ЛАБОРАТОРНАЯ РАБОТА №29

Тема: Табличные функции PL/SQL

Цель: Пробрести навыки создания и использования табличных функций на языке PL/SQL.

Теоретические сведения

Рассмотрим особенности табличных функций: что они собой представляют, как они работают и примеры простых табличных функций возвращающих коллекцию скалярных значений.

Что такое табличные функции?

Табличная функция – это функция, которая может быть вызвана в разделе FROM предложения SELECT. Такая функция возвращает коллекцию (обычно вложенные таблицы (nested tables) или виртуальные массивы (varrays)). Коллекция может быть с помощью оператора TABLE в набор строк и колонок, который может быть обработан средствами SQL предложения. Табличные функции удобно использовать в указанных ниже случаях:

- Объединение данных, описывающих состояние текущей сессии, с табличными данными. Пусть у нас есть данные хранящиеся в таблицах. Одновременно в текущей сессии (не в таблицах) также имеются данные, описывающих состояние этой сессии. Пусть имеется задача объединить эти два источника данных в одном SQL запросе. Эту задачу можно решить, используя оператор TABLE.
- Требуется программно построить набор данных, состоящих из множества строк и колонок. Например, требуется построить таблицу для вывода на веб странице, но структура данных в таблицах не соответствует структуре таблицы на веб-странице. Требуется создать программный код для генерации данных нужной структуры. С помощью табличной функции программист может сгенерировать данные нужной структурой.
- Создание параметризованного представления. Oracle не позволяет передавать параметры в представление. Но разработчик имеет возможность передавать параметры в функцию и использовать ее для генерации набора данных.
- Повышение эффективности выполнения параллельных запросов с помощью связанных (pipelined) табличных функций. Такого рода задачи характерны для хранилищ данных. Обычная табличная функция на является связанной. Имеется специальный синтаксис для обьявления табличной функции связанной.

- Снижение требований к объему памяти в особенности PGA (Process Global Area). Коллекции (которые строятся обычными табличными функциями) могут потреблять большой объем PGA. Однако, если табличная функция построенная как связанная (pipelined) требования к использованию PGA существенно снижаются.

Табличная функция, которая предназначена для использования в разделе FROM запроса, должна иметь следующие характеристики:

- Возвращаемый оператором RETURN тип данных. Этот тип данных должен быть коллекцией: вложенная таблица (nested table) или виртуальный массив (varray). В некоторых случаях можно использовать ассоциативные массивы. Это возвращаемый тип данных должен быть определен на уровне схемы (CREATE [OR REPLACE] TYPE) или в определении пакета (только для связанных (pipelined) функций).
- Все параметры функции должны иметь описатель IN и относиться к SQL-совместимому типу данных. (Например, нет возможности описать табличную функцию с параметром типа Boolean или записи.)
- Заключить вызов табличной функции в предложение TABLE. Заметим, что в версии ORACLE 12.1 введены новые синтаксические правила использования табличных функций. Эти новшества в основном касаются использования TABLE.

Рассмотрим пример создания и использования табличной функции, которая возвращается коллекцию строк

Создание табличных функций

Вначале следует возвращаемый тип данных. Определим тип данных – вложенная таблица. Элементами таблицы будут строки. Затем создадим табличную функцию, которая генерирует случайный набор строк. Наконец, создадим анонимный блок для демонстрации работы функции.

```
CREATE OR REPLACE TYPE strings_t
      IS TABLE OF VARCHAR2 (100);
CREATE OR REPLACE FUNCTION random_strings (count_in IN
INTEGER) RETURN strings_t AUTHID DEFINER
IS
l_strings strings_t := strings_t ();
BEGIN
   l_strings.EXTEND (count_in);
   FOR indx IN 1 .. count_in
    LOOP
       l_strings (indx) := DBMS_RANDOM.string('u', 10);
    END LOOP;
   RETURN l_strings;
END;
```

```
DECLARE l_strings strings_t := random_strings (5);
 BEGIN
    FOR indx IN 1 .. l_strings.COUNT
     LOOP
       DBMS_OUTPUT.put_line (l_strings (indx));
     END LOOP;
 END;
```

Далее продемонстрируем использование табличной функции в разделе FROM предложения SELECT.

Использование табличной функции в предложении TABLE

В разделе FROM предложения SELECT наряду с именами таблиц можно использовать следующее выражение на основе табличных функций:

```
TABLE (имя_функции(список параметров))
```

Имеется возможность и рекомендуется давать псевдонимы табличным выражениям, построенным на основе ключевого слова TABLE. Начиная с версии Oracle Database 12c, поддерживается возможность использования именованных выражений (named notation) при вызове табличной функции. Разработчик имеет возможность использования значений, которые возвращаются встроенными табличными функциями. Рассмотрим несколько примеров.

