

# Introducción a las Bases de Datos

Departamento de Lenguajes y  
Ciencias de la Computación

---

Introducción a las Bases de Datos is licensed under a  
[Creative Commons Reconocimiento-NoComercial-CompartirIgual 4.0 Internacional License](#)



# Objetivos

- Conocer los aspectos teóricos y prácticos de los sistemas de bases de datos
  - Justificación
  - Evolución
  - Clasificación
  - Nomenclatura
- Conocer qué es un SGBD

- El banco Malo guarda los datos de sus clientes en una tabla

| Cuenta Corriente | Saldo  |
|------------------|--------|
| 11223344         | 1.500  |
| 22334455         | 10.200 |
| 44556677         | 7.800  |
| 66778899         | 25.456 |
| 56565656         | 2.800  |

Queremos hacer un traspaso de 200 € desde la cuenta: 22334455 a la 56565656. Nuestro informático hace el siguiente programa:

| Cuenta Corriente | Saldo  |
|------------------|--------|
| 11223344         | 1.500  |
| 22334455         | 10.200 |
| 44556677         | 7.800  |
| 66778899         | 25.456 |
| 56565656         | 2.800  |

1. SaldoA  $\leftarrow$  Leer Saldo cuenta de origen
2. SaldoB  $\leftarrow$  Leer Saldo cuenta destino
3. SaldoA  $\leftarrow$  SaldoA - 200
4. Grabar SaldoA en cuenta de origen
5. SaldoB  $\leftarrow$  SaldoB + 200
6. Grabar SaldoB en cuenta de destino

¿Qué ocurre si justo antes de la instrucción 6 se produce un error, y la instrucción 6 no se puede completar?

¿Cambiar el orden de las instrucciones soluciona el problema?

- El banco Malo tiene muchos clientes

| Cuenta Corriente | Saldo  |
|------------------|--------|
| 11223344         | 1.500  |
| 22334455         | 10.200 |
|                  |        |
|                  |        |
|                  |        |
| ...              | ...    |
| 66778899         | 25.456 |
| 56565656         | 2.800  |

Queremos sumar todo el dinero que hay depositado en el banco.

Pero el banco tiene millones de clientes y la operación dura varios minutos

Cuando está a punto de terminar la suma, la cuenta 11223344 hace una transferencia de 500 € a la cuenta 56565656

Cuando el programa leyó de la primera cuenta leyó 1.500, pero cuando va a leer de la última, lee 3.300 puesto que la operación ya se ha completado

¿Cómo podemos resolver el problema?

- La cuenta 22334455 es compartida por Enrique y María. Los 2 sacan dinero a la vez

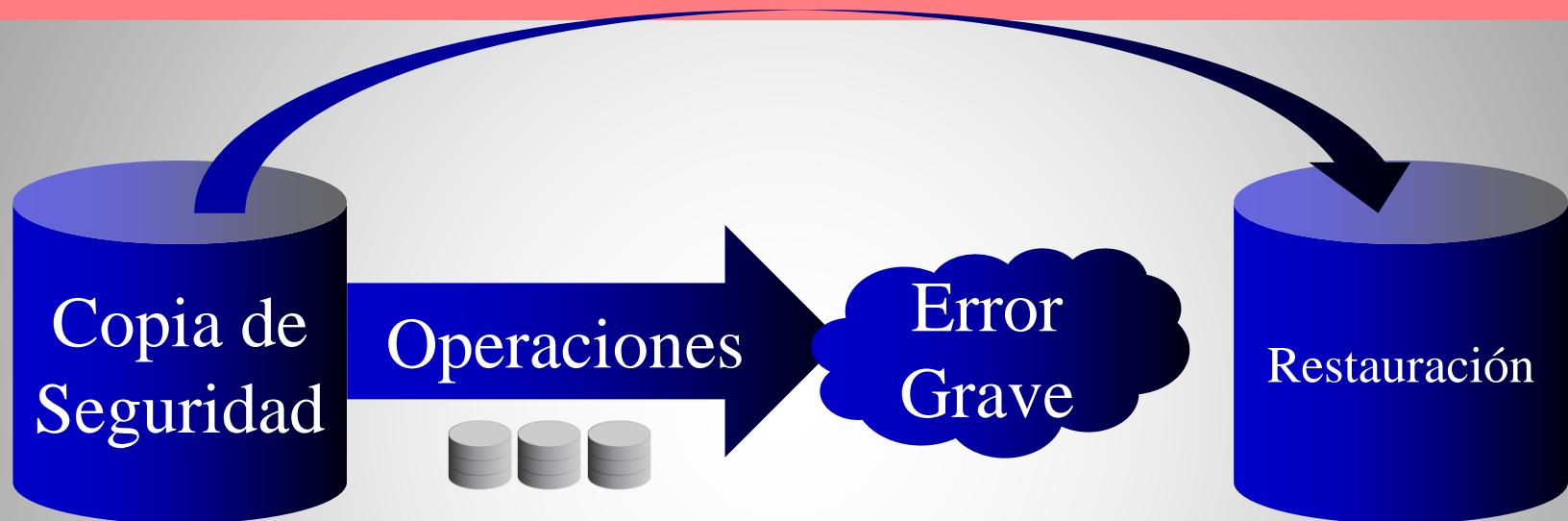
| Cuenta Corriente | Saldo  |
|------------------|--------|
| 11223344         | 1.500  |
| 22334455         | 10.200 |
| 44556677         | 7.800  |
| 66778899         | 25.456 |
| 56565656         | 2.800  |

ENRIQUE saca 200  
 1. SaldoA ← Leer Saldo  
 cuenta de origen  
 (10.200)  
 2. SaldoA ← SaldoA - 200  
 (10.000)  
 3. Grabar SaldoA en  
 cuenta de origen  
 (10.000)

MARIA saca 100  
 1. SaldoA ← Leer Saldo  
 cuenta de origen  
 (10.200)  
 2. SaldoA ← SaldoA - 100  
 (10.100)  
 3. Grabar SaldoA en  
 cuenta de origen  
 (10.100)

Tiempo

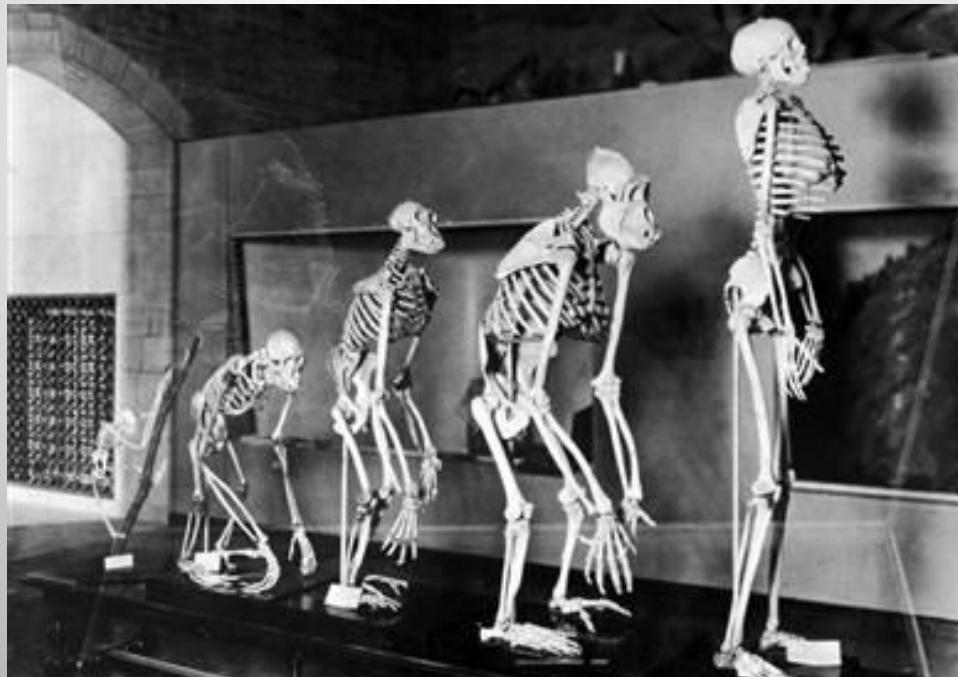
El valor que se graba finalmente es 10.100 ¿Es correcto?



- ¿Qué ocurre con las operaciones realizadas entre la copia y la restauración? ¿Todos los ingresos, transferencias, retiradas de efectivo... se pierden?

# ¿De dónde vienen las B.D.?

- Los sistemas evolucionan para adaptarse a los nuevos requerimientos de su entorno o a los nuevos avances tecnológicos.



## ¿De dónde vienen las B.D.?

- La mejora y abaratamiento de los sistemas de almacenamiento secundario posibilita guardar permanentemente ingentes cantidades de datos.
- Surge la necesidad de almacenar y recuperar los datos de forma eficiente.

**Now you can get our disk systems within 30 days ARO at the industry's lowest prices:**

- **80 Mbytes for under \$12K\***
- **300 Mbytes for under \$20K\***

Field-proven reliability, total software support and 30-day delivery. You've come to expect them all from us. And that's why we've become the world's largest independent supplier of minicomputer disk storage systems.

Now add low price. Lower than the minicomputer manufacturers, lower than any other independent—the lowest in the industry. Why? Because we buy more disk drives than anyone else, and we can afford to pass the OEM discounts on to you.

The prices listed above are for complete disk systems ready to plug into your minicomputer. Each system includes our high-performance controller, an appropriate minicomputer interface and the software of your choice.

When you buy disk systems from us, you'll save a lot more than a lot of money on the purchase price. You'll save precious time. Beginning with our 30-day delivery and continuing with our responsive, customized software support, we'll get your system up quickly—and keep it up. For complete OEM pricing information and technical details, contact the System Industries representative in your area.

**System  Industries**

An equal opportunity employer.  
535 Del Ray Avenue  
Sunnyvale, California 94086  
(408) 732-1650, Telex 346-459

\* OEM prices: 40-69 systems.



## ¿De dónde vienen las B.D.?

- Los sistemas de BD tienen gran presencia hoy día



- ¿Último acceso a una base de datos?

## ¿De dónde vienen las B.D.?

- IRS (Internal Revenue Service): 100 millones usuarios, 5 formularios / usuario, 400 caracteres / formulario. 800Gb



- Amazon: 15 millones visitas / día. 100 empleados actualizan la información diariamente.



# ¿De dónde vienen las B.D.?

- Sistemas orientados hacia el proceso
  - Datos integrados en el programa.
  - Aparecen los ficheros secuenciales
  - Ficheros jerárquicos y de acceso directo
- Sistemas orientados hacia los datos
  - Especial atención a las relaciones entre datos
  - En lo conceptual: Abstracción (grafos, tablas,...)
  - En lo tecnológico: Arquitectura a varios niveles.

# Desventajas del paradigma de ficheros

- Dificultad para compartir datos
- Falta de flexibilidad en seguridad
- Reglas de Integridad en el Software
- Dificultad para responder a nuevos requisitos.

# ¿Cuántas clases de bases de datos hay?



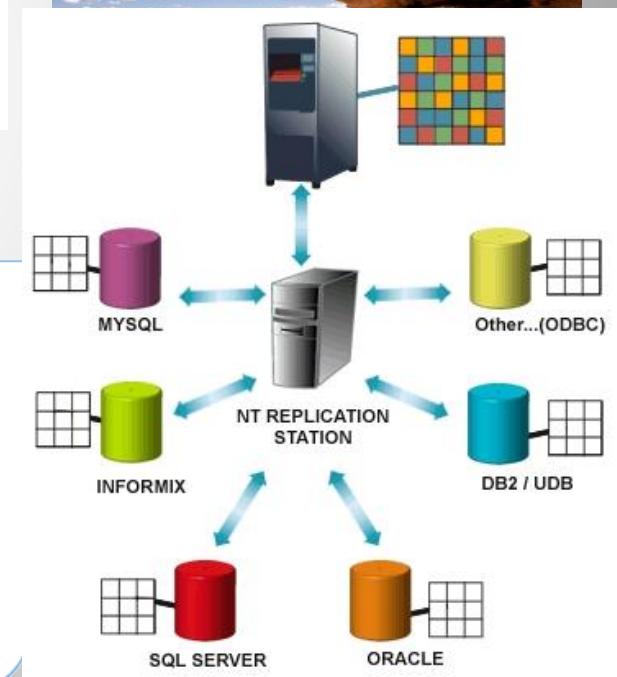
# Según su objetivo

- Propósito general: On-line Transaction Processing (OLTP)
- Análisis de datos: Data Warehousing
- Bases de datos en memoria
- Alta disponibilidad: Clusters



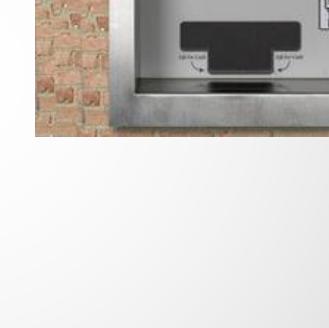
# Según su localización

- Local
- Centralizada
- Distribuida
  - Homogénea
  - Heterogénea



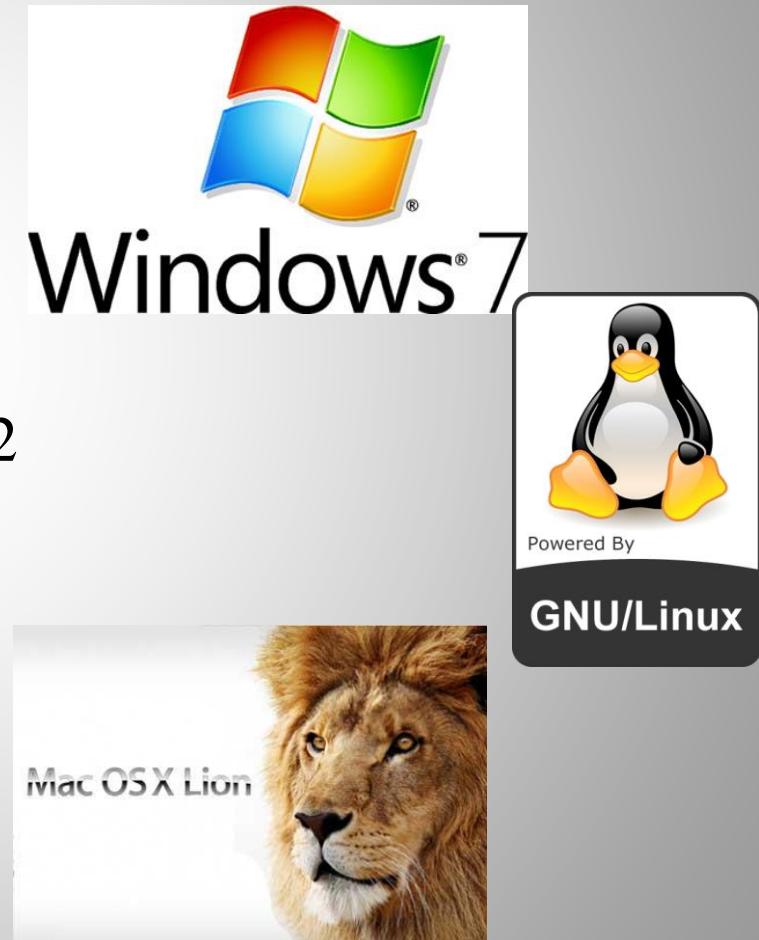
# Según su licencia

- De pago
  - Oracle, DB2, SQLServer, Informix, Access, Filemaker
- Software libre
  - MySQL, MariaDB, PostgreSQL
- Libre uso
  - Google (BigTable)



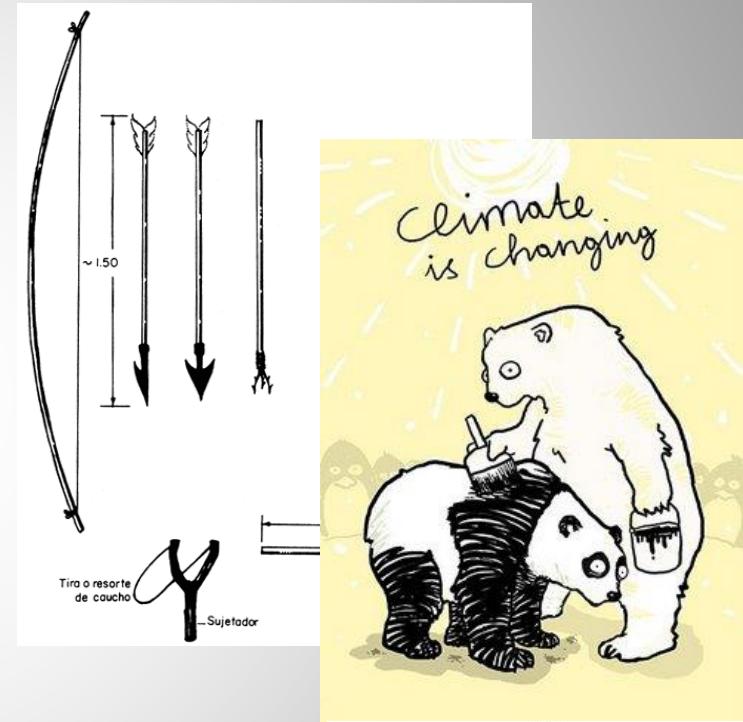
# Según el sistema operativo

- Windows
  - SQLServer, Oracle, PostgreSQL, MySQL
- Linux/Unix
  - MySQL, PostgreSQL, Oracle, DB2
- Mac
  - Oracle, MySQL, PostgreSQL



# Según su modelo de datos

- Árbol. BD Jerárquica
  - Grafo. BD en red
    - Rápidos y eficientes de ejecutar
  - Teoría de Conjuntos. BD Relacional
  - Lógica Clásica. BD Deductiva
  - Orientación a Objetos. BDOO
    - Flexibles, de rápido diseño y mantenimiento
  - Bases de datos XML
  - Bases de datos RDF
  - Bases de datos NoSQL



# Bases de Datos NoSQL

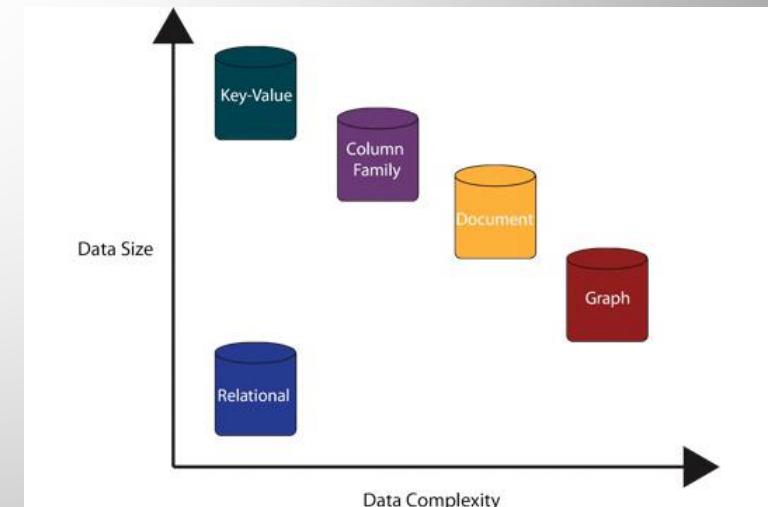
- NoSQL – "not only SQL" – es una categoría general de sistemas de gestión de bases de datos que difiere de los RDBMS en diferente modos:
  - No tienen esquemas, no permiten JOINs, no intentan garantizar ACID y escalan horizontalmente
  - Tanto las bases de datos NoSQL como las relacionales son tipos de Almacenamiento Estructurado.

# ACID vs. BASE

- En el mundo relacional estamos familiarizados con las transacciones ACID, que garantizan la consistencia y estabilidad de las operaciones pero requieren bloqueos sofisticados:
  - ACID = Atomicidad, Consistencia, (Isolation) aislamiento y Durabilidad
- Las BBDD NoSQL son repositorios de almacenamiento más optimistas, siguen el modelo BASE:
  - Basic Availability – el almacén funciona la mayoría del tiempo incluso ante fallos gracias al almacenamiento distribuido y replicado
  - Soft-state – los almacenes no tienen por qué ser consistentes ni sus réplicas en todo momento. El programador puede verificar esa consistencia.
  - Eventual consistency – la consistencia se da eventualmente
- BASE es una alternativa flexible a ACID para aquellos almacenes de datos que no requieren un adherencia estricta a las transacciones.

# Taxonomía de Soluciones NoSQL

- Los principales tipos de BBDD de acuerdo con su implementación son los siguientes:
  - Almacenes de Clave-Valor:
    - Voldemort
  - Almacenes de Familia de Columnas:
    - Cassandra, Hbase
  - Almacenes de Documentos:
    - MongoDB
  - Grafos:
    - AllegroGraph, VertexBD, Neo4j



# Arquitectura en tres niveles

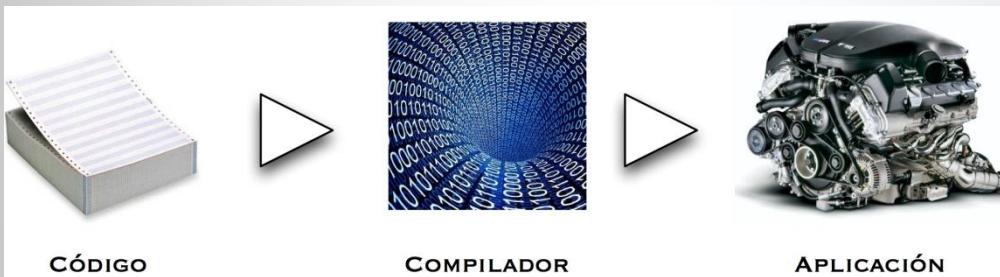
- Nivel **Externo**. Cercano a los usuarios
- Nivel **Conceptual**. Contenido Global
- Nivel **Interno**. Descripción a nivel físico

Informe ANSI/X3/SPARC. Standard Planning and Requirements Committee of the American National Standards Institute on Computers and Information Processing.

*Interim Report: ANSI/X3/SPARC Study Group on Data Base Management Systems 75-02-08, FDT - Bulletin of ACM SIGMOD. pp 1-140. 1975*

# Arquitectura en tres niveles

- Un **esquema** es la descripción de la estructura de un nivel de la BD.



# Arquitectura en tres niveles

## • Interno

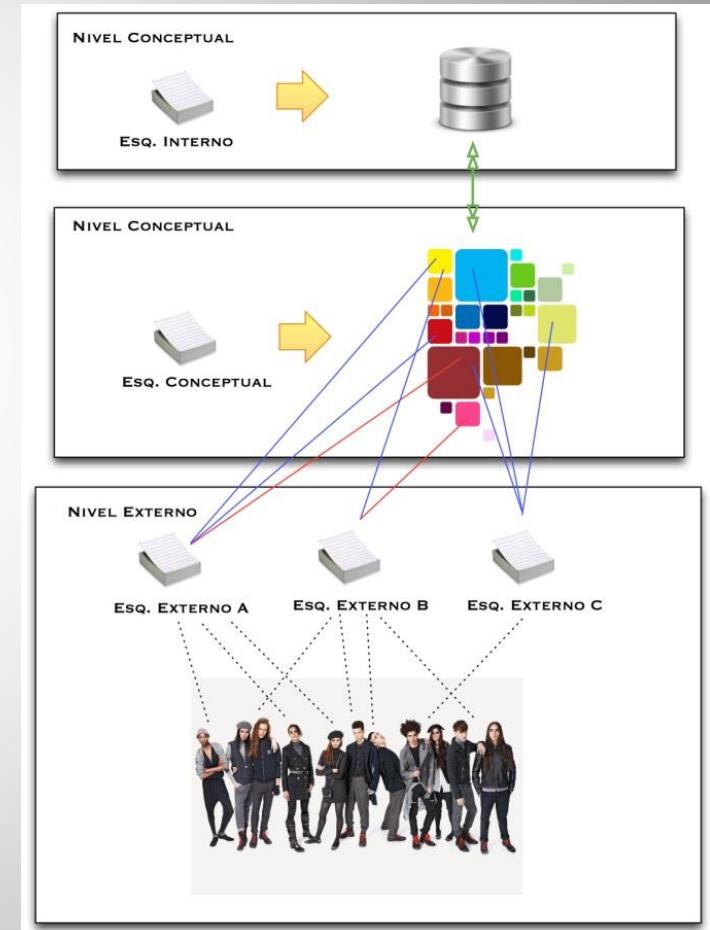
- Descripción de las estructuras de almacenamiento, métodos de recuperación eficiente, dispositivos, etc.

## • Conceptual

- Proporciona los nombres de las entidades, sus características y las relaciones que existen entre ellas.

## • Externo

- Visión que tienen los usuarios de los datos que utilizan y operaciones sobre ellos. Por tanto, existirán tantos subesquemas como tipos de usuarios tenga la BD.



## Arquitectura en tres niveles

Grabczewski, Edward (2005). "Un depósito de datos corporativo que apoya la investigación científica en el Reino Unido". Consejo para el laboratorio central de los consejos de investigación (CCLRC).

<https://epubs.stfc.ac.uk/work/35241>

# Arquitectura en tres niveles

- **Lenguajes de definición de datos (DDL).**

- Para definir los esquemas externo, conceptual e interno.

- **Lenguajes de manipulación de datos (DML)**

- Para realizar consultas y actualizaciones.

- **Lenguajes de Programación.**

- Ejemplo: PL/SQL

# Arquitectura en tres niveles

- Objetivo: **Independencia de los datos**. Propiedad de modificar un nivel de la base de datos sin que se vean afectados los niveles superiores.
  - Independencia **lógica**. Se puede modificar la estructura de la base de datos sin modificar las aplicaciones.
  - Independencia **física**. Se puede alterar la estructura física de la base de datos sin alterar su estructura lógica.
- Herramientas: correspondencias
- Ventaja: **persistencia de Software**



Área de trabajo A1



Área de trabajo A2

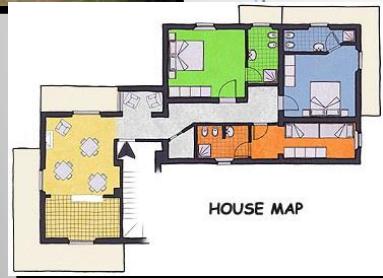


Área de trabajo B1

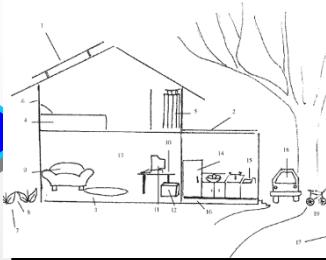


MA EXTERNO A

ESQUEMA E



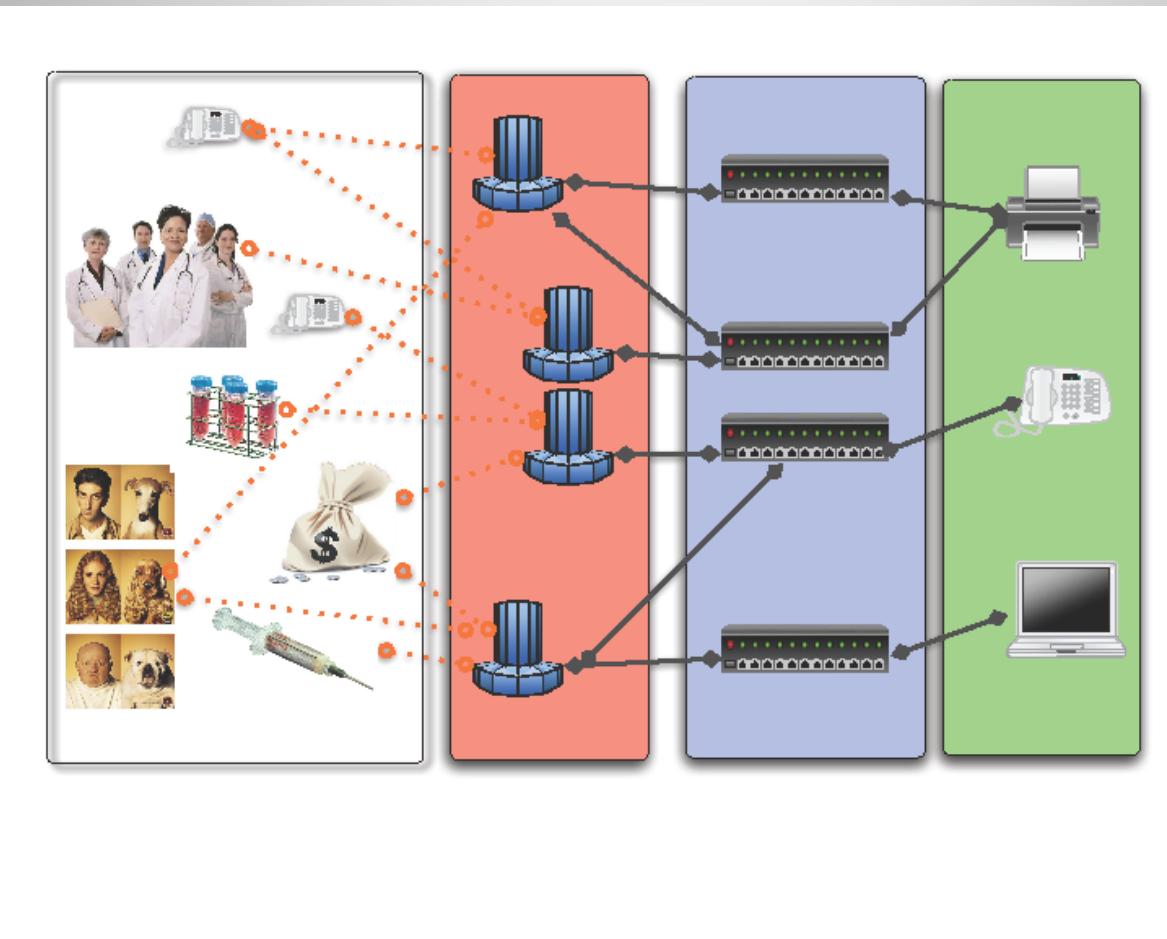
ESQUEMA  
CONCEPTUAL



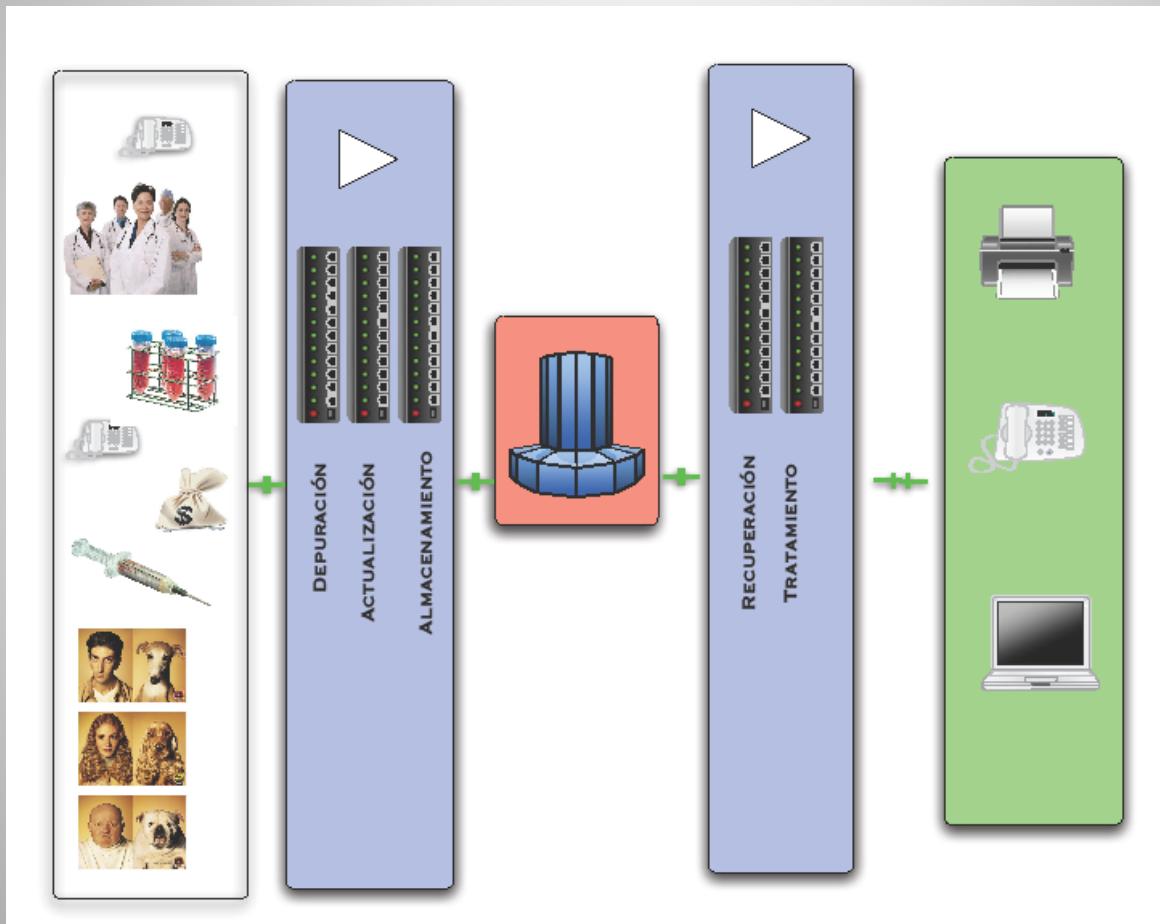
SQUEMA  
INTERNO



# Arquitectura de ficheros



# Arquitectura Base de Datos



# Sistema de Base de Datos

## •Base de datos

- Colección de datos que están lógicamente relacionados entre sí, que tienen una definición y una descripción comunes y que están estructurados de una forma particular.

+

## •Sistema Gestor de Base de Datos

- Colección de programas que aseguran el acceso a los datos.

# Base de Datos

- La base de datos contiene los propios datos y una definición de su estructura y restricciones. Básicamente lo que hemos definido en los esquemas.

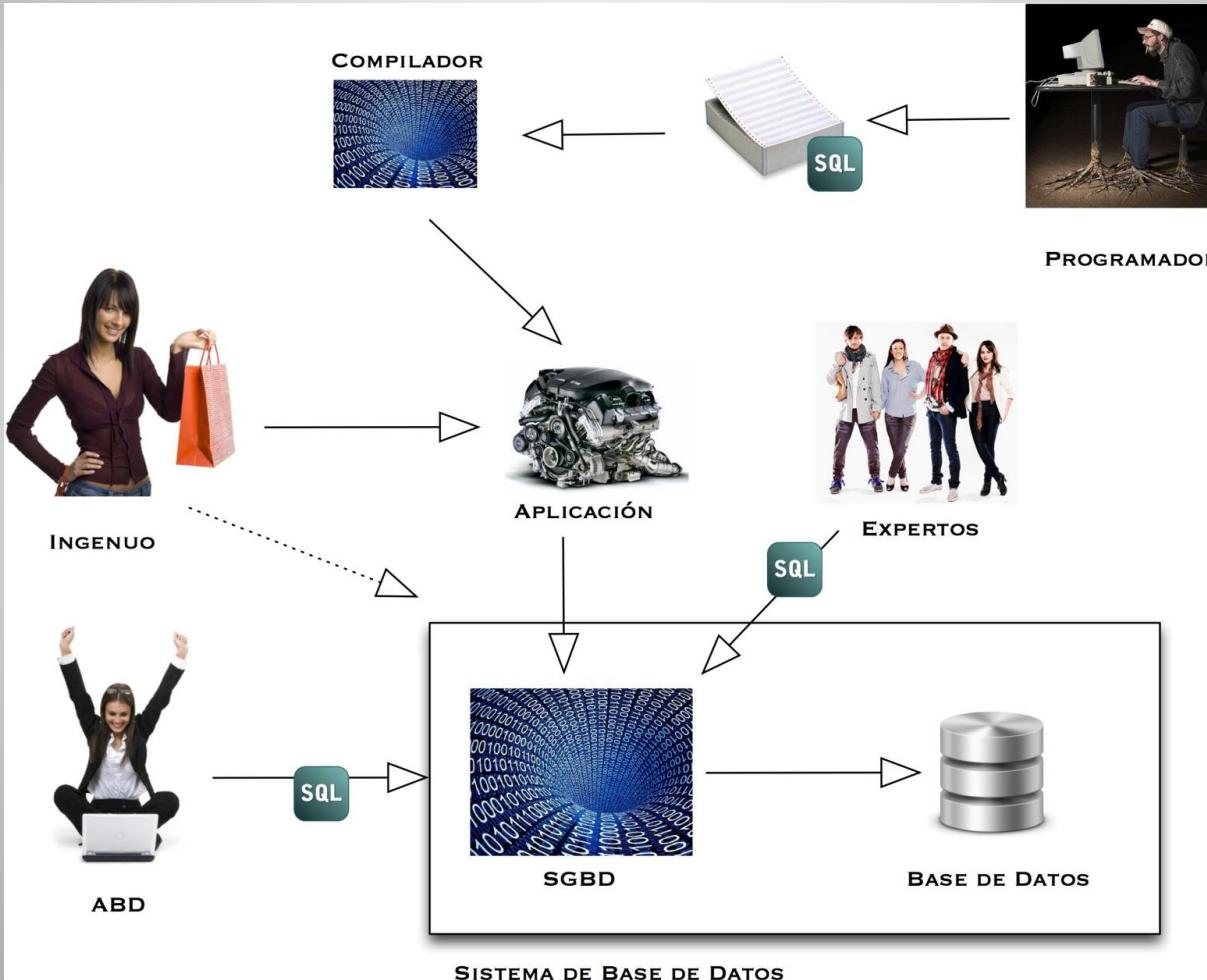
## • Metadatos

- Consiste en un catálogo completo con toda la descripción de la estructura y las restricciones de los datos.

# Sistema Gestor de Base de Datos

- Conjunto coordinado de programas, procedimientos, lenguajes, etc, que suministra, tanto a los usuarios informáticos, como no informáticos y al Administrador, los medios necesarios para describir, recuperar y manipular los datos integrados en la BD, asegurando su confidencialidad y seguridad.
- Tareas
  - Crea y mantiene los objetos de la BD
  - Permite modificar y operar con los datos
  - Rutinas de recuperación
  - Mantenimiento de la seguridad
  - Mantenimiento Reglas de Integridad

# Usuarios en los sistemas de BD



# El Administrador

- Gestiona la disponibilidad de la Base de Datos
- Describe esquemas y crea las Bases de Datos
- Gestiona las estructuras físicas
- Gestiona los dispositivos de almacenamiento
- Gestiona la seguridad

- Administración de la red
- Recuperación y backups
- Afinamiento de la BD

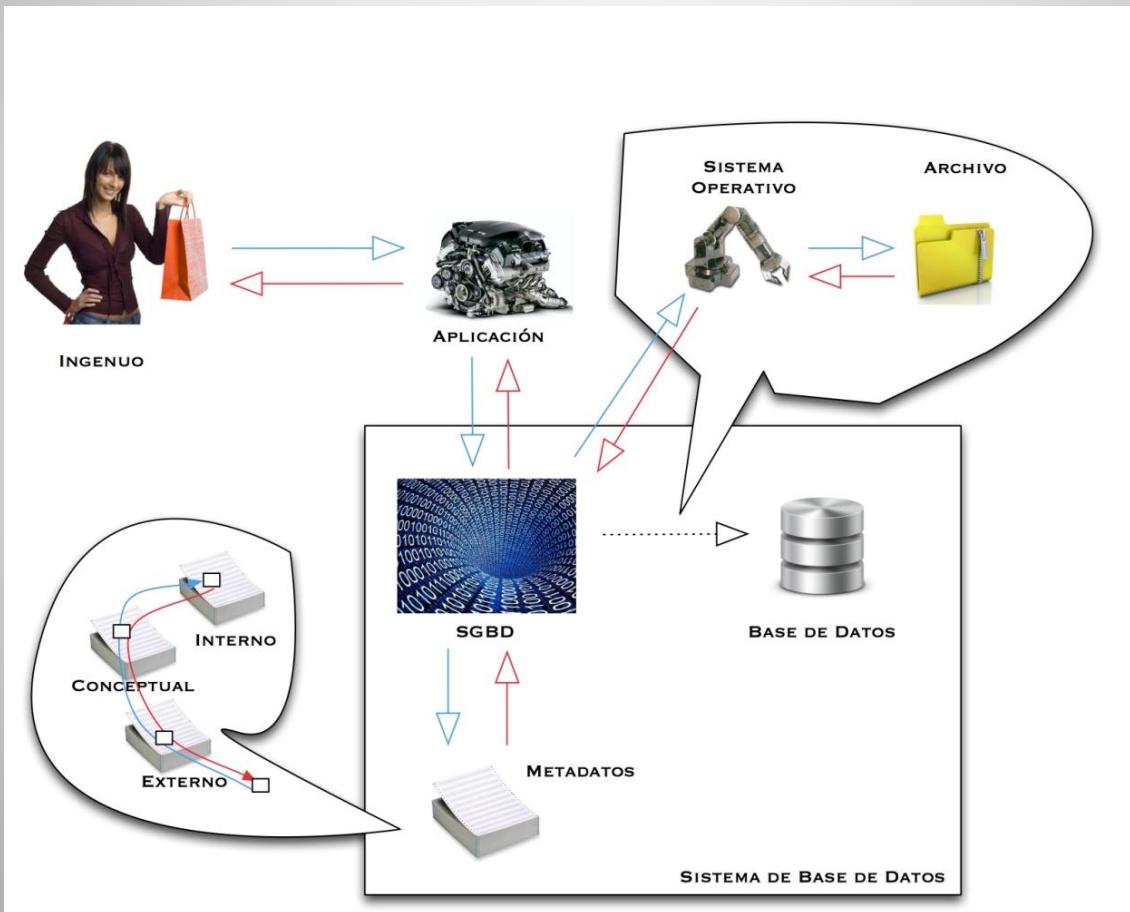


# Acceso a los sistemas de base de datos

- ¿Qué ocurre cuando accedemos a una base de datos?



