

C.F.G.S. DESARROLLO DE APLICACIONES WEB

**MÓDULO:
BASES DE DATOS**

**Unidad 2:
Interpretación de Diagramas Entidad/Relación.**

ÍNDICE DE CONTENIDOS

1.- Análisis y diseño de bases de datos.	4
2.- Que es el modelo Entidad-relación.	5
3.- Entidades.	6
3.1.- Tipos de Entidad.	7
4.- Atributos.	8
4.1.- Tipos de atributos.	10
4.2.- Claves.	10
4.3.- Atributos de una relación.	12
5.- Relaciones.	12
5.1.- Grado de una Relación.	13
5.2.- Cardinalidad de las relaciones.	15
5.3.- Cardinalidad de entidades.	17
6.- Simbología del modelo E/R.	18
7.- El modelo E/R extendido.	19
7.1.- Restricciones en las relaciones.	20
7.2.- Generalización y especificación.	22
7.3.- Agregación.	24
8.- Elaboración de modelo E/R.	26
8.1.- Identificación de entidades y relaciones.	26
8.2.- Identificación de atributos, claves, y jerarquías.	28
8.3.- Metodologías.	29
8.4.- Propiedades deseables de un diagrama E/R.	30
9.- Paso del diagrama E/R al modelo relacional.	31
10.- Normalización.	35
10.1.- Tipos de Dependencias.	36
10.2.- Formas Normales.	40
11.- Desnormalización.	45

OBJETIVOS

Esta unidad realiza un recorrido desde la fase de análisis hasta la fase de diseño de bases de datos.

Los objetivos principalmente serán:

- Interpretar el diseño lógico de bases de datos basado en el modelo relacional.
- Identificar la terminología propia del modelo relacional.
- Identificar la estructura de una base de datos relacional.
- Reconocer las restricciones del modelo relacional.

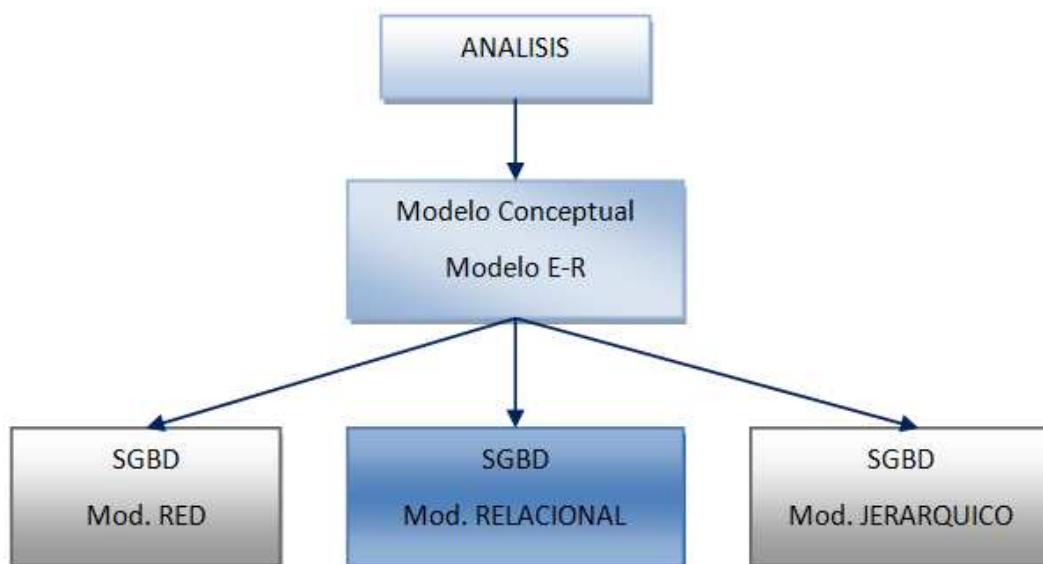
1.- Análisis y diseño de bases de datos.

Un modelo de datos es una colección de herramientas conceptuales que permiten llevar a cabo la descripción de los datos, sus relaciones, su semántica o significado y las restricciones que se les pueden aplicar. Sabemos que los SGBD cuentan con una arquitectura que simplifica, a los diferentes usuarios de la base de datos, su labor. El objetivo fundamental de esta arquitectura es separar los programas de aplicación de la base de datos física, proponiendo tres niveles de abstracción:

- Nivel **externo o visión del usuario**: visión total o parcial de un usuario o grupo de usuarios que utiliza la BD.
- Nivel **lógico o conceptual**: Representación de todos los datos definiendo una visión global de los mismos (datos, relaciones y restricciones).
- Nivel **interno o físico**: ficheros, organización, longitud, forma de acceso, tipo de registros, campos, índices...

El **Nivel lógico o conceptual** describe la estructura completa de la base de datos a través de lo que llamamos **Esquema Conceptual**, que se encarga de representar la información de una manera totalmente independiente del Sistema Gestor de Base de Datos.

El objetivo final es poder implementar el diseño en un modelo concreto de BD. (En nuestro caso pasaremos a un modelo relacional.)



Cuando hemos de desarrollar una base de datos se distinguen claramente dos fases de trabajo:

- **Análisis.**
- **Diseño.**

En la siguiente tabla se describen las etapas que forman parte de cada fase.

Fase de Análisis	Fase de Diseño
Análisis de entidades: Se trata de localizar y definir las entidades y sus atributos	Diseño de tablas
Análisis de relaciones: Definir las relaciones existentes entre entidades	Normalización
Obtención del Esquema Conceptual a través del modelo Entidad-Relación	Aplicación del retrodiseño si fuese necesario
Fusión de vistas: Se reúnen en un único esquema todos los esquemas existentes en función de las diferentes vistas de cada perfil de usuario.	Diseño de transacciones: Localización del conjunto de operaciones o transacciones que operan sobre el esquema conceptual.
Aplicación del enfoque de datos relacional	Diseño de las sendas de acceso: se formalizan los métodos de acceso dentro de la estructura de datos.

Llevando a cabo una correcta fase de análisis estaremos dando un paso determinante en el desarrollo de nuestras bases de datos. El hecho de saltarse el Esquema Conceptual conlleva un problema de pérdida de información respecto al problema real a solucionar. El esquema conceptual debe reflejar todos los aspectos relevantes del mundo real que se va a modelar.

Para la realización de esquemas que ofrezcan una visión global de los datos, Peter Chen en 1976 y 1977 presenta dos artículos en los que se describe el modelo Entidad/Relación (entity/relationship). Con el paso del tiempo, este modelo ha sufrido modificaciones y mejoras. Actualmente, el modelo entidad/relación extendido (ERE) es el más aceptado, aunque existen variaciones que hacen que este modelo no sea totalmente un estándar.

2.- Que es el modelo Entidad-relación.

Es una herramienta de referencia para la representación conceptual de problemas del mundo real. Su objetivo principal, es **facilitar el diseño de bases de datos permitiendo la especificación de un esquema que representa la estructura lógica completa de una base de datos**. Este esquema partirá de las descripciones textuales de la realidad, que establecen los requerimientos del sistema, buscando ser lo más fiel posible al comportamiento del mundo real para modelarlo.

El modelo de datos Entidad-Relación representa el significado de los datos, es un modelo semántico. De ahí que no esté orientado a ningún sistema físico concreto y tampoco tiene un ámbito informático puro de aplicación, ya que podría utilizarse para describir procesos de producción, estructuras de empresa, etc.

Además, las características actuales de este modelo favorecen la representación de cualquier tipo de sistema y a cualquier nivel de abstracción o refinamiento, lo cual da lugar a que se aplique tanto a la representación de problemas que vayan a ser tratados mediante un sistema informatizado, como manual.

Gracias al modelo Entidad-Relación, creado por **Peter Chen** en los años setenta, se puede representar el mundo real mediante una serie de símbolos y expresiones determinados. El modelo de datos Entidad/Relación (E/R ó E-R) está basado en una percepción consistente en objetos básicos llamados entidades y relaciones.

3.- Entidades.

Si utilizamos las bases de datos para guardar información sobre cosas que nos interesan o que interesan a una organización, hay que identificar esas cosas primero para poder guardar información sobre ellas. Para ello, vamos a describir un primer concepto, el de **Entidad**.

Una **entidad** puede ser un objeto físico, un concepto o cualquier elemento que queramos modelar, que tenga importancia para la organización y del que se desee guardar información. Cada entidad debe poseer alguna característica, o conjunto de ellas, que lo haga único frente al resto de objetos. Por ejemplo, podemos establecer una entidad llamada **ALUMNO** que tendrá una serie de características. El alumnado podría ser distinguido mediante su número de matrícula, o por su DNI, por ejemplo.

Entidad: objeto real o abstracto, con características diferenciadoras capaces de hacerse distinguir de otros objetos.

A la hora de identificar las entidades, hemos de pensar en nombres que tengan especial importancia dentro del lenguaje propio de la organización o sistema que vaya a utilizar dicha base de datos. Pero no siempre una entidad puede ser concreta, como un **cliente**, un **departamento**, un **empleado**... en ocasiones puede ser abstracta, como un **préstamo**, **una reserva en un hotel**...

Un **conjunto de entidades** serán un grupo de entidades que poseen las mismas características o propiedades. Por ejemplo, al conjunto de personas que realizan reservas para un hotel de montaña determinado, se les puede definir como el conjunto de entidades cliente. Por lo general, se suele utilizar el término **entidad** para identificar **conjuntos de entidades**.

Cada elemento del conjunto de entidades será una ocurrencia de entidad.

Si establecemos un símil con la Programación Orientada a Objetos, podemos decir que el concepto de **entidad** es análogo al de **instancia de objeto** y que el concepto de **conjunto de entidades** lo es al de **clase**.

En el modelo Entidad/Relación, la representación gráfica de las entidades se realiza mediante el nombre de la entidad encerrado dentro de un rectángulo. Se nombran con sustantivos en mayúscula y singular.

La representación de la entidad CLIENTE sería:



3.1.- Tipos de Entidad.

Las entidades pueden ser clasificadas en dos grupos:

■ **Entidades Fuertes o regulares:**

Son aquellas que tienen existencia por sí mismas, es decir, su existencia no depende de la existencia de otras entidades. Por ejemplo, en una base de datos hospitalaria, la existencia de instancias concretas de la entidad DOCTOR no depende de la existencia de instancias u objetos de la entidad PACIENTE.

En el modelo E/R las entidades fuertes se representan como hemos indicado anteriormente, con el nombre de la entidad encerrado dentro de un rectángulo.



■ **Entidades débiles:**

Son aquellas cuya existencia depende de la existencia de otras instancias de entidad.

Por ejemplo, consideremos las entidades EDIFICIO y AULA. Supongamos que puede haber aulas identificadas con la misma numeración pero en edificios diferentes. La numeración de cada aula no identificará completamente cada una de ellas. Para poder identificar completamente un aula es necesario saber también en qué edificio está localizada. Por tanto, la existencia de una instancia de una entidad débil depende de la existencia de una instancia de la entidad fuerte con la que se relaciona.

En el modelo E/R una entidad débil se representa con el nombre de la entidad encerrado en un rectángulo doble.

