**1. 背景**

最近有用户在做一些项目，使用到SQL SERVER 2012的一些新特性，比如SQL SERVER 提供的8个非常有用的分析函数，一开始我看了相关的文档，感觉内容很多，理解不清楚，不透彻。而我现在想来，其实不需要那么清楚，我觉得值要理解他的基本用法就足以应对工作，下面根据我的理解，以最简单的方式解析这些分析函数。

**1. 分析函数CUME\_DIST**

微软的定义：   
计算某个值在 SQL Server 2012 中的一组值内的累积分布。CUME\_DIST 计算某指定值在一组值中的相对位置。 对于行 r，假定采用升序，r 的 CUME\_DIST 是值低于或等于 r 的值的行数除以在分区或查询结果集中求出的行数。   
其实，我看了也不是很懂，我们看一个实例，就很清楚了。   
首先，我们构造一组数据：

DECLARE

@analytic TABLE(

name varchar(35) ,

dept varchar(35),

salary money

)

INSERT INTO @analytic

VALUES

--bd

('andy01','bd',15000),

('andy02','bd',12000),

('andy03','bd',12000),

('andy04','bd',10000),

('andy05','bd',8000),

--ca

('andy06','ca',20000),

('andy07','ca',18000),

('andy08','ca',18000),

('andy09','ca',15000),

('andy10','ca',12000),

('andy11','ca',12000),

('andy12','ca',10000),

('andy13','ca',8000),

('andy14','ca',8000),

('andy15','ca',8000)

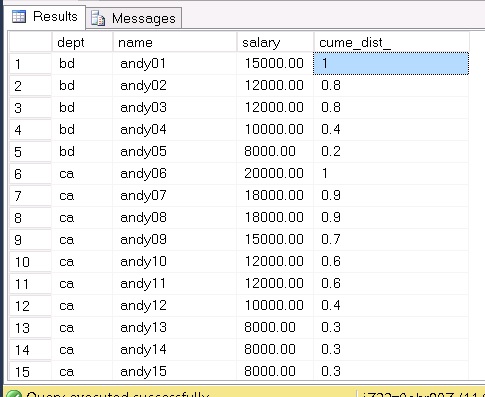
SELECT

dept,name ,salary,

CUME\_DIST() OVER(PARTITION BY dept ORDER BY salary) AS cume\_dist\_

FROM @analytic

ORDER BY dept,salary DESC

然后我们再看看结果：   


这是个什么意思呢？ 按照dept分组，根据salary逻辑排序，针对每一个分组里的每一个值，在该分组下等于或者小于自己的salary的分布百分比。举个例子，bd部门的andy02，salary为12000，那么等于或者小于这个12000的有4条，总共5条记录，因此那么CUME\_DIST()=4/5 = 0.8。 同理，其他也是如此计算，这下就明白了吧。

**2. 分析函数LAST\_VALUE**

微软的定义：   
返回 SQL Server 2012 中有序值集中的最后一个值。   
好像我还是不太懂。那么还是看看一个示例吧：

DECLARE

@analytic TABLE(

name varchar(35) ,

dept varchar(35),

salary money ,

hiredate date

)

INSERT INTO @analytic

VALUES

--bd

('andy01','bd',15000,'2002-01-09'),

('andy02','bd',12000,'2003-01-09'),

('andy03','bd',12000,'2003-02-09'),

('andy04','bd',10000,'2005-05-09'),

('andy05','bd',8000,'2003-06-09'),

--ca

('andy06','ca',20000,'2003-01-09'),

('andy07','ca',18000,'2005-02-09'),

('andy08','ca',18000,'2005-03-09'),

('andy09','ca',15000,'2004-01-09'),

('andy10','ca',12000,'2003-06-09'),

('andy11','ca',12000,'2002-09-09'),

('andy12','ca',10000,'2003-07-09'),

('andy13','ca',8000,'2003-08-09'),

('andy14','ca',8000,'2003-11-09'),

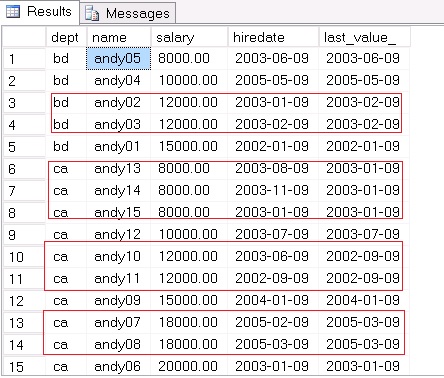
('andy15','ca',8000,'2003-01-09')

