第3章 查询语句

本章介绍 SQL 的具体语句及其应用。简单来说，SQL 包括数据定义语言（Data Definition Language，DDL）、数据操纵语言（Data manipulation Language，DML）和数据控制语言 DCL 三个方面的功能。其中，DDL 包括对数据库结构及表结构的定义和修改，DML 包括对数据的查询、更新、删除等功能，DCL 主要是对数据权限的控制功能。

由于在具体应用中，数据库的查询操作是使用最频繁的，因此 SQL 的数据查询语句功能非常强大，掌握 SQL 首先必须先掌握其查询语句的使用。本章及后续章节将详细介绍 SQL 查询语句的应用，本章着重介绍查询语句的简单使用，通过对 SQL 中的查询语句 SELECT 的结构和执行过程的讲解，对查询的基本结构做一个具体的介绍。简单查询是指对关系的简单投影操作，即对二维表的列查询，或只包含简单查询条件的查询。

3.1 SELECT 语句

SELECT 语句是 SQL 提供的唯一一个标准查询语句，对数据库的数据查询基本上都是使用该语句。SELECT 语句提供了非常丰富的查询功能，具体都体现在其参数中，本节将重点介绍 SELECT 的语句结构和执行过程。

3.1.1 SELECT 语句结构

在数据库中，数据查询是通过 SELECT 语句来完成的。SELECT 语句可以从数据库中按用户要求检索数据，并将查询结果以表格的形式返回。

SELECT 语句的完整语法结构非常复杂，要理解其中每一个子句是一个非常冗长、枯燥的过程。此处先列出 SELECT 语句所有的子句，主要如下：

```
SELECT statement :: =
< query_expression >
[ ORDER BY { order_by_expression | column_position [ ASC | DESC ] } [, … n] ]
[ COMPUTE { { AVG | COUNT | MAX | MIN | SUM } (expression) } [, … n]
[ BY expression [, … n] ] ]
[ FOR { BROWSE | XML { RAW | AUTO | EXPLICIT }
[ , XMLDATA ]
[ , ELEMENTS ]
```

```
[ , BINARY base64 ] }
[ OPTION (< query_hint > [, … n]) ]
< query expression > : : =
{ < query specification > | (< query expression >) }
[UNION [ALL] < query specification | (< query expression >) [ … n] ]
< query specification > : : =
SELECT [ ALL | DISTINCT ]
[ {TOP integer | TOP integer PERCENT} [ WITH TIES] ]
< select_list >
[ INTO new_table ]
[ FROM {< table_source >} [, … n] ]
[ WHERE < search_condition > ]
[ GROUP BY [ALL] group_by_expression [, … n]
[ WITH { CUBE | ROLLUP } ] ]
[ HAVING < search_condition > ]
```

读者可以发现,要准确理解其每一个子句是很困难的。本章将先就其用于简单查询中的相关子句做详细讲解。后面将继续向读者介绍该语句的相关内容。SELECT 语句用于基本查询的主要子句结构如下:

```
SELECT 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 ] ]
```

在上述结构中,关键字 SELECT、INTO 和 FROM 是必需的。一般来说,由 SELECT 子句、FROM 子句和可选的 INTO 子句组成的查询称为简单查询,其格式如下:

```
SELECT select_list
[INTO new_table_name ]
FROM table_list
```

其余的参数中,WHERE 子句后加条件语句,用于限制选取的记录条件; GROUP BY 子句后加分组关键字,用于将查询结果按某一字段进行分组显示; HAVING 子句后加条件,用于对分组进行条件限制; ORDER BY 子句后加排序关键字,用于将查询结果按某一字段进行升序或降序的排列,其中 ASC 表示升序、DESC 表示降序。这些参数将在后续章节中做详细介绍,本章将重点介绍简单查询的结构以及其相关子句。

3.1.2 SELECT 语句执行过程

在 SQL 中,语句的处理都遵循一定的执行顺序。一般来说,SQL 语句处理分两个或三个阶段,每个语句从用户进程传给服务器进程进行分析然后执行。如果是 SELECT 语句,则还需要将结果返回给用户。下面简要介绍 SQL 的执行流程。