Следует отметить, что СУБД Oracle автоматически использует строку "COLUMN_VALUE" в качестве имени колонки, возвращаемой табличной функцией. Пользователю предоставлена возможность переименовать колонку на сонове использования псевдонимов.

```
SELECT rs.COLUMN_VALUE my_string
   FROM TABLE (random_strings (5)) rs

SELECT COLUMN_VALUE my_string
   FROM TABLE (random_strings (count_in => 5))

SELECT SUM (LENGTH (COLUMN_VALUE)) total_length
     , AVG (LENGTH (COLUMN_VALUE)) average_length
       FROM TABLE (random_strings (5))
/* В версии ORACLE 12.1 и выше можно не использовать
ключевое слово TABLE */

SELECT rs.COLUMN_VALUE no_table
    FROM random_strings (5) rs
```

Предложение стало существенно проще и яснее.

Использование табличных функций в качестве источников данных предложения SELECT

Теперь, когда результаты табличной функции рассматриваются как набор строк и колонок, появляется возможность использования табличных функций в качестве обычного источника данных предложения SELECT. Например, можно использовать табличную функцию в операциях соединения (join), использовать операции над множествами, такими как UNION или INTERSECT, и тому подобное. Рассмотри несколько примеров на эту тему:

```
SELECT e.last_name
       FROM TABLE (random_strings (3)) rs
          , hr.employees e
    WHERE LENGTH (e.last_name) <= LENGTH (COLUMN_VALUE)

SELECT COLUMN_VALUE last_name
       FROM TABLE (random_strings (10)) rs
       UNION ALL
SELECT e.last_name
       FROM hr.employees e
       WHERE e.department_id = 100
```

Также возможно использование табличных функций в предложении SELECT в блоках PL/SQL:

```
BEGIN
     FOR rec IN (
         SELECT COLUMN_VALUE my_string
                FROM TABLE (random_strings (5)))
     LOOP
         DBMS_OUTPUT.put_line (rec.my_string);
     END LOOP;
END;
```

Левосторонняя корреляция и табличные функции

Левосторонняя корреляция в операции соединения join происходит, когда в качестве аргумента в табличную функцию передается значение из колонки таблицы или представления. Это методика часто используется во встроенных табличных функциях для обработки XMLTABLE и JSON_TABLE.

Следует запомнить, что для передачи данных колонки в функцию функция будет вызываться для каждой строки таблицы или представления. Безусловно, это может сказаться на производительности выполнения запроса. Следующие примеры демонстрируют указанную методику.

```
CREATE TABLE things (
   thing_id NUMBER
```

```
    , thing_name VARCHAR2 (100)
)

BEGIN
    INSERT INTO things VALUES (1, 'Thing 1');
    INSERT INTO things VALUES (2, 'Thing 2');
    COMMIT;
END;

CREATE OR REPLACE TYPE numbers_t IS TABLE OF NUMBER
CREATE OR REPLACE
    FUNCTION more_numbers (id_in IN NUMBER)
      RETURN numbers_t IS
    l_numbers numbers_t := numbers_t();
    BEGIN
       l_numbers.EXTEND (id_in * 5);
       FOR indx IN 1 .. id_in * 5
       LOOP
           l_numbers (indx) := indx;
       END LOOP;
       DBMS_OUTPUT.put_line ('more numbers');
       RETURN l_numbers;
    END;

BEGIN
    FOR rec IN (
       SELECT th.thing_name
            , t.COLUMN_VALUE thing_number
             FROM things th
            , TABLE (more_numbers (th.thing_id)) t)
    LOOP
       DBMS_OUTPUT.put_line   ('more   numbers   '   ||
rec.thing_number);
    END LOOP;
END;
```

Порядок описания табличных функций

Для обеспечения возможности использования табличной функции в предложении SELECT, она должна быть описана на уровне описания схемы БД (CREATE OR REPLACE FUNCTION – как это показано в примерах выше). Нет возможности описать табличную функцию во вложенных подпрограммах.

Рассмотрим пример описания табличной функции в пакете:

```
CREATE OR REPLACE
```

```
      TYPE strings_t IS TABLE OF VARCHAR2 (100);
CREATE OR REPLACE PACKAGE tf IS
   FUNCTION strings RETURN strings_t;
END;
CREATE OR REPLACE PACKAGE BODY tf IS
   FUNCTION strings RETURN strings_t
    IS
    BEGIN
       RETURN strings_t ('abc');
    END;
END;
SELECT COLUMN_VALUE my_string FROM TABLE (tf.strings)
```

Ссылка на вложенную или приватную процедуру не может быть разрешена на SQL уровне, на котором располагается SELECT запрос. Подобная структура использования табличной функции приведет к ошибке.

```
DECLARE FUNCTION nested_strings (count_in IN INTEGER)
RETURN strings_t
   IS
   BEGIN
       RETURN strings_t ('abc');
   END;
BEGIN
   FOR rec IN (SELECT * FROM TABLE (nested_strings()))
   LOOP
      DBMS_OUTPUT.PUT_LINE (rec.COLUMN_VALUE);
   END LOOP;
END;
PLS-00231: function 'NESTED_STRINGS' may not be used in
SQL
```

