Язык PL/SQL

Загадка:
Кто говорит на русском, читает на английском и знает еще не менее одного языка?

Ответ:
Программист

Корпоративные системы баз данных © М.Л. Цымблер

Содержание

- Краткая справка о PL/SQL
- Основные программные объекты

Краткая справка о PL/SQL

- *Язык PL/SQL (Programming Language/SQL)* – созданное в компании Oracle процедурное расширение языка SQL.
  - В программных объектах PL/SQL можно одновременно использовать операторы SQL и процедурные конструкции if , for, loop, goto и др.
  - Используя PL/SQL, можно определять и выполнять процедурные программные объекты: процедуры, функции, пакеты.
- Oracle Server и ряд других продуктов Oracle включают в себя *исполнитель запросов PL/SQL*.

Виды программных объектов PL/SQL

- *Неименованный блок* является частью приложения и не сохраняется в базе данных.
- *Хранимый блок* сохраняется в базе данных и может быть вызван по имени из приложения. При создании хранимого блока выполняется его компиляция и в базе данных сохраняется ее результат (псевдо-код).

Неименованный vs хранимый блок

```
declare
   t number;
   a number;
   d number;
begin
    create table statistics (
       when date,
       total number,
       alive number,
       dead number);
    select count(*) into t from s;
    select count(*) into a from sp
    where sp.sid=s.sid;
    d:=t-a;
    insert into statistics values (
       sysdate, t, a, d);
end;
```

```
create procedure Statistics(
     when in date,
     total out number,
     alive out number,
     dead out number) as
declare
     t number;
     a number;
     d number;
begin
    select count(*) into t from s;
    select count(*) into a from sp
    where sp.sid=s.sid;
    d:=t-a;
    when:=sysdate;
    total:=t;
    alive:=a;
    dead:=d;
end;
```


Структура блока PL/SQL

create procedure *имя*(*формальные параметры*)
as

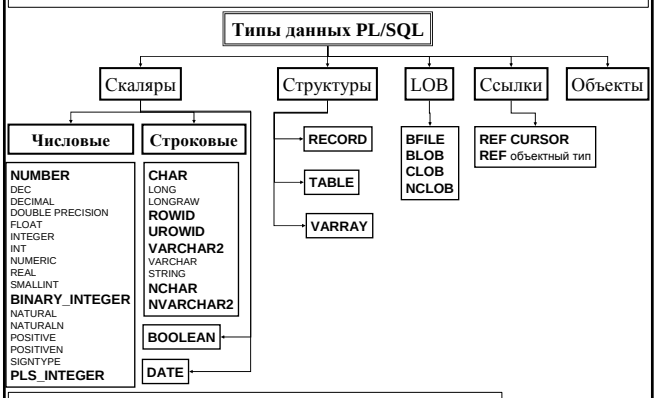
DECLARE
локальные переменные;

BEGIN
операторы, блоки

[EXCEPTION
обработка исключений]

END *имя*;

Типы данных



Корпоративные системы баз данных © М.Л. Цымблер 7

Переменные и константы

```
declare
-- Переменные
   i number;
   j number;
-- Константы
   default_rating constant number(3,2) := 15.00;
   errmsg constant char(20) := 'Ошибка!';
-- Инициализация переменных
   stop boolean := FALSE;
   str char(20) default 'N/A';
-- Ограничение целостности NOT NULL для переменных
   period number NOT NULL := 0;
-- Курсор
   cursor cur_s is select sid, name, rating from s;
-- Строка и столбец таблицы (курсора)
   s_rec cur_s%rowtype;
   s_name cur_s.name%type;
```

Корпоративные системы баз данных © М.Л. Цымблер 8

Пустой оператор NULL

```
if rating > 90 then
   compute_bonus(empno);
else
   NULL;
end if;

procedure compute_bonus(empno in number) is
begin
   NULL;
end compute_bonus;
```

Корпоративные системы баз данных © М.Л. Цымблер 9

Условные операторы

```
if условие then
   операторы;
end if;
```

```
if условие then
   операторы-1;
else
   операторы-2;
end if;
```

```
if условие-1 then
   операторы-1;
elsif условие-2 then
   операторы-2;
else
   операторы-3;
end if;
```


Операторы цикла

```
loop
   операторы;
   if условие then
      exit;
end loop;
```

```
loop
   операторы;
   exit when условие;
end loop;
```

```
while условие loop
   операторы;
end loop;
```

```
for счетчик in ниж_граница .. верх_граница loop
   операторы;
end loop;
```

```
for счетчик in reverse ниж_граница .. верх_граница loop
   операторы;
end loop;
```


Метки и операторы GOTO, EXIT

```
loop
   операторы;
   if условие then
      goto continue_label;
   операторы;
   <<continue_label>>
   NULL;
end loop;
```

```
<<outer_block>>
loop
   операторы;
  <<inner_block>>
   loop
      операторы;
      exit outer_block when условие
      exit inner_block when условие
   end loop inner_block;
   операторы;
end loop outer_block;
```

