Рекурсивные запросы в Oracle

Владимир Пржиялковский

Преподаватель технологий Oracle
prz@yandex.ru
http://open.oracle.tu2.ru

Июль 2010 г.

Ты то стоишь,
То начинаешь
Все сначала.
Путь учения
Не прост.

Мэйдзи, Путь

Шел Кондрат
В Ленинград,
А навстречу – двенадцать ребят.

К. Чуковский

Реферат

Рассматривается использование рекурсии в вынесенном во фразу WITH подзапросе в Oracle, разрешенное с версии 11.2. Возможности такой рекурсии сопоставляются с возможностями фразы CONNECT BY.

Введение

Рекурсивные запросы используются для обращения к иерархически связаным данным. Этого требуется не всегда: так, запросы по дереву можно свести к простому просмотру таблицы. Однако такое возможно, когда узлы дерева размечены особыми служебными значениями, а это делается рекурсивной процедурой. Если данные изменяются регулярно, рекурсивную процедуру приходится перевычислять часто, и в таких случаях идут на обычное хранение, а к рекурсии приходится прибегать в запросе.

До версии 11.2 в Oracle задача рекурсивных запросов к иерархически организованым данным решалась с помощью фразы CONNECT BY. В то же время в стандарте SQL:1999 была введена фраза WITH для вынесения подзапросов («факторизация» запроса), одна из двух вариантов которой решала задачу рекурсивныз запросов более общим (и стандартным !) образом. В Oracle первый вариант фразы WITH (простое вынесение подзапроса из основного текста) был воплощен в версии 9, а второй, рекурсивный, хотя и с некоторыми вольностями, – в версии 11.2.

Ниже для примера заводится таблица с иерархически связанными данными, далее показывается для сравнения обращение к этим данным с помощью CONNECT BY, а после приведены разные возможности употребления к этим данным рекурсивной фразы WITH.

Подготовка данных

Для дальнейших примеров создадим таблицу. Выполним:

```
CREATE TABLE route (
```

```
  node     VARCHAR2 ( 20 )
, parent   VARCHAR2 ( 20 )
, distance NUMBER   (  5 )
)
;
INSERT INTO route VALUES ( 'Ленинград', 'Москва',    696 );
INSERT INTO route VALUES ( 'Новгород',  'Москва',    538 );
INSERT INTO route VALUES ( 'Ленинград', 'Новгород',  179 );
INSERT INTO route VALUES ( 'Выборг',    'Ленинград', 135 );
COMMIT;
```

Обратите внимание, что создана не «таблица с расстояниями», а таблица с направленными маршрутами, предоставляющая расстояния в километрах между городами с точки зрения Москвы (здесь – единственная вершина иерархии). Такое представление данных и приводимые ниже запросы плохо подходят для решения более общей задачи поиска маршрута между двумя произвольными точками.

Примеры рекурсивных запросов с помощью CONNECT BY

Запрос вниз по иерархии от узла 'Москва' (присутствует только в качестве предка):

```
SQL> COLUMN way FORMAT a45
SQL> SELECT     SYS_CONNECT_BY_PATH ( node, '/' ) way
  2  FROM       route
  3  CONNECT BY PRIOR node = parent
  4  START WITH parent = 'Москва'
  5  ;

WAY
---------------------------------------------
/Ленинград
/Ленинград/Выборг
/Новгород
/Новгород/Ленинград
/Новгород/Ленинград/Выборг
```

Запрос вверх по иерархии от узла 'Выборг':

```
SQL> SELECT     SYS_CONNECT_BY_PATH ( node, '/' ) way
  2  FROM       route
  3  CONNECT BY node = PRIOR parent
  4  START WITH node = 'Выборг'
  5  ;

WAY
---------------------------------------------
/Выборг
/Выборг/Ленинград
/Выборг/Ленинград
/Выборг/Ленинград/Новгород
```

Построение рекурсивных запросов с помощью вынесения подзапроса во фразу WITH

С версии 11.2 фраза WITH может использоваться для формулирования рекурсивных запросов, в соответствии (неполном) со стандартом SQL:1999. В этом качестве она способна решать ту же задачу, что и CONNECT BY, однако (а) делает это похожим с СУБД других типов образом, (б) обладает более широкими возможностями, (в) применима не только к запросам по иерархии и (г) записывается значительно более замысловато.

