SELECT

功能说明：从数据库中检索数据行。

语　　法：SELECT [DISTINCT|ALL] select_list
FROM table_name_source [WHERE search_condition]
[GROUP BY group_by_expression]
[HAVING search_condition]
[ORDER BY order_expression[ASC|DESC]]

参数说明：

关键字	含　义
DISTINCT	删除查询结果中相同的行
ALL	默认的关键字。返回查询结果中的所有行
FROM	查询的数据源表
WHERE	列出搜索标准，用于选择要显示的数据行
GROUP BY	分组查询结果。根据group_by_expression中列出的列，归纳信息类型，汇总相关数据
HAVING	列出另外的行选择标准，以便根据GROUP BY子句产生的结果筛选行
ORDER BY	按照ORDER BY子句中的说明，对查询结果进行排列。这里的说明就是ASC和DESC关键字。ASC关键字表示升序排列结果，DESC关键字表示降序排列结果。如果没有指定任何一个关键字，那么ASC就是默认的关键字

使用要点：从SELECT语句可以看出，必需的子句只有SELECT子句和FROM子句，其他的子句都是可选的。

范　　例：

```
SELECT Tname From Teacher
```

运行结果：SELECT语句返回了Tname列的所有信息。

DISTINCT

功能说明：去除查询结果中的重复记录。

语　　法：SELECT DISTINCT column FROM table_name

使用要点：DISTINCT关键字去除column列重复的信息。如果SELECT子句查询的列为多列，那么只有这些列的信息重复时才被去除。

范　　例：

```
SELECT DISTINCT Tname From Teacher
```

运行结果：从Teacher表中去除Tname重复信息，即去除了查询结果中的重复值。

ORDER BY

功能说明：对查询结果进行排序。

语　　法：SELECT column1,column2…FROM table_name
ORDER BY column N

使用要点：ORDER BY子句一定要放在所有子句的最后。

范　　例：

① SAL字段的升序排列记录

```
SELECT TNAME,TNO,Sal
FROM Teacher
ORDER BY Sal
```

运行结果：查询结果按照Sal字段的升序顺序排列。

② 双字段记录的排序

```
SELECT Tname,Age,Tsext
          FROM Teacher
          ORDER BY Tsext,Age
```

运行结果：先依Tsex字段的记录排序，当Tsex字段的记录相同时，如均为“男”或“女”，再依Age字段的记录排序。

③ 查询结果按照某特定字段排序

```
SELECT Tname,Sal,Age,Tsex
          FROM Teacher
          ORDER BY 4,3
```

运行结果：查询结果首先依据第4个字段Tsex进行排序，当Tsex字段的记录相同时，再依据第3个字段Age进行降序排序。

④ 双字段的两次排序

```
SELECT Tname,Sal,Age,Tsex
          FROM Teacher
          ORDER BY Tsex,3 Desc
```

运行结果：查询结果首先依据Tsex进行升序排序，当Tsex字段的记录相同时，再依据第3个字段Age进行降序排序。

WHERE

功能说明：定义查询条件。WHERE子句获取FROM子句返回的结果集，并用WHERE定义的搜索条件对结果集筛选。

语　　法：SELECT column　FROM table_name
WHERE column condition value

使用要点：WHERE子句中定义的搜索条件可以是简单的单值比较条件，也可是使用各种运算符、组合子句的条件。

范　　例：

① 筛选结果，并且对结果排序

```
SELECT Tname, Dname, Age, Tsex
        FROM Teacher
        WHERE Dname='计算机'
        ORDER BY Age
```

运行结果：查询结果将院系为“计算机”的教师查询出来，同时对查询结果按年龄进行升序排序。

② 按照逻辑条件来查询数据，并且对结果排序

```
SELECT Tname,Dname,Age,Tsex
        FROM Teacher
        WHERE Age>=40
        ORDER BY Age
```

运行结果：查询结果将年龄大于等于40岁的教师查询出来，同时对查询结果按年龄进行升序排序。

③ 按照“<>”条件来查询数据，并且对结果排序

```
SELECT Tname,Dname,Age,Tsex
         FROM Teacher
         WHERE Dname<>'计算机'
         ORDER BY Dname
```

运行结果：查询结果将所在院系不是“计算机”的教师查询出来，同时对查询结果按院系进行升序排序。

BETWEEN

功能说明：在两个值之间进行比较筛选。

语　　法：test_expressionBETWEEN expression1 AND expression2

使用要点：运算符操作时，如“BETWEEN A1 AND A2”，在进行结果筛选时，满足查询要求的结果是A1到A2之间的信息，并且包含A1和A2。

范　　例：

① 查询数值在某区间的数据

```
SELECT Tname,Dname,Age,Tsex
        FROM Teacher
        WHERE Age BETWEEN 30 AND 50
        ORDER BY Age
```

