# Язык PL/SQL

| Версия Oracle | Версия PL/SQL |
|---|---|
| Oracle6 | 1.0 |
| Oracle7 | 2.0 |
| Oracle7.1 | 2.1 |
| Oracle7.3 | 2.3 |
| Oracle8 | 8.0 |
| Oracle8i | 8.1 |
| Oracle9i Release 1 | 9.0 |
| Oracle9i Release 2 | 9.2 |
| Oracle10g Release 1 | 10.0 |

```
SELECT * FROM sys.v_$version;
```

## Oracle8i

- Автономные транзакции
- Процедуры с правами вызывающего
- Динамический встроенный SQL (Native Dynamic SQL, NDS)
- Пакетные операции
- Новые возможности триггеров
- Вызов Java из PL/SQL

## Oracle9i

- Для SQL и PL/SQL один и тот же синтаксический анализатор
- Команды DML с ориентацией на записи
- Индексация коллекций не только целыми числами, но и строками

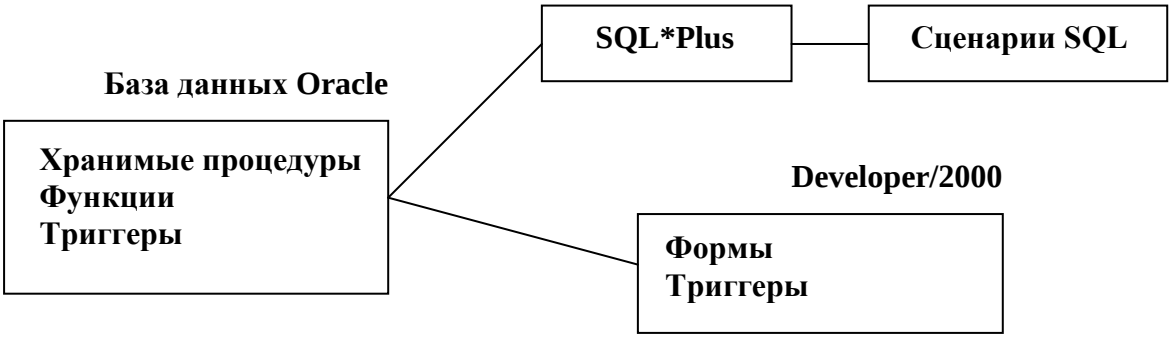
## Oracle10g

- Новые типы данных – BINARY_FLOAT и BINARY_DOUBLE
- Пакет DBMS_LOB поддерживает LOB от 8 до 128 терабайт
- Настройка строк литералов – конструкция q'!...!'

## Oracle11g

http://www.oraclebi.ru/files/presentations/imelnikov/plsql11g.pdf

## 1. Области применения PL/SQL



## 1.1. Файлы сценариев SQL

```
DECLARE
        …
        (объявление переменных)
BEGIN
        (команды SQL и PL/SQL)
[EXCEPTION
         (команды SQL и PL/SQL)]
END;
```

## 1.2. Хранимые процедуры

```
CREATE OR REPLACE PROCEDURE myproc
        (custid IN NUMBER, term IN NUMBER DEFAULT 30)
AS
        (объявления переменных)
BEGIN
        (операторы SQL и PL/SQL)
END;
```

## 1.3. Функции

```
Y = SIN(x) + 2;
```

```
CREATE OR REPLACE FUNCTION x_to_y
        (xin IN NUMBER ) RETURN NUMBER
IS
        (объявления переменных)
BEGIN
        (операторы SQL и PL/SQL)
        RETURN (выражение);
END;
```

# 1.4. Пакеты

**CREATE OR REPLACE PACKAGE *имя_пакета* IS**
**(*определение общедоступных переменных, процедур и функций*)**
**END [*имя_пакета*];**

