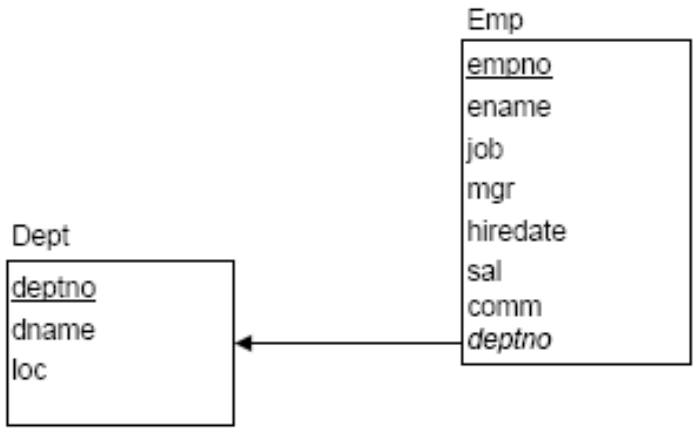
Cada alumno matriculado en la asignatura dispone de un usuario ORACLE cuyo identificador coincide con el de su usuario en la red upvnet (la contraseña coincide con el identificador de usuario).
La práctica propone una serie de ejercicios para los que se utilizará un esquema de ejemplo cuyo script sql de creación y carga está en el archivo “Esquema Empresa”




```
CREATE TABLE DEPT (
        DEPTNO NUMBER(2) NOT NULL,
        DNAME VARCHAR2(14),
        LOC VARCHAR2(13),
        CONSTRAINT DEPT_PRIMARY_KEY PRIMARY KEY (DEPTNO));
CREATE TABLE EMP (
        EMPNO NUMBER(4) NOT NULL,
        ENAME VARCHAR2(10),
        JOB VARCHAR2(9),
        MGR NUMBER(4) CONSTRAINT EMP_MGR_FK REFERENCES EMP (EMPNO),
        HIREDATE DATE,
        SAL NUMBER(7,2),
        COMM NUMBER(7,2),
        DEPTNO NUMBER(2) NOT NULL,
        CONSTRAINT EMP_DEPTNO_FK FOREIGN KEY (DEPTNO) REFERENCES DEPT
        (DEPTNO),
        CONSTRAINT EMP_EMPNO_PK PRIMARY KEY (EMPNO));
```

EJERCICIOS

1. Crea la BD Empresa
2. Crea un bloque PL/SQL que incremente el salario de todos los empleados en un 10% del salario medio de la empresa. Ten en cuenta:
   - La estructura de bloque de PL/SQL
   - La declaración de variables
   - Los tipos predefinidos
   - Un bloque PL/SQL puede contener varias sentencias SQL seguidas.
3. Crear un bloque PL/SQL que visualice la diferencia del salario medio de los empleados de los departamentos ACCOUNTING y SALES.
La complejidad y conocimientos son similares a los del ejemplo anterior. Utilice la sentencia print para mostrar el resultado como aparece en la transparencia 45 de BDV-SABD-P1 El lenguaje PLSQL de Oracle Transparencias
4. Crear un bloque PL/SQL que actualice el salario de cada empleados al salario medio de su departamento. Esta actualización se realizará sólo para

aquellos empleados cuyo salario actual sea inferior al salario medio de su departamento.
Este ejercicio debe realizarse con cursores utilizando estructuras de control de programación.
Se definirá un cursor que contendrá lso departamento y su salario medio y se recorrerá este cursor mediante un bucle FOR para actualizar el salario de los empleados que corresponda de cada departamento. Utilizaremos:

- cursores
- estructuras de control

5. Crear un bloque PL/SQL en el que se almacene en una tabla local los nombres de los empleados.
Visualizar posteriormente el contenido de dicha tabla.
Para este ejercicio se declarará una variable estructurada de tipo tabla.
Se declarará un cursor que contendrá a todos los empleados. Mediante un bucle FOR se recorrerán todos los empleados y se irán insertando (el nombre) en la variable de tipo tabla.
Por último se utilizará otro bucle para imprimir los nombres. Se utilizará la sentencia dbms_output.put_line(). Utilizaremos:

- tipos estructurados

SESIÓN 2 PROCEDIMIENTOS, FUNCIONES, PACKAGES
En esta práctica se va a trabajar con bloques PL/SQL pero nominados. La forma de trabajo será similar a la de la
práctica anterior pero en este caso acabaremos almacenando el bloque pl/sql e invocándolo después.
Lo más práctico es desarrollar primero los bloques pl/sql como anónimos, como los que se hicieron en la
primera sesión, y una vez los hayamos depurado los guardamos con la nominación apropiada.
Luego desde el mismo sqlworksheet podemos invocar los bloques.
La práctica propone una serie de ejercicios para los que se utilizará el mismo esquema de la sesión anterior

EJERCICIOS
1. Escribir un procedimiento con un parámetro de entrada del tipo del atributo salario, que muestre los
empleados que tienen dicho salario.
• En las transparencias 1-141 en adelante están definidas las procedures.
• Mire el ejemplo de la transparencia 1-147 para saber cómo se define un procedimiento con un
parámetro de entrada y cómo se invoca
• Use dbms_output.put_line para mostrar los resultados desde el bloque pl/sql. Utilice la
concatenación de strings si quiere hacer un mensaje complejo (string1||string2)
• Utilice un cursor para recorrer los empleados.
• Declare una variable de tipo registro empleado para usarla en la parte INTO de la sentencia
FETCH. Mire el ejemplo de la transparencia 1-111 para ver la declaración de una variable registro y

su uso en el FETCH.
2. Escribir un procedimiento que devuelva en el primer parámetro el nombre del empleado que tiene un
salario idéntico al valor recibido en el segundo parámetro. El procedimiento deberá cumplir los siguientes
puntos:
• Si el valor pasado no corresponde al salario de ningún empleado, se debe tratar la excepción
mostrando el error “Ningún empleado tienen un salario de <valor dado>”.
• Si el valor dado corresponde al salario de más de un empleado, se debe tratar la excepción mostrando
el error “Más de un empleado posee un salario de <valor dado>”.
• Si el valor pasado corresponde al salario de un único empleado, se debe devolver el nombre de dicho
empleado.
• Tratar cualquier otra excepción con mostrando el mensaje “Ocurrió algún error imprevisto”
• En las transparencias 1-141 en adelante están definidas las procedures.
• Mire el ejemplo de la transparencia 1-150 para saber cómo se define un procedimiento con un
parámetro de salida, cómo se invoca y como se ve su valor usando variable de entorno.
• En este procedimiento deberá utilizar la parte de excepciones. Recuerde que un bloque pl/sql
puede tener una sección para tratar excepciones. Las excepciones se pueden ver a partir de la
página 1-122
• Para este ejercicio usaremos el tratamiento de excepciones de Oracle. En la transparencia 1-127
tiene un resumen de excepciones implícitas que el propio Oracle dispara.
• Usted tendrá que capturar y tratar la excepción tal como indica la transparencia 1-129. Utilice
dbms_output.put_line para presentar los errores.
3. Escribir una función que devuelva la media del salario de un departamento cuyo número recibe como
parámetro. Probar su funcionamiento en la siguiente consulta: “obtener el nombre de los empleados y la
diferencia de su salario con el salario medio de su departamento”. Si el valor del número de departamento
no existe deberá lanzarse una excepción definida por el usuario con número de error 20202 y el mensaje
‘Departamento inexistente’.
• En las transparencias 1-160 en adelante están definidas las funciones.
• La captura de excepciones , en este caso, se hará utilizando el raise_application_error descrito en
la transparencia 1-137. Vea el ejemplo de la transparencia 1-139.
4. Crear un paquete que contenga un procedimiento denominado test_dep_job al que se le pasa un número
de departamento y un trabajo (job) y que verifique si existe un empleado en ese departamento con ese

trabajo. Utilizar:
• Una tabla indexada para almacenar las combinaciones válidas de departamento y trabajo.
• La tabla sólo se debe poblar una vez en la sesión.
• Lanzar un error de aplicación con el mensaje adecuado si la combinación no es válida.
Comprobar si el trabajo ‘CLERK’ es válido en el departamento 20 y lo mismo en el departamento 40.
• Un paquete es un objeto del esquema que agrupa lógicamente variables constantes tipos de datos
procedimientos o funciones.
• El paquete tiene dos partes especificación (T-1-181) y cuerpo (T-1-183)
• Para que la tabla solo se pueble una vez no tiene que estar contenida en el procedimiento sino
como código del propio paquete que se iniciará con un BEGIN
• En la trasnparencia 1-191 tiene un ejemplo de invocación.