# Лабораторная работа № 17

 Блоки. Процедуры. Курсоры. Триггеры. PL/SQL

|  |
| --- |
| Настройка SQL. Необходимо включить режим ECHO и вывести протокол лаб. работы в файл *<Фамилия студента>.txt*. Этот файл является отчетом о проделанной лаб. работе.  SQL> set echo on  SQL> spool d:\spool. txt  ……………………………………..  Завершить протокол лаб. работы (команда spool off) и направить результаты преподавателю.  SQL> spool off |

**Простые программы PL/SQL**

Самая простая программа имеет несколько объявлений и раздел выполняемого кода, состоящий из одной или нескольких команд SQL. Главная особенность – это то, что формат команды SELECT отличается от формата в SQL. После ключевого слова SELECT, должен присутствовать параметр INTO со списком переменных для каждого столбца команды SELECT, в которые должны быть помещены значения найденной строки (кортежа).

Обратите внимание, что в описании говорится о “строке (кортеже)”, а не о “строках (кортежах)”, так как команда SELECT в PL/SQL выполнится только, если результат запроса содержит одну строку (кортеж). Если запрос возвращает больше чем одну строку (кортеж), необходимо использовать курсор, как будет описано в следующей главе.

**Задание 1:**

CREATE TABLE T1(

e INTEGER,

f INTEGER

);

DELETE FROM T1;

INSERT INTO T1 VALUES(1, 3);

INSERT INTO T1 VALUES(2, 4);

/\* То, что написано выше – это просто SQL; далее - PL/SQL программа. \*/

|  |
| --- |
| Создадим файл сценария в БЛОКНОТЕ – c именем tt.sql (выполним файл сценария: SQL> @c:\tt)  DECLARE  a NUMBER;  b NUMBER;  BEGIN  SELECT e,f INTO a,b FROM T1 WHERE e>1;  INSERT INTO T1 VALUES(b,a);  END; |

Т.к. в таблице Т1 имеется только одна строка, в который первый компонент больше 1, а именно (2,4), команда INSERT вставляет (4,2) в T1.

**Процедуры**

PL/SQL очень похожи на процедуры других языков программирования. Ниже приводится пример PL/SQL процедуры addtuple1, которая получает целое число (i), как параметр, и добавляет строку (кортеж) (i, 'xxx') в таблицу Т1, которая определяется:

**Задание 2:**

CREATE TABLE T2 (

a INTEGER,

b CHAR(10)

);

|  |
| --- |
| Создадим файл сценария в БЛОКНОТЕ – c именем t1.sql (выполним файл сценария: SQL> @c:\t1)  CREATE or replace PROCEDURE addtuple1(i IN NUMBER) AS  BEGIN  INSERT INTO T2 VALUES(i, 'xxx');  END addtuple1;  В примере приведена процедура, в которой добавляется строка (кортеж) в таблицу Т2, а значения компонентов строки являются параметрами процедуры:  Создадим файл сценария в БЛОКНОТЕ – c именем t2.sql (выполним файл сценария: SQL> @c:\t2)  CREATE or replace PROCEDURE addtuple2(  x T2.a%TYPE,  y T2.b%TYPE)  AS  BEGIN  INSERT INTO T2(a, b)  VALUES(x, y);  END addtuple2; |

Процедура начинается с ключевых слов CREATE PROCEDURE, за которыми следует ее имя и список параметров. В качестве ключевого слова (описателя) вместо CREATE может использоваться OR REPLACE. Преимущество использования этого ключевого слова в том, что если процедура с каким-то именем уже определена, то новое определение с тем же именем не вызовет ошибки. С другой стороны, предыдущее определение процедуры с аналогичным именем заменится новым определением, и старая процедура перестанет существовать.

Количество параметров процедуры не ограничено. Каждый параметр сопровождается режимом и типом. В качестве режима можно задавать: IN (только чтение), OUT (только запись) и INOUT (чтение и запись). Замечание: в описании типа нельзя задавать длину. Например: CHAR(10) или VARCHAR(20) не допустимые описания, а CHAR или VARCHAR – допустимые. Количество символов (длина) будет определена при вызове процедуры по фактическому параметру (аргументу) функции.

За описанием параметров следует ключевое слово AS или IS. Это ключевое слово начинает тело процедуры, которое является блоком PL/SQL. В приведенном примере END используется с именем процедуры, хотя имя – это не обязательный параметр END. Раздел DECLARE не должен начинаться с ключевого слова DECLARE.