运行结果：查询结果将年龄在30~50岁（包含30岁和50岁）的教师查询出来，同时对查询结果按年龄进行升序排序。

② 查询介于某种字符字段之间的数据

```
SELECT Tname,Dname,Age,Tsex
        FROM Teacher
        WHERE Dname BETWEEN '计算机' AND'生物'
        ORDER BY Dname
```

运行结果：查询结果将所在院系介于“计算机”和“生物”的教师查询出来，同时对查询结果按院系进行升序排序。

strtoul

功能说明：将字符串转换成无符号长整数。

相关函数：atof，atoi，atol，strtod，strtol

头 文 件：#include <stdlib.h>

函数声明：unsigned long int strtoul(const char *nptr, char **endptr, int base);

函数说明：strtoul()会将参数nptr字符串根据参数base来转换成无符号的长整数。参数base范围从2至36，也可能为0。参数base表示转换时采用的进制方式，如base值为10则采用十进制，若base值为16则采用十六进制等。当base值为0时则是采用十进制作转换，但遇到如'0x'前置字符则会使用十六进制作转换。一开始strtoul()会扫描参数nptr字符串，跳过前面的空白字符，直到遇上数字或正负符号才开始作转换，再遇到非数字或字符串结束时（'\0'）结束转换并将结果返回。如果参数endptr不为NULL，则会将转换时所遇到不符合条件而终止的nptr中的字符指针由endptr返回。

返 回 值：返回转换后的长整数，否则返回ERANGE并将错误代码存入errno中。

错误代码：ERANGE指定的转换字符串超出合法范围。

范　　例：请参考strtol()。

toascii

功能说明：将整数转换成合法的ASCII字符。

相关函数：isascii，toupper，tolower

头 文 件：#include <ctype.h>

函数声明：int toascii(int c);

函数说明：toascii() 会将参数c转换成7位的unsigned char值，第8位则会清除，这个字符即会转成ASCII字符。

返 回 值：将转换成功的ASCII字符值返回。

范　　例：

```
#include   <stdlib.h>
main()
{
  int   a = 217;
  char  b;
  printf("before toascii() : a value = %d(%c)\n", a, a);
  b  =  toascii(a);
  printf("after toascii() : a value = %d(%c)\n", b, b);
}
```

执行结果：

```
before  toascii()  :  a  value  =  217()
after   toascii()  :  a  value  =  89(Y)
```

第4章

常用运算符

运算符是一种符号，用来指定要在一个或多个表达式中进行的操作。SQL中使用如下几种运算符：算术运算符、赋值运算符、位运算符、比较运算符、逻辑运算符、字符串串联运算符和一元运算符。本章介绍各种运算符的使用方法和示例。

!>

功能说明：判断左侧操作数的值不大于右侧操作数的真假。

语　　法：expression1 !> expression 2

使用要点：如果左侧操作数的值不大于右侧操作数，则结果为TRUE；否则结果为FALSE。如果任何一个操作数为NULL，则返回NULL。

范　　例：

```
SELECT Tname,Dname,Age,Tsex
        FROM Teacher
        WHERE Age !>50
        ORDER BY Age
```

运行结果：查询结果将年龄不大于50岁（包含50岁）的教师查询出来，同时对查询结果按年龄进行升序排序。

!<

功能说明：判断左侧操作数的值不小于右侧操作数的真假。

语　　法：expression1!<expression2

使用要点：当比较非空表达式时，如果左侧操作数的值不小于右侧操作数的值，则结果为TRUE，否则结果为FALSE。如果任何一个操作数为NULL，则返回NULL。

范　　例：

```
SELECT Tname,Dname,Age,Tsex
        FROM Teacher
        WHERE Dname !< '计算机'
        ORDER BY Dname
```

运行结果：查询结果将所在院系不小于“计算机”的教师查询出来，同时对查询结果按院系进行升序排序。

!=

功能说明：判断某个表达式是否不等于另一表达式的真假。

语　　法：expression1！=expression2

使用要点：如果任何一个操作数为NULL，或2个都为NULL，则返回NULL。其功能与<>（不等于）比较运算符相同。

范　　例：

```
SELECT Tname,Dname,Age,Tsex
        FROM Teacher
        WHERE Tsex != '女'
        ORDER BY Age
```

运行结果：查询结果将性别不为“女”的教师查询出来，同时对查询结果按年龄进行升序排序。

%

功能说明：匹配字符串。进行匹配的字符串可以包含任意个字符。

语　　法：LIKE '%'

