

SQLServer中Partition By及row_number 函数使用详解

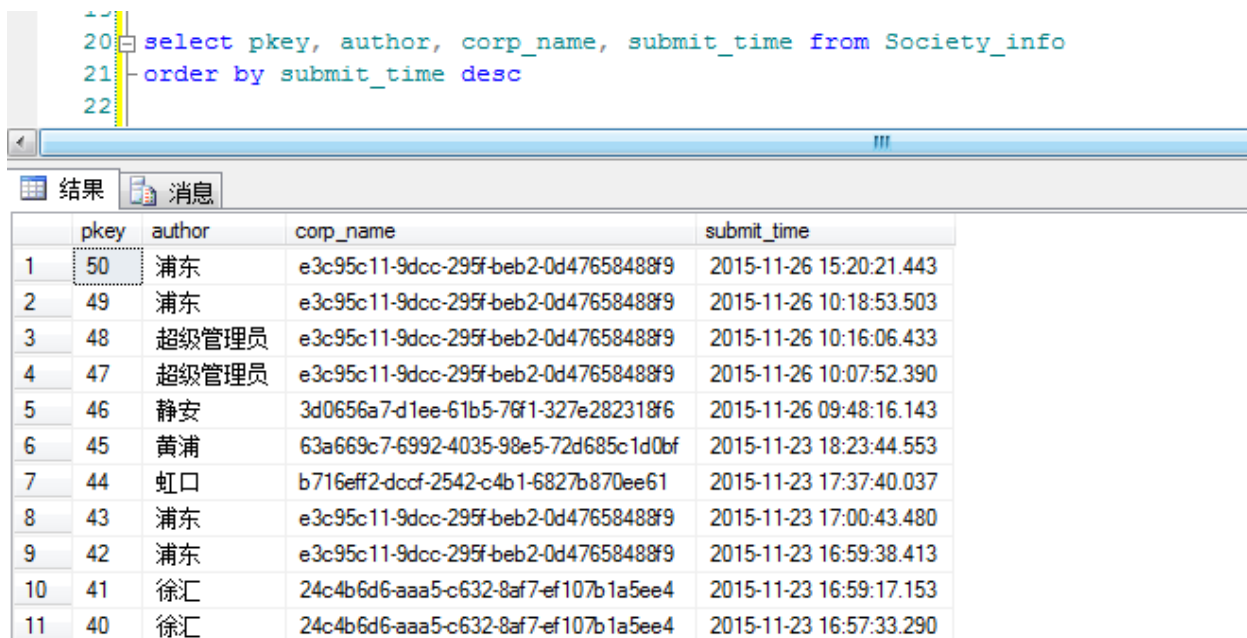
<http://www.jb51.net/article/75533.htm>

partition by关键字是分析性函数的一部分，partition by用于给结果集分组，如果没有指定那么它把整个结果集作为一个分组，本文给大家介绍SQLServer中Partition By及row_number 函数使用详解，需要的朋友参考下

partition by关键字是分析性函数的一部分，它和聚合函数不同的地方在于它能返回一个分组中的多条记录，而聚合函数一般只有一条反映统计值的记录，partition by用于给结果集分组，如果没有指定那么它把整个结果集作为一个分组。

今天群里看到一个问题，在这里概述下：查询出不同分类下的最新记录。一看这不是很简单的么，要分类那就用Group By;要最新记录就用Order By呗。然后在自己的表中试着做出来：

