

Темы

- Использование модульного дизайна в разработке
- Работа с процедурами:
 - Создание и вызов процедур
 - Параметры процедур и их режим
 - Формальные и фактические параметры
 - Запись параметров по имени и порядку

Модульная разработка с PL/SQL

- PL/SQL имеет блочную структуру. Блоки PL/SQL можно организовать в:
 - Анонимные блоки
 - Процедуры и функции
 - Пакеты
 - Триггеры
- Преимущество модульного подхода в разработке:
 - Проще сопровождение
 - Лучше безопасность и целостность
 - Лучше читается код

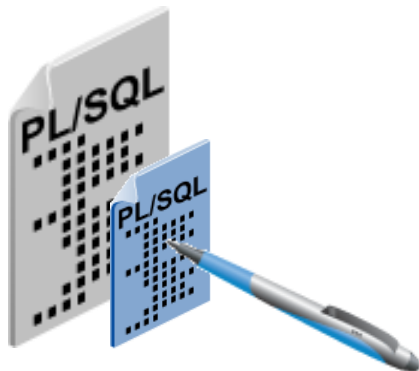
Анонимные блоки

Анонимные блоки:

- Минимальная структурная единица PL/SQL
- Вызов кода PL/SQL из приложений
- Могут быть вложенными в другой исполняемой секции PL/SQL

```
[DECLARE      -- Секция объявления
  variable declarations; ... ]
BEGIN        -- Исполняемая секция
  SQL or PL/SQL statements;
[EXCEPTION   -- Секция обработки исключений
  WHEN exception THEN statements; ]
END;         -- Конец блока
```

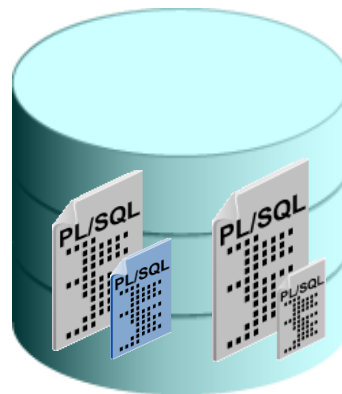
Преимущества использования подпрограмм PL/SQL



**Проще
сопровождать**



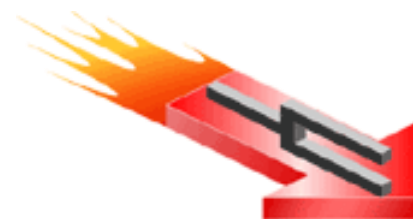
Проще читать код



**Подпрограммы:
Хранимые
процедуры и
функции**



**Лучше безопасность
и целостность
данных**



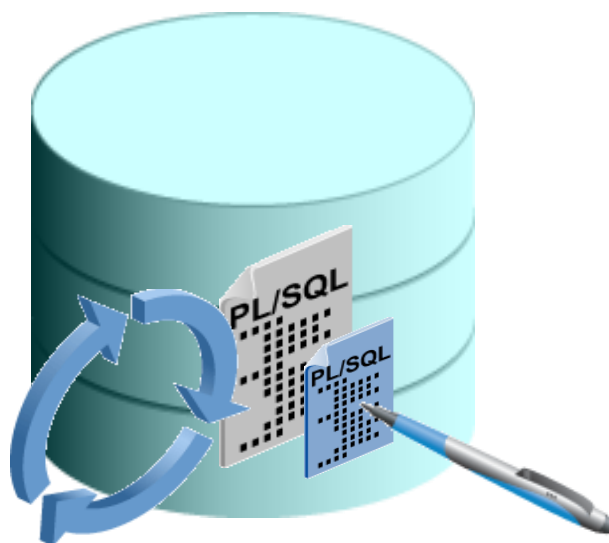
Производительность

Различия анонимных блоков и подпрограмм

Анонимный блок	Подпрограммы
Нет названия	Есть название
Компиляция каждый раз при запуске	Компиляция при создании
Не хранится в БД	Хранится в БД
Нельзя вызывать из других приложений	Можно обращаться из других приложений
Не возвращают значение	Функции должны возвращать значение
Нет параметров	Могут принимать параметры

Что такое процедуры?

