**MPC**

**主讲老师：张胜利**

**（一）日期函数：**

1. **GETDATE()**

**获取服务器当前日期和时间**

1. **DATEADD()**

**DATEADD (datepart , number, date )，计算增加以后的日期。**

**参数date为待计算的日期；参数number为增量；参数datepart为计量单位(可取值year、quarter季度、month、day、week、hour、minute、second等)**

**Eg:**

**SELECT DATEADD(MONTH ,-8,GETDATE());**

1. **FORMAT() 函数**

**FORMAT (value, format)函数用于对字段的显示进行格式化。**

**Eg:**

**Select FORMAT(GETDATE(),'yyyyMM01');**

1. **DATEDIFF()**

**DATEDIFF(datepart,startdate,enddate)：计算两个日期之间的差额。（enddate-startdate包括enddate这一天。）datepart 为计量单位，可取值参考DateAdd。**

**Eg:统计不同工龄的员工的个数（时间差，包括当天）：**

**select DateDiff(year,FInDate,getdate()),count(\*)**

**from T\_Employee group by DateDiff(year, FInDate,getdate())**

1. **DATEPART()**

**DATEPART (datepart,date)：返回一个日期的特定部分。**

**datepart 为计量单位，可取值参考DateAdd。类似于YEAR/Month/Day/Hour/Minute/Second(date);**

**当datepart为DW时 返回星期几（int类型，周日为第一天）**

1. **DateName(DW,date)**

**返回字符串星期几**

1. **EOMONTH（date,months）获取特定月份最后一天的日期**

**Eg:**

**如何获取上上个月最后一天日期？**

**Select EOMONTH（getDate(),-2）;**

**如何获取上个月第一天日期？**

**Select dateAdd(day,1,EOMONTH(getDate(),-2));**

**（二）转换函数：**

* **CAST( expression AS data\_type)**

**expression是要转换的对象，data\_type是要转换的数据类型**

**Eg:**

**SELECT CAST('123' AS INT)+1/CAST('2009-09-09' AS DATE);**

* **CONVERT( data\_type, expression,style)**

**expression是要转换的对象，data\_type是要转换的数据类型, style 规定日期/时间的输出格式(可选)。**

**Try\_convert ( data\_type,expression)用法跟convert是一样的，它的特点是不会抛出异常，出错返回null.**

**Eg:**

**SELECT CONVERT(DATETIME,'2012-12-12'),**

**CONVERT(nvarchar(100),GETDATE(),21)**

**DATEADD(DAY, 1,CONVERT(DATE,GETDATE())**

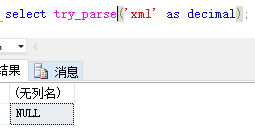
**CONVERT(VARCHAR, GETDATE(), 112)**

|  |  |
| --- | --- |
| **Style ID** | **Style 格式** |
| **100 或者 0** | **mon dd yyyy hh:miAM （或者 PM）** |
| **101** | **mm/dd/yy** |
| **102** | **yy.mm.dd** |
| **103** | **dd/mm/yy** |
| **104** | **dd.mm.yy** |
| **105** | **dd-mm-yy** |
| **106** | **dd mon yy** |
| **107** | **Mon dd, yy** |
| **108** | **hh:mm:ss** |
| **109或9** | **mon dd yyyy hh:mi:ss:mmmAM（或者 PM）** |
| **110** | **mm-dd-yy** |
| **111** | **yy/mm/dd** |
| **112** | **yymmdd** |
| **113或13** | **dd mon yyyy hh:mm:ss:mmm(24h)** |
| **114** | **hh:mi:ss:mmm(24h)** |
| **120或者20** | **yyyy-mm-dd hh:mi:ss(24h)** |
| **121或者21** | **yyyy-mm-dd hh:mi:ss.mmm(24h)** |
| **126** | **yyyy-mm-ddThh:mm:ss.mmm（没有空格）** |
| **130** | **dd mon yyyy hh:mi:ss:mmmAM** |
| **131** | **dd/mm/yy hh:mi:ss:mmmAM** |