(1) 分析(PARSE),分析是 SQL 语句处理的第一步,主要进行如下工作:

- 检查语法和根据字典来检查表名、列名。
- 确定用户执行语句的权限。
- 为语句确定最优的执行计划。
- 从 SQL 区中找出语句。

(2) 执行(EXECUTE)。

执行阶段执行的是被分析过的语句,对于 UPDATE、DELETE 语句,DBMS 先锁住有关的行,有些 DBMS 还需查找数据是否在数据缓冲区里,如果不在则从数据文件中将数据读到数据缓冲区中。

(3) 检索(FETCH)。

如果是 SELECT 语句,需要进行检索操作。执行结束,将数据返回给用户。SELECT 语句处理的步骤和流程将在下面具体介绍。

在 SQL 中,由于 SELECT 语句的子句众多,其执行也最为复杂。因此针对 SELECT 语句,SQL 规定了该语句的执行有着自己的过程。对于存在 FROM、WHERE、GROUP BY 和 HAVING 等子句的简单 SELECT 语句来说,其完整的执行过程如下:

- FROM 子句组装来自不同数据源的数据。
- WHERE 子句基于指定的条件对记录行进行筛选。
- GROUP BY 子句将数据划分为多个分组。
- 使用聚集函数进行计算。
- 使用 HAVING 子句筛选分组。

下面通过几种常用的 SELECT 语句格式介绍其执行过程。

- 只含有 FROM 子句的 SELECT 语句,其格式如下:

```
SELECT 列列表
FROM 表列表名/视图列表名
```

该种类型的 SELECT 语句最为简单,直接通过 FROM 子句组装来自不同数据源的数据,再执行 SELECT 子句即可。

- 只含有 FROM 和 WHERE 子句的 SELECT 语句,其格式如下:

```
SELECT 列列表
FROM 表列表名/视图列表名
WHERE 条件
```

该种类型的 SELECT 语句的执行顺序为:先 WHERE 后 SELECT。

- 含有 GROUP BY 和 HAVING 子句的 SELECT 语句,其格式如下:

```
SELECT 列列表
FROM 表列表名/视图列表名
WHERE 条件
GROUP BY (列列表)
HAVING 条件
```

该种类型的 SELECT 语句的执行顺序为：先 WHERE 再 GROUP BY 再 HAVING 后 SELECT。

• 含有 GROUP BY、HAVING 和 ORDER BY 子句的 SELECT 语句，其格式如下：

```
SELECT 列列表
FROM 表列表名/视图列表名
WHERE 条件
GROUP BY (列列表)
HAVING 条件
ORDER BY 列列表
```

该种类型的 SELECT 语句的执行顺序为：先 WHERE 再 GROUP 再 HAVING 再 SELECT 后 ORDER。

• 含有 JOIN 子句的 SELECT 语句，其格式如下：

```
SELECT 列列表
FROM 表 1
JOIN 表 2
ON 表 1.列 1 = 表 2.列 1…JOIN 表 N ON 表 N.列 1 = 表(N - 1).列 1
WHERE 表 1.条件 AND 表 2.条件…表 N.条件
```

该种类型的 SELECT 语句的执行顺序为：先 JOIN 再 WHERE 后 SELECT。

上述 5 种格式基本包含了 SELECT 语句用于简单查询的所有格式，再后续的章节中，会逐渐提到这几种查询格式，读者可注意其执行过程。

3.2 列查询

列查询是数据查询中最简单的一类查询，其使用的只有 SELECT 语句中必需的几个关键字。此处介绍的列查询不包括 WHERE 子句，即没有限制条件，分为单列查询、多列查询和所有列查询。在介绍具体的查询前，先简要介绍 SELECT 关键字及其子句。

