

# Unidad 2:

## Bases de Datos Relacionales

### Objetivos

- Explicar las características fundamentales del modelo Entidad-Relación (E-R)
- Describir la estructura del modelo relacional
- Realizar operaciones básicas sobre tablas

# Unidad 2:

## Bases de Datos Relacionales

1. Introducción
2. El modelo E-R
3. Estructura del modelo relacional

# Introducción

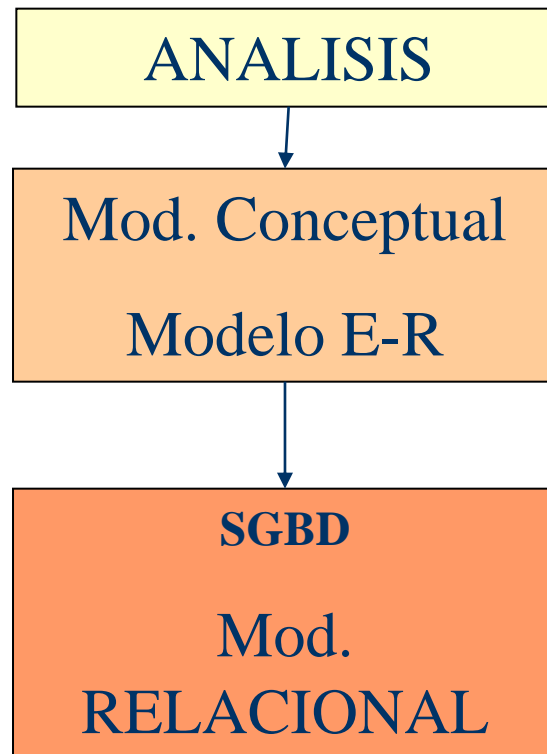
- Características básicas de uno de los modelos conceptuales más usados: E-R
- Características básicas del modelo relacional
- Paso del modelo conceptual E-R al modelo relacional (modelo lógico)
- Fundamentos del modelo relacional para la manipulación de los datos con SQL y PL/SQL

# Introducción

- existen varios esquemas a realizar para poder representar en forma de base de datos informática un problema
- Uno de los esquemas es el conceptual (representa la información independientemente del SGBD usado)
- Aparece en 1976 (Peter P. Chen): Entidad/Interrelación y con el tiempo evoluciona a lo que conocemos como E-R. Es un modelo conceptual independiente del SGBD

# El Modelo Entidad-Relación: E-R

## Diseño de una Base de Datos



# El Modelo Entidad-Relación: E-R

## Componentes del modelo E-R

- Entidades
- Interdependencias o Relaciones
- Atributos

# El Modelo Entidad-Relación: E-R

## Elementos del modelo E-R: ENTIDAD

*Objeto acerca del cual se pueda almacenar información en la BD*

Ejemplo: ALUMNOS, EMPLEADOS, HIJOS, DEPARTAMENTOS

Se representan por:

**nombre\_entidad**

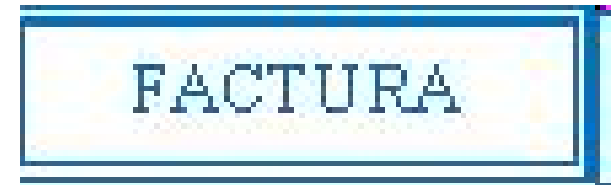
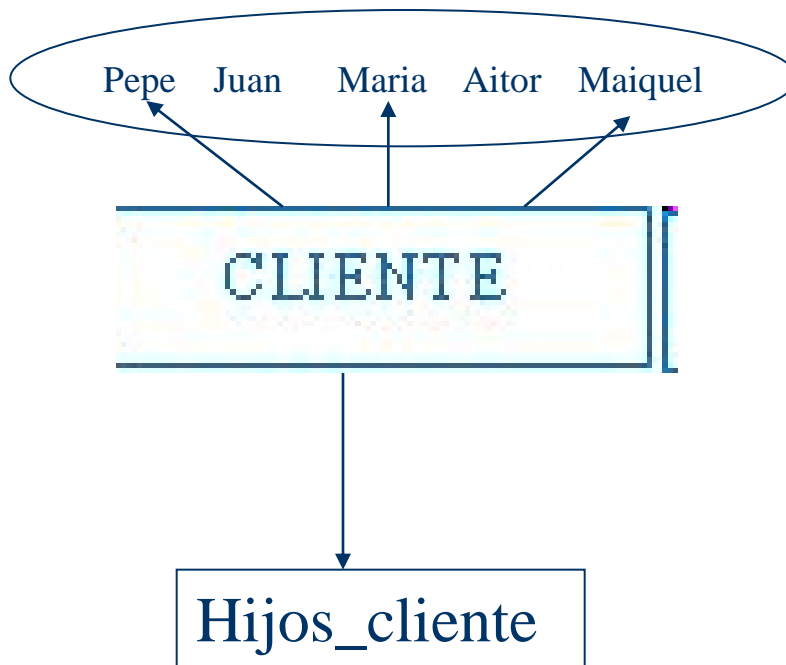
Ocurrencia: sería un ejemplar concreto de la entidad: p.e. Pepe

Tipos de Entidad:

- **Entidades Fuertes:** No dependen de otra entidad para su existencia.  
Ejemplo: EMPLEADO, DEPARTAMENTOS
- **Entidades Débiles:** Dependen de otra entidad para su existencia.  
Ejemplo: HIJOS\_EMPLEADO

# El Modelo Entidad-Relación: E-R

## Elementos del modelo E-R: ENTIDAD





# El Modelo Entidad-Relación: E-R

## Elementos del modelo E-R: **RELACIONES**

*Asociación entre dos o más entidades que generan información adicional a la de las entidades que las producen*

Se representan por:



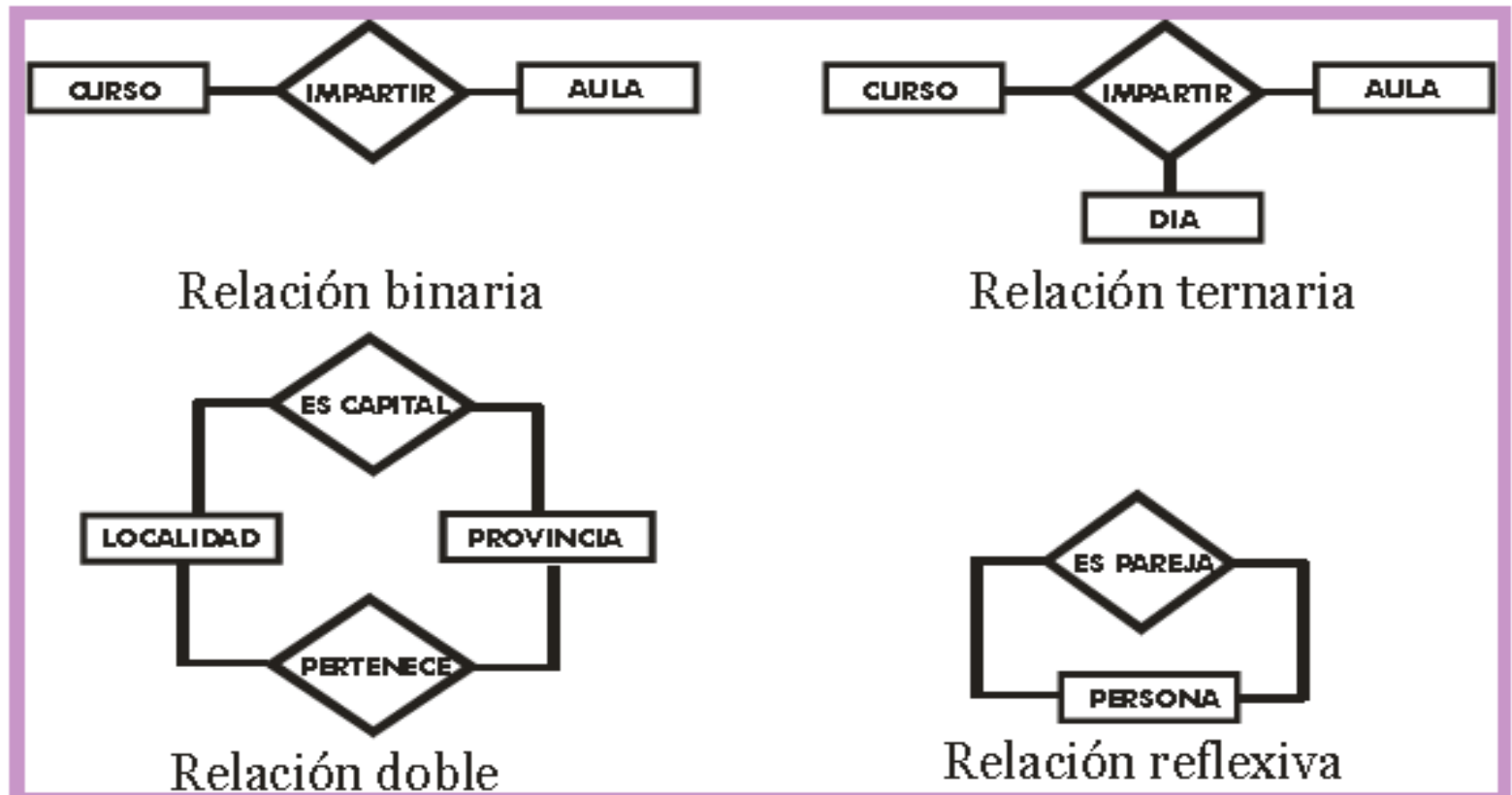
suelen ser verbos

Ejemplo:



# El Modelo Entidad-Relación: E-R

## Elementos del modelo E-R: **RELACIONES**



# El Modelo Entidad-Relación: E-R

## Elementos del modelo E-R: **RELACIONES**

Tipos de Relaciones:

- **Relaciones de GRADO 1:** Relación de una entidad consigo misma
- **Relaciones de GRADO 2:** Relación entre dos entidades
- **Relaciones de GRADO N:** Relación entre más de dos entidades

# El Modelo Entidad-Relación: E-R

## Relaciones de Grado 1

Se denomina *Relación Unaria o Recursiva*

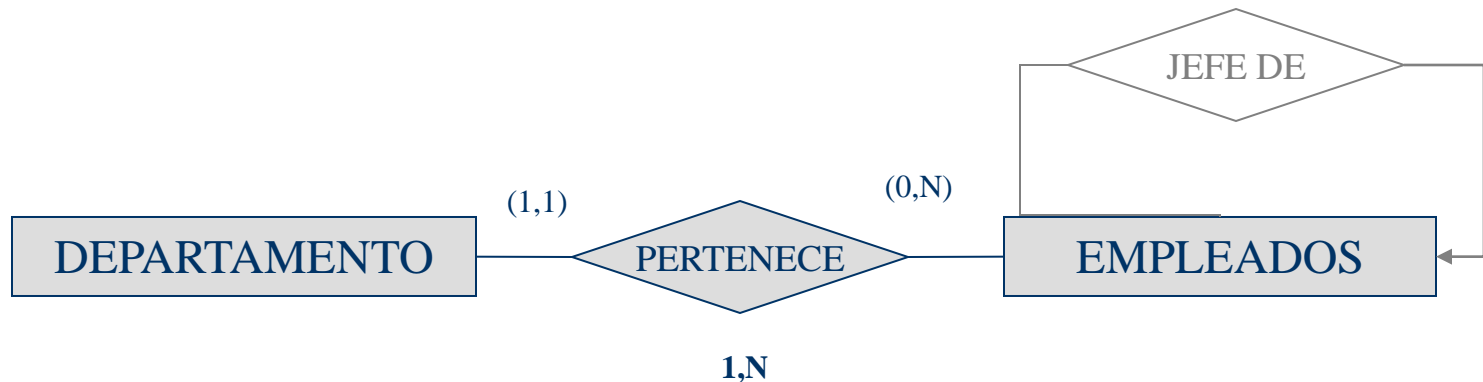


Otro ejemplo: Piezas de un artículo

# El Modelo Entidad-Relación: E-R

## Relaciones de Grado 2

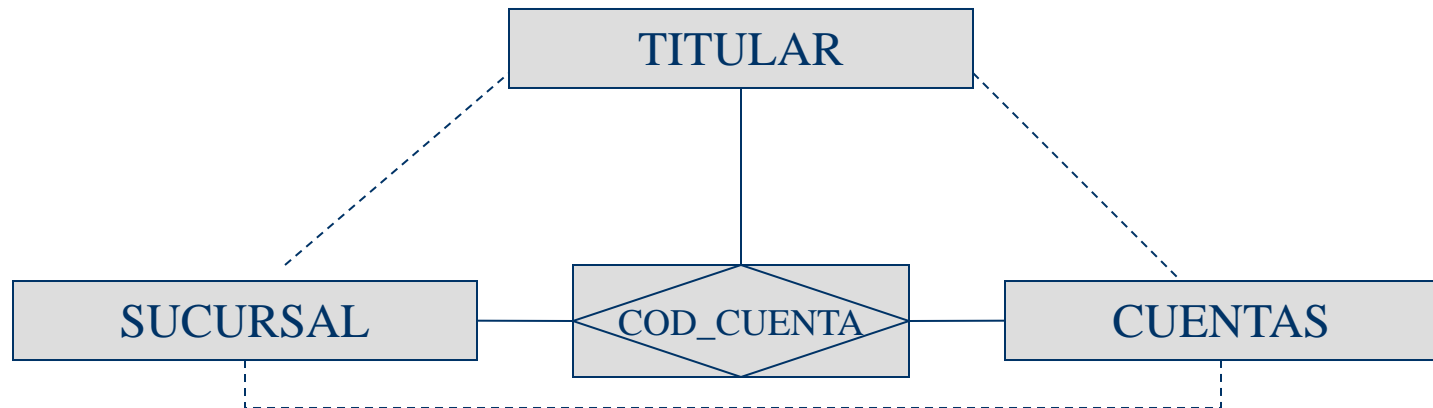
Se denomina *Relación binaria*



# El Modelo Entidad-Relación: E-R

## Relaciones de Grado N


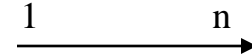
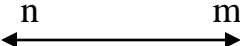
Aquella en la que intervienen más de dos entidades



# El Modelo Entidad-Relación: E-R

## Elementos del modelo E-R: **RELACIONES**

**Tipos de correspondencia** en una relación es la asociación entre dos entidades: (ej. Relación binaria)

- **1:1** 
- **1:n** 
- **n:m** 

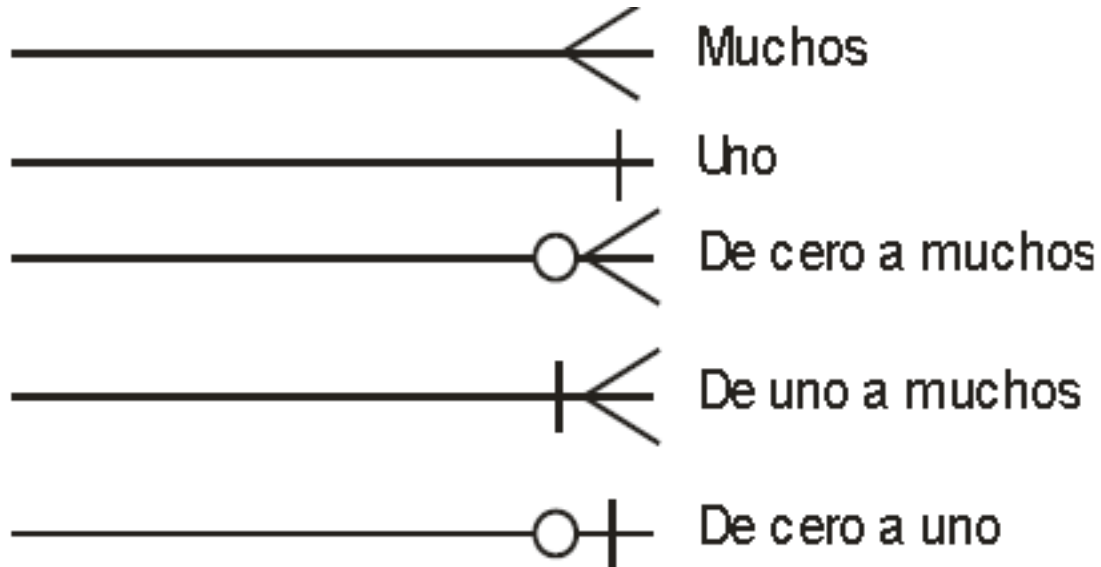
**Cardinalidad** del tipo de correspondencia, representa el número máximo y mínimo de ocurrencias de cada una de las entidades. (max,min)

