**Diseño lógico de Bases de Datos.**

**Caso práctico**

En la unidad anterior habíamos visto como Alejandra, contratada por sus amigos para informatizar su taller mecánico talleres FABER, había estudiado todas las opciones antes de decidirse por implantar un S.G.B.D.  
Antes de elegir uno de los SGBD que existen en el mercado tiene claro que debe diseñar un modelo flexible que dé respuesta a las necesidades actuales de gestión del taller, pero tiene que considerar también las posibilidades de crecimiento a medio y largo plazo.  
Alejandra explica a sus amigos que este proceso requiere un diseño lógico, previo al diseño físico y a la implantación definitiva, y que debe hacerse teniendo en cuenta todas las especificaciones, características e información sobre el funcionamiento del taller para que pueda dar respuesta a las necesidades actuales y futuras.  
En definitiva vamos a ayudar a Alejandra a construir un MODELO DE DATOS que represente la realidad de TALLERES FABER, aprenderemos con ella a realizar el diseño lógico para, en el tema siguiente, referirnos al diseño físico.

El modelado de datos es el primer paso para diseñar las bases de datos, y sirve de puente entre el mundo real y el modelo de base de datos que reside en el ordenador. Desempeña un papel fundamental ya que reduce las complejidades del mundo real y las convierte en abstracciones más fáciles de entender.

Un modelo de datos es la representación relativamente simple, generalmente gráfica, de estructuras de datos complejas del mundo real. Con relación a las bases de datos un modelo representa estructuras de datos y sus características, relaciones, restricciones y transformaciones.

**Un buen diseño de la base de datos es el fundamento de buenas aplicaciones, y un buen diseño se inicia con la construcción de un buen modelo**

**Modelos de datos.**

**Caso práctico**

Para empezar a diseñar nuestra base de datos empezaremos por elegir un modelo que nos permita representar la realidad de nuestro taller: los elementos que intervienen y las relaciones entre ellos. Como hemos visto existen distintos modelos tanto a nivel lógico como conceptual. A continuación veremos cuál es la elección más apropiada.

En el capítulo anterior hemos visto que una base de datos almacena datos mientras que el SGBD administra y controla el acceso a los datos y a la base de datos. Al estar el SGBD entre la aplicación y la base de datos se logra eliminar la mayoría de las limitaciones del sistema a cambio de tener una estructura física más compleja, sobre todo en **los modelos jerárquicos y en red**, que llegaron a tener unas estructuras físicas tan complicadas que restaban importancia al diseño eficaz de la base de datos.

**El modelo de base de datos relacional cambia esto, permitiendo al diseñador centrarse en la representación lógica de los datos y sus relaciones, en vez de ocuparnos sólo del almacenamiento físico.**

**Por ejemplo: Refiriéndonos a un ejemplo de automóviles es como si la base de datos relacional utilizara transmisión automática y nos libera de tener que utilizar la palanca de cambios mientras pisamos el embrague.**

Aunque el modelo relacional supuso una considerable mejora sobre los modelos anteriores carecía de las características necesarias para ser una herramienta eficaz para diseñar bases de datos. Es más fácil examinar las estructuras gráficamente que describirlas como texto, por tanto se buscaba un modelo de datos conceptual de carácter gráfico.

**El modelo E/R es la herramienta gráfica más aceptada y que mejor se adapta a las bases de datos relacionales.**

Este modelo no tiene en cuenta la implementación física de los datos, solo interesa el nivel conceptual, en cuanto a su aplicación práctica; el diseño de la base de datos se hace empleando este modelo y para implementarlo en un SGBD se traduce al modelo relacional como modelo de diseño a nivel lógico.

Veamos el ejemplo del modelo E/R y modelo relacional correspondiente a una representación de la relación que existe entre los propietarios que traen a sus animales a una clínica veterinaria.

|  |
| --- |
| **Relación veterinarios** |
| **Modelo ER - veterinarios** |
| **Modelo Relacional Veterinarios** |

**Esta representación gráfica fue la que popularizó el uso de los diagramas E/R como herramienta de modelado de datos a nivel conceptual y complementó los conceptos de modelo de datos relacional, con lo que se establecieron las bases de un diseño de bases de datos bien estructuradas que garanticen el diseño apropiado de datos relacionales.**

**Autoevaluación**

Rellena en el párrafo siguiente los espacios en blanco con las palabras que consideres apropiadas:

Para diseñar una base de datos bien estructurada actualmente se aplica el modelo  como modelo de datos a nivel  . Una vez obtenida la representación gráfica, se traduce al modelo  que es el modelo que implementan la mayoría de los SGBD como modelo a nivel  .

**Modelo Entidad-Relación.**

**Caso práctico**

En adelante veremos algunos objetos que intervienen en la actividad cotidiana de talleres Faber y de los cuales nos interesa guardar información:

* los clientes del taller.
* los vehículos que los clientes nos traen a reparar.
* la orden de reparación que se rellena antes de enviar el vehículo al taller.
* las facturas que se emiten con posterioridad a la reparación, etc.

Como veremos en esta unidad cada uno estos elementos se recogerán en nuestro modelo como una entidad y estudiaremos las relaciones que se dan entre estos objetos.

El modelo E/R lo introdujo Peter Chen en 1976 y produjo una representación gráfica de entidades y sus relaciones en una herramienta de modelado de datos a nivel conceptual.

Es un modelo muy extendido y potente para la representación de los datos. Se simboliza haciendo uso de grafos y de tablas.

**Se pretende 'visualizar' los objetos que pertenecen a la Base de Datos como entidades las cuales tienen unos atributos y se vinculan mediante relaciones.**

Las **características** del modelo E/R son:

* Abstracción de alto nivel respecto del mundo real, creando unos elementos (entidades) que representan la realidad.
* Los registros, conjunto de datos que contiene cada entidad, su nivel de abstracción y de detalle son independientes de su uso posterior, de las limitaciones de almacenamiento y de la velocidad de proceso y del sistema en el que se vaya a implementar la base de datos.
* Esa independencia física del soporte de almacenamiento permite que el número de entidades pueda crecer y modificarse.

Las exigencias o restricciones de este modelo son:

* La existencia de la clave primaria.
* La obligatoriedad de que las entidades estén asociadas mediante una relación y la imposibilidad de asociar dos relaciones entre sí.

**Ventajas**:

* Simplicidad conceptual: los diseños de bases de datos complejas se crean y se manejan con mucha más facilidad que con cualquier otro sistema.
* Representación visual: permite a los diseñadores, programadores y usuarios finales una representación de los datos y sus relaciones fácil de entender.
* Herramientas de comunicación efectivas: permite que el diseñador integrar las visualizaciones de los datos que tienen los distintos usuarios.
* Está muy bien integrado con el modelo relacional.

**Desventajas:**

* No puede representar algunas restricciones. Por ejemplo puede reflejar fácilmente que un profesor puede impartir entre 1 y 4 clases, pero no puede recoger que un profesor no pueda impartir más de 3 clases seguidas. Estas restricciones las tiene que manejar la aplicación.
* La representación de relaciones es limitada, recoge las relaciones entre las entidades pero no recoge, por ejemplo, las relaciones que puedan existir entre los atributos.
* En el modelo ER no hay comandos de manipulación de datos.
* Cuando se representan todos los atributos el modelo se congestiona, por tanto los diseñadores tienden a simplificarlo y con ello se pierde información.

**Para saber más**

Una de las cosas más importantes a la hora de establecer el modelo E/R que mejor se adapta a una situación del mundo real es establecer correctamente cada uno de sus elementos: las entidades, sus atributos, las relaciones que existen entre las entidades, la clave principal, las restricciones, etc.   
Aunque no existe un método propiamente dicho podemos seguir una serie de pasos que nos ayudarán a establecer nuestro diagrama E/R.

[Descargar Pasos para establecer un modelo E/R[pdf - 0.12 Mb]](http://fpadistancia.caib.es/pluginfile.php/167058/mod_resource/content/1/GBD02/GBD02_Contenidos/PasosER.pdf)

**Autoevaluación**

Sobre el modelo Entidad/Relación, selecciona las opciones que consideres correctas:

|  |  |
| --- | --- |
|  | En este modelo cada objeto del que queremos guardar información será una entidad y las asociaciones entre esos objetos serán las relaciones. |
|  |  |
|  | Cuando en una entidad no exista una clave primaria elegiremos uno cualquiera de sus atributos. |
|  |  |
|  | Se utiliza en una fase del diseño anterior al modelo relacional porque ambos se integran con facilidad. |
|  |  |
|  | Está muy aceptado, tanto por la facilidad de comprensión que ofrece su representación visual, como por los comandos para la manipulación de los datos. |
|  |  |
|  | Antes de elegir el modelo E/R para nuestra base de datos necesitamos saber en qué sistema lo vamos a implementar y cuál será el soporte de almacenamiento. |
|  |  |

**Entidades.**

Es una persona, lugar o cosa sobre la cual se tienen que reunir y guardar datos. Una entidad se representa por medio de un rectángulo con el nombre dentro del recuadro.

Por ejemplo la entidad CLIENTE.

Existen dos clases de entidades:

* **Fuertes**: son aquéllas que tienen existencia por sí mismas.   
  **Por ejemplo**: La entidad CLIENTE no depende de otras entidades para existir.
* **Débiles**: son aquéllas cuya existencia depende de otra u otros tipos de entidad. Una entidad puede ser fuerte o débil respecto de otras.   
  **Por ejemplo**: La entidad FACTURA será débil respecto a la entidad CLIENTE porque no existen facturas que no correspondan a un cliente, pero será fuerte respecto a la entidad LINEA refiriéndonos a las líneas de una factura.

