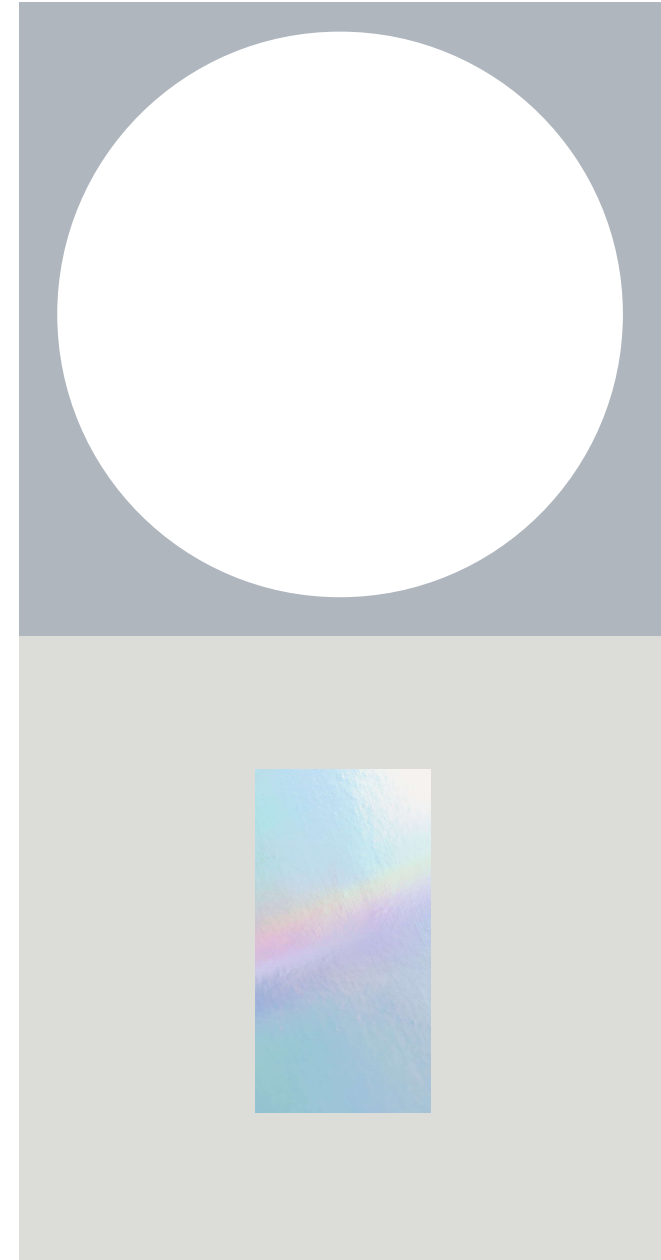


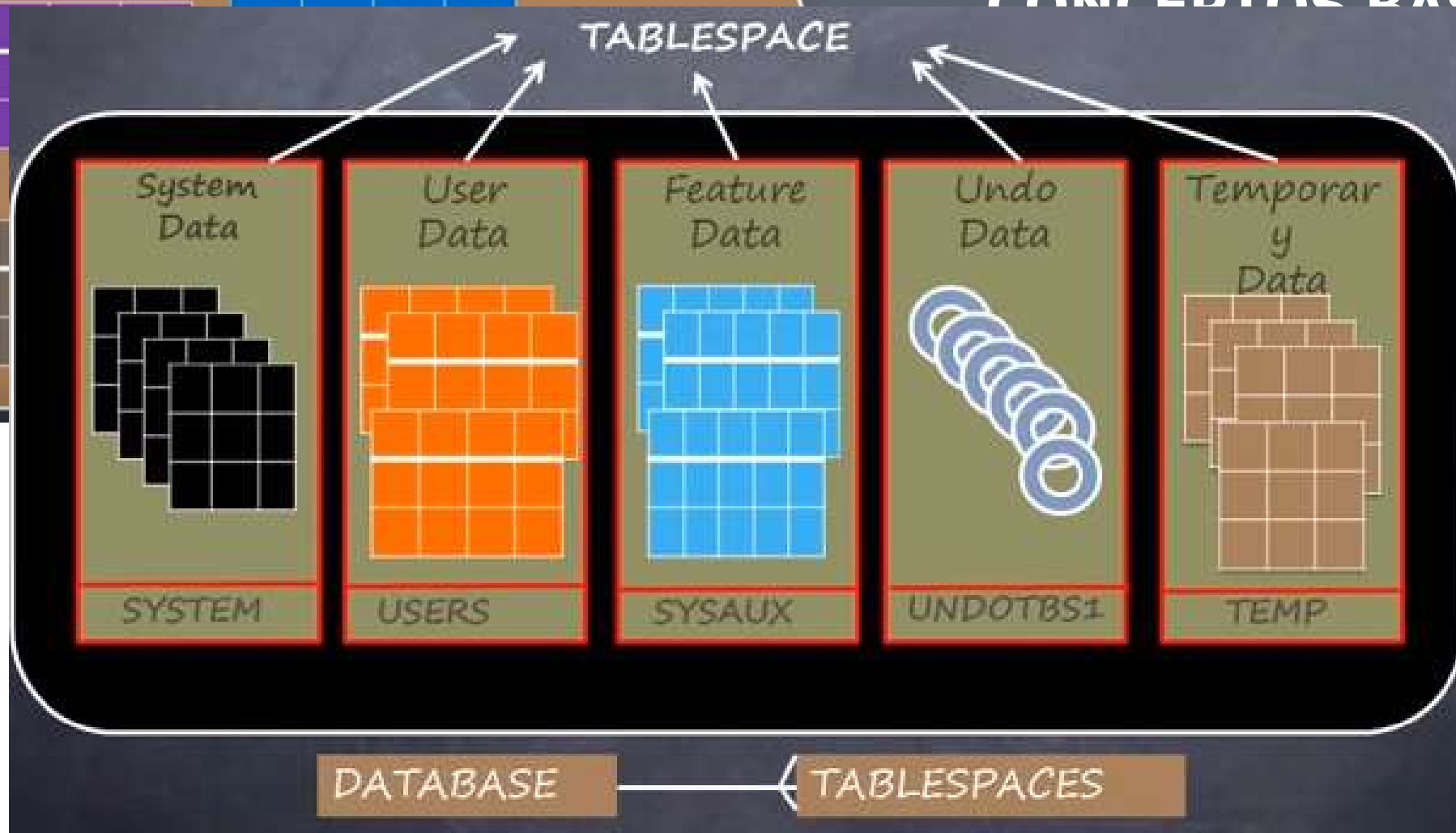
# UT4 – El lenguaje SQL

## Parte 1

- Conceptos básicos
- Tablespaces
- Ficheros y tablas
- Usuarios, privilegios, roles
- SQL – Tipos de datos y sentencias
- DDL – Creación de tablas
- DDL – Restricciones en tablas y campos