Las relaciones **n:m** dan lugar a una nueva entidad denominada **renacida**

# El Modelo Entidad-Relación: E-R

## Elementos del modelo E-R: **RELACIONES**

Otra forma de representar la correspondencia es:

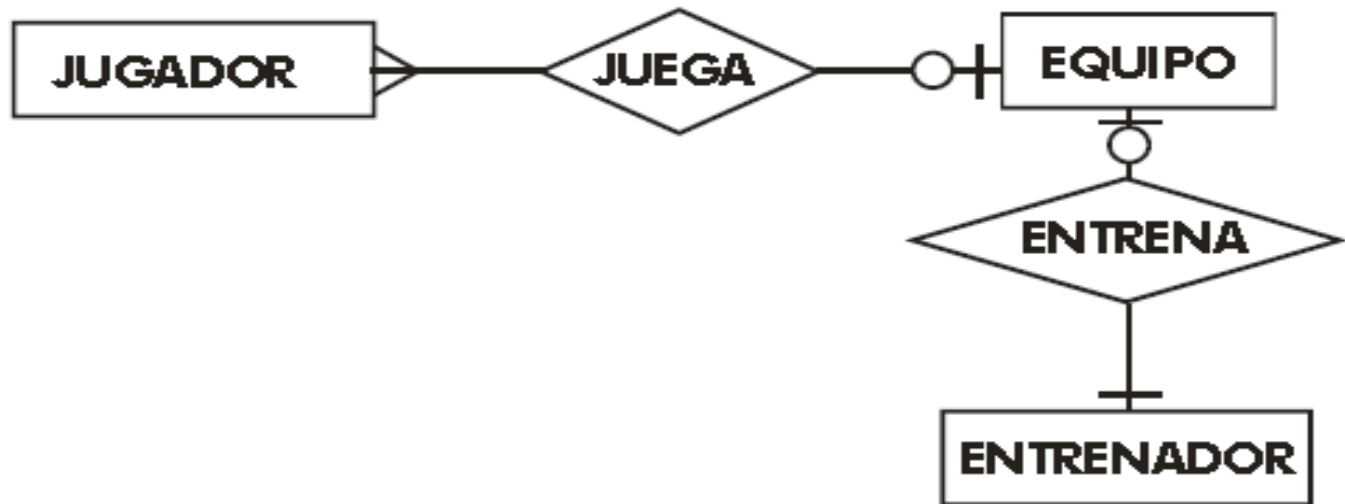




# El Modelo Entidad-Relación: E-R

## Elementos del modelo E-R: **RELACIONES**

Ejemplo:



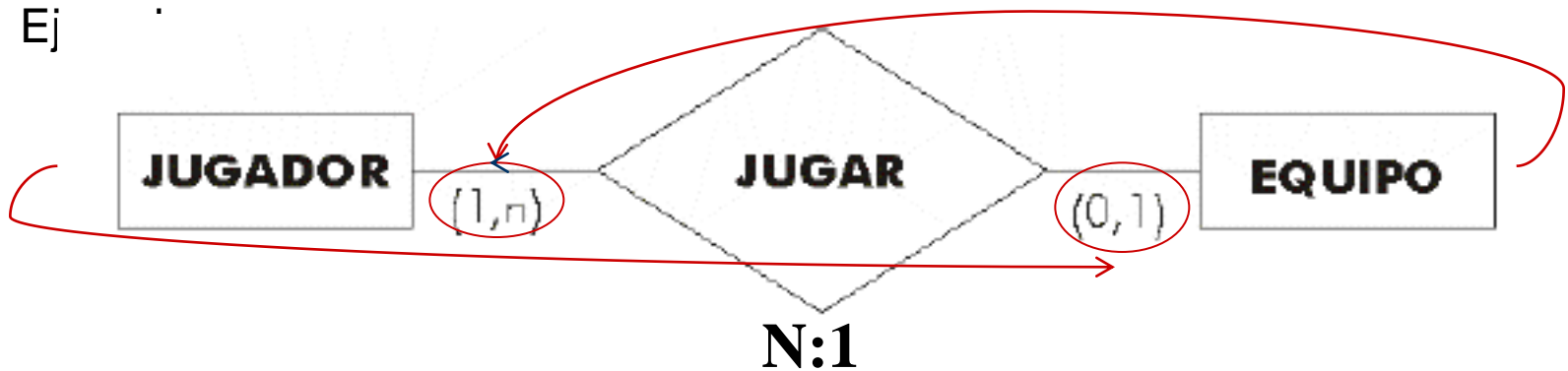
Cada equipo cuenta con varios jugadores. Un jugador juega como mucho en un equipo y podría no jugar en ninguno.

Cada entrenador podría entrenar a un equipo o ninguno, el cual tiene solo y solo un entrenador

# El Modelo Entidad-Relación: E-R

## Elementos del modelo E-R: **RELACIONES**

Ej



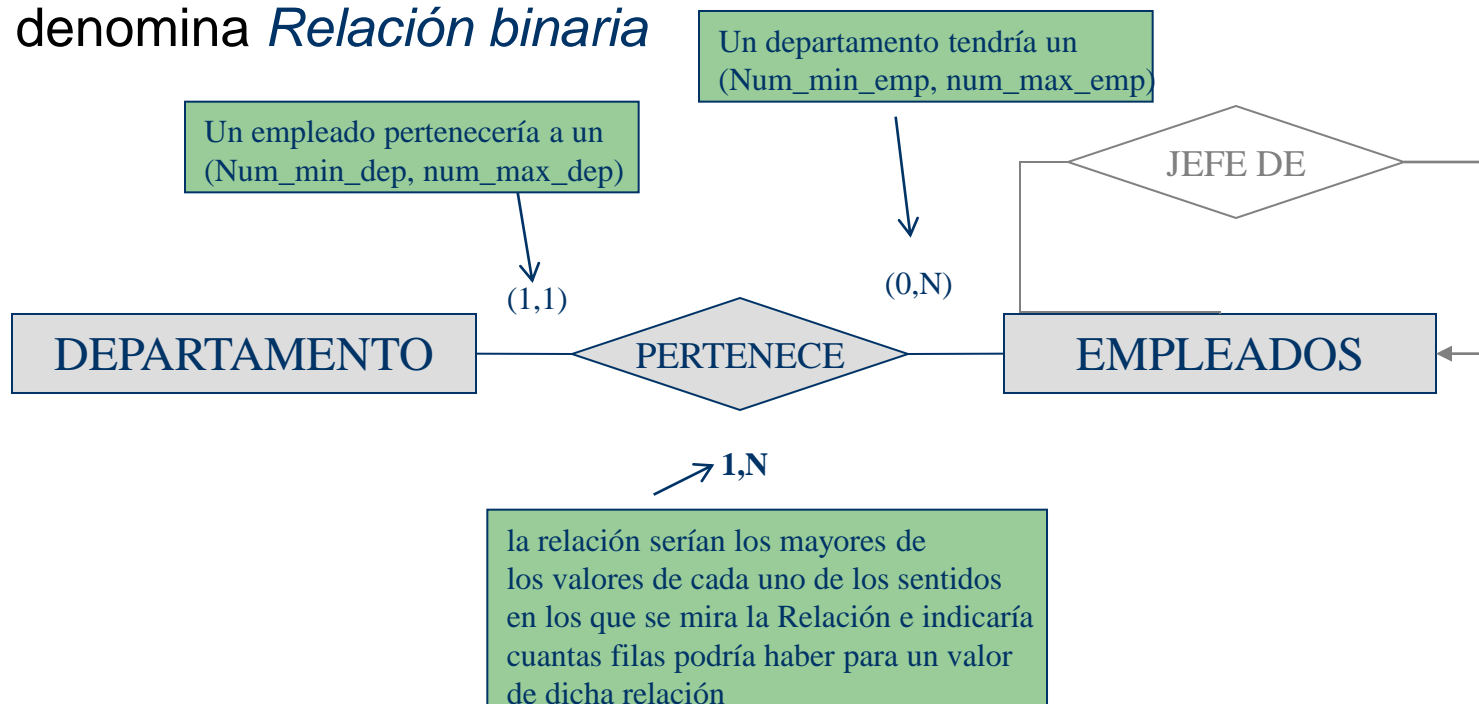
**CARDINALIDAD:** Un jugador tiene una cardinalidad mínima de 0 y máxima de 1 en un equipo, y un equipo tiene una cardinalidad mínima de 1 jugador (en realidad es más alta) y máxima de N jugadores

**RELACIÓN:** es N:1 un equipo tiene varios jugadores y un jugador solo pertenece a 0 o 1 equipo (la mayor de cada una de las cardinalidades que se relacionan)

# El Modelo Entidad-Relación: E-R

## Relaciones de Grado 2

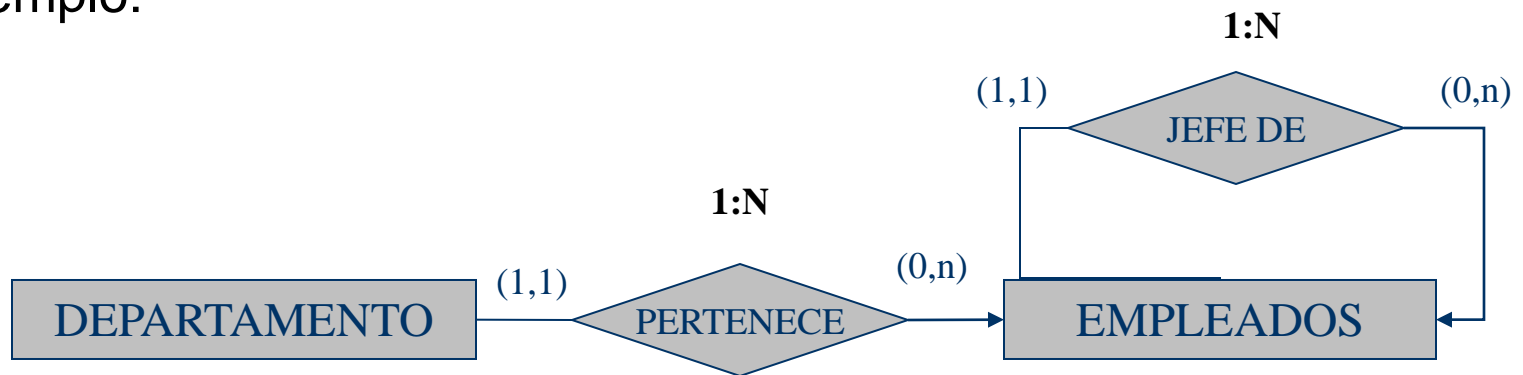
Se denomina *Relación binaria*



# El Modelo Entidad-Relación: E-R

## Elementos del modelo E-R: **RELACIONES**

Ejemplo:

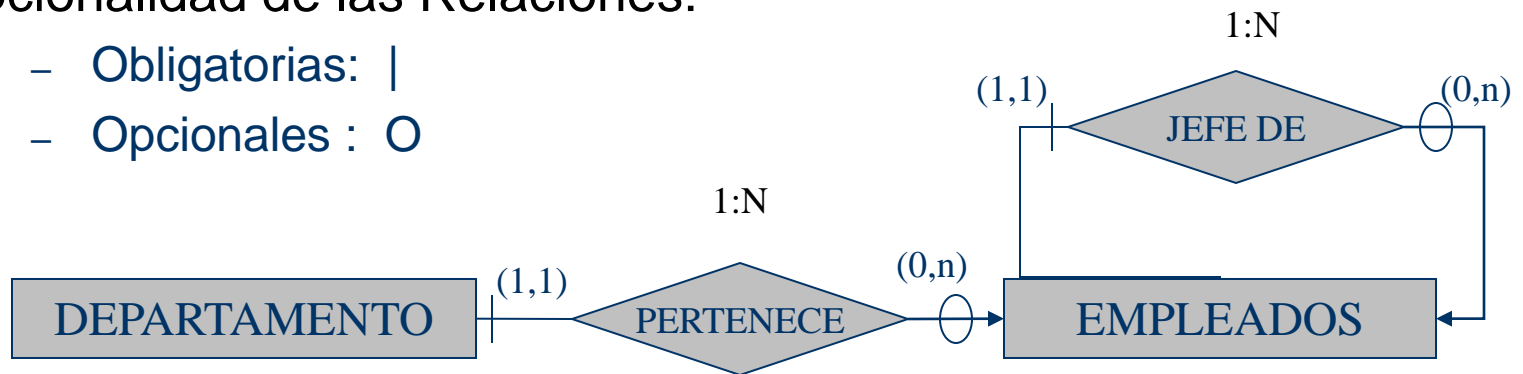


# El Modelo Entidad-Relación: E-R

## Elementos del modelo E-R: **RELACIONES**

Opcionalidad de las Relaciones:

- Obligatorias: |
- Opcionales : O



# El Modelo Entidad-Relación: E-R

## Elementos del modelo E-R: **ATRIBUTOS**

*Es cada una de las propiedades o características de una entidad o una relación*

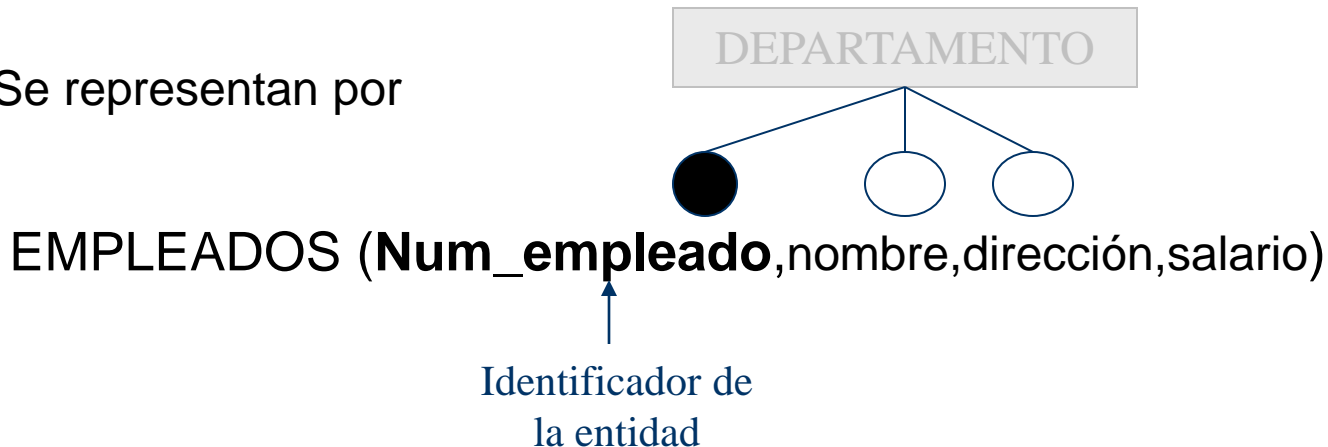
Entidad	Atributos
ALUMNO	ID_ALUM DNI NOMBRE DIRECCION TELEFONO
EMPLEADOS	NUM_EMPLEADO NOMBRE DIRECCIÓN SALARIO

