

Oracle Core
Тема 7
Процедуры и функции.
Пакеты.
Jobs и scheduler

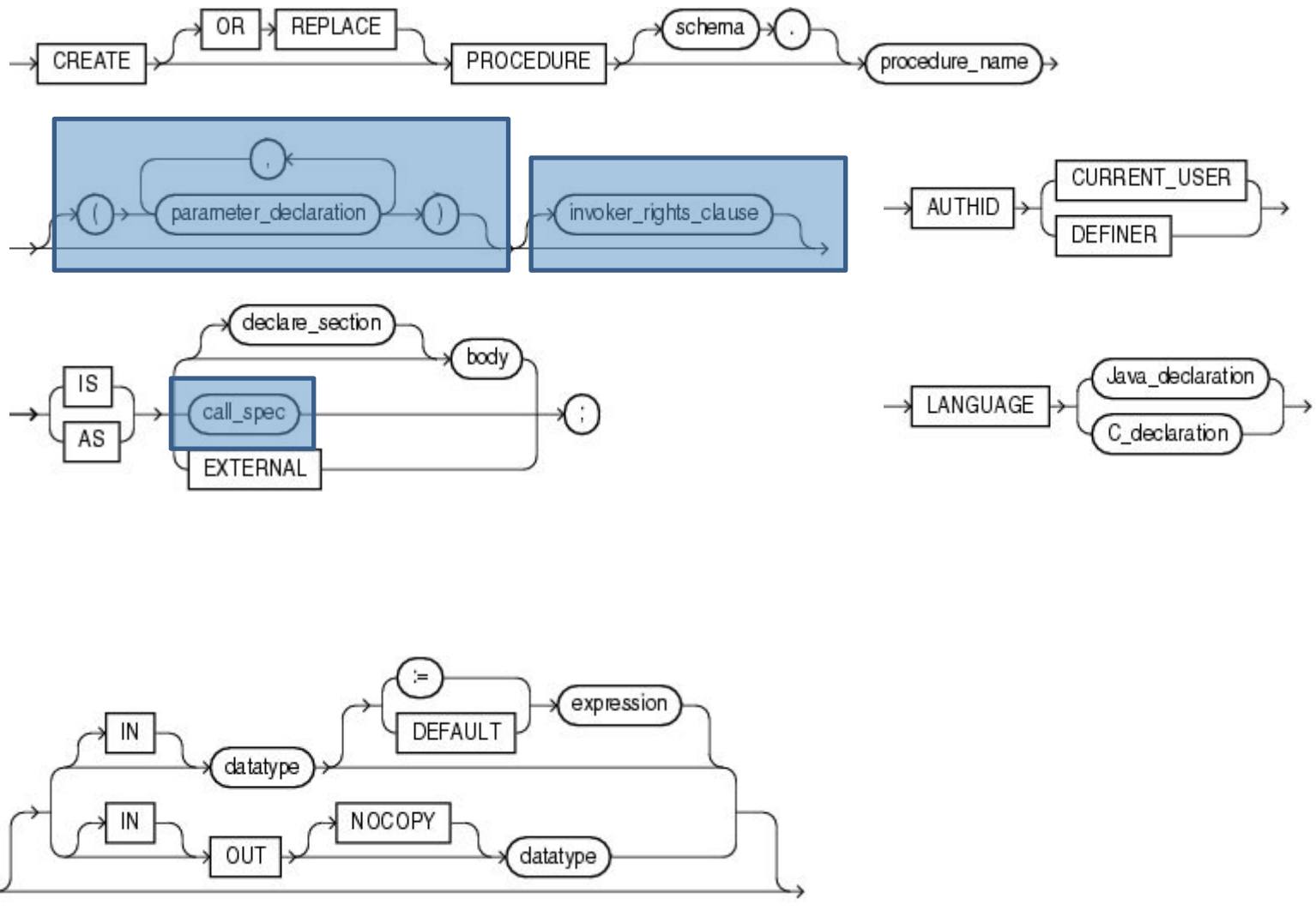
Содержание

- Структура процедуры/функции
- Структура пакета
- Перегрузка процедур/функций
- Глобальные переменные в пакетах
- INLINE Pragma
- SERIALLY_REUSABLE Pragma
- Conditional compilation Pragma
- Ограничения языка PL/SQL
- Package Writing Guidelines
- PL/SQL Source Text Wrapping
- Спецификация без тела
- Dependencies
- Запуск удаленных процедур (через dblink)
- Jobs
- Scheduler

Процедура

- SQL> CREATE OR REPLACE PROCEDURE p(p_1 NUMBER,
- 2 p_2 NUMBER DEFAULT 4,
- 3 p_3 OUT NUMBER) IS
- 4 v_1 NUMBER;
- 5 BEGIN
- 6 v_1 := p_2 * 2;
- 7 p_3 := p_1 + v_1;
- 8 END;
- 9 /
- Procedure created
- SQL> set serveroutput on ;
- SQL> exec dbms_output.enable(10000);
- PL/SQL procedure successfully completed
- SQL> DECLARE
- 2 n NUMBER;
- 3 BEGIN
- 4 p(p_1 => 10, p_3 => n);
- 5 dbms_output.put_line(n);
- 6 END,
- 7 /
- 18
- PL/SQL procedure successfully completed

