ORACLE 12c

PL/SQL Программные модули

Лекция 12

Программные модули

- Локальные
- Хранимые

Локальные программные модули

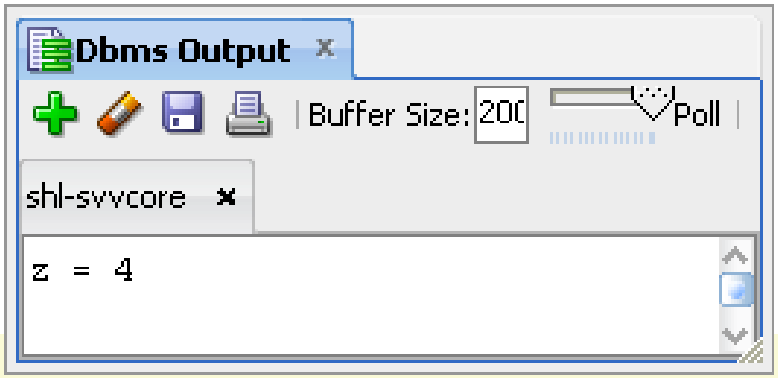
- Локальный программный модуль – это процедура или функция, определенная в секции декларации PL/SQL блока
- Объявление локальных процедур и функций должно размещаться в конце секции декларации после всех типов, записей, курсоров, переменных и исключений
- Локальные процедуры и функции могут быть использованы только в рамках блока, в котором они объявлены
- Локальные процедуры и функции могут быть перегружены

Перегрузка программных модулей

- Параметры должны отличаться семейством (number, character, datetime, boolean)
- Тип программного модуля должен отличаться – можно перегружать процедуру и функцию с одинаковым именем и списком параметров
- Число параметров должно быть разным

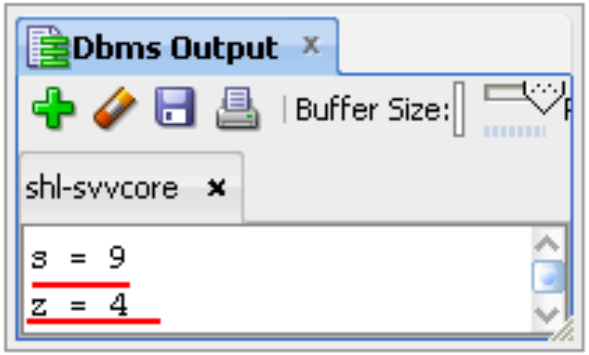
Локальные процедуры

```
-- 13/08.sql
declare
   x number(3):= 4;
   y number(3):= 5;
   z number(3);
   procedure summod5 (x1 number, x2 number, x3 out number )
   is
      z number(3):= 5;
      begin
        x3 := mod(x1+ x2,z);
      end  summod5;
begin
    summod5(x,y,z);
    dbms_output.put_line('z = '||z);

exception
   when others then dbms_output.put_line(sqlerrm);
end;
/
```



Локальные функции

```
-- 13/09.sql
declare
  x number(3):= 4;
  y number(3):= 5;
  z number(3);
  s number(5);
  function  summod5 (x1 number, x2 number, x3 out number)
    return number  is
        z number(3):= 5;
    begin
      x3 := mod(x1+ x2,z);
      return (x1+x2);
    end  summod5;
begin
    s := summod5(x,y,z);
    dbms_output.put_line('s = '||s);
    dbms_output.put_line('z = '||z);

 exception
   when others then dbms_output.put_line(sqlerrm);
end;
/
```



Программные модули

- Процедура
- Функция
- Пакет
- Триггер
- Объектный тип
- Хранимые процедуры на Java

Процедура

- Процедура – именованный модуль, который выполняет одно или несколько выражений и может принимать или возвращать значения через список параметров

Привилегии

- Для создания процедур необходима привилегия create procedure

```
SVVCORE@sh1>
SVVCORE@sh1> connect system/system@sh1;
Соединено.
sh1 - SYSTEM - 12.12.10
SYSTEM@sh1> grant create procedure to RLSVV;

Привилегии предоставлены.

SYSTEM@sh1>
SYSTEM@sh1>
```

Параметры

- Наименование
- Тип данных
- Режим передачи
- Начальное значение

Тип данных параметров

- PL/SQL или программно-определенный
- Не может быть ограничен по размеру
- Размер определяется через вызывающую программу или через связанное объявление переменной

Параметры

- Типы параметров:
  - IN
  - OUT
  - IN OUT
- При выполнении:
  - Значения OUT устанавливаются в NULL
  - Значения IN OUT остаются неизменными
  - При ошибке присвоения для параметров откатываются, кроме NOCOPY