SELECT

dept,name ,salary,hiredate,

LAST\_VALUE(hiredate) OVER(PARTITION BY dept ORDER BY salary) AS last\_value\_

FROM @analytic

我们再看看这个运行结果：   


这个意思是按照OVER子句中ORDER BY 根据salary排序，取salary最后行的hiredate值作为最后的LAST VALUE，重点在于salary有相同时需要取salary排序后的最后一条作为其他的LAST VALUE。

**3. 分析函数FIRST\_VALUE**

微软的定义：   
返回 SQL Server 2012 中有序值集中的第一个值。 从这个地方看起来，似乎跟LAST\_VALUE是相反的一个意思，实际上是不是这样，我们来看看实例：

DECLARE

@analytic TABLE(

name varchar(35) ,

dept varchar(35),

salary money ,

hiredate date

)

INSERT INTO @analytic

VALUES

--bd

('andy01','bd',15000,'2002-01-09'),

('andy02','bd',12000,'2003-01-09'),

('andy03','bd',12000,'2003-02-09'),

('andy04','bd',10000,'2005-05-09'),

('andy05','bd',8000,'2003-06-09'),

--ca

('andy06','ca',20000,'2003-01-09'),

('andy07','ca',18000,'2005-02-09'),

('andy08','ca',18000,'2005-03-09'),

('andy09','ca',15000,'2004-01-09'),

('andy10','ca',12000,'2003-06-09'),

('andy11','ca',12000,'2002-09-09'),

('andy12','ca',10000,'2003-07-09'),

('andy13','ca',8000,'2003-08-09'),

('andy14','ca',8000,'2003-11-09'),

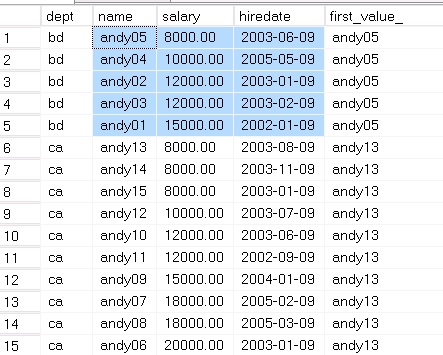
('andy15','ca',8000,'2003-01-09')

SELECT

dept,name ,salary,hiredate,

FIRST\_VALUE(name) OVER(PARTITION BY dept ORDER BY salary) AS first\_value\_

FROM @analytic



显然，这个与LAST\_VALUE不是同一个意思，OVER子句根据ORDER BY 来排序，按dept分组来确定这个分组的第一个值，而不是根据salary的值来确定的，所以与LAST\_VALUE是不一样的，将FIRST\_VALUE(name)修改为FIRST\_VALUE(hiredate)后，对比看得更清楚。这个很有蒙蔽性。

**4. 分析函数LEAD**

微软的定义：   
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。   
感觉这个好理解多了。不过我们还是看看一个示例来说明：

DECLARE

@analytic TABLE(

name varchar(35) ,

dept varchar(35),

salary money ,

hiredate date

)

INSERT INTO @analytic

VALUES

--bd

('andy01','bd',15000,'2002-01-09'),

('andy02','bd',12000,'2003-01-09'),

('andy03','bd',12000,'2003-02-09'),

('andy04','bd',10000,'2005-05-09'),

('andy05','bd',8000,'2003-06-09'),

--ca

('andy06','ca',20000,'2003-01-09'),

('andy07','ca',18000,'2005-02-09'),

('andy08','ca',18000,'2005-03-09'),

('andy09','ca',15000,'2004-01-09'),

('andy10','ca',12000,'2003-06-09'),

('andy11','ca',12000,'2002-09-09'),

('andy12','ca',10000,'2003-07-09'),

('andy13','ca',8000,'2003-08-09'),

('andy14','ca',8000,'2003-11-09'),

('andy15','ca',8000,'2003-01-09')