3.2.1 SELECT 子句

SELECT 子句指定需要通过查询返回的表的列，其完整的语法如下：

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

其中，各参数的说明如下：

- 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。对 DISTINCT 选项来说，Null 值被认为是相同的值。
- TOP n [PERCENT]——指定返回查询结果的前 n 行数据。如果 PERCENT 关键字指定的话，则返回查询结果的前百分之 n 行数据。
- WITH TIES——此选项只能在使用了 ORDER BY 子句后才能使用当指定此项时，除了返回由 TOP n (PERCENT) 指定的数据行外，还要返回与 TOP n (PERCENT)返回的最后一行记录中由 ORDER BY 子句指定的列的列值相同的数据行。
- select_list select_list——是所要查询的表的列的集合，多个列之间用逗号分开。
- * ——通配符，返回所有对象的所有列。
- table_name | view_name | table_alias. —— * 限制通配符 * 的作用范围。凡是带 * 的项，均返回其中所有的列。
- 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 IDENTITY 列。如果 FROM 子句中有多个表含有 IDENTITY 列，则必须在 IDENTTYCOL 选项前加上表名，如 Table1. IDENTITYCOL。
- ROWGUIDCOL 返回表的 ROWGUIDCOL 列。同 IDENTITYCOL 选项相同，当要指定多个 ROWGUIDCOL 列时，选项前必须加上表名，如 Table1. ROWGUIDCOL。
- column_alias：在返回的查询结果中用此别名替代列的原名。column_alias 可用于 ORDER BY 子句，但不能用于 WHERE GROUP BY 或 HAVING 子句。如果查询是游标声明命令 DECLARE CURSOR 的一部分，则 column_alias 还不能用于 FOR UPDATE 子句。

上述的 SELECT 子句过于复杂，在实际应用中，如下的格式用得较多：

```
SELECT select_list
```

其中，select_list 描述结果集的列，其是一个逗号分隔的表达式列表。每个表达式同时定义格式(数据类型和大小)和结果集列的数据来源。通常，每个选择列表表达式都是对数据所在的源表或视图中的列的引用，但也可能是对任何其他表达式(例如，常量或 SQL 函数)的引用。在选择列表中使用 * 表达式可指定返回源表的所有列。

为方便 SELECT 语句的实例讲解，此处设定数据表，如表 3.1 和表 3.2 所示。表 3.1 表是一个学生表 STUDENT，包含学号(SNO)、姓名(SNAME)、性别(SGENTLE)、年龄(SAGE)、出生年月(SBIRTH)和系部(SDEPT)六个字段。

表 3.1 学生表 STUDENT

SNO	SNAME	SGENTLE	SAGE	SBIRTH	SDEPT
990001	张三	男	20	1987-8-4	计算机
990002	陈林	女	19	1988-5-21	外语
990003	吴忠	男	21	1986-4-12	工商管理
990005	王冰	女	20	1987-2-16	体育
990012	张忠和	男	22	1985-8-28	艺术
990026	陈维加	女	21	1986-7-1	计算机
990028	李莎	女	21	1986-10-21	计算机

表 3.2 是一个课程表 COURSE，主要包含课程号(CNO)、课程名称(CNAME)和学分(CGRADE)三个字段。

表 3.2 课程表 COURSE

CNO	CNAME	CGRADE
001	计算机基础	2
003	数据结构	4
004	操作系统	4
006	数据库原理	4
007	软件工程	4

3.2.2 单列查询

单列查询在前面的章节中也提到过不少，其是查询中最简单的一种，其操作的对象是数据表中的某一个字段，返回的是表中的某一个列。其基本格式如下：

```
SELECT 列名
FROM 表名/视图名
```

单列查询中的列名只能是一个字段名，而 FROM 子句中既可带基本表，也可带视图，具体内容在后续章节还将介绍。例如，针对上述学生表 STUDENT，查询其中所有学生的姓名。

实现代码如下：

```
SELECT SNAME
FROM STUDENT
```

执行上述查询后，其查询结果如图 3.1 所示。

从上述示例可以看出，单列查询显示的结果简单明了，其显示的是基本表中所有记录的指定字段值。用户需要预先知道表的结构和字段名。

图 3.1 单列查询应用

3.2.3 多列查询

多列查询和单列查询是对应的，其操作对象为数据表中的

某几个字段，返回的是表中的多个列。其基本格式如下：

```
SELECT 列名 1,列名 2,…,列名 N
FROM 表列表名/视图列表名
```

多列查询中的列名可以是多各个字段名，而 FROM 子句中同样既可带基本表也可带视图，而且其允许多个基本表和多个视图。

例如，针对上述学生表 STUDENT，查询其中所有学生的姓名、性别和所属系别，其实现代码如下，查询结果如图 3.2 所示。

```
SELECT SNAME,SGENTLE,SDEPT
FROM STUDENT
```