首先呢我把表中的数据按照提交时间倒序出来：



```
20 select pkey, author, corp_name, submit_time from Society_info
21 order by submit_time desc
22
```

	pkey	author	corp_name	submit_time
1	50	浦东	e3c95c11-9dcc-295f-beb2-0d47658488f9	2015-11-26 15:20:21.443
2	49	浦东	e3c95c11-9dcc-295f-beb2-0d47658488f9	2015-11-26 10:18:53.503
3	48	超级管理员	e3c95c11-9dcc-295f-beb2-0d47658488f9	2015-11-26 10:16:06.433
4	47	超级管理员	e3c95c11-9dcc-295f-beb2-0d47658488f9	2015-11-26 10:07:52.390
5	46	静安	3d0656a7-d1ee-61b5-76f1-327e282318f6	2015-11-26 09:48:16.143
6	45	黄浦	63a669c7-6992-4035-98e5-72d685c1d0bf	2015-11-23 18:23:44.553
7	44	虹口	b716eff2-dccf-2542-c4b1-6827b870ee61	2015-11-23 17:37:40.037
8	43	浦东	e3c95c11-9dcc-295f-beb2-0d47658488f9	2015-11-23 17:00:43.480
9	42	浦东	e3c95c11-9dcc-295f-beb2-0d47658488f9	2015-11-23 16:59:38.413
10	41	徐汇	24c4b6d6-aaa5-c632-8af7-ef107b1a5ee4	2015-11-23 16:59:17.153
11	40	徐汇	24c4b6d6-aaa5-c632-8af7-ef107b1a5ee4	2015-11-23 16:57:33.290

“corp_name”就是分类的GUID（请原谅我命名的随意性）。OK，这里按照最开始的想法加上Group By来看一下显示效果：

32	
33	<code>select corp_name, submit_time from Society_info</code>
34	<code>group by corp_name, submit_time</code>
35	<code>order by corp_name, submit_time</code>
36	

结果		消息
	corp_name	submit_time
1	24c4b6d6-aaa5-c632-8af7-ef107b1a5ee4	2015-11-23 16:57:33.290
2	24c4b6d6-aaa5-c632-8af7-ef107b1a5ee4	2015-11-23 16:59:17.153
3	3d0656a7-d1ee-61b5-76f1-327e282318f6	2015-11-26 09:48:16.143
4	63a669c7-6992-4035-98e5-72d685c1d0bf	2015-11-23 18:23:44.553
5	b716eff2-dccf-2542-c4b1-6827b870ee61	2015-11-23 17:37:40.037
6	e3c95c11-9dcc-295f-beb2-0d47658488f9	2015-11-23 16:59:38.413
7	e3c95c11-9dcc-295f-beb2-0d47658488f9	2015-11-23 17:00:43.480
8	e3c95c11-9dcc-295f-beb2-0d47658488f9	2015-11-26 10:07:52.390
9	e3c95c11-9dcc-295f-beb2-0d47658488f9	2015-11-26 10:16:06.433
10	e3c95c11-9dcc-295f-beb2-0d47658488f9	2015-11-26 10:18:53.503
11	e3c95c11-9dcc-295f-beb2-0d47658488f9	2015-11-26 15:20:21.443

呃，嗯。这尼玛和想象中的结果不一样啊，看来写代码还是要理性分析问题，意念是无法控制结果滴！

既然要求是不同分类的数据，除了使用Group By之外，还有别的函数能用吗？度娘了一下结果还真有，over (partition by) 函数，那么它和平时用的Group By有什么区别呢？

Group By除了对结果进行单纯的分组之外呢，一般都和聚合函数一起使用，Partition By也具有分组功能，属于Oracle的分析函数，在这里就不详细的不啦不啦不啦了。

看代码：

```
2 select corp_name, submit_time, t from (
3 select author, corp_name, submit_time,
4 ROW_NUMBER() over(partition by corp_name order by submit_time desc ) as t
5 from Society_info a ) b
```