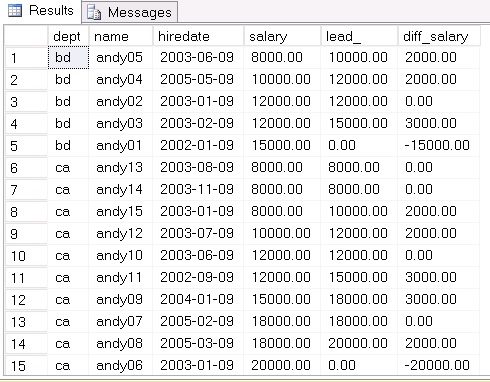
SELECT

dept,name,hiredate,salary,

LEAD(salary,1,0) OVER(PARTITION BY dept ORDER BY salary) AS lead\_,

(LEAD(salary,1,0) OVER(PARTITION BY dept ORDER BY salary)-salary) AS diff\_salary

FROM @analytic

我们再看看结果：   


这个还是比较容易理解的。按照dept分区，根据salary排序，比较当前记录和后一条记录（偏移量为1）的salary值的差值，这个非常的使用

# 5. 分析函数 LAG

微软定义：   
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。  
还是比较拗口，最近有用户在做一些项目，我们还是看看示例比较好理解，看看是不是这样。

DECLARE

@analytic TABLE(

name varchar(35) ,

dept varchar(35),

salary money ,

hiredate date

)

INSERT INTO @analytic

VALUES

--bd

('andy01','bd',15000,'2002-01-09'),

('andy02','bd',12000,'2003-01-09'),

('andy03','bd',12000,'2003-02-09'),

('andy04','bd',10000,'2005-05-09'),

('andy05','bd',8000,'2003-06-09'),

--ca

('andy06','ca',20000,'2003-01-09'),

('andy07','ca',18000,'2005-02-09'),

('andy08','ca',18000,'2005-03-09'),

('andy09','ca',15000,'2004-01-09'),

('andy10','ca',12000,'2003-06-09'),

('andy11','ca',12000,'2002-09-09'),

('andy12','ca',10000,'2003-07-09'),

('andy13','ca',8000,'2003-08-09'),

('andy14','ca',8000,'2003-11-09'),

('andy15','ca',8000,'2003-01-09')

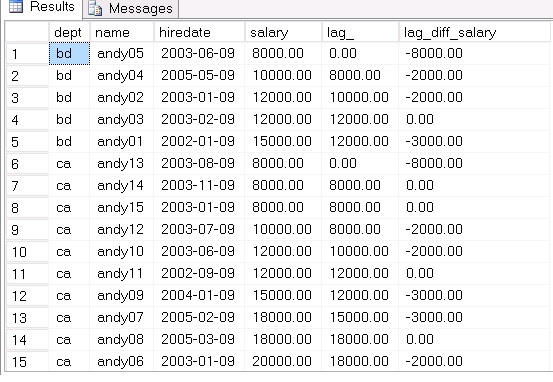
SELECT

dept,name,hiredate,salary,

LAG(salary,1,0) OVER(PARTITION BY dept ORDER BY salary) AS lag\_,

(LAG(salary,1,0) OVER(PARTITION BY dept ORDER BY salary)-salary) AS lag\_diff\_salary

FROM @analytic

我们再看看结果：   


从这个结果看，确实如此，与LEAD相反的行为。这里不多说了，从示例看的非常清楚。

# 6. 分析函数PERCENT\_RANK

微软定义：   
计算 SQL Server 2012 中一组行内某行的相对排名。 使用 PERCENT\_RANK 计算一个值在查询结果集或分区中的相对位置。 PERCENT\_RANK 类似于 CUME\_DIST 函数。   
只看这个，是看不出来啥意思的。我们还是从下面的实例可以看得很清楚：

DECLARE

@analytic TABLE(

name varchar(35) ,

dept varchar(35),

salary money ,

hiredate date

)