Одним из нововведений ORACLE 12.1 является возможность использования предложения WITH для описания функции непосредственно в внутри предложения SELECT. Подобная функция может быть использована в качестве табличной функции (следует, однако, отметить, что такой синтаксис не поддерживается в LiveSQL; но он работает в SQL Developer, SQLcl и SQL*Plus):

```
WITH
    FUNCTION strings RETURN strings_t IS
     BEGIN
       RETURN strings_t ('abc');
     END;
SELECT COLUMN_VALUE my_string FROM TABLE (strings)
```

Допустимые типы коллекций для табличных функций

Нужно учитывать две обстоятельства касающихся типов коллекций, используемых в операторе RETURN табличной функции:

1. Тип коллекции должен быть явно объявлен. В этом случае SQL интерпретатор сможет найти ссылку на этот тип.
2. Тип коллекции (или атрибуты в этом типе) должны быть SQL-совместимыми. Например, нет возможности возвратить коллекцию Boolean из табличной функции.

Интерпретатор SQL может определить ссылку на типы данных PL/SQL программ, если эти типы определены на уровне описания схемы или в описании пакета. При этом для описания типа данных для табличной функции в описании пакет нужно использовать связанные (*pipelined)* табличные функции (они описаны выше в данной лабораторной работе). Рассмотрим следующий пример.. Функции strings_sl и strings_pl могут быть успешно вызваны в качестве табличных функций.

```
CREATE OR REPLACE TYPE
    sl_strings_t IS TABLE OF VARCHAR2 (100);
CREATE OR REPLACE PACKAGE tf IS
  TYPE strings_t IS TABLE OF VARCHAR2 (100);
  FUNCTION strings RETURN strings_t;
  FUNCTION strings_sl RETURN sl_strings_t;
  FUNCTION strings_pl RETURN strings_t PIPELINED;
END;
CREATE OR REPLACE PACKAGE BODY tf IS
  FUNCTION strings RETURN strings_t IS
   BEGIN
       RETURN strings_t ('abc');
   END;
  FUNCTION strings_sl RETURN sl_strings_t IS
   BEGIN
       RETURN sl_strings_t ('abc');
   END;
  FUNCTION strings_pl RETURN strings_t PIPELINED IS
   BEGIN
       PIPE ROW ('abc');
       RETURN;
   END;
END;
 SELECT COLUMN_VALUE my_string
        FROM TABLE (tf.strings)
 SELECT COLUMN_VALUE my_string
        FROM TABLE (tf.strings_sl)
 SELECT COLUMN_VALUE my_string
        FROM TABLE (tf.strings_pl) /
```

Но, если попытаться использовать функцию strings, получим ошибку "ORA-00902: invalid datatype". Причиной возникновения ошибки является

ссылка на функцию, которая описана в пакете и при этом не использовано ключевое слово pipelined.

```
SELECT COLUMN_VALUE my_string FROM TABLE (tf.strings) /
```

На данном этапе рассмотрено достаточное количество вопросов для построения табличных функций

Основные положения:

- Необходимо определить тип возвращаемого функцией значения оператором RETURN. Обычно таким типом является вложенная таблица или массив (varray). Иногда можно использовать в качестве возвращаемого типа ассоциативный массив. Этот тип должен быть определен на уровне описания схемы базы данных (оператор CREATE [OR REPLACE] TYPE) или в описании пакета (только для связанных (pipelined) табличных функций).
- Для всех параметров функции необходимо указать, что они являются входными (ключевое слово IN) и имеют тип совместимый типами данных SQL. (Например, нельзя вернуть коллекцию значений типа Boolean.)
- Требуется поместить вызов табличной функции в оператор TABLE. Между тем в ORACLE 12.1 и следующих появилась возможность не использовать оператор TABLE.

Порядок выполнения работы

1. Выполните примеры, приведенные в данных методических указаниях.
2. Составьте табличную функцию для БД HR, которая для каждой должности возвращает список сотрудников. Входным параметром функции является JOB_ID. Функция должна возвращать коллекцию строк FIRST_NAME||' '||LAST_NAME. Привести примеры ее использования.
3. Составьте анонимный блок, который выводит последовательность чисел от 1 до количества дне в текущем месяце. Указание – использовать цикл for … loop … end loop;

Содержание отчета.

1. Название работы
2. Цель работы
3. Листинги запросов
4. Скриншоты результатов выполнения запросов.
5. Выводы

Контрольные вопросы

1. Укажите назначение и особенности использования табличных функций.
2. Для чего используются табличные функции?
3. Какие типы данных можно использовать для коллекций возвращаемых значений табличных функций?
4. Приведите примеры использования табличных функций.
5. Для чего используется конструкция TABLE при вызове табличной функции.
6. В каком разделе оператора SELECT используются табличные функции.
7. Приведите пример использования табличных функций в операциях соединения (join).