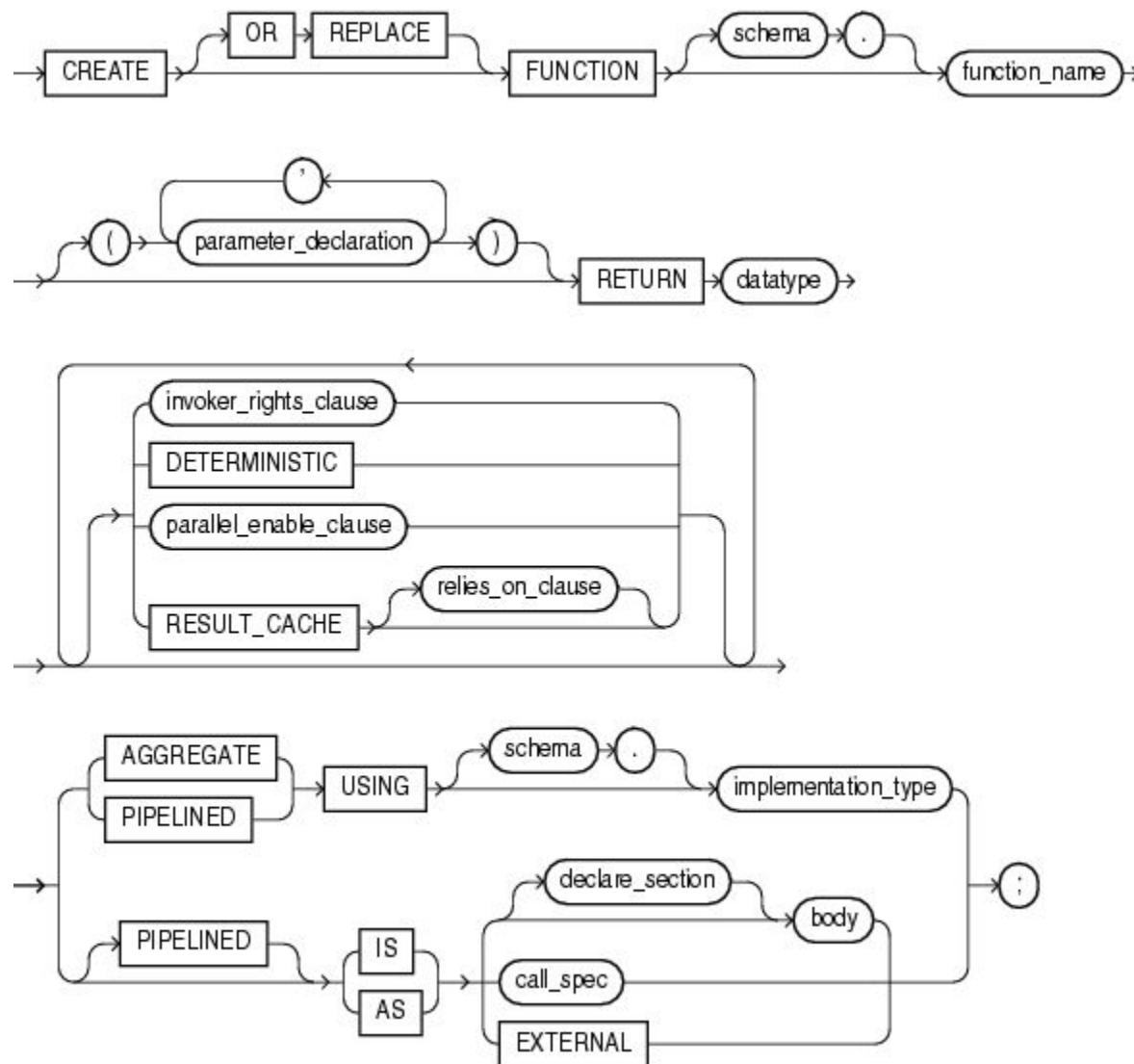
Структура процедуры



Функция

- SQL> create or replace function fnc_Multiply(n1 number, n2 number)
- 2 return number is
- 3 begin
- 4 return n1*n2;
- 5 end;
- 6 /
- Function created
- SQL> set serveroutput on;
- SQL> exec dbms_output.enable(10000);
- PL/SQL procedure successfully completed
- SQL> select **fnc_Multiply(10,4.5)** as mult from dual;
- **MULT**
- -----
- **45**

