之间的关系。因为一个书名不可能与多个出版商相匹配，你不需要通过第三个表来指明这两个表之间的关系。要指明表 titles 和表 publishers 之间的关系，你只要让这两个表有一个公共的字段就可以了。在数据库 pubs 中，表 titles 和表 publishers 都有一个名为 pub\_id 的字段。如果你想得到书名及其出版商的一个列表，你可以使用如下的语句：

```
SELECT  title, pub_name FROM  titles, publishers
        WHERE titles.pub_id=publishers.pub_id
```

当然，如果一本书是由两个出版商联合出版的，那么你需要第三个表来代表这种关系。

通常，当你预先知道两个表的字段间存在“多对多”关系时，就使用第三个表来关联这两个表。反之，如果两个表的字段间只有“一对一”或“一对多”关系，你可以使用公共字段来关联它们。

## 操作字段

通常，当你从一个表中取出字段值时，该值与创建该表时所定义的字段名联系在一起。如果你从表 authors 中选择所有的作者名字，所有的值将会与字段名 au\_lname 相联系。但是在某些情况下，你需要对字段名进行操作。在 SELECT 语句中，你可以在缺省字段名后面仅跟一个新名字来取代它。例如，可以用一个更直观易读的名字 Author Last Name 来代替字段名 au\_lname：

```
SELECT au_lname "Author Last Name" FROM authors
```

当这个 SELECT 语句执行时，来自字段 au\_lname 的值会与“Author Last Name”相联系。查询结果可能是这样：

```
Author Last Name
.....
White
Green
```

```

Carson
O'Leary
Straight
...
(23 row(s) affected)

```

注意字段标题不再是 au\_lname，而是被 Author Last Name 所取代。

你也可以通过执行运算，来操作从一个表返回的字段值。例如，如果你想把表 titles 中的所有书的价格加倍，你可以使用下面的 SELECT 语句：

```
SELECT price*2 FROM titles
```

当这个查询执行时，每本书的价格从表中取出时都会加倍。但是，通过这种途径操作字段不会改变存储在表中的书价。对字段的运算只会影响 SELECT 语句的输出，而不会影响表中的数据。为了同时显示书的原始价格和涨价后的新价格，你可以使用下面的查询：

```
SELECT price "Original price", price*2 "New price" FROM titles
```

当数据从表 titles 中取出时，原始价格显示在标题 Original price 下面，加倍后的价格显示在标题 New price 下面。结果可能是这样：

```

original price          new price
.....
19.99                  39.98
11.95                  23.90
2.99                   5.98
19.99                  39.98
...
(18 row(s) affected)

```

你可以使用大多数标准的数学运算符来操作字段值，如加 (+)，减 (-)，乘 (\*) 和除 (/)。你也可以一次对多个字段进行运算，例如：

```
SELECT price*ytd_sales "total revenue" FROM titles
```

在这个例子中，通过把价格与销售量相乘，计算出了每种书的总销售额。这个 SELECT 语句的结果将是这样的：

```

total revenue
.....
81,859,05
46,318,20
55,978,78
81,859,05
40,619,68
...

```



```
(18 row(s) affected)
```

最后，你还可以使用连接运算符（它看起来像个加号）来连接两个字符型字段：

```
SELECT au_fname+" "+au_lname "author name" FROM authors
```

在这个例子中，你把字段 `au_fname` 和字段 `au_lname` 粘贴在一起，中间用一个逗号 隔开，并把查询结果的标题指定为 `author name`。这个语句的执行结果将是这样的：

```
author names
.....
Johnson White
Marjorie Green
Cheryl Carson
Michael O'Leary
Dean Straight
...
(23 row(s) affected)
```

可以看到，SQL 为你提供了对查询结果的许多控制。你应该在 ASP 编程过程中充分利用这些优点。使用 SQL 来操作查询结果几乎总是比使用有同样作用的脚本效率更高。

## 排序查询结果

本章的介绍中曾强调过，SQL 表没有内在的顺序。例如，从一个表中取第二个记录是没有意义的。从 SQL 的角度看来，没有一个记录在任何其他记录之前。

然而，你可以操纵一个 SQL 查询结果的顺序。在缺省情况下，当记录从表中取出时，记录不以特定的顺序出现。例如，当从表 `authors` 中取出字段 `au_lname` 时，查询结果显示成这样：

```
au_lname
.....
White
Green
Carson
O'Leary
Straight
...
(23 row(s) affected)
```

看一列没有特定顺序的名字是很不方便的。如果把这些名字按字母顺序排列，读起来就会容易得多。通过使用 `ORDER BY` 子句，你可以强制一个查询结果按升序排列，就像这样：

```
SELECT au_lname FROM authors ORDER BY au_lname
```

当这个 `SELECT` 语句执行时，作者名字的显示将按字母顺序排列。`ORDER BY` 子句将作者名字按升序排列。

你也可以同时对多个列使用 ORDER BY 子句。例如，如果你想同时按升序显示字段 au\_lname 和字段 au\_fname，你需要对两个字段都进行排序：

```
SELECT au_lname, au_fname FROM authors ORDER BY au_lname , au_fname
```

这个查询首先把结果按 au\_lname 字段进行排序，然后按字段 au\_fname 排序。记录将按如下的顺序取出：

au_lname	au_fname
.....	
Bennet	Abraham
Ringer	Albert
Ringer	Anne
Smith	Meander
...	
(23 row(s) affected)	

注意有两个作者有相同的名字 Ringer。名为 Albert Ringer 的作者出现名为 Anne Ringer 的作者之前，这是因为姓 Albert 按字母顺序应排在姓 Anne 之前。如果你想把查询结果按相反的顺序排列，你可以使用关键字 DESC。关键字 DESC 把查询结果按降序排列，如下例所示：

```
SELECT au_lname, au_fname FROM authors
WHERE au_lname="Ringer" ORDER BY au_lname , au_fname DESC
```

这个查询从表 authors 中取出所有名字为 Ringer 的作者记录。ORDER BY 子句根据作者的名字和姓，将查询结果按降序排列。结果是这样的：

au_lname	au_fname
.....	
Ringer	Anne
Ringer	Albert
(2 row(s) affected)	

注意在这个表中，姓 Anne 出现在姓 Albert 之前。作者名字按降序显示。你也可以按数值型字段对一个查询结果进行排序。例如，如果你想按降序取出所有书的价格，你可以使用如下的 SQL 查询：

```
SELECT price FROM titles ORDER BY price DESC
```

这个 SELECT 语句从表中取出所有书的价格，显示结果时，价格低的书先显示，价格高的书后显示。

#### 警告：

不是特别需要时，不要对查询结果进行排序，因为服务器完成这项工作要费些力气。这意味着带有 ORDER BY 子句的 SELECT 语句执行起来比一般的 SELECT 语句花的时间长。

## 取出互不相同的记录

一个表有可能在同一列中有重复的值。例如，数据库 pubs 的表 authors 中有两个作者的名字是 Ringer。如果你从这个表中取出所有的名字，名字 Ringer 将会显示两次。

在特定情况下，你可能只有兴趣从一个表中取出互不相同的值。如果一个字段有重复的值，你也许希望每个值只被选取一次，你可以使用关键字 DISTINCT 来做到这一点：

```
SELECT DISTINCT au_lname FROM authors WHERE au_lname="Ringer"
```

当这个 SELECT 语句执行时，只返回一个记录。通过在 SELECT 语句中包含关键字 DISTINCT，你可以删除所有重复的值。例如，假设有一个关于新闻组信息发布的表，你想取出所有曾在这个新闻组中发布信息的人的名字，那么你可以使用关键字 DISTINCT。每个用户的名字只取一次——尽管有的用户发布了不止一篇信息。

### 警告：

如同 ORDER BY 子句一样，强制服务器返回互不相同的值也会增加运行开销。福气不得不花费一些时间来完成这项工作。因此，不是必须的时候不要使用关键字 DISTINCT。

## 创建新表

前面说过，数据库中的所有数据存储在表中。数据表包括行和列。列决定了表中数据的类型。行包含了实际的数据。

例如，数据库 pubs 中的表 authors 有九个字段。其中的一个字段名为 au\_lname，这个字段被用来存储作者的名字信息。每次向这个表中添加新作者时，作者名字就被添加到这个字段，产生一条新记录。

通过定义字段，你可以创建一个新表。每个字段有一个名字和一个特定的数据类型（数据类型在后面的“字段类型”一节中讲述），例如字段 au\_lname 存储的是字符型数据。一个字段也可以存储其它类型的数据。

使用 SQL Sever，创建一个新表的方法是很多的。你可以可执行一个 SQL 语句或使用 SQL 事务管理器（SQL Enterprise Manager）来创建一个新表。在下一节里，你将学会如何用 SQL 语句来创建一个新表。

## 用 SQL 创建新表

### 注意：

如果你还没有建立自己的数据库，现在就跳回到第三章创建这个库。你绝不能向 master, tempdb 或任何其他任何系统数据库中添加数据。

从 SQL Sever 程序组（在任务栏中）中启动 ISQL/w 程序。出现查询窗口后，从窗口顶部的下拉列表中选择你在第三章所创建的数据库。下一步，在查询窗口中键入下面的 SQL 语句，单击执行查询按钮，执行这个语句：

```
CREATE TABLE guestbook (visitor VARCHAR(40), comments TEXT, entrydate
DATETIME)
```

如果一切正常，你会在结果窗口中看到如下的文字（如果出现异常，请参阅第三章）：

```
This command did not return data ,and it did not return any rows
```

祝贺你，你已经建立了你的第一个表！

你所创建的表名为 `guestbook`，你可以使用这个表来存储来访问你站点访问者的信息。你是用 `CREATE TABLE` 语句创建的这个表，这个语句有两部分：第一部份指定表的名字；第二部份是括在括号中的各字段的名称和属性，相互之间用逗号隔开。