# El Modelo Entidad-Relación: E-R

## Elementos del modelo E-R: **ATRIBUTOS**

Un atributo se identifica por su nombre, y el atributo que identifica a la entidad se denomina **ATRIBUTO IDENTIFICADOR PRINCIPAL**

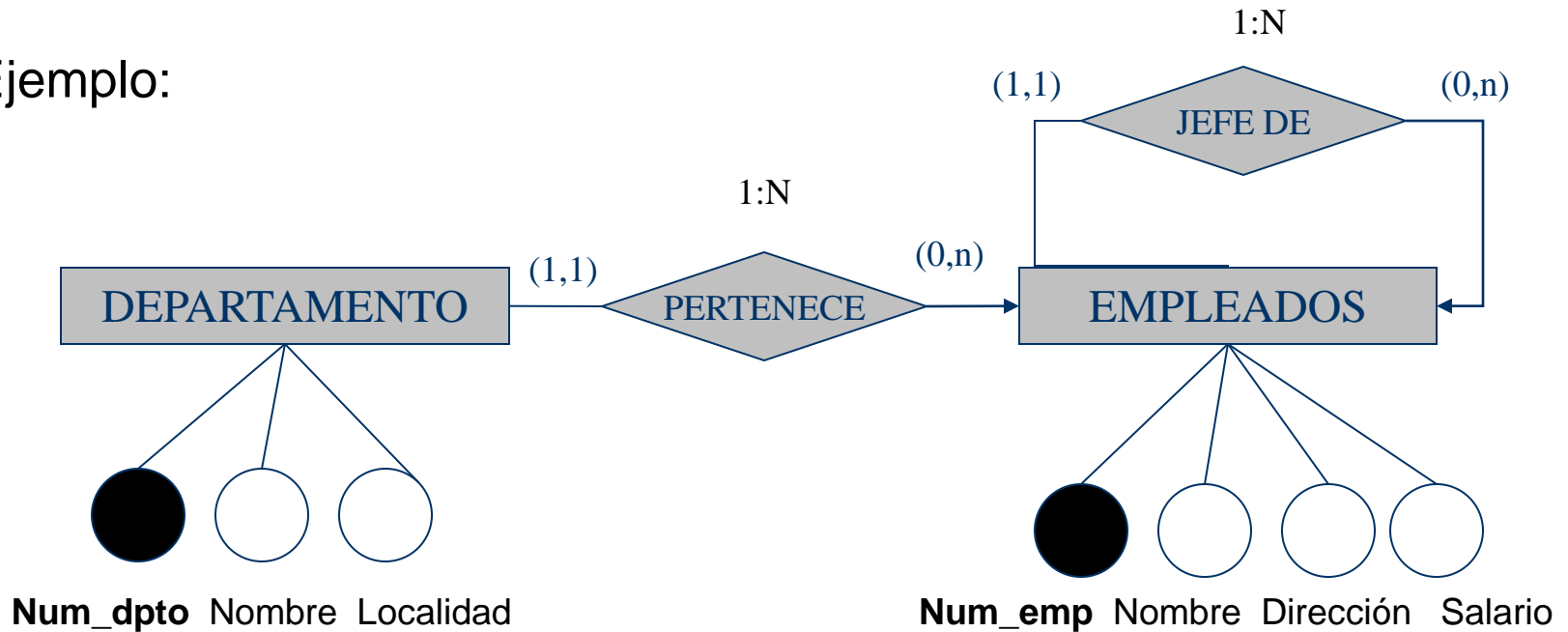
Se representan por



# El Modelo Entidad-Relación: E-R

## Elementos del modelo E-R: **ATRIBUTOS**

Ejemplo:



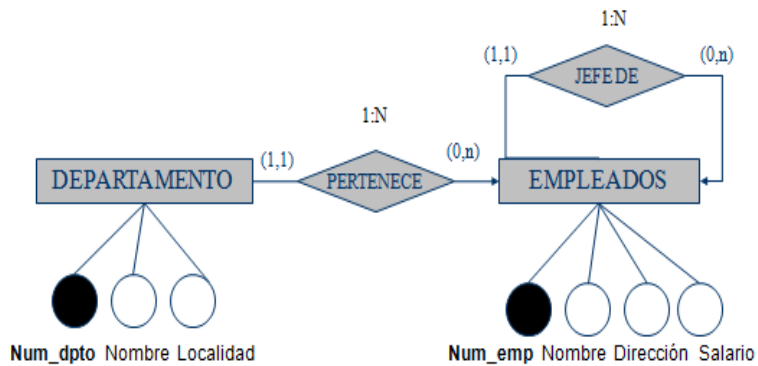


# Del E-R al Modelo Relacional

E-R



MODELO  
RELACIONAL



DEPARTAMENTO

NUM_DPTO	NOM_DPTO	PRESUPUESTO
D1	Marketing	60000
D2	Desarrollo	100000
D3	Investigación	30000

EMPLEADOS

NUM_EMP	APELLIDOS	NUM_DPTO	Jefe
E1	Fernandez	D1	
E2	Gonzalez	D2	E1
E3	Lopez	D2	E2
E4	Menendez	D3	E1

# Estructura del Modelo Relacional

El modelo relacional: Es una implementación en un SGBD real.

- Es un modelo simple y potente
- Se basa en las matemáticas relacionales
- El elemento principal es la relación
- Su sencillez ha facilitado la aparición de lenguajes de consultas e interfaces muy sencillos

# Estructura del Modelo Relacional

## Elementos del modelo Relacional

- Relación
- Dominios
- Claves
- Vistas
- Índices

# Estructura del Modelo Relacional

## Elementos del modelo relacional: **RELACION**

- ❑ Se representa mediante una **TABLA**
- ❑ Tabla: es una entidad o una asociación de entidades
  - ATRIBUTOS (columnas)
  - TUPLA (fila)
- ❑ No existen filas duplicadas y de una tabla se derivan los conceptos:
  - **cardinalidad**: número de filas de la tabla
  - **Grado**: número de columnas
  - **Valor**: valor de la intersección de una fila y una columna
  - **valor NULL**: ausencia de información

# Estructura del Modelo Relacional

## Elementos del modelo relacional: **RELACION**

Num_emp	Nombre	Salario	Num_dpto	Fecha_alta
1024	Armando Guerra Constante	1000	10	18/11/1990
1025	Dolores Fuertes	1200	15	29/1/1980

Cardinalidad: 2 (número de filas)

Grado:5 (número de atributos)

# Estructura del Modelo Relacional

## Elementos del modelo relacional: **RELACION**

- De la definición de una relación se definen las siguientes propiedades (características de una relación):
  - En una relación no existen tuplas/filas repetidas o duplicadas
  - Las tuplas de una relación no están ordenadas, no es significativo
  - Los atributos de una relación no tienen orden
  - Todos los valores de los atributos son atómicos, solo toman un valor

# Estructura del Modelo Relacional

## Elementos del modelo relacional: **DOMINIOS**

- ❑ Conjunto de valores que puede tomar un atributo
- ❑ Tipos de dominios
  - Generales: están entre un máximo y un mínimo  
salario: 9999
  - Restringidos: pertenecen a un conjunto de valores específicos  
sexo: H o M

# Estructura del Modelo Relacional

## Elementos del modelo relacional: **DOMINIOS**

- ❑ Tipos de datos más comunes de un dominio:
  - **Texto**: almacena cadenas de caracteres (números con los que **no** vamos a realizar operaciones matemáticas, letras o símbolos).
  - **Numérico**: almacena números con los que vamos a realizar operaciones matemáticas.
  - **Fecha/hora**: almacena fechas y horas.
  - **Sí/No**: almacena datos que solo tienen dos posibilidades (verdadero/falso).
  - **Autonumérico**: valor numérico secuencial que el SGBD incrementa de modo automático al añadir un registro (fila).
  - **Memo**: almacena texto largo (mayor que un tipo texto).
  - **Moneda**: se puede considerar un subtipo de Numérico ya que almacena números, pero con una característica especial, y es que los valores representan cantidades de dinero.
  - **Objeto OLE**: almacena gráficos, imágenes o textos creados por otras aplicaciones



# Estructura del Modelo Relacional

## Elementos del modelo relacional: **CLAVES**

- ❑ Atributo o conjunto de atributos que identifica unívocamente cada una de las ocurrencias de la identidad. NO ADMITEN REDUNDANCIA
- ❑ Tipos de claves:
  - Candidata
  - Primaria o primary key
  - Alternativas
  - Ajena o extranjera o foreign key

# Estructura del Modelo Relacional

## Elementos del modelo relacional: **CLAVES**

- Tipos de claves:
  - **Candidata**: aquellos atributos que pueden ser claves
  - **Primaria o primary key**: de las claves candidatas, el seleccionado como campo clave
  - **Alternativas**: de las claves candidatas, las que no han sido seleccionadas
  - **Ajena o extranjera o foreign key**: son atributos de una tabla que son campos claves o claves únicos de otra tabla o relación

# Estructura del Modelo Relacional

## Elementos del modelo relacional: **CLAVES**

### DEPARTAMENTO

NUM_DPTO	NOM_DPTO	PRESUPUESTO
D1	Marketing	60000
D2	Desarrollo	100000
D3	Investigación	30000

¿claves candidatas?

¿clave primaria?

¿claves alternativas?

### EMPLEADOS

NUM_EMP	APELLIDOS	NUM_DPTO	Jefe
E1	Fernandez	D1	
E2	Gonzalez	D2	E1
E3	Lopez	D2	E2
E4	Menendez	D3	E1

# Estructura del Modelo Relacional

## Elementos del modelo relacional: **CLAVES**

### DEPARTAMENTO

NUM_DPTO	NOM_DPTO	PRESUPUESTO
D1	Marketing	60000
D2	Desarrollo	100000
D3	Investigación	30000

### EMPLEADOS

NUM_EMP	APELLIDOS	NUM_DPTO	SALARIO
E1	Fernandez	D1	900
E2	Gonzalez	D2	1000
E3	Lopez	D2	1000
E4	Menendez	D3	800

Clave  
candidatas

# Estructura del Modelo Relacional

## Elementos del modelo relacional: **CLAVES**

NUM_DPTO	NOM_DPTO	PRESUPUESTO
D1	Marketing	60000
D2	Desarrollo	100000
D3	Investigación	30000

¿claves ajenas?

Clave alternativas

Clave primaria

NUM_EMP	APELLIDOS	NUM_DPTO	SALARIO
E1	Fernandez	D1	900
E2	Gonzalez	D2	1000
E3	Lopez	D2	1000
E4	Menendez	D3	800

# Estructura del Modelo Relacional

## Elementos del modelo relacional: **CLAVES**

NUM_DPTO	NOM_DPTO	PRESUPUESTO
D1	Marketing	60000
D2	Desarrollo	100000
D3	Investigación	30000

NUM_EMP	APELLIDOS	NUM_DPTO	SALARIO
E1	Fernandez	D1	900
E2	Gonzalez	D2	1000
E3	Lopez	D2	1000
E4	Menendez	D3	800

Clave primaria

Clave ajena

# Estructura del Modelo Relacional

## Elementos del modelo relacional: **VISTAS**

- ❑ Tabla virtual surgida a partir de filas de una o varias tablas que sirven de base.
- ❑ Corresponde al esquema externo o visión que necesita el usuario de la BD.



# Estructura del Modelo Relacional

## Restricciones del modelo

características de una relación  
*(sus 4 propiedades)*

+

regla de integridad de entidad  
*(ninguna clave puede ser nula)*

→ **restricc. inherentes del modelo**

Conjunto de dominios a verificar

+

regla de integridad referencial  
*(no debe tener valores de clave  
ajena sin concordancia)*

→ **restricciones de usuario**



# Estructura del Modelo Relacional

## Regla de Integridad de entidad

*Ningún componente de la clave primaria de una relación puede ser nulo (NULL)*

**IMPORTANTE:** NULO: no es un valor, indica que un atributo no tiene valor.

NULO (NULL) es distinto de espacio en blanco, de CERO, de VERDADERO y distinto también de FALSO

En todas las BD relacionales se utiliza un operador llamado ES NULO (IS NULL) que devuelve VERDADERO si el valor con el que se compara es NULO.

```
select * from ALUMNOS where NOTA1 is null
```

```
select * from ALUMNOS where NOTA1 is not null
```

# Estructura del Modelo Relacional

## Regla de Integridad referencial

La BD no debe tener valores de clave ajena sin concordancia, es decir que si existe una clave ajena, existe ese valor como clave primaria en otra tabla (**INCONSISTENCIA**)

NUM_DPTO	NOM_DPTO	PRESUPUESTO
D1	Marketing	60000
D2	Desarrollo	100000
D3	Investigación	30000

NUM_EMP	APELLIDOS	NUM_DPTO	SALARIO
E1	Fernandez	D1	900
E2	Gonzalez	D2	1000
E3	Lopez	D2	1000
E4	Menendez	D3	800

Clave primaria

Clave ajena

# Estructura del Modelo Relacional

## Regla para las claves ajenas

Para garantizar que las claves ajenas cumplen la INTEGRIDAD REFERENCIAL, se les debe de especificar que sucede cuando se actúa con la correspondiente clave primaria a la que referencian:

- **RESTRICT**: no se permite borrar la clave primaria
- **CASCADE**: borra la tupla (fila) de la clave ajena
- **SET NULL**: pone el valor de la clave ajena a NULL
- **SET DEFAULT**: pone el valor de la clave ajena a un valor

# Estructura del Modelo Relacional

## Elementos del modelo relacional: **INDICES**

- ❑ Un **índice** es una estructura de datos que permite acceder a diferentes filas de una misma tabla a través de un campo o campos . Esto permite un acceso mucho más rápido a los datos.
- ❑ son independientes, lógica y físicamente de los datos, es por eso que pueden ser creados y eliminados en cualquier momento, sin afectar a las tablas ni a otros índices.
- ❑ Son actualizados de forma automática, al modificar una tabla, lo que puede implicar que si existen muchos índices una operación implica actualizar tabla e índices, lo que puede suponer una ralentización.
- ❑ Se utilizan cuando se hacen consultas frecuentes en una rango de filas o en una fila de una tabla

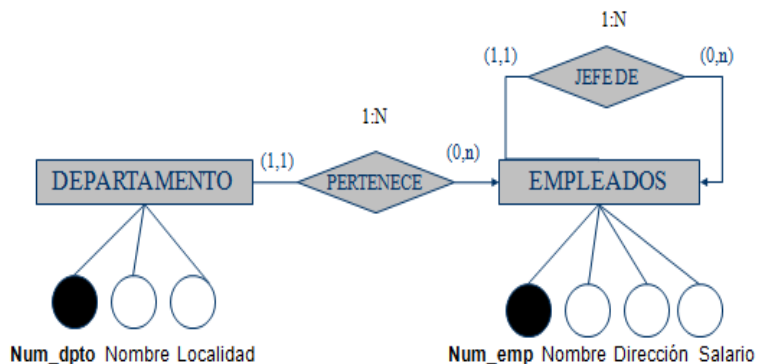
# Estructura del Modelo Relacional

## Paso del modelo E/R al modelo RELACIONAL

**E-R**



**MODELO  
RELACIONAL**



**DEPARTAMENTO**

NUM_DPTO	NOM_DPTO	PRESUPUESTO
D1	Marketing	60000
D2	Desarrollo	100000
D3	Investigación	30000

