

Interoperabilidad Ingeniería Biomédica

Integrating the Healthcare Enterprise (IHE)



UNIVERSIDAD CES
Un compromiso con la excelencia

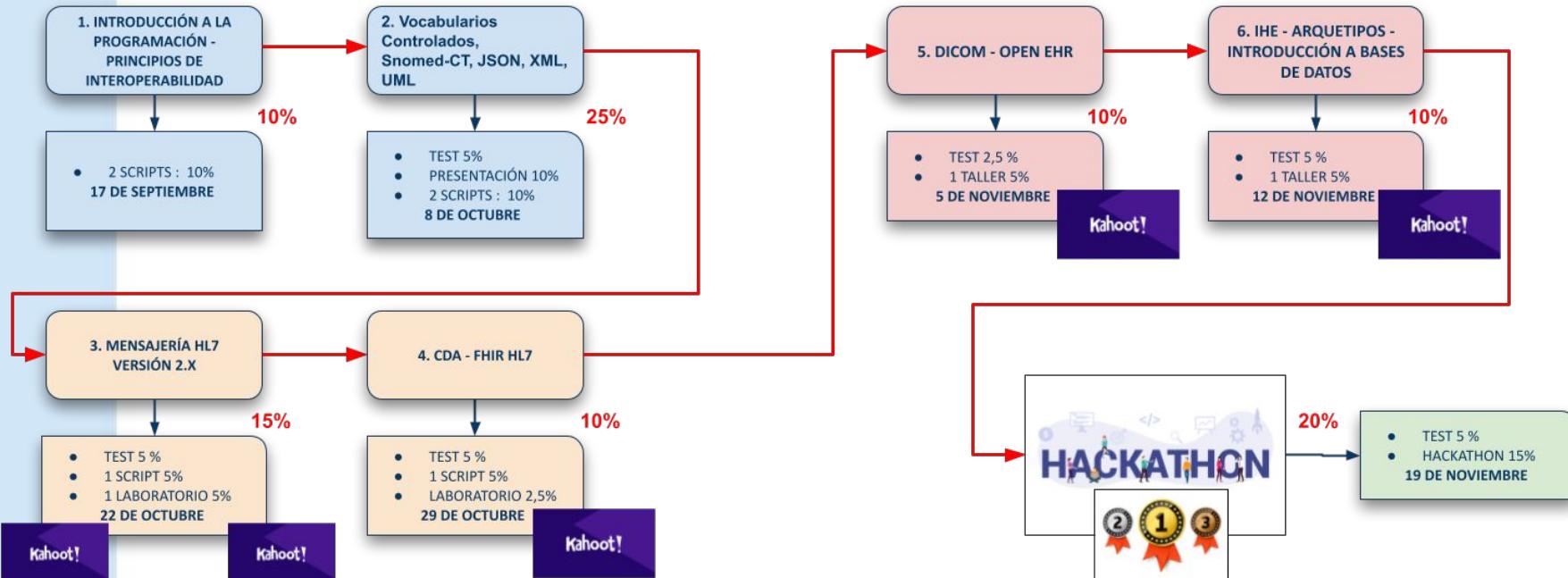
Sesión 6
Ing. Pedro Ortiz Tamayo

La Universidad CES es la propietaria y titular de todos los derechos de propiedad intelectual asociados al presente contenido. La comunicación pública del mismo se realiza, única y exclusivamente, con fines de divulgación e información. Por lo tanto, el material no se podrá usar para propósitos diferentes a los indicados.

La presente divulgación no implica licencia, cesión o autorización de uso o explotación de ningún tipo de derechos de propiedad intelectual diferentes sobre el mismo. La copia, reproducción total o parcial, modificación, adaptación, traducción o distribución, infringe los derechos de la Universidad y causa daños por los que se podrá ser objeto de las acciones civiles y penales correspondientes y de las medidas cautelares que se consideren pertinentes o necesarias. Las opiniones expresadas por los autores o participes no constituyen ni comprometen la posición oficial o institucional de la Universidad CES.



PLAN DE TRABAJO INTEROPERABILIDAD - INGENIERÍA BIOMÉDICA

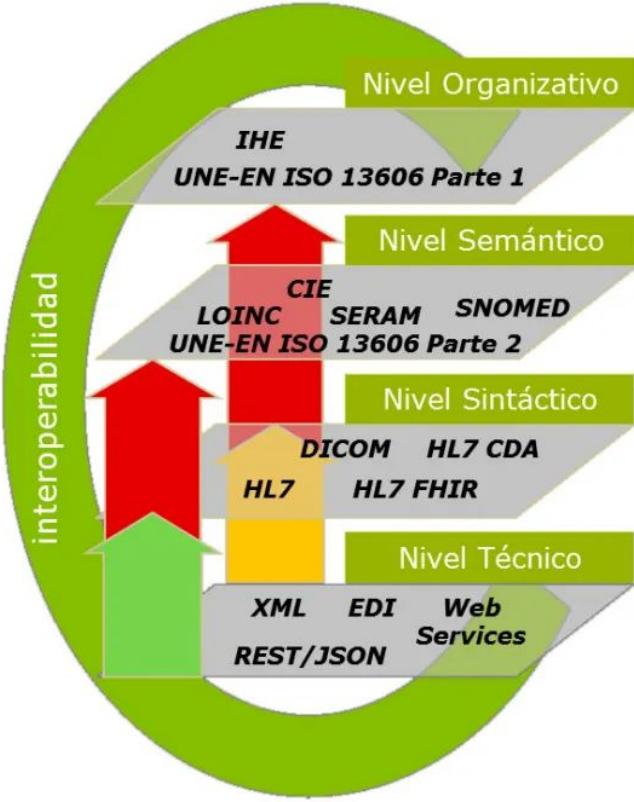


Agenda :

1. IHE
2. Arquetipos en Medicina
3. Bases de Datos
4. Modelo de Datos
5. Diagrama Entidad-Relación
6. Modelo Relacional
7. Lenguaje SQL



UNIVERSIDAD CES
Un compromiso con la excelencia



Interoperabilidad por Niveles:

- **Técnico**
 - XML
 - RST/JSON
- **Sintáctico**
 - DICOM
 - HL7 Ver. 2.x
 - HL7 CDA
 - HL7 FHIR
- **Semántico**
 - Snomed-CT
- **Organizativo**
 - IHE



UNIVERSIDAD CES

Un compromiso con la excelencia



Integrating
the Healthcare
Enterprise

Integrating the Healthcare Enterprise (IHE)



UNIVERSIDAD CES
Un compromiso con la excelencia

IHE ¿QUE ES?

La sigla IHE corresponde a Integrating the Healthcare Enterprise, que puede traducirse como “**integrando empresas sanitarias**”. Es una iniciativa de profesionales de la sanidad (que incluye a colegios profesionales de médicos) y empresas proveedoras, con el objetivo de mejorar la comunicación entre los sistemas de información que se utilizan en la atención al paciente.



UNIVERSIDAD CES
Un compromiso con la excelencia

IHE ¿QUE HACE?

IHE define perfiles de integración que utilizan estándares ya existentes para la constitución de sistemas, de manera que proporcionen una interoperabilidad efectiva y un flujo de trabajo eficiente. También, IHE nos permite alcanzar el nivel de integración exigible en la era de la historia clínica electrónica.



UNIVERSIDAD CES
Un compromiso con la excelencia

IHE ¿QUE HACE?

- No es un estándar.
- Sí es una recomendación de uso de estándares existentes.
- No es una compañía privada.
- Sí es una organización sin fines de lucro, cuya misión es el desarrollo y la promoción de su recomendación en particular y de los estándares médicos en general.
- No es un equipamiento, un software ni un lenguaje de programación.
- Sí es un conjunto de especificaciones que forman un marco técnico. De hecho, cuando un equipamiento o software cumple estas especificaciones, tiene conformidad con IHE.
- No es exclusivamente para informáticos. Sí es de interés para todo el entorno de salud.



UNIVERSIDAD CES
Un compromiso con la excelencia

IHE ¿COMO COLABORA?

El trabajo de IHE es seleccionar, sobre la base de una metodología de trabajo establecida y teniendo en cuenta el **requerimiento de interoperabilidad**, los estándares que mejor lo cubren . **Esto implica seleccionar los estándares adecuados para cada caso y elaborar un perfil de implementación.**



IHE ¿COMO COLABORA?

Cada perfil de integración de IHE describe una necesidad clínica de fusión de sistemas y la solución para llevarla a cabo. Define, también, los componentes funcionales, a los que denominamos actores IHE , y específica, con el mayor grado de detalle posible, las transacciones que cada actor debe llevar a cabo; siempre, basadas en estándares como DICOM y HL7.



IHE BENEFICIOS

1. IHE para los clínicos

- Ayuda a proporcionar información clínica fácilmente accesible , sin pérdidas, errores ni redundancias.
- Optimiza el flujo de tareas , mejorando la eficacia y la eficiencia.
- Favorece el intercambio de información entre diferentes sistemas.



UNIVERSIDAD CES
Un compromiso con la excelencia

IHE BENEFICIOS

2. IHE para los profesionales de IT

- Facilita la definición y la integración de diferentes sistemas, evita procesos y diseños a medida, costosos y de resultados dudosos.
- Garantiza resultados fiables de integración .



UNIVERSIDAD CES
Un compromiso con la excelencia

IHE BENEFICIOS

3. IHE para los gerentes de salud

- Facilita la tarea de la toma de decisiones a la hora de adquirir diferentes sistemas, gracias a los certificados de integración .
- Por lo tanto, contribuye a disminuir los gastos, aumenta la eficiencia del personal, minimiza los problemas de integración y, en definitiva, mejora la atención para la salud.



UNIVERSIDAD CES
Un compromiso con la excelencia

IHE BENEFICIOS

4. IHE para la reducción de costos y esfuerzo



UNIVERSIDAD CES
Un compromiso con la excelencia

IHE ORGANIZACIÓN

IHE International tiene base en Illinois, Estados Unidos, con patrocinio principal de HIMSS y la RSNA. Dispone de un directorio para cada dominio internacional, nacional y las organizaciones regionales. Cuenta con patrocinadores, en tanto que los dominios son patrocinados por las sociedades médicas/profesionales que desarrollan soluciones técnicas a los casos de uso.

También cuenta con comités de implementación regional que promueven el uso de las especificaciones IHE en sus áreas geográficas y trabajan en conjunto con los gobiernos locales y los ministerios de Salud.



UNIVERSIDAD CES
Un compromiso con la excelencia

IHE ESTRUCTURA Y MIEMBROS

Solo las organizaciones pueden ser miembros de IHE International y firman un acuerdo de propiedad intelectual (similar al acuerdo IP DICOM). Únicamente los miembros tienen posibilidad de votar en los comités. La participación sin derecho de voto en las actividades es libre. La membresía en IHE International es gratis.



IHE PROCESO DE ADOPCIÓN

1. Primer paso: identificación del problema

IHE aborda las necesidades de integración entre los departamentos y los dominios clínicos, así como a través de las organizaciones.



UNIVERSIDAD CES
Un compromiso con la excelencia

IHE PROCESO DE ADOPCIÓN

2. Segundo paso: definir el perfil de implementación

Todos los documentos IHE se someten a revisión pública para garantizar el más amplio consenso posible. Luego, se publican y están disponibles libremente para descarga en el sitio web de IHE



UNIVERSIDAD CES
Un compromiso con la excelencia

IHE PROCESO DE ADOPCIÓN

3. Tercer paso: Connectathon

Se realiza una convocatoria abierta a la participación de los proveedores que implementan el perfil. Los participantes ponen a prueba sus sistemas con una suite de herramientas de software en un Connectathon anual de IHE.



UNIVERSIDAD CES
Un compromiso con la excelencia

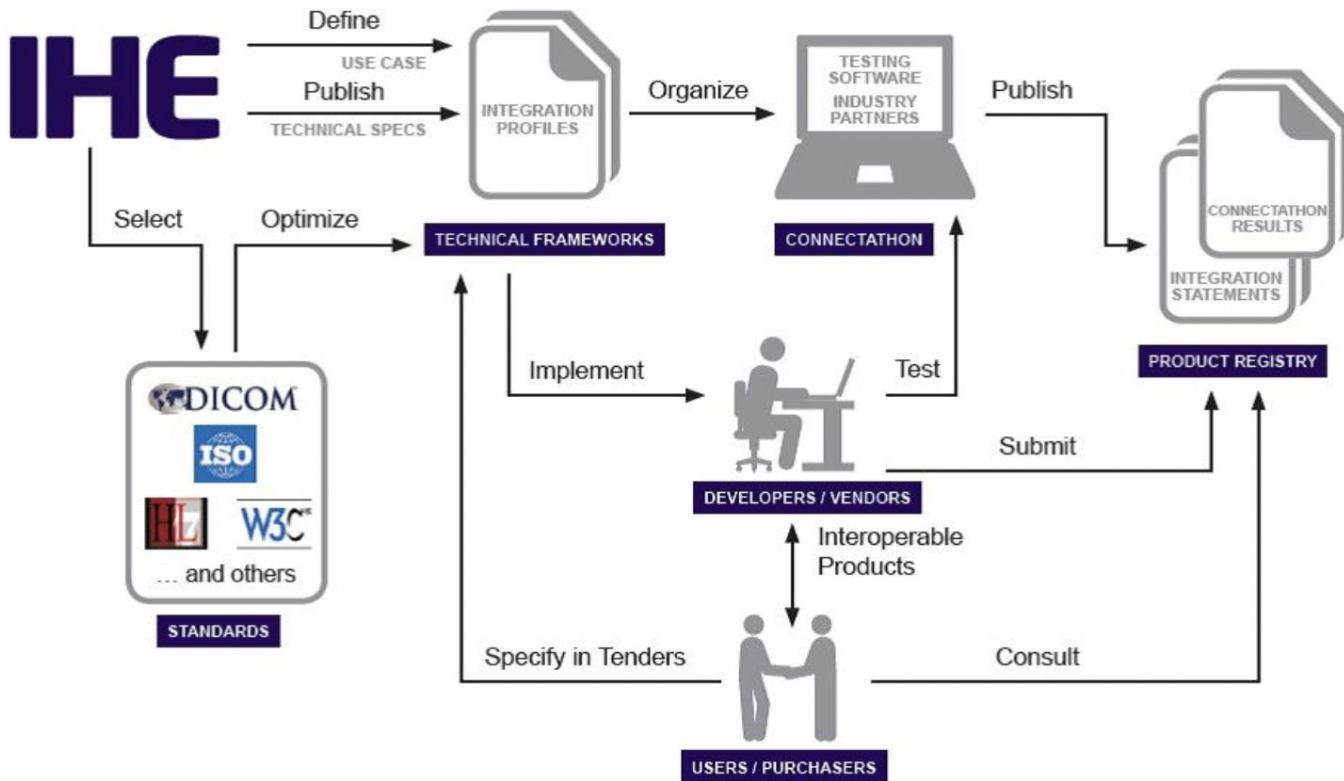
IHE PROCESO DE ADOPCIÓN

4. Cuarto paso: publicación

Los proveedores de software publican que su producto es compatible con el perfil IHE desarrollado. Se realizan demostraciones en importantes eventos de la industria, como HIMSS.



UNIVERSIDAD CES
Un compromiso con la excelencia



UNIVERSIDAD CES
Un compromiso con la excelencia

IHE TENDENCIA MUNDIAL DEL USO DE IHE

Sin duda, el crecimiento de IHE es cada vez más notable. No solo porque ya el mercado necesita imperiosamente interoperar, sino porque no hay mejor forma de hacerlo que con reglas y especificaciones claras.

IHE cuenta con más de 175 miembros en toda la organización, localizados en el mundo.



UNIVERSIDAD CES
Un compromiso con la excelencia

ARQUETIPO

Arquetipos en Salud



UNIVERSIDAD CES
Un compromiso con la excelencia

ARQUETIPO ¿QUE ES?

Un arquetipo es el primer modelo de alguna cosa. El concepto, en este sentido, puede vincularse a un prototipo: el molde original en que se produce por primera vez un objeto.

Los arquetipos son patrones de los cuales derivan otros elementos o ideas. Puede tratarse de algo físico o simbólico, siempre capaces de generar algo más a partir de sí mismos.



ARQUETIPOS EN MEDICINA

Los arquetipos clínicos son definiciones semánticas de los conceptos clínicos y por lo tanto proporcionan un enfoque sistemático a la representación de la definición de cualquier estructura de datos en una HCE; aunque el enfoque de un arquetipo si se le toma desde un punto de vista genérico, podría representar **estructuras de datos** para cualquier especialidad o área.



ARQUETIPOS EN MEDICINA

Los arquetipos representan una **declaración formal y de consenso** de las estructuras de los datos clínicos, esto debido a que especifica toda la información que un médico podría querer informar sobre un escenario clínico particular. Juega un papel importante en la determinación de cómo la **información clínica está representada y organizada dentro de las HCE**.



ARQUETIPOS EN MEDICINA

Un arquetipo proporciona una forma estandarizada de especificar jerarquías de datos clínicos de HCE y los tipos de valores de datos que se pueden almacenar dentro de cada tipo de entrada

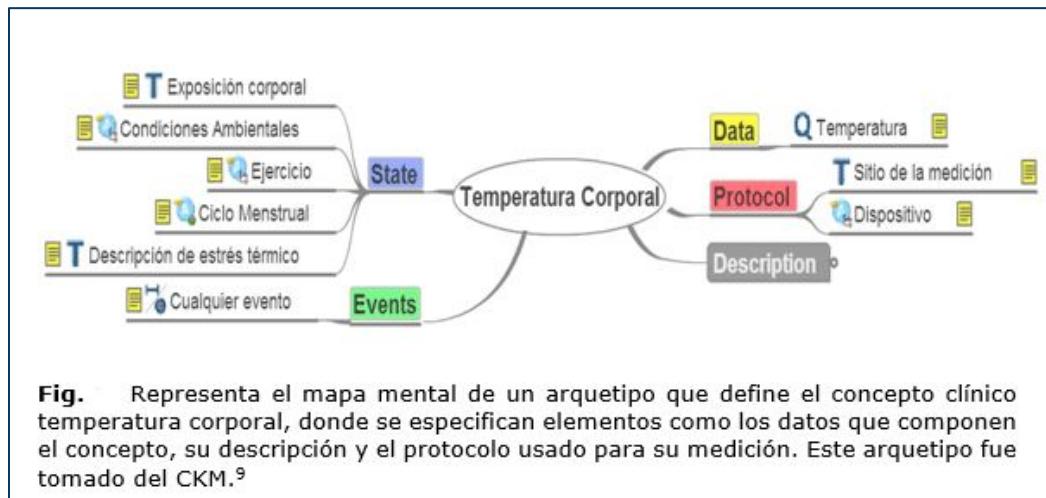
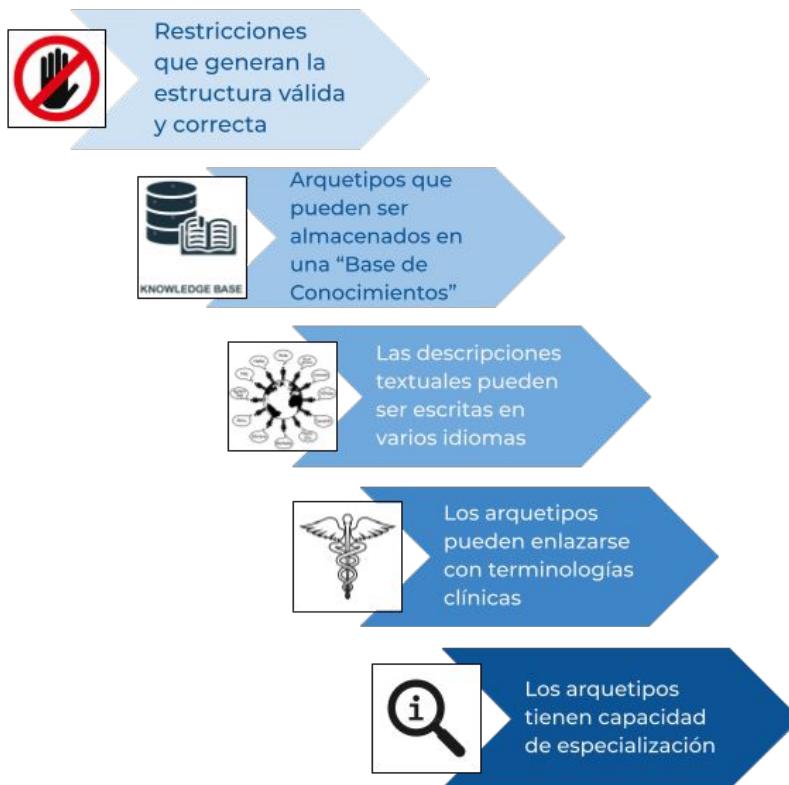


Fig. Representa el mapa mental de un arquetipo que define el concepto clínico temperatura corporal, donde se especifican elementos como los datos que componen el concepto, su descripción y el protocolo usado para su medición. Este arquetipo fue tomado del CKM.⁹