# CONCEPTOS BÁSICOS






**DATABASE** - (Is a centralized repository of related data)

# TABLESPACES

```
select distinct tablespace_name  
from dba_segments;
```

Salida de Script x  Resultado de la Consulta x

   SQL | Todas las Filas Recuperadas: 4 en 0,12 se

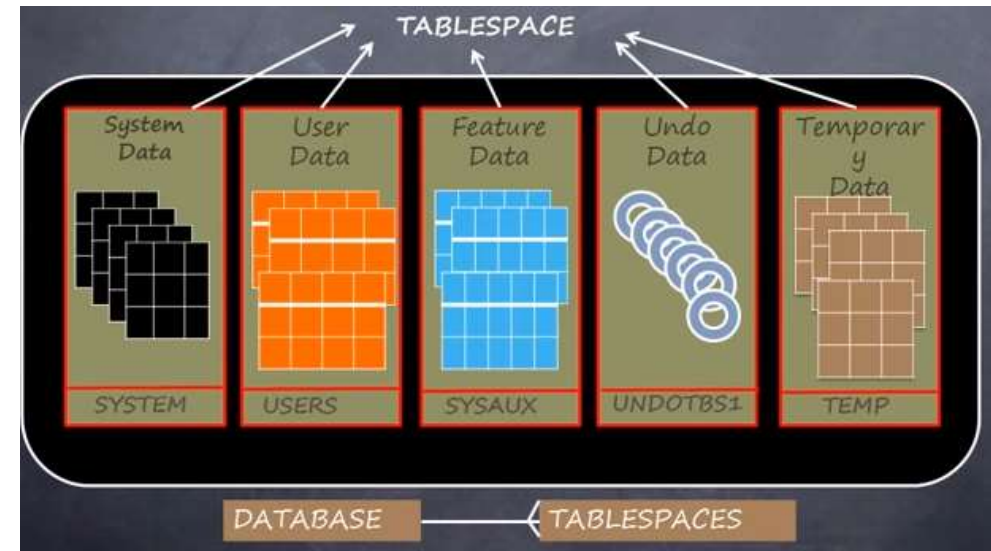
TABLESPACE\_NAME

1 SYSAUX

2 UNDOTBS1

3 USERS

4 SYSTEM



# FICHEROS Y TABLAS

- *Control Files* : Contains information about the database. The file usually has a .ctl extension.
- *Redo Log Files* : Stores a sequential record of changes that occur in the database. The file usually has a .log extension.
- *Datafiles* : Stores data of the database. The file usually has a .dbf extension.
- *Archive Log Files* : Stores offline copies of redo log files for media recovery purposes (Optional)

```
select table_name from tabs;
```

da de Script x		Resultado de la Consulta x	
SQL   Todas las Filas Recuperadas: 156 en			
TABLE_NAME			
7	REPCAT\$_INSTANTIATION_DDL		
8	REPCAT\$_EXTENSION		
9	REPCAT\$_SITES_NEW		
0	LOGSTDBY\$PARAMETERS		
1	LOGSTDBY\$EVENTS		

TABLE_NAME	
1	Tprofesores
2	Tmodulos
3	TESPECIALIDADES
4	TCursos
5	TCOMUNIDADES
6	TALUMNOS

listener.ora

sqlnet.ora

tnsnames.ora

## Usuarios

Un **usuario** es un conjunto de **permisos** que se asignan y aplican a una conexión de base de datos.

Además, tiene otras funciones:

- Ser el propietario de ciertos objetos (tablas, vistas, etc.). **ESQUEMA**
- Realiza las copias de seguridad.
- Define una cuota de almacenamiento.
- Define el tablespace por defecto para los objetos de un usuario en Oracle.

**SCHEMA = USER**

# TRABAJANDO CON USUARIOS

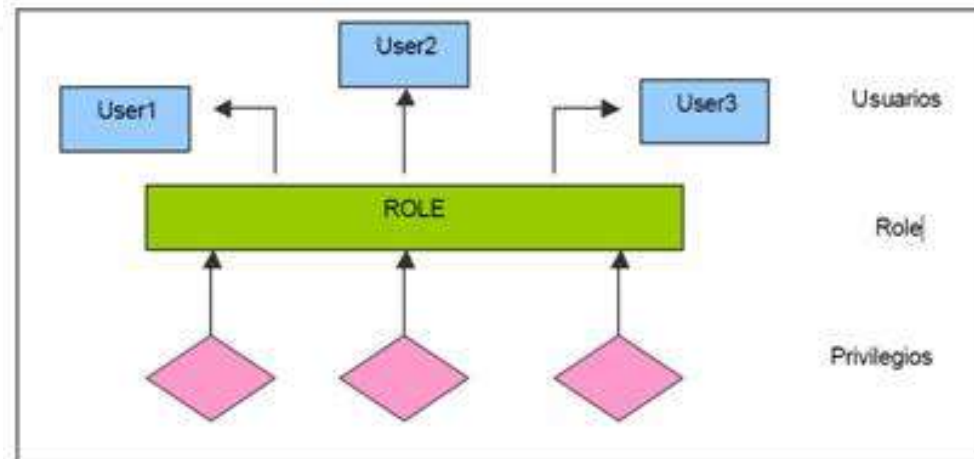
```
select username from dba_users where username no
```

Salida de Script x Resultado de la Consulta x

SQL | Todas las Filas Recuperadas: 4 en 0,031 segundos

USERNAME

1	ANDREA
2	MIGUEL
3	JULIA
4	APEX_040000



# GESTIÓN DE TABLESPACES

```
CREATE TABLESPACE julia_data DATAFILE 'C:\Users\Julia\Documents\oracle-scripts\juli  
SIZE 10M  
AUTOEXTEND ON MAXSIZE 200M  
EXTENT MANAGEMENT LOCAL UNIFORM SIZE 64K;
```

Salida de Script x Resultado de la Consulta x  
Tarea terminada en 0,188 segundos

tablespace JULIA\_DATA creado.

```
SELECT tablespace_name, status FROM dba_tablespaces;
```

Salida de Script x Resultado de la Consulta x  
SQL | Todas las Filas Recuperadas: 6 en 0,001 segundos

TABLESPACE_NAME	STATUS
1 SYSTEM	ONLINE
2 SYSAUX	ONLINE
3 UNDOTBS1	ONLINE
4 TEMP	ONLINE
5 USERS	ONLINE
6 JULIA_DATA	ONLINE

```
DROP TABLESPACE julia_data;
```

Salida de Script x Resultado de la Consulta x  
Tarea terminada en 0,068 segundos

tablespace JULIA\_DATA borrado.

**SQL:** Structured Query Language. Es un lenguaje de consulta (Lenguaje de 4ª generación).

Permite, además, la **manipulación y la gestión** de una bbdd.

Lenguaje para todo tipo de usuarios:

**Usuarios finales**

**Programadores**

**Administradores**

Lenguaje **Portable** a diferentes entornos.