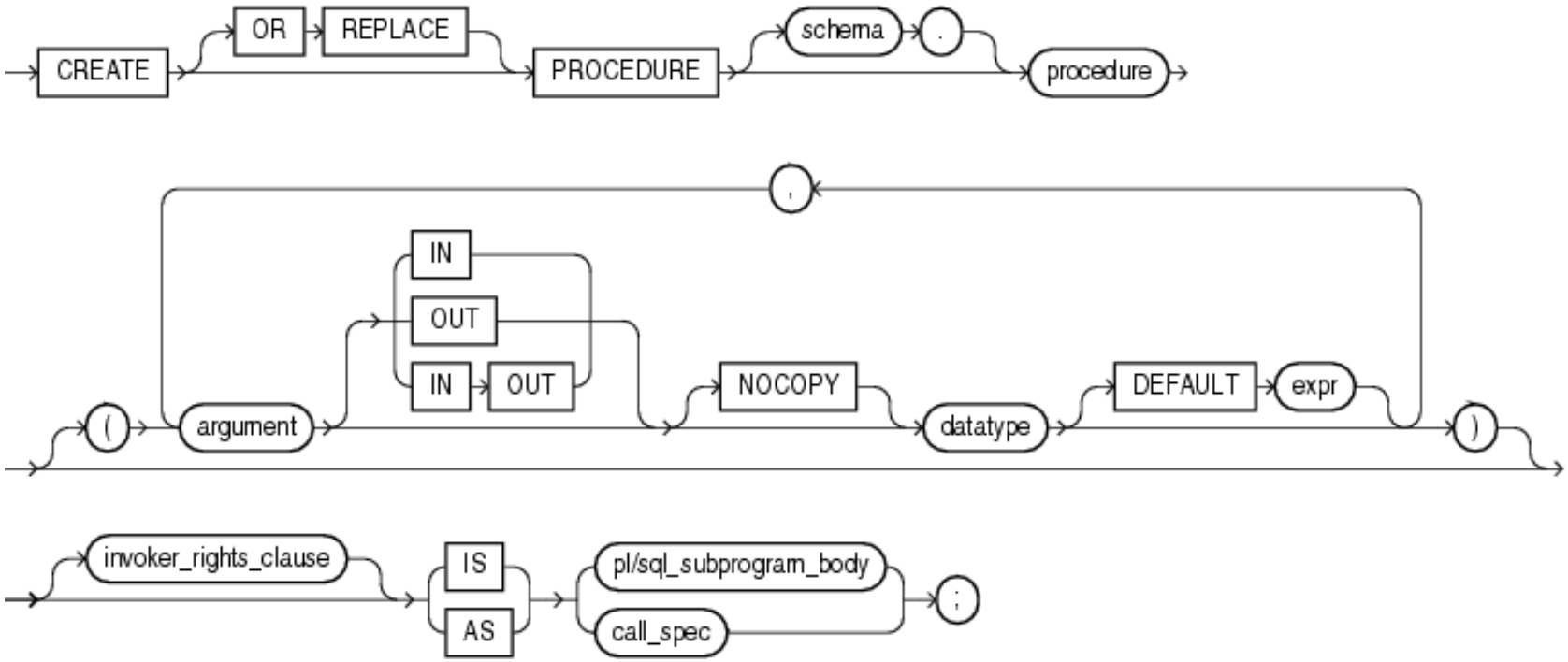
Значения по умолчанию

- IN, IN OUT
- Можно не задавать при вызове

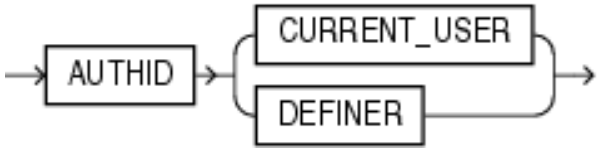
Передача параметров

- Позиционный – каждое значение в списке аргументов вызова ставится в соответствие формальному параметру по порядку.
  Empid_to_name(23, name, surname);
- Именованный – явно связывает аргументы при вызове с параметрами по именам.
  Empid_to_name(in_id =>23, out_name=> name, out_surname =>surname);
- Можно комбинировать оба метода, пока позиционные аргументы стоят слева.
  - *Empid_to_name(23, name, out_surname =>surname);*

