前　言

随着计算机技术的不断提高，计算机的应用面在不断拓宽。尤其是在信息技术大发展的时代，信息的存储、信息的管理、信息的获取已成为各个领域适应市场发展的基本需求。因此数据库技术已渗透到各种应用中。并已成为各信息管理系统、决策支持系统、地理信息系统、办公信息系统等各类信息处理应用系统的核心技术。

数据库技术与各类边缘技术互相渗透，互相结合，使得数据库技术的应用前景更为广阔。微软的 SQL Server 数据库管理系统正是应这一市场需求而开发的。

SQL Server 数据库管理系统是目前市场上非常流行的一种支持 C/S 和 B/S 模式的数据库产品。它有入门容易、开发成本低、运行和维护费用低、技术更新快、兼容性好、功能完整等特点，在电子商务、数据仓库、信息管理系统等小中型应用系统解决方案中扮演着重要的角色，为企业的信息管理提供了强大的技术支持，近几年来具有很高的市场占有量。

本书在介绍数据库基本概念、数据库应用系统设计方法基础上，以教学管理数据库应用系统为背景，讨论了应用 SQL Server 提供的语言和工具，设计完成数据库应用系统的基本方法和思想。

鉴于多年的教学实践，考虑教学特点以及目前人才市场对数据库应用系统开发者能力的要求，我们在书中将理论基础和应用实践的比例进行了最佳的调整。

书中的内容包括两部分：理论篇——数据库原理；应用篇——SQL Server 技术。二者的比例基本是 4∶6。

其中理论篇包括：数据库系统基础、关系运算、关系的规范化理论及数据库系统设计方法等；应用篇包括：SQL Server 基础、物理数据库的设计、数据库的操作、数据的查询与统计、视图、存储过程、触发器、Transact-SQL 语言编程方法、数据的备份与恢复等。

本书结构清晰合理，系统性强，实用性好。各章涉及到的重要概念，均有相关的例题以帮助读者理解和掌握。

为了便于教师课程教学的安排，本书附有教学学时分配指南；为了便于学生课后复习，各章后附有实习实例、内容小结、习题等。

本书的第 1、2 章由张凤琴同志编写；第 3、4 章由张青凤同志编写；第 5、8 章由李淑玲同志编写；第 6、7、9 章由张水平、王蓉同志编写；第 10 章由詹金珍同志编写。全书由沈钧毅教授审阅，并提出宝贵的修改意见，在此表示感谢。

本书可作为计算机专业的教学用书，也可作为参加相关计算机专业自学考试用书及相关技术人员的参考用书。

本书按照由浅入深，循序渐进的原则编写。各章内容既有联系，又有相对独立性，可根据不同教学要求有重点、有选择地组织教学，其中带“*”的章节为可选内容。附表给出了根据不同学时要求各章节的学时分配，供教学时参考。

本书编写内容不足之处在所难免，恳请专家和读者批评指正。

需要全本请访问 www.ertongbook.com

附表　各层次教学学时分配指南

章　节	学　时　分　配							
	70 学时		60 学时		50 学时		40 学时	
总学时	理论	实习	理论	实习	理论	实习	理论	实习
	46	24	40	20	30	20	24	16
第 1 章 数据库系统概述	4	0	4	0	2	0	4	0
第 2 章 关系运算	4		2		2	0		
第 3 章 关系的规范化理论	4		2		2	0		
第 4 章 数据库设计方法	4		4		2	0		
第 5 章 SQL Server 概述	2	2	2	2	2	2	2	2
第 6 章 物理数据库创建与维护	4	4	4	4	4	4	4	4
第 7 章 数据库的查询操作	8	6	8	6	8	6	6	6
第 8 章 Transact-SQL 语言	6	6	6	4	4	6	4	2
第 9 章 存储过程及触发器	4	4	4	2	4	4	4	2
第 10 章 数据库的维护	2	2	2	2	2	2	0	
复习	2		2		2		2	

作者

2007 年 11 月

第 7 章　数据库操作

建立数据库的目的是为了使用数据库中的数据，即所谓对数据库中数据的操作，包括数据的插入、删除、修改、查询、统计等。为了提高数据处理速度，使其数据的处理达到最佳效果，就必须采取一定的措施，例如根据具体的应用特点，为表建立不同的索引，为不同的应用建立视图等。本章将对 SQL Server 所提供的数据操纵语言 DML 及其工具对数据库的各种操作方法进行详细的论述。