	SNAME	SGENTLE	SDEPT
▶	张三	男	计算机
	陈林	女	外语
	吴忠	男	工商管理
	王冰	女	体育
	张忠和	男	艺术
	陈维加	女	计算机
	李莎	女	计算机

图 3.2 多列查询应用

与单列查询类似，多列查询显示的结果是基本表中所有记录的指定字段值，结果简单明了，但是用户需要预先知道表的结构和字段名。

3.2.4 对数据列进行算术运算

上述 SELECT 的简单查询介绍的都是对基本表或视图中的内容进行查询，并将数据返回显示，数据表中的值是什么，返回的值就是什么。在实际应用中，用户经常需要对数据表中数据进行简单的运算后再显示出来。

例如，有员工工资表（EMPLOYEE）如表 3.3 所示，其有员工号（NUMBER）和工资（SALARY）两个字段，现在需要对其中每个员工的工资加薪 15%，并将加薪后的结果返回显示。

表 3.3 员工工资表

员工号	工资
0001	852.00
0002	792.00
0003	1054.00
0006	684.00

需要对每个员工加薪 15%后显示结果，可以使用 SELECT 语句来实现，这就需要对数据列进行算术运算后的 SELECT 查询语句。其具体实现代码如下：

```
SELECT NUMBER,SALARY * 1.15
FROM EMPLOYEE
```

同样,针对课程表 COURSE,如要实现将其所有课程的学分都加 1,并将运算结果显示出来,可采用如下代码:

```
SELECT CNO,CNAME,CGRADE + 1
FROM COURSE
```

在 SQL Server 2008 的查询分析器中执行上述语句,其结果显示如图 3.3 所示。

	CNO	CNAME	Expr1
▶	001	计算机基础	3
	003	数据结构	5
	004	操作系统	5
	006	数据库原理	5
	007	软件工程	5

图 3.3 对数据列进行算术运算

需要注意的是,对数据列进行算术运算的 SELECT 子句并不会改变原基本表或视图中的数据,一旦查询释放,其运算结果也被释放,而不会写入到原表或视图中,数据表中的数据仍然如图 3.4 所示。这是因为 SELECT 语句只是查询语句,而不是数据操作的 SQL 语句。

	CNO	CNAME	CGRADE
▶	001	计算机基础	2
	003	数据结构	4
	004	操作系统	4
	006	数据库原理	4
	007	软件工程	4

图 3.4 原数据表不变

3.2.5 为数据列指定别名

在 3.2.4 节中对数据列进行算术运算显示的结果图 3.3 中,可以发现,其经过运算后的列不会以原有的表或视图的字段名显示出来,而是显示默认的“无列名”。为了查询后的结果更具有可读性,通常可以为该类查询结果指定通俗易懂的别名,其指定格式如下:

```
SELECT 列名 1, 别名 1,列名 2,别名 2,…,列名 N,别名 N
FROM 表列表名/视图列表名
```

用户可根据显示需要,可以对表或视图的一个显示字段指定别名,也可对多个显示字段指定,指定格式只需在原字段名即表名后加空格再加别名即可。

例如,在上一小节中将学分加 1 的示例中,如果将学分加 1 后的列指定别名 GRADE1,表示更改后的学分,则实现代码可如下:

```
SELECT CNO,CNAME,CGRADE + 1 GRADE1
FROM COURSE
```

执行结果如图 3.5 所示。可以看出,计算后的学分字段由“无列名”变成指定的别名 GRADE1,这就让用户更容易理解查询结果。

	CNO	CNAME	GRADE1
▶	001	计算机基础	3
	003	数据结构	5
	004	操作系统	5
	006	数据库原理	5
	007	软件工程	5

图 3.5 指定别名

不仅仅是经过算术运算后的列可以指定别名,用户可以为任意数据表中的字段名指定想要的别名。例如,以中文列名显示学分表 COURSE 中的所有数据记录。实现语句如下:

```
SELECT CNO 课程号,CNAME 课程名称,CGRADE 学分
FROM COURSE
```