**EMPLEADOS**

NUM_EMP	APELLIDOS	NUM_DPTO	Jefe
E1	Fernandez	D1	
E2	Gonzalez	D2	E1
E3	Lopez	D2	E2
E4	Menendez	D3	E1

# Estructura del Modelo Relacional

## Paso del modelo E/R al modelo RELACIONAL

Obtenido el modelo E-R, definimos el modelo lógico:

- ❑ Entidad → tabla
- ❑ Atributo → columna de la tabla
- ❑ El identificador único → clave primaria
- ❑ Una relación N:M → nueva tabla con clave primaria resultante de la concatenación de las entidades que asocia
- ❑ Una relación 1:N → se propaga la clave primaria de la entidad con cardinalidad 1 a la tabla de la entidad con cardinalidad N y desaparece la relación

# Estructura del Modelo Relacional

## Paso del modelo E/R al modelo RELACIONAL

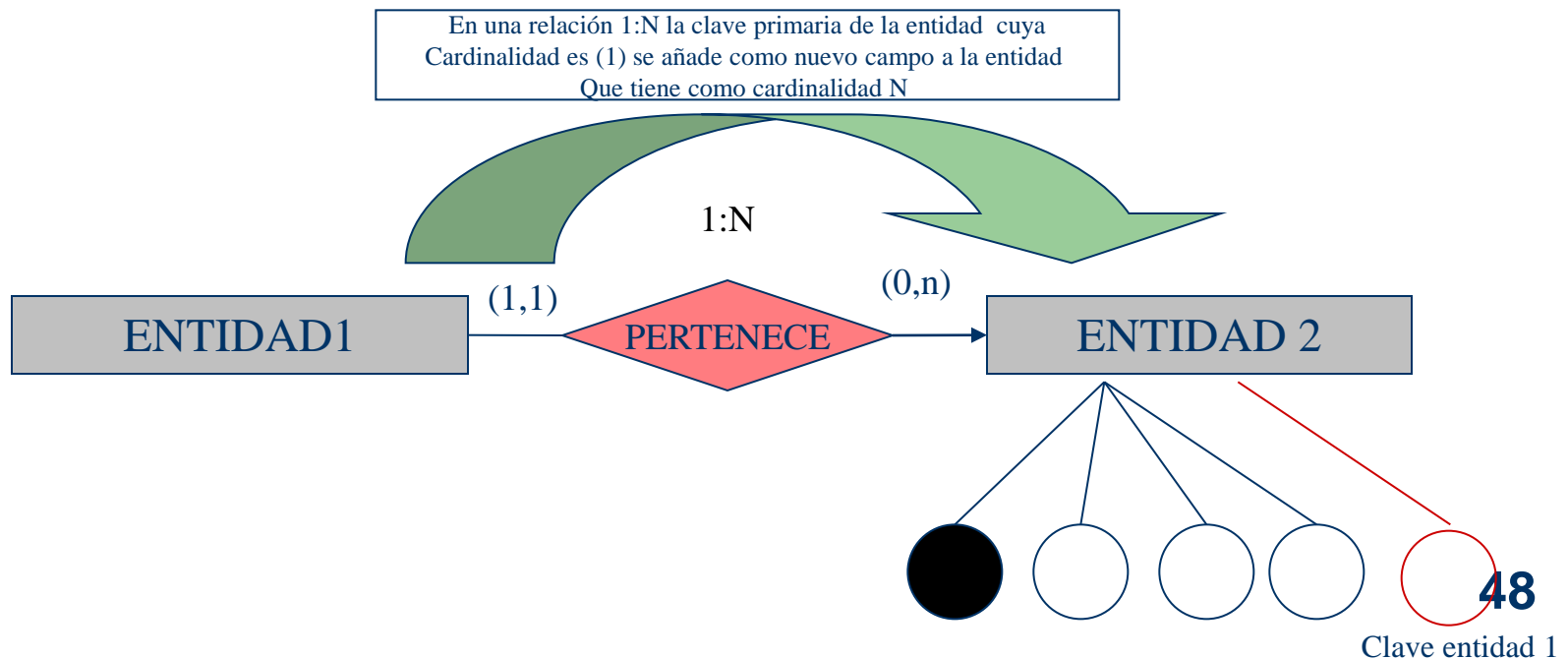
Obtenido el modelo E-R, definimos el modelo lógico:

- ❑ Una **relación 1:1**, se tiene en cuenta la cardinalidad de las entidades que participan:
  - Si una entidad tiene cardinalidad (0,1) y otra (1,1) propagar la clave de la entidad con cardinalidad (1,1) a la tabla resultante de la entidad de cardinalidad (0,1)
  - Si ambas tienen cardinalidad (1,1) propagar la clave de cualquiera de ellas a la otra entidad

# Estructura del Modelo Relacional

## Paso del modelo E/R al modelo RELACIONAL

### Relación 1:N



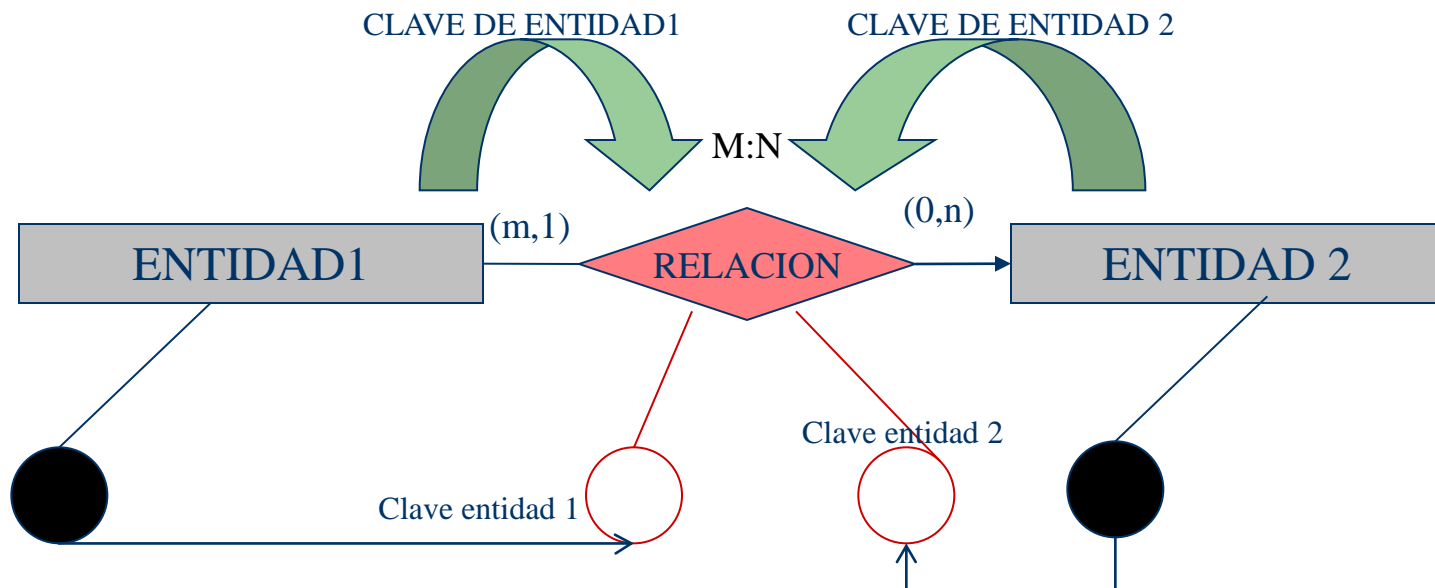


# Estructura del Modelo Relacional

## Paso del modelo E/R al modelo RELACIONAL

### Relación N:M

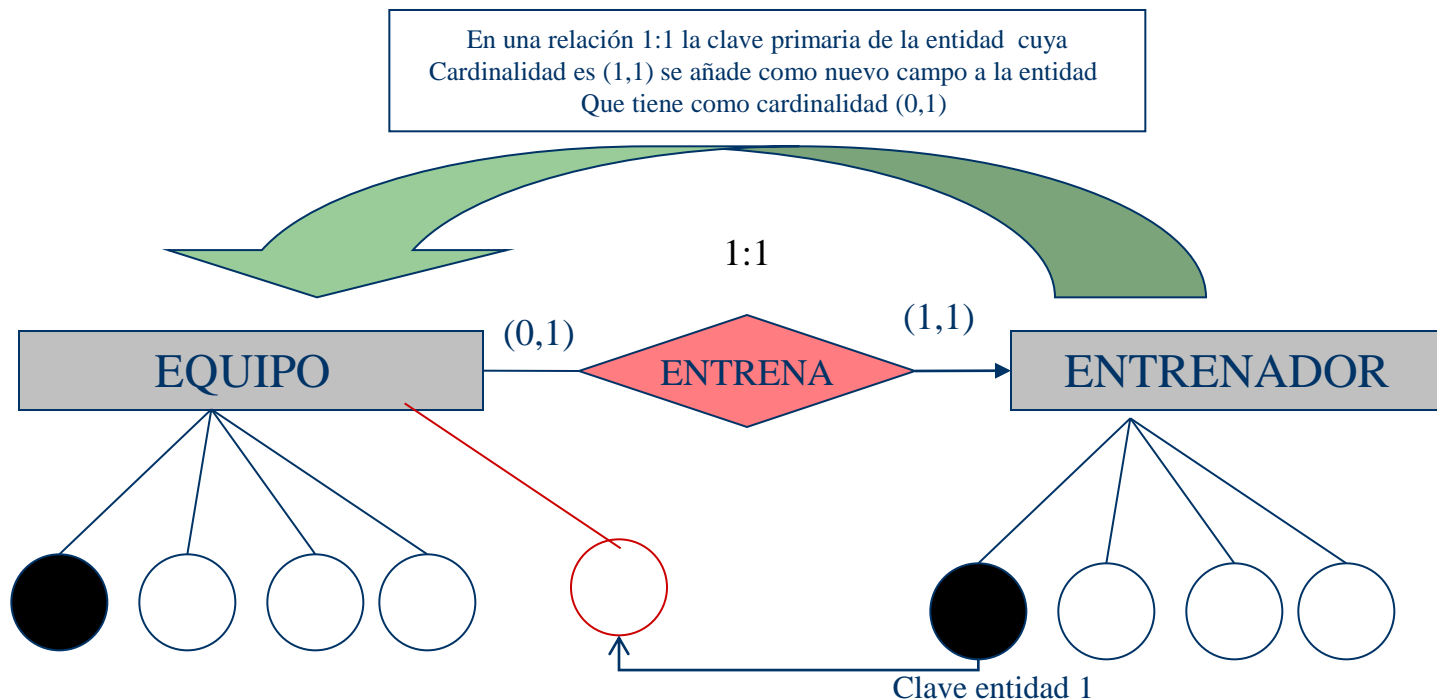
En una relación M:N la relación da lugar a una nueva tabla, que estará formada por las claves primarias de las dos entidades y los atributos propios de la relación



# Estructura del Modelo Relacional

## Paso del modelo E/R al modelo RELACIONAL

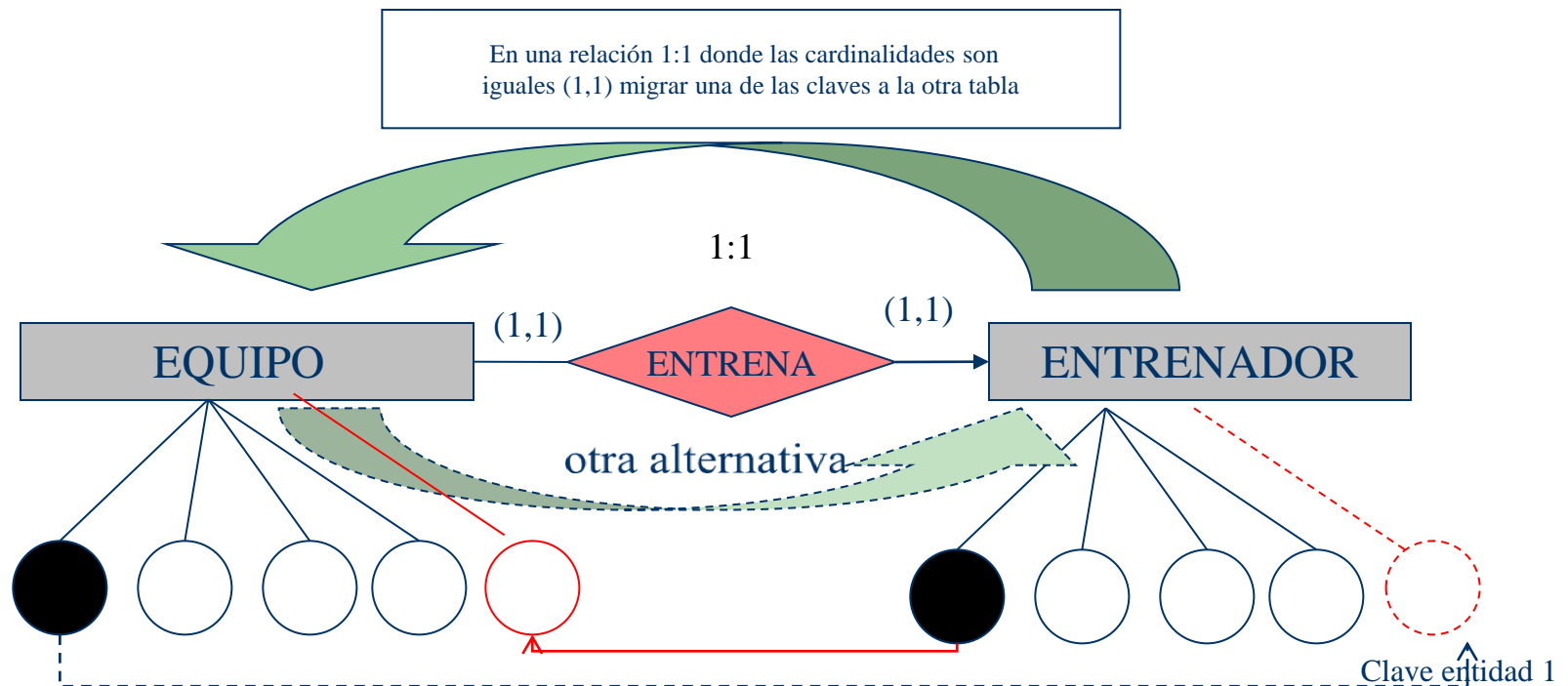
### Relación 1:1



# Estructura del Modelo Relacional

## Paso del modelo E/R al modelo RELACIONAL

### Relación 1:1



# Estructura del Modelo Relacional

## Paso del modelo E/R al modelo RELACIONAL

**Ejemplo:**

Entidades	Atributos
Departamento	Numero Departamento Nombre Departamento Localidad
Empleado	Numero Empleado Nombre Salario Comisión Fecha_alta

- Un departamento puede tener 0 o más empleados, pero un empleado pertenece a un solo departamento
- Un jefe puede tener 0 o más empleados a su cargo, pero un empleado solo puede tener un jefe y solo uno