# Acceso a los sistemas de base de datos



# Autoevaluación



# Tema 2. Diseño Lógico de Bases de Datos

Departamento de Lenguajes y  
Ciencias de la Computación

Diseño Lógico de Bases de Datos is licensed under a  
[Creative Commons Reconocimiento-NoComercial-CompartirIgual 4.0 Internacional License](#)



- Conocer el ciclo de vida del diseño de una base de datos
- Conocer los componentes de un modelo Entidad/Relación
- Desarrollar un modelo Entidad/Relación

- Introducción
- Modelo Entidad/Relación
- Modelando las entidades
- Modelando las relaciones
- Otros elementos del diagrama
- Entidades Débiles
- Relaciones Es\_un: subentidades.

# El ciclo de vida del diseño de BD

- Análisis de requisitos
- Diseño Lógico
  - Modelado Conceptual
  - Integración de vistas
  - Modelado Relacional
  - Normalización
- Diseño Físico
- Monitorización y afinamiento.

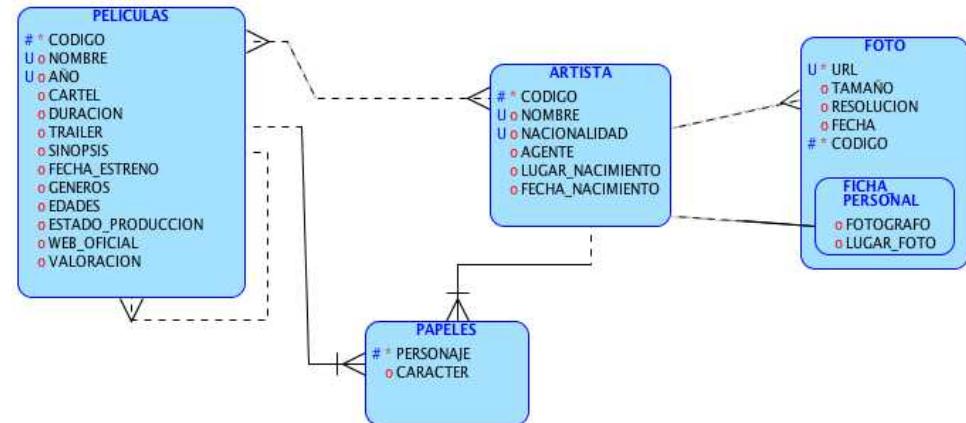
# Análisis de requisitos

- Entrevistas con los usuarios.
- Datos, relaciones y restricciones.
- Plataforma software (opcional).



# Diseño Lógico: diseño conceptual.

- Modelo Entidad/Relación.
- Todos los datos y sus relaciones.
- Esquema global.



# Diseño Lógico: integración de vistas.

- Diseños grandes.
- Varias personas en el análisis.
- Eliminación de redundancia e inconsistencia.



# Diseño Lógico: modelado relacional.

- Tablas.
- SQL.
- Reglas de transformación (algoritmo).



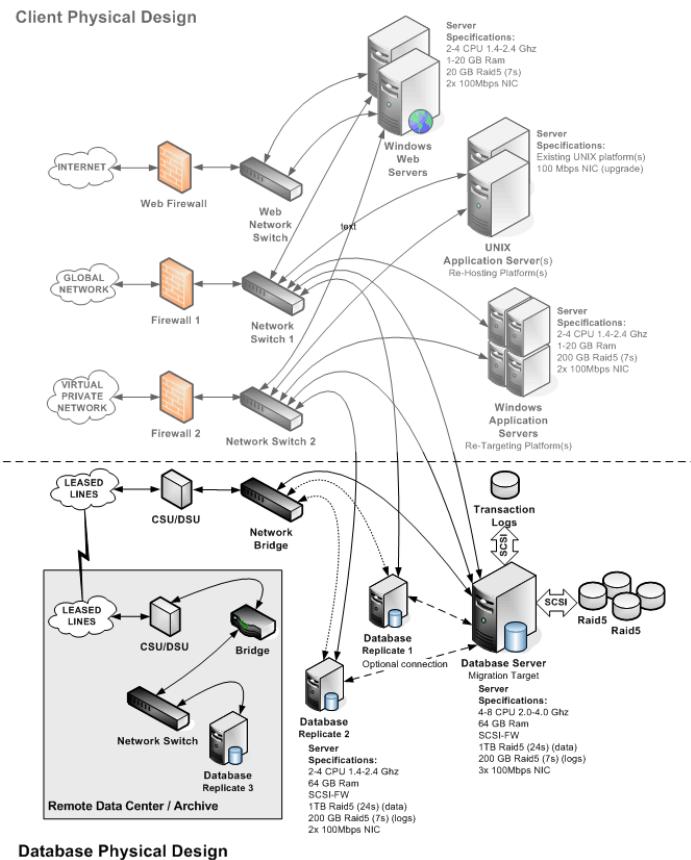
# Diseño Lógico: normalización.

- Técnica eliminación anomalías.
- Disponer de BD sin redundancia y fáciles de mantener.



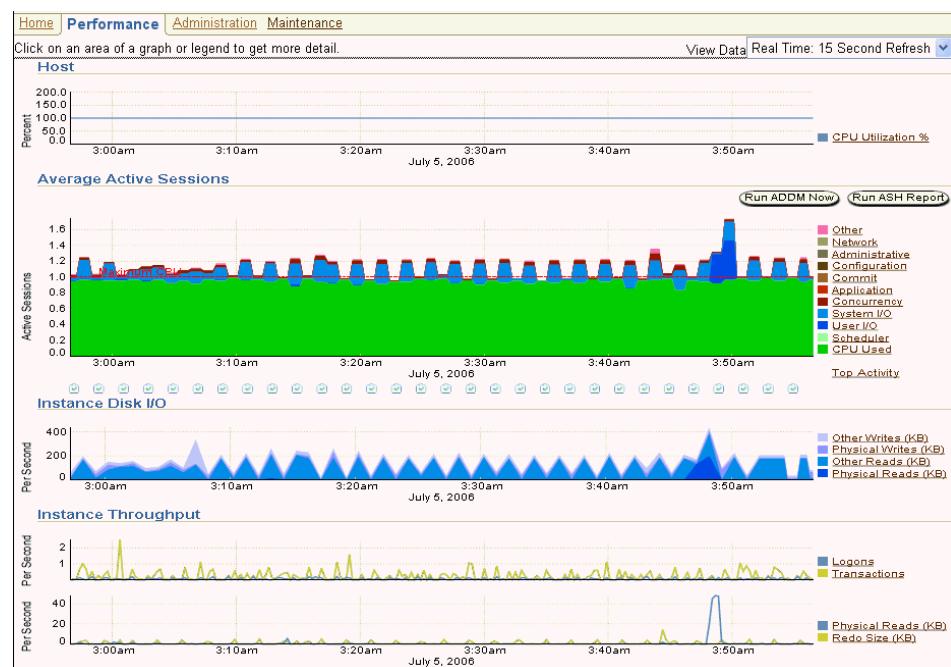
# Diseño físico.

- Métodos de acceso (índices).
- Particionamiento y clusters.
- Rendimiento.
- Desnormalización.



# Monitorización y afinamiento

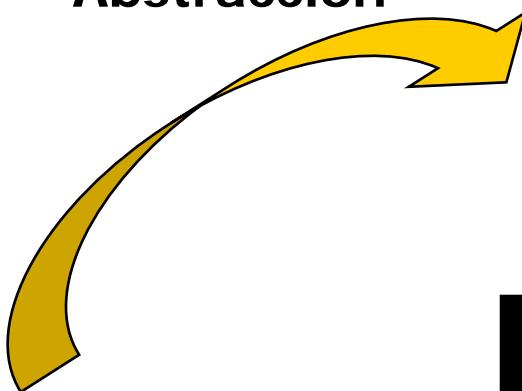
- Implementación (DDL).
- Modificaciones
  - Rendimiento
  - Requisitos



# Nuestro proceso de diseño

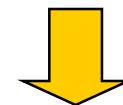


Abstracción



Modelo E/R

ENTIDADES  
ATRIBUTOS  
RELACIONES



Modelo Relacional

```
CREATE TABLE veterinarios
```

```
CREATE TABLE vacunas
```



Base de Datos

# Requerimientos

“Vamos a gestionar el Departamento de Recursos Humanos de una gran Compañía. Para ello, necesitamos obtener información sobre cada *empleado*. Necesitamos conocer su *nombre*, *apellidos*, *puesto*, *fecha de ingreso* y *salario*. Algunos empleados reciben una *comisión*. Cada empleado tiene asignado un *único número de empleado*.

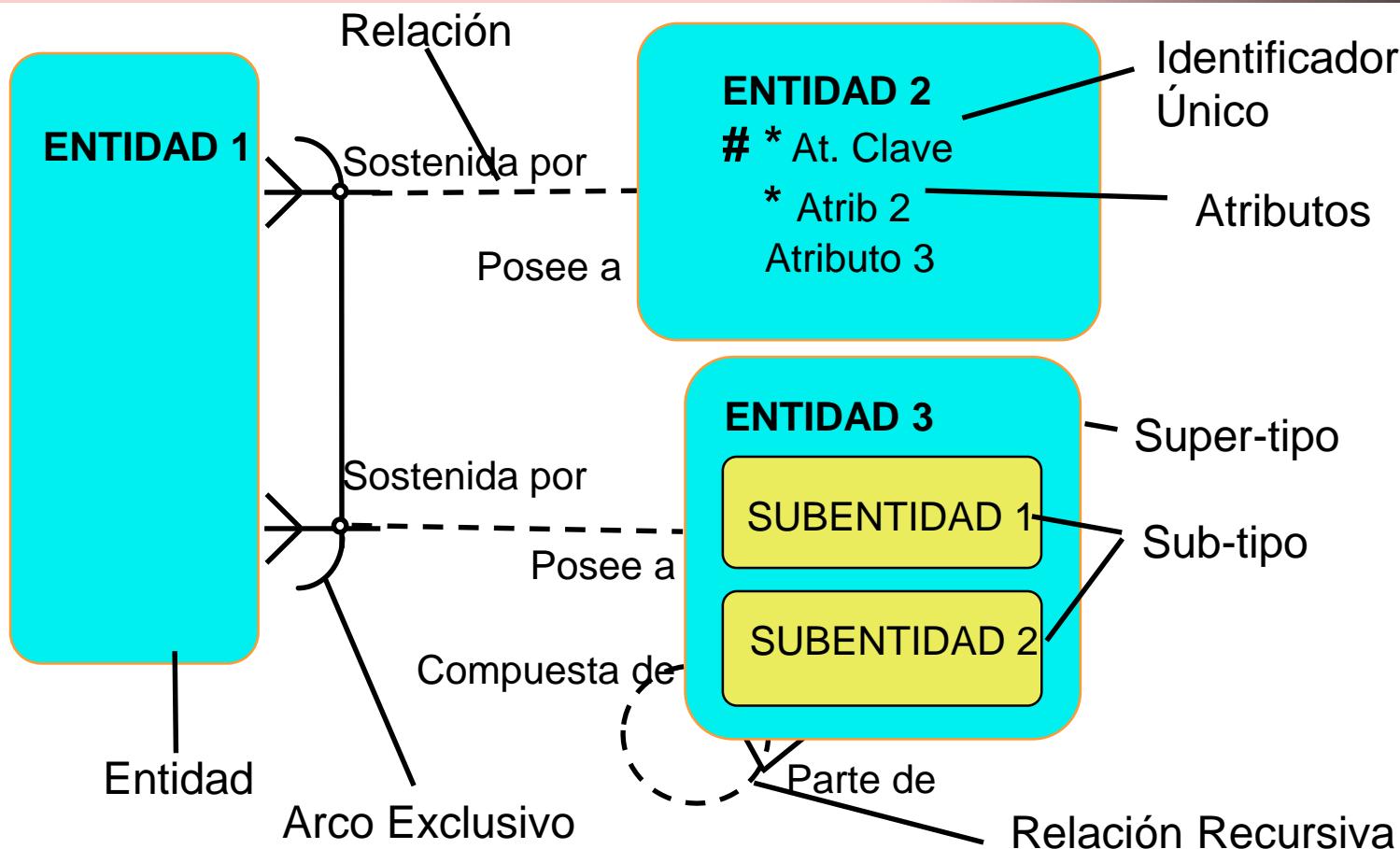
La Compañía se divide en *departamentos*. Cada empleado está *asignado a un departamento*. Necesitamos conocer el *departamento al que se adscribe cada empleado* y su *localización*. Cada *departamento tiene un número único*.

Algunos empleados son jefes. Debemos conocer cada *jefe de empleados* y los *empleados que están a su cargo*”.

# Requerimientos

- Identificar sujetos del modelo
- Identificar sus propiedades
- Identificar sus interconexiones

# Elementos Modelo E/R



# Entidad

“*Seres distinguibles del mundo real*”

- Un objeto de interés
- Real o abstracto
- Un nombre o sustantivo
- Algo sobre lo que la organización necesita información

# Atributo

“*Propiedades de las entidades*”

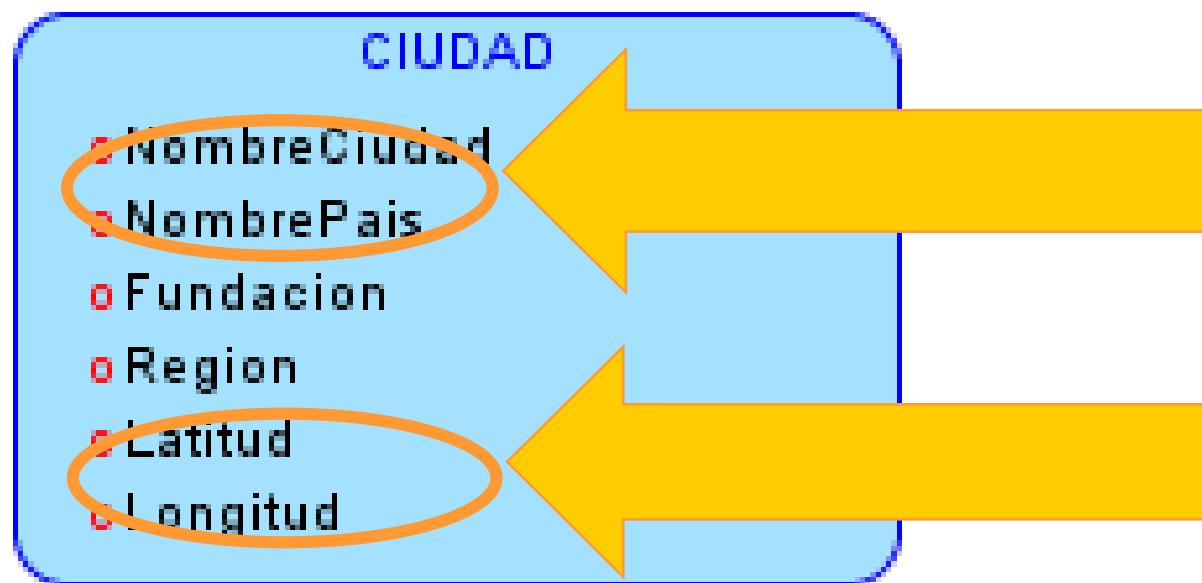
- Se usan para describir las entidades
- Especifican elementos de información:  
**DATOS**
- Una entidad posee muchos atributos

# Diagrama de Entidades

- Rectángulo ‘suave’
- Un nombre único en mayúsculas
- Atributos en minúsculas



# Restricción de identificación: CLAVE



# Modelando Entidades

## Construcción BOTTOM-UP:

- Identificar los atributos
- Surgirán las entidades
- Identificar el objeto (sustantivos)
- Dibujar un rectángulo para ella

# Ejemplo

“Una empresa dedicada a la formación imparte cursos. Cada uno de ellos tiene un código, un nombre, y un precio. Los cursos tienen una duración de uno a cuatro días.

Existen instructores de los que necesitamos conocer su nombre y teléfono.

Los alumnos pueden matricularse en varios cursos a la vez, ¡incluso de todos!

Debemos conocer el nombre y teléfono de los alumnos.”

# Ejemplo

“Una empresa dedicada a la formación imparte cursos. Cada uno de ellos tiene un **código**, un **nombre**, y un **precio**. Los cursos tienen una **duración** de uno a cuatro días.

Existen instructores de los que necesitamos conocer su **nombre** y **teléfono**.

Los alumnos pueden matricularse en varios cursos a la vez, ¡incluso de todos!

Debemos conocer el **nombre** y **teléfono** de los alumnos.”

# Ejemplo

nombre  
teléfono

código  
nombre  
precio  
duración

nombre  
teléfono

# Ejemplo

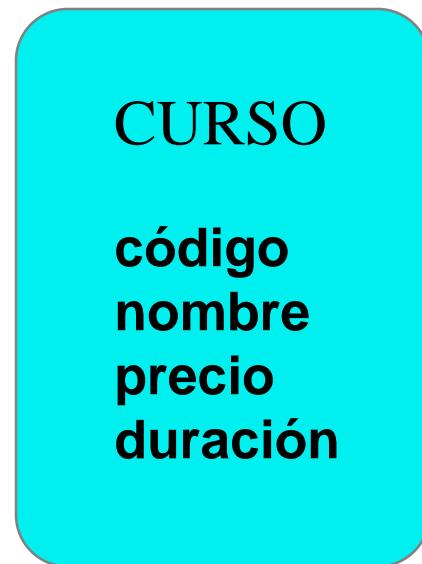
“Una empresa dedicada a la formación imparte CURSOS. Cada uno de ellos tiene un código, un nombre, y un precio. Los cursos tienen una duración de uno a cuatro días.

Existen INSTRUCTORES de los que necesitamos conocer su nombre y teléfono.

Los ALUMNOs pueden matricularse en varios cursos a la vez, ¡incluso de todos!

Debemos conocer el nombre y teléfono de los alumnos.”

# Ejemplo



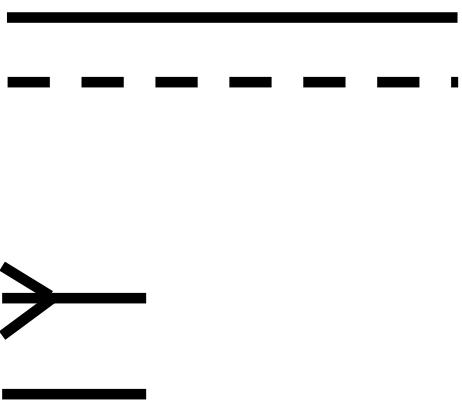
# Relaciones

“*Conexiones entre las entidades*”

- Esquema binario
- Doble dirección
- Nombradas
- Orden
- Obligatoriedad

# Diagramas

- Una línea que conecta dos entidades
- Nombres de la relación en minúsculas



- Opcionalidad

Obligatoria - *debe ser*

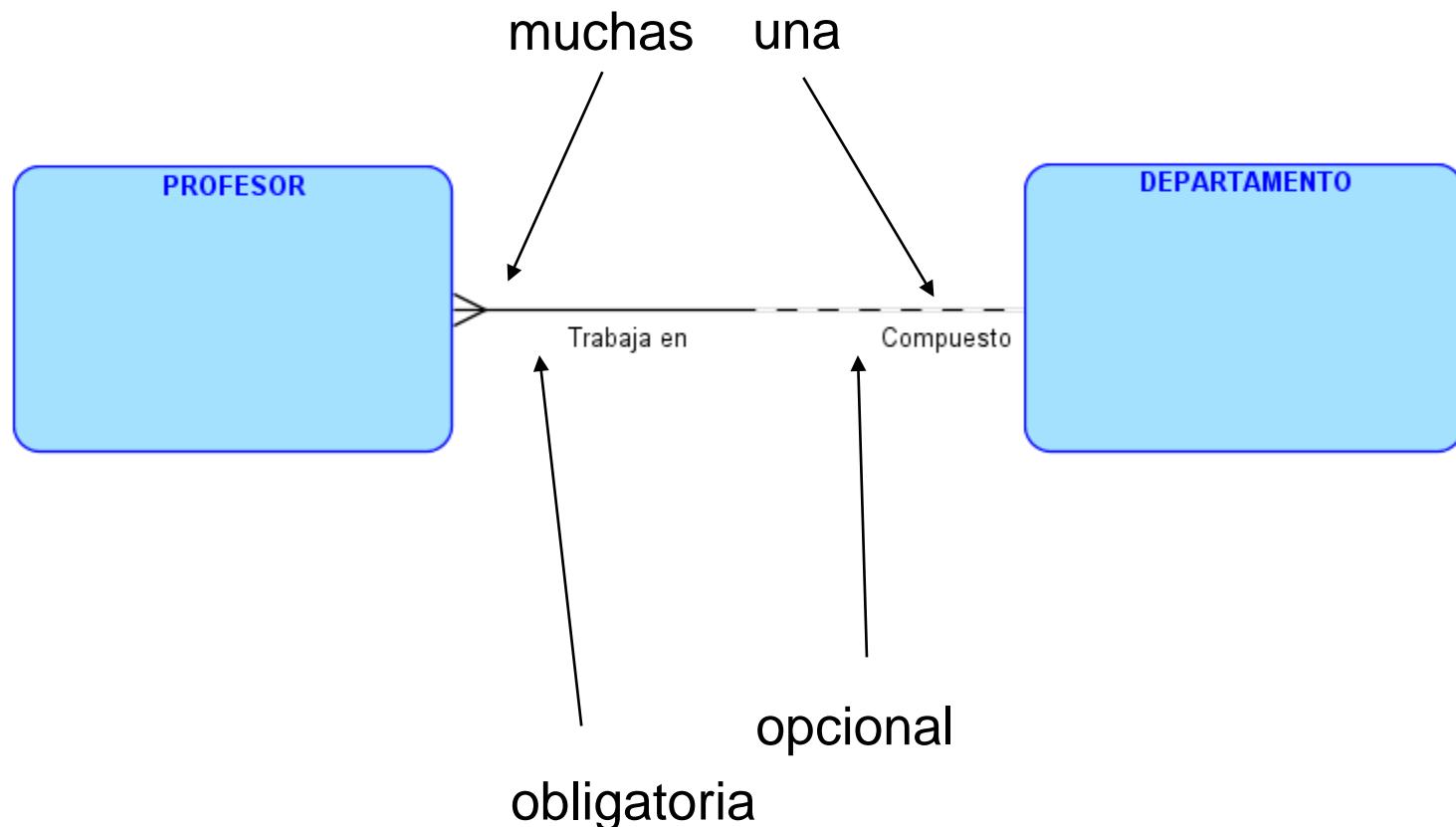
Opcional - *puede ser*

- Grado

Una o más

Una y sólo una

# Diagramas



# Semántica de las Relaciones

Cada

entidad 1

( Debe ser  
o  
puede ser )

Nombre de  
la relación

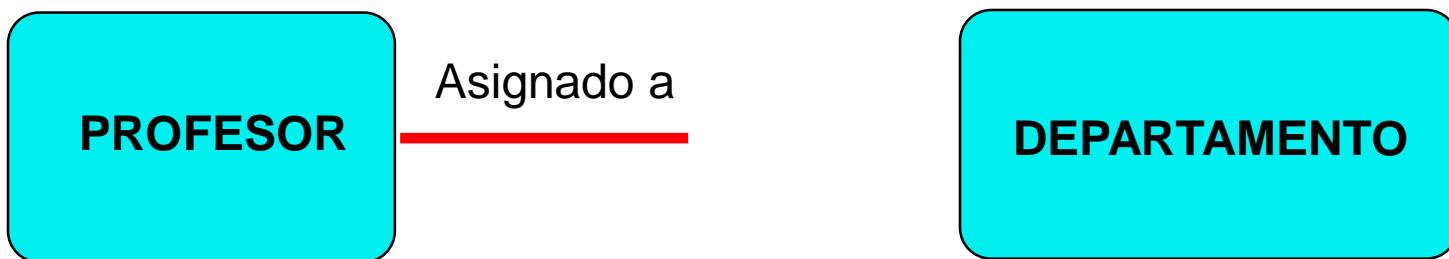
( Una o más  
o  
una y sólo una )

entidad 2

# Ejemplo

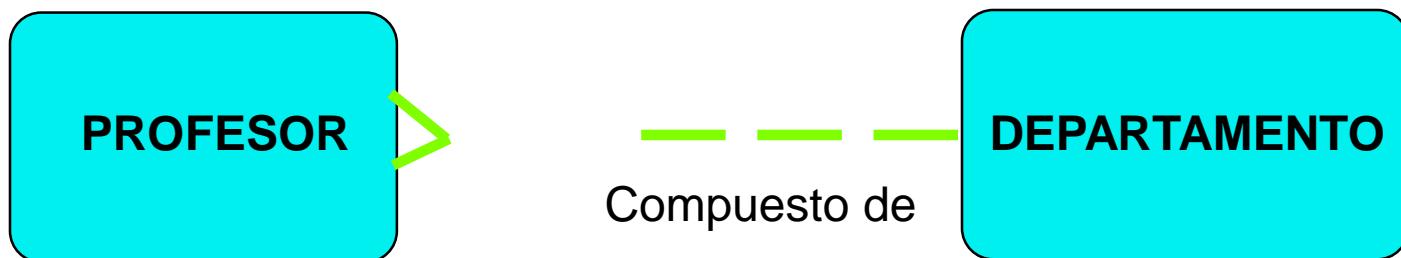


# Ejemplo



Cada **PROFESOR** debe ser asignado a uno y sólo un **DEPARTAMENTO**

# Ejemplo



Cada DEPARTAMENTO puede estar compuesto de uno o más **PROFESORES**

# Ejemplo



Cada DEPARTAMENTO puede ser responsable de uno o más PROFESORES

# Tipos de Relación



Muchos a uno  
M:1

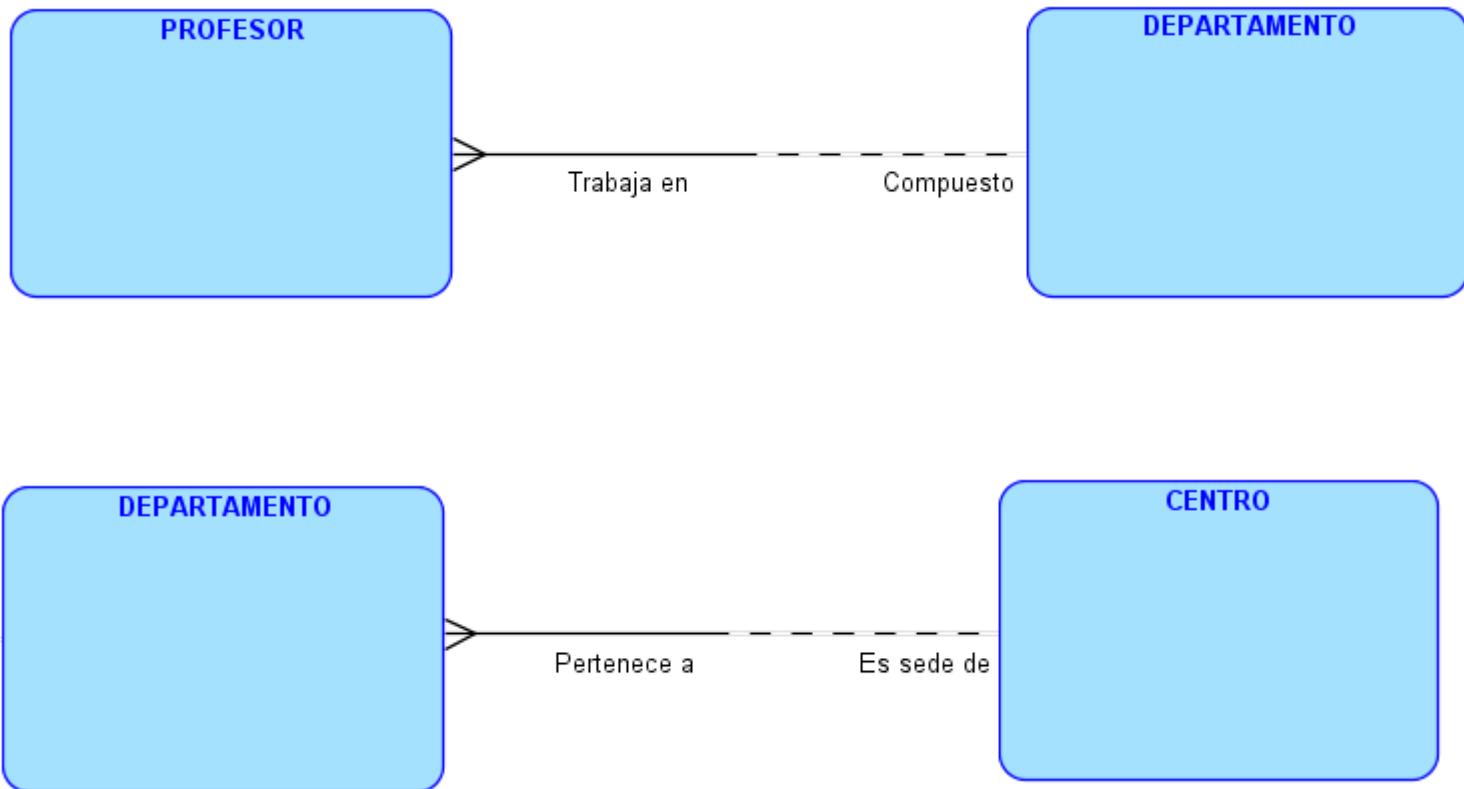


Muchos a muchos  
M:M

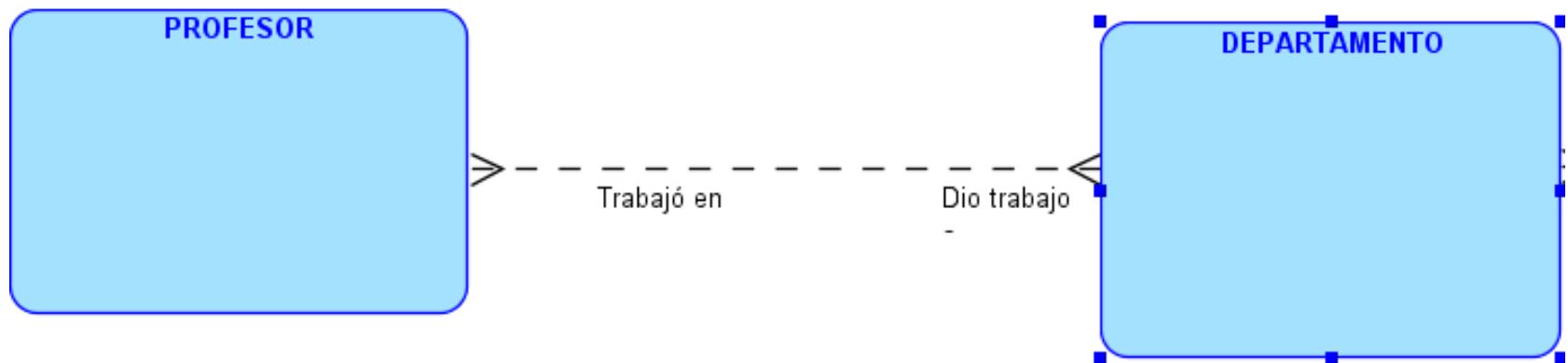


Uno a uno  
1:1

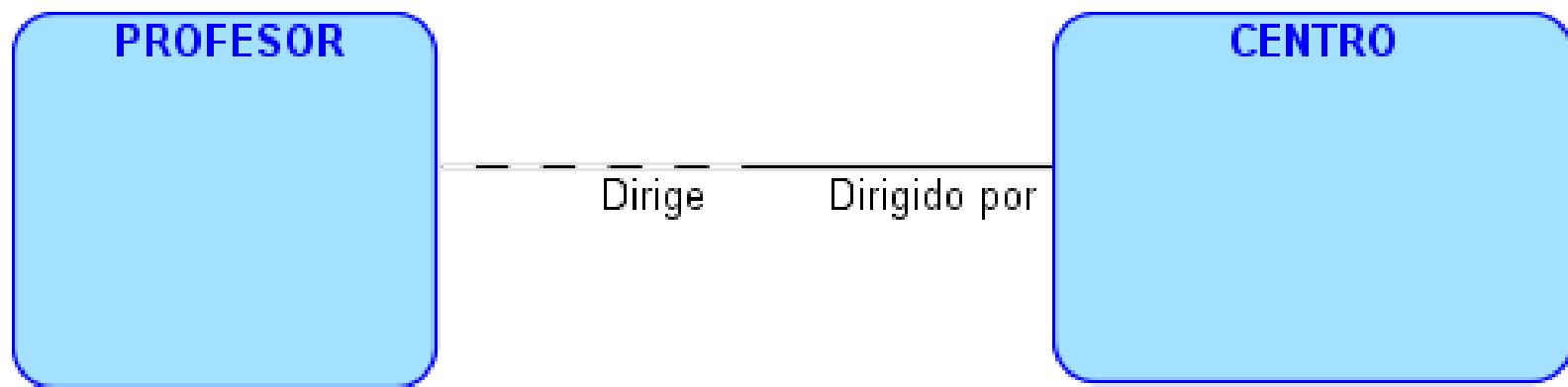
# Relaciones Muchos a Uno



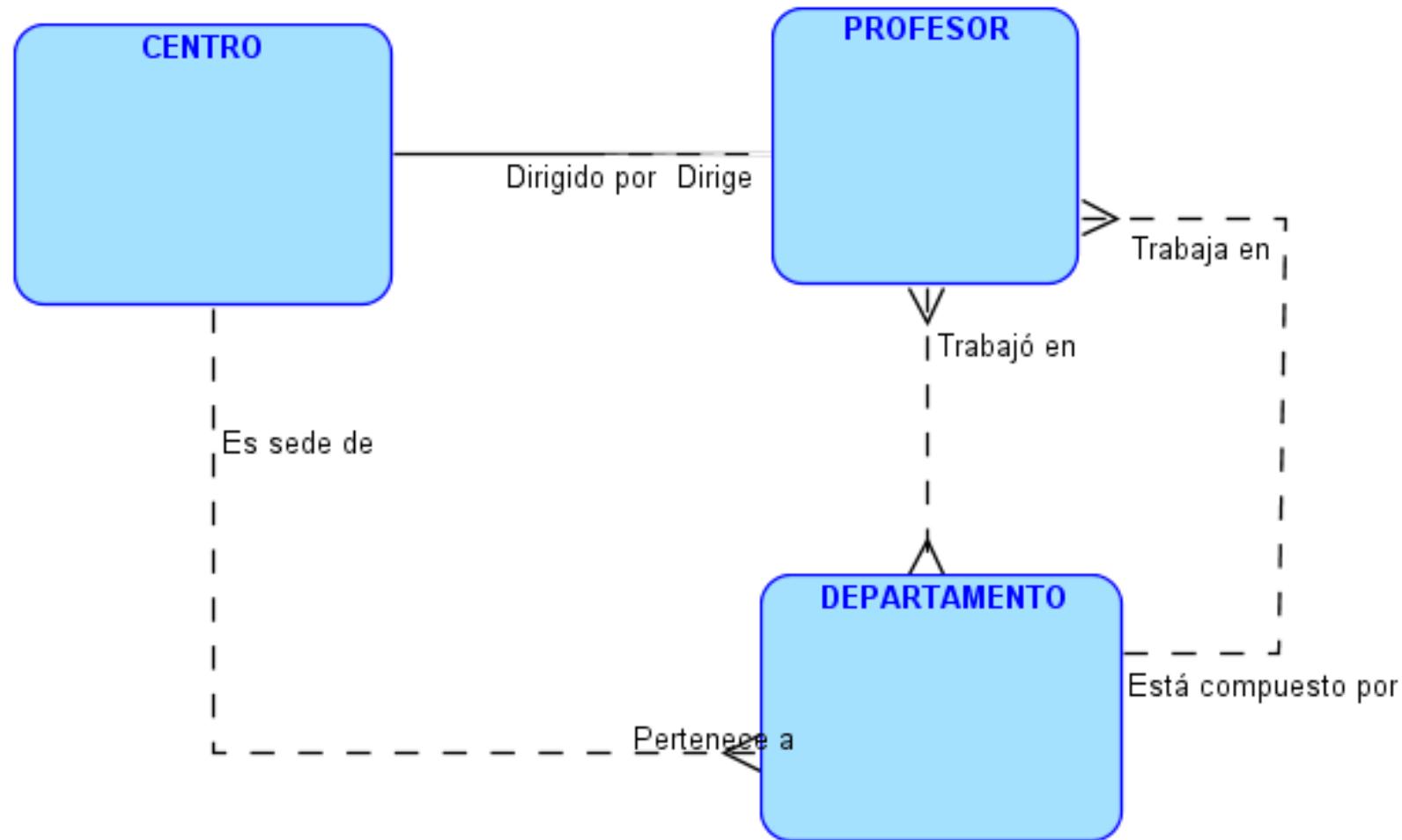
# Relaciones Muchos a Muchos



# Relaciones Uno a Uno



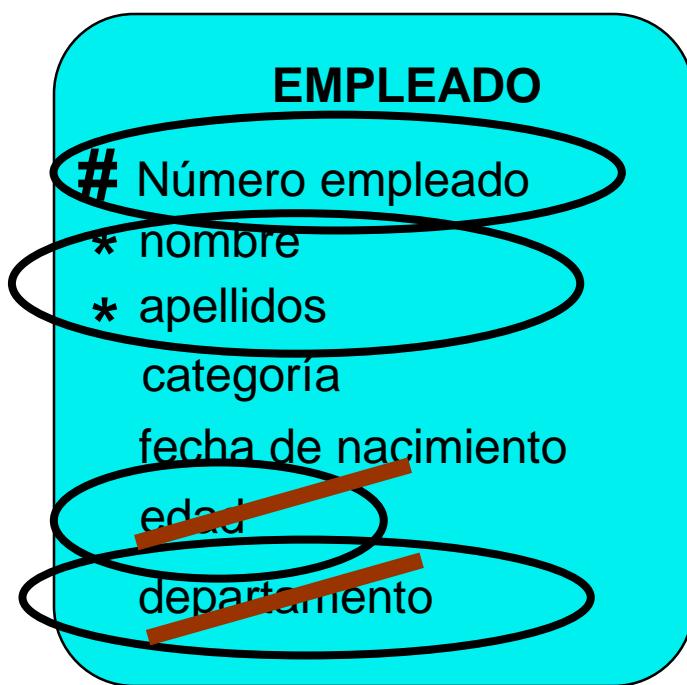
# Todas las Relaciones Juntas



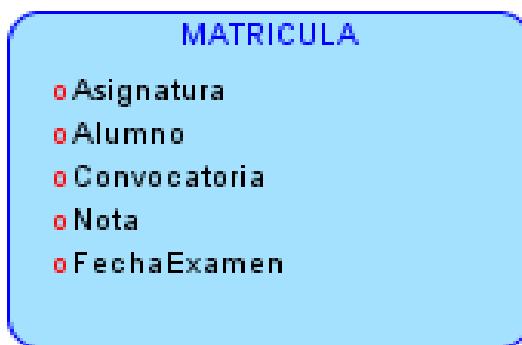
# Atributos

- Los atributos son los datos del mundo real
- No tienen estructura. Atomicidad.
- No son computables a partir de otros atributos
- Son mono-valuados
- Pueden ser requeridos
- **NUNCA REPRESENTA OTRA ENTIDAD !!!!**

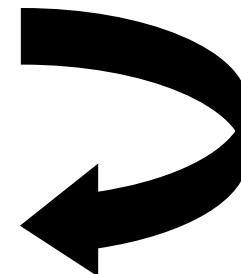
# Atributos



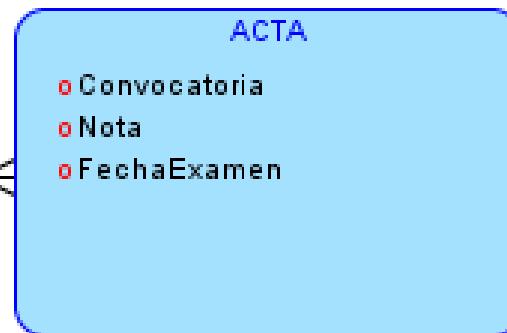
# Valores Simples



OJO: multivaluado



Otra entidad



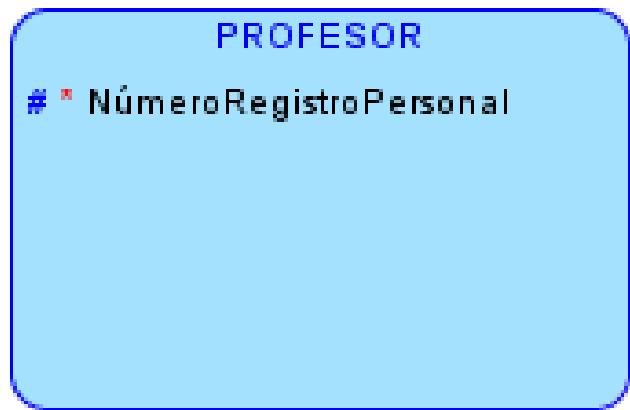
# Claves

Cada instancia de una entidad debe ser identificada de manera única

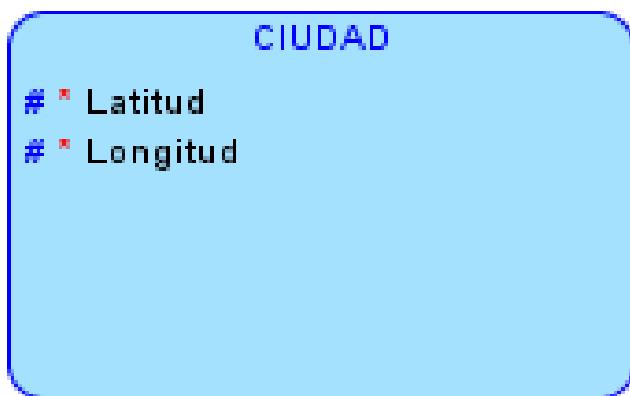


Regla de completitud: siempre puede determinarse un subconjunto de atributos que la identifica.

