第十章

子查询因子化
subquery factoring
即with 子句
ANSI 标准术语为：公共表表达式（common table expression, CTE）

10.1 标准用法

eg: 没有进行子查询因子化的交叉数据分析查询

```
select * from (
    select /*+ gather_plan_statistics */
    product, channel, quarter, country, quantity_sold
    from (
        select prod_name product, country_name country, channel_id channel, substr(calendar_quarter_desc,6,2) quarter,
sum(amount_sold) amount_sold, sum(quantity_sold) quantity_sold
        from sh.sales
        join sh.times on times.time_id = sales.time_id
        join sh.custoemrs on customers.cust_id = sales.cust_id
        join sh.countries on countries.country_id = customers.country_id
        join sh.products on products.prod_id = sales.prod_id
        group by prod_name, country_name, channel_id, substr(calendar_quarter_desc, 6, 2)
    )
) pivot (
    sum(quantity_sold)
    for (channel, quarter) in
        (5,'02') as catalog_q2,
        (4,'01') as internet_q1,
        (4,'04') as internet_q4,
        (2,'02') as partners_q2,
        (9,'03') as tele_q3
    )
)
order by product, country;
```

eg: 进行子查询因子化的交叉表

```
select sales_countries as (
    select /*+ gather_plan_statistics */
    cu.cust_id, co.country_name
    from sh.countries co, sh.customers cu
    where cu.country_id = co.country_id
),
top_sales as (
    select p.prod_name, sc.country_name, s.channel_id, t.calendar_quarter_desc, s.amount_sold, s.quantity_sold
    from sh.sales s
    join sh.times t on t.time_id = s.time_id
    join sh.customers c on c.cust_id = s.cust_id
    join sales_countries sc on sc.cust_id = c.cust_id
    join sh.products p on p.prod_id = s.prod_id
),
sales_rpt as (
    select prod_name product, country_name country, channel_id channel, substr(calendar_quarter_desc, 6, 2) quarter,
sum(amount_sold) amount_sold, sum(quantity_sold_ quantity_sold
    from top_sales
    group by prod_name, country_nama,e channel_id, substr(calendar_quarter_desc, 6, 2)
)
select * from
(
    select product, channel, quarter, country, quantity_sold
    from sales_rpt
) pivot (
    sum(quantity_sold)
    for (channel, quarter) in
    (
        (5,'02') as catalog_q2,
        (4,'01') as internet_q1,
        (4,'04') as internet_q4,
        (2,'02') as partners_q2,
        (9,'03') as tele_q3
    )
)
order by product, country;
```

10.2 SQL 优化

10.2.1 测试执行计划

eg: WITH和MATERIALIZE

```
with cust as (
    select /*+ materialize gather_plan_statistics */
    b.cust_income_level, a.country_name
    from sh.customers b
    join sh.countries a on a.country_id = b.country_id
)
select country_name, cust_income_level, count(country_name) country_cust_count
from cust c
having count(country_name) >
(
    select count(*) * .01
    from cust c2
)
or count(cust_income_level) >=
(
    select median(income_level_count)
    from (
        select cust_income_level, count(*) *.25 income_level_count
        from cust
        group by cust_income_level
    )
)
group by country_name, cust_income_level
order by 1,2;
```

eg: WITH和INLINE提示

```
wuth cust as (
    select /*+ inline gather_plan_statistics */
    b.cust_income_level, a.country_name
    from sh.customers b
    join sh.countries a on a.country_id = b.country_id
)
...
```