Структура функции



Пакеты



Структура спецификации пакета

```
CREATE OR REPLACE PACKAGE pkg_test AS

    c_hello_world CONSTANT VARCHAR2(50) := 'Hello, world!';

    TYPE t_name IS RECORD(
        last_name     VARCHAR2(100 CHAR),
        first_name    VARCHAR2(100 CHAR),
        middle_name   VARCHAR2(100 CHAR));

    ex_sun_is_shining EXCEPTION;

    TYPE cur_huge_data IS REF CURSOR RETURN dual%ROWTYPE;

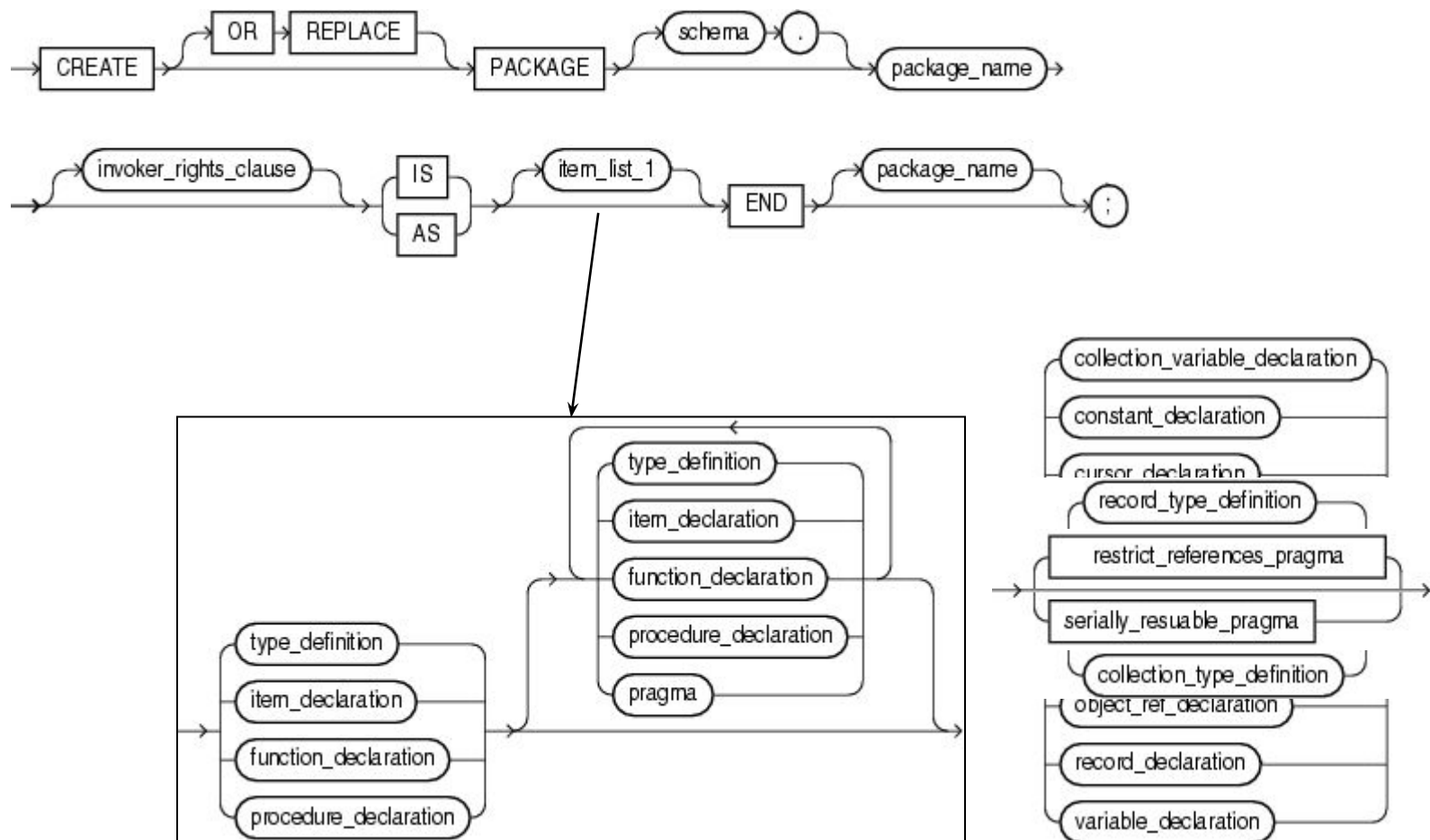
    TYPE t_table_indexed IS TABLE OF VARCHAR2(1000) INDEX BY BINARY_INTEGER;

    PROCEDURE make_me_happy(p_summa NUMBER);

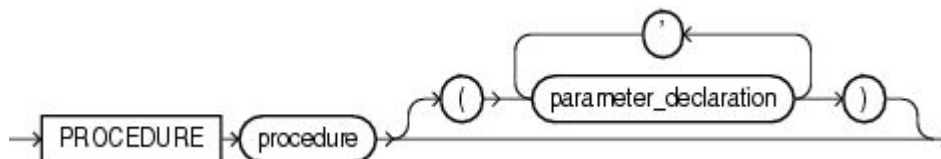
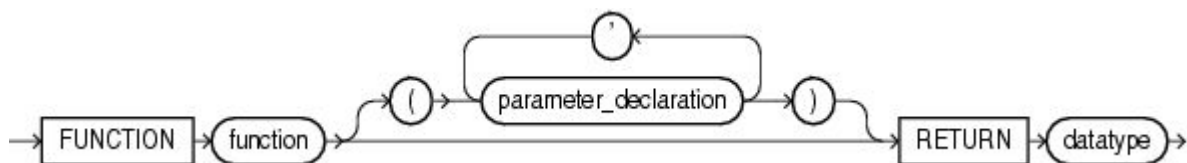
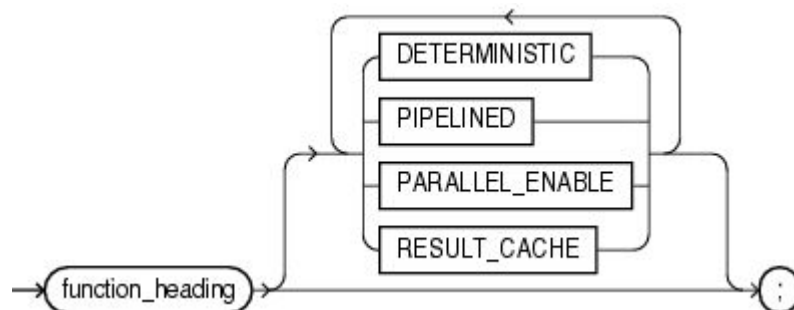
    FUNCTION am_i_happy(p_summa NUMBER) RETURN VARCHAR2;

END;
```