# Claves Simples y Compuestas



Clave simple



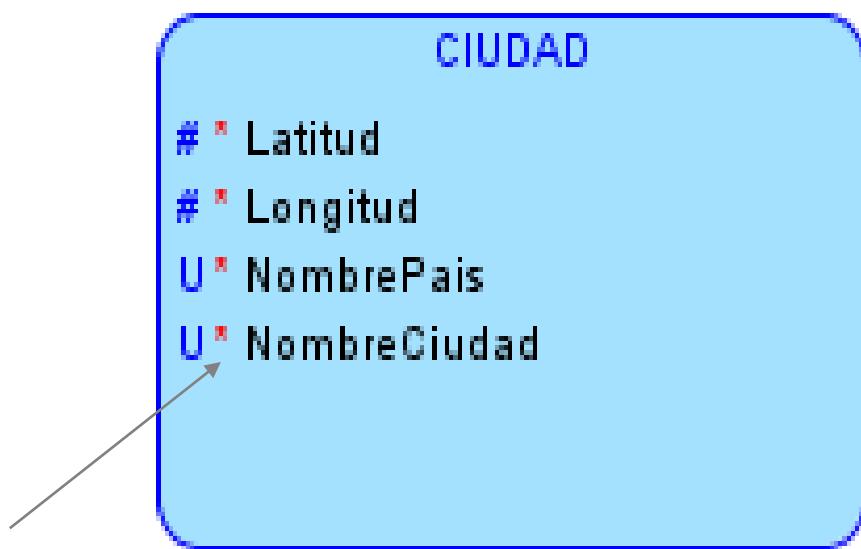
Clave compuesta

# Claves Primarias y Candidatas

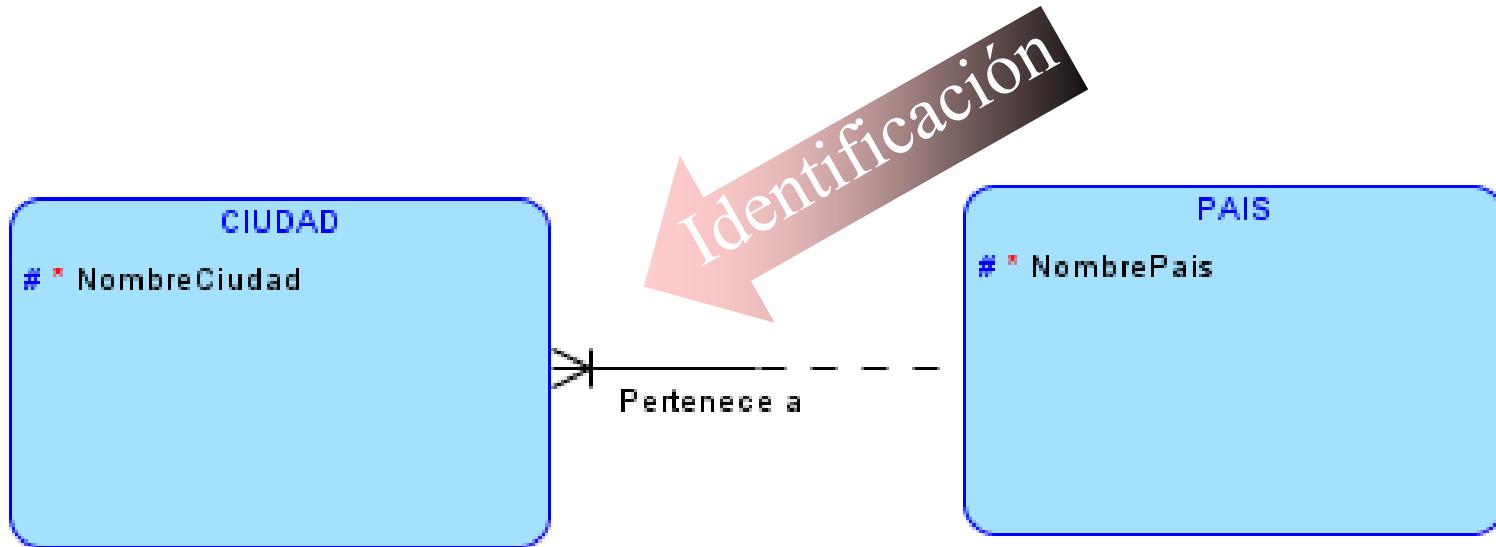
## CIUDAD

# \* Latitud  
# \* Longitud  
U o NombrePais  
U o NombreCiudad

# Claves Candidatas y obligatoriedad

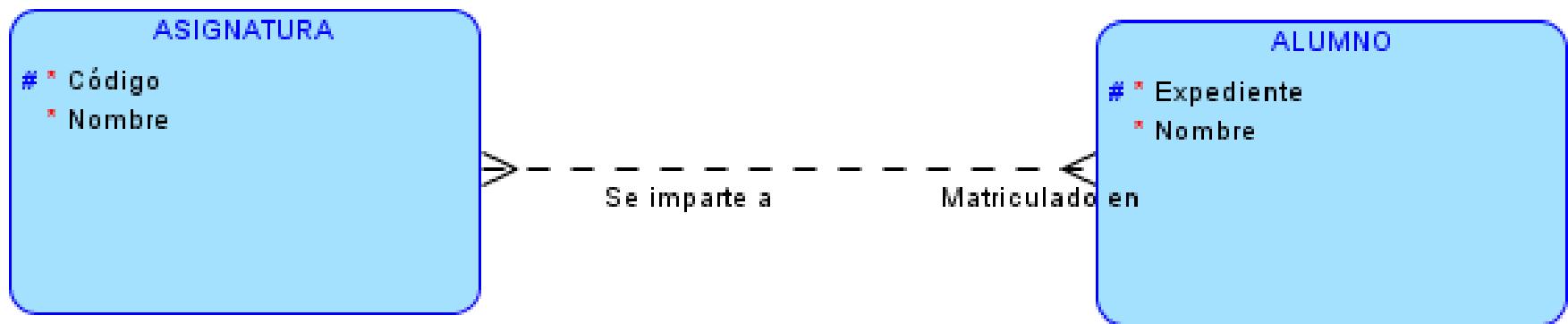


# Entidad débil. Clave prestada



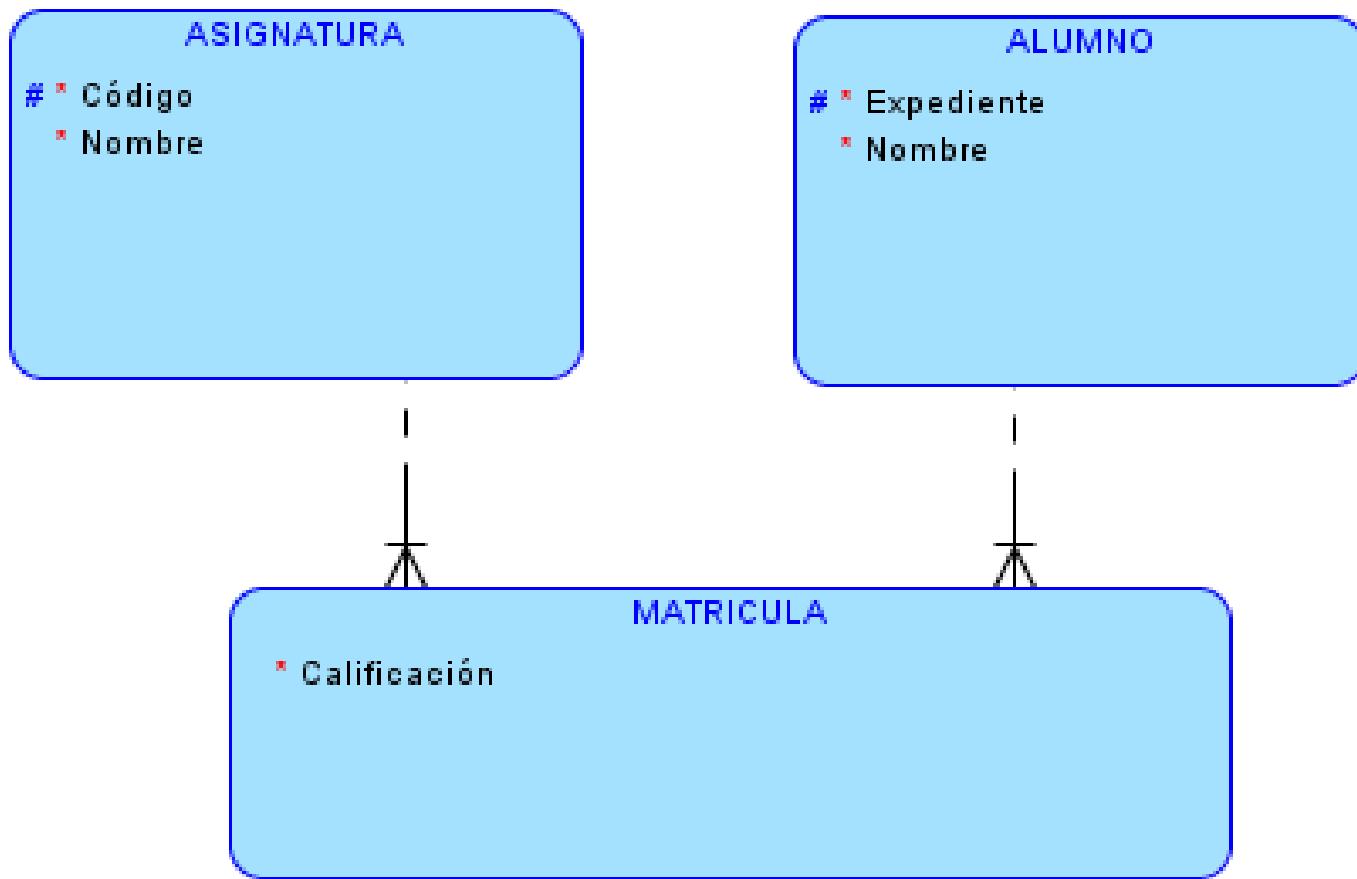
¿cómo se identifica la ciudad?

# Entidad débil y relación M:M

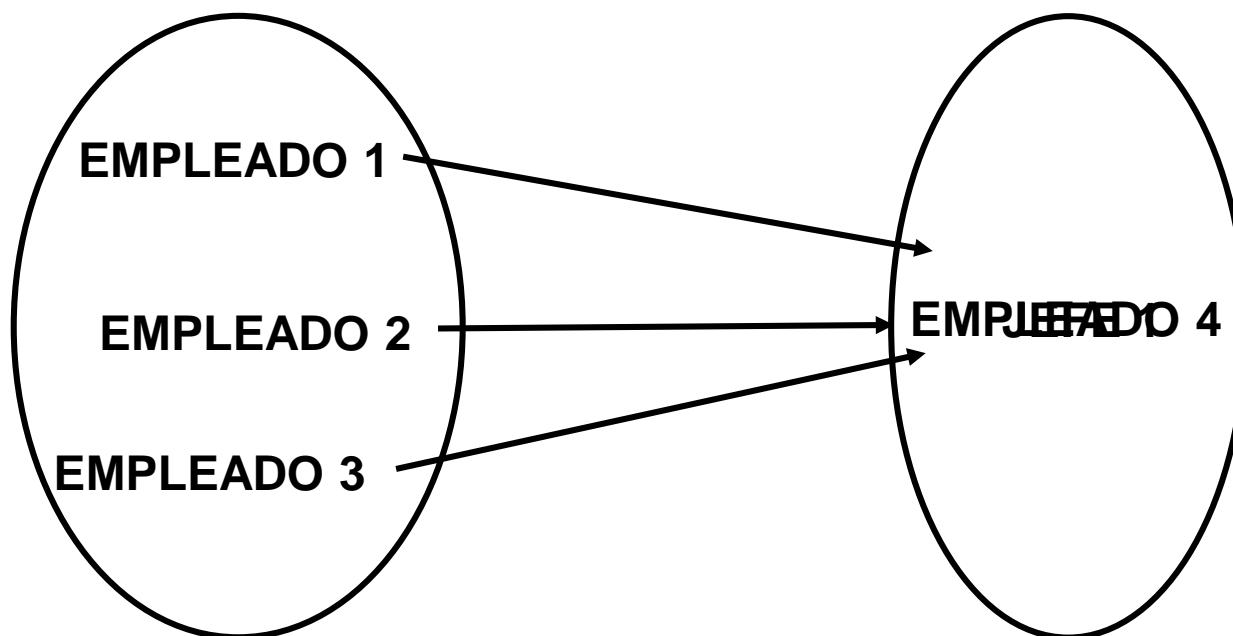


¿A quién pertenece el atributo  
**CALIFICACIÓN**?

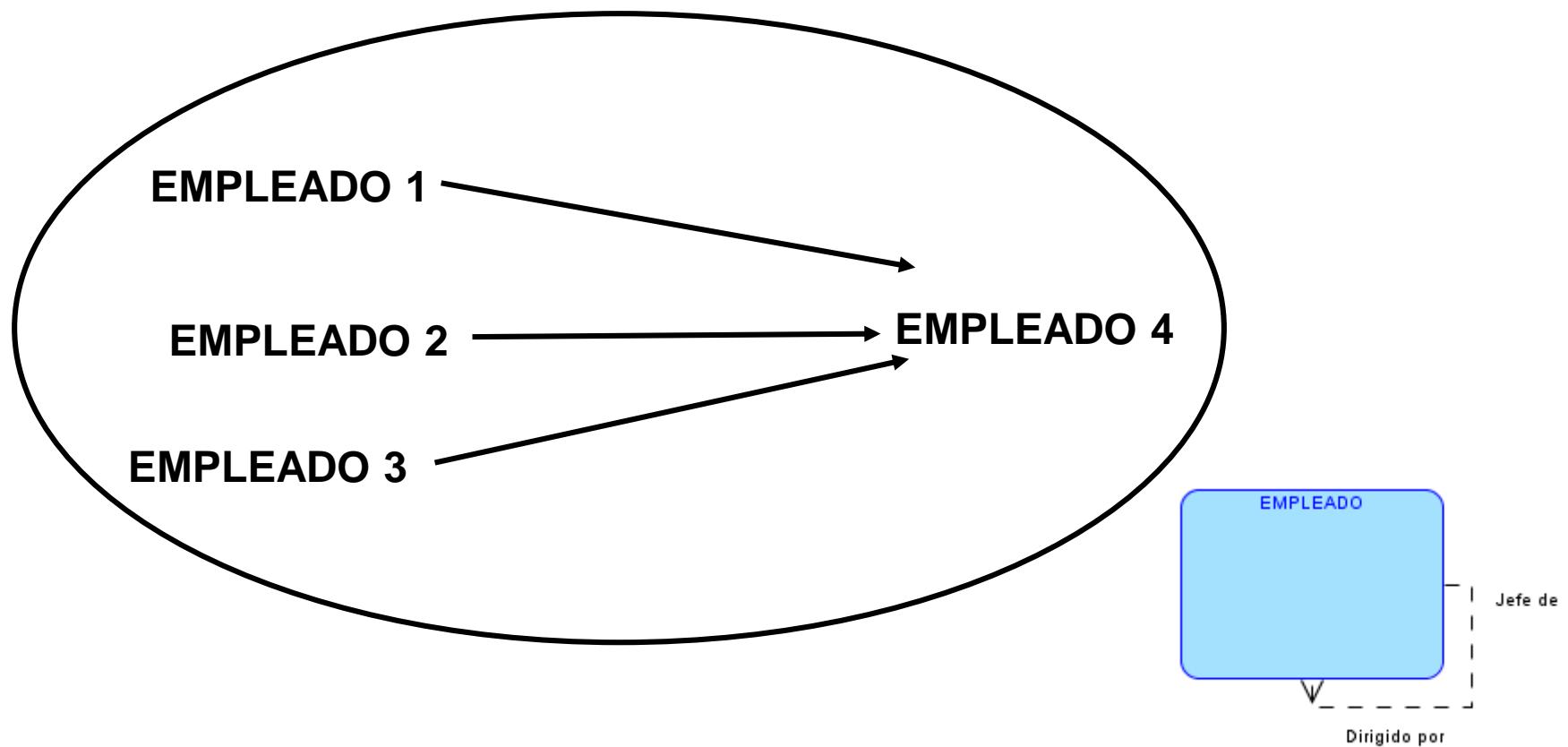
# Entidades débiles y relación M:M



# Relaciones Reflexivas



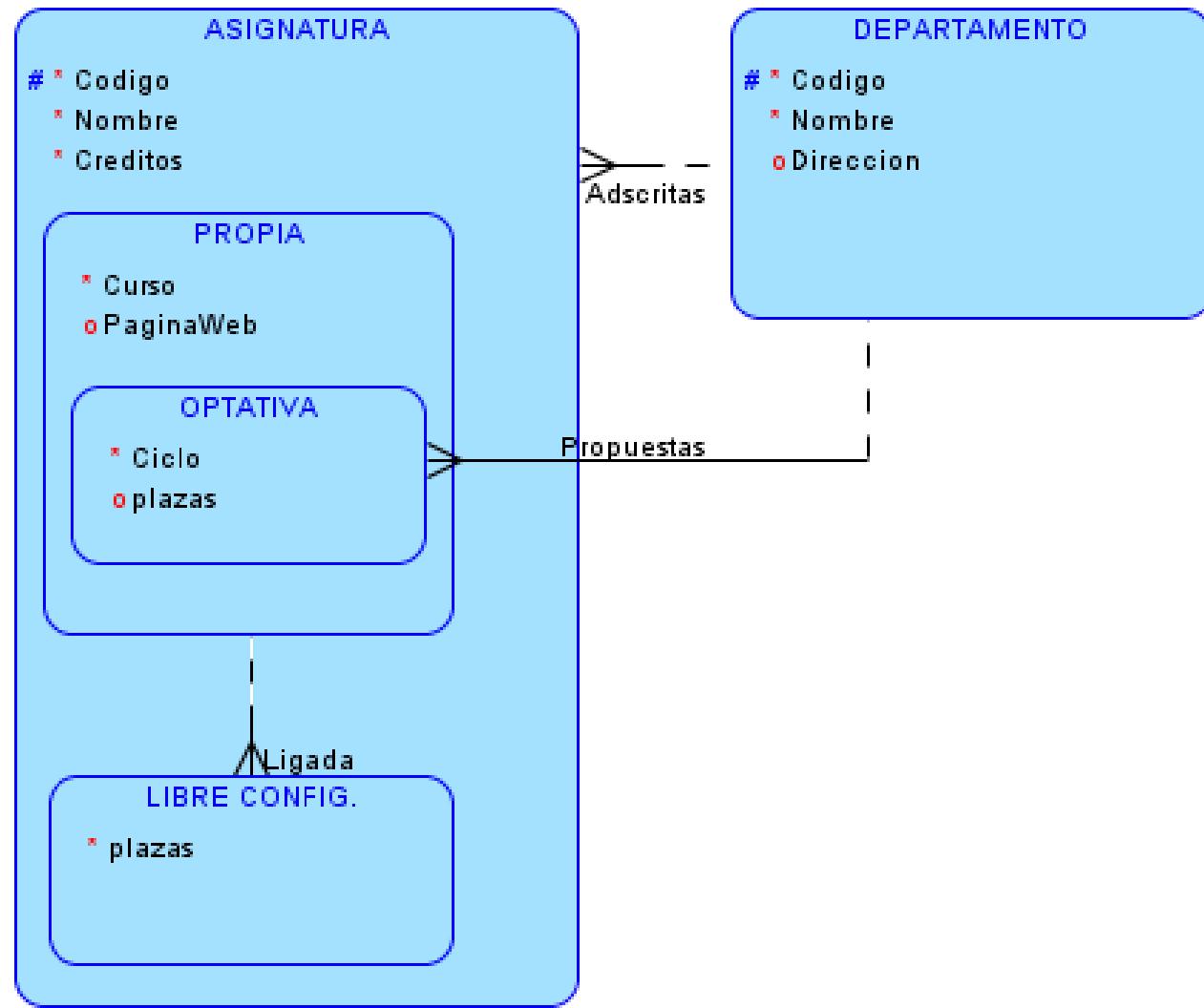
# Relaciones Reflexivas



# Relación Es\_un. Subentidades



# Relación Es\_un. Subentidades



# 1 Introducción

## 1.1 Entorno

### Entorno de trabajo

- Realizar diseño de bases de datos **relacionales**.
- Usar modelo **Entidad/Relación**
- Soporte para DBMS Oracle
- Gratis

### Oracle Data Modeler



- Ingeniería inversa y directa.
- Notación Entidad/Relación: Barker o Bachman.
- Diseño lógico, diseño relacional y diseño físico.
- Soporta tablas, usuarios, objetos de la BBDD, tablespace, etc.
- Otros: Integración de vistas, migración a otros SGBD, etc.

### Enlaces de Interés

- Página Principal: [www.oracle.com/technetwork/developer-tools/datamodeler](http://www.oracle.com/technetwork/developer-tools/datamodeler)
- Descarga con registro (usar buzón de correo UMA).
- Documentación en diferentes formatos.
- Enlace en Campus Virtual.

### Diseño en tres capas

- Modelar la realidad
- Llegando a un producto final

## Diseño Lógico

- Modelo Entidad/Relación.
- Todos los elementos, aún sin mucho detalle.

## Diseño Relacional

- Modelo Relacional
- Especificación de todos los elementos con su implementación relacional

## Diseño Físico

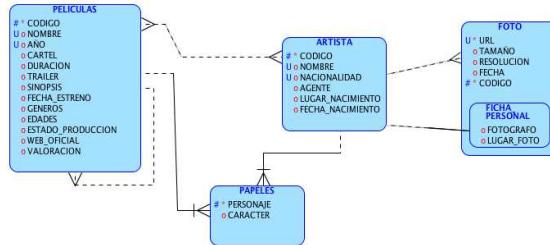
- DBMS concreto
- Detalles de almacenamiento

## 1.2 Ejemplo

### Películas comerciales

- Información de películas.
- Datos de artistas (actores y directores).
- Documentación sobre fotos.

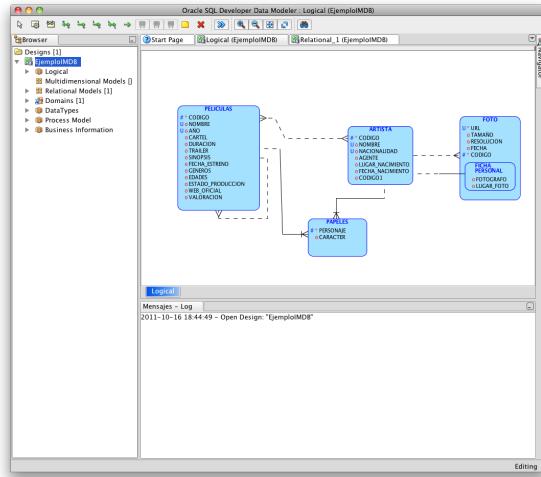
### Películas comerciales



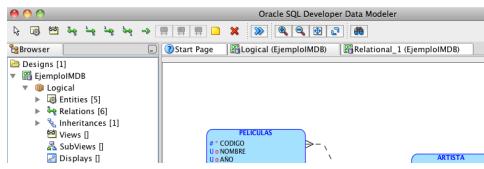
## 2 Data Modeler

### 2.1 Introducción

#### Aspecto general



## Aspecto general



- Pestaña diseño lógico / relacional
- Navegación (Browser).
- Botón de generador

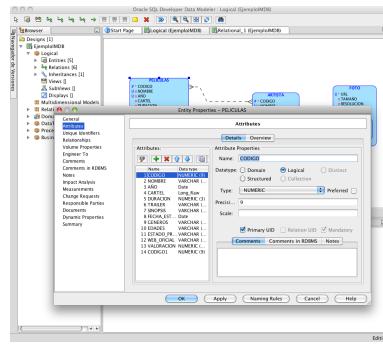
## 2.2 Diseño Lógico

### Elementos



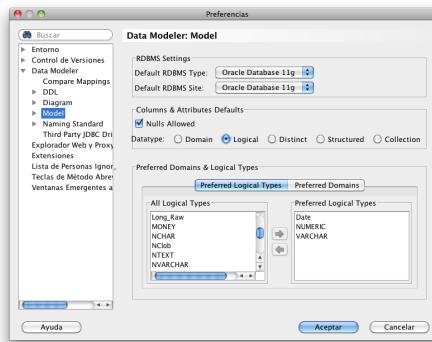
- Entidades
- Relaciones Fuertes (m:m, 1:m, 1:1)
- Relaciones Débiles y Es\_un

## Navegación: Entidades



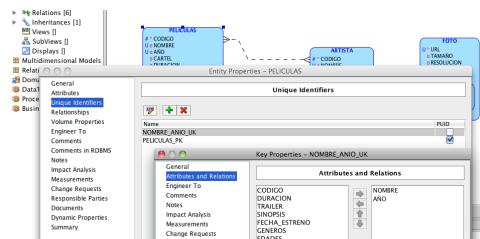
- Atributos, tipos y restricciones (PK, NULL).
- Dominios o tipos base (seleccionar en preferencias).

## Personalizar Tipos de Datos



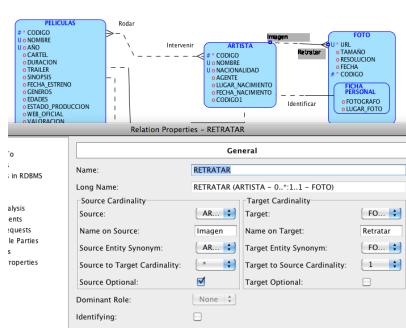
- Herramientas :: Preferencias
- Seleccionamos tipos de datos preferidos y por defecto *Logical*

## Navegación: Claves candidatas



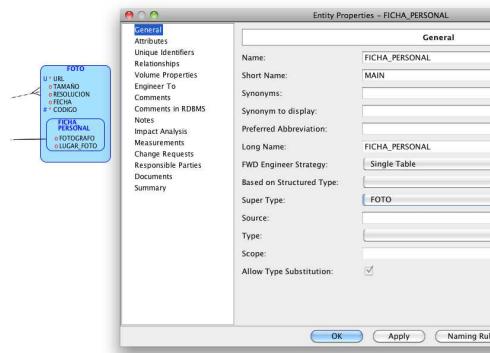
- Simples y compuestas
- Independiente de la obligatoriedad

## Navegación: Relaciones



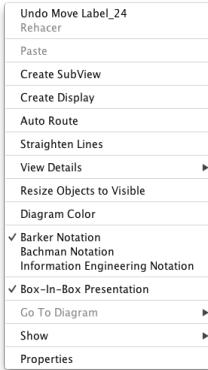
- Orden o Cardinalidad
- Carácter de obligatoriedad
- Debilidad
- Nombres de lectura

## Navegación: Relaciones



- Fuertes y débiles
- Relaciones Es\_Un

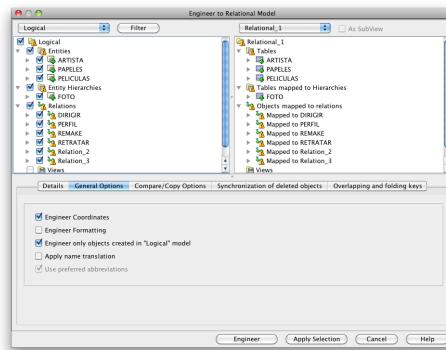
## Opciones vista



- Mover relaciones: Auto Route
- Barker Notation
- Sub-entidades: Box-in-Box

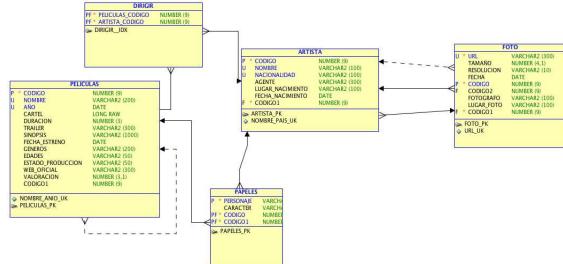
### 2.3 Diseño Relacional

#### Generacion



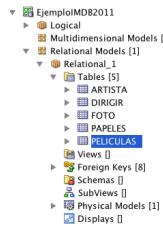
- Tablas, relaciones y atributos.
- Seleccionar objetos del modelo lógico

#### Diagrama



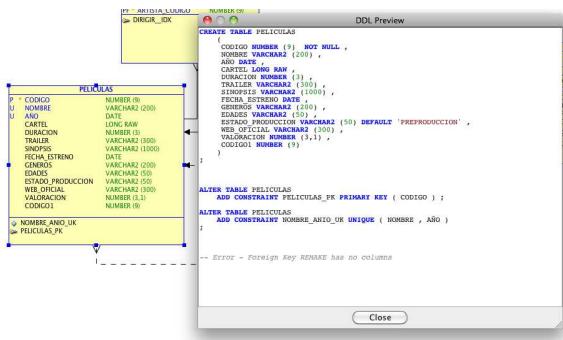
- Relaciones m:m
- Entidades débiles

## Elementos



- Tablas
- Claves Foráneas

## SQL tablas



- Atributos
- Restricciones
- Valores por defecto

## Tema 3. Modelo Relacional

# Bases de Datos Relacionales

Departamento de Lenguajes y  
Ciencias de la Computación

# Concepto intuitivo de tabla

- Es intuitiva la forma de almacenar la información en una base de datos relacional?

| PROVINCIA | CMUN | NOMBRE           | HOMBRES | MUJERES |
|-----------|------|------------------|---------|---------|
| Albacete  | 1    | Abengibre        | 386     | 396     |
| Albacete  | 2    | Alatoz           | 340     | 270     |
| Albacete  | 3    | Albacete         | 84233   | 87888   |
| Albacete  | 4    | Albatana         | 380     | 364     |
| Albacete  | 5    | Alborea          | 394     | 360     |
| Albacete  | 6    | Alcadozo         | 372     | 327     |
| Albacete  | 7    | Alcalá del Júcar | 667     | 583     |
| Albacete  | 8    | Alcaraz          | 774     | 755     |
| Albacete  | 9    | Almansa          | 12359   | 12478   |
| Albacete  | 10   | Alpera           | 1196    | 1130    |
| Albacete  | 11   | Ayna             | 385     | 332     |
| Albacete  | 12   | Balazote         | 1228    | 1150    |
| Albacete  | 14   | Ballestero, El   | 226     | 216     |
| Albacete  | 13   | Balsa de Ves     | 102     | 70      |
| Albacete  | 15   | Barrax           | 979     | 936     |

# Concepto intuitivo de tabla

- Es intuitiva la forma de almacenar la información en una base de datos relacional?

| Código | Operador  | Logo  | Facilidad de uso | Número     | Voz   | MMS   | SMS  | Protocolos | Tipo      | Rid      |
|--------|-----------|---|------------------|------------|---|-------|------|------------|-----------|----------|
| 1      | Movistar  |    | 0,15€            | Movistar   | 0,03€ Movistar<br>0,36 resto                  | 1€    | 0,15 | 18 meses   | Conectado |          |
| 2      | Movistar  |    | 0,15€            | Mi gente   | 0,03€ Movistar<br>0,56 elegidos<br>0,36 resto | 1€    | 0,15 | 18 meses   | Conectado |          |
| 3      | Orange    |    | 0,15€            | Hola       | 0,23€ Orange<br>0,46 resto                    | 0,60€ | 0,15 |            | Tarjeta   |          |
| 4      | Vodafone  |    | 0,15€            | Contrato 1 | 0,01€ Vodafone<br>0,35€ resto                 |       | 0,15 | Si         | Conectado |          |
| 5      | Pepephone |   | 0,15€            | Pulpo Pepe | 0,01€   |       | 0,09 | No         | Conectado | Vodafone |
| 6      | Simyo     |  | 0,15€            | 0/8        | 0,00€ Simyo<br>0,36 resto                     | 0,34€ | 0,10 | No         | Contrato  | Orange   |

**TARIFAS**

| Operador | Fecha Inicio   | Líneas     |
|----------|----------------|------------|
| Movistar | Julio/1995     | 22.808.649 |
| Vodafone | Diciembre/1995 | 15.992.982 |
| Orange   | Enero/1996     | 11.123.755 |

**REDES**

# Conceptos de BD relacionales

- Codd propone el modelo relacional para bases de datos en 1970.
- Ésta es la base para la construcción de los SGBDR.
- El modelo relacional consiste en:
  - Conjunto de objetos o relaciones
  - Conjunto de operadores que actúan sobre los objetos
  - Reglas para asegurar la integridad y consistencia del propio modelo.

# Definición de una Base de Datos Relacional

- Una relación consta de:
  - Un esquema: conjunto de pares (atributo,dominio).
  - Un cuerpo: conjunto de tuplas de pares (atributo,valor).

# Definición de una Base de Datos Relacional

Ejemplo de relación:

Tabla: **DEPARTAMENTOS**

| DEPTNO | DNOMBRE       | DLUGAR  |
|--------|---------------|---------|
| 01     | MANTENIMIENTO | MALAGA  |
| 10     | CUENTAS       | MADRID  |
| 20     | I+D           | MALAGA  |
| 30     | VENTAS        | SEVILLA |
| 40     | MARKETING     | MADRID  |

**ESQUEMA**

**CUERPO**

Esquema de DEPARTAMENTOS

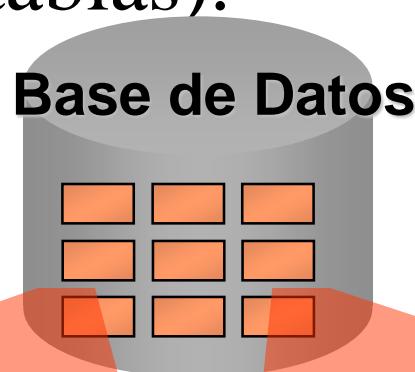
{ (DEPTNO, Número), (DNOMBRE, Cadena), (DLUGAR, Cadena) }

Cuerpo de DEPARTAMENTOS

{[(DEPTNO, 01), (DNOMBRE, MANTENIMIENTO),(DLUGAR,MALAGA)],  
[(DEPTNO, 10), (DNOMBRE,CUENTAS),(DLUGAR,MADRID)], ... }

# Definición de una Base de Datos Relacional

Una base de datos relacional es un conjunto de relaciones (o tablas).



| EMPNO | ENOMBRE | CARGO     | DEPTNO |
|-------|---------|-----------|--------|
| 7839  | KING    | PRESIDENT | 10     |
| 7698  | BLAKE   | ASESOR    | 30     |
| 7782  | CLARK   | ASESOR    | 10     |
| 7566  | JONES   | ASESOR    | 20     |

| DEPTNO | DNOMBRE   | DLUGAR  |
|--------|-----------|---------|
| 10     | CUENTAS   | MADRID  |
| 20     | I+D       | MALAGA  |
| 30     | VENTAS    | SEVILLA |
| 40     | MARKETING | MADRID  |

# Terminología de las BDR

| 1      | 2      | 3           | 4    | 5         | 6     |       |        |
|--------|--------|-------------|------|-----------|-------|-------|--------|
| NUMERO | NOMBRE | PUESTO      | JEFE | ANTIG     | SALAR | COMIS | DEPTNO |
| 7839   | KING   | PRESIDENTE  | 7839 | 17-NOV-81 | 5000  |       | 10     |
| 7698   | BLAKE  | ASESOR      | 7839 | 01-MAY-81 | 2850  |       | 30     |
| 7782   | CLARK  | ASESOR      | 7839 | 09-JUN-81 | 2450  |       | 10     |
| 7566   | JONES  | ASESOR      | 7839 | 02-ABR-81 | 2975  |       | 20     |
| 7654   | MARTIN | JEFE VENTAS | 7698 | 28-SEP-81 | 1250  | 1400  | 30     |
| 7499   | ALLEN  | JEFE VENTAS | 7698 | 20-FEB-81 | 1600  | 300   | 30     |
| 7844   | TURNER | JEFE VENTAS | 7698 | 08-SEP-81 | 1500  | 0     | 30     |
| 7900   | JAMES  | EMPLEADO    | 7698 | 03-DIC-81 | 950   |       | 30     |
| 7521   | WARD   | JEFE VENTAS | 7698 | 22-FEB-81 | 1250  | 500   | 30     |
| 7902   | FORD   | ANALISTA    | 7566 | 03-DIC-81 | 3000  |       | 20     |
| 7369   | SMITH  | EMPLEADO    | 7902 | 17-DIC-80 | 800   |       | 20     |
| 7788   | SCOTT  | ANALISTA    | 7566 | 09-DIC-82 | 3000  |       | 20     |
| 7876   | ADAMS  | EMPLEADO    | 7788 | 12-ENE-83 | 1100  |       | 20     |
| 7934   | MILLER | EMPLEADO    | 7782 | 23-ENE-82 | 1300  |       | 10     |

1. Fila o Tupla
2. Clave Primaria
3. Columna
4. Clave Foránea
5. Dato o Valor
6. Valor Nulo

# Definición de una Base de Datos Relacional

- Los dominios se completan con el valor NULL.
- Claves candidatas. Subconjunto de atributos del cuerpo que permite identificar a cada elemento del cuerpo de forma única.
  - Clave minimal.
  - Clave Primaria. De entre las candidatas se elige una como primaria → Criterios de selección.
- Clave foránea.

## MODELO ORIENTADO A VALORES

# Relaciones. Funcionamiento.

- Cada fila de una tabla se identifica mediante la Clave Primaria (PK).
- Se pueden relacionar datos de varias tablas mediante las Claves Foráneas (FK).

Tabla: **EMPLEADOS**

| NÚMERO | NOMBRE | CARGO     | DEPTNO |
|--------|--------|-----------|--------|
| 7839   | KING   | PRESIDENT | 10     |
| 7698   | BLAKE  | ASRSOR    | 30     |
| 7782   | CLARK  | ASESOR    | 10     |
| 7566   | JONES  | ASESOR    | 20     |



Clave Primaria

Tabla: **DEPARTAMENTOS**

| DEPTNO | NOMBRE    | LUGAR   |
|--------|-----------|---------|
| 10     | CUENTAS   | MADRID  |
| 20     | I + D     | MALAGA  |
| 30     | VENTAS    | SEVILLA |
| 40     | MARKETING | MADRID  |



Clave Foránea



Clave Primaria

# Claves

- ¿Cuántas claves candidatas puede haber en una relación? ¿Una o varias?
- ¿Cuántas primarias?
- ¿Cuántas foráneas?
- Las claves foráneas ¿se establecen sobre las primarias o las candidatas?
- ¿Qué es una clave primaria simple?
- ¿Es mejor una primaria simple?

# Reglas del Modelo Relacional

## **Propiedad de la clave primaria: IDENTIFICAR**

Primera Regla de Integridad (de la *Entidad*):  
Las componentes de una clave primaria no  
pueden ser nulos.