# Estructura del Modelo Relacional

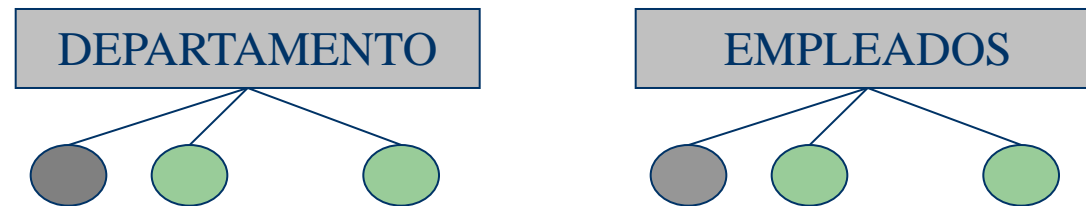
## Paso del modelo E/R al modelo RELACIONAL

### Se pide:

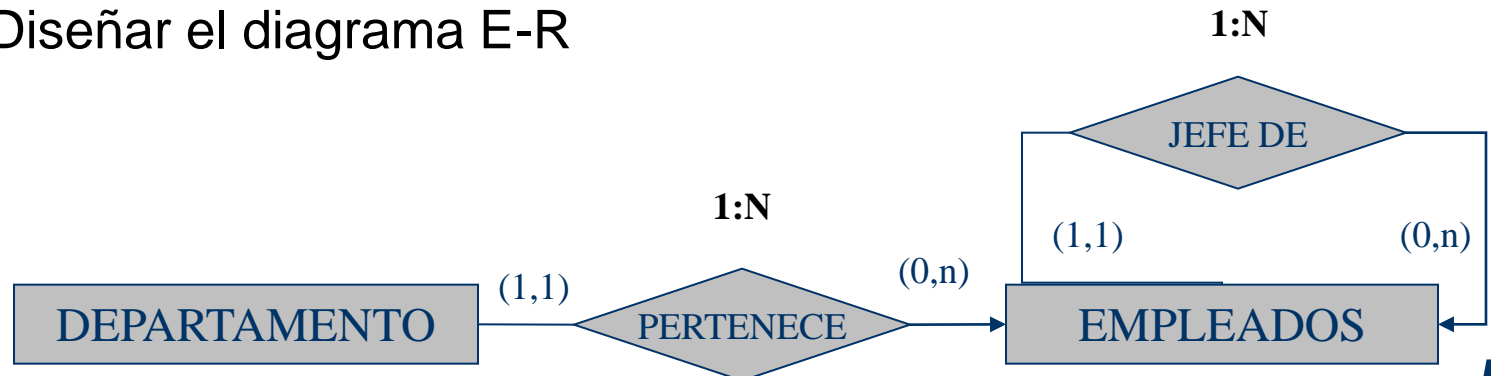
1. Asignar un nombre a las entidad y representar sus atributos
2. Diseñar el diagrama E-R
3. Representar de forma escrita las distintas entidades del modelo E-R con sus atributos, en su correspondiente equivalencia en el modelo relacional
4. Indicar las claves principales de cada una de las tablas
5. Transformar las relaciones del diagrama E-R a su correspondencia en el modelo relacional.
6. Indicar el modelo relacional final obtenido e indicar si existen claves ajenas, claves alternativas en las tablas

# Estructura del Modelo Relacional

1. Representar de forma escrita y gráfica las distintas tablas correspondientes a cada una de las entidades con todos sus atributos



2. Diseñar el diagrama E-R



# Estructura del Modelo Relacional

3. Representar de forma escrita las distintas entidades con sus atributos del modelo E-R en su correspondiente equivalencia en el modelo relacional

**EMPLEADOS**(Num\_Empleado,Nombre,Salario,Comisión,Fecha\_alta)  
**DEPARTAMENTO**(Num\_ dpto,Nom\_dpto,Localización)

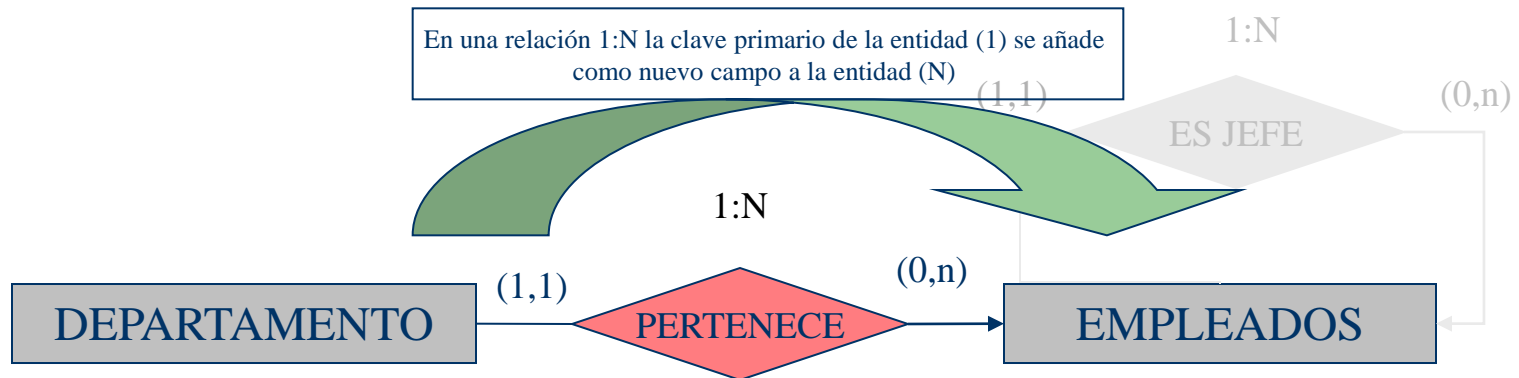
4. Indicar las claves principales de cada una de las tablas

**EMPLEADOS**(Num\_Empleado,Nombre,Salario,Comisión,Fecha\_alta)  
**DEPARTAMENTO**(Num\_ dpto,Nom\_dpto,Localización)

# Estructura del Modelo Relacional

5. Transformar las relaciones del diagrama E-R a su correspondencia en el modelo relacional

Relación: **PERTENECE**



**EMPLEADOS**(Num\_Empleado,Nombre,Salario,Comisión,Fecha\_alta,Num\_dpto)

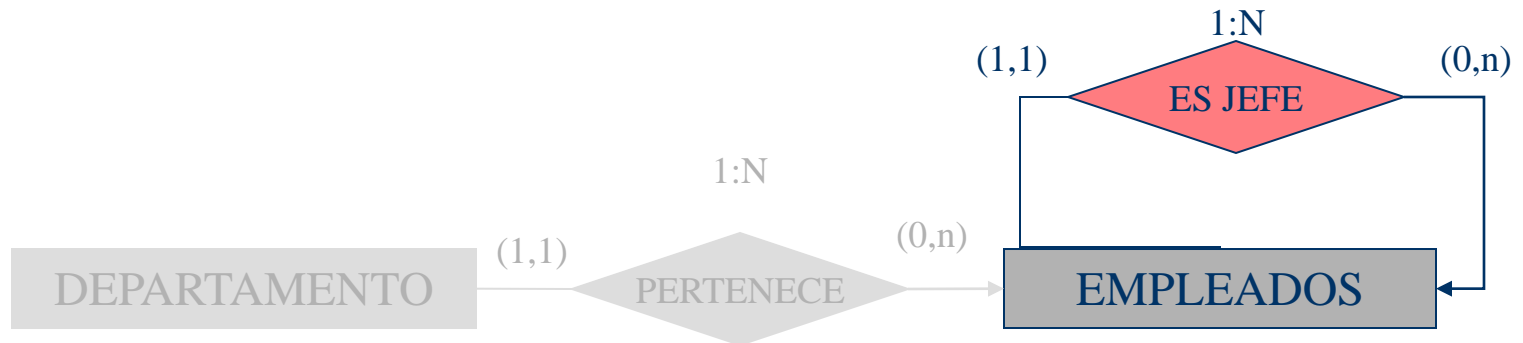


# Estructura del Modelo Relacional

## 5. Transformar las relaciones del diagrama E-R a su correspondencia en el modelo relacional

Relación: **ES JEFE**

En una relación 1:N la clave primario de la entidad (1) se añade como nuevo campo a la entidad (N) en este caso se añadiría un Nuevo campo a la entidad empleado llamado jefe que corresponderá con la clave primaria de un empleado



**EMPLEADOS**(Num\_Empleado, Nombre, Salario, Comisión, Fecha\_alta, Num\_dpto, jefe)

# Estructura del Modelo Relacional

6. Indicar el modelo relacional final obtenido e si existen claves ajenas, claves alternativas en las tablas

**EMPLEADOS**(Num\_Empleado,Nombre,Salario,Comisión,Fecha\_alta,**Num\_dpto**,jefe)

**DEPARTAMENTO**(Num\_dpto,Nom\_dpto,Localización)

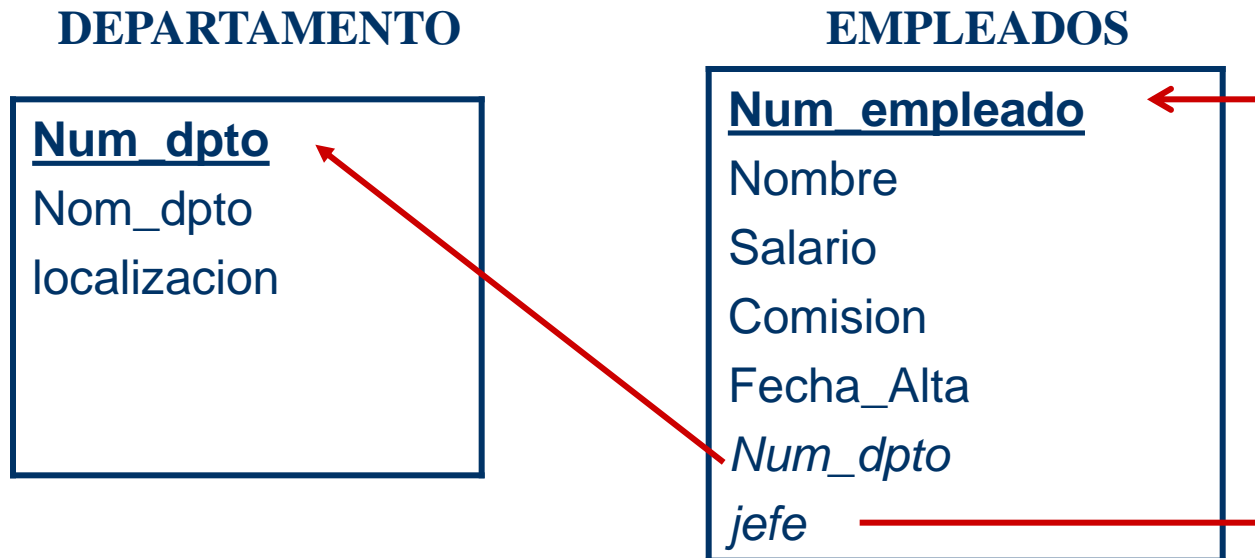
En EMPLEADOS aparecen una clave ajena (num\_dpto) y una clave alternativa (Nombre)

En DEPARTAMENTO aparece una clave alternativa (Nom\_dpto)

# Estructura del Modelo Relacional

**EMPLEADOS**(Num\_Empleado,Nombre,Salario,Comisión,Fecha\_alta,**Num\_dpto**,jefe)

**DEPARTAMENTO**(Num\_dpto,Nom\_dpto,Localización)



# Estructura del Modelo Relacional

## Ejercicio:

Se necesita obtener un listado con la siguiente información

- Código del cliente
- Nombre del cliente
- Dirección
- Código del material
- Descripción del material
- Cantidad de pedido
- Fecha de pedido
- Precio Unitario
- Número de Pedido

NOTA: recordad que una relación N:M da lugar a una nueva tabla que contendrá las claves primarias de las tabla que relaciona y los atributos propios de dicha relación

# Estructura del Modelo Relacional

Se pide:

1. Identificar las entidades y darles un nombre
2. Definir el modelo E-R sabiendo que un artículo puede ser solicitado por distintos clientes y un cliente puede solicitar distintos artículos.
3. Transformar el modelo E-R al modelo relacional
4. Indicar las claves principales de cada una de las tablas
5. Indicar el modelo relacional final obtenido

# USUARIOS. PRIVILEGIOS. ROLES

## 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.

Pero es muy importante diferenciar entre los distintos tipos de usuario:  
administrador, usuario final,....

# USUARIOS. PRIVILEGIOS. ROLES

## PRIVILEGIOS

- Capacidad o permiso que se le da a un usuario para realizar determinadas operaciones y/o acceder a determinados objetos
- Pueden ser de dos tipos:
  - *Privilegios sobre los objetos*: permiten acceder y realizar cambios en los datos de los usuarios  
Ejemplo: consulta de tablas de otro usuario, insertar, modificar
  - *Privilegios del sistema*: dan derecho a ejecutar un tipo de comando SQL o a realizar alguna acción sobre objetos  
Ejemplo. Crear tablespaces, crear vistas,...

# USUARIOS. PRIVILEGIOS. ROLES

## ROLES

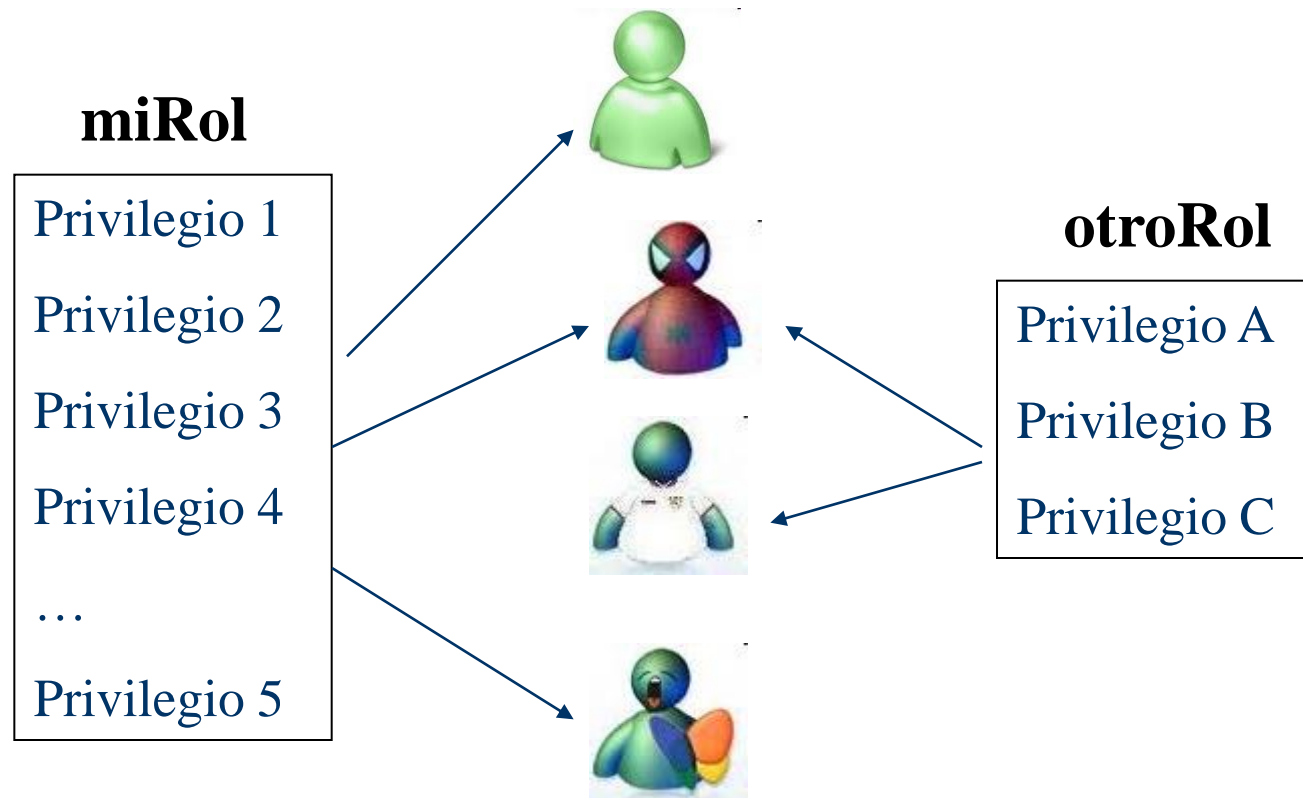
- Conjunto de privilegios que recibe un nombre común para facilitar la tarea de asignación a los usuarios o a otros roles
- Los privilegios que se engloban bajo un rol pueden ser tanto de sistema y como de objeto.