ARQUETIPOS EN MEDICINA - CARACTERÍSTICAS

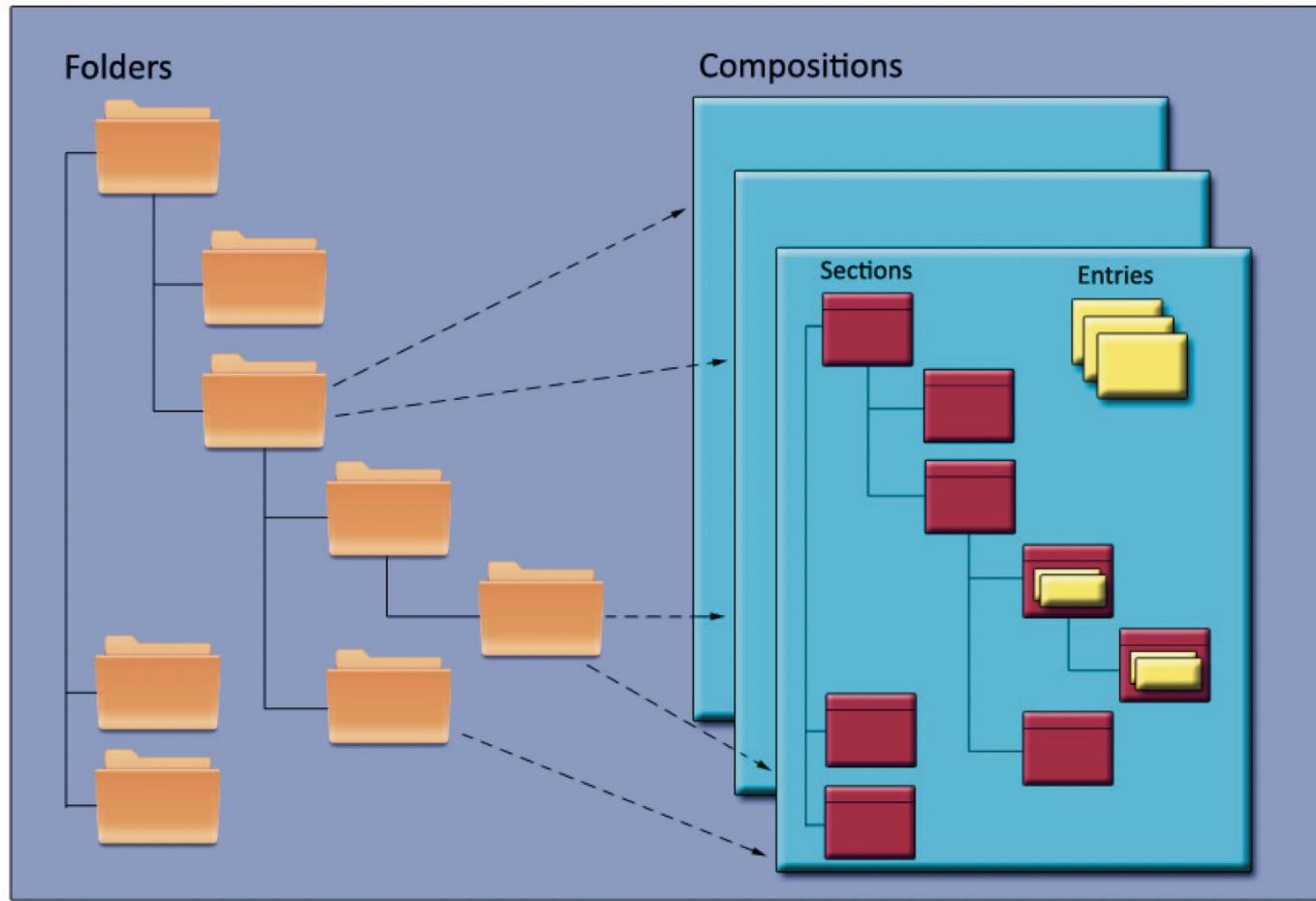


DISEÑO DE ARQUETIPOS EN MEDICINA

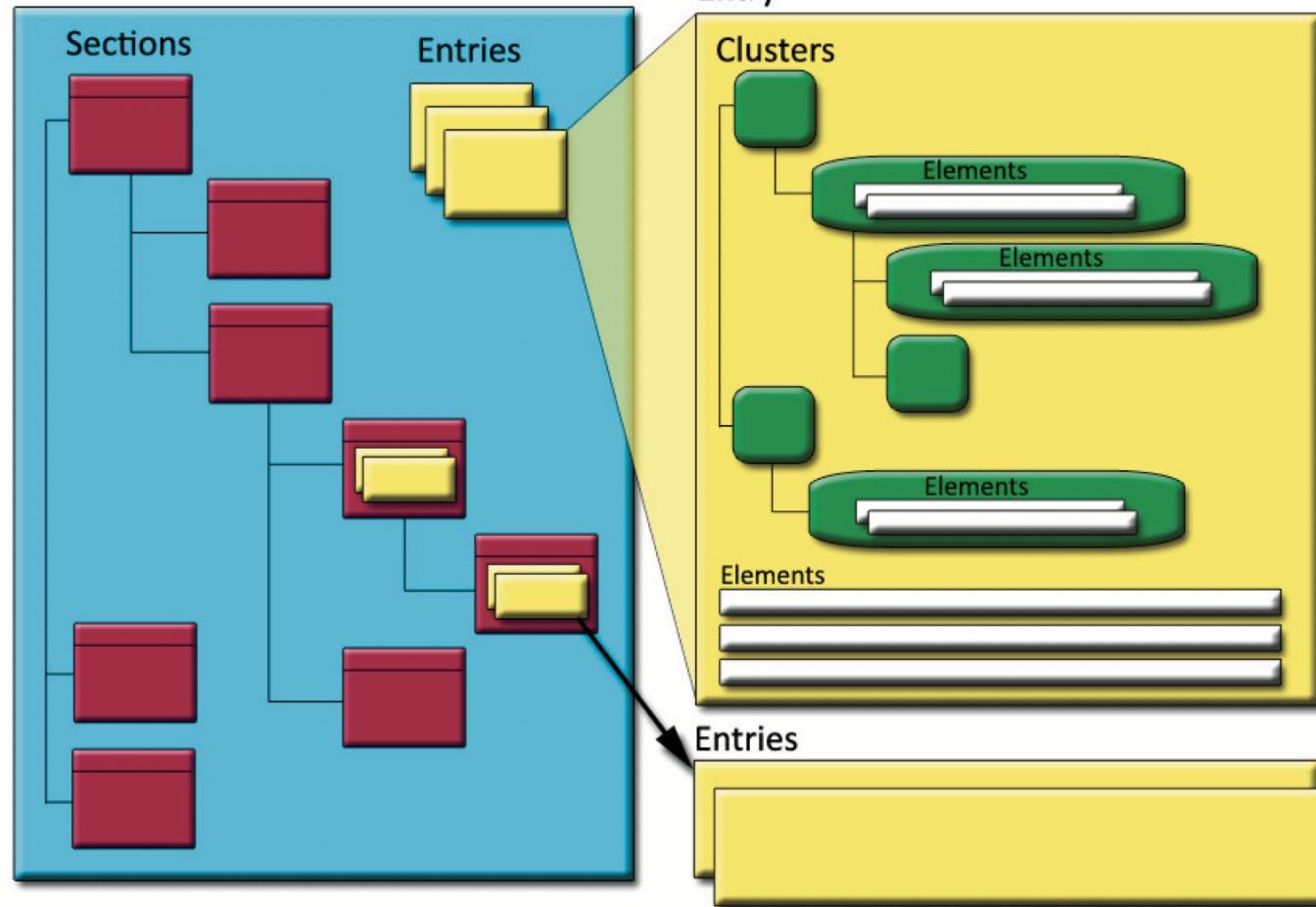
Si bien el uso de arquetipos se ha extendido rápidamente a nivel mundial, no existe formalmente una metodología que guíe su diseño por lo que se han analizado varios ejemplos de desarrollo e implementación, decidiéndose por la alternativa de varios arquetipos primarios enlazados y conformando un Arquetipo Organizativo, y no la de un único Arquetipo.



EHR_Extract



Composition



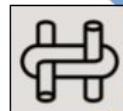
ARQUETIPOS EN MEDICINA - ELEMENTOS



EHR Extract: Contenedor global, almacena toda la HCE



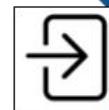
Folder: Datos de organización de alto nivel dentro de la HCE



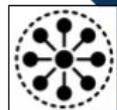
Composition: Es el conjunto de información dedicado a una HCE, como resultado de un encuentro clínico



Section: Representa los datos dentro de una composición que pertenece a un solo evento clínico.



Entry: información registrada en la HCE como resultado de una acción clínica, una observación, una interpretación, o un objetivo clínico



Cluster: Son estructuras de datos con varias partes anidadas tales como series de tiempo, lista o tabla



Element: contiene un único valor de datos.

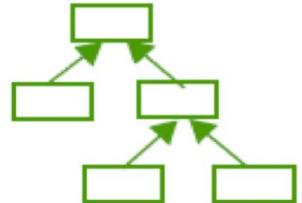


UNIVERSIDAD CES
Un compromiso con la excelencia

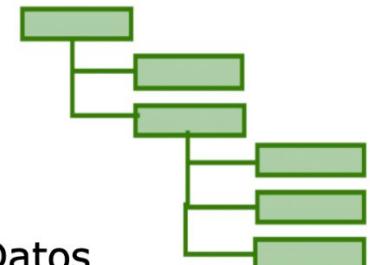
CEN/ISO 13606 Doble Modelo

Información

Modelo de Referencia



Instancias

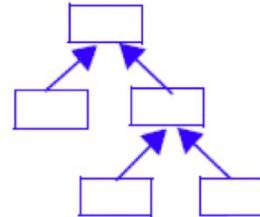


Datos

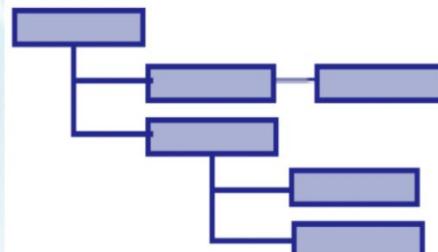
Se basa en ...

Conocimiento

Modelo de Arquetipos



Instancias



Arquetipos

Restringe en
tiempo de
ejecución



Introducción a las Bases de Datos

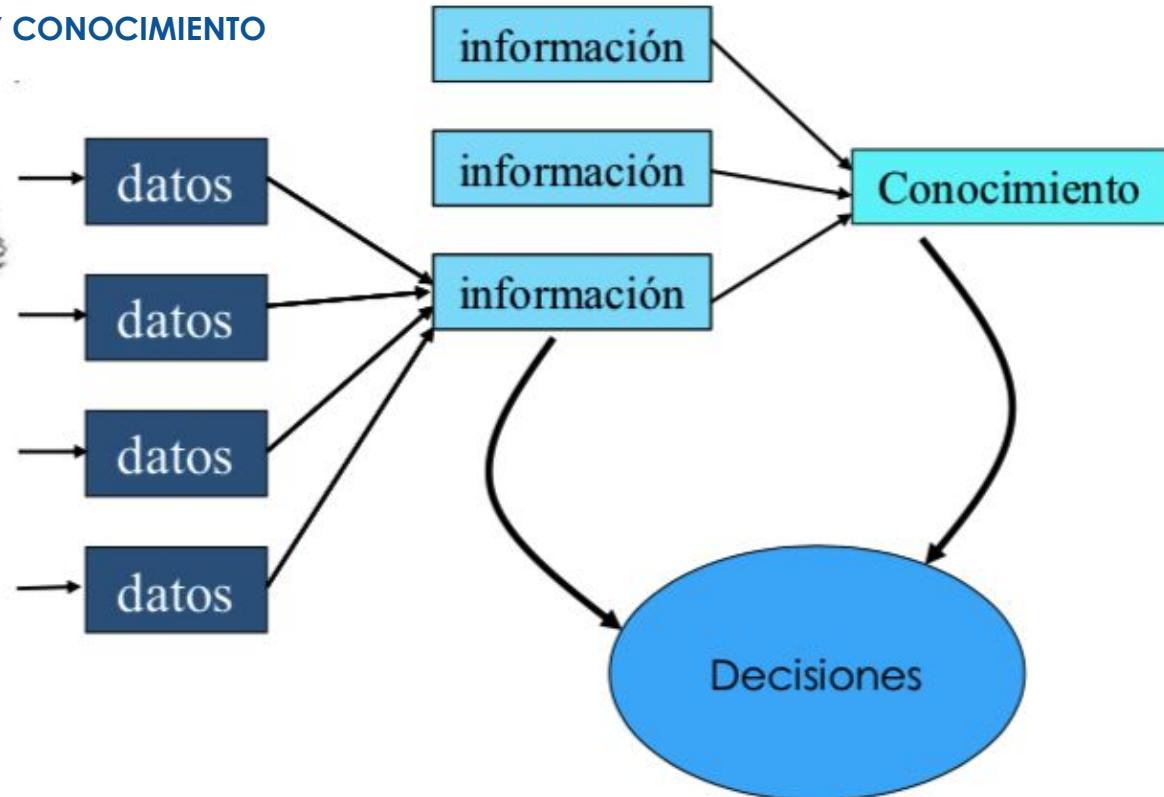
PRINCIPIOS DE LOS MODELOS DE DATOS

TIPOS DE MODELOS

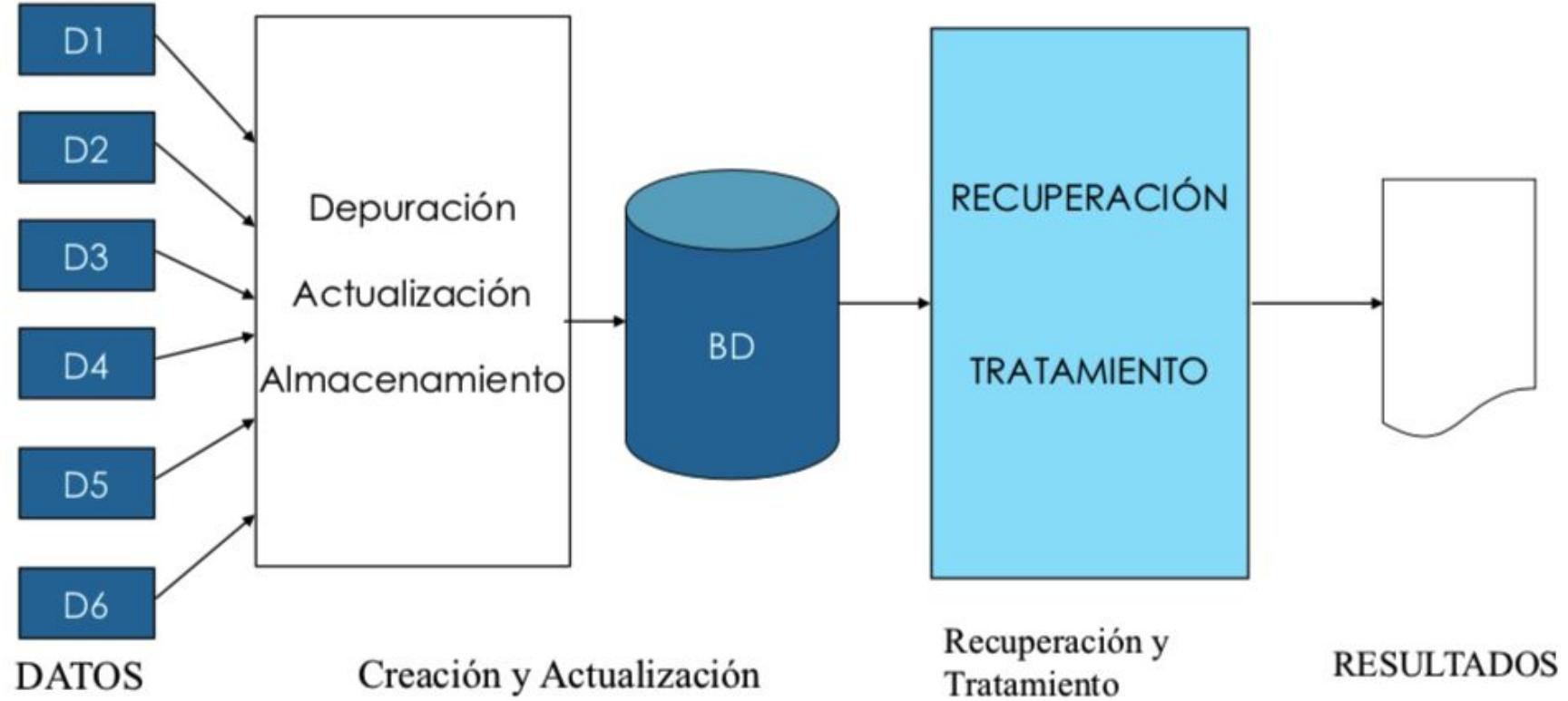
DATOS - INFORMACIÓN Y CONOCIMIENTO



Realidad



UNIVERSIDAD CES
Un compromiso con la excelencia



VENTAJAS DE LOS SISTEMAS ORIENTADOS A DATOS

- Redundancias de los datos minimizada y controlada
- Manejo de la integridad en la información
- Compartimiento de los datos en los programas
- Se mantienen las normas
- Programación fácil
- Independencia entre la estructura de los datos y los programas de búsqueda de datos



DESVENTAJAS DE LOS SISTEMAS ORIENTADOS A DATOS

- Inversión inicial elevada en software (SW), hardware (HW) y capacitación para:
 - Necesidad de SW especial
 - Necesidad de HW
- El diseño de bases de datos no es trivial.



¿QUÉ ES UNA BASE DE DATOS?

Colección o depósito de **datos integrados**, con **redundancia controlada** y con una estructura que refleje las **interrelaciones** y restricciones existentes en el mundo real; los datos que han de ser **compartidos** por diferentes usuarios y aplicaciones, deben mantenerse **independientes** de éstas, y su definición y descripción, únicas para cada tipo de datos, han de estar almacenadas junto con los mismos. Los procedimientos de **actualización** y **recuperación**, comunes y bien determinados, habrán de ser capaces de conservar la **integridad, seguridad y confidencialidad** de los datos.



EVOLUCIÓN

los medios de almacenamiento



UNIVERSIDAD CES
Un compromiso con la excelencia

EVOLUCIÓN

las bases de datos sistematizadas



UNIVERSIDAD CES
Un compromiso con la excelencia

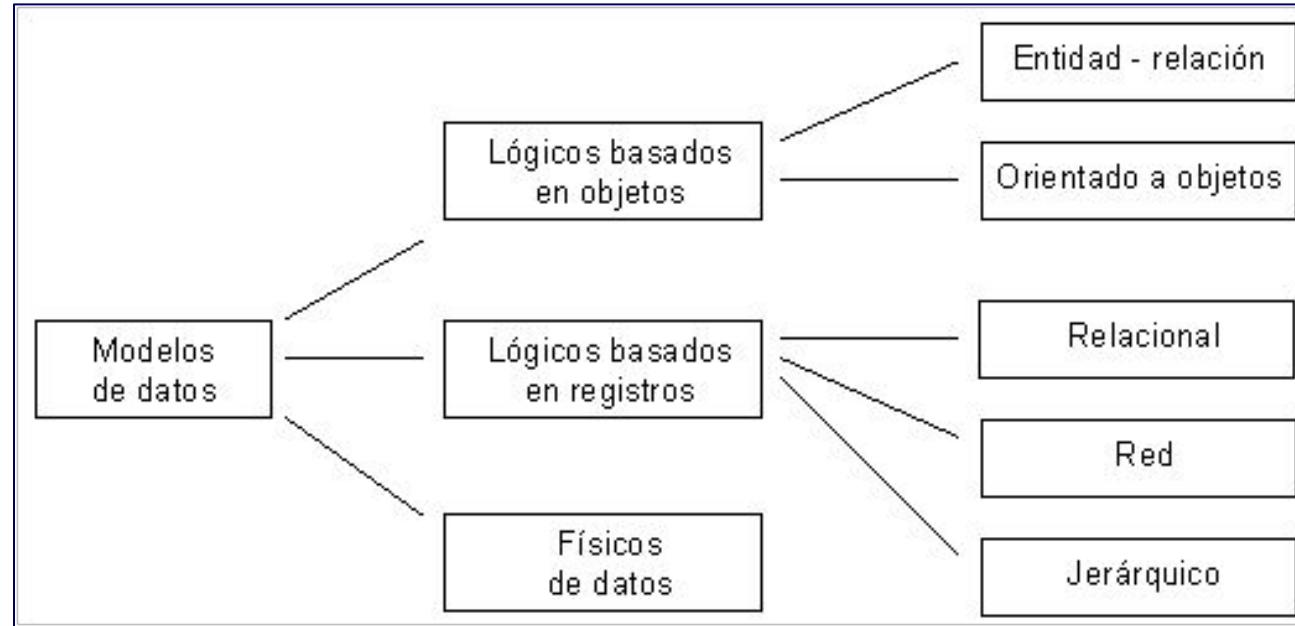
¿QUÉ ES UNA BASE DE DATOS?

- Una base de datos (BD) es una colección de información organizada de forma que un programa de computador pueda seleccionar rápidamente los fragmentos de datos que necesite.
- Las bases de datos tradicionales se organizan por campos, registros y archivos.
 - Un **campo** es una pieza única de información
 - Un **registro** es un sistema completo de campos
 - Un **archivo** es una colección de registros.