# Reglas del Modelo Relacional

Clave Foránea: relaciones entre tablas

**Segunda Regla de Integridad (de *Referencia*):**  
**Las componentes de una clave foránea son nulas o son iguales que el valor de alguna una primaria en una tabla del modelo.**

# Redundancia

- Un dato A es deducible de otros B, C, D, ..., si conocidos los valores de B, C, D, ... queda determinado el valor de A.
- Si además de tener almacenados B, C, D, ... tenemos también almacenado A, estamos ante un caso de redundancia

# Efectos de la redundancia

- Mayor espacio de almacenamiento. Menos importante
- Posible incoherencia. Muy importante

# Dependencia Funcional

- Entre dos atributos X e Y de una relación R hay una Dependencia Funcional (DF) si siempre que dos tuplas de R coincidan en sus valores de X, coinciden también en sus valores de Y

## Ejemplo

| Operador  | Líneas   | Red      | Líneas Red |
|-----------|----------|----------|------------|
| Movistar  | 22808649 | Movistar | 22808649   |
| Orange    | 15992461 | Orange   | 15992461   |
| Vodafone  | 11123755 | Vodafone | 11123755   |
| Pepephone | 32600    | Vodafone | 11123755   |
| Simyo     | 32600    | Orange   | 15992461   |

# Dependencia Funcional

- Operador → Líneas
- Red → Líneas Red

¿Hay redundancia en esta tabla?

## Ejemplo

| Operador  | Líneas   | Red      | Líneas Red |
|-----------|----------|----------|------------|
| Movistar  | 22808649 | Movistar | 22808649   |
| Orange    | 15992461 | Orange   | 15992461   |
| Vodafone  | 11123755 | Vodafone | 11123755   |
| Pepephone | 32600    | Vodafone | 11123755   |
| Simyo     | 32600    | Orange   | 15992461   |

# Operaciones de Conjuntos

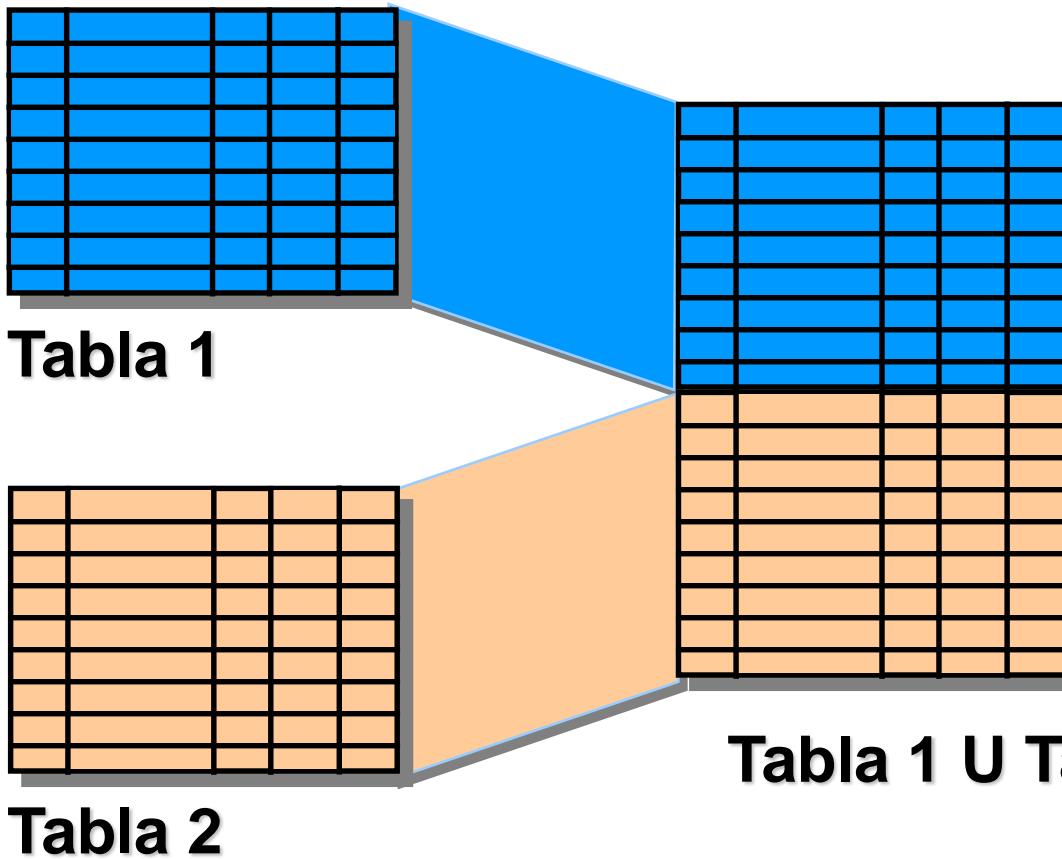
- **Unión Compatibles:**

- Misma aridad (*Mismo número de columnas*)
- Isomorfismo entre esquemas basado en igualdad de dominios (*Columnas del mismo tipo, o tipo compatible*)
  - **Unión.**
  - **Intersección.**
  - **Diferencia.**

- **Libres:**

- Siempre se puede realizar
  - **Producto Cartesiano.**

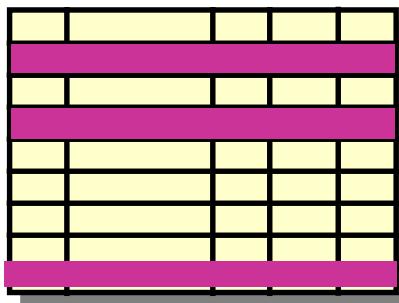
# Operaciones relacionales. Operaciones de Conjuntos Unión



Cuidado: Los elementos que se encuentran en las 2 tablas no se repiten en la unión

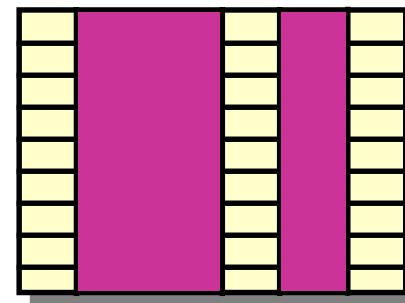
# Operaciones relacionales

# Selección



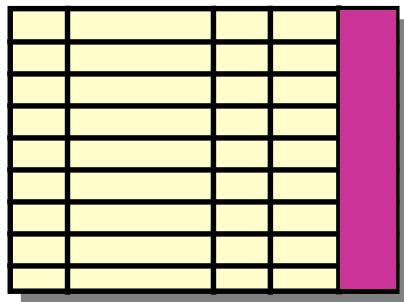
**Tabla 1**

# Proyección

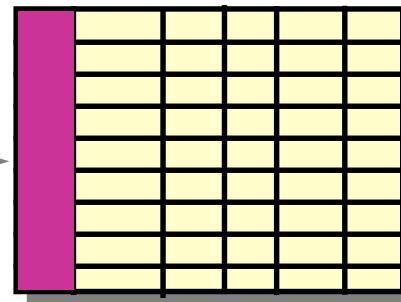


**Tabla 1**

## Reunión (join)



**Tabla 1**



## Tabla 2

# Álgebra relacional

- Operaciones de Conjuntos:
  - $R \cup S, R \cap S, R \times S, R - S.$
- Selección:  $\sigma_{\text{Predicado}} R$
- Proyección:  $\Pi_{\text{Atributos}} R$
- Reunión:  $R * S = \Pi_{\Omega \setminus FK} (\sigma_{FK=PK} R \times S)$
- Renombramiento:  $\rho(R)$

# Ejemplo de Algebra Relacional

¿Cual es el operador con mas líneas?

OP

| Operador  | Líneas   |
|-----------|----------|
| Movistar  | 22808649 |
| Orange    | 15992461 |
| Vodafone  | 11123755 |
| Pepephone | 32600    |
| Simyo     | 31880    |

# Ejemplo de Algebra Relacional

¿Cual es el operador con mas líneas?

$$OP \times \rho(OP)$$

| Operador  | Líneas   | Operador | Líneas   |
|-----------|----------|----------|----------|
| Movistar  | 22808649 | Movistar | 22808649 |
| Orange    | 15992461 | Movistar | 22808649 |
| Vodafone  | 11123755 | Movistar | 22808649 |
| Pepephone | 32600    | Movistar | 22808649 |
| Simyo     | 31880    | Movistar | 22808649 |
| Movistar  | 22808649 | Orange   | 15992461 |
| Orange    | 15992461 | Orange   | 15992461 |
| Vodafone  | 11123755 | Orange   | 15992461 |
| ...       |          |          |          |

# Ejemplo de Algebra Relacional

¿Cual es el operador con mas líneas?

$$\sigma_{\text{Líneas} < \rho(\text{Líneas})}(\text{OP} \times \rho(\text{OP}))$$

| Operador  | Líneas   | Operador | Líneas   |
|-----------|----------|----------|----------|
| Orange    | 15992461 | Movistar | 22808649 |
| Vodafone  | 11123755 | Movistar | 22808649 |
| Pepephone | 32600    | Movistar | 22808649 |
| Simyo     | 31880    | Movistar | 22808649 |
| Vodafone  | 11123755 | Orange   | 15992461 |
| Pepephone | 32600    | Orange   | 15992461 |
| Simyo     | 31880    | Orange   | 15992461 |
| ...       |          |          |          |

# Ejemplo de Algebra Relacional

¿Cual es el operador con mas líneas?

$$\Pi_{\text{Operador}} \sigma_{\text{Líneas} < \rho(\text{Líneas})} (\text{OP} \times \rho(\text{OP}))$$

| Operador  |
|-----------|
| Orange    |
| Vodafone  |
| Pepephone |
| Simyo     |
| Vodafone  |
| Pepephone |
| Simyo     |
| ...       |

# Ejemplo de Algebra Relacional

¿Cual es el operador con mas líneas?

$$\Pi_{\text{Operador}} \text{OP} - \Pi_{\text{Operador}} \sigma_{\text{Líneas} < \rho(\text{Líneas})} (\text{OP} \times \rho(\text{OP}))$$

| Operador |
|----------|
| Movistar |

# Reglas de Codd

## 1. Regla de Información

Toda la información se presenta mediante tablas y sólo mediante tablas

## 2. Regla de acceso garantizado:

Se accede sólo por nombre de columna y valor de clave candidata

## 3. Manejo sistemático de valores nulos

Se debe disponer de una representación de valores desconocidos y no aplicables diferente de los valores normales

# Reglas de Codd

## 4. Catálogo activo en línea basado en el modelo relacional

Debe estar a disposición de los usuarios con el mismo lenguaje de consulta que la base de datos.

## 5. Sublenguaje de datos completo:

- o Sintaxis lineal
- o Utilización interactiva y mediante lenguajes de programación
- o Definición de datos, manipulación completa de datos
- o Restricciones de seguridad, integridad y manejo de transacciones

# Reglas de Codd

## 6. Actualización de vistas

Se deben poder actualizar todas las vistas que en teoría se puedan actualizar

## 7. Inserción, modificación y borrado de alto nivel:

Se debe modificar, insertar y borrar todo un conjunto de tuplas a la vez

## 8. Independencia física de los datos

## 9. Independencia lógica de los datos

# Reglas de Codd

## 10. Independencia de integridad

La integridad no forma parte de los programas de aplicación sino del esquema conceptual

## 11. Independencia a la distribución

## 12. No subversión

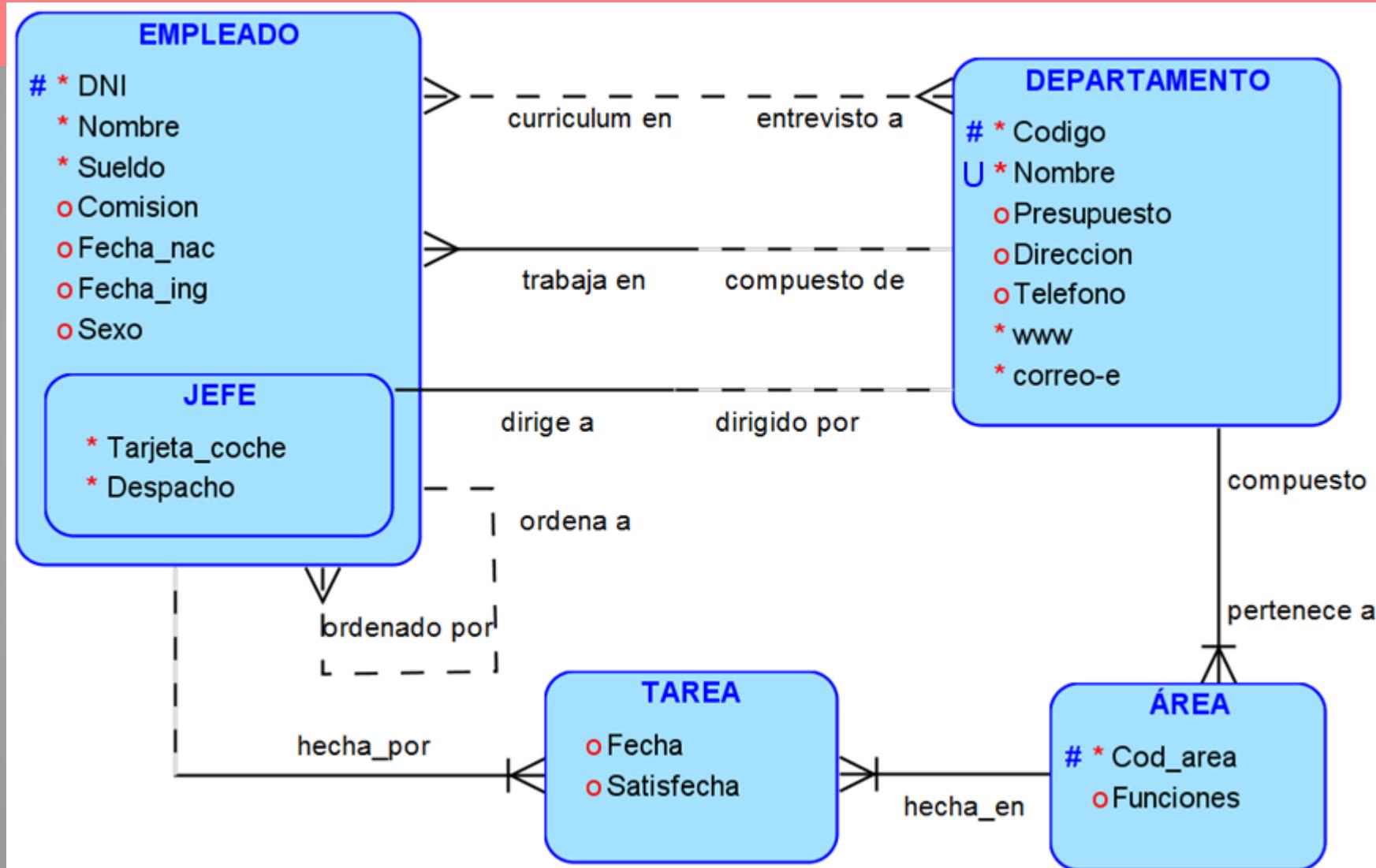
Si el sistema ofrece una interfaz de acceso a bajo nivel, dicho acceso no se podrá usar para salvar restricciones de integridad o seguridad

# Equivalencia entre el Modelo E/R y el Modelo Relacional (Aproximación Data Modeler)

Generación de un modelo de bases de datos  
relacionales a partir del modelo E/R



# Ejemplo: Departamentos y Empleados



- No hay una única equivalencia. Veremos la transformación automática que hace **Oracle Data Modeler**
- Convertir las entidades es fácil
- Convertir las relaciones y las restricciones es lo más “creativo”
- El esquema conceptual sólo no es “funcional”:
  - Los índices son fundamentales (esquema interno), aunque Data Modeler no crea la mayoría de ellos

# Entidad Regular



- Por cada Entidad Regular (también llamada Fuerte o Maestra) se crea una Tabla
- Se crean las claves:
  - Primarias: PRIMARY KEY
  - Candidatas: NOT NULL y UNIQUE
- Se crean los atributos con sus tipos. Los que sean obligatorios serán NOT NULL

# Entidad Regular

- Si se desea, en este momento pueden especificarse las restricciones sobre atributos y sobre tabla: CONSTRAINT
- Los índices de las claves (PRIMARY KEY y UNIQUE) los crea Oracle automáticamente
- Deben crearse índices para los atributos por los que se vayan a realizar búsquedas: CREATE INDEX

# Entidad Regular

```
CREATE TABLE EMPLEADO (
    DNI NUMBER(9) PRIMARY KEY,          #
    Nombre VARCHAR2(15) NOT NULL,        *
    Sueldo NUMBER(5) NOT NULL,
    Comision NUMBER(5),
    Fecha_Nacimiento DATE,
    Fecha_Ingreso DATE,
    Sexo CHAR(1),
    CONSTRAINT sexo_valido CHECK (Sexo IN ('H','M'))
);
```

```
CREATE INDEX Emp_Nombre_Idx ON Empleado(Nombre);
```

# Entidad Regular

Si se quiere, las restricciones pueden añadirse en sentencias separadas

```
CREATE TABLE DEPARTAMENTO(
```

Codigo NUMBER PRIMARY KEY, ← #1

Nombre VARCHAR2(25) NOT NULL,

Presupuesto NUMBER(7),

Direccion VARCHAR2(35),

Telefono VARCHAR2(9),

www VARCHAR2(90) NOT NULL,

Correo-e VARCHAR2(30) NOT NULL

```
);
```

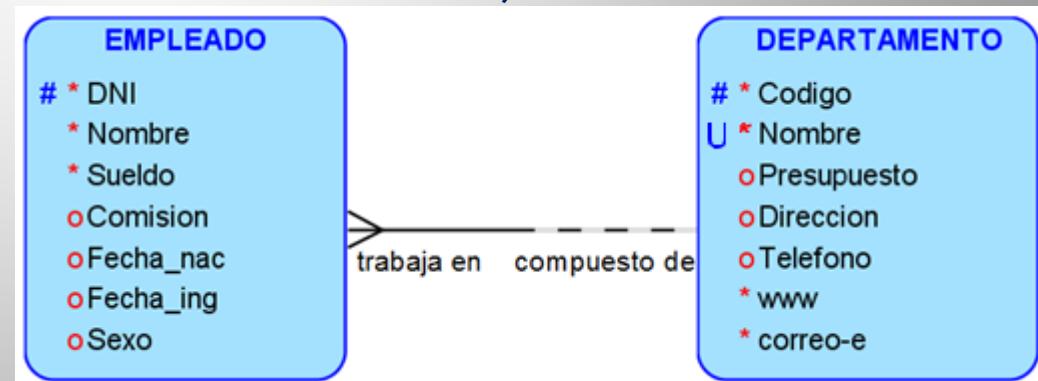
```
ALTER TABLE Departamento ADD CONSTRAINT UK_NOMBRE UNIQUE (Nombre);
```

~~```
CREATE INDEX Dpt_Nombre_Idx ON Departamento(Nombre);
```~~

#2\*

## Relación 1:M

- Las Relaciones de Uno a Muchos entre 2 entidades  $A$  y  $B$  se implementan en la tabla generada para la entidad  $B$ , creando un atributo nuevo que referencia a la clave de la otra Entidad  $A$  (FOREIGN KEY)
- Si es obligatoria en el lado  $B$ , el campo creado debe ser NOT NULL para obligar a que exista la referencia
- Si es obligatoria en el lado de  $A$  se puede hacer mediante programación de bases de datos (*triggers*, proc. almacenados, etc.)
- Las relaciones obligatorias por el lado  $A$  no son frecuentes

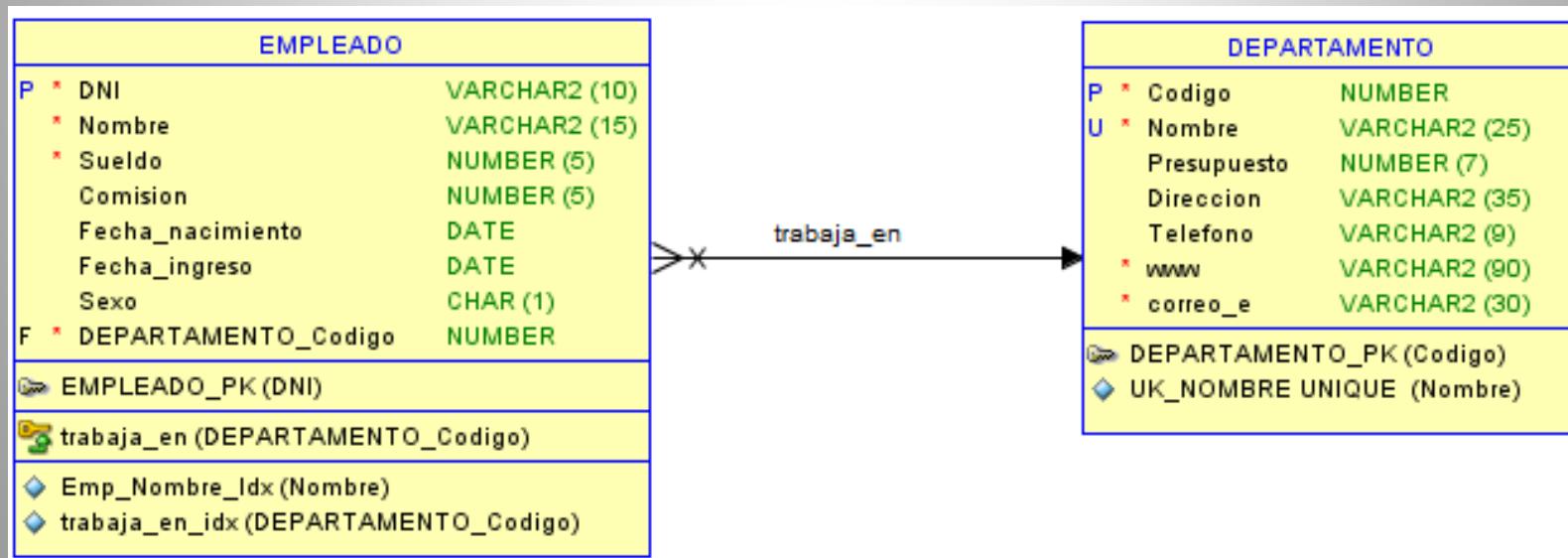


## Relación 1:M

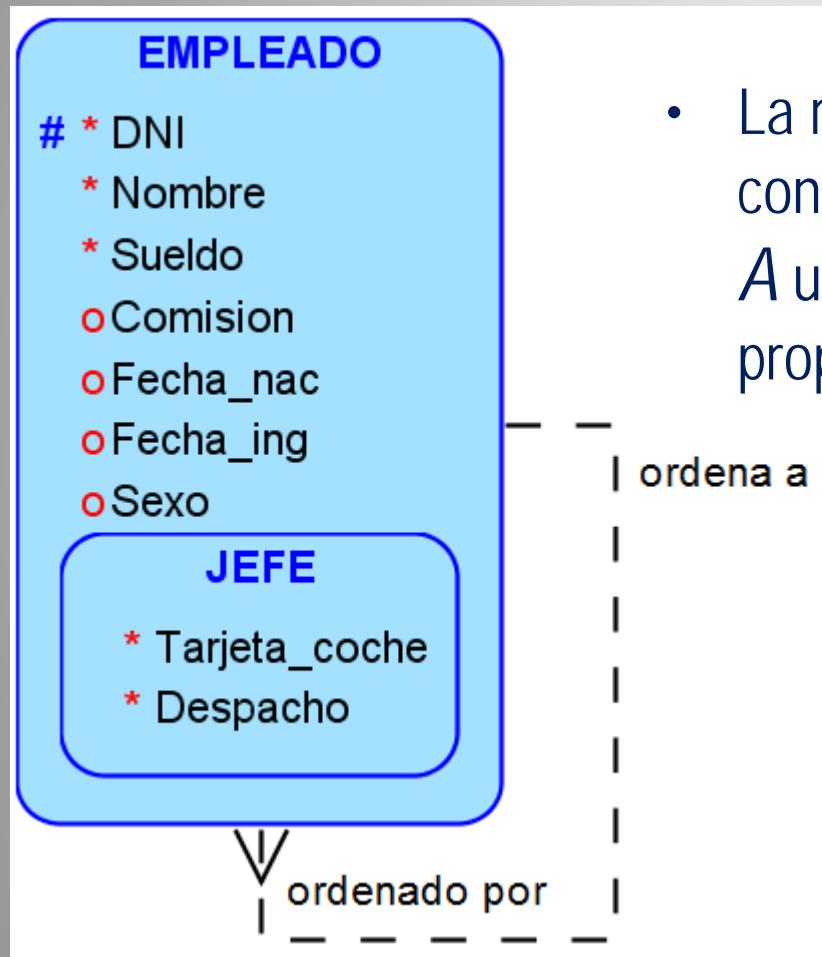
ALTER TABLE Empleado ADD (departamento\_codigo NUMBER NOT NULL);

ALTER TABLE Empleado ADD CONSTRAINT trabaja\_en FOREIGN KEY (departamento\_codigo) REFERENCES Departamento (codigo);

CREATE INDEX trabaja\_en\_idx ON Empleado (departamento\_codigo);



## Relación 1:M Reflexiva



- La relación reflexiva de una Entidad *A* consigo misma se implementa añadiéndole a *A* un campo que sea una referencia a su propia clave primaria.

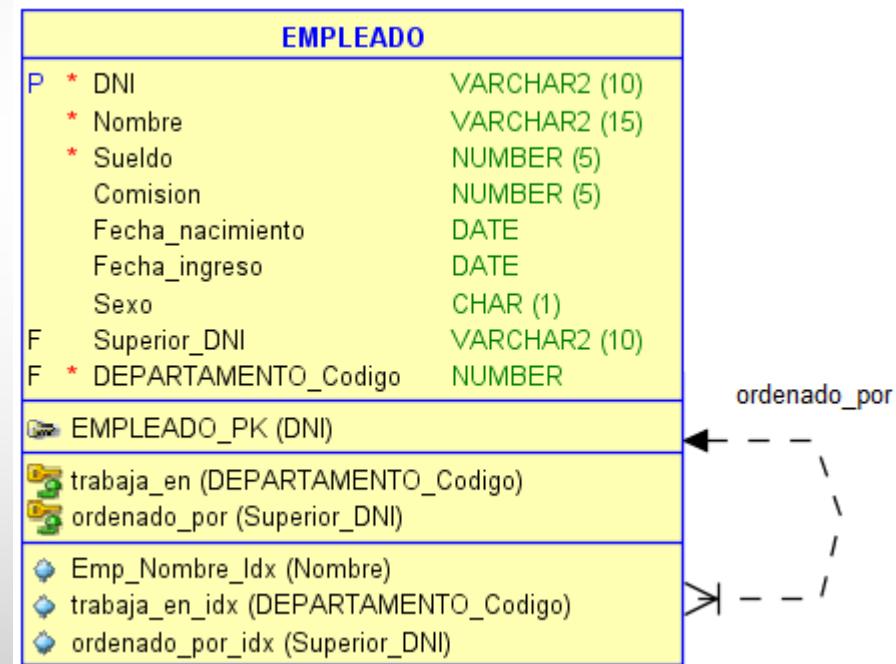
## Relación 1:M Reflexiva

ALTER TABLE Empleado ADD (superior\_DNI NUMBER);

ALTER TABLE Empleado ADD CONSTRAINT ordenado\_por FOREIGN KEY (superior\_DNI) REFERENCES Empleado (DNI);

CREATE INDEX ordenado\_por\_idx

ON Empleado (superior\_DNI);



## Ejemplo de datos en las tablas

### Departamento

| Codigo | Nombre       | Presupuesto | ... |
|--------|--------------|-------------|-----|
| ING    | Ingeniería   | 12000,00    | ... |
| FNZ    | Contabilidad | 7500,50     | ... |

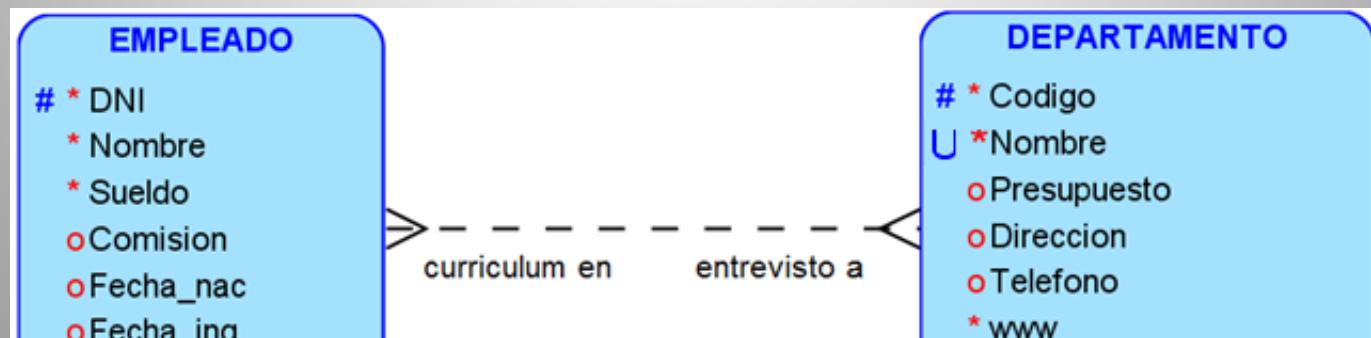


### Empleado

| DNI       | Nombre       | Sueldo   | ... | Sexo | DEPARTAMENTO_codigo | superior_DNI |
|-----------|--------------|----------|-----|------|---------------------|--------------|
| 33387199W | Eduardo Gris | 30000,00 | ... | H    | ING                 | 77623977J    |
| 24887120L | Pedro Rojas  | 28675,23 | ... | H    | ING                 |              |
| 77623977J | Ana Negrín   | 35098,44 | ... | M    | FNZ                 |              |

## Relación M:M

- Una Relación R de muchos a muchos (M:M) entre 2 entidades  $A$  y  $B$  se implementa creando una tabla nueva  $T_R$
- $T_R$  tendrá una referencia obligatoria a cada una de las tablas generadas para las entidades  $A$  y  $B$
- La yuxtaposición de las claves de  $A$  y  $B$  debe ser PRIMARY KEY
- Suele ser necesario crear un índice por cada clave foránea



# Relación M:M

`CREATE TABLE Entrevista (`

`departamento_codigo NUMBER NOT NULL REFERENCES Departamento (codigo),  
empleado_DNI NUMBER NOT NULL REFERENCES Empleado (DNI),  
PRIMARY KEY (departamento_codigo, empleado_DNI)`

`) ;`

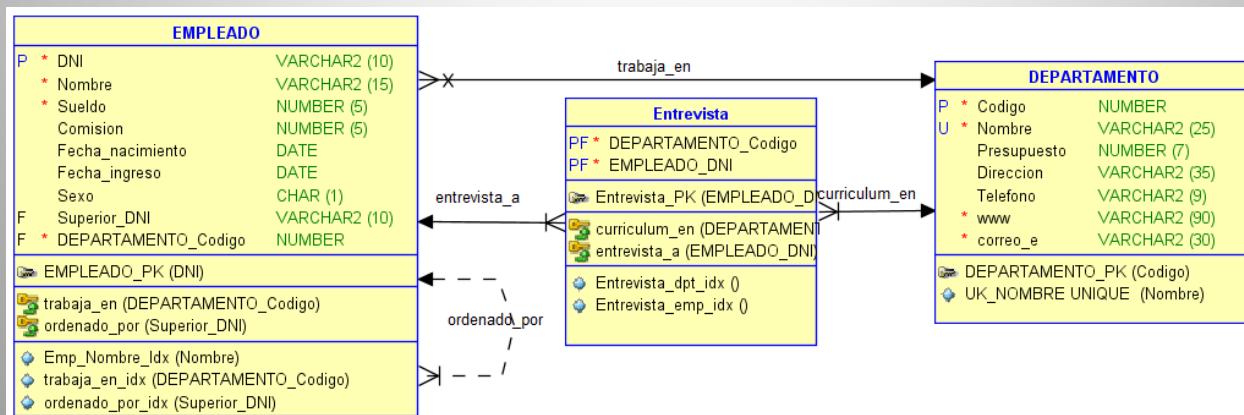
índice automático  
yuxtapuesto

relaciona

~~`CREATE INDEX Entrevista_dpt_idx ON Entrevista(departamento_Codigo);`~~

~~`CREATE INDEX Entrevista_emp_idx ON Entrevista (empleado_DNI);`~~

búsquedas



## Relación M:M

- El índice por departamento\_codigo facilita saber qué empleados se han entrevistado en un departamento. Este índice se crea implícitamente por la PRIMARY KEY
- El índice por empleado\_DNI facilita saber en qué departamentos se ha entrevistado un empleado
- La PRIMARY KEY asegura que un empleado sólo pueda entrevistarse una vez en un mismo departamento. Con poner UNIQUE NOT NULL podría haber sido suficiente pero Data Modeler lo pone como PK

## Relación M:M

### Departamento

| Codigo | Nombre       | Presupuesto | ... |
|--------|--------------|-------------|-----|
| ING    | Ingeniería   | 12000,00    | ... |
| FNZ    | Contabilidad | 7500,50     | ... |

### Empleado

| DNI       | Nombre       | Sueldo   | ... | Sexo | departamento_Codigo | superior_DNI |
|-----------|--------------|----------|-----|------|---------------------|--------------|
| 33387199W | Eduardo Gris | 30000,00 | ... | H    | ING                 | 77623977J    |
| 24887120L | Pedro Rojas  | 28675,23 | ... | H    | ING                 |              |
| 77623977J | Ana Negrín   | 35098,44 | ... | M    | FNZ                 |              |

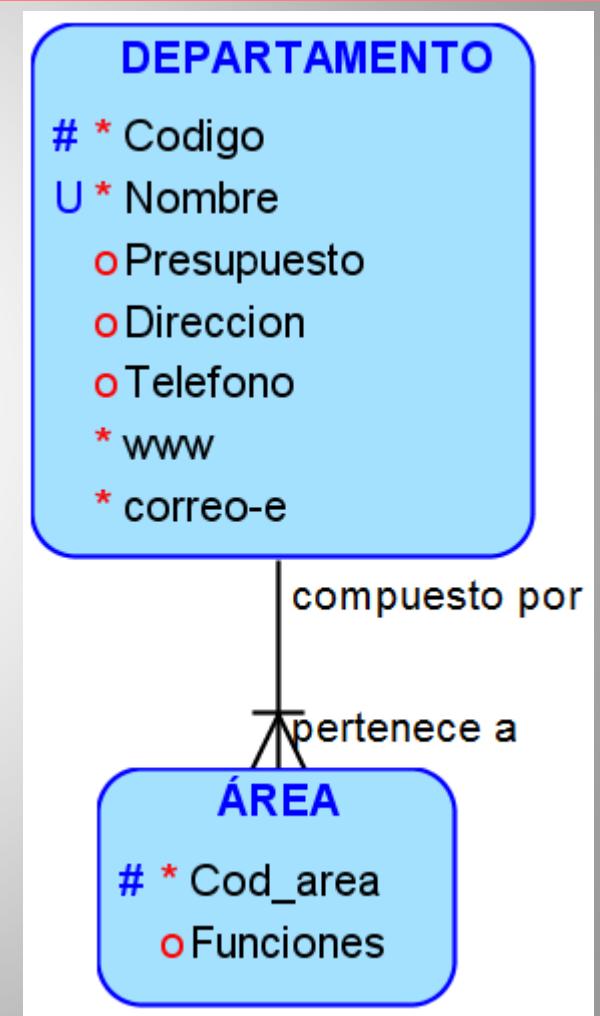
### Entrevista

| departamento_Codigo | empleado_DNI |
|---------------------|--------------|
| ING                 | 33387199W    |
| FNZ                 | 77623977J    |
| FNZ                 | 33387199W    |

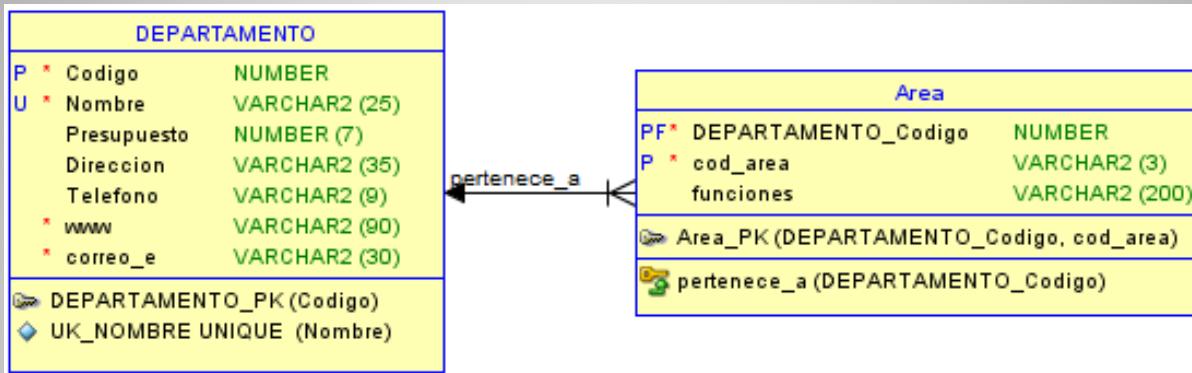
- Eduardo se ha entrevistado en Ingeniería y en Contabilidad
- Ana se ha entrevistado en Contabilidad

# Entidad débil

- Una entidad débil  $D$  de una regular  $R$  se implementa incluyendo una referencia no nula en  $D$  hacia  $R$
- Dado que  $D$  es una entidad se necesita una clave primaria (puede participar en relaciones)
- Data Modeler le asocia como PK una clave compuesta que incluye la que hereda de su entidad maestra



# Entidad débil



```
CREATE TABLE Area (
    departamento_Codigo NUMBER REFERENCES Departamento (codigo),
    cod_area      VARCHAR2 (3),
    funciones     VARCHAR2 (200),
    PRIMARY KEY (departamento_codigo, cod_area)
);
```

índice automático  
yuxtapuesto

- Departamento\_Codigo es **clave foránea no nula** (porque la entidad débil depende del Departamento)
- Cod\_area es **no nula** porque forma parte de una clave (compuesta)

# Entidad débil

## Departamento

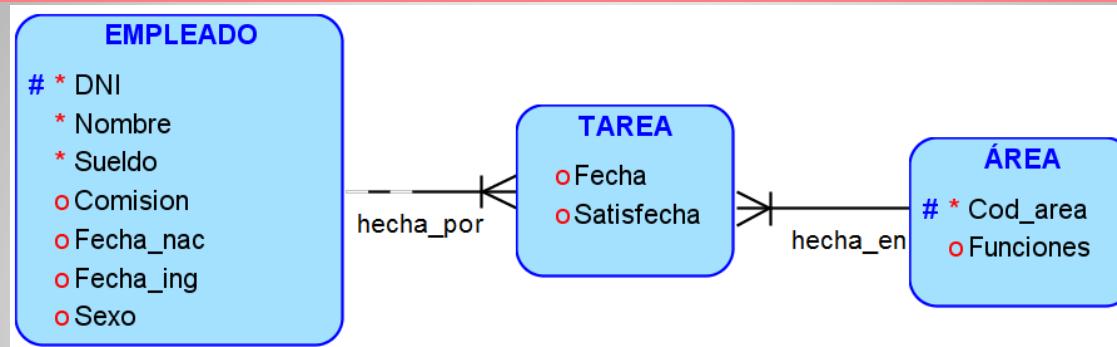
| Código | Nombre       | Presupuesto | ... |
|--------|--------------|-------------|-----|
| ING    | Ingeniería   | 12000,00    | ... |
| FNZ    | Contabilidad | 7500,50     | ... |

## Área

| departamento_Código | cod_area | funciones              |
|---------------------|----------|------------------------|
| ING                 | PRG      | Programación integral  |
| ING                 | PRL      | Personal cualificado   |
| FNZ                 | HBL      | Habilitación           |
| ING                 | HDW      | Hardware y rendimiento |
| FNZ                 | PRL      | Personal               |

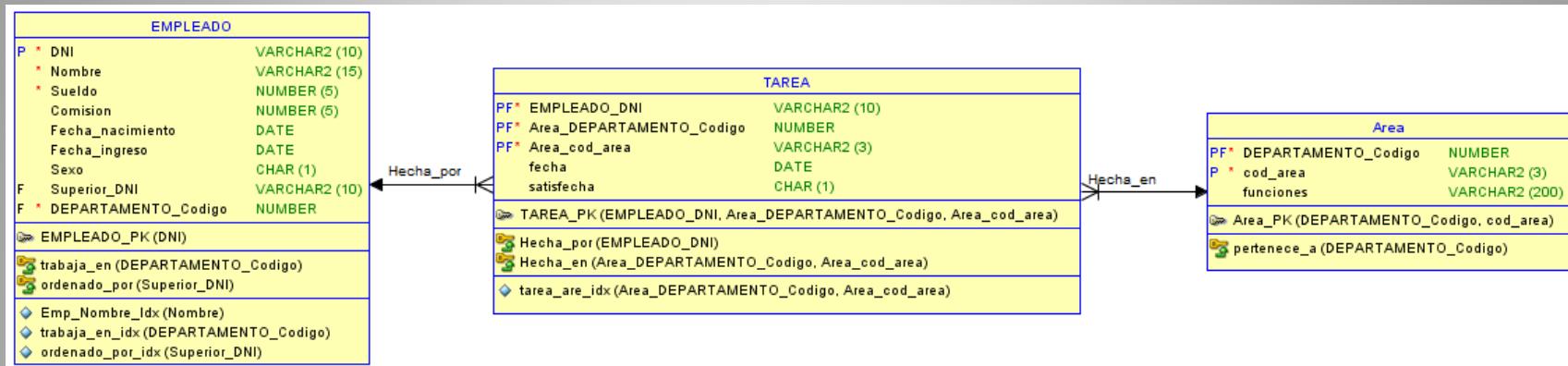
No hay problema en que se repita el código del área

## Entidad débil (m:m)



- Las Relaciones M:M entre 2 relaciones *A* y *B* que contiene atributos generan una tabla  $T_R$  con referencia a las claves primarias de *A* y *B*
- Data Modeler hace que la clave primaria sea la yuxtaposición de las foráneas
- Suele ser necesario crear un índice por cada clave foránea

# Entidad débil (m:m)