- Подпрограмма, которая выполняет действие
- Хранится в БД как объект схемы
- Можно вызывать повторно после создания



Procedures

Создание процедуры командой SQL

CREATE OR REPLACE

- Используйте команду `CREATE` для создания процедуры в СУБД
- Добавьте опцию `OR REPLACE` для замены существующей процедуры.

```
CREATE [OR REPLACE] PROCEDURE procedure_name  
  [(parameter1 [mode] datatype1,  
    parameter2 [mode] datatype2, ...)]  
IS|AS
```

```
  [local_variable_declarations; ...]  
BEGIN  
    -- actions;  
END [procedure_name];
```

Блок PL/SQL

Рекомендации по наименованию переменных PL/SQL

Структура PL/SQL	Рекомендация	Пример
Переменная	<i>v_variable_name</i>	v_rate
Константа	<i>c_constant_name</i>	c_rate
Параметр подпрограммы	<i>p_parameter_name</i>	p_id
Bind (host) переменная	<i>b_bind_name</i>	b_salary
Курсор	<i>cur_cursor_name</i>	cur_emp
Запись (Record)	<i>rec_record_name</i>	rec_emp
Тип	<i>type_name_type</i>	ename_table_type
Исключение (Exception)	<i>e_exception_name</i>	e_products_invalid
Ссылка на файл	<i>f_file_handle_name</i>	f_file

Параметры и режимы параметров

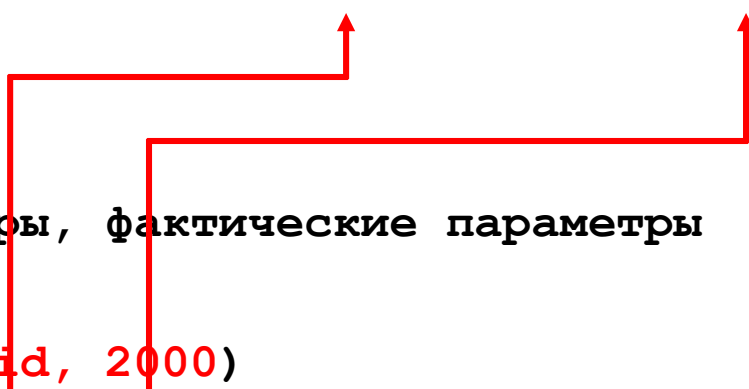
- Объявляются после названия подпрограммы в секции заголовка
- Используются для передачи данных от вызывающей программы в подпрограмму
- Работают как локальные переменные в подпрограмме с учетом режима работы параметра:
 - Входные параметры (`IN`) используется по умолчанию и служит для передачи значений в процедуру
 - Выходные параметры (`OUT`) используется для передачи значения из подпрограммы в вызывающую программу
 - Режим `IN OUT` используется для передачи значения в подпрограмму и обратно

Формальные и фактические параметры

- Формальные параметры: локальные переменные в процедуре, объявленные в спецификации
- Фактические параметры: Значения (литералы), переменные и выражения, которые передаются в процедуру.

```
-- Объявление процедуры, формальные параметры
CREATE PROCEDURE raise_sal(p_id NUMBER, p_sal NUMBER) IS
BEGIN
  . . .
END raise_sal;

-- Вызов процедуры, фактические параметры
v_emp_id := 100;
raise_sal(v_emp_id, 2000)
```