上述 SELECT 语句的执行结果如图 3.6 所示。

	课程号	课程名称	学分
▶	001	计算机基础	2
	003	数据结构	4
	004	操作系统	4
	006	数据库原理	4
	007	软件工程	4

图 3.6 中文列名输出

3.2.6 查询所有列

在实际应用中,用户经常不清楚基本表或视图的结构和具体字段名称,那么上述这些针对具体列的查询就无法实现。因此,SQL 提供了一个查询所有列的 SELECT 语句形式,即使用户不清楚数据表的结构,通过该语句可显示出其所有字段名和所有记录。查询所有列的 SELECT 语句格式为:

```
SELECT *
FROM 表名/视图名
```

通过该语句显示出的字段名都是基本表或视图的真实字段名称,不能为其设置别名。例如,下列语句列出学生表 STUDENT 的所有记录。

```
SELECT *
FROM STUDENT
```

执行该语句后,返回的结果为 STUDENT 表中所有数据记录,其列名即该表的真实字

段名称，运行结果如图 3.7 所示。

SNO	SNAME	SGENTLE	SAGE	SBIRTH	SDEPT
990001	张三	男	20	1987-08-04 00:...	计算机
990002	陈林	女	19	1988-05-21 00:...	外语
990003	吴忠	男	21	1986-04-12 00:...	工商管理
990005	王冰	女	20	1987-02-16 00:...	体育
990012	张忠和	男	22	1985-08-28 00:...	艺术
990026	陈维加	女	21	1986-07-01 00:...	计算机
990028	李莎	女	21	1986-10-21 00:...	计算机

图 3.7　查询所有列

3.2.7　使用 DISTINCT 关键字

在完整的 SELECT 子句中，可以发现，其后可带有 ALL 或者是 DISTINCT 参数。其中，默认的是 ALL 参数，表示选取符合条件的所有数据记录。而 DISTINCT 参数表示选取符合条件的数据记录，并当选取的记录中有重复记录时，去除重复记录。使用 DISTINCT 关键字的 SELECT 语句在实际应用中是使用非常多的，其列查询的格式如下：

```
SELECT DISTINCT 列名 1, 别名 1,列名 2 别名 2,…,列名 N 别名 N
FROM 表列表名/视图列表名
```

例如，在课程表 COURSE 中，如其数据记录中，有两条相同的记录，如表 3.4 所示。

表 3.4　有重复记录的课程表 COURSE

CNO	CNAME	CGRADE
001	计算机基础	2
003	数据结构	4
004	操作系统	4
004	操作系统	4
006	数据库原理	4
007	软件工程	2

在表 3.4 中，CNO 为“004”的记录是重复记录，如果使用选取所有列的 SELECT 语句来返回该表中所有数据，则使用如下语句：

```
SELECT *
FROM COURSE
```

其返回结果如图 3.8 所示。

读者可发现，返回的结果中共有 6 条记录，包含 1 条重复的 CNO 为“004”的记录。这是因为上述语句没有使用 DISTINCT 关键字，系统默认的是 ALL 参数，这在实际应用中往往是不需要的，去除重复的记录语句如下，其返回结果如图 3.9 所示。

```
SELECT DISTINCT *
FROM COURSE
```

	CNO	CNAME	CGRADE
▶	001	计算机基础	2
	003	数据结构	4
	004	操作系统	4
	004	操作系统	4
	006	数据库原理	4
	007	软件工程	4

图 3.8 带有重复记录的查询

	CNO	CNAME	CGRADE
▶	001	计算机基础	2
	003	数据结构	4
	004	操作系统	4
	006	数据库原理	4
	007	软件工程	4

图 3.9 使用 DISTINCT 去除重复记录

需要注意的是,DISTINCT 关键字并不是去除数据表中的重复记录,而是去除查询结果中的重复记录。例如,下列两条查询语句分别返回课程表 COURSE 的学分字段。