```
CREATE TABLE TAREA (
    EMPLEADO_DNI           VARCHAR2 (10) NOT NULL ,
    Area_DEPARTAMENTO_Codigo NUMBER NOT NULL ,
    Area_cod_area           VARCHAR2 (3) NOT NULL ,
    fecha                   DATE ,
    satisfecha              CHAR (1)
);
```

## Entidad débil (m:m)

```
ALTER TABLE TAREA ADD CONSTRAINT TAREA_PK PRIMARY KEY  
( EMPLEADO_DNI, Area_DEPARTAMENTO_Codigo, Area_cod_area ) ;
```



índice automático  
yuxtapuesto

```
ALTER TABLE TAREA ADD CONSTRAINT Hecha_en FOREIGN KEY (Area_DEPARTAMENTO_Codigo, Area_cod_area ) REFERENCES Area (DEPARTAMENTO_Codigo, cod_area) ;
```

```
ALTER TABLE TAREA ADD CONSTRAINT Hecha_por FOREIGN KEY (EMPLEADO_DNI ) REFERENCES EMPLEADO ( DNI ) ;
```

```
CREATE INDEX tarea_are_idx ON TAREA (Area_DEPARTAMENTO_Codigo ASC, Area_cod_area ASC) ;
```

~~CREATE INDEX tarea\_emp\_idx ON TAREA (EMPLEADO\_DNI ASC);~~

# Entidad débil (m:m)

## Empleado

| DNI       | Nombre       | Sueldo   | ... | Sexo | departamento_codigo | superior_DNI |
|-----------|--------------|----------|-----|------|---------------------|--------------|
| 33387199W | Eduardo Gris | 30000,00 | ... | H    | ING                 | 77623977J    |
| 24887120L | Pedro Rojas  | 28675,23 | ... | H    | ING                 |              |
| 77623977J | Ana Negrín   | 35098,44 | ... | M    | FNZ                 |              |

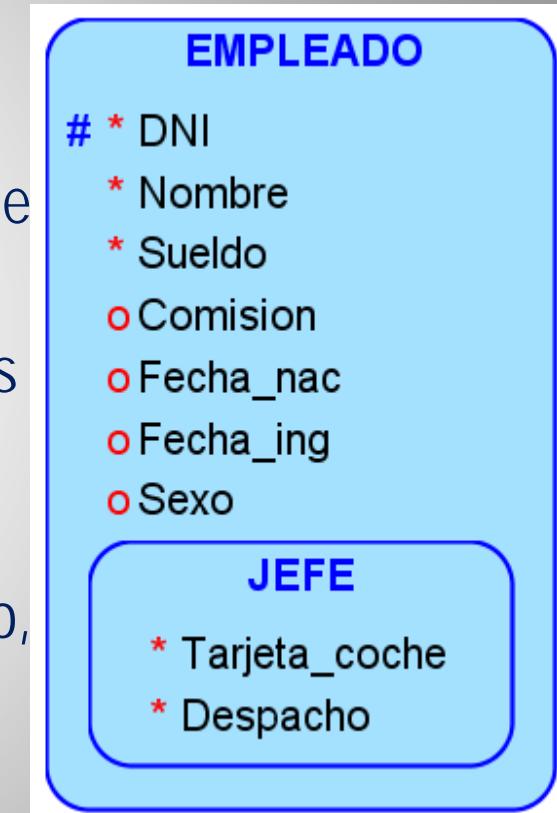
## Tarea

## Área

| departamento_codigo | cod_area | funciones              | empleado_dni | area_departamento_codigo | Area_cod_area | fecha   | satisficha |
|---------------------|----------|------------------------|--------------|--------------------------|---------------|---------|------------|
| ING                 | PRG      | Programación           | 33387199W    | ING                      | PRG           | 3/3/16  | S          |
| ING                 | PRL      | Personal cualificado   | 33387199W    | ING                      | PRL           | 3/9/16  | S          |
| FNZ                 | HBL      | Habilitación           | 24887120L    | FNZ                      | HBL           | 12/5/16 | S          |
| ING                 | HDW      | Hardware y rendimiento | 77623977J    | FNZ                      | HDW           | 3/9/16  | N          |
| FNZ                 | PRL      | Personal               | 24887120L    | PRL                      | PRL           | 1/4/15  | S          |
|                     |          |                        |              |                          |               | 22/1/16 | N          |

# Relación ES\_UN o SUBENTIDAD

- Una subentidad  $S$  de una regular  $R$  se implementa incluyendo una referencia en  $S$  hacia  $R$
- La clave Primaria de  $S$  será la misma que la clave de la entidad regular  $R$
- Esa es la diferencia con la relación de debilidad. Las entidades débiles tienen su propia clave, aunque dependan de otra entidad por debilidad
- Existen otras posibilidades de Implementación como, por ejemplo, el uso de discriminantes



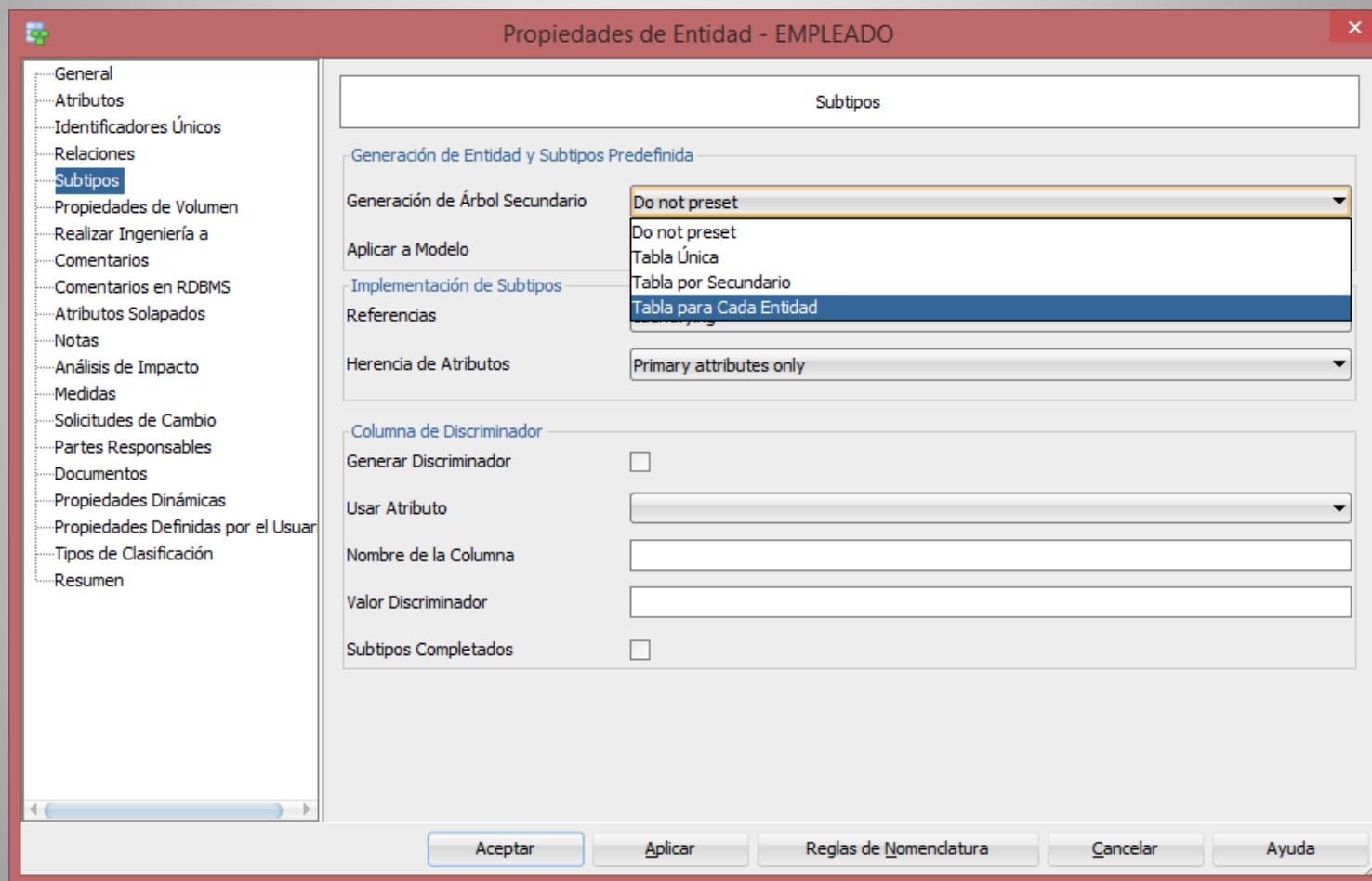
Entidades

Relaciones

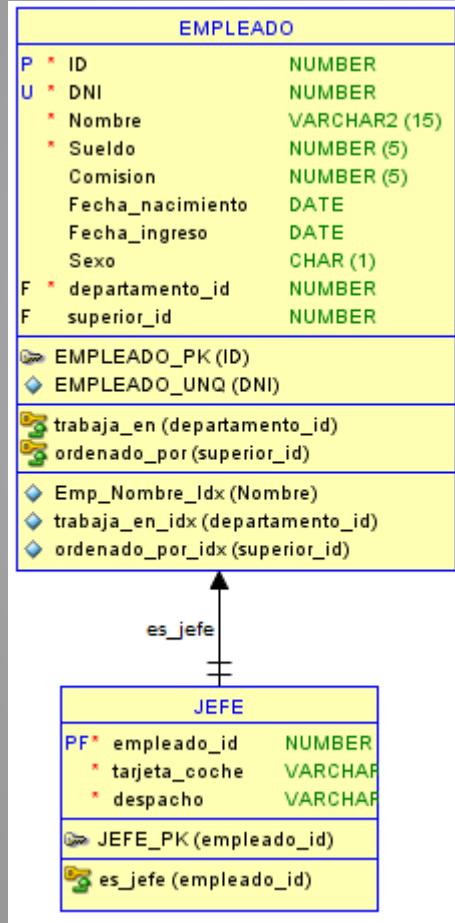
Optimización

Relación 1:M  
Relación M:M  
Entidad Débil  
**SubEntidad**  
Relación 1:1

# Relación ES\_UN o SUBENTIDAD



# Relación ES\_UN o SUBENTIDAD



```

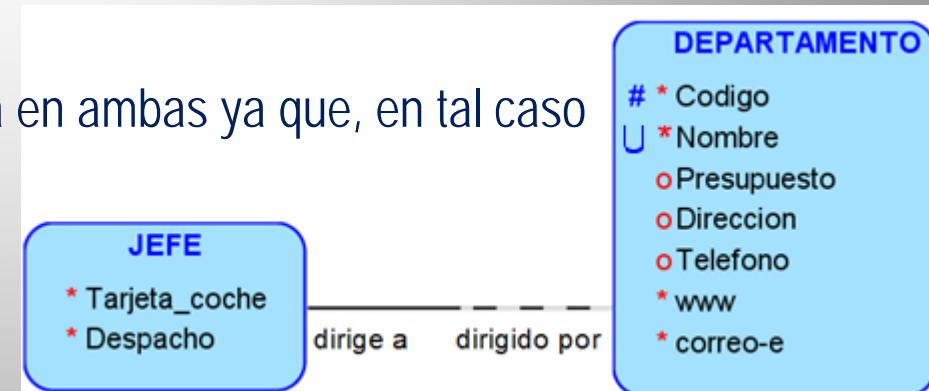
CREATE TABLE JEFE (
    empleado_id NUMBER NOT NULL ,
    tarjeta_coche VARCHAR2 (16) NOT NULL ,
    despacho     VARCHAR2 (10) NOT NULL
);

ALTER TABLE JEFE ADD CONSTRAINT JEFE_PK PRIMARY KEY
( empleado_id );

ALTER TABLE JEFE ADD CONSTRAINT es_jefe FOREIGN KEY
( empleado_id ) REFERENCES EMPLEADO ( ID );
    
```

## Relación 1:1

- Una Relación 1 a 1 entre dos Entidades *A* y *B* se implementa según la obligatoriedad:
  - Si es obligatoria en *A* y no en *B* se implementa en *A* agregando un atributo no nulo que referencia a la clave primaria de *B*. Ese atributo debe tener la restricción UNIQUE para impedir que varias filas de *A* referencia a la misma de *B*. Si la clave de *B* es compuesta, se añadirán tantos atributos como atributos tenga la clave.
  - Si es opcional en ambas entidades, se puede implementar en cualquiera de las 2. Por motivos de eficiencia, se suele hacer en la que se prevea que tenga menos filas.
  - Es poco frecuente que sea obligatoria en ambas ya que, en tal caso suelen ser la misma entidad y, por tanto, se implementan en una sola tabla.



# Relación 1:1

ALTER TABLE JEFE

ADD (dirige\_departamento\_id NUMBER NOT NULL);

ALTER TABLE JEFE ADD CONSTRAINT JEFE\_DPTO\_UNQ  
UNIQUE (dirige\_departamento\_id) ;

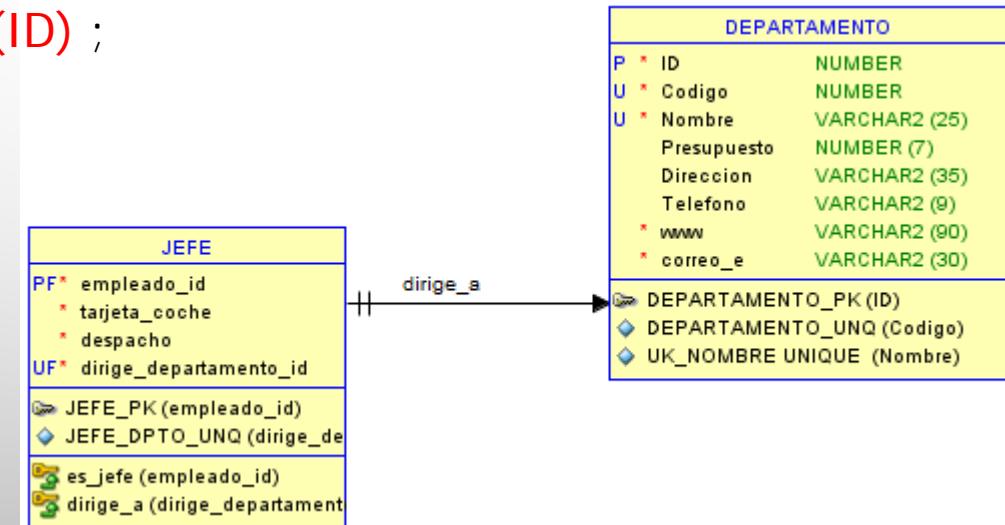
ALTER TABLE JEFE ADD CONSTRAINT dirige\_a  
FOREIGN KEY (dirige\_departamento\_id)  
REFERENCES DEPARTAMENTO (ID) ;

uno a uno

relaciona

Toda restricción  
UNIQUE crea un  
índice  
automáticamente

**obligatoria**



# Relación 1:1

## Departamento

| Codigo | Nombre       | Presupuesto | ... |  |
|--------|--------------|-------------|-----|--|
| ING    | Ingeniería   | 12000,00    | ... |  |
| FNZ    | Contabilidad | 7500,50     | ... |  |

## Empleado

| DNI       | Nombre       | Sueldo   | ... | Sexo | departamento_codigo | superior_DNI |
|-----------|--------------|----------|-----|------|---------------------|--------------|
| 33387199W | Eduardo Gris | 30000,00 | ... | H    | ING                 | 77623977J    |
| 24887120L | Pedro Rojas  | 28675,23 | ... | H    | ING                 |              |
| 77623977J | Ana Negrín   | 35098,44 | ... | M    | FNZ                 |              |

## Jefe

| empleado_dni | tarjeta | despacho | Dirige_dpto_cod |
|--------------|---------|----------|-----------------|
| 33387199W    | 200843  | 3.2.11B  | ING             |
| 77623977J    | 771003  | 7.0.1C   | FNZ             |

- Eduardo dirige Ingeniería  
 Ana dirige Contabilidad

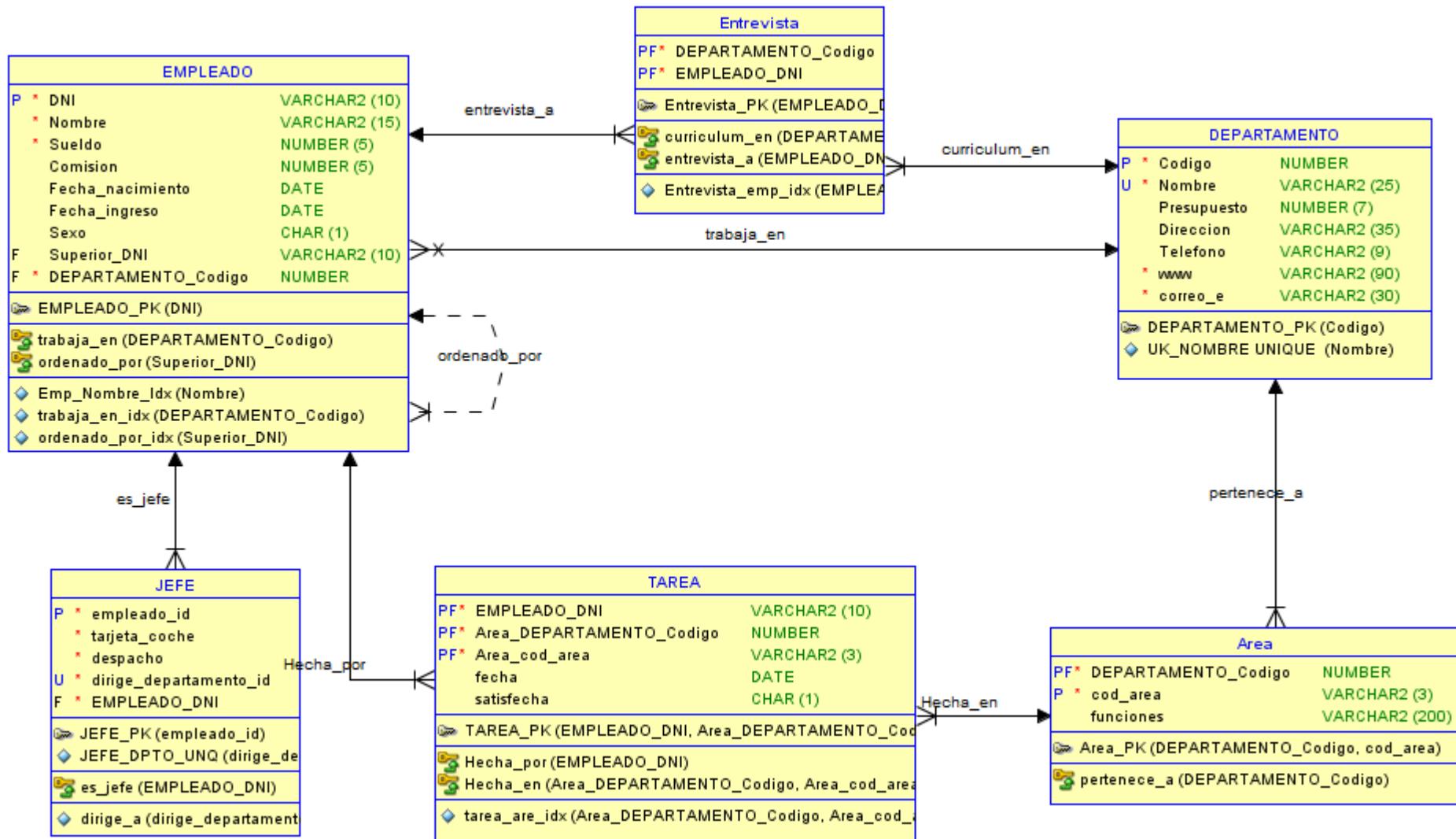
## Buenas prácticas

- En general, hay que evitar claves compuestas
- Los índices son básicos. Una base de datos de verdad sin índices es inservible
- Si las claves primarias no son identificadores internos entonces el diseño puede complicarse mucho
- Usar secuencias para llenar las claves

## Secuencias

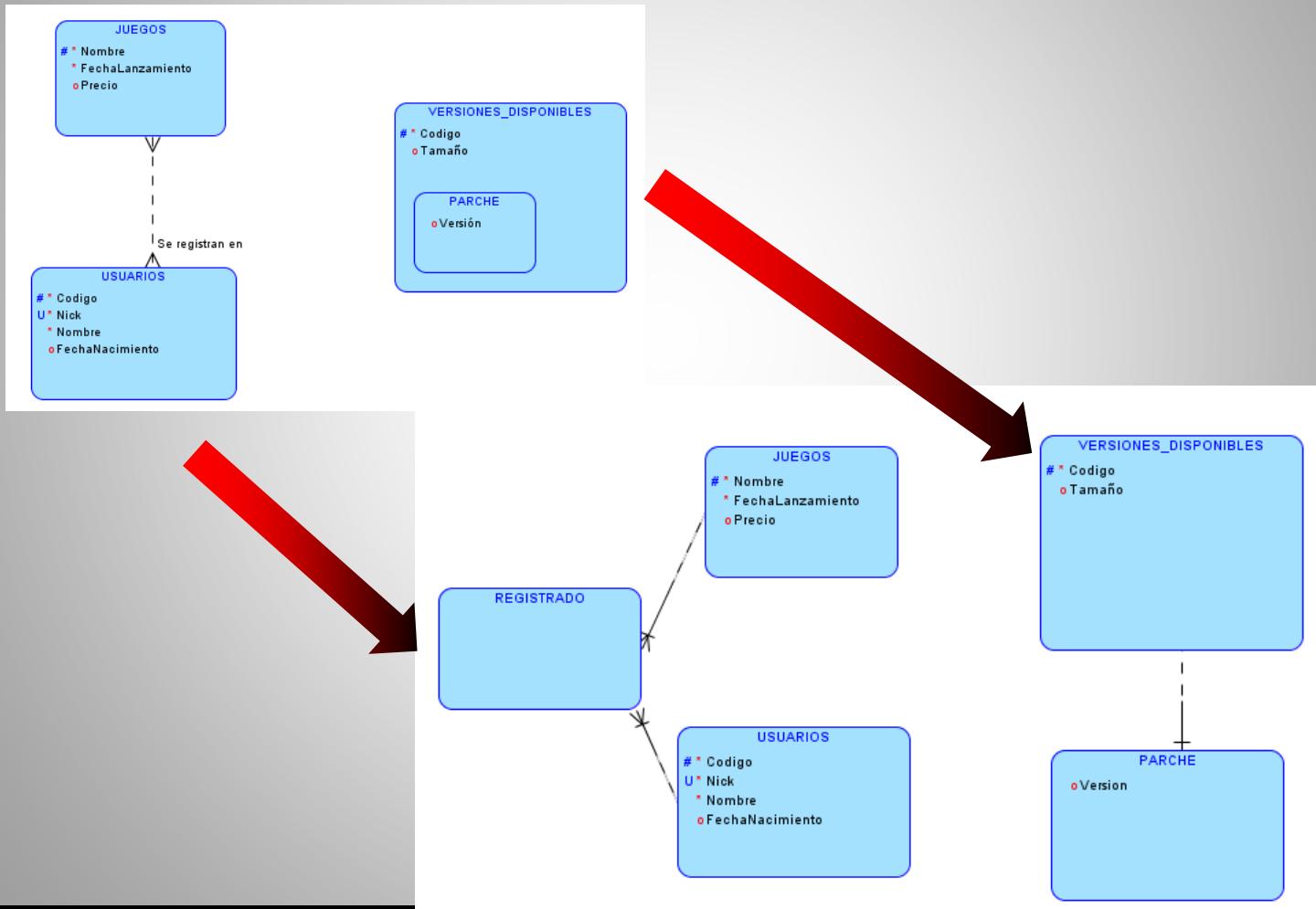
- Una secuencia se crea con CREATE SEQUENCE:  
CREATE SEQUENCE Empleado\_id\_SEQ;
- Y se usa en un INSERT con la expresión:  
Empleado\_id\_SEQ.next\_val
- Ejemplo:  
INSERT INTO EMPLEADO VALUES(  
Empleado\_id\_SEQ.next\_val,  
'33789456w',  
'Juan Pedro', ...)

# Diagrama Final



# Equivalencias

¿Hay diferencias entre estos diagramas?



## Tema 6. Introducción a la Normalización

# Bases de Datos

Departamento de Lenguajes y  
Ciencias de la Computación



<http://www.lcc.uma.es>

Normalización de Bases de Datos

## Información redundante y anomalías en tuplas

- Mezclar atributos de distintas entidades no es adecuado
- Provoca redundancia en el almacenamiento
- Provoca anomalías de inserción, modificación y borrado
- Imaginemos la relación  
EMP\_PROJ ( Empleado#, Proyecto#, ENombre, PNombre, N\_horas)
  - Cambiar el nombre del proyecto 1 de ‘AEROESPACIAL’ a ‘MANTEINIMENTO’ produce que cambie para los 100 empleados trabajando en el proyecto 1. **Anomalía de modificación.**
  - No podemos insertar un proyecto a menos que asignemos al menos un empleado.  
**Anomalía de inserción.**
    - A la inversa, no podemos insertar un empleado a menos que exista un proyecto para él.
    - Cuando un proyecto se elimina, se eliminará la información de todos los empleados asignados a ese proyecto. Además, si se elimina el único empleado de un proyecto, eliminaremos también ese proyecto. **Anomalía de borrado.**

<http://www.lcc.uma.es>

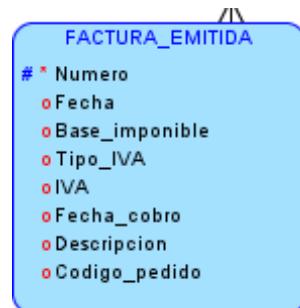
Introducción a la Normalización



¿Esto pasa sólo si mezclas atributos de distintas entidades?

**Anomalía de modificación**

¿Qué ocurre si el gobierno modifica el porcentaje de un tipo de IVA en las facturas vigentes?



<http://www.lcc.uma.es>

Normalización de Bases de Datos



## Semántica de la relación de atributos

- En el diseño de una base de datos, cada tupla en una relación (tabla) debe representar una entidad o instancia de la relación.
  - Los atributos de diferentes entidades (EMPLEADOS, DEPARTAMENTOS, PROYECTOS) no deberían mezclarse en una misma relación.
  - Utilizar claves foráneas para referenciar otras entidades
  - La entidad y sus atributos deben formar una unidad tan independiente como sea posible.
- Por tanto, el diseño que realicemos debe estar, en la medida de lo posible, libre de redundancias y anomalías.
- El objetivo de la **normalización** es proporcionar un procedimiento sistemático para ello.

<http://www.lcc.uma.es>

Introducción a la Normalización



## Dependencias funcionales

| Indicativo | Numero       | País   | Abonado               |
|------------|--------------|--------|-----------------------|
| 34         | 952131000    | España | Universidad de Málaga |
| 34         | 951299316    | España | Turismo Andaluz       |
| 34         | 951299300    | España | Turismo Andaluz       |
| 1          | 800-275-2273 | EEUU   | Apple                 |
| 1          | 800-426-9400 | EEUU   | Microsoft             |
| 1          | 951299300    | Canadá | Peter Smith           |
| 1          | 800-555-0077 | EEUU   | Turismo Andaluz       |
| ...        | ...          | ...    | ...                   |

Pais, Numero → Abonado  
Pais → Indicativo  
Son dependencias funcionales. ¿Hay algunas más?

### Preguntas:

¿Hay valores repetidos en la tabla?  
¿Hay redundancia en la tabla?  
¿Dónde? ¿Es mala la redundancia?

**Preguntas:** ¿Es igual  $A \rightarrow B, C$  que  $A \rightarrow B, A \rightarrow C$ ?

### Definición

Entre los atributos  $A_1, A_2, \dots, A_n$  y  $B$  de una relación  $R$  hay una Dependencia Funcional (DF) si siempre que dos tuplas de  $R$  coincidan en sus valores de  $A_1, A_2, \dots, A_n$ , coinciden también en sus valores de  $B$ .

**Notación:**  $A_1, A_2, \dots, A_n \rightarrow B$

Generalizamos la definición:  $A_1, A_2, \dots, A_n \rightarrow B_1, B_2, \dots, B_m$   
Si  $A \rightarrow B$ ,  $A$  se denomina el determinante y  $B$  el determinado



<http://www.lcc.uma.es>

Introducción a la Normalización

## Claves

### Claves y DFs

Dada la relación  $R$ , decimos que un subconjunto de sus atributos  $K \subseteq \{A_1, A_2, \dots, A_n\}$  es clave si:

- Es posible identificar una única fila por el valor en la clave (Identificación).
- Es imposible que dos tuplas de  $R$  coincidan en la clave (Indistinción).
- La clave determinan funcionalmente a todos los atributos de la relación.  $K \rightarrow A_1, A_2, \dots, A_n$ . (Determinación).

### ¿Claves?

**Definición:** Dada  $R$  con clave  $K$ , se dice minimal si  $\nexists K' \subset K$  que sea clave.

**Definición:** Las claves de una relación que tienen una clave minimal como subconjunto propio se dicen Super-claves.

La clave **primaria** es una clave seleccionada de entre todas las claves de una relación. ¿qué criterios utilizar?



<http://www.lcc.uma.es>

Normalización de Bases de Datos

## Normalización

- Es el proceso de descomponer tablas o relaciones “*mal*” construidas, generalmente dividiéndola en relaciones más pequeñas.
- Una **forma normal** es un condición que cumple una relación de acuerdo a sus claves y dependencias.
- Las tres primeras formas normales (1FN,2FN,3FN) y la forma normal de Boyce-Codd (FNBC) están basadas en sus claves y dependencias funcionales.
- La cuarta forma normal (4FN) se basa en las claves y las dependencias multivaluadas (MVD) de una relación. La quinta forma normal (5FN) se basa en las claves y las dependencias de unión (JD) de una relación.
- No obstante lo anterior, con la FNBC se obtiene un diseño de la BD correcto, por lo que en este tema no se estudian la 4FN ni la 5FN.
- Para asegurar el mejor diseño posible de una base de datos aparte de las formas normales anteriores es necesario tener en cuenta **otras propiedades**.



## Definiciones previas

- Se define un **atributo primario** como aquel que es miembro de alguna clave candidata. Cualquier otro atributo es considerado un atributo no primario.
- Una DF  $A \rightarrow B$  es **plena** si no existe otra DF  $C \rightarrow B$  donde  $C$  es un subconjunto propio de  $A$ .
- Una DF  $A \rightarrow B$  es **transitiva** si existe un atributo  $C$  con  $C \not\subseteq A$  y  $B \not\subseteq C$  tal que existen las dependencias funcionales  $A \rightarrow C$  y  $C \rightarrow B$ .



## 1FN

- La primera forma normal es aquella que no permite atributos compuestos o multivalor; es decir, atributos cuyos valores para un tupla individual no son atómicos.
  - Se considera que cualquier tabla, dada las restricciones del DDL está en 1FN.

| Indicativo | Numero                 | País   | Abonado               |
|------------|------------------------|--------|-----------------------|
| 34         | 952131000              | España | Universidad de Málaga |
| 34         | 951299316<br>951299300 | España | Turismo Andaluz       |
| 1          | 800-275-2273           | EEUU   | Apple                 |
| 1          | 800-426-9400           | EEUU   | Microsoft             |
| 1          | 951299300              | Canadá | Peter Smith           |
| 1          | 800-555-0077           | EEUU   | Turismo Andaluz       |
| ...        | ...                    | ...    | ...                   |



| Indicativo | Numero       | País   | Abonado               |
|------------|--------------|--------|-----------------------|
| 34         | 952131000    | España | Universidad de Málaga |
| 34         | 951299316    | España | Turismo Andaluz       |
| 34         | 951299300    | España | Turismo Andaluz       |
| 1          | 800-275-2273 | EEUU   | Apple                 |
| 1          | 800-426-9400 | EEUU   | Microsoft             |
| 1          | 951299300    | Canadá | Peter Smith           |
| 1          | 800-555-0077 | EEUU   | Turismo Andaluz       |
| ...        | ...          | ...    | ...                   |



<http://www.lcc.uma.es>

Introducción a la Normalización

## 2FN

- Un esquema de relación R se encuentra en 2FN si todo atributo no primario de R se encuentra en una DF plena con la clave primaria.
- Si R no cumple la 2FN tiene que descomponerse en varias relaciones 2FN, por cada clave parcial con sus atributos dependientes.

normalización

| Indicativo | Numero       | País   | Abonado               |
|------------|--------------|--------|-----------------------|
| 34         | 952131000    | España | Universidad de Málaga |
| 34         | 951299316    | España | Turismo Andaluz       |
| 34         | 951299300    | España | Turismo Andaluz       |
| 1          | 800-275-2273 | EEUU   | Apple                 |
| 1          | 800-426-9400 | EEUU   | Microsoft             |
| 1          | 951299300    | Canadá | Peter Smith           |
| 1          | 800-555-0077 | EEUU   | Turismo Andaluz       |
| ...        | ...          | ...    | ...                   |



| Numero       | País   | Abonado               | Indicativo | País   |
|--------------|--------|-----------------------|------------|--------|
| 952131000    | España | Universidad de Málaga | 34         | España |
| 951299316    | España | Turismo Andaluz       | 1          | EEUU   |
| 951299300    | España | Turismo Andaluz       | 1          | Canadá |
| 800-275-2273 | EEUU   | Apple                 | ...        | ...    |
| 800-426-9400 | EEUU   | Microsoft             |            |        |
| 951299300    | Canadá | Peter Smith           |            |        |
| 800-555-0077 | EEUU   | Turismo Andaluz       |            |        |
| ...          | ...    | ...                   |            |        |



<http://www.lcc.uma.es>

Introducción a la Normalización

## 3FN

- Un esquema de relación R se encuentra en 3FN si se está en 2FN y ningún atributo no primario de R es transitivamente dependiente de la clave primaria a través de atributos no primarios. Si la dependencia transitiva es a través de una clave, no hay problema.
- Si R no cumple la 3FN tiene que descomponerse en varias relaciones 3FN que incluyan los atributos no primarios que determinen funcionalmente otros atributos no primarios.

normalización

| Nacimiento | Número       | País   | Abonado     | Número       | País   | Abonado     | Nacimiento | Abonado     |
|------------|--------------|--------|-------------|--------------|--------|-------------|------------|-------------|
| 1974       | 952131000    | España | Francisca   | 952131000    | España | Francisca   | 1974       | Francisca   |
| 1974       | 951299316    | España | Lucas       | 951299316    | España | Lucas       | 1974       | Lucas       |
| 1964       | 951299300    | España | Lolo        | 951299300    | España | Lolo        | 1964       | Lolo        |
| 1998       | 800-275-2273 | EEUU   | Antonio     | 800-275-2273 | EEUU   | Antonio     | 1998       | Antonio     |
| 1998       | 800-426-9400 | EEUU   | Miguel      | 800-426-9400 | EEUU   | Miguel      | 1998       | Miguel      |
| 1989       | 951299300    | Canadá | Peter Smith | 951299300    | Canadá | Peter Smith | 1989       | Peter Smith |
| 1979       | 800-555-0077 | EEUU   | Tomás       | 800-555-0077 | EEUU   | Tomás       | 1979       | Tomás       |
| ...        | ...          | ...    | ...         | ...          | ...    | ...         | ...        | ...         |



<http://www.lcc.uma.es>

Introducción a la Normalización

## Definiciones generales de las FN

- Las definiciones anteriores consideran sólo las claves primarias, aunque también pueden definirse en función de cualquier clave candidata.
- Un esquema de relación R se encuentra en 2FN si cualquier atributo no primario de R se encuentra en una DF plena con *todas las claves* de R.
- Un esquema de relación R se encuentra en 3FN si cuando existe una DF  $X \rightarrow A$  en R entonces se cumple una de las siguientes condiciones:
  - a) X es una superclave de R
  - b) A es un atributo primario de R
- La FNBC se puede definir como una 3FN en la que no se permite b).



<http://www.lcc.uma.es>

Introducción a la Normalización

## FNBC

- Un esquema de relación R se encuentra en FNBC si cuando existe una DF  $X \rightarrow A$  en R entonces se cumple que X es una superclave de R.
- Dicho de otro modo, una relación está en FNBC si y solo si está en 3FN y cada determinante es una clave candidata
- Cada forma normal es estrictamente más fuerte que la anterior
- Existen relaciones que están en 3FN pero no en FBC. El objetivo es conseguir que todas las relaciones de nuestro diseño estén en FNBC.



<http://www.lcc.uma.es>

Introducción a la Normalización

## FNBC. Ejemplo. Direcciones Postales:

| CPost | Dir                   | Ciudad  |
|-------|-----------------------|---------|
| 3000  | C/ Las Flores N°17    | Merida  |
| 4858  | Av. Bolívar este N°72 | Maracay |

A Ciudades diferentes le corresponden códigos postales distintos.

En este caso hay dependencia entre el Código Postal  $\rightarrow$  Ciudad, ya que, conocido el Código Postal se puede conocer la Ciudad, y además Dirección, Ciudad  $\rightarrow$  Código Postal. Existe pues un determinante, Código Postal que no es clave candidata

Para transformar la tabla en una tabla en FNBC se crea una tabla de Códigos Postales y Ciudades, eliminando de la tabla original la Ciudad, obteniéndose dos tablas, una con los atributos Dirección y Código Postal y otra con el Código Postal y la Ciudad

Tablas en FNBC:

| CÓDIGOS_CIUDADES |         |
|------------------|---------|
| CPost            | Ciudad  |
| 3000             | Merida  |
| 4858             | Maracay |

| CÓDIGOS_DIRECCIONES |                       |
|---------------------|-----------------------|
| CPost               | Dir                   |
| 3000                | C/ Las Flores N°17    |
| 4858                | Av. Bolívar este N°72 |



<http://www.lcc.uma.es>

Normalización de Bases de Datos

## FNBC. Otro Ejemplo.

Consideremos una empresa donde un trabajador puede trabajar en varios departamentos. En cada departamento hay varios responsables, pero cada trabajador sólo tiene asignado uno.

Tendríamos una tabla con las columnas:

**IDTrabajador, IDDepartamento, IDResponsable**

La única clave candidata es IDTrabajador (que será por tanto la clave primaria).

Si añadimos la limitación de que el responsable sólo puede serlo de un departamento, este detalle produce una dependencia funcional ya que: **IDResponsable → IDDepartamento**

Por lo tanto hemos encontrado un determinante (IDResponsable) que sin embargo no es clave candidata. Por ello, esta tabla no está en FNBC. En este caso la redundancia ocurre por mala selección de clave. La repetición del par [IDDepartamento + IDResponsable] es innecesaria y evitable

Una forma sencilla de comprobar si una relación se encuentra en FNBC consiste en comprobar, además de que esté en 3FN, lo siguiente:

- (1) Si no existen claves candidatas compuestas (con varios atributos), está en FNBC.
- (2) Si existen varias claves candidatas compuestas y éstas tienen un elemento común, puede no estar en FNBC. Sólo si, para cada dependencia funcional en la relación, el determinante es una clave candidata, estará en FNBC.



# **SQL El lenguaje de Consulta Estructurado**

Introducción, Proyección, Ordenación,  
Operaciones de Conjunto, Selección



## ¿Qué es SQL?

- El lenguaje de Consulta Estructurado (Structured Query Language) es un lenguaje de control e interacción con un SGBD.
- La primera versión de SQL fue desarrollado por IBM como lenguaje de descripción y manipulación de datos de la base de datos relacional System R a principios de los setenta.
- Permite
  - la creación y modificación de esquemas externos, conceptuales e internos (Data Definition Language)
  - la consulta y el mantenimiento de los datos. (Data Manipulation Language)

## Un poco de historia

- 1970 Codd define el modelo relacional
- 1974 Se inicia la construcción de System R
- 1979 ORACLE es la primera BDR comercial con SQL
- 1982 ANSI forma el Comité de Estandares de SQL
- 1986 ISO (International Standards Organization) y  
ANSI (American National Standards Institute)  
desarrollaron una versión estándar llamada  
SQL86 o SQL1
- 1989 2º estándar ANSI/ISO de SQL (SQL/89)

## ¿Qué es SQL?

- 1992 3<sup>er</sup> estándar ANSI/SQL (SQL/92 o SQL-2)
- 1999 Publicación de SQL-3. Añade expresiones regulares, consultas recursivas (relaciones jerárquicas) y disparadores (triggers).
- 2003 Introduce algo de XML y columnas autonuméricas.
- 2005 SQL-2005. Aumenta la integración de XML en SQL.
- 2008 SQL-2008. Se incluyen los disparadores tipo INSTEAD OF y el comando TRUNCATE.

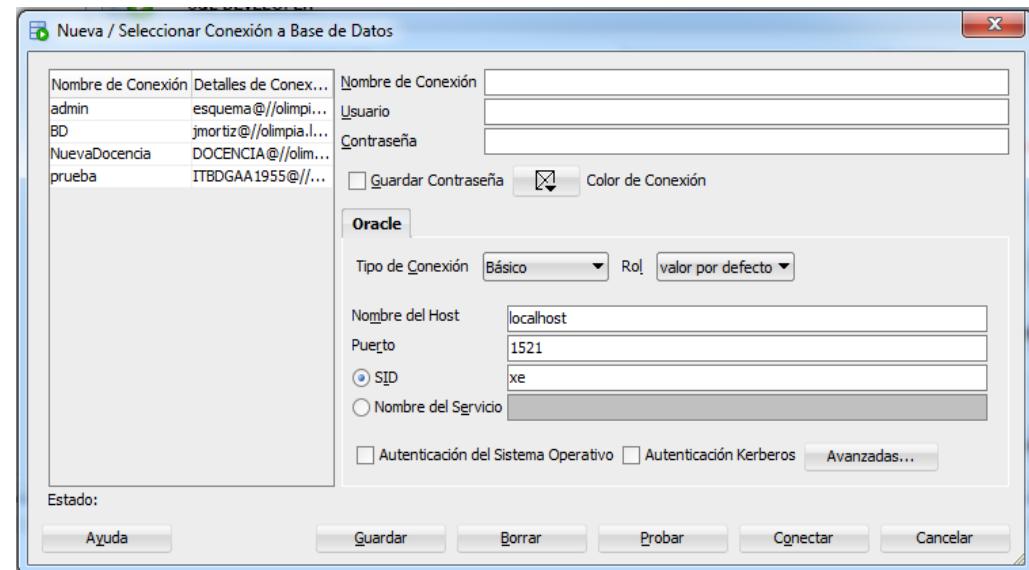
## Interactuando con la BD ORACLE



- Arrancar el servidor ORACLE y una instancia de BD
- Uso de un cliente
  - SQL\*PLUS (Oracle 8.1.6)
  - SQL\*PLUS Worksheet (ORACLE 8i)
  - iSQLPLUS (ORACLE 9i)
  - SQL Developer
  - Otros clientes de terceras partes.

## Conexión

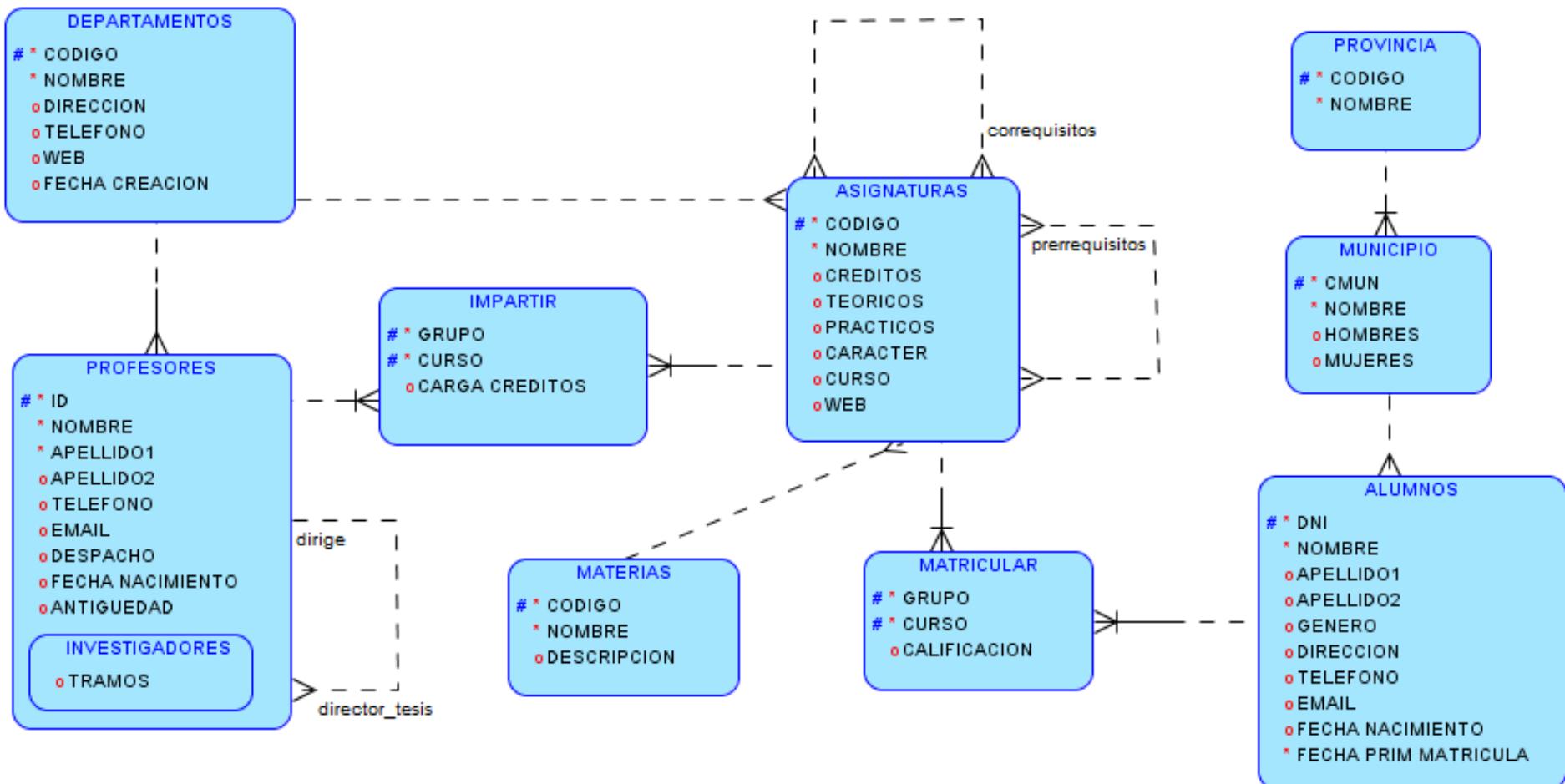
- Datos Conexión:
  - Localhost: olimpia.lcc.uma.es
  - SID: apolo
  - Puerto: 1521
- Cuenta general:
  - Usuario: Alumnobd1
  - Clave: bd
- Tablas en usuario: docencia



# Introducción al SQL

## La sentencia SELECT

### Base de Datos



## Mostrar Información

- La sentencia **SELECT** permite mostrar el cuerpo de una tabla, es decir, el conjunto de filas que están almacenadas en ella.

**SELECT \* FROM *tabla***

- La sentencia **DESC** permite mostrar el esquema de una tabla, es decir, el conjunto de columnas que la forman y sus dominios.

**DESC *tabla***

## Mostrar Información

- Mostrar el esquema de la tabla *alumnos*

DESC alumnos;

| Nombre               | Nulo     | Tipo           |
|----------------------|----------|----------------|
| DNI                  | NOT NULL | NUMBER (9)     |
| NOMBRE               | NOT NULL | VARCHAR2 (20)  |
| APELLIDO1            |          | VARCHAR2 (20)  |
| APELLIDO2            |          | VARCHAR2 (20)  |
| GENERO               |          | VARCHAR2 (4)   |
| DIRECCION            |          | VARCHAR2 (100) |
| TELEFONO             |          | VARCHAR2 (9)   |
| EMAIL                |          | VARCHAR2 (40)  |
| FECHA_NACIMIENTO     |          | DATE           |
| FECHA_PRIM_MATRICULA | NOT NULL | DATE           |
| CPRO                 |          | NUMBER (2)     |
| CMUN                 |          | NUMBER (5)     |

# Introducción al SQL

## La sentencia SELECT

Introducción

Proyección

Ordenación

Operaciones de conjunto

Selección

## Mostrar Información

- Mostrar el cuerpo de la tabla *alumnos*