结果

消息

	corp_name	submit_time	t
1	24c4b6d6-aaa5-c632-8af7-ef107b1a5ee4	2015-11-23 16:59:17.153	1
2	24c4b6d6-aaa5-c632-8af7-ef107b1a5ee4	2015-11-23 16:57:33.290	2
3	3d0656a7-d1ee-61b5-76f1-327e282318f6	2015-11-26 09:48:16.143	1
4	63a669c7-6992-4035-98e5-72d685c1d0bf	2015-11-23 18:23:44.553	1
5	b716eff2-dccf-2542-c4b1-6827b870ee61	2015-11-23 17:37:40.037	1
6	e3c95c11-9dcc-295f-beb2-0d47658488f9	2015-11-26 15:20:21.443	1
7	e3c95c11-9dcc-295f-beb2-0d47658488f9	2015-11-26 10:18:53.503	2
8	e3c95c11-9dcc-295f-beb2-0d47658488f9	2015-11-26 10:16:06.433	3
9	e3c95c11-9dcc-295f-beb2-0d47658488f9	2015-11-26 10:07:52.390	4
10	e3c95c11-9dcc-295f-beb2-0d47658488f9	2015-11-23 17:00:43.480	5
11	e3c95c11-9dcc-295f-beb2-0d47658488f9	2015-11-23 16:59:38.413	6

over(partition by corp_name order by submit_time desc) as t 。就是按照corp_name 分类并按时间倒序出来，“t” 这里一列呢就是不同corp_name类出现的次数，需求是只查询出不同分类的最新提交数据，那么我们只需要针对“t”再进行一次筛选即可：

```

24 |
25 | select author, corp_name, submit_time, t from (
26 | select author, corp_name, submit_time
27 | , ROW_NUMBER() over(partition by corp_name order by submit_time desc) as t
28 | from Society_info a) b
29 | where b.t=1
30 |
31 |

```

	author	corp_name	submit_time	t
1	徐汇	24c4b6d6-aaa5-c632-8af7-ef107b1a5ee4	2015-11-23 16:59:17.153	1
2	静安	3d0656a7-d1ee-61b5-76f1-327e282318f6	2015-11-26 09:48:16.143	1
3	黄浦	63a669c7-6992-4035-98e5-72d685c1d0bf	2015-11-23 18:23:44.553	1
4	虹口	b716eff2-dccf-2542-c4b1-6827b870ee61	2015-11-23 17:37:40.037	1
5	浦东	e3c95c11-9dcc-295f-beb2-0d47658488f9	2015-11-26 15:20:21.443	1

好啦，结果已经出来，不求各位看官喜欢，但求看在我头像中的胸器望点个赞， 好人一生平安哦！！

ps: SQL Server数据库partition by 与ROW_NUMBER()函数使用详解

关于SQL的partition by 字段的一些用法心得

先看例子：

```
if object_id('TESTDB') is not null drop table TESTDB
```

```
create table TESTDB(A varchar(8), B varchar(8))
```

```
insert into TESTDB
```

```
select 'A1', 'B1' union all
```

```
select 'A1', 'B2' union all
```

```
select 'A1', 'B3' union all
```

```
select 'A2', 'B4' union all
```

```
select 'A2', 'B5' union all
```

```
select 'A2', 'B6' union all
```

```
select 'A3', 'B7' union all
```

```
select 'A3', 'B3' union all
```

```
select 'A3', 'B4'
```

-- 所有的信息

```
SELECT * FROM TESTDB
```

```
A B
```

```
-----
```

```
A1 B1
```

```
A1 B2
```

```
A1 B3
```

```
A2 B4
```

```
A2 B5
```

A2 B6

A3 B7

A3 B3

A3 B4

-- 使用PARTITION BY 函数后

```
SELECT *,ROW_NUMBER() OVER(PARTITION BY A ORDER BY A DESC) NUM FROM TESTDB
```

```
A  B  NUM
```