Синтаксис



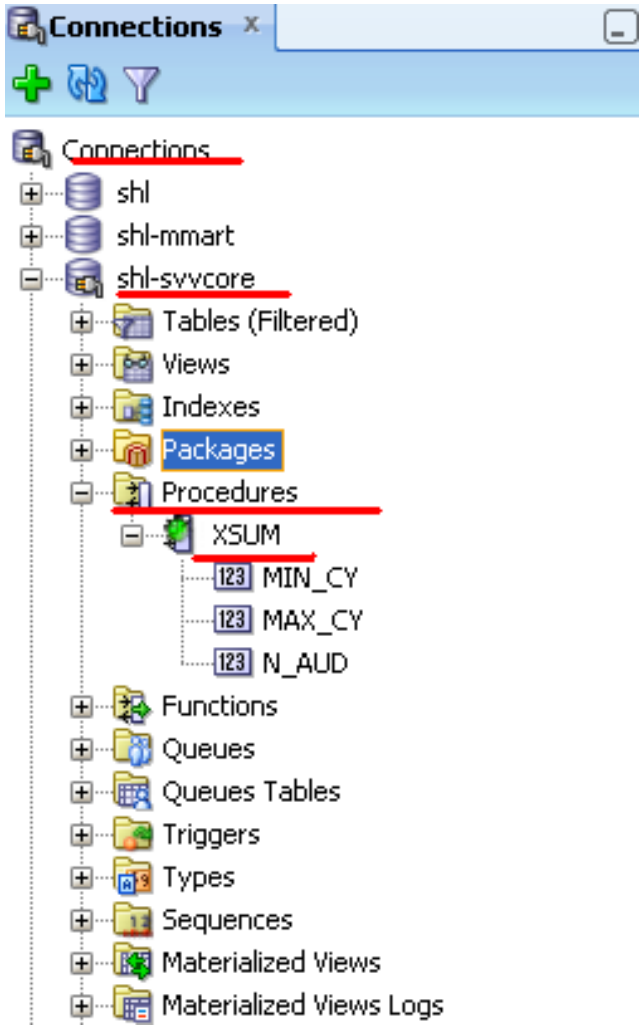
invoker_rights_clause

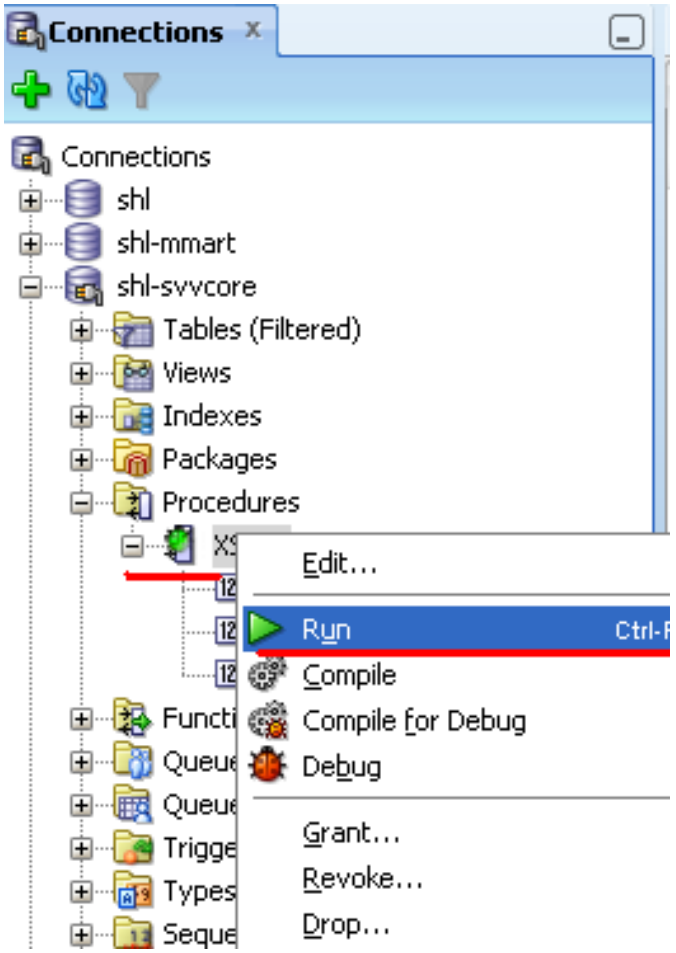


Процедуры

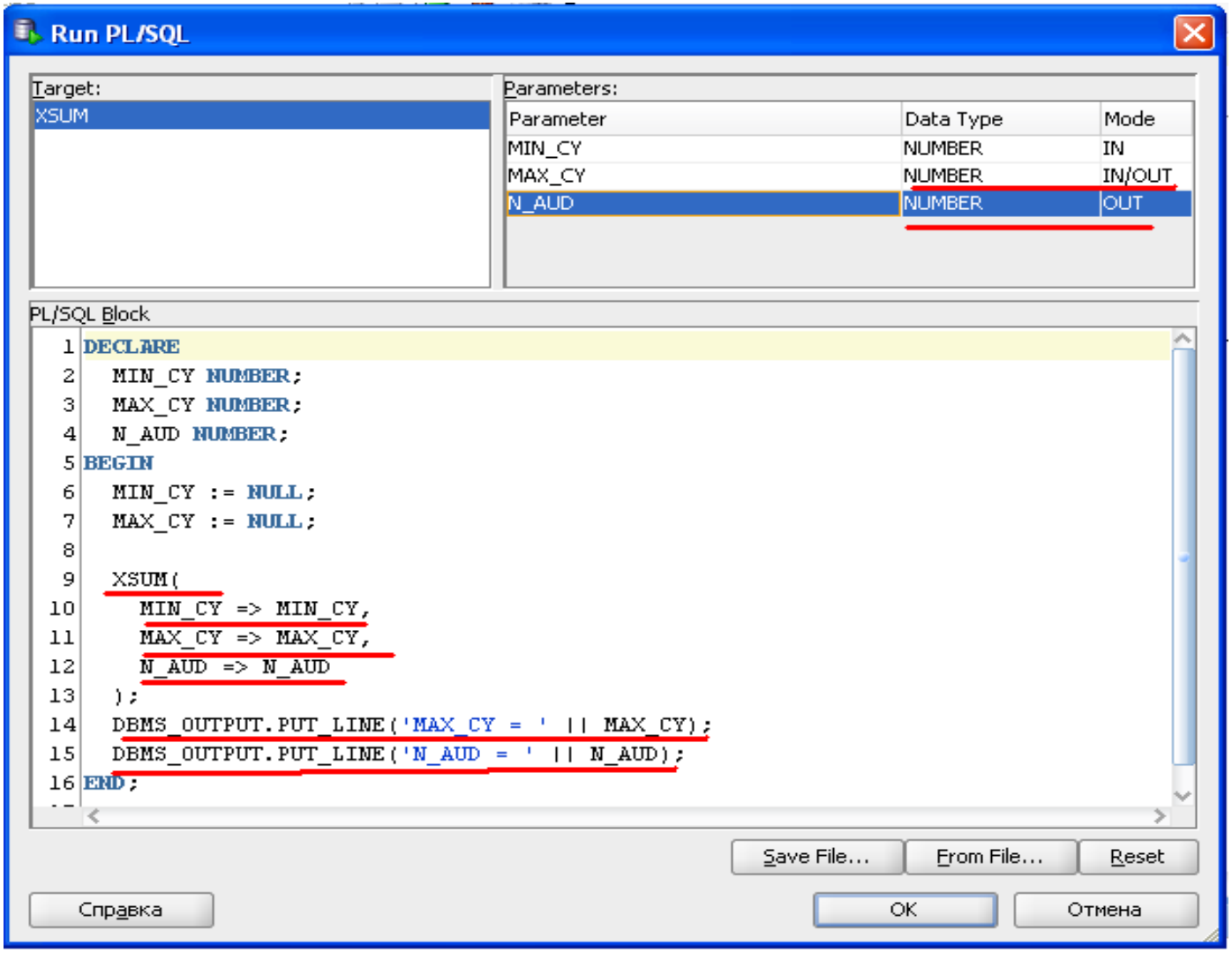
```
  -- 16/01.sql
 --select * from dba_sys_privs where grantee = 'RLSVV';
-- select * from user_role_privs where username = 'SVVCORE';
--  grant create procedure to RLSVV

create or replace procedure svvcore.xsum(
          min_cy  in     svvcore.auditorium.auditorium_capacity%type,  -- минимальная вместимость
          max_cy  in out svvcore.auditorium.auditorium_capacity%type,  -- максимальная вместимость
          n_aud   out     number                                      -- колличество
                                          )
   is
    m_max_cy  svvcore.auditorium.auditorium_capacity%type;
    m_n_aud   number := 0;
   begin
     select count(*), max(auditorium_capacity) into m_n_aud, m_max_cy from svvcore.auditorium
     where auditorium_capacity >= min_cy and auditorium_capacity <=  max_cy;
   exception
      when others then dbms_output.put_line(sqlerrm);
end xsum;
```