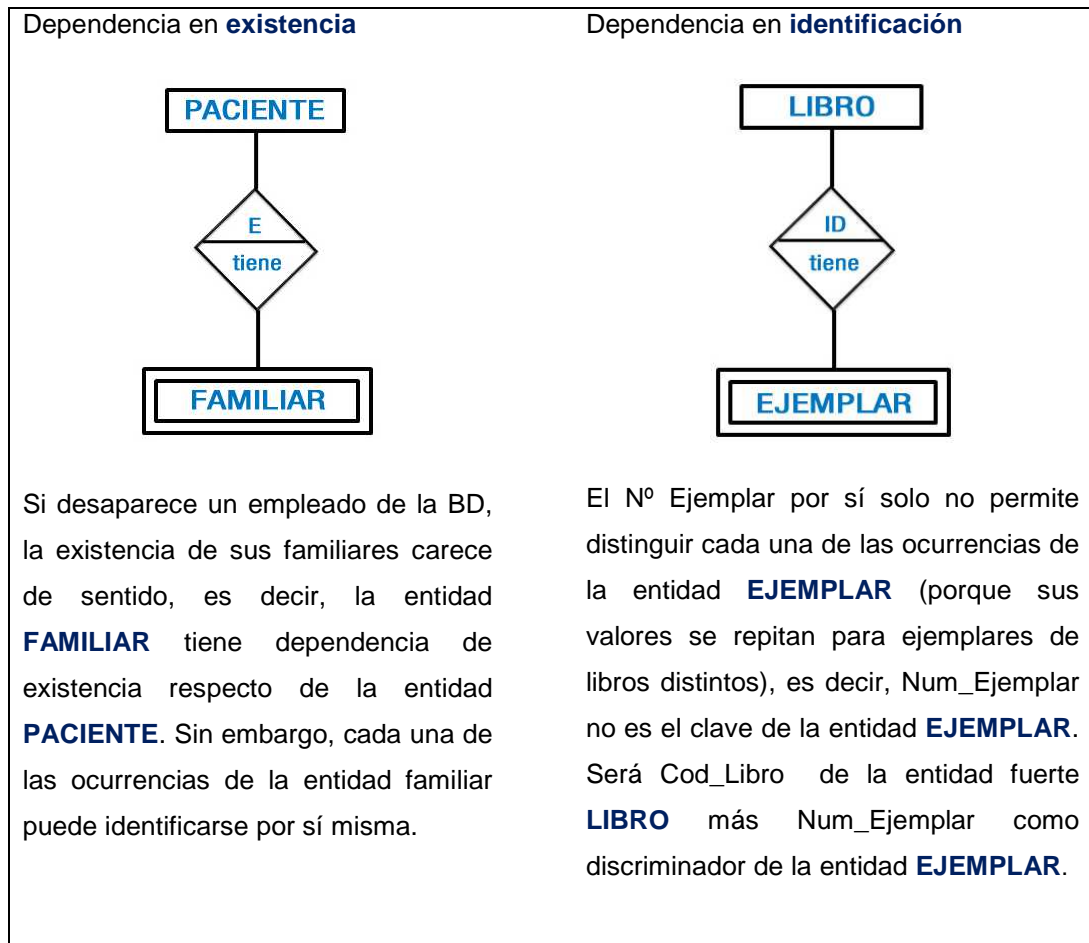
La representación de la entidad AULA será:



Las entidades débiles presentan dos tipos de dependencia:

- **Dependencia en existencia:** entre entidades, si desaparece una instancia de entidad fuerte desaparecerán las instancias de entidad débiles que dependan de la primera. La representación de este tipo de dependencia incluirá una **E** en el interior de la relación débil.
- **Dependencia en identificación:** debe darse una dependencia en existencia y además, una ocurrencia de la entidad débil no puede identificarse por sí misma, debiendo hacerse mediante la clave de la entidad fuerte asociada. La representación de este tipo de dependencia incluirá una **ID** en el interior de la relación débil.

Ejemplo de entidades débiles:



4.- Atributos.

Cada entidad se describe como un conjunto de atributos. Éstos describen características o propiedades que posee cada miembro de un conjunto de entidades. El mismo atributo establecido para un conjunto de entidades o, lo que es lo mismo, para un tipo de entidad, almacenará información parecida para cada ocurrencia de entidad. Pero, cada ocurrencia de entidad tendrá su propio valor para cada atributo.

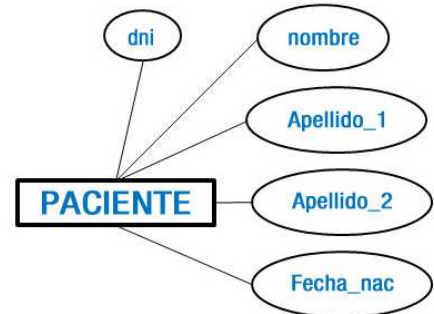
Atributo:

Cada una de las propiedades o características que tiene un tipo de entidad o un tipo de relación se denomina atributo; los atributos toman valores de uno o varios dominios.

Por tanto, un atributo se utilizará para guardar información sobre alguna característica o propiedad de una entidad o relación. Ejemplos de atributos pueden ser: altura, color, peso, DNI, fecha, etc. todo dependerá de la información que sea necesaria almacenar.

En el modelo Entidad/Relación los atributos de una entidad son representados mediante el nombre del atributo rodeado por una elipse. La elipse se conecta con la entidad mediante una línea recta. Cada atributo debe tener un nombre único que haga referencia al contenido de dicho atributo. Los nombres de los atributos se deben escribir en letra minúscula.

Representación de algunos de los atributos para la entidad **PACIENTE**.

**Dominio:**

Al conjunto de valores permitidos para un atributo se le denomina dominio.

Todos los posibles valores que puede tomar un atributo deberán estar dentro del dominio. Varios atributos pueden estar definidos dentro del mismo dominio. Por ejemplo, los atributos nombre, apellido primero y apellido segundo de la entidad PACIENTE, están definidos dentro del dominio de cadenas de caracteres de una determinada longitud.

Por ejemplo:

El dominio del atributo Luis se toma del dominio Nombre por ser uno de los nombres posibles.

El dominio para el atributo nota media de un estudiante sería (0,10) porque el valor más bajo posible es el 0 y el más alto el 10.

El dominio del atributo teléfono estaría dentro del conjunto de combinaciones de 9 cifras que sean números de teléfonos válidos.

El dominio del atributo sexo de un cliente sólo tiene dos valores posibles M o F, etc.

A menudo se confunden los dominios con los tipos de datos, aunque no son lo mismo.

Tipo de datos es un concepto físico, mientras que dominio es un concepto lógico.

Aunque los dominios suelen ser amplios (números enteros, reales, cadenas de caracteres, etc.), a la hora de llevar a cabo el desarrollo de una base de datos, es mejor establecer unos límites adecuados para que el sistema gestor de la base de datos lleve a cabo las verificaciones oportunas en los datos que se almacenen, garantizando así la integridad de éstos.

4.1.- Tipos de atributos.

No todos los atributos son iguales. Existen varias características que hacen que los atributos asociados a una entidad o relación sean diferentes, los clasificaremos según varios criterios.

■ Atributos obligatorios y opcionales.

- **Atributo obligatorio:** es aquél que ha de estar siempre definido para una entidad o relación. Por ejemplo La Clave principal.
- **Atributo opcional:** es aquél que podría ser definido o no para la entidad. Es decir, puede haber ocurrencias de entidad para las que ese atributo no esté definido o no tenga valor.

■ Simples o compuestos.

- **Atributo simple o atómico:** es un atributo que no puede dividirse en otras partes o atributos, presenta un único elemento. No es posible extraer de este atributo partes más pequeñas que puedan tener significado.
- **Atributo compuesto:** son atributos que pueden ser divididos en subpartes, éstas constituirán otros atributos con significado propio. Por ejemplo la dirección(Calle, numero, ciudad)

■ Atributos monovaluados o multivaluados.

- **Atributo monovaluado:** es aquél que tiene un único valor para cada ocurrencia de entidad.
- **Atributo multivaluado:** es aquél que puede tomar diferentes valores para cada ocurrencia de entidad. Por ejemplo: diferentes emails para una misma persona.

■ Atributos derivados o almacenados:

el valor de este tipo de atributos puede ser obtenido del valor o valores de otros atributos relacionados. Un ejemplo clásico de atributo derivado es la edad. Por ejemplo si se ha almacenado en algún atributo la fecha de nacimiento, la edad es un valor calculable a partir de dicha fecha.

4.2.- Claves.

Hemos visto que un atributo se identifica por su nombre, y que hay un tipo de atributo que es obligatorio, a este atributo se la llama atributo identificador o clave.

Está claro que es necesario identificar correctamente cada ocurrencia de entidad o relación, de este modo el tratamiento de la información que se almacena podrá realizarse adecuadamente. Esta distinción podría llevarse a cabo tomando todos los valores de todos los atributos de una entidad o relación. Pero, en algunas ocasiones, sabemos que puede no ser necesario utilizar todos, bastando con un subconjunto de ellos. Aunque puede ocurrir que ese subconjunto tenga idénticos valores para varias entidades, por lo que cualquier subconjunto no será válido.

Por tanto, los valores de los atributos de una entidad deben ser tales que permitan identificar unívocamente a la entidad. En otras palabras, no se permite que ningún par de entidades tengan exactamente los mismos valores de sus atributos.

Clave:

Atributo o conjunto de atributos que identifica unívocamente cada una de las ocurrencias de la entidad.

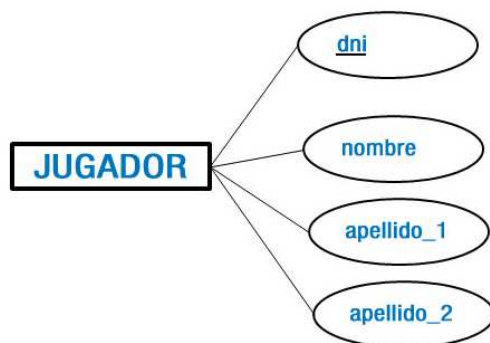
No admite redundancias.

Tipos de claves.

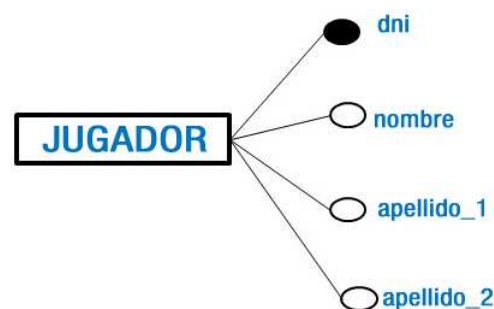
- **Clave candidata:** cualquier atributo que pueda ser clave.
- **Clave primaria** o principal: (**primary key**): Es la clave candidata seleccionada por el diseñador de la base de datos.
Tiene que cumplir:
 - ✓ No puede contener valores nulos.
 - ✓ Ha de ser sencilla de crear.
 - ✓ No puede cambiar con el tiempo.
- **Clave alternativa:** Es cada una de las claves candidatas no seleccionadas se le denomina clave alternativa.
- **Clave Ajena** o extranjera o **foreign key**: son atributos de una entidad que son campos claves de otra entidad o relación.

La representación en el modelo Entidad/Relación de las claves primarias puede realizarse de dos formas:

- Si se utilizan elipses para representar los atributos, se subrayarán aquellos que formen la clave primaria.



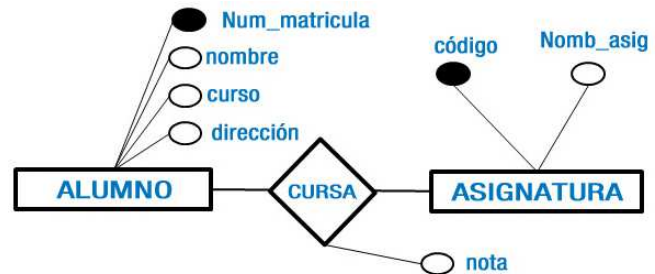
- Si se utilizan círculos para representar los atributos, se utilizará un círculo negro en aquellos que formen la clave primaria.



4.3.- Atributos de una relación.

Una relación entre entidades puede tener atributos que la describan, en este caso el atributo colgará de la relación

Consideremos la relación **curso** entre las entidades **ALUMNO** y **ASIGNATURA**. Podríamos asociar a la relación **curso** un atributo **nota** para especificar la nota que ha obtenido un alumno/a en una determinada asignatura.



Otro ejemplo típico son las relaciones que representan históricos. Este tipo de relaciones suele constar de datos como fecha y hora. Cuando se emite una factura a un cliente o se le facilita un duplicado de la misma, es necesario registrar el momento en el que se ha realizado dicha acción. Para ello, habrá que crear un atributo asociado a la relación entre la entidad CLIENTE y FACTURA que se encargue de guardar la fecha de emisión.

En el modelo **Entidad/Relación** la representación de atributos asociados a relaciones es exactamente igual a la que utilizábamos para entidades. Podremos utilizar una elipse con el nombre del atributo en su interior, conectada con una línea a la relación, o bien, un círculo blanco conectado con una línea a la relación y junto a él, el nombre del atributo. En el gráfico puedes ver esta segunda representación.

En ocasiones cuando una relación tiene atributos, significa que la relación oculta una entidad que hasta ese momento no se ha definido, a esa entidad se le denomina **entidad asociada**. Se representará mediante un rombo inscrito en un rectángulo.



5.- Relaciones.

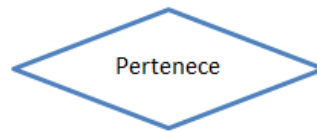
La relación o interrelación es un elemento del modelo **Entidad/Relación** que permite relacionar datos entre sí. En una relación se asocia un elemento de una entidad con otro de otra entidad.

Relación:

Es una asociación entre dos o más entidades que genera información adicional a las entidades que intervienen.

La representación gráfica en el modelo Entidad/Relación corresponde a un **rombo** en cuyo interior se encuentra inscrito el **nombre de la relación**.