7.1　表数据的操作

创建数据表的目的是为了存储和管理数据。我们经常需要通过向数据表中插入数据以实现数据的存储，或是通过修改、删除数据表中的数据实现对表的管理。下面介绍使用企业管理器和 Transact-SQL 语言插入、修改和删除表数据的方法。

7.1.1　用企业管理器进行数据的操作

用企业管理器进行数据的插入、修改和删除非常方便，只需要打开设计好的表，在表中进行数据的编辑（插入、删除、修改）即可。

打开表的方法很简单：在企业管理器下，打开你的数据库，并选中所要操作的表，单击右键，如图 7－1 所示。单击【打开表】，系统会弹出图 7－2 所示的数据表窗口。在这里进行数据



图 7－1　在企业管理器下打开表

的各种操作。用户可以通过单击新行来添加数据，还可以进入单独的数据域进行数据的修改。此外，亦可选定相应的数据行进行删除，具体方法为：首先选定欲删除的数据行，然后按下键盘上的 DELETE 键，系统会弹出确认对话框，点击“是”，即可删除该数据行。

注意：在表中进行数据的编辑时，一定要遵守表定义时的数据类型、主键、是否允许为空等各种约束，否则数据将无法录入。另外，使用企业管理器只适合操作一些少量、分散的数据，不适合成批或灵活地处理数据。

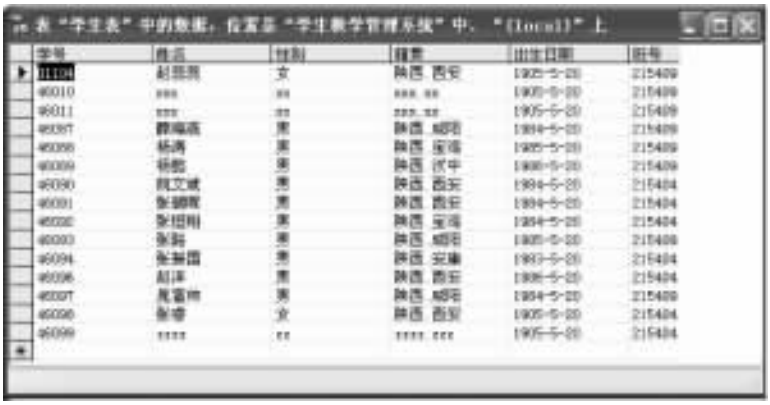


图 7-2 企业管理器中的数据表

7.1.2 用 Transact-SQL 语言进行数据的操作

用 Transact-SQL 语言对数据进行操作通常是在程序中对数据进行动态处理时使用，尤其是批量数据的插入或前端工具软件（如 VB、C#、JAVA）等处理的中间结果要存入到关系中时要用到。

1. 插入数据

SQL Server 支持多种向数据表中插入数据的方法，最为常用的是使用 Transact-SQL 中的 INSERT 语句。其基本语法为

INSERT

[INTO] table_name [(column_list)]

VALUES data_values

其中参数意义如下：

column_list：由逗号分隔的列名列表，用来指定为其提供数据的列。如果没有指定 column_list，表或者视图中的所有列都将接收数据。

data_values：作为一行或者多行数据插入已命名的表中。根据 VALUES 子句的不同，有以下两种数据插入的形式。

(1) 使用 INSERT... VALUES 插入一条记录。

使用 VALUES 子句，INSERT 语句变为

INSERT [INTO] table_name [(column_list)] VALUES (expression_list)

VALUES 子句引入要插入的数据值的列表。值被指定为逗号分隔的标量表达式列表，表达式的数据类型、精度和小数字数必须与列表对应的列一致，或者可以隐性地转换为列表中的

对应列。

注意：如果 INTO 字句中没有指定任何列名，则新插入的记录必须在每个属性列上均有值。

【例 7-1】 向学生表中添加一行新记录(01201101，张会映，女，陕西. 西安，01/12/1975，012011)。

解：可由如下语句实现：

```
INSERT
INTO 学生表 (学号,姓名,性别,籍贯,出生日期,班号)
VALUES ('01201101','张会映','女','陕西. 西安',01/12/1997,'012011')
```

