

原 Oracle group by及其若干相关函数的一些说明 收藏

Oracle的group by除了基本用法以外，还有3种扩展用法，分别是rollup、cube、grouping sets。

假设有一个表test，有A、B、C、D、E5列。

如果使用group by rollup(A,B,C)，首先会对(A、B、C)进行GROUP BY，然后对(A、B)进行GROUP BY，然后是(A)进行GROUP BY，最后对全表进行GROUP BY操作。roll up的意思是“卷起”，这也可以帮助我们理解group by rollup就是对选择的列从右到左以一次少一列的方式进行grouping直到所有列都去掉后的grouping(也就是全表grouping)，对于n个参数的rollup，有n+1次的grouping。以下2个sql的结果集是一样的：

Select A,B,C,sum(E) from test group by rollup(A,B,C)和

Select A,B,C,sum(E) from test group by A,B,C

union all

Select A,B,null,sum(E) from test group by A,B

union all

Select A,null,null,sum(E) from test group by A

union all

Select null,null,null,sum(E) from test

cube的意思是立方，对cube的每个参数，都可以理解为取值为参与grouping和不参与grouping两个值的一个维度，然后所有维度取值组合的集合就是grouping的集合，对于n个参数的cube，有2^n次的grouping。如果使用group by cube(A,B,C)，则首先会对(A、B、C)进行GROUP BY，然后依次是(A、B)，(A、C)，(A)，(B、C)，(B)，(C)，最后对全表进行GROUP BY操作，一共是2^3=8次grouping。同rollup一样，也可以用基本的group by加上结果集的union all写出一个与group by cube结果集相同的sql：

用户操作

[发私信] [加为好友]

订阅我的博客

0

位读者

POWERED BY FEEDSKY

订阅





+ 订阅到 鲜果

+ 订阅到 Google

+ 订阅到 抓虾

roland_wg的公告

文章分类

-  Linux 操作系统
-  MSSQL
-  MYSQL
-  ORACLE
-  Windows
-  存储
-  生活点滴
-  数据库理论
-  网络

收藏

Select A,B,C,sum(E) from test group by cube(A,B,C);

Select A,B,C,sum(E) from test group by A,B,C

union all

Select A,B,null,sum(E) from test group by A,B

union all

Select A,null,C,sum(E) from test group by A,C

union all

Select A,null,null,sum(E) from test group by A

union all

Select null,B,C,sum(E) from test group by B,C

union all

Select null,B,null,sum(E) from test group by B

union all

Select null,null,C,sum(E) from test group by C

union all

Select null,null,null,sum(E) from test

- grouping sets就是对参数中的每个参数做grouping，也就是有几个参数做几次grouping，例如使用group by grouping sets(A,B,C)，则对(A),(B),(C)进行group by，如果使用group by grouping sets((A,B),C),则对(A,B),(C)进行group by。甚至groupin g by grouping set(A,A)都是语法允许的，也就是对(A)进行2次group by, grouping sets的参数允许重复

需要说明的几点：

- 在rollup和cube的说明中分别给出了用基本group by加结果集union all给出了结果集相同的sql，但这只是为了理解的方便而给出的sql，并不说明rollup和cube与基本group by加结果集union all等价。实际上两者的内部机制是安全不一样的，前者除了写法简洁以外，运行时不需多次扫描表，效率远比后者高。
- 3种扩展用法的参数可以是源表中的某一个具体的列，也可以是若干列经过计算而形成的一个新列（比如说A+B，A||B），也可以是这两种列的一个集合（例如（A+B，C）），对于grouping set更是特殊，可以是空集合(),表示对全表进行group by。
- Group by的基本用法以及这3种扩展用法可以组合使用，也就是说可以出现group by A,rollup(A,B)这样的用法，oracle将对

[BLOG](#)

[IT类](#)

[IT公司网站](#)

[IP存储](#)

[创新科存储技术有限公司](#)

[北京北亚数据恢复中心](#)

[北京建科海纳科技有限公司-中小型存储服务器](#)

[大恒存储](#)

[IT专业技术门户网站](#)

[DOIT存储论坛](#)

[IT168——全原创IT主流咨询](#)

[IT专家网——以精准定位和领先的数据库营销服务于跨行业的CIO和CTO](#)

