Universidad Nacional del Litoral

FACULTAD DE INGENIERÍA
Y CIENCIAS HÍDRICAS

Ingeniería en Informática

Ingeniería de Software I

TEMA V – El modelo de conceptual

PRIMERA PARTE
EL MODELO CONCEPTUAL

Introducción

Gran parte de las labores efectuadas para un desarrollo informático, involucra el modelado del sistema que desea el usuario. En el tema anterior, se vio el modelado de las funciones y el análisis de los procesos del sistema, técnica que permite confeccionar diagramas que reflejan la esencia de cuáles son las tareas que hacen transformaciones sobre los datos en el contexto del sistema de información. En este tipo de modelo, el elemento fundamental es el constituido por los procesos, que mueven la información dentro del sistema a partir de los datos y estímulos recibidos desde fuera de las fronteras del mismo. Este tipo de metodología, permite tener una visión global bastante conveniente de los componentes funcionales del sistema, pero – como se indicara oportunamente –, sin dar detalles de ellos. Para mostrar los detalles, se recurría a dos herramientas textuales de apoyatura al modelado adicionales: el Diccionario de Datos y la Especificación de Procesos.

A pesar de que se utilicen estas herramientas de modelado adicionales y de que estos diagramas muestren la existencia de uno o más grupos de datos almacenados, deliberadamente transmiten muy poco acerca de sus detalles. Todos los sistemas almacenan y usan información acerca del ambiente en el cual interactúan; a veces, la información es mínima, pero en la mayoría de los sistemas actuales, es bastante compleja. No sólo se debe conocer en detalle qué información hay en cada almacén de datos, sino que también interesa conocer la relación que existe entre ellos. Este aspecto del sistema es tratado por el Modelo Conceptual de Datos.

5.1 Definiciones previas

5.1.1 Noción de entidad o sujeto

*Una **entidad** o sujeto es la representación de un objeto material o inmaterial con significado (cosa real o imaginaria) del universo exterior al que se le asocian atributos o propiedades que la caracterizan.*

Por ejemplo, al sujeto Cliente del universo exterior, le corresponderá en el sistema de información una entidad CLIENTE a la cual se le asociarán propiedades como Código de cliente, Apellido y Nombre, Domicilio, etc.

Gráficamente, se denotará mediante un rectángulo, indicándose en la parte superior del mismo, el nombre de la entidad y, separado por una línea, los otros objetos que la componen. Los nombres de las entidades deben ser expresados en singular.

5.1.2 Noción de relación

*Una **relación** es la consideración por el sistema de información del hecho de que existe una asociación entre objetos del universo exterior la cual tomará la forma de asociación entre las entidades correspondientes.*

Por ejemplo, entre la entidad **PEDIDO** (número, fecha), y la entidad **ARTÍCULO** (código, descripción, precio unitario), puede existir la relación ***FORMADO POR*** que expresa la asociación que existe entre el objeto **PEDIDO** y determinados objetos del tipo **ARTÍCULO**. La providencia colocada como nombre de la relación se corresponde a la construcción sintáctica que se desea lograr, y dependerá de quien es considerado como *sujeto* y quien como *objeto*. De esta manera, puede decirse que el **pedido CONTIENE artículo**, o bien **artículo *CONFORMA* pedido**. Su significación es la misma. Es común que se utilice como sujeto a la entidad de mayor jerarquía en la relación, siendo éstas normalmente las que representan hechos o eventos. En el ejemplo dado, la entidad **ARTÍCULO** no representa ningún hecho o evento, sino que es estructural. No ocurre lo mismo con la entidad **PEDIDO** que representa precisamente el hecho de haber recibido un requerimiento para adquirir o comprar artículos. De acuerdo a lo expresado, la forma más natural de armar la relación sería **PEDIDO *FORMADO POR* ARTÍCULO.**

A una relación se le pueden asignar propiedades, exactamente igual que a las entidades. La existencia de relaciones con propiedades es bastante común, sin embargo, algunos autores en sus metodologías, las consideran como entidades especiales a las que denominan **entidades débiles**. En el ejemplo considerado, la relación ***FORMADO POR*** puede ser portadora de la propiedad **cantidad** que expresa para cada **ARTÍCULO** del **PEDIDO** (contenido en el pedido), la cantidad solicitada.

La nomenclatura que inicialmente se denotará para indicar las relaciones, será una elipse con una línea horizontal en el eje mayor y se colocará en la parte superior el nombre de la relación, y en la parte inferior, los atributos o propiedades que la caracterizan.