* **parse( string\_value AS data\_type)和try\_parse**

**parse 为 Sql Server 的新特性。 expression 必须要为被转换类型的字符串形式.跟cast用法是一样的。**

**Try\_parse ( string\_value AS data\_type)用法跟parse是一样的，它的特点是不会抛出异常，出错返回null**

****

* **iif(布尔表达式,value1,value2)**

**也就是说如果表达式成立，那么就返回value1，如果不成立，就返回value2.**

**Eg:**

**Select iif(try\_parse(@var as decimal(36,9))is null,'ture','false');------>true**

**（三）分组**

1. **按照某列分组**

**如果有WHERE的话，GROUP和Order by（用于按指定的列对结果集进行排序）**

**都必须放在WHERE之后**

**SELECT 分组列名 AS 列别名,聚合函数1,聚合函数2 FROM <TBname>**

**GROUP BY 分组列名**

1. **按照多列分组**

**SELECT 列名1,列名2,聚合函数 FROM <TBname> GROUP BY 列名1，列名2**

**--直接用逗号隔开多个列名即可**

**注：**

**Select子句后面只能是聚合函数 或GROUP BY 的列名**

1. **group by roolup(A,B,C)按照一定的规则进行分组**
2. **group by grouping sets()自定义分组**

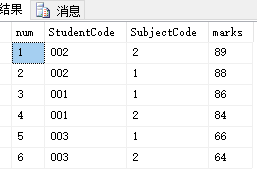
**（四）排序函数**

* **row\_number()**

**是没有重复值的排序(即使两天记录相等也是不重复的)**

**注意:使用row\_number函数时必须要用over子句选择对某一列进行排序才能生成序号。其基本原理是先使用over子句中的排序语句对记录进行排序，然后按照这个顺序生成序号。**

**Eg：**

**select ROW\_NUMBER() over(order by marks desc) num,\* from StudentMarks;**

* **dense\_rank()**

**dense\_rank函数的功能与rank函数类似。在各个分组内，rank()是跳跃排序，有两个第一名时接下来就是第四名，dense\_rank()是连续排序，有两个第一名时仍然跟着第二名。**

**Eg:**

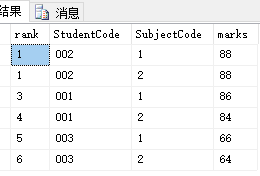
**select DENSE\_RANK() over(order by marks desc) dense\_rank,\* from StudentMarks;**

* **rank()**

**是跳跃排序,rank函数考虑到了over子句中排序字段值相同的情况，如果使用rank函数来生成序号，over子句中排序字段值相同的序号是一样的，后面字段值不相同的序号将跳过相同的排名号排下一个，也就是相关行之前的排名数加一，可以理解为根据当前的记录数生成序号，后面的记录依此类推。**

**Eg:**

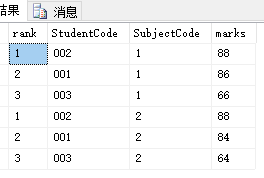
**select RANK() over(order by marks desc) rank,\* from StudentMarks ;**



**如果要分组排序呢请按学科进行排序**

**partition by 与group by不同之处在于前者返回的是分组里的每一条数据，并且可以对分组数据进行排序操作。后者只能返回聚合之后的组的数据统计值的记录。**

**select RANK() over(partition by subjectcode order by marks desc) rank,\* from StudentMarks ;**



* **Ntile() over(order by columnname)**

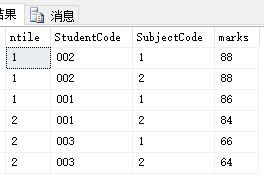
**ntile后面的数字，是要把查询得到的结果平均分为几组**

**ntile函数可以对序号进行分组处理，将有序分区中的行分发到指定数目的组中。 各个组有编号，编号从一开始。 对于每一个行，ntile 将返回此行所属的组的编号。这就相当于将查询出来的记录集放到指定长度的数组中，每一个数组元素存放一定数量的记录。ntile函数为每条记录生成的序号就是这条记录所有的数组元素的索引（从1开始）。也可以将每一个分配记录的数组元素称为“桶”。ntile函数有一个参数，用来指定桶数。**

