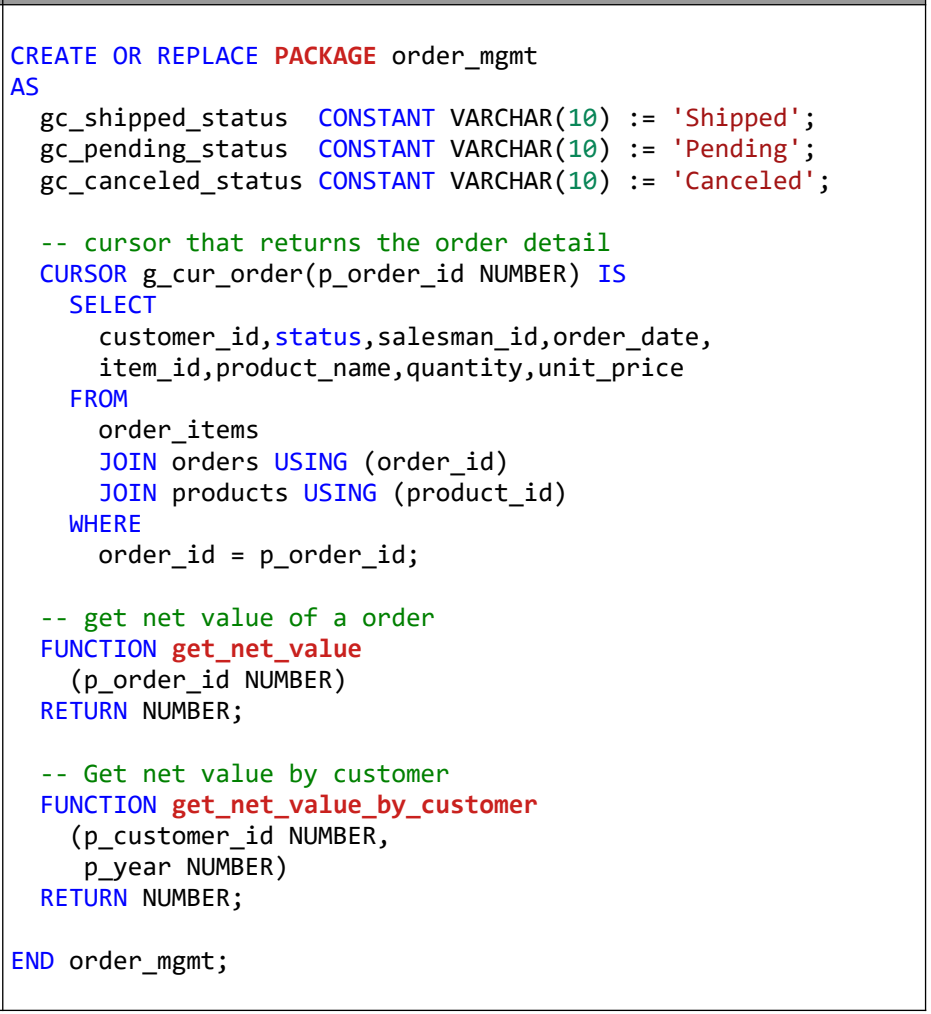
Packages

Definició paquets	Exemples
Un paquet té dues part: • Especificació (PACKAGE) • Cos del paquet (PACKAGE BODY) A **l'especificació** del paquet es declaren els elements públics , aquells que podran ser invocats des d'altres objectes o codi pl/sql. Això vol dir que els elements declarats en l'especificació del paquet es poden accedir des de qualsevol lloc de l'esquema on s'ha creat el paquet, per exemple des d'un altre paquet. El paquet d'especificació no conté la implementació dels elements públics. Per exemple, en el cas de procediments o funcions a l'especificació només es defineix la capçalera , el nom i paràmetres de la funció, però NO el seu cos. El paquet d'especificació **PACKAGE** pot existir independentment sense estar vinculat a un PACKAGE BODY, si cap dels elements que el component necessita implementació, per exemple un paquet no només de defineixen constants. Els elements típics d'una paquet d'especificació són: • Procedures • Functions • Cursors • Types, variables, and constants	 ``` CREATE OR REPLACE PACKAGE order_mgmt AS gc_shipped_status CONSTANT VARCHAR(10) := 'Shipped'; gc_pending_status CONSTANT VARCHAR(10) := 'Pending'; gc_canceled_status CONSTANT VARCHAR(10) := 'Canceled'; -- cursor that returns the order detail CURSOR g_cur_order(p_order_id NUMBER) IS SELECT customer_id,status,salesman_id,order_date, item_id,product_name,quantity,unit_price FROM order_items JOIN orders USING (order_id) JOIN products USING (product_id) WHERE order_id = p_order_id; -- get net value of a order FUNCTION get_net_value (p_order_id NUMBER) RETURN NUMBER; -- Get net value by customer FUNCTION get_net_value_by_customer (p_customer_id NUMBER, p_year NUMBER) RETURN NUMBER; END order_mgmt; ```

Definició paquets

Cada cursor o programa(procedure or function) declarat en l'especificació del paquet ha de tenir la seva correspondència en el cos del paquet **PACKAGE BODY**.