Вызов процедуры

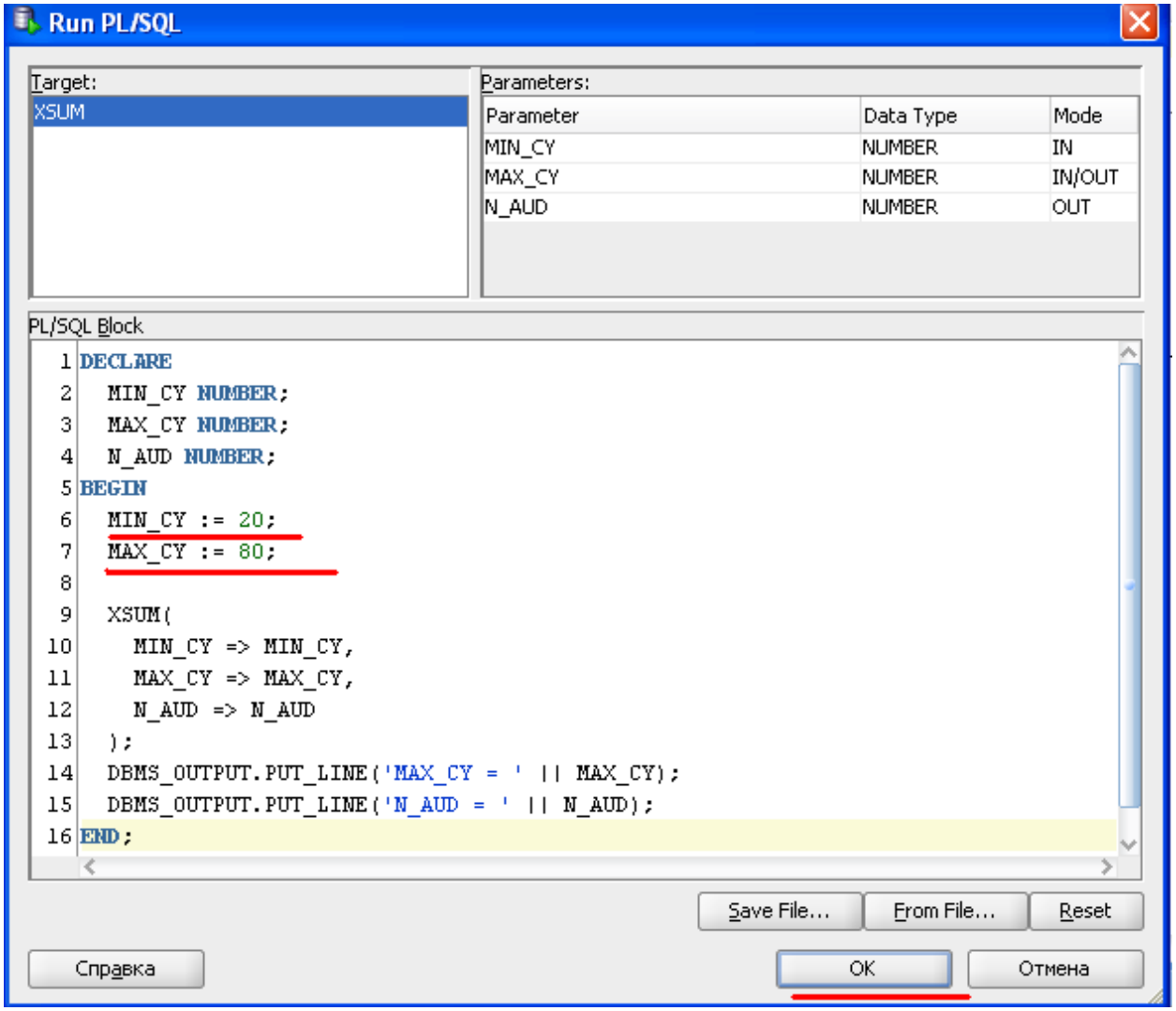




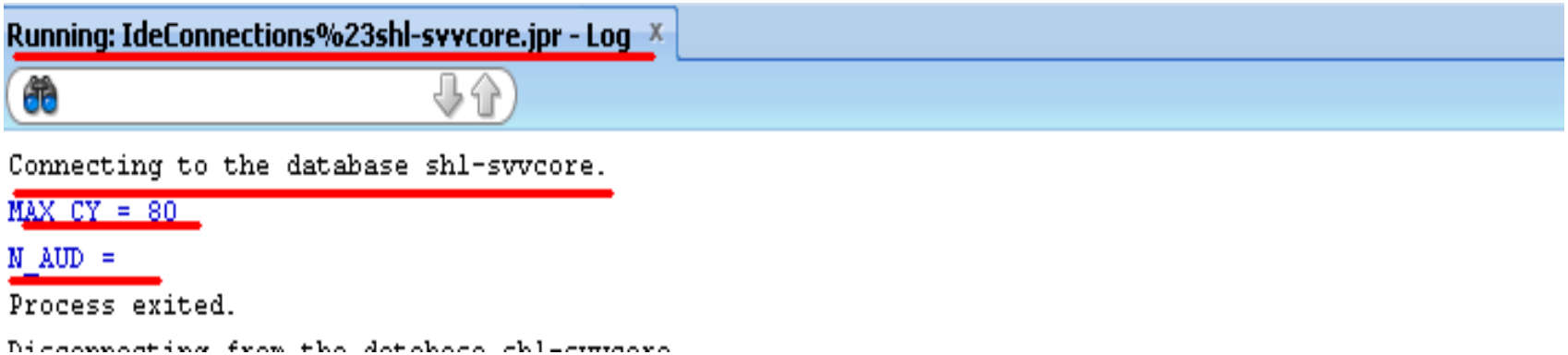
Отладка


Run PL/SQL
Target:
XSUM
Parameters:
Parameter
Data Type
Mode
MIN_CY
NUMBER
IN
MAX_CY
NUMBER
IN/OUT
N_AUD
NUMBER
OUT
PL/SQL Block
DECLARE
MIN_CY NUMBER;
MAX_CY NUMBER;
N_AUD NUMBER;
BEGIN
MIN_CY := NULL;
MAX_CY := NULL;
XSUM(
MIN_CY => MIN_CY,
MAX_CY => MAX_CY,
N_AUD => N_AUD
);
DBMS_OUTPUT.PUT_LINE('MAX_CY = ' || MAX_CY);
DBMS_OUTPUT.PUT_LINE('N_AUD = ' || N_AUD);
END;
Save File...
From File...
Reset
Справка
OK
Отмена

Отладка


Run PL/SQL
Target:
XSUM
Parameters:
Parameter
Data Type
Mode
MIN_CY
NUMBER
IN
MAX_CY
NUMBER
IN/OUT
N_AUD
NUMBER
OUT
PL/SQL Block
1 DECLARE
2 MIN_CY NUMBER;
3 MAX_CY NUMBER;
4 N_AUD NUMBER;
5 BEGIN
6 MIN_CY := 20;
7 MAX_CY := 80;
8
9 XSUM(
10 MIN_CY => MIN_CY,
11 MAX_CY => MAX_CY,
12 N_AUD => N_AUD
13);
14 DBMS_OUTPUT.PUT_LINE('MAX_CY = ' || MAX_CY);
15 DBMS_OUTPUT.PUT_LINE('N_AUD = ' || N_AUD);
16 END;
Save File...
From File...
Reset
Справка
OK
Отмена

Отладка


Running: IdeConnections%23shl-svvcore.jpr - Log
Connecting to the database shl-svvcore.
MAX_CY = 80
N_AUD =
Process exited.

Переменные

```
   -- 16/01.sql
 --select * from dba_sys_privs where grantee = 'RLSVV';
-- select * from user_role_privs where username = 'SVVCORE';
--  grant create procedure to RLSVV

create or replace procedure svvcore.xsum(
          min_cy  in     svvcore.auditorium.auditorium_capacity%type,  -- минимальная вместимость
          max_cy  in out svvcore.auditorium.auditorium_capacity%type,  -- максимальная вместимость
          n_aud   out     number                                      -- колличество
                                          )
   is
    m_max_cy  svvcore.auditorium.auditorium_capacity%type;
    m_n_aud   number := 0;
   begin
     select count(*), max(auditorium_capacity) into m_n_aud, m_max_cy from svvcore.auditorium
     where auditorium_capacity >= min_cy and auditorium_capacity <=  max_cy;
     max_cy := m_max_cy;
     n_aud := m_n_aud;
   exception
       when others then dbms_output.put_line(sqlerrm);
end xsum;
```

Вызов процедур

```
SVVCORE@sh1> exec :max_cy:=10;

Процедура PL/SQL успешно завершена.

SVVCORE@sh1> exec xsum(20,:max_cy,:n_aud);

Процедура PL/SQL успешно завершена.

SVVCORE@sh1> select :max_cy, :n_aud from dual;

   :MAX_CY     :N_AUD
---------- ----------
                    0
```

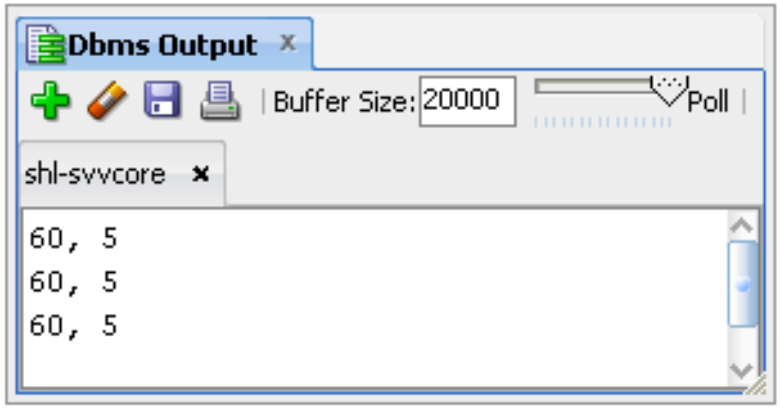
```
SVVCORE@sh1>
SVVCORE@sh1> var max_cy number;
SVVCORE@sh1> var n_aud  number;
SVVCORE@sh1> var
переменная   max_cy
тип данных   NUMBER

переменная   n_aud
тип данных   NUMBER
SVVCORE@sh1> exec :max_cy:=20;

Процедура PL/SQL успешно завершена.

SVVCORE@sh1> exec :max_cy:=80;

Процедура PL/SQL успешно завершена.

SVVCORE@sh1> exec xsum(20,:max_cy,:n_aud);

Процедура PL/SQL успешно завершена.

SVVCORE@sh1> select :max_cy, :n_aud from dual;

   :MAX_CY     :N_AUD
---------- ----------
        60          5
```

Вызов процедур

```
  -- 16/02.sql
--select * from dba_sys_privs where grantee = 'RLSVV';
-- select * from user_role_privs where username = 'SVVCORE';
--  grant create procedure to RLSVV

create or replace procedure svvcore.xsum(
          min_cy  in     svvcore.auditorium.auditorium_capacity%type,  -- минимальная вместимость
          max_cy  in out svvcore.auditorium.auditorium_capacity%type,  -- максимальная вместимость
          n_aud   out     number                                       -- колличество
                                        )
   is
     begin
     select count(*), max(auditorium_capacity) into n_aud, max_cy from svvcore.auditorium
     where auditorium_capacity >= min_cy and auditorium_capacity <=  max_cy;
     exception
       when others then dbms_output.put_line(sqlerrm);
end xsum;
```

Вызов процедур

```
-- 16/03.sql
declare
  max_cy number(3) := 80;
  n_aud  number(3):= 0;
begin
  svvcore.xsum(20, max_cy, n_aud);
  dbms_output.put_line(max_cy || ', '|| n_aud);

  xsum(n_aud=>n_aud,   min_cy =>20, max_cy=>max_cy);
  dbms_output.put_line(max_cy || ', '|| n_aud);

  xsum(20, n_aud=>n_aud,  max_cy=>max_cy);
  dbms_output.put_line(max_cy || ', '|| n_aud);

end;
```