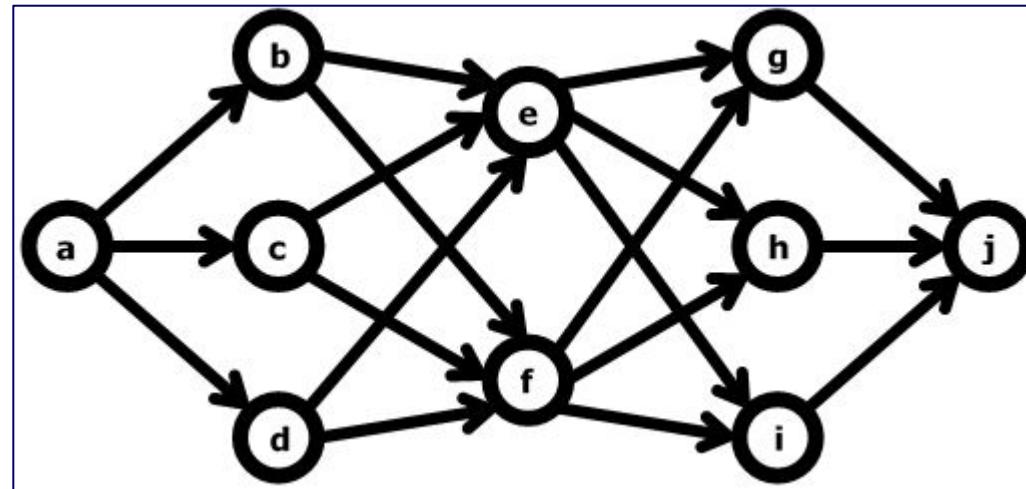


UNIVERSIDAD CES
Un compromiso con la excelencia

TIPOS DE BASES DE DATOS POR EL MODELO

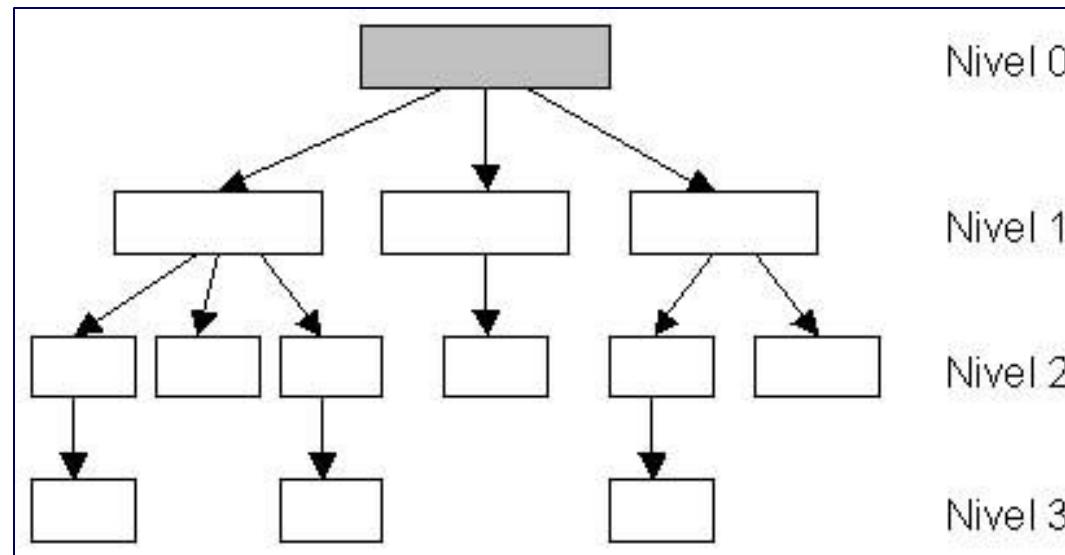


BASE DE DATOS EN RED



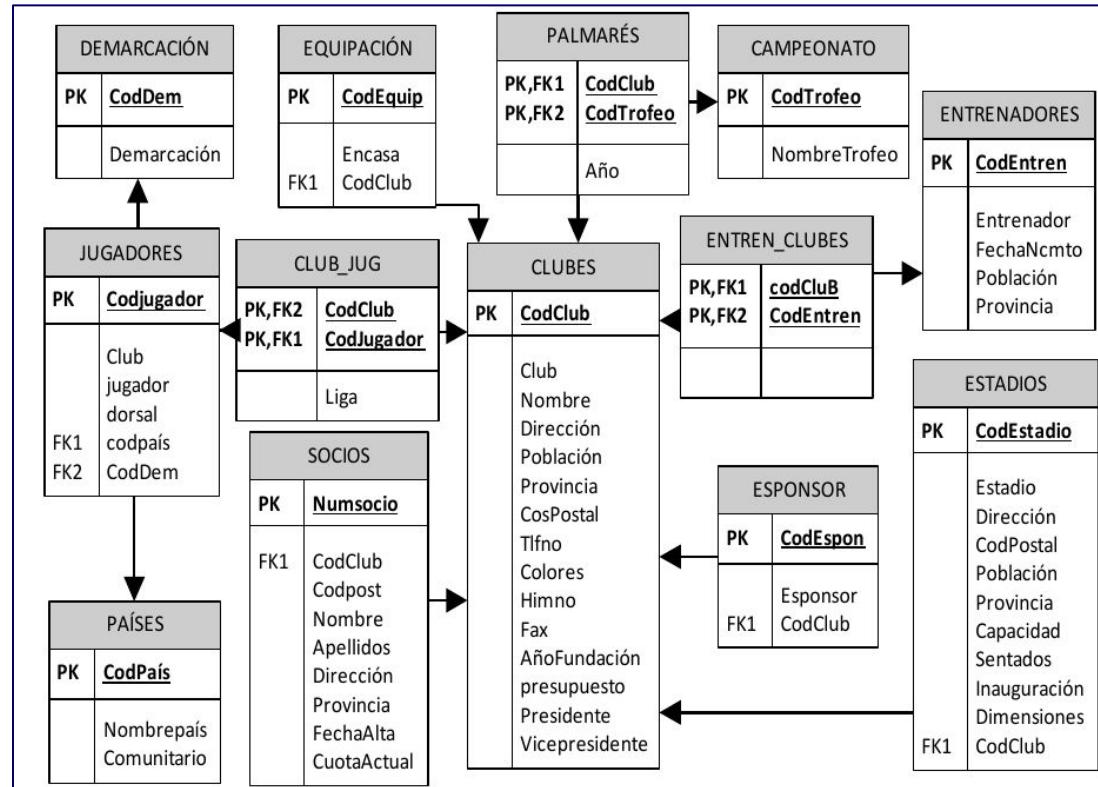
UNIVERSIDAD CES
Un compromiso con la excelencia

BASE DE DATOS JERÁRQUICA

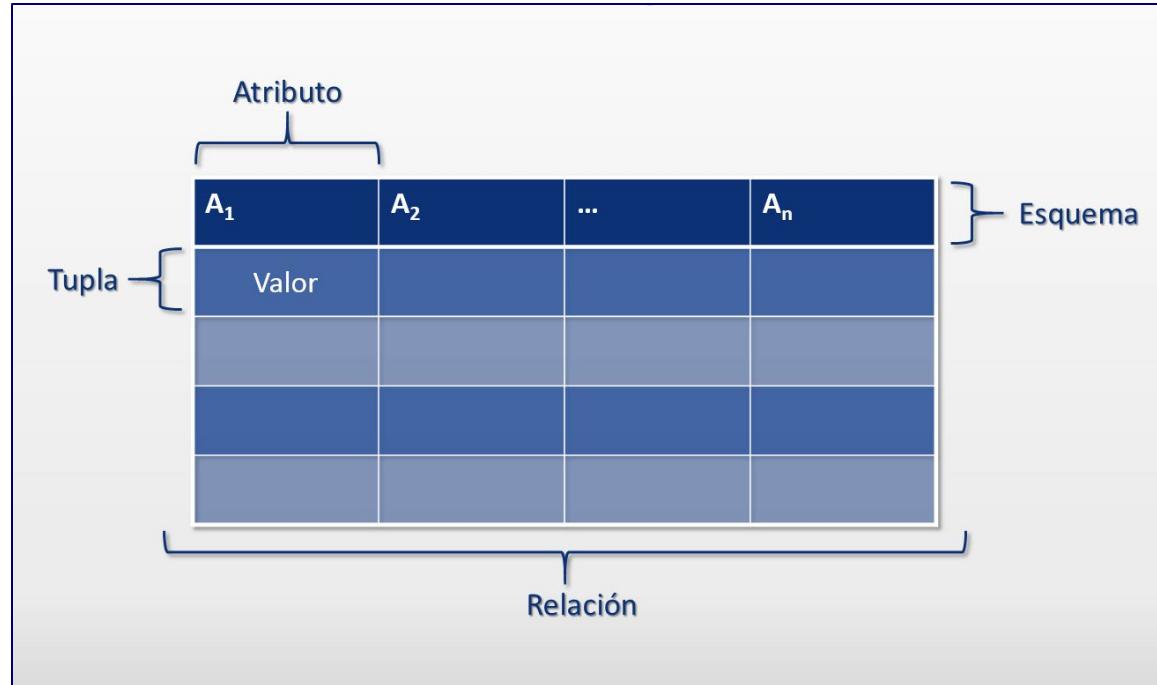


UNIVERSIDAD CES
Un compromiso con la excelencia

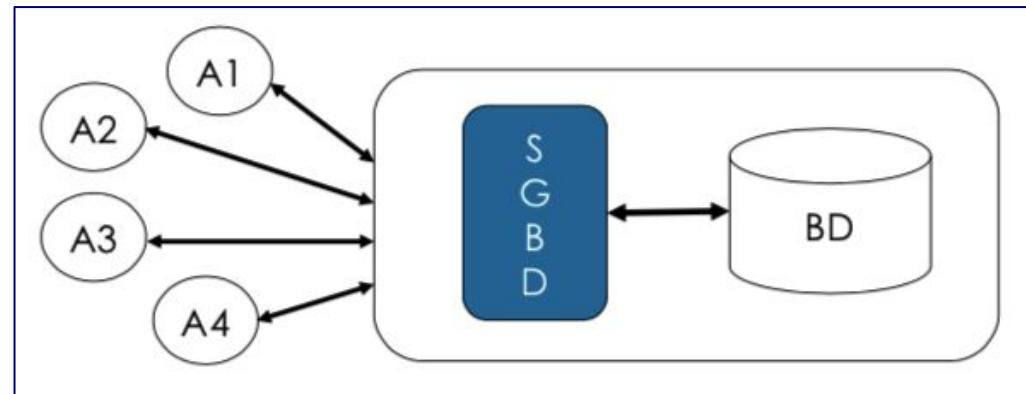
BASE DE DATOS RELACIONAL



BASE DE DATOS RELACIONAL



- **SGBD** es una colección de programas que permiten a los usuarios crear y manipular una base de datos.
- Un **SGBD** es un software de propósito general que facilita el proceso de definir, construir y manipular bases de datos de diversas aplicaciones.



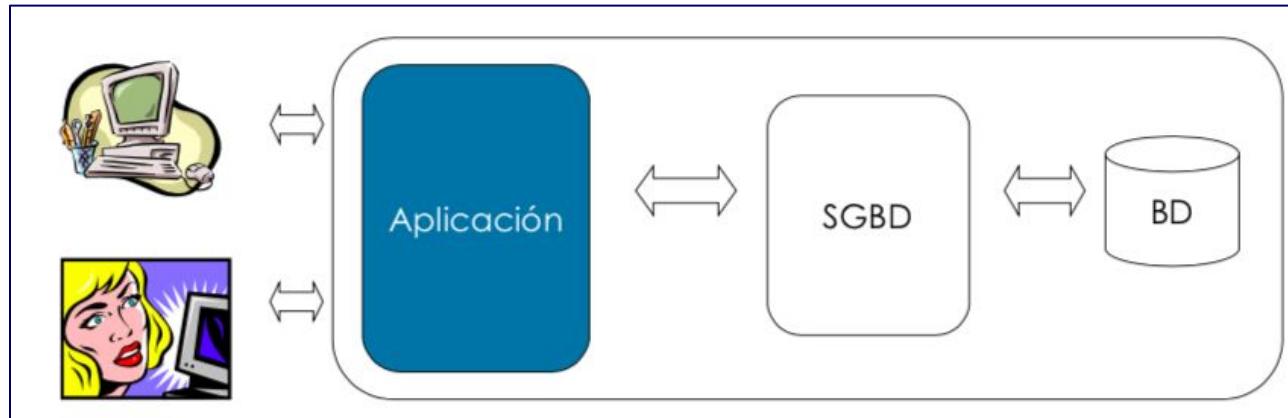


- **Ejemplos:**
 - Access.
 - SQL Server.
 - Oracle
 - Informix
 - MySQL
 - Paradox
 - Postgres



SISTEMA DE BASES DE DATOS

Un Sistema de Bases de Datos está determinado por una base de datos, un sistema de gestión de bases de datos y los programas de una aplicación determinada.



MODELOS DE DATOS

- Conjunto de Herramientas para describir:
 - Los datos
 - Las asociaciones entre los datos
 - La semántica de los datos
 - Las restricciones de los datos
 - Las operaciones para especificar cómo recuperar y modificar los datos.



MODELOS DE DATOS

Categorías

- { • Bajo Nivel (Física)
• Representacionales/ de implementación (Lógica)
• Alto Nivel(Conceptuales) }



UNIVERSIDAD CES
Un compromiso con la excelencia

MODELOS DE DATOS

Categorías

Modelos Conceptuales:

- Proporcionan representaciones de cómo las personas perciben los datos.
- Son independientes de la tecnología de implementación.
- No son directamente implementables.
- Ej. Modelo Entidad – Relación



UNIVERSIDAD CES
Un compromiso con la excelencia

Categorías

Modelos Representacionales :

Esconden detalles de almacenamiento físico, pero pueden ser directamente implementados en un SGBD.

Incluyen:

Modelos que representan datos utilizando Estructuras de Registros.

Ej. Modelo jerárquico, Modelo de red, Modelo relacional.

Nuevas familias de modelos de implementación de alto nivel. Ej. Modelo OO.



MODELOS DE DATOS

Categorías

Modelos de Bajo Nivel:

Describen detalles de forma de almacenar los datos, representando la información como:

Formato de los registros.

Orden de los registros.

Caminos de acceso, etc.



MODELOS DE DATOS

Es importante distinguir entre:

Modelo:

Conjunto de herramientas para describir los datos.

Esquema de una BD :

Descripción de una BD, generada a partir del modelo (Diagrama de esquema: forma de visualización).

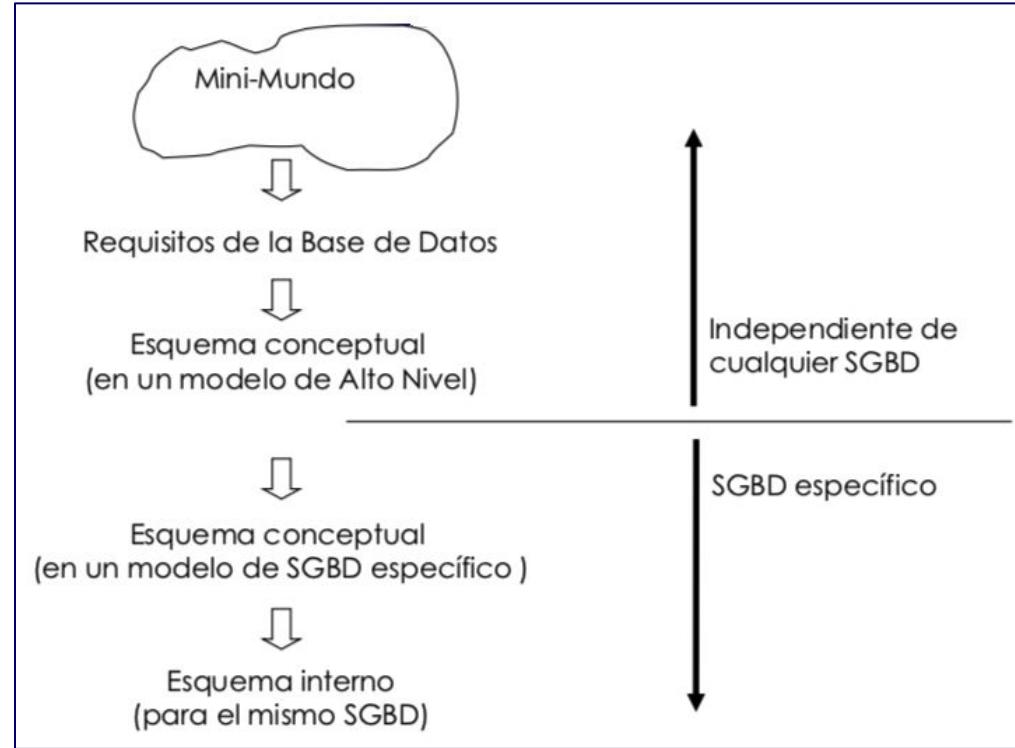
Instancia de una BD :

Los datos de la base de datos en un momento de tiempo particular.



UNIVERSIDAD CES
Un compromiso con la excelencia

PROYECTO DE BASE DE DATOS



LENGUAJES DE BASE DE DATOS

Lenguaje de definición de datos

(Data Definition Language – DDL)

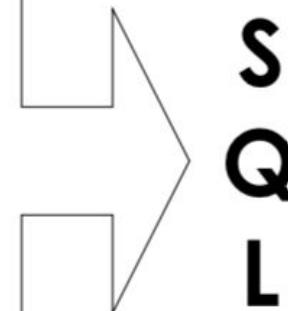
Es utilizado para definir los esquemas

Lenguaje de manipulación de datos

(Data Manipulation Language – DML)

Es utilizado por los usuarios para manipular datos.

Manipular : recuperación, inserción, eliminación, modificación



Structured Query Language

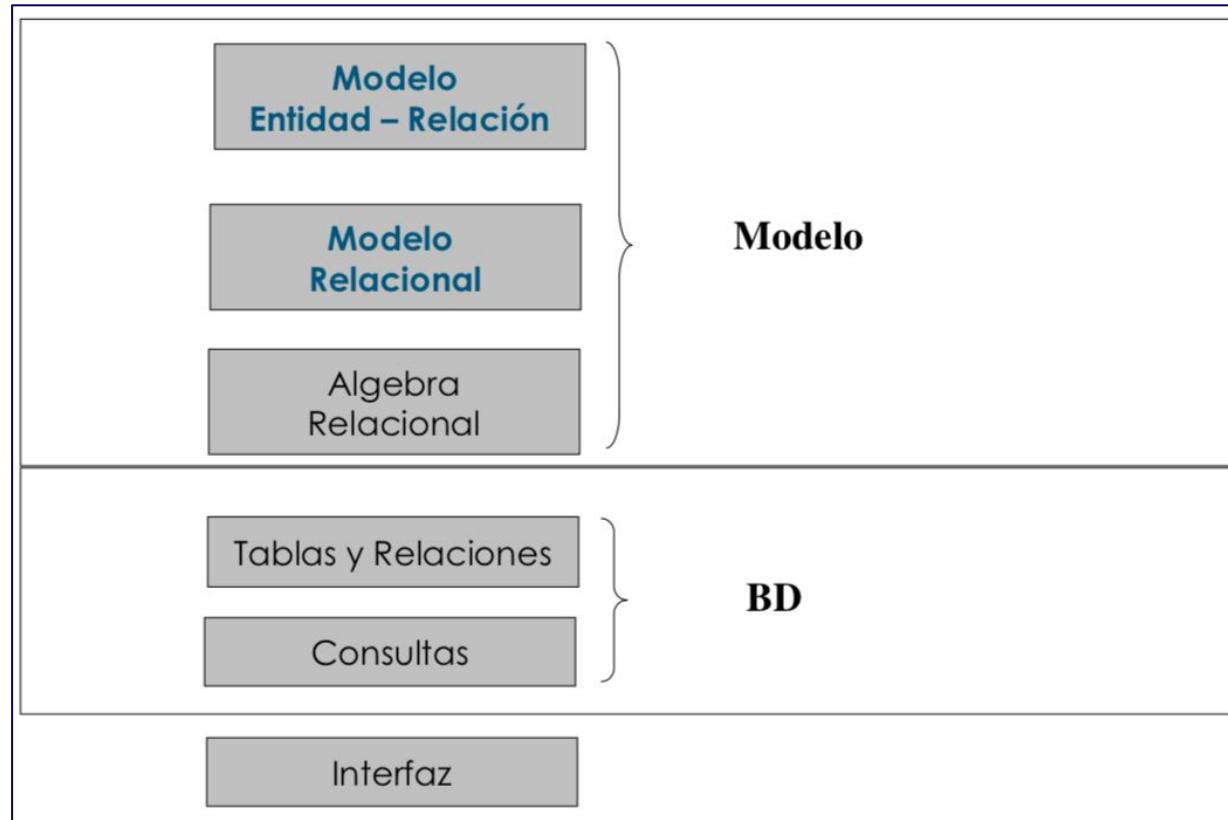
Unifica DDL y DML en un único lenguaje;

Es un lenguaje no procedimental;

Engloba otros aspectos, como por ejemplo los relacionados con administración de BD;

SQL es un standard de ISO (International Standards Organization) y ANSI (American National Standards Institute)





FASES EN EL DISEÑO DE BASES DE DATOS

1. Análisis de requisitos del problema
2. Diseño conceptual (Esquema de alto nivel)
3. Diseño Lógico (Independiente del SGBD)



ENTIDAD

- Cualquier objeto (real o abstracto) sobre el cual queremos tener información y que tiene existencia por sí mismo.
- "...una persona, lugar, cosa, concepto o suceso, real o abstracto de interés para la empresa", ANSI, 1977
- Puede ser un objeto con una existencia física, ej. carro, empleado, producto, alumno, etc.
- O un objeto con existencia conceptual, ej. curso, profesión, etc.