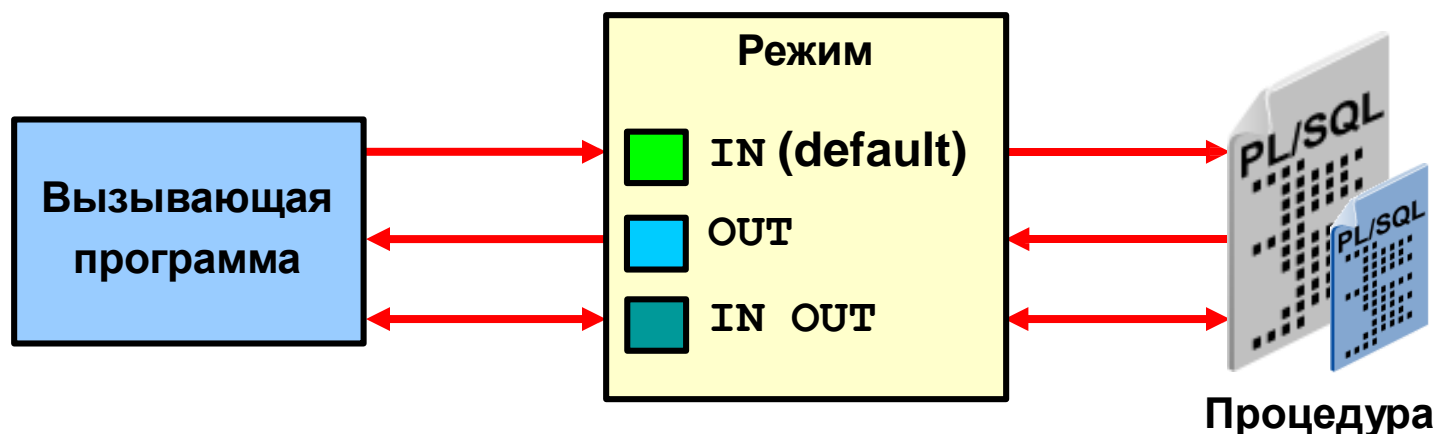


The diagram consists of red lines and arrows. Two vertical lines originate from the first two arguments of the procedure call, `v_emp_id` and `2000`. These lines extend upwards and then turn horizontally to the right, ending in arrows that point to the corresponding formal parameters in the procedure definition, `p_id` and `p_sal`.

Режим параметров в процедуре

- Режим параметров указывается при объявлении формальных параметров в заголовке процедуры.
- По умолчанию используется входной режим `IN` если никакой режим не указан.

```
CREATE PROCEDURE proc_name(param_name [mode] datatype)  
...
```

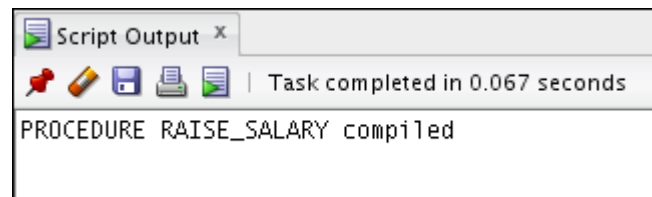


Режимы параметров

IN	OUT	IN OUT
По умолчанию	Надо указывать	Надо указывать
Значения передаются в подпрограмму	Значение возвращается в вызывающее окружение	Значение передается в подпрограмму, затем возвращается в вызывающее окружение
Формальный параметр действует как константа	Не инициализирован	Инициализированная переменная
Фактический параметр может быть литерал, выражение, константа, переменная	Переменная	Переменная
Можно присвоить значение по умолчанию	Нет значения по умолчанию	Нет значения по умолчанию

Использование входного режима IN

```
CREATE OR REPLACE PROCEDURE raise_salary
  (p_id      IN employees.employee_id%TYPE,
   p_percent IN NUMBER)
IS
BEGIN
  UPDATE employees
  SET    salary = salary * (1 + p_percent/100)
  WHERE  employee_id = p_id;
END raise_salary;
/
```



```
EXECUTE raise_salary(176, 10)
```