其中,如下语句返回的是所有记录的学分数据,该语句返回结果如图 3.10 所示。

```
SELECT CGRADE
FROM COURSE
```

如下语句返回的是去除查询中重复记录的学分数据,该语句返回结果如图 3.11 所示,读者可发现,即使数据表中这些记录不是重复的,但是查询中这些记录是重复的,因此使用关键字 DISCTINCT 后,这些重复记录被去除了。

	CGRADE
▶	2
	4
	4
	4
	4

图 3.10 字段的重复值

	CGRADE
▶	2
	4

图 3.11 去除字段重复值

```
SELECT DISTINCT CGRADE
FROM COURSE
```

关键字 DISTINCT 除了使用到此处的简单查询中,还可以用到各种复合查询中,这在后续章节中还将介绍到。一般来说,只要涉及了查询,都可以使用到 DISTINCT 关键字。

3.2.8 使用 TOP 关键字

在 SELECT 子句中,读者可以发现它还能添加 TOP 参数。该参数的作用是用来限制返回到结果集中的记录的数目,在 SELECT 中,其可以使用下面两种表达方式:

- TOP N。
- TOP N PERCENT。

TOP N 格式表示的是显示返回记录中的前 N 条记录。例如,在如表 3.1 所示的学生表 STUDENT 中,一共有 7 条记录,使用如下语句可以取得其所有的 7 条记录。

```
SELECT *
FROM STUDENT
```

因为该语句是返回所有数据，因此符合条件的是所有的 7 条数据记录。但是，如果实际应用中不需要显示这么多，例如只需要显示前面 5 条，那么就需要使用 TOP 关键字来实现，TOP 关键字的使用格式如下所示：

```
SELECT TOP N 列名 1, 别名 1,列名 2,别名 2,…,列名 N,别名 N
FROM 表列表名/视图列表名
```

例如，显示学生表 STUDENT 的前 5 条记录，可使用如下语句实现，其返回结果如图 3.12 所示。

```
SELECT TOP 5 *
FROM STUDENT
```

	SNO	SNAME	SGENTLE	SAGE	SBIRTH	SDEPT
▶	990001	张三	男	20	1987-08-04 00:...	计算机
	990002	陈林	女	19	1988-05-21 00:...	外语
	990003	吴忠	男	21	1986-04-12 00:...	工商管理
	990005	王冰	女	20	1987-02-16 00:...	体育
	990012	张忠和	男	22	1985-08-28 00:...	艺术

图 3.12　显示前 5 条记录

TOP N PERCENT 格式则返回查询结果的前百分之 N 行数据，取整数。例如，下列语句表示的是返回学生表 STUDENT 中的前 5%行的数据。

```
SELECT TOP 5 PERCENT *
FROM STUDENT
```

由于学生表 STUDENT 中一共只有 7 条记录，上述查询语句中取所有记录也只有 7 条，根据显示前 5%行，还不到 1 条记录。此处，SQL 规定取整数，其返回结果如图 3.13 所示。

	SNO	SNAME	SGENTLE	SAGE	SBIRTH	SDEPT
▶	990001	张三	男	20	1987-08-04 00:...	计算机

图 3.13　显示前 5%行的记录

3.3　INTO 子句

INTO 子句用于把查询结果存放到一个新建的表中。SELECT…INTO 句式不能与 COMPUTE 子句一起使用。INTO 子句的使用语法如下：

```
INTO new_table
```

参数 new_table 指定了新建的表的名称。新表的列由 SELECT 子句中指定的列构成，新表中的数据行是由该查询的结果指定的。例如，将学生表 STUDENT 中所有学生的姓名、性别、年龄和所属系部 4 个字段取出，放入新表 STU 中，其实现代码如下：

```
SELECT SNAME,SGENTLE,SAGE,SDEPT
INTO STU
FROM STUDENT
```

在 SQL Server 2008 的查询分析器中执行上述代码，其执行结果如图 3.14 所示。

图 3.14 INTO 子句执行结果

读者可以看到，上述语句执行后系统并没有给出新的 STU 表的结构和数据，只返回该语句执行造成的记录操作的数目。如需要查看新表 STU 的结构和数据，可使用下列代码：

```
SELECT SNAME,SGENTLE,SAGE,SDEPT
FROM STU
```