Generalmente el nombre de la relación se identifica con un verbo en singular (activo o pasivo)



Cuando debas dar un nombre a una relación procura que éste haga referencia al objetivo o motivo de la asociación de entidades.

El rombo estará conectado con las entidades a las que relaciona, mediante líneas rectas, que podrán o no acabar en punta de flecha según el tipo de relación.

Las relaciones siempre operan en los dos sentidos.

Por ejemplo:

La relación entre JUGADOR y EQUIPO se definen en dos direcciones:



- ✓ Un JUGADOR pertenece a un EQUIPO
- ✓ A un EQUIPO pertenecen muchos JUGADORES

Para describir y definir adecuadamente las relaciones existentes entre entidades, es imprescindible conocer los siguientes conceptos:

- Grado de la relación.
- Cardinalidad de la relación.
- Cardinalidades de las entidades.

5.1.- Grado de una Relación.

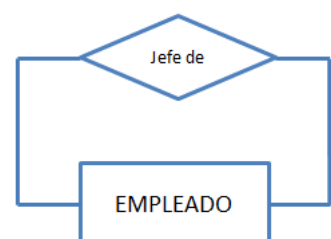
En una relación pueden intervenir una o varias entidades.

Grado de una relación: Número de entidades que participan en una relación.

En función del grado se pueden establecer diferentes tipos de relaciones.

■ Relación Unaria o de grado 1:

Es aquella relación en la que participa **una única entidad**. También llamadas reflexivas o recursivas, ya que la relación es de una entidad consigo misma.



■ Relación Binaria o de grado 2:

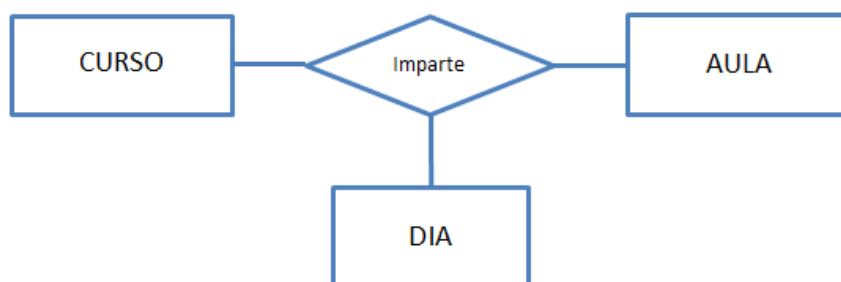
Es aquella relación en la que participan **dos entidades**.



En general, tanto en una primera aproximación, como en los sucesivos refinamientos, el esquema conceptual de la base de datos buscará tener sólo este tipo de relaciones.

■ Relación Ternaria o de grado 3:

Es aquella relación en la que participan **tres entidades** al mismo tiempo.

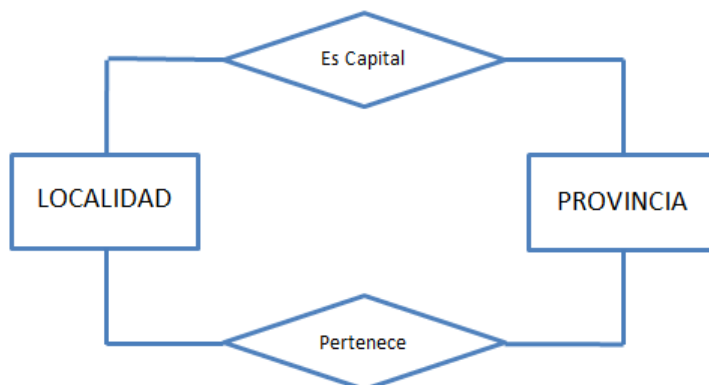


■ Relación N-aria o de grado n:

Es aquella relación que involucra **n entidades**. Este tipo de relaciones no son usuales y deben ser simplificadas hacia relaciones de menor grado.

■ Relación doble:

Ocurre cuando **dos entidades** están relacionadas a través de **dos relaciones**.



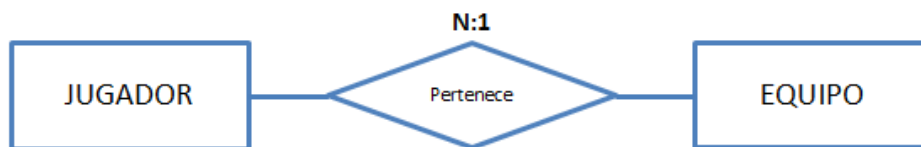
5.2.- Cardinalidad de las relaciones.

En matemáticas, el cardinal de un conjunto es el número de elementos que lo forman. Este concepto puede extrapolarse a las relaciones con las que estamos tratando.

Cardinalidad de una relación:

Es el número máximo de ocurrencias de cada entidad que pueden intervenir en una ocurrencia de relación. La cardinalidad vendrá expresada siempre para relaciones entre dos entidades. Dependiendo del número de ocurrencias de cada una de las entidades pueden existir relaciones uno a uno, uno a muchos, muchos a uno y muchos a muchos.

Por ejemplo:



La cardinalidad indicará el número de ocurrencias de la entidad **JUGADOR** que se relacionan con cada ocurrencia de la entidad **EQUIPO** y viceversa. Podríamos hacer la siguiente lectura: un jugador pertenece a un equipo y a un equipo pueden pertenecer varios jugadores.

Podríamos representar las cardinalidades mediante las etiquetas **1:1**, **1:N**, **N:1**, **M:N** que se leerían respectivamente: uno a uno, uno a muchos, muchos a uno y muchos a muchos.

Veamos en detalle el significado de cada una de estas cardinalidades:

■ Relaciones uno a uno (1:1)

Sean las entidades A y B, **una** instancia u ocurrencia de la entidad A se relaciona únicamente **con otra** instancia de la entidad B y viceversa.

Por ejemplo:

Para cada ocurrencia de la entidad **ALUMNO** sólo habrá una ocurrencia relacionada de la entidad **EXPEDIENTE** y viceversa. O lo que es lo mismo, un alumno tiene un expediente asociado y un expediente sólo pertenece a un único alumno.

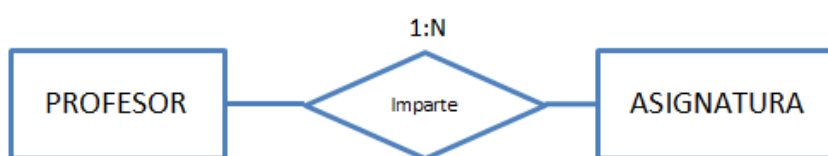


■ Relaciones uno a muchos (1:N)

Sean las entidades A y B, **una** ocurrencia de la entidad A se relaciona **con muchas** ocurrencias de la entidad B y una ocurrencia de la entidad B sólo estará relacionada con una única ocurrencia de la entidad A.

Por ejemplo:

Para cada ocurrencia de la entidad **PROFESOR** puede haber varias ocurrencias de la entidad **ASIGNATURA** y para varias ocurrencias de la entidad **ASIGNATURA** sólo habrá una ocurrencia relacionada de la entidad **PROFESOR** (si se establece que una asignatura sólo puede ser impartida por un único profesor). O lo que es lo mismo, un profesor puede impartir varias asignaturas y una asignatura sólo puede ser impartida por un único profesor.



■ Relaciones muchos a uno (N:1)

Igual que la relación 1:N .

■ Relaciones muchos a muchos (M:N)




Sean las entidades A y B, una ocurrencia de la entidad A está relacionado **con muchas** ocurrencias de la entidad B y una ocurrencia de la entidad B está relacionado **con muchas** ocurrencias de la entidad A

Por ejemplo:

Un alumno puede estar matriculado en varias asignaturas y en una asignatura pueden estar matriculados varios alumnos.



La cardinalidad de las relaciones puede representarse de varias maneras en los esquemas del modelo Entidad/Relación. Otra forma de representarlo sería:

- Relaciones uno a uno: **1: 1** 
- Relaciones uno a muchos: **1: N** 
- Relaciones muchos a muchos: **N: M** 

5.3.- Cardinalidad de entidades.

La cardinalidad con la que una entidad participa en una relación especifica el número mínimo y el número máximo de correspondencias en las que puede tomar parte cada ocurrencia de dicha entidad. Indica el número de relaciones en las que una entidad puede aparecer.

La cardinalidad de una entidad:

Se representa con el número mínimo y máximo de correspondencias en las que puede tomar parte cada ocurrencia de dicha entidad, entre paréntesis.

Su representación gráfica será, por tanto, una etiqueta del tipo (0,1), (1,1), (0,n) o (1,n).

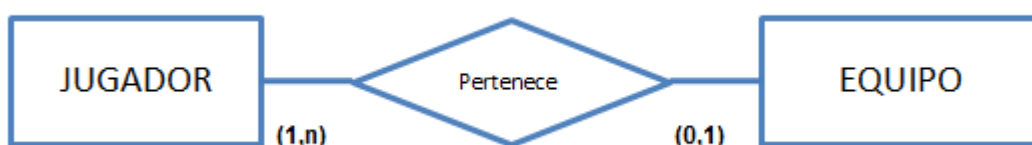
El significado del primer y segundo elemento del paréntesis corresponde a:

(cardinalidad mínima, cardinalidad máxima):

- **Cardinalidad mínima:** Indica el número mínimo de asociaciones en las que aparecerá cada ocurrencia de la entidad (el valor que se anota es de cero o uno, aunque tenga una cardinalidad mínima de más de uno, se indica sólo un uno). El valor 0 se pondrá cuando la participación de la entidad sea opcional.
- Sean las entidades A y B, la participación de la entidad A en una relación es **obligatoria total** si la existencia de cada una de sus ocurrencias necesita **como mínimo de una ocurrencia** de la entidad B. En caso contrario, la participación es opcional o parcial.
- **Cardinalidad máxima:** Indica el número máximo de relaciones en las que puede aparecer cada ocurrencia de la entidad. Puede ser uno, otro valor concreto mayor que uno (tres por ejemplo) o muchos (se representa con n).

Por ejemplo:

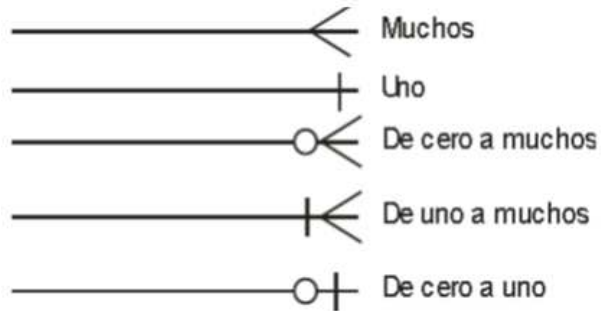
Un **JUGADOR** pertenece como mínimo a ningún **EQUIPO** y como máximo a uno (**0,1**) y, por otra parte, a un **EQUIPO** pertenece como mínimo un **JUGADOR** y como máximo varios (**1, n**).



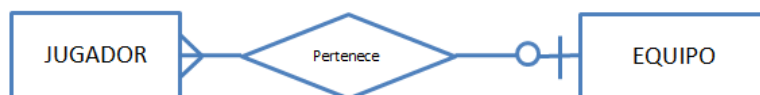
Como puedes ver, la cardinalidad (0,1) de **JUGADOR** se ha colocado junto a la entidad **EQUIPO** para representar que un jugador puede no pertenecer a ningún equipo o como máximo a uno. Para la cardinalidad de **EQUIPO** ocurre igual, se coloca su cardinalidad junto a la entidad **JUGADOR** para expresar que en un equipo hay como mínimo un jugador y máximo varios.

Ten en cuenta que cuando se representa la cardinalidad de una entidad, el paréntesis y sus valores han de colocarse junto a la entidad con la que se relaciona. Es decir en el lado opuesto a la relación.