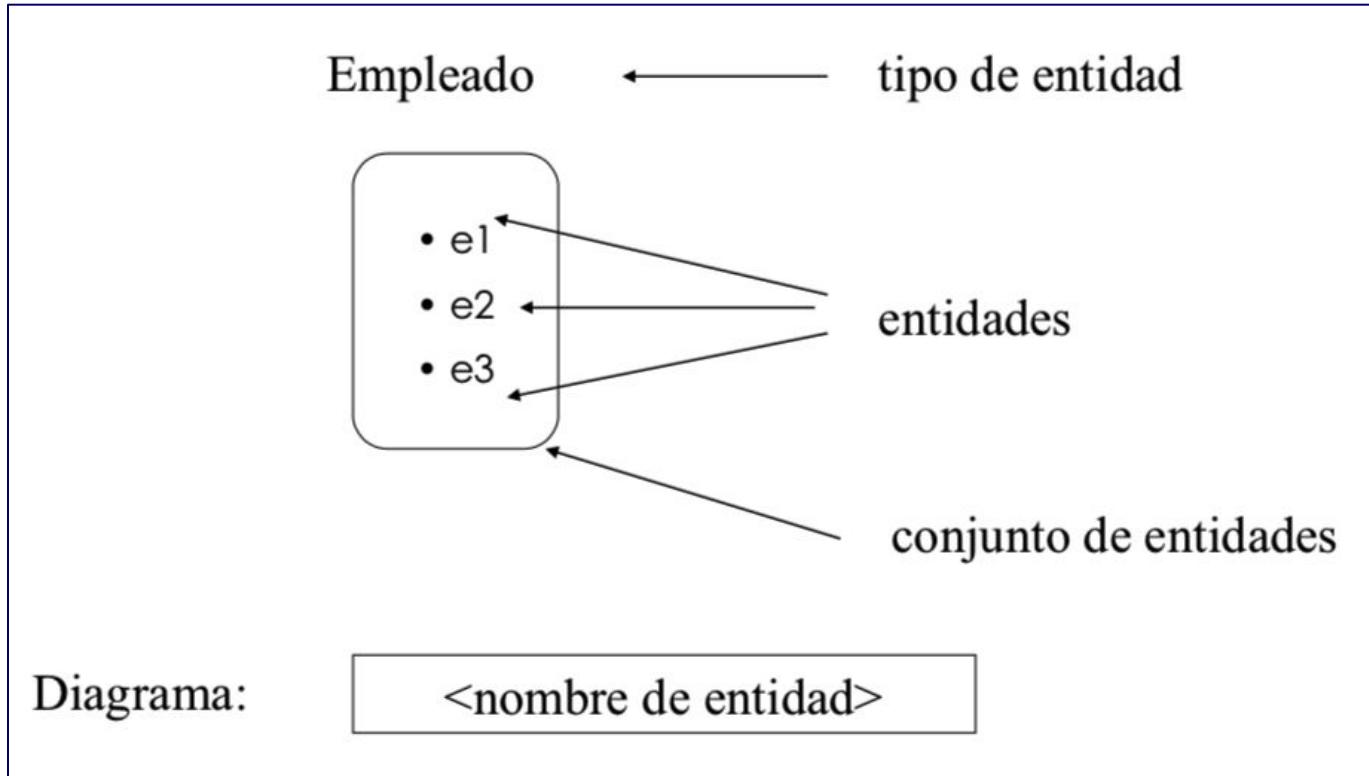


ENTIDAD

CONCEPTOS	EJEMPLOS
Tipo de entidad (estructura genérica)	Paciente
Entidad o Instancia (instancia del tipo)	"Juan <u>Pérez</u> <u>Díaz</u> "
Conjunto de entidades $\{e \mid p(e)\}$ Donde: e instancia del tipo de entidad E p(e) predicado asociado a E	Persona que se trata en un hospital



ENTIDAD



ENTIDAD

Representación gráfica:

Empleado

Reglas que debe cumplir una entidad: (Tardieu 1979)

- tener existencia propia
- cada ocurrencia de un tipo de entidad debe distinguirse de las demás
- todas las ocurrencias de un tipo de entidad deben tener las mismas características

Entidades:

regulares o fuertes: existen por sí mismas

débiles : su existencia depende de otro tipo de entidad

Representación:



UNIVERSIDAD CES

Un compromiso con la excelencia

ATRIBUTOS

- ▶ Propiedades que describen a las entidades

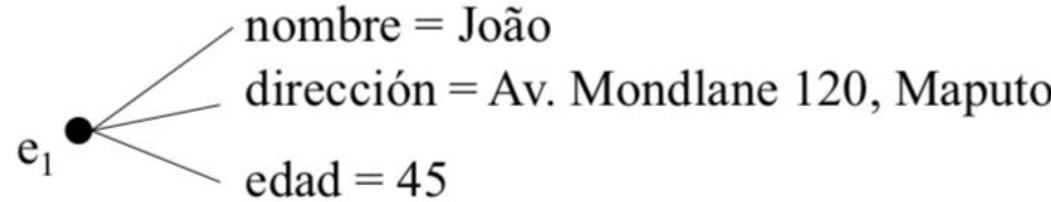
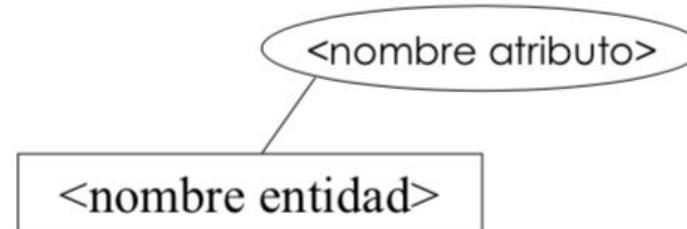


Diagrama:



ATRIBUTOS

- ▶ **Compuestos**
 - ▶ Atributos que pueden ser divididos en subpartes con significados
 - ▶ **Simples o atómicos**
 - ▶ Atributos que no son divisibles
-
- ▶ **Univvalorados**
 - ▶ Atributos que tienen apenas un único valor.
 - ▶ **Multivvalorados**
 - ▶ Atributos que pueden tener un conjunto de valores.



UNIVERSIDAD CES
Un compromiso con la excelencia

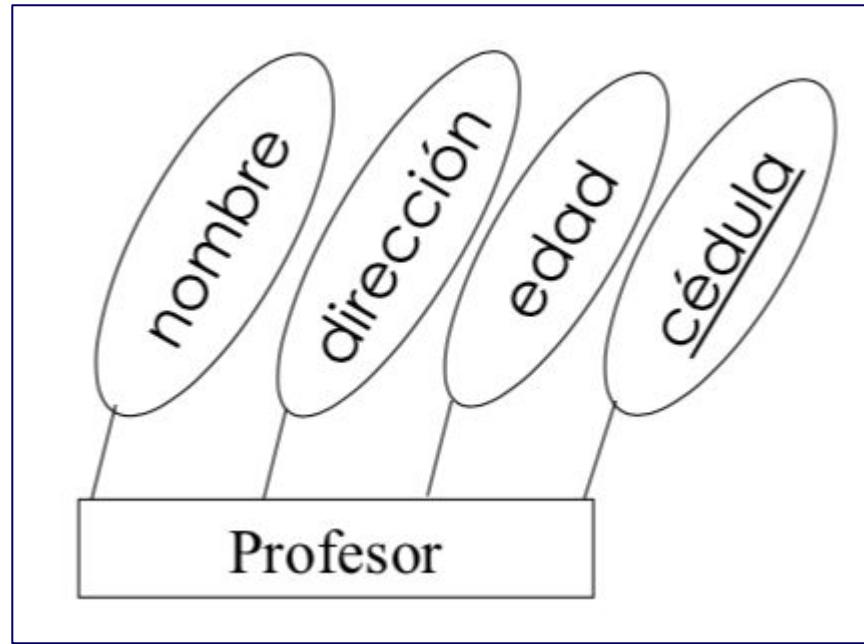
ATRIBUTOS

- ▶ Un tipo de entidad tiene, normalmente, atributos cuyos valores son distintos para cada entidad. Tal atributo (o atributos) es llamado atributo(s) llave, y su valor puede ser usado para identificar cada entidad.

- ▶ Algunos tipos de entidades pueden tener más de un atributo llave llamadas llaves candidatas, de ellas es seleccionada una para ser la llave primaria



ATRIBUTOS



UNIVERSIDAD CES
Un compromiso con la excelencia

RELACIONES

- ▶ Relación

- ▶ Es una asociación entre varias entidades

- ▶ $r_i = (e_1, e_2, \dots, e_n)$

- ▶ Conjunto de relaciones

- ▶ $R = \{(e_1, e_2, \dots, e_n) \mid e_1 \in E_1, e_2 \in E_2, \dots, e_n \in E_n\}$

- ▶ Tipo de relaciones

- ▶ Abstracción para describir relaciones



RELACIONES

- ▶ Indica el número de tipos de entidades participantes.
- ▶ Pueden ser: binarias, ternarias, unarias.

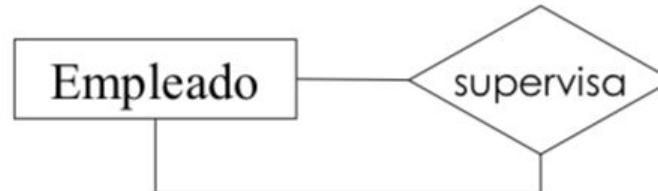
Diagrama:



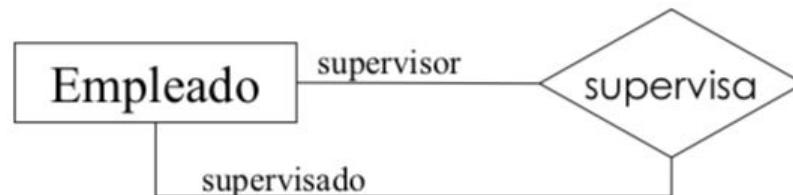
UNIVERSIDAD CES
Un compromiso con la excelencia

RELACIONES

- ▶ Unarias o reflexiva

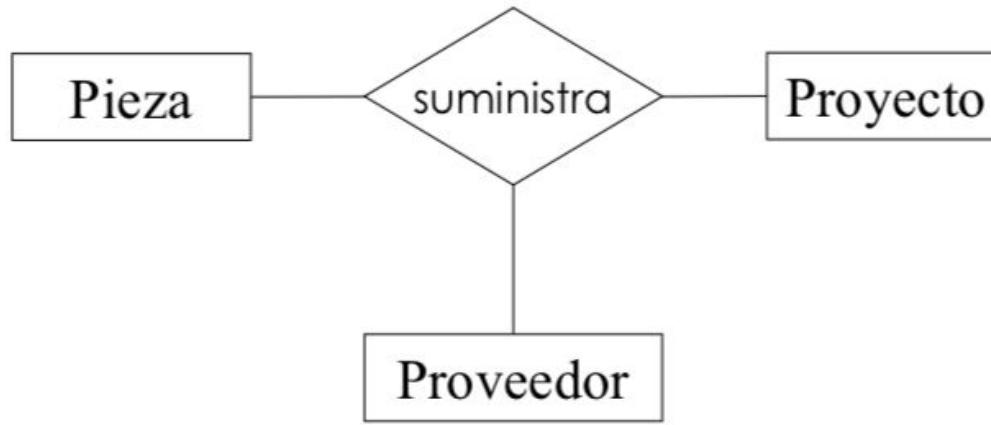


- ▶ El tipo de relación supervisa relaciona un empleado con su supervisor.
- ▶ En las relaciones unarias se debe indicar el papel de cada entidad:



RELACIONES

► Ternaria

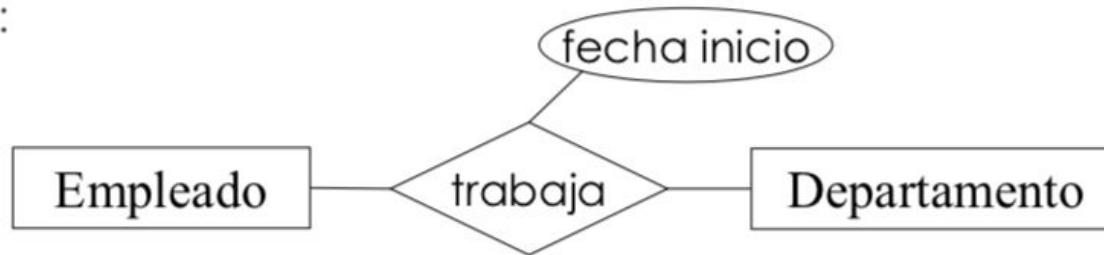


- El proveedor **s** suministra una pieza **p** para el proyecto **j**.



RELACIONES

- ▶ Un atributo puede ser una propiedad de un tipo de relación.
- ▶ Ejemplo:

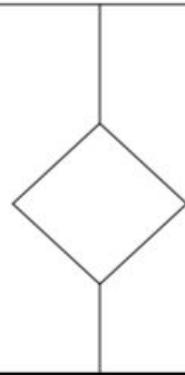


- ▶ El empleado trabaja en el departamento a partir de una fecha determinada.



RELACIONES

e. fuerte



e. débil

Su existencia **depende de la existencia** de la entidad fuerte



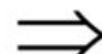
UNIVERSIDAD CES

Un compromiso con la excelencia

RELACIONES

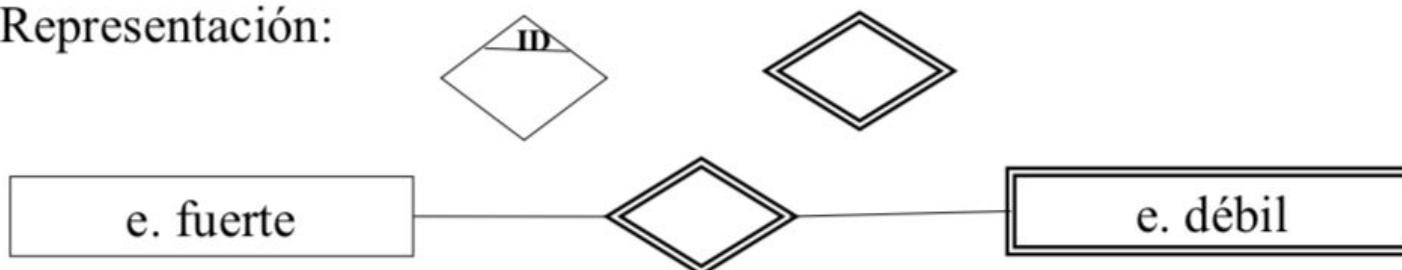
- ▶ Si el tipo de entidad débil no posee los atributos requeridos para identificar a sus instancias.

Dependencia de
Identificación



Dependencia de
Existencia

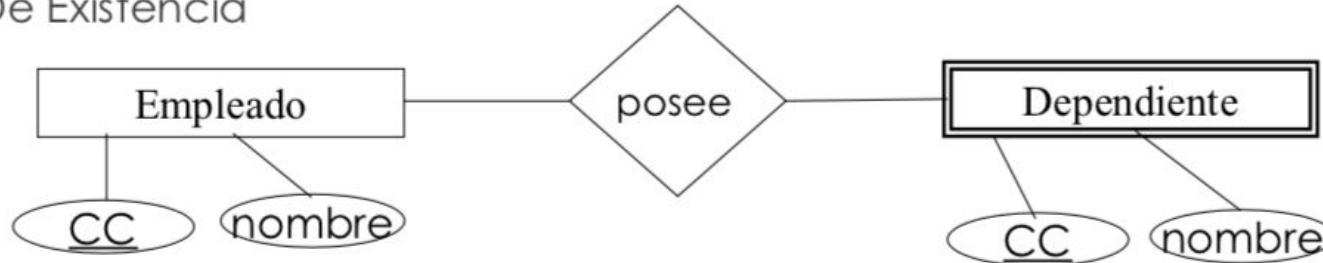
Representación:



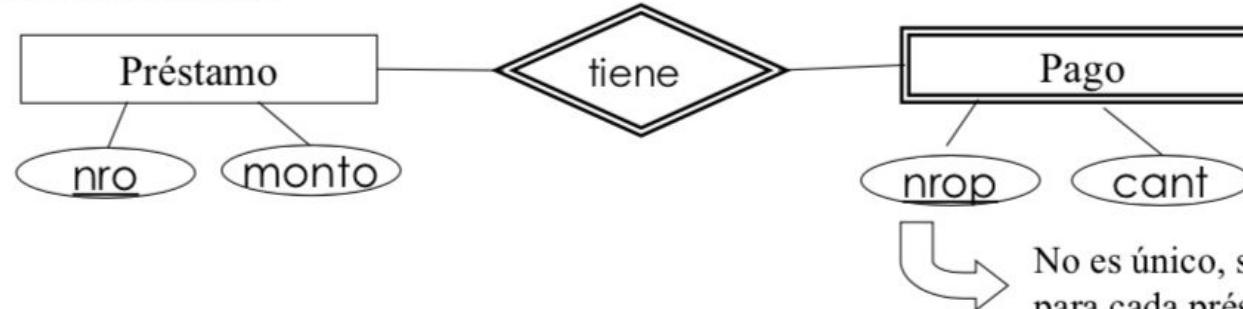
UNIVERSIDAD CES
Un compromiso con la excelencia

RELACIONES

► De Existencia



► De Identificación



No es único, sino único
para cada préstamo



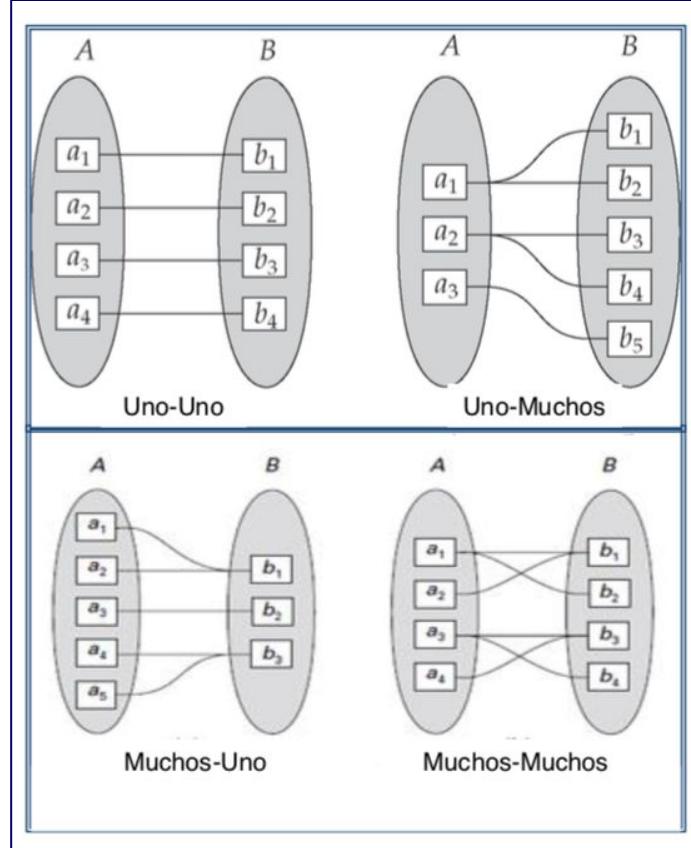
RELACIONES

Cardinalidad

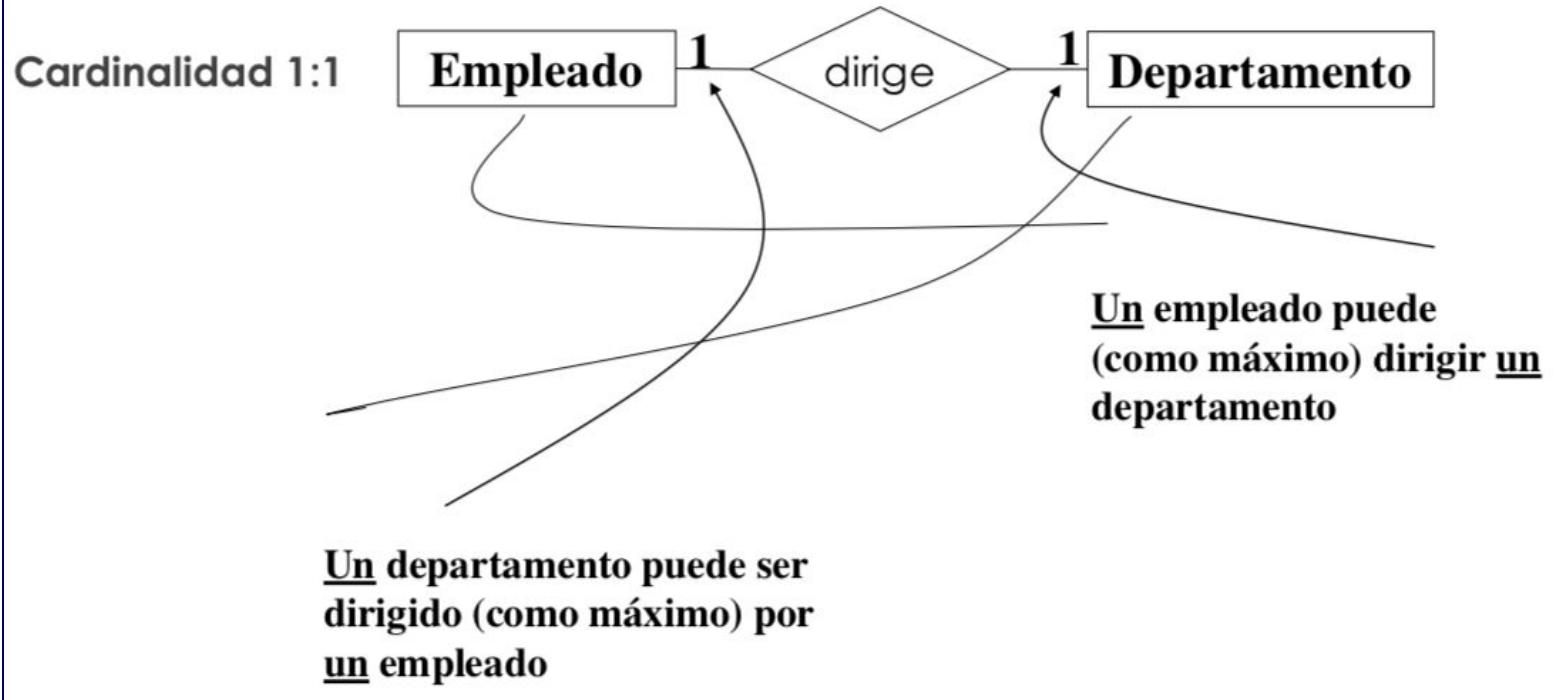
- ▶ Restringe el número de entidades con las cuales puede estar asociada a otra entidad en un determinado conjunto de relaciones
 - ▶ Especifica la cantidad máxima de relaciones en que una entidad puede participar.
 - ▶ Las cardinales pueden ser:
 - ▶ 1 : 1 (uno - uno)
 - ▶ 1 : M (uno - muchos)
 - ▶ M : M (muchos - muchos)