或者

```
SELECT *
FROM STU
```

执行上述代码后的返回结果如图 3.15 所示。

SNAME	SGENTLE	SAGE	SDEPT
张三	男	20	计算机
陈林	女	19	外语
吴忠	男	21	工商管理
王冰	女	20	体育
张忠和	男	22	艺术
陈维加	女	21	计算机
李莎	女	21	计算机

图 3.15 INTO 子句创建的新表 STU

需要注意的是，如果 SELECT 子句中指定了计算列，则一定要为该字段指定别名。例如，将学生表 STUDENT 中所有学生的姓名、性别、年龄和所属系部 4 个字段取出，并将所有学生的年龄加 1 后放入新表 STU2 中，其实现代码如下：

```
SELECT SNAME,SGENTLE,SAGE + 1 SAGE1,SDEPT
INTO STU2
FROM STUDENT
```

上述语句的 SAGE+1 为一个计算列，此处为其指定了别名 SAGE1，该别名在新建的 STU2 表中对应的列不是计算列，而是一个实际存储在表中的列，其中的数据由执行 SELECT…INTO 语句时计算得出。因此，上述语句执行的结果为创建了新表 STU2，该表的结构和数据记录通过执行下列语句返回，执行结果如图 3.16 所示。

```
SELECT *
FROM STU2
```

	SNAME	SGENTLE	SAGE1	SDEPT
▶	张三	男	21	计算机
	陈林	女	20	外语
	吴忠	男	22	工商管理
	王冰	女	21	体育
	张忠和	男	23	艺术
	陈雏加	女	22	计算机
	李莎	女	22	计算机

图 3.16　INTO 子句创建包含计算列指定别名的新表 STU2

需要注意的是，SQL Server 2008 的 SQL 解释器中，使用 INTO 子句创建的都是基本表而非临时表，在许多关系数据库管理系统中，INTO 子句创建的是临时表。

此外，通过 INTO 子句，用户可以复制表的结构和数据。例如，将学生表 STUDENT 中的表结构和所有数据记录复制到新表 STU 中，使用如下代码可以实现：

```
SELECT *
INTO STU
FROM STUDENT
```

INTO 子句的使用比较灵活，在许多应用场合中，灵活运用 INTO 子句能够为解决一些特殊问题提供好的方法。读者可仔细理解其用法。

3.4　FROM 子句

FROM 子句指定需要进行数据查询的表，只要 SELECT 子句中有要查询的列，就必须使用 FROM 子句，因此，FROM 子句是 SELECT 查询语句中必不可少的子句。在 SELECT 语句中，FROM 子句主要用于指定 SELECT 语句查询及与查询相关的表或视图，在 FROM 子句中最多可指定 256 个表或视图，之间用逗号分隔。

3.4.1　FROM 子句语法

在前面提到的应用中，FROM 子句的应用非常简单，例如，返回学生表 STUDENT 的

所有数据记录的 SELECT 语句如下：

```
SELECT *
FROM STUDENT
```

这些语句中的 FROM 往往只加一个或多个基本表、视图的名称，用于指定 SELECT 子句指定的列名所属的数据表。事实上，FROM 子句的完整语法如下：

```
FROM {< table_source >} [, … n]
< table_source > :: =
table_name [ [AS] table_alias ] [ WITH ( < table_hint > [, … n]) ]
| view_name [ [AS] table_alias ]
| rowset_function [ [AS] table_alias ]
| OPENXML
| derived_table [AS] table_alias [ (column_alias [, … n] ) ]
| < joined_table >
< joined_table > :: =
< table_source > < join_type > < table_source > ON < search_condition >
| < table_source > CROSS JOIN < table_source >
| < joined_table >
< join_type > :: =
[ INNER | { { LEFT | RIGHT | FULL } [OUTER] } ]
[ < join_hint > ]
JOIN
```