10.2.2 跨多个会话执行的测试

10.2.3 测试查询改变的影响

eg: 修改后的收入查询——INLINE

```
with cust as (
    select /*+ inline gather_plan_statistics */
    b.cust_income_level, a.country_name
    from sh.customers b
    join sh.countries a on a.country_id = b.country_id
),
median_income_set as (
    select /*+ inline */ cust_income_level, count(*) income_level_count
    from cust
    group by cust_income_level
    having count(cust_income_level) > (
        select median(income_level_count) income_level_count
        from (
            select cust_income_level, count(*) income_level_count
            from cust
            group by cust_income_level
        )
    )
)
select country_name, cust_income_level, count(country_name) country_cust_count
from cust c
having count(country_name) >
(
    select count(*) * .01
    from cust c2
)
or cust_income_level in (select mis.cust_income_level from median_income_set mis)
group by country_name, cust_income_level;
```
eg: 修改后的收入查询——MATERIALIZE
```
with cust as (
    select /*+ materialize gather_plan_statistics */
    b.cust_income_level, a.country_name
    from sh.customers b
    join sh.countries a on a.country_id = b.country_id
),
...
```

10.2.4 寻找其他优化机会

eg: 用来计算成本的老SQL语句
```
select /*+ gather_plan_statistics */
    substr(prod_name, 1, 30) prod_name, channel_desc,
    (select avg(c2.unit_cost) from sh.costs c2 where c2.prod_id = c.prod_id and c2.channel_id = c.channel_id and c2.time_id
between to_date('01/01/2000','mm/dd/yyyy') and to_date('12/31/2000')) avg_cost,
    (select min(c2.unit_cost) from sh.costs c2 where c2.prod_id = c.prod_id and c2.channel_id = c.channel_id and c2.time_id
between to_date('01/01/2000','mm/dd/yyyy') and to_date('12/31/2000','mm/dd/yyyy')) min_cost,
    (select max(c2.unit_cost) from sh.costs c2 where c2.prod_id = c.prod_id and c2.channel_id = c.channel_id and c2.time_id
between to_date('01/01/2000','mm/dd/yyyy') and to_date('12/31/2000','mm/dd/yyyy')) max_cost
from (
    select distinct pr.prod_id, pr.prod_name, ch.channel_id, ch.channel_desc, sh.products pr, sh.costs co
    where ch.channel_id = co.channel_id
    and co.prod_id = pr.prod_id
    and co.time_id between ('01/01/2000','mm/dd/yyyy') and to_date('12/31/2000','mm/dd/yyyy')
) c
order by prod_name, channel_desc;
```
eg: 使用WITH子句进行重构后的老SQL语句
```
with bookends as (
    select to_date('01/01/2000','mm/dd/yyyy') begin_date,
    to_date('12/31/2000','mm/dd/yyyy') end_date
    from dual
```

```
),
prodmaster as (
    select distinct pr.prod_id, pr.prod_name, ch.channel_id, ch.channel_desc
    from sh.channels ch, sh.products pr, sh.costs co
    where ch.channel_id = co.channel_id
    and co.prod_id = pr.prod_id
    and co.time_id between (select begin_date from bookends)
    and (select end_date from bookends)
),
cost_compare as (
    select prod_id, channel_id, avg(c2.unit_cost avg_cost, min(c2.unit_cost) min_cost, max(c2.unit_cost) max_cost
    from sh.costs c2
    where c2.time_id between (select begin_date from bookends)
    and (select end_date from bookends)
    group by c2.prod_id, c2.channel_id
)
select /*+ gather_plan_statistics */
    substr(pm.prod_name, 1, 30) prod_name, pm.channel_desc, cc.avg_cost, cc.min_cost, cc.max_cost
    from prodmaster pm
    join cost_compare cc on cc.prod_id = pm.prod_id
    and cc.channel_id = pm.channel_id
order by pm.prod_name, pm.channel_desc;
```