Las entidades débiles se representan con un doble rectángulo:

**Reflexiona**

Cada uno de los elementos que incluye la entidad se denomina ocurrencia de entidad o instancia de entidad.  
Un conjunto de entidades semejantes se conoce como conjunto de entidades. Por ejemplo el conjunto de los 5 empleados del taller mecánico constituyen el conjunto de entidades CLIENTE.  
En realidad se debería hablar de conjunto de entidades, pero en la práctica se utiliza la palabra entidad para referirnos al conjunto de entidades aunque no sea muy correcto, por tanto hablaremos de la entidad CLIENTE en lugar del conjunto de entidades CLIENTE.  
Del mismo modo el nombre de la entidad se debería escribir en mayúsculas y en singular, pero es frecuente utilizar el plural en muchos ejemplos

**Autoevaluación**

Solo una de las siguientes opciones muestra una entidad débil respecto a otra. Marca la opción que consideres acertada.

|  |  |
| --- | --- |
|  | Relación PROVEEDORES suministran ARTICULOS |
|  | Relación ALUMNOS obtienen CALIFICACIONES |
|  | La relación ALUMNOS se matriculan en ASIGNATURAS |

**Atributos.**

Cada entidad se describe como un conjunto de atributos. Un atributo describe una característica en particular de la entidad, es decir cada una de las propiedades que posee la entidad de la que se desea guardar información.  
**Por ejemplo**: La entidad CLIENTES tendrá atributos como: Código de Cliente, DNI, Apellidos, Nombre, Dirección, Teléfono.

Los atributos se representan mediante una elipse horizontal con el nombre en su interior, unidos por una línea a la entidad a la que pertenecen. (En algunos casos se representa también el dominio.)

**Atributos compuestos y simples**: Un atributo compuesto es un atributo que se puede subdividir en otros. Un atributo simple no se puede subdividir.  
Un **ejemplo**claro es el atributo dirección que puede dividirse en: calle, número, localidad, provincia y código postal.

La forma de representar estos atributos varía en función de los autores y en muchos casos no se representan en el modelo E/R. Un ejemplo de representación gráfica sería:

**Atributos de un solo valor y atributos de valores múltiples o multivaluados:**

1. Atributos de un solo valor son los que pueden tener un único valor,   
**Por ejemplo**: El DNI, el número de la seguridad social, etc. Eso no significa que sea un atributo simple, por ejemplo el número de habitación de un hotel se puede dividir: el primer dígito es la planta, los dos siguientes corresponden a la habitación dentro de la planta.

2. Atributos de valores múltiples son aquellos que pueden tener muchos valores.   
**Por ejemplo**: El número de teléfono de un cliente (puede tener uno o más números fijos, varios móviles, etc.), el color de un coche a veces puede tomar distintos valores: color de la carrocería, de las molduras y del techo, etc.

Ejemplo de representación de un atributo de valores múltiples:

A menudo nos podemos encontrar con esta otra forma de representar atributos múltiples:

Aunque en el modelo E/R se pueden manejar atributos con valores múltiples, esto no pueden ejecutarse en un SGBD relacional por tanto habrá que adoptar una de estas soluciones:

* Crear atributos nuevos para cada uno de los componentes de atributo múltiple
* Crear una entidad nueva que contenga los componentes del atributo de valores múltiples.

**Atributo derivado**: Un atributo derivado es aquel que se puede deducir de otro/s atributos/s mediante un algoritmo. Puesto que se puede calcular, no se guarda en la base de datos.   
**Por ejemplo**: La edad de una persona, que se puede calcular a partir de la fecha de nacimiento y la fecha del sistema. El importe de un pedido que se puede obtener multiplicando precio de cada unidad \* unidades pedidas.

**Dominio de un atributo**: Es el conjunto de los posibles valores que ese atributo puede poseer.   
**Por ejemplo**: El dominio del atributo Luis se toma del dominio Nombre por ser un uno de los nombres posibles para los clientes. El dominio para el atributo nota media de un estudiante sería (0,10) porque el valor más bajo posible es el 0 y el mas 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.

**Claves.**

Se denomina clave o clave primaria al conjunto de atributos que identifican de forma única a cada ejemplo de entidad.  
Al menos hay una clave, la formada por todos los atributos.

**Por ejemplo**: En la entidad CLIENTES podríamos encontrar las siguientes claves: él código de cliente, el conjunto apellidos + nombre, el DNI, el nombre, apellidos y la dirección, etc.

Clave candidata: Es cada una de las claves que está formada por el mínimo número de campos posibles.

En el **ejemplo**anterior serían el DNI y el Código de cliente.

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.

El atributo o conjunto de atributos que formen esta clave se representan subrayados. Lo ideal es que la clave primaria esté formada por un solo atributo. A cada una de las claves candidatas no seleccionadas se le denomina clave alternativa.

En el **ejemplo**anterior podría ser el Código de cliente.

Clave ajena foránea (foreign key): Es el atributo o conjunto de atributos de una entidad que forman la clave primaria en otra entidad. Se utilizan para representar las relaciones entre las tablas.

Es indiferente que los atributos que formen la clave ajena tengan distinto nombre, lo que interesa es que su contenido pertenezca al mismo dominio que la clave principal, aunque es preferible que tengan el mismo nombre para facilitar el trabajo.

**Por ejemplo**: La entidad CLIENTES con los atributos que habíamos expuesto: CodCliente (primary key), DNI, Apellidos, Nombre, Dirección y Teléfono; podría relacionarse con la entidad VEHÍCULOS cuyos atributos podrían ser: Matrícula (primary key), Marca, Modelo, Color y FechaMatriculación. El CodCliente de la entidad VEHÍCULOS sería una clave ajena (foreign key), porque es la clave principal en la entidad CLIENTES.

Clave artificial: es un atributo creado por el equipo de diseño de la base de datos, cuyo contenido es aleatorio o secuencial, no repetitivo. Al ser un atributo único, sirve para crear claves primarias más sencillas que una formada por la unión de varios atributos. En su creación se ha de tener en cuenta que ha de ser única e invariable. Suelen tener nombres como: código, identificador, número de....

**Por ejemplo**: Queremos recoger los datos de nuestra agenda de teléfonos: Nombre, apellidos y teléfono. La clave principal podría estar formada por el conjunto de atributos: Apellidos y Nombre (podemos tener amigos con nombres que coincidan o hermanos cuyos apellidos sean iguales). Como se trata de una clave compleja en la que puede haber confusiones añadimos una clave artificial denominada CodigoAgenda que podría ser un número consecutivo.

**Ejercicio resuelto**

Queremos informatizar una pequeña clínica veterinaria de nuestro barrio donde los vecinos traen a sus mascotas. Cuando un vecino acude a la consulta por primera vez, se recoge la siguiente información:

* Del propietario de la mascota: DNI, Apellidos, Nombre, Dirección, Teléfono. Hay que tener en cuenta que algunos vecinos llevan animales que están abandonados y no tienen, por tanto, ningún propietario.
* Del animal: Nombre, Raza, Fecha de nacimiento, Peso, Altura, Vacunas, y otros datos recogidos como Descripción. (Por ejemplo una mancha en la pata, etc.)

Tanto la primera vez como cuando el propietario lleva de nuevo a su mascota para una consulta se necesita guardar:

* Fecha de la consulta, motivo de la consulta, peso, diagnóstico y tratamiento

Se pide:

* Obtener una lista con las entidades
* Colocar los atributos de cada entidad
* Elegir las claves principales

**Relaciones.**

Una relación es la asociación de dos o más entidades. Generalmente se las identifica con un verbo (activo o pasivo) y se representan mediante un rombo. Las relaciones siempre operan en los dos sentidos

**Por ejemplo**: La relación entre CLIENTES y VEHICULOS se definen en dos direcciones:

* Un CLIENTE es propietario de un VEHICULO
* Cada VEHICULO es propiedad de un CLIENTE

**Entidad asociada**: Normalmente las relaciones no tienen atributos, pero puede ocurrir que los tenga. 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.

Estas entidades darán lugar a una tabla que contenga esos atributos cuando se conviertan al modelo relacional.

**Grado de una relación**: Se define como el número de entidades que participan en una relación.

* Cuando solo participa una entidad se denominan de grado uno o relaciones reflexivas.
* Cuando participan dos entidades en una relación se denominan binarias o de grado dos.
* Si participan 3 entidades se denominan de grado 3 o ternarias.
* Cuando participan más de 3 entidades se denominan n-arias.

Las relaciones pueden tener cualquier grado pero cuando se pueda se intentará que sean binarias.

Ejemplos de relaciones de grado 1, grado 2 y grado 3:

**Reflexiona**

Lo mismo que en el caso de las entidades hablamos de relación cuando tendríamos que hablar de **Conjunto de relaciones**

**Correspondencia y cardinalidad.**

En el modelo E-R se representan ciertas restricciones a las que deben ajustarse los datos contenidos en una base de datos.

Tipo de correspondencia: expresa el número de entidades a las que puede asociarse otra entidad mediante una relación. Los tipos de correspondencia para relaciones binarias son:

* Relaciones **1:1**
* Relaciones **1:N**
* Relaciones **M:N**

A esta clasificación se la denomina también **conectividad**.

**Relaciones 1:1** (uno a uno): a cada elemento de la primera entidad le corresponde sólo uno de la segunda entidad, y a la inversa.

**Por ejemplo**: Cada empleado ocupa un puesto de trabajo y cada puesto de trabajo es ocupado por un solo empleado.

**Relaciones 1: N**(uno a muchos): a cada elemento de la primera entidad le corresponde uno o más elementos de la segunda entidad, y a cada elemento de la segunda entidad le corresponde uno sólo de la primera entidad.

**Por ejemplo**: Un proveedor suministra muchas piezas. Cada pieza solo nos la suministra un único proveedor.

**Relaciones N: M** (muchos a muchos): a cada elemento de la primera entidad le corresponde uno o más elementos de la segunda entidad, y a cada elemento de la segunda entidad le corresponden uno o más elementos de la primera.

**Por ejemplo**: Cada mecánico puede intervenir varias reparaciones y una misma reparación la llevan a cabo varios mecánicos.

**Cardinalidad de una relación**: Expresa el número de ocurrencias de una entidad asociadas con una ocurrencia de la entidad relacionada. Se coloca al lado de las entidades y se representa con el formato:

**(1,n)**: el primer valor representa el valor mínimo y el segundo valor representa el máximo

**Por ejemplo**: En la relación CLIENTES traen VEHICULOS respecto a los clientes que traen a reparar sus vehículos a nuestro taller, las cardinalidades serían:

* Un cliente trae a reparar como mínimo un vehículo y como máximo varios.
* Cada vehículo es traído al taller por un cliente como mínimo y como máximo.

Las cardinalidades son muy útiles a nivel de aplicación.

