Методы сбора, хранения, обработки и анализа данных

Лекция 9

Программные конструкции PL/SQL

На этой лекции

- Расширенные возможности применения функций
- Работа с пакетами

Применение функций

- Вызов программно-определенных функций
- Табличные функции
- Детерминированные функции

Требования для функций

- Хранимая функция
- Все параметры в режиме IN
- Типы данных – либо SQL, либо PL/SQL

Ограничения для функций

- Нельзя изменять таблицы БД (только в автономных транзакциях)
- Может изменять пакетные переменные только в SELECT, VALUES, SET

Согласованное чтение для функций

- Модель согласованного чтения – запрос видит данные, которые существуют в БД на время начала выполнения запроса
- Нарушение – при длительном запросе

```
FUNCTION total_sales (id_in IN account.account_id%TYPE)
RETURN NUMBER
    IS
  CURSOR tot_cur IS
    SELECT SUM (sales) total
    FROM orders
    WHERE account_id = id_in
    AND year = TO_NUMBER (TO_CHAR (SYSDATE, 'YYYY'));
  tot_rec tot_cur%ROWTYPE;
BEGIN
    OPEN tot_cur;
        FETCH tot_cur INTO tot_rec;
    RETURN tot_rec.total;
END;
```

```
SELECT name, total_sales (account_id)
FROM account
WHERE status = 'ACTIVE';
```

Табличные функции

- Функция возвращает таблицу
- Виды:
  - Потоковые
  - Конвейерные

Табличные функции

```oracle
create or replace function pet_family(dad_in in t_pet, mom_in in t_pet, dob_in in date)
    return tt_pet
is
    l_count pls_integer;
    rc tt_pet := tt_pet();

    procedure extend_collection (pet_in IN t_pet)
        is
    begin
        rc.extend;
        rc(rc.last) := pet_in;
    end;
  begin
    extend_collection(dad_in);
    extend_collection(mom_in);
    if dad_in.breed <> mom_in.breed
        then
        extend_collection(t_pet('himera', dad_in.breed||' '||mom_in.breed, dob_in));
    else
     case mom_in.breed
        when 'RABBIT' then l_count:= 12;
        when 'DOG' then l_count:= 4;
        when 'KANGAROO' then l_count:= 1;
     end case;
     for idx in 1 .. l_count
        loop
         extend_collection(t_pet('BABY_'||mom_in.breed||idx, mom_in.breed, dob_in));
        end loop;
     end if;
    return rc;
  end;
```

```oracle
create type t_pet is object (
    name varchar2(20),
    breed varchar2(20),
    dob date);

create type tt_pet is table of t_pet;
```

Табличные функции

```
select pets.name, pets.dob
from table(pet_family(t_pet('Johnny', 'RABBIT', sysdate-30),
                      t_pet('Lucy', 'RABBIT', sysdate-20),
                      sysdate - 1)) pets;
```

```
select pets.name, pets.dob
from table(pet_family(t_pet('Johnny', 'RABBIT', sysdate-30),
                      t_pet('Lucy', 'DOG', sysdate-20),
                      sysdate - 1)) pets;
```

```
select pets.name, pets.dob
from table(pet_family(t_pet('Johnny', 'DOG', sysdate-30),
                      t_pet('Lucy', 'DOG', sysdate-20),
                      sysdate - 1)) pets;
```

	NAME	DOB
1	Johnny	13.11.23
2	Lucy	23.11.23
3	BABY RABBIT1	12.12.23
4	BABY RABBIT2	12.12.23
5	BABY RABBIT3	12.12.23
6	BABY RABBIT4	12.12.23
7	BABY RABBIT5	12.12.23
8	BABY RABBIT6	12.12.23
9	BABY RABBIT7	12.12.23
10	BABY RABBIT8	12.12.23
11	BABY RABBIT9	12.12.23
12	BABY RABBIT10	12.12.23
13	BABY RABBIT11	12.12.23
14	BABY RABBIT12	12.12.23

