UT 3.1: DISEÑO FÍSICO RELACIONAL
(Creación , Modificación y Borrado de tablas en ORACLE)

Cómo cambiar de una PDB a otro.

ALTER SESSION SET CONTAINER=XEPDB1;

Cómo cambiar de una PDB al contenedor.

ALTER SESSION SET CONTAINER=CDB$ROOT;

Cómo conectar SYSTEM a una PDB llamada XEPDB1.

CONNECT SYSTEM/contraseña@LOCALHOST:1521/XEPDB1

Cómo conectar SYSTEM al contenedor raíz de XE (CDB$ROOT).

CONNECT SYSTEM/contraseña@LOCALHOST:1521/XE

Intérprete de comandos SQL*Plus:

sqlplus [{ usuario**[/**password**>][@<**cadena_conexión**>]**

Ejecución de script:

start fichero

@ fichero

Visualiza la descripción de una tabla:

DESC[RIBE] [esquema**.]**nombre_tabla**;**

SENTENCIAS DE CREACIÓN, MODIFICACIÓN, ELIMINACIÓN DE TABLAS

CREATE TABLE [esquema.]Nombre_tabla
[(Definición_columna [,Definición_columna] ...
[Constraint_tabla]
) [Opciones_ tabla];

- **Definición_columna:**

nombre_columna tipo_datos **GENERATED BY DEFAULT AS IDENTITY**
[{ CONSTRAINT [nombre_constraint**]]**
{NOT NULL | NULL}|
[DEFAULTdefault_value**]|**
[UNIQUE [KEY] |
[PRIMARY KEY]|
[REFERENCES nombre_tabla(columna)**][ON DELETE CASCADE/SET NULL]]**
CHECK (expresion**)**
}]...

- **GENERATED BY DEFAULT AS IDENTITY:** genera un nuevo entero para la columna cada vez que se inserte una nueva fila en la tabla
- **Constraint_de_tabla:**

[CONSTRAINT [nombre_constraint**]]**
{PRIMARY KEY(nombre_columna**,...)|**
UNIQUE(nombre_columna**,...)|**
FOREIGN KEY(nombre_columna1**,...) REFERENCES** nombre_tabla1**(**nombre_columna2**,...)**
[ON DELETE CASCADE | SET NULL}]
CHECK (expresion**)}**

- **TO_DATE(**'23/05/2010','DD/MM/YYYY'**)**
- **Opciones_Tabla**:

  TABLESPACE nombre_tablespace
  STORAGE (INITIAL valor_inicial **NEXT** valor_siguiente
  MINEXTENTS minimo
  MAXEXTENTS {maximo**/UNLIMITED }**
  PCTINCREASE incremento**)**

- En Oracle, se pueden consultar las vistas: **USER_TABLES, DBA_TABLES Y ALL_TABLES.**

ALTER TABLE nombre_tabla especificación_alter **;**

especificación_alter:

{ADD definición_columna
|ADD (definición_columna**, ...)**
|ADD [CONSTRAINT [símbolo**]] PRIMARY KEY (**nombre_columna**, ...)**
|ADD [CONSTRAINT [símbolo**]] UNIQUE (**nombre_columna**, ...)**
|ADD [CONSTRAINT [símbolo**]] FOREIGN KEY (**nombre_columna, ... **)**
REFERENCES nombre_tabla**(**nombre_columna,...**)**
|ADD [CONSTRAINT [símbolo**]] CHECK (**condición)
|RENAME COLUMN anterior_nombre_columna **TO** nuevo_nombre_columna
|MODIFY definición_columna
|DROP COLUMN nombre_columna
|DROP PRIMARY KEY [CASCADE]
|DROP FOREIGN KEY nombre_constraint
|DROP CONSTRAINT nombre_constraint
| MOVE nombre_tablespace
}

DROP [TEMPORARY] TABLE nombre_tabla **[CASCADE CONSTRAINTS];**

RENAME nombre_tabla **TO** nuevo_nombre_tabla adolece

TIPOS DE DATOS

TIPO DATO	TIPO DATO
VARCHAR2(tamaño)	NUMBER(p,s)
CHAR(tamaño)	DATE

OPERADORES ARITMÉTICOS

Operador	Significado	Ejemplo
+	Suma	A+B
-	Resta	A-B
/	División	A/B
*	Multiplicación	A*B
%	Resto de la división	A%B

OPERADORES DE COMPARACIÓN O RELACIONALES

Operador	Significado
!=, <>	Distinto
> , >=	Mayor , Mayor o igual que
< , <=	Menor, Menor o igual
LIKE	Se utiliza para unir cadenas de caracteres. **%** : Representa cualquier cadena de caracteres de 0 o mas caracteres. **_:** Representa un único carácter cualquiera.
IN	Igual a cualquiera de los miembros entre paréntesis
NOT IN	Distinto a cualquiera de los miembros entre paréntesis
BETWEEN	Contenido en el rango
NOT BETWEEN	Fuera del rango

INFORMACIÓN DE LAS CONSTRAINT

Las siguientes vistas de diccionario de datos tiene información sobre las constraint:

- **USER_CONSTRAINTS** (owner, constraint_name, table_name, constraint_type, ...): vista que contiene información de todas las constraint de un usuario.
- Constraint_type puede ser: C: CHECK,P: PK,R: FK,U: UNIQUE
- **USER_CONS_COLUMNS**: vista que contiene las columnas que forman parte de una constraint.(OWNER,CONSTRAINT_NAME, TABLE_NAME, COLUMN_NAME, POSITION)

ÍNDICES

CREATE [UNIQUE] INDEX nombre_indice **ON** nombre_tabla **(**nombre_col_1 **[ASC|DESC],...);**

ALTER INDEX nombre_indice **REBUILD;**

DROP INDEX nombre_indice**;**

CREACIÓN Y BORRADO DE SECUENCIAS

SECUENCIAS

CREATE SEQUENCE nombre_secuencia

[INCREMENT BY entero**]**

[START WITH entero**]**

[MAXVALUE entero **| NOMAXVALUE]**

[MINVALUE entero **| NOMINVALUE]**

[CYCLE | NOCYCLE]

[ORDER | NOORDER]

[CACHE entero **| NOCACHE];**

DROP SEQUENCE nombre_secuencia**;**

- **N_secuencia.CURRVAL:** devuelve el valor actual de la secuencia.
- **N_secuencia.NEXTVAL:** devuelve el siguiente valor de la secuencia.

UT 3.2: USUARIOS Y PERMISOS EN ORACLE

CREACIÓN, MODIFICACIÓN y BORRADO DE USUARIOS

CREATE USER nombre_usuario
IDENTIFIED BY contraseña
[DEFAULT TABLESPACE nombre_tablespace**]**
[TEMPORARY TABLESPACE nombre_tablespace**]**
[QUOTA {entero **{K|M} | UNLIMITED} ON** nombre_tablespace
[PROFILE perfil**]**
[ACCOUNT LOCK/UNLOCK];

ALTER USER nombre_usuario
IDENTIFIED BY contraseña
[DEFAULT TABLESPACEnombre_tablespace**]**
[TEMPORARY TABLESPACEnombre_tablespace**]**
[QUOTA {entero {K|M} | UNLIMITED} ON nombre_tablespace
[PROFILE perfil**]**
[ACCOUNT UNLOCK/LOCK]
[DEFAULT ROLE nombre_rol1 [,nombre_rol2]...|**ALL**|**NONE] ;**

DROP USER nombre_usuario **[CASCADE];**

SHOW USER

Las tablas del diccionario de datos para saber toda la información de los usuarios son:

USER_USERS
DBA_USERS (todos los usuarios, debe de tener permisos dba para ver esta tabla)

ROLES

CREATE ROLEnombre_rol ;
DROP ROLE nombre_rol**;**

Roles del Sistema	Privilegios
CONNECT	ALTER SESSION, CREATE CLUSTER, CREATE DATABASE LINK, CREATE SEQUENCE, CREATE SESSION, CREATE SYNONYM, CREATE TABLE, CREATE VIEW
RESOURCE	CREATE CLUSTER, CREATE PROCEDURE, CREATE TABLE, CREATE SEQUENCE, CREATE TRIGGER
DBA	TODOS LOS PRIVILEGIOS DEL SISTEMA

PERMISOS SOBRE OBJETOS

GRANT {privilegio1 **[,** privilegio2**]... | ALL [PRIVILEGES]}**
[(columna1 **[,** columna2**]...)]**
ON [usuario**.]**objeto
TO {usuario **|** rol **| PUBLIC} [, {**usuario **|** rol **| PUBLIC} ...]**
[WITH GRANT OPTION];

REVOKE {privilegio1 [, privilegio2]**... | ALL [PRIVILEGES]**
ON [usuario.]objeto
FROM {usuario **|** rol **| PUBLIC} [, {**usuario **|** rol **| PUBLIC} ...];**

	Tabla	Vista	Secuencia	Procedimiento
ALTER	X		X	
SELECT	X	X		
INSERT	X	X		
DELETE	X	X		
UPDATE	X	X		
EXECUTE				X
INDEX	X			
REFERENCES	X(FK)			
ALL				

PERMISOS DEL SISTEMA

GRANT {privilegio | rol} [, {privilegio | rol},...]
TO {usuario | rol | PUBLIC} [, {usuario | rol | PUBLIC} ...]
[WITH ADMIN OPTION];

REVOKE {privilegio **|** rol**} [, {**privilegio **|** rol**},...]**
FROM {usuario **|** rol **| PUBLIC} [, {**usuario **|** rol **| PUBLIC}..];**

Privilegio	Descripción
ANALYZE	
ANALIZE ANY	Analiza cualquier tabla o índice de la base de datos
AUDIT	
AUDIT ANY	Auditar cualquier objeto de la base de datos
AUDIT SYSTEM	Activar y desactivar la auditoría
DATABASE	
ALTER DATABASE	Modificar la base de datos. Añadir ficheros a los tablespaces, siempre que se tenga el privilegio del sistema operativo.
DATABASE LINK	
CREATE DATABASE LINK	Crear enlaces de base de datos privados en el esquema propio
INDEX	
CREATE ANY INDEX	Crear índices sobre cualquier tabla en cualquier esquema
ALTER ANY INDEX	Modificar índices sobre cualquier tabla en cualquier esquema
DROP ANY INDEX	Borrar índices sobre cualquier tabla en cualquier esquema
LIBRARY	
CREATE LIBRARY	Crear librerías externas en el esquema propio
CREATE ANY LIBRARY	Crear librerías externas en cualquier esquema
DROP LIBRARY	Eliminar librerías externas en el esquema propio
DROP ANY LIBRARY	Eliminar Crear librerías externas en cualquier esquema
PRIVILEGE	
GRANT ANY PRIVILEGE	Conceder cualquier privilegio del sistema, pero no sobre objetos
PROCEDURE	
CREATE PROCEDURE	Crear un procedimiento en el esquema propio
CREATE ANY PROCEDURE	Crear procedimientos bajo cualquier esquema
ALTER ANY PROCEDURE	Modificar cualquier procedimiento de la base de datos
DROP ANY PROCEDURE	Borrar cualquier procedimiento de la base de datos
EXECUTE ANY PROCEDURE	Ejecutar cualquier procedimiento de la base de datos
PROFILE	
CREATE PROFILE	Crear perfiles
ALTER PROFILE	Modificar cualquier perfil de la base de datos
DROP PROFILE	Borrar cualquier perfil de la base de datos
ALTER RESOURCE COST	Calcular los costes de recursos utilizados en todas las sesiones de los usuarios
PUBLIC DATABASE LINK	
CREATE PUBLIC DATABASE LINK	Crear enlaces de base de datos públicos
DROP PUBLIC DATABASE LINK	Borrar enlaces de base de datos públicos
PUBLIC SYNONYM	
CREATE PUBLIC SYNONYM	Crear sinónimos públicos
DROP PUBLIC SYNONYM	Borrar sinónimos públicos

ROLE	
CREATE ROLE	Crear roles
ALTER ANY ROLE	Modificar cualquier rol en la base de datos
DROP ANY ROLE	Borrar cualquier rol de la base de datos
GRANT ANY ROLE	Conceder cualquier rol de la base de datos
ROLLBACK SEGMENT	
CREATE ROLLBACK SEGMENT	Crear un segmento de rollback
ALTER ROLLBACK SEGMENT	Modificar un segmento de rollback
DROP ROLLBACK SEGMENT	Borrar un segmento de rollback
SESSION	
CREATE SESSION	Conectarse a la base de datos
ALTER SESSION	Ejecutar el comando alter session para cambiar parámetros de entorno
RESTRICTED SESSION	Conectarse a la base de datos que haya sido arrancada con el comando STARTUP RESTRICT
SEQUENCE	
CREATE SEQUENCE	Crear una secuencia en el propio esquema
CREATE ANY SEQUENCE	Crear una secuencia bajo cualquier secuencia
ALTER ANY SEQUENCE	Modificar cualquier secuencia de la base de datos
DROP ANY SEQUENCE	Borrar cualquier secuencia de la base de datos
SYNONYM	
CREATE SYNONYM	Crear sinónimos en el esquema propio
CREATE ANY SYNONYM	Crear sinónimos bajo cualquier esquema
DROP ANY SYNONYM	Borrar cualquier sinónimo de la base de datos
SYSTEM	
ALTER SYSTEM	Ejecutar el comando ALTER SYSTEM
TABLE	
CREATE TABLE	Crear una tabla en el esquema propio
CREATE ANY TABLE	Crear tablas bajo cualquier esquema
ALTER ANY TABLE	Modificar la estructura de cualquier tabla de la base de datos
DROP ANY TABLE	Borrar cualquier tabla de la base de datos
EXPORT ANY TABLE	Realizar una exportación de cualquier tabla de la base de datos
LOCK ANY TABLE	Bloquear cualquier tabla o vista de la base de datos
COMMENT ANY TABLE	Comentar cualquier tabla de la base de datos
SELECT ANY TABLE	Realizar consultas de cualquier tabla de la base de datos
INSERT ANY TABLE	Realizar inserciones en cualquier tabla de la base de datos
UPDATE ANY TABLE	Realizar actualizaciones en cualquier tabla de la base de datos
DELETE ANY TABLE	Realizar borrados de filas de cualquier tabla de la base de datos
TABLESPACE	
CREATE TABLESPACE	Crear un tablespace en la base de datos
ALTER TABLESPACE	Modificar tablespaces
MANAGE TABLESPACE	Poder poner un tablespace OFF/ON LINE, y ejecutar el comando ALTER TABLESPACE BEGIN/END BACKUP
DROP TABLESPACE	Eliminar tablespaces
UNLIMITED TABLESPACE	Utilizar cantidades de espacio ilimitado en cualquier tablespace de la base de datos
TRIGGER	
CREATE TRIGGER	Crear un disparador en el esquema propio
CREATE ANY TRIGGER	Crear disparadores bajo cualquier esquema de la base de datos
ALTER ANY TRIGGER	Modificar cualquier disparador de la base de datos
DROP ANY TRIGGER	Borrar cualquier disparador de la base de datos
USER	
CREATE USER	Crear un usuario
BECOME USER	Convertirse de forma temporal en otro usuario
ALTER USER	Modificar cualquier usuario de la base de datos
DROP USER	Eliminar a un usuario
VIEW	
CREATE VIEW	Crear una vista en el esquema propio
CREATE ANY VIEW	Crear vistas bajo cualquier esquema
DROP ANY VIEW	Borrar cualquier vista de la base de datos

PERFILES

Define unos límites en la utilización de recursos de la BD.