Ej: para ver los privilegios de sistemas asociados a los roles asignados a un usuario

```
// Select * from role_sys_privs;
```



# USUARIOS. PRIVILEGIOS. ROLES



# SQL

## SQL

- **SQL**: Structured Query Language. Es un lenguaje de consulta (Lenguaje de 4ª generación)
- Permite, además, la **manipulación y la gestión** de una BD
- Lenguaje para todo tipo de usuarios:
  - **Usuarios finales**
  - **Programadores**
  - **Administradores**
- Lenguaje **Portable** a diferentes entornos
- Otros lenguajes lo utilizan embebido para acceder a BD (JAVA, .NET,.....)

# SQL

## SQL

- El usuario que emplea SQL **especifica qué**, no dónde ni cómo.
- Permite construir **consultas** que **devuelvan una fila, un grupo de filas, datos de una tabla entera** o un **campo** de una tabla.
- SQL no solo es un lenguaje de consulta, permite también **actualizaciones, definición de la estructura de datos y control** (conexiones seguras)
- Aunque SQL está estandarizado, siempre es recomendable revisar la documentación del SGBD con el que estemos trabajando para conocer su sintaxis concreta, ya que algún comando, tipo de dato, etc., puede no seguir el estándar.

# ELEMENTOS DEL LENGUAJE. NORMAS DE ESCRITURA

El lenguaje SQL está compuesto por:

- Comandos
- Cláusulas
- Operadores
- Funciones
- Literales

Todos estos elementos se combinan en las instrucciones y se utilizan para crear, actualizar y manipular bases de datos.

# ELEMENTOS DEL LENGUAJE

El lenguaje SQL está compuesto por:

- **COMANDOS:** Van a ser las instrucciones que se pueden crear en SQL. Se pueden distinguir en tres grupos que veremos con más detenimiento a lo largo de las siguientes unidades:
  - **De definición de datos (DDL, Data Definition Language)**, que permiten crear y definir nuevas bases de datos, tablas, campos, etc.
  - **De manipulación de datos (DML, Data Manipulation Language)**, que permiten generar consultas para ordenar, filtrar y extraer datos de la base de datos.
  - **De control y seguridad de datos (DCL, Data Control Language)**, que administran los derechos y restricciones de los usuarios.

# ELEMENTOS DEL LENGUAJE

El lenguaje SQL está compuesto por:

- **CLÁUSULAS:** Llamadas también condiciones o criterios, son palabras especiales que permiten modificar el funcionamiento de un comando.

*Ej. FROM, WHERE, GROUP BY, HAVING, BY*

- **OPERADORES:** Permiten crear expresiones complejas. Pueden ser aritméticos (+, -, \*, /, ...) o lógicos (<, >, >=, <=, < >, And, Or, not, between, like, in ).

# ELEMENTOS DEL LENGUAJE

El lenguaje SQL está compuesto por:

- **FUNCIONES:** Para conseguir valores complejos. Por ejemplo, la función promedio para obtener la media de un salario.

*Ej. AVG, COUNT, SUM, MAX, MIN*

- **LITERALES:** Les podemos llamar también constantes y serán valores concretos, como por ejemplo un número, una fecha, un conjunto de caracteres, etc.

# NORMAS DE ESCRITURA

Normas a tener en cuenta:

- Todas las instrucciones **terminan con un signo de punto y coma.**

*select \* from empleados ;*

- No se distingue entre mayúsculas y minúsculas.
- Cualquier comando puede ser partido con saltos de línea o espacios para facilitar su lectura y comprensión.
- Los comentarios comienzan por /\* y terminan con \*/ (excepto en algunos SGBD).

*/\* ..... comentario .....\*/*



# Tipos de sentencias SQL

## TIPOS DE 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 BD. Usada por programadores y administradores. (CREATE, DROP, ALTER,...)

- **DCL** (Data Control Language)

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

# Elementos del Lenguaje SQL

## ESTRUCTURA DE UNA SENTENCIA SQL

**SENTENCIA** esta formada por:

- un nombre de sentencia que describe la operación a realizar
- Por una o varias cláusulas (obligatorias u opcionales)

Una sentencia no es *case sensitive*: se puede usar mayúsculas o minúsculas.

*Ej. Modelo de sentencia delete*

*DELETE FROM tabla [WHERE condición];*

*Cuando una sentencia forma parte de un bloque, función o procedimiento debe finalizar en ;*

# Elementos del Lenguaje SQL

## REGLAS PARA NOMBRAR OBJETOS EN ORACLE

- Los nombres deben tener como máximo 30 caracteres
- No pueden ser palabras reservadas del lenguaje
- No pueden contener comillas
- El nombre debe comenzar por una letra
- Usan los siguientes caracteres: a..z, A..Z, 0..9, \_, \$, # (no se aconseja utilizar estos dos últimos)
- Es indiferente el uso de mayúsculas y minúsculas

# Elementos del Lenguaje SQL

## NOMBRES DE COLUMNAS

Para referenciar una columna se puede indicar de las siguientes formas:

- **nombreColumna**
- **nombreTabla.nombreColumna**
- **esquemaUsuario.nombreTabla.nombreColumna**

### NOTA:

*Siempre que la sentencia afecte a más de una tabla es aconsejable especificar la tabla a la que pertenece cada columna. Si estamos trabajando en un esquema y la tabla pertenece al esquema no es necesario indicar el esquemaUsuario.*

# Elementos del Lenguaje SQL

## COMENTARIOS

- Sentencia SQL:

*/\* comentarios \*/*

- Script SQL:

*REM comentario*

## LITERALES

*'cadena de caracteres'*

# Tipos de datos SQL

## TIPOS DE DATOS

### ➤ CHAR(longitud)

- Almacena cadena de caracteres de longitud fija entre 1 y 255
- Las columnas tienen una longitud fija.
- Si se introduce **una cadena de menor longitud** que la definida se **rellenará con blancos a la derecha** hasta que quede completa.
- Si se introduce una cadena de **mayor longitud** que la fijada Oracle devolverá **un error**.

# Tipos de datos SQL

## TIPOS DE DATOS

### ➤ VARCHAR2 (longitud):

- Almacena cadena de caracteres de longitud variable (max. 2000 caracteres)
- Las columnas de este tipo tienen una **longitud variable**.
- Si se introduce **una cadena de menor longitud** que la que esta definida se almacenará con esa longitud.
- Si se introduce **una cadena de mayor longitud a la fijada**, Oracle devolverá **error**.

# Tipos de datos SQL

## TIPOS DE DATOS

### ➤ NUMBER (precisión, escala):

- almacena datos numéricos, tanto enteros como decimales, con o sin signo
- “**precisión**”: el número total de dígitos que va a tener el dato que se define: el rango de 1 a 38.
- “**escala**”: número de dígitos a la derecha del punto decimal: rango de –84 a 127. Una escala negativa indica redondear tantos dígitos a la izquierda del punto decimal
- Si no se indica escala, el valor de esta es 0
- Si no se indica precisión ni escala, se almacenará el valor tal y como se introduzca



# Tipos de datos SQL

## TIPOS DE DATOS: Ejemplo

Valor	Formato	Almacenamiento
7456123.89	NUMBER	7456123.89
7456123.89	NUMBER(9)	**7456124
7456123.89	NUMBER(9,2)	7456123.89
7456123.89	NUMBER(9,1)	*7456123.9
7456123.89	NUMBER(6)	ERROR
7456123.89	NUMBER(15,1)	*****7456123.9
7456123.89	NUMBER(7,-2)	7456100
7456123.89	NUMBER(-7,2)	ERROR, precisión tiene que estar entre 1 y 38

# Tipos de datos SQL

## TIPOS DE DATOS

### ➤ LONG:

- Almacena cadenas de caracteres de longitud variable (hasta 2Gb). Este tipo de datos está obsoleto (en desuso), en su lugar se utilizan los datos de tipo LOB (CLOB, NCLOB). Oracle recomienda que se convierta el tipo de datos LONG a alguno LOB si aún se está utilizando.
- Restricciones:
  - Sólo se puede definir una columna LONG por tabla
  - No puede aparecer en restricciones de integridad (CONSTRAINTS).
  - No sirve para indexar ni utilizar como argumento en funciones
  - Una función almacenada no puede devolver un valor LONG.
  - No es posible su uso en cláusulas WHERE, GROUP BY, ORDER BY, CONNECT BY, DISTINCT BY, ni con operaciones de UNIÓN, INTERSERC Y MINUS
  - No se puede referenciar como subconsulta en la creación de tablas, ni en inserción de filas.
  - Las variables o argumentos de bloques PL/SQL no se pueden declarar como tipo LONG.

# Tipos de datos SQL

## TIPOS DE DATOS

### ➤ DATE:

- Almacena información de fecha y hora. Para cada tipo DATE almacena la siguiente información:  
*Siglo / años / mes / días / horas / minutos / segundos.*
- Por defecto el valor para el formato de la fecha es una cadena de caracteres del tipo **dd/mm/yy**. Se puede modificar con la orden:

**ALTER\_SESSION** variando el parámetro **NLS\_DATE\_FORMAT**

# Tipos de datos SQL

## TIPOS DE DATOS

### ➤ OTROS TIPOS DE DATOS FECHA

- **DATETIME**: combinación de fecha y hora: yy-MM-dd hh-mm-ss con rango desde 1/1/1001 hasta 31/12/9999
- **TIMESTAMP**: igual que el anterior pero desde 1/1/1970 31/12/2037
- **TIMESTAMP WITH TIME ZONE**
- **TIME STAMP WITH LOCAL TIME ZONE**
- **INTERVAL YEAR TO MONTH**
- **INTERVAL DAY TO SECOND**

# Tipos de datos SQL

## TIPOS DE DATOS

### ➤ RAW(varchar2) y LONG RAW(long):

- Almacena datos binarios
- RAW es igual a varchar2 pero maneja cadena de bytes en vez de caracteres Maximo 255 bytes
- LONG RAW es igual a LONG y se emplea para almacenar gráficos, sonidos,...

Ya está en desuso y se sustituye por los tipo LOB

# Tipos de datos SQL

## TIPOS DE DATOS

LOB (BLOB, CLOB, NCLOB, BFILE)	Permiten almacenar y manipular bloques grandes de datos no estructurados (tales como texto, imágenes, videos, sonidos, etc) en formato binario o del carácter	Admiten hasta 8 terabytes (8000 GB). Una tabla puede contener varias columnas de tipo LOB. Soportan acceso aleatorio. Las tablas con columnas de tipo LOB no pueden ser replicadas.
BLOB	Permite almacenar datos binarios no estructurados	Admiten hasta 8 terabytes
CLOB	Almacena datos de tipo carácter	Admiten hasta 8 terabytes
NLOB	Almacena datos de tipo carácter.	Admiten hasta 8 terabytes. Guarda los datos según el juego de caracteres Unicode nacional

# Tipos de datos SQL

## TIPOS DE DATOS

### ➤ ROWID:

- Identifica de forma única la dirección de cada fila. Esta columna utiliza **una representación binaria de la localización física** de la fila.

# Tipos de datos SQL

## TIPOS DE DATOS

### Ejemplos:

ASIGNATURA	CHAR(15)
APELLIDO	VARCHAR2(10)
SUMA	NUMBER
SALARIO_MED	NUMBER(9,2)
TEXTO	LONG
FECHA_ALTA	DATE



# DDL

## 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 BD

## CREATE DATABASE

- Son creadas por el administrador de la BD, y permite crear el contenedor que almacenará los distintos esquemas de los usuarios que van a utilizarla.
- El comando para crear una BD

`CREATE DATABASE NombredemiBasedeDatos;`

*Ej. `CREATE DATABASE BD_empresa`*

# DDL: Creación de tablas

## CREACION DE 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

## CREATE TABLE

Formato:

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

- **Colum1, Colum2,...**: Nombres de las columnas que forman la fila
- **Tipo dato**: indica el tipo de dato que va a contener
- **TABLESPACE espacio\_de\_tabla**: señala el TABLESPACE para almacenar la tabla, si no se indica lo hará en el tablespace por defecto
- **NOT NULL**: especifica si la columna puede contener valores nulos o no

# DDL: Creación de tablas

## CREATE TABLE

Ejemplo:

```
CREATE TABLE USUARIOS (  
    id_usuario NUMBER(4),  
    nombre VARCHAR(25),  
    apellidos VARCHAR(25)  
);
```

```
Create table ALUMNOS  
(  
    NUM_MATRICULA number(6) NOT NULL,  
    NOMBRE          varchar2(15) NOT NULL,  
    FECHA_NTO       date,  
    DIRECCION       varchar2(30),  
    LOCALIDAD       varchar2(15)  
) tablespace users;
```

# DDL: Creación de tablas

## CREATE TABLE

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

```
Select * from user_tables;
```

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

```
// objetos que son propiedad del usuario
```

```
Select * from USER_OBJECTS;
```

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

```
Select * from USER_CATALOG;
```

# DDL

## INTEGRIDAD DE DATOS

- **INTEGRIDAD:** Hace referencia al hecho de que 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.

Ejemplo VENTAS, ARTICULOS. Nunca se dará la situación de insertar una venta con un artículo inexistente.

