Power Up Your Apps with Recursive Subquery Factoring

Jared Still

2014

Pythian
love your data®

About Me

- Prefer cmdline to GUI
- Like to know how things work
- Perl aficionado
- Oak Table Member
- Oracle ACE
- Started Oracle-L
- Twitter: @PerlDBA
- jkstill@gmail.com
- Hobby: Performance Driving
- They pay me to do this?

About Pythian

- **Recognized Leader:**
  - Global industry-leader in remote database administration services and consulting for Oracle, Oracle Applications, MySQL and Microsoft SQL Server
  - Work with over 250 multinational companies such as Forbes.com, Fox Sports and Nordion to help manage their complex IT deployments
- **Expertise:**
  - Pythian's data experts are the elite in their field. We have the highest concentration of Oracle ACEs on staff—9 including 2 ACE Directors—and 2 Microsoft MVPs.
  - Pythian holds 7 Specializations under Oracle Platinum Partner program, including Oracle Exadata, Oracle GoldenGate & Oracle RAC
- **Global Reach & Scalability:**
  - Around the clock global remote support for DBA and consulting, systems administration, special projects or emergency response

What Will You Learn?

- Some performance benefits of RSF vs CONNECT BY
- How to duplicate CONNECT BY functionality with RSF

CONNECT BY Refresher

- **l_10_0a.sql** – connect by
- START WITH: ‘King’ as MGR_ID is null
- ‘level’ is depth of iteration
  - Used to provide indentation via lpad()
- Simple to do
  - Becomes difficult for more complex data

What is Recursive Subquery Factoring?

- Anchor member
- Recursive member
- Joined by UNION ALL
- Search by
  - Depth First
  - Breadth First
- **l_10_0b.sql**

Why Use Recursive Subquery Factoring?

- ANSI: Recursive Common Table Expression
- ANSI Compatible
  - Identical in SQL Server
- Will CONNECT BY be enhanced in later releases?

RSF: differences from CONNECT BY

l_10_0b.sql – includes ordering siblings ♦

```
with emp_recurse (ename,empno,mgr,deptno,lvl) as ( -- recursive query
   select e.ename, e.empno, e.mgr, e.deptno, 1 as lvl
   from scott.emp e where e.mgr is null
        -- anchor member
   union all
        -- recursive member
   select e.ename, e.empno, e.mgr,  e.deptno , empr.lvl + 1 as lvl
   from scott.emp e
   join emp_recurse empr on empr.empno = e.mgr
)
   search depth first by ename desc set order1 -- sibling order reversed
   --search breadth first by ename set order1 -- display in order of levels
select lpad(' ', lvl*2-1,' ') || er.ename ename
        , er.empno
        , er.mgr
        , er.deptno
from emp_recurse er
```

What Can RSF do

- … that CONNECT BY cannot?
- Generate test data

```
20:37:08 ora11203fs.jks.com - jkstill@js01 SQL> select max(level) from dual connect by level <=
5000000;
select max(level) from dual connect by level <= 5000000
                            *
ERROR at line 1:
ORA-30009: Not enough memory for CONNECT BY operation
```

- 65M of memory allocated

What Can RSF do cont…

- Try again with RSF
- Used same amount memory, but does not fail

```
  1  with gen (id) as (
  2     select 0 id from dual
  3     union all
  4     select gen.id + 1 as id
  5     from gen
  6     where id < 5000000
  7  )
  8* select max(id) from gen;
MAX(ID)
----------
   5000000
```

RSF is environment friendly

- Use CONNECT BY to generate test data
  - **gen_test_CB.sql** fails with ORA-30009
- **gen_test_RSF.sql** succeeds
- Monitor Mem and TEMP
  - get_spid.sql in SQL session
  - show_temp.sh
  - Also can use: watch –n 1 ps –p $PID –o pid,rss
  - Bug 17834663 - Include SQL ID for statement that created a temporary segment in GV$SORT_USAGE (Doc ID 17834663.8)
- RSF uses TEMP – CONNECT BY does not
- This is extreme usage, but useful to know for large hierarchies

oradebug dump heapdump 5

```
[ora11203fs]$ ./ha.pl
js01_ora_24891_BEFORE.trc
           free:       170,992
       freeable:       445,032
           perm:       380,576
       recreate:        98,336
          Total:     1,094,936
[trace]$ ./ha.pl

js01_ora_24967_AFTER-RSF.trc
           free:       195,952
       freeable:       498,064
           perm:       380,576
       recreate:        98,336
          Total:     1,172,928
```

```
[trace]$ ./ha.pl
js01_ora_24959_AFTER-CB.trc
           free:       195,952
       freeable:    65,420,456
           perm:       380,576
       recreate:        98,336
          Total:    66,095,320
```

What Else Can RSF Do?