如果没有指定列表，则 VALUES 子句中指定值的顺序必须与表中列的顺序一致。

【例 7-2】 向成绩表中添加一条记录。

解：由下列语句添加数据：

```
INSERT
INTO 成绩表(学号,课号)
VALUES ('01201101','012001')
```

新插的记录“成绩”列由于未给出，故取空值(在创建成绩表时，成绩列应允许为空)

(2) 使用 INSERT...SELECT 插入批量数据。

被插入的数据来自于 SELECT 子句查询的结果，SELECT 查询语句是 SQL 语言的核心语句，关于该语句我们将在后续课程中详细讲解。

使用 SELECT 子句，INSERT 语句变为

```
INSERT [INTO] table_name [(column_list)]
SELECT (select_list)
FROM table_name
WHERE search_conditions
```

INSERT 语句中的 SELECT 子查询可用于将一个或多个其他表的值添加到表中并可同时插入多行。子查询的 select_list 必须与 INSERT 语句的 column_list 匹配。如果没有指定列的列表，select_list 必须与插入的表的列正方向匹配。

【例 7-3】 将所有课程成绩不及格的成绩记录插入到一个成绩表 1 中(假定表 1 已创建，该表与成绩表结构完全相同)。

解：下列语句可实现数据插入：

```
INSERT
INTO 成绩表 1 (学号,课号,成绩)
SELECT 学号,课号,成绩
FROM 成绩表
WHERE 成绩<60
```

2. 数据的修改

对数据表中一列或多列的值进行修改时，最常用的方法是使用 UPDATE 语句。其简单语法如下：

```
UPDATE table_name
```

需要全本请访问 www.ertongbook.com

SET column_name ={ expression | DEFAULT | NULL }

FROM table_source

[WHERE search_conditions]

参数说明如下：

SET 子句包含要更新的列和新值的列表(用逗号分隔)，FROM 子句指定为 SET 子句中的表达式提供值的表，WHERE 子句指定条件限定所要更新的行，如果省略 WHERE 子句，表示要修改所有的记录。

(1) 修改指定记录的值。

下面举例说明 UPDATE 语句的使用。

【例 7-4】 将学生表中，学号为“01201101”的学生性别改为“男”。

解：由下列语句实现：

```
UPDATE 学生表
SET 性别='男'
WHERE 学号='01201101'
```

该例只修改了一条记录的值。

【例 7-5】 由于计算机系改名为“信息技术系”，要求将班级表中，计算机系的班级“所在系”改为“信息技术系”。

解：由下列语句实现：

```
UPDATE 班级表
SET 所在系='信息技术系'
WHERE 所在系='计算机系'
```

该例修改了多条记录的值。

(2) 修改指定表中的所有记录的值。

这里不使用 WHERE 子句，则 UPDATE 语句会修改表中的每一行。

【例 7-6】 将学生表中(假定该表中有“年龄”列)，所有学生的年龄均增加一岁。

解：由下列语句实现：

```
UPDATE 学生表
SET 年龄=年龄+1
```

(3) 带 select 子句的修改语句。

由于限定修改条件的列不在要修改的表中，此时不能简单地通过 WHERE 构造修改条件，而要借助于 select 子句进行，如下列所示。

【例 7-7】 将教师号是“0120001”的教师所授课程的全体学生的成绩设置为零。

解：由于“教师号”在授课表中，成绩又在成绩表中，而课号是其两个表的共同属性。因此具体实现语句如下：

```
UPDATE 成绩表
SET 成绩=0
WHERE 课号 in
  ( SELECT 课号
    FROM 授课表
```

```
    WHERE 教师号='0120001'
  )
```

3. 数据的删除

删除语句的具体格式如下：

```
DELETE
FROM table_name
[WHERE] search_condition
```

table_name 指定要从中删除行的表。该表中所有符合 WHERE 搜索条件的行都将被删除。如果没有指定 WHERE 子句，将删除表中的所有行。WHERE 子句指定搜索条件。

注意：当省略 WHERE 子句，意味着要删除指定表中的所有记录。另外 DELETE 语句只删除表中的数据，并不删除表结构。

当要从数据表中删除数据时，最常用的方法是使用 DELETE 语句，当然，如果要删除数据表中的所有数据，TRUNCATE TABLE 是一个更好的选择。下面就分别来介绍这两种方式。

