

## 1. Процедуры и функции

### 1.1. Создание процедур и функций

```
CREATE [OR REPLACE] PROCEDURE имя_процедуры
  [ (параметр [{IN | OUT | IN OUT}] тип,
    ...
    (параметр [{IN | OUT | IN OUT}] тип)) {IS | AS}
  /* раздел объявлений */
BEGIN
  /* выполняемый раздел */
EXCEPTION
  /* раздел исключительных ситуаций */
END [имя_процедуры];
```

```
CREATE [OR REPLACE] FUNCTION имя_функции
  [ (параметр [{IN | OUT | IN OUT}] тип,
    ...
    (параметр [{IN | OUT | IN OUT}] тип))
  RETURN возвращаемый_тип {IS | AS}
  /* раздел объявлений */
BEGIN
  /* выполняемый раздел */
EXCEPTION
  /* раздел исключительных ситуаций */
END [имя_функции];
```

### 1.2. Вызов процедур и функций

```
DECLARE
  nRtg NUMBER (4) := 200;
  v1   NUMBER;
BEGIN
  v1 := Func_1(nRtg);
  Proc2; --если параметров нет, скобки можно опустить
END;
```

### 1.3. Удаление процедур и функций

```
DROP PROCEDURE имя_процедуры
DROP FUNCTION имя_функции
```

## 1.4. Параметры подпрограмм

Вид	Описание
IN	Значение фактического параметра передается в процедуру при ее вызове. В процедуре формальный параметр рассматривается как константа. После завершения процедуры фактический параметр не изменяется.
OUT	Любое значение, которое имеет фактический параметр при вызове процедуры, игнорируется. В процедуре формальный параметр рассматривается как неиницированная переменная. Когда процедура завершается, содержимое формального параметра присваивается фактическому параметру (в Oracle 8i этот режим можно изменить с помощью модификатора NOCOPY, позволяющего применять параметр по ссылке).
IN OUT	Комбинацию видов IN и OUT. Значение фактического параметра передается в процедуру при ее вызове. В процедуре формальный параметр рассматривается в качестве инициализированной переменной, и можно как записать в него значение, так и считать значение из него. После завершения процедуры содержимое формального параметра присваивается фактическому параметру (в Oracle 8i этот режим можно изменить с помощью модификатора NOCOPY, как и для параметра OUT).

При описании процедур запрещается указывать длину параметров типа CHAR и VARCHAR2, а также точность и/или масштаб параметров типа NUMBER

Только использование атрибута %TYPE накладывает ограничение на формальный параметр

```
CREATE OR REPLACE PROCEDURE myProc
  (p_Par1 IN OUT cust.rating%TYPE,
   p_Par2 IN VARCHAR2 DEFAULT NULL) IS
BEGIN
  p_Par1 := 250;
END myProc;
```

```
CREATE OR REPLACE PROCEDURE Proc_1 IS
  nRtg NUMBER (4) := 120;
  vVal VARCHAR(20) := 'Tokyo';
BEGIN
  MyProc(nRtg, vVal);
  MyProc(p_Par2 => vVal, p_Par1 => nRtg);
END myProc;
```

## 2. Пакеты PL/SQL

### 2.1. Спецификация и тело пакета

#### 2.1.1. Спецификация пакета

```
CREATE OR REPLACE PACKAGE lib IS
    TYPE sal_type_rec IS sal%ROWTYPE;
    MAXDATE CONSTANT DATE := to_date('31.12.4712 AD','dd.mm.yyyy AD');
    FUNCTION bool_to_char(p_Bool IN BOOLEAN) RETURN VARCHAR2;
    PROCEDURE debug_on;
    PROCEDURE debug_off;
END lib;
/
```

#### 2.1.2. Тело пакета

```
CREATE OR REPLACE PACKAGE BODY lib IS
    debug_flag BOOLEAN;
    sal_rec sal_type_rec;

    FUNCTIO bool_to_char(p_Bool IN BOOLEAN) RETURN VARCHAR2 IS
        Str VARCHAR2(5);
    BEGIN
        IF (p_Bool) THEN str := 'TRUE';
        ELSIF (NOT p_Bool) THEN str := 'FALSE';
        ELSE str := 'NULL';
        END IF;
        RETURN (str);
    END bool_to_char;

    PROCEDURE debug_on IS    -- включить отладку
    BEGIN
        Debug_flag := TRUE;
    END debug_on;

    PROCEDURE debug_off IS  -- выключить отладку
    BEGIN
        Debug_flag := FALSE;
    END debug_off;

    BEGIN
        Debug_flag := FALSE;
    END lib;
/
```

### 2.2. Общедоступные и закрытые объявления

- Общедоступные объекты – объявленные в спецификации пакета
- Закрытые объекты – объявленные только внутри тела пакета

## 2.3. Ссылки на элементы пакета

**<имя\_схемы>.<имя\_пакета>.<имя\_объекта>**

```
DECLARE
  Str VARCHAR2(10);
  vBool BOOLEAN DEFAULT TRUE;
BEGIN
  str := study.lib.bool_to_char(vBool);
  DBMS_OUTPUT.put_line('str is '||str);
END;
/
```

## 2.4. Стандартные пакеты Oracle

- DBMS\_ – пакеты для работы с базой данных
- UTL\_ – утилиты общего назначения.

```
SELECT object_name, object_type, status FROM dba_objects
WHERE owner='SYS' AND object_type LIKE 'PACKAGE%'
ORDER BY object_name, object_type;
```

```
SELECT table_name, grantee FROM all_tab_privs
WHERE grantor='SYS' AND privilege='EXECUTE'
ORDER BY table_name;
```