# DDL

## RESTRICCIONES

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

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

## RESTRICCIONES EN CREATE TABLE

Tipos de restricción sobre una tabla:

- ❑ Claves primarias
- ❑ Claves ajenas
- ❑ Obligatoriedad
- ❑ Valores por defecto
- ❑ Verificación de condiciones

# DDL: Restricciones

## CONSTRAINT

Para definir las restricciones se 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

# DDL: Restricciones

## 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];
  
```

# DDL: Restricciones

## RESTRICCIONES EN CREATE TABLE

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

Ejemplo:

```
Create table EMPLEADO (  
    nombre          varchar2(25)      primary key,  
    edad            number             check (edad between 18 and 35),  
    cod_provincia   number(2)          references PROVINCIAS  
                                         on delete cascade  
);
```

# DDL: Restricciones

## RESTRICCIONES EN CREATE TABLE

- **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.
- Se puede hacer referencia a varias columnas en una única restricción (por ejemplo, declarando dos columnas como clave primaria o ajena).

# DDL: Restricciones

## 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];

```

# DDL: Restricciones

## RESTRICCIONES EN CREATE TABLE

### – Restricción de tabla:

```
CREATE TABLE EMPLEADO
(
  NOMBRE          VARCHAR(25),
  EDAD            NUMBER,
  COD_PROVINCIA   NUM(2),
  CONSTRAINT EMPLEADO_PK PRIMARY KEY (NOMBRE),
  CONSTRAINT EDAD_CK CHECK (EDAD BETWEEN 18 AND 35),
  CONSTRAINT EMPLEADO_FK FOREIGN KEY (COD_PROVINCIA)
                      REFERENCE PROVINCIAS ON DELETE CASCADE
);
```

# DDL: Restricciones

## RESTRICCIONES EN CREATE TABLE

### – Restricción de tabla:

```
CREATE TABLE EMPLEADO
(
  NOMBRE          VARCHAR(25),
  EDAD            NUMBER,
  COD_PROVINCIA   NUMBER(2),
  PRIMARY KEY     (NOMBRE),
  CHECK           (EDAD BETWEEN 18 AND 35),
  FOREIGN KEY     (COD_PROVINCIA) REFERENCES PROVINCIAS
                  ON DELETE CASCADE
);
```



# DDL: Restricciones

## 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

Ej.

```
CREATE TABLE USUARIOS (  
    Login VARCHAR2(25) CONSTRAINT usu_log_UK  UNIQUE  
);
```

```
CREATE TABLE USUARIOS ( Login VARCHAR2 (25) UNIQUE);
```

```
CREATE TABLE USUARIOS (  
    Login VARCHAR2 (25),  
    Correo VARCHAR2 (25),  
    CONSTRAINT Usuario_UK UNIQUE (Login, Correo)  
);
```

# DDL: Restricciones

## Restricción NOT NULL

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

Ej.

```
CREATE TABLE USUARIOS (  
    F_Nacimiento DATE CONSTRAINT usu_fcha_nto NOT NULL  
);
```

```
CREATE TABLE USUARIOS (  
    F_Nacimiento DATE NOT NULL  
);
```

# DDL: Restricciones

## 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

# DDL: Restricciones

## Clave primaria. Restricción PRIMARY KEY

Formato para definir PRIMARY KEY:

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

# DDL: Restricciones

## Clave primaria. Restricción PRIMARY KEY

```
Create table BLOQUEPISOS (
  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 viv_pk PRIMARY KEY
                (calle, numero, piso, puerta)
); /* restriccion de tabla*/
```

```
Create table BLOQUEPISOS (
  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)
); /* restriccion de tabla*/
```

# DDL: Restricciones

## Clave primaria. Restricción PRIMARY KEY

/\* restriccion de columna\*/

**Create table** ZONAS

(

**COD\_ZONA**

**number(3) PRIMARY KEY,**

NOMBREZONA

**varchar2(15) not null,**

MASDATOS

**varchar2(60)**

);

----

/\* restriccion de columna otra forma de ponerla\*/

**Create table** ZONAS

(

**COD\_ZONA**

**number(3) CONSTRAINT ZONAS\_PK PRIMARY KEY,**

NOMBREZONA

**varchar2(15) not null,**

MASDATOS

**varchar2(60)**

);

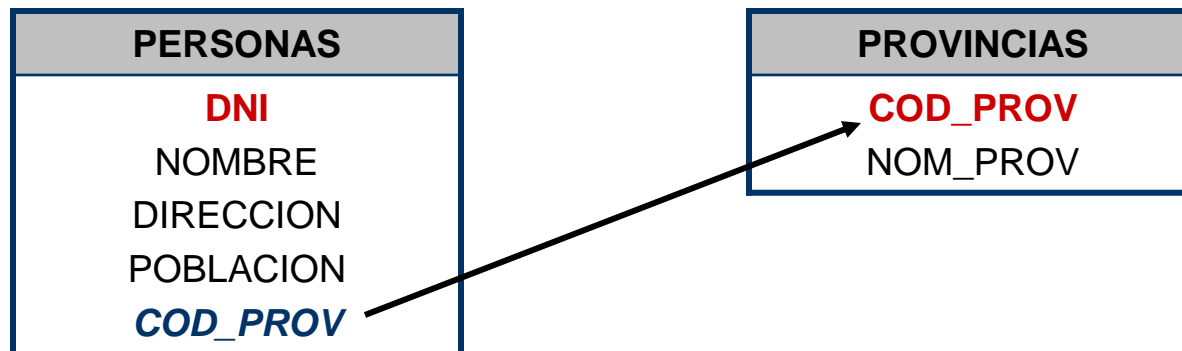
# DDL: Restricciones

## 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)

## DDL: Restricciones

### Claves ajenas. Restricción FOREIGN KEY





# DDL: Restricciones

## 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];
```

# DDL: Restricciones

## 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];
```

# DDL: Restricciones

## Claves ajenas. Restricción FOREIGN KEY

En nuestro ejemplo:

*PERSONAS* (*dni*, *nombre*, *direccion*, *población*, *cod\_prov*)

*PROVINCIAS* (*cod\_prov*, *nom\_prov*)

Create table PERSONAS (

dni                                    number(8) PRIMARY KEY,

nombre                                varchar2(15),

direccion                            varchar2(25),

poblacion                            varchar2(20),

cod\_prov                              number(2) NOT NULL,

**FOREIGN KEY**                    (*cod\_prov*) **REFERENCES**                    *PROVINCIAS*

);

También podríamos poner:

cod\_prov                    number(2) NOT NULL **REFERENCES**                    *PROVINCIAS*

# DDL: Restricciones

## Claves ajenas. Restricción FOREIGN KEY

En nuestro ejemplo:

PERSONAS (dni, nombre, direccion, población, **cod\_prov**)

PROVINCIAS (cod\_prov, nom\_prov)

```
Create table PROVINCIAS (  
cod_prov          number(2) PRIMARY KEY,  
nom_prov          varchar2(15)  
);
```

Será necesario crear primero la tabla PROVINCIAS y después la tabla PERSONAS (que referencia a PROVINCIAS) para que no se produzca error en ORACLE. Para borrar será primero la tabla que hace la referencia (PERSONAS) y después la referenciada (PROVINCIAS).

# DDL: Restricciones

## Claves ajenas. Restricción FOREIGN KEY. Opción ON DELETE CASCADE

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

Create table PERSONAS (

    dni                    number(8) PRIMARY KEY,

    nombre                varchar2(15),

    direccion             varchar2(25),

    poblacion             varchar2(20),

    cod\_prov              number(2) NOT NULL,

**FOREIGN KEY (cod\_prov) REFERENCES PROVINCIAS**  
    **ON DELETE CASCADE**

);

# DDL: Restricciones

## Claves ajenas. Restricción FOREIGN KEY. Opción ON DELETE SET NULL

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

Create table PERSONAS (

    dni                    number(8) PRIMARY KEY,

    nombre                varchar2(15),

    direccion             varchar2(25),

    poblacion             varchar2(20),

    cod\_prov              number(2) NOT NULL,

    FOREIGN KEY (cod\_prov) REFERENCES PROVINCIAS

    ON DELETE SET NULL

);

# DDL: Restricciones

## RESTRICCIONES EN CREATE TABLE

### – Claves ajenas. Restricción FOREIGN KEY

#### 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.

# DDL: Restricciones

## RESTRICCIONES EN CREATE TABLE

### – 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.



# DDL: Restricciones

## RESTRICCIONES EN CREATE TABLE

### – CONSTRAINT

Create table PROVINCIAS

```
(  
  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 línea 1:

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

# DDL: Restricciones

## RESTRICCIONES EN CREATE TABLE

### – CONSTRAINT

Create table PERSONAS (

<b>dni</b>	<b>number(8)</b>		
	<b>CONSTRAINT</b>	<b>PER_PK</b>	<b>PRIMARY KEY,</b>
nombre	varchar2(15),		
direccion	varchar2(25),		
poblacion	varchar2(20),		
cod_prov	number(2) NOT NULL,		
<b>CONSTRAINT</b>	<b>PER_FK</b>		
	<b>FOREIGN KEY (cod_prov)</b>	<b>REFERENCES</b>	<b>PROVINCIAS</b>

);

**PER\_PK**, es el nombre de la restricción de la clave primaria.

**PER\_FK**, restricción de la clave ajena

Cualquier error que se produzca referente a ella vendrá referenciado por:

**nom\_usuario.FER\_PK o FER\_FK** (ej. Asignar cod\_prov inexistente)

# DDL: Restricciones

## RESTRICCIONES DEFAULT

DEFAULT: permite asignar un valor por defecto

```
CREATE TABLE USUARIOS ( Pais VARCHAR2(20) DEFAULT ' España ' );
```

```
CREATE TABLE USUARIOS ( Fecha_ingreso DATE DEFAULT SYSDATE);
```

# DDL: Restricciones

## RESTRICCIONES VALIDACIÓN CHECK

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

```
CREATE TABLE USUARIOS (  
    Credito NUMBER(4) CHECK (Credito BETWEEN 0 AND 2000)  
);
```

```
CREATE TABLE EMPLEADOS(  
    .....  
    funcionario char(1),  
    constraint FUNCIONARIO_CK check (funcionario in ('S','N')),  
    ....);
```

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

# DDL: Restricciones

## RESTRICCIONES VALIDACIÓN CHECK

```
CREATE TABLE T_EJEMPLO(  
    ....
```

```
    campo tipo check (expresion),
```

*donde expresion puedes ser una comparación, igualdad, etc*

```
    campo tipo constraint nom_CK check ( campo in (valor1, valor2,..)),
```

*donde valor pueden ser un 'carácter' , 'cadena' o numero*

```
);
```

# DDL: Restricciones

## RESTRICCIONES VALIDACIÓN CHECK

### Ejemplos

edad number(4) check (edad >18 and edad <65)

sexo char(1) constraint SEXO\_CK check( sexo in ('H','M'))

dia\_semana varchar2(10) constraint dia\_sem\_ck check (dia\_semana in ('LUNES', 'MARTES', 'MIERCOLES', 'JUEVES', 'VIERNES'))

fcha\_onto date constraint fcha\_onto\_ck check (sysdate > fcha\_onto)

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)

# DDL: Borrado de tablas

## DROP TABLE

Permite suprimir una tabla de la base de datos.

- Un usuario solo puede borrar sus propias tablas
- Solo el administrador o algún usuario al que se le otorgara privilegios puede borrar las tablas de otro usuario
- Al borrar una tabla se suprimen los índices y privilegios asociados a ella.
- Las vistas y sinónimos asociados a una tabla dejan de funcionar **pero no desaparecen**, por lo que habría que eliminarlos.

# DDL: Borrado de tablas

## DROP TABLE

Formato:

```
DROP TABLE [esquema_usuario.]nombre_tabla  
[CASCADE CONSTRAINTS];
```

La opción **CASCADE CONSTRAINTS** permite suprimir todas las restricciones de integridad referencial que se refieran a claves de la tabla borrada.

Ejemplo: `DROP TABLE EJEMPLO;`



# DDL: Borrado de tablas

## DROP TABLE

Ejemplo: tabla PROVINCIAS y PERSONAS (con clave ajena a PROVINCIAS)

**DROP TABLE PROVINCIAS; → produce ERROR!!!**

Produciría un error debido a que existen tablas (en este caso PERSONAS) que tienen alguna columna que es clave ajena y que referencian a la tabla PROVINCIAS, por lo que si eliminamos la tabla PROVINCIAS, la columna PERSONAS.COD\_PROV no tendría sentido al referenciar a una tabla que no existe.

Para evitar esto sería necesario borrar la tabla PROVINCIAS, pero indicando que elimine también todas las referencias de otras tablas a PROVINCIAS

**DROP TABLE PROVINCIAS CASCADE CONSTRAINTS;**

# DDL: limpiar tablas

## TRUNCATE TABLE

Permite suprimir todas las filas de una tabla y liberar el espacio ocupado para otros usos sin que desaparezca la definición de la tabla de la base de datos.

Una instrucción TRUNCATE no permite ROLLBACK

Formato:

```
TRUNCATE TABLE [usuario.]nom_tabla [{DROP | REUSE} STORAGE];
```

# DDL: limpiar tablas

## TRUNCATE TABLE

Formato:

```
TRUNCATE TABLE [usuario.]nom_tabla [{DROP | REUSE} STORAGE];
```

**DROP STORAGE:** opción por defecto, libera el espacio de las filas borradas

**REUSE STORAGE:** borra las filas, pero mantiene el espacio reservado para nuevas filas.

No se puede truncar una tabla cuya clave primaria sea clave ajena de otra tabla y exista integridad referencial definida. Habrá que desactivar la restricción.

Ej. `TRUNCATE TABLE PROVINCIAS` (producirá error hasta que se desactive la restricción, en este caso suponiendo que otra tabla p.e. `PERSONAS` tengan clave ajena a `PROVINCIAS`)

DIFERENCIAS entre DELETE y TRUNCATE

# DDL: Modificación de tablas

## ALTER TABLE

Permite modificar la estructura de las tablas.

Las tablas se pueden modificar:

- **Cambiando la definición de una columna**
- **Añadiendo/borrando una columna**

Formato:

```
ALTER TABLE nombre_tabla
{
    [ADD      (columna tipo)]
    [DROP COLUMN (columna [, columna ]...)]
    [MODIFY (columna tipo [, columna tipo]...)]
    [RENAME COLUMN NombreAntiguo TO NombreNuevo]
    [ADD      CONSTRAINT restricción]
    [DROP     CONSTRAINT restricción]
    [DISABLED CONSTRAINT restricción]
    [ENABLE  CONSTRAINT restricción]
};
```

# DDL: Modificación de tablas

## ALTER TABLE

[ADD (columna tipo)]: Añade columnas a una tabla

- Si la columna no está definida como NOT NULL, se puede añadir en cualquier momento
- Si la columna está definida como NOT NULL, una opción:
  - se define primero la columna sin especificar NOT NULL
  - A continuación se le asigna un valor para cada fila
  - Se modifica la columna a NOT NULL

# DDL: Modificación de tablas

## ALTER TABLE

[MODIFY (columna tipo [, columna tipo]....)]: Modifica una o más columnas existentes. Habrá que tener en cuenta:

- Se puede aumentar la longitud de la columna en cualquier momento
- Al disminuir la longitud de una columna con datos no se puede dar menor tamaño que el máximo valor almacenado
- Si la columna es NULL en todas las filas de la tabla, se puede disminuir y modificar el tipo de dato
- La opción MODIFY ... NOT NULL sólo será posible cuando la tabla no contenga ninguna fila con valor nulo en la columna

# DDL: Modificación de tablas

## ALTER TABLE

[**DROP COLUMN** (columna [, columna ]...)]: Borra una columna de una tabla.

- Recordar que no se pueden borrar todas las columnas de una tabla
- Tampoco se pueden eliminar las claves primarias referenciadas por claves ajenas

# DDL: Modificación de tablas

## ALTER TABLE

*[ADD CONSTRAINT restricción]: Permite añadir una restricción*

*[DROP CONSTRAINT restricción]: Permite borrar una restricción*

*[DISABLE CONSTRAINT restricción]: Permite desactivar una restricción*

*[ENABLE CONSTRAINT restricción]: Permite activar una restricción*



# DDL: Modificación de tablas

## ALTER TABLE

### Ejemplos

```
ALTER TABLE EJEMPLO ADD (SEXO CHAR(1), IMPORTE NUMBER(4));
```

```
ALTER TABLE EJEMPLO ADD ( SEXO CHAR(1) NOT NULL, IMPORTE  
NUMBER(4));
```

La tabla debería estar vacía, sino lo esta la restricción NOT NULL produciría un error.