CREATE PROFILE nombre_perfil **LIMIT**
nombre_recurso **{**entero **[K|M] | UNLIMITED |DEFAULT}**
[nombre_recurso **{**entero **[K|M] | UNLIMITED |DEFAULT}]...**

Nombre_recurso: Define unos límites en la utilización de recursos de la BD:

- **SESSIONS_PER_USER**
- **CONNECT_TIME**
- **CPU_PER_SESSION**
- **IDLE_TIME**
- **FAILED_LOGIN_ATTEMPTS**
- **PASSWORD_LIFE_TIME**
- **PASSWORD_GRACE_TIME**
- **PASSWORD_REUSE_TIME**
- **PASSWORD_REUSE_MAX**

ALTER PROFILE nombre_perfil **LIMIT**
nombre_recurso {entero **[K|M] | UNLIMITED |DEFAULT}**
[nombre_recurso {entero **[K|M] | UNLIMITED |DEFAULT}]...**

DROP PROFILE nombre_perfil **[CASCADE]**

TABLESPACES

CREATE TABLESPACE n_tablespace
DATAFILE 'n_archivo**' [SIZE** entero **[K|M]] [REUSE]**
, 'n_archivo2' **[SIZE** entero **[K|M]] [REUSE]]...**
[DEFAULT STORAGE
(INITIAL tamaño **NEXT** tamaño **MINEXTENDS** tamaño **MAXEXTENDS** tamaño **PCTINCREASE** valor**)]**
[ONLINE | OFFLINE];

CREATE TABLESPACE n_tablespace
DATAFILE 'n_archivo' **[SIZE** entero **[K|M]]**
[AUTOEXTEND ON [NEXT int K | M] [MAXSIZE {int {K | M}/UNLIMITED]]

ALTER TABLESPACE n_tablespace **{**
[ADD DATAFILE
'n_archivo' **[SIZE** entero **[K|M]] [REUSE]**
[AUTOEXTEND ON [MAXSIZE {int {K | M}/UNLIMITED]]
[,'n_archivo' **[SIZE** entero **[K|M]] [REUSE]**

```
                [AUTOEXTEND ON [MAXSIZE {int {K | M}/UNLIMITED]]
        ]
[RENAME TO nuevo_nombre
[RENAME DATAFILE 'archivo_anterior' TO 'archivo_nuevo' ]
[DEFAULT STORAGE clausulas_almacenamiento]
[ONLINE | OFFLINE]
};
```

```
DROP TABLESPACE n_tablespace  [INCLUDING CONTENTS];
```

Vistas del diccionario de datos:

- **USER_FREE_SPACE**: espacio libre en los tablespaces que tenga cuota el usuario activo
- **DBA_FREE_SPACE**: espacio libre en todos los tablespaces
- **DBA_TABLESPACES**: informa sobre los tablespaces existentes
- **DBA_TS_QUOTAS**: cuotas de los diferentes usuarios en los tablespaces. Siempre que no tengan asignado el privilegio UNLIMITES TABLESPACE.
- **DBA_DATA_FILES**: ficheros asignados a cada tablespace.

UT 4: CONSULTAS (SENTENCIA SELECT)

SENTENCIA SELECT

SELECT [DISTINCT] select_expresion **[,**select_expresion**] ...**
[FROM referencias_tablas**]**
[WHERE filtro**]**
[GROUP BY expresion **[,** expresion**] ....]**
[HAVING filtro_grupos**]**
[ORDER BY {nombre_columnas **|** expresion **|** posición**} [ASC | DESC] , ...]**

select_expresion:

{nombre_columna **[**alias**] | * |** expresión **}**

referencias_tablas:

nombre_tabla **[**alias**][,**nombre_tabla **[**alias**]]**
| nombre_tabla **[** alias**] INNER JOIN** nombre_tabla **[**alias**]**
[ON condición**]**
| nombre_tabla **[**alias**] LEFT [OUTER] JOIN** nombre_tabla**[**alias**]**
ON condición
| nombre_tabla **[**alias**] RIGHT [OUTER] JOIN** nombre_tabla **[**alias**]**
ON condición

- Convertir una expresión nula a 0: **NVL(A,0)**
- Elementos que pueden formar parte de las expresiones:
  - Operandos : constantes o variables
  - Operadores aritméticos: **+,-,*,/,%**
  - Operadores relacionales: **=,<,>,>=,<=,<>**
  - Operadores lógicos :**AND, OR,NOT**
  - Paréntesis: **()**
  - Funciones (NVL(),SUBSTR(),AVG(),etc.)
  - **Operador de pertenencia a conjuntos :IN**
    nombre_columna **IN** (Valor1, Valor2 ...)
  - **Operador de rango: BETWEEN**
    nombre_columna **BETWEEN** Valor1 AND Valor2
  - **Operadores IS e IS NOT** permiten verificar si un campo es *o no es nulo* respectivamente.
  - **Operador LIKE :** patrón de búsqueda: **%:** cualquier carácter y **_** un solo carácter

CONSULTAS DE RESUMEN

- Resume la información de varios registros.

- **Funciones de grupo:**

SUM (Expresión**)**	**AVG (**Expresión**)**	**MIN (**Expresión**)**
MAX (Expresión**)**	**COUNT(**Columna**)**	**COUNT (*)**

AGRUPACIÓN : GROUP BY

- Se denomina agrupación de registros a un conjunto de registros que cumplen que tienen una o varias columnas con el mismo valor.
- En las agrupaciones sólo es posible poner una columna si se incluye ésta en el **GROUP BY** , además podemos poner funciones de resumen**.**

FILTROS DE GRUPOS HAVING

- Los filtros de grupos deben realizarse mediante el uso de la cláusula **HAVING** puesto que WHERE actúa antes de agrupar los registros.

CONCATENAR CADENAS

- En **Oracle** se utiliza el operando **||** para concatenar una o más expresiones (También se puede utiliza **concat,** pero sólo se pueden poner 2 expresiones)
  SELECT 'El empleado '|| APELLIDO|| 'tiene el oficio : ' || OFICIO as "EMPLEADOS" FROM EMPLE;

- En **MySQL** :**concat** : concatena o une expresiones

  SELECT concat('El empleado :' ,APELLIDO,'tiene el oficio: ',OFICIO) as "EMPLEADOS"
  FROM EMPLE

SUBCONSULTAS

- Una **subconsulta** es una consulta encerrada entre paréntesis que después incluiremos en la consulta principal.

  SELECTcolumas|expresión [columas|expresión]..
  FROMtabla1[, tabla2],...
  WHEREcolumna=
  (SELECT columna **FROM** table **WHERE……);**

- **Test de Comparación:** operadores : =, >=, <=, <>, >y <
  campo operador (subconsulta)
  Ejemplo: cantidad >=(select numero....)