Подпрограммы

```
PROCEDURE имя [(список формальных параметров)] IS
[локальные переменные]
BEGIN
   операторы;
[EXCEPTION
   обработчики исключений]
END имя;

FUNCTION имя [(список формальных параметров)] RETURN тип IS
[локальные переменные]
BEGIN
   операторы (один из которых имеет вид RETURN выражение);
[EXCEPTION
   обработчики исключений]
END имя;
```


Режимы передачи параметров

Режим / Характеристика	IN	OUT	IN OUT
Задание	По умолчанию	Явно	Явно
Назначение	Передача параметра в подпрограмму	Возврат значения из подпрограммы	Передача начального значения в подпрограмму и возврат измененного значения
Поведение формального параметра	Как константа. Параметру не может быть присвоено значение.	Как переменная. Параметру должно быть присвоено значение.	Как переменная с начальным значением. Параметру может быть присвоено значение.
Фактический параметр	Константа, переменная с начальным значением, выражение	Переменная	Переменная
Передача параметра	По ссылке	По значению (или по ссылке, если указано NOCOPY)	По значению (или по ссылке, если указано NOCOPY)

Пример: режим IN

```
-- Внутри подпрограммы IN параметр ведет себя как константа
procedure debit (acctno IN integer, amount IN real) IS
   min_purchase constant real default 10.0;
   service_charge constant real default 0.50;
begin
   if amount < min_purchase then
      amount := amount + service_charge; -- Синтаксическая ошибка!
   end if;
   ...
end debit;
```

Пример: режим OUT

```
-- Внутри подпрограммы OUT параметр ведет себя как переменная
procedure calc_bonus (empno IN integer, bonus OUT real) IS
   hire_date date;
   bonus_missing exception;
begin
   select sal * 0.10, hiredate into bonus, hire_date from emp
      where empno = empno;
   if bonus is null then
      raise bonus_missing;
   end if;
   if months_between(sysdate, hire_date) > 36 then
       bonus := bonus + 300;
   end if;
   ...
exception
   when bonus_missing then
      ...
end calc_bonus;

calc_bonus(123, amount + service_charge); -- Синтаксическая ошибка!
```

Корпоративные системы баз данных © М.Л. Цымблер 16

Обязательная передача параметров по ссылке

- Параметры OUT и IN OUT по умолчанию передаются по значению.
- Если фактические OUT и IN OUT параметры хранят большие структуры данных (массивы, записи, экземпляры объектов), то это требует много памяти, а их копирование замедляет работу.
- С ключевым словом NOCOPY параметры OUT и IN OUT передаются по ссылке.

```
declare
   type Staff IS varray(500) OF emp%rowtype;
   procedure reorganize (my_staff IN OUT NOCOPY Staff) IS ...
```

Корпоративные системы баз данных © М.Л. Цымблер 17

Синтаксис передачи параметров

```
procedure myproc(param1 in integer, param2 in real, param3 out date) is …
…
declare
   arg1 integer;
   arg2 real;
   arg3 date;
begin
-- Позиционная нотация
   myproc(arg1, arg2, arg3);
-- Именная нотация
   myproc(param3=>arg3, param1=>arg1, param2=>arg2);
-- Комбинированная нотация: позиционная предшествует именной!
   myproc(arg1, param3=>arg3, param2=>arg2);
end;
```

Корпоративные системы баз данных © М.Л. Цымблер 18

Значения параметров по умолчанию

```
procedure myproc(param1 IN integer DEFAULT 256,
          param2 IN real DEFAULT 0.00) is …
…

begin
-- Позиционная нотация
  myproc;
  myproc(123);
-- Именная нотация
  myproc(param2=>0.05);
  myproc(param1=>123);
end;
```


Область действия и видимость переменных

```
declare
   A char;
   B real;
begin
  -- Здесь видны: A (char), B (real)
   declare
      A integer;
      C real;
   begin
      -- Здесь видны: A (integer), B (real), C (real)
   end;
   declare
      D real;
   begin
      -- Здесь видны: A (char), B (real), D (real)
   end;
    -- Здесь видны: A (char), B (real)
end;
```


Курсоры

- *Курсор (cursor)* – область разделяемого пула SGA, относящаяся к конкретному запросу SQL.
  - *Явные курсоры* определяются с помощью ключевых слов declare cursor.
  - *Неявные курсоры* создаются СУБД Oracle для всех DML-запросов.
- Интерфейс курсора:
  - **open** – открыть курсор
  - **fetch** – передвинуть указатель курсора на следующую строку
  - **close** – закрыть курсор
- Атрибуты курсора:
  - **%isopen** – возвращает True, если курсор открыт
  - **%found** – возвращает True, если последняя операция fetch успешна
  - **%notfound**– возвращает True, если последняя операция fetch не успешна
  - **%rowcount**– счетчик успешных операций fetch
  - **%type** – тип данных «столбец курсора»
  - **%rowtype**– тип данных «запись курсора»