表 `guestbook` 有三个字段：`visitor`、`comments` 和 `entrydate`。`visitor` 字段存储访问者的名字，`comments` 字段存储访问者对你站点的意见，`entrydate` 字段存储访问者访问你站点的日期和时间。

注意每个字段名后面都跟有一个专门的表达式。例如，字段名 `comments` 后面跟有表达式 `TEXT`。这个表达式指定了字段的数据类型。数据类型决定了一个字段可以存储什么样的数据。因为字段 `comments` 包含文本信息，其数据类型定义为文本型。

字段有许多不同的数据类型。下一小节讲述 SQL 所支持的一些重要的数据类型。

## 字段类型

不同的字段类型用来存放不同类型的数据。创建和使用表时，更应该理解五种常用的字段类型：字符型，文本型，数值型，逻辑性和日期型。

## 字符型数据

字符型数据非常有用。当你需要存储短的字符串信息时，你总是要用到字符型数据。例如，你可以把从 HTML form 的文本框中搜集到的信息放在字符型字段中。

要建立一个字段用来存放可变长度的字符串信息，你可以使用表达式 `VARCHAR`。考虑你前面创建的表 `guestbook`：

```
CREATE TABLE guestbook (visitor VARCHAR(40),comments TEXT,entrydate
                           DATETIME)
```

在这个例子中，字段 `visitor` 的数据类型为 `VARCHAR`。注意跟在数据类型后面的括号中的数字。这个数字指定了这个字段所允许存放的字符串的最大长度。在这个例子中，字段 `visitor` 能存放的字符串最长为四十个字符。如果名字太长，字符串会被截断，只保留四十个字符。

`VARCHAR` 类型可以存储的字符串最长为 255 个字符。要存储更长的字符串数据，可以使用文本型数据（下一节中讲述）。

另一种字符型数据用来存储固定长度的字符数据。下面是一个使用这种数据类型的例子：

```
CREATE TABLE guestbook (visitor CHAR(40),comments TEXT,entrydate
                           DATETIME)
```

在这个例子中，字段 `visitor` 被用来存储四十个字符的固定长度字符串。表达式 `CHAR` 指定了这个字段应该是固定长度的字符串。

`VARCHAR` 型和 `CHAR` 型数据的这个差别是细微的，但是非常重要。假如你向一个长度为四十个字符的 `VARCHAR` 型字段中输入数据 `Bill Gates`。当你以后从这个字段中取出此数据时，你取出的数据其长度为十个字符——字符串 `Bill Gates` 的长度。

现在假如你把字符串输入一个长度为四十个字符的 `CHAR` 型字段中，那么当你取出数据时，所取出的数据长度将是四十个字符。字符串的后面会被附加多余的空格。

当你建立自己的站点时,你会发现使用 VARCHAR 型字段要比 CHAR 型字段方便的多。使用 VARCHAR 型字段时,你不需要为剪掉你数据中多余的空格而操心。

VARCHAR 型字段的另一个突出的好处是它可以比 CHAR 型字段占用更少的内存和硬盘空间。当你的数据库很大时,这种内存和磁盘空间的节省会变得非常重要。

## 文本型数据

字符型数据限制了字符串的长度不能超过 2 55 个字符。而使用文本型数据,你可以存放超过二十亿个字符的字符串。当你需要存储大串的字符时,应该使用文本型数据。

这里有一个使用文本型数据的例子:

```
CREATE TABLE guestbook (visitor VARCHAR(40),comments TEXT,entrydate
                           DATETIME)
```

在这个例子中,字段 comments 被用来存放访问者对你站点的意见。注意文本型数据没有长度,而上一节中所讲的字符型数据是有长度的。一个文本型字段中的数据通常要么为空,要么很大。

当你从 HTML form 的多行文本编辑框(TEXTAREA)中收集数据时,你应该把收集的信息存储于文本型字段中。但是,无论何时,只要你能避免使用文本型字段,你就应该不适用它。文本型字段既大且慢,滥用文本型字段会使服务器速度变慢。文本型字段还会吃掉大量的磁盘空间。

### 警告:

一旦你向文本型字段中输入了任何数据(甚至是空值),就会有 2K 的空间被自动分配给该数据。除非删除该记录,否则你无法收回这部分存储空间。

## 数值型数据

SQL Sever 支持许多种不同的数值型数据。你可以存储整数、小数、和钱数。

通常,当你需要在表中的存放数字时,你要使用整型(INT)数据。INT 型数据的表数范围是从 -2, 147, 483, 647 到 2, 147, 483, 647 的整数。下面是一个如何使用 INT 型数据的例子:

```
CREATE TABLE visitlog (visitor VARCHAR(40),numvisits INT)
```

这个表可以用来记录你站点被访问的次数。只要没有人访问你的站点超过 2, 147, 483, 647 次, nubvisits 字段就可以存储访问次数。

为了节省内存空间,你可以使用 SMALLINT 型数据。SMALLINT 型数据可以存储从 -32768 到 32768 的整数。这种数据类型的使用方法与 INT 型完全相同。

最后,如果你实在需要节省空间,你可以使用 TINYINT 型数据。同样,这种类型的使用方法与 INT 型相同,不同的是这种类型的字段只能存储从 0 到 255 的整数。TINYINT 型字段不能用来存储负数。

通常,为了节省空间,应该尽可能的使用最小的整型数据。一个 TINYINT 型数据只占用一个字节;一个 INT 型数据占用四个字节。这看起来似乎差别不大,但是在比较大的表中,字节数的增长是很快的。另一方面,一旦你已经创建了一个字段,要修改它是很困难的。因此,为安全起见,你应该预测以下,一个字段所需要存储的数值最大有可能是多大,然后选择适当的数据类型。

为了能对字段所存放的数据有更多的控制,你可以使用 NUMERIC 型数据来同时表示一个数的整数部分和小数部分。NUMERIC 型数据使你能表示非常大的数——比 INT 型数据要大得多。一个 NUMERIC 型字段可以存储从  $-10^{38}$  到  $10^{38}$  范围内的数。NUMERIC 型数据还使你能表示有小数部分的数。例如,你可以在 NUMERIC 型字段中存储小数 3.14。

当定义一个 NUMERIC 型字段时，你需要同时指定整数部分的大小和小数部分的大小。这里有一个使用这种数据类型的例子：

```
CREATE TABLE numeric_data (bignumber NUMERIC(28,0),  
                             fraction    NUMERIC (5,4) )
```

当这个语句执行时，将创建一个名为 numeric\_data 的包含两个字段的表。字段 bignumber 可以存储直到 28 位的整数。字段 fraction 可以存储有五位整数部分和四位小数部分的小数。

一个 NUMERIC 型数据的整数部分最大只能有 28 位，小数部分的位数必须小于或等于整数部分的位数，小数部分可以是零。

你可以使用 INT 型或 NUMERIC 型数据来存储钱数。但是，专门有另外两种数据类型用于此目的。如果你希望你的网点能挣很多钱，你可以使用 MONEY 型数据。如果你的野心不大，你可以使用 SMALLMONEY 型数据。MONEY 型数据可以存储从-922, 337, 203, 685, 477.5808 到 922, 337, 203, 685, 477.5807 的钱数。如果你需要存储比这还大的金额，你可以使用 NUMERIC 型数据。

SMALLMONEY 型数据只能存储从-214, 748.3648 到 214, 748.3647 的钱数。同样，如果可以的话，你应该用 SMALLMONEY 型来代替 MONEY 型数据，以节省空间。下面的例子显示了如何使用这两种表示钱的数据类型：

```
CREATE TABLE products (product VARCHAR(40),price MONEY,  
                        Discount_price SMALLMONEY)
```

这个表可以用来存储商品的折扣和普通售价。字段 price 的数据类型是 MONEY，字段 discount\_price 的数据类型是 SMALLMONEY。

## 存储逻辑值

如果你使用复选框 (CHECKBOX) 从网页中搜集信息，你可以把此信息存储在 BIT 型字段中。BIT 型字段只能取两个值：0 或 1。这里有一个如何使用这种字段的例子：

```
CREATE TABLE opinion (visitor VARCHAR(40),good BIT)
```

这个表可以用来存放对你的网点进行民意调查所得的信息。访问者可以投票表示他们是否喜欢你的网点。如果他们投 YES，就在 BIT 型字段中存入 1。反之，如果他们投 NO，就在字段中存入 0（在下一章里，你将学会如何计算投票）。

当心，在你创建好一个表之后，你不能向表中添加 BIT 型字段。如果你打算在一个表中包含 BIT 型字段，你必须在创建表时完成。

## 存储日期和时间

当你建立一个网点时，你也许需要记录在一段时间内的访问者数量。为了能够存储日期和时间，你需要使用 DATETIME 型数据，如下例所示：

```
CREATE TABL visitorlog( visitor VARCHAR (40), arrivaltime DATETIME ,  
                       departuretime DATETIME)
```

这个表可以用来记录访问者进入和离开你网点的日期和时间。一个 DATETIME 型的字段可以存储

的日期范围是从 1 7 5 3 年 1 月 1 日第一毫秒到 9999 年 12 月 31 日最后一毫秒。

如果你不需要覆盖这么大范围的日期和时间,你可以使用 SMALLDATETIME 型数据。它与 DATETIME 型数据同样使用,只不过它能表示的日期和时间范围比 DATETIME 型数据小,而且不如 DATETIME 型数据精确。一个 SMALLDATETIME 型的字段能够存储从 1 9 0 0 年 1 月 1 日到 2 0 7 9 年 6 月 6 日的日期,它只能精确到秒。

DATETIME 型字段在你输入日期和时间之前并不包含实际的数据,认识这一点是重要的。在下一章,你将学习怎样使用大量的 SQL 函数来读取和操作日期和时间(参见下面的“缺省值”一节)。你也可以在 VBScript 和 JScript 中使用日期和时间函数来向一个 DATETIME 型字段中输入日期和时间。

