数据库中聚合函数count() count(字段) sum(字段) avg(字段) min(字段) max(字段)

## 8.2.1 求和函数——SUM()

求和函数SUM( )用于对数据求和，返回选取结果集中所有值的总和。语法如下。

SELECT SUM(column\_name)

FROM table\_name

说明：SUM()函数只能作用于数值型数据，即列column\_name中的数据必须是数值型的。

**实例1 SUM函数的使用**

从TEACHER表中查询所有男教师的工资总数。TEACHER表的结构和数据可参见5.2.1节的表5-1，下同。实例代码：

**SELECT SUM(SAL) AS BOYSAL**

**FROM TEACHER**

**WHERE TSEX='男'**

**实例2 SUM函数对NULL值的处理**

从TEACHER表中查询年龄大于40岁的教师的工资总数。实例代码：

**SELECT SUM(SAL) AS OLDSAL**

**FROM TEACHER**

**WHERE AGE>=40**

当对某列数据进行求和时，如果该列存在NULL值，则SUM函数会忽略该值。

## 8.2.2 计数函数——COUNT()

COUNT()函数用来计算表中记录的个数或者列中值的个数，计算内容由SELECT语句指定。使用COUNT函数时，必须指定一个列的名称或者使用星号，星号表示计算一个表中的所有记录。两种使用形式如下。

COUNT(\*)，计算表中行的总数，即使表中行的数据为NULL，也被计入在内。

COUNT(column)，计算column列包含的行的数目，如果该列中某行数据为NULL，则该行不计入统计总数。

1．使用COUNT(\*)函数对表中的行数计数

COUNT(\*)函数将返回满足SELECT语句的WHERE子句中的搜索条件的函数。

**实例3 COUNT(\*)函数的使用**

查询TEACHER表中的所有记录的行数。实例代码：

**SELECT COUNT(\*) AS TOTALITEM**

**FROM TEACHER**

在该例中，SELECT语句中没有WHERE子句，那么认为表中的所有行都满足SELECT语句，所以SELECT语句将返回表中所有行的计数，结果与5.2.1节的表5-1列出的TEACHER表的数据相吻合。

如果DBMS在其系统表中存储了表的行数，COUNT(\*)将很快地返回表的行数，因为这时，DBMS不必从头到尾读取表，并对物理表中的行计数，而直接从系统表中提取行的计数。而如果DBMS没有在系统表存储表的行数，将具有NOT NULL约束的列作为参数，使用COUNT()函数，则可能更快地对表行计数。

注意

COUNT(\*)函数将准确地返回表中的总行数，而仅当COUNT()函数的参数列中没有NULL值时，才返回表中正确的行计数，所以仅当受NOT NULL限制的列作为参数时，才可使用COUNT( )函数代替COUNT(\*)函数。

**2．使用COUNT( )函数对一列中的数据计数**

COUNT( )函数可用于对一列中的数据值计数。与忽略了所有列的COUNT(\*)函数不同，COUNT( )函数逐一检查一列（或多列）中的值，并对那些值不是NULL的行计数。

**实例4 查询多列中所有记录的行数**

查询TEACHER表中的TNO列、TNAME列以及SAL列中包含的所有数据行数。实例代码：

**SELECT COUNT(TNO) AS TOTAL\_TNO, COUNT(TNAME) AS TOTAL\_TNAME,**

**COUNT(SAL) AS TOTAL\_SAL**

**FROM TEACHER**

可见，TNO列与TNAME列由于其中不含有NULL值，所以其计数与使用COUNT(\*)函数对TEACHER表中的记录计数结果相一致，而SAL列由于其中有两行数据为NULL，所以这两列没有被计入在内，计数结果也就是8。

**3．使用COUNT( )函数对多列中的数据计数**

COUNT( )函数不仅可用于对一列中的数据值计数，也可以对多列中的数据值计数。如果对多列计数，则需要将要计数的多列通过连接符连接后，作为COUNT( )函数的参数。下面将结合具体的多列计数的实例，说明其使用过程。