- **Test de pertenencia a conjunto IN :** comprueba que una o varias columnas están dentro de los resultados que devuelva una subconsulta.
- **Test cuantificados ALL y ANY:** Los test cuantificados sirven para comparar una expresión con todos los registros de la subconsulta (ALL) o algunos de los registros de la subconsulta (ANY).
- **Subconsultas correlacionadas o consultas externas:** Una consulta es correlacionada cuando necesitas algún valor de la consulta principal para poder resolver la subconsulta.
- **Test de existencia: EXISTS:** nos devuelve VERDADERO si la subconsulta devuelve algún registro y FALSO si no devuelve filas-

```
SELECT columnas
        FROM tabla
        WHERE [NOT] EXISTS (subconsulta)
```

- **Subconsultas: GROUP BY ... HAVING:**

```
SELECTcolumna
        FROMtabla
        GROUPBYcolumna
        HAVINGcount(*)> (SELECT count(*) FROM tabla2);
```

CONSULTAS DERIVADAS

- Las consultas con tablas derivadas son aquellas que utilizan sentencias SELECT en la cláusula FROM en lugar de nombres de tablas. (En MySQL deben de llevar un alias), por ejemplo:

```
SELECT *
FROM (SELECT CODIGOEMPLEADO, NOMBRE
            FROM EMPLEADOS
            WHERE CODIGOOFICINA='OFI1')
        AS TABLA_DERIVADA;
```

CONSULTAS : UNION, INTERSECT Y MINUS

- Combina los resultados de ambas consultas, convirtiendo las filas duplicadas en una sola fila.

```
SELECT col1,col2 FROM TABLA1 WHERE condicion
UNION
SELECT col1,col2 FROM TABLA2 WHERE condicion;
```

- Devuelve las filas comunes a ambas Consultas. Convierte a una sola fila los duplicados.

```
SELECT col1,col2 FROM TABLA1 WHERE condicion
MINUS
SELECT col1,col2 FROM TABLA2 WHERE condicion;
```

- Devuelve las filas que están en la primera consulta y que no están en la segunda.

```
SELECT col1,col2 FROM TABLA1 WHERE condicion
INTERSECT
SELECT col1,col2 FROM TABLA2 WHERE condicion;
```

RESUMEN DE FUNCIONES EN ORACLE

FUNCIONES ARITMÉTICAS.

Funciones de valores simples.	
ABS(n)	Devuelve el valor absoluto de (n).
CEIL(n)	Obtiene el valor entero inmediatamente superior o igual a "n".
FLOOT(n)	Devuelve el valor entero inmediatamente inferior o igual a "n".
MOD (m, n)	Devuelve el resto resultante de dividir "m" entre "n".
NVL (valor, expresión)	Sustituye un valor nulo por otro valor.
POWER (m, exponente)	Calcula la potencia de un numero.
ROUND (numero [, m])	Redondea números con el numero de dígitos de precisión indicados.
SIGN (valor)	Indica el signo del "valor".
SQRT(n)	Devuelve la raíz cuadrada de "n".
TRUNC (numero, [m])	Trunca números para que tengan una cierta cantidad de dígitos de precisión.
VARIANCE (valor)	Devuelve la varianza de un conjunto de valores.

Funciones de grupos	
AVG(n)	Calcula el valor medio de "n" ignorando los valores nulos.
COUNT (* | Expresión)	Cuenta el numero de veces que la expresión evalúa algún dato con valor no nulo. La opción "*" cuenta todas las filas seleccionadas.
MAX (expresión)	Calcula el máximo.
MIN (expresión)	Calcula el mínimo.
SUM (expresión)	Obtiene la suma de los valores de la expresión.

Funciones de listas.	
GREATEST (valor1, valor2...)	Obtiene el mayor valor de la lista.
LEAST (valor1, valor2...)	Obtiene el menor valor de la lista.

FUNCIONES DE CADENA DE CARACTERES.

Funciones que devuelven valores carácter.	
CHR(n)	Devuelve el carácter cuyo valor en binario es equivalente a "n".
CONCAT (cad1, cad2)	Devuelve "cad1" concatenada con "cad2".
LOWER (cad)	Devuelve la cadena "cad" en minúsculas.
UPPER (cad)	Devuelve la cadena "cad" en mayúsculas.
INITCAP (cad)	Convierte la cadena "cad" a tipo titulo.
LPAD (cad1, n[,cad2])	Añade caracteres a la izquierda de la cadena hasta que tiene una cierta longitud.
RPAD (cad1, n[,cad2])	Añade caracteres a la derecha de la cadena hasta que tiene una cierta longitud.
LTRIM (cad [,set])	Suprime un conjunto de caracteres a la izquierda de la cadena.
RTRIM (cad [,set])	Suprime un conjunto de caracteres a la derecha de la cadena.
REPLACE (cad, cadena_busqueda [, cadena_sustitucion])	Sustituye un carácter o caracteres de una cadena con 0 o mas caracteres.
SUBSTR (cad, m [,n])	Obtiene parte de una cadena.
TRANSLATE (cad1, cad2, cad3)	Convierte caracteres de una cadena en caracteres diferentes, según un plan de sustitución marcado por el usuario.

Funciones que devuelven valores numéricos.	
ASCII(cad)	Devuelve el valor ASCII de la primera letra de la cadena "cad".
INSTR (cad1, cad2 [, comienzo [,m]])	Permite una búsqueda de un conjunto de caracteres en una cadena pero no suprime ningún carácter después.
LENGTH (cad)	Devuelve el numero de caracteres de cad.

FUNCIONES PARA EL MANEJO DE FECHAS

SYSDATE	Devuelve la fecha del sistema.
ADD_MONTHS (fecha, n)	Devuelve la fecha "fecha" incrementada en "n" meses.
LASTDAY (fecha)	Devuelve la fecha del último día del mes que contiene "fecha".
MONTHS_BETWEEN (fecha1, fecha2)	Devuelve la diferencia en meses entre las fechas "fecha1" y "fecha2".
NEXT_DAY (fecha, cad)	Devuelve la fecha del primer día de la semana indicado por "cad" después de la fecha indicada por "fecha".

FUNCIONES DE CONVERSIÓN

TO_CHAR	Transforma un tipo DATE ó NUMBER en una cadena de caracteres.
TO_DATE	Transforma un tipo NUMBER ó CHAR en DATE.
TO_NUMBER	Transforma una cadena de caracteres en NUMBER.

TO_CHAR(fecha,'formato'):

Formato	Significado
CC	Siglo
SCC	Siglo. Si es AC (Antes de Cristo), lleva un signo —
YYYY	Año, formato de 4 dígitos
SYYY	Año, formato de 4 dígitos. Si es AC lleva un signo —
YY	Año, formato de 2 dígitos
YEAR	Año, escrito en letras y en inglés (por ejemplo, 'TWO THOUSAND TWO')
SYEAR	Ídem, pero si es AC lleva el signo —
BC	Antes o Después de Cristo (AC o DC) para usar con los anteriores, por ejemplo YYYY BC
Meses	
Q	Trimestre: Ene-Mar=1, Abr-Jun=2, Jul-Sep=3, Oct-Dic=4
MM	Número de mes (1-12)
RM	Número de mes en números romanos (I-XII)
MONTH	Nombre del mes completo rellenado con espacios hasta 10 espacios (SEPTIEMBRE)
FMMONTH	Nombre del mes completo, sin espacios adicionales
MON	Tres primeras letras del mes: ENE, FEB,...
Semanas	
WW	Semana del año (1-52)
W	Semana del mes (1-5)
Días	
DDD	Día del año (1-366)
DD	Día del mes (1-31)
D	Día de la semana (1-7)
DAY	Nombre del día de la semana rellenado a 9 espacios (MIÉRCOLES)
FMDAY	Nombre del día de la semana, sin espacios
DY	Tres primeras letras del nombre del día de la semana
DDTH	Día (ordinal): 7TH
DDSPTH	Día ordinal en palabra, en inglés: SEVENTH
horas	
HH	Hora del día (1-12)
HH12	Hora del día (1-12)
HH24	Hora del día (1-24)