## 字段属性

上一节介绍了如何建立包含不同类型字段的表。在这一节中,你将学会如何使用字段的三个属性。这些属性允许你控制空值,缺省值和标识值。

## 允许和禁止空值

大多数字段可以接受空值(NULL)。当一个字段接受了空值后,如果你不改变它,它将一直保持空值。空值(NULL)和零是不同的,严格的说,空值表示没有任何值。为了允许一个字段接受空值,你要在字段定义的后面使用表达式 NULL。例如,下面的表中两个字段都允许接受空值:

```
CREATE TABLE empty (empty1 CHAR (40) NULL,empty2 INT NULL(
```

### 注意:

BIT 型数据不能是空值。一个这种类型的字段必须取 0 或者 1。

有时你需要禁止一个字段使用空值。例如,假设有一个表存储着信用卡号码和信用卡有效日期,你不会希望有人输入一个信用卡号码但不输入有效日期。为了强制两个字段都输入数据,你可以用下面的方法建立这个表:

```
CREATE TABLE creditcards (creditcard_number CHAR(20) NOT NULL,
                           Creditcard_expire DATETIME NOT NULL)
```

注意字段定义的后面跟有表达式 NOT NULL。通过包含表达式 NOT NULL,你可以禁止任何人只在一个字段中插入数据,而不输入另一个字段的数据。

你将会发现,在你建设自己的网点过程中,这种禁止空值的能力是非常有用的。如果你指定一个字段不能接受空值,那么当你试图输入一个空值时,会有错误警告。这些错误警告可以为程序调试提供有价值的线索。

## 缺省值

假设有一个存储地址信息的表,这个表的字段包括街道、城市、州、邮政编码和国家。如果你预计地址的大部分是在美国,你可以把这个值作为 country 字段的缺省值。

为了在创建一个表时指定缺省值,你可以使用表达式 DEFAULT。请看下面这个在创建表时使用缺省值的例子:

```
CREATE TABLE addresses (street VARCHAR(60) NULL,
```

```
city VARCHAR(40) NULL,
state VARCHAR(20) NULL
zip VARCHAR(20) NULL,
country VARCHAR(30) DEFAULT 'USA')
```

在这个例子中，字段 `country` 的缺省值被指定为美国。注意单引号的使用，引号指明这是字符型数据。为了给非字符型的字段指定缺省值，不要把该值扩在引号中：

```
CREATE TABLE orders(price MONEY DEFAULT $38.00,
quantity INT DEFAULT 50,
entrydate DATETIME DEFAULT GETDATE())
```

在这个 `CREATE TABLE` 语句中，每个字段都指定了一个缺省值。注意 `DATETIME` 型字段 `entrydate` 所指定的缺省值，该缺省值是函数 `Getdate()` 的返回值，该函数返回当前的日期和时间。

## 标识字段

每个表可以有一个也只能有一个标识字段。一个标识字段是唯一标识表中每条记录的特殊字段。例如，数据库 `pubs` 中的表 `jobs` 包含了一个唯一标识每个工作标识字段：

job_id	job_desc
1	New Hire Job not specified
2	Chief Executive officer
3	Bushness Operations Manager
4	Chief Financial Officier
5	Publisher

字段 `job_id` 为每个工作提供了唯一的一个数字。如果你决定增加一个新工作，新增记录的 `job_id` 字段会被自动赋给一个新的唯一值。

为了建立一个标识字段，你只需在字段定义后面加上表达式 `IDENTITY` 即可。你只能把 `NUMERIC` 型或 `INT` 型字段设为标识字段，这里有一个例子：

```
CREATE TABLE visitorID (theid NUMERIC(18) IDENTITY, name VARCHAR(40))
```

这个语句所创建的表包含一个名为 `theid` 的标识字段。每当一个新的访问者名字添加到这个表中时，这个字段就被自动赋给一个新值。你可以用这个表为你的站点的每一个用户提供唯一标识。

### 技巧：

建立一个标示字段时，注意使用足够大的数据类型。例如你使用 `TINYINT` 型数据，那么你只能向表中添加 255 个记录。如果你预计一个表可能会变得很大，你应该使用 `NUMERIC` 型数据。

标识字段的存在会使你想尝试许多不可能的事情。例如，你也许想利用标识字段来对记录进行基于它们在表中位置的运算。你应该抛弃这种意图。每个记录的标识字段的值是互不相同的，但是，这并不禁止一个标识字段的标识数字之间存在间隔。例如，你永远不要试图利用一个表的标识字段来取出表中的前十个记录。这种操作会导致失败，比如说 6 号记录和 7 号记录根本不存在。



## 使用 SQL 事务管理器创建新表

你可以使用前面几节所讲的方法创建新表。但是，使用事务管理器创建新表会更容易。这一节介绍如何使用这个程序创建新表。

从任务栏的 SQL Sever 程序组中选择 SQL Enterprise Manager，启动该程序，你会看到如图 10.4 所示的窗口。浏览服务管理器窗口中的树形结构，选择名为 Database 的文件夹。打开文件夹 Database 后，选择你在第三章中所建立的数据库。

### 注意：

如果你还没有创建自己的数据库，回到第三章创建它。你决不要向 master, tempdb 或任何其它系统数据库中添加数据。

在选择了数据库之后，你会看到一个名为 Group/users 的文件夹和一个名为 objects 的文件夹。打开文件夹 objects，你会看到许多文件夹，其中一个名为 Tables。用右键单击文件夹 Tables 并选择 New table，就会出现如图 10.5 所示的窗口。

你可以使用 Manager Tables 窗口来创建一个新表。Manager Tables 窗口有 7 个列：Key, Column, Name, Datatype, Size, Nulls 和 Default。Manager Tables 窗口中的每一行标明表中一个字段的信息。

图 10.4  
10.5

要建立一个新表，你至少要输入一行信息。在名为 Column Name 的列下面键入 mycolumn。下一步，选择 Datatype 列，并从下拉列表中选择 CHAR。当你在这两个列中输入信息后，窗口将是如图 10.6 所示的样子。

图 10.6

你已经建立了一个只有一个字段的简单的表。单击保存按钮保存这个新表。当要求你输入新表的名字时，输入 mytable 并单击 OK。现在这个表已经保存到了你的数据库中。如果你打开服务管理器窗口中的文件夹 Tables，你会看到你所建立的新表被列出。你可以双击该表的图表来编辑它，这时 Manager Tables 窗口会重新出现，你可以增加新的字段并重新保存。

用 SQL 事务管理器可以做的工作，你都可以用 SQL 语句来实现。但是，事务管理器使得建表过程变得更加简单。

## 向表中添加数据

下一章将讨论如何使用 SQL 向一个表中插入数据。但是，如果你需要向一个表中添加许多条记录，使用 SQL 语句输入数据是很不方便的。幸运的是，Microsoft SQL Sever 带有一个称为 Microsoft Query 的客户端应用程序，这个程序使得向表中添加数据变得容易了。

启动位于任务栏 SQL Sever 程序组中的 Microsoft Query 程序。从窗口顶部的菜单中选择 File|New Query。这时会显示一个 Select Data Source 对话框（见图 10.7）。选择你的数据源名字并单击 Use。

图 10.7

输入你的登录帐号和密码后，程序要求你选择一个表和一个数据库。选择你在上一节中所建立的表（mytable），单击按钮 Add，然后单击按钮 Close 关闭该对话框。

在窗口的左上角会出现一个对话框，框中是取自表 mytable 的一列字段名。你可以双击任何一个字段，把它添加到主窗口中。如果你双击星号（\*）字符，所有的字段都会被添加到主窗口中。

如果你的表中有记录，它们现在已经出现在主窗口的字段标题下面了。但是，因为你刚刚建立了这个表，表还是空的。要添加新记录，选择 Records|Allow Editing，主窗口中就会出现一条新记录。输入一行数据完成这个记录，就向表中添加了一条新记录。

图 10.8

当你转到下一条新记录时，你向上一条记录中输入的值会自动被保存。如果你需要，你可以用 Microsoft Query 向表中输入几百条记录。

## 删除和修改表

你应该在建立表之前仔细设计它们，因为你在改变一个已经存在的表时会受到很大的限制。例如，一旦已经建立了一个表，你就不能删除表中的字段或者改变字段的数据类型。在这种情况下你能做的是删除这个表，然后重头开始（参见第十一章“中级 SQL”中的“使用 SQL 创建记录 and 表”一节）。

要删除一个表，你可以使用 SQL 语句 DROP TABLE。例如，又从数据库中彻底删除表 mytable，你要使用如下的语句：

```
DROP TABLE mytable
```

### 警告：

使用 DROP TABLE 命令时一定要小心。一旦一个表被删除之后，你将无法恢复它。

当你建设一个站点时，你很可能需要向数据库中输入测试数据。而当你准备向世界提供你的网点时，你会想清空表中的这些测试信息。如果你想清除表中的所有数据但不删除这个表，你可以使用 TRUNCATE TABLE 语句。例如，下面的这个 SQL 语句从表 mytable 中删除所有数据：

```
TRUNCATE TABLE mytable
```

虽然你不能删除和修改已经存在的字段，但你可以增加新字段。最容易的实现方法是使用 SQL 事务管理器中的 Manager Tables 窗口。你也可以使用 SQL 语句 ALTER TABLE。下面是一个如何使用这种语句的例子：

```
ALTER TABLE mytable ADD mynewcolumn INT NULL
```

这个语句向表 mytable 中增加了一个新字段 mynewcolumn。当你增加新字段时，你必须允许它接受空值，因为表中原来可能已经有了许多记录。

## 总结

这一章向你介绍了 SQL。使用 SQL，你可以操作 Microsoft SQL Sever 数据库。你已经学会了使用 SELECT 语句从数据库中取出数据，你还学会了怎样使用 CREATE TABLE 语句和 SQL 事务管理器