Например:

... AS

<объявления\_локальных\_переменных>

BEGIN

<тело\_процедуры>

END;

Команда run; является последней командой, которая создает процедуру, но выполняет ее. Для вызова (выполнения) процедуры используется команда PL/SQL, в которой процедура вызывается как выполняемая команда. Например:

|  |
| --- |
| Создадим файл сценария для исполнения процедур в БЛОКНОТЕ – c именем t0.sql (выполним файл сценария: SQL> @c:\t0)  BEGIN  addtuple1(99);  addtuple2(10, 'abc');  END; |

Ниже приведенный приме показывает применение режима OUT параметра процедуры:

CREATE TABLE T3 (

a INTEGER,

b INTEGER

);

|  |
| --- |
| Создадим файл сценария в БЛОКНОТЕ – c именем t3.sql (выполним файл сценария: SQL> @c:\t3)  CREATE PROCEDURE addtuple3(a NUMBER, b OUT NUMBER)  AS  BEGIN  b := 4;  INSERT INTO T3 VALUES(a, b);  END; |

|  |
| --- |
| Дополним файл сценария – c именем t0.sql (выполним файл сценария: SQL> @c:\t0)  DECLARE  v NUMBER;  BEGIN  addtuple1(99);  addtuple2(10, 'abc');  addtuple3(10, v);  END; |

Обратите внимание, что даже если параметры объявлены как OUT или INOUT, соответствующие им фактические параметры при вызове функции указываются. Т.к., фактический параметр для OUT или INOUT должен быть "именующим выражением" (lvalue), как в примере выше для переменной v. Константа или литеральный параметр не могут передаваться в OUT/INOUT параметры.

Можно определять функции вместо процедур. В объявлении функции, за списком параметров следует ключевое слово RETURN с типом возвращаемого значения:

CREATE FUNCTION <имя\_функции>(<список\_параметров>) RETURN <тип> AS ...

Команда RETURN <выражение>;, которая должна присутствовать в теле функции, прекращает выполнение функции и возвращает значение <выражение>.

Для поиска созданных функций и процедур, можно использовать SQL запрос:

select object\_type, object\_name

from user\_objects

where object\_type = 'PROCEDURE'

or object\_type = 'FUNCTION';

Для удаления созданных процедур или функций используется команда:

drop procedure <имя\_процедуры>;

drop function <имя\_функции>;

**Курсоры**

Курсор – это переменная, состоящая из строк некоторой таблицы или результата запроса.

Сформировав курсор, можно в программе читать и обрабатывать каждую строку курсора. Если курсор построен на основе таблицы, то можно модифицировать и удалять текущую позицию курсора.

Например:

В примере показывается применение цикла для просмотра и модификации курсора. Пусть имеется таблица Т1(e,f), строки которой являются парами целых чисел; требуется удалить из Т1 строки, первый компонент которых меньше второго и добавить в Т1 обратную строку.

1) DECLARE

/\* Переменные для хранения результата: \*/

2) a T1.e%TYPE;

3) b T1.f%TYPE;

/\* Объявление курсора: \*/

4) CURSOR T1Cursor IS

5) SELECT e, f

6) FROM T1

7) WHERE e < f

8) FOR UPDATE;

9) BEGIN

10) OPEN T1Cursor;

11) LOOP

/\* Перебираем строки запроса, сформулированного выше, \*/

/\* и заносим строки в PL/SQL переменные: \*/

12) FETCH T1Cursor INTO a, b;

/\* Если строк больше нет, то выходим из цикла: \*/

13) EXIT WHEN T1Cursor%NOTFOUND;

/\* Удаляем текущую строку: \*/

14) DELETE FROM T1 WHERE CURRENT OF T1Cursor;

/\* Вставляем обратную строку : \*/

15) INSERT INTO T1 VALUES(b, a);

16) END LOOP;

/\* Освобождаем курсор: \*/

17) CLOSE T1Cursor;

18) END;

Комментарии к программе:

* Строка (1) открывает раздел объявлений
* В строках (2) и (3) объявляются переменные a и b, которые имеют тот же тип, что и столбцы e и f таблицы T1. Хотя известно, что типы столбцов INTEGER, тип переменных PL/SQL не указывается явно, а копируется из соответствующих столбцов (сравните с предыдущим примером, где было объявлено, что соответствующие переменные будут типа NUMBER).
* В строках (4) по (8) определяется курсор T1Cursor. Он представляет собой таблицу, полученную в результате SELECT-FROM-WHERE запроса, в котором отбираются строки из Т1, чья первая компонента меньше или равна второй. Строка (8) объявляет курсор FOR UPDATE, так как предполагается модификация Т1 на основе курсора (см.строку (14)). В общем случае, FOR UPDATE не обязателен, если курсор не предполагается использовать для модификации таблицы.
* Строка (9) открывает раздел выполнимого кода программы.
* Строка (10) открывает курсор, это является обязательным.
* Строки с (11) по (16) – это команда цикла PL/SQL. Цикл размещается между LOOP и END LOOP. В цикле:
  + В строке (12) значения из строки (кортежа) курсора заносятся в локальные переменные. В общем случае, в команде FETCH должно присутствовать столько переменных, сколько компонент у строки (кортежа), отбираемой из курсора. Так как запрос, сформулированный в строках с (5) по (7), отбирает два значения, то в команде FETCH указаны две переменные соответствующего (правильного) типа.
  + В строке (13) проверяется условие выхода из цикла. Оператор % NOTFOUND, примененный к имени курсора, возвратит истину тогда, когда попытка выборки из курсора очередной строки не находит строку.
  + В строке (14) - команда SQL DELETE, которая удаляет из таблицы Т1 текущую строку (кортеж). Для удаления текущей строки в команде сформулировано условие - WHERE CURRENT OF T1Cursor.
  + В строке (15) - команда SQL INSERT, которая добавляет обратную строку в Т1.
* В строке (17) закрывается курсор.
* Строка (18) завершает PL/SQL программу.