[IT商网——提供专业权威的资讯展示和海量、准确、及时的信息服务，IT厂商渠道部门、渠道商及IT产品购买者](#)

[天极网——提供IT产品应用、评测、行情报价、导购、软件下载、互动社区等内容](#)

[搜新网——个人、流媒体软件专注网站](#)

[比特网——提供全面、深刻的IT原创报道与产业评论和权威实用的企业信息化资讯](#)

[至顶网](#)

[计世网——专业价值 行业网群](#)

[其他门户网站](#)

[专业技术blog](#)

[IBM_SystemX](#)

[Alibaba DBA Team](#)

出现在group by中的每种用法的grouping列集合做笛卡尔积然后对其中的每一个元素做group by。这话说起来挺绕口，举例说明吧，group by A,rollup(A,B)，基本用法的grouping集合是(A),rollup(A,B)的grouping集合是((A,B),(A),()),两个集合的笛卡尔积集合是((A,A,B),(A,A),(A))，所以会首先对(A,A,B)做group by，然后对(A,A)做group by，最后对(A)做group by。实际上对(A,A,B)做group by和对(A,B)做group by两者是完全等价的(group by A,A,B结果和group by A,B完全一样)，同理对(A,A)做group by和对(A)做group by也是等价的。简化后的结果就是首先对(A,B)做group by，然后对(A)做group by，最后再对(A)做group by。下面给出两个等价的sql以便理解：

Select A,B,sum(E) from test1 group by A,rollup(A,B);

Select A,B,sum(E) from test1 group by A,B

Union all

Select A,null,sum(E) from test1 group by A

Union all

Select A,null,sum(E) from test1 group by A;

相关函数grouping(),grouping_id(),group_id()的用法：

- grouping():参数只有一个，而且必须为group by中出现的某一列，表示结果集的一行是否对该列做了grouping。对于对该列做了grouping的行而言，grouping()=0，反之为1；
- grouping_id():参数可以是多个，但必须为group by中出现的列。Grouping_id()的返回值其实就是参数中的每列的grouping()值的二进制向量，例如如果grouping(A)=1，grouping(B)=0，则grouping_id(A,B)的返回值就是二进制的10，转成10进制就是2。
- group_id():无参数。见上面的说明3)，group by对某些列的集合会进行重复的grouping，而实际上绝大多数情况下对结果集中的这些重复行是不需要的，那就必须有办法剔出这些重复grouping的行。当结果集中有n条重复grouping而形成的行时，每行的group_id()分别是0,1,...,n，这样我们在条件中加入一个group_id()<1就可以剔出这些重复grouping的行了。

扩展用法及相关函数的应用例子：

数据库ACCINF表结构为：区域号area_no，机构号org_no，账户号acc_no，账户余额acc_bal，现有一个 报表要求先按机构统计每个机构的总余额，最后是所有区域的总余额，也就是说报表格式为：

区域1 机构1 余额数

区域1 机构2 余额数

区域2 机构3 余额数

区域2 机构4 余额数

[ccnp_Server](#)

[gzzg's BLOG](#)

[MKing's Blog](#)

[NinGoo's Notes | Database & Datawarehouse](#)

[Oracle粉丝网-九鼎云州](#)

[SELECT blog FROM Fenng.Thought](#)

[VonGates 學習筆記](#)

[yangtingkun的个人空间](#)

[梦想有多远登录](#)

[菩提老祖的博客](#)

[蚊子窝](#)

存档

[2009年09月\(5\)](#)

[2009年08月\(12\)](#)

[2009年07月\(20\)](#)

[2009年06月\(13\)](#)

[2009年05月\(6\)](#)

[2009年04月\(4\)](#)

[2009年03月\(3\)](#)

[2009年02月\(1\)](#)

[2009年01月\(7\)](#)

[2008年12月\(4\)](#)

[2008年11月\(1\)](#)

[2008年10月\(6\)](#)

[2008年09月\(16\)](#)

[2008年08月\(22\)](#)

总合计 余额数

以往的sql是通过先计算出各机构的总余额，然后在程序里将所有的余额相加计算出所有区域的总余额。但使用扩展用法之后的sql就非常简单了：

```
Select nvl(area_no,'总合计'),org_no,sum(acc_bal) from acc_inf group by rollup((area_no,org_no));
```