**Participación**:

El concepto de participación está relacionado con las cardinalidades mínimas.

* La participación de cualquier entidad en una relación puede ser parcial o total. Si no es posible que una entidad exista a no ser que participe en la asociación, la participación es total; en caso contrario, la participación es parcial.
* Para definir el grado de participación se utilizan también los términos opcional u obligatoria. Cuando un PROFESOR necesita impartir al menos una CLASE, la entidad CLASE es **obligatoria**, si un profesor puede estar registrado sin necesidad de impartir alguna CLASE, la entidad CLASE es**opcional.**

**Por ejemplo**: En la relación anterior si cada cliente que tenemos recogido en nuestra base de datos nos ha traído al taller al menos un vehículo, en ese caso la participación de la entidad CLIENTES en la relación es total. En el caso de que pudiéramos tener registrado algún cliente que no hubiera traído ningún coche al taller (clientes potenciales) la participación serial **parcial**.

Participación total u obligatoria: cardinalidad mínima 1  
Participación parcial u opcional: cardinalidad mínima 0

**Relaciones fuertes y débiles.**

**Relaciones fuertes y débiles (identificadoras o no identificadoras):**

En el punto anterior hemos clasificado las entidades en entidades débiles y entidades fuertes, en función de que dependan o no de otra entidad para existir. Dado que las relaciones unen entidades, la existencia o no de dependencia entre estas afecta también a las relaciones.

La existencia de entidades fuertes y débiles da origen a relaciones en las que se asocian ambos tipos de entidades, de modo que una entidad débil dependa de otra fuerte para existir como las facturas y los clientes, ya que no hay facturas si no hay clientes. Un tipo diferente de dependencia de origen se da cuando la entidad débil es un subconjunto de la entidad fuerte, como en el caso de las entidades cliente y persona

La dependencia entre una entidad débil y otra fuerte puede ser de dos tipos:

**Dependencia en existencia: (EX)** Se dice que hay dependencia en existencia cuando las ocurrencias de la entidad débil no pueden existir si desaparece la ocurrencia de la entidad fuerte. En este tipo la entidad débil puede ser identificada sin necesidad de identificar la entidad fuerte de la que depende

**Cuando un empleado de nuestra empresa se da de baja será necesario dar de baja a todos sus familiares porque esa información ya no tendría sentido.**

**Dependencia en identificación: (ID)**Cuando además de la condición anterior, la entidad débil no se puede identificar únicamente mediante los atributos propios de la misma, es necesario incluir la clave de la entidad fuerte de la que depende.

**Para identificar un ejemplar tomamos la identificación del libro y le añadimos la referente a ese ejemplar para diferenciarla de otros ejemplares del mismo libro.**

**Relaciones débiles (relación no identificativa):**

Si una entidad es independiente de la existencia de otra entidad, la relación entre ellas se describe como una relación débil o como una relación no identificativa.

Una relación débil existe cuando la clave primaria de la entidad relacionada (hijo) no contiene la clave primaria de la entidad padre.

**Relaciones fuertes (relación identificativa)**

Una relación fuerte se da cuando las entidades que relaciona son dependientes *en existencia* y *en identificación*; es decir que la clave principal de la entidad relacionada contiene la clave principal de la entidad padre. En estos casos la clave principal de la entidad padre es clave ajena y parte de la clave principal en la entidad hijo.

**Por ejemplo:** En la relación FACTURAS tienen LINEAS, la clave principal de la entidad LINEAS se forma con la clave de la entidad FACTURAS junto con el número de línea.

**Ejercicio resuelto**

Volviendo a la pequeña clínica veterinaria de nuestro barrio, en el apartado anterior establecimos las entidades, sus atributos y elegimos las claves principales.

En estos momentos debemos establecer las relaciones entre esas entidades utilizando la representación gráfica del modelo entidad-relación. Para ello deberás determinar:

* Las relaciones que asocian a las entidades PROPIETARIOS, ANIMALES y CONSULTAS
* Las cardinalidades mínimas y máximas con las que participa cada entidad en la relación.
* Los tipos de correspondencia de las relaciones.

Se pide la representación gráfica del modelo entidad-relación, así como la respuesta a las siguientes cuestiones:

1. ¿La participación de cada entidad es total o parcial? ¿Hay alguna entidad que participe en la relación de forma parcial?
2. Explica si se trata de relaciones fuertes o débiles.

**Modelo Entidad-Relación ampliado.**

**Caso práctico**

Para presentar los siguientes conceptos vamos a partir de un ejemplo de entidad que se encuentra en la mayoría de empresas: la entidad empleados. En la mayoría de empresas existen distintos tipos de empleados, cuya descripción es difícil de recoger utilizando una misma entidad.

Por ejemplo en un centro de enseñanza nos podemos encontrar con profesores, personal de administración, personal de limpieza y mantenimiento, etc. De los profesores, por ejemplo, tendremos que recoger información que no se requiere para el resto de personal. Por ejemplo: el número de registro personal, la especialidad, el cuerpo al que pertenece, la fecha de alta en el cuerpo, etc.

Si recogemos a todo el personal en la misma entidad los que no sean profesores tendrían muchos valores nulos en esos atributos innecesarios.

Por otra parte existen muchos atributos comunes a todo el personal como: apellidos, nombre, DNI, dirección, fecha de contratación, etc.

Esto es lo que se pretende resolver empleando jerarquías de generalización.

Originalmente, el modelo entidad-relación sólo incluía los conceptos de entidad, relación, dominio y atributo. Más tarde, se añadieron otros conceptos, como los atributos compuestos y las jerarquías de generalización, en lo que se ha denominado modelo entidad-relación extendido, o ampliado.

Algunos aspectos correspondientes al modelo ampliado se han visto en los apartados anteriores vamos a referirnos pues a las jerarquías de generalización

**Generalización y jerarquías de generalización.**

La generalización es una técnica de abstracción que permite extraer de un conjunto de entidades una serie de atributos comunes y una serie de atributos específicos, de forma que los atributos comunes describen el **supertipo** y los atributos específicos los **subtipos*.***

Una de las características más importantes de las jerarquías es la **herencia** por la que los subtipos heredan los atributos del **supertipo**.

De la misma forma si un supertipo participa en una relación, sus subtipos también.

La jerarquía de generalización recoge la relación entre entidades del tipo padre-hijo o supertipo-subtipo.

**El supertipo es la entidad de mayor nivel y contiene los atributos comunes.**

**El subtipo es la entidad de menor nivel que contiene los atributos específicos.**

**Tipos de jerarquías.**

La generalización puede ser:

* **Total*:*** cuando todas las ocurrencias del supertipo pertenecen a alguno de los subtipos.   
  En nuestro **ejemplo** es total si todos los empleados del centro son o profesores o administrativos o de mantenimiento. Todos los tipos de empleado están incluidos en la clasificación.
* **Parcial**: Cuando puede haber ocurrencias en el supertipo que no pertenezcan a ninguno de los subtipos.

**Por ejemplo:** Si en nuestro centro existen empleados que no sean ni profesores, ni administrativos ni de mantenimiento.

* **Exclusiva**: Cuando una ocurrencia del supertipo no puede estar a la vez incluida en más de un subtipo.

**Por ejemplo:** que un profesor no puede ser a la vez administrativo o de mantenimiento y viceversa.

* **Solapada**: Cuando una ocurrencia del supertipo puede estar a la vez en varios subtipos.

**Por ejemplo:** Si un empleado puede ser a la vez administrativo y de mantenimiento.

Las cardinalidades de las jerarquías son:

* **(1,1)** en el supertipo
* **(0,1)** en los subtipos, para las exclusivas
* **(1,1) o (0,1)** en los subtipos, para las solapadas

**Ejemplos de representación gráfica de los distintos tipos de jerarquía**

**Ejercicio resuelto**

Queremos recoger la información correspondiente a las aulas de un instituto de secundaria con la especialidad de informática:

* De cada aula se quiere recoger la siguiente información: NºAula, Piso, Pasillo, Nº plazas. El número de aula es único y distinto para todas ellas.
* Cuando las aulas son salas de ordenadores necesitamos saber: nº de ordenadores, escáner y nº de impresoras, así como otros equipamientos que pudieran instalarse.
* Si se trata de laboratorios queremos saber el tipo (de ciencias, de idioma, etc.) y el equipamiento que tiene.
* Todas las aulas, sean comunes, laboratorios o salas de ordenadores, pueden tener o no proyector y pizarra interactiva.
* Existen otros tipos de aulas no recogidos en el modelo anterior como gimnasio, biblioteca, etc.
* Cada aula se utiliza únicamente para las funciones que tiene asignadas (no se imparte una clase normal en un laboratorio, etc.)

Construye el diagrama E/R del enunciado anterior.

**Represenatación gráfica del modelo Entidad-Relación.**

**Caso práctico**

En este punto hemos repasado con Alejandra todos los conceptos necesarios para elaborar correctamente un modelo ER. Con esto podríamos llevar a cabo la primera fase de nuestro objetivo: obtener un modelo conceptual flexible e independiente.

A continuación veremos que, con el paso del tiempo, se han ido desarrollando para este modelo distintas simbologías con la intención de adaptarse a las herramientas de diseño basadas en la utilización del ordenador, que paralelamente han ido surgiendo.

Afortunadamente encontramos en el mercado una gran variedad de herramientas, tanto de software libre como propietario, que nos van a ayudar a diseñar este modelo, corregir los posibles errores y en muchos casos nos facilitarán la generación del código (herramientas CASE).

El modelo de datos E/R, a pesar de tener algunas desventajas, como herramienta de modelado cada vez está más extendida. De hecho los vendedores han agregado tantas extensiones a la presentación básica que sigue siendo la herramienta de diseño de bases de datos más utilizada.

En los ejemplos desarrollados a lo largo de la unidad hemos utilizado los símbolos del modelo Chen, sin embargo se han desarrollado otros estilos que se adaptan mejor a las herramientas de modelado de bases de datos basados en el ordenador.

Los más conocidos son:

* Diagrama entidad-relación de Chen: es el que hemos visto hasta ahora.
* Diagrama de pata de gallo
* Rein85
* IDEF1X

**El modelo de Chen**, tal y como lo hemos estudiado en este tema, está basado en los trabajos de Chen (modelo básico) y en las mejoras incorporadas por Teorey, Yang y Fry (modelo ampliado). Ha sido el modelo en que se han basados las herramientas CASE dominante desde finales de los 80. Aunque actualmente no sea el modelo dominante como generador de modelos ER, todas las herramientas actuales tienen su origen en él.