La cardinalidad de entidades también puede representarse en el modelo Entidad/Relación con la siguiente notación



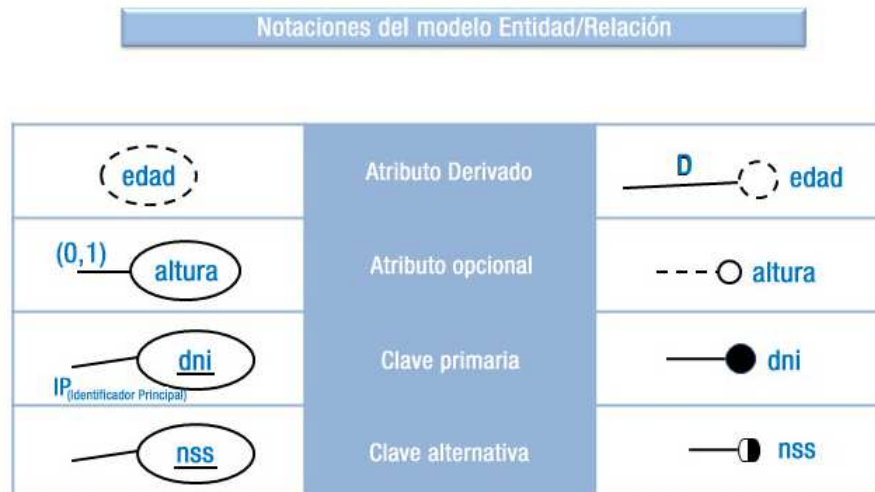
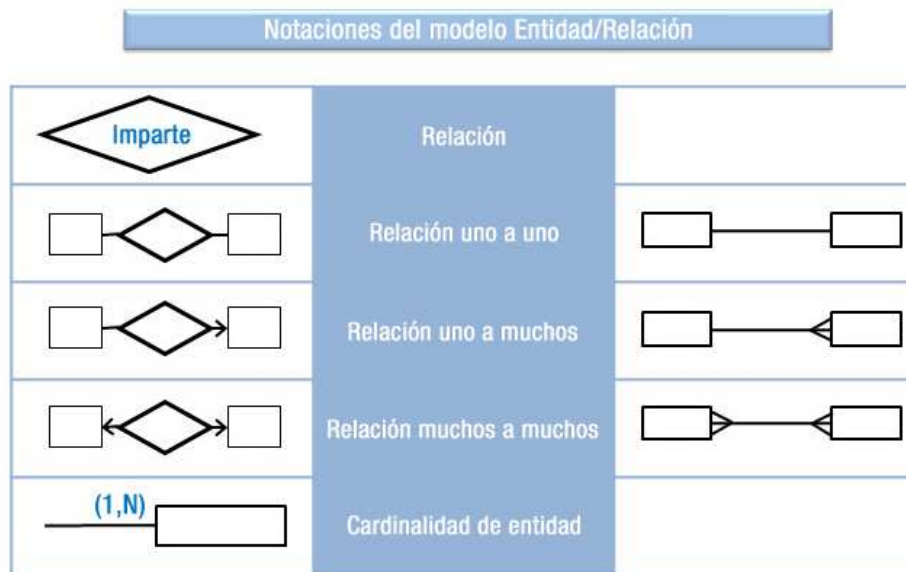
Por tanto, el anterior ejemplo quedaría representado así:



6.- Simbología del modelo E/R

Resumen básico de los símbolos utilizados en el modelo Entidad/Relación. Existen diferentes maneras de representar los mismos elementos.

Notaciones del modelo Entidad/Relación		
	Entidad	
	Entidad Débil	
	Atributo simple o atómico	
	Atributo compuesto	
	Atributo multivaluado	



7.- El modelo E/R extendido.

Tiene como objetivo solventar algunos problemas de representación con la terminología tradicional del modelo E-R.

Con el modelo Extendido se consigue mejorar la capacidad de representar los esquemas aportando más información..

Algunas nuevas características son:

- Tipos de restricciones sobre las relaciones.
- Generalización y Especialización
- Conjunto de entidades de nivel más alto y más bajo
- Herencia de atributos
- Agregación.

7.1.- Restricciones en las relaciones.

■ Restricción de exclusividad.

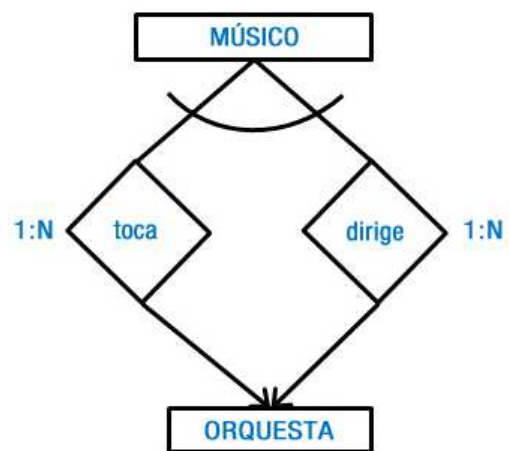
Se da cuando existe **una entidad que participa en dos o más relaciones** y cada ocurrencia de dicha entidad **sólo puede pertenecer a una de las relaciones únicamente**.

Si la ocurrencia de entidad pertenece a una de las relaciones, no podrá formar parte de la otra. O se produce una relación o se produce la otra pero **nunca ambas a la vez**.

La representación gráfica de una restricción de exclusividad se realiza **mediante un arco** que engloba a todas aquellas relaciones que son exclusivas.

Por ejemplo:

Supongamos que un músico puede dirigir una orquesta o tocar en ella, pero no puede hacer las dos cosas simultáneamente. Existirán por tanto, dos relaciones **dirige** y **toca**, entre las entidades **MUSICO** y **ORQUESTA**, estableciéndose una relación de exclusividad entre ellas.



El ejemplo solo plantea la relación de músico a orquesta, no de orquesta a músico.

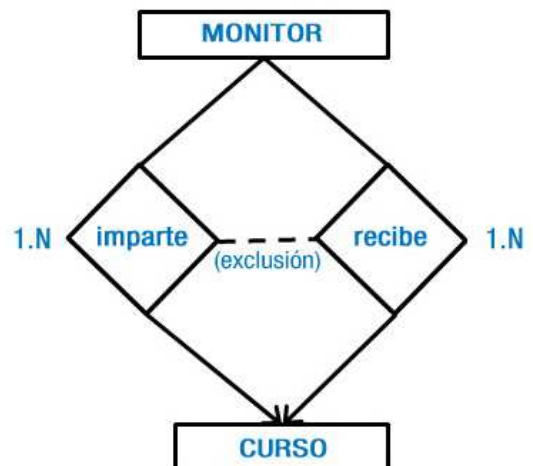
■ Restricción de exclusión.

Se produce cuando **las ocurrencias de las entidades sólo pueden asociarse utilizando una única relación**.

La representación gráfica es mediante **una línea discontinua** entre las dos relaciones.

Por ejemplo:

Supongamos que un monitor puede impartir diferentes cursos de perfeccionamiento para monitores, y que éste puede a su vez recibirlos. Pero un monitor **si imparte un determinado curso, no podrá estar recibéndolo simultáneamente y viceversa**. Se establecerá, por tanto, una restricción de exclusión



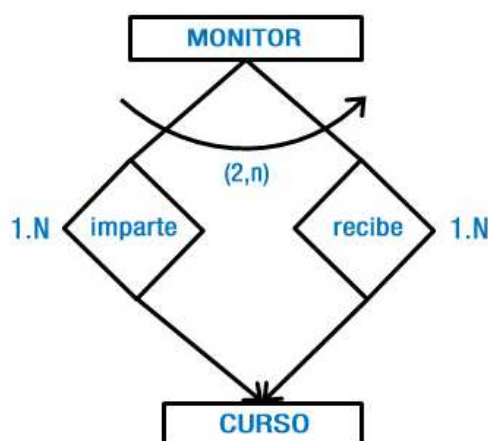
■ Restricción de inclusividad.

Se aplica cuando es necesario modelar **situaciones en las que para que dos ocurrencias de entidad se asocien a través de una relación, tengan que haberlo estado antes a través de otra relación.**

Se representará **mediante un arco acabado en flecha**, que partirá desde la relación que ha de cumplirse primero hacia la otra relación. Se indicará junto al arco **la cardinalidad mínima y máxima** de dicha restricción de inclusividad.

Por ejemplo:

Siguiendo con el ejemplo anterior, supongamos que para que un monitor pueda impartir cursos sea necesario que reciba previamente dos cursos. Puedes ser, que los cursos que el monitor deba recibir no tengan que ser los mismos que luego pueda impartir. Aplicando una restricción de inclusividad entre las relaciones **imparte** y **recibe**, estaremos indicando que cualquier ocurrencia de la entidad **MONITOR** que participa en una de las relaciones (**imparte**) tiene que participar obligatoriamente en la otra (recibe).



En el ejemplo, **(2,n)** indica que un monitor ha de recibir 2 cursos antes de poder impartir varios.

■ Restricción de inclusión.

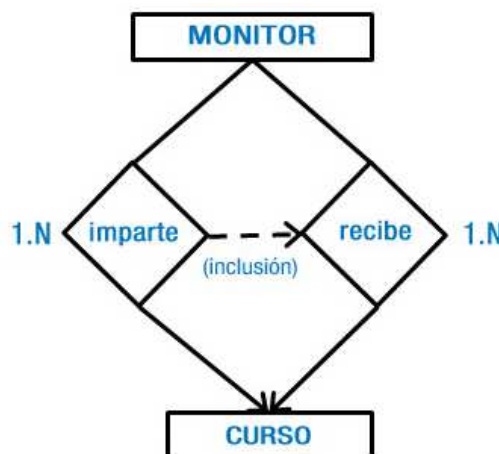
En algunas ocasiones aplicar una restricción de **inclusividad** no representa totalmente la realidad a modelar, entonces se hace necesario aplicar una restricción de inclusión que es aún más fuerte.

La representación gráfica es mediante **una línea discontinua entre las dos relaciones, acabada en flecha**, que partirá desde la relación que ha de cumplirse primero hacia la otra relación.

Por ejemplo:

Si hemos de modelar que un monitor pueda impartir un curso, si previamente lo ha recibido, entonces tendremos que aplicar una restricción de inclusión. Con ella toda ocurrencia de la entidad **MONITOR** que esté asociada a una ocurrencia determinada de la entidad **CURSO**, a través de la relación **imparte**, ha de estar unida a la misma ocurrencia de la entidad **CURSO** a través de la relación **recibe**.

Es decir que para impartir un curso debe haberlo recibido previamente.



7.2.- Generalización y especificación.

Cuando estamos diseñando una base de datos puede que nos encontremos con conjuntos de entidades que posean características comunes, lo que permitiría crear un tipo de entidad de nivel más alto que englobase dichas características. Y a su vez, puede que necesitemos dividir un conjunto de entidades en diferentes subgrupos de entidades por tener éstas, características diferenciadoras. Este proceso de refinamiento **ascendente/descendente**, permite expresar **mediante la generalización** la existencia de tipos de entidades de nivel superior que engloban a conjuntos de entidades de nivel inferior.

A los conjuntos de **entidades de nivel superior** también se les denomina **superclase o supertipo**. A los conjuntos de **entidades de nivel inferior** se les denomina **subclase o subtipo**.

Por tanto, existirá la posibilidad de realizar **una especialización de una superclase en subclases**, y análogamente, establecer **una generalización de las subclases en superclases**.

La generalización es la reunión en una superclase o supertipo de entidad de una serie de subclases o subtipos de entidades, que poseen características comunes. Las subclases tendrán otras características que las diferenciarán entre ellas.

¿Cómo detectamos una generalización? Podremos identificar una generalización cuando encontremos una serie de atributos comunes a un conjunto de entidades, y otros atributos que sean específicos. Los atributos comunes conforman la superclase o supertipo y los atributos específicos la subclase o subtipo.

Las jerarquías se caracterizan por un concepto que hemos de tener en cuenta, **la herencia**. A través de la herencia los atributos de una superclase de entidad son heredados por las subclases. Si una superclase interviene en una relación, las subclases también lo harán.

¿Cómo se representa una generalización o especialización? Existen varias notaciones, pero hemos de convenir que la relación que se establece entre una superclase de entidad y todos sus subtipos se expresa a través de las palabras **ES UN**, o en notación inglesa **IS A**, que correspondería con **ES UN TIPO DE**. Partiendo de este punto, una jerarquía se representa **mediante un triángulo invertido**, sobre él quedará la entidad superclase y conectadas a él a través de líneas rectas, las subclases.



Por ejemplo:



Vemos que las subclases **INVITADO**, **REGISTRADO** y **ADMINISTRADOR** constituyen subclases de la superclase **USUARIO**. Cada una de ellas aporta sus propias características y heredan las pertenecientes a su superclase.

Una **generalización/especialización** podrá tener las siguientes restricciones semánticas:

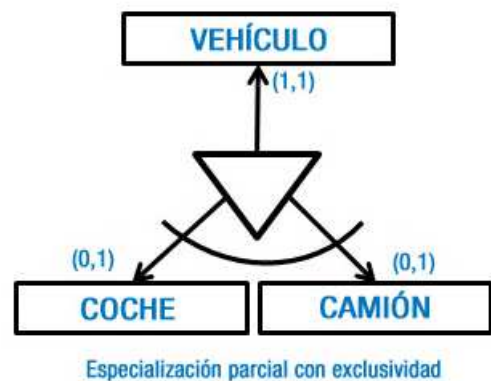
- **Totalidad:**

Una generalización/especialización será total **si todo ejemplar** de la superclase pertenece **a alguna** de las subclases.