(1) 删除表中指定记录的值。

【例 7-8】 删除学生表中班号为“012011”的所有学生的信息。

解：由下列语句实现：

```
DELETE
FROM 学生表
WHERE 班号='012011'
```

如果不指定搜索条件，DELETE 语句将删除学生表中的所有数据。如：

```
DELETE
FROM 学生表
```

注意：若无意中错误使用了 DELETE 语句，可以从备份数据库中对数据进行恢复。有关这方面的知识，请参见第 10 章的相关内容。

(2) 带 select 子句的删除语句。

与上述带 select 子句的修改语句类似，如果限定删除的条件列不在要删除的表中，也要借助于 select 子句进行，如例 7-9 所示。

【例 7-9】 将教师号是“0120001”的教师所授课程的全体学生的成绩记录从成绩表中删除。

解：具体实现语句如下：

```
DELETE
FROM 成绩表
WHERE 课号 in
  ( SELECT 课号
    FROM 授课表
    WHERE 教师号='0120001'
  )
```

(3) 用 TRUNCATE TABLE 语句删除数据。

TRUNCATE TABLE 语句也可以删除数据表中的所有数据，而且它通常比不带条件的

DELETE 语句要快,因为 DELETE 语句要记录对每行的删除操作,而 TRUNCATE TABLE 语句只记录整个数据页的释放,与 DELETE 语句相同,它只删除表内的数据。

TRUNCATE TABLE 的语法是:

TRUNCATE TABLE table_name

table_name 指定要从中删除行的表。

【例 7-10】 删除成绩表中所有的记录。

解:TRUNCATE TABLE 学生表

注意:由于 TRUNCATE TABLE 语句是一种无日志记录的方法,用该语句删除数据后无法进行数据恢复,因此使用 TRUNCATE TABLE 应十分谨慎。

以上简述了利用 Transact-SQL 语言对数据表中数据进行操作的主要方法,我们还可以利用企业管理器完成管理数据的任务。

7.2 索引的处理

索引是 SQL Server 的另一种数据对象,通过对表中的数据建立索引,可提高数据的访问速度。它类似于图书的索引,数目的索引可以帮助读者无须翻阅全书,就可以快速找到所需要的内容。

表的索引就是一张按照建立索引的列或列的组合进行排序的表,这个表保存着指向对应记录的指针。

假定教学管理数据库中有几万学生的信息,我们要在学生表中查找学号为"01201101"的学生,如果没有对表中的学号列建立索引,则查询操作须遍历表中几万条记录,才能以找到相应的数据。可以想象,执行这种查询操作会花费很多时间。相反,如果对学号列建立索引,则该索引将包括一个指向数据的指针,系统按照指针就能很快找到数据。从而大大加快查询速度。

7.2.1 索引的分类

SQL Server 按照索引列的重复值、索引表的物理顺序以及索引列数等的情况对索引进行了分类,共划分了三种类型:

(1) 唯一索引和非唯一索引;

(2) 聚集索引和非聚集索引;

(3) 单列索引和复合索引。

1. 唯一索引和非唯一索引

唯一索引要求被索引列不能有重复值(包括 NULL 值),而非唯一索引则没有这一要求。

在建立唯一索引时,如果 SQL Server 发现索引列存在重复值,则将停止建立索引并显示错误信息。当唯一索引建立后,SQL Server 将禁止向数据表中插入重复的索引值。

当为数据表定义一个主键时,SQL Server 将为主键默认地建立一个索引,称为主键索引。可以看出,主键索引是唯一索引的特殊情况。

2. 聚集索引和非聚集索引

在聚集索引中，表中记录的物理顺序（记录在计算机中实际存储的顺序）与索引顺序（键值的逻辑顺序）相同，一个表只能有一个聚集索引。聚集索引比非聚集索引有更快的检索速度。对于非聚集索引，行的物理排序独立于索引排序。

注意：一个数据表只能建立一个聚集索引。另外建立聚集索引由于记录的物理顺序与索引键值的逻辑顺序要一致，因此建立索引时，数据更新一旦涉及到键值的列被修改，索引的更新（或者说维护）比非聚集索引花费时间。