**El modelo pata de gallo**, desarrollado por Bachman, es una herramienta de modelado muy extendida por usar una notación fácil de entender. Si en el modelo Chen utiliza la designación 1: N para el tipo de correspondencia entre relaciones y (0,1), (1,1), (0,n), (1,n) para las cardinalidades, el modelo pata de gallo reúne esta información en un solo símbolo. Sin embargo este modelo no puede representar cardinalidades distintas de 0, 1 o n como sí puede hacerse en el modelo Chen (por ejemplo (3,30) para designar que a una clase deben asistir como mínimo 3 y como máximo 30 alumnos). Algunas herramientas comerciales que utilizan este modelo evitan este problema añadiendo las estas cardinalidades en el diagrama usando texto y definiéndolas en el directorio de datos.

**El modelo Rein85**fue desarrollado por Reiner en 1985. Basado en las mismas convenciones que el de pata de gallo, sin embargo sus símbolos son diferentes. Este modelo no reconoce cardinalidades, únicamente trata de conectividad. Su nivel de abstracción es mayor.

**El modelo IDEF1X** fue desarrollado a finales de los 70 por la fuerza aérea de EE.UU mediante el programa ICAM (fabricación asistida por ordenador). Se trata de un programa que pretendía aumentar la productividad de la fabricación aplicando tecnología informática. Se desarrollaron unos estudios que obtuvieron métodos gráficos para definir funciones, estructuras de datos y dinámicas de empresas de manufacturas. Se convirtió en una herramienta de modelado de datos de manufactura en general, aunque utiliza menos símbolos que otros modelos y por tanto proporciona menos detalles.

**Para saber más**

Puedes ver una comparación de los distintos símbolos utilizados por cada uno de los modelos citados en el siguiente enlace:

[Descargar comparativa de símbolos del modelo E-R](http://fpadistancia.caib.es/pluginfile.php/167058/mod_resource/content/1/GBD02/GBD02_Contenidos/Comparativa_ER.pdf) (0.08 Mb)

**Herramientas para construir diagramas E/R.**

Actualmente hay muchas herramientas para construir diagramas E/R en el mercado; a lo largo de este curso necesitarás familiarizarte con alguna de ellas. A continuación te mostramos algunos ejemplos:

**Recomendación**

A continuación te mostramos diferentes sitios web donde puedes descargarte distinto software para construir diagramas E/R

* **Software libre**: Herramientas que puedan interpretar y generar modelos ER, SQL y hacer análisis de base de datos MySQL son: Workbench y DBDesigner (código abierto).

[Acceder a la página de descarga de Workbench](http://dev.mysql.com/downloads/workbench/5.0.html)

[Acceder a DBdesigner](http://www.fabforce.net/dbdesigner4/index.php)

Algunas herramientas de software libre se utilizan para generar únicamente el diagrama, sin tener conocimiento de lo que significan, ni generan SQL. Estos incluyen Kivio (incluida en la suite KDE Koffice ) y Dia. (los diagramas de DIA, sin embargo, puede ser traducido con tedia2sql)

[Acceder a la página de Dia](http://live.gnome.org/Dia)

* **Freeware**: Una herramienta que puede generar la base de datos y la capa de código de la aplicación es el Editor de RISE.

[Acceder a la página del editor de RISE](http://www.freedownloadmanager.org/es/downloads/Editor_RISE_55134_p/)

* **Software propietario**: Algunos de los diagramas ER de propiedad son:

herramientas ARIS, dbForge Studio para MySQL, DeZign para bases de datos, ER/Studio, Devgems Data Modeler, ERwin, Oracle Designer, PowerDesigner, Rational Rose, Enterprise Architect de Sparx, SQLyog, Toad Data Modeler, Maestro SQL, Microsoft Visio, Analista visible, y paradigma visual.

[Acceder a la página de Toad Data Modeler](http://www.quest.com/toad-data-modeler/)

[Acceder a la págian de Microsoft Visio](http://technet.microsoft.com/es-es/evalcenter/ee390821.aspx)

**Para saber más**

Aunque las notaciones más extendidas son la de Chen y Pata de gallo, existen herramientas de modelado en el mercado que utilizan otras simbologías. Si quieres ampliar esta información puedes visitar los siguientes enlaces:

[Acceder a la comparativa de todos los modelos](http://en.wikipedia.org/wiki/Entity-relationship_model)

[Acceder a la definición en wikipedia del Modelo IDEF1X](http://en.wikipedia.org/wiki/IDEF1X)

[Acceder a un ejemplo de una base de datos de casas de colonias](http://www.dataprix.com/214-ejemplo-base-datos-casas-colonias)

Sobre el modelo pata de gallo (o pata de cuervo, según otros autores) existe más documentación porque es un modelo muy utilizado por las herramientas de diseño de bases de datos o herramientas CASE.

Algunos ejemplos sobre el modelo pata de gallo son:

[Acceder a ejemplos sobre el modelo de pata de gallo 1](http://www2.cs.uregina.ca/~bernatja/crowsfoot.html)

[Acceder a ejemplos sobre el modelo de pata de gallo 2](http://www.scribd.com/doc/19078534/DISEODEBASESDEDATOSCAPITULOXIII)

**Ejercicio resuelto**

Tenemos la siguiente información:

* Un cliente puede tener muchos vehículos.
* Cada vehículo es de un cliente.
* La participación de la entidad VEHICULOS es opcional.
* Puede haber clientes registrados que no tengan ningún vehículo.

Construir el diagrama Entidad - Relación utilizando los modelos vistos anteriormente

**Modelo Relacional.**

**Caso práctico**

Veremos ahora como se transforma un modelo Entidad-Relación en un Modelo Relacional más próximo al ordenador.

E.F.Codd es el padre del modelo relacional, que definió en un artículo publicado en 1970. Este modelo está basado en dos teorías matemáticas: la teoría de conjuntos y la lógica de predicados de primer orden, que nos proporcionan los elementos necesarios para crear una base de datos relacional; pero no es necesario aprender estas teorías para poder utilizar el modelo relacional.  
Este modelo persigue, al igual que la mayoría de los modelos de datos, los siguientes **objetivos**:

* Independencia física de los datos: el modo de almacenamiento de los datos no debe influir en su manipulación lógica.
* Independencia lógica de los datos: los cambios que se realicen en los objetos de la base de datos no deben repercutir en los programas y usuarios que accedan a ella.
* Flexibilidad: presentar los datos a los usuarios de la forma más adecuada a la aplicación que utilicen.
* Uniformidad: presentación de las estructuras lógicas en forma de tablas, fáciles de comprender y manipular por los usuarios.
* Sencillez: las características anteriores, junto con los lenguajes de usuario hacen que este modelo sea fácil de entender y utilizar por el usuario.

Para Codd los datos se estructuran en forma de relaciones; entendiendo la relación como un elemento de la teoría matemática de las relaciones y que se corresponde con la idea de tabla o entidad. Los conceptos básicos de este modelo son:

* Interrelación es la asociación entre dos tablas.
* El conjunto de columnas que representan las propiedades de la tabla y que se denominan atributos.
* El conjunto de filas que se denominan tuplas y que contienen los valores que toman cada uno de los atributos para cada elemento de la relación.

Para Codd lo importante es el diseño lógico, un modelo abstracto, por lo que no indica nada acerca de la implementación de este modelo en SGBD comerciales, sin embargo los trabajos de Codd y otros investigadores dieron lugar a diversas bases de datos comerciales, DB2 de IBM, Oracle, etc.; y es el modelo lógico en el que se basan la mayoría de los S.G.B.D. existentes actualmente en el mercado.

Un SGBD sólo necesita que el usuario pueda percibir la base de datos como un conjunto de tablas. Esta percepción sólo se aplica a la estructura lógica de la base de datos (en el nivel externo y conceptual de la arquitectura de tres niveles ANSI-SPARC). No se aplica a la estructura física de la base de datos, que se puede implementar con distintas estructuras de almacenamiento.

El modelo relacional, como todo modelo de datos tiene una parte estática constituida por los objetos permitidos y las restricciones, y una parte dinámica o conjunto de operaciones definidas sobre la estructura, que permite la transición entre estados de la base de datos. La parte dinámica está constituida por elálgebra relacional y el cálculo relacional

**Codd logró un nuevo concepto con una nueva terminología, el modelo relacional, cuyo eje son las relaciones que se dan entre los atributos. Son los atributos los que dan origen a las tablas ya que existen relaciones entre ellos. Las tablas forman bases de datos.**

**Reflexiona**

Suele haber confusión entre la terminología que utilizan las bases de datos y los sistemas de archivos. En ocasiones encontraremos que se hace referencia a las filas como registros, a las columnas como campos y a las tablas como archivos. La diferencia está en que la tabla en una base de datos es un concepto lógico, mientras que los conceptos archivo, registro y campo son conceptos físicos.

**Autoevaluación**

En el siguiente párrafo rellena los espacios en blanco con las palabras que consideres acertadas:

En el modelo relacional, el modelo el almacenamiento de los datos no influye en su manipulación. A esto se le denomina independencia .   
En este modelo los cambios en los objetos no repercuten en los programas. A esto se denomina independencia  .   
Para el usuario la base de datos es un conjunto de tablas constituidas por columnas o  y por filas o  .

**Estructura del modelo relacional.**

Las relaciones se utilizan para almacenar información sobre los objetos que se representan en la base de datos.

Se representa gráficamente como una tabla bidimensional en la que las filas corresponden a registros individuales y las columnas corresponden a campos o atributos de esos registros.