```
SELECT *
FROM alumnos;
```

| DNI      | NOMBRE         | APELLIDO1  | APELLIDO2         | GENERO | DIRECCION           | TELEFONO  | EMAIL                   | FECHA_NACIMIENTO |
|----------|----------------|------------|-------------------|--------|---------------------|-----------|-------------------------|------------------|
| 25748955 | Guillermo      | Villanueva | Pedraza           | MASC   | C/Rueda 75-1G.      | 617898999 | devott@fairmail.com     | 22/12/97         |
| 87906454 | Manuel         | Fernandez  | Ortega            | MASC   | C/Latina 15-2G.     | 617568999 | mfo@mimail.com          | 02/12/98         |
| 87675655 | Justo          | Iglesias   | Aramillo          | MASC   | C/Mirolta 25-1G.    | 617886999 | JIA@fairmail.com        | 08/12/98         |
| 73635423 | Rafael         | Villanueva | Galvez            | MASC   | C/Madrid 45-1G.     | 617884999 | Eroto@fairmail.com      | 20/11/98         |
| 73143332 | Maria Jose     | Lillo      | Gamez             | FEM    | C/Paris 75-1G.      | 616368999 | jmlillo@fairmail.com    | 18/10/97         |
| 23456456 | Victoria       | Mayordomo  | Toro              | FEM    | C/Quito 75-1G.      | 628898999 | rgueo@micromail.com     | 12/09/97         |
| 67856744 | Jose           | Gil        | Hilo              | MASC   | C/Tanger 5-1G.      | 628898999 | spider@micromail.com    | 24/08/97         |
| 23423424 | David          | Alarcon    | Garcia y Muntaner | MASC   | C/Fuente Paz 25-1G. | 619238999 | ttguor@micromail.com    | 09/07/97         |
| 67867888 | Isidro         | Villanueva | Fernandez         | MASC   | C/Cornisa 35-1G.    | 617857999 | dalarcong@micromail.com | 16/07/99         |
| 65565643 | Nicolas        | Bersabe    | Alba              | MASC   | C/Barqueros 45-1G.  | 952656969 | bersabe@micromail.com   | 30/05/99         |
| 97800676 | Carlos Alberto | Acosta     | Castilla          | MASC   | C/Zapateros 55-1G.  | 952398999 | acostaCA@mimail.com     | 30/04/96         |
| 45645644 | Rodrigo        | Castro     | Picon             | MASC   | C/Rodaderos 13-1G.  | 952893299 | rcp56@mimail.com        | 01/04/96         |
| 23423332 | Benjamin       | Ruiz       | Palacio           | MASC   | C/Italia 32-1G.     | 951898999 | tres@mimail.com         | 02/03/96         |
| 56756777 | Gema           | Jimenez    | Larios            | FEM    | C/Filipinas 134-1G. | 952898119 | (null)                  | 03/02/97         |

...

## Proyección

- La proyección sirve para elegir qué atributos queremos en el resultado de una consulta

`SELECT Lista_Atributos FROM tabla`

- En la lista de atributos se puede especificar un alias para cada atributo.

Atrib1 "Alias completo", Atrib2 "Alias2",...

## Proyección

- Listar el Nombre y Código de departamento de todos los profesores

```
SELECT Nombre "NOMBRE DE PILA",
       Departamento
  FROM profesores;
```

| NOMBRE DE PILA | DEPARTAMENTO |
|----------------|--------------|
| Michael        | 4            |
| Manuel         | 1            |
| Enrique        | 2            |
| Miguel         | 1            |
| Maria del Mar  | 2            |
| Sergio         | 1            |
| Carlos         | 3            |
| Ana            | 3            |
| Miguel         | 4            |
| Juana          | 4            |
| Antonio        | 4            |

## Eliminación de tuplas repetidas

- Las **tuplas** repetidas en el resultado de la consulta pueden eliminarse con DISTINCT.
- Listar los nombres diferentes de profesores.

```
SELECT DISTINCT Nombre  
FROM profesores;
```

| NOMBRE        |
|---------------|
| Michael       |
| Enrique       |
| Sergio        |
| Juana         |
| Miguel        |
| Maria del Mar |
| Manuel        |
| Antonio       |
| Carlos        |
| Ana           |

## Ordenación

- La cláusula **ORDER BY** permite ordenar las tuplas del resultado de una consulta

**ORDER BY Lista\_Atributos**

- Para cada atributo se puede indicar el sentido de la ordenación:
  - ASC, ascendente (por defecto)
  - DESC, descendente

## Ordenación

- Listar el Nombre y Departamento de todos los profesores ordenados por departamento (decreciente) y por orden alfabético dentro del departamento

```
SELECT Nombre, Departamento  
FROM   profesores  
ORDER BY Departamento DESC,  
        Nombre ASC;
```

| NOMBRE        | DEPARTAMENTO |
|---------------|--------------|
| Antonio       | 4            |
| Juana         | 4            |
| Michael       | 4            |
| Miguel        | 4            |
| Ana           | 3            |
| Carlos        | 3            |
| Enrique       | 2            |
| Maria del Mar | 2            |
| Manuel        | 1            |
| Miguel        | 1            |
| Sergio        | 1            |

# Identificación por posición

- Aunque no es recomendable, si la ordenación se realiza utilizando alguna de las columnas de la proyección se puede utilizar su posición en la cláusula ORDER BY.

```
SELECT Nombre, Departamento  
FROM profesores  
ORDER BY 2 DESC,  
        1 ASC;
```

| NOMBRE        | DEPARTAMENTO |
|---------------|--------------|
| Antonio       | 4            |
| Juana         | 4            |
| Michael       | 4            |
| Miguel        | 4            |
| Ana           | 3            |
| Carlos        | 3            |
| Enrique       | 2            |
| Maria del Mar | 2            |
| Manuel        | 1            |
| Miguel        | 1            |
| Sergio        | 1            |

## Operaciones de conjunto

- Dado que los resultados de las consultas son relaciones (conjuntos de tuplas) se puede utilizar operaciones de conjuntos para combinar los resultados de varias sentencias SELECT.
- Operaciones:  
**UNION, UNION ALL, INTERSECT, MINUS**
- Las relaciones deben ser unión-compatibles.  
En caso de que no lo sean, se puede conseguir la unión-compatibilidad gracias a la proyección.

## Operaciones de conjunto

- Mostrar todos los emails de las personas existentes en la base de datos

```
SELECT email  
FROM profesores
```

UNION

```
SELECT email  
FROM alumnos;
```

| EMAIL           |
|-----------------|
| 11087825@uma.es |
| 11569523@uma.es |
| 12127891@uma.es |
| 13504134@uma.es |
| 14156094@uma.es |
| 14785585@uma.es |
| 15940923@uma.es |
| 16891547@uma.es |
| 17647730@uma.es |
| 20058694@uma.es |
| 21012656@uma.es |
| 22903027@uma.es |
| 26651441@uma.es |
| 27921947@uma.es |
| 28179524@uma.es |
| ...             |

## Conservación de tuplas repetidas

- Las operaciones UNION, MINUS e INTERSECT eliminan tuplas repetidas.
- Ej: Mostrar todos los nombres de personas existentes en la base de datos, sin eliminar repeticiones.

```
SELECT nombre  
FROM profesores  
  
UNION ALL  
  
SELECT nombre  
FROM alumnos;
```

| NOMBRE        |
|---------------|
| Michael       |
| Manuel        |
| Enrique       |
| Miguel        |
| Maria del Mar |
| Sergio        |
| Carlos        |
| Ana           |
| Miguel        |
| Juana         |
| Antonio       |
| Guillermo     |
| Manuel        |
| Justo         |
| Rafael        |
| ...           |

## Producto Cartesiano

- Se puede utilizar en relaciones que no son Unión-compatibles

FROM *Lista\_tablas*

- Se pueden utilizar alias de tabla:  
FROM *tabla1 Alias1, tabla2 alias2, ...*
- Se utiliza la notación punto para identificar atributos con el mismo nombre en diferentes tablas (*tabla.atributo*)
- Tiene un **alto coste computacional**.

# Introducción al SQL

## La sentencia SELECT

Introducción  
Proyección  
Ordenación  
Operaciones de conjunto  
Selección

- Listar todas las combinaciones posibles de matriculación de todos los alumnos

```
SELECT AL.Nombre, ASI.Nombre  
FROM alumnos AL, asignaturas ASI;
```

| NOMBRE     | NOMBRE_1             |
|------------|----------------------|
| Guillermo  | Teoria de la señal   |
| Manuel     | Teoria de la señal   |
| Justo      | Teoria de la señal   |
| Rafael     | Teoria de la señal   |
| Maria Jose | Teoria de la señal   |
| Victoria   | Teoria de la señal   |
| Jose       | Teoria de la señal   |
| ...        |                      |
| Guillermo  | Prácticas en empresa |
| Manuel     | Prácticas en empresa |
| Justo      | Prácticas en empresa |
| ...        |                      |
| Guillermo  | Calculo Numerico     |
| Manuel     | Calculo Numerico     |
| Justo      | Calculo Numerico     |
| Rafael     | Calculo Numerico     |
| ...        |                      |

# Selección

- La cláusula **WHERE** permite la seleccionar un subconjunto del cuerpo de una tabla basándose en un predicado

**WHERE** *Predicado*

- Toda fila que satisfaga el predicado aparecerá en el resultado de la consulta.
- Operadores relacionales: `=` , `!=` , `<` , `>` , `<=` , `>=`
- Operadores booleanos: AND, OR, NOT
- Especiales: BETWEEN, LIKE

# Introducción al SQL

## La sentencia SELECT

Introducción  
Proyección  
Ordenación  
Operaciones de conjunto  
Selección

## Selección

- Listar el Nombre de los profesores del departamento de código 1, ordenados alfabéticamente.

```
SELECT Nombre  
FROM profesores  
WHERE departamento = 1  
ORDER BY Nombre ;
```

| NOMBRE |
|--------|
| Manuel |
| Miguel |
| Sergio |

- Listar el nombre y primer apellido de los profesores cuyo nombre empiece por 'M' y que pertenezcan a los departamentos con código entre 2 y 4

```
SELECT Nombre, Apellido1  
FROM profesores  
WHERE departamento BETWEEN 2 AND 4  
AND nombre LIKE 'M%';
```

| NOMBRE        | APELLIDO1 |
|---------------|-----------|
| Michael       | Brown     |
| Maria del Mar | Roldán    |
| Miguel        | Hermoso   |

## Funciones predefinidas

- Existen funciones predefinidas que se pueden utilizar tanto en el predicado de la cláusula WHERE como en la parte de proyección (SELECT ...)
- Cada tupla resultado de la consulta mostrará el resultado de aplicar la función utilizada al valor concreto para esa tupla de los atributos que se usan como parámetros de la función.
- [http://docs.oracle.com/cd/B19306\\_01/server.102/b14200/functions001.htm#i88893](http://docs.oracle.com/cd/B19306_01/server.102/b14200/functions001.htm#i88893)

## Funciones sobre cadenas de caracteres

- **CONCAT(a,b)** ó **(a || b)** concatena las cadenas *a* y *b*. Si *a* y *b* no son de tipo cadena se convierten antes de concatenarlos.
- **SUBSTR(cad, pos, len)** devuelve la subcadena de la cadena *cad*, que comienza en la posición *pos* y tiene longitud *len*.
  - La primera posición de la cadena es la 1.
  - Si *pos* = 0, entonces se considera que se desea empezar en 1.
  - Si *pos* < 0, la posición se determina empezando a contar desde el final de la cadena
- **LOWER(cad)** devuelve una cadena en minúsculas con los mismos símbolos que la cadena pasada.
- **UPPER(cad)** devuelve una cadena en mayúsculas con los mismos símbolos que la cadena pasada.
- etc

## Funciones sobre cadenas de caracteres

- Mostrar un listado con una sola columna con alias 'Id' que muestre para cada profesor una cadena formada por las 3 primeras letras del nombre seguidas de las 3 últimas letras de su primer apellido.

| Id       |
|----------|
| Micown   |
| Maniso   |
| Enrler   |
| Migtiz   |
| Angora   |
| Cesmez   |
| Carsassi |
| Ananez   |
| Migoso   |
| Juadez   |
| Anteava  |

```
SELECT substr(nombre,1,3) || substr(apellido1,-3,3) "Id"  
from profesores;
```

# Funciones numéricas

- Aritméticas:  
+, -, /, \*, MOD, POWER,...
- Trigonométricas:  
SIN, COS, TAN, ASIN, ACOS, ATAN,...
- **ROUND** (n,decimales) redondea n al número de decimales indicado
  - ROUND(n) = ROUND(n,0)
- **TRUNC** (n,decimales) trunca n al número de decimales indicado.
  - TRUNC(n) = TRUNC(n,0)
- etc.

## Funciones numéricas

- Mostrar el resultado de multiplicar 3.1416 y 2.86, tal cual, redondeado y truncado a 3 decimales.

```
SELECT 3.1416*2.86, round(3.1416*2.86,3), trunc(3.1416*2.86,3)  
FROM dual;
```

| 3.1416*2.86 | ROUND(3.1416*2.86,3) | TRUNC(3.1416*2.86,3) |
|-------------|----------------------|----------------------|
| 8,984976    | 8,985                | 8,984                |

## Funciones sobre fechas

- **ADD\_MONTHS**(fechalinicio, m) devuelve la fecha resultante de sumar *m* meses a la *fechalinicio*.
  - Si *fecha2* es más reciente que *fecha1* el número devuelto es negativo.
- **MONTHS\_BETWEEN** (fecha1, fecha2) devuelve la diferencia en meses entre *fecha1* y *fecha2*.
  - Si *fecha2* es más reciente que *fecha1* el número devuelto es negativo.
- *fecha1* – *fecha2*, devuelve la diferencia en días entre dos fechas.
  - Si *fecha2* es más reciente que *fecha1* el número devuelto es negativo.
- **NEXT\_DAY** (fechalinicio, diaSemana) devuelve la fecha posterior a *fechalinicio* correspondiente al siguiente día que coincida con el día de la semana dado.
- **SYSDATE** devuelve la fecha actual del sistema.
- etc.

## Funciones sobre fechas

- Devolver la fecha en que cae el siguiente miércoles a partir del día de hoy

```
SELECT NEXT_DAY(SYSDATE,'Miércoles')  
FROM DUAL;
```

- Mostrar el nombre de cada profesor junto a su edad.

```
SELECT nombre,  
       months_between(sysdate,fecha_nacimiento)/12 "Edad"  
FROM profesores
```

| NOMBRE  | Edad                                      |
|---------|-------------------------------------------|
| Michael | 74,78894016328156113102349661489446435683 |
| Manuel  | 48,85076812027080844285145360414177618475 |
| Enrique | 43,87227349661489446435682994822779769017 |
| Miguel  | 67,26743478693747510951812027080844285142 |
| Angel   | 35,88033801274392672242134607726005575467 |
| Cesar   | 42,59539177618478693747510951812027080842 |
| Carlos  | 85,26743478693747510951812027080844285142 |
| Ana     | (null)                                    |
| Miguel  | (null)                                    |
| Juana   | 24,14915521704500199123855037833532457192 |
| Antonio | 46,1437788729589804858622062923138191955  |

## Funciones de conversión

- **TO\_NUMBER** (cad) transforma la cadena de caracteres a número si el formato es el adecuado.
- **TO\_CHAR** (param) convierte el número o fecha dada a cadena.
  - TO\_CHAR(param,formato) realiza la conversión utilizando el formato dado
  - Se puede utilizar de TO\_CHAR para extraer información del tipo fecha (sólo el mes, sólo el año, sólo el día de la semana, ...)
- **TO\_DATE** (cad,formato) transforma una cadena a fecha siguiendo el formato dado.
- Para conocer cómo especificar el formato hay que consultar la referencia SQL
- [http://docs.oracle.com/cd/B19306\\_01/server.102/b14200/sql\\_elements004.htm#i34510](http://docs.oracle.com/cd/B19306_01/server.102/b14200/sql_elements004.htm#i34510)

## Funciones de conversión

- Mostrar un listado con el nombre, el año y el día de la semana de nacimiento de cada profesor

```
SELECT      nombre,  
            TO_CHAR(fecha_nacimiento,'yyyy') "Año",  
            TO_CHAR(fecha_nacimiento,'DAY') "Día"  
FROM profesores;
```

| NOMBRE  | Año    | Día       |
|---------|--------|-----------|
| Michael | 1942   | SÁBADO    |
| Manuel  | 1968   | LUNES     |
| Enrique | 1972   | DOMINGO   |
| Miguel  | 1949   | LUNES     |
| Angel   | 1980   | DOMINGO   |
| Cesar   | 1974   | MIÉRCOLES |
| Carlos  | 1931   | SÁBADO    |
| Ana     | (null) | (null)    |
| Miguel  | (null) | (null)    |
| Juana   | 1992   | LUNES     |
| Antonio | 1970   | MIÉRCOLES |

## Pseudo-columnas

- Toda tabla proporciona las pseudo-columnas ROWNUM y ROWID. Dichas columnas no están almacenadas en la tabla y sólo pueden consultarse.
- **ROWNUM**, contiene un identificador único para cada *tupla resultado* de la consulta. El identificador indica el orden en que fue seleccionada dicha tupla para la consulta.
- **ROWID**, contiene un identificador único para cada *fila* del *sistema*. Dicho identificador coincide con la localización en disco para encontrar dicha fila.

## Pseudo-columnas

- Mostrar el nombre, rowid y rownum de sólo tres profesores.

```
SELECT nombre, rowid, rownum  
FROM profesores  
WHERE rownum < 4;
```

```
DESC profesores;
```

| NOMBRE  | ROWID                | ROWNUM |
|---------|----------------------|--------|
| Michael | AAASKrAAFAAAAADDAAA  | 1      |
| Manuel  | AAASKrAAFAAAAADDAAAB | 2      |
| Enrique | AAASKrAAFAAAAADDAAAC | 3      |

| Nombre           | Nulo     | Tipo           |
|------------------|----------|----------------|
| NRP              | NOT NULL | VARCHAR2 (20)  |
| NOMBRE           | NOT NULL | VARCHAR2 (20)  |
| APELLIDO1        | NOT NULL | VARCHAR2 (20)  |
| APELLIDO2        |          | VARCHAR2 (20)  |
| DEPARTAMENTO     |          | NUMBER (3)     |
| TELEFONO         |          | VARCHAR2 (4)   |
| EMAIL            |          | VARCHAR2 (100) |
| DESPACHO         |          | VARCHAR2 (10)  |
| FECHA_NACIMIENTO |          | DATE           |
| ANTIGUEDAD       |          | DATE           |
| DIRECTOR_TESIS   |          | VARCHAR2 (20)  |

# Ejercicios

- Mostrar:
  1. Grupo y calificación de todos los alumnos matriculados en la asignatura de código 112.
  2. Código de los profesores que no han dirigido tesis doctorales
  3. Código de los profesores que son directores de tesis y/o imparten asignaturas.
  4. Nombres de alumnos sin su inicial.
  5. Nombres de asignaturas que comiencen por 'B'.
  6. Dni de los alumnos que han aprobado alguna asignatura.
  7. Nombre -> fechaPrimMatrícula, para alumnos con nombre de más de 5 letras.
  8. Cantidad de meses desde la fecha de primera matrícula hasta el 1/12/2013.
  9. El nombre de una asignatura de menos de 6 créditos.

# **SQL El lenguaje de Consulta Estructurado**

Manejo de valores nulos,  
Reunión



## Manejo de valores nulos

- SQL utiliza una lógica trivaluada:  
TRUE, FALSE, NULL.
- La aplicación de una función a un valor nulo devuelve un valor nulo.  
 $F(\text{NULL}, \dots) = \text{NULL}$
- La evaluación de un predicado donde una de sus partes es nula devuelve nulo.  $P(\text{NULL}, \dots) = \text{NULL}$ 
  - ¡Ojo! NULL es valor de verdad diferente de TRUE y de FALSE.

## Manejo de valores nulos

- **expr1 IS [NOT] NULL** devuelve TRUE si expr1 es (no es) NULL.
  - Diferente de expr1 = NULL.
- **NVL (expr1, expr2)** devuelve
  - expr1 si expr1 IS NOT NULL
  - expr2 si expr1 IS NULL
- expr1 y expr2 han de ser del mismo tipo

## Manejo de valores nulos

- Listar el nombre y primer apellido de los profesores que no tienen email

```
SELECT Nombre, Apellido1  
FROM profesores  
WHERE email IS NULL;
```

| NOMBRE  | APELLIDO1  |
|---------|------------|
| Michael | Brown      |
| Miguel  | Ortiz      |
| Miguel  | Hermoso    |
| Juana   | Hernandez  |
| Antonio | Villanueva |

```
SELECT Nombre, Apellido1  
FROM profesores  
WHERE email = NULL;
```

| NOMBRE  | APELLIDO1  |
|---------|------------|
| Michael | Brown      |
| Miguel  | Ortiz      |
| Miguel  | Hermoso    |
| Juana   | Hernandez  |
| Antonio | Villanueva |

Ninguna fila se selecciona porque ninguna hace TRUE el predicado

## Manejo de valores nulos

- Listar el primer apellido y el código del director de tesis de cada profesor. En caso de no tener, se muestra 'No tiene'

```
SELECT apellido1,  
       NVL(director_tesis, 'No Tiene')  
FROM profesores;
```

| APELLIDO1  | NVL(DIRECTOR_TESIS,'NOTIENE') |
|------------|-------------------------------|
| Brown      | A-34-CEU-01                   |
| Enciso     | C-34-TU-00                    |
| Soler      | No Tiene                      |
| Ortiz      | A-89-CEU-99                   |
| Roldán     | A-89-CEU-99                   |
| Gálvez     | C-34-TU-00                    |
| Fernández  | No Tiene                      |
| Jiménez    | No Tiene                      |
| Hermoso    | H-32-TU-99                    |
| Hernandez  | No Tiene                      |
| Villanueva | No Tiene                      |

## Reunión (join)

- Operación adecuada para enlazar información de dos tablas. Conceptualmente, se puede entender como un producto cartesiano al que se le aplica una selección posterior.

```
SELECT lista_atributos
FROM lista_tablas
WHERE condiciones_de_reunión
```

- *Condición\_de\_reunión*: tabla1.atributo1 = tabla2.atributo2
- Una reunión correcta de  $n$  tablas necesitará, al menos,  $n-1$  condiciones de reunión.
- La operación de la reunión evita la ejecución del producto cartesiano y, por consiguiente, su alto coste asociado.

## Reunión. Ejemplo paso a paso

- Mostrar el código de las asignaturas asignadas a Manuel Enciso
- Vamos a definir la consulta en SQL de forma constructiva.

# Reunión. Ejemplo paso a paso

- Mostrar asignación docente añadiendo datos del profesor correspondiente.

```
SELECT *
FROM Impartir I, profesores P;
```

¡¡Hace falta la condición de reunión!!

| PROFESOR    | ASIGNATURA | GRUPO | CURSO | CARGA_CREDITOS | ID      | NOMBRE  | APELLIDO1 | APELLIDO2 | DEPARTAMENTO | TELEFONO | EMAIL  | DESPACHO |
|-------------|------------|-------|-------|----------------|---------|---------|-----------|-----------|--------------|----------|--------|----------|
| C-34-TU-00  | 112 B      | 15/16 |       | 7              | J-98766 | Michael | Brown     | (null)    |              | 4 (null) | (null) | (null)   |
| C-34-TU-00  | 114 A      | 15/16 |       | 6              | J-98766 | Michael | Brown     | (null)    |              | 4 (null) | (null) | (null)   |
| D-92-PC-92  | 113 A      | 15/16 |       | 6              | J-98766 | Michael | Brown     | (null)    |              | 4 (null) | (null) | (null)   |
| D-92-PC-92  | 113 B      | 15/16 |       | 3              | J-98766 | Michael | Brown     | (null)    |              | 4 (null) | (null) | (null)   |
| C-67-CEU-98 | 116 A      | 15/16 |       | 6              | J-98766 | Michael | Brown     | (null)    |              | 4 (null) | (null) | (null)   |
| A-89-CEU-99 | 112 A      | 15/16 |       | 7              | J-98766 | Michael | Brown     | (null)    |              | 4 (null) | (null) | (null)   |
| A-89-CEU-99 | 140 A      | 15/16 |       | (null)         | J-98766 | Michael | Brown     | (null)    |              | 4 (null) | (null) | (null)   |
| G-34-TEU-96 | 140 A      | 15/16 |       | (null)         | J-98766 | Michael | Brown     | (null)    |              | 4 (null) | (null) | (null)   |
| G-34-TEU-96 | 112 B      | 15/16 |       | (null)         | J-98766 | Michael | Brown     | (null)    |              | 4 (null) | (null) | (null)   |
| H-77-TU-99  | 123 A      | 15/16 |       | 4              | J-98766 | Michael | Brown     | (null)    |              | 4 (null) | (null) | (null)   |
| H-32-TU-99  | 123 A      | 13/14 |       | 4              | J-98766 | Michael | Brown     | (null)    |              | 4 (null) | (null) | (null)   |
| H-32-TU-99  | 114 C      | 15/16 |       | 6              | J-98766 | Michael | Brown     | (null)    |              | 4 (null) | (null) | (null)   |
| F-12-AY-02  | 133 A      | 15/16 |       | 6              | J-98766 | Michael | Brown     | (null)    |              | 4 (null) | (null) | (null)   |
| F-12-AY-02  | 101 B      | 15/16 |       | 7              | J-98766 | Michael | Brown     | (null)    |              | 4 (null) | (null) | (null)   |
| F-12-AY-02  | 133 A      | 14/15 |       | 6              | J-98766 | Michael | Brown     | (null)    |              | 4 (null) | (null) | (null)   |
| F-12-AY-02  | 101 B      | 14/15 |       | 7              | J-98766 | Michael | Brown     | (null)    |              | 4 (null) | (null) | (null)   |
| C-34-TU-00  | 140 B      | 14/15 |       | 9              | J-98766 | Michael | Brown     | (null)    |              | 4 (null) | (null) | (null)   |
| C-34-TU-00  | 140 A      | 15/16 |       | 9              | J-98766 | Michael | Brown     | (null)    |              | 4 (null) | (null) | (null)   |

...

# Reunión. Ejemplo paso a paso

- Mostrar asignación docente añadiendo datos del profesor correspondiente.

```
SELECT *
FROM Impartir I, profesores P
WHERE I.profesor=P.id;
```

| PROFESOR    | ASIGNATURA | GRUPO | CURSO | CARGA_CREDITOS | ID          | NOMBRE               | APELLIDO1 | APELLIDO2       | DEPARTAMENTO | TELEFONO      | EMAIL                   |
|-------------|------------|-------|-------|----------------|-------------|----------------------|-----------|-----------------|--------------|---------------|-------------------------|
| A-89-CEU-99 | 112 A      | 15/16 |       | 7              | A-89-CEU-99 | Manuel               | Enciso    | Garcia-Oliveros |              | 1 3309        | enciso_lcc@uma.es       |
| A-89-CEU-99 | 140 A      | 15/16 |       | (null)         | A-89-CEU-99 | Manuel               | Enciso    | Garcia-Oliveros |              | 1 3309        | enciso_lcc@uma.es       |
| C-34-TU-00  | 140 A      | 13/14 |       | 4              | C-34-TU-00  | Enrique              | Soler     | Castillo        |              | 2 2789        | esc@gisum.lcc.uma.es    |
| C-34-TU-00  | 112 B      | 15/16 |       | 7              | C-34-TU-00  | Enrique              | Soler     | Castillo        |              | 2 2789        | esc@gisum.lcc.uma.es    |
| C-34-TU-00  | 114 A      | 15/16 |       | 6              | C-34-TU-00  | Enrique              | Soler     | Castillo        |              | 2 2789        | esc@gisum.lcc.uma.es    |
| C-34-TU-00  | 140 A      | 15/16 |       | 9              | C-34-TU-00  | Enrique              | Soler     | Castillo        |              | 2 2789        | esc@gisum.lcc.uma.es    |
| C-34-TU-00  | 140 B      | 14/15 |       | 9              | C-34-TU-00  | Enrique              | Soler     | Castillo        |              | 2 2789        | esc@gisum.lcc.uma.es    |
| C-67-CEU-98 | 116 A      | 15/16 |       | 6              | C-67-CEU-98 | Miguel               | Ortiz     | (null)          |              | 1 2875        | (null)                  |
| D-92-PC-92  | 113 B      | 15/16 |       | 3              | D-92-PC-92  | Maria del Mar Roldán | García    |                 |              | 2 2875        | mar@ctima.uma.es        |
| D-92-PC-92  | 113 A      | 15/16 |       | 6              | D-92-PC-92  | Maria del Mar Roldán | García    |                 |              | 2 2875        | mar@ctima.uma.es        |
| F-12-AY-02  | 133 A      | 15/16 |       | 6              | F-12-AY-02  | Sergio               | Gálvez    | Rojas           |              | 1 7151        | cesar.lcc.uma.es@uma.es |
| F-12-AY-02  | 101 B      | 14/15 |       | 7              | F-12-AY-02  | Sergio               | Gálvez    | Rojas           |              | 1 7151        | cesar.lcc.uma.es@uma.es |
| F-12-AY-02  | 133 A      | 14/15 |       | 6              | F-12-AY-02  | Sergio               | Gálvez    | Rojas           |              | 1 7151        | cesar.lcc.uma.es@uma.es |
| F-12-AY-02  | 101 B      | 15/16 |       | 7              | F-12-AY-02  | Sergio               | Gálvez    | Rojas           |              | 1 7151        | cesar.lcc.uma.es@uma.es |
| G-34-TEU-96 | 112 B      | 15/16 |       | (null)         | G-34-TEU-96 | Ana                  | Jiménez   | Santos          | 3 (null)     | rqq@lccuma.es |                         |
| G-34-TEU-96 | 140 A      | 15/16 |       | (null)         | G-34-TEU-96 | Ana                  | Jiménez   | Santos          | 3 (null)     | rqq@lccuma.es |                         |

...

## Reunión. Ejemplo paso a paso

- Mostrar asignación de docencia para el profesor Manuel Enciso

```
SELECT *
FROM impartir I, profesores P
WHERE I.profesor=P.id
AND UPPER(P.Nombre) = 'MANUEL'
AND UPPER(P.Apellido1) = 'ENCISO';
```

| PROFESOR    | ASIGNATURA | GRUPO | CURSO  | CARGA_CREDITOS | ID     | NOMBRE | APPELLIDO1      | APPELLIDO2 | DEPARTAMENTO | TELEFONO | EMAIL             | ... |
|-------------|------------|-------|--------|----------------|--------|--------|-----------------|------------|--------------|----------|-------------------|-----|
| A-89-CEU-99 | 112 A      | 15/16 | 7      | A-89-CEU-99    | Manuel | Enciso | Garcia-Oliveros |            | 1 3309       |          | enciso_lcc@uma.es |     |
| A-89-CEU-99 | 140 A      | 15/16 | (null) | A-89-CEU-99    | Manuel | Enciso | Garcia-Oliveros |            | 1 3309       |          | enciso_lcc@uma.es |     |

## Reunión. Ejemplo paso a paso

- Mostrar el código de las asignaturas asignadas a Manuel Enciso

```
SELECT I.asignatura "Codigo Asignatura"  
FROM impartir I, profesores P  
WHERE I.profesor=P.id  
AND UPPER(P.Nombre) = 'MANUEL'  
AND UPPER(P.Apellido1) = 'ENCISO';
```

| Codigo Asignatura |
|-------------------|
| 112               |
| 140               |

## Self join

- Reunión de una tabla consigo misma.
- Utilizar alias de tablas y atributos es fundamental para distinguir cuál de las dos instancias de la tabla estamos usando.

```
SELECT      t1.atrib_n "atribn tabla1",
            t2.atrib_n "atribn tabla2"
FROM tabla t1, tabla t2
WHERE t1.atrib_i = t2.atrib_j
```

## Self join

- Mostrar los nombres de las parejas de profesores que pertenecen al mismo departamento.