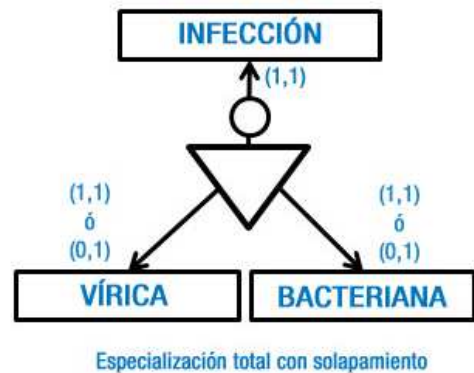
- **Parcialidad:**

Una generalización/especialización será parcial **si no todos los ejemplares** de la superclase pertenecen **a alguna** de las subclases.



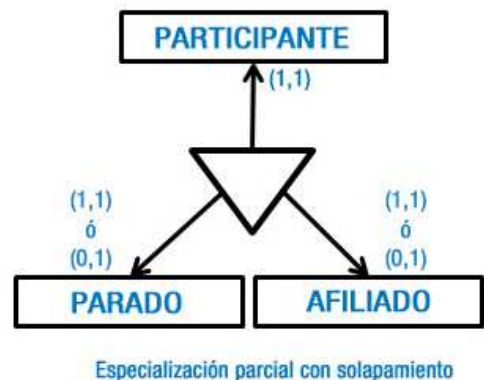
- **Solapamiento:**

Una generalización/especialización presentará solapamiento **si un mismo ejemplar** de la superclase puede pertenecer **a más de una** subclase.



- **Exclusividad:**

Una generalización/especialización presentará exclusividad **si un mismo ejemplar** de la superclase pertenece **sólo a una** subclase.



7.3.- Agregación.

En el modelo Entidad/Relación no es posible representar relaciones entre relaciones.

La **agregación** es una **abstracción** a través de la cual las relaciones se tratan como entidades de nivel más alto, siendo utilizada para expresar **relaciones entre relaciones** o **entre entidades y relaciones**.

La **representación gráfica** de una agregación se caracteriza **por englobar con un rectángulo las entidades y relación a abstraer**. De este modo, se crea una nueva entidad agregada que puede participar en otras relaciones con otras entidades.

Es un tipo especial de interrelación en la cual la **cardinalidad mínima y máxima** de la entidad agregada **siempre será (1,1)** no indicándose por ello en el esquema.

Por Ejemplo:

Supongamos que hemos de modelar la siguiente situación:

Una empresa de selección de personal realiza entrevistas a diferentes aspirantes. Puede ser que, de algunas de estas entrevistas a aspirantes, se derive una oferta de empleo, o no.

Vamos a presentar tres soluciones, las dos primeras erróneas y una tercera correcta, utilizando una agregación.

• **Solución 1:**

Errónea ya que estaríamos representando que, por cada entrevista realizada por una empresa a un aspirante, se genera una oferta de empleo.



• **Solución 2:**

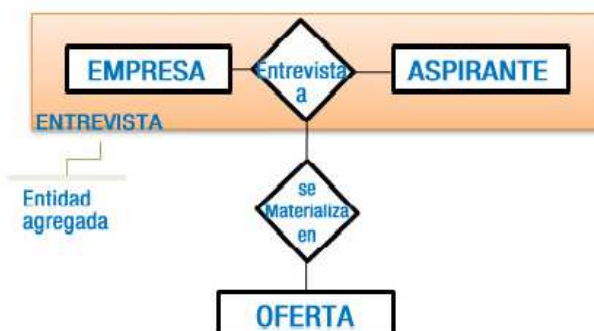
Errónea, porque en el modelo E/R no pueden establecerse relaciones entre varias relaciones



• **Solución 3:**

En el modelo E/R extendido, podemos crear una entidad agregada llamada ENTREVISTA compuesta por la relación "Entrevista a" que existe entre EMPRESA y ASPIRANTE.

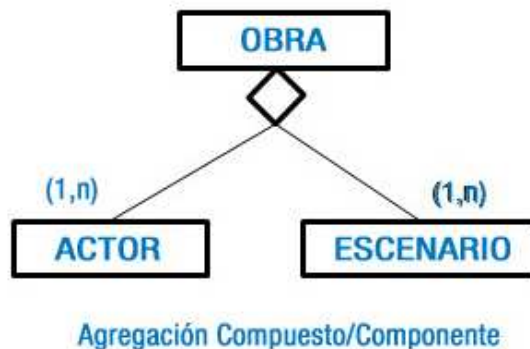
Luego entre esta nueva entidad y OFERTA podemos establecer la relación "se materializa en".



Existen dos clases de agregaciones:

• **Compuesto/componente:**

Un todo se obtiene por la unión de diversas partes, que pueden ser objetos distintos y que desempeñan papeles distintos en la agregación. Teniendo esto en cuenta, esta abstracción permite representar que un todo o agregado se obtiene por la unión de diversas partes o componentes que pueden ser tipos de entidades distintas y que juegan diferentes roles en la agregación.

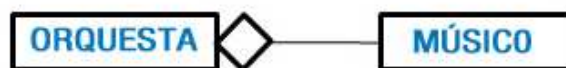


- **Miembro/Colección:**

Un todo se obtiene por la unión de diversas partes del mismo tipo y que desempeñan el mismo papel en la agregación.

Teniendo esto en cuenta, esta abstracción permite representar un todo o agregado como una colección de miembros, todos de un mismo tipo de entidad y todos jugando el mismo rol. Esta agregación puede incluir una restricción de orden de los miembros dentro de la colección (indicando el atributo de ordenación).

Es decir, permite establecer un orden entre las partes.



Agregación Miembro/Colección

8.- Elaboración de modelo E/R.

Sabemos que en la fase de diseño conceptual de la base de datos, en la que nos encontramos, hemos de generar el diagrama E/R que **representará de manera más sencilla** el problema real a modelar, independientemente del Sistema Gestor de Base de Datos. Este esquema será como un plano que facilite la comprensión y solución del problema. Este diagrama estará compuesto por la representación gráfica, a través de la simbología vista, de los requisitos o condiciones que se derivan del problema a modelar.

Saltarnos este paso en el proceso de creación e implementación de una base de datos, supondría pérdida de información. Por lo que esta fase, requerirá de la creación de uno o varios esquemas previos más cercanos al mundo real, antes del paso a tablas del modelo relacional.

Los diagramas no siempre se crean del mismo modo y, en ocasiones, hay que retocarlos e incluso rehacerlos. A través de la resolución de diferentes problemas y la elaboración de múltiples diagramas, obtendrás la destreza necesaria para generar esquemas que garanticen una posterior y correcta conversión del modelo Entidad/Relación al modelo Relacional.

8.1.- Identificación de entidades y relaciones.

Lo primero que hemos de tener a nuestra disposición para poder generar un diagrama E/R adecuado es el conjunto de requerimientos, requisitos o condiciones que nuestra base de datos ha de cumplir. Es lo que se denomina el **documento de especificación de requerimientos**. En otras palabras, el enunciado del problema a modelar. Cuanto más completa y detallada sea la información de la que dispongamos, mucho mejor.

Suponiendo que conocemos la simbología del modelo Entidad/Relación y que entendemos su significado ¿Cómo empezamos?

Las etapas para la creación del diagrama E/R se detallan a continuación:

- **Identificación de entidades:**

Es un proceso bastante intuitivo. Para localizar aquellos elementos que serán las entidades de nuestro esquema, analizaremos la especificación de requerimientos en **busca de nombres o sustantivos**. Si estos nombres se refieren a objetos importantes dentro del problema probablemente serán entidades. Tendremos en cuenta que nombres referidos a características, cualidades o propiedades no se convertirán en entidades.

Otra forma de identificar entidades es localizando objetos o elementos que existen por sí mismos. Por ejemplo: **VEHICULO**, **PIEZA**, etc. En otras ocasiones, la localización de varias características o propiedades puede dejar ver la existencia de una entidad.

¿Esto puede ser una entidad o no? Es una pregunta que se repite mucho cuando estamos en esta etapa. Algunos autores indican que para poder considerarse como entidad se deben cumplir tres reglas:

- ✓ **Existencia propia.**
- ✓ **Cada ejemplar** de un tipo de entidad debe poder **ser diferenciado del resto** de ejemplares.
- ✓ **Todos los ejemplares** de un tipo de entidad deben tener las **mismas propiedades**.

El número de entidades obtenidas debe ser manejable y según se vayan identificando se les otorgará nombres, **preferiblemente en mayúsculas**, representativos de su significado o función. De esta manera el diagrama será cada vez más legible.

- **Identificación de relaciones:**

Localizadas las entidades, debemos establecer qué relación existe entre ellas. Para ello, analizaremos de nuevo el documento de especificación de requerimientos **en busca de verbos o expresiones verbales** que conecten unas entidades con otras. En la gran mayoría de ocasiones encontraremos que las relaciones se establecen entre dos entidades (relaciones binarias), pero prestaremos especial atención a las relaciones entre más entidades y a las relaciones recursivas o relaciones **unarias**.

Cada una de las relaciones establecidas deberá tener asignado un nombre, **preferiblemente en minúsculas**, representativo de su significado o acción.

Dependiendo de la notación elegida, **el siguiente paso será la representación de la cardinalidad** (mínima y máxima) de las entidades participantes en cada relación y del tipo de correspondencia de la relación (1 a 1, 1 a muchos o muchos a muchos).

Si hemos encontrado alguna relación **recursiva**, reflexiva o unaria, hemos de representar en nuestro esquema los roles desempeñados por la entidad en dicha relación.

8.2.- Identificación de atributos, claves, y jerarquías.

Sólo con la localización de entidades y relaciones no está todo hecho. Hemos de completar el proceso realizando las siguientes tareas:

- **Identificación de atributos:**

Volvemos sobre el documento de especificación de requerimientos para buscar nombres relativos a características, propiedades, identificadores o cualidades de entidades o relaciones. Resulta más sencillo si nos preguntamos ¿Qué información es necesario tener en cuenta de una u otra entidad o relación? Quizás no todos los atributos estén reflejados directamente en el documento de especificación de requerimientos, aplicando el sentido común el diseñador podrá establecerlos en algunos casos y en otros, será necesario consultar e indagar en el problema.

Tendremos en cuenta si los atributos localizados son simples o compuestos, derivados o calculados y si algún atributo o conjunto de ellos se repite en varias entidades. Si se da este último caso, deberemos detenernos y plantear la posibilidad de establecer una jerarquía de especialización, o bien, dejar las entidades tal y como han sido identificadas.

Cada atributo deberá tener **asignado un nombre, preferiblemente en minúsculas**, representativo de su contenido o función. Además, siempre es recomendable recopilar la siguiente información de cada atributo:

- ✓ Nombre y descripción.
- ✓ Atributos simples que lo componen, si es atributo compuesto.
- ✓ Método de cálculo, si es atributo derivado o calculado.

En el caso de encontrar atributos asociados a relaciones con cardinalidad uno a muchos, se valorará asignar ese atributo o atributos a la entidad con mayor cardinalidad participante en la relación.

- **Identificación de claves:**

Del conjunto de atributos de una entidad se establecerán **una o varias claves candidatas**, escogiéndose una de ellas como **clave o llave primaria** de la entidad.

Esta clave estará formada por **uno o varios atributos que identificarán de manera unívoca** cada ocurrencia de entidad. El proceso de identificación de claves permitirá determinar la fortaleza (al menos una clave candidata) o debilidad (ninguna clave candidata) de las entidades encontradas.

Se representará la existencia de esta clave primaria mediante la notación elegida para la elaboración el diagrama E/R. Del mismo modo, se deberán representar adecuadamente las entidades fuertes o débiles.

- **Determinación de jerarquías:**

Es probable que existan **entidades con características comunes** que puedan ser generalizadas en una entidad de nivel superior o superclase (jerarquía de generalización). Pero también, puede ser necesario expresar en el esquema **las particularidades de diferentes** ejemplares de un tipo de entidad, por lo que se crearán subclases o subtipos de una superclase o supertipo (jerarquía de especialización). Para ello, habrá que analizar con detenimiento el documento de especificación de requerimientos.

Si se identifica algún tipo de jerarquía, se deberá representar adecuadamente según el tipo de notación elegida, determinando si la jerarquía es total/parcial o exclusiva/con solapamiento.

8.3.- Metodologías.