如果使用上面的函数，还可以有其它的写法，一个例子是：

```
Select nvl(area_no,'总合计'),org_no,sum(acc_bal) from acc_inf group by rollup(area_no,org_no) having (grouping_id(area_no,org_no)=0 or grouping_id(area_no,org_no)=3)
```

#####

2005-05-10 09:40 First Publish.
Oracle的GROUP BY语句除了最基本的语法外，还支持ROLLUP和CUBE语句。如果是ROLLUP(A, B, C)的话，首先会对(A、B、C)进行GROUP BY，然后对(A、B)进行GROUP BY，然后是(A)进行GROUP BY，最后对全表进行GROUP BY操作。如果是GROUP BY CUBE(A, B, C)，则首先会对(A、B、C)进行GROUP BY，然后依次是(A、B)，(A、C)，(A)，(B、C)，(B)，(C)，最后对全表进行GROUP BY操作。 grouping_id()可以美化效果：

Oracle的GROUP BY语句除了最基本的语法外，还支持ROLLUP和CUBE语句。

除本文内容外，你还可参考：
分析函数参考手册：<http://xsb.itpub.net/post/419/33028>
分析函数使用例子介绍：<http://xsb.itpub.net/post/419/44634>

```
SQL> create table t as select * from dba_indexes;
```

表已创建。

```
SQL> select index_type, status, count(*) from t group by index_type, status;
```

INDEX_TYPE	STATUS	COUNT(*)

LOB	VALID	51
NORMAL	N/A	25
NORMAL	VALID	479
CLUSTER	VALID	11

下面来看看ROLLUP和CUBE语句的执行结果。

```
SQL> select index_type, status, count(*) from t group by rollup(index_type, status);
```

INDEX_TYPE	STATUS	COUNT(*)

LOB	VALID	51
LOB		51
NORMAL	N/A	25
NORMAL	VALID	479
NORMAL		504
CLUSTER	VALID	11

```
CLUSTER 11
566
```

已选择8行。

```
SQL> select index_type, status, count(*) from t group by cube(index_type, status);
```

```
INDEX_TYPE STATUS COUNT(*)
-----
566
N/A 25
VALID 541
LOB 51
LOB VALID 51
NORMAL 504
NORMAL N/A 25
NORMAL VALID 479
CLUSTER 11
CLUSTER VALID 11
```

已选择10行。

查询结果不是很一目了然，下面通过Oracle提供的函数GROUPING来整理一下查询结果。

```
SQL> select grouping(index_type) g_ind, grouping(status) g_st, index_type, status, count(*)
2 from t group by rollup(index_type, status) order by 1, 2;
```

```
G_IND G_ST INDEX_TYPE STATUS COUNT(*)
-----
0 0 LOB VALID 51
0 0 NORMAL N/A 25
0 0 NORMAL VALID 479
0 0 CLUSTER VALID 11
0 1 LOB 51
0 1 NORMAL 504
0 1 CLUSTER 11
1 1 566
```

已选择8行。

这个查询结果就直观多了，和不带ROLLUP 语句的GROUP BY相比，ROLLUP增加了对INDEX_TYPE的GROUP BY统计和对所有记录的GROUP BY统计。

也就是说，如果是ROLLUP(A, B, C)的话，首先会对(A、 B、 C)进行GROUP BY，然后对(A、 B)进行GROUP BY，然后是(A)进行GROUP BY，最后对全表进行GROUP BY操作。

下面看看CUBE语句。

```
SQL> select grouping(index_type) g_ind, grouping(status) g_st, index_type, status, count(*)
2 from t group by cube(index_type, status) order by 1, 2;
```

```
G_IND G_ST INDEX_TYPE STATUS COUNT(*)
-----
0 0 LOB VALID 51
0 0 NORMAL N/A 25
0 0 NORMAL VALID 479
0 0 CLUSTER VALID 11
0 1 LOB 51
```

```
0 1 NORMAL 504
0 1 CLUSTER 11
1 0 N/A 25
1 0 VALID 541
1 1 566
```

已选择10行。

和ROLLUP相比，CUBE又增加了对STATUS列的GROUP BY统计。

如果是GROUP BY CUBE(A, B, C)，则首先会对(A、 B、 C)进行GROUP BY，然后依次是(A、 B)，(A、 C)，(A)，(B、 C)，(B)，(C)，最后对全表进行GROUP BY操作。