说明

关于如何使用连接符连接多列可参见本书的7.2节。

**实例5 使用COUNT( )函数对多列中的数据计数**

统计TEACHER表中的TNO列、TNAME列和SAL列中分别包含的数据行数，以及TNO列和TNAME列、TNAME列和SAL列一起包含的数据行数。实例代码：

**SELECT COUNT(TNO) AS TOTAL\_TNO, COUNT(TNAME) AS TOTAL\_TNAME,**

**COUNT(SAL) AS TOTAL\_SAL,**

**COUNT(CAST(TNO AS VARCHAR(5)) + TNAME) AS T\_NONAME,**

**COUNT(TNAME + CAST(SAL AS VARCHAR(5))) AS T\_NAMESAL**

**FROM TEACHER**

在进行两列的连接时，由于它们的数据类型不一致，因此要使用CAST表达式将它们转换成相同的数据类型。

在7.2.1节已经讲过，如果在被连接的列中的任何一列有NULL值时，那么连接的结果为NULL，则该列不会被COUNT( )函数计数。

注意

COUNT( )函数只对那些传递到函数中的参数不是NULL的行计数。

**4．使用COUNT函数对满足某种条件的记录计数**

也可以在SELECT语句中添加一些子句约束来指定返回记录的个数。

实例6 使用COUNT函数对满足某种条件的记录计数

查询TEACHER表中女教师记录的数目。实例代码：

**SELECT COUNT(\*) AS TOTALWOMEN**

**FROM TEACHER**

**WHERE TSEX='女'**

这时结果为6而不是前面的所有记录10。之所以可以通过WHERE子句定义COUNT()函数的计数条件，这与SELECT语句各个子句的执行顺序是分不开的。前面已经讲过，DBMS首先执行FROM子句，而后是WHERE子句，最后是SELECT子句。所以COUNT()函数只能用于满足WHERE子句定义的查询条件的记录。没有包括在WHERE子句的查询结果中的记录，都不符合COUNT()函数。

## 8.2.3 最大/最小值函数—MAX()/MIN()

当需要了解一列中的最大值时，可以使用MAX()函数；同样，当需要了解一列中的最小值时，可以使用MIN()函数。语法如下。

**SELECT MAX (column\_name) / MIN (column\_name)**

**FROM table\_name**

说明：列column\_name中的数据可以是数值、字符串或是日期时间数据类型。MAX()/MIN()函数将返回与被传递的列同一数据类型的单一值。

**实例7 MAX()函数的使用**

查询TEACHER表中教师的最大年龄。实例代码：

**SELECT MAX (AGE) AS MAXAGE**

**FROM TEACHER**

然而，在实际应用中得到这个结果并不是特别有用，因为经常想要获得的信息是具有最大年龄的教师的教工号、姓名、性别等信息。

然而SQL不支持如下的SELECT语句。

**SELECT TNAME, DNAME, TSEX, MAX (AGE)**

**FROM TEACHER**

因为聚合函数处理的是数据组，在本例中，MAX函数将整个TEACHER表看成一组，而TNAME、DNAME和TSEX的数据都没有进行任何分组，因此SELECT语句没有逻辑意义。同样的道理，下面的代码也是无效的。

**SELECT TNAME, DNAME, TSEX,SAL ,AGE**

**FROM TEACHER**

**WHERE AGE＝MAX (AGE)**

解决这个问题的方法，就是在WHERE子句中使用子查询来返回最大值，然后再基于这个返回的最大值，查询相关信息。

**实例8 在WHERE子句中使用子查询返回最大值**

查询TEACHER表中年纪最大的教师的教工号、姓名、性别等信息。

实例代码：

**SELECT TNAME, DNAME, TSEX, SAL, AGE**

**FROM TEACHER**

**WHERE AGE＝(SELECT MAX (AGE) FROM TEACHER)**

MAX()和MIN()函数不仅可以作用于数值型数据，也可以作用于字符串或是日期时间数据类型的数据。

**实例9 MAX()函数用于字符型数据**

如下面代码：

**SELECT MAX (TNAME) AS MAXNAME**

**FROM TEACHER**

可见，对于字符串也可以求其最大值。

说明

对字符型数据的最大值，是按照首字母由A～Z的顺序排列，越往后，其值越大。当然，对于汉字则是按照其全拼拼音排列的，若首字符相同，则比较下一个字符，以此类推。

当然，对与日期时间类型的数据也可以求其最大/最小值，其大小排列就是日期时间的早晚，越早认为其值越小，如下面的实例。

实例10 MAX()、MIN()函数用于时间型数据

从COURSE表中查询最早和最晚考试课程的考试时间。其中COURSE表的结构和数据可参见本书6.1节的表6-1。实例代码：

**SELECT MIN (CTEST) AS EARLY\_DATE,**

**MAX (CTEST) AS LATE\_DATE**

**FROM COURSE**

可见，返回结果的数据类型与该列定义的数据类型相同。

注意

确定列中的最大值（最小值）时，MAX( )（MIN( )）函数忽略NULL值。但是，如果在该列中，所有行的值都是NULL，则MAX( )/MIN( )函数将返回NULL值。

## 8.2.4 均值函数——AVG()

函数AVG()用于计算一列中数据值的平均值。语法如下。

**SELECT AVG (column\_name)**

**FROM table\_name**

说明：AVG()函数的执行过程实际上是将一列中的值加起来，再将其和除以非NULL值的数目。所以，与SUM( )函数一样，AVG()函数只能作用于数值型数据，即列column\_name中的数据必须是数值型的。