Las metodologías o estrategias disponibles para la elaboración del esquema conceptual son las siguientes:

- **Metodología Descendente (Top-Down):**

Se trata de partir de un esquema general e ir descomponiendo éste en niveles, cada uno de ellos con mayor número de detalles. Se parte de objetos muy abstractos, que se refinan paso a paso hasta llegar al esquema final.

- **Metodología Ascendente (Bottom-Up):**

Inicialmente, se parte del nivel más bajo, los atributos. Se irán agrupando en entidades, para después ir creando las relaciones entre éstas y las posibles jerarquías hasta obtener un diagrama completo. Se parte de objetos atómicos que no pueden ser descompuestos y a continuación se obtienen abstracciones u objetos de mayor nivel de abstracción que forman el esquema.

- **Metodología Dentro-fuera (Inside-Out):**

Inicialmente se comienza a desarrollar el esquema en una parte del papel y a medida que se analiza la especificación de requerimientos, se va completando con entidades y relaciones hasta ocupar todo el documento.

- **Metodología Mixta:**

Es empleada en problemas complejos. Se dividen los requerimientos en subconjuntos que serán analizados independientemente. Se crea un esquema que servirá como estructura en la que irán interconectando los conceptos importantes con el resultado del análisis de los subconjuntos creados. Esta metodología utiliza las técnicas ascendente y descendente. Se aplicará la técnica descendente para dividir los requerimientos y en cada subconjunto de ellos, se aplicará la técnica ascendente.

Cualquiera de estas metodologías puede ser válida, todo dependerá de lo fácil y útil que te resulte aplicarlas. Probablemente y, casi sin ser consciente de ello, tú mismo crearás tu propia metodología combinando las existentes.

8.4.- Propiedades deseables de un diagrama E/R.

Cuando construimos un diagrama Entidad/Relación existen una serie de propiedades o características que éste debería cumplir. Quizá no se materialicen todas, pero hemos de intentar cubrir la gran mayoría de ellas. De este modo, conseguiremos que nuestros diagramas o esquemas conceptuales tengan mayor calidad.

Estas características o propiedades deseables se desglosan a continuación:

- **Compleitud:**

Un diagrama E/R será completo si es posible verificar que cada uno de los requerimientos está representado en dicho diagrama y viceversa, cada representación del diagrama tiene su equivalente en los requerimientos.

- **Corrección:**

Un diagrama E/R será correcto si emplea de manera adecuada todos los elementos del modelo Entidad/Relación. La corrección de un diagrama puede analizarse desde dos vertientes:

- ✓ **Corrección sintáctica:**

Se producirá cuando no se produzcan representaciones erróneas en el diagrama.

- ✓ **Corrección semántica:**

Se producirá cuando las representaciones signifiquen exactamente lo que está estipulado en los requerimientos.

Posibles errores semánticos serían: la utilización de un atributo en lugar de una entidad, el uso de una entidad en lugar de una relación, utilizar el mismo identificador para dos entidades o dos relaciones, indicar erróneamente alguna cardinalidad u omitirla, etc.

- **Minimalidad:**

Un diagrama E/R será mínimo si se puede verificar que al eliminar algún concepto presente en el diagrama, se pierde información. Si un diagrama es redundante, no será mínimo.

- **Sencillez:**

Un diagrama E/R será sencillo si representa los requerimientos de manera fácil de comprender, sin artificios complejos.

- **Legibilidad:**

Un diagrama E/R será legible si puede interpretarse fácilmente. La legibilidad de un diagrama dependerá en gran medida del modo en que se disponen los diferentes elementos e interconexiones. Esta propiedad tiene mucho que ver con aspectos estéticos del diagrama.

- **Escalabilidad:**

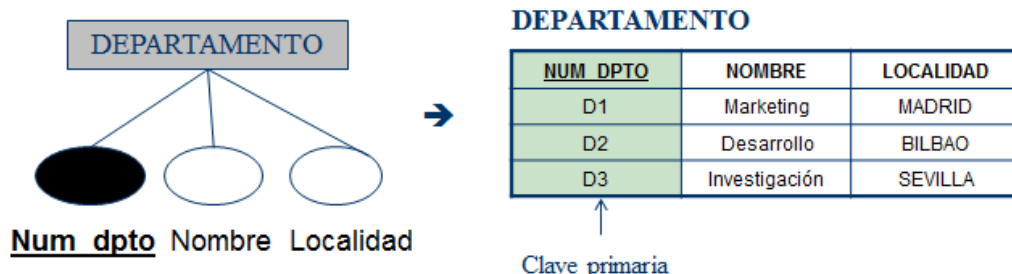
Un diagrama E/R será escalable si es capaz de incorporar posibles cambios derivados de nuevos requerimientos.

9.- Paso del diagrama E/R al modelo relacional.

Obtenido el modelo E-R, definimos el modelo lógico de BD en el que vamos a implementar dicho modelo, es decir, pensamos ya en el SGBD que se va a utilizar (en nuestro caso Oracle):

- Toda **entidad** se transforma **en una tabla**
- Todo **atributo** se transforma **en columna** dentro de la tabla.
- El **atributo clave** de la entidad se convierte **en clave primaria** de la tabla y se representará **subrayado** en la tabla.

Ejemplo: Una entidad → tabla
un atributo → columna de la tabla

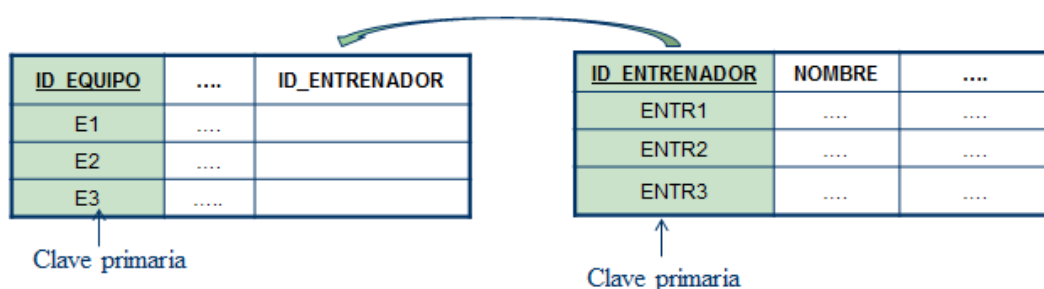


- **Relaciones 1:1** podrán generar una nueva tabla o propagar la clave en función de la cardinalidad.
- ✓ Si una entidad tiene cardinalidad (0,1) y otra (1,1): Se propaga la clave de la entidad con cardinalidad (1,1) a la entidad que tenga (0,1).



ENTRENADOR (dniEnt, nombre, apellidos, dirección, teléfono, apodo)

EQUIPO (nombreEg, campo, ciudad, dniEnt)



- ✓ **Si ambas tienen cardinalidad (1,1):** hay dos opciones.



Opción 1:

Se propaga la clave de cualquier tabla a la otra, según cuál sea la que tenga accesos más frecuentes.

PAIS (codPais, nombrePa, numeHaPa, codCap)

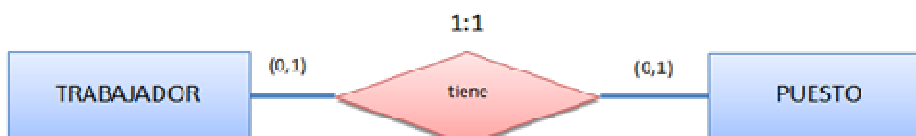
CAPITAL (codCap, nombreCap, numeHaCap)

Opción 2:

Se crea una única tabla, fusionando los atributos de ambas entidades. La clave primaria sería cualquiera de las dos claves primarias, y la otra clave será clave alternativa. El nombre de la tabla será el de la entidad más importante desde el punto de vista conceptual.

PAIS (codPais, nombrePa, numeHaPa, codCap, nombreCap, numeHaCap)

- ✓ **Si ambas tienen cardinalidad (0,1):** La relación se convierte en una nueva tabla a partir de la relación con las dos claves de ambas y algún atributo de la relación si lo hubiera.



TRABAJADOR (dniTrab, nombreTrab, apellidosTrab, direccionTrab, telefonoTrab)

PUESTO (codigoPu, nombrePu, lugarPu)

TRABAJADOR-PUESTO (dniTrab, codigoPu, fechaContrato)

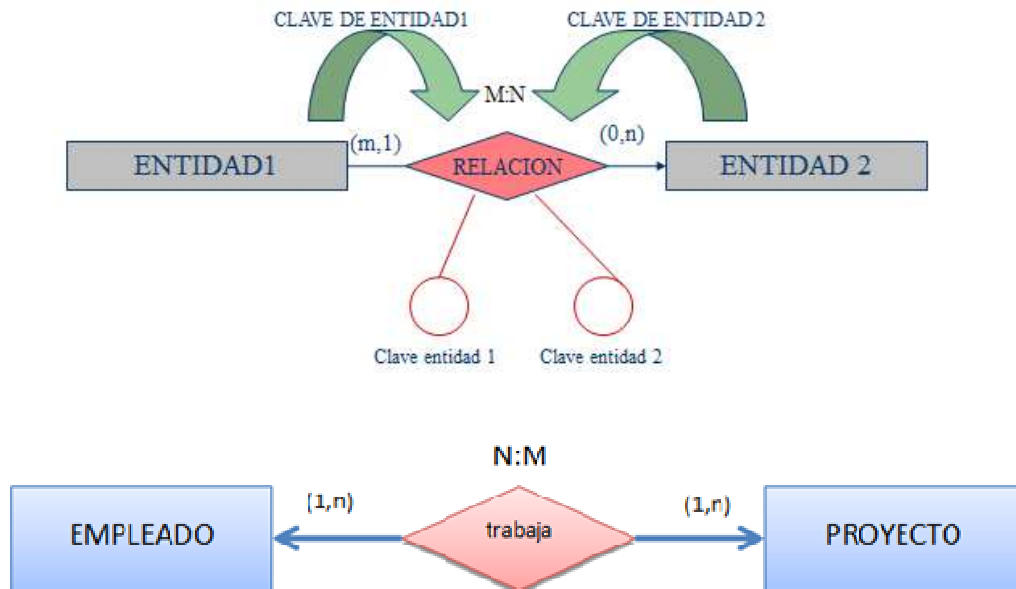
- **Relaciones 1:N:** En estas relaciones la clave primaria de la entidad cuya cardinalidad es (1) se añade como campo a la entidad que tiene cardinalidad N



DEPARTAMENTO (codigoDep, nombreDep)

EMPLEADO (dniEmp, nombreEmp, apellidosEmp, direccionEmp, telefonoEmp, codigoDep)

- **Relaciones N:M:** De estas relaciones se crea una nueva tabla que tendrá como clave primaria la concatenación de los atributos clave de las entidades que asocia y con los atributos propios de la relación si los hay.

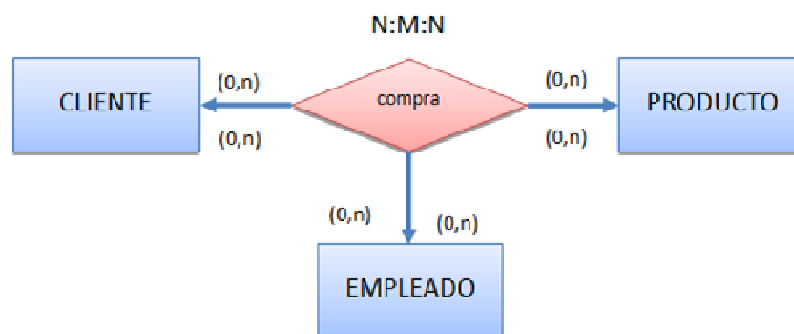


EMPLEADO (codEm, nombreEm, apellidosEmp, direccionEmp, telefonoEm)

PROYECTO (codPro, nombrePro,

EMPLE-PRO (codEmp, codPro, fechaEntrega)

- **Relaciones N:M:N:** En estas relaciones se crea una nueva tabla que tendrá como clave primaria la concatenación de los atributos clave de las entidades que asocia y con los atributos propios de la relación si los hay.



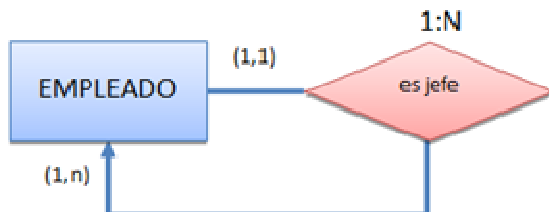
CLIENTE (dniCli, nombreCli, apellidosCli, direccionCli, telefonoCli)

PRODUCTO (codigoPro, descripción, precio)

EMPLEADO (dniEmp, nombreEmp, apellidosEmp, direccionEmp, telefonoEmp)

COMPRA (dniCli, codigoPro, dniEmp, fecha)

- **Relaciones recursivas:** Este tipo de relaciones se tratan de la misma forma que las otras, solo hay que suponer que se trata de una relación binaria normal en la que las dos entidades son iguales.