Otros lenguajes lo utilizan embebido para acceder a BBDD (JAVA, .NET,....).



UT 4 - ELSQL-- Introducción al SQL



## 2 Overview of PL/SQL

Advantages of PL/SQL

Main Features of PL/SQL

Architecture of PL/SQL

Protecting Sensitive Information in PL/SQL

## 3 PL/SQL Language Fundamentals

## 4 PL/SQL Data Types

SQL Data Types

PLS\_INTEGER and BINARY\_INTEGER Data Types

User-Defined PL/SQL Subtypes

## 4 PL/SQL Data Types

Every PL/SQL constant, variable, parameter, and function receive valid values and operations.

This chapter explains **scalar data types**, which store values.

A scalar data type can have subtypes. A subtype has the same valid operations as its base type.

PL/SQL predefines many types and subtypes.

The PL/SQL scalar data types are:

- The SQL data types
- PLS\_INTEGER

### Scalar Types

BINARY\_DOUBLE  
BINARY\_FLOAT  
BINARY\_INTEGER  
DEC  
DECIMAL  
DOUBLE PRECISION  
FLOAT  
INT  
INTEGER  
NATURAL  
NATURALN  
NUMBER  
NUMERIC  
PLS\_INTEGER  
POSITIVE  
POSITIVEN  
REAL  
SIGNTYPE  
SMALLINT

CHAR  
CHARACTER  
LONG  
LONG RAW  
NCHAR  
NVARCHAR2  
RAW  
ROWID  
STRING  
UROWID  
VARCHAR  
VARCHAR2

BOOLEAN

DATE

### Composite Types

RECORD  
TABLE  
VARRAY

### Reference Types

REF CURSOR  
REF object\_type

### LOB Types

BFILE  
BLOB  
CLOB  
NCLOB

# TIPOS DE SENTENCIAS SQL

- **TIPOS DE “SUB” LENGUAJES**

**DML (Data Manipulation Language)**

Permite manipular y consultar la información que contienen las tablas. (SELECT, INSERT, DELETE, UPDATE,...)

**DDL (Data Description Language)**

Crea y mantiene la estructura de la bbdd. Usada por programadores y administradores. (CREATE, DROP, ALTER,...)

**DCL (Data Control Language)**

Permite gestionar el acceso –confidencialiad- (GRANT, REVOKE,..) y las transacciones a la bbdd (COMMIT, ROLLBACK,...)

# Tipos de sentencias SQL

## SQL Language Statements

```
graph TD; A[SQL Language Statements] --> B[DML]; A --> C[DDL]; A --> D[DCL]; A --> E[TCL]; B --> B1[SELECT]; B --> B2[INSERT]; B --> B3[UPDATE]; B --> B4[DELETE]; C --> C1[CREATE]; C --> C2[ALTER]; C --> C3[DROP]; D --> D1[GRANT]; D --> D2[REVOKE]; E --> E1[BEGIN]; E --> E2[TRAN]; E --> E3[COMMIT]; E --> E4[TRAN]; E --> E5[ROLLBACK];
```

DML

SELECT  
INSERT  
UPDATE  
DELETE

DDL

CREATE  
ALTER  
DROP

DCL

GRANT  
REVOKE

TCL

BEGIN  
TRAN  
COMMIT  
TRAN  
ROLLBACK

## • LENGUAJE DESCRIPCIÓN/DEFINICION DE DATOS

- Permite crear, modificar y eliminar objetos de la base de datos (es decir, los metadatos).
- En un principio, un usuario puede crear sus propios objetos, a no ser que el administrador le retire los privilegios que permitan crear distintos objetos
- En Oracle, cada usuario de una base de datos tiene sus propios objetos (tablas, vistas, índices,...), y este conjunto de objetos se denomina ESQUEMA, que tendrá el mismo nombre que el usuario con el que se ha accedido.

*NOTA: Las instrucciones DDL generan acciones que no se pueden deshacer, por eso es conveniente usarlas con precaución y tener copias de seguridad cuando manipulamos la base de datos.*



# DDL: CREACIÓN DE BBDD

Son creadas por el administrador de la bbdd, y permite crear el contenedor que almacenará los distintos esquemas de los usuarios que van a utilizarla.

El comando para crear una bbdd

CREATE DATABASE **NombredemiBasedeDatos**;

```
CREATE DATABASE Basurilla2
ON PRIMARY
    (name= basurilla2,
    filename="c:\bbdds\basurilla2_data.mdf",
    size=6MB,
    maxsize=15MB,
    filegrowth=1MB)
LOG ON
    (name=gescom_log,
    filename="c:\bbdds\basurilla2_log.ldf",
    size=2MB,
    maxsize=15MB,
    filegrowth=1)
```

```
DROP DATABASE Basurilla2
```

Mensajes

Comandos completados correctamente.

# DDL: CREACIÓN DE TABLAS

Las **tablas** son los objetos donde vamos a almacenar los datos.

## **Tendremos que tener en cuenta:**

- Qué nombre le vamos a dar a la tabla.
- Qué nombre le vamos a dar a cada una de las columnas.
- Qué tipo y tamaño de datos vamos a almacenar en cada columna.
- Qué restricciones tenemos sobre los datos.
- Alguna otra información adicional que necesitemos



# DDL: CREACIÓN DE TABLAS

## Formato:

```
CREATE TABLE [esquema.]nom_tabla  
(  
    colum1    tipo_dato        [not null],  
    colum2    tipo_dato        [not null],  
    .....  
);
```

- Colum1, Colum2,...: Nombres de las columnas que forman la fila
- Tipo dato: indica el tipo de dato que va a contener
- NOT NULL: especifica si la columna puede contener valores nulos o no

```
CREATE TABLE tlibros(  
    codigo NUMBER(5),  
    titulo VARCHAR2(40),  
    codigoautor NUMBER(4),  
    codigoeditorial NUMBER(3),  
    precio NUMBER(5,2)  
);
```


# DDL: CREACIÓN DE TABLAS

**Los usuarios pueden consultar las tablas creadas por medio de la VISTA USER\_TABLES.**

Existen otras dos VISTAS que permiten obtener información de los objetos que son propiedad del usuario.