5.1.3 Noción de propiedad

*El concepto de **propiedad** se corresponde a la noción del atributo que caracteriza a la entidad o a la relación.*

Un atributo es cualquier detalle que sirve para calificar, identificar, clasificar, cuantificar o expresar el estado de una entidad o una relación. En general, es cualquier descripción de una característica de importancia. Se corresponden con uno y solamente un tipo de dato, básicamente numérico, alfanumérico o tipo fecha.

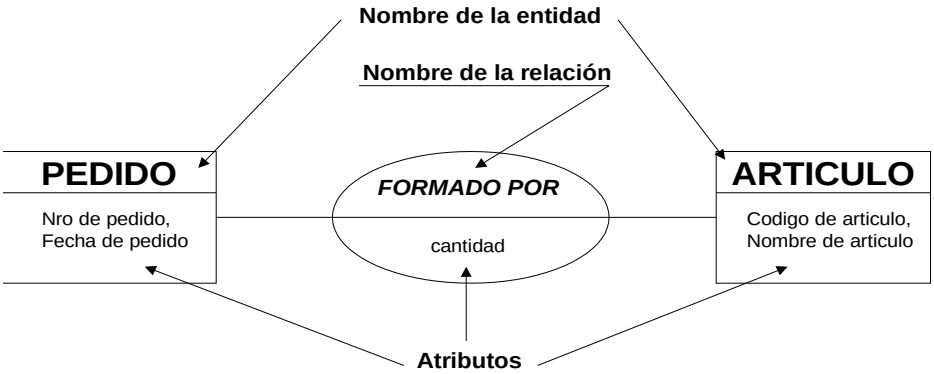
Los atributos o propiedades, pueden tener normas de validación y ciertas restricciones de formato, por ejemplo, el valor del atributo **importe de sueldo** no puede ser menor que cero, el valor del atributo **tipo de documento** puede ser 1, 2, 3, 4 ó 5 (según una norma de codificación) y no tomar ningún otro valor; el valor del atributo **fecha de entrega** de un pedido, no puede ser anterior al valor del atributo **fecha del pedido** original, etc.

5.1.4 Noción de dominio

*Un **dominio** es una abstracción que representa un conjunto de reglas de validación, restricciones en los formatos, valores posibles a tomar, tipo de dato y otras características propias de los atributos presentes en el modelo.*

Con la definición aportada, ahora un atributo, no necesariamente estará vinculado a un tipo de dato y además expresar sus limitaciones para solamente esa propiedad, sino que puede estar asociado a un **dominio**, y es éste el que tiene todas las características que deberá heredar el atributo. Por ejemplo si de define el dominio **Domicilio** y se define que será una cadena de caracteres no nula de ochenta símbolos de longitud, entonces por ejemplo **dirección postal de cliente**, **domicilio legal de cliente**, **domicilio comercial de cliente**, **domicilio de la empresa** etc. será conveniente y más sencillos que estén asociados al dominio y no tener que definir todas las limitantes para cada uno de los atributos. Además permite el mantenimiento de un formato estándar para todos los datos de características semejantes.

A manera de ejemplo, se graficará la entidad **PEDIDO** y la entidad **ARTÍCULO** vinculadas por la relación ***FORMADO POR***.



5.2 Clasificación de las entidades

5.2.1 Entidades permanentes

Las entidades permanentes son aquellas que se conservan constantemente en la base de información, pero que se pueden actualizar en cualquier momento. Corresponden a la estructura, **y** no representan hechos. Por ejemplo: **CLIENTE**.

Las propiedades de una entidad permanente pueden cambiar, pero la entidad en sí es estable. Las propiedades de una entidad permanente pueden ser de dos tipos:

- Propiedades de **filiación**:
  Son las que corresponden a su descripción. Por ejemplo: **Nombre del cliente** de una entidad **CLIENTE**.

- Propiedades de **situación:**

  Son las que expresan en que estado se encuentra la entidad permanente en un instante dado, situación que representa en general una acumulación o una posición. Por ejemplo, **stock disponible** de una entidad **ARTÍCULO**.

5.2.2 Entidades de tipo movimiento

Las entidades de tipo movimiento, son las que conservan hechos. Se trata de movimientos memorizados. Estas entidades son las imágenes de los eventos que portan estos movimientos. Están ligadas a los hechos circunstanciales. Una entidad de tipo movimiento, es el registro de un evento en el sistema de información.