	NAME	DOB
1	Johnny	13.11.23
2	Lucy	23.11.23
3	himera	12.12.23

	NAME	DOB
1	Johnny	13.11.23
2	Lucy	23.11.23
3	BABY DOG1	12.12.23
4	BABY DOG2	12.12.23
5	BABY DOG3	12.12.23
6	BABY DOG4	12.12.23

Конвейерные функции

```oracle
-------- pipeline
create type TypeTestObject as object
(
 object_name varchar2(500),
 object_id  number,
 object_type varchar2(10)
);
-- nested table
create type TypeTestList as table of TypeTestObject;

-- pipeline function
create or replace function testFunction(pObject_type in varchar2)
          return TypeTestList pipelined as
begin
  for i in (
      select tao.OBJECT_NAME, tao.OBJECT_ID, tao.OBJECT_TYPE
        from all_objects tao
       where tao.OBJECT_TYPE = pObject_type
     )
 loop
   pipe row (TypeTestObject(i.OBJECT_NAME, i.OBJECT_ID, i.OBJECT_TYPE));
  end loop;
 return;
end;
```

	OBJECT_NAME	OBJECT_ID	OBJECT_TYPE
1	DUAL	142	TABLE
2	SYSTEM PRIVILEGE MAP	447	TABLE
3	TABLE PRIVILEGE MAP	450	TABLE
4	USER PRIVILEGE MAP	453	TABLE
5	STMT AUDIT OPTION MAP	456	TABLE

	OBJECT_NAME	OBJECT_ID	OBJECT_TYPE
1	DM$EXPIMP ID SEQ	1275	SEQUENCE
2	SCHEDULER$ JOBSUFFIX S	7187	SEQUENCE
3	HS BULK SEQ	17280	SEQUENCE
4	XDB$NAMESUFF SEQ	18438	SEQUENCE
5	TMP COORD OPS	83649	SEQUENCE

```oracle
--use
select *
 from table(testFunction('TABLE')) t;

select *
 from table(testFunction('SEQUENCE')) t;
```

Детерминированные функции

```
CREATE OR REPLACE FUNCTION initials (name_in IN VARCHAR2)
    RETURN VARCHAR2 DETERMINISTIC
IS
    BEGIN
     RETURN (substr(name_in, 1, 1)||'. '||
        substr(name_in, INSTR(name_in,' ', 1, 1)+1, 1)||'.');
    END;
```

```
select name, initials(name) from salesreps;
```

	NAME	INITIALS(NAME)
1	Sam Clark	S. C.
2	Mary Jones	M. J.
3	Bob Smith	B. S.
4	Larry Fitch	L. F.
5	Bill Adams	B. A.
6	Sue Smith	S. S.
7	Dan Roberts	D. R.
8	Tom Snyder	T. S.
9	Paul Cruz	P. C.
10	Nancy Angelli	N. A.

Работа с пакетами

- Работа с данными пакета
  - Секция инициализации
  - Курсоры
  - Исключения
  - Переменные
- Повторная инициализация пакетов
- Использование пакетов

Работа с пакетами

```
-- PACKAGES
CREATE OR REPLACE PACKAGE time_pkg IS
   FUNCTION  GetTimestamp RETURN DATE;
   PROCEDURE ResetTimestamp(new_time DATE DEFAULT SYSDATE);
END time_pkg;
```

```
CREATE OR REPLACE PACKAGE BODY time_pkg IS
   StartTimeStamp DATE := SYSDATE;
   FUNCTION GetTimestamp RETURN DATE IS
   BEGIN
      RETURN StartTimeStamp;
   END GetTimestamp;
   PROCEDURE ResetTimestamp(new_time DATE DEFAULT SYSDATE)
   IS
   BEGIN
      StartTimeStamp := new_time;
   END ResetTimestamp;
   BEGIN
    dbms_output.put_line('Package is initialized');
   EXCEPTION
    WHEN NO_DATA_FOUND
         THEN dbms_output.put_line('Error inirializing');
END time_pkg;
```