A1 B1 1

A1 B2 2

A1 B3 3

A2 B4 1

A2 B5 2

A2 B6 3

A3 B7 1

A3 B3 2

A3 B4 3

可以看到结果中多出一列NUM 这个NUM就是说明了相同行的个数，比如A1有3个，他就给每个A1标上是第几个。

-- 仅仅使用ROW_NUMBER() OVER的结果

```
SELECT *,ROW_NUMBER() OVER(ORDER BY A DESC)NUM FROM TESTDB
```

```
A  B  NUM
```

A3 B7 1

A3 B3 2

A3 B4 3

A2 B4 4

A2 B5 5

A2 B6 6

A1 B1 7

A1 B2 8

A1 B3 9

可以看到它只是单纯标出了行号。

-- 深入一点应用

```
SELECT A = CASE WHEN NUM = 1 THEN A ELSE '' END, B
FROM (SELECT A, NUM = ROW_NUMBER() OVER(PARTITION BY A ORDER BY A DESC) FROM
TESTDB) T
```

A B

A1 B1

B2

B3

A2 B4

B5

B6

A3 B7

B3

B4

接下来我们就通过几个实例来一一介绍ROW_NUMBER()函数的使用。

实例如下：

1. 使用row_number()函数进行编号，如

```
select email, customerID, ROW_NUMBER() over(order by psd) as rows from
QT_Customer
```

原理：先按psd进行排序，排序完后，给每条数据进行编号。

2. 在订单中按价格的升序进行排序，并给每条记录进行排序代码如下：

```
select DID, customerID, totalPrice, ROW_NUMBER() over(order by totalPrice) as rows
from OP_Order
```

3. 统计出每一个客户的所有订单并按每一个客户下的订单的金额 升序排序，同时给每一个客户的订单进行编号。这样就知道每个客户下几单了。

如图：

SQL Server数据库ROW_NUMBER()使用详解

代码如下：

```
select ROW_NUMBER() over(partition by customerID order by totalPrice) as
rows, customerID, totalPrice, DID from OP_Order
```

4. 统计每一个客户最近下的订单是第几次下的订单。

SQL Server数据库ROW_NUMBER()使用详解

代码如下：

```
with tabs as
(
select ROW_NUMBER() over(partition by customerID order by totalPrice) as
rows,customerID,totalPrice, DID from OP_Order
)
select MAX(rows) as '下单次数',customerID from tabs group by customerID
```

5. 统计每一个客户所有的订单中购买的金额最小，而且并统计改订单中，客户是第几次购买的。

如图：

A rectangular box with a light gray background containing the text "SQL Server 数据库 ROW_NUMBER() 使用详解".

上图：rows表示客户是第几次购买。

思路：利用临时表来执行这一操作。

1. 先按客户进行分组，然后按客户的下单的时间进行排序，并进行编号。
2. 然后利用子查询查找出每一个客户购买时的最小价格。
3. 根据查找出每一个客户的最小价格来查找相应的记录。

代码如下：

```
with tabs as
(
select ROW_NUMBER() over(partition by customerID order by insDT) as
rows,customerID,totalPrice, DID from OP_Order
)
select * from tabs
where totalPrice in
(
select MIN(totalPrice)from tabs group by customerID
)
```

6. 筛选出客户第一次下的订单。

A rectangular box with a light gray background containing the text "SQL Server 数据库 ROW_NUMBER() 使用详解".

思路。利用rows=1来查询客户第一次下的订单记录。

代码如下：

```
with tabs as
(
select ROW_NUMBER() over(partition by customerID order by insDT) as rows,* from
```

OP_Order

)

```
select * from tabs where rows = 1
```

```
select * from OP_Order
```

7. rows_number() 可用于分页

思路：先把所有的产品筛选出来，然后对这些产品进行编号。然后在where子句中进行过滤。

8. 注意：在使用over等开窗函数时，over里头的分组及排序的执行晚于“where, group by, order by”的执行。

如下代码：

```
select
```

```
ROW_NUMBER() over(partition by customerID order by insDT) as rows,
```

```
customerID, totalPrice, DID
```

```
from OP_Order where insDT>'2011-07-22'
```

以上代码是先执行where子句，执行完后，再给每一条记录进行编号。