SPHH	Hora del día, en palabra, inglés: SEVEN
AM	am o pm, para usar con HH, como 'HH:MI am'
PM	am o pm
A.M.	a.m. o p.m.
P.M.	a.m. o p.m.
Minutos y segundos	
MI	Minutos (0-59)
SS	Segundos (0-59)
SSSS	Segundos después de medianoche (0-86399)

TO_CHAR(número,'formato')

Formato	Descripción
9	valor retornado con el número especificado de dígitos y los 0 por la izquierda se convierten en espacios en blanco
0	como 9, pero en lugar de espacios en blanco usa ceros
. (period)	punto decimal
, (comma)	separador de grupo (miles)
PR	retorna el valor negativo en anglebrackets
S	retorna el valor negativo con el signo menos
L	símbolo monetario
D	punto decimal
G	separador de grupos
MI	retorna el signo menos en la posición especificada (si número < 0)
RN	retorna el número como número romano(número debe ser entre1 y 3999)
V	arg1 * (10 ^ n); - retorna un valor multiplicado por 10^n (donde 'n' es número de '9's despues de 'V'). El to_char() no soporta el uso de 'V' y punto decimal juntos, ejemplo "99.9V99".
EEEE	numeroscientificos . ahora no soportados.

OTRAS FUNCIONES

- Sustituye un valor por otro:
  DECODE(var1,valor1,codigo1,valor2,codigo2,...,valor_por_defecto)
- Devuelve en número de bytes que ocupa una expresión:
  VSIZE(expresión)
- **USER**: Devuelve el nombre del usuario actual

```
SQL>SELECT USER FROM DUAL;
USER
----------
DAM101
```

UT 5: TRATAMIENTO DE DATOS

INSERCIÓN DE REGISTROS

INSERT INTO nombretabla [(columna1,columna2,....)] **VALUES** (valor [,valor]...);

INSERT INTO nombretabla1 [(columna1,columna2,....)]
SELECT {columna1[,columna2].../*}
FROM nombre_tabla;

ACTUALIZACIÓN DE REGISTROS

UPDATE nombretabla
SET columna1=valor1,..., columnaN=valorN
WHERE condición;

UPDATE n_tabla
SET columna1=valor1,..., colunmaN=valorN
WHERE (columna1,columna2...)**= (SELECT** columna1,columna2,...**FROM .. WHERE...);**

UPDATE n_tabla
SET (columna1, columna2...)**=(SELECT** columna1, columna2... **FROM ...WHERE)**
WHERE condición;

ELIMINACIÓN DE REGISTROS

DELETE [**FROM**] nombretabla**WHERE** condición;

DELETE [FROM] Nombre_tabla
[WHERE condición **= (SELECT ...);**

TRANSACCIONES

- **Oracle:** por defecto **OFF**
  SET AUTOCOMMIT OFF/ON
- **COMMIT :**para confirmar una transacción y almacenarla definitivamente.
- **ROLLBACK :** para abortar una transacción y que no se almacenen en la base de datos.

ACCESO CONCURRENTE A LOS DATOS

- En **Oracle**, el nivelpor defecto es **READ COMMITED** y, además de éste, solo permite **SERIALIZABLE**. Se puede cambiar ejecutando el comando:
  SET TRANSACTION ISOLATION LEVEL {READ COMMITTED|SERIALIZABLE};

CREACIÓN DE VISTAS

CREATE [OR REPLACE] VIEW [esquema.]nombre_vista **[(**lista_columnas**)] AS**
sentencia_select**;**
DROP VIEW nombre_vista**;**

UD6: PROGRAMACIÓN DE BASE DE DATOS: PL/SQL (Oracle)

CONCEPTOS INICIALES DE PL/SQL

Bloque

```
[DECLARE
        /* Sección declarativa */
]
BEGIN
        /* Sección ejecutable  */
[EXCEPTION
        */ Sección de excepciones  */
]
END;
```

Identificadores

- Comienzan con una letra + (letras , números, $, _,#)
- Máximo: 30 caracteres.

Comentarios.

--	Comentarios en una sola línea
/* */	Comentarios en varias líneas

Declaración de variables.

<Nombre_Var> <tipo_dato> [CONSTANT] [NOT NULL] [:= / DEFAULT expr];

TIPOS PL/SQL

- **Tipos Derivados:**

  Nombre_Variable NombreTabla.NombreColumna%TYPE;

  - Atributo %TYPE: Permite crear una variable del mismo tipo que una columna de una tabla

  NombreVariable NombreTabla%ROWTYPE;

  - Atributo %ROWTYPE: crea variables que tengan el mismo tipo que una tabla.

- **Entrada por teclado:** &
  V_EMPNO EMPLE.EMP_NO%TYPE:=&EMPLEADO;
  V_SALARIO NUMBER:=&SALARIO;
- Clasificación:
  - **Escalar**: contienen un valor simple
  - **Compuesto**: permiten que se definan y manipulen grupos de valores (colecciones: registros, arrays…)
  - **Referenciado**: llamados *punteros*, designan otros artículos de programa.
  - **LOB**: (CLOB, BLOB, BFILE, NCLOB) contienen valores, llamados *localizadores* que indican la localización de lobs (imágenes,…) que pueden estar almacenados externamente.

Tipos escalares

Numéricos	
NUMBER(P,S)	Numèrico entero o punto flotante.P:precisión(38), S:escala(-84,127)
BINARY_INTEGER	Valores enteros. No almacenar en BD
PLS_INTEGER	Igual que birary, pero genera error en desbordamiento
Carácter	
VARCHAR2(N)	Cadenas de longitud variable (PL:32767, BD:2000,4000(v8))
CHAR(N)	Cadenas de longitud fija (PL:32767,DB:255)
LONG(N)	Cadena de longitud variable(PL:2G,DB:32767)
RAW	
RAW(N)	Almacena datos binarios(PL:32767,BD:255)
LONG RAW(N)	Igual RAW(PL:32767,BD:2G)
DATE	Almacena datos tipo fecha (7bytes)
ROWID	Clave que identifica unívocamente a cada fila, se almacena en hexadecimal: BBBBBBBB.RRRR.FFFF
BOLEAN	TRUE, FALSE, NULL

Tipos compuestos:

- Registros
- Colecciones:
  - VARRAY
  - Tablas Anidadas
  - Tablas Indexadas

OPERADORES

TIPO OPERADOR	
Asignación	**:=**
Lógicos	**AND, OR, NOT**
Concatenación	**||**
Comparación	Igual = distinto != menor que < mayor que >menor o igual <= mayor o igual >= IS NULL BETWEEN LIKE IN
Aritméticos	+, -, *, /, **(potencia)

INTRODUCIR POR TECLADO

N NUMBER(5):= **&**NUMERO;
V VARCHAR2(20):='**&**ALFANUMERICO';

SALIDA POR PANTALLA

DBMS_OUTPUT.PUT_LINE ('NO EXISTE ESE REGISTRO') ;

ESTRUCTURAS DE CONTROL.