RELACIONES



RELACIONES



RELACIONES

Cardinalidad 1:M



Un empleado puede
(como máximo) trabajar
en un departamento

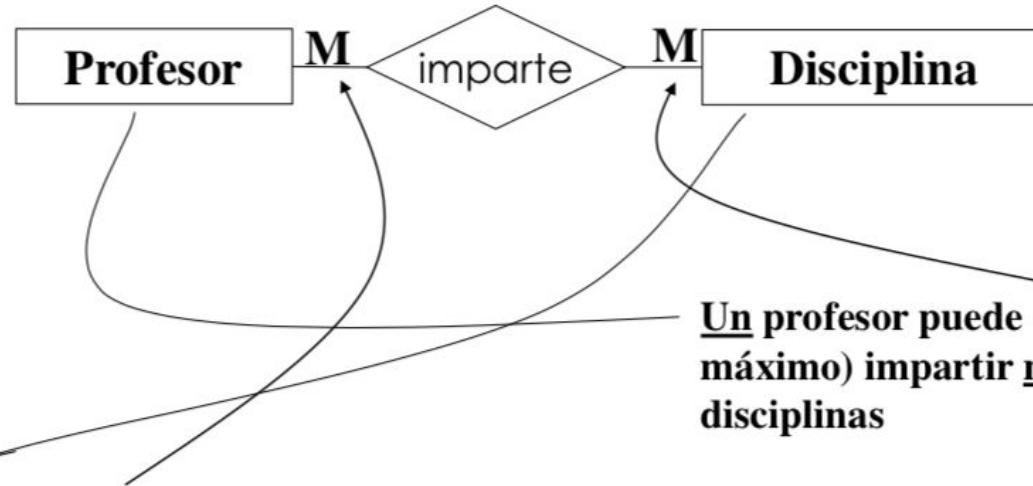
En un departamento pueden
trabajar (como máximo)
muchos empleados



UNIVERSIDAD CES
Un compromiso con la excelencia

RELACIONES

Cardinalidad M:M



Un profesor puede (como máximo) impartir muchas disciplinas

Una disciplina puede ser impartida (como máximo) por muchos profesores



UNIVERSIDAD CES
Un compromiso con la excelencia

RELACIONES

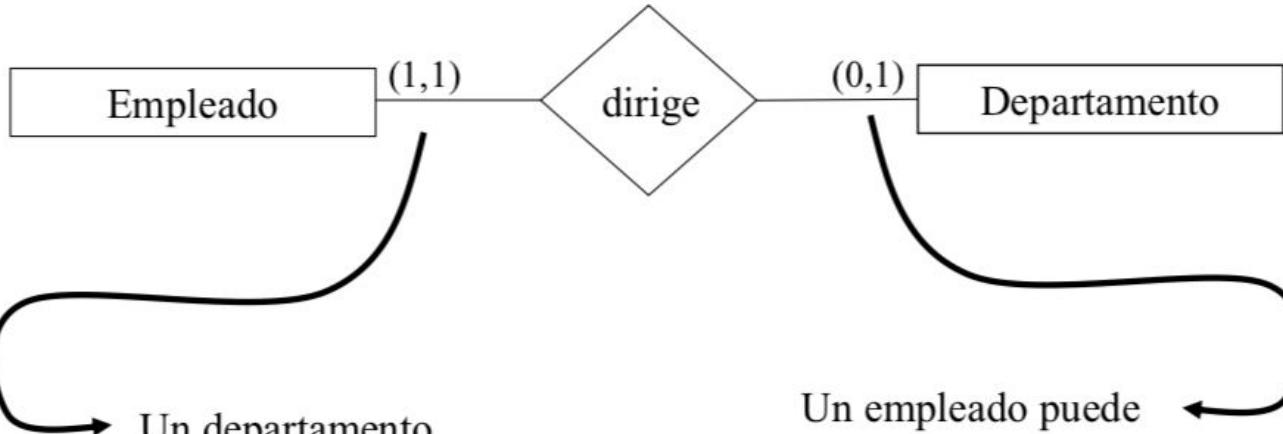
- ▶ Las cardinalidades anteriores se pueden especificar mejor utilizando esta notación.
- ▶ Notación (cardinalidad mínima, cardinalidad máxima)
- ▶ Cardinalidad mínima
 - ▶ Número mínimo de entidades con las cuales puede estar asociada otra entidad en un conjunto de relaciones.
- ▶ Cardinalidad máxima
 - ▶ Número máximo de entidades con las cuales puede estar asociada otra entidad en un conjunto de relaciones.



UNIVERSIDAD CES
Un compromiso con la excelencia

RELACIONES

- ▶ Ejemplo:



Un departamento debe ser dirigido por lo menos por **un** empleado y como máximo por **un** empleado

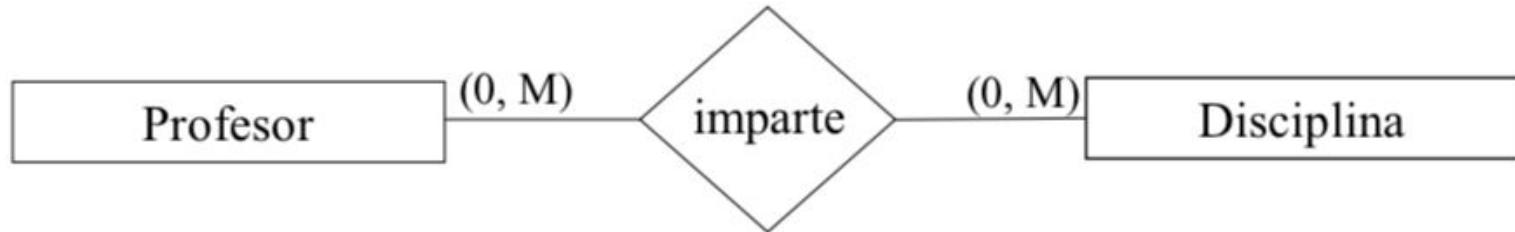
Un empleado puede que no dirija **ningún** (0) departamento y como máximo puede dirigir uno.



UNIVERSIDAD CES
Un compromiso con la excelencia

RELACIONES

- ▶ Ejemplo:



RELACIONES

- ▶ Ejemplo:



ESPECIALIZACIÓN - GENERALIZACIÓN

► Especialización

- Método de diseño descendente; designamos subgrupos (subconjuntos) dentro de un conjunto de entidades que son distintas de otras entidades en ese conjunto.

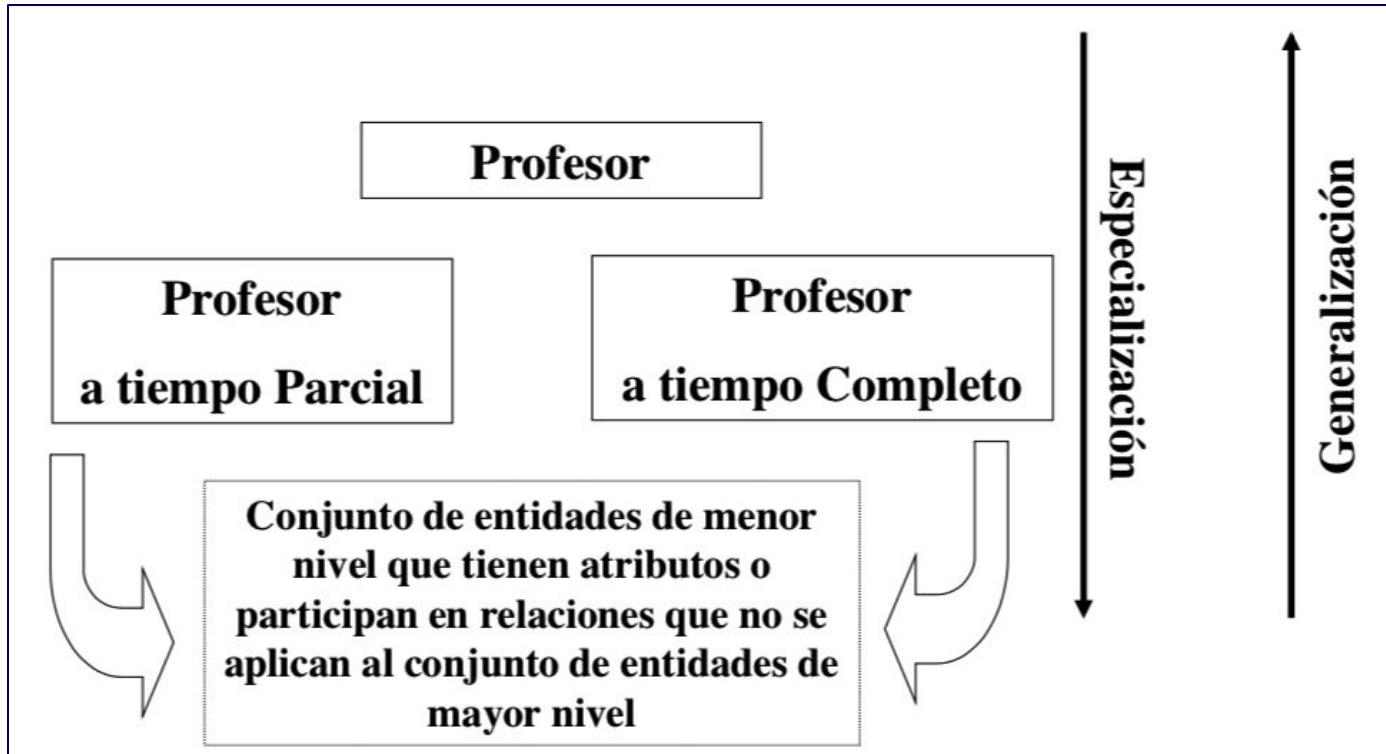
► Generalización

- Método de diseño ascendente; combinamos un conjunto de entidades de menor nivel en un conjunto de entidades que comparten las mismas características.



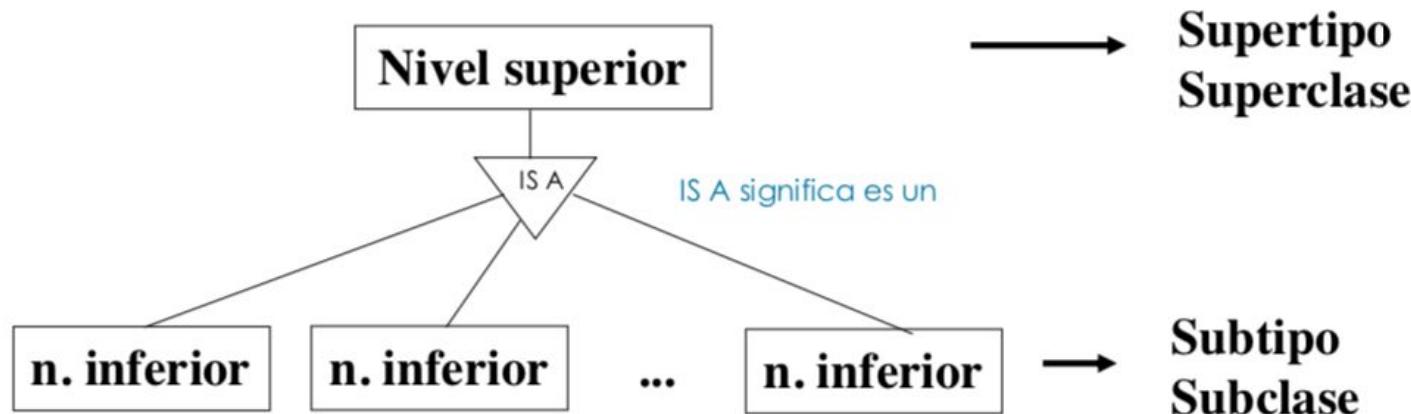
UNIVERSIDAD CES
Un compromiso con la excelencia

ESPECIALIZACIÓN - GENERALIZACIÓN



ESPECIALIZACIÓN - GENERALIZACIÓN

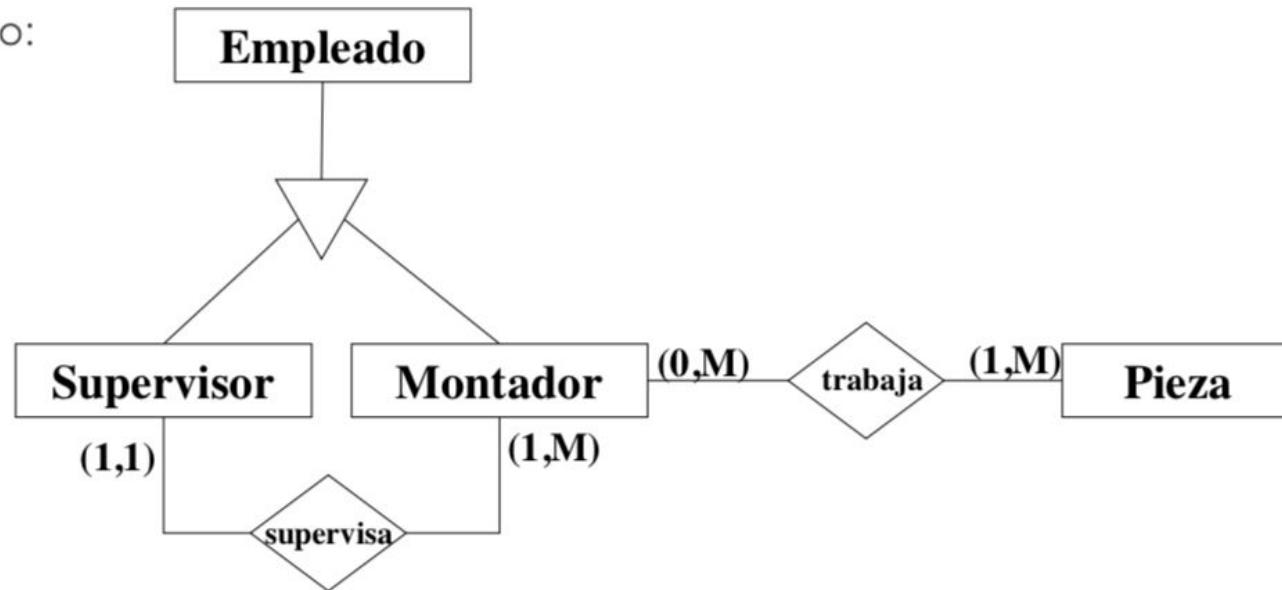
Notación



UNIVERSIDAD CES
Un compromiso con la excelencia

ESPECIALIZACIÓN - GENERALIZACIÓN

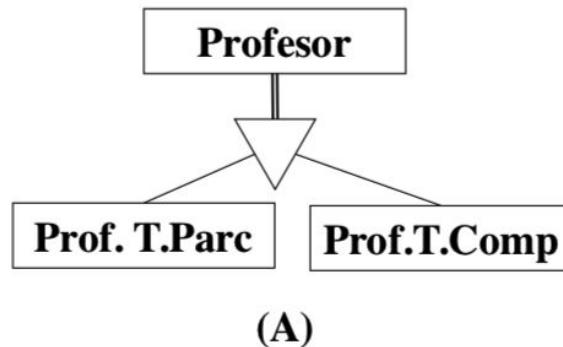
► Ejemplo:



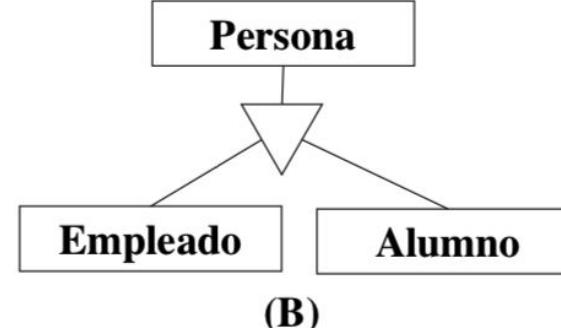
ESPECIALIZACIÓN - GENERALIZACIÓN

► Restricción de pertenencia

- A. Mutuamente exclusiva
- B. No mutuamente exclusiva



(A)



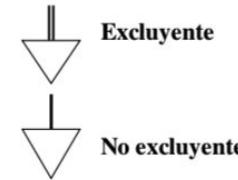
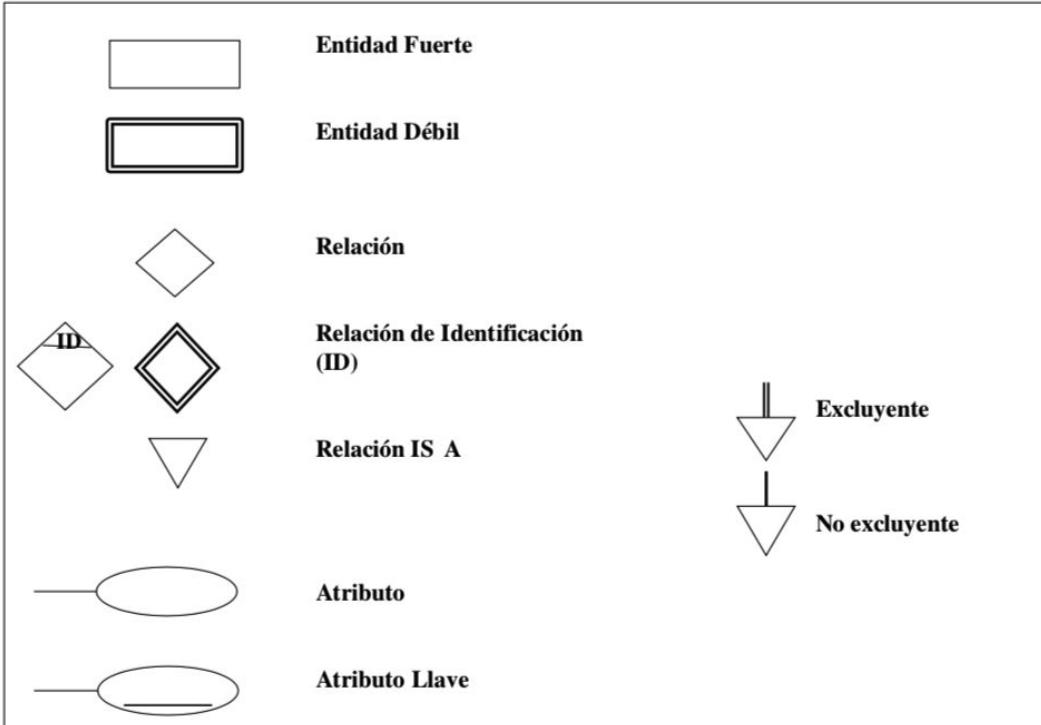
(B)