A banda de la implementació dels elements declarats en l'especificació, un **PACKAGE BODY** pot tenir elements propis, privats, es a dir només es poden fer servir dins del BODY

La sintaxis de creació del **PACKAGE BODY** és la següent:

```
CREATE [OR REPLACE] PACKAGE BODY
[schema_name.]<package_name> IS
    declarations
    implementations;
[BEGIN
EXCEPTION]
END <package_name>;
```

A package body can have an initialization part which consists of statements that initialize public variables and do other one-time setup tasks. The initialization part only runs once at the first time the package is referenced. It can also include an exception handler.

Exemples de com podem cridar les funcions i elements del paquet anterior:

1. `SELECT order_mgmt.get_net_value_by_customer(1,2017) FROM dual;`
2.
```
BEGIN
  DBMS_OUTPUT.PUT_LINE(order_mgmt.gc_shipped_status);
END;
```

Exemples

```
CREATE OR REPLACE PACKAGE BODY order_mgmt AS
  -- get net value of a order
  FUNCTION get_net_value(p_order_id NUMBER)
  RETURN NUMBER   IS
    ln_net_value NUMBER
  BEGIN
        SELECT SUM(unit_price * quantity)
        INTO ln_net_value
        FROM order_items
        WHERE order_id = p_order_id;
        RETURN p_order_id;
  EXCEPTION
    WHEN no_data_found THEN
            DBMS_OUTPUT.PUT_LINE( SQLERRM );
  END get_net_value;
-- Get net value by customer
  FUNCTION get_net_value_by_customer
    (p_customer_id NUMBER, p_year NUMBER)
  RETURN NUMBER IS
    ln_net_value NUMBER
  BEGIN
        SELECT SUM(quantity * unit_price)
      INTO ln_net_value
        FROM order_items
        JOIN orders USING (order_id)
        WHERE extract(YEAR FROM order_date) = p_year
          AND customer_id  = p_customer_id
          AND status   = gc_shipped_status;
        RETURN ln_net_value;
  EXCEPTION
    WHEN no_data_found THEN
            DBMS_OUTPUT.PUT_LINE( SQLERRM );
  END get_net_value_by_customer;
END order_mgmt;
```