Se han introducido algunos **elementos que intervienen en el modelo relacional**. De forma más detallada son:

* Una **relación**es una tabla con columnas y filas. El usuario percibe la base de datos como un conjunto de tablas. Una tabla sirve para almacenar los datos de una entidad.
* Un **dominio**es el conjunto de valores que puede tomar un atributo. Cada atributo se define sobre un dominio. Se trata de un conjunto finito de valores homogéneos y atómicos.  
  Los valores de cada columna pertenecen a un dominio que se define previamente. Todos los dominios tienen un nombre y un tipo de datos asociado. Existen dos tipos de dominios:
  + **Dominios generales**: son aquéllos cuyos valores están comprendidos entre un máximo y un mínimo. Ejemplo el código postal formado por todos los números positivos de cinco cifras.
  + **Dominios restringidos**: son los que pertenecen a un conjunto de valores específico. Ejemplo el sexo que puede tomar los valores H o M.
* El concepto de dominio es muy importante a la hora de definir operaciones que pueden tener sentido o no (por ejemplo no tiene sentido comparar una ciudad con un número de teléfono), pero su implementación en un SGBD es muy compleja.
* Un **atributo**es el nombre de una columna de una relación. En otros términos un atributo es el papel o rol que desempeña un dominio en una relación. Representa el uso del dominio para una determinada relación. Aporta un significado semántico a un dominio. Ejemplo: atributo nombre definido sobre el dominio: conjunto de 15 caracteres.
* Una **tupla**es cada fila de una relación (o tabla). El orden de las tuplas no es relevante.
* El **grado**de una relación es el número de atributos que contiene, es decir el número de columnas de la tabla.
* La **cardinalidad**de una relación es el número de tuplas que contiene.
* El **valor**es la intersección entre una fila y una columna.

Veamos el siguiente ejemplo: Relación EMPLEADOS

**Características de una relación.**

La visualización desde un punto de vista lógico de una base de datos relacional se corresponde con un conjunto de tablas relacionadas.  
Para el usuario una relación o tabla es un conjunto de entidades (simplificando, sustituimos el término conjunto de entidades por el de entidad)  
Una relación es una tabla que tiene una serie de características:

* Una tabla se percibe como una estructura bidimensional compuesta de filas y columnas
* Cada tabla o relación tiene un nombre que la diferencia de las demás.
* Cada fila de la tabla se denomina tupla y representa la ocurrencia de una entidad.
* No admiten filas duplicadas
* El orden de las filas no es significativo.
* Cada columna representa un atributo, y cada columna tiene un nombre distinto.
* El orden de las columnas es irrelevante. No están ordenadas.
* La tabla es plana, es decir, en la intersección fila/columna solo puede haber un valor, no se admiten atributos multivaluados.
* Cada tabla debe tener un atributo o conjunto de atributos que identifique a cada fila de forma única.
* Cada columna tiene un mismo tipo de datos y un intervalo de valores específico: dominio de un atributo.

Tipos de relaciones:

En un SGBD relacional pueden existir varios tipos de relaciones, aunque no todos manejan todos los tipos. Unas se almacenan en la base de datos y otras son resultados de consultas.

* **Relaciones base**. Son relaciones reales que tienen nombre y forman parte directa de la base de datos almacenada (son autónomas).
* **Vistas**. También denominadas relaciones virtuales, son relaciones con nombre y derivadas a partir de una consulta: se representan mediante la definición de la consulta en términos de otras relaciones con nombre, no poseen datos almacenados propios.
* **Instantáneas**. Son relaciones con nombre y derivadas. Pero a diferencia de las vistas, son reales, no virtuales: están representadas no sólo por su definición en términos de otras relaciones con nombre, sino también por sus propios datos almacenados. Son relaciones de sólo de lectura y se refrescan periódicamente por el sistema.
* **Resultados de consultas**. Son las relaciones resultantes de alguna consulta especificada. Pueden o no tener nombre y no persisten en la base de datos.
* **Resultados intermedios**. Son las relaciones que contienen los resultados de las subconsultas. Normalmente no tienen nombre y tampoco persisten en la base de datos.
* **Resultados temporales**. Son relaciones con nombre, similares a las relaciones base o a las instantáneas, pero la diferencia es que se destruyen automáticamente en algún momento apropiado.

ESQUEMA DE UNA BASE DE DATOS RELACIONAL

**Claves primarias y ajenas.**

Cuando un atributo o combinación de atributos de una relación poseen un conjunto de valores que identifican únicamente a cada elemento de la relación, se denomina clave primaria.

La clave primaria identifica inequívocamente a cada fila de una tabla. Sus características son:

* Es única y mínima (a veces la fila se puede identificar por un único atributo, pero otras veces es necesario recurrir a más de uno).
* Existe y es conocida y pública
* Es NO NULA
* No es ambigua
* No se permiten claves primarias redundantes (repetidas). (no se puede descartar ningún atributo de la clave para identificar la fila)

Cuando una clave se compone de más de un atributo se denomina clave compuesta.  
  
Existen tablas con uno o más atributos o grupos de atributos que poseen características de clave primaria. A ese atributo o grupo de atributos que pueden originar una clave primaria de su tabla se les denomina claves candidatas.

Características de las claves candidatas:

* Al menos hay una por tabla y está formada por el conjunto de todos los atributos (puesto que no puede haber dos filas iguales)
* Ha de tener el mínimo número posible de atributos. Entre uno y el total de campos de la tabla
* No han de poseer campos superfluos
* La clave candidata seleccionada se convierte en clave primaria. El resto de claves candidatas se denominan claves alternativas.

Cuando un atributo o combinación de atributos de una relación poseen un conjunto de valores que no forman la clave primaria en la tabla, pero son el conjunto de atributos que forman la clave primaria en otra tabla se les denomina clave ajena

Las claves ajenas, al compartir atributos, permiten relacionar tablas, por lo que facilitan la relación entre las mismas, para originar nuevas tablas o vistas.

Características de las claves ajenas:

* Su dominio ha de ser el mismo que el dominio de la clave primaria de la tabla a la que haga referencia.
* Se permite que el nombre de los atributos sea diferente de la clave ajena y que el nombre del atributo de la clave primaria y de la tabla asociada sean diferentes.
* Su valor puede estar duplicado o ser nulo
* Los atributos que la forman pueden formar parte de la clave primaria de la tabla a la que pertenecen.
* Una tabla que no se relacione con otras no tiene claves ajenas.

**Autoevaluación**

Rellenar los espacios en blanco con la palabra correcta teniendo en cuenta el siguiente esquema:

**Tabla EMPLEADOS**:

CLAVES CANDIDATAS: IDEMPLEADO,    
CLAVE PRINCIPAL:    
CLAVES ALTERNATIVAS:    
CLAVE AJENA: 

**Tabla SECCIONES**:

CLAVES CANDIDATAS: IDSECCION,    
CLAVE PRINCIPAL:    
CLAVE ALTERNATIVA: 

**Restricciones.**

En todos los modelos de datos existen restricciones que deben tenerse en cuenta a la hora de diseñar una base de datos. Los datos almacenados en la base de datos deben cumplir con una serie de reglas para garantizar que sean correctos. El modelo relacional, como todo modelo de datos, también presenta ocurrencias no permitidas. Los tipos de restricciones pueden ser:

1. **Restricciones inherentes al modelo**: Son restricciones propias del modelo e indican las características propias de una relación, por tanto han de cumplirse obligatoriamente.
   * Ausencia de tuplas repetidas.
   * Irrelevancia del orden de las tuplas.
   * Irrelevancia del orden de los atributos.
   * Cada atributo solo puede tomar un único valor del dominio al que pertenece.

Algunas de ellas se exigen desde el modelo matemático como:

1. Otras restricciones inherentes son:
   * La **restricción de clave primaria (PRIMARY KEY)**, permite declarar uno o varios atributos como clave primaria de una relación.   
     En nuestro **ejemplo**la clave principal de la relación CLIENTES sería el CodCliente.
   * **Integridad de clave**: es una restricción que exige que todos los atributos de la clave primaria (PRIMARY KEY), han de contener valores no nulos (NOT NULL).   
     En el **ejemplo**de la relación FACTURAS- LINEAS, la clave principal de la tabla LINEAS es una clave compuesta de los campos NumFactura+NumLinea. Esta restricción implica que ambos atributos deben contener valores que no sean nulos.
   * **Integridad referencial (FOREIGN KEY)**: Consiste en que no puede haber un valor en una clava ajena de una tabla, si antes no existe en la tabla de la que ese campo o conjunto de campos formen la clave primaria.   
     Para explicar este concepto denominamos tabla hija a la tabla que contiene la clave ajena y tabla padre a la tabla que contiene la clave principal.  
     Se permiten valores nulos en las claves ajenas; es decir que una clave ajena debe coincidir con un valor de clave primaria de la relación a la que apunta o tener valor nulo.   
     Por **ejemplo**: Para relacionar un vehículo con su propietario establecemos una clave ajena (CodCliente) en la tabla VEHÍCULOS que coincide con el CodCliente correspondiente en la tabla CLIENTES. La integridad referencial establece que no podemos introducir ningún código de cliente (clave ajena) en la tabla (VEHICULOS) si no existe previamente ese código en la tabla CLIENTES (clave principal).
2. **Restricciones de usuario**: Son impuestas para cada problema concreto, desde la definición de los dominios de un campo a condiciones impuestas a un campo de acuerdo con el valor de otros. Para establecer estas condiciones disponemos de las siguientes restricciones:
   * **Restricciones de verificación: (CHECK)** comprueban en una actualización si se cumplen las condiciones exigidas en la restricción o no, antes de ejecutarla.   
     Por **ejemplo**: Que la edad de un trabajador sea mayor o igual a 16
   * **Restricción de unicidad: (UNIQUE)**permite definir claves alternativas. Los valores de los atributos no pueden repetirse.

Por **ejemplo**: En la tabla CLIENTES la clave principal seleccionada es el CodCliente, pero el atributo DNI, que es clave alternativa en dicha tabla, tampoco admite valores duplicados. Para ello establecemos una restricción de tipo UNIQUE para este campo.

* + **Restricción de obligatoriedad: (NOT NULL)** permite declarar si uno o varios atributos pueden tomar valores nulos. Por definición una clave principal no puede contener valores nulos en ninguno de los atributos que la componen (integridad de clave) por tanto se definirán como NOT NULL.   
    Por **ejemplo**: La restricción NOT NULL puede ser necesaria también con otro tipo de atributos como el DNI citado en el ejemplo anterior, el NOMBRE, etc.
  + Los **disparadores (TRIGGER)**son acciones, métodos según la terminología orientada a objetos, que ejecuta un objeto, como respuesta a un evento o acción que se realiza sobre él, cuando se cumple una determinada condición.   
    Por **ejemplo**si tenemos dos tablas:  
    una con los datos de los vehículos de nuestros clientes (matrícula y resto de datos que identifican al vehículo) y otra que recoge cada una de las entradas en el taller de esos vehículos (matrícula, fecha y hora), podemos crear un trigger que cada vez que se inserte un vehículo en la tabla de vehículos añada una fila en la tabla entradas en taller con los datos anteriores.