// objetos que son propiedad del usuario

//tablas, vistas, sinónimos y secuencias propiedad del usuario



```
//  
SELECT * FROM tabs;  
  
SELECT * FROM user_objects;  
  
SELECT * FROM user_catalog;
```



# TAREA 1 -- LA TÍPICA DE EMPLEADOS Y DEPARTAMENTOS

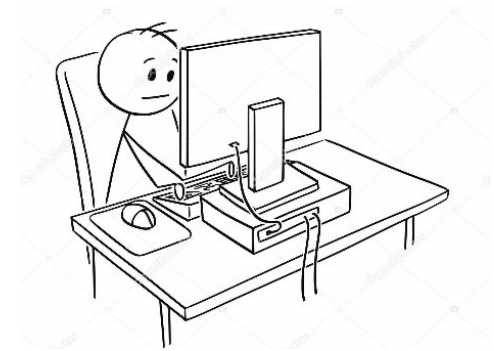


TABLA DEPARTAMENTOS

CodigoDepartamento	NombreDepartamento	Ubicación

TABLA DE EMPLEADOS

CodigoEmpleado	Nombre	Apellidos	Dirección	Teléfono	FechaNto	Sexo	Casado	Sueldo	CodigoDepartamento

```
-- TAREA 1 -- PS
CREATE TABLE tdepartamentos (
    cd_dpto NUMBER(2),
    nombre_dpto VARCHAR2(15),
    ubicacion VARCHAR2(10)
;

```

```
CREATE TABLE templeados (
    emp_no NUMBER(3),
    nombre_emp VARCHAR2(10),
    apellidos VARCHAR2(20),
    direccion VARCHAR2(10),
    telefono VARCHAR2(10),
    fec_nac DATE,
    sexo CHAR(1),
    casado CHAR(1),
    sueldo NUMBER(10),
    dept_no NUMBER(2)
;

```

## Tarea 1 -- La típica de empleados y departamentos -- PS

TABLA DEPARTAMENTOS		
CodigoDepartamento	NombreDepartamento	Ubicación

TABLA DE EMPLEADOS									
CodigoEmpleado	Nombre	Apellidos	Dirección	Teléfono	FechaNto	Sexo	Casado	Sueldo	CodigoDepartamento

**INTEGRIDAD DE DATOS** → los datos de la base de datos han de ajustarse a restricciones antes de almacenarse en ella.

**RESTRICCIÓN DE INTEGRIDAD:** será una regla que restringe el rango de valores para una o más columnas de la tabla.

**INTEGRIDAD REFERENCIAL:** garantiza que los valores de las columnas de una tabla dependan de los valores de otra columna de otra tabla.

Supongamos -- TVENTAS y TARTICULOS.

Nunca se dará la situación de insertar una venta con un artículo inexistente.

## RESTRICCIONES -- CONSTRAINTS

**RESTRICCIÓN:** condición que una o varias columnas deben cumplir obligatoriamente.

~~nombre restricción~~

Los diferentes tipos de restricción que existen son:

- PRIMARY KEY
- UNIQUE
- FOREIGN KEY
- CHECK
- DEFAULT

2 1 PRIMARY KEY

Es importante poner nombre a las restricciones, sino lo hará el propio SGBD (SYS\_C00n donde "n" será un número). Se aconseja la siguiente regla a la hora de poner nombre a las restricciones:

- Tres letras para el nombre de la tabla.
- Carácter de subrayado.
- Tres letras con la columna afectada por la restricción.
- Carácter de subrayado.
- Dos letras con la abreviatura del tipo de restricción. La abreviatura puede ser:

**PK = Primary Key.**  
**FK = Foreign Key.**  
**NN = Not Null.**  
**UK = Unique.**  
**CK = Check (validación).**

## DDL -- EJEMPLO

```
7 CREATE TABLE tlibros(  
8     codigo NUMBER(5),  
9     titulo VARCHAR2(40),  
0     codigoautor NUMBER(4),  
1     codigoeditorial NUMBER(3),  
2     precio NUMBER(5,2) DEFAULT 0,  
3     CONSTRAINT pk_tlibros_codigo PRIMARY KEY (codigo),  
4     CONSTRAINT uq_tlibros_tituloautor UNIQUE (titulo,codigoautor),  
5     CONSTRAINT ck_tlibros_codigoeditorial CHECK (codigoeditorial IS NOT NULL),  
6     CONSTRAINT fk_tlibros_editorial FOREIGN KEY (codigoeditorial) REFERENCES tedit  
7     CONSTRAINT fk_tlibros_autores FOREIGN KEY (codigoautor) REFERENCES tautores(co  
8     CONSTRAINT ck_tlibros_precioonulo CHECK (precio IS NOT NULL) DISABLE,  
9     CONSTRAINT ck_precio_positivo CHECK (precio>=0)
```

# DEFINIENDO TABLAS



# TIPOS DE RESTRICCIÓN SOBRE UNA TABLA

Claves primarias

Claves ajenas

Obligatoriedad

Valores por defecto

Verificación de condiciones

Valores únicos



Para definir las restricciones de utiliza la cláusula **CONSTRAINT** y pueden ser:

- **Restricción de columna:** Restringe una sola columna y se define como parte de la definición de la columna
- **Restricción de tabla:** Restringe un grupo de columnas y se define al final de la definición de la tabla, una vez definida todas las columnas



# RESTRICCIONES EN CREATE TABLE

- **Restricción de columna:** Restringe una sola columna y se define como parte de la definición de la columna

```
CREATE TABLE nom_tabla (  
  column1      TIPO_DE_DATO  
              [CONSTRAINT nombre_restricción]  
              [NOT NULL]  
              [UNIQUE]  
              [PRIMARY KEY]  
              [DEFAULT valor]  
              [REFERENCES NOM_TABLA [(columna [,columna])] [on delete cascade]]  
              [CHECK condicion],  
  column2      TIPO_DE_DATO  
  .....  
) [TABLESPACE espacio_de_tabla];
```

Ejemplos →



## RESTRICCIONES EN CREATE TABLE

```
CREATE TABLE empleados (  
    nombre          varchar2(25)      PRIMARY KEY,  
    edad            number             CHECK (edad between 18 and 35),  
    cod_provincia   number(2)          REFERENCES tprovincias  
    ON DELETE CASCADE  
);
```

# RESTRICCIONES EN CREATE TABLE

## Restricción de tabla:

```
CREATE TABLE nom_tabla (  
    Colum1 TIPO_DE_DATO,  
    Colum2 TIPO_DE_DATO,  
    Colum3 TIPO_DE_DATO,  
    .....  
    [CONSTRAINT nombre_restricción  
        { [Unique] | [PRIMARY KEY] (columna [,columna]) },  
    [constraint nombre_restricción  
        [FOREIGN KEY (columna, [columna])  
        REFERENCES nombre_tabla [(columna [,columna])]  
        [ON DELETE CASCADE]],  
    [CONSTRAINT nombre_restricción  
        [CHECK (condicion)],  
    .....  
) [TABLESPACE espacio_de_tabla];
```

Ejemplos



# RESTRICCIONES EN CREATE TABLE

```
CREATE TABLE tlibros(  
  codigo NUMBER(5),  
  titulo VARCHAR2(40),  
  codigoautor NUMBER(4),  
  codigoeditorial NUMBER(3),  
  precio NUMBER(5,2) DEFAULT 0,  
  CONSTRAINT pk_tlibros_codigo PRIMARY KEY (codigo),  
  CONSTRAINT uq_tlibros_tituloautor UNIQUE (titulo,codigoautor),  
  CONSTRAINT ck_tlibros_codigoeditorial CHECK (codigoeditorial IS NOT NULL),  
  CONSTRAINT fk_tlibros_editorial FOREIGN KEY (codigoeditorial) REFERENCES tedit  
  CONSTRAINT fk_tlibros_autores FOREIGN KEY (codigoautor) REFERENCES tautores(co  
  CONSTRAINT ck_tlibros_precioonulo CHECK (precio IS NOT NULL) DISABLE,  
  CONSTRAINT ck_precio_positivo CHECK (precio>=0)
```