3. 单列索引和复合索引

单列索引根据表中的单个列建立索引，复合索引则依据表中的多个列建立索引。复合索引可以代替多个单列索引并且所需开销更少。一个表的复合索引允许的最大列数是 16，而这些列定义的最大长度之和不得超过 900 个字节。

另外，建复合索引时列的顺序可以和表中列的顺序不同。

7.2.2　索引的建立

在 SQL Server 中可用以下几种方法建立索引：

（1）使用创建索引向导；

（2）使用企业管理器；

（3）使用 Transact-SQL 中语言中的 CREATE INDEX 命令；

（4）使用 Transact-SQL 中语言中的 CREATE TABLE，ALTER TABLE 命令。

这里只简述前三种方法。第四中方法读者可以查阅有关 CREATE TABLE、ALTER TABLE 命令。

1. 使用创建索引向导

使用创建索引向导与使用创建数据库向导创建数据库类似，具体步骤如下：

（1）启动企业管理器并连接服务器，在上部的菜单栏中，单击【工具】→【向导】依次展开【数据库】目录树，单击【创建索引向导】项，在随后打开的创建索引向导的欢迎对话框中，单击【下一步】按钮。

（2）在系统弹出的创建索引对话框中（见图 7－3），分别选择要建索引的数据库和表，例如我们选择的数据库表分别是“教学管理数据库”及“学生表”，并单击【下一步】按钮。系统弹出当前索引信息对话框。在此，系统显示了所选对象的所有已建的索引，以供设计者了解情况。

（3）单击【下一步】按钮，弹出如图 7－4 所示的对话框中，列出了该表（学生表）的所有列，从中选择为其建索引的列（一个或多个）。如果选择一个列，则建立的是单列索引，否则是复合索引。如我们希望以“学号”建索引，则选择“学号”列，单击【下一步】按钮。

（4）系统随后弹出如图 7－5 所示的指定索引选项对话框，借助于该对话框，我们可以进一步确定索引选项，如确定是聚集索引还是非聚集索引（如果该表中已存在聚集索引，该项就不能再选择），是否是唯一索引等等。选择后单击【下一步】按钮。

（5）在系统弹出的图 7－6 所示的完成创建索引向导对话框中，给出了系统缺省的索引名称，设计者可以进行修改，单击【完成】按钮，并在随后的对话框中，单击【确定】即可。



图 7-3 选择数据库和表对话框



图 7-4 选择列对话框



图 7-5 指定索引选项对话框



图 7-6 完成创建索引向导对话框

2. 使用企业管理器建立索引

使用企业管理器建立索引的具体步骤如下：

(1) 启动企业管理器，依次展开【数据库】文件夹，展开要建索引的数据库(教学管理数据库)，单击【表】，用鼠标单击右边列表中要建索引的表名(如学生表)，在弹出的快捷菜单中，单击【所有任务】菜单中的【管理索引…】，如图 7-7 所示。

(2) 在系统弹出的管理索引对话框中(如图 7-8 所示)，可以在上部通过菜单选择要建索引的数据库和表。



图 7-7 企业管理器弹出的快捷菜单



图 7-8 管理索引对话框

(3) 单击对话框下部的【新建】按钮，打开图 7-9 所示的新建索引对话框，在该对话框中，可以指定索引的名称、被索引的列以及索引选项(是否是唯一索引、是否是聚集索引)等。



图 7-9　新建索引界面

(4) 单击下部的【编辑】按钮，进入编辑索引对话框，修改和编辑所选定的索引。

(5) 单击下部的【删除】按钮，可以删除当前选定的索引对话框，确认弹出的对话框即可。

这里应指定索引的名称、被索引的列以及索引选项等。

3. 使用 Transact-SQL 语言建立索引

在 Transact-SQL 中，使用 CREATE INDEX 语句来建立索引，其语法如下：

CREATE [UNIQUE] [CLUSTERED|NONCLUSTERED]

INDEX index_name ON table_name (column [ASC | DESC] [,...n])

其中的参数意义如下：

- UNIQUE：指示建立的索引类型为唯一索引。
- CLUSTERED：指示建立的索引类型为聚集索引。NONCLUSTERED 为非聚集索引。
- index_name：要建立的索引的名称。索引名称应当遵循 SQL Server 的命名规则，并保证同一表中的索引名称是唯一的。
- table_name：被索引的表名。
- column：被索引的列名。
- ASC 和 DESC：指定索引列的排序方式，默认为 ASC(升序)。