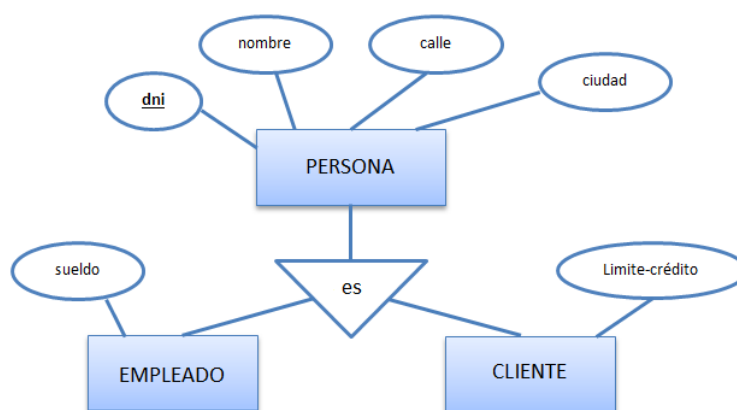


La transformamos en:



EMPLEADO (codEm, nombreEm, apellidosEmp, direccionEmp, telefonoEm, codEmJefe)

- **Generalización:** Hay tres posibilidades.
 - 1.- Se crea una tabla para el conjunto de entidades de nivel más alto y para cada conjunto de entidades de nivel más bajo se crea una tabla que incluya una columna para cada uno de los atributos de ese conjunto de entidades más una columna por cada atributo de la clave primaria del conjunto de las entidades de nivel más alto.



PERSONA (dni, nombre, calle, ciudad)

EMPLEADO (dni, sueldo)

CLIENTE (dni, línea-credito)

- 2.- En el caso de que la generalización sea **disjunta/exclusiva** y **completa/total** no se crea una tabla para el conjunto de entidades de nivel más alto sino que para cada conjunto de entidades de nivel más bajo se crea una tabla que incluya una columna por cada atributo del conjunto de entidades, más una columna por cada atributo del conjunto de entidades del nivel más alto.

EMPLEADO (dni, nombre, calle, ciudad, sueldo)

CLIENTE (dni, nombre, calle, ciudad,, línea-credito)

- 3.- Englobar todos los atributos en una sola entidad, añadiendo el atributo discriminante como una columna más y dejando en blanco los atributos que no le correspondan.

PERSONA (dni, nombre, calle, ciudad, **tipo**, sueldo, límite-crédito)

Donde **tipo** = EMPLEADO o CLIENTE

Esta solución es buena cuando las subclases se diferencian en muy pocos atributos y las relaciones que los asocian con el resto de las entidades del esquema son las mismas para todos. Tiene la ventaja de que las consultas son más rápidas, pero el inconveniente de tener valores nulos en los atributos que no son de dicho subtipo.

10.- Normalización.

Siempre que se diseña un sistema informático, (no solo una base de datos), se ha de medir su calidad, y si no se cumplen determinados criterios de calidad, hay que realizar de forma iterativa, sucesivos refinamientos en el diseño, para alcanzar la calidad deseada.

Uno de los parámetros que mide la calidad de una base de datos es **la forma normal** en la que se encuentra su diseño. Esta forma normal puede alcanzarse cumpliendo ciertas restricciones que impone cada forma normal al conjunto de atributos de un diseño.

Si en nuestro diseño de base de datos hemos seguido el modelo E/R parece que deberíamos obtener tablas bien estructuradas pero es posible que un buen diseño produzca tablas defectuosas. Diseñar tablas bien estructuradas nos ayudará a controlar la redundancia de los datos y evitar la inconsistencia. Estos problemas, cuando existan, pueden ser eliminados mediante el proceso de **normalización**.

Normalización:

Es un proceso que consiste en imponer a las tablas del modelo relacional una serie de restricciones consiguiendo que las tablas contengan los atributos necesarios y suficientes para describir la realidad de la entidad que representan y separando aquellos que podrían generar la creación de otra tabla garantizando:

- Suprimir dependencias erróneas entre atributos.
- Optimizar procesos de inserción, modificación y borrado en las bases de datos

Se aplica mediante una serie de etapas denominadas formas normalizadas o formas normales. Las 3 primeras de esas etapas se denominan **1FN**, **2FN** y **3FN**. Cada una de ellas representa un nivel superior al anterior. Para la mayoría de las bases de datos 3FN es el nivel al que se debería llegar. En algunas aplicaciones, sin embargo, llegaremos hasta la forma normal **Boyce-Codd**, después la 4FN, y en aplicaciones muy especializadas de investigación estadística será necesario llegar más allá, hasta la 5FN.

Es un proceso que se realiza en varias etapas secuenciales. Cada etapa está asociada a una forma normal, que establece unos requisitos a cumplir por la tabla sobre la que se aplica..

Como hemos indicado, el paso de una forma normal a otra es consecutivo, si no se satisface una determinada forma normal no puede pasarse al análisis de la siguiente. Según vamos avanzando en la normalización, los requisitos a cumplir serán cada vez más restrictivos, lo que hará que nuestro esquema relacional sea cada vez más robusto.

Como norma general, para garantizar que no existan problemas en la actualización de datos, es recomendable aplicar el proceso de normalización hasta Tercera Forma Normal o incluso hasta Forma Normal de Boyce-Codd.

10.1.- Tipos de Dependencias.

Para aplicar correctamente la normalización es necesario partir del concepto de dependencia. La dependencia es un conjunto de restricciones que se imponen a determinados atributos de las tablas.

■ Dependencias funcionales:

Se dice que un atributo tiene dependencia funcional de otro cuando a cada valor del primero le corresponde un solo valor del segundo.

Si A y B son atributos de una tabla, diremos que **B depende funcionalmente de A**, si y solo si, para cada valor de A sólo puede existir un único valor de B.

La dependencia funcional siempre se establece entre atributos de una misma tabla. El atributo A se denomina determinante, ya que A determina el valor de B.

Para representar esta dependencia funcional utilizamos la siguiente notación:

A → B

Hay que indicar que A y B podrían ser un solo atributo o un conjunto de ellos.

Por ejemplo.

Si consideramos la tabla (o relación) **EMPLEADO** (**dni**, **nombre**, **direccion**, **telefono**, **departamento**) es evidente que el nombre de una persona depende funcionalmente del **dni**, es decir, para un **dni** determinado existe sólo un nombre que le corresponda. Sin embargo, si consideramos el ejemplo al revés, **dni** no depende funcionalmente de **nombre**, ya que para un mismo nombre puede haber muchos **dni** diferentes.

¡Piensa en la cantidad de personas con el mismo nombre y apellidos que hay en este país, y cada uno tiene su DNI, que lo identifica!

En este caso, si suponemos que cada persona tiene un único teléfono y una única dirección, y que además trabaja en un único departamento de la empresa, las dependencias funcionales quedan como siguen:

EMPLEADO

dni \longrightarrow **nombre, direccion, telefono, departamento**

No obstante, si en nuestro problema necesitáramos almacenar (o quisiéramos permitir que se almacenara) más de una dirección para una misma persona, o más de un teléfono, o que trabajara en más de un departamento, esos tres atributos no dependerían funcionalmente de **DNI**. A eso es a lo que nos referimos cuando decimos que las dependencias funcionales vienen definidas por la semántica del problema.

Otro ejemplo:

PRODUCTOS (**codigoProducto**, **nombre**, **precio**, **descripción**)

Podemos decir que **codigoProducto** \rightarrow **nombre**, puesto que un código de producto solo puede tener asociado un único nombre,

■ **Dependencia funcional completa:**

Cuando un atributo depende de otro que es un atributo compuesto (un conjunto de atributos), se dice que la dependencia funcional es completa si el atributo dependiente no depende de ningún subconjunto del atributo compuesto.

En una dependencia funcional $A \rightarrow B$, cuando A es un conjunto de atributos formado por $A1, A2, A3, \dots$, decimos que la **dependencia funcional es completa** si B solo depende de A , no de ningún subconjunto de A .

Para representar esta dependencia funcional completa utilizamos la siguiente notación:

$$A \Rightarrow B$$

Si A está formado por A_1, A_2

$$A \longrightarrow B$$

B depende funcionalmente de A

Pero

$$A_1 \not\longrightarrow B \quad (B \text{ No depende funcionalmente de } A_1)$$

y

$$A_2 \not\longrightarrow B \quad (B \text{ No depende funcionalmente de } A_2)$$

Por ejemplo:

Si consideramos la relación:

PROYECTO (cod_proyecto, dni_empleado, fecha_incorporacion, horas_dedicadas)

Está claro que los atributos **fecha_incorporacion** y **horas_dedicadas** tienen dependencia funcional completa respecto de **cod_proyecto** y **dni_empleado**. Es evidente que la **fecha_incorporacion** a un proyecto de un empleado depende tanto del proyecto al que se incorpora como del empleado que se incorpora. Un mismo empleado puede pertenecer a varios proyectos a los que se incorpora en diferentes fechas. Igual ocurre con el atributo **horas_dedicadas**.

■ Dependencia funcional elemental:

Se produce cuando A y B forman una dependencia funcional completa y además B es un único atributo.

Si tenemos una dependencia completa $A \Rightarrow B$ diremos que es una dependencia funcional elemental si B es un atributo, y no un conjunto de atributos.

■ Dependencia funcional trivial:

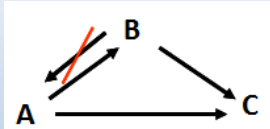
Es la dependencia que tiene un atributo con relación a otro compuesto cuando forma parte de él. Se produce cuando B es un subconjunto de A.

Una dependencia funcional $A \rightarrow B$ es trivial cuando B es parte de A. Esto ocurre cuando A es un conjunto de atributos y A es un subconjunto de B.

■ Dependencia funcional transitiva:

Es una dependencia de un atributo no principal de otro no principal.

Dados tres atributos A , B y C , se dice que existe una dependencia transitiva entre A y C , si B depende funcionalmente de A ($A \rightarrow B$) y C depende funcionalmente de B ($B \rightarrow C$), pero A no depende funcionalmente de B ($B \not\rightarrow A$), entonces ocurre que A produce una dependencia funcional transitiva sobre C .

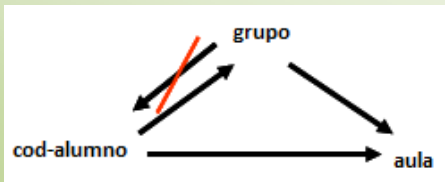


Se representa por

$A \dashrightarrow C$

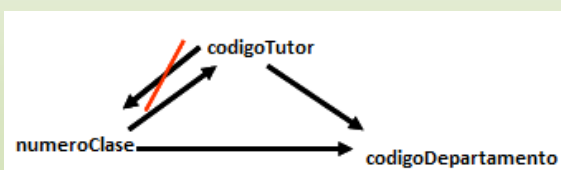
Por ejemplo

Si consideramos la siguiente relación **ALUMNO** (**cod_alumno**, grupo, aula) claramente se aprecia que los atributos **grupo** y **aula** dependen funcionalmente del atributo **cod_alumno**, y a su vez el atributo **aula** depende funcionalmente del atributo **grupo**, por lo que **aula** tiene una dependencia funcional transitiva respecto a **cod_alumno** a través de **grupo**.



Otro ejemplo:

Si A es el atributo **numeroClase** de un instituto, y B es el atributo **codigoTutor**. Entonces $A \rightarrow B$ (el tutor depende funcionalmente de **numeroClase**). Si C representa el **codigoDepartamento**, entonces $B \rightarrow C$ (el **codigoDepartamento** depende funcionalmente del **codigoTutor**, cada tutor solo puede estar en un departamento). Como ocurre que $B \not\rightarrow A$ (el **codigoClase** no depende funcionalmente del **codigoTutor**, un **codigoTutor** se puede corresponder con varios códigos de clase). Entonces $A \dashrightarrow C$ (**codigoDepartamento** depende transitivamente de **codigoClase**).



10.2.- Formas Normales.

Una vez conocidos los conceptos sobre los que se basa el proceso de normalización, se han de llevar a cabo una serie de etapas consecutivas en las que se aplicarán las propiedades de cada una de las formas normales definidas por Codd.

■ Primera Forma Normal.