10.2.5 将子查询因子化应用到PL/SQL中

eg: 用PL/SQL来生成消费者报告

```
create global temporary table cust3year (cust_id number);
create global temporary table sales3year(
    cust_id number,
    prod_category varchar2(50),
    total_sale number
)
/
begin
    execute immediate 'truncate table cust3year';
    execute immediate 'truncate table sales3year';
    insert into cust3year
    select cust_id --, count(cust_years) year_count
    from (
        select distinct cust_id, trunc(time_id,'YEAR') cust_years
        from sh.sales
    )
    group by cust_id
    having count(cust_years) >= 3;
    for crec in (select cust_id from cust3year)
    loop
        insert into sales3year
        select s.cust_id, p.prod_category, sum(co.unit_price * s.quantity_sold)
        from sh.sales s
        join sh.products p on p.prod_id = s.prod_id
        join sh.costs co on co.prod_id= s.prod_id
            and cotime_id = s.time_id
        join sh.customers cu on cu.cust_id = s.cust_id
        where s.cust_id = crec.cust_id
        group by s.cust_id, p.prod_category;
    end loop;
end;
/
```

```
break on report
compute sum of total_sale on report
select c3.cust_id, c.cust_last_name, c.cust_first_name, s.prod_category, s.total_sale
from cust3year c3
join sales3year s on s.cust_id = c3.cust_id
join sh.customers c on c.cust_id = c3.cust_id
order by 1,4;
```

eg: 使用WITH子句来生成消费者报告

```
with custyear as (
    select cust_id, extract(year from time_id) sales_year
    from sh.sales
    where extract(year from time_id) between 1998 and 2002
    group by cust_id, extract(year from time_id)
),
custselect as (
    select distinct cust_id
    from (
        select cust_id, count(*) over (partition by cust_id) year_count
        from cust_year
    )
    where year_count >= 3  -- 3 or more years as a customer during period
)
select cu.cust_id, cu.cust_last_name, cu.cust_first_name, p.prod_category, sum(co.unit_price * s.quantity_sold) total_sale
from custselect cs
join sh.sales s on s.cust_id = cs.cust_id
join sh.products p on p.prod_id = s.prod_id
join sh.costs co on co.prod_id = s.prod_id
and co.time_id = s.time_id
join sh.customers cu on cu.cust_id = cs.cust_id
group by cu.cust_id, cu.cust_last_name, cu.cust_first_name, p.prod_category
order by cu.cust_id;
```

10.3 递归子查询

递归子查询因子化
recursive subquery factoring
ANSI标准中称为 递归公共表达式 recursive common table expression
替换了 CONNECT BY 的功能

10.3.1 一个CONNECT BY的例子

eg: 基本的CONNECT BY

```
select lpad(' ', level*2-1,' ') || emp.emp_last_name emp_last_name, emp.emp_first_name, emp.employee_id, emp.mgr_last_name,
emp.mgr_first_name, emp.manager_id, department_name
from (
    select /*+ inline gather_plan_statistics */
    e.last_name emp_last_name, e.first_name emp_first_name, e.employee_id, d.department_id, e.manager_id, d.department_name,
es.last_name mgr_last_name, es.first_name mgr_first_name
    from hr.employees e
    left outer join hr.departments d on d.department_id = e.department_id
    left outer join hr.employees es on es.employee_id = e.manager_id
) emp
connect by prior emp.employee_id = emp.manager_id
start with emp.manager_id is null
order siblings by emp.emp_last_name;
```

10.3.2 使用RSF例子

eg: 基本的递归子查询因子化

```
with emp as (
    select /*+ inline gather_plan_statistics*/
    e.last_name, e.first_name, e.employee_id, e.manager_id, d.department_name
    from hr.employees e
    left outer join hr.departments d on d.department_id = e.department_id
),
emp_recurse (last_name, first_name, employee_id, manager_id, department_name, lvl) as (
    select e.last_name, e.first_name, e.employee_id, e.manager_id, e.department_name, 1 as lvl
    from emp e where e.manager_id is null
    union all
    select emp.last_name, emp.first_name, emp.employee_id, emp.manager_id, emp.department_name, empr.lvl + 1 as lvl
    from emp
    join emp_recurse empr on empr.employee_id = emp.manager_id
)
search depth first by last_name set order1
select lpad(' ', lvl*2-1, ' ') || er.last_name last_name, er.first_name, er.department_name
from emp_recurse er;
```