下面我们给出为班级表建立索引的例子。

【例 7-11】 为班级表建立索引，要求以“班号”列建索引，并按升序建非聚集索引。

解：具体实现命令如下：

```
CREATE NONCLUSTERED
INDEX 班级_学号 idex ON 班级表(学号)
```

7.2.3　索引的删除

我们可以通过企业管理器或 Transact-SQL 中的 DROP INDEX 语句删除索引。具体方法如下：

需要全本请访问 www.ertongbook.com

1. 使用企业管理器删除索引

在图 7-8 中,选中要删除的索引,然后单击【Delete】按钮,在弹出的对话框内选择【Yes】。

2. 使用 Transact-SQL 删除索引

在 Transact-SQL 中,使用 DROP INDEX 语句来删除索引,其语法如下:

DROP INDEX table_name . index_name [, . . . n]

其中参数意义如下:

- table_name:要删除索引的表名称。
- index_name:要删除的索引名称。

例如:

DROP INDEX 学生表. 学号_index

这一语句将删除学生表表中的学号_index 索引。

注意:通过设置主键约束或唯一约束所建立的索引不可删除,只能通过删除约束或删除表的方法来删除这类索引。

7.2.4 索引的应用

当创建数据库并优化其性能时,应该为数据查询所使用的列创建索引,索引的使用会为我们带来很多好处,但索引有利必有弊,在应用索引的过程中必须注意到:索引虽然可以加快数据检索的速度,但它会使数据的插入、删除和更新变慢。尤其是聚集索引,数据是按照逻辑顺序存放在一定的物理位置,当变更数据时根据新的数据顺序需要将许多数据进行物理位置的移动,这将增加系统的负担。对非聚集索引,数据更新时也需要更新索引页,因而会占用系统时间。因此建立索引时,应注意以下几点:

(1) 由于在一个表中使用太多的索引会影响数据库的性能,对于一个经常会改变的表应该尽量限制表只使用一个聚集索引和不超过 3～4 个非聚集索引,对事务处理特别繁重的表其索引应尽量不超过 3 个。

(2) 要注意建立索引的时机,数据更新量很大时,在更新之前,应删除索引,更新结束后,查询之前再建立索引。

(3) 要注意选择用来建立索引的列,以下的列较适合用来建立索引:

① 作表的主键或外键的列。

② 其值具有唯一性的列。

③ 经常被作为查询条件的列。

④ 被排序的列。

与之相反,以下列不适合用来建立索引。

① 在查询中很少被引用的列。

② 重复值较多的列。

③ 数据类型为 bit、text 或 image 的列。

数据库设计者必须权衡利弊,在利用索引所带来的好处的同时,尽量减少设计不合理的索引对数据库性能的负面影响。

7.3　数据查询

数据查询是数据库的核心操作，其操作是由 Transact-SQL 的 SELECT 语句完成的。与一般的编程语言相比，Transact-SQL 语言的过程性要低得多，因此其编程效率很高。读者看到的 SELECT 语句虽然不很复杂，但其功能很丰富，使用方式也很灵活。

7.3.1　SQL Server 查询分析器

SQL Server 的查询分析器是一个图形化的数据库编程接口，是 SQL Server 客户端应用程序的重要组成部分。查询分析器以自由的文本格式编辑 SQL 语句，语句中的保留字用彩色显示，并提供快速生成 SQL 代码的大量模板，同时使用图形化的方式显示执行 SQL 语句的逻辑步骤和 SQL 语句的执行效率评估。在我们学习用 Transact-SQL 语言对数据库中的数据进行操作和处理之前，应首先掌握和了解其查询分析器的基本应用方法。