# RESTRICCIÓN UNIQUE

Permite que no se puedan repetir valores en la columna  
Oracle crea un índice automáticamente cuando se habilita esta restricción y lo borra al deshabilitarla

```
CREATE TABLE tusuarios (  
    login VARCHAR2(25) CONSTRAINT usu_log_UK    UNIQUE  
);
```

```
CREATE TABLE tusuarios (login VARCHAR2(25) UNIQUE);
```

```
CREATE TABLE tusuarios (  
    login VARCHAR2 (25),  
    correo VARCHAR2 (25),
```

```
CONSTRAINT tusuarios_uk UNIQUE (login, correo));
```

error que empieza en la línea: 5 del comando :

insert into tusuarios values('2')

mensaje de error -

ORA-00001: unique constraint (SYSTEM.USU\_LOG\_UK) violated

# RESTRICCIÓN NOT NULL

Una columna con una restricción NOT NULL obliga a que tenga un valor que no sea nulo

```
CREATE TABLE tusuarios (  
  login VARCHAR2(25),  
  f_nacimiento DATE CONSTRAINT usu_fcha_nto NOT NULL  
);
```

```
CREATE TABLE tusuarios (  
  login VARCHAR2(25),  
  f_nacimiento DATE NOT NULL  
);
```

error que empieza en la línea: 16 del comando :

insert into tusuarios values('2', NULL)

informe de error -

ORA-01400: cannot insert NULL into ("SYSTEM"."TUSUARIOS"."F\_NACIMIENTO")

# CLAVE PRIMARIA. RESTRICCIÓN PRIMARY KEY

- **Clave primaria:** columnas o conjunto de columnas que identifican unívocamente a cada fila.
- Automáticamente se crea un índice que facilita el acceso a la tabla.
- Debe de ser única, no nula y obligatoria
- Solo una primary key por tabla
- Si esta clave referencia una columna o columnas de otra tabla se denomina clave ajena

**NOTA:** La primary key automáticamente aparece como NOT NULL en la descripción de la tabla

Primary Key



Employee Id must be Unique

Employee Id	Name	Salary
101	Hitesh	250000
102	Gaurav	40000
103	Anjali	25000
104	Gaurav	60000
105	Poonam	25000

## CLAVE PRIMARIA. RESTRICCIÓN PRIMARY KEY

RESTRICCIÓN DE COLUMNA	<pre>CREATE TABLE nombre_tabla (     colum1 TIPO_DE_DATO [<b>CONSTRAINT</b> nombre_restricción] <b>PRIMARY KEY</b>,     colum2 TIPO_DE_DATO,     ..... ) [TABLESPACE espacio_de_tabla];</pre>
RESTRICCIÓN DE TABLA	<pre>CREATE TABLE nombre_tabla (     colum1 TIPO_DE_DATO,     colum2 TIPO_DE_DATO,     .....     [<b>CONSTRAINT</b> nombre_restricción] <b>PRIMARY KEY</b> (columna [, columna]),     ..... ) [TABLESPACE espacio_de_tabla];</pre>



## CLAVE PRIMARIA. RESTRICCIÓN PRIMARY KEY

```
CREATE TABLE TBLOQUEPISOS (  
  CALLE          VARCHAR2(30) NOT NULL,  
  NUMERO         NUMBER(3) NOT NULL,  
  PISO           NUMBER(2) NOT NULL,  
  PUERTA         CHAR(1) NOT NULL,  
  CODIGO_POSTAL  NUMBER(5),  
  METROS         NUMBER(5),  
  COMENTARIOS   VARCHAR2(30),  
  COD_ZONA       NUMBER(2),  
  DNI            VARCHAR2(10),  
  CONSTRAINT TBLQ_PK PRIMARY KEY  
    (CALLE, NUMERO, PISO, PUERTA)  
); /* RESTRICCIÓN DE TABLA*/
```

```
CREATE TABLE TBLOQUEPISOS (  
  CALLE          VARCHAR2(30) NOT NULL,  
  NUMERO         NUMBER(3) NOT NULL,  
  PISO           NUMBER(2) NOT NULL,  
  PUERTA         CHAR(1) NOT NULL,  
  CODIGO_POSTAL  NUMBER(5),  
  METROS         NUMBER(5),  
  COMENTARIOS   VARCHAR2(30),  
  COD_ZONA       NUMBER(2),  
  DNI            VARCHAR2(10),  
  PRIMARY KEY  
    (CALLE, NUMERO, PISO, PUERTA)  
); /* RESTRICCIÓN DE TABLA*/
```

# CLAVE PRIMARIA. RESTRICCIÓN PRIMARY KEY

```
CREATE TABLE tzonas (  
    cod_zona          NUMBER(3) PRIMARY KEY,  
    nombrezona        VARCHAR2(15) NOT NULL,  
    masdatos          VARCHAR2(60)  
);
```

error que empieza en la línea: 16 del comando :

```
insert into tzonas values (NULL,'21','HHHHHHHOOOOOOLLLLLLAAAAA')
```

Informe de error -

ORA-01400: cannot insert NULL into ("SYSTEM"."TZONAS"."COD\_ZONA")

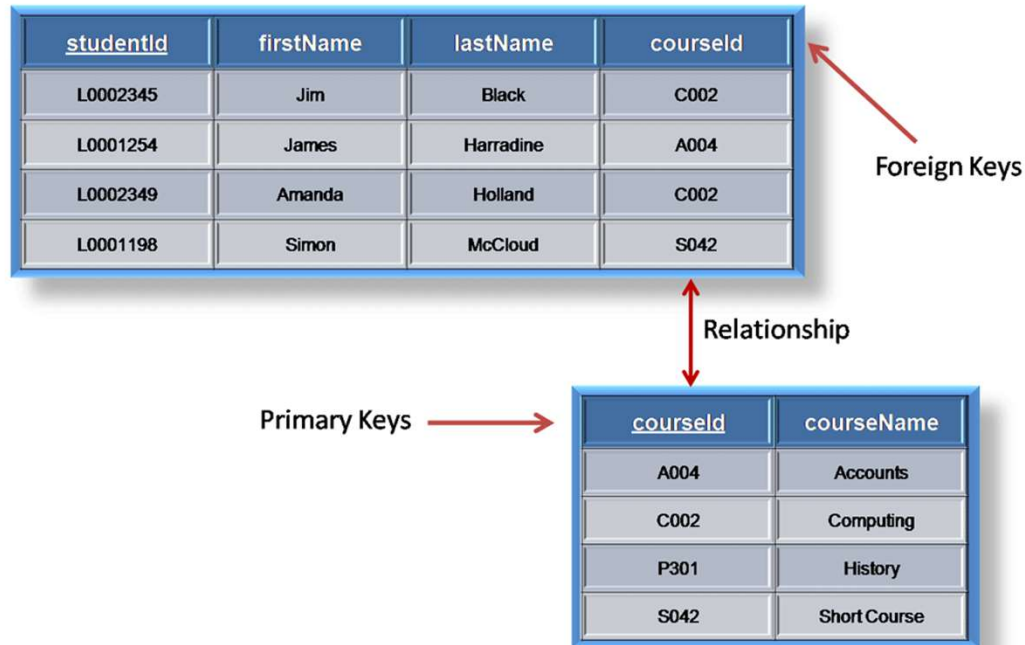
```
CREATE TABLE tzonas (  
    cod_zona          NUMBER(3) CONSTRAINT zonas_pk PRIMARY KEY,  
    nombrezona        VARCHAR2(15) NOT NULL,  
    masdatos          VARCHAR2(60)  
);
```