来创建新表。最后，你学会了如何指明一系列重要的字段属性。

下一章将介绍如何使用索引来增强 SQL 查询的操作。还将通过许多其它的 SQL 语句和函数，使你的 SQL 知识得到进一步扩充。

## 中级 SQL

### 本章内容

- 创建索引
- SQL 核心语句
- 集合函数
- 其它常用的 SQL 表达式，  
函数，和过程

第十章“SQL 基础”向你初步介绍了 SQL。你学会了如何用 SELECT 语句进行查询，你还学会了如何建立自己的表。在这一章里，你将加深你的 SQL 知识。你将学习如何建立索引来加快查询速度。你还将学会如果用更多的 SQL 语句和函数来操作表中的数据。

## 建立索引

假设你想找到本书中的某一个句子。你可以一页一页地逐页搜索，但这会花很多时间。而通过使用本书的索引，你可以很快地找到你要搜索的主题。

表的索引与附在一本书后面的索引非常相似。它可以极大地提高查询的速度。对一个较大的表来说，通过加索引，一个通常要花费几个小时来完成的查询只要几分钟就可以完成。因此没有理由对需要频繁查询的表增加索引。

### 注意：

当你的内存容量或硬盘空间不足时，也许你不想给一个表增加索引。对于包含索引的数据库，SQL Sever 需要一个可观的额外空间。例如，要建立一个聚簇索引，需要大约 1.2 倍于数据大小的空间。要看一看一个表的索引在数据库中所占的空间大小，你可以使用系统存储过程 `sp_spaceused`，对象名指定为被索引的表名。

## 聚簇索引和非聚簇索引

假设你已经通过本书的索引找到了一个句子所在的页码。一旦已经知道了页码后，你很可能漫无目的翻寻这本书，直至找到正确的页码。通过随机的翻寻，你最终可以到达正确的页码。但是，有一种找到页码的更有效的方法。

首先，把书翻到大概一半的地方，如果要找的页码比半本书处的页码小，就书翻到四分之一处，否则，就把书翻到四分之三的地方。通过这种方法，你可以继续把书分成更小的部分，直至找到正确的页码附近。这是找到书页的非常有效的一种方法。

SQL Sever 的表索引以类似的方式工作。一个表索引由一组页组成，这些页构成了一个树形结构。根页通过指向另外两个页，把一个表的记录从逻辑上分成两个部分。而根页所指向的两个页又分别把记录分割成更小的部分。每个页都把记录分成更小的分割，直至到达叶级页。

索引有两种类型：聚簇索引和非聚簇索引。在聚簇索引中，索引树的叶级页包含实际的数据：记录的索引顺序与物理顺序相同。在非聚簇索引中，叶级页指向表中的记录：记录的物理顺序与逻辑顺序没有必然的联系。

聚簇索引非常象目录表，目录表的顺序与实际的页码顺序是一致的。非聚簇索引则更象书的标准索引表，索引表中的顺序通常与实际的页码顺序是不一致的。一本书也许有多个索引。例如，它也许同时有主题索引和作者索引。同样，一个表可以有多个非聚簇索引。

通常情况下，你使用的是聚簇索引，但是你应该对两种类型索引的优缺点都有所理解。

每个表只能有一个聚簇索引，因为一个表中的记录只能以一种物理顺序存放。通常你要对一个表按照标识字段建立聚簇索引。但是，你也可以对其它类型的字段建立聚簇索引，如字符型，数值型和日期时间型字段。

从建立了聚簇索引的表中取出数据要比建立了非聚簇索引的表快。当你需要取出一定范围内的数据时，用聚簇索引也比用非聚簇索引好。例如，假设你用一个表来记录访问者在你网点上的活动。如果你想取出在一定时间段内的登录信息，你应该对这个表的 DATETIME 型字段建立聚簇索引。

对聚簇索引的主要限制是每个表只能建立一个聚簇索引。但是，一个表可以有不止一个非聚簇索引。实际上，对每个表你最多可以建立 249 个非聚簇索引。你也可以对一个表同时建立聚簇索引和非聚簇索引。

假如你不仅想根据日期，而且想根据用户名从你的网点活动日志中取数据。在这种情况下，同时建立一个聚簇索引和非聚簇索引是有效的。你可以对日期时间字段建立聚簇索引，对用户名字段建立非聚簇索引。如果你发现你需要更多的索引方式，你可以增加更多的非聚簇索引。

非聚簇索引需要大量的硬盘空间和内存。另外，虽然非聚簇索引可以提高从表中取数据的速度，它也会降低向表中插入和更新数据的速度。每当你改变了一个建立了非聚簇索引的表中的数据时，必须同时更新索引。因此你对一个表建立非聚簇索引时要慎重考虑。如果你预计一个表需要频繁地更新数据，那么不要对它建立太多非聚簇索引。另外，如果硬盘和内存空间有限，也应该限制使用非聚簇索引的数量。

## 索引属性

这两种类型的索引都有两个重要属性：你可以用两者中任一种类型同时对多个字段建立索引（复合索引）；两种类型的索引都可以指定为唯一索引。

你可以对多个字段建立一个复合索引，甚至是复合的聚簇索引。假如有一个表记录了你的网点访问者的姓和名字。如果你希望根据完整姓名从表中取数据，你需要建立一个同时对姓字段和名字字段进行的索引。这和分别对两个字段建立单独的索引是不同的。当你希望同时对不止一个字段进行查询时，你应该建立一个对多个字段的索引。如果你希望对各个字段进行分别查询，你应该对各字段建立独立的索引。

两种类型的索引都可以被指定为唯一索引。如果对一个字段建立了唯一索引，你将不能向这个字段输入重复的值。一个标识字段会自动成为唯一值字段，但你也可以对其它类型的字段建立唯一索引。假设你用一个表来保存你的网点的用户密码，你当然不希望两个用户有相同的密码。通过强制一个字段成为唯一值字段，你可以防止这种情况的发生。

## 用 SQL 建立索引

为了给一个表建立索引，启动任务栏 SQL Sever 程序组中的 ISQL/w 程序。进入查询窗口后，输入下面的语句：

```
CREATE INDEX mycolumn_index ON mytable (mycolumn)
```

这个语句建立了一个名为 mycolumn\_index 的索引。你可以给一个索引起任何名字，但你应该在索引名中包含所索引的字段名，这对你将来弄清楚建立该索引的意图是有帮助的。

### 注意：

在本书中你执行任何 SQL 语句，都会收到如下的信息：

This command did not return data, and it did not return any rows

这说明该语句执行成功了。

索引 mycolumn\_index 对表 mytable 的 mycolumn 字段进行。这是个非聚簇索引，也是个非唯一索引。（这是一个索引的缺省属性）

如果你需要改变一个索引的类型，你必须删除原来的索引并重建一个。建立了一个索引后，你可以用下面的 SQL 语句删除它：

```
DROP INDEX mytable.mycolumn_index
```

注意在 DROP INDEX 语句中你要包含表的名字。在这个例子中，你删除的索引是 mycolumn\_index，它是表 mytable 的索引。

要建立一个聚簇索引，可以使用关键字 CLUSTERED。）记住一个表只能有一个聚簇索引。（这里

有一个如何对一个表建立聚簇索引的例子：

```
CREATE CLUSTERED INDEX mycolumn_clust_index ON mytable(mycolumn)
```

如果表中有重复的记录，当你试图用这个语句建立索引时，会出现错误。但是有重复记录的表也可以建立索引；你只要使用关键字 `ALLOW_DUP_ROW` 把这一点告诉 SQL Sever 即可：

```
CREATE CLUSTERED INDEX mycolumn_cindex ON mytable(mycolumn)
WITH ALLOW_DUP_ROW
```

这个语句建立了一个允许重复记录的聚簇索引。你应该尽量避免在一个表中出现重复记录，但是，如果已经出现了，你可以使用这种方法。

要为一个表建立唯一索引，可以使用关键字 `UNIQUE`。对聚簇索引和非聚簇索引都可以使用这个关键字。这里有一个例子：

```
CREATE UNIQUE COUSTERED INDEX myclumn_cindex ON mytable(mycolumn)
```

这是你将经常使用的索引建立语句。无论何时，只要可以，你应该尽量对一个对一个表建立唯一聚簇索引来增强查询操作。

最后，要建立一个对多个字段的索引——复合索引——在索引建立语句中同时包含多个字段名。下面的例子对 `firstname` 和 `lastname` 两个字段建立索引：

```
CREATE INDEX name_index ON username(firstname,lastname)
```

这个例子对两个字段建立了单个索引。在一个复合索引中，你最多可以对 16 个字段进行索引。

## 用事务管理器建立索引

用事务管理器建立索引比用 SQL 语句容易的多。使用事务管理器，你可以看到已经建立的索引的列表，并可以通过图形界面选择索引选项。

使用事务管理器你可以用两种方式建立索引：使用 `Manage Tables` 窗口或使用 `Manage Indexes` 窗口。

要用 `Manage Tables` 窗口建立一个新索引，单击按钮 `Advanced Options`（它看起来象一个前面有一加号的表）。这样就打开了 `Advanced Options` 对话框。这个对话框有一部分标名为 `Primary Key`（见图 11.1）。

图 11.1

要建立一个新索引，从下拉列表中选择你想对之建立索引的字段名。如果你想建立一个对多字段的索引，你可以选择多个字段名。你还可以选择索引是聚簇的还是非聚簇的。在保存表信息后，索引会自动被建立。在 `Manage Tables` 窗口中的字段名旁边，会出现一把钥匙。

你已经为你的表建立了“主索引”。主索引必须对不包含空值的字段建立。另外，主索引强制一个字段成为唯一值字段。

要建立没有这些限制的索引，你需要使用 `Manage Indexes` 窗口。从菜单中选择 `Manage | Indexes`，打开 `Manage Indexes` 窗口。在 `Manage Indexes` 窗口中，你可以通过下拉框选择表和特定的索引。（见图 11.2）。要建立一个新索引，从 `Index` 下拉框中选择 `New Index.`，然后就可以选择要对之建