.

**Триггеры**

Триггер – это специальная процедура языка PL/SQL. Обычная процедура начинает выполняться после ее вызова специальной командой, триггер же начинает выполняться, когда происходит определенное событие с таблицей. Событиями, которые вызывают выполнение триггера, являются операции вставки, удаления или обновления, выполняемые командами INSERT, DELETE или UPDATE. Триггеры выполняются либо до, либо после выполнения этих команд. Различают два варианта триггеров: триггеры, которые выполняются при обращении команды к строке таблицы, триггеры, которые выполняются до (или после) выполнения всей команды.

Синтаксис команды в Oracle для определения триггера (здесь представлен упрощенный формат команды, для просмотра полного формата команды воспользуйтесь командой HELP CREATE TRIGGER в sqlplus):

CREATE [OR REPLACE] TRIGGER <имя\_триггера>

{BEFORE|AFTER} {INSERT|DELETE|UPDATE} ON <имя\_таблицы>

[FOR EACH ROW [WHEN (<условие\_триггера>)]]

<тело\_триггера>

Обратите внимание на следующие замечания:

* Для таблиц можно создать только BEFORE или AFTER триггеры. (INSTEAD OF триггеры допустимы только для представлений, если необходимо выполнять модификации представлений)
* В определении триггера можно использовать ключевое слово OR для определения события. Для события UPDATE можно использовать опцию OF для определения списка столбцов таблица, при обновлении которых будет выполняться триггер.

Например:

... INSERT ON R ...

... INSERT OR DELETE OR UPDATE ON R ...

... UPDATE OF A, B OR INSERT ON R ...

* Если указан параметр FOR EACH ROW, то триггер вызывается при каждом обращении к строке, в противном случае – при выполнении команды.
* Если определен триггер для строки, то можно задать условия в опции WHEN, ограничивающие вызовы триггера. В условии не допустимы подзапросы. Если опция WHEN не указана, то триггер вызывается при каждом обращении к строке таблицы.
* <тело\_триггера> - это PL/SQL блок, а не последовательность SQL команд. В Oracle есть несколько ограничений на то, что можно разместить в <тело\_триггера>. Эти ограничения связаны с ситуациями подобными: один триггер выполняет действие, которое приводит к выполнению другого триггера, выполнение которого, в свою очередь, приводит в вызову третьего триггера и т.о. организуется бесконечный цикл вызововтриггеров. Ограничения на <тело\_триггера>:
  + Нельзя использовать команды модификации самой таблицы, т.к. эти изменения приведут к вызову данного триггера.
  + Нельзя использовать команды модификации таблиц, связанных с данной, посредством внешнего ключа.

Например:

Пусть определены 2 таблицы:

CREATE TABLE T4 (a INTEGER, b CHAR(10));

CREATE TABLE T5 (c CHAR(10), d INTEGER);

Определим триггер, который будет добавлять строку (кортеж) в Т5, если добавляется строка (кортеж) в Т4. Причем, триггер будет проверять, если в добавляемой строке первая компонента меньше или равна 10, и, только если такая компонента есть, вТ5 добавляется обратная строка (кортеж):

CREATE TRIGGER trig1

AFTER INSERT ON T4

FOR EACH ROW

WHEN (NEW.a <= 10)

BEGIN

INSERT INTO T5 VALUES(:NEW.b, :NEW.a);

END trig1;

Использование специальных имен NEW и OLD позволяет ссылаться на новые и старые строки (кортежи) таблицы. Замечание: В теле триггера перед NEW и OLD размещается символ ":", а в WHEN условии – нет!

Команда CREATE TRIGGER завершается точкой и командой run;, как обычно в PL/SQL. Команда run; завершает создание триггера, но не вызывает его выполнение. Только событие, связанное с триггером, а именно вставка строки в таблицу Т4, вызовет его выполнение.

Для просмотра информации о созданных триггерах используется команды:

select trigger\_name from user\_triggers;

select trigger\_type, table\_name, triggering\_event

from user\_triggers

where trigger\_name = '<имя\_триггера>';

Для удаления триггера используется команда:

drop trigger <имя\_триггера>;

Для определения свойства доступности или недоступности триггера используется команда:

alter trigger <имя\_триггера> {disable|enable};

**Поиск ошибок**

PL/SQL не всегда сообщает об ошибках компиляции. Вместо информации об ошибках, выдается сообщение "procedure created with compilation errors" ("процедура создана с ошибками компиляции"). Если ошибка не понятна, используйте команду для просмотра ошибок:

show errors procedure <имя\_процедуры>;

Аналогичная команда для просмотра ошибок триггера:

show errors trigger <имя\_триггера>;

Кроме того, используя команду "SHO ERR" (сокращение от "SHOW ERRORS ") можно просмотреть ошибки компиляции.