```
ALTER TABLE EJEMPLO MODIFY (SEXO CHAR(1) NOT NULL, NOMBRE  
VARCHAR(5));
```

Podría dar error si la columna nombre tiene valores y son mayores de 5

```
ALTER TABLE EJEMPLO DROP COLUMN (SEXO);
```

# DDL: Modificación de tablas

## ALTER TABLE

### Ejemplos:

- Para añadir una restricción de valor único

```
ALTER TABLE EMPLE ADD CONSTRAINT apellido_uq UNIQUE (apellido);
```

- Para añadir la opción no nula al apellido

```
ALTER TABLE EMPLE ADD CONSTRAINT APELL_NONULO CHECK (APELLIDO NOT NULL);
```

```
ALTER TABLE EMPLE ADD CONSTRAINT EMPLE_CK (NUM_DEP IN(10,20,30,40,50));
```

- Para borrar una restricción

```
ALTER TABLE EMPLE DROP CONSTRAINT APELLIDO_UQ;
```

```
ALTER TABLE EMPLE DROP CONSTRAINT SYS_C0095;
```

# DDL: Renombrado de tablas

## RENAME

Permite cambiar el nombre a una tabla, vista o un sinónimo.

Oracle invalida todos los objetos que dependan del objeto renombrado, como las vistas, los sinónimos que hacen referencia a la tabla renombrada.

Formato:

*RENAME nombre\_old TO nombre\_new;*

Ejemplo:

create synonym ALUM from ALUMNOS;  
rename ALUMNOS to TALUMNOS;

//sinónimo para tabla ALUMNOS  
// al renombrar la tabla, el sinónimo  
ALUM deja de funcionar

# DDL: Indices

## INDICES

- Permiten acelerar el tiempo de respuesta en las consultas.
- Es un objeto de la base de datos que se asocia a una tabla y al que se asocia una o varias columnas de la tabla

Formato para crear:

```
CREATE INDEX nombre_indice  
ON nombre_tabla (columna [,columna]...)  
[STORAGE clausula_almacenamiento]  
[TABLESPACE nombre_tablespace]  
[otras cláusulas];
```

Formato para borrar:

```
DROP INDEX NombreIndice;
```

# DCL

## DCL (Data Control Language)

Permite gestionar:

- el acceso (gestión de usuarios)
- La confidencialidad- (GRANT, REVOKE,...)
- y las transacciones a la BD (COMMIT, ROLLBACK,...)

# DCL: Usuarios

## CREATE USER

Se debe de tener privilegios para la creación de usuarios (CREATE USER) o ser DBA de la base de datos

Formato:

```
CREATE USER nombre_usuario  
IDENTIFIED BY clave_acceso  
[DEFAULT          TABLESPACE          espacio_tabla]  
[TEMPORARY        TABLESPACE          espacio_tabla]  
[QUOTA {entero {K | M} | UNLIMITED} ON  espacio_tabla]  
[PROFILE          perfil];
```

Si no se indican las cláusulas opcionales, coge las opciones por defecto

# DCL: Usuarios

## MODIFICACION DE USUARIOS

Se debe tener privilegios para modificar usuarios (ALTER USER) o ser DBA de la base de datos

Formato:

```
ALTER USER nombre_usuario  
IDENTIFIED BY clave_acceso  
[DEFAULT TABLESPACE espacio_tabla]  
[TEMPORARY TABLESPACE espacio_tabla]  
[QUOTA {entero {K | M} | unlimited} ON espacio_tabla]  
[PROFILE perfil];
```

# DCL: Usuarios

## BORRADO DE USUARIOS

Formato:

**DROP USER** nombre\_usuario [CASCADE];

CASCADE: suprime todos los objetos asociados al usuario antes de borrar el usuario



# DCL: Usuarios

## VISTAS DEL DICCIONARIO PARA USUARIOS

- **ALL\_USERS**: Permite ver todos los usuarios creados
- **DBA\_USERS**: para comprobar los tablespaces asignados a cada usuario (solo usuario administrador)
- **USER\_USERS**: para ver la información del usuario actual

# DCL: Privilegios

## PRIVILEGIOS

- Capacidad de un usuario de realizar determinadas operaciones y/o acceder a determinados objetos
- Una vez creado un usuario es necesario darle privilegios para que pueda hacer algo:
  - privilegios específicos de una acción
  - privilegios asociados a roles predefinidos en ORACLE
  - Privilegios asociados a roles definidos por el DBA

# DCL: Privilegios

## PRIVILEGIOS

Los privilegios asociados a los roles predefinidos en ORACLE 10g:

ROLES	PRIVILEGIOS
CONNECT	CREATE SESSION
RESOURCE	CREATE TRIGGER CREATE SEQUENCE CREATE TYPE CREATE PROCEDURE CREATE CLUSTER CREATE OPERATOR CREATE INDEX TYPE CREATE TABLE

# DCL: Privilegios

## PRIVILEGIOS

Los privilegios asociados a los roles predefinidos en ORACLE:

ROLES	PRIVILEGIOS
DBA	TODOS LOS PRIVILEGIOS DEL SISTEMA
EXP_FULL_DATABASE	SELECT ANY TABLE BACKUP ANY TABLE INSERT UPDATE DELETE sobre tablas SYS.INCVID, SYS.INCFILL y SYS.INCEXP
IMP_FULL_DATABASE	BECOME USER

# DCL: Privilegios

## PRIVILEGIOS

### Tipos de privilegios a definir sobre la BD

- *Privilegios sobre los objetos*: permiten acceder y realizar cambios en los datos de los usuarios  
Ejemplo: consulta de tablas de otro usuario, insertar o borrar datos en una tabla
- *Privilegios del sistema*: dan derecho a ejecutar un tipo de comando SQL o a realizar alguna acción sobre objetos  
Ejemplo. Crear tablespaces, crear vistas, crear tablas, consultar tablas, borrar tablas ...

# DCL: Privilegios

## PRIVILEGIOS sobre objetos

PRIVILEGIOS SOBRE OBJETOS	TABLA	VISTA	SECUENCIA	PROCEDURE
ALTER	X		X	
DELETE	X	X		
EXECUTE				X
CREATE INDEX	X			
INSERT	X	X		
REFERENCES	X			
SELECT	X	X	X	
UPDATE	X	X		

# DCL: Privilegios

## GRANT: ASIGNAR PRIVILEGIOS SOBRE OBJETOS

```
GRANT {priv_objeto [,priv_obeto] .... | ALL [PRIVILEGES]}  
      [(columna [,columna]....)]  
ON [usuario.]objeto  
TO {usuario | rol | PUBLIC} [, {usuario | rol | PUBLIC} ...]  
[WITH GRANT OPTION];
```

**WITH GRANT OPTION** → permite que el receptor del privilegio o rol lo asigne a otros usuarios o roles

Ejemplo:

```
GRANT select, insert ON tabla1 TO U3;  
GRANT all ON tabla1 TO U3;  
GRANT all ON tabla1 TO PUBLIC;
```

# DCL: Privilegios

## GRANT: ASIGNAR PRIVILEGIOS DEL SISTEMA

Permiten ejecutar un tipo de comando SQL, o a realizar alguna acción sobre objetos de un tipo especificado.

### **Sesiones**

**CREATE SESSION database:** Conectarse a una BD

### **Sinónimos**

**CREATE SYNONYM** Crear sinónimos en el esquema

**CREATE ANY SYNONYM** Crear sinónimos privados en cualquier esquema.

**CREATE PUBLIC SYNONYM** Crear sinónimos públicos

**DROP ANY SYNONYM** Borrar sinónimos privados en cualquier esquema.

**DROP PUBLIC SYNONYM** Borrar sinónimos públicos.



# DCL: Privilegios

## GRANT: ASIGNAR PRIVILEGIOS DEL SISTEMA

### *Tablas*

<b>CREATE ANY TABLE</b>	Crear tablas en cualquier esquema
<b>ALTER ANY TABLE</b>	Modificar cualquier tabla o vista en el esquema
<b>BACKUP ANY TABLE</b>	Permite exportar objetos del esquema de otros usuarios
<b>DELETE ANY TABLE</b>	Borrar filas de tablas o vistas en cualquier esquema
<b>DROP ANY TABLE</b>	Borrar o truncar tablas en cualquier esquema
<b>INSERT ANY TABLE</b>	Insertar filas en tablas y vistas de cualquier esquema
<b>LOCK ANY TABLE</b>	Bloquear tablas y vistas de cualquier esquema
<b>UPDATE ANY TABLE</b>	Actualizar filas en tablas y vistas de cualquier esquema
<b>SELECT ANY TABLE</b>	Consultar tablas y vistas en cualquier esquema.

# DCL: Privilegios

## GRANT: ASIGNAR PRIVILEGIOS DEL SISTEMA

### *Usuarios*

**CREATE USER** Crear usuarios.

**ALTER USER** Modificar cualquier usuario.

**DROP USER** Borrar usuarios.

### *Vistas*

**CREATE VIEW** Crear vistas en el esquema

**CREATE ANY VIEW** Crear vistas en cualquier esquema

**DROP ANY VIEW** Borrar vistas en cualquier esquema

### *Administración:*

**DBA** privilegio de DBA

# DCL: Privilegios

## GRANT: ASIGNAR PRIVILEGIOS DEL SISTEMA

### *Perfiles*

**CREATE PROFILE** Crear perfiles

**ALTER PROFILE** Modificar perfiles

**DROP PROFILE** Borrar perfiles

### *Roles*

**CREATE ROLE** Crear roles

**ALTER ANY ROLE** Modificar cualquier role de la BD

**DROP ANY ROLE** Borrar roles.

**GRANT ANY ROLE** Asignar cualquier role en la BD.

# DCL: Privilegios

## GRANT: ASIGNAR PRIVILEGIOS DEL SISTEMA

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

### Ejemplo:

```
Grant CREATE SESSION to usr1;           // privilegios de inicio de sesión  
Grant CONNECT to usr1;                   // privilegios descritos en el rol CONNECT  
Grant DBA to usr1, juan;                 // privilegios de DBA  
Grant CREATE VIEW to milagros;           // privilegios para crear vistas  
Grant SELECT ANY TABLE to PUBLIC;
```

# DCL: Privilegios

## REVOKE: RETIRADA DE PRIVILEGIOS

- *Formato para retirar privilegios de objetos a los usuario o roles:*  
REVOKE { priv\_objeto [, priv\_objeto].... | ALL [PRIVILEGES]}  
ON [usuario.]objeto  
FROM {usuario | rol | PUBLIC} [{usuario | rol | PUBLIC} ].....;
- *Formato para retirar privilegios de sistema o roles a usuarios o para retirar privilegios a roles:*  
REVOKE { priv\_sistema | rol } [{priv\_sistema | rol } ] ....  
FROM {usuario | rol | PUBLIC} [{usuario | rol | PUBLIC} ].....;

# DCL: Privilegios

## REVOKE: RETIRADA DE PRIVILEGIOS

Ejemplo:

```
REVOKE SELECT, UPDATE ON tabla 1 FROM francisco;
```

```
REVOKE ALL ON tabla1 FROM francisco, juan;
```

```
REVOKE DROP USER FROM milagros;
```

```
REVOKE SELECT ANY TABLE FROM PUBLIC;
```

```
REVOKE DBA FROM juan, pedro;
```

# DCL: Privilegios

## VISTAS CON INFORMACION DE PRIVILEGIOS

Para conocer los privilegios concedidos o recibidos por los usuarios sobre los objetos o a nivel de sistema, se pueden consultar las siguientes vistas del diccionario de datos:

- **SESSION\_PRIVS**: privilegios del usuario activo
- **USER\_SYS\_PRIVS**: privilegios de sistema asignados al usuario
- **DBA\_SYS\_PRIVS**: privilegios del sistema asignado a los usuarios y a los roles
- **USER\_TAB\_PRIVS**: concesiones sobre objetos que son propiedad del usuario, concedidos o recibidos por éste.
- **USER\_TAB\_PRIVS\_MADE**: concesiones sobre objetos que son propiedad del usuario (asignadas).
- **USER\_TAB\_PRIVS\_RECD**: concesiones sobre objetos que recibe el usuario.

# DCL: Privilegios

## VISTAS CON INFORMACION DE PRIVILEGIOS

- **USER\_TAB\_GRANTS**: concesiones en objetos para los que el usuario es el propietario, el que concedió el privilegio o al que se le concedió el privilegio.
- **USER\_TAB\_GRANTS\_MADE**: todas las concesiones hechas en objeto que son propiedad del usuario.
- **USER\_TAB\_GRANTS\_RECD**: concesiones en objetos en las que el usuario es aquél al que se ha concedido el privilegio (concesiones recibidas)
- **ALL\_TAB\_GRANTS, ALL\_TAB\_GRANTS\_MADE, ALL\_TAB\_GRANTS\_RECD**: son iguales que las anteriores vistas, con la diferencia de que aparecen las concesiones de todos los usuarios.
- **USER\_COL\_GRANTS**: concesiones en columnas para las que el usuario es el propietario, el que concedió el privilegio o al que se le concedió el privilegio.
- **USER\_COL\_GRANTS\_MADE, USER\_COL\_GRANTS\_RECD**: iguales que las anteriores vistas pero para columnas.
- **ALL\_COL\_GRANTS, ALL\_COL\_GRANTS\_MADE, ALL\_COL\_GRANTS\_RECD**: son iguales que las anteriores vistas, pero aparecen concesiones de todos los usuarios de la base de datos.
- **USER\_COL\_PRIVS**: concesiones sobre columnas en las que el usuario es el propietario, asigna el privilegio o lo recibe.
- **USER\_COL\_PRIVS\_MADE**: todas las concesiones sobre columnas de objetos que son propiedad del usuario.
- **USER\_COL\_PRIVS\_RECD**: concesiones sobre columnas recibidas por el usuario.



# RESUMEN

# ELEMENTO SQL

## Comandos DDL. Lenguaje de Definición de Datos

Comando	Descripción
CREATE	Crea nuevas tablas, campos e índices,
DROP	Elimina tablas, índices.
ALTER	Modifica objetos del sistema tablas, ...

# ELEMENTO SQL

## Comandos DML. Lenguaje de Manipulación de Datos

Comando	Descripción
SELECT	Consultar filas que satisfagan un criterio determinado
INSERT	Para Insertar datos en una tabla en una operación
UPDATE	Modificar valores de campos y filas específicos
DELETE	Eliminar filas de una tabla

# ELEMENTO SQL

## Comandos DCL. Lenguaje de Control de Datos

Comando	Descripción
GRANT	Permite dar permisos/privilegios a uno o varios usuarios o roles para realizar tareas determinadas
REVOKE	Permite eliminar permisos/privilegios previamente concedidos con GRANT

# ELEMENTO SQL

## Cláusulas

Cláusulas	Descripción
<b>FROM</b>	Especifica la tabla de la que se van a seleccionar filas
<b>WHERE</b>	Especifica las condiciones que deben de reunir las filas que se van a seleccionar
<b>GROUP BY</b>	Separa las filas seleccionadas en grupos específicos
<b>HAVING</b>	Expresa la condición que debe de satisfacer cada grupo
<b>ORDER BY</b>	Ordena las filas seleccionadas de acuerdo a un orden específico

# ELEMENTO SQL

## Operadores lógicos

Operador	Descripción
And	Evalúa dos condiciones y devuelve un valor de verdad solo si ambas son ciertas
Or	Evalúa dos condiciones y devuelve un valor de verdad si alguna de las dos es cierta o bien ambas lo son
not	Devuelve el valor contrario a la expresión

# ELEMENTO SQL

## Operadores Comparación

Operador	Descripción
<	Menor que
>	Mayor que
<>	Distinto de
<=	Menor o igual
>=	Mayor o igual
=	Igual
BETWEEN	Entre un intervalo de valores
LIKE	Para comparar “Como.....”
IN	Aparece en un conjunto de valores

# ELEMENTO SQL

## Funciones

Función	Descripción
<b>AVG</b>	Calcula el promedio de los valores de un campo determinado
<b>COUNT</b>	Devuelve el número de filas de la selección
<b>SUM</b>	Suma todos los valores de un campo determinado
<b>MAX</b>	Devuelve el máximo de un campo determinado
<b>MIN</b>	Devuelve el mínimo de un campo determinado



# ELEMENTO SQL

## Literales

Literales	Descripción
23/03/97	Literal fecha
María	Literal caracteres
5	Literal número