- Fibonacci anyone?

```
with fibonacci (idx, fibvalue, prev_fibvalue) as (
   select
      0 as idx , 0 as fibvalue , 0 as prev_fibvalue
   from dual
   union all
   select
      f.idx + 1 as idx
      , f.fibvalue + decode(f.prev_fibvalue,0,1,f.prev_fibvalue) fibvalue
      , f.fibvalue prev_fibvalue
   from fibonacci f
   where f.idx < 42
)
select fibvalue
from fibonacci
order by idx
```

What Else Can RSF Do?

- Factorial
  - factorial.sql - Google for (ugly) CONNECT BY

```
with factorial (idx,fctr) as (
   select
      0 as idx , 1 as fctr
   from dual
   union all
   select
      f.idx + 1 as idx
      , (f.idx + 1) * f.fctr fctr
   from factorial f
   where f.idx <= 7
)
select idx,  fctr
from factorial
order by idx
```

RSF Restrictions – from the Docs

- Restrictions on Recursive Member
  - The DISTINCT keyword or a GROUP BY clause
  - The MODEL_CLAUSE
  - An aggregate function. However, analytic functions are permitted in the select list.
  - Subqueries that refer to recursive query_name.
  - Outer joins that refer to recursive query_name as the right table

RSF Differences cont.

Breadth first (default)

```
ENAME                           EMPNO        MGR     DEPTNO
------------------------- ---------- ---------- ----------
 KING                            7839                   10
   BLAKE                         7698       7839        30
   CLARK                         7782       7839        10
   JONES                         7566       7839        20
     ALLEN                       7499       7698        30
     FORD                        7902       7566        20
     JAMES                       7900       7698        30
     MARTIN                      7654       7698        30
     MILLER                      7934       7782        10
     SCOTT                       7788       7566        20
     TURNER                      7844       7698        30
     WARD                        7521       7698        30
       ADAMS                     7876       7788        20
```

RSF Differences cont.

Depth first (looks like std connect by)

```
ENAME                               EMPNO        MGR     DEPTNO
------------------------- ---------- ---------- ----------
 KING                                 7839                    10
   BLAKE                              7698       7839         30
     ALLEN                            7499       7698         30
     JAMES                            7900       7698         30
     MARTIN                           7654       7698         30
     TURNER                           7844       7698         30
     WARD                             7521       7698         30
   CLARK                              7782       7839         10
     MILLER                           7934       7782         10
   JONES                              7566       7839         20
     FORD                             7902       7566         20
       SMITH                          7369       7902         20
```

RSF: Replace the LEVEL function

- Anchor Member
  – 1 as LVL
- Recursive Member
  – LVL + 1 as LVL
- Select
  – lpad(' ', r.lvl*2-1,' ') || r.last_name

RSF: SYS_CONNECT_BY_PATH

l_10_18.sql - ♦

```
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
```

Pythian love your data®

RSF: CONNECT_BY_ROOT

- Show the ‘root’ of the connect path - ♦
  - **l_10_20.sql** demo
  - root is available at any level in the path
  - Easily duplicated with sys_connect_by_path

```
case instr(sys_connect_by_path(last_name,':'),':',-1,1)
   when 1 then last_name
   else substr(
      sys_connect_by_path(last_name,':'), 2,
      instr(sys_connect_by_path(last_name,':'),':',2)-2
end root
```

RSF: CONNECT_BY_ROOT cont.