**实例11 AVG()函数的应用**

从TEACHER表中查询所有教师的平均年龄。实例代码：

**SELECT AVG (AGE) AS AVG\_AGE**

**FROM TEACHER**

在计算平均值时，AVG()函数将忽略NULL值。因此，如果要计算平均值的列中有NULL值，计算均值时，要特别注意。

**实例12 AVG()函数对NULL值的处理**

从TEACHER表中查询所有教师的平均工资。实例代码：

**SELECT AVG (SAL) AS AVG\_AGE1，SUM(SAL)/COUNT(\*) AS AVG\_AGE2,**

**SUM(SAL)/COUNT(SAL) AS AVG\_AGE3**

**FROM TEACHER**

可以发现得到了不同的结果。实际上，“AVG(SAL)”与“SUM(SAL)/COUNT(SAL)”语句是等价的。因为AVG(SAL)语句的执行过程实际上是将SAL列中的值加起来，再将其和（也就等价于SUM(SAL)）除以非NULL值的数目（也就等价于COUNT(SAL)）。而语句“SUM(SAL)/COUNT(\*)”则不然，因为COUNT(\*)返回的是表中所有记录的个数，而不管SAL列中的数值是否为NULL。

注意

AVG()函数在计算一列的平均值时，忽略NULL值。但是，如果在该列中，所有行的值都是NULL，则AVG()函数将返回NULL值。

如果不想对列中的所有值求平均，则可在WHERE子句中使用搜索条件来限制用于计算均值的行。

**实例13 在WHERE子句中使用搜索条件来限制用于计算均值的行**

从TEACHER表中查询所有计算机系教师的平均年龄。实例代码：

**SELECT AVG (AGE) AS AVGCOMPUTER\_AGE**

**FROM TEACHER**

**WHERE DNAME = '计算机'**

当执行SELECT语句时，DBMS将表中的每行对WHERE子句中的搜索条件“DNAME = '计算机'”求值。只有那些搜索条件为True时，行中的AGE值才传到均值函数AVG (AGE)中。

当然，除了显示表中某列的平均值，还可用AVG()函数作为WHERE子句的一部分。与前面介绍的MAX()函数一样，不能直接用于WHERE子句，必须以子查询的形式。

**实例14 AVG()函数作为WHERE子句中搜索条件的一部分**

从TEACHER表中查询所有年龄高于平均年龄的教师的信息。实例代码：

**SELECT \***

**FROM TEACHER**

**WHERE AGE >= (SELECT AVG (AGE) FROM TEACHER)**

**ORDER BY AGE**

## 8.2.5 聚合分析的重值处理

前面介绍的5种聚合函数，可以作用于所选列中的所有数据（不管列中的数据是否有重置），也可以只对列中的非重值进行处理，即把重复的值只取一次进行聚合分析。当然，对于MAX()/MIN()函数来讲，重值处理意义不大。

可以使用ALL关键字指明对所选列中的所有数据进行处理，使用DISTINCT关键字指明对所选列中的非重值数据进行处理。以AVG()函数为例，语法如下。

**SELECT AVG ([ALL/DISTINCT] column\_name)**

**FROM table\_name**

说明：[ALL/DISTINCT]在缺省状态下，默认是ALL关键字，即不管是否有重值，处理所有数据。其他聚合函数的用法与此相同。

注意

Microsoft Access数据库不支持在聚合函数中使用DISTINCT关键字。

**实例15 聚合分析的重值处理**

从TEACHER表中查询工资SAL列中存在的所有记录数。实例代码：

**SELECT COUNT(ALL SAL) AS ALLSAL\_COUNT**

**FROM TEACHER**

当然，在代码中去除ALL关键字，也可以得到相同的结果。而如果从TEACHER表中，查询工资SAL列中存在的不同记录的数目，可采用如下代码。

**SELECT COUNT(DISTINCT SAL) AS DISTINCTSAL\_COUNT**

**FROM TEACHER**

对比两个结果，使用DISTINCT关键字后，工资SAL列中的重值并没有列入统计的范围之内。另外还要强调一点，在所有5种聚合函数中，除了COUNT(\*)函数外，其他的函数在计算过程中都忽略NULL值，即把NULL值的行排除在外，不进行分析。

## 8.2.6 聚合函数的组合使用

前面介绍的实例中，聚合函数都是单独使用的。聚合函数也可以组合使用，即在一条SELECT语句中，可以使用多个聚合函数。

**实例16 使用多个聚合函数**

如下面的代码：

**SELECT COUNT(\*) AS num\_items,**

**MAX(SAL) AS max\_sal,**

**Min(AGE) AS min\_age,**

**SUM(SAL)/COUNT(SAL) AS avg\_sal,**

**AVG(DISTINCT SAL) AS disavg\_sal**

**FROM TEACHER**

该例在一条SELECT语句中，几乎用到了所有的聚合函数。其中num\_items为TEACHER表所有记录的条目，max\_sal为TEACHER表中记录的最高工资，min\_age为TEACHER表中记录的最小年龄，avg\_sal为所有TEACHER表中的工资记录的平均值，disavg\_sal为TEACHER表中所有不同的工资记录的平均值。