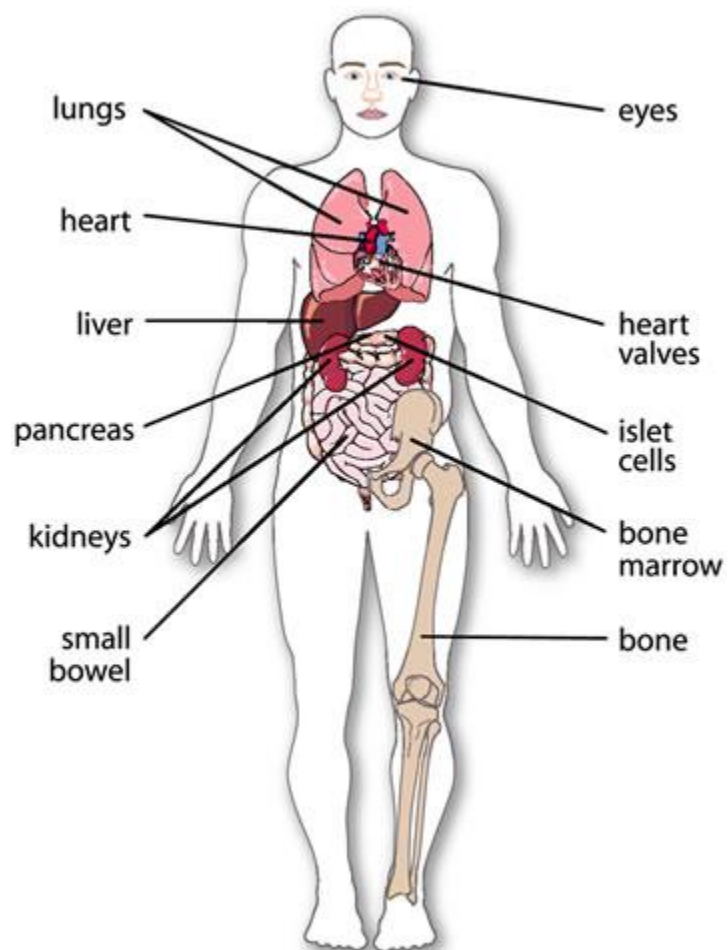

Структура спецификации пакета



Спецификация function/procedure



Тело пакета



Тело пакета

```
CREATE OR REPLACE PACKAGE BODY pkg_test AS
  PROCEDURE make_me_happy(p_summa NUMBER) IS
  BEGIN
    IF p_summa > 100500
    THEN
      dbms_output.put_line('You are happy!');
    ELSE
      dbms_output.put_line('I can't, sorry :-( ');
    END IF;
  END;
BEGIN
  dbms_output.put_line('Let's start');
END;
```

SQL> begin

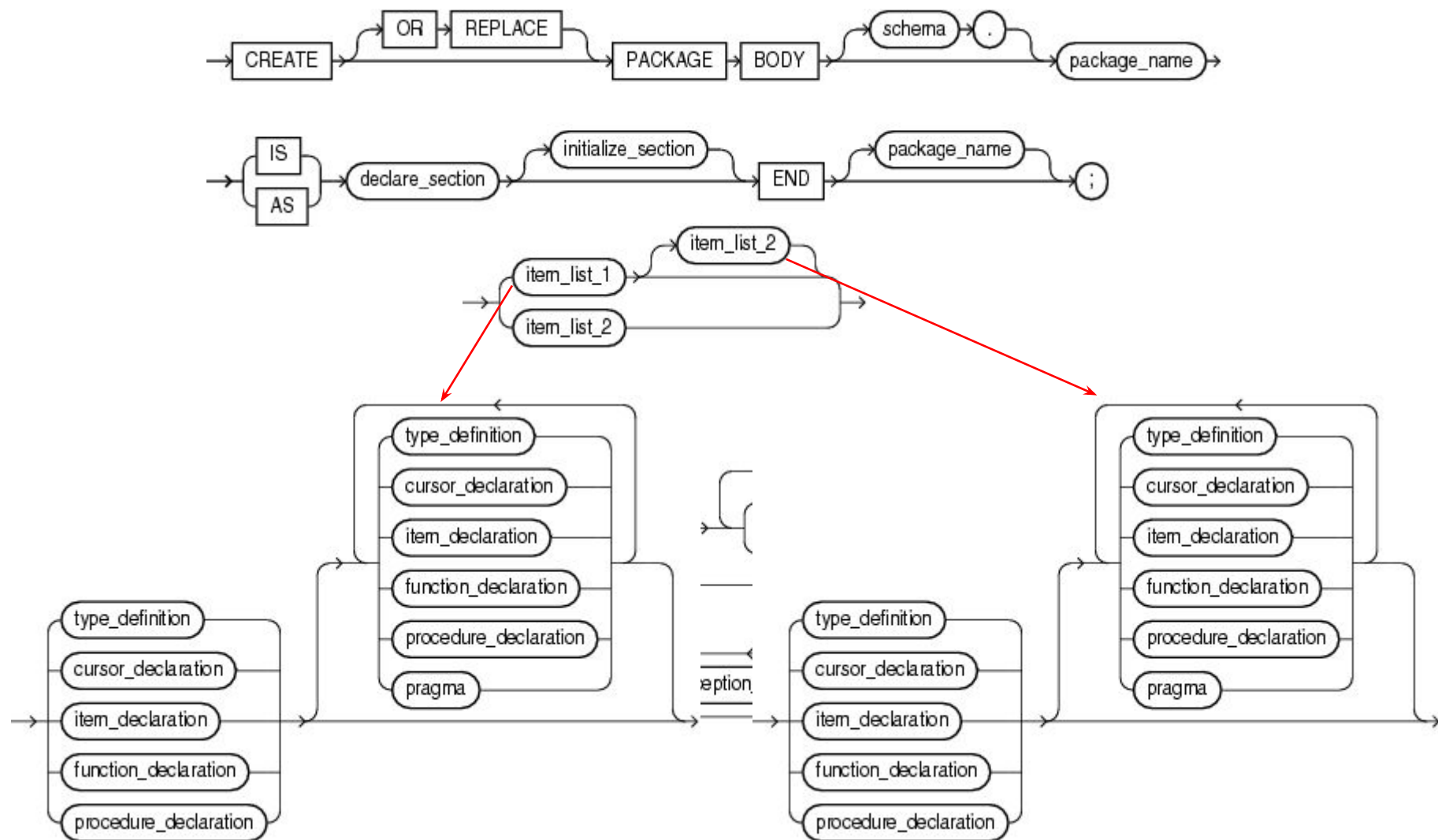
```
2  pkg_test.make_me_happy(500);
3  pkg_test.make_me_happy(100501);
4  end;
5  /
```

Let's start

I can't, sorry :-(

You are happy!