**CREATE OR REPLACE PACKAGE BODY *имя_пакета* IS**
**(*определение локальных переменных, процедур и функций*)**
**(*реализация процедур и функций*)**
**[BEGIN**
**(*инициализация*)]**
**END [*имя_пакета*];**

```
CREATE OR REPLACE PACKAGE score_all
AS
    PROCEDURE score_answers
        (custid IN NUMBER, term IN NUMBER DEFAULT 30);
END;

CREATE OR REPLACE PAKAGE BODY score_all
AS
    PROCEDURE score_answers
        (custid IN NUMBER, term IN NUMBER DEFAULT 30)
    IS
        (определение переменных)
    BEGIN
        (операторы SQL и PL/SQL)
    END;

    PROCEDURE letter_grade
    IS
        (объявления переменных)
    BEGIN
        (операторы SQL и PL/SQL)
    END;
END;
```

## 1.5. Триггеры

```
CREATE OR REPLACE TRIGGER populate_total
    AFTER UPDATE OF hole_sccore ON golf_score
    FOR EACH ROW
DECLARE
BEGIN
        (операторы SQL и PL/SQL)
END;
```

## 2. Синтаксис PL/SQL

- команды и операторы заканчиваются точкой с запятой:

  ```
  x := 2 * y + 1;
  ```

- после операторов PL/SQL, которые стоят в начале программного блока (IF или BEGIN) точка с запятой не ставится

  ```
  IF y = 7 THEN
  …
  END IF;
  ```

- текстовые значения должны быть заключены в одинарные (а не двойные) кавычки

  ```
  IF last_name = 'SMITH'  THEN …
  ```

- допускается выход только в конце процедуры. Оператор EXIT позволяет выйти из цикла, но не позволяет осуществить выход из процедуры в середине программы.

## 3. Структура блока

```
DECLARE
      (список объявлений переменных)
BEGIN
      (список команд SQL и PL/SQL)
EXCEPTION
      (обработчики исключений)
END;
```

## 4. Передача значений в процедуры и из процедур

```
Test_proc(x, y, output_v);
```

```
CREATE OR REPLACE PROCEDURE test_proc
(xin IN NUMBER, yin IN CHAR, return_code OUT NUMBER)
…
AS
…
```

## 5. Переменные в PL/SQL

- CHAR
- VARCHAR2
- NUMBER
- DATE
- BINARY_INTEGER
- BOOLEAN
- %TYPE
- %ROWTYPE

```
last_name(1) := 'Smith';
last_name(2) := 'Jones';
last_name(3) := 'Davis';
```

```
TYPE first_name_list IS TABLE OF VARHAR2(20)
INDEX BY binary_integer;
first_name   first_name_list;
```

```
TYPE last_name_list IS VARRAY(15) OF VARHAR2(20);
last_name   last_name_list;
```

```
CREATE OR REPLACE PACKAGE loan_calc
AS
prime_rate NUMBER(4,2);
…
PROCEDURE …
…
```

# 6. Операторы языка. Управление ходом выполнения программ

## 6.1. Составление выражений

- **Оператор присваивания (:=).**
  **y := 6 * x + 2;**
- **Арифметические операции и оператор конкатенации (||).**
  **y := 3 * x + 4/2 - 8;**
  **full_name := first_name || ' ' || last_name;**
- **Функции, встроенные и разрабатываемые пользователем.**
  **y := SIN(x);**

## 6.2. Конструкция IF … THEN … ELSE …

```
I.  IF условие THEN
        (операторы SQL или PL/SQL)
    END IF;
```

```
II.  IF условие THEN
        (операторы SQL или PL/SQL)
     ELSIF условие THEN
        (операторы SQL или PL/SQL)
     ELSE
        (операторы SQL или PL/SQL)
     END IF;
```

## 6.2. Оператор выбора CASE

```
CASE выражение
WHEN значение1 THEN действия;
WHEN значение2 THEN действия;
…
ELSE действия;
END CASE;
```

```
DECLARE
        N number;
BEGIN
        N := Last_day(sysdate);
        CASE N
        WHEN 28 THEN
        update my_tab set priznak=1;
```

```
        WHEN 29 THEN
        update my_tab set priznak=1;
        WHEN 30 THEN
        update my_tab set priznak=2;
        ELSE
        update my_tab set priznak=3;
        END CASE;
END;
/
```

## 6.3. Цикл LOOP

```
BEGIN
        cnt := 0;
        LOOP
                INSERT INTO test VALUES(2 * cnt);
                cnt := cnt +1;
                IF cnt >= 8 THEN  EXIT;  END IF;
        END LOOP;
END;
```