Работа с пакетами

```oracle
-- package use
declare
    idx PLS_INTEGER := 1;
begin
  dbms_output.put_line(to_char(time_pkg.gettimestamp, 'dd/mm/yyyy hh24:mi:ss'));
  commit;
  while idx < 1000000
    loop
      idx := idx + 1;
    end loop;
  dbms_output.put_line(to_char(time_pkg.gettimestamp, 'dd/mm/yyyy hh24:mi:ss'));
  time_pkg.resetTimestamp;
  dbms_output.put_line(to_char(time_pkg.gettimestamp, 'dd/mm/yyyy hh24:mi:ss'));
end;
```

```
Package is initialized
13/12/2023 17:11:40
13/12/2023 17:11:40
13/12/2023 17:11:40

13/12/2023 17:11:40
13/12/2023 17:11:40
13/12/2023 17:11:55
```

Работа с пакетами

```oracle
-- PACKAGES
CREATE OR REPLACE PACKAGE time_pkg IS
   PRAGMA SERIALLY_REUSABLE;
   FUNCTION  GetTimestamp RETURN DATE;
   PROCEDURE ResetTimestamp(new_time DATE DEFAULT SYSDATE);
END time_pkg;
```

```oracle
CREATE OR REPLACE PACKAGE BODY time_pkg IS
   PRAGMA SERIALLY_REUSABLE;
   StartTimeStamp DATE := SYSTIMESTAMP;
   FUNCTION GetTimestamp RETURN DATE IS
   BEGIN
      RETURN StartTimeStamp;
   END GetTimestamp;

   PROCEDURE ResetTimestamp(new_time DATE DEFAULT SYSDATE)
   IS
   BEGIN
      StartTimeStamp := new_time;
   END ResetTimestamp;
   BEGIN
    dbms_output.put_line('Package is initialized');
   EXCEPTION
    WHEN NO_DATA_FOUND
         THEN dbms_output.put_line('Error inirializing');
END time_pkg;
```

Работа с пакетами

```oracle
-- package use
declare
    idx PLS_INTEGER := 1;
begin
  dbms_output.put_line(to_char(time_pkg.gettimestamp, 'dd/mm/yyyy hh24:mi:ss'));
  commit;
  while idx < 1000000
    loop
      idx := idx + 1;
    end loop;
  dbms_output.put_line(to_char(time_pkg.gettimestamp, 'dd/mm/yyyy hh24:mi:ss'));
  time_pkg.resetTimestamp;
  dbms_output.put_line(to_char(time_pkg.gettimestamp, 'dd/mm/yyyy hh24:mi:ss'));
end;
```

```
Package is initialized
13/12/2023 17:13:14
13/12/2023 17:13:14
13/12/2023 17:13:14

Package is initialized
13/12/2023 17:13:22
13/12/2023 17:13:22
13/12/2023 17:13:22
```

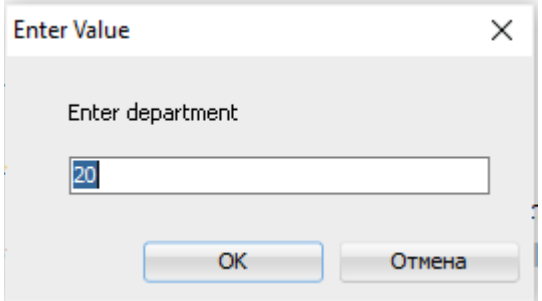
Работа с пакетами – курсоры и исключения

```
CREATE OR REPLACE PACKAGE emp_pkg
IS
CURSOR by_deptno_cur (deptno_in IN emp.deptno%TYPE)
IS SELECT * FROM emp WHERE deptno = deptno_in;

CURSOR by_sal_cur (lowsal_in IN emp.sal%TYPE,
                   hisal_in IN emp.sal%TYPE)
RETURN emp%rowtype;
no_such_dept EXCEPTION;
END emp_pkg;
```