Тело пакета



Перегрузка функций

```
CREATE OR REPLACE PACKAGE pkg_test AS
```

```
    FUNCTION f_overload(p_1 NUMBER) RETURN NUMBER ;  
    FUNCTION f_overload(p_1 VARCHAR2) RETURN VARCHAR2 ;
```

```
end;  
/
```

```
CREATE OR REPLACE
```

```
FUNCTION
```

```
BEGIN
```

```
RETURN
```

```
END;
```

```
FUNCTION
```

```
BEGIN
```

```
RETURN hello, p_1,
```

```
END;
```

```
END;  
/
```

```
SQL> select pkg_test.f_overload(p_1 => 1234) col from dual;
```

```
COL
```

```
-----
```

```
2468
```

```
SQL> select pkg_test.f_overload(p_1 => '1234') col from dual;
```

```
COL
```

```
-----
```

```
Hello, 1234
```

Глобальные переменные в пакетах

```
CREATE OR REPLACE PACKAGE pkg_test IS
```

```
  n INTEGER
```

```
END;
```

```
/
```

```
CREATE OR  
IS
```

```
BEGIN
```

```
  n := 0
```

```
END;
```

```
/
```

Первая сессия

```
SQL> set serverout on;
```

```
SQL> begin
```

```
  2  pkg_test.n:=2;
```

```
  3  dbms_output.put_line(pkg_test.n);
```

```
  4 end;
```

```
  5 /
```

2

Вторая сессия

```
SQL> set serverout on ;
```

```
SQL> begin
```

```
  2  dbms_output.put_line(pkg_test.n);
```

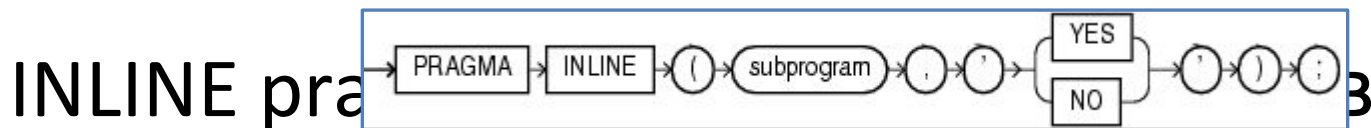
```
  3 end;
```

```
  4 /
```

0

Y pkg_test

INLINE Pragma



подпрограммы должен быть встроен.

Действует непосредственно на

Если подпрограмма перегружена, встраивание будет применимо ко всем подпрограммам с таким именем

PLSQL_OPTIMIZE_LEVEL=2, YES – подпрограмма будет встроена

PLSQL_OPTIMIZE_LEVEL=3, YES – наивысший приоритет для встраивания

Pragma PROCEDURE p1 (x PLS_INTEGER) IS ...

...

Если в **PRAGMA INLINE (p1, 'YES');**

PL/SQL x:= p1(1) + p1(2) + 17; -- These 2 invocations to p1 are **inlined**

...

[Develo](#) x:= p1(3) + p1(4) + 17; -- These 2 invocations to p1 are **not inlined**

ВЫКЛЮ ...

ть
[on](#)

SERIALLY_REUSABLE Pragma

```
CREATE OR REPLACE PACKAGE pkg IS
  n NUMBER := 5;
END pkg;
/

CREATE OR REPLACE PACKAGE sr_pkg IS
  PRAGMA SERIALLY_REUSABLE;
  n NUMBER := 5;
END sr_pkg;
/

BEGIN
  pkg.n := 10;
  sr_pkg.n := 10;
END;
/

BEGIN
  DBMS_OUTPUT.PUT_LINE('pkg.n: ' || pkg.n);
  DBMS_OUTPUT.PUT_LINE('sr_pkg.n: ' || sr_pkg.n);
END;
/
```

Result:

pkg.n: 10
sr_pkg.n: 5

Условная компиляция позволяет вам настраивать функциональность PL/SQL без удаления исходного кода.

- Использование новых возможностей СУБД и отключение этих возможностей, когда приложение работает на более старых версиях СУБД
- Активация отладчика или операторов трассировки в разработческом окружении и скрытие их, когда приложение работает в продуктовой среде

Conditional compilation Pragma

- Preprocessor Control Tokens

Появилас

Идентифицирует код, который обрабатывается перед компиляцией PL/SQL блока

\$plsql_idenfier (\$IF, \$THEN,\$ELSE,\$ELSEIF, \$ERROR)

версии

- Selection Directives

10.1.0.4

Выбирает исходный код для компиляции

- Error Directives

Генерирует пользовательскую ошибку во время компиляции

- Inquiry Di

Предостав

- Static Exp

Выражени

появляться

```
ALTER SESSION SET
PLSQL_CCFlags = 'Some_Flag:1, Some_Flag:2, PLSQL_CCFlags:99'
/
BEGIN
  DBMS_OUTPUT.PUT_LINE($$Some_Flag);
  DBMS_OUTPUT.PUT_LINE($$PLSQL_CCFlags);
END;
/
Result:
2
99
```

е может

Спецификация пакета без тела

```
CREATE OR REPLACE PACKAGE pkg_test IS
    c_package_name constant varchar2(100) := 'PKG_TEST';
    ex_sun_is_shining EXCEPTION;
```

```
END;
```

```
/
```

```
SQL> SELECT pkg_test.c_package_name FROM dual;
ORA-06553: PLS-221: 'C_PACKAGE_NAME' is not a procedure or
is undefined
```

```
2
3    do.object_type
4 FROM dba_objects do
5 WHERE do.owner = USER
6 AND   do.object_name = 'PKG_TEST';
```

OWNER	OBJECT_NAME	OBJECT_TYPE
DBADMIN	PKG_TEST	PACKAGE

Ограничения языка PL/SQL

PL/SQL основан на программном языке ADA. Как результат он использует вариант Descriptive Intermediate Attributed Notation for Ada (DIANA).