```
SELECT P1.Nombre || ' ' || P1.Apellido1 "Nombre1",
       P2.Nombre || ' ' || P2.Apellido1 "Nombre2"
  FROM Profesores P1, Profesores P2
 WHERE P1.departamento = P2.departamento;
```

¡¡Se empareja cada profesor consigo mismo!!

| Nombre1              | Nombre2              |
|----------------------|----------------------|
| Antonio Villanueva   | Michael Brown        |
| Juana Hernandez      | Michael Brown        |
| Miguel Hermoso       | Michael Brown        |
| Michael Brown        | Michael Brown        |
| Sergio Gálvez        | Manuel Enciso        |
| Miguel Ortiz         | Manuel Enciso        |
| Manuel Enciso        | Manuel Enciso        |
| Maria del Mar Roldán | Enrique Soler        |
| Enrique Soler        | Enrique Soler        |
| Sergio Gálvez        | Miguel Ortiz         |
| Miguel Ortiz         | Miguel Ortiz         |
| Manuel Enciso        | Miguel Ortiz         |
| Maria del Mar Roldán | Maria del Mar Roldán |
| Enrique Soler        | Maria del Mar Roldán |
| Sergio Gálvez        | Sergio Gálvez        |
| Miguel Ortiz         | Sergio Gálvez        |
|                      | ...                  |

## Self join

- Mostrar los nombres de las parejas de profesores que pertenecen al mismo departamento.

```
SELECT P1.Nombre || ' ' || P1.Apellido1 "Nombre1",
       P2.Nombre || ' ' || P2.Apellido1 "Nombre2"
  FROM Profesores P1, Profesores P2
 WHERE P1.departamento = P2.departamento
   AND P1.id != P2.id
```

¡¡Se repiten las parejas!!

| Nombre1              | Nombre2              |
|----------------------|----------------------|
| Antonio Villanueva   | Michael Brown        |
| Juana Hernandez      | Michael Brown        |
| Miguel Hermoso       | Michael Brown        |
| Sergio Gálvez        | Manuel Enciso        |
| Miguel Ortiz         | Manuel Enciso        |
| Maria del Mar Roldán | Enrique Soler        |
| Sergio Gálvez        | Miguel Ortiz         |
| Manuel Enciso        | Miguel Ortiz         |
| Enrique Soler        | Maria del Mar Roldán |
| Miguel Ortiz         | Sergio Gálvez        |
| Manuel Enciso        | Sergio Gálvez        |
| Ana Jiménez          | Carlos Fernández     |
| Carlos Fernández     | Ana Jiménez          |
| Antonio Villanueva   | Miguel Hermoso       |
| Juana Hernandez      | Miguel Hermoso       |
| Michael Brown        | Miguel Hermoso       |
| Antonio Villanueva   | Juana Hernandez      |

## Self join

- Mostrar los nombres de las parejas de profesores que pertenecen al mismo departamento.

```
SELECT P1.Nombre || ' ' || P1.Apellido1 "Nombre1",
       P2.Nombre || ' ' || P2.Apellido1 "Nombre2"
  FROM Profesores P1, Profesores P2
 WHERE P1.departamento = P2.departamento
       AND P1.id < P2.id
```

| Nombre1            | Nombre2              |
|--------------------|----------------------|
| Antonio Villanueva | Michael Brown        |
| Juana Hernandez    | Michael Brown        |
| Manuel Enciso      | Miguel Ortiz         |
| Enrique Soler      | Maria del Mar Roldán |
| Miguel Ortiz       | Sergio Gálvez        |
| Manuel Enciso      | Sergio Gálvez        |
| Ana Jiménez        | Carlos Fernández     |
| Antonio Villanueva | Miguel Hermoso       |
| Juana Hernandez    | Miguel Hermoso       |
| Michael Brown      | Miguel Hermoso       |
| Antonio Villanueva | Juana Hernandez      |

## Combinación de self Join y reunión

- Hay que tener especial cuidado para que no se produzca el producto cartesiano.
- Ej: Mostrar el nombre de las parejas de alumnos que han estado matriculados en las mismas asignaturas.

```
SELECT AL1.Nombre, AL2.Nombre
FROM Alumnos AL1, Matricular M1,
      Alumnos AL2, Matricular M2
WHERE      AL1.dni = M1.Alumno
          AND AL2.dni = M2.Alumno
          AND M1.Asignatura = M2. Asignatura
          AND AL1.dni < AL2.dni;
```

¡¡Si los alumnos coinciden en más de una asignatura salen repeticiones!!

| NOMBRE  | NOMBRE_1       |
|---------|----------------|
| Stephan | ANTONIO JESUS  |
| Stephan | SERGIO         |
| Stephan | DANIEL         |
| Stephan | RAFAEL         |
| Stephan | ADRIAN         |
| Stephan | ALFONSO        |
| Stephan | Benjamin       |
| Stephan | Sergio         |
| Stephan | Carlos Alberto |
| Stephan | Carlos Alberto |
| Stephan | Manuel         |
| Stephan | Jaime          |
| Stephan | Rafael         |
| Stephan | Rafael         |
| Stephan | Isidro         |
| Stephan | Isidro         |
|         | ...            |

## Combinación de self Join y reunión

- Si deseamos eliminar todas las repeticiones en este tipo de consultas habrá que usar DISTINCT.
- Ej: Mostrar el nombre de las parejas de alumnos que han estado matriculados en las mismas asignaturas.

```
SELECT DISTINCT AL1.Nombre, AL2.Nombre
FROM Alumnos AL1, Matricular M1,
      Alumnos AL2, Matricular M2
WHERE      AL1.dni = M1.Alumno
          AND AL2.dni = M2.Alumno
          AND M1.Asignatura = M2. Asignatura
          AND AL1.dni < AL2.dni;
```

| NOMBRE   | NOMBRE_1       |
|----------|----------------|
| Stephan  | ALFONSO        |
| Stephan  | Jose           |
| Stephan  | Arcadio        |
| Stephan  | Carlos         |
| Stephan  | PABLO          |
| Stephan  | Raziz          |
| Stephan  | Angel          |
| Stephan  | Marlon         |
| Stephan  | Esther         |
| Benjamín | RAFAEL         |
| Benjamín | ELOY           |
| Benjamín | RUBEN          |
| Benjamín | Carlos Alberto |
| Benjamín | Justo          |
| Benjamín | Laura          |
| Benjamín | ISABEL         |

## Natural Join

- Hay varias formas de especificar la operación de reunión completamente en la Cláusula FROM.
  - FROM *tabla1* NATURAL JOIN *tabla2*
  - FROM *tabla1* JOIN *tabla2* USING *atributos*
  - FROM *tabla1* JOIN *tabla2* ON *condición\_reunión*
  - FROM *tabla1* OUTER JOIN *tabla2* ON *condición\_reunión*
- Siempre habrá una consulta SQL equivalente especificando la condición de reunión en el WHERE.
- La reunión natural (**NATURAL JOIN**) realiza la reunión utilizando todos los atributos comunes (columnas con el mismo nombre) de las dos tablas.
  - Por tanto, en el resultado no habrá columnas repetidas (innecesarios los alias de tabla)

## Natural Join

- Mostrar un listado donde a cada alumno del curso 15/16 se le añade el identificador de cada profesor que le da clase.

```
SELECT M.Alumno, I.Profesor
FROM impartir I, matricular M
WHERE I.Asignatura = M.asignatura
      AND I.Grupo = M.Grupo
      AND I.Curso = M.Curso
      AND I.Curso = '15/16';
```

Condición  
de reunión

| ALUMNO   | PROFESOR    |
|----------|-------------|
| 12312342 | G-34-TEU-96 |
| 12312342 | C-34-TU-00  |
| 12312342 | C-67-CEU-98 |
| 25748955 | G-34-TEU-96 |
| 25748955 | C-34-TU-00  |
| 34234244 | D-92-PC-92  |
| 34234444 | A-89-CEU-99 |
| 45634322 | C-67-CEU-98 |
| 46563277 | D-92-PC-92  |
| 46563277 | C-67-CEU-98 |
| 54354355 | G-34-TEU-96 |
| 54354355 | C-34-TU-00  |
| 67867433 | D-92-PC-92  |
| 67867888 | C-67-CEU-98 |
| 23456456 | D-92-PC-92  |

```
SELECT Alumno, Profesor
FROM impartir NATURAL JOIN matricular
WHERE Curso = '15/16';
```

# Natural Join

- **Problema:** se utilizan TODOS los atributos con nombre idéntico.  
A veces no interesa reunir utilizando todos los atributos, sino sólo alguno de ellos.
- **Soluciones:**
  1. Utilizar otro tipo de reunión.
  2. Crear tablas temporales en el FROM.
- Efecto secundario de la solución 2:
  - determinación de orden de ejecución.

- Suponiendo que el primer apellido de profesores y alumnos fuera único, mostrar nombre y fecha de matrícula de los profesores que son también alumnos

```
SELECT P.Nombre, Fecha_prim_matricula
FROM profesores P, alumnos A
WHERE P.apellido1 = A.apellido1 ;
```

| NOMBRE  | FECHA_PRIM_MATRICULA |
|---------|----------------------|
| Antonio | 30/08/15             |
| Antonio | 30/08/14             |
| Antonio | 10/09/13             |
| Miguel  | 05/09/15             |
| Antonio | 11/09/13             |

```
SELECT Nombre, Fecha_prim_matricula
FROM profesores NATURAL JOIN
      (alumnos sin atributos comunes excepto apellido1);
```

```
SELECT Nombre, Fecha_prim_matricula
FROM profesores NATURAL JOIN
      (SELECT Apellido1, Fecha_prim_matricula FROM Alumnos);
```

## Join Using

- Operación de reunión que permite especificar los atributos **comunes** que se utilizarán en las condiciones de reunión.

FROM *tabla1* JOIN *tabla2* USING ( *atributos\_seleccionados* )

- *atributos\_seleccionados*:  
    atributo\_común1 , atributo\_común2 , ...
- El resto de atributos comunes no seleccionados se tratan como atributos normales y no se utilizan en la condición de reunión
- Los alias de tabla serán necesarios si hay atributos comunes no seleccionados con USING que queremos utilizar en la proyección.

## Join Using

- Suponiendo que el primer apellido de profesores y alumnos fuera único, mostrar nombre y fecha de matrícula de los profesores que son también alumnos

```
SELECT P.Nombre, A.Fecha_prim_matricula  
FROM Profesores P JOIN Alumnos A USING (Apellido1);
```

## Conditional Join

- **Problema:** si los atributos de reunión no se llaman igual, no podemos utilizar NATURAL JOIN ni JOIN USING.
- **Solución:** especificar la condición de reunión que queremos, indicando los atributos que se solapan en las dos tablas.

`FROM tabla1 JOIN tabla2 ON condición_reunión`

## Conditional Join

- Mostrar el numero de tramos de investigación del profesor Enrique Soler

```
SELECT I.Tramos
FROM Profesores P, Investigadores I
WHERE P.Id = I.Id_profesor
      AND upper(P.Nombre) = 'ENRIQUE'
      AND upper(P.Apellido1) = 'SOLER';
```

|                                                                                            |
|--------------------------------------------------------------------------------------------|
|  TRAMOS |
| 2                                                                                          |

```
SELECT I.Tramos
FROM Profesores P JOIN Investigadores I ON P.Id = I.Id_profesor
WHERE upper(P.Nombre) = 'ENRIQUE'
      AND upper(P.Apellido1) = 'SOLER' ;
```

# Ejercicios

- Mostrar ...
  1. Nombre y nota de los alumnos que han aprobado Bases de Datos.
  2. Nombre del alumno junto a nombre de asignatura y turno en el que está matriculado.
  3. Nombre de alumno, nombre de asignatura aprobada, la nota en ella y el nombre del profesor de la asignatura.

## Outer Join

- La reunión externa (**OUTER JOIN**) de dos tablas  $A$  y  $B$  es una operación de reunión que devuelve todas las filas que cumplen la condición de reunión (como un JOIN normal) y, además, aquellas filas de la tabla  $A$  y/o la tabla  $B$  que no cumplen la condición de reunión.

FROM *tabla1* [LEFT/RIGHT/FULL] **OUTER JOIN** *tabla2* ON *condición\_reunión*

- $A$  **LEFT OUTER JOIN**  $B$  añade las filas de  $A$  que no cumplen *condición\_reunión*
- $A$  **RIGHT OUTER JOIN**  $B$  añade las filas de  $B$  que no cumplen *condición\_reunión*
- $A$  **FULL OUTER JOIN**  $B$  añade las filas de  $A$  y  $B$  que no cumplen *condición\_reunión*.

## Outer Join

- Hacer un listado de **todas** las asignaturas, que muestre el nombre de asignatura junto al nombre de la materia asociada

```
SELECT A.Nombre "ASIGNATURA", M.Nombre "MATERIA"  
FROM Asignaturas A, Materias M  
WHERE A.Cod_materia = M.Codigo;
```

| ASIGNATURA                       | MATERIA                       |
|----------------------------------|-------------------------------|
| Programación Orientada a Objetos | Ingeniería del Software       |
| Ingeniería Web                   | Ingeniería del Software       |
| Administración de Bases de Datos | Ingeniería del Software       |
| Bases de Datos                   | Ingeniería del Software       |
| Estructura de Computadores       | Dispositivos hardware         |
| Computación Altas Prestaciones   | Dispositivos hardware         |
| Dispositivos Electronicos        | Dispositivos hardware         |
| Prácticas en empresa             | Complementos de formación     |
| Estadística                      | Fundamentos de la Informática |
| Calculo Numérico                 | Fundamentos de la Informática |
| Matemática Discreta              | Fundamentos de la Informática |
| Lógica Computacional             | Fundamentos de la Informática |

Las asignaturas que no tienen materia asignada no salen en el listado

## Outer Join

- Hacer un listado de **todas** las asignaturas, que muestre el nombre de asignatura junto al nombre de la materia asociada

```
SELECT A.Nombre "ASIGNATURA", M.Nombre "MATERIA"  
FROM Asignaturas A LEFT OUTER JOIN Materias M  
ON (A.Cod_materia = M.Codigo);
```

Para las filas añadidas,  
los atributos de la otra  
tabla se asignan a **null**

| ASIGNATURA                       | MATERIA                       |
|----------------------------------|-------------------------------|
| Programación Orientada a Objetos | Ingeniería del Software       |
| Ingeniería Web                   | Ingeniería del Software       |
| Bases de Datos                   | Ingeniería del Software       |
| Administración de Bases de Datos | Ingeniería del Software       |
| Computación Altas Prestaciones   | Dispositivos hardware         |
| Dispositivos Electronicos        | Dispositivos hardware         |
| Estructura de Computadores       | Dispositivos hardware         |
| Prácticas en empresa             | Complementos de formación     |
| Matemática Discreta              | Fundamentos de la Informática |
| Estadística                      | Fundamentos de la Informática |
| Logica Computacional             | Fundamentos de la Informática |
| Calculo Numerico                 | Fundamentos de la Informática |
| Modelos Computacionales          | (null)                        |
| Sistemas Operativos              | (null)                        |
| Teoria de Automatas              | (null)                        |
| Teoria de la señal               | (null)                        |

## Outer Join

- Mostrar el código de alumno junto al código de profesor que le da clase de todos los alumnos y todos los profesores de las asignaturas 112 y 114.

```
SELECT I.Profesor, M.Alumno
FROM   Impartir I  FULL OUTER JOIN  Matricular M
       ON (I.Asignatura = M.Asignatura AND
           I.Grupo = M.Grupo AND
           I.Curso = M.Curso)
WHERE  nvl(I.Asignatura,0) in (112,114) OR
       nvl(M.Asignatura,0) in (112,114)
```

- Salen
  - Los profesores de las asignaturas sin grupo asignado y/o sin alumnos en sus grupos.
  - Los alumnos matriculados en las asignaturas sin grupo asignado y/o sin profesor asignado.

| PROFESOR    | ALUMNO   |
|-------------|----------|
| C-34-TU-00  | 12312342 |
| G-34-TEU-96 | 12312342 |
| D-92-PC-92  | 23423332 |
| (null)      | 23423332 |
| A-89-CEU-99 | 23423424 |
| (null)      | 23456456 |
| D-92-PC-92  | 25744665 |
| C-34-TU-00  | 25748955 |
| G-34-TEU-96 | 25748955 |
| (null)      | 25756456 |
| A-89-CEU-99 | 34234244 |
| (null)      | 34234444 |
| (null)      | 34545346 |
| C-34-TU-00  | 34558955 |
| C-34-TU-00  | 34567567 |
| D-92-PC-92  | 45345311 |
| (null)      | 45345311 |
|             | ...      |

## Ejercicio

1. Mostrar todos los datos de TODOS los profesores junto a los datos de sus directores de tesis, en caso de tenerlos.

# **SQL El lenguaje de Consulta Estructurado**

Subconsultas,  
Consultas Negativas



# Subconsultas

- Una **subconsulta** es una sentencia **SELECT** que aparece **dentro** de otra sentencia **SELECT** que llamaremos consulta principal.
- Se puede encontrar en la lista de selección, en la cláusula **WHERE** o en la cláusula **HAVING** de la consulta principal.
- Una subconsulta tiene la misma sintaxis que una sentencia **SELECT** normal exceptuando que aparece **encerrada entre paréntesis**, no puede contener la cláusula **ORDER BY** y tiene algunas restricciones en cuanto a número de columnas según el lugar donde aparece en la consulta principal.

# Subconsultas

- Listar el nombre de las asignaturas que tienen más créditos que la asignatura llamada 'Bases de Datos'.

```
SELECT Nombre
FROM Asignaturas
WHERE creditos > (SELECT creditos FROM Asignaturas
                     WHERE upper(Nombre) = 'BASES DE DATOS');
```

| NOMBRE               |
|----------------------|
| Prácticas en empresa |

## Consultas anidadas

- La subconsulta se ejecuta una vez por cada fila de la consulta principal.
- La subconsulta puede utilizar todos los atributos de las tablas que aparecen en el FROM de la consulta principal.
  - Imprescindible un alias de tabla.
- Las consultas anidadas imponen una cierta estructura (y orden de ejecución) a las consultas.
  - Ventaja: Suelen ser más fáciles de interpretar por el usuario, si el nivel de anidamiento es limitado.
  - Desventaja: más lentas de ejecutar.

## Consultas Anidadas

- Listar el nombre de las asignaturas que tienen mas créditos que las asignaturas de segundo curso.

```
SELECT Nombre  
FROM Asignaturas  
WHERE creditos > (SELECT creditos FROM Asignaturas  
WHERE curso = 2 );
```

Error: single-row subquery  
returns more than one row

El resultado de la  
subconsulta devuelve  
más de una tupla

| CREDITOS |
|----------|
| 6        |
| 6        |
| 6        |
| 4        |
| 4        |
| 6        |

## ANY / ALL

- Cuando las subconsultas que se utilizan en una expresión de comparación devuelven más de una tupla hay que combinar los operadores de comparación (<, >, =, !=,...) con ANY / ALL
- Operador **ANY** Subconsulta. Será verdad si algún valor del resultado de la subconsulta hace cierta la comparación.
- Operador **ALL** Subconsulta. Será verdad si todos los valores del resultado de la subconsulta hacen cierta la comparación.

## Consultas Anidadas

- Listar el nombre de las asignaturas que tienen mas créditos que **ALGUNA DE** las asignaturas de segundo curso.

```
SELECT Nombre
FROM Asignaturas
WHERE creditos > ANY (SELECT creditos FROM Asignaturas
                        WHERE curso = 2 ) ;
```

| NOMBRE                           |
|----------------------------------|
| Prácticas en empresa             |
| Programación Orientada a Objetos |
| Bases de Datos                   |
| Teoría de la señal               |
| Lógica Computacional             |
| Matemática Discreta              |
| Sistemas Operativos              |
| Estructura de Computadores       |
| Calculo Numérico                 |
| Dispositivos Electrónicos        |
| Modelos Computacionales          |
| Ingeniería Web                   |
| Teoría de Automatas              |

## Consultas Anidadas

- Listar el nombre de las asignaturas que tienen mas créditos que **TODAS** las asignaturas de segundo curso.

```
SELECT Nombre
FROM Asignaturas
WHERE creditos > ALL (SELECT creditos FROM Asignaturas
                      WHERE curso = 2 ) ;
```

| NOMBRE                           |
|----------------------------------|
| Bases de Datos                   |
| Programación Orientada a Objetos |
| Prácticas en empresa             |

## IN / EXISTS

- Los operadores IN y EXISTS se aplican a conjuntos de tuplas.
- *elem IN conjunto\_tuplas*
  - La expresión es cierta si el elemento pertenece al conjunto dado.
- **EXISTS** *conjunto\_tuplas*
  - La expresión es cierta si el conjunto de tuplas no está vacío. Es decir, existe al menos una tupla en el conjunto.
- Se pueden utilizar con el resultado de subconsultas para emular las reuniones.

## IN / EXISTS

- Mostrar el código de las asignaturas que imparte el profesor Manuel Enciso

```
SELECT I.Asignatura "Codigo Asignatura"  
FROM profesores P, Impartir I  
WHERE I.profesor = P.id  
      AND upper(P.Nombre) = 'MANUEL'  
      AND upper(P.Apellido1) = 'ENCISO';
```

| Codigo Asignatura |
|-------------------|
| 112               |
| 140               |

Se obtiene el id de  
Manuel Enciso

```
SELECT Asignatura "Codigo Asignatura"  
FROM Impartir  
WHERE profesor IN (SELECT id FROM Profesores  
                      WHERE upper(Nombre) = 'MANUEL'  
                      AND upper(Apellido1) = 'ENCISO' );
```

Se obtienen los  
profesores con ese id

## IN / EXISTS

- Mostrar el código de las asignaturas que imparte el profesor Manuel Enciso

```
SELECT I.Asignatura "Codigo Asignatura"  
FROM profesores P, Impartir I  
WHERE I.profesor = P.id  
      AND upper(P.Nombre) = 'MANUEL'  
      AND upper(P.Apellido1) = 'ENCISO';
```

| Codigo Asignatura |
|-------------------|
| 112               |
| 140               |

```
SELECT I.Asignatura "Codigo Asignatura"  
FROM Impartir I  
WHERE EXISTS (SELECT * FROM Profesores P  
                  WHERE I.profesor = P.Id  
                  AND upper(P.Nombre) = 'MANUEL'  
                  AND upper(P.Apellido1) = 'ENCISO');
```

Se obtienen las asignaturas para las que existe un profesor llamado Manuel Enciso que la imparte (**I.profesor** = **P.id**)

## ¿Qué devuelven las siguientes consultas?

```
SELECT Asignatura "Codigo Asignatura"  
FROM Impartir  
WHERE EXISTS (SELECT *  
              FROM Profesores , Impartir  
              WHERE profesor = id  
              AND upper(nombre) = 'MANUEL'  
              AND upper(apellido1) = 'ENCISO') ;
```

```
SELECT Asignatura "Codigo Asignatura"  
FROM Impartir  
WHERE EXISTS (SELECT *  
              FROM Profesores , Impartir  
              WHERE profesor = id  
              AND upper(nombre) = 'PEDRO'  
              AND upper(apellido1) = 'ENCISO') ;
```

## Condición de reunión compuesta

- Mostrar el código de los alumnos que reciben clase del profesor de código C-34-TU-00

```
SELECT M.Alumno "Codigo Alumno"  
FROM Matricular M, Impartir I  
WHERE I.Asignatura=M.Asignatura  
      AND I.Grupo=M.Grupo  
      AND I.Curso=M.Curso  
      AND I.Profesor = 'C-34-TU-00';
```

```
SELECT Alumno "Codigo Alumno"  
FROM Matricular  
WHERE (Asignatura, Grupo, Curso) IN  
      (SELECT Asignatura, Grupo, Curso  
       FROM Impartir  
       WHERE Profesor = 'C-34-TU-00') ;
```

| Codigo Alumno |
|---------------|
| 12312342      |
| 25748955      |
| 54354355      |

Se obtienen los alumnos de esas asignaturas

Se obtienen las asignaturas impartidas por el profesor

## Condición de reunión compuesta

- Mostrar el código de los alumnos que reciben clase del profesor de código C-34-TU-00

```
SELECT M.Alumno "Codigo Alumno"  
FROM Matricular M, Impartir I  
WHERE I.Asignatura=M.Asignatura  
      AND I.Grupo=M.Grupo  
      AND I.Curso=M.Curso  
      AND I.Profesor = 'C-34-TU-00';
```

| Codigo Alumno |
|---------------|
| 12312342      |
| 25748955      |
| 54354355      |

```
SELECT Alumno "Codigo Alumno" FROM Matricular M  
WHERE EXISTS (SELECT *  
               FROM Impartir I  
               WHERE I.Asignatura=M.Asignatura  
                     AND I.Grupo=M.Grupo  
                     AND I.Curso=M.Curso  
                     AND I.Profesor = 'C-34-TU-00') ;
```

## Condición de reunión compuesta

- Mostrar el código de los alumnos que reciben clase del profesor de código C-34-TU-00

```
SELECT M.Alumno "Codigo Alumno"  
FROM Matricular M NATURAL JOIN Impartir I  
WHERE I.Profesor = 'C-34-TU-00';
```

| Codigo Alumno |
|---------------|
| 12312342      |
| 25748955      |
| 54354355      |

```
SELECT Alumno "Codigo Alumno" FROM Matricular M  
WHERE EXISTS (SELECT *  
                FROM Impartir I  
                WHERE I.Asignatura=M.Asignatura  
                AND I.Grupo=M.Grupo  
                AND I.Curso=M.Curso  
                AND I.Profesor = 'C-34-TU-00') ;
```

Alumnos matriculados en asignaturas en las que existe un profesor que las imparte (I.Asignatura=M.Asignatura & I.curso=M.Curso & I.Grupo=M.Grupo) con código dado

# Consultas Negativas

- El operador NOT en combinación con los operadores IN y EXISTS se puede utilizar para comprobar exhaustivamente que en todas las tuplas no se da una condición.
- *elem NOT IN conjunto\_tuplas*
  - La expresión es cierta si el elemento no pertenece al conjunto. Es decir, el elemento no es ninguno de los que aparecen en el conjunto.
- **NOT EXISTS** *conjunto\_tuplas*
  - La expresión es cierta si el conjunto de tupas está vacío. Es decir, no existe elemento en el conjunto de tuplas.

## Consultas negativas

- Mostrar los datos de los profesores que no imparten la asignatura 112.

```
SELECT * FROM PROFESORES  
WHERE id IN (SELECT PROFESOR FROM IMPARTIR  
              WHERE ASIGNATURA != 112);
```

```
SELECT profesor  
FROM impartir  
WHERE asignatura = 112;
```

¡¡ Salen profesores que no deberían salir!!  
Son aquellos que imparten otra asignatura  
además de la 112

| PROFESOR    |
|-------------|
| A-89-CEU-99 |
| C-34-TU-00  |
| G-34-TEU-96 |

| ID          | NOMBRE        | APELLIDO1 |
|-------------|---------------|-----------|
| A-89-CEU-99 | Manuel        | Enciso    |
| C-34-TU-00  | Enrique       | Soler     |
| C-67-CEU-98 | Miguel        | Ortiz     |
| D-92-PC-92  | Maria del Mar | Roldán    |
| F-12-AY-02  | Sergio        | Gálvez    |
| G-34-TEU-96 | Ana           | Jiménez   |
| H-32-TU-99  | Miguel        | Hermoso   |
| H-77-TU-99  | Juana         | Hernandez |
| ...         |               |           |

- Condición positiva: una tupla tiene un valor diferente al dado.

## Consultas negativas

- Mostrar los datos de los profesores que no imparten la asignatura 112.

```
SELECT * FROM profesores  
WHERE Id NOT IN (SELECT Profesor FROM impartir  
WHERE Asignatura = 112);
```

```
SELECT * FROM profesores P  
WHERE NOT EXISTS  
(SELECT Profesor FROM impartir I  
WHERE I.profesor = P.Id  
AND I.Asignatura = 112);
```

Profesores para los que no existe un profesor que imparte clase en la asignatura 112 que tiene su mismo código (I.profesor = P.Id)

# Ejercicios

- Mostrar ...
1. Código de las asignaturas que imparten los profesores que imparten la asignatura 112. Se deben eliminar las repeticiones.
  2. Nombre de los departamentos que no tienen ninguna asignatura con más de 6 créditos.
  3. Dni de alumnos matriculados en alguna asignatura impartida por el profesor de mayor antigüedad.
  4. Mostrar las parejas de profesores que no tienen ningún alumno en común.

# **SQL El lenguaje de Consulta Estructurado**

Funciones de agregación,  
Anidamiento avanzado



# Funciones de agregación

- Las funciones de agregación operan sobre un conjunto de tuplas y devuelven un único valor que será el resultado de dicha operación.
- Funciones:
  - **SUM** (expr): Suma los valores
  - **MIN** (expr) : Calcula el valor mínimo
  - **MAX** (expr) : Calcula el valor máximo
  - **COUNT** (expr) / **COUNT(\*)** : Cuenta el número de tuplas
  - **AVG** (expr) : Calcula la media aritmética.
    - $\text{AVG} (\text{expr}) = \text{sum} (\text{expr}) / \text{count(expr)}$
- Se pueden utilizar para calcular información que está almacenada de forma implícita en la base de datos.

# Funciones de agregación

- La mayoría, excepto COUNT(\*), ignoran los valores NULL.
  - Se puede usar la función NVL para forzar que se considere los valores NULL como un valor determinado.
- Se puede utilizar DISTINCT en la expresión que se le pasa como parámetro a las funciones de agregación para tener en cuenta sólo los valores distintos.
- Las funciones de agregación devuelven NULL si no hay filas en la tabla necesaria para resolver la expresión o todas contienen NULL.
  - COUNT nunca devuelve NULL.

# Funciones de agregación

- Contar cuántos teléfonos distintos de profesores hay en la base de datos.

```
SELECT COUNT(DISTINCT telefono) FROM profesores;
```

COUNT(DISTINCTTELEFONO)  
5

- Contar cuántos teléfonos no nulos de profesores hay en la base de datos

```
SELECT COUNT(telefono) FROM profesores;
```

COUNT(TELEFONO)  
7

- Contar cuántos profesores hay en la base de datos

```
SELECT COUNT(*) FROM profesores;
```

COUNT(\*)  
11

# Funciones de agregación

- Mostrar el número de alumnos que se han matriculado en la asignatura ‘Base de Datos’ y la media de todas las notas de la asignatura, ignorando las calificaciones nulas.

```
SELECT COUNT(*) "Matriculados" ,  
       AVG(decode(calificacion,'MH',10,'SB',9,'NT',7,'AP',5,'SP', 0)) "Media"  
FROM matricular M, asignaturas A  
WHERE M.asignatura = A.codigo  
AND upper(A.nombre) = 'BASES DE DATOS' ;
```

| Matriculados | Media |
|--------------|-------|
| 32           | 2,5   |

- ¿Qué habría que modificar si quisiéramos contar sólo los alumnos cuya calificación no es nula?
- ¿Qué habría que modificar si quisiéramos tener en cuenta las calificaciones nulas en la media?

## GROUP BY

- La cláusula **GROUP BY** permite dividir las tuplas seleccionadas para el resultado por la cláusula WHERE en conjuntos de tuplas disjuntas, sobre cada uno de los cuales se va a aplicar funciones de agregación.

```
SELECT Agregación1, Agregación2, ...
  FROM tablas
 WHERE predicado
 GROUP BY expr1, expr2, ..., exprn
 ORDER BY criterios_ordenación
```

- Se creará un grupo de tuplas por cada valor diferente de (*expr1,expr2,...,exprn*)
- La consulta aplica las funciones de agregación *Agregación1, Agregación2,..* a cada grupo, obteniéndose una tupla resultado para cada uno.

## GROUP BY

- Mostrar el primer apellido más “pequeño” (en sentido lexicográfico) de los profesores de cada departamento.

```
SELECT MIN(apellido1) "Menor apellido", departamento
FROM profesores
GROUP BY departamento;
```

| DEPARTAMENTO    | NOMBRE     | APELLIDO1 | ... |
|-----------------|------------|-----------|-----|
| 4 Michael       | Brown      | ...       |     |
| 1 Manuel        | Enciso     | ...       |     |
| 2 Enrique       | Soler      | ...       |     |
| 1 Miguel        | Ortiz      | ...       |     |
| 2 Maria del Mar | Roldán     | ...       |     |
| 1 Sergio        | Gálvez     | ...       |     |
| 3 Carlos        | Fernández  | ...       |     |
| 3 Ana           | Jiménez    | ...       |     |
| 4 Miguel        | Hermoso    | ...       |     |
| 4 Juana         | Hernandez  | ...       |     |
| 4 Antonio       | Villanueva | ...       |     |

GROUP BY



## GROUP BY

- Queremos agrupar a los profesores por decenios (el decenio 2 lo forman los profesores con 20-29 años, el decenio 3, profesores con 30-39, ...). Necesitamos un listado ordenado por decenio, en el que se muestre el número de decenio y la fecha de nacimiento del profesor más joven del decenio.

```
SELECT trunc(months_between(sysdate,fecha_nacimiento)/12/10) "Decenio",
       MAX(fecha_nacimiento) "Fecha"
  FROM profesores
 GROUP BY trunc(months_between(sysdate,fecha_nacimiento)/12/10)
 ORDER BY trunc(months_between(sysdate,fecha_nacimiento)/12/10);
```

| Decenio | Fecha    |
|---------|----------|
| 2       | 14/09/92 |
| 3       | 21/12/80 |
| 4       | 01/01/68 |
| 6       | 01/08/49 |
| 7       | 24/01/42 |
| 8       | 01/08/31 |
| (null)  | (null)   |

```
SELECT trunc(months_between(sysdate,fecha_nacimiento)/12/10) "Decenio",
       MAX(fecha_nacimiento) "Fecha"
  FROM profesores
 GROUP BY trunc(months_between(sysdate,fecha_nacimiento)/12/10)
 ORDER BY "Decenio"
```

## GROUP BY

- Cuando se utiliza agrupación (GROUP BY) y/o funciones de agregación, en la lista de selección **SÓLO** podemos utilizar expresiones que tengan funciones de agregación
  - Se pueden usar expresiones que no utilicen funciones de agregación si son exactamente las expresiones que aparecen en el GROUP BY (ver ejemplo anterior)
- Ej:

```
SELECT MIN(apellido1) "Menor apellido", nombre  
FROM profesores  
GROUP BY departamento;
```

Error "not a GROUP BY expression"

- Si las tuplas se agrupan por valor de departamento, ¿cuál de todos los nombres de profesores de cada departamento tomamos para incluirlo en la única tupla resultado para ese departamento?

# HAVING

- La cláusula **HAVING** permite filtrar los grupos que no satisfagan un determinado predicado.

```
SELECT Agregación1, Agregación2, ...
FROM tablas
WHERE predicado_filtrar_filas
GROUP BY expr1, expr2, ..., exprn
HAVING predicado_filtrar_grupos
ORDER BY criterios_ordenación
```

- El predicado de la cláusula WHERE se utiliza para eliminar filas que no se usarán en el resultado.
- La cláusula GROUP BY distribuye las filas que quedan en grupos.
- La cláusula HAVING se asegura de que los grupos que se usarán en la salida hacen verdad el predicado\_filtrar\_grupos.

## HAVING

- Listar el código, la nota media y el número de alumnos matriculados en cada una de las asignaturas, pero sólo de las asignaturas que tienen un grupo 'B' y más de 10 alumnos.

```
SELECT asignatura, COUNT(*) "Matriculados",
       round(AVG(decode(calificacion,'MH',10,'SB',9,'NT',7,'AP',5,'SP',0)),2) "Media"
  FROM Matricular
 WHERE Grupo = 'B'
 GROUP BY asignatura
 HAVING COUNT(*) > 10 ;
```

| ASIGNATURA | Matriculados | Media |
|------------|--------------|-------|
| 144        | 25           | 2,33  |
| 113        | 23           | 2,67  |
| 116        | 11           | 2     |
| 112        | 19           | 1,15  |
| 110        | 12           | 5,75  |
| 114        | 15           | 1,75  |
| 115        | 14           | 1,5   |

## HAVING

- Las funciones de agregación **sólo** pueden utilizarse en la lista de selección, teniendo en cuenta las limitaciones ya comentadas, en la cláusula HAVING y en la cláusula ORDER BY.
- Ej: Listar el código, la nota media y el número de alumnos matriculados en cada una de las asignaturas, pero sólo de las asignaturas que tienen un grupo 'B' y más de 10 alumnos. *El resultado ha de estar ordenado por número de alumnos matriculados.*

```
SELECT asignatura, COUNT(*) "Matriculados",
       round(AVG(decode(calificacion,'MH',10,'SB',9,'NT',7,'AP',5,'SP',0)),2) "Media"
  FROM Matricular
 WHERE Grupo = 'B'
 GROUP BY asignatura
 HAVING COUNT(*) > 10
 ORDER BY COUNT(*);
```

## Orden de Ejecución de la sentencia SELECT

- 5.** SELECT lista\_selección
- 1.** FROM tablas
- 2.** WHERE predicado\_filtrar\_filas
- 3.** GROUP BY expr1, . . . , exprk
- 4.** HAVING predicado\_filtrar\_grupos
- 6.** ORDER BY criterios\_ordenación

## Ejercicios

1. Se desea conocer si los alumnos o las alumnas del curso '15/16' son más estudiados. Para ello se necesita un listado con la nota media de los alumnos de dicho curso divididos por género.
2. Calcular la media de cada asignatura donde la nota mínima sea superior a 4.
3. Listado con el código de profesor, total de créditos que imparte y código de departamento al que pertenece ordenado por departamento. Queremos que salgan primero todos los profesores del departamento 4, después del 3, ...
4. Para los municipios de la provincia de Málaga, queremos agruparlos por primera letra de su nombre. Para cada grupo queremos calcular el total de habitantes de los municipios del grupo, el número de habitantes del municipio con menos habitantes del grupo y el número de habitantes del municipio con más habitantes del grupo.
5. ¿Qué habría que modificar si queremos visualizar sólo los grupos con menos de 10000 habitantes?

## Anidamiento Avanzado

- El resultado de una sentencia SELECT sigue un esquema relacional.
- Dicho resultado se puede usar como una relación. Ya lo hemos hecho con las operaciones de conjunto (UNION, MINUS,...) y con las subconsultas.
- ¿Por qué no usarla en la cláusula **FROM**?
- Ventajas del anidamiento:
  - Estructurar la consulta.
  - Adaptar el orden de ejecución de la consulta a nuestras necesidades.

## Anidamiento Avanzado

- Obtener toda la información de las dos asignaturas con más créditos.

```
SELECT *
FROM asignaturas
WHERE rownum <= 2 AND creditos IS NOT NULL
ORDER BY creditos DESC;
```

| CODIGO | NOMBRE               | DEPARTAMENTO | CREDITOS | ... |
|--------|----------------------|--------------|----------|-----|
| 140    | Prácticas en empresa | 1            | 9        | ... |
| 200    | Teoria de la señal   | 4            | 6        | ... |

Problema: ROWNUM (cláusula WHERE) se calcula antes del ORDER BY

La consulta extrae información de dos asignaturas y después las ordena.

## Anidamiento Avanzado

- Obtener toda la información de las dos asignaturas con más créditos.

```
SELECT *
FROM ( SELECT * FROM asignaturas
        WHERE creditos IS NOT NULL
        ORDER BY creditos DESC )
WHERE rownum <= 2;
```

| CODIGO | NOMBRE               | DEPARTAMENTO | CREDITOS | ... |
|--------|----------------------|--------------|----------|-----|
| 140    | Prácticas en empresa | 1            | 9        | ... |
| 112    | Bases de Datos       | 1            | 7        | ... |

En este caso se ordenan las asignaturas y después se seleccionan las dos primeras.

## Anidamiento Avanzado

- Obtenga toda la información de las asignaturas, pero sólo de aquellas en las que haya exactamente 25 alumnos matriculados.

```
SELECT asig.*  
FROM asignaturas asig  
WHERE (25, asig.codigo) IN (  
    SELECT COUNT (DISTINCT alumno), asignatura  
    FROM matricular  
    GROUP BY asignatura)
```

| CODIGO | NOMBRE                  | DEPARTAMENTO | CREDITOS | TEORICOS | PRACTICOS | CARACTER | CURSO | WEB                    | COD_MATERIA |
|--------|-------------------------|--------------|----------|----------|-----------|----------|-------|------------------------|-------------|
| 113    | Calculo Numerico        |              | 2        | 6        | 3         | 3 TR     | 2     | www.fermat.uma.es      | 4           |
| 144    | Modelos Computacionales |              | 1        | 6        | 5         | 1 TR     | 3     | (null)                 | (null)      |
| 112    | Bases de Datos          |              | 1        | 7        | 5         | 3 TR     | 3     | www.lcc.uma.es/~enciso | 1           |

¿ Y si quisiéramos que saliera también el número de alumnos matriculados en esas asignaturas?

## Anidamiento Avanzado

- Obtenga toda la información de las asignaturas junto al número de alumnos matriculados en ellas, pero sólo de aquellas en las que haya exactamente 25 alumnos matriculados.