Se dice que una tabla está en **Primera Forma Normal (1FN o FN1)** sí, y sólo sí, todos los atributos de la misma contienen valores atómicos o simples, es decir, no hay grupos repetitivos.

Dicho de otra forma, estará en 1FN si los atributos no clave, dependen funcionalmente de la clave.

Los valores de los atributos deben de ser valores atómicos simples del dominio. No puede haber dos valores en una misma casilla y las columnas repetidas deben de eliminarse y colocarse en otra tabla.

Por ejemplo:

<u>codigoAlumno</u>	Apellido1	Apellido2	Nombre	Edad	Asignaturas
AsirM1	Álvarez	González	Manuel	19	Redes BDatos
AsirM2	López	Ruiz	María	20	Redes BDatos Sis
AsirM3	Gómez	González	Eva	19	Redes

Esta tabla no está en 1FN pues tiene la columna asignatura con más de un valor por fila.

Para resolver este problema primero se descomponen aquellas filas en las que los atributos tienen más de un valor en tantas filas como valores haya, cada una con un valor.

Nos quedaría la siguiente relación.

<u>codigoAlumno</u>	Apellido1	Apellido2	Nombre	Edad	Asignaturas
AsirM1	Álvarez	González	Manuel	19	Redes
AsirM1	Álvarez	González	Manuel	19	BDatos
AsirM2	López	Ruiz	María	20	Redes
AsirM2	López	Ruiz	María	20	BDatos
AsirM2	López	Ruiz	María	20	Sis
AsirM3	Gómez	González	Eva	19	Redes

Para normalizar esta tabla hay que dar los siguientes pasos:

- ✓ Se crea, a partir de la tabla inicial, una nueva tabla cuyos atributos son los que presentan dependencia funcional de la clave primaria.

ALUMNOS				
<u>codigoAlumno</u>	Apellido1	Apellido2	Nombre	Edad
AsirM1	Álvarez	González	Manuel	19
AsirM2	López	Ruiz	María	20
AsirM3	Gómez	González	Eva	19

La clave de esta tabla será la misma clave primaria de la tabla inicial. Esta tabla ya estará en 1FN.

- ✓ Con los atributos restantes que no dependen funcionalmente de la clave, una nueva tabla y se eliminarán de la antigua.

La clave primaria de la nueva entidad estará formada por la concatenación de uno o varios de sus atributos y la clave primaria de la antigua entidad.

ASIGNATURAS	
<u>codigoAlumno</u>	<u>Asignaturas</u>
AsirM1	Redes
AsirM1	BDatos
AsirM2	Redes
AsirM2	BDatos
AsirM2	Sis
AsirM3	Redes

Comprobaremos si esta segunda tabla está en 1FN. Si es así, la tabla inicial ya estará normalizada a 1FN y el proceso termina. Si no está en 1FN, tomaremos la segunda tabla como tabla inicial y repetiremos el proceso.

■ Segunda Forma Normal.

Una tabla está en **Segunda Forma Normal (2FN o FN2)** sí, y sólo sí, está en 1FN y, además, todos los atributos que no forman parte de la clave principal tienen dependencia funcional de la clave completa y no de parte de ella.

Es obvio que una tabla que esté en 1FN y cuya clave esté compuesta por un único atributo, estará en 2FN.

Por Ejemplo:

Partimos de esta tabla que está en 1FN

FACTURAS-1							
<u>nuFactura</u>	<u>linea</u>	referencia	cantidad	codCliente	fecha	cif	nombre
008976	1	786543	12	654321	21-9-15	11223344B	Luis Suarez
008976	2	564322	48	654321	21-9-15	11223344B	Luis Suarez
008985	1	456789	100	123456	23-9-15	1234567J	Juan Pérez
008985	2	564322	36	123456	23-9-15	1234567J	Juan Pérez
008985	3	556677	50	123456	23-9-15	1234567J	Juan Pérez

Para normalizar esta tabla hay que dar los siguientes pasos:

- ✓ Se crea, a partir de la tabla inicial, una nueva tabla con los atributos que dependen funcionalmente de forma completa de la clave. La clave de esta tabla será la misma clave primaria de la tabla inicial. Esta tabla ya estará en 2FN.

LINEAS-FACTURAS			
<u>nuFactura</u>	<u>linea</u>	referencia	cantidad
008976	1	786543	12
008976	2	564322	48
008985	1	456789	100
008985	2	564322	36
008985	3	556677	50

- ✓ Con los atributos restantes, se crea otra tabla que tendrá por clave el subconjunto de atributos de la clave inicial de los que dependen de forma completa.

FACTURAS-2				
<u>nuFactura</u>	<u>codCliente</u>	fecha	cif	nombre
008976	654321	21-9-15	11223344B	Luis Suarez
008985	123456	23-9-15	1234567J	Juan Pérez

Se comprueba si esta tabla está en 2FN. Si es así, la tabla inicial ya está normalizada y el proceso termina. Si no está en 2FN, tomamos esta segunda tabla como tabla inicial y repetiremos el proceso.

■ Tercera forma Normal.

Una tabla está en **Tercera Forma Normal (3FN o FN3)** sí, y sólo sí, está en 2FN y, además, cada atributo que no está en la clave primaria no depende transitivamente de la clave primaria.

Un atributo **NO CLAVE**, no debe ni puede depender de otro atributo no clave de su tabla. En otras palabras, si cada uno de los atributos de la entidad dependen solo de la clave primaria.

Por Ejemplo:

TRABAJADOR			
<u>NSS</u>	nombre	puesto	salario
1111	Juan Pérez	Jefe de Área	3000
2222	José Sánchez	Administrativo	1500
3333	Ana Díaz	Administrativo	1500

Vemos que el salario depende del **puesto** (que no es campo clave) y no depende del NSS que es la clave.

Para normalizar esta tabla hay que dar los siguientes pasos:

- ✓ Se crea, a partir de la tabla inicial, una nueva tabla con los atributos que no poseen dependencias transitivas de la clave primaria.

TRABAJADOR		
<u>NSS</u>	nombre	puesto
1111	Juan Pérez	Jefe de Área
2222	José Sánchez	Administrativo
3333	Ana Díaz	Administrativo

- ✓ Con los atributos restantes, se crea otra tabla (Es decir con los atributos no clave que intervienen en la dependencia transitiva), y se elige uno de ellos como clave primaria, si cumple los requisitos para ello.

CATEGORIA	
puesto	salario
Jefe de Área	3000
Administrativo	1500
Administrativo	1500

Se comprueba si esta tabla está en 3FN. Si es así, la tabla inicial ya está normalizada y el proceso termina. Si no está en 3FN, tomamos esta segunda tabla como tabla inicial y repetiremos el proceso.

■ Forma Normal de Boyce Codd.

Una tabla está en **Forma Normal de Boyce-Codd (FNBC o BCFN)** sí, y sólo sí, está en 3FN y todo determinante es una clave candidata. Un determinante será todo atributo simple o compuesto del que depende funcionalmente de forma completa algún otro atributo de la tabla.

Aquellas tablas en la que todos sus atributos forman parte de la clave primaria, estarán en FNBC. Por tanto, si encontramos un determinante que no es clave candidata, la tabla no estará en FNBC. Esta redundancia suele ocurrir por una mala elección de la clave.

Dicho de otra forma:

Una tabla está en FNBC si cualquier atributo solo facilita información sobre claves candidatas, y no sobre atributos que no formen parte de ninguna clave candidata.

La FNBC es un caso especial de la 3FN, por tanto en la mayoría de los casos al aplicar la 3FN se cumplen los requerimientos de FNBC, pero no siempre.

Una tabla está en 3FN y no está en FNBC cuando un atributo que no es clave determina a un atributo de la clave.

Si una tabla tiene una sola clave candidata la 3FN y la FNBC son equivalentes.

Si la clave primaria está formada por un solo atributo y está en 3FN, ya está en FNBC.

Por Ejemplo:

ORGANIZACION				
<u>DNI-Trabajador</u>	Nombre-Trabajador	Cod Depart	Nombre-Depart	Responsable
123456A	Alex	111	Producción	Carmen
234567B	Arturo	111	Producción	Martin
345678C	Carlos	222	Ventas	Julio
345678C	Carlos	111	Producción	Carmen
456789D	Ana	111	Producción	Iñigo
567890E	María	222	Ventas	Silvia
567890E	María	111	Producción	Martin
678901F	Eva	222	Ventas	Julio
789012G	José	222	Ventas	Silvia

La cuestión es que un trabajador puede trabajar en varios departamentos. En dicho departamento hay varios responsables, pero cada trabajador solo tiene asignado uno.

El detalle importante que no se ha tenido en cuenta, es que el o la responsable solo puede ser responsable de un departamento.

Este detalle último produce una dependencia funcional ya que: Responsable → Departamento.

Por lo tanto hemos encontrado un determinante que no es clave candidata. No está por tanto en FNBC. En este caso la redundancia del departamento es completamente evitable.

La solución sería:

PERSONAL		
<u>DNI-Trabaj</u>	Nombre-Trabaj	Responsable
123456A	Alex	Carmen
234567B	Arturo	Martin
345678C	Carlos	Julio
345678C	Carlos	Carmen
456789D	Ana	Iñigo
567890E	María	Silvia
567890E	María	Martin
678901F	Eva	Julio
789012G	José	Silvia

RESPONSABLE		
<u>Cod Depart</u>	Nombre-Depart	<u>Responsable</u>
111	Producción	Carmen
111	Producción	Martin
222	Ventas	Julio
111	Producción	Iñigo
222	Ventas	Silvia

En las formas de Boyce-Codd hay que tener cuidado al descomponer ya que se podría perder información por una mala descomposición.

RESUMEN

1ª Forma:

Se dice que una entidad está en 1ª Forma Normal (1FN) si no contiene grupos repetitivos, es decir, los atributos dependen funcionalmente de la clave.

2ª Forma:

Se dice que una entidad está en 2ª Forma Normal (2FN) si está en 1FN y cada atributo que no pertenezca a la clave tiene una dependencia funcional completa de la clave, es decir si cada uno de los atributos de la entidad depende de toda la clave.

3ª Forma:

Se dice que una entidad está en 3ª Forma Normal (3FN) si está en 2FN y cada atributo que no pertenezca a la clave no depende transitivamente de dicha clave, es decir si cada uno de los atributos de la entidad dependen solo de la clave.

■ Otras Formas Normales.

Existen también la **Cuarta Forma Normal (4FN o FN4)**, **Quinta Forma Normal (5FN o FN5)** y **Forma Normal de Dominio-Clave (DKFN)**, aunque se ha recomendado normalizar hasta 3FN o FNBC. La 4FN se basa en el concepto de Dependencias Multivaluadas, la 5FN en las Dependencias de Join o de reunión y la DKFN en las restricciones impuestas sobre los dominios y las claves.

11.- Desnormalización.

Cada vez que avanzamos en el nivel de normalización se necesitan más uniones entre las entidades y eso ralentiza la respuesta del sistema. Como la respuesta rápida a las demandas del usuario debe tenerse en cuenta a la hora del diseño de una base de datos, a veces es necesario **desnormalizar** alguna parte de la misma.

Desnormalizar es transformar una base de datos en un nivel de normalización inferior. No obstante es necesario tener en cuenta que a cambio de un desempeño más rápido deberemos soportar una mayor redundancia de datos.

Los diseñadores de una base de datos deben conciliar 3 requerimientos a menudo incompatibles entre sí:

- Un diseño apropiado
- Velocidad de procesamiento
- Almacenamiento de la información, controlando las redundancias.

En ocasiones es necesario aceptar una cierta redundancia para que la base de datos cumpla mejor con el objetivo de mostrar la información adecuadamente.

La combinación de normalización y modelado E/R produce un modelo relacional con estructuras de tabla apropiadas

GLOSARIO

Abstracción

Separación mental de las cualidades de una cosa y de su realidad física para considerarlas aisladamente. Representación artística de personas o cosas de manera abstracta, tomando sus características esenciales y no su forma real.

Recursividad

Característica de repetirse indefinidamente que presentan algunos hechos.

Redundancia

Repetición innecesaria o inútil de un concepto o elemento.

Retrodiseño

Cuando se aplica el enfoque de datos relacional al esquema conceptual obtenido en la fase de análisis, es probable que los procesos que se lleven a cabo modifiquen la equivalencia entre entidades y tablas. Por tanto, el retrodiseño implicará la modificación del esquema conceptual para mantener dicha equivalencia, y la vuelta a un paso anterior en la secuencia de análisis de la base de datos.

Unario

Interviene un único elemento. Asociado a uno y no a varios.

Unívocamente

Que solamente tiene un significado o una interpretación posible.