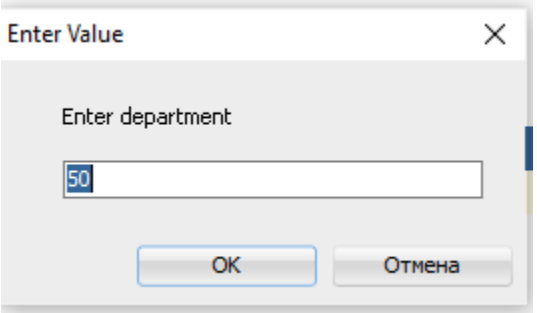
```
CREATE OR REPLACE PACKAGE BODY emp_pkg
IS
CURSOR by_sal_cur (lowsal_in IN emp.sal%TYPE,
                   hisal_in IN emp.sal%TYPE)
RETURN emp%ROWTYPE
IS SELECT * FROM emp
    WHERE sal between lowsal_in and hisal_in;

END emp_pkg;
```

Работа с пакетами – исключения

```
ACCEPT p_deptno PROMPT 'Enter department'
declare
v_deptno emp.deptno%type := &p_deptno;
v_emp_count pls_integer;
begin
    select count(*) into v_emp_count
    from emp
    where deptno = v_deptno;
    if v_emp_count = 0 then raise emp_pkg.no_such_dept;
        else dbms_output.put_line(' There are '||
                    v_emp_count||' emps in dept');
    end if;
exception
    when emp_pkg.no_such_dept
        then dbms_output.put_line('No such department');
end;
```




```
There are 5 emps in dept
```




```
No such department
```

Работа с пакетами – исключения

```oracle
CREATE OR REPLACE PACKAGE emp_pkg
IS
CURSOR by_deptno_cur (deptno_in IN emp.deptno%TYPE)
IS SELECT * FROM emp WHERE deptno = deptno_in;

CURSOR by_sal_cur (lowsal_in IN emp.sal%TYPE,
                   hisal_in IN emp.sal%TYPE)
RETURN emp%rowtype;
no_such_dept EXCEPTION;
pragma EXCEPTION_INIT(no_such_dept, 100);
END emp_pkg;
```

```oracle
-- pragma
declare
v_deptno emp.deptno%type :=50;
v_dept dept%rowtype;
begin
    select * into v_dept
    from dept
    where deptno = v_deptno;
    dbms_output.put_line(v_dept.dname ||' '||v_dept.loc);
exception
    when emp_pkg.no_such_dept
        then dbms_output.put_line('No such department');
end;
```

```
No such department
```

Работа с пакетами – курсоры

```
-- use package cursor
begin
    for rec in emp_pkg.by_sal_cur(2000,4000)
    loop
        dbms_output.put_line(rec.ename ||' '||rec.sal);
    end loop;
end;
```

```
BLAKE 2850
CLARK 2450
JONES 2975
FORD 3000
SCOTT 3000
```

```
begin
    for rec in emp_pkg.by_deptno_cur(20)
    loop
        dbms_output.put_line(rec.ename ||' '||rec.sal||' '||rec.deptno);
    end loop;
end;
```

```
JONES 2975 20
FORD 3000 20
SMITH 800 20
SCOTT 3000 20
ADAMS 1100 20
```

Работа с пакетами – курсоры

```
-- already open
declare
 rec emp_pkg.by_sal_cur%rowtype;
begin
    open emp_pkg.by_sal_cur(2000,4000);
    fetch emp_pkg.by_sal_cur into rec;
    dbms_output.put_line(rec.ename ||' '||rec.sal);
end;
```

```
BLAKE 2850
```

```
Error report -
ORA-06511: PL/SQL: курсор уже открыт
ORA-06512: на  "TEST_USER.EMP_PKG", line 6
ORA-06512: на  line 4
06511. 00000 -  "PL/SQL: cursor already open"
*Cause:    An attempt was made to open a cursor that was already open.
*Action:   Close cursor first before reopening.
```

Курсор вне пакета

```
--simple cursor - automatically closed
declare
 rec emp%rowtype;
 cursor emp_curs is select * from emp;
begin
    open emp_curs;
    fetch emp_curs into rec;
    dbms_output.put_line(rec.ename ||' '||rec.sal);
end;
```

```
KING 400

KING 400
```

Вопросы?