## 6.4. Цикл WHILE

```
BEGIN
        cnt := 0;
        WHILE cnt < 8  LOOP
                INSERT INTO test VALUES(2 * cnt);
                cnt := cnt +1;
        END LOOP;
END;
```

## 6.5. Цикл FOR

```
BEGIN
        cnt := 0;
        FOR cnt IN  [REVERSE]  1 .. 8  LOOP
                INSERT INTO test VALUES(2 * cnt);
        END LOOP;
END;
```

## 6.6. Оператор перехода GOTO

```
GOTO имя_метки;
<<имя_метки>>


BEGIN
        DBMS_OUTPUT.PUT_LINE('Начало блока');
        GOTO m_last;

        DBMS_OUTPUT.PUT_LINE('Пропускаемый оператор');
        RETURN ;

        <<m_last>>
        DBMS_OUTPUT.PUT_LINE('Конец блока');
END;
/
```

## 6.7. Цикл CURSOR FOR

```
DECLARE
        CURSOR cursor_name IS …
BEGIN
        FOR for_name IN cursor_name LOOP
                (команды SQL и PL/SQL)
        END LOOP;
END;
```

# 7. Оператор FORALL

**FORALL** ***индекс_записи*** **IN** ***начальное значение .. конечное значение***
***SQL-инструкция***

## Атрибуты неявного курсора для оператора FORALL

| | |
|---|---|
| **SQL%FOUND** | **TRUE, если последняя DML-инструкция модифицировала хотя бы одну строку** |
| **SQL%NOTFOUND** | **TRUE, если последняя DML -инструкция не модифицировала ни одной строки** |
| **SQL%ROWCOUNT** | **Количество строк, модифицированных всеми DML-инструкциями** |
| **SQL%ISOPEN** | **Всегда FALSE. Пользоваться атрибутом не следует** |
| **SQL%BULK_ROWCOUNT** | **Возвращает коллекцию с информацией о количестве строк, обработанных каждой DML-инструкцией, выполненной в операторе FORALL** |

```
CREATE OR REPLACE PROCEDURE add_sal (
            sal_num IN num_arr,
            sal_comm IN comm_arr)    IS
BEGIN

    FORALL idx  IN   sal_num.FIRST .. sal_num.LAST
        UPDATE sal  SET  comm = sal_comm(idx)
            WHERE snum = sal_num(idx);
END;
```

# 8. Предложение BULK COLLECT

**. . . BULK COLLECT INTO *имя_коллекции* [, *имя_коллекции*] . . .**

**Функционально эквивалентные программные блоки:**

```
DECLARE
    CURSOR c_sal IS
      SELECT * FROM sal;
    TYPE t_arr_sal IS TABLE  OF c_sal%ROWTYPE
      INDEX BY binary_integer;
    arr_sal  t_ arr _sal;
BEGIN
    FOR v_sal  IN  c_sal  LOOP
        arr_sal(c_sal%ROWCOUNT).snum  :=  v_sal.snum;
        arr_sal(c_sal%ROWCOUNT).sname  :=  v_sal.sname;
        arr_sal(c_sal%ROWCOUNT).city  :=  v_sal.city;
        arr_sal(c_sal%ROWCOUNT).comm  :=  v_sal.comm;
    END LOOP;
    -- работа с массивом arr_sal
END;
```

```
DECLARE
    CURSOR c_sal IS
      SELECT * FROM sal;
    TYPE t_arr_sal IS TABLE  OF c_sal%ROWTYPE
      INDEX BY binary_integer;
    arr_sal  t_ arr _sal;
BEGIN
    OPEN  c_sal;
    FETCH  c_sal BULK COLLECT INTO arr_sal;
    CLOSE  c_sal;
    -- работа с массивом arr_sal
END;
```

```
DECLARE
    TYPE t_arr_sal IS TABLE  OF sal%ROWTYPE
    INDEX BY binary_integer;
    arr_sal  t_ arr _sal;
BEGIN
    SELECT * BULK COLLECT INTO arr_sal FROM sal;
    -- работа с массивом arr_sal
END;
```