**Restricciones y operaciones relacionales.**

Además de definir las claves ajenas hay que tener en cuenta las operaciones de borrado y actualización que se realizan sobre las filas de la tabla relacionada.   
A la hora de hacer operaciones de inserción, actualización o borrado aparecen problemas, ya que hay que tener en cuenta los tres tipos de restricciones establecidas, sobre todo las restricciones de clave primaria e integridad de clave.

Respecto a la **actualización de campos** hay que tener en cuenta:

* No pueden modificarse los atributos que forman parte de la clave primaria para dar origen a registros con valores de clave primaria redundantes.
* La modificación de atributos que forman la clave principal no puede dar lugar a que aparezcan en esos atributos valores nulos.
* Para aceptar los cambios en un atributo, se han de cumplir las restricciones de clave, referenciales y de usuario que hayan sido definidas para ese atributo en la tabla a la que pertenezca.
* Si se modifica el valor de un atributo que forme parte de la clave primaria, habrá que sustituir los valores de los atributos que forman claves ajenas, en los que aparece el valor antes de modificar el contenido del atributo, por su nuevo valor. Es lo que se denomina actualización en cascada.

Respecto de la de **inserción de campos**:

* No pueden añadirse registros si alguno de los atributos que forman parte de la clave primaria posee un valor nulo.
* No pueden añadirse registros si el conjunto de valores de los atributos que forman parte de la clave primaria origina una clave primaria redundante
* Como consecuencia de la actualización en cascada, para añadir un valor a un atributo que forma parte de la clave ajena, ese valor ha de existir en la clave primaria a la que hace referencia la clave ajena.
* Para aceptar los cambios en un atributo se han de cumplir las restricciones de clave, referenciales y de usuario que hayan sido definidas para ese atributo en la tabla a la que pertenezca.

Respecto del **borrado de campos**:

Al borrar una fila que tenga asociados, mediante la clave primaria, otras filas en otras tablas, se pierde la consistencia de la tabla. Para solucionarlo existen dos posibilidades:

* Borrado en cascada: consiste en borrar todas las filas de las tablas en los que aparezca como clave ajena el conjunto de atributos que forman la clave primaria del registro a borrar.
* Para borrar un registro cuya clave primaria es clave ajena en otras tablas, primero hay que borrar los registros con la clave primaria asociada, para después borrar el registro deseado.

Teniendo en cuenta las consideraciones anteriores las posibilidades son las siguientes:

* Borrado y/o modificación en cascada (**CASCADE**). El borrado o modificación de una fila en la relación padre (relación con la clave primaria) ocasiona un borrado o modificación de las filas relacionadas en la relación hija (relación que contiene la clave ajena).
* Borrado y/o modificación restringido (**RESTRICT**). En este caso no es posible realizar el borrado o la modificación de las filas de la relación padre si existen filas relacionadas en la relación hija.
* Borrado y/o modificación con puesta a nulos (**SET NULL**). Esta restricción permite poner la clave ajena en la tabla referenciada a NULL, si se produce el borrado o modificación en la tabla primaria o padre.
* Borrado y/o modificación con puesta a valor por defecto (**SET DEFAULT**). El valor que se pone en las claves ajenas de la tabla referenciada es un valor por defecto que se habrá especificado en la creación de la tabla.

**Paso del diagrama Entidad-Relación al modelo relacional.**

**Caso práctico**

Una vez conseguido el esquema conceptual de datos mediante el modelo E/R, para su implementación es necesario transformarlo a esquema lógico mediante el modelo relacional.   
Ayudaremos a Alejandra en este proceso aplicando una serie de reglas de transformación que se describen a continuación.

La transformación se realiza empleando las siguientes reglas:

* Toda **entidad**se transforma en una tabla.
* Todo **atributo**se transforma en columna dentro de la tabla.
* El identificador único de la entidad se convierte en **clave primaria.**
* Como las **relaciones**del modelo E/R no tienen equivalente en el modelo relacional, ya que sólo existen tablas y operaciones entre ellas, es necesario aplicar lo siguiente:
  + En las **relaciones M:N** 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. Esta tabla posee dos claves ajenas, una por cada entidad con la que está relacionada.
  + En las **relaciones 1:N** la entidad del lado N de la relación añade el conjunto de campos necesarios para incorporar a sus atributos la totalidad de la clave primaria de la entidad del lado 1, creando una clave ajena, de modo que se puedan relacionar ambas tablas mediante operadores relacionales. El nombre de la relación desaparece.
  + Las **relaciones 1:1** se transforman en función de las cardinalidades:
    - propagando cualquiera de los atributos identificadores y sus atributos asociados creando una única tabla con el conjunto de los atributos de ambas entidades. La clave primaria sería cualquiera de las dos. Ambas entidades participan con cardinalidades (1,1)
    - propagando la clave de cualquier tabla a la otra, según cuál sea la que tenga accesos más frecuentes. Ambas cardinalidades (1,1)
    - propagar la clave de la entidad con cardinalidad (1,1) a la entidad que tenga (0,1).
    - crear una nueva tabla a partir de la relación con las dos claves de ambas. Cuando ambas tablas tienen cardinalidades (0,1)

Veremos un ejemplo de aplicación de esas reglas:

Además de las reglas de transformación que acabamos de ver y que se aplican con carácter general, existen otros aspectos a tener en cuenta a la hora de obtener un esquema relacional a partir de un modelo entidad-relación.

**Debes conocer**

Para ampliar la información sobre las reglas de transformación pulsar el siguiente enlace:

[Ampliación reglas de transformación del modelo E/R al modelo relacional](http://fpadistancia.caib.es/pluginfile.php/167058/mod_resource/content/1/GBD02/GBD02_Contenidos/Transformacion_er.pdf) (0.25 MB)

**Ejercicio resuelto**

Partiendo del ejercicio resuelto del apartado 2.2.2 en el que se establece el modelo entidad-relación entre las entidades PROPIETARIOS, ANIMALES Y CONSULTAS.

Se pide obtener el modelo relacional correspondiente.

**Ejercicio resuelto**

Partiendo del ejercicio resuelto del apartado 2.3.1 en el que se establece el modelo entidad-relación correspondiente a la clasificación de la entidad AULA en los subtipos: COMÚN, S. ORDENADORES y LABORATORIO.

Obtener el modelo relacional correspondiente.

**Normalización.**

**Caso práctico**

Una vez que hayamos conseguido establecer un modelo lógico para nuestra base de datos que recoja toda la información necesaria sobre el funcionamiento del taller podríamos plantearnos el siguiente paso que es el diseño físico de la base de datos, pero aunque el paso del modelo entidad-relación al modelo relacional se haya hecho aplicando ciertas reglas, las tablas obtenidas pueden presentar problemas que se derivan de las restricciones que pueden existir entre atributos de distintas tablas o entre atributos de la misma tabla. La n  
A esas restricciones se les denomina dependencias.  
Así pues tendremos que dar un paso más en nuestro diseño lógico y aplicar una técnica de denominada normalización que consiste en verificar el modelo para eliminar las inconsistencias que pudiera tener.

Vamos a estudiar las dependencias existentes en los atributos para conseguir que nuestras tablas se encuentren en un nivel de normalización aceptable.

**B es funcionalmente dependiente de A**

**Dependencia funcional transitiva**

**El proceso de normalización.**

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 Un buen diseño de base de datos debe ser acorde con las buenas estructuras de datos. 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 asignar atributos a las entidades verificando que se cumplan ciertas reglas. Reduce las redundancias de datos y ayuda a eliminar las anomalías que se derivan de las redundancias.

La normalización se basa en lograr la independencia de los datos con respecto a las aplicaciones que los usan, obteniendo unas tablas con una estructura óptima y eficaz.

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 4FN, y en aplicaciones muy especializadas de investigación estadística será necesario llegar más allá.

Este proceso se basa en la descomposición sin pérdida de las tablas que están en una forma normal inferior, obteniendo una forma normal superior. Se descompone la tabla en otras con menor cantidad de atributos, sin que haya pérdida de información.

**Dependencias funcionales y formas normales.**

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.

Se denominan **dependencias** a las relaciones que existen entre los atributos en el mundo real y que son recogidas en el modelo lógico de la base de datos.

**Dependencias funcionales**: se producen cuando tenemos una tabla con una serie de atributos. Se dice que un atributo tiene dependencia funcional de otro cuando a cada valor del primero le corresponde un solo valor del segundo.

**Para una definición más formal diremos que: Si X e Y son subconjuntos de atributos de una tabla, diremos que X tiene dependencia funcional de Y (o también que X determina a Y) si cada valor de X tiene asociado siempre un único valor de Y.**

**X   -->    Y   (se lee: X determina a Y*)***

Las dependencias funcionales pueden ser:

**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.

**Es decir; en una dependencia funcional X -->   Y, cuando X es un conjunto de atributos, decimos que la dependencia funcional es completa si solo depende de X, no de ningún subconjunto de X.**

**X  → Y**

**Dependencia funcional elemental:** Una dependencia funcional es elemental si Y posee un único atributo.

**Si tenemos una dependencia completa X→Y diremos que es una dependencia funcional elemental si Y 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 Y es un subconjunto de X

**Una dependencia funcional   X → Y   es trivial cuando Y es parte de X. Esto ocurre cuando X es un conjunto de atributos y X es un subconjunto de Y.**

**Dependencia funcional transitiva*.*** Es una dependencia de un atributo no principal de otro no principal.

**Cuando se tienen tres atributos X, Y y Z se cumple que X → Y,  Y → Z  pero  no Y → X (Y no determina X). Por tanto Z tiene dependencia transitiva con respecto a X, a través de Y.**

**Dependencias multivaluadas.**Se producen cuando un atributo puede tomar múltiples valores en la misma tabla, independientemente de los valores que tomen el resto de los atributos.