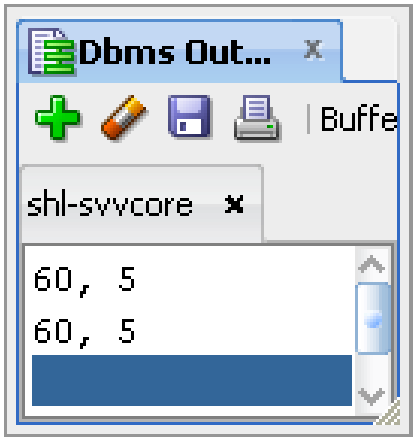


Значения по умолчанию - DEFAULT

```
  -- 16/04.sql
create or replace procedure svvcore.xsum(
          min_cy  in     svvcore.auditorium.auditorium_capacity%type default 20,  -- минимальная вместимость
          max_cy  in out svvcore.auditorium.auditorium_capacity%type,  -- максимальная вместимость
          n_aud   out      number                                      -- колличество
                                        )
     is
         no_max_cy exception;
     begin
       select count(*), max(auditorium_capacity) into n_aud, max_cy from svvcore.auditorium
       where auditorium_capacity >= min_cy and auditorium_capacity <=  max_cy;
       if n_aud is null
       then raise no_max_cy;
       end if;
     exception
       when no_max_cy then return;
       when others then dbms_output.put_line(sqlerrm);
end xsum;
```

Значения по умолчанию - DEFAULT

```
-- 16/05.sql
declare
  max_cy number(3) := 80;
  n_aud  number(3):= 0;
begin
  --svvcore.xsum(max_cy, n_aud);
  --dbms_output.put_line(max_cy || ', '|| n_aud);

  xsum(n_aud=>n_aud, max_cy=>max_cy);
  dbms_output.put_line(max_cy || ', '|| n_aud);

  xsum(n_aud=>n_aud,  max_cy=>max_cy);
  dbms_output.put_line(max_cy || ', '|| n_aud);

 end;
```



Компиляция

- OR REPLACE – перестроение уже существующего модуля, привилегии на выполнение сохраняются
- AUTHID – определяет, как будет выполняться модуль и разрешаться имена в БД:
  - DEFINER – (по умолчанию) от имени владельца модуля
  - CURRENT_USER - от имени пользователя, выполняющего модуль

Вызов процедуры пользователем, не являющимся владельцем

```
SVVCORE@sh1> connect svvguest/svvguest@sh1;
Соединено.
sh1 - SVVGUEST - 12.12.10
SVVGUEST@sh1> var
Переменная   max_cy
Тип данных   NUMBER

Переменная   n_aud
Тип данных   NUMBER
SVVGUEST@sh1> exec :max_cy:=80;

Процедура PL/SQL успешно завершена.

SVVGUEST@sh1> exec xsum(20,:max_cy,:n_aud);
BEGIN xsum(20,:max_cy,:n_aud); END;

      *
Ошибка в строке 1:
ORA-06550: Строка 1, столбец 7:
PLS-00201: идентификатор 'XSUM' должен быть объявлен
ORA-06550: Строка 1, столбец 7:
PL/SQL: Statement ignored
```

```
SVVGUEST@sh1> exec svvcore.xsum(20,:max_cy,:n_aud);
BEGIN svvcore.xsum(20,:max_cy,:n_aud); END;

      *
ошибка в строке 1:
ORA-06550: Строка 1, столбец 7:
PLS-00201: идентификатор 'SVVCORE.XSUM' должен быть объявлен
ORA-06550: Строка 1, столбец 7:
PL/SQL: Statement ignored
```

AUTHID {CURRENT_USER|DEFINER}

```
  -- 16/07.sql
create or replace procedure svvcore.xxsum(
          min_cy  in      svvcore.auditorium.auditorium_capacity%type default 20,  -- минимальная в
          max_cy  in out svvcore.auditorium.auditorium_capacity%type,  -- максимальная вместимость
          n_aud   out     number                                       -- колличество
                                        )
     authid current user is
         no_max_cy exception;
     begin
       select count(*), max(auditorium_capacity) into n_aud, max_cy from svvcore.auditorium
       where auditorium_capacity >= min_cy and auditorium_capacity <=  max_cy;
       if n_aud is null
       then raise no_max_cy;
       end if;
     exception
       when no_max_cy then return;
       when others then dbms_output.put_line(sqlerrm);
end xxsum;
```

SQL-оператор CALL вызова процедур

```
SVVGUEST@sh1>
SVVGUEST@sh1>
SVVGUEST@sh1> CALL svvcore.xsum(20,:max_cy,:n_aud);

Вызов завершен.

SVVGUEST@sh1> select :max_cy, :n_aud from dual;

   :MAX_CY     :N_AUD
---------- ----------
        60          5
```

USER_PROCEDURES

```
SVVGUEST@shl> connect svvcore/svvcore@shl;
Соединено.
shl - SVVCORE - 13.12.10
SVVCORE@shl> select * from user_procedures;

OBJECT_NAME                    PROCEDURE_NAME                 AGG PIP
------------------------------ ------------------------------ --- ---
XSUM                                                          NO  NO
XXSUM                                                         NO  NO
CALENDARPKG                    XINSERT                        NO  NO
CALENDARPKG                    NSEMESTER                      NO  NO

SVVCORE@shl>
SVVCORE@shl>
```