立索引的字段。单击按钮 Add，把字段加入到索引中。

图 11. 2

你可以为你的索引选择许多不同的选项。例如，你可以选择该索引是聚簇的还是非聚簇的。你还可以指定该索引为唯一索引。设计好索引后，单击按钮 Build，建立该索引。

#### 注意：

唯一索引是指该字段不能有重复的值，而不是只能建立这一个索引。

## SQL 核心语句

在第十章，你学会了如何用 SQL SELECT 语句从一个表中取数据。但是，到现在为止，还没有讨论如何添加，修改或删除表中的数据。在这一节中，你将学习这些内容。

## 插入数据

向表中添加一个新记录，你要使用 SQL INSERT 语句。这里有一个如何使用这种语句的例子：

```
INSERT mytable (mycolumn) VALUES ('some data')
```

这个语句把字符串'some data'插入表 mytable 的 mycolumn 字段中。将要被插入数据的字段的名字在第一个括号中指定，实际的数据在第二个括号中给出。

INSERT 语句的完整句法如下：

```
INSERT [INTO] {table_name|view_name} [(column_list)] {DEFAULT VALUES |
    Values_list | select_statement}
```

如果一个表有多个字段，通过把字段名和字段值用逗号隔开，你可以向所有的字段中插入数据。假设表 mytable 有三个字段 first\_column, second\_column, 和 third\_column。下面的 INSERT 语句添加了一条三个字段都有值的完整记录：

```
INSERT mytable (first_column, second_column, third_column)
    VALUES ('some data', 'some more data', 'yet more data')
```

#### 注意：

你可以使用 INSERT 语句向文本型字段中插入数据。但是，如果你需要输入很长的字符串，你应该使用 WRITETEXT 语句。这部分内容对本书来说太高级了，因此不加讨论。要了解更多的信息，请参考 Microsoft SQL Sever 的文档。

如果你在 INSERT 语句中只指定两个字段和数据会怎么样呢？换句话说，你向一个表中插入一条新记录，但有一个字段没有提供数据。在这种情况下，有下面的四种可能：

- 如果该字段有一个缺省值，该值会被使用。例如，假设你插入新记录时没有给字段 third\_column 提供数据，而这个字段有一个缺省值'some value'。在这种情况下，当新记录建立时会插入值'some value'。
- 如果该字段可以接受空值，而且没有缺省值，则会被插入空值。



- 如果该字段不能接受空值，而且没有缺省值，就会出现错误。你会收到错误信息：

The column in table mytable may not be null.

- 最后，如果该字段是一个标识字段，那么它会自动产生一个新值。当你向一个有标识字段的表中插入新记录时，只要忽略该字段，标识字段会给自己赋一个新值。

#### 注意：

向一个有标识字段的表中插入新记录后，你可以用 SQL 变量 @@identity 来访问新记录的标识字段的值。考虑如下的 SQL 语句：

```
INSERT mytable (first_column) VALUES('some value')
```

```
INSERT anothertable(another_first,another_second)
VALUES(@@identity,'some value')
```

如果表 mytable 有一个标识字段，该字段的值会被插入表 anothertable 的 another\_first 字段。这是因为变量 @@identity 总是保存最后一次插入标识字段的值。

字段 another\_first 应该与字段 first\_column 有相同的数据类型。但是，字段 another\_first 不能是应该标识字段。Another\_first 字段用来保存字段 first\_column 的值。

## 删除记录

要从表中删除一个或多个记录，需要使用 SQL DELETE 语句。你可以给 DELETE 语句提供 WHERE 子句。WHERE 子句用来选择要删除的记录。例如，下面的这个 DELETE 语句只删除字段 first\_column 的值等于 'Delete Me' 的记录：

```
DELETE mytable WHERE first_column='Delete Me'
```

DELETE 语句的完整句法如下：

```
DELETE [FROM] {table_name|view_name} [WHERE clause]
```

在 SQL SELECT 语句中可以使用的任何条件都可以在 DELETE 语句的 WHERE 子句 中使用。例如，下面的这个 DELETE 语句只删除那些 first\_column 字段的值为 'goodbye' 或 second\_column 字段的值为 'so long' 的记录：

```
DELETE mytable WHERE first_column='goodbye' OR second_column='so long'
```

如果你不给 DELETE 语句提供 WHERE 子句，表中的所有记录都将被删除。你不应该有这种想法。如果你想删除应该表中的所有记录，应使用第十章所讲的 TRUNCATE TABLE 语句。

#### 注意：

为什么要用 TRUNCATE TABLE 语句代替 DELETE 语句？当你使用 TRUNCATE TABLE 语句时，记录的删除是不作记录的。也就是说，这意味着 TRUNCATE TABLE 要比 DELETE 快得多。

## 更新记录

要修改表中已经存在的一条或多条记录，应使用 SQL UPDATE 语句。同 DELETE 语句一样，UPDATE 语句可以使用 WHERE 子句来选择更新特定的记录。请看这个例子：

```
UPDATE mytable SET first_column='Updated!' WHERE second_column='Update Me!'
```

这个 UPDATE 语句更新所有 second\_column 字段的值为 'Update Me!' 的记录。对所有被选中的记录，字段 first\_column 的值被置为 'Updated!'。

下面是 UPDATE 语句的完整句法：

```
UPDATE {table_name|view_name} SET [{table_name|view_name}]
    {column_list|variable_list|variable_and_column_list}
    [, {column_list2|variable_list2|variable_and_column_list2}...
    [, {column_listN|variable_listN|variable_and_column_listN}]]
    [WHERE clause]
```

#### 注意：

你可以对文本型字段使用 UPDATE 语句。但是，如果你需要更新很长的字符串，应使用 UPDATETEXT 语句。这部分内容对本书来说太高级了，因此不加讨论。要了解更多的信息，请参考 Microsoft SQL Sever 的文档。

如果你不提供 WHERE 子句，表中的所有记录都将被更新。有时这是有用的。例如，如果你想把表 titles 中的所有书的价格加倍，你可以使用如下的 UPDATE 语句：

你也可以同时更新多个字段。例如，下面的 UPDATE 语句同时更新 first\_column, second\_column, 和 third\_column 这三个字段：

```
UPDATE mytable SET first_column='Updated!'
                  Second_column='Updated!'
                  Third_column='Updated!'
                  WHERE first_column='Update Me!'
```

#### 技巧：

SQL 忽略语句中多余的空格。你可以把 SQL 语句写成任何你最容易读的格式。

## 用 SELECT 创建记录和表

你也许已经注意到，INSERT 语句与 DELETE 语句和 UPDATE 语句有一点不同，它一次只操作一个记录。然而，有一个方法可以使 INSERT 语句一次添加多个记录。要作到这一点，你需要把 INSERT 语句与 SELECT 语句结合起来，象这样：

```
INSERT mytable (first_column, second_column)
SELECT another_first, another_second
FROM anothertable
WHERE another_first='Copy Me!'
```

这个语句从 anothertable 拷贝记录到 mytable. 只有表 anothertable 中字段 another\_first 的值为 'Copy Me!' 的记录才被拷贝。

当为一个表中的记录建立备份时, 这种形式的 INSERT 语句是非常有用的。在删除一个表中的记录之前, 你可以先用这种方法把它们拷贝到另一个表中。

如果你需要拷贝整个表, 你可以使用 SELECT INTO 语句。例如, 下面的语句创建了一个名为 newtable 的新表, 该表包含表 mytable 的所有数据:

```
SELECT * INTO newtable FROM mytable
```

你也可以指定只有特定的字段被用来创建这个新表。要做到这一点, 只需在字段列表中指定你想要拷贝的字段。另外, 你可以使用 WHERE 子句来限制拷贝到新表中的记录。下面的例子只拷贝字段 second\_column 的值等于 'Copy Me!' 的记录的 first\_column 字段。

```
SELECT first_column INTO newtable  
FROM mytable  
WHERE second_column='Copy Me!'
```

使用 SQL 修改已经建立的表是很困难的。例如, 如果你向一个表中添加了一个字段, 没有容易的办法来去除它。另外, 如果你不小心把一个字段的数据类型给错了, 你将没有办法改变它。但是, 使用本节中讲述的 SQL 语句, 你可以绕过这两个问题。

例如, 假设你想从一个表中删除一个字段。使用 SELECT INTO 语句, 你可以创建该表的一个拷贝, 但不包含要删除的字段。这使你既删除了该字段, 又保留了不想删除的数据。

如果你想改变一个字段的数据类型, 你可以创建一个包含正确数据类型字段的新表。创建好该表后, 你就可以结合使用 UPDATE 语句和 SELECT 语句, 把原来表中的所有数据拷贝到新表中。通过这种方法, 你既可以修改表的结构, 又能保存原有的数据。

## 集合函数

到现在为止, 你只学习了如何根据特定的条件从表中取出一条或多条记录。但是, 假如你想对一个表中的记录进行数据统计。例如, 如果你想统计存储在表中的一次民意测验的投票结果。或者你想知道一个访问者在你的站点上平均花费了多少时间。要对表中的任何类型的数据进行统计, 都需要使用集合函数。

Microsoft SQL 支持五种类型的集合函数。你可以统计记录数目, 平均值, 最小值, 最大值, 或者求和。当你使用一个集合函数时, 它只返回一个数, 该数值代表这几个统计值之一。

### 注意:

要在你的 ASP 网页中使用集合函数的返回值, 你需要给该值起一个名字。要作到这一点, 你可以在 SELECT 语句中, 在集合函数后面紧跟一个字段名, 如下例所示:

```
SELECT AVG(vote) 'the_average' FROM opinion
```

在这个例子中, vote 的平均值被命名为 the\_average。现在你可以在你的 ASP 网页的数据库方法中使用这个名字。

## 统计字段值的数目

函数 COUNT ( ) 也许是最有用的集合函数。你可以用这个函数来统计一个表中有多少条记录。这里有一个例子：

```
SELECT COUNT (au_lname) FROM authors
```