IF-THEN-ELSE

```
IF expresión_boleana1 THEN
        secuencia_de_órdenes1;
[ELSIF expresión_boleana2 THEN
        secuencia_de_órdenes2;]
[ELSE
        secuencia_de_órdenes3;]
END IF;
```

SENTENCIA CASE:

```
CASE <expresion>
WHEN <valor_comprobacion1> THEN
   instrucciones1;
WHEN <valor_comprobacion2> THEN
   instrucciones2;
.....
[ELSE
   instrucciones;]
END CASE;
```

```
CASE
WHEN <condición1> THEN
   instrucciones1;
WHEN <condición2> THEN
   instrucciones2;
.....
[ELSE
   instrucciones;]
END CASE;
```

BUCLES SIMPLES

```
LOOP
   instrucciones;
   ...;
   EXIT WHEN condición;
   ...;
   instrucciones;
   ...;
END LOOP;
```

```
LOOP
   instrucciones;
   ...;
   IF condición THEN
         EXIT;
   END IF;
   instrucciones;
   ...;
END LOOP;
```

BUCLES WHILE

```
WHILE condición LOOP
   secuencia_de_órdenes
END LOOP;
```

BUCLES FOR

```
FOR contador_bucle IN [REVERSE]
  límite_inferior..límite_superior LOOP
   secuencia_de_órdenes;
END LOOP;
```

SENTENCIA : SELECT, INSERT, UPDATE,DELETE

SELECT:

```
SELECT <Columna/s> INTO <variable/s>
FROM <Tabla/s>
WHERE  <condición>
....
```

INSERT, UPDATE Y DELETE: Tienen la misma sintaxis que en SQL:

```
DELETE [FROM] nombretabla
        WHERE condición;

INSERT INTO nombretabla [(columna1,columna2,....)]  VALUES (valor [,valor]...);
UPDATE nombretabla
        SET columna1=valor1,..., columnaN=valorN
        WHERE condición;
```

PROCEDIMIENTOS

```
CREATE OR REPLACE PROCEDURE nombre_procedimiento
    [(parametro1 [IN|OUT|IN OUT] <tipo> [{:=|DEFAULT} <valor>,...)]
IS|AS
        declaraciones;
BEGIN
        instrucciones;
[EXCEPTION
        gestión_de_excepciones;]
END [nombre_procedimiento];
```

FUNCIONES

```
CREATE OR REPLACE FUNCTION nombre_función
        [(parametro1 [IN|OUT|IN OUT] <tipo>
                [{:=|DEFAULT} <valor>,...)]
RETURN <tipo_del_valor_devuelto>
IS|AS
        declaraciones;
BEGIN
        órdenes ejecutables;
        RETURN <expr>;
[EXCEPTION
        gestión_de_excepciones;]
END <n_función>;
```

CURSORES

Declarar el cursor

```
CURSOR  nombre_cursor  IS sentencia_SELECT;
```

Apertura del cursor

```
OPEN nombre_cursor;
```

Extracción de los datos del cursor.

```
FETCH  nombre_cursor  INTO  lista_variables;
FETCH  nombre_cursor  INTO  registro_pl/sql;
```

Cierre del cursor.

```
CLOSE  nombre_cursor;
```

Atributos del cursor.

%FOUND	Devuelve TRUE si la última sentencia FETCH devuelve una fila
%NOTFOUND	Devuelve TRUE si la última sentencia FETCH no devuelve ninguna fila
%ISOPEN	Devuelve TRUE si el cursor está abierto
%ROWCOUNT	Devuelve el número de filas devueltas por el cursor hasta el momento

Bucle FOR en un cursor

```
FOR vble_registro IN n_cursor LOOP
        sentencias;
END LOOP;
```

Cursores parametrizados.

```
DECLARE
        CURSOR c_empleados(p_dep  VARCHAR2(7)) IS
                SELECT * FROM empleados
                        WHERE codemp=p_dep;
BEGIN
        OPEN c_empleados('D123');
```

Cursores con SELECT FOR UPDATE.

- Se utiliza para extraer los datos y después modificarlos o borrarlos.
- Pasos:
  1. Declarar el cursor con SELECT ..FOR UPDATE
  2. Órdenes DELETE o UPDATE con la opción WHERE CURRENT OF.

FOR UPDATE.

- Es la última cláusula de la orden SELECT. Sintaxis:

  SELECT ... FROM .... FOR UPDATE [OF referencia_columna][NOWAIT]

- Referencia_columna: una o varias columnas de la tabla para actualizar, si no se especifica ninguna se pueden modificar todas.
- NOWAIT: si los registros a los que se quieren acceder están bloqueados por otro usuario la sentencia SELECT espera hasta que se desbloquean, si no queremos que espere se pondrá la cláusula NOWAIT.

WHERE CURRENT OF.

- Esta cláusula se emplea en las órdenes DELETE o UPDATE cuando el cursor se haya declarado con la opción SELECT .. FOR UPDATE. Se actualizará o borrará la fila que actualmente se haya extraído.
- Sintaxis:

  WHEN CURRENT OF cursor

CURSORES VARIABLES

- Definición :

  ```
  TYPE T_CURSOR IS REF CURSOR
          RETURN EMPLE%ROWTYPE;
  ```

- Declaración de la variable

  ```
  CEMPLE T_CURSOR;
  ```

- Asociarlo a una consulta:

  OPEN nombre_variable_cursor **FOR** sentencia_select;

  OPEN CEMPLE FOR SELECT * FROM EMPLE WHERE DEPT_NO=10;

EXCEPCIONES

```
EXCEPTION
WHEN nombre_excepción THEN
        <tratamiento>
WHEN nombre_excepción THEN
        <tratamiento>
....
WHEN OTHERS THEN
        <tratamiento>
        */ Sección de excepciones  */
END;
```

Excepciones internas definidas por ORACLE:

EXCEPCIÓN	DESCRIPCIÓN	SQLCODE
ACCESS_INTO_NULL	El programa intentó asignar valores a los atributos de un objeto no inicializado	-6530
CASE_NOT_FOUND	Ninguna de las opciones existentes en las cláusulas WHEN de un CASE se selecciona, y no hay ninguna cláusula ELSE.	
COLLECTION_IS_NULL	El programa intentó asignar valores a una tabla anidada aún no inicializada	-6531
CURSOR_ALREADY_OPEN	El programa intentó abrir un cursor que ya se encontraba abierto. Recuerde que un cursor de ciclo FOR automáticamente lo abre y ello no se debe especificar con la sentencia OPEN	-6511
DUP_VAL_ON_INDEX	El programa intentó almacenar valores duplicados en una columna que se mantiene con restricción de integridad de un índice único (unique index)	-1
INVALID_CURSOR	El programa intentó efectuar una operación no válida sobre un cursor	-1001
INVALID_NUMBER	En una sentencia SQL, la conversión de una cadena de caracteres hacia un número falla cuando esa cadena no representa un número válido	-1722
LOGIN_DENIED	El programa intentó conectarse a Oracle con un nombre de usuario o password inválido	-1017
NO_DATA_FOUND	Una sentencia SELECT INTO no devolvió valores o el programa referenció un elemento no inicializado en una tabla indexada	+100
NOT_LOGGED_ON	El programa efectuó una llamada a Oracle sin estar conectado	-1012
PROGRAM_ERROR	PL/SQL tiene un problema interno	-6501
ROWTYPE_MISMATCH	Los elementos de una asignación (el valor a asignar y la variable que lo contendrá) tienen tipos incompatibles. También se presenta este error cuando un parámetro pasado a un subprograma no es del tipo esperado	-6504
SELF_IS_NULL	El parámetro SELF (el primero que es pasado a un método MEMBER) es nulo	-30625
STORAGE_ERROR	La memoria se terminó o está corrupta	-6500