### DBMS\_OUTPUT в утилите SQL\*Plus

```
SET SERVEROUTPUT ON
BEGIN
  DBMS_OUTPUT.enable;
  DBMS_OUTPUT.put_line('Hello, World!');
END;
/
```

Размер буфера должен быть в диапазоне от 2 000 до 1 000 000 байт

```
DBMS_OUTPUT.enable(100000);
```

### 3. Триггеры базы данных

#### 3.1. Триггеры DML

```

CREATE TABLE sal_stats (
    snum      NUMBER(4),
    total_ords NUMBER(4),
    total_amt  NUMBER(7,2));

CREATE OR REPLACE TRIGGER Update_Sal_Stats
    AFTER INSERT OR DELETE OR UPDATE ON Ord
DECLARE
    CURSOR c_stat IS
        SELECT snum Id, COUNT(*) cnt, SUM(amt) summa FROM Ord
        GROUP BY snum;
BEGIN
    DELETE FROM sal_stats;
    FOR v_stat IN c_stat LOOP
        INSERT INTO sal_stats(snum, total_ords, total_amt)
            VALUES(v_stat.Id, v_stat.cnt, v_stat.summa);
    END LOOP;
END Update_Sal_Stats;

```

#### 3.2. Триггеры INSTEAD OF

```

CREATE OR REPLACE VIEW Sal_Cust AS
    SELECT sname, cnum, cname, city, rating FROM sal s, cust c
    WHERE s.snum=c.snum;

```

Нельзя:

```

INSERT INTO Sal_Cust (sname, cnum, cname, city, rating)
    VALUES ('Peel',2010,'Mary','London',100);

```

```

CREATE OR REPLACE TRIGGER Sal_Cust_Insert
    INSTEAD OF INSERT ON Sal_Cust
DECLARE
    v_snum sal.snum%TYPE;
BEGIN
    SELECT snum INTO v_snum FROM sal WHERE sname = :new.sname;
    INSERT INTO Cust (cnum, cname, city, rating, snum)
        VALUES (2010,'Mary','London',100, v_snum);
END Sal_Cust_Insert;

```

### 3.3. Системные триггеры

```
CREATE TABLE ddl_cre (
  user_id    NUMBER,
  obj_type   VARCHAR2(20),
  obj_name   VARCHAR2(20),
  cre_date   DATE);

CREATE OR REPLACE TRIGGER Cre_Log
  AFTER CREATE ON SCHEMA
BEGIN
  INSERT INTO ddl_cre (user_id, obj_type, obj_name, cre_date)
    VALUES (USER, DICTIONARY_OBJ_TYPE,
             DICTIONARY_OBJ_NAME, SYSDATE);
END Cre_Log;
```

### 3.4. Создание триггеров

```
CREATE [OR REPLACE] TRIGGER имя_триггера
  {BEFORE | AFTER | INSTEAD OF} активизирующее_событие
  [конструкция_REFERENCING]
  [WHEN условие_срабатывания]
  [FOR EACH ROW]
  тело_триггера;
```

Внутри строкового триггера можно обращаться к строке – :old и :new.

*активизирующая\_таблица*%%ROWTYPE;

:new.поле

Таблица 1. :old и :new

Активизирующий оператор	:old	:new
INSERT	Не определена – во всех полях содержатся NULL-значения	Значения, которые будут введены после выполнения оператора
UPDATE	Исходные значения, содержащиеся в строке перед обновлением данных	Новые значения, которые будут введены после выполнения оператора
DELETE	Исходные значения, содержащиеся в строке перед ее удалением	Не определена – во всех полях содержатся NULL-значения

```

CREATE OR REPLACE TRIGGER Gen_Sal_Snum
  BEFORE INSERT ON Sal
  FOR EACH ROW
BEGIN
  SELECT SQ_Sal.NEXTVAL INTO :new.snum FROM dual;
  :new.snum := SQ_Sal.NEXTVAL; --c Oracle 11g
END Gen_Sal_Snum;

INSERT INTO Sal (sname, city, comm) VALUES ('Tonny', 'Berlin', 0.12);

```

REFERENCING [OLD AS *имя\_old*] [NEW AS *имя\_new*];

```

CREATE OR REPLACE TRIGGER Gen_Sal_Snum
  BEFORE UPDATE OF comm ON Sal
  FOR EACH ROW
  WHEN (old.comm. > 0.12)
BEGIN
  /* Тело триггера */
END Gen_Sal_Snum;

```

Функции – INSERTIG, UPDATING и DELETING

```

CREATE OR REPLACE TRIGGER Log_Change
  BEFORE INSERT OR UPDATE OR DELETE ON Sal
  FOR EACH ROW
DECLARE
  v_Ch_Type VARCHAR2(1);
BEGIN
  IF INSERTING THEN
    v_Ch_Type := 'I';
  ELSIF UPDATING THEN
    v_Ch_Type := 'U';
  ELSE
    v_Ch_Type := 'D';
  END IF;
  INSERT INTO Audit (Ch_User_Name, Ch_Type, Ch_Date)
    VALUES(USER, v_Ch_Type, SYSDATE);
END Log_Change;

```

- В триггере нельзя задавать операторы управления транзакциями: COMMIT, ROLLBACK, SAVEPOINT или SET TRANSACTION
- В процедурах и функциях, которые вызываются в теле триггера, тоже нельзя задавать никаких операторов управления транзакциями

### 3.5. Удаление и запрещение триггеров

**DROP TRIGGER *имя\_триггера*;**

**ALTER TRIGGER *имя\_триггера* {DISABLE | ENABLE};**