INSERT INTO @analytic

VALUES

--bd

('andy01','bd',15000,'2002-01-09'),

('andy02','bd',15000,'2003-01-09'),

('andy03','bd',12000,'2003-02-09'),

('andy04','bd',10000,'2005-05-09'),

('andy05','bd',8000,'2003-06-09'),

--ca

('andy06','ca',20000,'2003-01-09'),

('andy07','ca',18000,'2005-02-09'),

('andy08','ca',18000,'2005-03-09'),

('andy09','ca',15000,'2004-01-09'),

('andy10','ca',12000,'2003-06-09'),

('andy11','ca',12000,'2002-09-09'),

('andy12','ca',10000,'2003-07-09'),

('andy13','ca',8000,'2003-08-09'),

('andy14','ca',8000,'2003-11-09'),

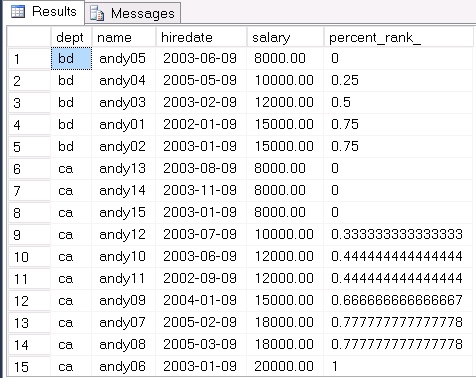
('andy15','ca',8000,'2003-01-09')

SELECT

dept,name,hiredate,salary,

PERCENT\_RANK() OVER(PARTITION BY dept ORDER BY salary) AS percent\_rank\_

FROM @analytic



从这个图可以清楚的：  
最小的SALARY的 PERCENT\_RANK() 始终是0，但有唯一一个最大SALARY是1.其他值的算法是这样的：   
PERCENT\_RANK() = CEILING(小于自己的记录数/（总的记录数-1）)。这样来看是不是要简单清晰得多了。

# 7. 分析函数PERCENTILE\_DISC

微软定义：   
计算 SQL Server 2012 中整个行集内或行集的非重复分区内已排序值的特定百分位数。 对于给定的百分位数的值 P，PERCENTILE\_DISC 对 ORDER BY 子句中表达式的值进行排序，并返回具有最小 CUME\_DIST 值且大于或等于 P 的值（遵照相同的排序规范）。 例如，PERCENTILE\_DISC (0.5) 将计算表达式的第 50 百分位数（也即中值）。 PERCENTILE\_DISC 基于列值的离散分布来计算百分位数；结果等于列中的一个特定值。   
好晦涩的说法，我也没有理解，我们还是看示例：

DECLARE

@analytic TABLE(

name varchar(35) ,

dept varchar(35),

salary money ,

hiredate date

)

INSERT INTO @analytic

VALUES

--bd

('andy01','bd',15000,'2002-01-09'),

('andy02','bd',15000,'2003-01-09'),

('andy03','bd',12000,'2003-02-09'),

('andy04','bd',10000,'2005-05-09'),

('andy05','bd',8000,'2003-06-09'),

--ca

('andy06','ca',20000,'2003-01-09'),

('andy07','ca',18000,'2005-02-09'),

('andy08','ca',18000,'2005-03-09'),

('andy09','ca',15000,'2004-01-09'),

('andy10','ca',11000,'2003-06-09'),

('andy11','ca',1100,'2002-09-09'),

('andy12','ca',10000,'2003-07-09'),

('andy13','ca',8000,'2003-08-09'),

('andy14','ca',8000,'2003-11-09'),

('andy15','ca',8000,'2003-01-09')

SELECT

dept,name,hiredate,salary,

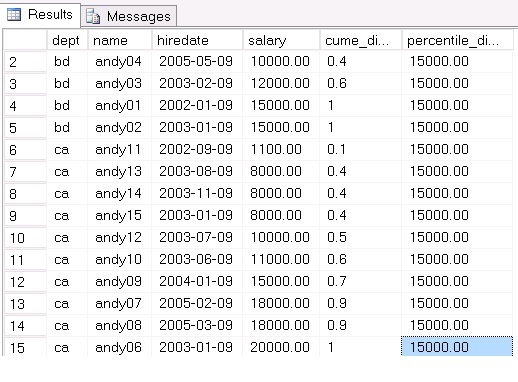
CUME\_DIST() OVER (PARTITION BY dept ORDER BY salary) cume\_dist\_ ,

PERCENTILE\_DISC (0.7) WITHIN GROUP( ORDER BY salary)