SUBSCRIPT_BEYOND_COUNT	El programa está tratando de referenciar un elemento de un array indexado que se encuentra en una posición más grande que el número real de elementos de la colección	-6533
SUBSCRIPT_OUTSIDE_LIMIT	El programa está referenciando un elemento de un array utilizando un número fuera del rango permitido (por ejemplo, el elemento "-1")	-6532
SYS_INVALID_ROWID	La conversión de una cadena de caracteres hacia un tipo *rowid* falló porque la cadena no representa un número	-1410
TIMEOUT_ON_RESOURCE	Se excedió el tiempo máximo de espera por un recurso en Oracle	-51
TOO_MANY_ROWS	Una sentencia SELECT INTO devuelve más de una fila	-1422
VALUE_ERROR	Ocurrió un error aritmético, de conversión o truncamiento. Por ejemplo, sucede cuando se intenta calzar un valor muy grande dentro de una variable más pequeña	-6502
ZERO_DIVIDE	El programa intentó efectuar una división por cero	-1476

- Excepciones definidas por el usuario

```
DECLARE
    Nombre_excepción    EXCEPTION;
```

- Levantar excepción:
  RAISE nombre_excepción;
- Para acceder a los errores de Oracle que no tienen asignada una excepción:
  SQLCODE: código del error
  SQLERRM: mensaje asociado al error
- Errores sin excepción asignada: Asociamos a esa excepción el código de error de ORACLE.
  PRAGMA EXCEPTION_INIT(nombre_excepcion, codigo**);**
- Errores generados por el usuario:
  RAISE_APPLICATION_ERROR (numero_error, mensaje_error**)**
  Rango del nº de error :-20000, -20999

DISPARADORES: TRIGGERS

Formato para la creación de un triggers:

```
CREATE [OR REPLACE] TRIGGER Nombre_del_trigger
    {BEFORE | AFTER |INSTEAD OF}
    {DELETE | INSERT | UPDATE [OF columna1 [,columna2...]]} ...
    ON {tabla |vista}
    [FOR EACH  {ROW |[WHEN (condición)] | STATEMENT}]
/* Comienza el cuerpo del trigger */
[DECLARE
    <declaraciones>
]
BEGIN
    <sentencias>
[EXCEPTION
    <gestión de excepciones>
]
END ;
```

Valores para NEW y OLD.:

- Al utilizar los valores :old (valor anterior) y :new(valor nuevo) debemos tener en cuenta el evento del disparo:
  - **DELETE:** debemos hacer referencia siempre a **:old.nombrecolumna**, ya que :new no existe
  - **INSERT**: debemos hacer referencia siempre a **:new.nombrecolumna**, ya que :old no existe
  - **UPDATE:** se pueden usar las dos

Múltiples eventos de disparo:

INSERTING	Devuelve TRUE si el evento que disparó el trigger fue un comando INSERT
DELETING	Devuelve TRUE si el evento que disparó el trigger fue un comando DELETE
UPDATING	Devuelve TRUE si el evento que disparó el trigger fue un comando UPDATE
UPDATING(columna)	Devuelve TRUE si el evento que disparó el trigger fue un comando UPDATE y la columna especificada ha sido actualizada

Trigger con múltiples eventos de disparo:

```
CREATE or REPLACE TRIGGER nombre_trigger
{BEFORE|AFTER} evento1 OR evento2 OR... ON tabla1
BEGIN
        IF INSERTING THEN
                ...
        ELSIF DELETING THEN
                ...
        ELSIF UPDATING('COLUMNA') THEN
                ...
        END IF;
END;
```

Trigger de Sustitución (Vistas):

```
CREATE [OR REPLACE] TRIGGER nombre_trigger
INSTEAD OF
   {DELETE | INSERT | UPDATE [OF columna1 [,columna2...]]}
   {OR {DELETE|INSERT|UPDATE}...}
ON n_vista
[REFERENCING OLD AS .... , NEW AS ...]
FOR EACH ROW
DECLARE
        ...
BEGIN
        ...
EXCEPTION
        ...
END;
```

REGISTROS

- Declaración del tipo:

```
TYPE tipo_registro IS RECORD(
        campo1  tipo1 [NOT NULL]  [:=expr1] ,
        campo2  tipo2 [NOT NULL ] [:=expr2 ],
        .....
        campoN  tipoN [NOT NULL]  [:=exprN ] );
```

- Declaración de la variable:

```
DECLARE
        nombre_registro   n_tipo;
```

- Ejemplo:

```
DECLARE
        TYPE t_alumnos IS RECORD(
                dni             number(9),
                nombre          varchar2(50),
                nota            number(4,2))

        r_alumnos t_alumnos;
```

COLECCIONES :VARRAYS

- Definir el tipo en un bloque anónimo:

```
TYPE nombre_tipo_array IS
        VARRAY(num_elementos) OF tipo_elemento [NOT NULL];
```

- Podemos definirlos el tipo en base de datos para que pueda ser usado cuando se quiera:

```
CREATE OR REPLACE
        TYPE nombre_tipo_array IS
                VARRAY(num_elementos) OF tipo_elemento [NOT NULL];
```

  - Ejemplo:

```
TYPE T_EMPLE IS VARRAY(5) OF EMPLE.EMP_NO%TYPE;
```

- Declarar la variable:

```
Nombre_variable nombre_tipo_array;
```

Ejemplo:

```
VA_EMPLE T_EMPLE;
```

- Inicializar la variable cargando valores:

En la misma declaración:

```
VA_EMPLE T_EMPLE:=T_EMPLE(1,2,3,4,5);
```

En la zona ejecutable :

```
VA_EMPLE:=T_EMPLE(1,2,3,4,5);
```

COLECCIONES : TABLAS ANIDADAS

- Definir el tipo en un bloque anónimo:

```
TYPE nombre_tipo IS
        TABLE  OF tipo_elementos [NOT NULL];
```

- Ejemplo**:**

  TYPE T_DIAS IS TABLE OF VARCHAR2(20);
- Podemos definirlos el tipo en base de datos para que pueda ser usado cuando se quiera:

  CREATE OR REPLACE
  TYPE nombre_tipo **IS**
  TABLE OF tipo_elementos [NOT NULL];
- Declaramos e inicializamos las variables:

  nombre_var:= nombre_tipo();
- Para añadir filas nuevas a la tabla se utiliza el método EXTEND:

  nombre_var.EXTEND;

COLECCIONES : TABLAS INDEXADAS

- Definir el tipo en un bloque anónimo:

  TYPE tipo_tabla **IS**
  TABLE OF tipo_elemento **[NOT NULL]**
  INDEX BY [PLS_INTEGER/ BINARY_INTEGER/VARCHAR2(longitud**)];**
- Declarar la variable:

  nombre_var tipo_tabla;

  No hace falta inicializarlas y tampoco hay que reservar espacio para los nuevos elementos con EXTEND.

Atributos de las tablas:

- **Vble_tabla.FIRST:** devuelve el valor de la clave o índice del primer elemento de la tabla.
- **Vble_tabla.LAST:** devuelve el último elemento de la tabla:
- FOR i IN vble_tabla.FIRST..vble_tabla.LAST LOOP
- **Vble_tabla.PRIOR(n):** devuelve el elemento anterior al elemento **n.**
- **Vble_tabla.NEXT(n):** devuelve el elemento posterior al elemento **n.**
- **Vble_tabla.EXISTS(n):** devuelve TRUE cuando existe el elemento **n**
- **Vble_tabla.COUNT**: devuelve el número de filas que tiene una tabla.
- **Vble_tabla.DELETE[(n1 [, n2])]:** Borra elementos de una tabla. Si n1 y n2 no están se borra la tabla entera.