Por ejemplo, para el caso de la entidad **PEDIDO** - memorizado en el sistema de información -, es el resultado del evento **PEDIDO *PASADO POR UN* CLIENTE**. Una particularidad que presentan, es que existe un momento en que ya no se puede actualizar una entidad tipo movimiento (no puede modificarse un pedido una vez que ya fue pasado).

5.3 Clasificación de las relaciones

5.3.1 Relaciones permanentes

Son relaciones entre entidades permanentes que se conservan constantemente y de las que se puede modificar sus propiedades en cualquier momento. Son relaciones estructurales. Por ejemplo dada la entidad **EMPLEADO** y la entidad **OFICINA**, la relación ***ASIGNADO A*** es de carácter permanente (es el vínculo lógico que deberá existir entre las oficinas y los empleados).

5.3.2 Relaciones de tipo movimiento

Son relaciones entre entidades permanentes o de tipo movimiento que representan el registro de un evento (relaciones coyunturales). Por ejemplo dada la entidad **PEDIDO** y la entidad **ARTÍCULO**, la relación ***FORMADO POR***, es de tipo movimiento.

5.4 Clasificación de las propiedades

La primera clasificación que establecemos es relativa a la naturaleza de la propiedad. Se dice que una propiedad es concatenada si es posible descomponerla o dividirla en otras propiedades. Una propiedad es elemental si no puede descomponerse (ha llegado al máximo grado de división y es una primitiva). Esta clasificación depende en gran medida del sistema que es objeto del estudio. Las propiedades que para uno son elementales, tal vez para otro no lo sean y admitan subdivisiones. El criterio para definir si la propiedad es elemental o no, debe ser definido por quien hace el estudio del sistema que establecerá si vale o no la pena separar o unificar conceptos.

Por ejemplo en el caso de indicar un domicilio, tal vez se coloque como que:

Dirección = Calle + Número + Piso + Departamento (caso de concatenación)

La propiedad **piso** o cualquiera de los otros términos involucrados, constituye una propiedad **elemental** (de acuerdo al criterio de quien ha indicado esto). Tal vez esto sea de utilidad pues permitirá realizar por ejemplo, una consulta de manera más sencilla que devuelva todos los datos de los clientes que viven en una calle determinada pero en un quinto piso. En este caso, la propiedad **dirección** es **concatenada**. Puede ocurrir también que el domicilio de un cliente simplemente sea considerado como un elemento más en cuyo caso no requeriría ser subdividido, razón por la cual domicilio ya es una propiedad elemental.

Otra forma de clasificar una propiedad, es por su grado de persistencia. Una propiedad cuyo valor no puede obtenerse a partir de ninguna deducción (dato primitivo), necesariamente debe ser **memorizada** en la base de información del sistema. Otras no hace falta que lo sean pues pueden obtenerse de manera indirecta y en estos casos se dice que son **deducibles** o **calculadas**.

La última forma de clasificar las propiedades, es relativa a su tipo. Se distinguen tres clases de propiedades:

- **Códigos:** Son informaciones sintéticas representativas de objetos materiales o inmateriales del universo exterior, según una ley de correspondencia rigurosa (sistema de codificación) por la que a todo objeto existente se le asocia un valor y sólo un valor del código de forma tal que a dos objetos diferentes les corresponden dos valores diferentes de código. Un código puede servir para identificar una entidad, es decir, de propiedad característica de esa entidad (clave de identificación).

- **Etiquetas:** Son datos alfanuméricos, cualitativos, simples cadenas de caracteres, que pueden ser suministrados por el sistema o sobre los que se pueden hacer clasificaciones o comparaciones, pero que no pueden participar en cálculos.

- **Cantidades:** Son datos numéricos, cuantitativos, que pueden participar en cálculos.

5.5 Tipos y ocurrencias

Un tipo, es un conjunto de elementos que tienen las mismas características (clase). Una ocurrencia de un tipo, es un elemento en particular perteneciente a dicho conjunto (objeto).

Entidad - Tipo:

Un tipo de entidad o entidad - tipo, es una clase de entidades particulares que tienen propiedades análogas.

Ocurrencia de Entidad - Tipo:

Una ocurrencia de entidad - tipo, es una entidad concreta perteneciente a este tipo.

Por ejemplo:

CLIENTE es una **entidad – tipo**.
El cliente **Juan Pérez** es una ocurrencia de esta entidad - tipo.