Общий алгоритм вычисления фразой WITH таков:

Результат := **пусто**;
Добавок := исходный **SELECT ...**;

Пока *Добавок* **не пуст выполнять**:
Результат := *Результат*
{UNION ALL | UNION | INTERSECT | EXCEPT}
Добавок;
Добавок := рекурсивный **SELECT ... FROM** *Добавок* **…**;
конец цикла;

Предложение SELECT для исходного множества строк Oracle называет опорным (anchor) членом фразы WITH. Предложение SELECT для получения добавочного множества строк Oracle называет рекурсивным членом.

Простой пример

Простой пример употребления фразы WITH для построения рекурсивного запроса:

```
WITH
numbers ( n ) AS (
   SELECT 1 AS n FROM dual -- исходное множество -- одна строка
      UNION ALL            -- символическое «объединение» строк
   SELECT n + 1 AS n       -- рекурсия: добавок к предыдущему результату
   FROM   numbers          -- предыдущий результат в качестве источника данных
   WHERE  n < 5            -- если не ограничить, будет бесконечная рекурсия
)
SELECT n FROM numbers      -- основной запрос
;
```

Операция UNION ALL здесь используется символически, в рамках определенного контекста, для указания способа рекурсивного накопления результата.

Ответ:

```
         N
----------
         1
         2
         3
         4
         5
```

Строка с n = 1 получена из опорного запроса, а остальные строки из рекурсивного. Из примера видна оборотная сторона рекурсивных формулировок: при неаккуратном планировании они допускают «бесконечное» выполнение (на деле – пока хватит ресурсов СУБД для сеанса или же пока администратор не прервет запрос или сеанс). С фразой CONNECT BY «бесконечное» выполнение в принципе невозможно. Программист обязан отнестись к построению рекурсивного запроса ответственно.

Еще один вывод из этого примера: подобно случаю с CONNECT BY, вынесенный рекурсивный подзапрос применим вовсе необязательно только к иерархически организованным данным.

Пример с дополнительным разъяснением способа выполнения:

```
SQL> WITH
  2    anchor1234 ( n ) AS (              -- обычный
  3       SELECT 1 FROM dual UNION ALL
  4       SELECT 2 FROM dual UNION ALL
  5       SELECT 3 FROM dual UNION ALL
  6       SELECT 4 FROM dual
  7    )
  8  , numbers ( n ) AS (                 -- рекурсивный
  9       SELECT n FROM anchor1234
 10          UNION ALL
 11       SELECT n + 1 AS n
 12       FROM   numbers
```

```
 13       WHERE  n < 5
 14    )
 15  SELECT n FROM numbers
 16  ;

         N
----------
         1 ←опорный запрос
         2 ←опорный запрос
         3 ←опорный запрос
         4 ←опорный запрос
         2 ←рекурсия 1
         3 ←рекурсия 1
         4 ←рекурсия 1
         5 ←рекурсия 1
         3 ←рекурсия 2
         4 ←рекурсия 2
         5 ←рекурсия 2
         4 ←рекурсия 3
         5 ←рекурсия 3
         5 ←рекурсия 4
```

Использование предыдущих значений при выполнении рекурсии

Рекурсивные запросы с фразой WITH позволяют программисту больше, нежели запросы с CONNECT BY (тоже рекурсивные). Например, они позволяют накапливать изменения, и не испытывают необходимости в функциях LEVEL или SYS_CONNECT_BY_PATH, имея возможность легко их моделировать.

Пример запроса по маршрутам из Москвы с подсчетом километража:

```
WITH stepbystep ( node, way, distance ) AS (
  SELECT node, parent || '-' || node, distance
  FROM   route
  WHERE  parent = 'Москва'
     UNION ALL
  SELECT r.node
       , s.way || '-' || r.node
       , r.distance + s.distance
  FROM route r
       INNER JOIN
       stepbystep s
       ON ( s.node = r.parent )
  )
SELECT way, distance FROM stepbystep
/
```

Ответ:

```
WAY                                      DISTANCE
---------------------------------------- ----------
Москва-Ленинград                                696
Москва-Новгород                                 538
Москва-Новгород-Ленинград                       717
Москва-Ленинград-Выборг                         831
Москва-Новгород-Ленинград-Выборг                852
```