Existe dependencia funcional multivaluada si dados 3 atributos de una tabla, para cada valor del primer atributo existen múltiples valores del segundo y no existen ninguna relación entre el tercer atributo y el primero, a no ser a través del segundo atributo.

**Se representa como:   X→→ Y     y se lee  X  multidetermina  Y.**

**Este tipo de atributos implica redundancia ya que el resto de los atributos se repiten tantas veces como valores diferentes tenga el atributo multivaluado.**

**Dependencias de unión.** Para comprender este tipo de dependencias es necesario explicar previamente los conceptos:

* **Proyección:**Creación de una tabla cuyos elementos forman un subconjunto de una tabla dada. Se incluyen todas las filas y algunas columnas.
* **Unión:**Formar, a partir de dos tablas, una nueva con todos los campos de una de ellas y los registros de ambas, excepto los repetidos. Ambas tablas han de tener el mismo grado y las mismas columnas.

**Se dice que hay dependencia de unión o producto si una tabla tiene dependencia de unión con varias de sus proyecciones y se puede obtener la tabla por medio de la unión de dichas proyecciones.**

**Autoevaluación**

**Partiendo de la siguiente relación, seleccionar la alternativa que refleje las dependencias existentes entre sus atributos**:

MODULOS ( **ClaveModulo**, NombreModulo, Horas, Aula, ClaveCiclo, NombreCiclo)

|  |  |
| --- | --- |
|  | ClaveMódulo → NombreMódulo, Horas, Aula, ClaveCiclo, NombreCiclo  ClaveCiclo → NombreCiclo |
|  | ClaveMódulo → NombreMódulo, Horas, Aula, ClaveCiclo, NombreCiclo |
|  | ClaveMódulo → NombreMódulo, Horas, Aula  ClaveCiclo → NombreCiclo |
|  | ClaveCiclo → NombreCiclo, ClaveMódulo, NombreMódulo, Aula, Horas  ClaveMódulo → NombreMódulo |

**1FN.**

Para que una base de datos esté en primera forma normal cada columna de una tabla debe ser atómica, es decir, que cada atributo debe contener un único valor del dominio.

En consecuencia los atributos en cada tabla de una base de datos no podrán tener una lista de valores (ni del mismo dominio ni de otro) y, por tanto, cada atributo tiene que tener un nombre único.  
Las restricciones de la primera forma normal coinciden con las condiciones de una tabla en el modelo relacional por lo tanto siempre es obligatorio aplicar la 1FN.

Para aplicar esta forma normal nos podemos encontrar con los siguientes casos:

* Un atributo compuesto,   
  Por **ejemplo**: El nombre del cliente: Juan Martínez Arce. Podemos considerar el nombre como un dato atómico o, si nos interesa, separarlo en Apellido1, Apellido2, Nombre.
* Un atributo multivaluado,  
  Por **ejemplo**: CodCliente, ....., teléfono. En el caso de que quisiéramos recoger distintos números de teléfono para cada cliente tendríamos que crear una nueva tabla que recoja los números de clientes y los CodCliente para relacionar con la anterior.

**Ejemplo**:

CLIENTES: **CodCliente**, Nombre, Dirección, CPostal, Teléfono.

De la tabla de CLIENTES de nuestro taller tomamos como ejemplo los atributos CodCliente y Nombre. Podemos determinar que a cada código de cliente le corresponde un único nombre de cliente. Por tanto hay una dependencia funcional entre ellos: el código de cliente determina el nombre de ese cliente (no al revés). Eso no quiere decir que si nos sabemos el Código de un cliente sepamos también el nombre, sino que para cada código de cliente solo puede haber un nombre.

CodCliente → Nombre

**Como vemos en la mayoría de los casos no será necesario aplicar este proceso ya que nuestras tablas ya se encontrarán en 1FN si hemos hecho un buen diseño conceptual y lógico para la base de datos.**

**Autoevaluación**

**¿Está la siguiente tabla en 1 FN? Selecciona la respuesta correcta.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CodigoAlumno** | **Apellido1** | **Apellido2** | **Nombre** | **Asignaturas** |
| Asir\_M\_01 | Álvarez | Salces | Carmen | Redes, Base de datos |
| Asir\_M\_02 | Calle | Martín | Javier | Redes, Base de datos, Sistemas Operativos |
|  |  |  |  |  |

|  |  |
| --- | --- |
|  | Sí, porque el Nombre del alumno que es un atributo compuesto se ha dividido en datos atómicos. |
|  | No, porque no hay dependencia entre la clave CodigoAlumno y las Asignaturas. |
|  | No, porque el atributo Asignaturas es un atributo multivaluado. |
|  | Sí, porque todos los atributos tienen dependencia respecto de la clave CodigoAlumno. |

**2FN.**

Una tabla está en segunda forma normal cuando está en 1FN y además todos los atributos que no forman parte de la clave

principal tienendependencia funcional de la clave completa y no de parte de ella.   
Significa, por tanto, que en una relación sólo se debe almacenar información sobre un tipo de entidad, y que se traduce en que los atributos que no aporten información directa sobre la clave principal deben almacenarse en una relación separada.

Por **ejemplo**: En la tabla PRODUCTIVIDAD recogemos los tiempos que utiliza cada empleado en realizar determinadas reparaciones con relación a los tiempos previstos, para calcular la productividad de ese empleado.

PRODUCTIVIDAD: **NumReparacion, CodEmpleado, FechaActuación** NombreEmpleado, PuestoTrabajo, SalarioBase, NumHoras, DescriReparación, TiempoPrevisto, ImporteReparación. TiempoEmpleado.

Dependencias:

**NumReparacion, CodEmpleado, FechaActuación**→TiempoEmpleado  
CodEmpleado → NombreEmpleado, PuestoTrabajo, SalarioBase  
NumReparación → DescriReparación, TiempoPrevisto, ImporteReparación

Tenemos dependencias parciales, es decir, dependencias basadas en una parte de la clave primaria que producirán redundancias al insertar nuevas filas en la tabla. Para eliminar esas dependencias crearemos nuevas tablas que recojan las dependencias:

**PRODUCTIVIDAD:** **NumReparacion, CodEmpleado FechaActuación**, TiempoEmpleado.  
**EMPLEADOS:** **CodEmpleado**, NombreEmpleado, PuestoTrabajo, SalarioBase  
**REPARACIONES:** **NumReparacion**, DescriReparación, TiempoPrevisto, ImporteReparación.

**Estas tablas están en 2FN ya que:**

* **Están en 1FN**
* **Ningún atributo depende solamente de una parte de la clave primaria.**

Aunque nuestras tablas se encuentren en 2FN, todavía presentan anomalías; ya que el Salario Base de todos los empleados que tengan el mismo puesto de trabajo coincide, por tanto, si queremos hacer algún cambio en el salario base asociado a un puesto de trabajo determinado, debemos hacer el cambio en todos los empleados que tengan ese puesto o generaremos inconsistencias.

**Cuando la clave primaria de una tabla se compone de un solo atributo automáticamente está en 2FN**

**Autoevaluación**

La siguiente relación no está en 2 FN. De las opciones planteadas selecciona la que cumpla las condiciones necesarias.

CONSULTA (CodPaciente,CodMedico, FechaConsulta, NombrePaciente, Diagnóstico, Tratamiento, Especialidad)

|  |  |
| --- | --- |
|  | CONSULTA (**CodPaciente**, NombrePaciente, Diagnóstico, Tratamiento, FechaConsulta) MEDICO (**CodMedico**, Especialidad) |
|  | CONSULTA (**CodPaciente, CodMedico, FechaConsulta**, Diagnóstico, Tratamiento) MEDICO (**CodMedico**, Especialidad) PACIENTE (**CodPaciente**, NombrePaciente) |
|  | MEDICO (**CodMedico**, Especialidad) PACIENTE (**CodPaciente**, NombrePaciente) CONSULTA (**FechaConsulta**, Diagnóstico, Tratamiento) |

**3FN.**

Consiste en eliminar toda dependencia funcional transitiva.  
Una base de datos está en 3FN si está en 2FN y no existen atributos que no pertenezcan a la clave primaria que puedan ser conocidos mediante otro atributo que no forme parte de la clave primaria. Es decir, que no existan dependencias funcionales transitivas.

Siguiendo con el **ejemplo**anterior vemos que existen dependencias transitivas:

**EMPLEADOS**: **CodEmpleado**, NombreEmpleado, PuestoTrabajo, SalarioBase

PuestoTrabajo → SalarioBase

Para eliminar esta dependencia guardamos esos atributos en una nueva tabla. Sin embargo dejamos el determinante (PuestoTrabajo) en la tabla original como clave ajena para poder relacionar ambas tablas.

De esta forma nuestro ejemplo quedaría en 3FN con las siguientes tablas:

**PRODUCTIVIDAD: NumReparacion, CodEmpleado, FechaActuación**, TiempoEmpleado.  
**REPARACIONES: NumReparacion**, DescriReparación, TiempoPrevisto, ImporteReparación.  
**EMPLEADOS: CodEmpleado**, NombreEmpleado, PuestoTrabajo  
**PUESTOS: PuestoTrabajo**, SalarioBase

**Estas tablas están en 3FN ya que:**

* **Están en 2FN**
* **No existen dependencias transitivas**

Aunque las tablas anteriores están en 3FN y no presentan dependencias parciales ni transitivas aún podemos mejorar el diseño teniendo en cuenta los siguientes aspectos:

* La clave principal de la tabla PUESTOS puede dar lugar a errores si se cometen imprecisiones al rellenar los datos. Sería conveniente añadir un CodigoPuesto que evite problemas de pérdida de integridad, pero se crearán dependencias:  
  CodPuesto → PuestoTrabajo, SalarioBase   
  PuestoTrabajo → SalarioBase
* Se trata de una dependencia transitiva ya que el PuestoTrabajo determina el SalarioBase sin ser clave principal, pero debemos asumirla para evitar problemas de integridad referencial.
* El campo NombreEmpleado de la tabla EMPLEADOS, deberíamos descomponerlo en EmpApellido1, EmpApellido2 y EmpNombre para cumplir con el requerimiento de atomicidad y facilitar algunas operaciones como listas ordenadas por apellidos, etc.
* En la tabla EMPLEADOS faltan campos como FechaContratación, CuotaSegSocial, etc.
* En la tabla PRODUCTIVIDAD se supone que un empleado solo puede introducirse una actuación por cada empleado y cada tipo de reparación. Si el empleado realiza dos actuaciones correspondientes a un mismo tipo de reparación en el mismo día no podría registrase. Es conveniente añadir un CódigoActuación que sería la clave principal y mantendríamos NumReparación y CodEmpleado como claves ajenas.

