**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:

CREATE TABLE T2 (

a INTEGER,

b CHAR(10)

);

CREATE PROCEDURE addtuple1(i IN NUMBER) AS

BEGIN

INSERT INTO T2 VALUES(i, 'xxx');

END addtuple1;

.

run;

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;

.

run;

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:

BEGIN addtuple1(99); END;

.

run;

В примере приведена процедура, в которой добавляется строка (кортеж) в таблицу Т2, а значения компонентов строки являются параметрами процедуры:

CREATE PROCEDURE addtuple2(

x T2.a%TYPE,

y T2.b%TYPE)

AS

BEGIN

INSERT INTO T2(a, b)

VALUES(x, y);

END addtuple2;

.

run;

После выполнения следующего блока в таблицу Т2 будет добавлена строка (кортеж)(10, 'abc'):

BEGIN

addtuple2(10, 'abc');

END;

.

run;

Ниже приведенный приме показывает применение режима OUT параметра процедуры:

CREATE TABLE T3 (

a INTEGER,

b INTEGER

);

CREATE PROCEDURE addtuple3(a NUMBER, b OUT NUMBER)

AS

BEGIN

b := 4;

INSERT INTO T3 VALUES(a, b);

END;

.

run;

DECLARE

v NUMBER;

BEGIN

addtuple3(10, v);

END;

.

run;

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>;