Ограничение	Лимит
Количество строк кода на объект	~6 000 000
Bind переменные, которые могут быть переданы в UNIT	32768
Обработчиков исключений на UNIT	65536
Полей в record-е	65536
Уровней вложенности блоков	256
Количество параметров в explicit курсоре, функции и процедуре	65536
Размер идентификатора в символах	30
Размер строки в байтах	32767
Размер триггера	32K

Package Writing Guidelines

- Познакомьтесь с пакетами, которые предоставляет Oracle DataBase и не пишите свои пакеты, которые дублируют эту функциональность (всего 239 пакетов тут [Oracle Database PL/SQL Packages and Types Reference](#)).
- Делайте пакеты такими, чтобы в будущем их можно было переиспользовать.
- Разрабатывайте пакеты и пакеты тела пакета.
 - В спецификации пакета должен быть только код инициализации.
 - Это должен быть код инициализации пакета.
 - При этом код инициализации должен быть простым, а при необходимости – сложным.
 - При описании тела пакета должен быть только код реализации.
 - Код инициализации получается более комплексный и лучше документированный.
 - В секции инициализации всегда можно поймать исключение.
- Объявляйте курсоры в спецификации, а описывайте в теле пакета.

```
CREATE PACKAGE emp_stuff AS
  CURSOR c1 RETURN employees%ROWTYPE; -- Declare cursor
END emp_stuff;

CREATE PACKAGE BODY emp_stuff AS
  CURSOR c1 RETURN employees%ROWTYPE IS
    SELECT * FROM employees WHERE salary > 2500; -- Define cursor
END emp_stuff;
```

PL/SQL Source Text Wrapping

Можно зашифровать следующие типы модулей, чтобы никто не мог посмотреть исходный код:

- Package specification
- Package body
- Type specification
- Type body
- Function
- Procedure

Ограничения:

- Проблемы с обратной совместимостью (нельзя заврапленный в 11-м оракле файл перенести в 10-й)
- Это не безопасный путь для хранения паролей или имен таблиц (юзай [Oracle Database Vault Administrator's Guide](#))
- Исходный код триггера заврапить нельзя, врапим процедуру и ее вызываем в триггере

PL/SQL Source Text Wrapping Guidelines

- Вwrap только тело пакета и не wrap спецификацию, это поможет другим разработчикам посмотреть информацию в спецификации, которая им может понадобиться при разработке
- Вwrap файлы в самом конце разработки, ты не сможешь редактировать заwrapленный файл. Если надо внести правки, правь оригинальный файл и потом wrap его снова
- Перед дистрибуцией заwrapленного файла посмотри его в текстовом редакторе и убедись, что все важные части заwrapлены

Dependencies (Зависимости)



Dependencies (Зависимости)

- Если определение объекта А ссылается на объект В, то А зависит от В.
- А - **dependent object** (зависимый объект) от В
- В - **referenced object** (объект, на который ссылается) А
- (USER/ALL/DBA)_DEPENDENCIES – описание зависимостей между объектами

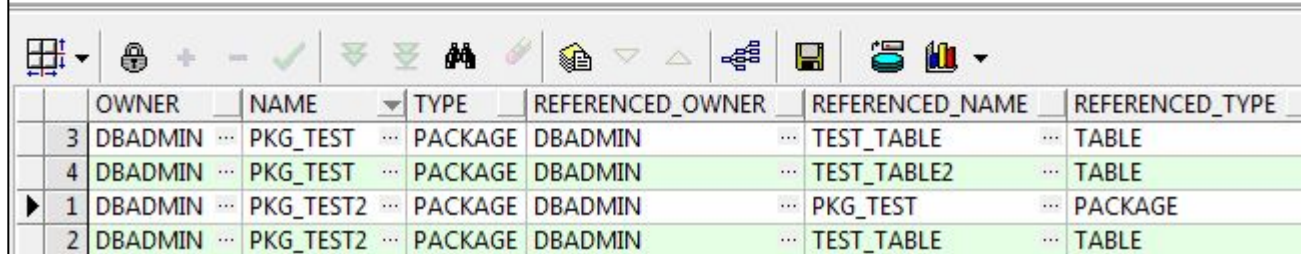
Dependencies (Зависимости)

- А зависит от В, В зависит от С, то А **direct dependent**(прямо зависит) от В, А **indirect dependent**(косвенно зависит) of С.
- Если изменение С делает невалидным В, В делает невалидным А. Это называется **cascading invalidation**.
- **Coarse-grained invalidation**
(инвалидируются все зависимости) и **fine-grained invalidation** (в зависимости операций, ссылка в конце презентации)

Dependencies (Зависимости)

```
create table test_t  
create table test_t
```

```
SELECT t.owner,  
       t.name,  
       t.type,  
       t.referenced_owner,  
       t.referenced_name,  
       t.referenced_type  
FROM   dba_dependencies t  
WHERE  t.owner = USER  
AND    t.name in ('PKG_TEST', 'PKG_TEST2');
```



	OWNER	NAME	TYPE	REFERENCED_OWNER	REFERENCED_NAME	REFERENCED_TYPE
3	DBADMIN	PKG_TEST	PACKAGE	DBADMIN	TEST_TABLE	TABLE
4	DBADMIN	PKG_TEST	PACKAGE	DBADMIN	TEST_TABLE2	TABLE
1	DBADMIN	PKG_TEST2	PACKAGE	DBADMIN	PKG_TEST	PACKAGE
2	DBADMIN	PKG_TEST2	PACKAGE	DBADMIN	TEST_TABLE	TABLE

```
CREATE OR REPLACE PACKAGE pkg_test AS
```

```
    t_1 test_table%ROWTYPE;
```

```
    t_2 test_table2%rowtype;
```