Definició paquets	Exemples
Un altre exemple de PACKAGE i PACKAGE BODY `CREATE OR REPLACE PACKAGE pck_get_set IS` `PROCEDURE set_record (p_emp_rec IN emp%ROWTYPE);` `FUNCTION get record (p_emp no IN NUMBER)` `RETURN emp%ROWTYPE;` `END pck_get_set;` Aquest és un exemple de codi per cridar als procediments i funcions del paquet que hem creat `DECLARE` `l_emp_rec emp%ROWTYPE;` `l_get_rec emp%ROWTYPE;` `BEGIN` `--inserir l'empleat 1004` `l_emp_rec.emp_no:=1004;` `l_emp_rec.emp_name:='CCC';` `l_emp_rec.salary~20000;` `l_emp_rec.manager:='BBB';` `pck_get_set.set_record(1_emp_rec);` `--obtenir les dades de l'empleat 1004` `l_get_rec:=pck_get_set.get_record(1004);` `dbms_output.put_line` `('Employee name: '||l_get_rec.emp_name);` `END;`	Aquest és un exemple del body del paquet , **pck_get_set ,** especificat a l'esquerre `CREATE OR REPLACE PACKAGE BODY pck_get_set IS` `PROCEDURE set_record(p_emp_rec IN emp%ROWTYPE) IS` `BEGIN` `INSERT INTO emp VALUES` `(p_emp_rec.emp_name,p_emp_rec.emp_no,` `p_emp_rec.salary,p_emp_rec.manager);` `COMMIT;` `END set_record;` `FUNCTION get_record(p_emp_no IN NUMBER)` `RETURN emp%ROWTYPE IS` `l_emp_rec emp%ROWTYPE;` `BEGIN` `SELECT * INTO l_emp_rec FROM emp` `WHERE emp_no=p_emp_no` `RETURN l_emp_rec;` `END get_record;` `END pck_get_set;`

Més conceptes i informació sobre paquets

OVERLOADING A PROCEDURE	
There can be multiple subprograms within a package having similar names. This feature is useful if we want to have homogenous parameters with heterogeneous data types. The concept of overloading within the package allows the programmers to mention clearly the type of action they want to perform. **Coding Implementation with procedure overloading. (Package created)** `CREATE PACKAGE overloadingprocedure AS` `Procedure overl_method (p varchar2);` `Procedure overl_method (numbr number);` `END overloadingprocedure;` **Coding Implementation with procedure overloading. (Package body created)** `CREATE OR REPLACE PACKAGE BODY overloadingprocedure AS` `--procedure implemented` `Procedure overl_method (p varchar2) AS` `BEGIN` `DBMS_OUTPUT.PUT_LINE ('First Procedure: ' || p);` `END;` `--procedure implemented` `Procedure overl_method (numbr number) AS` `BEGIN` `DBMS_OUTPUT.PUT_LINE ('Second Procedure: ' || numbr);` `END;` `END;`	**Els procediments del paquet creat s'invocarien de la següent manera:** `BEGIN` `overloadingprocedure.overl_method ('Software Testing Help');` `overloadingprocedure.overl_method (1);` `END;`

Package Information In PL/SQL	PLSQL Package Dependency
All the relevant details like the source of the package, subprograms, and overloaded items are stored in data definition tables after a package is created. **The list of the data definition tables are as follows:** • **USER_PROCEDURES:** This table contains subprogram information like the overloaded items, object_id, and so on for the current user. • **ALL_PROCEDURES:** This table contains subprogram information like the overloaded items, object_id, and so on for all the users. • **USER_SOURCE:** This table contains the information on the object source for the current user. • **ALL_SOURCE:** This table contains the information on the object source for all the users. • **ALL_OBJECT:** This table contains the information on the package like the creation_date, object_id, and other object detail for all the users.	**The package dependencies in PL/SQL are listed below:** • A package specification is an independent identity. • Package body is reliant on the package specification. • A package body can only be compiled separately. However, if a package specification is compiled then the body needs to be compiled again. • Function or a procedure inside a package body that depends on the private elements should be implemented post declaration of the private elements.

Webgrafia

Enllaços web	
...packages-pl-sql.html	Oracle PL/SQL Package: Type, Specification, Body
...testinghelp.com/pl-sql-packages/	PL SQL Package: Oracle PL/SQL Package Tutorial With Examples