Запрос по маршрутам из Выборга аналогичен, но с поправкою на симметрию, вызванной движением по иерархии снизу вверх, а не сверху вниз:

```
WITH stepbystep ( parent, way, distance ) AS (
  SELECT parent, node || '-' || parent, distance
  FROM   route
```

```
  WHERE  node = 'Выборг'
     UNION ALL
  SELECT r.parent
       , s.way || '-' || r.parent
       , r.distance + s.distance
  FROM   route r
         INNER JOIN
         stepbystep s
         ON ( s.parent = r.node )
  )
SELECT way, distance FROM stepbystep
/
```

Ответ:

```
WAY                                        DISTANCE
---------------------------------------- ----------
Выборг-Ленинград                                135
Выборг-Ленинград-Москва                         831
Выборг-Ленинград-Новгород                       314
Выборг-Ленинград-Новгород-Москва                852
```

Обработка зацикленности данных

Пример организации зацикленности в сведениях о маршрутах:

```
INSERT INTO route VALUES ( 'Новгород', 'Выборг', 135 );
```

Реакция на появление цикла (уже получается не иерархия) в этом случае отлична от имевшейся для CONNECT BY и будет:

```
ERROR:
ORA-32044: cycle detected while executing recursive WITH query
```

Упражнение. Проверить это самостоятельно.

Для предупреждения зацикливания вычислений вводится специальное указание CYCLE, где следует указать перечень (в общем случае) столбцов для распознавания хождения по кругу, придумать название столбца-индикатора (он автоматически включается в конечный ответ) и задать пару символов: для обозначения незацикленной строки и для обозначения строки, где было зафиксировано повторение значений в различительных столбцах:

```
WITH stepbystep ( node, way, distance ) AS (
  SELECT node, parent || '-' || node, distance
  FROM   route
  WHERE  parent = 'Москва'
     UNION ALL
  SELECT r.node
       , s.way || '-' || r.node
       , r.distance + s.distance
  FROM route r
       INNER JOIN
       stepbystep s
       ON ( s.node = r.parent )
  )
  CYCLE node SET cyclemark TO 'X' DEFAULT '-'
SELECT way, distance, cyclemark FROM stepbystep
/
```

Ответ:

```
WAY                                          DISTANCE C
```

```
---------------------------------------- ---------- -
Москва-Ленинград                                696 -
Москва-Новгород                                 538 -
Москва-Новгород-Ленинград                       717 -
Москва-Ленинград-Выборг                         831 -
Москва-Новгород-Ленинград-Выборг                852 -
Москва-Ленинград-Выборг-Новгород                966 -
Москва-Ленинград-Выборг-Новгород-Ленинград     1145 X
Москва-Новгород-Ленинград-Выборг-Новгород       987 X
```

Упорядочение результата

Для придания порядка строкам результата в запросах с CONNECT BY используется особая конструкция ORDER BY SIBLINGS. Аналогично в вынесенном рекурсивном запросе используется специальное указание SEARCH. В рамках последнего, в частности, задается вымышленое имя столбца, в котором СУБД автоматически проставит числовые значения, и который включит автоматически в порождаемый набор столбцов, допуская в последующей обработке его использование во фразе ORDER BY для осуществления упорядочения. Пример:

```
ROLLBACK;

WITH stepbystep ( node, way, distance ) AS (
  SELECT node, parent || '-' || node, distance
  FROM   route
  WHERE  parent = 'Москва'
     UNION ALL
  SELECT r.node
       , s.way || '-' || r.node
       , r.distance + s.distance
  FROM route r
       INNER JOIN
       stepbystep s
       ON ( s.node = r.parent )
  )
  SEARCH DEPTH FIRST BY node DESC SET orderval
SELECT   way, distance, orderval
FROM     stepbystep
ORDER BY orderval DESC
/
```

Ответ:

```
WAY                                      DISTANCE   ORDERVAL
---------------------------------------- ---------- ----------
Москва-Ленинград-Выборг                         831          5
Москва-Ленинград                                696          4
Москва-Новгород-Ленинград-Выборг                852          3
Москва-Новгород-Ленинград                       717          2
Москва-Новгород                                 538          1
```

Подробности и прочие свойства построений указания SEARCH приведены в документации по Oracle.