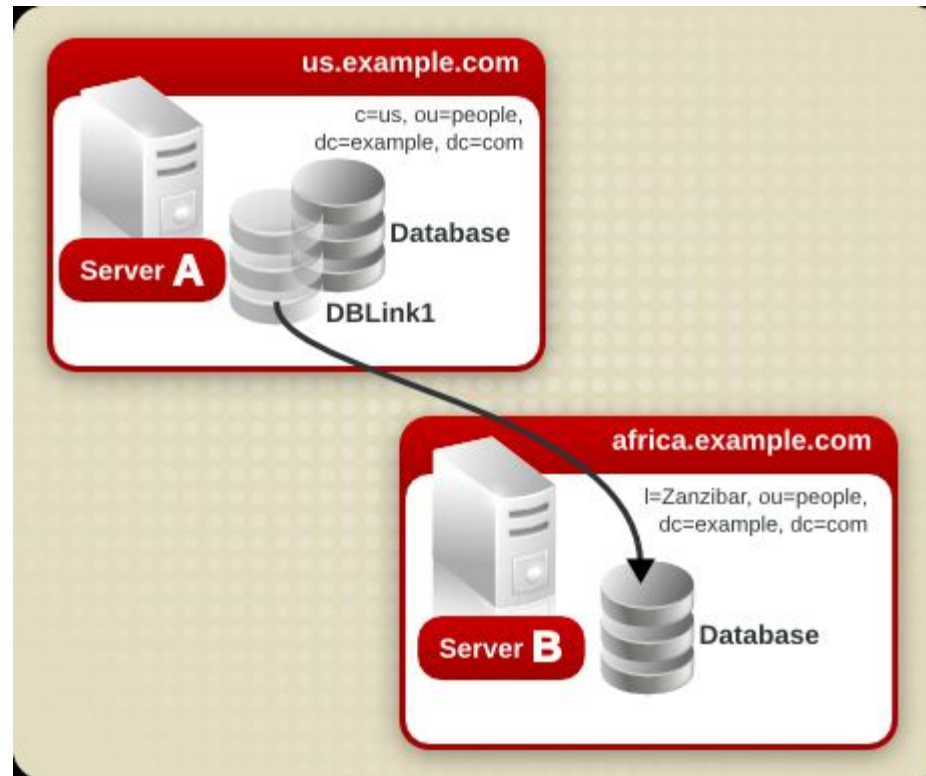
```
END;
```

```
CREATE OR REPLACE PACKAGE pkg_test2 AS
```

```
    t_1 test_table%ROWTYPE := pkg_test.t_1;
```

```
END;
```

DBlink



Запуск удаленных процедур (через dblink)

```
SQL> select sysdate@DB_LINK_TEST from dual;  
SYSDATE
```

06.11.2015

```
SQL> select get_test@DB_LINK_TEST from dual;  
GET_TEST
```

Hello world

Dbms_Job

Job предназначен для периодического запуска задач.

Указываем, *что, когда и с каким интервалом* запускать.

Информация о джобах - DBA_JOBS

О выполняющихся джобах - DBA_JOBS_RUNNING

Создание/управление джобами - DBMS_JOB

Особенность:

- Транзакционный
- Нет имени, номер задать нельзя, это выходной параметр

Jobs

```
DBMS_JOB.SUBMIT (  
    job          OUT BINARY_INTEGER,  
    what         IN   VARCHAR2 ,  
    next_date    IN   DATE DEFAULT sysdate,  
    interval     IN   VARCHAR2 DEFAULT 'null',  
    no_parse     IN   BOOLEAN DEFAULT FALSE,  
    instance     IN   BINARY_INTEGER DEFAULT any_instance,  
    force        IN   BOOLEAN DEFAULT FALSE);
```

Parameter	Description
job	Номер джоба
what	PL/SQL-процедура для запуска (или PL/SQL блок или даже insert 😊)
next_date	Следующая дата запуска джоба
interval	Выражение, возвращающее тип Date следующего запуска джоба. Должно вычисляться для каждого будущего запуска или быть null.
no_parse	Если false – Oracle парсит what джоба, если true – оракл будет парсить what при первом запуске джоба. Зачем это? Например, создать джоб до создания таблицы, с которой работает джоб.
instance	Instance на котором должен запуститься джоб
force	Если true – любой положительный номер instance-а подходит, если false, то указанный instance должен быть запущен, иначе вы получите исключение.

Scheduler

DBMS_JOB считается устаревшей технологией и Oracle рекомендует использовать более мощный Scheduler (с 10g):

- Логирование запуска джобов
- Простой и мощный синтаксис (проще, но гораздо мощнее cron-a)
- Запуск джобов за пределами БД, в ОС
- Управление ресурсами между различными классами джоба
- Возможность просмотра и графического управлений (Oracle SQL developer)
- Возможность использования ресурсного плана (например, не запускать джоб при загрузке процессора более 70%)
- Возможность указать максимальную длительность работы
- Возможность указать последовательно задачи, которые надо выполнить, проверку результатов выполнения задач и ветвление

Scheduler

Информация о джобах шедуллера - DBA_SCHEDULER_JOBS

О выполняющихся джобах - DBA_SCHEDULER_RUNNING_JOBS

Создание/управление джобами - DBMS_SCHEDULER

Пример интерпретации

'FREQ=DAILY; BYDAY=MON,TUE,WED,THU,FRI,SAT,SUN;

'FREQ=MINUTELY; BYHOUR=22;'

'FREQ=HOURLY; BYDAY=MON,TUE,WED,THU,FRI,SAT,SUN;

'FREQ=DAILY; BYDAY=MON,TUE,WED,THU,FRI,SAT,SUN;

BYHOUR=22;'

```
BEGIN
  DBMS_SCHEDULER.CREATE_JOB (
    job_name      => 'my_java_job',
    job_type      => 'EXECUTABLE',
    job_action     => '/usr/bin/java myClass',
    repeat_interval => 'FREQ=MINUTELY',
    enabled        => TRUE
  );
END;
```

Scheduler

```
BEGIN
  DBMS_SCHEDULER.CREATE_JOB(
    job_name          => 'pavlik_job',
    job_type          => 'PLSQL_BLOCK',
    job_action        => 'BEGIN INSERT INTO employees VALUES (7935,
''SALLY'', ''DOGAN'', ''sally.dogan@examplecorp.com'', NULL,
SYSDATE, ''AD_PRES'', NULL, NULL, NULL, NULL); END;',
    start_date        => SYSDATE,
    repeat_interval    => 'FREQ = DAILY; INTERVAL = 1');
END;
/

DBMS_SCHEDULER.DROP_JOB('pavlik_job');
```