其中，各参数说明如下：

- table_source——指明 SELECT 语句要用到的表、视图等数据源。
- table_name [[AS] table_alias]——指明表名和表的别名。
- view_name [[AS] table_alias]——指明视图名称和视图的别名。
- rowset_function [[AS] table_alias]——指明行统计函数和统计列的名称。
- OPENXML——提供一个 XML 文档的行集合视图。
- WITH ([,…n])——指定一个或多个表提示。通常 SQL Server 的查询优化器会自动选取最优执行计划，除非是特别有经验的用户，否则不用此选项。
- derived_table [AS] table_alias——指定一个子查询，从数据库中返回数据行。
- column_alias——指明列的别名，用于替换查询结果中的列名。
- joined_table——指定由连接查询生成的查询结果。有关连接与连接查询的介绍参见第 5 章。
- join_type——指定连接查询操作的类型。
- INNER——指定返回两个表中所有匹配的行。如果没有 join_type 选项，此选项就为系统默认。
- LEFT [OUTER]——返回连接查询左边的表中所有的相应记录，而右表中对应于左表无记录的部分，用 NULL 值表示。
- RIGHT [OUTER]——返回连接查询右边的表中所有的相应记录，而左表中对应于右表无记录的部分，用 NULL 值表示。

- FULL [OUTER]——返回连接的两个表中的所有记录。无对应记录的部分用NULL值表示。
- join_hint——指定一个连接提示或运算法则。如果指定了此选项，则INNER LEFT RIGHT或FULL选项必须明确指定。通常SQL Server的查询优化器会自动选取最优执行计划，除非是特别有经验的用户，否则最好不用此选项。join_hint的语法如下：

```
<join_hint> :: = { LOOP | HASH | MERGE | REMOTE }
```

其中，LOOP | HASH | MERGE选项指定查询优化器中的连接是循环、散列或合并的。REMOTE选项指定连接操作由右边的表完成。当左表的数据行少于右表，才能使用REMOTE选项。当左表和右表都是本地表时，此选项不必使用。

- JOIN：指明特定的表或视图将要被连接。
- ON <search_condition>：指定连接的条件。
- CROSS JOIN：返回两个表交叉查询的结果。

FROM子句的复杂应用主要在表的连接和多表查询上，该部分内容在第4.6节和第5章中将详细讲解，读者在此只需大概了解FROM子句的格式。

3.4.2 表的别名

在此节，读者主要需要了解的是表的别名的使用。表的别名与列的别名不同，列的别名用于显示的时候方便用户理解其字段内容，提高易读性。而表除了可以提高语句的易读性外，还能将SQL语句变得更简短精练。此外，灵活运行表的别名还可以将一个表当成多个表使用，在一些特殊应用中会有意想不到的好处。

在FROM子句中，可用以下两种格式为表或视图指定别名：

- 表名 as 别名。
- 表名 别名。

例如，在选取学生表STUDENT中所有学生的学号、姓名、性别、年龄和所属系部等字段的语句中，给STUDENT表指定别名S1，实现语句如下：

```
SELECT SNO, SNAME, SGENTLE, SAGE, SDEPT
FROM STUDENT AS S1
```

执行上述语句后，表STUDENT中的数据不变，而别名S1表中的数据记录则根据返回的结果，如图3.17所示。

当然，根据别名的书写格式，上述语句可改写为：

```
SELECT SNO, SNAME, SGENTLE, SAGE, SDEPT
FROM STUDENT S1
```

其执行效果相同。此外，SELECT不仅能从表或视图中检索数据，其还能够从其他查询语句所返回的结果集合中查询数据。

	SNO	SNAME	SGENTLE	SAGE	SDEPT
▶	990001	张三	男	20	计算机
	990002	陈林	女	19	外语
	990003	吴忠	男	21	工商管理
	990005	王冰	女	20	体育
	990012	张忠和	男	22	艺术
	990026	陈维加	女	21	计算机
	990028	李莎	女	21	计算机

图 3.17 别名为 S1 的表数据记录

3.5 小结

本章介绍 SQL 语句的相关内容，本章介绍的是简单查询。在介绍简单查询的实现之前，给出了 SELECT 语句的完整格式，让读者对 SELECT 语句的子句做一个大致了解。接着着重讲解查询的实现，简单查询使用的是 SELECT 语句，该语句只包含 SELECT 子句、INTO 子句和 FROM 子句。本章是 SQL 的查询语句 SELECT 语句的基础，读者应熟练掌握该章的数据查询思路，包括 DISTINCT 和 TOP 等关键字的使用，为后续章节的学习打下好的基础。