除了使用GROUPING函数，还可以使用GROUPING_ID来标识GROUP BY结果。

```
SQL> select grouping_id(index_type, status) g_ind, index_type, status, count(*)
2 from t group by rollup(index_type, status) order by 1;
```

```
G_IND INDEX_TYPE STATUS COUNT(*)
-----
0 LOB VALID 51
0 NORMAL N/A 25
0 NORMAL VALID 479
0 CLUSTER VALID 11
1 LOB 51
1 NORMAL 504
1 CLUSTER 11
3 566
```

已选择8行。

```
SQL> select grouping_id(index_type, status) g_ind, index_type, status, count(*)
2 from t group by cube(index_type, status) order by 1;
```

```
G_IND INDEX_TYPE STATUS COUNT(*)
-----
0 LOB VALID 51
0 NORMAL N/A 25
0 NORMAL VALID 479
0 CLUSTER VALID 11
1 LOB 51
1 NORMAL 504
1 CLUSTER 11
2 N/A 25
2 VALID 541
3 566
```

已选择10行。

grouping_id()可以美化效果：

```
select DECODE(GROUPING_ID(C1), 1, '合计', C1) D1,
DECODE(GROUPING_ID(C1, C2), 1, '小计', C2) D2,
DECODE(GROUPING_ID(C1, C2, C1 + C2), 1, '小计', C1 + C2) D3,
count(*),
GROUPING_ID(C1, C2, C1 + C2, C1 + 1, C2 + 1),
GROUPING_ID(C1)
```

```
from T2
group by rollup(C1, C2, C1 + C2, C1 + 1, C2 + 1);
```

=====

1.报表合计专用的**Rollup**函数

销售报表

广州 1月 2000元

广州 2月 2500元

广州 4500元

深圳 1月 1000元

深圳 2月 2000元

深圳 3000元

所有地区 7500元

以往的查询SQL:

```
Select area,month,sum(money) from SaleOrder group by area,month
```

然后广州，深圳的合计和所有地区合计都需要在程序里自行累计

1.其实可以使用如下SQL:

```
Select area,month,sum(total_sale) from SaleOrder group by rollup(area,month)
```

就能产生和报表一模一样的纪录

2.如果year不想累加，可以写成

```
Select year,month,area,sum(total_sale) from SaleOrder group by year, rollup(month,area)
```

另外Oracle 9i还支持如下语法:

```
Select year,month,area,sum(total_sale) from SaleOrder group by rollup((year,month),area)
```

3.如果使用Cube(area,month)而不是RollUp(area,month)，除了获得每个地区的合计之外，还将获得每个月份的合计，在报表最后显示。

4.Grouping让合计列更好读

RollUp在显示广州计时，月份列为NULL，但更好的做法应该是显示为"所有月份"

Grouping就是用来判断当前Column是否是一个合计列，1为yes，然后用Decode把它转为"所有月份"

```
Select Decode(Grouping(area),1,'所有地区',area) area, Decode(Grouping(month),1,'所有月份',month), sum(money) From SaleOrder Group by RollUp(area,month);
```

2.对多级层次查询的**start with.....connect by**

比如人员组织,产品类别,Oracle提供了很经典的方法

```
SELECT LEVEL, name, emp_id,manager_emp_id FROM employee START WITH manager_emp_id is null CONNECT BY PRIOR emp_id = manager_emp_id;
```

上面的语句demo了全部的应用,start with指明从哪里开始遍历树,如果从根开始,那么它的manager应该是Null,如果从某个职员开始,可以写成emp_id='11'

CONNECT BY 就是指明父子关系,注意PRIOR位置

另外还有一个LEVEL列,显示节点的层次

3.更多报表/分析决策功能

3.1 分析功能的基本结构

分析功能() over(partition子句,order by子句,窗口子句)

概念上很难讲清楚,还是用例子说话比较好.