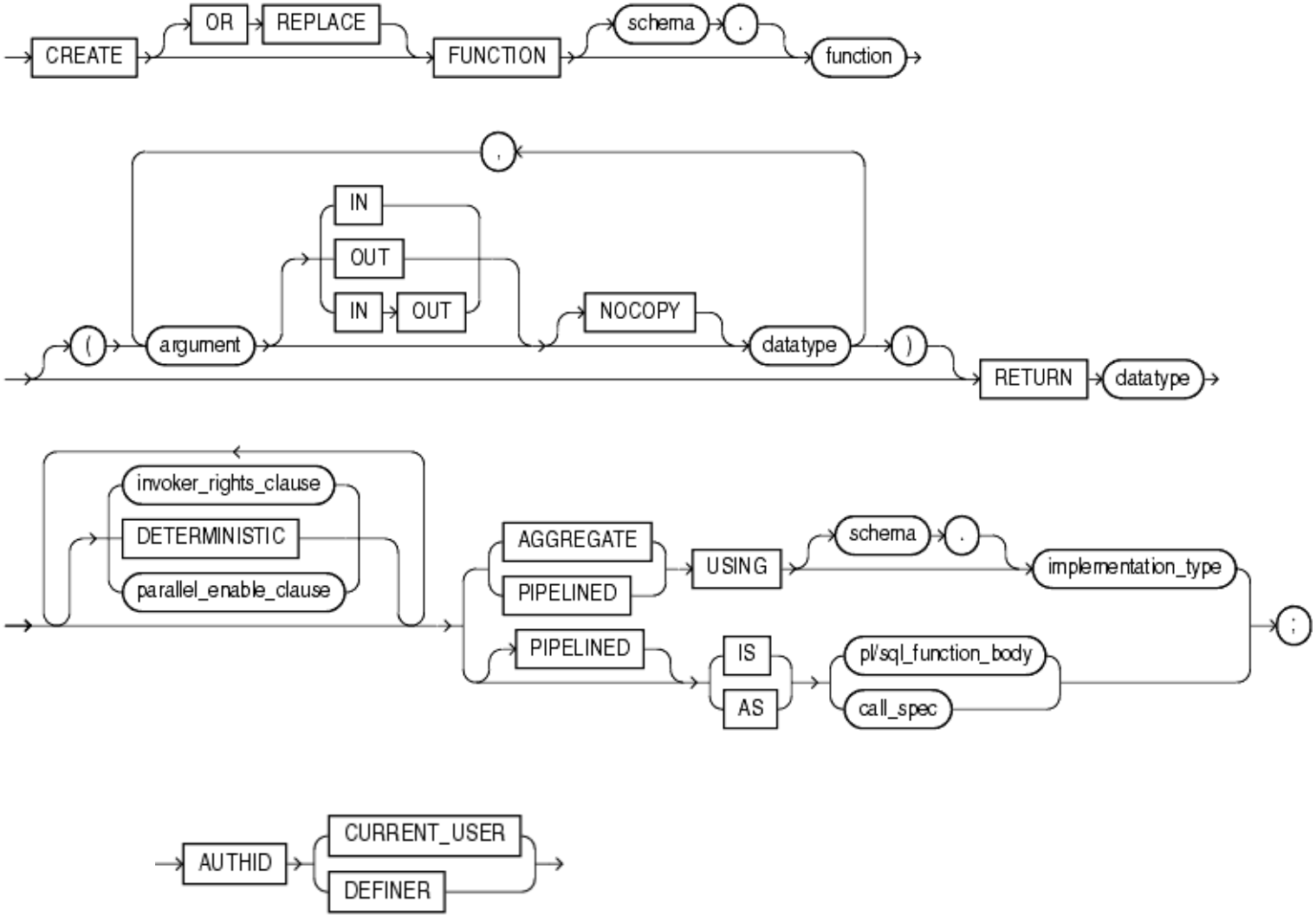
USER_SOURCE

```
SVVCORE@shl> column text format a100
SVVCORE@shl> column name format a10
SVVCORE@shl> select type, name, line, text from user_source where name ='XSUM'
  2  /

TYPE         NAME             LINE TEXT
------------ ---------- ---------- ----------------------------------------------------------------------------------------------------
PROCEDURE    XSUM                1 procedure         xsum(
PROCEDURE    XSUM                2            min_cy  in    svvcore.auditorium.auditorium_capacity%type default 20,  -- минималь
                                   стимость

PROCEDURE    XSUM                3            max_cy  in out svvcore.auditorium.auditorium_capacity%type,  -- максимальная вмест
PROCEDURE    XSUM                4            n_aud   out     number                                       -- колличество
PROCEDURE    XSUM                5                                      )
PROCEDURE    XSUM                6       is
PROCEDURE    XSUM                7           no_max_cy exception;
PROCEDURE    XSUM                8       begin
PROCEDURE    XSUM                9         select count(*), max(auditorium_capacity) into n_aud, max_cy from svvcore.auditorium

TYPE         NAME             LINE TEXT
------------ ---------- ---------- ----------------------------------------------------------------------------------------------------
PROCEDURE    XSUM               10         where auditorium_capacity >= min_cy and auditorium_capacity <=  max_cy;
PROCEDURE    XSUM               11         if n_aud is null
PROCEDURE    XSUM               12         then raise no_max_cy;
PROCEDURE    XSUM               13         end if;
PROCEDURE    XSUM               14       exception
PROCEDURE    XSUM               15         when no_max_cy then return;
PROCEDURE    XSUM               16         when others then dbms_output.put_line(sqlerrm);
PROCEDURE    XSUM               17  end xsum;
PROCEDURE    XSUM               18
PROCEDURE    XSUM               19
PROCEDURE    XSUM               20

20 строк выбрано.
```

Функция

- Функция – именованный модуль, который выполняет ноль или более выражений через фразу Return
- Может быть вызвана следующим образом:
  - В присвоении начального значения переменной
  - В выражении присвоения
  - В булевом выражении
  - В SQL запросе
  - Как аргумент в списке параметров другой функции или процедуры

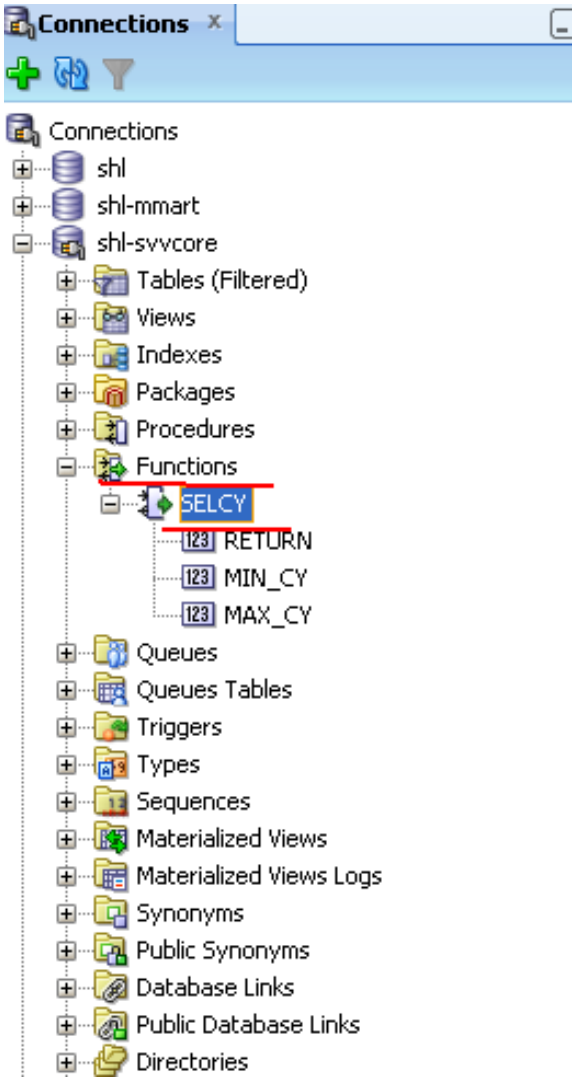
Функции


CREATE
OR
REPLACE
FUNCTION
schema
.
function
(
argument
IN
OUT
IN
OUT
NOCOPY
datatype
)
RETURN
datatype
invoker_rights_clause
DETERMINISTIC
parallel_enable_clause
AGGREGATE
PIPELINED
USING
schema
.
implementation_type
PIPELINED
IS
AS
pl/sql_function_body
call_spec
;
AUTHID
CURRENT_USER
DEFINER