UNIVERSIDAD CES
Un compromiso con la excelencia

MODELO ER

Resumen MER



UNIVERSIDAD CES
Un compromiso con la excelencia

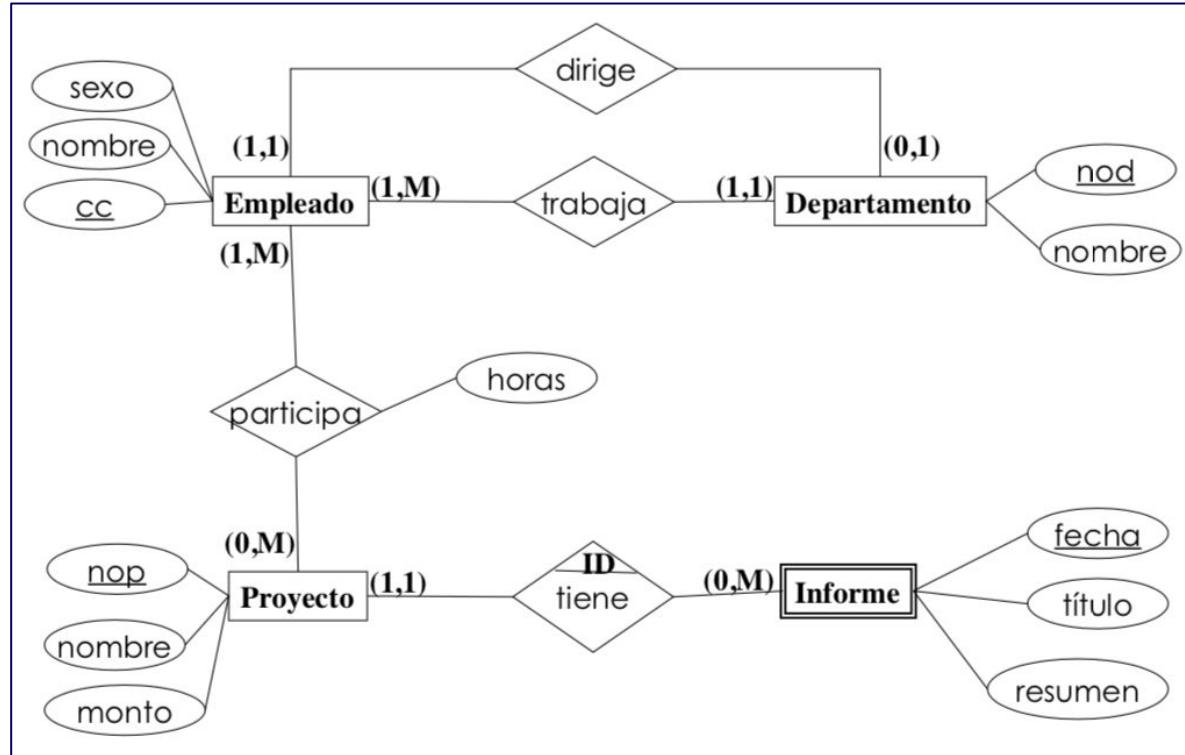
EJERCICIO

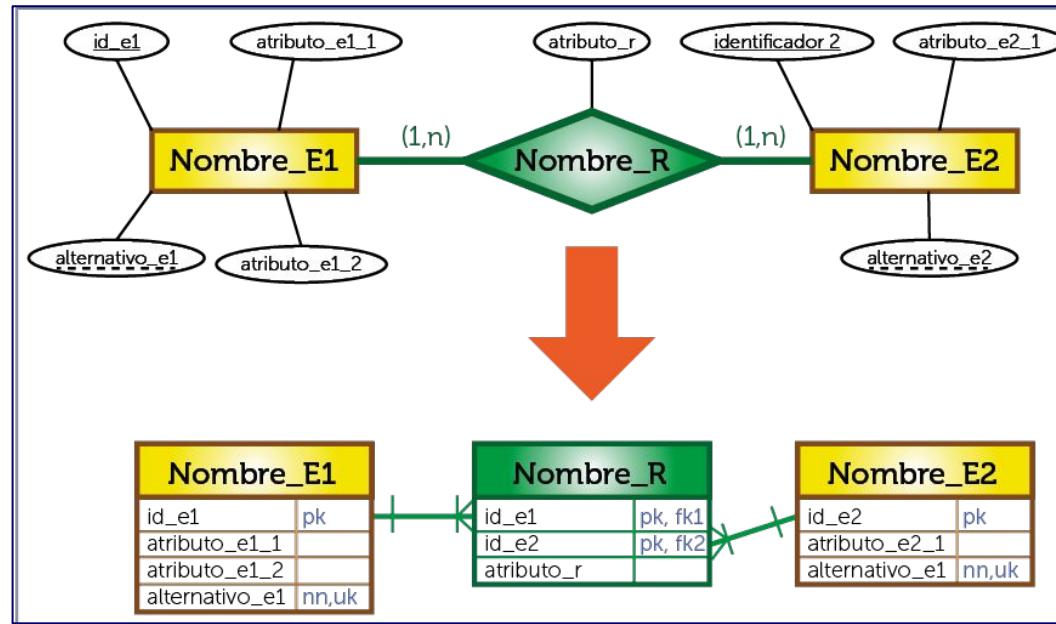
- ▶ Una empresa está dividida en departamentos. Cada departamento tiene un nombre, un número y un gerente.
- ▶ Los empleados participan en proyectos. Cada proyecto tiene un nombre, un monto asignado y un número único.
- ▶ Para cada empleado, se necesita guardar el nombre, la cédula y sexo.
- ▶ Un empleado pertenece a un departamento y puede trabajar en varios proyectos.
- ▶ Tomar nota del número de horas por semana que un empleado trabaja en un proyecto dado.
- ▶ En el transcurso del desarrollo de los proyectos se hacen informes parciales de los mismos, en fechas determinadas. De cada informe se necesita tener la fecha en que se realizó, el título y el resumen.



UNIVERSIDAD CES
Un compromiso con la excelencia

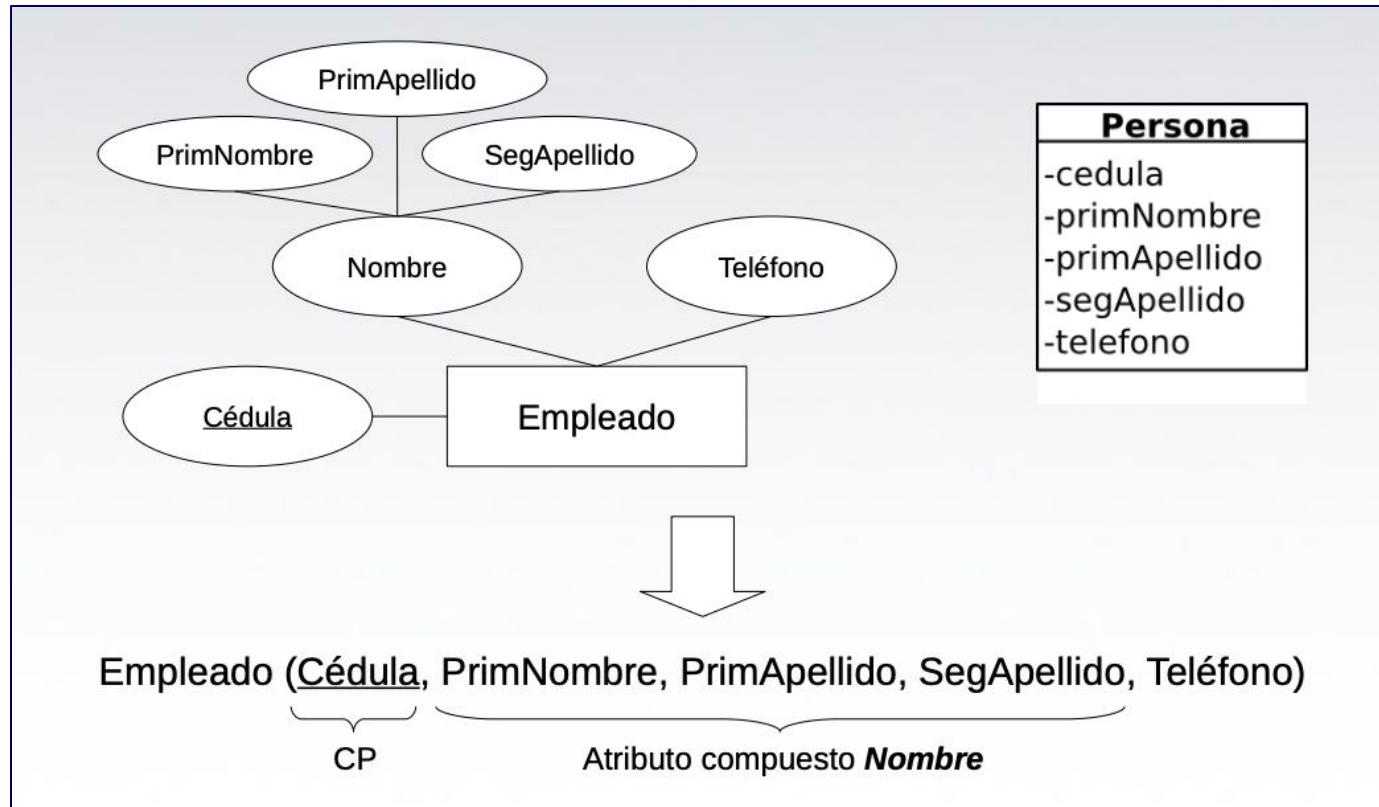
EJERCICIO



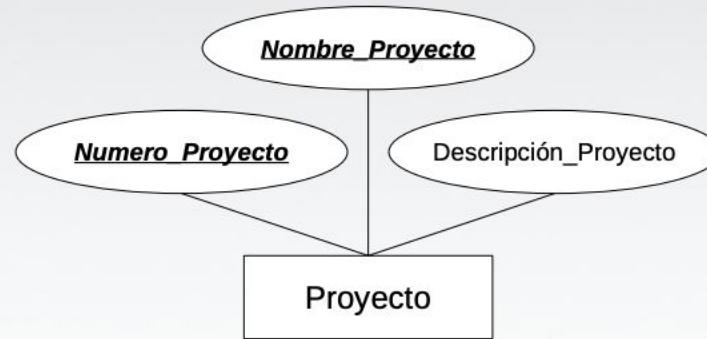


Transformación del Diagrama ER al Modelo Relacional

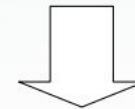




En caso de que más de un atributo sea parte de la clave primaria:



Proyecto
-numero
-nombre
-descripción

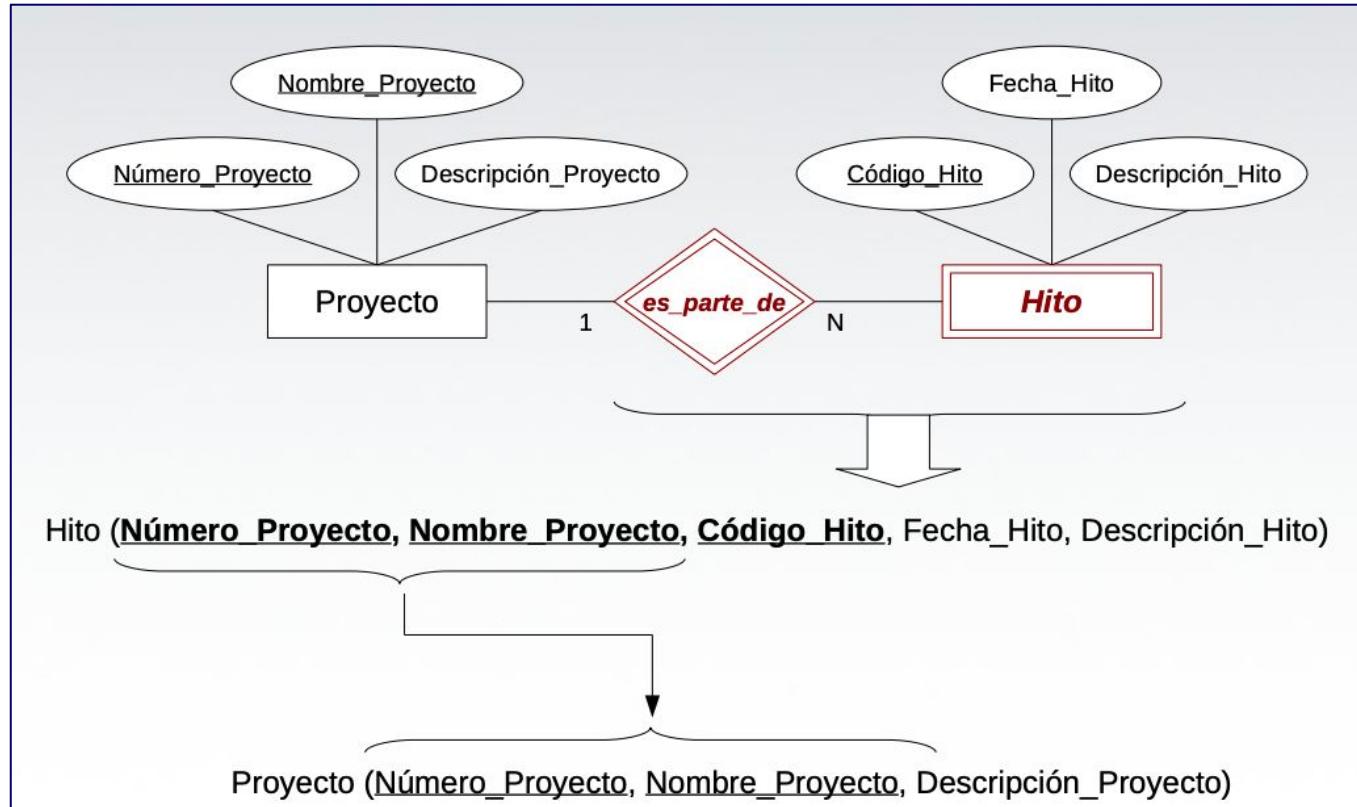


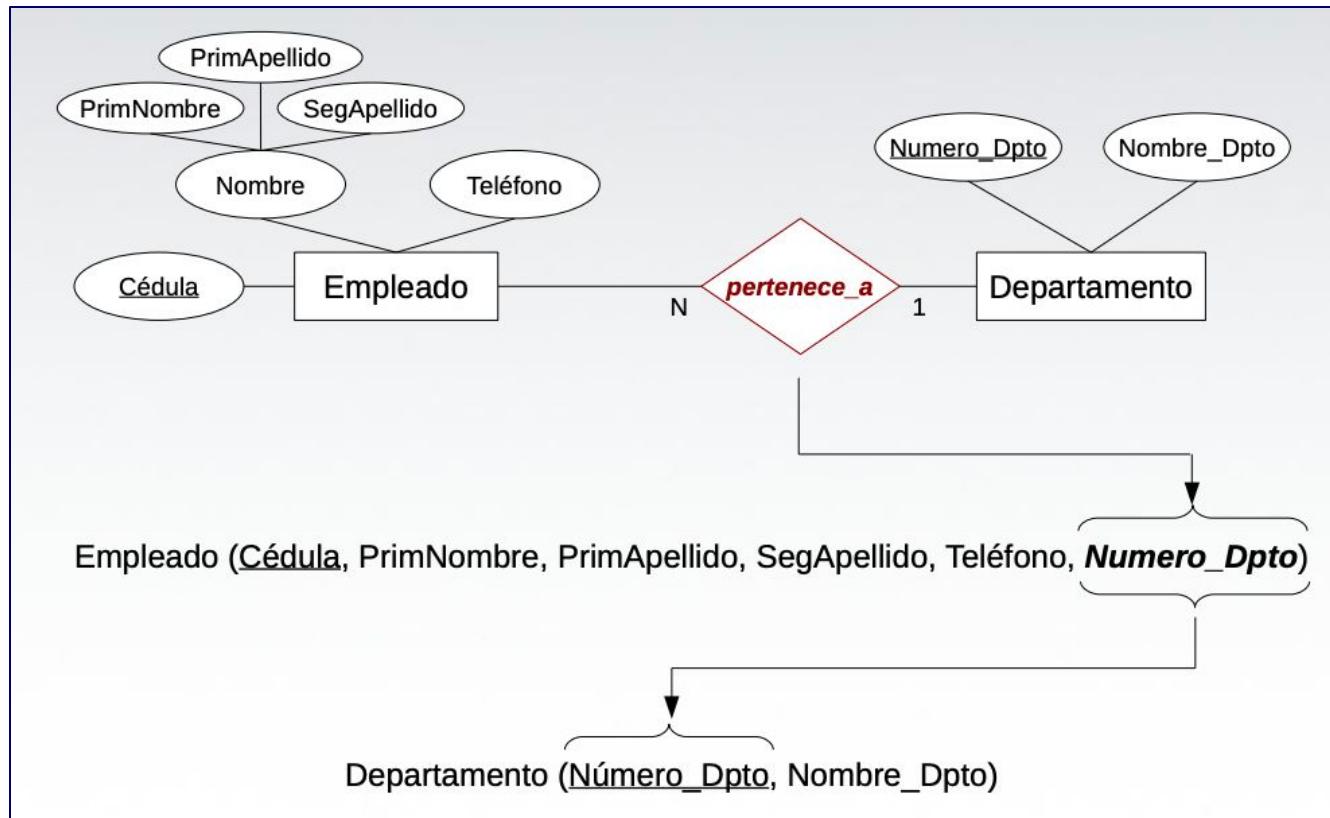
Proyecto (Número_Proyecto, Nombre_Proyecto, Descripción_Proyecto)