这个例子计算表 authors 中名字 (last name) 的数目。如果相同的名字出现了不止一次，该名字将会被计算多次。如果你想知道名字为某个特定值的作者有多少个，你可以使用 WHERE 子句，如下例所示：

```
SELECT COUNT (au_lname) FROM authors WHERE au_lname='Ringer'
```

这个例子返回名字为 'Ringer' 的作者的数目。如果这个名字在表 authors 中出现了两次，则函数的返回值是 2。

假如你想知道有不同名字的作者的数目。你可以通过使用关键字 DISTINCT 来得到该数目。如下例所示：

```
SELECT COUNT (DISTINCT au_lname) FROM authors
```

如果名字 'Ringer' 出现了不止一次，它将只被计算一次。关键字 DISTINCT 决定了只有互不相同的值才被计算。

通常，当你使用 COUNT ( ) 时，字段中的空值将被忽略。一般来说，这正是你所希望的。但是，如果你仅仅想知道表中记录的数目，那么你需要计算表中所有的记录——不管它是否包含空值。下面是一个如何做到这一点的例子：

```
SELECT COUNT (*) FROM authors
```

注意函数 COUNT ( ) 没有指定任何字段。这个语句计算表中所有记录所数目，包括有空值的记录。因此，你不需要指定要被计算的特定字段。

函数 COUNT ( ) 在很多不同情况下是有用的。例如，假设有一个表保存了对你站点的质量进行民意调查的结果。这个表有一个名为 vote 的字段，该字段的值要么是 0，要么是 1。0 表示反对票，1 表示赞成票。要确定赞成票的数量，你可以所有下面的 SELECT 语句：

```
SELECT COUNT (vote) FROM opinion_table WHERE vote=1
```

## 计算字段的平均值

使用函数 COUNT ( )，你可以统计一个字段中有多少个值。但有时你需要计算这些值的平均值。使用函数 AVG ( )，你可以返回一个字段中所有值的平均值。

假如你对你的站点进行一次较为复杂的民意调查。访问者可以在 1 到 10 之间投票，表示他们喜欢你站点的程度。你把投票结果保存在名为 vote 的 INT 型字段中。要计算你的用户投票的平均值，你需要使用函数 AVG ( )：

```
SELECT AVG (vote) FROM opinion
```

这个 SELECT 语句的返回值代表用户对你站点的平均喜欢程度。函数 AVG ( ) 只能对数值型字段使用。这个函数在计算平均值时也忽略空值。

## 计算字段值的和

假设你的站点被用来出售卡片，已经运行了两个月，是该计算赚了多少钱的时候了。假设有一个名为 `orders` 的表用来记录所有访问者的订购信息。要计算所有订购量的总和，你可以使用函数 `SUM ()`：

```
SELECT SUM(purchase_amount) FROM orders
```

函数 `SUM ()` 的返回值代表字段 `purchase_amount` 中所有值的平均值。字段 `purchase_amount` 的数据类型也许是 `MONEY` 型，但你也可以对其它数值型字段使用函数 `SUM ()`。

## 返回最大值或最小值

再一次假设你有一个表用来保存对你的站点进行民意调查的结果。访问者可以选择从 1 到 10 的值来表示他们对你站点的评价。如果你想知道访问者对你站点的最高评价，你可以使用如下的语句：

```
SELECT MAX(vote) FROM opinion
```

你也许希望有人对你的站点给予了很高的评价。通过函数 `MAX ()`，你可以知道一个数值型字段的所有值中的最大值。如果有人对你的站点投了数字 10，函数 `MAX ()` 将返回该值。

另一方面，假如你想知道访问者对你站点的最低评价，你可以使用函数 `MIN ()`，如下例所示：

```
SELECT MIN(vote) FROM opinion
```

函数 `MIN ()` 返回一个字段的值中的最小值。如果字段是空的，函数 `MIN ()` 返回空值。

## 其它常用的 SQL 表达式，函数，和过程

这一节将介绍一些其它的 SQL 技术。你将学习如何从表中取出数据，其某个字段的值处在一定的范围，你还将学习如何把字段值从一种类型转换成另一种类型，如何操作字符串和日期时间数据。最后，你将学会一个发送邮件的简单方法。

## 通过匹配一定范围的值来取出数据

假设你有一个表用来保存对你的站点进行民意调查的结果。现在你想向所有对你的站点的评价在 7 到 10 之间的访问者发送书面的感谢信。要得到这些人的名字，你可以使用如下的 `SELECT` 语句：

```
SELECT username FROM opinion WHERE vote>6 and vote<11
```

这个 `SELECT` 语句会实现你的要求。你使用下面的 `SELECT` 语句也可以得到同样的结果：

```
SELECT username FROM opinion WHERE vote BETWEEN 7 AND 10
```

这个 `SELECT` 语句与上一个语句是等价的。使用哪一种语句是编程风格的问题，但你会发现使用表达式 `BETWEEN` 的语句更易读。

现在假设你只想取出对你的站点投了 1 或者 10 的访问者的名字。要从表 opinion 中取出这些名字，你可以使用如下的 SELECT 语句：

```
SELECT username FROM opinion WHERE vote=1 or vote=10
```

这个 SELECT 语句会返回正确的结果，没有理由不使用它。但是，存在一种等价的方式。使用如下的 SELECT 可以得到相同的结果：

```
SELECT username FROM opinion WHERE vote IN (1,10)
```

注意表达式 IN 的使用。这个 SELECT 语句只取出 vote 的值等于括号中的值之一的记录。

你也可以使用 IN 来匹配字符数据。例如，假设你只想取出 Bill Gates 或 President Clinton 的投票值。你可以使用如下的 SELECT 语句：

```
SELECT vote FROM opinion WHERE username IN ('Bill Gates','President Clinton')
```

最后，你可以在使用 BETWEEN 或 IN 的同时使用表达式 NOT。例如，要取出那些投票值不在 7 到 10 之间的人的名字，你可以使用如下的 SELECT 语句：

```
SELECT username FROM opinion WHERE vote NOT BETWEEN 7 and 10
```

要选取那些某个字段的值不在一系列值之中的记录，你可以同时使用 NOT 和 IN，如下例所示：

```
SELECT vote FROM opinion
      WHERE username NOT IN ('Bill Gates','President Clinton')
```

你不是必须在 SQL 语句中使用 BETWEEN 或 IN，但是，要使你的查询更接近自然语言，这两个表达式是有帮助的。

## 转换数据

SQL Server 足够强大，可以在需要的时候把大部分数值从一种类型转换为另一种类型。例如，要比较 SMALLINT 型和 INT 型数据的大小，你不需要进行显式的类型转换。SQL Server 会为你完成这项工作。但是，当你想在字符型数据和其它类型的数据之间进行转换时，你的确需要自己进行转换操作。例如，假设你想从一个 MONEY 型字段中取出所有的值，并在结果后面加上字符串“US Dollars”。你需要使用函数 CONVERT ( )，如下例所示：

```
SELECT CONVERT (CHAR(8),price)+'US Dollars' FROM orders
```

函数 CONVERT ( ) 带有两个变量。第一个变量指定了数据类型和长度。第二个变量指定了要进行转换的字段。在这个例子中，字段 price 被转换成长度为 8 个字符的 CHAR 型字段。字段 price 要被转换成字符型，才可以在它后面连接上字符串‘US Dollars’。

当向 BIT 型，DATETIME 型，INT 型，或者 NUMERIC 型字段添加字符串时，你需要进行同样的转换操作。例如，下面的语句在一个 SELECT 语句的查询结果中加入字符串‘The vote is’，该 SELECT 语句返回一个 BIT 型字段的值：

```
SELECT 'The vote is'+CONVERT(CHAR(1),vote) FROM opinion
```

下面是这个语句的结果示例：

```
The vote is 1
The vote is 1
The vote is 0
(3 row(s) affected)
```

如果你不进行显式的转换，你会收到如下的错误信息：

```
Implicit conversion from datatype 'varchar' to 'bit' is not allowed.
Use the CONVERT function to run this query.
```

## 操作字符串数据

SQL Server 有许多函数和表达式，使你能对字符串进行有趣的操作，包括各种各样的模式匹配和字符转换。在这一节中，你将学习如何使用最重要的字符函数和表达式。

### 匹配通配符

假设你想建立一个与 Yahoo 功能相似的 Internet 目录。你可以建立一个表用来保存一系列的站点名称，统一资源定位器（URL），描述，和类别，并允许访问者通过在 HTML form 中输入关键字来检索这些内容。

假如有一个访问者想从这个目录中得到其描述中包含关键字 trading card 的站点的列表。要取出正确的站点列表，你也许试图使用这样的查询：

```
SELECT site_name FROM site_directory WHERE site_desc='trading card'
```

这个查询可以工作。但是，它只能返回那些其描述中只有 trading card 这个字符串的站点。例如，一个描述为 We have the greatest collection of trading cards in the world! 的站点不会被返回。

要把一个字符串与另一个字符串的一部分相匹配，你需要使用通配符。你使用通配符和关键字 LIKE 来实现模式匹配。下面的语句使用通配符和关键字 LIKE 重写了上面的查询，以返回所有正确站点的名字：

```
SELECT SITE_name FROM site_directory
      WHERE site_desc LIKE '%trading card%'
```

在这个例子中，所有其描述中包含表达式 trading card 的站点都被返回。描述为 We have the greatest collection of trading cards in the world! 的站点也被返回。当然，如果一个站点的描述中包含 I am trading cardboard boxes online，该站点的名字也被返回。

注意本例中百分号的使用。百分号是通配符的例子之一。它代表 0 个或多个字符。通过把 trading card 括在百分号中，所有其中嵌有字符串 trading card 的字符串都被匹配。

现在，假设你的站点目录变得太大而不能在一页中完全显示。你决定把目录分成两部分。在第一页，你想显示所有首字母在 A 到 M 之间的站点。在第二页，你想显示所有首字母在 N 到 Z 之间的站点。要得到第一页的站点列表，你可以使用如下的 SQL 语句：

```
SELECT site_name FROM site_directory WHERE site_name LIKE '[A-M]'
```