```
SELECT asig.*, 25 "Matriculados"
FROM asignaturas asig
WHERE (25, asig.codigo) IN (
    SELECT COUNT (DISTINCT alumno), asignatura
    FROM matricular
    GROUP BY asignatura)
```

| CODIGO | NOMBRE                  | DEPARTAMENTO | CREDITOS | TEORICOS | PRACTICOS | CARA... | CURSO | WEB                    | COD_MATERIA | Matridulados |
|--------|-------------------------|--------------|----------|----------|-----------|---------|-------|------------------------|-------------|--------------|
| 113    | Calculo Numerico        |              | 2        | 6        | 3         | 3 TR    | 2     | www.fermat.uma.es      | 4           | 25           |
| 144    | Modelos Computacionales |              | 1        | 6        | 5         | 1 TR    | 3     | (null)                 | (null)      | 25           |
| 112    | Bases de Datos          |              | 1        | 7        | 5         | 3 TR    | 3     | www.lcc.uma.es/~enciso | 1           | 25           |

## Anidamiento Avanzado

- Obtenga la información de las asignaturas, pero sólo de aquellas en las que haya al menos 25 alumnos matriculados.

```
SELECT asig.*  
FROM asignaturas asig  
WHERE 25 >= (SELECT COUNT (DISTINCT alumno)  
               FROM matricular  
               WHERE asig.codigo = asignatura)
```

```
SELECT asig.*  
FROM asignaturas asig  
WHERE codigo IN  
      (SELECT asignatura FROM matricular  
       GROUP BY asignatura  
       HAVING COUNT (DISTINCT alumno) >=25)
```

| CODIGO | NOMBRE                  |     |
|--------|-------------------------|-----|
| 113    | Calculo Numerico        | ... |
| 144    | Modelos Computacionales | ... |
| 116    | Logica Computacional    | ... |
| 112    | Bases de Datos          | ... |
| 200    | Teoria de la señal      | ... |

## Anidamiento Avanzado

- ¿Y si queremos añadir el número de alumnos de cada asignatura en la consulta anterior?

```
SELECT asig.*, numero
FROM asignaturas asig JOIN
    (SELECT COUNT (DISTINCT alumno) as numero, asignatura
     FROM matricular
     GROUP BY asignatura)
    ON (asig.codigo = asignatura)
WHERE numero >=25
```

| CODIGO | NOMBRE                  | NUMERO | ... |
|--------|-------------------------|--------|-----|
| 113    | Calculo Numerico        | 25     | ... |
| 144    | Modelos Computacionales | 25     | ... |
| 116    | Logica Computacional    | 29     | ... |
| 112    | Bases de Datos          | 25     | ... |
| 200    | Teoria de la señal      | 87     | ... |



Se asigna un identificador al resultado de la expresión

## Ejercicios

1. Mostrar los datos del profesor más antiguo de cada departamento.
2. Queremos dar un premio a los alumnos más estudiados. Para ello se debe mostrar un listado con los tres alumnos (sólo 3) que hayan aprobado más asignaturas. Para cada alumno se ha de mostrar su dni, el número de asignaturas aprobadas y la nota media de las asignaturas aprobadas. Se considerará que la calificación ‘MH’ equivale a 10 puntos, el ‘SB’ a 9, el ‘NT’ a 7 y el ‘AP’ a 5. La nota media estará redondeada a dos decimales.
3. Obtener un informe con 4 columnas que muestre el dni, nombre completo, email y edad (en años) de los alumnos matriculados en la asignatura que tiene más alumnos matriculados.
4. La tasa de rendimiento de una titulación es la razón entre el número de créditos aprobados por los alumnos y el número de créditos en los que se han matriculado los alumnos. Se desea calcular dicha tasa para cada curso académico.

# **SQL El lenguaje de Consulta Estructurado**

Operaciones de Manipulación,  
Vistas, Metadatos



# Operaciones de Manipulación

- SQL proporciona operaciones para manipular la información contenida en las tablas:
  - **INSERT**: permite insertar nuevas filas en las tablas.
  - **UPDATE**: modifica la información de filas existentes
  - **DELETE**: borra filas de las tablas
- También permite definir **transacciones**, que son conjuntos de operaciones de manipulación que se tratan atómicamente, como si fueran una única operación. Las operaciones pertenecientes a una transacción se ejecutan todas, o bien, no se ejecuta ninguna, es decir, se deja al sistema en el estado previo a la ejecución de la primera operación de la transacción.

# Operación de inserción

- Para añadir una fila a una tabla se utiliza

`INSERT INTO tabla VALUES fila_a_insertar`

- Para añadir un conjunto de filas a una tabla se utiliza

`INSERT INTO tabla Subconsulta`

- Si el orden de las columnas en *fila\_a\_insertar* o en el resultado de *Subconsulta* es diferente al orden de las columnas de tabla, hay que indicar el orden en el que vienen los datos

`tabla(col1, col2, col3, ..., coln)` indica que la primera columna de *fila\_a\_insertar* o *Subconsulta* se insertará en *col1* de la tabla, la segunda columna en *col2*, ...

# Operación de inserción

- Matricular al alumno con dni 12127891 en el grupo A de la asignatura con código 112 del curso 16/17

```
INSERT INTO matricular VALUES ('12127891',112,'A','16/17',null)
```

- Matricular a todos los alumnos en el grupo B de la asignatura con código 111 del curso 16/17

```
INSERT INTO Matricular (Asignatura, Grupo, Curso, Alumno)  
SELECT 111, 'B','16/17', dni  
FROM Alumnos ;
```

Los atributos no indicados se asignan al valor nulo o al valor por defecto definido para ellos si existe.

# Operación de modificación

- Para modificar datos de las filas se utiliza

**UPDATE** *tabla*

**SET** *atributo1 = expr1, atributo2 = expr2,... atributon = exprn*

**WHERE** *condición*

- Sólo las filas que satisfacen la condición serán modificadas.
- La parte WHERE es opcional. Si no existe, se modifican todas las filas.
- Podemos cambiar más de un atributo de una misma fila. Para cada uno de ellos hay que indicar el nuevo valor que tomará.
- Se pueden utilizar subconsultas tanto en la parte WHERE como en la parte SET.

# Operación de modificación

- Dar aprobado general a los alumnos del grupo B de la asignatura con código 112 del curso 15/16

```
UPDATE Matricular
```

```
SET calificacion = 'AP'
```

```
WHERE grupo = 'B' and asignatura = 112 and curso ='15/16';
```

- Asignar 6 créditos a todas las asignaturas que tengan grupo B

```
UPDATE asignaturas
```

```
SET creditos = 6
```

```
WHERE codigo IN ( SELECT asignatura  
                  FROM matricular  
                  WHERE grupo = 'B'  
                );
```

# Operación de modificación

- Asignar el despacho 2-2-43-A y el teléfono 2755 a todos los profesores del departamento de código 3.

UPDATE profesores

```
SET telefono = '2755', despacho = '2-2-43-A'  
WHERE departamento = 3;
```

- Cambiar el número de créditos de cada asignatura por el número de matrículas que se han realizado de esa asignatura.

UPDATE asignaturas **asig**

```
SET creditos = ( SELECT COUNT(*)  
                  FROM matricular  
                 WHERE asignatura = asig.codigo ) ;
```

# Operación de eliminación

- Para eliminar filas de una tabla se utiliza

**DELETE FROM *tabla***

**WHERE *condición***

- Sólo las filas que satisfacen la condición serán borradas.
- La parte WHERE es opcional. Si no existe, se borran todas las filas.
- Se pueden utilizar subconsultas en la parte WHERE.

# Operación de modificación

- Eliminar todas las matrículas de la asignatura con código 112.

```
DELETE FROM matricular  
WHERE asignatura = 112;
```

- Borrar las asignaturas que no tienen a nadie matriculado.

```
DELETE FROM asignaturas asig  
WHERE NOT EXISTS (  
    SELECT *  
    FROM matricular  
    WHERE asignatura = asig.codigo  
);
```

# Transacciones

- En Oracle una transacción se puede definir mediante un conjunto de sentencias dentro de un bloque PL/SQL (lenguaje de programación incrustado en Oracle)

BEGIN

    sentencias

END

- Las sentencias del bloque deben terminar con
  - **COMMIT**. Se confirman los cambios y se guardan en la base de datos.
  - **ROLLBACK**. Se deshacen los cambios y se vuelve al estado de la base de datos previo.

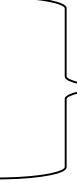
# Transacciones. Ejemplo

```
DECLARE
    importe NUMBER;  ctaOrigen VARCHAR2(23);  ctaDestino VARCHAR2(23);
BEGIN
    importe := 100;
    ctaOrigen := '2530 10 2000 1234567890';
    ctaDestino := '2532 10 2010 0987654321';

    UPDATE CUENTAS SET SALDO = SALDO – importe WHERE CUENTA = ctaOrigen;
    UPDATE CUENTAS SET SALDO = SALDO + importe WHERE CUENTA = ctaDestino;
    INSERT INTO MOVIMIENTOS VALUES (ctaOrigen, ctaDestino, importe*(-1), SYSDATE);
    INSERT INTO MOVIMIENTOS VALUES (ctaDestino, ctaOrigen, importe, SYSDATE);

    COMMIT;
EXCEPTION
    WHEN OTHERS THEN
        dbms_output.put_line('Error en la transaccion:' || SQLERRM);
        dbms_output.put_line('Se deshacen las modificaciones');
        ROLLBACK;
END;
```

## Elementos del nivel externo

- En el nivel externo se determinan las parcelas de información del esquema conceptual a las que los usuarios de la base de datos tienen acceso.
  - CREATE
  - ALTER
  - DROP

**VIEW** permiten crear, modificar y borrar vistas para los usuarios.
- También hay que indicar qué operaciones pueden realizar los usuarios sobre los elementos del nivel conceptual. Es decir hay que definir permisos.
  - **GRANT**, concede permisos a los usuarios
  - **REVOKE**, revoca permisos de los usuarios.

## Vistas

- Una vista se puede considerar una tabla virtual que se basa en una o más tablas o vistas

**CREATE VIEW** *nombreVista* (*atributo1, atributo2, ..., atributon*)  
**AS** *subconsulta\_de\_definición*.

- La subconsulta\_de\_definición determina las columnas que formarán parte de la vista.
  - Si ninguna de dichas columnas es el resultado de aplicar funciones u operaciones aritméticas, entonces no tendremos que especificar nombres de atributos. Se utilizan los mismos que en las tablas en las que se basa.
- Se puede utilizar como una tabla más.
- La vista siempre muestra la información actualizada.
  - Cambios en las tablas base se reflejan inmediatamente en la vista.

## Vistas

- Crear una vista que muestre el código junto al año de entrada en la universidad de los alumnos cuyo primer apellido comienza por la letra M.

```
CREATE VIEW Alumnos_con_Experiencia(Codigo, Cuando) AS
(SELECT Dni, TO_CHAR(Fecha_Prim_Matricula,'yyyy')
FROM Alumnos
WHERE Apellido1 LIKE 'M%' );
```

- Usar la vista Alumnos\_con\_Experiencia para mostrar la información de los alumnos con más de 2 años de antigüedad en la universidad.

```
SELECT *
FROM Alumnos_con_Experiencia
WHERE to_number(to_char(sysdate,'yyyy')) - to_number(Cuando) > 2;
```

| CODIGO   | CUANDO |
|----------|--------|
| 23456456 | 2014   |
| 12312342 | 2013   |
| 25423411 | 2014   |
| 62512427 | 2014   |
| 38260925 | 2014   |
| 47345210 | 2014   |
| 36982815 | 2013   |
| 28294639 | 2013   |
| 29707760 | 2014   |
| 12127891 | 2013   |
| 31333317 | 2014   |

## Vistas. Opciones

```
CREATE [OR REPLACE] [FORCE] VIEW nombreVista (atr1, atr2, ..., atrn)
AS subconsulta_de_definición
[WITH READ ONLY]
[WITH CHECK OPTION]
```

- OR REPLACE: permite sobrescribir la vista si ya existe.
- FORCE: crea la vista aunque las tablas base no existan o no se tengan permisos sobre ellas.
- WITH READ ONLY: impide que se pueda modificar la vista.
- WITH CHECK OPTION: impide modificar las filas de la vista si el resultado no coincide con alguna de las filas que devolvería la subconsulta.

## Actualización de vistas

- Podemos realizar operaciones de actualización del contenido de una vista. Dichos cambios serán propagados a las tablas base correspondientes si es posible.
- Para que una vista sea actualizable debe cumplir:
  - Sus columnas deben hacer referencia a columnas de alguna tabla base (no pueden ser el resultado de alguna expresión)
  - La subconsulta no puede tener:
    - Operaciones de conjunto
    - DISTINCT
    - Funciones de agregación
    - GROUP BY
    - ORDER BY.
    - Reuniones

## Actualización de vistas

- Crear una vista que muestre el la información de las asignaturas con código menor de 130.

```
CREATE VIEW asignaturas_menor_130
AS select * from asignaturas where codigo < 130;
```

- Insertar, usando la vista, la asignatura Programación de Videojuegos, con código 1 y el resto de los valores a null o al valor definido por defecto.

```
INSERT INTO asignaturas_menor_130(codigo,nombre)
VALUES (1, 'Programación de Videojuegos');
COMMIT;
```

- Mostrar el contenido de la vista

```
SELECT * FROM asignaturas_menor_130;
```

| CODIGO | NOMBRE                           | ... |
|--------|----------------------------------|-----|
| 1      | Programación de Videojuegos      | ... |
| 101    | Programación Orientada a Objetos | ... |
| 110    | Sistemas Operativos              | ... |
| 111    | Estadística                      | ... |
| 112    | Bases de Datos                   | ... |
| 113    | Calculo Numérico                 | ... |
| 114    | Teoría de Automatas              | ... |
| 115    | Administración de Bases de Datos | ... |
| 116    | Lógica Computacional             | ... |
| 122    | Estructura de Computadores       | ... |
| 123    | Computación Altas Prestaciones   | ... |

# Actualización de vistas

- Insertar la asignatura 'Aprendizaje Computacional' con código 300

```
INSERT INTO asignaturas_menor_130(codigo,nombre)  
VALUES (300, 'Aprendizaje Computacional');  
COMMIT;
```

- Mostramos la vista

```
SELECT * FROM asignaturas_menor_130;
```

- ¡¡Pero la nueva asignatura se ha insertado!!

```
SELECT * FROM asignaturas;
```

| CODIGO | NOMBRE                           | ... |
|--------|----------------------------------|-----|
| 1      | Programación de Videojuegos      | ... |
| 101    | Programación Orientada a Objetos | ... |
| 110    | Sistemas Operativos              | ... |
| 111    | Estadística                      | ... |
| 112    | Bases de Datos                   | ... |
| 113    | Calculo Numérico                 | ... |
| 114    | Teoría de Automatas              | ... |
| 115    | Administración de Bases de Datos | ... |
| 116    | Lógica Computacional             | ... |
| 122    | Estructura de Computadores       | ... |
| 123    | Computación Altas Prestaciones   | ... |

| CODIGO | NOMBRE                           | ... |
|--------|----------------------------------|-----|
| 200    | Teoría de la señal               | ... |
| 140    | Prácticas en empresa             | ... |
| 113    | Calculo Numérico                 | ... |
| 114    | Teoría de Automatas              | ... |
| 115    | Administración de Bases de Datos | ... |
| 116    | Lógica Computacional             | ... |
| 112    | Bases de Datos                   | ... |
| 110    | Sistemas Operativos              | ... |
| 122    | Estructura de Computadores       | ... |
| 133    | Ingeniería Web                   | ... |
| 134    | Dispositivos Electrónicos        | ... |
| 144    | Modelos Computacionales          | ... |
| 123    | Computación Altas Prestaciones   | ... |
| 111    | Estadística                      | ... |
| 101    | Programación Orientada a Objetos | ... |
| 201    | Matemática Discreta              | ... |
| 1      | Programación de Videojuegos      | ... |
| 300    | Aprendizaje Computacional        | ... |

## Actualización de vistas

- Para evitarlo utilizamos la opción WITH CHECK OPTION

```
CREATE OR REPLACE VIEW asignaturas_menor_130
AS
select * from asignaturas where codigo < 130
WITH CHECK OPTION;
```

- Ahora al insertar una fila que la subconsulta no devolvería (código mayor o igual a 130) el SGBD devuelve un error.

Informe de error -

Error SQL: ORA-01402: violación de la cláusula WHERE en la vista WITH CHECK OPTION  
01402. 00000 - "view WITH CHECK OPTION where-clause violation"

- Por último, borramos las filas introducidas en el ejemplo mediante la vista.

```
DELETE FROM asignaturas_menor_130 where codigo =1 or codigo = 300;
COMMIT;
```

# Permisos

- Tipos de permisos:

- SELECT, permiso de lectura
- INSERT, permiso de inserción de filas nuevas
- UPDATE, permiso de modificación de filas
- DELETE, permiso de eliminación de filas.

- Conceder permisos sobre una tabla/vista a un usuario:

**GRANT *permisos* ON *tabla* TO *usuario* [WITH GRANT OPTION];**

- La opción WITH GRANT OPTION permite al usuario receptor del permiso concederlo a otros usuarios.

- Quitar permisos sobre una tabla/vista a un usuario:

**REVOKE *permisos* ON *tabla* FROM *usuario***

## Permisos

- Dar permisos de lectura e inserción al usuario ENCISO sobre la vista ALUMNOS\_CON\_EXPERIENCIA

```
GRANT SELECT, INSERT ON Alumnos_con_Experiencia  
TO enciso ;
```

- Quitar el permiso de lectura de la tabla matricular a todos los usuarios

```
REVOKE SELECT ON matricular  
TO public;
```

# Metadatos

- Los **metadatos** son datos que describen otros datos. En el ámbito de las bases de datos, los metadatos son datos almacenados en tablas del sistema que describen la estructura de los elementos de la base de datos (tablas, columnas, restricciones, vistas,...)
- El conjunto de tablas que contienen metadatos se denomina **diccionario de datos** o catálogo del sistema.
- En Oracle las tablas del diccionario de datos sólo son leídas y modificadas por el SGBD.
- Sin embargo, existen una serie de vistas públicas que los usuarios pueden utilizar para consultar información del diccionario de datos.

## Vistas del diccionario de datos

- El nombre de las vistas del diccionario de datos comienza por alguno de los siguientes prefijos:
  - **ALL**, vista que describe elementos a los que el usuario tiene acceso.
  - **DBA**, vista que describe elementos de la base de datos completa. Sólo los administradores tienen permiso para utilizar estas vistas.
  - **USER**, vista que describe elementos creados por el usuario.
- Algunas de las vistas de usuario son
  - USER\_TABLES, USER\_TAB\_COLUMNS
  - USER\_CONSTRAINTS, USER\_CONS\_COLUMNS
  - USER\_VIEWS
- Todas tienen sus vistas “hermanas” para administradores (dba) y acceso extendido (all)

# Metadatos sobre tablas

- **ALL\_TABLES** describe las tablas a las que tiene acceso el usuario  
Algunos atributos interesantes son
  - TABLE\_NAME, nombre la tabla
  - TABLESPACE\_NAME, espacio de tablas en el que se creó la tabla
  - OWNER, el creador de la tabla
- **USER\_TABLES** describe las tablas creadas por el usuario  
Tiene los mismos atributos que ALL\_TABLES excepto OWNER.
- ¿Cómo se obtendría el nombre de todas las tablas a las que tiene acceso el usuario actual y que no ha creado él?
- ¿Cómo podemos borrar todas las tablas que hemos creado?

# Metadatos sobre tablas

- **USER\_TAB\_COLUMNS** describe las columnas de las tablas del usuario

Atributos:

- TABLE\_NAME, nombre de la tabla o vista que contiene la columna
- COLUMN\_NAME , nombre de la columna
- DATA\_TYPE, tipo de datos de la columna
- DATA\_LENGTH, longitud del tipo de datos en bytes
- DATA\_PRECISION, DATA\_SCALE: precisión y longitud del tipo NUMBER
- NULLABLE, obligatoriedad del atributo
- COLUMN\_ID, orden del atributo al crearse la tabla.
- ...

# Metadatos sobre tablas

- Simular el comando DESC para la tabla profesores.

```
SELECT column_name "Nombre", decode(nullable,'N','NOT NULL',' ') "Nulo",
data_type || decode(data_type,'DATE','','NUMBER','(' || data_precision || ')','(' || data_length || ')') "Tipo"
FROM user_tab_columns
WHERE upper(table_name) = 'PROFESORES';
```

DESC profesores;

| Nombre           | Nulo     | Tipo           |
|------------------|----------|----------------|
| ID               | NOT NULL | VARCHAR2 (20)  |
| NOMBRE           | NOT NULL | VARCHAR2 (20)  |
| APELLIDO1        | NOT NULL | VARCHAR2 (20)  |
| APELLIDO2        |          | VARCHAR2 (20)  |
| DEPARTAMENTO     | NOT NULL | NUMBER (3)     |
| TELEFONO         |          | VARCHAR2 (4)   |
| EMAIL            |          | VARCHAR2 (100) |
| DESPACHO         |          | VARCHAR2 (10)  |
| FECHA_NACIMIENTO |          | DATE           |
| ANTIGUEDAD       |          | DATE           |
| DIRECTOR_TESIS   |          | VARCHAR2 (20)  |

| Nombre           | Nulo     | Tipo                |
|------------------|----------|---------------------|
| ID               | NOT NULL | VARCHAR2 (20)       |
| NOMBRE           | NOT NULL | VARCHAR2 (20)       |
| APELLIDO1        | NOT NULL | VARCHAR2 (20)       |
| APELLIDO2        |          | VARCHAR2 (20)       |
| DEPARTAMENTO     |          | NOT NULL NUMBER (3) |
| TELEFONO         |          | VARCHAR2 (4)        |
| EMAIL            |          | VARCHAR2 (100)      |
| DESPACHO         |          | VARCHAR2 (10)       |
| FECHA_NACIMIENTO |          | DATE                |
| ANTIGUEDAD       |          | DATE                |
| DIRECTOR_TESIS   |          | VARCHAR2 (20)       |

# Metadatos sobre restricciones

- **USER\_CONSTRAINTS** describe las restricciones creadas por el usuario

Atributos:

- CONSTRAINT\_NAME Nombre (único) de la restricción
- TABLE\_NAME, tabla que contiene la restricción
- CONSTRAINT\_TYPE, tipo de restricción
- SEARCH\_CONDITION, predicado si restricción de tipo 'C' (Check)
- R\_CONSTRAINT\_NAME, restricción que define la clave destino si restricción de tipo 'R' (clave foránea)
- ...

## Metadatos sobre restricciones

- Extraer las parejas de tablas tales que la primera componente tiene una clave foránea hacia la segunda.

```
SELECT r1.table_name, r2.table_name
FROM user_constraints r1, user_constraints r2
WHERE r1.r_constraint_name = r2.constraint_name;
```

| TABLE_NAME     | TABLE_NAME_1  |
|----------------|---------------|
| CORREQUISITOS  | ASIGNATURAS   |
| CORREQUISITOS  | ASIGNATURAS   |
| IMPARTIR       | ASIGNATURAS   |
| MATRICULAR     | ASIGNATURAS   |
| PRERREQUISITOS | ASIGNATURAS   |
| PRERREQUISITOS | ASIGNATURAS   |
| MATRICULAR     | ALUMNOS       |
| ASIGNATURAS    | DEPARTAMENTOS |
| PROFESORES     | DEPARTAMENTOS |
| IMPARTIR       | PROFESORES    |
| INVESTIGADORES | PROFESORES    |
| PROFESORES     | PROFESORES    |
| ASIGNATURAS    | MATERIAS      |
| MUNICIPIO      | PROVINCIA     |
| ALUMNOS        | MUNICIPIO     |

# Metadatos sobre restricciones

- **USER\_CONS\_COLUMNS** describe columnas que aparecen en las restricciones creadas por el usuario

## Atributos

- CONSTRAINT\_NAME, nombre de la restricción en la que aparece la columna
- TABLE\_NAME, nombre de la tabla en la que aparece la columna
- COLUMN\_NAME, nombre de la columna que aparece en la restricción
- POSITION, posición de la columna en la definición de la restricción

## Metadatos sobre restricciones

- Mostrar los atributos que forman parte de la clave primaria de la tabla Matricular.

```
SELECT column_name
FROM user_cons_columns RCOL, user_constraints R
WHERE RCOL.constraint_name = R.constraint_name AND
      R.constraint_type = 'P' AND upper(R.table_name) = 'MATRICULAR';
```

| COLUMN_NAME |
|-------------|
| ALUMNO      |
| ASIGNATURA  |
| GRUPO       |
| CURSO       |

- Determinar a qué atributos referencian las claves foráneas definidas en la tabla Matricular. Mostrar Atr\_origen, Atr\_destino, Tabla\_destino.

```
SELECT RCOL1.column_name, RCOL2.column_name, RCOL2.table_name
FROM user_cons_columns RCOL1, user_constraints R1, user_constraints R2, user_cons_columns RCOL2
WHERE   RCOL1.constraint_name = R1.constraint_name AND
        R1.r_constraint_name = R2.constraint_name AND
        R2.constraint_name = RCOL2.constraint_name and
        R1.constraint_type = 'R' AND R1.table_name = 'MATRICULAR';
```

| COLUMN_NAME | COLUMN_NAME_1 | TABLE_NAME  |
|-------------|---------------|-------------|
| ASIGNATURA  | CODIGO        | ASIGNATURAS |
| ALUMNO      | DNI           | ALUMNOS     |

# Otros Objetos de la Base de Datos

Disparadores y Trabajos



## Definición de TRIGGER

- Son elementos que especifican acciones desencadenadas por una operación sobre:
  - Operación DML: INSERT, DELETE o UPDATE
  - Operación DDL o evento de la Base de datos
- Al contrario que un procedimiento o función no son llamados sino que se ejecutan de forma implícita.
- Un Disparador No admite Argumentos
- Vista USER\_TRIGGERS

## Utilidad de los TRIGGERS

- Sus aplicaciones son inmensas, como por ejemplo:
  - Mantenimiento de **Restricciones** de Integridad complejas.
    - Ej: Restricciones de Estado (como que el sueldo sólo puede aumentar).
  - Auditoría de una Tabla, registrando los cambios efectuados y la identidad del que los llevó a cabo.
  - Lanzar cualquier acción cuando una tabla es modificada.

## Creación

```
CREATE [ OR REPLACE ] TRIGGER <NombreT>
{BEFORE | AFTER} <Suceso_Disparo> ON <Tabla>
[ FOR EACH ROW [ WHEN <Condición_Disparo> ] ]
    <Bloque_del_TRIGGER>;
```

- <Suceso\_Disparo> es la operación **DML** que efectuada sobre <Tabla> disparará el *trigger*: INSERT, DELETE o UPDATE
  - » Puede haber varios sucesos separados por la palabra OR.
  - » Si la orden UPDATE lleva una lista de atributos el Disparador sólo se ejecutará si se actualiza alguno de ellos: UPDATE OF <Lista\_Atributos>

## Creación

```
CREATE [ OR REPLACE ] TRIGGER <NombreT>
{BEFORE | AFTER} <Suceso_Disparo> ON <Tabla>
[ FOR EACH ROW [WHEN <Condición_Disparo>] ]
    <Bloque_del_TRIGGER>;
```

- Temporización: BEFORE (anterior) o AFTER (posterior).
  - Define si el disparador se activa antes o después de que se ejecute la operación DML causante del disparo.
- Nivel: a Nivel de Tabla o a Nivel de Fila (FOR EACH ROW).
  - Nivel de Tabla: Se activan sólo una vez, antes o después de la Sentencia u operación DML.
  - Nivel de Fila: Se activan una vez por cada Fila afectada por la operación DML (una misma operación DML puede afectar a varias filas).

## Creación

```
CREATE [ OR REPLACE ] TRIGGER <NombreT>
{BEFORE | AFTER} <Suceso_Disparo> ON <Tabla>
[ FOR EACH ROW [WHEN <Condición_Disparo>] ]
    <Bloque_del_TRIGGER>;
```

- **<Bloque\_del\_TRIGGER>**
  - El cuerpo es un bloque de PL/SQL.
  - No tiene órdenes de control de transacciones. Idem para rutinas llamadas por el disparador.
  - No se pueden declarar variables de tipo LONG o LONG RAW.
  - No hay libertad absoluta de acceso a tablas (problema de la tabla mutante).
  - Pueden usarse los predicados INSERTING, UPDATING o DELETING en el cuerpo para discriminar el suceso.

## Ejemplo

- **Ejemplo:** Guardar en una tabla de control la fecha y el usuario que modificó la tabla Empleados:
- (NOTA: Este trigger es **AFTER** y a nivel de tabla)

```
CREATE OR REPLACE TRIGGER Control_Empieados
    AFTER INSERT OR DELETE OR UPDATE ON Empleados
BEGIN
    INSERT INTO Ctrl_Empieados (Tabla, Usuario, Fecha)
        VALUES ('Empleados', USER, SYSDATE);
END Control_Empieados;
```

## Orden de Ejecución

1. Ejecutar el disparador tipo BEFORE a nivel de tabla.
2. Para cada fila a la que afecte la orden
  - a) Ejecutar disparador BEFORE a nivel de fila, sólo si dicha fila cumple la condición de la cláusula WHEN (si existe).
  - b) Ejecutar la propia orden.
  - c) Ejecutar disparador AFTER a nivel de fila, sólo si dicha fila cumple la condición de la cláusula WHEN (si existe).
3. Ejecutar el disparador tipo AFTER a nivel de orden.

## Variables de acoplamiento

Cuando el Trigger es de nivel de fila se pueden utilizar 2 variables de acoplamiento, :old y :new

|        | :old                                                                                                                                                           | :new |     |     |    |     |     |                                                                                                                                                                |     |     |                                                                                                                                                                |    |     |     |    |     |     |    |     |     |
|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|------|-----|-----|----|-----|-----|----------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|-----|----------------------------------------------------------------------------------------------------------------------------------------------------------------|----|-----|-----|----|-----|-----|----|-----|-----|
| DELETE | <table border="1"><tr><td>IB</td><td>456</td><td>VLC</td></tr><tr><td>IB</td><td>875</td><td>AGP</td></tr><tr><td>AA</td><td>456</td><td>AGP</td></tr></table> | IB   | 456 | VLC | IB | 875 | AGP | AA                                                                                                                                                             | 456 | AGP | <table border="1"><tr><td>IB</td><td>456</td><td>VLC</td></tr><tr><td></td><td></td><td></td></tr><tr><td>AA</td><td>456</td><td>AGP</td></tr></table>         | IB | 456 | VLC |    |     |     | AA | 456 | AGP |
| IB     | 456                                                                                                                                                            | VLC  |     |     |    |     |     |                                                                                                                                                                |     |     |                                                                                                                                                                |    |     |     |    |     |     |    |     |     |
| IB     | 875                                                                                                                                                            | AGP  |     |     |    |     |     |                                                                                                                                                                |     |     |                                                                                                                                                                |    |     |     |    |     |     |    |     |     |
| AA     | 456                                                                                                                                                            | AGP  |     |     |    |     |     |                                                                                                                                                                |     |     |                                                                                                                                                                |    |     |     |    |     |     |    |     |     |
| IB     | 456                                                                                                                                                            | VLC  |     |     |    |     |     |                                                                                                                                                                |     |     |                                                                                                                                                                |    |     |     |    |     |     |    |     |     |
|        |                                                                                                                                                                |      |     |     |    |     |     |                                                                                                                                                                |     |     |                                                                                                                                                                |    |     |     |    |     |     |    |     |     |
| AA     | 456                                                                                                                                                            | AGP  |     |     |    |     |     |                                                                                                                                                                |     |     |                                                                                                                                                                |    |     |     |    |     |     |    |     |     |
| UPDATE | <table border="1"><tr><td>IB</td><td>456</td><td>VLC</td></tr><tr><td>IB</td><td>875</td><td>AGP</td></tr><tr><td>AA</td><td>456</td><td>AGP</td></tr></table> | IB   | 456 | VLC | IB | 875 | AGP | AA                                                                                                                                                             | 456 | AGP | <table border="1"><tr><td>IB</td><td>456</td><td>VLC</td></tr><tr><td>IB</td><td>999</td><td>VLC</td></tr><tr><td>AA</td><td>456</td><td>AGP</td></tr></table> | IB | 456 | VLC | IB | 999 | VLC | AA | 456 | AGP |
| IB     | 456                                                                                                                                                            | VLC  |     |     |    |     |     |                                                                                                                                                                |     |     |                                                                                                                                                                |    |     |     |    |     |     |    |     |     |
| IB     | 875                                                                                                                                                            | AGP  |     |     |    |     |     |                                                                                                                                                                |     |     |                                                                                                                                                                |    |     |     |    |     |     |    |     |     |
| AA     | 456                                                                                                                                                            | AGP  |     |     |    |     |     |                                                                                                                                                                |     |     |                                                                                                                                                                |    |     |     |    |     |     |    |     |     |
| IB     | 456                                                                                                                                                            | VLC  |     |     |    |     |     |                                                                                                                                                                |     |     |                                                                                                                                                                |    |     |     |    |     |     |    |     |     |
| IB     | 999                                                                                                                                                            | VLC  |     |     |    |     |     |                                                                                                                                                                |     |     |                                                                                                                                                                |    |     |     |    |     |     |    |     |     |
| AA     | 456                                                                                                                                                            | AGP  |     |     |    |     |     |                                                                                                                                                                |     |     |                                                                                                                                                                |    |     |     |    |     |     |    |     |     |
| INSERT | <table border="1"><tr><td>IB</td><td>456</td><td>VLC</td></tr><tr><td>AA</td><td>456</td><td>AGP</td></tr></table>                                             | IB   | 456 | VLC | AA | 456 | AGP | <table border="1"><tr><td>IB</td><td>456</td><td>VLC</td></tr><tr><td>IB</td><td>875</td><td>AGP</td></tr><tr><td>AA</td><td>456</td><td>AGP</td></tr></table> | IB  | 456 | VLC                                                                                                                                                            | IB | 875 | AGP | AA | 456 | AGP |    |     |     |
| IB     | 456                                                                                                                                                            | VLC  |     |     |    |     |     |                                                                                                                                                                |     |     |                                                                                                                                                                |    |     |     |    |     |     |    |     |     |
| AA     | 456                                                                                                                                                            | AGP  |     |     |    |     |     |                                                                                                                                                                |     |     |                                                                                                                                                                |    |     |     |    |     |     |    |     |     |
| IB     | 456                                                                                                                                                            | VLC  |     |     |    |     |     |                                                                                                                                                                |     |     |                                                                                                                                                                |    |     |     |    |     |     |    |     |     |
| IB     | 875                                                                                                                                                            | AGP  |     |     |    |     |     |                                                                                                                                                                |     |     |                                                                                                                                                                |    |     |     |    |     |     |    |     |     |
| AA     | 456                                                                                                                                                            | AGP  |     |     |    |     |     |                                                                                                                                                                |     |     |                                                                                                                                                                |    |     |     |    |     |     |    |     |     |

## Ejemplos

Si se actualiza el código de una Pieza, actualizar el código de los Suministros en los que se usaba:

-- Actualizar Suministros.Pieza si cambia Pieza.Codigo:

```
CREATE OR REPLACE TRIGGER Actualiza_SuministrosPieza
    BEFORE UPDATE OF Codigo ON Pieza FOR EACH ROW
BEGIN
    UPDATE Suministros SET Pieza = :new.Codigo
    WHERE Pieza = :old.Codigo;
END Actualiza_SuministrosPieza;
```

## Ejemplos

- Guardar en una tabla de control la fecha y el usuario que modifica la tabla Asignaturas. Se guarda una sola fila por cada sentencia INSERT, independientemente del número de filas involucradas.
- Creamos la tabla

```
CREATE TABLE Ctr_Tablas
  Usuario VARCHAR2( 30 ) ,
  Fecha DATE ,
  Tabla VARCHAR2( 30 ) ,
  Operacion VARCHAR2( 30 ) );
```

## Ejemplos

```
CREATE OR REPLACE TRIGGER Control_Asignaturas
AFTER INSERT OR DELETE OR UPDATE ON Asignaturas
BEGIN
    IF INSERTING THEN -- Se ejecuta solo si es de INSERT
        INSERT INTO Ctrl_Tablas (Tabla,Usuario,Fecha,Operacion)
        VALUES ('Asignaturas', USER, SYSDATE, 'INSERT');
    ELSIF DELETING THEN
        INSERT INTO Ctrl_Tablas (Tabla,Usuario,Fecha, Operacion)
        VALUES ('Asignaturas', USER, SYSDATE, 'DELETE');
    ELSE      -- UPDATING
        INSERT INTO Ctrl_Tablas (Tabla,Usuario,Fecha, Operacion)
        VALUES ('Asignaturas', USER, SYSDATE, 'UPDATE');
    END IF;
END Control Asignaturas;
```

## Disparadores de Sustitución

- Disparador que se ejecuta en lugar de la orden DML (ni antes ni después, sino sustituyéndola).
  - **Sólo pueden definirse sobre Vistas.**
  - **Se activan en lugar de la Orden DML** que provoca el disparo, o sea, la orden disparadora no se ejecutará nunca.
  - **Deben tener Nivel de Filas.**
  - Se declaran usando INSTEAD OF en vez de BEFORE/AFTER.

## Ejemplos

- Queremos crear una vista que guarde información agrupada del número de alumnos matriculados en cada asignatura
- Creamos la vista

```
CREATE VIEW Global_Matriculas AS
SELECT asignatura, COUNT(*) Num_Matriculas
FROM Matricular
GROUP BY asignatura;
```

## Ejemplos

- Se quiere crear un disparador que borre todas las matrículas de una asignatura si se borra una tupla sobre la vista Global\_Matriculas

```
CREATE OR REPLACE TRIGGER Borrar_en_Global
INSTEAD OF DELETE ON Global_Matriculas
FOR EACH ROW
BEGIN
    DELETE FROM matriculas
    WHERE asignatura = :OLD.asignatura;
END Borrar_en_Global;
```

## Jobs

- Oracle Scheduler: planificador de trabajos DBMS\_SCHEDULER
- JOB = PROGRAM + SCHEDULE
- ¿Qué se puede ejecutar?:
  - Unidades de **programa** de bases de datos (STORED\_PROCEDURE, PLSQL\_BLOCK)
  - **Programas** externos (ejecutables, scripts, etc.) de forma local o en otras bases de datos (EXTERNAL)
- ¿Cuándo se ejecuta?:
  - Time-based scheduling. Se define la fecha y la repetición
  - Event-based scheduling. Cuando falla una transacción, llega un fichero, etc.
  - Dependency scheduling → Chains. Se pueden definir cadenas complejas de trabajos, con ramas, etc.

## Ejecución de los Trabajos

Una Trabajo es un objeto y el dueño es el usuario que lo crea  
El programa se ejecuta con las credenciales del trabajo (salvo que se indique lo contrario)

### **Al ejecutarse:**

- Cuando es hora de ejecutarse, se despierta un proceso para ejecutar el programa
  1. Se recogen todos los metadatos necesarios, argumentos de programa e información de los privilegios
  2. Se crea una sesión con las credenciales del dueño del trabajo, se comienza una transacción y se ejecuta el programa del trabajo.
  3. Cuando se completa, se hace commit y se termina la transacción.
  4. Se cierra la sesión.

### **Cuando el trabajo termina:**

- Se planifica la siguiente ejecución si es necesario.
- Se modifica la tabla de trabajos para indicar si ha terminado o si se tiene que ejecutar de nuevo.
- Se inserta una entrada en la tabla de log de trabajos.
  - USER\_SCHEDULER\_JOB\_LOG

## Ejemplo

```
BEGIN

    DBMS_SCHEDULER.CREATE_JOB (
        job_name          => 'mi_tarea',
        job_type          => 'PLSQL_BLOCK',
        job_action         => 'BEGIN insert into prueba values
(17,USER,sysdate); END; ',
        start_date         => SYSDATE+1,
        repeat_interval   => 'FREQ=SECONDLY; INTERVAL=10' ,
        end_date           => '30/MAY/2018 20.00.00',
        enabled            => TRUE,
        comments           => 'Inserta una tupla cada 10
segundos a partir de mañana y hasta el 30 de mayo de
2018' );
END;
```

## Gestión de trabajos

- Ver los trabajos:
  - `select * from user_scheduler_jobs;`
- Borrar un trabajo (o varios):
  - `DBMS_SCHEDULER.DROP_JOB ('job1, job2, job3');`
- Deshabilitarlo:
  - `DBMS_SCHEDULER.DISABLE ('job1, job2, job3');`
- Habilitarlo:
  - `DBMS_SCHEDULER.ENABLE ('job1, job2, job3');`

## Planificación

- Cada viernes:
  - FREQ=WEEKLY; BYDAY=FRI;
- Cada 2 viernes:
  - FREQ=WEEKLY; INTERVAL=2; BYDAY=FRI;
- El último día de cada mes
  - FREQ=MONTHLY; BYMONTHDAY=-1;
- El penúltimo día de cada mes:
  - FREQ=MONTHLY; BYMONTHDAY=-2;
- Cada año, el 10 de marzo:
  - FREQ=YEARLY; BYMONTH=MAR; BYMONTHDAY=10;