- Anchor member
  - ':' || e.last_name || ':' as path
- Recursive member
  - er.path || e.last_name || ':' as path
- SELECT
  - substr(path,2,instr(path,':',2)-2) root
- Demo
  - **l_10_21.sql**
  - **l_10_20-fix.sql**

RSF: Cycles – deal with cycle errors

- ORA-01436: CONNECT BY loop in user data
- connect_by_iscycle – detect cycle error
- nocyle – ignore cyle error
- Demo with connect by
  - l_10_22.sql
  - l_10_23.sql

RSF: Cycles – deal with cycle errors cont.

- RSF has the CYCLE clause
  - nocycle not needed (legitimate double negative?)
  - CYCLE employee_id SET is_cycle TO '1' DEFAULT '0'
  - IS_CYCLE column created to show error row
  - Better than CONNECT BY
    - Shows the row that is the source of the error
- Demo with RSF
  - l_10_24.sql
  - l_10_22-fix.sql

RSF: CONNECT_BY_ISLEAF - end of the hierarchy

- Returns 1 when at a leaf node in hierarchy - ♦

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

```
LAST_NAME                  CONNECT_BY_ISLEAF
-------------------------- -----------------
Kochhar                                    0
  Baer                                     1
  Greenberg                                0
    Chen                                   1
    Faviet                                 1
    Popp                                   1
…
```

RSF: CONNECT_BY_ISLEAF cont.

- What good is connect_by_isleaf?
  - How about walking **UP** the hierarchy?

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

LAST_NAME                 CONNECT_BY_ISLEAF
------------------------- -----------------
Urman                                     0
  Greenberg                               0
    Kochhar                               0
      King                                1
```

RSF: CONNECT_BY_ISLEAF cont.

- Find all leaf nodes

```
select  e.last_name last_name, connect_by_isleaf
from hr.employees e
where connect_by_isleaf = 1
start with e.manager_id is null
connect by prior e.employee_id = e.manager_id
order  by e.last_name

LAST_NAME                 CONNECT_BY_ISLEAF
------------------------- -----------------
Abel                                      1
Ande                                      1
Atkinson                                  1
…
Vishney                                   1
Walsh                                     1
Whalen                                    1
```

RSF: CONNECT_BY_ISLEAF cont.

- Walk hierarchy backwards for all leaf nodes - ♦

```
with leaves as (
   select last_name last_name
   from hr.employees e
   where connect_by_isleaf = 1
   start with e.manager_id is null
   connect by prior e.employee_id = e.manager_id
   order siblings by e.last_name
)
select lpad(' ',2*(level-1)) || e.last_name last_name, connect_by_isleaf
from hr.employees e
start with e.last name in ( select last name from leaves )
connect by prior e.manager_id = e.employee_id  -- reversed the relationship
order siblings by e.last_name
```

- Demo – l_10_25a.sql

Pythian
love your data

RSF: CONNECT_BY_ISLEAF cont.

- No native functionality in RSF for isleaf
  - gotta DIY
  - The RSF SQL is a little more complex
  - See l_10_26.sql for code – won’t fit on page here
  - The RSF SQL is a little more complex, but robust
- Walk UP through the hierarchy
  - l_10_26a.sql
  - Code changes (3) on next slide

```
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.last_name = ‘Urman'
   union all
   select e.manager_id, nvl(e.employee_id,null) employee_id,  e.last_name, emp.lvl + 1 as lvl
      , decode(e.manager_id,null,1,0) isleaf
   from hr.employees e
   join emp on emp.manager_id = e.employee_id
   left outer join leaves l on l.employee_id = e.employee_id
)
search depth first by last_name set order1
select lpad(' ',2*(lvl-1)) || last_name last_name, isleaf
from emp
```

RSF: Find Leaf Node with LEAD() ?

- Works only with Depth First
  - l_10_27.sql – depth first works
  - l_10_28.sql only change is to BREADTH FIRST
    - Most nodes now show as leaves
  - (code on next slide)

RSF: Find Leaf Nodes with LEAD() ? Cont.

```
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
```

Download Presentation

- http://bit.ly/recursive-subquery-factoring

Thank you – Q&A

To contact us

- sales@pythian.com
- 1-877-PYTHIAN

To follow us

- http://www.pythian.com/blog
- http://www.facebook.com/pages/The-Pythian-Group/163902527671
- @pythian
- http://www.linkedin.com/company/pythian