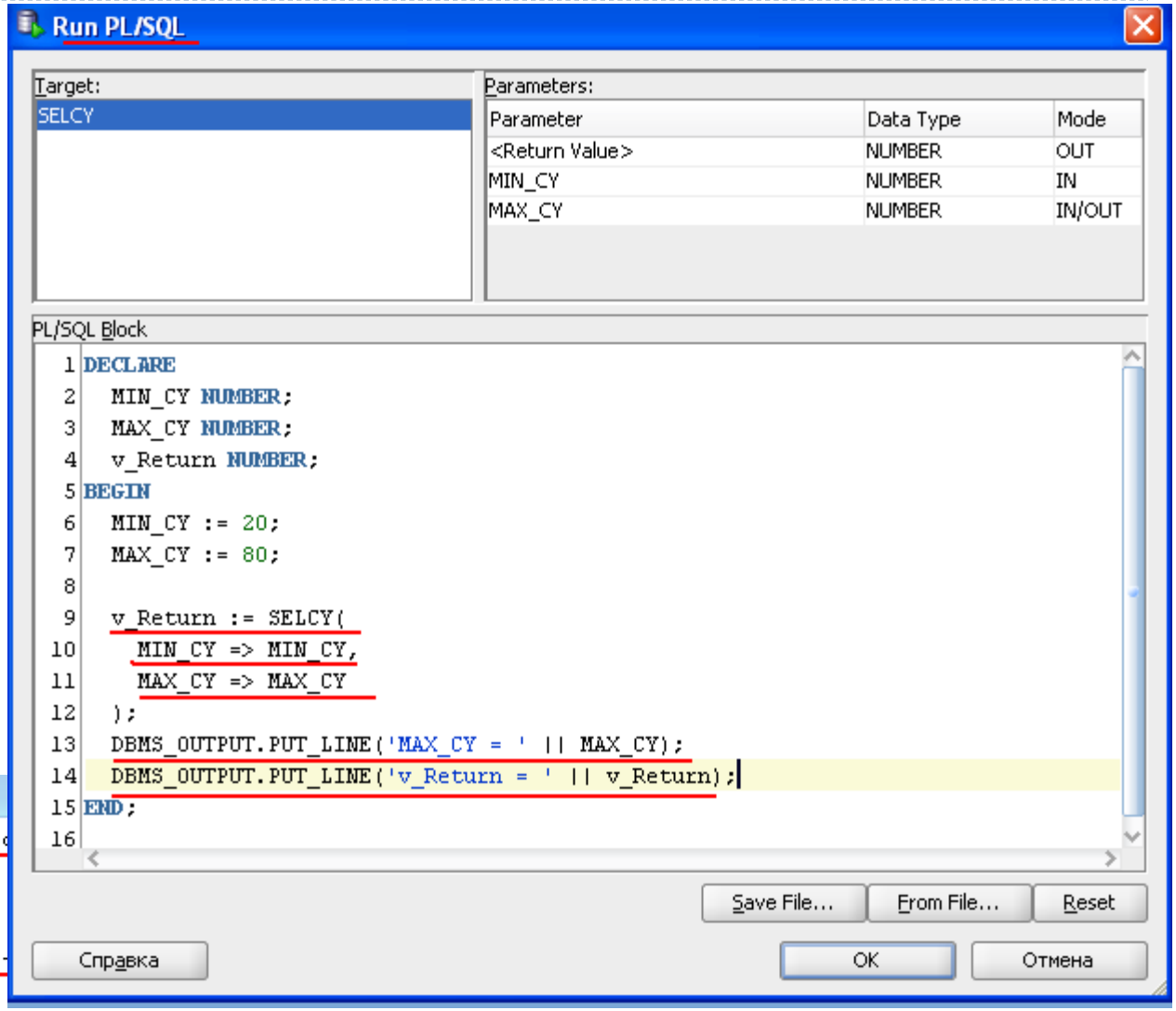
Функция

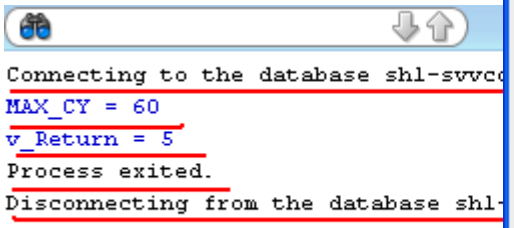
```
  -- 16/10 .sql
create or replace function selcy(
                            min_cy auditorium.auditorium_capacity%type,
                            max_cy in out auditorium.auditorium_capacity%type
                            )
  return number is
     rc  number(5);
  begin
     select count(*), max(auditorium_capacity) into rc, max_cy from svvcore.auditorium
     where auditorium_capacity >= min_cy and auditorium_capacity <=  max_cy group by max_cy;
     return rc;
  exception
     when others then return -1;
end selcy;
```

SQL Developer



Отладка


Run PL/SQL
Target:
SELCY
Parameters:
Parameter
Data Type
Mode
<Return Value>
NUMBER
OUT
MIN_CY
NUMBER
IN
MAX_CY
NUMBER
IN/OUT
PL/SQL Block
1 DECLARE
2 MIN_CY NUMBER;
3 MAX_CY NUMBER;
4 v_Return NUMBER;
5 BEGIN
6 MIN_CY := 20;
7 MAX_CY := 80;
8
9 v_Return := SELCY(
10 MIN_CY => MIN_CY,
11 MAX_CY => MAX_CY
12);
13 DBMS_OUTPUT.PUT_LINE('MAX_CY = ' || MAX_CY);
14 DBMS_OUTPUT.PUT_LINE('v_Return = ' || v_Return);
15 END;
16
Save File...
From File...
Reset
Справка
OK
Отмена


Connecting to the database shl-svvc
MAX_CY = 60
v_Return = 5
Process exited.
Disconnecting from the database shl-

Применение функций в SELECT

```
SVVCORE@sh1>
SVVCORE@sh1> select selcy(20,80) from dual;
select selcy(20,80) from dual
       *
ошибка в строке 1:
ORA-06572: Функция SELCY имеет внешний аргумент


SVVCORE@sh1>
SVVCORE@sh1>
SVVCORE@sh1>
SVVCORE@sh1> var max_cy number;
SVVCORE@sh1> exec :max_cy:=80;

Процедура PL/SQL успешно завершена.

SVVCORE@sh1> select selcy(20,:max_cy) from dual;
select selcy(20,:max_cy) from dual
       *
ошибка в строке 1:
ORA-06572: Функция SELCY имеет внешний аргумент
```

Функция без параметров

```
create or replace function maximumcy
  return number is
     rc  number(5);
  begin
     select max(auditorium_capacity) into rc from svvcore.auditorium;
     return rc;
  exception
     when others then return -1;
end maximumcy;
```

Вызов в SELECT

```
  -- l6/ll .sql
create or replace function maxcy(
                              min_cy auditorium.auditorium_capacity%type,
                              max_cy  auditorium.auditorium_capacity%type
                              )
  return number is
     rc  number(5);
  begin
     select count(*) into rc from svvcore.auditorium
     where auditorium_capacity >= min_cy and auditorium_capacity <=  max_cy ;
     return rc;
  exception
     when others then return -l;
end maxcy;
```

```
SVVCORE@shl>
SVVCORE@shl>
SVVCORE@shl>
SVVCORE@shl> select maxcy(20,80) from dual;

MAXCY(20,80)
------------
           5
```

Ключевые слова

- DETERMINISTIC – функция детерминирована, если она возвращает одно и то же значение при вызове с теми же параметрами
- AGGREGATE USING – используется для агрегатных функций.

DETERMINISTIC

```
  -- l6/l3 .sql
create or replace function maximumcy_d
  return number deterministic  is
     rc  number(5);
  begin
     select max(auditorium_capacity) into rc from svvcore.auditorium;
     return rc;
  exception
     when others then return -1;
end maximumcy_d;
```

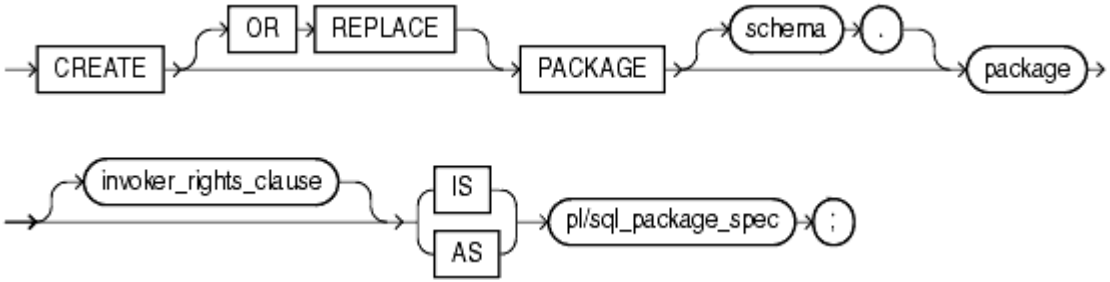
Пакеты

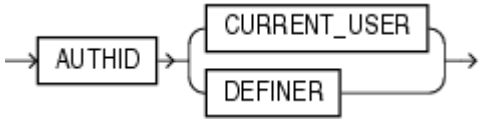
- Пакеты - коллекция PL/SQL объектов, сгруппированных вместе.
- Преимущества:
  - Скрытие информации
  - Объектно-ориентированный дизайн
  - Постоянство объектов в транзакциях
  - Улучшенная производительность
- Можно включать в пакет: процедуры, функции, константы, исключения, курсоры, переменные, TYPE выражения, записи, REF курсоры