**下面的SQL语句使用ntile函数对StudentMarks表进行了装桶处理：**

**Eg:**

**select ntile(2) over(order by marks desc) ntile,\* from StudentMarks**



**（五）其他函数**

* **CASE()**

1. **CASE 列名**

**WHEN value1 THEN ReturnValue1 WHEN value2 THEN ReturnValue2**

**ELSE DefaultReturnValue END**

**Eg:**

**SELECT FName,(CASE FLevel WHEN 1 THEN 'VIP客户' WHEN 2 THEN '高级客户'**

**WHEN 3 THEN '普通客户' ELSE '客户类型错误' END) as FLevelName FROM T\_Customer**

1. **CASE**

**WHEN 表达式1 THEN ReturnValue1 WHEN 表达式2 THEN ReturnValue2 ELSE DefaultReturnValue END**

**注:ELSE不一定要有，但必须有END**

**Eg:**

**SELECT F\_NAME,(CASE WHEN F\_SALARY<2000 THEN '穷人'**

**WHEN 2000<=F\_SALARY AND F\_SALARY<3000 THEN '中产'**

**WHEN 3000<=F\_SALARY AND F\_SALARY<5000 THEN '富人'**

**WHEN 5000<=F\_SALARY THEN '贵族' ELSE '未知' END )AS 收入水平**

* **LEAD和LAG**

**LEAD访问相同结果集的后续行中的数据，而不使用 SQL Server 2012 中的自联接。 LEAD 以当前行之后的给定物理偏移量来提供对行的访问。 在 SELECT 语句中使用此分析函数可将当前行中的值与后续行中的值进行比较。**

**LAG访问相同结果集的先前行中的数据，而不使用 SQL Server 2012 中的自联接。 LAG 以当前行之前的给定物理偏移量来提供对行的访问。 在 SELECT 语句中使用此分析函数可将当前行中的值与先前行中的值进行比较。**

**语法：LEAD/LAG ( scalar\_expression [ ,offset ] , [ default ] ) OVER ( [ partition\_by\_clause ] order\_by\_clause )scalar\_expression，要返回的值基于指定的偏移量。 这是一个返回单个（标量）值的任何类型的表达式。scalar\_expression 不能为分析函数**

**offset默认值为1，指的是前（后）几行的数据值 ， offset 可以是列、子查询或其他求值为正整数的表达式，或者可隐式转换为bigint。offset 不能是负数值或分析函数。Default是指结果为空的默认值， offset 可以是列、子查询或其他求值为正整数的表达式，或者可隐式转换为bigint。offset不能是负数值或分析函数。**

* **write的用法**

**write更新的字段必须是varchar(max) 或是nvarchar(max)的类型。write(char,index,index2)**

**char是要加的字符，index加的位置，index2是占的空间。**

* **COUNT() :求某一列的行数**
* **ISNULL(expression,value)**

**expression是列名，如果不为空则返回具体的列值，如果未空，则用value填充。**

**Eg:**

**SELECT ISNULL(FName,'佚名') as 姓名 FROM T\_Employe**

**——》如果名字为空，则用佚名来代替。**

* **用 ANY、SOME 或 ALL 修改的比较运算符**

**SOME 是与 ANY 等效的。**

**以 > 比较运算符为例：**

**>ALL 表示大于每一个值。换句话说，它表示大于最大值。例如，>ALL (1, 2, 3) 表示大于 3。**

**>ANY 表示至少大于一个值，即大于最小值。因此 >ANY (1, 2, 3) 表示大于 1。**

**=ANY 运算符与 IN 等效。**

**< >ANY 运算符则不同于 NOT IN：< >ANY 表示不等于 a，或者不等于 b，或者不等于 c。NOT IN 表示不等于 a、不等于 b 并且不等于 c。**

**<>ALL 与 NOT IN 表示的意思相同。**

**Eg:**

**select \* from student where student\_no <> all(select StudentCode from StudentMarks);**