OVER(PARTITION BY dept) AS percentile\_disc\_

FROM @analytic

看看结果，我们来分析分析：



其实这个算法很简单：

1. 首先要计算出CUME\_DIST() 的值
2. 其次要比较CUME\_DIST() 的值和PERCENTILE\_DISC (x)给定的x
3. 如果CUME\_DIST() 的值等于x，在各个分组上PERCENTILE\_DISC (x)对应的值（ ORDER BY \*）

都会等于对应的ORDER BY的salary值

4.若CUME\_DIST() 的值不等于x，在各个分组上PERCENTILE\_DISC (x)对应的值（ ORDER BY \*）

都会等于接近于但大于x 的CUME\_DIST() 值的ORDER BY的salary值（有点拗，不过看图就明白了）

# 8. 分析函数PERCENTILE\_CONT

微软定义：   
基于 SQL Server 2012 列值的连续分布计算百分位数。 将内插结果，且结果可能不等于列中的任何特定值。   
同样，我们理解这句话存在困难，我们还是看看示例：

DECLARE

@analytic TABLE(

name varchar(35) ,

dept varchar(35),

salary money ,

hiredate date

)

INSERT INTO @analytic

VALUES

--bd

('andy01','bd',15000,'2002-01-09'),

('andy02','bd',15000,'2003-01-09'),

('andy03','bd',12000,'2003-02-09'),

('andy04','bd',10000,'2005-05-09'),

('andy05','bd',8000,'2003-06-09'),

--ca

('andy06','ca',20000,'2003-01-09'),

('andy07','ca',18000,'2005-02-09'),

('andy08','ca',18000,'2005-03-09'),

('andy09','ca',15000,'2004-01-09'),

('andy10','ca',11000,'2003-06-09'),

('andy11','ca',1100,'2002-09-09'),

('andy12','ca',10000,'2003-07-09'),

('andy13','ca',8000,'2003-08-09'),

('andy14','ca',8000,'2003-11-09'),

('andy15','ca',8000,'2003-01-09')

SELECT

dept,name,hiredate,salary,

ROW\_NUMBER() OVER (PARTITION BY dept ORDER BY salary) row\_number\_ ,

PERCENTILE\_CONT(0.5) WITHIN GROUP( ORDER BY salary)

OVER(PARTITION BY dept) AS percentile\_cont\_

FROM @analytic

--n=5

--x=0.5

--rn=(1+x\*(n-1))=(1+0.5\*(5-1))=3

--crn=ceiling(rn)=3

--frn=floor(rn)=3

--percentile\_cont= 12000

--n=10

--x=0.5

--rn=(1+x\*(n-1))=(1+0.5\*(10-1))=5.5

--crn=ceiling(rn)=6

--frn=floor(rn)=5

--

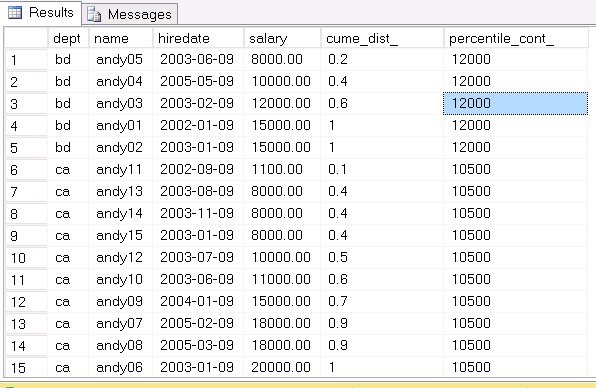
/\*percentile\_cont=

(crn - rn) \* (value of expression for row at frn)

+(rn - frn) \* (value of expression for row at crn)

--select (6-5.5)\*10000.00+(5.5-5)\*11000

\*/

请看看下图：   


这个算法比较复杂，我推测了一下，算法如下：  
n为每个分组的记录数  
x为percentile\_cont的百分位参数值  
rn是计算出来的row number，但这个是逻辑计算出来的，不是对应物理的数据行  
rn=(p+x\*(n-1))  
crn是ceiling(rn)  
frn是floor(rn)

所以算法就很清楚了：

1. 如果(crn = frn = rn) 那么percentile\_cont= (value of expression from row at rn)
2. 如果(crn = frn= rn) 不满足，percentile\_cont=(crn- rn) (value of expression for row at frn) +(rn- frn) (value of expression for row at crn)