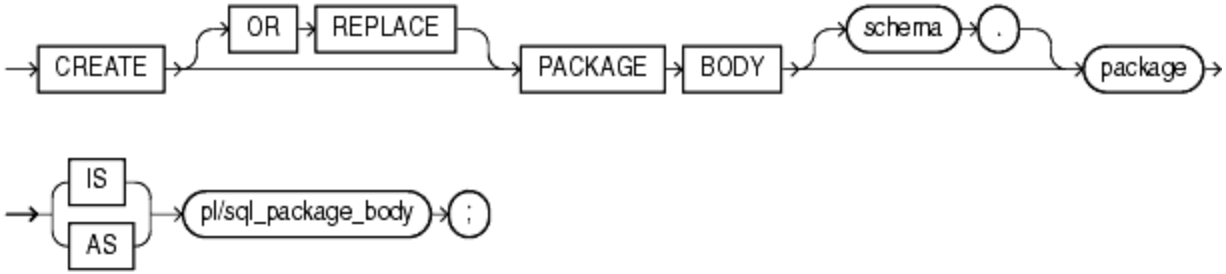
Пакеты

- Спецификация пакета (package) – обязательна, содержит список объектов для общего доступа из других модулей или приложения
- Реализация пакета (package body) – содержит весь программный код для реализации процедур и функций и спецификации, приватные объекты и секцию инициализации

Спецификация пакета







Пример заголовка пакета

```
create or replace
package teacherpkg as


  type teacher_rec is  record
  (
   t          teacher.teacher%type,
   tn         teacher.teacher_name%type,
   pp         teacher.pulpit%type
  );

  exc_xinsert  exception;
  exc_xupdate  exception;
  exc_xdelete  exception;
  procedure xinsert(tr teacher_rec);
  function  xupdate(tr teacher_rec) return boolean;
  function  xdelete(t  teacher.teacher%type) return boolean;

 end  teacherpkg;
  -- ============================================== --
```

Пример использования пакета

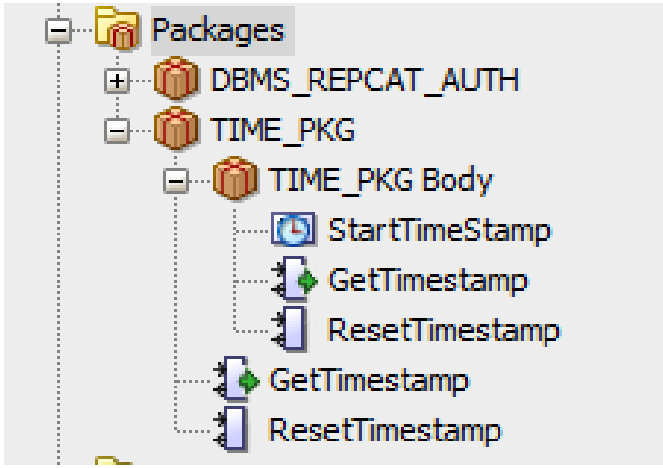
```
declare
 rec teacherpkg.teacher_rec;
begin
  rec.t := 'БНР';
  rec.tn := 'Бендер Остап Ибрагимович';
  rec.pp := 'ИСиТ';
  teacherpkg.xinsert(rec);
  commit;
  rec.tn := 'Бурнаш Михаил Антонович';
  if teacherpkg.xupdate(rec)
     then dbms_output.put_line('xupdate = ok');
     else dbms_output.put_line('xupdate = error');
  end if;
  commit;

   if teacherpkg.xdelete(rec.t)
     then dbms_output.put_line('xdelete = ok');
     else dbms_output.put_line('xdelete = error');
  end if;
  commit;
exception
  when teacherpkg.exc_xdelete then dbms_output.put_line('xdelete:'||sqlerrm);
  when teacherpkg.exc_xinsert then dbms_output.put_line('xinsert:'||sqlerrm);
  when teacherpkg.exc_xupdate then dbms_output.put_line('xupdate:'||sqlerrm);

end;
```

Пример спецификации пакета

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

Пример реализации пакета

```
CREATE OR REPLACE PACKAGE BODY time_pkg IS
   StartTimeStamp DATE := SYSDATE;
   -- StartTimeStamp is package data.
-- Function
   FUNCTION GetTimestamp RETURN DATE IS
   BEGIN
      RETURN StartTimeStamp;
   END GetTimestamp;
-- Procedures
   PROCEDURE ResetTimestamp(new_time DATE DEFAULT SYSDATE)
   IS
   BEGIN
      StartTimeStamp := new_time;
   END ResetTimestamp;
-- Initialization section
   BEGIN
    null;
   EXCEPTION
    WHEN NO_DATA_FOUND
         THEN dbms_output.put_line('not initialized');
END time_pkg;
```



Пример использования пакета

```
-- package use
begin
  dbms_output.put_line(to_char(time_pkg.gettimestamp, 'dd/mm/yyyy hh24:mi:ss'));

  dbms_lock.sleep(3);

  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;
```

```
14/12/2019 08:55:59
14/12/2019 08:55:59
14/12/2019 08:56:02
```

```
-- package use
begin
  dbms_output.put_line(to_char(time_pkg.gettimestamp, 'dd/mm/yyyy hh24:mi:ss'));
  dbms_lock.sleep(3);
  dbms_output.put_line(to_char(time_pkg.gettimestamp, 'dd/mm/yyyy hh24:mi:ss'));
end;
```

```
14/12/2019 08:56:02
14/12/2019 08:56:02
```

```
14/12/2019 08:56:02
14/12/2019 08:56:02

14/12/2019 08:56:02
14/12/2019 08:56:02
```

Пример пакета SERIALLY REUSABLE

```
-- 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;

CREATE OR REPLACE PACKAGE BODY time_pkg IS
   PRAGMA SERIALLY_REUSABLE;
   StartTimeStamp DATE := SYSTIMESTAMP;
   -- StartTimeStamp is package data.
-- Function
   FUNCTION GetTimestamp RETURN DATE IS
   BEGIN
      RETURN StartTimeStamp;
   END GetTimestamp;
-- Procedures
   PROCEDURE ResetTimestamp(new_time DATE DEFAULT SYSDATE)
   IS
   BEGIN
      StartTimeStamp := new_time;
   END ResetTimestamp;
-- Initialization section
   BEGIN
    null;
   EXCEPTION
    WHEN NO_DATA_FOUND
         THEN dbms_output.put_line('not initialized');
END time_pkg;
```

Пример пакета SERIALLY REUSABLE

```
-- package use
begin
  dbms_output.put_line(to_char(time_pkg.gettimestamp, 'dd/mm/yyyy hh24:mi:ss'));
  dbms_lock.sleep(3);
  dbms_output.put_line(to_char(time_pkg.gettimestamp, 'dd/mm/yyyy hh24:mi:ss'));
end;
```

```
14/12/2019 08:58:55
14/12/2019 08:58:55

14/12/2019 08:59:08
14/12/2019 08:59:08
```

Пакеты

- Вызов пакета:
  - Package_name.package_element;
- Структуры данных, объявленные в пакете, называются пакетными данными
- Пакетные переменные сохраняют свое состояние от одной транзакции к другой и являются глобальными данными

Пакеты

- AUTHID {CURRENT_USERIDEFINER}
- Словарь: USER_PROCEDURES, USER_SOURCE
- ALTER PACKAGE COMPILE PACKAGE
- ALTER PACKAGE COMPILE BODY
- DROP PACKAGE

Вопросы?