利用查询分析器可完成以下工作：

(1) 可将在该窗口下用 Transact-SQL 语言编写的程序进行解释执行之后，存储在一个后缀为“.sql”的文件中，以供以后多次使用。

(2) 可快速地插入、修改、删除表中的数据。

(3) 跟踪执行存储过程。

启动查询分析器有两种方法：

(1) 单击 Click【开始】and【程序】，然后从【Microsoft SQL Server】中选择【查询分析器】即可。

(2) 从企业管理器中，单击【工具】，并从弹出的菜单中选择【查询分析器】即可。具体过程如图 7－10 所示。

打开后的查询分析器如图 7－11 所示。



图 7－10　查询分析器的启动

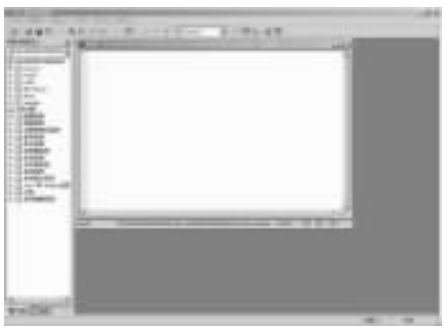


图 7－11　查询分析器的界面

在编辑窗口中可以编辑和调试用 Transact-SQL 语言编写的程序，包括存储过程、触发器等，并可进行查询优化和分析查询过程等操作。

如果需要保存编写好的程序，单击图 7－11 所示的菜单栏中保存按钮即可。在图 7－11 中的左边给出了系统所有的数据库及其对象(每个数据库所定义的表、触发器、存储过程、视

图、函数、用户定义的数据类型等等)、各种公用对象(包括配置函数、游标函数、时间和日期函数、数学函数、系统数据类型等等),以供编程时方便查找所需要的信息。

下面我们先从 SELECT 语句的基本语法入手,通过不同应用示例由浅入深,逐一说明其中每个子句的功能特点和使用方法。SELECT 语句的基本格式如下:

```
SELECT [ALL][DISTINCT]select_list
[INTO new_table_name]
FROM table_list
[WHERE search_conditions]
[GROUP BY group_by_list] [HAVING search_conditions]
[ORDER BY order_list [ASC | DESC] ]
```

SELECT 语句执行的过程简述如下:

(1) 首先根据 WHERE 子句的条件,从 FROM 子句指定的表或视图中找到符合条件的数据记录;

(2) 然后再根据 SELECT 子句中的选择列表达式,从这些记录中筛选出所需要的列的值作为查询结果;

(3) 若定义了 INTO 子句,则将查询结果保存到另一张表中;

(4) 若定义了 GROUP 子句,则将查询结果按分组表达式进行分组,若 GROUP 子句中含有 HAVING 子句,则只输出满足条件的组;

(5) 若定义了 ORDER 子句,则查询结果按 order_list 给出的表达式按照升序或降序进行排序并输出。

SELECT 语句可以完成简单的单表查询、复杂的连接查询、嵌套查询和集合查询。由于 SELECT 语句的复杂性,我们下面结合前述教学管理数据库,根据不同的查询需求,介绍 SELECT语句的用法。

7.3.2 单表查询

单表查询通常是指只涉及一个表的查询。可分如下几种情况讨论。

1. 选择表中的若干列

在表中选择列的查询实际上就是我们讨论过的投影运算。

(1) 结果集包括所有列

在 SELECT 后面使用“*”号,将查询表中的所有列。

【例 7-12】 查询学生表中的所有信息。

解:可由如下程序实现:

```
SELECT *
FROM 学生表
```

(2) 结果集包括部分列

在 SELECT 后面指定要在结果集中包含列的列名,列名之间用逗号分开。

【例 7-13】 查询学生表中的学号与姓名。

解:可由如下程序实现:

```
SELECT 学号,姓名
```

FROM 学生表

例 7－12 和例 7－13 的执行结果如图 7－12 所示。

(3) 消除结果集中重复行

表中是不存在重复记录值的，但一旦查询结果投影到若干列后，可能出现重复的行，如果在 SELECT 语句中使用 DISTINCT 关键字，查询结果中如果有重复值的行，则系统会自动进行去重处理。

【例 7－14】 查询全校中学生的籍贯情况。

解：可由如下程序实现：

SELECT 籍贯

FROM 学生表

执行结果集中会有重复的籍贯出现，要想去掉这些籍贯重复的行，应在 SELECT 语句中使用 DISTINCT 关键字：

SELECT DISTINCT 籍贯

FROM 学生表

两条语句执行结果的比较如图 7－13 所示。



图 7－12　例 7－12 和例 7－13 的执行结果



图 7－13　例 7－14 的执行结果

(4) 限制结果集中行的数目

缺省情况下，SELECT 语句查询的结果集包括表中满足条件的所有行。在实际应用中，我们往往只需要其中的一部分数据，这可以在 SELECT 语句中使用 TOP 关键字，以限制结果集中的行数。具体地说，就是在 SELECT 后面使用 TOP n[PERCENT]，指定返回查询结果的前 n 行数据。如果指定 PERCENT 关键词，则按照查询结果的 n％返回数据。