Scheduler. Периодичность

```
requery_clause = "FREQ" "=" "YEARLY" | "MONTHLY" | "WEEKLY" |  
"DAILY" | "HOURLY" | "MINUTELY" | "SECONDLY"
```

```
interval_clause = "INTERVAL" "=" intervalnum
```

```
intervalnum = 1 through 999
```

```
bymonth_clause = "BYMONTH" "=" monthlist
```

```
monthlist = monthday ( "," monthday)*
```

```
month = numeric_month | char_month
```

```
numeric_month = 1 | 2 | 3 ... 12
```

```
char_month = "JAN" | "FEB" | "MAR" | "APR" | "MAY" | "JUN" |
```

```
"JUL" | "AUG" | "SEP" | "OCT" | "NOV" | "DEC"
```

```
byweekno_clause = "BYWEEKNO" "=" weeknumber_list - 1 through 53
```

```
byyearday_clause = "BYYEARDAY" "=" yearday_list - 1 through 366
```

```
bymonthday_clause = "BYMONTHDAY" "=" monthday_list - 1 through 31
```

```
byday_clause = "BYDAY" "=" byday_list - day = "MON" | "TUE" | "WED"...
```

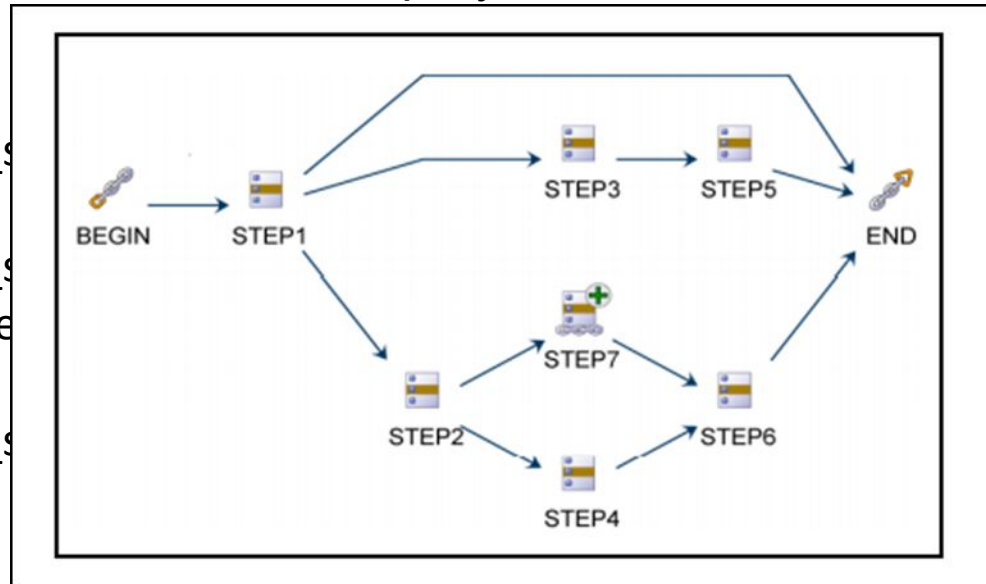
```
byhour_clause = "BYHOUR" "=" hour_list - hour = 0 through 23
```

```
byminute_clause = "BYMINUTE" "=" minute_list - minute = 0 through 59
```

```
bysecond_clause = "BYSECOND" "=" second_list - second = 0 through 59
```

Scheduler. Chain-ы

Chain(цепочка) – именованная последовательность задач связанная вместе. Chain-ы являются средством, с помощью которого можно реализовать зависимости между задачами, в которых задачи стартуют в зависимости от результата одной или более предыдущих задач.



```
sys.dbms_scheduler.define_chain_rule(chain_name=>
```

```
...
```

```
sys.dbms_scheduler.define_chain_rule(chain_name=>
```

```
step_name=>
```

```
...
```

```
sys.dbms_scheduler.define_chain_rule(chain_name=>
```

```
CHAIN_MAIN',
```

```
SCHEMA.CHAIN_MAIN',
```

```
SCHEMA.CHAIN_MAIN',
```

```
action => 'START "DO_DATA","PREP_AND_DO_START"
```

```
...
```

```
sys.dbms_scheduler.define_chain_rule(chain_name=> 'SCHEMA.CHAIN_MAIN',
```

```
rule_name => 'SCHEMA.RULE_2_1',
```

```
condition => 'PREP_AND_DO_START COMPLETED',
```

```
....
```

```
sys.dbms_scheduler.enable(name => 'SCHEMA.CHAIN_MAIN');
```

Summarizing

Пакеты – инкапсуляция логически связанного кода в единое целое

Перегрузка процедур и функций как добавление нового функционала без изменения старого

Глобальные переменные пакеты - средство обмена данным между разными исполняемыми блоками кода

INLINE Pragma - указывает, что вызов подпрограммы должен быть встроен. Действует непосредственно на последующее описание или оператор, и на некоторые типы операторов

SERIALLY_REUSABLE Pragma - указывает, что состояние пакета необходимо только для одного серверного вызова, после этого вызова место для хранения переменных пакетов может быть использовано повторно, уменьшая перерасход памяти для длительных операций.

Conditional compilation Pragma - условная компиляция

Package Writing Guidelines – рекомендации оракл по написани кода пакетов

PL/SQL Source Text Wrapping – шифрование кода

Спецификация без тела - это хранение констант, исключений и типов и т.д.

Dependency – зависимости БД

DB Link – средство исполнения процедур и функций на удаленных БД

DBMS_JOB – простой средство эпизодического/периодического запуска заданий

Scheduler – новое, мощное средство БД (с 10g) для запуска заданий как БД так и ОС

Список использованных материалов

- [Функция](#)
- [Процедура](#)
- [Спецификация пакета](#)
- [Тело пакета](#)
- [PL/SQL Блок](#)
- [INLINE Pragma](#)
- [SERIALLY_REUSABLE Pragma](#)
- [Conditional Compilation](#)
- [Package Writing Guidelines](#)
- [PL/SQL Source Text Wrapping](#)
- [Compiling PL/SQL Units for Native Execution](#)
- [Dependency](#)
- [DBLink](#)
- [DBMS_JOB](#)
- [SCHEDULER](#)
- [PL/SQL Language Fundamentals](#)
- [PL/SQL limits](#)