10.3.3 RSF的限制条件

distinct关键字或group by子句

model子句

聚合函数（但select列表中可以使用分析函数）

引用query_name的子查询

引用query_name作为右表的外联结

10.3.4 与CONNECT BY的不同点

level 换成了 lvl

新特性：search depth first

10.4 复制CONNECT BY的功能

connect by 中有一些层级查询运算符、位列和一个函数在RSF中是无法直接使用的。但功能可以在RSF中复制。

10.4.1 level 伪列

eg: level 伪列

```
select lpad(' ', level*2-1, ' ') || e.last_name last_name, level
from hr.employees e
connect by prior e.employee_id = e.manager_id
start with e.manager_id is null
order siblings by e.last_name;
```

eg: 创建一个LVL列

```
with emp_recurse(employee_id, manager_id, last_name, lvl) as (
    select e.employee_id, null, e.last_name, 1 as lvl
    from hr.employees e
    where e.manager_id is null
    union all
    select e1.employee_id, e1.manager_id, e1.last_ame, e2.lvl + 1 as lvl
    from hr.employees e1
    join emp_recurse e2 on e2.employee_id = e1.manager_id
)
search depth first by last_name set last_name order
select lpad(' ', r.lvl*2-1, ' ') || r.last_name last_name, r.lvl
from emp_recurse r
order by last_name_order;
```

10.4.2 sys_connect_by_path 函数

该函数用来返回组成员层级的知道当前行的值。

eg:

```
select lpad(' ', 2*(level-1)) || e.last_name last_name, sys_connect_by_path(last_name, ';') path
from hr.employees e
```

```
start with e.manager_id is null
connect by prior e.employee_id = e.manager_id
order siblings by e.last_name;
```
eg: 建立你自己的SYS_CONNECT_BY_PATH函数
```
with emp_recurse(employee_id, manager_id, last_name, lvl, path) as (
    select e.employee_id, null, e.last_name, 1 as lvl, ':' || to_char(e.last_name) as path
    from hr.employees e
    where e.manager_id is null
    union all
    select e1.employee_id, e1.manager_id, e1.last_name, e2.lvl + 1 as lvl, e2.path || ':' || e1.last_name as path
    from hr.employees e1
    join emp_recurse e2 on e2.employee_id = e1.manager_id
)
search depth first by last_name set last_name_order
select lpad(' ', r.lvl*2-1, ' ') || r.last_name last_name, r.path
from emp_recurse r
order by last_name_order;
```
eg: 逗号分隔的路径
```
with emp_recurse (employee_id, manager_id, last_name, lvl, path) as (
    select e.employee_id, null, e.last_name, 1 as lvl, e.last_name as path
    from hr.employees e
    where e.manager_id is null
    union all
    select e1.employee_id, e1.manager_id, e1.last_name, e2.lvl + 1 as lvl, e2.path || ',' || e1.last_name as path
    from hr.employees e1
    join emp_recurse e2 on e2.employee_id = e1.manager_id
)
search depth first by last_name set last_name_order
select lpad(' ', r.lvl*2-1, ' ') || r.last_name last_name, r.path
from emp_recurse r
order by last_name_order;
```

10.4.3 connect_by_root 运算符

connect_by_root对connect by 语法进行了强化，使得它可以返回当前行的根节点。

eg: connect_by_root
```
update hr.employees set manager_id = null where last_name = 'Kochar';
select /*+ inline gather_plan_statistics */
    level, lpad(' ',2*(level-1)) || last_name last_name, first_name, connect_by_root last_name as root,
sys_connect_by_path(last_name, ':') path
    from hr.employees
    where connect_by_root last_name = 'Kochar'
    connect by prior employee_id = manager_id
    start with manager_id is null;
```
eg: 复制connect_by_root运算符的功能
```
update hr.employees set manager_id = null where last_name = 'Kochar';
with emp_recurse(employee_id, manager_id, last_name, lvl, path) as (
    select /*+ gather_plan_statistics */
    e.employee_id, null as manager_id, e.last_name, 1 as lvl, ':' || e.last_name || ':' as path
    from hr.employees e
    where e.manager_id is null
    union all
    select e.employee_id, e.last_name, er.lvl + 1 as lvl, er.path || e.last_name || ':' as path
    from hr.employees e
    join emp_recurse er on er.employee_id = e.manager_id
    join hr.employees e2 on e2.employee_id = e.manager_id
)
search depth first by last_name set order1,
emps as (
    select lvl, last_name, path, substr(path, 2, instr(path, ':',2)-2) root
    from emp_recurse
)
select lvl, lpad(' ',2*(lvl-1)) || last_name last_name, root, path
from emps
where root = 'Kochar';
```