# CLAVES AJENAS. RESTRICCIÓN FOREIGN KEY

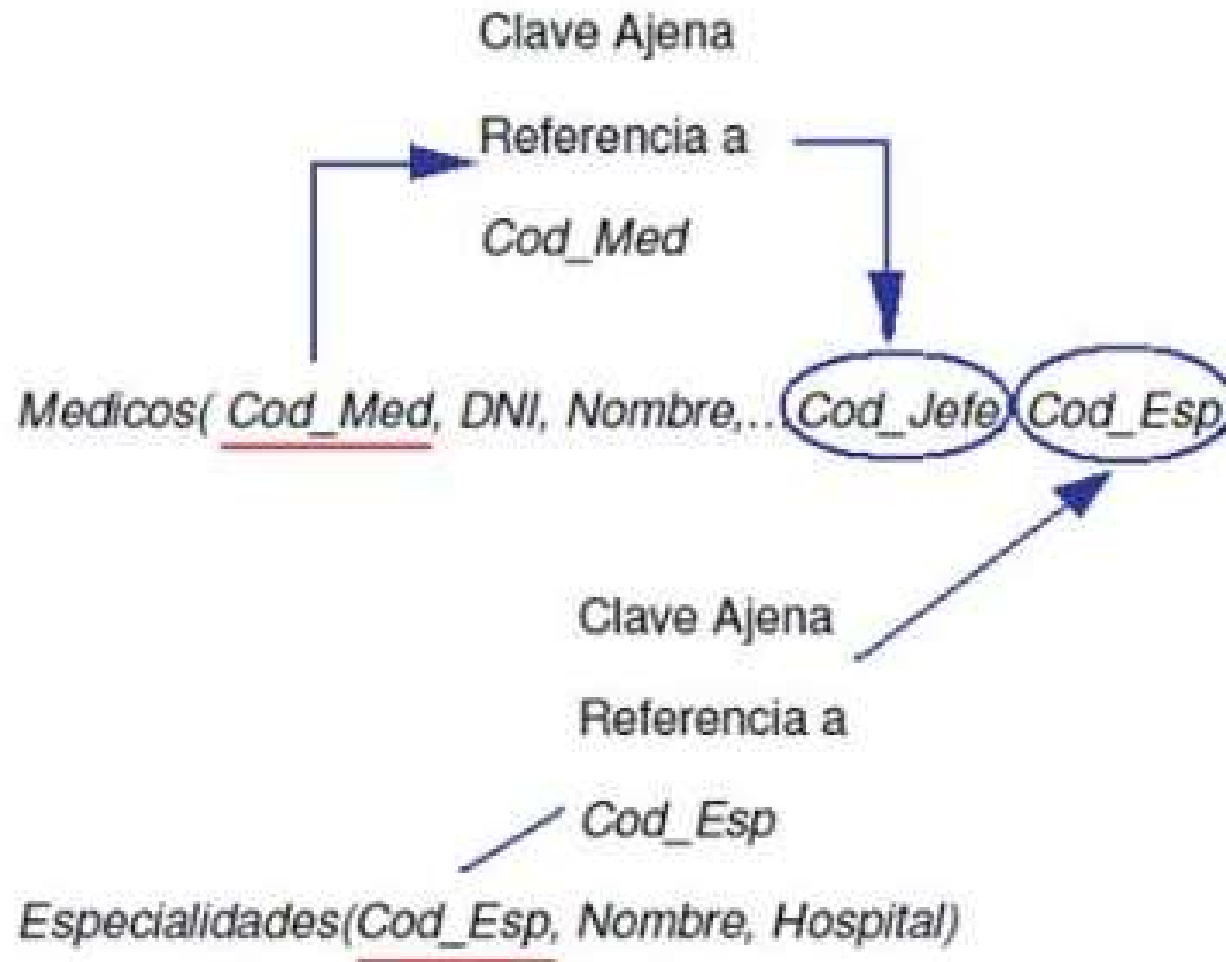
**Clave ajena:** es una o varias columnas asociadas a una clave primaria o a una clave única que pertenece a otra o la misma tabla

Puede existir más de una clave ajena en una tabla.

Las claves ajenas deben de tener un valor NULL o un valor igual al de la clave referenciada (REGLA DE INTEGRIDAD REFERENCIAL)



# CLAVES AJENAS. RESTRICCIÓN FOREIGN KEY



# CLAVES AJENAS. RESTRICCIÓN FOREIGN KEY

Podemos utilizar cualquiera de los dos formatos siguientes para definir claves ajenas:

**FORMATO 1:** la clave ajena se define en la restricción de la columna usando  
**REFERENCES**

```
CREATE TABLE nombre_tabla
(
    colum1 TIPO_DE_DATO
        [CONSTRAINT nombre_restricción]
        REFERENCES nombre_tabla [(columna)]
        [ON DELETE {CASCADE | SET NULL} ],
    colum2 TIPO_DATO,
    .....
) [TABLESPACE espacio_de_tabla];
```

# CLAVES AJENAS. RESTRICCIÓN FOREIGN KEY

Podemos utilizar cualquiera de los dos formatos siguientes para definir claves ajenas:

**FORMATO 2:** la clave ajena se define al final de todas la columnas usando **FOREIGN KEY** y **REFERENCES**

```
CREATE TABLE nombre_tabla (  
    colum1    TIPO_DE_DATO,  
    colum2    TIPO_DE_DATO,  
    .....  
    [CONSTRAINT nombre_restricción  
        FOREIGN KEY (columna [, columna])  
        REFERENCES nombre_tabla [(columna [, columna])]  
        [ON DELETE {CASCADE / SET NULL}],  
    .....  
)[TABLESPACE espacio_de_tabla];
```

## TAREA 2 – La de las personas y las provincias

**Diseña las siguientes tablas con las PKs y FKs que te indica la imagen:**

TPERSONAS (cd\_pers, dni, nombre, dirección, población, **cd\_prov (FK)**)