在这个例子中使用了表达式[A-M]，只取出那些首字母在 A 到 M 之间的站点。中括号（[]）用来匹配处在指定范围内的单个字符。要得到第二页中显示的站点，应使用这个语句：

```
SELECT site_name FROM site_directory
WHERE site_name LIKE '[N-Z]'
```

在这个例子中，括号中的表达式代表任何处在 N 到 Z 之间的单个字符。

假设你的站点目录变得更大了，你现在需要把目录分成更多页。如果你想显示那些以 A，B 或 C 开头的站点，你可以用下面的查询来实现：

```
SELECT site_name FROM site_directory WHERE site_name LIKE '[ABC]'
```

在这个例子中，括号中的表达式不再指定一个范围，而是给出了一些字符。任何一个其名字以这些字符中的任一个开头的站点都将被返回。

通过在括号内的表达式中同时包含一个范围和一些指定的字符，你可以把这两种方法结合起来。例如，用下面的这个查询，你可以取出那些首字母在 C 到 F 之间，或者以字母 Y 开头的站点：

```
SELECT site_name FROM site_directory WHERE site_name LIKE '[C-FY]'
```

在这个例子中，名字为 Collegescape 和 Yahoo 的站点会被选取，而名字为 Magicw3 的站点则不会被选取。

你也可以使用脱字符（^）来排除特定的字符。例如，要得到那些名字不以 Y 开头的站点，你可以使用如下的查询：

```
SELECT site_name FROM site_directory WHERE site_name LIKE '[^Y]'
```

对给定的字符或字符范围均可以使用脱字符。

最后，通过使用下划线字符（\_），你可以匹配任何单个字符。例如，下面这个查询返回每一个其名字的第二个字符为任何字母的站点：

```
SELECT site_name FROM site_directory WHERE site_name LIKE 'M_crosoft'
```

这个例子既返回名为 Microsoft 的站点，也返回名为 Macrosoft 的站点。但是，名字为 Moocrosoft 的站点则不被返回。与通配符 '%' 不同，下划线只代表单个字符。

#### 注意：

如果你想匹配百分号或下划线字符本身，你需要把它们括在方括号中。如果你想匹配连字符（-），应把它指定为方括号中的第一个字符。如果你想匹配方括号，应把它们也括在方括号中。例如，下面的语句返回所有其描述中包含百分号的站点：

```
SELECT site_name FROM site_directory WHERE site_desc LIKE '%[%]'
```

## 匹配发音



Microsoft SQL 有两个允许你按照发音来匹配字符串的函数。函数 SOUNDEX ( ) 给一个字符串分配一个音标码，函数 DIFFERENCE ( ) 按照发音比较两个字符串。当你不知道一个名字的确切拼写，但多少知道一点它的发音时，使用这两个函数将有助于你取出该记录。

例如，如果你建立一个 Internet 目录，你也许想增加一个选项，允许访问者按照站点名的发音来搜索站点，而不是按名字的拼写。考虑如下的语句：

```
SELECT site_name FROM site_directory
      WHERE DIFFERENCE(site_name , 'Microsoft')>3
```

这个语句使用函数 DEFFERENCE ( ) 来取得其名字的发音与 Microsoft 非常相似的站点。函数 DIFFERENCE ( ) 返回一个 0 到 4 之间的数字。如果该函数返回 4，表示发音非常相近；如果该函数返回 0，说明这两个字符串的发音相差很大。

例如，上面的语句将返回站点名 Microsoft 和 Macrosoft。这两个名字的发音与 Microsoft 都很相似。如果你把上一语句中的大于 3 改为大于 2，那么名为 Zicrosoft 和 Megasoft 的站点也将被返回。最后，如果你只需要差别等级大于 1 即可，则名为 Picosoft 和 Minisoft 的站点也将被匹配。

要深入了解函数 DIFFERENCE ( ) 是如何工作的，你可以用函数 SOUNDEX ( ) 来返回函数 DIFFERENCE ( ) 所使用的音标码。这里有一个例子：

```
SELECT site_name 'site name', SOUNDEX(site_name) 'sounds like'
```

这个语句选取字段 site\_name 的所有数据及其音标码。下面是这个查询的结果：

site name	sounds like
Yahoo	Y000
Mahoo	M000
Microsoft	M262
Macrosoft	M262
Minisoft	M521
Microshoft	M262
Zicrosoft	Z262
Zaposoft	Z121
Millisoft	M421
Nanosoft	N521
Megasoft	M221
Picosoft	P221

(12 row(s) affected)

如果你仔细看一下音标码，你会注意到音标码的第一个字母与字段值的第一个字母相同。例如，Yahoo 和 Mahoo 的音标码只有第一个字母不同。你还可以发现 Microsoft 和 Macrosoft 的音标码完全相同。

函数 DIFFERENDE ( ) 比较两个字符串的第一个字母和所有的辅音字母。该函数忽略任何元音字母（包括 y），除非一个元音字母是一个字符串的第一个字母。

不幸的是，使用 SOUNDEX ( ) 和 DIFFERENCE ( ) 有一个欠缺。WHERE 子句中包含这两个函数的查询执行起来效果不好。因此，你应该小心使用这两个函数。

## 删除空格

有两个函数，LTRIM（）和 RTRIM（），可以用来从字符串中剪掉空格。函数 LTRIM（）去除应该字符串前面的所有空格；函数 RTRIM（）去除一个字符串尾部的所有空格。这里有一个任何使用函数 RTRIM（）的例子：

```
SELECT RTRIM(site_name) FROM site_directory
```

在这个例子中，如果任何一个站点的名字尾部有多余的空格，多余的空格将从查询结果中删去。你可以嵌套使用这两个函数，把一个字符串前后的空格同时删去：

```
SELECT LTRIM(RTRIM(site_name)) FROM site_directory
```

你会发现，在从 CHAR 型字段中剪掉多余的空格时，这两个函数非常有用。记住，如果你把一个字符串保存在 CHAR 型字段中，该字符串会被追加多余的空格，以匹配该字段的长度。用这两个函数，你可以去掉无用的空格，从而解决这个问题。

## 操作日期和时间

日期和时间函数对建立一个站点是非常有用的。站点的主人往往对一个表中的数据何时被更新感兴趣。通过日期和时间函数，你可以在毫秒级跟踪一个表的改变。

### 返回当前日期和时间

通过函数 GETDATE（），你可以获得当前的日期和时间。例如，语句 SELECT GETDATE（）返回如下结果：

```
.....
NOV 30 1997 3:29AM
(1 row(s) affected)
```

显然，如果你将来使用这个函数，你得到的日期将比这个时间晚，或者更早。

函数 GETDATE（）可以用来作为 DATETIME（）型字段的缺省值。这对插入记录时保存当时的时间是有用的。例如，假设有一个表用来保存你站点上的活动日志。每当有一个访问者访问到你的站点时，就在表中添加一条新记录，记下访问者的名字，活动，和进行访问的时间。要建立一个表，其中的记录包含有当前的日期和时间，可以添加一个 DATETIME 型字段，指定其缺省值为函数 GETDATE（）的返回值，就象这样：

```
CREATE TABLE site_log (
    username VARCHAR(40),
    useractivity VARCHAR(100),
    entrydate DATETIME DEFAULT GETDATE())
```

## 转换日期和时间

你也许已经注意到，在上一节的例子中，函数 GETDATE（）的返回值在显示时只显示到秒。实际上，SQL Sever 内部时间可以精确到毫秒级（确切地说，可以精确到 3.33 毫秒）。

要得到不同格式的日期和时间，你需要使用函数 CONVERT ( )。例如，当下面的这个语句执行时，显示的时间将包括毫秒：

```
SELECT CONVERT (VARCHAR (30), GETDATE (), 9)
```

注意例子中数字 9 的使用。这个数字指明了在显示日期和时间时使用哪种日期和时间格式。当这个语句执行时，将显示如下的日期和时间：

```
.....
Nov 30 1997 3:29:55:170AM
(1 row(s) affected)
```

在函数 CONVERT ( ) 中你可以使用许多种不同风格的日期和时间格式。表 11.1 显示了所有的格式。

表 11.1 日期和时间的类型

类型值	标准	输出
0	Default	mon dd yyyy hh:miAM
1	USA	mm/dd/yy
2	ANSI	yy.mm.dd
3	British/French	dd/mm/yy
4	German	dd.mm.yy
5	Italian	dd-mm-yy
6	-	dd mon yy
7	-	mon dd,yy
8	-	hh:mi:ss
9		Default + milliseconds--mon dd yyyy hh:mi:ss:mmmAM(or )
10	USA	mm-dd-yy
11	JAPAN	yy/mm/dd
12	ISO	yymmdd
13	Europe	Default + milliseconds--dd mon yyyy hh:mi:ss:mmm(24h)
14	-	hh:mi:ss:mmm(24h)

类型 0, 9, 和 13 总是返回四位的年。对其它类型，要显示世纪，把 style 值加上 100。类型 13 和 14 返回 24 小时时钟的时间。类型 0, 7, 和 13 返回的月份用三位字符表示（用 Nov 代表 November）。

对表 11.1 中所列的每一种格式，你可以把类型值加上 100 来显示有世纪的年（例如，00 年将显示为 2000 年）。例如，要按日本标准显示日期，包括世纪，你应使用如下的语句：

```
SELECT CONVERT (VARCHAR (30), GETDATE (), 111)
```

在这个例子中，函数 CONVERT ( ) 把日期格式进行转换，显示为 1997/11/30

抽取日期和时间

在许多情况下，你也许只想得到日期和时间的一部分，而不是完整的日期和时间。例如，假设你想列出你的站点目录中每个站点被查询的月份。这时你不希望完整的日期和时间把网页弄乱。为了抽取日期的特定部分，你可以使用函数 DATEPART ()，象这样：

```
SELECT site_name 'Site Name',
DATEPART(mm,site_entrydate) 'Month Posted' FROM site_directory
```