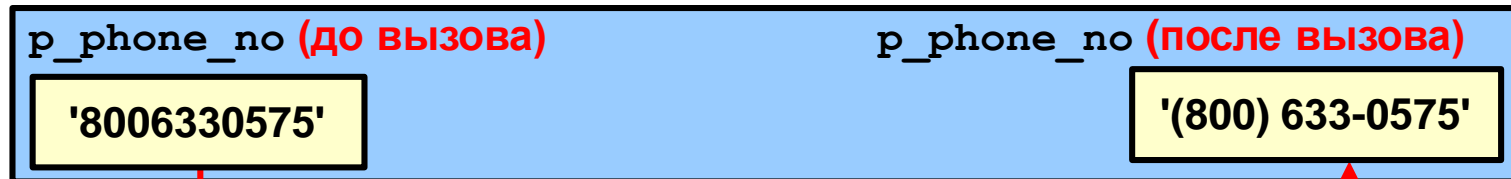
Использование выходного параметра OUT

```
CREATE OR REPLACE PROCEDURE query_emp
  (p_id      IN  employees.employee_id%TYPE,
   p_name    OUT employees.last_name%TYPE,
   p_salary  OUT employees.salary%TYPE) IS
BEGIN
  SELECT  last_name, salary INTO p_name, p_salary
  FROM    employees
  WHERE   employee_id = p_id;
END query_emp;
/
```

```
SET SERVEROUTPUT ON
DECLARE
  v_emp_name employees.last_name%TYPE;
  v_emp_sal   employees.salary%TYPE;
BEGIN
  query_emp(171, v_emp_name, v_emp_sal);
  DBMS_OUTPUT.PUT_LINE(v_emp_name||' earns '||
    to_char(v_emp_sal, '$999,999.00'));
END;
/
```

Использование режима INOUT

Вызывающее окружение



```
CREATE OR REPLACE PROCEDURE format_phone
  (p_phone_no IN OUT VARCHAR2) IS
BEGIN
  p_phone_no := '(' || SUBSTR(p_phone_no,1,3) ||
                ')' || SUBSTR(p_phone_no,4,3) ||
                '-' || SUBSTR(p_phone_no,7);
END format_phone;
/
```

```
anonymous block completed
B_PHONE_NO
-----
8006330575

anonymous block completed
B_PHONE_NO
-----
(800) 633-0575
```

Просмотр выходных параметров через DBMS_OUTPUT.PUT_LINE

После вызова процедуры OUT параметры принимают новые значения, и их можно отобразить при помощи процедуры DBMS_OUTPUT.PUT_LINE.

```
SET SERVEROUTPUT ON
```

```
DECLARE
```

```
  v_emp_name employees.last_name%TYPE;
```

```
  v_emp_sal   employees.salary%TYPE;
```

```
BEGIN
```

```
  query_emp(171, v_emp_name, v_emp_sal);
```

```
  DBMS_OUTPUT.PUT_LINE('Name: ' || v_emp_name);
```

```
  DBMS_OUTPUT.PUT_LINE('Salary: ' || v_emp_sal);
```

```
END;
```

```
anonymous block completed  
Name: Smith  
Salary: 7400
```


Удаление процедуры

- Используйте команду `DROP` :

```
DROP PROCEDURE raise_salary;
```

Просмотреть информацию о процедуре в Data Dictionary Views

```
SELECT text
FROM   user_source
WHERE  name = 'ADD_DEPT' AND type = 'PROCEDURE'
ORDER BY line;
```

	TEXT
1	PROCEDURE add_dept(
2	p_name departments.department_name%TYPE:= 'Unknown',
3	p_loc departments.location_id%TYPE DEFAULT 1700) IS
4	
5	BEGIN
6	INSERT INTO departments (department_id, department_name, location_id)
7	VALUES (departments_seq.NEXTVAL, p_name, p_loc);
8	END add_dept;