10.4.4 connect_by_iscycle 伪列和 nocycle 参数

eg: 通过 connect_by_iscycle 检查循环

```
select lapd(' ',2*(level-1)) || last_name last_name, first_name, employee_id, level, connect_by_iscycle
from hr.employees
start with employee_id = 100
connect by nocycle prior employee_id = manager_id;
```

eg: 在递归查询中检查循环

```
with emp (employuee_id, manager_id, last_name, first_name, lvl) as (
    select e.employee_id, null as manager_id, e.last_name, e.first_name, 1 as lvl
    from hr.employees e
    where e.employee_id = 100
    union all
    select e.employee_id, e.manager_id, e.last_name, e.first_name, emp.lvl + 1 as lvl
    from hr.employees e
    join emp on emp.employee_id = e.manager_id
)
search depth first by last_name set orders
cycle employee_id set is_cycle to '1' default '0'
select lpad(' ',2*(lvl-1)) || last_name last_name, first_name, employee_id, lvl, is_cycle
from emp
order by order1;
```

10.4.5 connect_by_isleaf 伪列

用来在层级数据中识别叶子节点。

eg:

```
select lpad(' ',2*(level-1)) || e.last_name last_name, connect_by_isleaf
from hr.employees e
start with e.manager_id is null
connect by prior e.employee_id = e.manager_id
order siblings by e.last_name;
```

eg: 在递归查询中找出叶子节点

```
with leaves as (
    select employee_id
    from hr.employees
    where employee_id not in (
        select manager_id
        from hr.employees
        where manager_id is not null
    )
),
emp(manager_id, employee_id, last_name, lvl, isleaf) as (
    select e.manager_id, e.employee_id, e.last_name, 1 as lvl, 0 as isleaf
    from hr.employees e
    where e.manager_id is null
    union all
    select e.manager_id, nvl(e.employee_id, null) employee_id, e.last_name, emp.lvl + 1 as lvl, decode(l.employee_id, null,0,1)
isleaf
    from hr.employees e
    join emp on emp.employee_id = e.manager_id
    left outer join leaves l on l.employees_id = e.employee_id
)
```

```
search depth first by last_name set orders1
select lpad(' ',2*(lvl-1)) || last_name last_name, isleaf
from emp;
```

eg: 利用 lead() 来寻找叶子节点

```
with emp(manager_id, employee_id, last_name, lvl) as (
    select e.manager_id, e.employee_id, e.last_name, 1 as lvl
    from hr.employees e
    where e.manager_id is null
    union all
    select e.manager_id, nvl(e.employee_id, null) employee_id, e.last_name, emp.lvl + 1 as lvl
    from hr.employees e
    join emp on emp.employee_id = e.manager_id
)
search depth first by last_name set last_name_order
select lpad(' ',2*(lvl-1)) || last_name last_name,
lvl, lead(lvl) over (order by last_name_order) leadlvlorder,
case when (lvl - lead(lvl) over (order by last_name_order)) < 0 then 0 else 1 end isleaf
from emp;
```

eg: 使用 breadth first 的 lead()

```
with emp(manager_id, employee_id, last_name, lvl) as (
    select e.manager_id, e.employee_id, e.last_name, 1 as lvl
    from hr.employees e
    where e.manager_id is null
    union all
    select e.manager_id, nvl(e.employee_id, null) employee_id, e.last_name, emp.lvl + 1 as lvl
    from hr.employees e
    join emp on emp.employee_id = e.manager_id
)
search breadth first by last_name set last_name_order
select lpad(' ',2*(lvl-1)) || last_name last_name, lvl,
lead(lvl) over (order by last_name_order) leadlvlorder,
case
when (lvl - lead(lvl) over (order by last_name_order)) < 0
then 0
else 1
end isleaf
from emp;
```