函数 DATEPART () 的参数是两个变量。第一个变量指定要抽取日期的哪一部分；第二个变量是实际的数据。在这个例子中，函数 DATEPART () 抽取月份，因为 mm 代表月份。下面是这个 SELECT 语句的输出结果：

Site Name	Month Posted
.....	
Yahoo	2
Microsoft	5
Magicw3	5
(3 row(s) affected)	

Month Posted 列显示了每个站点被查询的月份。函数 DATEPART () 的返回值是一个整数。你可以用这个函数抽取日期的各个不同部分，如表 11.2 所示。

表 11.2 日期的各部分及其简写

日期部分	简写	值
year	yy	1753--9999
quarter	qq	1--4
month	mm	1--12
day of year	dy	1--366
day	dd	1--31
week	wk	1--53
weekday	dw	1--7 (Sunday--Saturday)
hour	hh	0--23
minute	mi	0--59
second	ss	0--59
milisecond	ms	0--999

当你需要进行日期和时间的比较时，使用函数 DATEPART () 返回整数是有用的。但是，上例中的查询结果 (2, 5) 不是十分易读。要以更易读的格式得到部分的日期和时间，你可以使用函数 DATENAME ()，如下例所示：

```
SELECT site_name 'Site Name'
DATENAME(mm,site_entrydate) 'Month Posted'
FROM site_directory
```

函数 DATENAME () 和函数 DATEPART () 接收同样的参数。但是，它的返回值是一个字符串，而不是一个整数。下面是上例该用 DATENAME () 得到的结果：

Site Name	Month Posted
.....	
Yahoo	February
Microsoft	June
Magicw3	June
(3 row(s) affected)	

你也可以用函数 DATENAME ( ) 来抽取一个星期中的某一天。下面的这个例子同时抽取一周中的某一天和日期中的月份：

```
SELECT site_name 'Site Name',
DATENAME(dw,site_entrydate)+ '-' + DATENAME (mm,site_entrydate)
'Day and Month Posted' FROM site_directory
```

这个例子执行时，将返回如下的结果：

Site Name	Day and Month Posted
.....	
Yahoo	Friday - February
Microsoft	Tuesday - June
Magicw3	Monday - June
(3 row(s) affected)	

## 返回日期和时间范围

当你分析表中的数据时，你也许希望取出某个特定时间的数据。你也许对特定的某一天中——比如说 2000 年 12 月 25 日——访问者在你站点上的活动感兴趣。要取出这种类型的数据，你也许会试图使用这样的 SELECT 语句：

```
SELECT * FROM weblog WHERE entrydate="12/25/2000"
```

不要这样做。这个 SELECT 语句不会返回正确的记录——它将只返回日期和时间是 12/25/2000 12:00:00:000AM 的记录。换句话说，只有刚好在午夜零点输入的记录才被返回。

### 注意：

在本节的讨论中，假设字段 entrydate 是 DATETIME 型，而不是 SMALLDATETIME 型。本节的讨论对 SMALLDATETIME 型字段也是适用的，不过 SMALLDATETIME 型字段只能精确到秒。

问题是 SQL Sever 将用完整的日期和时间代替部分日期和时间。例如，当你输入一个日期，但不输入时间时，SQL Sever 将加上缺省的时间“12: 00: 00: 000AM”。当你输入一个时间，但不输入日期时，SQL Sever 将加上缺省的日期“Jan 1 1900”。

要返回正确的记录，你需要适用日期和时间范围。有不只一种途径可以做到这一点。例如，下面的这个 SELECT 语句将能返回正确的记录：

```
SELECT * FROM weblog
WHERE entrydate>="12/25/2000" AND entrydate<"12/26/2000"
```

这个语句可以完成任务，因为它选取的是表中的日期和时间大于等于 12/25/2000 12:00:00:000AM 并小于 12/26/2000 12:00:00:000AM 的记录。换句话说，它将正确地返回 2000 年圣诞节这一天输入的每一条记录。

另一种方法是，你可以使用 LIKE 来返回正确的记录。通过在日期表达式中包含通配符 “%”，你可以匹配一个特定日期的所有时间。这里有一个例子：

```
SELECT * FROM weblog WHERE entrydate LIKE 'Dec 25 2000%'
```

这个语句可以匹配正确的记录。因为通配符 “%” 代表了任何时间。

使用这两种匹配日期和时间范围的函数，你可以选择某个月，某一天，某一年，某个小时，某一分钟，某一秒，甚至某一毫秒内输入的记录。但是，如果你使用 LIKE 来匹配秒或毫秒，你首先需要使用函数 CONVERT () 把日期和时间转换为更精确的格式（参见前面“转换日期和时间”一节）。

## 比较日期和时间

最后，还有两个日期和时间函数对根据日期和时间取出记录是有用的。使用函数 DATEADD () 和 DATEDIFF (), 你可以比较日期的早晚。例如，下面的 SELECT 语句将显示表中的每一条记录已经输入了多少个小时：

```
SELECT entrydate 'Time Entered'
DATEDIFF(hh,entrydate,GETDATE()) 'Hours Ago' FROM weblog
```

如果当前时间是 2000 年 11 月 30 号下午 6 点 15 分，则会返回如下的结果：

Time Entered	Hours Ago
Dec 30 2000 4:09PM	2
Dec 30 2000 4:13PM	2
Dec 1 2000 4:09PM	698

(3 row(s) affected)

函数 DATEDIFF () 的参数是三个变量。第一个变量指定日期的某一部分。在这个例子中，是按小时对日期进行比较，（要了解日期各部分的详细内容，请参考表 11.2）在日期 2000 年 11 月 1 日和 2000 年 11 月 30 日的指定时间之间有 689 个小时。另外两个参数是要进行比较的时间。为了返回一个正数，较早的时间应该先给。

函数 DATEADD () 把两个日期相加。当你需要计算截止日期这一类的数据时，这个函数是有用的。例如，假设访问者必须先注册才能使用你的站点。注册以后，他们可以免费使用你的站点一个月。要确定什么时候他们的免费时间会用完，你可以使用如下的 SELECT 语句：

```
SELECT username 'User Name',
DATEADD(mm,1,firstvisit_date) 'Registration Expires'
FROM registration_table
```

函数 DATEADD () 的参数有三个变量。第一个变量代表日期的某一部分（参见表 11.2），这个例子用到了代表月份的 mm。第二个变量指定了时间的间隔——在本例中是一个月。最后一个变量是一

个日期，在这个例子中，日期是取自 DATETIME 型字段 firstvisit\_date. 假设当前日期是 June 30, 2000，这个语句将返回如下的内容：

User Name	Registration Expires
.....	
Bill Gates	Jul 30 2000 4:09PM
President Clinton	Jul 30 2000 4:13PM
William Shakespeare	Jul 1 2000 4:09PM
(3 row(s) affected)	

#### 注意：

与你预料的相反，使用函数 DATEADD ( ) 把一个日期加上一个月，它并不加上 30 天。这个函数只简单地把月份值加 1。这意味着在 11 月注册的人将比在 2 月注册的人多得到 2 天或 3 天的时间。要避免这个问题，你可以用函数 DATEADD ( ) 直接增加天数，而不是月份。

## 发送邮件

你可以用 SQL Sever 发送简单的 e\_mail 信息。要做到这一点，你需要在你的系统中安装邮件服务器，如 Microsoft Exchange Sever(参见第四章“Exchange Active Sever, Index Sever, 和 NetShow”)。你还需要配置 SQL Sever 以识别邮件服务器。

要让 SQL Sever 能识别邮件服务器，启动事务管理器并从菜单中选择 Sever|SQL Mail|Configure，这时会出现一个如图 11.3 所示的对话框。输入你在邮件服务器中注册的用户名和口令，然后单击 OK。

#### 注意：

如果你使用 Microsoft Exchange Sever, 配置 SQL Sever 的过程将会大大不同。你需要在同一个（域）用户帐号下运行 Microsoft SQL Sever 和 Exchange Sever。你还需要在安装了 SQL Sever 的机器上安装 Exchange Cliect 并给这个帐号创建一个配置文件。完成这些之后，你就可以在 SQL Mail Configuration 对话框中输入该配置文件的名字。

图 11.3

在发送邮件之前，你要先启动 SQL Mail。从菜单中选择 Sever|SQL Mail|Start。如果你的邮件服务器配置正确，并且你输入了正确的用户名和口令，则 SQL Mail 会成功启动。

#### 注意：

你可以把 SQL Sever 配置为自动启动邮件服务。要做到这一点，在 Set Sever Options 对话框（从菜单中选择 Sever|SQL Sever|Configure）中选择 Auto Start Mail Client 即可。

要发送一个邮件，你可以使用名为 xp\_sendmail 的扩展存储过程。这里有一个如何使用这个过程的例子：

```
master..xp_sendmail "president@whitehouse.gov", "Hello Mr. President"
```

这个过程调用向 e\_mail 地址 [president@whitehouse.gov](mailto:president@whitehouse.gov) 发送一个简单的 email 信息：“Hello Mr. President”。你可以用任何其它的 email 地址和信息取代上例中相应的内容，但是，你所发送的信

息不能超过 255 个字符长度。

当你想随时了解你的站点数据库的状态时，存储过程 `xp_sendmail` 是有用处的。例如，你可以向一个页面管理程序发送信息。如果你的站点出了什么问题，你马上就可以知道。下一章将讲述更多有关存储过程的内容。

## 总结

这一章加深了你的 SQL 知识。你学会了如何建立索引，使你的查询速度更快。你还学会了如何插入，删除和更新一个表中的数据，如何使用集合函数得到一个表中数据的统计信息。最后，你学会了许多有价值的表达式，函数和过程，用来操作字符串，日期和时间及邮件。

下一章将进一步加深你对 Microsoft SQL Sever 的掌握。你将学习如何用 SQL 来进行程序设计，如何建立存储过程，触发器和执行计划。更另人兴奋的是，你将学会让 SQL Sever 自动创建网页的一个简单方法。