TPROVINCIAS (cd\_prov, nom\_prov)

## TAREA 2 – La de las personas y las provincias -- ps

TPERSONAS (cd\_pers, dni, nombre, dirección, población, cd\_prov (FK))

TPROVINCIAS (cd\_prov, nom\_prov)

```
CREATE TABLE tprovincias (  
    cd_prov NUMBER(2) PRIMARY KEY,  
    nom_prov VARCHAR2(15)  
);
```

```
CREATE TABLE tpersonas (  
    cd_pers NUMBER(2) PRIMARY KEY,  
    dni VARCHAR2(10),  
    nombre VARCHAR2(20),  
    direccion VARCHAR2(30),  
    poblacion VARCHAR2(10),  
    cd_prov NUMBER(2) NOT NULL,  
    CONSTRAINT FK_pers_prov  
        FOREIGN KEY (cd_prov)  
        REFERENCES tprovincias(cd_prov)  
);
```

Informe de error -

ORA-02449: claves únicas/primarias en la tabla referidas por cla  
02449. 00000 - "unique/primary keys in table referenced by fore

\*Cause: An attempt was made to drop a table with unique or  
primary keys referenced by foreign keys in another ta

\*Action: Before performing the above operations the table, dro  
foreign key constraints in other tables. You can see



## CLAVES AJENAS. RESTRICCIÓN FOREIGN KEY. OPCIÓN ON DELETE CASCADE

Se utiliza cuando al borrar las filas asociadas con claves primarias (TPROVINCIAS), deseamos que se borren automáticamente las correspondientes filas con clave ajena, dependientes de otras tablas (TPERSONAS).

```
CREATE TABLE tpersonas (  
    cd_pers NUMBER(2) PRIMARY KEY,  
    dni VARCHAR2(10),  
    nombre VARCHAR2(20),  
    direccion VARCHAR2(30),  
    poblacion VARCHAR2(10),  
    cd_prov NUMBER(2) NOT NULL,  
    CONSTRAINT FK_pers_prov  
        FOREIGN KEY (cd_prov)  
        REFERENCES tprovincias(cd_prov)  
        ON DELETE CASCADE  
);
```

## CLAVES AJENAS. RESTRICCIÓN FOREIGN KEY. OPCIÓN ON DELETE SET NULL

Se utiliza cuando al borrar las filas asociadas con claves primarias (TPROVINCIAS), pero no queremos que se borren las correspondientes filas con clave ajena, dependientes de otras tablas (TPERSONAS), en este caso pone a NULL dicho campo en TPERSONAS

```
CREATE TABLE tpersonas (  
    cd_pers NUMBER(2) PRIMARY KEY,  
    dni VARCHAR2(10),  
    nombre VARCHAR2(20),  
    direccion VARCHAR2(30),  
    poblacion VARCHAR2(10),  
    cd_prov NUMBER(2) NOT NULL,  
    CONSTRAINT FK_pers_prov  
        FOREIGN KEY (cd_prov)  
        REFERENCES tprovincias(cd_prov)  
        ON DELETE SET NULL  
);
```

# Restricciones en create table

## RESUMEN

- **RESTRICCIONES DE BORRADO Y ACTUALIZACIÓN:** Una fila no se puede borrar si es referenciada por una clave ajena. Tampoco será posible la actualización de la clave primaria.
- **Borrado en cascada (ON DELETE CASCADE):** Si borramos una fila de una tabla maestra, todas las filas detalle cuya clave ajena sea referenciada se borrarán automáticamente. La restricción se declara en la tabla detalle. El mensaje “n filas borradas” (n row deleted) solo indica las filas borradas en la tabla maestra.
- **Borrado en cascada (ON DELETE SET NULL):** Si borramos una fila de una tabla maestra, y todas las filas detalle cuya clave ajena sea referenciada se actualizarán a null.

## CONSTRAINT

- Permite definir un nombre a las restricciones, tanto de clave primaria como de clave ajena, facilitando la interpretación de errores cuando estos se produzcan.
- Por defecto si no se utiliza el CONSTRAINT, ORACLE asigna un nombre a la restricción: SYS\_C00n, donde "n" → número asignado automáticamente por ORACLE.

# RESTRICCIONES EN CREATE TABLE

## CONSTRAINT

```
CREATE TABLE tprovincias  
(  
  cod_prov NUMBER(2)   CONSTRAINT PROV_PK  PRIMARY KEY,  
  nom_prov      VARCHAR2(15)  
);
```

**PROV\_PK**, es el nombre de la restricción de la clave primaria. Cualquier error que se produzca referente a ella vendrá referenciado por:

nom\_usuario.**PROV\_PK**

**Ejemplo:** Dar de alta una provincia con un código que ya existe  
error en linea 1:

ORA-0001: restricción única (**U7. PROV\_PK**) violada



## RESTRICCIONES EN CREATE TABLE

```
CREATE TABLE tlibros(  
  codigo NUMBER(5),  
  titulo VARCHAR2(40),  
  codigoautor NUMBER(4),  
  codigoeditorial NUMBER(3),  
  precio NUMBER(5,2) DEFAULT 0,  
  CONSTRAINT pk_tlibros_codigo PRIMARY KEY (codigo),  
  CONSTRAINT uq_tlibros_tituloautor UNIQUE (titulo,codigoautor),  
  CONSTRAINT ck_tlibros_codigoeditorial CHECK (codigoeditorial IS NOT NULL),  
  CONSTRAINT fk_tlibros_editorial FOREIGN KEY (codigoeditorial) REFERENCES tedito(  
  CONSTRAINT fk_tlibros_autores FOREIGN KEY (codigoautor) REFERENCES tautores(co  
  CONSTRAINT ck_tlibros_precioonulo CHECK (precio IS NOT NULL) DISABLE,  
  CONSTRAINT ck_precio_positivo CHECK (precio>=0)
```

# RESTRICCIONES DEFAULT

DEFAULT: permite asignar un valor por defecto

```
CREATE TABLE tusuarios_def (  
    id_usuario VARCHAR2(3) PRIMARY KEY,  
    pais VARCHAR2(20) DEFAULT 'España',  
    fec_ing DATE DEFAULT SYSDATE  
);  
  
INSERT INTO tusuarios_def VALUES ('001', DEFAULT, DEFAULT);  
INSERT INTO tusuarios_def VALUES ('002', DEFAULT, DEFAULT);  
INSERT INTO tusuarios_def VALUES ('003', DEFAULT, DEFAULT);
```

la de Script x	Resultado de la Consulta x	
  SQL	Todas las Filas Recuperadas: 3 en 0,004 segundos	
ID_USUARIO	PAIS	FEC_ING
001	España	10/11/18
002	España	10/11/18
003	España	10/11/18

# RESTRICCIONES VALIDACIÓN CHECK

CHECK: permite comprobar que los valores que se van a introducir en el campo cumplen una restricción

```
CREATE TABLE tcreditos (  
    credito    NUMBER(4) CHECK (credito BETWEEN 0 AND 2000),  
    funcionario CHAR(1),  
    CONSTRAINT funcionario_ck CHECK (funcionario in ('S','N'))  
);
```

Error que empieza en la línea: 8 del comando :

insert into tcreditos values (15000)

Informe de error -

ORA-01438: value larger than specified precision allowed for this column

Error que empieza en la línea: 10 del comando :

insert into tcreditos values (1500,'a')

Informe de error -

ORA-02290: check constraint (SYSTEM.FUNCIONARIO\_CK) violated

Una misma columna puede tener varios CHECK asociados a ella, para ello ponemos varios CONSTRAINT seguidos y separados por comas.