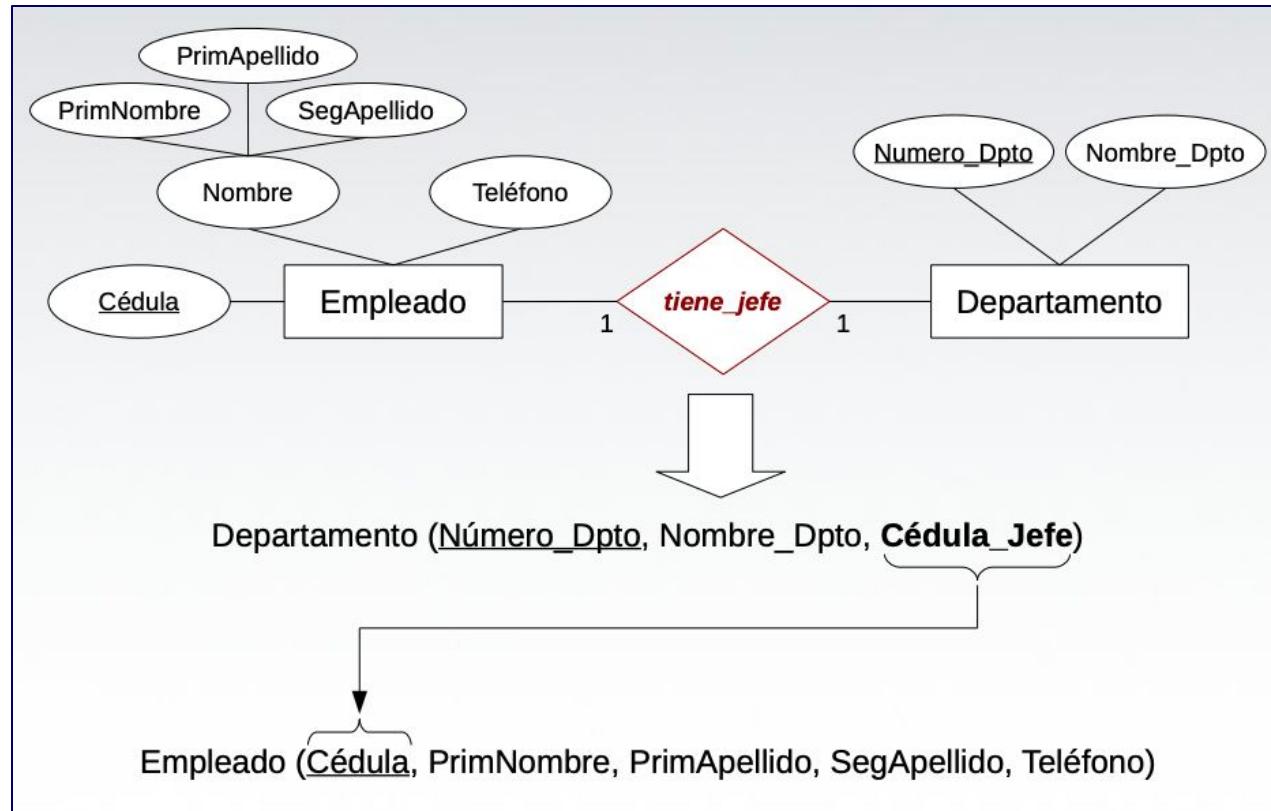
CP Compuesta

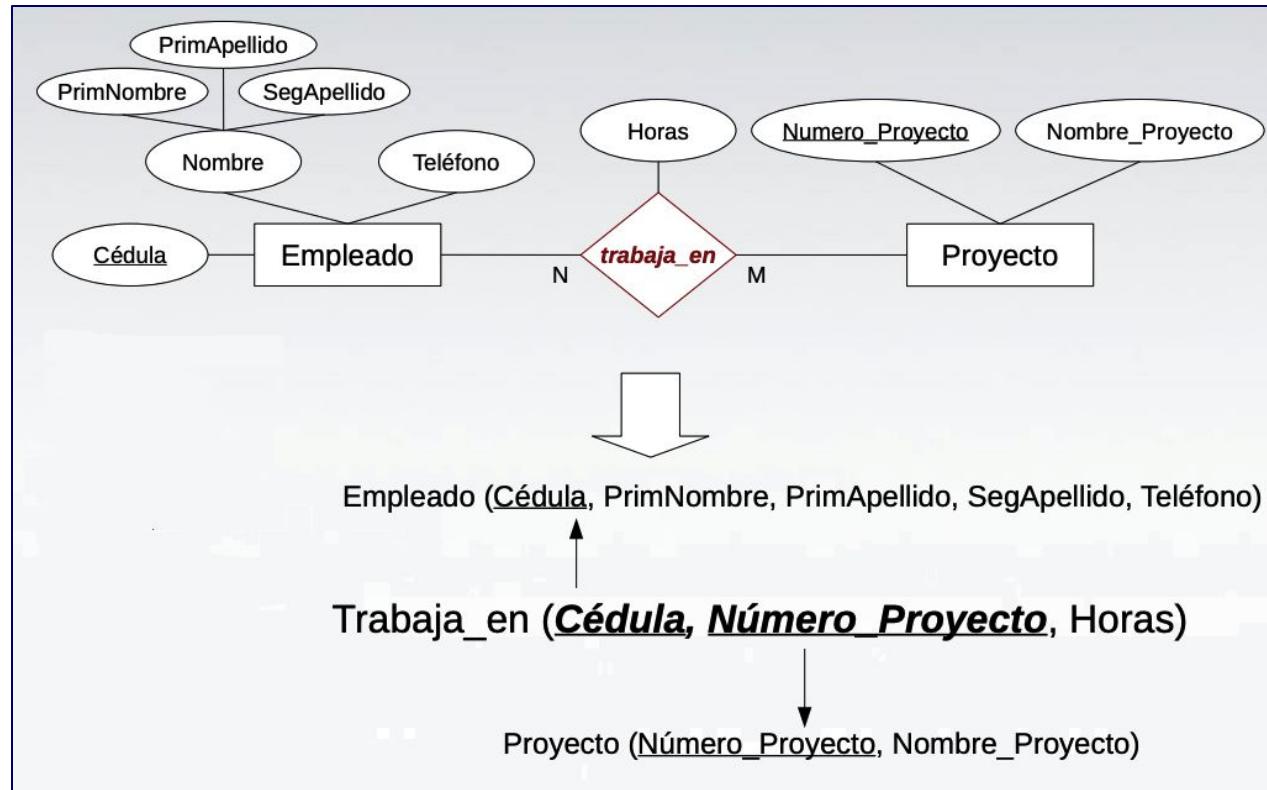


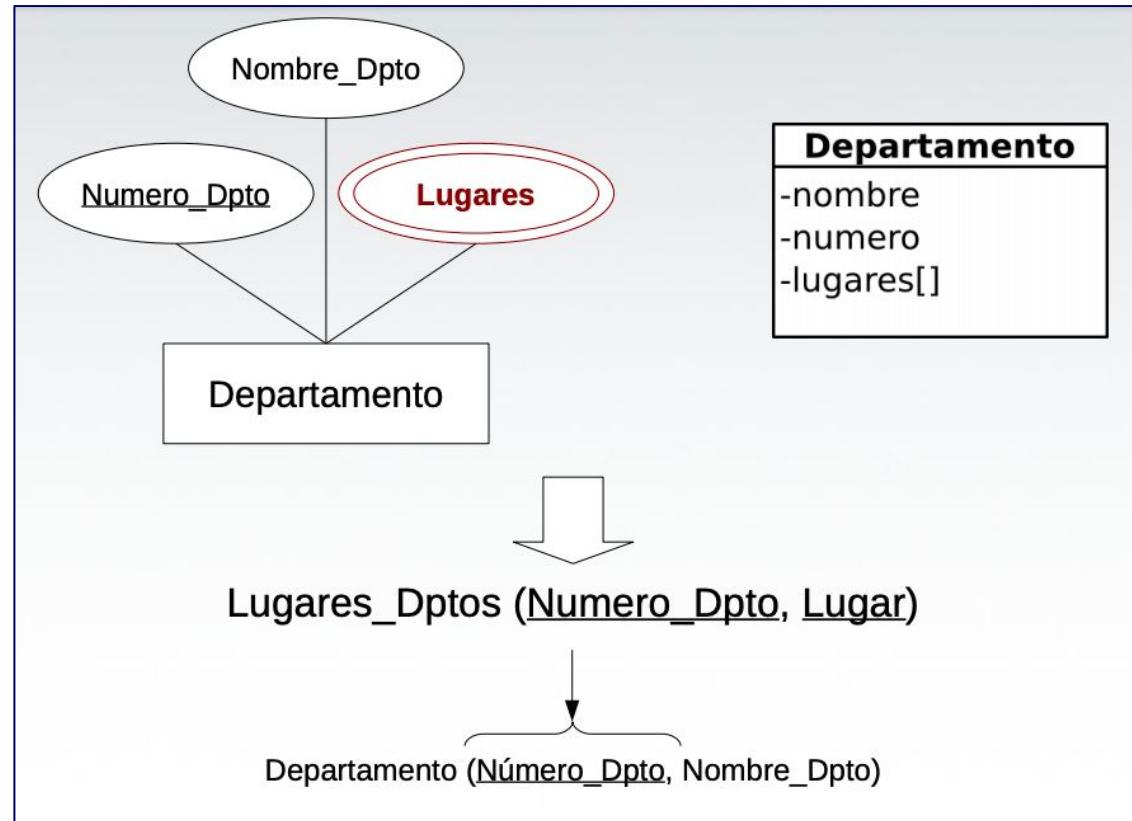
UNIVERSIDAD CES
Un compromiso con la excelencia













Empleado (Cédula, Nombre, Apellido, Dirección, Salario)

Profesor (Cédula, Nombre, Apellido, Dirección, Costo_Hora)

Estudiante (Cédula, Nombre, Apellido, Dirección, Carrera)

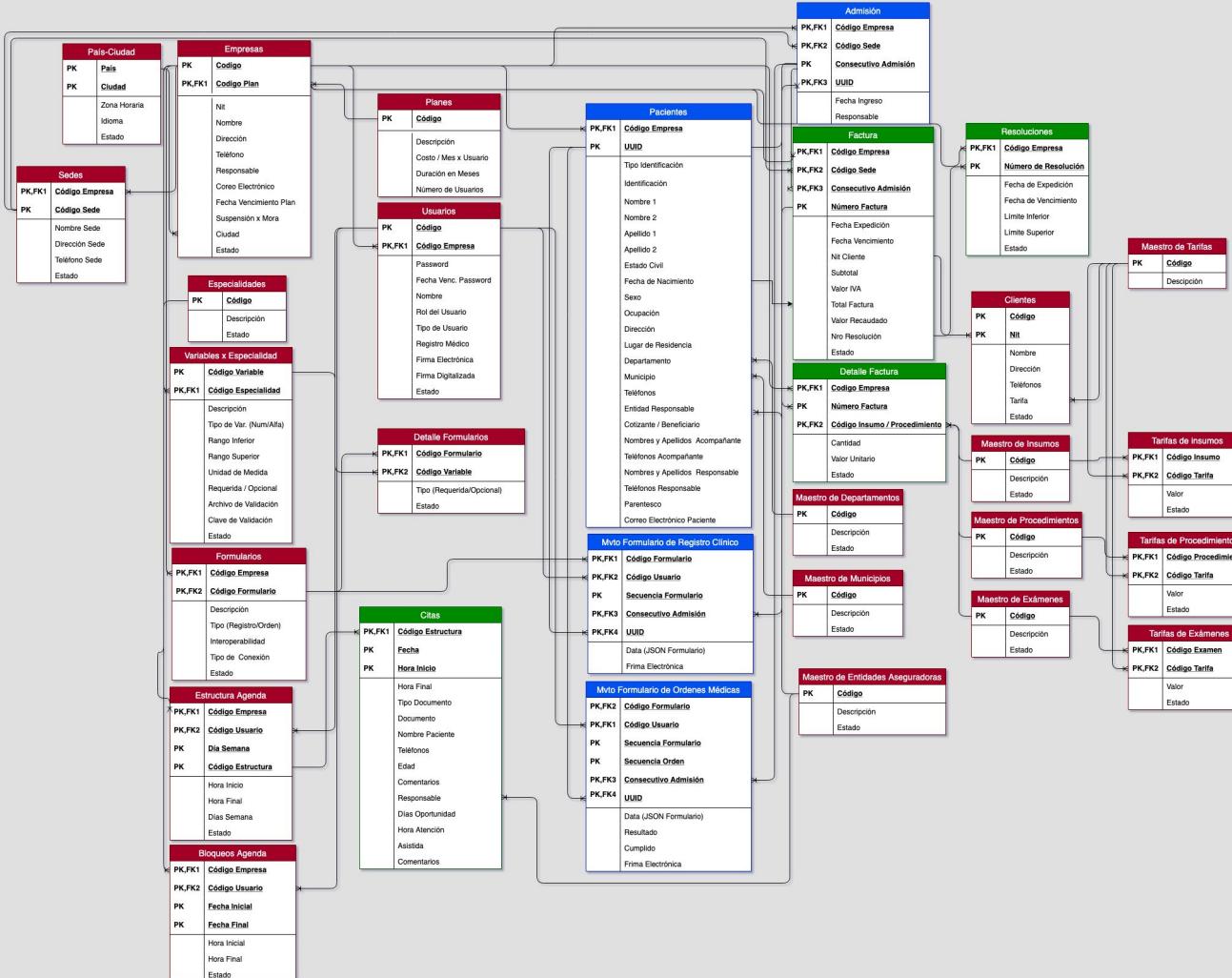




Persona (Cédula, Nombre, Apellido, Dirección, **Tipo**, Salario, Costo_Hora, Carrera)

Donde **Tipo** puede ser 0 para la subclase Empleado, 1 para la subclase Profesor o 2 para la subclase Estudiante





EJEMPLO DE MODELO RELACIONAL

- Una empresa vende productos a varios clientes. Se necesita conocer los datos personales de los clientes (nombre, apellidos, dni, dirección y fecha de nacimiento).
- Cada producto tiene un nombre y un código, así como un precio unitario. Un cliente puede comprar varios productos a la empresa, y un mismo producto puede ser comprado por varios clientes.
- Los productos son suministrados por diferentes proveedores.
- Se debe tener en cuenta que un producto sólo puede ser suministrado por un proveedor, y que un proveedor puede suministrar diferentes productos.
- De cada proveedor se desea conocer el NIF, nombre y dirección.

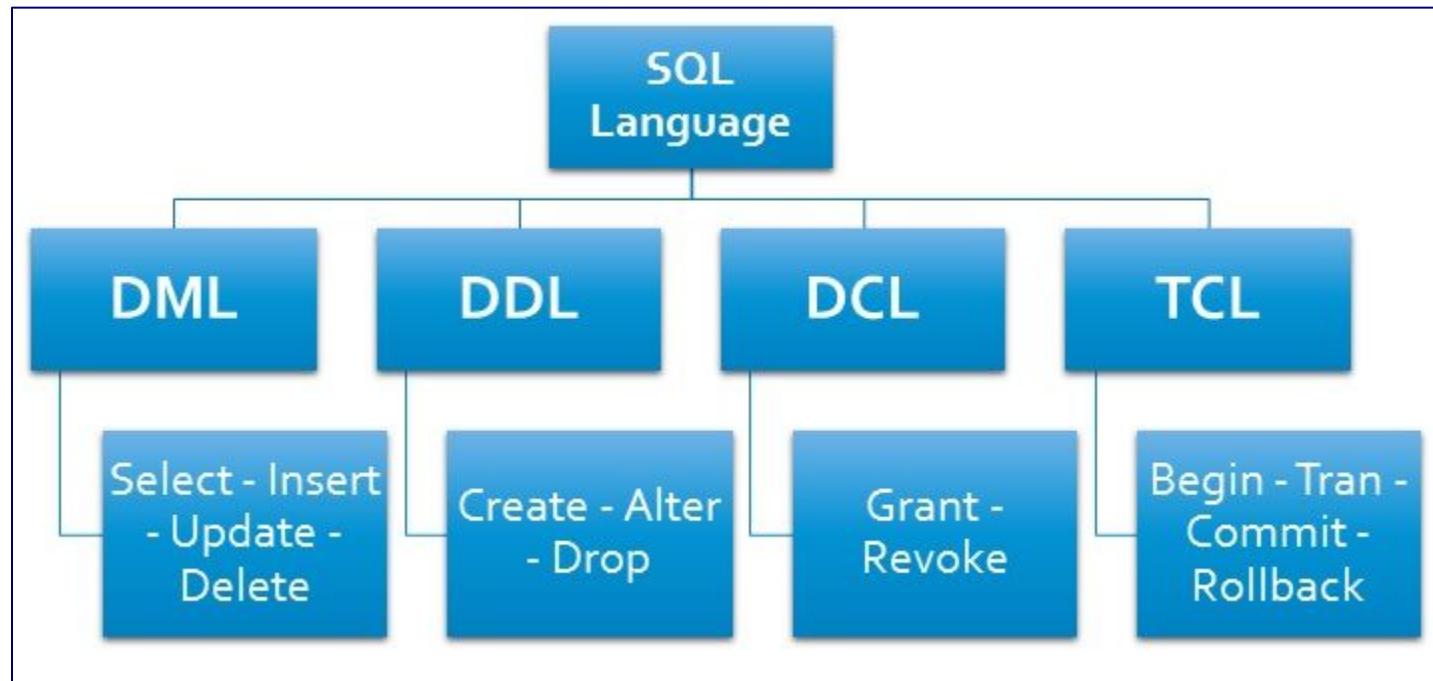


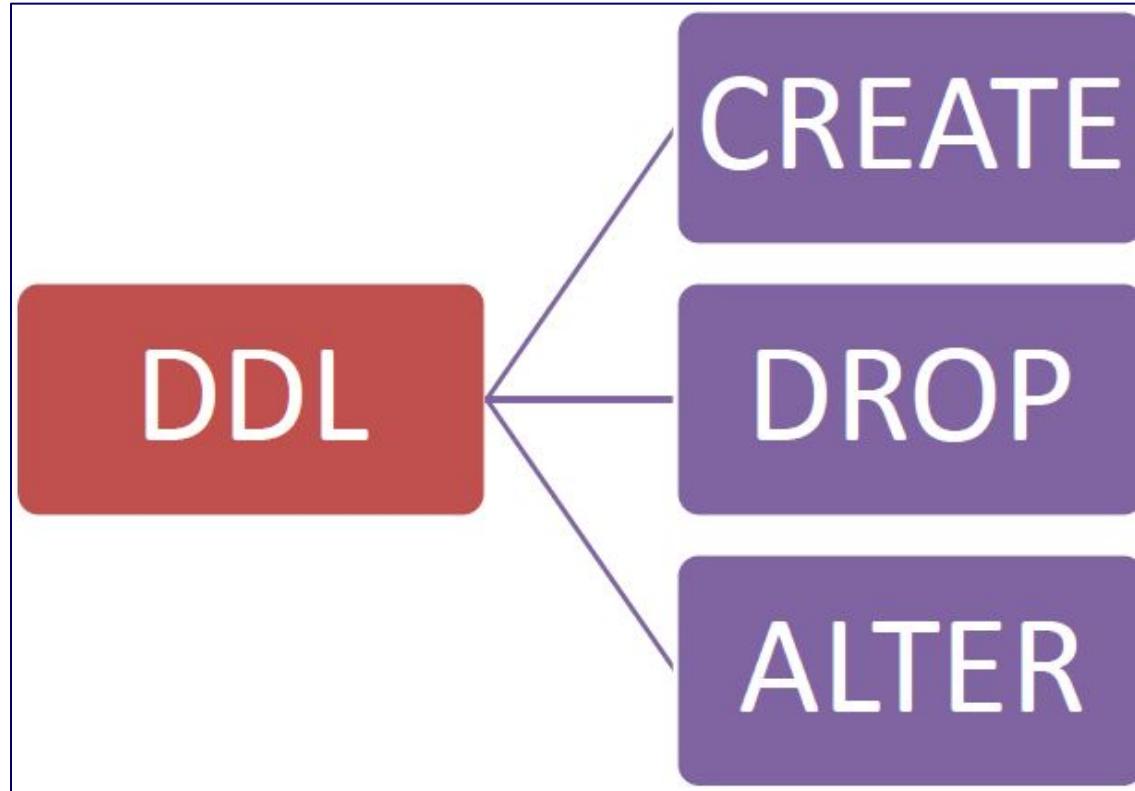
- La clínica “SAN PATRÁS” necesita llevar un control informatizado de su gestión de pacientes y médicos.
- De cada paciente se desea guardar el código, nombre, apellidos, dirección, población, provincia, código postal, teléfono y fecha de nacimiento.
- De cada médico se desea guardar el código, nombre, apellidos, teléfono y especialidad.
- Se desea llevar el control de cada uno de los ingresos que el paciente hace en el hospital.
- Cada ingreso que realiza el paciente queda registrado en la base de datos. De cada ingreso se guarda el código de ingreso (que se incrementará automáticamente cada vez que el paciente realice un ingreso), el número de habitación y cama en la que el paciente realiza el ingreso y la fecha de ingreso.
- Un médico puede atender varios ingresos, pero el ingreso de un paciente solo puede ser atendido por un único médico.
- Un paciente puede realizar varios ingresos en el hospital





UNIVERSIDAD CES
Un compromiso con la excelencia





La sentencia CREATE DATABASE se utiliza para crear bases de datos.

Sintaxis CREATE DATABASE:

`CREATE DATABASE nombreBaseDatos`

Ejemplo CREATE DATABASE

`CREATE DATABASE mibasededatos`



UNIVERSIDAD CES
Un compromiso con la excelencia

La sentencia **CREATE TABLE** se utiliza para crear una tabla en una base de datos existente.

Sintaxis CREATE TABLE

```
CREATE TABLE nombretabla
{
    nombrecolumna1 tipodato1,
    nombrecolumna2 tipodato2,
    nombrecolumna3 tipodato3,
    ..
}
```

Ejemplo CREATE TABLE

```
CREATE TABLE personas
{
    nombre varchar(255),
    apellido1 varchar(255),
    apellido2 varchar(255),
    dep int
}
```

Esta sentencia creará la base de datos 'personas' con 4 columnas.

Las columnas 'nombre', 'apellido1' y 'apellido2' son de tipo 'varchar', es decir, acepta valores alfanuméricos hasta una longitud máxima de 255 caracteres.

La columna 'dep' es de tipo 'int', es decir, acepta sólo números.

Existen diferentes tipos de datos, algunos son iguales en todas las bases de datos (MySQL, ORACLE, DB2, ..) y otros pueden ser particulares para ser usados únicamente en alguna de estas bases de datos.



CREATE INDEX se utiliza para crear índices en una tabla.

Un índice sirve para buscar datos rápidamente, y no tener que recorrer toda la tabla secuencialmente en busca alguna fila concreta.

Si una columna es índice de una tabla, al buscar por un valor de esa columna, iremos directamente a la fila correspondiente. La búsqueda así es mucho más óptima en recursos y más rápida en tiempo.

Si esa columna de búsqueda no fuese índice, entonces tendríamos que recorrer de forma secuencial la tabla en busca de algún dato. Por eso, es importante crear un índice por cada tipo de búsqueda que queramos hacer en la tabla.

Actualizar una tabla con índices tarda más tiempo porque también hay que actualizar los índices, así que solo se deben poner índices en las columnas por las que buscamos frecuentemente.

Se pueden crear índices ÚNICOS, es decir, índices que no admiten valores duplicados.

Sintaxis para SQL **CREATE INDEX**

```
CREATE INDEX nombreindice  
ON nombretabla (nombrecolumna)
```

La columna que forma parte de este índice admite valores duplicados en su columna.

Sintaxis para SQL **CREATE UNIQUE INDEX**

```
CREATE UNIQUE INDEX nombreindice  
ON nombretabla (nombrecolumna)
```

La columna que forma parte de este índice NO admite valores duplicados en su columna, porque es una clave única.



La sentencia **DROP** se utiliza para borrar definitivamente un índice, tabla o base de datos.

DROP INDEX

Sintaxis DROP INDEX para MySQL

```
ALTER TABLE nombretabla  
DROP INDEX nombreindice
```

Sintaxis DROP INDEX para DB2 y ORACLE

```
DROP INDEX nombreindice
```

Sintaxis DROP INDEX para ACCESS

```
DROP INDEX nombreindice  
ON nombretabla
```

Sintaxis DROP INDEX para SQLSERVER

```
DROP INDEX nombretabla.nombreindice
```

DROP TABLE

Se utiliza **DROP TABLE** para borrar definitivamente una tabla

```
DROP TABLE nombretabla
```

DROP DATABASE

Se utiliza para borrar una base de datos definitivamente.

```
DROP DATABASE nombrebasededatos
```



Este comando **SQL TRUNCATE** se utiliza para eliminar o borrar los datos que contiene una tabla.

Es útil cuando sólo se quiere borrar los datos, pero no se quiere borrar la tabla.

Este comando deja vacía una tabla, es decir, sin datos.

```
TRUNCATE TABLE nombretabla
```



La sentencia **SQL ALTER** se utiliza para añadir, eliminar o modificar columnas de una tabla.

Sintaxis SQL ALTER

Para añadir una nueva columna a una tabla

```
ALTER TABLE nombretabla  
ADD nombrecolumna tipodatocolumna
```

Para borrar una columna de una tabla

```
ALTER TABLE nombretabla  
DROP COLUMN nombrecolumna
```

Para modificar el tipo de dato de una columna de una tabla

```
ALTER TABLE nombretabla  
ALTER COLUMN nombrecolumna tipodatocolumna
```

Ejemplos de SQL ALTER

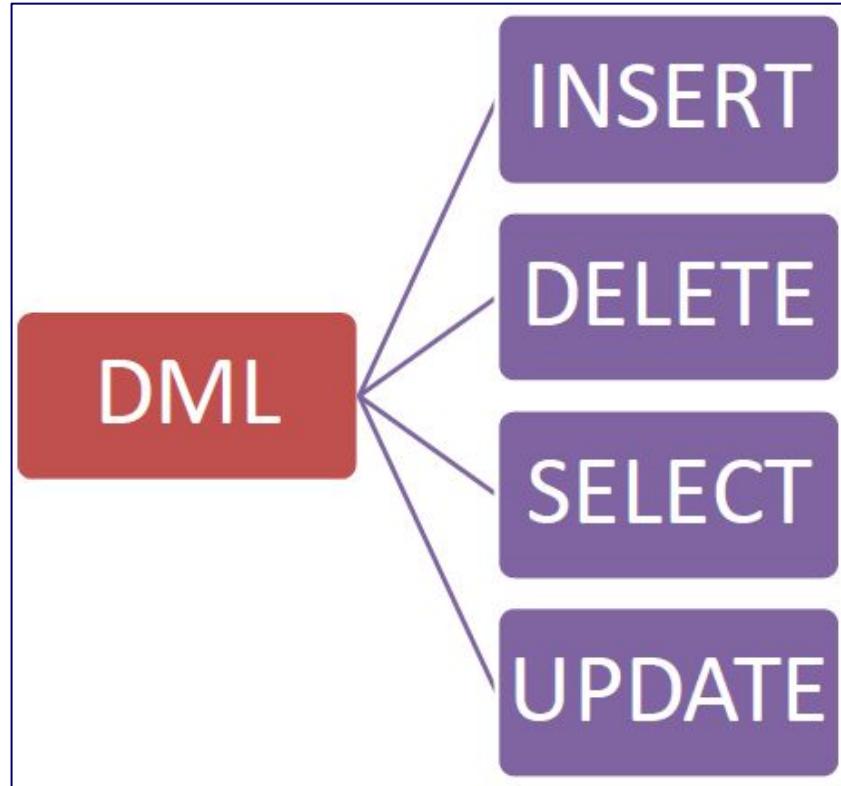
per	nombre	apellido1	apellido2
1	ANTONIO	PEREZ	GOMEZ
2	ANTONIO	GARCIA	RODRIGUEZ
3	PEDRO	RUIZ	GONZALEZ

Dada la siguiente tabla de 'personas', queremos añadir una nueva columna, denominada 'fechadenacimiento'

```
ALTER TABLE personas  
ADD fechadenacimiento date
```

per	nombre	apellido1	apellido2	fechadenacimiento
1	ANTONIO	PEREZ	GOMEZ	
2	ANTONIO	GARCIA	RODRIGUEZ	
3	PEDRO	RUIZ	GONZALEZ	





La sentencia INSERT INTO se utiliza para insertar nuevas filas en una tabla.