**Autoevaluación**

Rellena los espacios en blanco con las palabras correctas:

En la relación ENFERMEDADES (**IdEnfermedad**, Descripción, Síntomas, Medicamento, EfectosSecundarios), podemos decir que todos los dependen funcionalmente de la  por eso la relación está en  pero no está en  ya que existe una dependencia funcional  . El atributo  puede ser conocido a través de valor del atributo  .

**FNBC.**

Una tabla esta en forma normal de Boyce-Codd si todo determinante en ella es una clave candidata. Dicho de otra forma:  
Una tabla está en FNBC si cualquier atributo sólo 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**

Ejemplo:   
X+Y → Z, F  
Z → Y

Esta tabla estaría en 3FN porque no tiene dependencias ni parciales, ni transitivas, pero no está en FNBC.

Para corregir esto es necesario:  
• Cambiar la clave primaria de X+Y a X+Z porque Z resulta ser determinante de Y  
• Eliminar las dependencias parciales que se han creado ya que Y depende de Z:

X+Z → F  
Z → Y

Por **ejemplo**:   
Tenemos una tabla con los atributos:  
CALIFICACIONES: **CodAlumno, CodProfesor**, CodClase, Calificación

Y las siguientes dependencias:

CodAlumno + CodProfesor → CodClase, Calificación  
CodClase → CodProfesor.

El resultado para que las tablas estén en FNBC será:

CALIFICACIONES: **CodAlumno, CodClase**, , Calificación  
CLASES: **CodClase**, CodProfesor.

**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**

**Autoevaluación**

Dada la siguiente relación:  
FACTURA (**NumFactura**, FechaFactura, CodigoCliente, ImporteTotal)

¿Podemos afirmar que cumple la FNBC?. Selecciona las respuestas acertadas

|  |  |
| --- | --- |
|  | No, porque no cumple la 3FN. |
|  |  |
|  | Sí, porque cumple la 3FN y solo hay una clave candidata. |
|  |  |
|  | Sí, porque la clave principal está formada por un único atributo y está en 3FN. |
|  |  |
|  | No, porque un atributo que no es clave determina a un atributo de la clave. |
|  |  |
|  | Sí, porque ningún atributo de la clave determina a un atributo que no es clave |
|  |  |

**4FN.**

Es posible que nos encontremos con bases de datos mal diseñadas o creadas a partir de hojas de cálculo, en el que existan atributos con valores múltiples.  
Las dependencias multivaluadas son una consecuencia de la 1FN, que prohíbe que un atributo de una fila tenga un conjunto de valores.   
Si tenemos dos o más atributos multivaluados independientes en la misma tabla, tendremos que repetir todos los valores de uno de los atributos con cada valor del otro atributo para que las filas de la tabla sean consistentes.  
Para evitar esto se descompone la tabla en otras dos, manteniéndose en cada una de ellas una dependencia múltiple, siendo la clave de las nuevas tablas la combinación de todos los campos. Veamos esto con un ejemplo:

Por **ejemplo**: Consideramos la tabla MECÁNICOS supongamos que se dan las siguientes alternativas:

* Un mecánico puede trabajar en una avería con un ayudante.
* Cada mecánico puede trabajar en varias averías. En este caso el mecánico ME-001 trabaja en las averías AV-555 y AV-552
* Cada mecánico puede tener varios ayudantes. El mecánico ME-001 trabaja con los ayudantes AYU-23 y AYU-20.
* Las averías y los ayudantes del mecánico no tienen relación directa.

|  |  |  |
| --- | --- | --- |
| **MECANICO** | **AVERIA** | **MECANICO** |
| ME-001 | AV-555 | AYU-23 |
| ME-001 | AV-552 | AYU-20 |
| ME-002 | AV-551 | AYU-18 |

En este caso la clave principal estaría formada por los tres atributos.

Para evitar las múltiples redundancias y los problemas de actualización de esta tabla que tiene dependencias multivaluadas será necesario dividir la tabla en dos. La clave de cada tabla estará formada por los dos atributos.

|  |  |  |  |
| --- | --- | --- | --- |
| **MECANICO** | **AVERIA** | **MECANICO** | **AYUDANTE** |
| ME-001 | AV-555 | ME-001 | AYU-23 |
| ME-001 | AV-552 | ME-001 | AYU-20 |
| ME-002 | AV-551 | ME-002 | AYU-18 |

**Una tabla está en 4FN si está en FNBC y las únicas dependencias funcionales multivaluadas que existen son las dependencias funcionales de la clave con los atributos que no formen parte de la clave. Es decir que toda dependencia funcional multivaluada se determina por una clave candidata (se trata de dependencias triviales)**

**5FN.**

La quinta forma normal se refiere a dependencias que son extrañas, las dependencias de unión. Tiene que ver con tablas que pueden dividirse en subtablas, pero que no pueden reconstruirse.

Se emplea cuando en una misma tabla existe información redundante, con pocos atributos o cuando una tabla posee una gran cantidad de atributos y se hace por ello inmanejable.

**Se dice que una tabla está en 5FN si está en 4FN y las únicas dependencias que existen son las dependencias de unión de una tabla con sus proyecciones relacionándose entre sí mediante la clave primaria o cualquier clave candidata.**

Para conseguir que una tabla que está en 4FN esté en 5FN, se divide la tabla en tantas como sea necesario. Estas tablas tendrán en común los campos que formaban la clave principal en la tabla original.

Para conseguir que una tabla que está en 4FN que tenga gran cantidad de atributos o pocos atributos y muchos registros, esté en 5FN, se divide la tabla en tantas como sea necesario. Estas tablas tendrán en común con una o más de las otras tablas la existencia de una o más claves ajenas.

**Ejemplo1**: Tenemos una tabla EMPLEADOS que contiene la información sobre los empleados de nuestra empresa.

EMPLEADOS (CodEmpleado, DNI, Apellido1, Apellido2, Nombre, Dirección, CP, Teléfono, FechaNacimiento, EstadoCivil, NumHijos, FechaAlta, Categoría, Puesto, NSegSocial, IRPF, ....)

Se trata de una tabla con muchos atributos, lo cual ralentiza las consultas. Podemos dividir esa tabla en:

EMPLEADOS\_ Personal (CodEmpleado, DNI, Apellido1, Apellido2, Nombre, Dirección, CP, ......)  
EMPLEADOS\_Profesional (CodEmpleado, Categoría, Puesto, NSegSocial, IRPF, .....)  
EMPLEADOS\_ Familiar (CodEmpleado, EstadoCivil, NumHijos, ..)

**Ejemplo2**: Tenemos una tabla de ALQUILERES de vehículos que tiene pocos atributos con información redundante.

ALQUILERES (Matricula, CodCliente, Fecha)

Esta tabla contiene múltiples registros con los datos que se recogen a diario, de forma que si queremos consultar los vehículos alquilados por un determinado cliente la velocidad de respuesta será elevada. Sería conveniente partir esta tabla en otras:

VEHICULO-CLIENTE (Matricula, CodCliente)  
VEHICULO-FECHA (Matricula, Fecha)  
FECHA-CLIENTE (Fecha, CodCliente)

**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 desnormalizaralguna 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

**Anexo.- Licencias de recursos.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Recurso (1)** | **Datos del recurso (1)** | **Recurso (2)** | **Datos del recurso (2)** |
|  | Autoría: ElFisgon.  Licencia: CC by-nc-nd.  Procedencia: http://www.flickr.com/photos/salya\_yasal/2942806332/sizes/l/. |  | Autoría: amenish.dewatermere.  Licencia: CC by-nc-nd.  Procedencia: http://www.flickr.com/photos/amenish/4043015228/in/photostream/. |
|  | Autoría: Pattybot.  Licencia: CC by .  Procedencia: http://www.flickr.com/photos/29678921@N07/2777746597/. |  | Autoría: Edtruji.  Licencia: Licencia libre GNU, CC by-sa.  Procedencia: http://es.wikipedia.org/wiki/Archivo:DependenciaFunional.png. |
|  | Autoría: Edtruji.  Licencia: Licencia libre GNU, CC by-sa.  Procedencia: http://es.wikipedia.org/wiki/Archivo:DependenciaFunional2.png. |  | Autoría: Jonata.  Licencia: PD (public domain).  Procedencia: http://www.openclipart.org/detail/17436. |
|  | Autoría: pst.  Licencia: --.  Procedencia: Elaboración propia con openclipart http://www.openclipart.org/detail/17436. |  | Autoría: Jcaceres93.  Licencia: Dominio público.  Procedencia: http://commons.wikimedia.org/wiki/File:Conjunto\_de\_llaves\_con\_un\_llavero\_para\_una\_mochila.jpg. |
|  | Autoría: Tizio.  Licencia: Dominio público.  Procedencia: http://commons.wikimedia.org/wiki/File:Tree-decomposition-2.svg. |  | Autoría: Dake.  Licencia: CC by-sa.  Procedencia: http://commons.wikimedia.org/wiki/File:Knapsack.svg. |
|  | Autoría: pst .  Licencia: Dominio público.  Procedencia: Montaje realizado sobre imagen original en http://www.openclipart.org/detail/23507. |  | Autoría: pst .  Licencia: Dominio público.  Procedencia: Montaje realizado sobre imagen original en http://www.openclipart.org/detail/23507 . |
|  | Autoría: pst .  Licencia: Dominio público.  Procedencia: Montaje realizado sobre imagen original en http://www.openclipart.org/detail/23507 . |  | Autoría: pst .  Licencia: Dominio público.  Procedencia: Montaje realizado sobre imagen original en http://www.openclipart.org/detail/23507. |
|  | Autoría: pst .  Licencia: Dominio público.  Procedencia: Montaje realizado sobre imagen original en http://www.openclipart.org/detail/23507. |  |  |