# RESTRICCIONES VALIDACIÓN CHECK

## Ejemplos

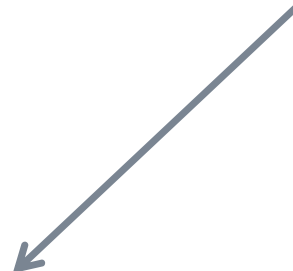
```
edad NUMBER(4) CHECK (edad >18 AND edad <65)
sexo CHAR(1) CONSTRAINT CK_sexo CHECK( sexo IN ('h','m'))
dia_semana VARCHAR2(10) CONSTRAINT CK_dia_sem
    CHECK (dia_semana IN ('lunes','martes','miercoles','jueves','viernes'))
dpt_no NUMBER (2) CONSTRAINT dpt_no_ck CHECK (dpt_no IN (10,20,30,40))
dpt_no NUMBER(2) CHECK (dpt_no =10 OR dpt_no=20)
```

# VIENDO LAS CONSTRAINTS

## USER\_CONSTRAINTS

USER\_CONSTRAINTS describes all constraint definitions on tables owned by the current user.

```
CREATE TABLE tautores(
  codigo NUMBER(4) NOT NULL
    CONSTRAINT CK_tautores_codigo
      CHECK (codigo>=0),
  nombre VARCHAR2(30) NOT NULL ,
  CONSTRAINT PK_tautores_codigo
    PRIMARY KEY (codigo),
  CONSTRAINT UQ_tautores_nombre
    UNIQUE (nombre)
);
```



Nombre	¿Nulo?	Tipo
OWNER		VARCHAR2(120)
CONSTRAINT_NAME	NOT NULL	VARCHAR2(30)
CONSTRAINT_TYPE		VARCHAR2(1)
TABLE_NAME	NOT NULL	VARCHAR2(30)
SEARCH_CONDITION		LONG
R_OWNER		VARCHAR2(120)
R_CONSTRAINT_NAME		VARCHAR2(30)
DELETE_RULE		VARCHAR2(9)
STATUS		VARCHAR2(8)
DEFERRABLE		VARCHAR2(14)
DEFERRED		VARCHAR2(9)
VALIDATED		VARCHAR2(13)
GENERATED		VARCHAR2(14)
BAD		VARCHAR2(3)
RELY		VARCHAR2(4)
LAST_CHANGE		DATE
INDEX_OWNER		VARCHAR2(30)
INDEX_NAME		VARCHAR2(30)
INVALID		VARCHAR2(7)
VIEW_RELATED		VARCHAR2(14)

	⚡ CONSTRAINT_NAME	⚡ CONSTRAINT_TYPE	SEARCH_CONDITION	⚡ STATUS	⚡ VALIDATED	⚡ TABLE_NAME
1	SYS_C007236	C	"CODIGO" IS NOT NULL	ENABLED	VALIDATED	TAUTORES
2	SYS_C007237	C	"NOMBRE" IS NOT NULL	ENABLED	VALIDATED	TAUTORES
3	CK_TAUTORES_CODIGO	C	codigo>=0	ENABLED	VALIDATED	TAUTORES
4	PK_TAUTORES_CODIGO	P	(null)	ENABLED	VALIDATED	TAUTORES
5	UQ_TAUTORES_NOMBRE	U	(null)	ENABLED	VALIDATED	TAUTORES

# VIENDO LAS CONSTRAINTS

```
-- Julia Triana -- BBDD -- DAM1  
-- Trabajando con CONSTRAINTS
```

```
CREATE TABLE ttablamolona (  
    campo1 NUMBER(3) PRIMARY KEY,  
    campo2 VARCHAR2(30) NOT NULL,  
    campo3 VARCHAR2(30) UNIQUE,  
    campo4 VARCHAR2(30),  
    campo5 NUMBER(3),  
    campo6 VARCHAR2(30),  
    CONSTRAINT UQCAMPO4 UNIQUE (campo4),  
    CONSTRAINT CHCAMPO5MAYOR33 CHECK (campo5>=33),  
    CONSTRAINT FKCAMPO6 FOREIGN KEY (campo6) REFERENCES totratabla_molona(campo1)  
);
```

Salida de Script x Resultado de la Consulta x Resultado de la Consulta 1 x

SQL | Todas las Filas Recuperadas: 6 en 0,005 segundos

	OWNER	CONSTRAINT_NAME	CONSTRAINT_TYPE	TABLE_NAME
1	DAM1	FKCAMPO6	R	TTABLAMOLONA
2	DAM1	SYS C007396	C	TTABLAMOLONA
3	DAM1	CHCAMPO5MAYOR33	C	TTABLAMOLONA
4	DAM1	SYS C007398	P	TTABLAMOLONA
5	DAM1	SYS C007399	U	TTABLAMOLONA
6	DAM1	UQCAMPO4	U	TTABLAMOLONA

## Tarea 3 – La de la complicación de los empleados y dptos.

```
-- TAREA 1 -- PS
CREATE TABLE tdepartamentos (
  cd_dpto NUMBER(2),
  nombre_dpto VARCHAR2(15),
  ubicacion VARCHAR2(10)
);

CREATE TABLE templeados (
  emp_no NUMBER(3),
  nombre_emp VARCHAR2(10),
  apellidos VARCHAR2(20),
  direccion VARCHAR2(10),
  telefono VARCHAR2(10),
  fec_nac DATE,
  sexo CHAR(1),
  casado CHAR(1),
  sueldo NUMBER(10),
  dept_no NUMBER(2)
);
```



Partimos de la misma idea que teníamos en la Tarea1 pero....

- Los campos cd\_dpto y emp\_no serán PKs en sus tablas correspondientes.
- Los departamentos solo podrán estar en Almeria, Soria o Valladolid.
- En obligatorio que se introduzca la fecha de nacimiento.
- El campo "casado" solo podrá tener los valores S o N. Ambos en mayúscula.
- Por defecto, todos los trabajadores tendrán un sueldo de 3000,50€.
- El campo dept\_no de la tabla de empleados será una FK hacia el campo cd\_dpto en la tabla de departamentos.
- No se podrán repetir números de teléfono entre los trabajadores.