Es posible insertar una nueva fila en una tabla de dos formas distintas:

```
INSERT INTO nombre_tabla  
VALUES (valor1, valor2, valor3, .)
```

```
INSERT INTO nombre_tabla (columna1, columna2,  
columna3,.)  
VALUES (valor1, valor2, valor3, .)
```

Ejemplo:

Dada la siguiente tabla personas:

nombre	apellido1	apellido2
ANTONIO	PEREZ	GOMEZ
LUIS	LOPEZ	PEREZ
ANTONIO	GARCIA	BENITO

Si queremos insertar una nueva fila en la tabla personas, lo podemos hacer con cualquiera de las dos sentencias siguientes:

```
INSERT INTO personas  
VALUES ('PEDRO', 'RUIZ', 'GONZALEZ')  
INSERT INTO personas (nombre, apellido1,  
apellido2)  
VALUES ('PEDRO', 'RUIZ', 'GONZALEZ')
```



La sentencia **DELETE** sirve para borrar filas de una tabla.

La sintaxis de SQL DELETE es:

```
DELETE FROM nombre_tabla  
WHERE nombre_columna = valor
```

Si queremos borrar todos los registros o filas de una tabla, se utiliza la sentencia:

```
DELETE * FROM nombre_tabla;
```

Ejemplo de SQL DELETE para borrar una fila de la tabla personas

nombre	apellido1	apellido2
ANTONIO	PEREZ	GOMEZ
LUIS	LOPEZ	PEREZ
ANTONIO	GARCIA	RODRIGUEZ
PEDRO	RUIZ	GONZALEZ

Si queremos borrar a la persona LUIS LOPEZ PEREZ, podemos ejecutar el comando:

```
DELETE FROM personas  
WHERE nombre = 'LUIS'  
AND apellido1 = 'LOPEZ'  
AND apellido2 = 'PEREZ'
```

La tabla 'personas' resultante será:

nombre	apellido1	apellido2
ANTONIO	PEREZ	GOMEZ
ANTONIO	GARCIA	RODRIGUEZ
PEDRO	RUIZ	GONZALEZ



La sintaxis de SQL UPDATE es:

```
UPDATE nombre_tabla  
SET columna1 = valor1, columna2 = valor2  
WHERE columna3 = valor3
```

La cláusula SET establece los nuevos valores para las columnas indicadas.

La cláusula WHERE sirve para seleccionar las filas que queremos modificar.

Ojo: Si omitimos la cláusula WHERE, por defecto, modificará los valores en todas las filas de la tabla.

Ejemplo del uso de SQL UPDATE

nombre	apellido1	apellido2
ANTONIO	PEREZ	GOMEZ
LUIS	LOPEZ	PEREZ
ANTONIO	GARCIA	BENITO
PEDRO	RUIZ	GONZALEZ

Si queremos cambiar el apellido2 'BENITO' por 'RODRIGUEZ' ejecutaremos:

```
UPDATE personas  
SET apellido2 = 'RODRIGUEZ'  
WHERE nombre = 'ANTONIO'  
AND apellido1 = 'GARCIA'  
AND apellido2 = 'BENITO'
```

Ahora la tabla 'personas' quedará así:

nombre	apellido1	apellido2
ANTONIO	PEREZ	GOMEZ
LUIS	LOPEZ	PEREZ
ANTONIO	GARCIA	RODRIGUEZ
PEDRO	RUIZ	GONZALEZ



Una de las sentencias SQL más importantes es SELECT, ya que permite realizar consultas sobre los datos almacenados en la base de datos.

Sintaxis SQL SELECT

```
SELECT * FROM nombretabla
```

```
SELECT columna1, columna2 FROM nombretabla
```

Para los ejemplos, tendremos la siguiente tabla de personas denominada "personas"

Estos son los datos almacenados en la tabla "personas"

nombre	apellido1	apellido2
ANTONIO	PEREZ	GOMEZ
ANTONIO	GARCIA	BENITO
LUIS	LOPEZ	PEREZ

Si queremos consultar todos los datos de la tabla "personas"

```
SELECT * FROM personas
```

Este será el resultado:

nombre	apellido1	apellido2
ANTONIO	PEREZ	GOMEZ
ANTONIO	GARCIA	BENITO
LUIS	LOPEZ	PEREZ



La cláusula WHERE se utiliza para hacer filtros en las consultas, es decir, seleccionar solamente algunas filas de la tabla que cumplan una determinada condición.

El valor de la condición debe ir entre comillas simples ".

Por ejemplo:

Seleccionar las personas cuyo nombre sea ANTONIO

```
SELECT * FROM personas  
WHERE nombre = 'ANTONIO'
```

nombre	apellido1	apellido2
ANTONIO	PEREZ	GOMEZ
ANTONIO	GARCIA	BENITO



Conjunción Disyunción

p	q	$p \wedge q$
v	v	v
v	f	f
f	v	f
f	f	f

p	q	$p \vee q$
v	v	v
v	f	v
f	v	v
f	f	f



UNIVERSIDAD CES
Un compromiso con la excelencia

Los operadores **AND** y **OR** se utilizan para filtrar resultados con 2 condiciones.

El operador **AND** mostrará los resultados cuando se cumplan las 2 condiciones.

Condición1 AND condición2

El operador **OR** mostrará los resultados cuando se cumpla alguna de las 2 condiciones.

Condición1 OR condición2

En la tabla personas

nombre	apellido1	apellido2
ANTONIO	PEREZ	GOMEZ
ANTONIO	GARCIA	BENITO
LUIS	LOPEZ	PEREZ

La siguiente sentencia (ejemplo AND) dará el siguiente resultado:

```
SELECT * FROM personas  
WHERE nombre = 'ANTONIO'  
AND apellido1 = 'GARCIA'
```

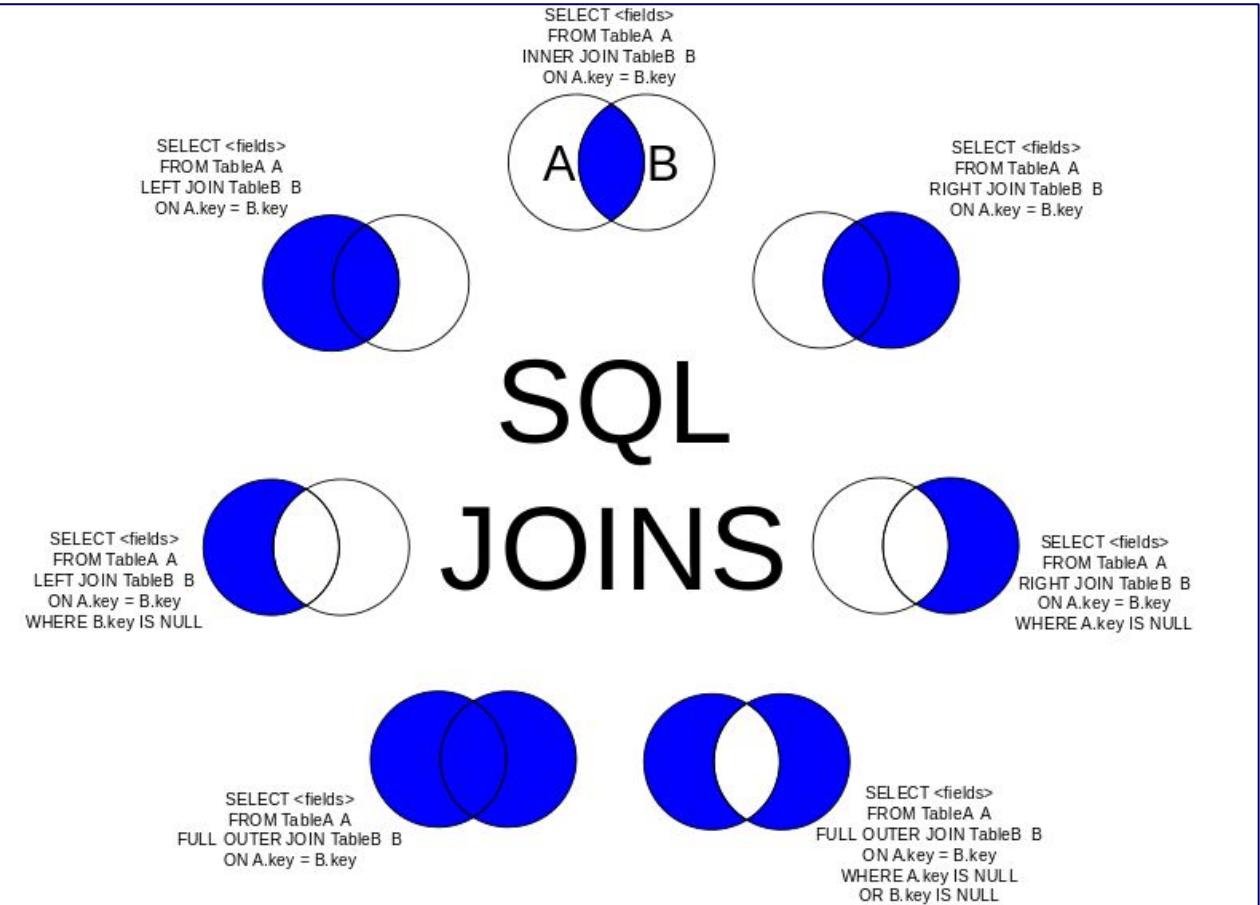
nombre	apellido1	apellido2
ANTONIO	GARCIA	BENITO

La siguiente sentencia (ejemplo OR) dará el siguiente resultado:

```
SELECT * FROM personas  
WHERE nombre = 'ANTONIO'  
OR apellido1 = 'GARCIA'
```

nombre	apellido1	apellido2
ANTONIO	PEREZ	GOMEZ
ANTONIO	GARCIA	BENITO





Devolver todos los campos de una tabla (**SELECT ***)

```
SELECT *  
FROM CLIENTES
```

Con el * indicamos que queremos devolver todos los campos. Si CLIENTES dispone de los campos *idCliente*, *nombre* y *descripcion*, lo anterior sería equivalente a:

```
SELECT idCliente, nombre, descripcion  
FROM CLIENTES
```

Obviamente, al querer todos los campos, esto es innecesario y es por tanto más conveniente emplear el asterisco (*). También sería equivalente emplear la notación completa:

```
SELECT CLIENTES.idCliente, CLIENTES.nombre, CLIENTES.descripcion  
FROM CLIENTES
```

Al tener únicamente una tabla involucrada, podemos referirnos a los campos sin calificar, dado que no hay duda de a qué tabla se refiere. Cuando veamos consultas sobre varias tablas comprenderemos la necesidad de incluir esta notación calificada (TABLA.campo).



Devolver un subconjunto de los campos de una tabla (**SELECT DISTINCT**)

```
SELECT cp, ciudad  
FROM DIRECCION
```

Esta consulta devolverá únicamente los campos *cp* (código postal) y *ciudad* de la tabla DIRECCION. Al tener un subconjunto de los campos, éstos no tienen por qué incluir a la clave de la tabla, por lo que no tienen por qué ser únicos. Así, si tenemos muchos registros referidos a distintas calles y números de ese mismo código postal y ciudad, nos encontraremos muchos registros repetidos. Esto puede evitarse haciendo:

```
SELECT DISTINCT cp, ciudad  
FROM CLIENTES
```

Así se eliminan los registros repetidos, devolviendo únicamente una vez cada par *cp, ciudad*. Esta selección de un subconjunto de los datos de la tabla y excluyendo repetidos se denomina en álgebra relacional *proyección*.



Devolver un subconjunto de los registros de una tabla (WHERE)

```
SELECT numero, calle  
FROM DIRECCION  
WHERE ciudad = 'Sevilla'
```

Esta consulta devolvería el número y la dirección de todas las direcciones pertenecientes a la ciudad de Sevilla. Como vemos, con WHERE indicamos la condición que deben cumplir los registros de la tabla para ser devueltos en la consulta. En este caso tenemos una condición simple dada por la comparación de igualdad (`=`) entre al campo (`ciudad`) y un literal de tipo cadena, entre comillas simples ('Sevilla').



SQL - SENTENCIAS DML

Operadores relacionales

Al margen del signo de igualdad empleado anteriormente, se pueden usar n las condiciones simples de las consultas los operadores relacionales habituales, devolviendo siempre un valor booleano (lógico):

Operador	Significado
<	Menor que
>	Mayor que
<>	Distinto de
<=	Menor ó Igual que
>=	Mayor ó Igual que
=	Igual que

```
SELECT nombre
FROM CLIENTES
WHERE edad <= 32
```

Adicionalmente, disponemos de operadores de comparación adicionales, también devolviendo valores booleanos (lógicos) True o False según si se cumplen o no las condiciones:

BETWEEN: para indicar un intervalo de valores.

```
SELECT nombre
FROM CLIENTES
WHERE edad BETWEEN 20 AND 35
```



LIKE: empleado para comparar [patrones de texto](#) pudiendo incluir comodines como los siguientes:

Comodín	Descripción
%	Sustituto para cero o más caracteres.
_	Sustituto para exactamente un carácter
[lista caracteres]	Cualquier carácter de la lista
[^lista caracteres]	Cualquier carácter que no esté en la lista
[!lista caracteres]	Cualquier carácter que no esté en la lista

```
SELECT num, calle, cp  
FROM DIRECCION  
WHERE ciudad LIKE 'Val%'
```

Esta consulta devolvería los datos de las direcciones de toda ciudad que comience por *Val* y siga por cualquier número de caracteres, incluyendo cero caracteres. Por ejemplo, *Valladolid* o *Valencia*. En el enlace anterior sobre patrones de texto podemos practicar directamente desde la web.



IN: empleado para comparar con una lista de valores fijados de modo que devuelva True si el campo indicado pertenece a la lista.

```
SELECT num, calle, direccion  
FROM DIRECCION  
WHERE ciudad IN ('Sevilla', 'Córdoba', 'Huelva', 'Cádiz')
```

Operadores lógicos (*AND, OR, NOT*)

Los [operadores lógicos](#) nos sirven para componer expresiones de filtrado a partir de las anteriores:

Operador	Significado
AND	Y lógico
OR	O lógico
NOT	Negación lógica

La precedencia y asociatividad es la habitual definida en la Lógica. En cualquier caso, cuando incluya expresiones que empleen varios de estos operadores es recomendable usar paréntesis para evitar errores. Por ejemplo:

```
SELECT *  
FROM DIRECCION  
WHERE ciudad = 'Sevilla' AND cp = 41009 OR ciudad = 'Córdoba' AND NOT cp = 14010
```

Devuelve los registros pertenecientes a direcciones que tengan el código postal 41009 de Sevilla o bien que no tengan el 14010 de Córdoba. La mayor precedencia la adopta el operador *NOT* sobre la condición *cp = 14010*; a continuación los *AND* se aplican sobre *ciudad = 'Sevilla'* *AND cp = 41009* y *ciudad = 'Córdoba'* *AND NOT cp = 14010*; por último se aplica el *OR* sobre la fórmula completa. La misma consulta se puede expresar de forma más clara con paréntesis:

```
SELECT *  
FROM DIRECCION  
WHERE (ciudad = 'Sevilla' AND cp = 41009) OR  
(ciudad = 'Córdoba' AND (NOT cp = 14010))
```

O bien si el *NOT* nos parece más evidente, podemos excluir el paréntesis interior, a nuestra gusto siempre conservando el significado que queríamos dar la operación.



Ordenación

Ordenar según criterios (ORDER BY)

Podemos ordenar los registros devueltos por una consulta por el campo o campos que estimemos oportunos:

```
SELECT *
  FROM CIUDAD
 ORDER BY provincia ASC, numhabitantes DESC
```

Esta consulta devolvería todas las ciudades ordenadas por provincia en orden ascendente, y dentro de los de la misma provincia ordenaría las ciudades por orden descendente del número de habitantes. Si no indicamos ASC ni DESC, el comportamiento por defecto será el orden ascendente (ASC).



Consultas agrupadas (GROUP BY)

Las consultas anteriores recuperaban, trabajaban con, y mostraban información a nivel de cada registro individual de la base de datos. Así, si tenemos un producto con un determinado precio, podemos devolver el precio mediante `SELECT precioLinea` o bien operar sobre él como en `SELECT precioLinea * 0.85`.

Ahora bien, podemos querer obtener información que no proviene de un registro individual sino de la agrupación de información, como es el caso de contar el número de líneas de pedido, sumar el precio de todas las líneas por cada pedido, etc. Para ello, debemos emplear funciones agregadas y en la mayoría de los casos agrupar por algún campo.

Así, para ver el número total de registros podemos hacer:

```
SELECT COUNT(*)  
FROM LINEAPEDIDO
```

Si por el contrario deseamos obtener el total de líneas por pedido, debemos indicar que agrupe por `idPedido`, lo que contará todos los registros con el mismo `idPedido` y calculará su cuenta:

```
SELECT idPedido, COUNT(*)  
FROM LINEAPEDIDO  
GROUP BY idPedido
```

Lo mismo se puede aplicar a otras funciones como la suma, indicando en ese caso aparte de la agrupación el campo que queremos sumar:

```
SELECT idPedido, SUM(precioLinea)  
FROM LINEAPEDIDO  
GROUP BY idPedido
```



Consultas por intersecciones entre tablas (JOIN)

Podemos generar una consulta que obtenga datos de varias tablas, pudiendo establecer a su vez criterios sobre otras. Veamos los tipos fundamentales (una buena explicación se puede encontrar [aqui](#)). Por ejemplo, dados los clientes de nuestro concesionario y nuestras ventas de coches a clientes en los diversos concesionarios,

VENTA			
cifc	cifcl	codcoche	color
1	1	1	blanco
1	2	5	rojo
2	3	8	blanco
2	1	6	rojo
3	4	11	rojo
4	5	14	verde

CLIENTE			
cifcl	nombre	apellidos	ciudad
1	Luis	García	Madrid
2	Antonio	López	Valencia
3	Juan	Martín	Madrid
4	Maria	García	Madrid
5	Javier	González	Barcelona
6	Ana	López	Barcelona

podemos obtener una consulta que obtenga datos del cliente y de la venta. Por ejemplo, nombre, apellidos, codcoche y color, mediante una consulta del tipo:

```
SELECT nombre, apellidos, codcoche, color  
FROM CLIENTE, VENTA  
WHERE CLIENTE.cifcl = VENTA.idCliente
```



INNER JOIN

INNER JOIN implícito

Por ejemplo, para obtener los datos del cliente y el pedido en la misma consulta:

```
SELECT nombreCliente, idPedido, fechaPedido  
FROM CLIENTE, PEDIDO  
WHERE CLIENTE.idCliente = PEDIDO.idCliente
```

Como vemos, podemos empezar escribiendo tal cuál qué datos nos piden (SELECT), de dónde podemos obtenerlos (FROM) y qué criterio (WHERE). Esta es la versión más antigua de SQL, aunque se sigue empleando, conociéndose como **JOIN implícito** ya que no se usa por ningún lado la palabra **JOIN**, pero se está haciendo la intersección por la foreign key, en este caso la columna *idCliente*.



INNER JOIN explícito

La forma más habitual de INNER JOIN es la que intersecta las tablas indicadas en con INNER JOIN por el campo indicado por ON.

```
SELECT nombreCliente, idPedido, fechaPedido  
FROM CLIENTE INNER JOIN PEDIDO  
ON cliente.idCliente = pedido.idCliente
```

Este comportamiento se explica y puede probar [aquí](#), en este [otro enlace](#) o finalmente [aquí](#).

Como vemos, es equivalente al INNER JOIN implícito, pero en lugar de establecer una condición sobre los campos de las relaciones en el WHERE, establece la condición asociada a la relación entre ambas tablas a través de la FOREIGN KEY, en este caso desde el atributo (campo) idCliente en la relación (tabla) PEDIDO, FOREIGN KEY que hace referencia al atributo idCliente de la relación CLIENTE.



FIN SESIÓN 6

¡Gracias!



UNIVERSIDAD CES
Un compromiso con la excelencia