Пример работы с курсором

```
function sum_parts() return number is
   cur cursor is
      select p.price*sp.qty as sum from p, sp
      where p.p#=sp.p#;
   rec cur%rowtype;
   sum cur.sum%type;
begin
   sum := 0;
   open cur;
   loop
      fetch cur into rec;
      exit when cur%notfound;
      sum:= sum + cur.sum;
   end loop;
   close cur;
   return sum;
end;
```

- Курсоры следует использовать, когда те же действия нельзя реализовать с помощью запроса SQL.

```
select sum(p.price*sp.qty)
into sum_parts
from p, sp where p.p#=sp.p#
```


Пример работы с курсором

```
function sum_parts(from in date, to in date, from_city in string) return number is
   cur cursor (start in sp.spdate%type,
               stop in sp.spdate%type,
               city in p.city%type) is
      select p.price*sp.qty as sum from p, sp
      where p.p#=sp.p# and
            p.city=city and sp.spdate between start and stop;
   rec cur%rowtype;
   sum cur.sum%type;
begin
   sum := 0;
   for rec in cur (from, to, from_city) loop
      sum:= sum + cur.sum;
   end loop;
   return sum;
end;
```


Пакеты

- Подпрограммы, переменные, константы и др. программные объекты могут быть объединены *пакет (package)*.
- Пакет имеет *интерфейсную секцию* (описание экспортируемых объектов) и *тело* (реализация экспортируемых объектов).

Пример пакета

```
-- Интерфейс пакета
create package dbms_pkg as
   default_rating constant number(3,2) := 15.00;
   sp_cur cursor return sp%rowtype;
function sp_qty(s# number, recalc_date date) return number;
end dbms_pkg;
-- Тело пакета
create package body dbms_pkg as
sp_cur cursor is select * from sp;
function sp_qty(sid number, recalc_date date) return number is
   …
end sp_qty;
end dbms_pkg;
```


Обработка исключений

- *Исключение (exception)* – предикат ошибки в блоке. В случае возникновении ошибки *возбуждается (raise)* исключение, при этом выполнение блока прерывается и управление передается *обработчику исключений*.
- *Системное исключение* – предикат ошибки времени выполнения; имеет предопределенное имя (ZERO_DIVIDE, NO_DATA_FOUND и др.), возбуждается автоматически.
- *Пользовательское исключение* – предикат логической ошибки; имя определяется пользователем, возбуждается оператором RAISE.

Некоторые системные исключения

Исключение	Ошибка Oracle	SQLcode	Семантика
ZERO_DIVIDE	ORA-01476	-1476	Деление на 0
NO_DATA_FOUND	ORA-01403	100	Оператор SELECT INTO не вернул ни одной строки
DUP_VAL_ON_INDEX	ORA-00001	-1	Вставка повторяющегося значения первичного ключа
VALUE_ERROR	ORA-06502	-6502	Несовпадение размеров типа переменной и присваемого значения
CURSOR_ALREADY_OPEN	ORA-06511	-6511	Открытие курсора, уже открытого ранее
INVALID_CURSOR	ORA-01001	-1001	Некорректная операция с курсором (например, закрытие ранее не открытого курсора)

Обработка исключений: пример 1

```
procedure inc_salary(empid in number, amount in number) is
    sal number;
begin
   select salary into sal from emp
   where empno=empid;
   update emp set salary=sal+amount
   where empno=empid;
exception
   when no_data_found then
      insert into audit_emp values (sysdate, empid,
                        'Нет данных о зарплате');
   when others then
      rollback;
end;
```


Обработка исключений: пример 2

```
procedure inc_salary(empid in number, amount in number) is
    sal number;
    null_salary exception;
begin
   select salary into sal from emp
   where empno=empid;
   if sal is null then
      raise null_salary;
   end if;
   update emp set salary=sal+amount
   where empno=empid;
exception
   when null_salary then
      insert into audit_emp values (sysdate, empid, 'Нет данных о зарплате');
   when others then
      rollback;
end;
```


Обработка исключений: пример 3

```
procedure inc_salary(empid in number, amount in number) is
    sal number;
begin
   select salary into sal
   from emp where empno=empid;
   if sal is null then
      raise_application_error(-20101, 'Нет данных о зарплате!');
   else
      update emp set
          salary=sal+amount where empno=empid;
   end if;
end;
```

Обработка исключений: пример 4

```
declare
   ratio number(3,1);
begin
   delete from stats
   where empno = 'xyz';
   select price / nvl(earnings, 0)
   into ratio
   from stocks where symbol = 'xyz';
   insert into stats (symbol, ratio)
     values ('xyz', ratio);
exception
   when zero_divide then ...
end;
```

```
declare
   ratio number(3,1);
begin
   delete from stats
   where symbol = 'xyz';
   begin
      select price / nvl(earnings, 0)
      into ratio
      from stocks where symbol = 'xyz';
   exception
      when zero_divide then
           ratio := -1;
   end;
   insert into stats (symbol, ratio)
     values ('xyz', ratio);
exception
   when others then ...
end;
```

Корпоративные системы баз данных © М.Л. Цымблер 31