3.2 Row_Number 和 Rank, DENSE_Rank

用于选出Top 3 sales这样的报表

当两个业务员可能有相同业绩时,就要使用Rank和Dense_Rank

比如

金额 RowNum Rank Dense_Rank

张三 4000 元 1 1 1

李四 3000 元 2 2 2

钱五 2000 元 3 3 3

孙六 2000 元 4 3 3

丁七 1000 元 5 5 4

这时,应该把并列第三的钱五和孙六都选进去,所以用Ranking功能比RowNumber保险.至于Desnse还是Ranking就看具体情况了。

SELECT salesperson_id, SUM(tot_sales) sp_sales, RANK() OVER (ORDER BY SUM(tot_sales) DESC) sales_rank FROM orders GROUP BY salesperson_id

3.3 NTILE 把纪录平分成甲乙丙丁四等

比如我想取得前25%的纪录,或者把25%的纪录当作同一个level平等对待,把另25%当作另一个Level平等对待

SELECT cust_nbr, SUM(tot_sales) cust_sales, NTILE(4) OVER (ORDER BY SUM(tot_sales) DESC) sales_quartile FROM orders GROUP BY cust_nbr ORDER BY 3, 2 DESC;

NTITLE(4)把纪录以 SUM(tot_sales)排序分成4份.

3.4 辅助分析列和Windows Function

报表除了基本事实数据外,总希望旁边多些全年总销量,到目前为止的累计销量,前后三个月的平均销量这样的列来参考.

这种前后三个月的平均和到目前为止的累计销量就叫windows function, 见下例

SELECT month, SUM(tot_sales) monthly_sales, SUM(SUM(tot_sales)) OVER (ORDER BY month ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) max_preceeding FROM orders GROUP BY month ORDER BY month;

SELECT month, SUM(tot_sales) monthly_sales, AVG(SUM(tot_sales)) OVER (ORDER BY month ROWS BETWEEN 1 PRECEDING AND 1 FOLLOWING) rolling_avg


```
FROM orders GROUP BY month ORDER BY month;
```

Windows Function的关键就是Windows子句的几个取值

1 PRECEDING 之前的一条记录

1 FOLLOWING 之后的一条记录

UNBOUNDED PRECEDING 之前的所有记录

CURRENT ROW 当前纪录

4.SubQuery总结

SubQuery天天用了,理论上总结一下.SubQuery 分三种

1.Noncorrelated 子查询 最普通的样式.

2.Correlated Subqueries 把父查询的列拉到子查询里面去,头一回cyt教我时候理解了半天.

3.Inline View 也被当成最普通的样式用了.

然后Noncorrelated 子查询又有三种情况

1.返回一行一列 where price < (select max(price) from goods)

2.返回多行一列 where price>= ALL (select price from goods where type=2)

or where NOT price< ANY(select price from goods where type=2)

最常用的IN其实就是=ANY()

3.返回多行多列 一次返回多列当然就节省了查询时间

```
UPDATE monthly_orders SET (tot_orders, max_order_amt) = (SELECT COUNT(*), MAX(sale_price) FROM cust_order) DELETE FROM line_item WHERE (order_nbr
, part_nbr) IN (SELECT order_nbr, part_nbr FROM cust_order c)
```

=====

```
/*-----理解grouping sets
select a, b, c, sum( d ) from t
group by grouping sets ( a, b, c )
```

等效于

```
select * from (
select a, null, null, sum( d ) from t group by a
union all
select null, b, null, sum( d ) from t group by b
union all
select null, null, c, sum( d ) from t group by c
)
*/
```

```
select DECODE(GROUPING_ID(C1), 1, '合计', C1) D1,
DECODE(GROUPING_ID(C1, C2), 1, '小计', C2) D2,
DECODE(GROUPING_ID(C1, C2, C1 + C2), 1, '小计', C1 + C2) D3,
```

```
count(*),
GROUPING_ID(C1, C2, C1 + C2, C1 + 1, C2 + 1),
GROUPING_ID(C1)
from T2
group by rollup(C1, C2, C1 + C2, C1 + 1, C2 + 1);
```

发表于 @ 2009年07月03日 15:40:00 | [评论\(0\)](#) | [举报](#) | [收藏](#)

旧一篇:Oracle裸设备(raw device)问答 | 新一篇:RDA是Remote Diagnostic Agent 的简称

给roland_wg的留言

姓 名：

主 页：

校验码：



只有已注册用户才能发表评论！[请登录或注册](#)

提交留言