使用要点：该通配符既可以用作前缀也可以用作后缀。

范　　例：

```
SELECT Tname,Dname,Age,Tsex
        FROM Teacher
        WHERE Tname like '李%'
        ORDER BY Tname
```

运行结果：查询结果将姓“李”的教师查询出来，同时对查询结果按姓名进行升序排序。

%

功能说明：返回两数相除后的余数。

语　　法：dividend % divisor

参数说明：

关 键 字	含　义
dividend	要执行除法运算的数值表达式。dividend必须为整数和货币数据类型类别中任意数据类型的有效表达式，或者为numeric数据类型
divisor	要除被除数的数值表达式。divisor必须为整数和货币数据类型类别中任意数据类型的有效表达式，或者为numeric数据类型

使用要点：取模算术运算符可以和列名、数值常量或任何具有整数和货币数据类型类别或numeric数据类型的有效表达式组合一起用于SELECT语句的选择列表中。

范　　例：

```
SELECT TOP(100)ProductID, UnitPrice, OrderQty,
CAST((UnitPrice) AS int) % OrderQty AS Modulo
FROM Sales.SalesOrderDetail;
```

运行结果：查询结果返回产品ID号、产品单价、除以每种产品的单价后得到的模（余数）、转换为整数值。

&

功能说明：对两个整数值执行位与逻辑运算。

语　　法：expression1 & expression2

使用要点：&位运算符将在两个表达式之间执行位与逻辑运算，从两个表达式取对应的位。当且仅当输入表达式中两个位的值都为1时，结果中的位才被设置为1；否则，结果中的位被设置为0。

范　　例：

```
SELECT a_int_value & b_int_value
FROM bitwise;
```

运行结果：170（a_int_value或A）的二进制表示形式是0000 0000 1010 1010。75（b_int_value或B）的二进制表示形式是0000 0000 0100 1011。对上述两个值执行“位与”运算将产生二进制0000 0000 0000 1010，即十进制数10。

第10章

游标函数

在数据库中，游标是一个十分重要的概念。游标提供了一种灵活的手段，可以对表中检索出的数据进行操作。就本质而言，游标实际上是一种能从包括多条数据记录的结果集中每次提取一条记录的机制。本章将讲解游标函数的具体应用。

@@CURSOR_ROWS

功能说明：返回当前限定行的数目。

语　　法：@@CURSOR_ROWS

使用要点：如果上一个游标是异步打开的，则@@CURSOR_ROWS返回的数字是负数。如果sp_configure cursor threshold的值大于0，且游标结果集中的行数大于游标阈值，则异步打开键集驱动程序或静态游标。

返回值	说　明
-m	游标被异步填充。返回值(-m)是键集中当前的行数
-1	游标为动态游标。因为动态游标可反映所有更改，所以游标符合条件的行数不断变化。因此，永远不能确定已检索到所有符合条件的行
0	没有已打开的游标，对于上一个打开的游标没有符合条件的行，或上一个打开的游标已被关闭或被释放
n	游标已完全填充。返回值(n)是游标中的总行数

范　　例：

```
USE AdventureWorks;
SELECT @@CURSOR_ROWS;
DECLARE Name_Cursor CURSOR FOR
SELECT LastName,@@CURSOR_ROWS FROM Person.Contact;
OPEN Name_Cursor;
FETCH NEXT FROM Name_Cursor;
SELECT @@CURSOR_ROWS;
CLOSE Name_Cursor;
DEALLOCATE Name_Cursor;
```

运行结果：0

```
LastName
Achong
-1
```

@@FETCH_STATUS

功能说明：返回上一条游标FETCH语句的状态。

语　　法：@@FETCH_STATUS

使用要点：由于@@FETCH_STATUS对于在一个连接上的所有游标都是全局性的，所以要谨慎使用@@FETCH_STATUS。在执行一条FETCH语句后，必须在对另一游标执行另一FETCH语句前测试@@FETCH_STATUS。在此连接上出现任何提取操作之前，@@FETCH_STATUS的值没有定义。

返回值	说　明
0	FETCH语句成功
−1	FETCH语句失败或行不在结果集中
−2	提取的行不存在

范　　例：

```
DECLARE Employee_Cursor CURSOR FOR
SELECT EmployeeID, Title FROM AdventureWorks. HumanResources.
Employee;
OPEN Employee_Cursor;
FETCH NEXT FROM Employee_Cursor;
WHILE @@FETCH_STATUS = 0
BEGIN
FETCH NEXT FROM Employee_Cursor;
END;
CLOSE Employee_Cursor;
DEALLOCATE Employee_Cursor;
```

运行结果：用@@FETCH_STATUS控制WHILE循环中的游标活动。