PAQUETES

- **Cabecera o Especificación**

  CREATE [OR REPLACE] PACKAGE nombre_paquete **AS**
  Declaraciones de tipos, cursores, excepciones,... públicos (accesibles desde el paquete y exterior:n_paquete.n_objeto)
  Especificación de subprogramas (cabeceras=nombre, parámetros y tipo de retorno en funciones)
  END [n_paquete];

- **Cuerpo del paquete:**

  CREATE [OR REPLACE] PACKAGE BODY nombre_paquete **AS**
  declaración de tipos, variables, cursores,... privados
  cuerpo de los subprogramas
  BEGIN
  instrucciones iniciales;
  END [n_paquete];

UD7: BASE DE DATOS OBJETO-RELACIONAL

OBJETOS

- Definir el tipo:

```
CREATE or REPLACE TYPE nombreObjeto AS OBJECT (
 atributo TIPO,
 atributo TIPO,
 ...,
 MEMBER FUNCTION nombreFuncion RETURN Tipo,
 MEMBER FUNCTION nombreFuncion2(Var TIPO,...)
        RETURN Tipo,
 MEMBER PROCEDURE nombreProcedimiento,
 MEMBER PROCEDURE nombreFuncion2(Var TIPO,...)
);
```

- **Inicialización de objetos:**

```
variable_objeto := NEW Nombre_Tipo_Objeto
                (valor_atributo1, valor_atributo2, ...);
variable_objeto := Nombre_Tipo_Objeto
                (valor_atributo1, valor_atributo2, ...);
```

- **Sintaxis de crear cuerpo:**

```
CREATE OR REPLACE TYPE BODY nombre_del_tipo AS
        <implementación de los métodos>
END;
```

- **<implementación de los métodos>** tiene el siguiente formato:

```
[STATIC|MEMBER|PROCEDURE NombreProc [(param1,parám2,...)]
IS
        Declaraciones;
BEGIN
        Instrucciones;
END;
[STATIC|MEMBER|CONSTRUCTOR] FUNCTION NombreFunc
                [(parametrol, parámetro2,...)) RETURN tipo_valor_retorno
IS
        Declaraciones;
BEGIN
        Instrucciones;
END;
```

- Para indicar que un tipo de objeto es **heredado** de otro hay que usar la **palabra reservada UNDER** y además hay que tener en cuenta que el tipo de objeto del que **hereda** debe tener la **propiedad NOT FINAL**.
- Por defecto, los tipos de objeto se declaran como **FINAL**, es decir, que no se puede crear un tipo de objeto que herede de él.
- Igualmente si un **método** es **FINAL** los subtipos no pueden redefinirlo.
- La cláusula **OVERRIDING** se utiliza para redefinir el método.

MÉTODOS MAP Y ORDER

- Los métodos **MAP** consisten en una función que devuelve un valor de tipo escalar (CHAR, VARCHAR2, NUMBER, DATE, ...) que será el que se **utilice en las comparaciones y ordenaciones** aplicando los criterios establecidos para este tipo de datos.
- Sólo **puede haber un método MAP u ORDER.**
- Un método **ORDER** utiliza los atributos del objeto sobre el que se ejecuta para realizar un cálculo y compararlo con otro objeto del mismo tipo que toma como argumento de entrada. Este método devuelve un valor negativo si el parámetro de entrada es mayor que el atributo, un valor positivo si ocurre lo contrario y cero si ambos son iguales

```
CREATE OR REPLACE TYPE USUARIO AS OBJECT (
        LOGIN VARCHAR2(30),
        NOMBRE VARCHAR2(30),
        APELLIDOS VARCHAR2(40),
        F_INGRESO DATE,
        CREDITO NUMBER,
        MAP MEMBER FUNCTION ORDENARUSUARIO RETURN VARCHAR2
);

CREATE OR REPLACE TYPE BODY USUARIO AS
        MAP MEMBER FUNCTION ORDENARUSUARIO RETURN VARCHAR2 IS
        BEGIN
                RETURN (APELLIDOS || ' ' || NOMBRE);
        END ORDENARUSUARIO;
END;
```

TABLAS DE OBJETOS

- Una vez definidos los objetos se pueden utilizar para definir nuevos tipos o para definir tablas de ese tipo de objetos.
- Una tabla de objetos es una tabla en la que cada fila se almacene un objeto de ese tipo.
- Se accede a los atributos de esos objetos como si se tratasen de columnas de la tabla.

```
CREATE TABLE OF USUARIO (PRIMAY KEY LOGIN);

CREATE TABLE OF PERSONAS(
        DNI             VARCHAR2(9) PRIMARY KEY,
        DATOS           USUARIO
);
```

- **Tablas de objetos: VALUE:** Cuando se quiera **hacer referencia a un objeto en lugar de alguno de sus atributos**, se puede utilizar la función **VALUE** junto con el nombre de la tabla de objetos o su alias, **dentro de una sentencia SELECT**.

```
INSERT INTO Favoritos
        SELECT VALUE(u) FROM Usuario u
         WHERE u.credito >= 100;
```

- Comparaciones:

```
SELECT u.login
    FROM UsuariosObj u JOIN Favoritos f
        ON VALUE(u)=VALUE(f);
```

- Usando la **cláusula INTO se puede guardar en variables el objeto obtenido** en las consultas usando la función **VALUE**.

```
DECLARE
        u1 Usuario;
        u2 Usuario;
BEGIN
        SELECT VALUE(u) INTO u1
                FROM UsuariosObj u
                WHERE u.login = 'luitom64';
        dbms_output.put_line(u1.nombre);
        u2 := u1;
        dbms_output.put_line(u2.nombre);
END;
```

REFERENCIAS: REF Y DEREF

- Mediante el operador REF asociado a un atributo se pueden definir referencias a otros objetos.
- Un atributo de este tipo almacena una referencia al objeto del tipo definido e implementa una relación de asociación entre los dos tipos de objetos.

```
CREATE TYPE EMPLEADO_T AS OBJECT(
        NOMBRE VARCHAR2(30) ,
        JEFE REF EMPLEADO T
);
CREATE TABLE EMPLEADO OF EMPLEADO_T;
```

- Para acceder al objeto referido por un **REF** se utiliza el operador **DEREF**, en el ejemplo se visualiza el nombre del empleado y los datos del jefe de cada empleado:

```
DECLARE
        EMP             EMPLEADO_T;
        VNOMBRE         EMPLEADO.NOMBRE%TYPE;
BEGIN
        FOR C IN (SELECT NOMBRE, DEREF(E.JEFE) D FROM EMPLEADO E) LOOP
                DBMS_OUTPUT.PUT_LINE('Empleado : '||C.NOMBRE);
                IF C.D IS NULL THEN
                        DBMS_OUTPUT.PUT_LINE('NO TIENE JEFE');
                ELSE
                        DBMS_OUTPUT.PUT_LINE('SU JEFE ES :  '||C.D.NOMBRE);
        END IF;
        END LOOP;
END;
```

- La siguiente consulta obtiene el identificador del objeto cuyo nombre es GIL:

```
SELECT REF(P) INTO EMP FROM EMPLEADO P WHERE NOMBRE='GIL';
```

OBJETOS Y COLECCIONES

- Definir el tipo en un bloque anónimo:

```
TYPE nombre_tipo
        AS TABLE OF tipo_objeto;
```

- Podemos definirlos el tipo en base de datos para que pueda ser usado cuando se quiera:

```
CREATE OR REPLACE  TYPE nombre_tipo
AS TABLE OF tipo_objeto;
```