【例 7－15】 假定学生表中，出生日期已按升序排序，要求查找年龄最大的前 6 名学生的所有情况。

解：可以用下列方式完成：

SELECT TOP 6　*

FROM 学生表；

【例 7－16】 假定学生表中，出生日期已按升序排序，要求查找年龄最大的前 20％学生的

所有情况。

使用 PERCENT：

```
SELECT TOP 20 PERCENT *
FROM 学生表；
```

执行结果如图 7－14 所示。

(5) 使用列的别名

在显示结果中，为使列名便于阅读，可以为数据列指定列的别名。有 3 种方式：

(1) 别名＝列名；

(2) 列名 别名；

(3) 列名 AS 别名。

【例 7－17】 在课程表中，查询所有课程及所使用的教材，要求在结果集中将列名“教材”换为别名“使用教材”。

解：可由如下程序实现：

```
SELECT 课程名,教材 AS 使用教材
FROM 课程表；
```

其执行结果如图 7－15 所示。



图 7－14 例 7－15 和例 7－16 的执行结果



图 7－15 例 7－17 的执行结果

以上的用法实际上就是 SELECT 子句及其参数的应用。这里对 SELECT 子句作一小结。

SELECT 子句指定需要通过查询返回的表的列，其语法如下：

```
SELECT [ALL | DISTINCT ]
[TOP n [PERCENT] [WITH TIES] ]
<select_list>
其中<select_list> ::=
  { *
    | { table_name | view_name | table_alias }. *
    | { column_name | expression | IDENTITYCOL | ROWGUIDCOL }
    [[AS] column_alias ]
    | column_alias= expression
```

}[,...n]

各参数意义如下：

- ALL：指明查询结果中可以显示值相同的列。ALL 是系统默认的。
- DISTINCT：指明查询结果中如果有值相同的列则只显示其中的一列。

注意：对 DISTINCT 来说，NULL 被认为是相同的值。

- TOP n [PERCENT]：指定返回查询结果的前 n 行数据。如果指定 PERCENT 关键词，则按照查询结果的 n%返回数据。
- WITH TIES：返回附加的行，这些行中 ORDER BY 列中的值与 TOP n(PERCENT)指定的行中的最后一行的 ORDER BY 列的值相同。此选项只能在使用了 ORDER BY 子句后才能使用。
- select_list：是所要查询的表中列的集合。多个列之间用逗号分开。
- *：使用该通配符返回所有列。
- table_name | view_name | table_alias. *：限制通配符 * 的作用范围。
- column_name：指定返回的列名。
- expression：表达式。可以为列名、常量、函数或它们的组合。
- IDENTITYCOL：返回标识列。
- ROWGUIDCOL：返回行全局唯一标识列。
- column_alias：是查询结果集内替换列名的别名。

2. 选择表中的若干列

用户在查询数据库时往往只需要一部分满足条件的信息。在这种情况下就需要在 SELECT语句中加入条件以选择所需的记录，这时就用到 WHERE 子句。WHERE 子句中的条件由各种运算符和 NOT、AND、OR 所组成。这些运算符主要有：

- 比较大小：>、>=、<、<=、=、<>、!=、!>、!<，以及 NOT 与上述运算符的组合
- 确定范围：BETWEEN AND、NOT BETWEEN AND
- 模式匹配：LIKE、NOT LIKE
- 空值判断：IS NULL、IS NOT NULL
- 隶属判断：IN、NOT IN
- 逻辑判断：AND、OR、NOT

下面我们分类结合例子来学习 WHERE 子句的使用方法。

(1) 比较大小的条件子句

比较大小的条件子句在查询中用的比较多，下面举例说明。

【例 7-18】 查询成绩表中成绩低于 60 分的所有记录。

解：可由如下程序实现：

```
SELECT *
FROM 成绩表
WHERE 成绩<60;
```

【例 7-19】 查询计算机系所有女生的基本信息。

解：可由如下程序实现：

需要全本请访问 www.ertongbook.com