Un concepto especial de las entidades es el **identificativo o clave de identificación**, y lo que permite es distinguir una ocurrencia en particular, de cualquier otra perteneciente a la misma entidad – tipo. El identificativo es una propiedad o concatenación de propiedades que caracteriza cada ocurrencia de la entidad - tipo. Una entidad – tipo puede tener más de un medio alternativo de identificación única. En los gráficos, este concepto aparecerá subrayado o en negrita para diferenciarlo de los otros. De acuerdo al ejemplo anterior, se tiene que:

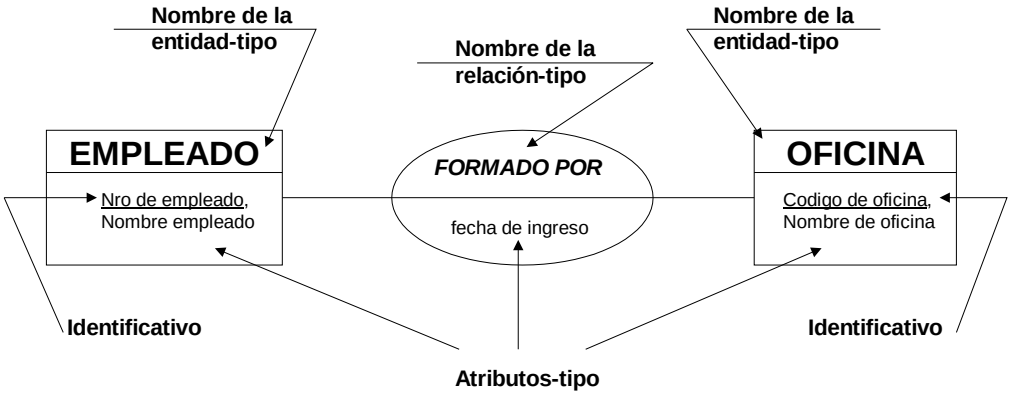
Relación-tipo:

Una relación-tipo, es una relación definida entre diversas entidades-tipo. Cada conjunto de ocurrencias de entidades que componen la relación-tipo, constituye una ocurrencia de la relación-tipo.

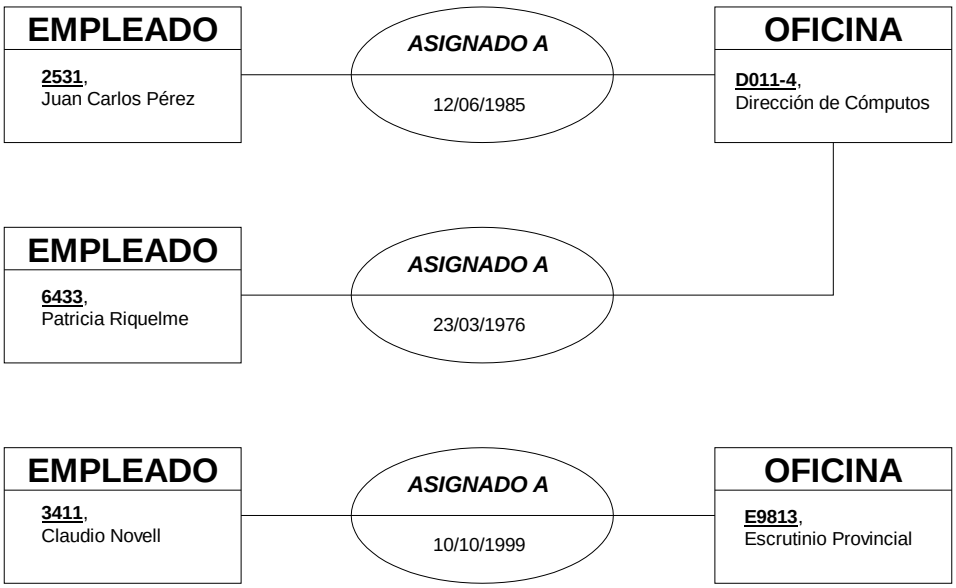
Propiedad-tipo:

Una propiedad-tipo, es una clase de propiedades semejantes. Una ocurrencia de una propiedad-tipo, es un valor tomado por esta propiedad.

Entidad - tipo, Relación - tipo y Atributos - tipo



Ocurrencias de Entidad - tipo, Relación - tipo y Atributos - tipo



En el ejemplo, se ve que los empleados **Pérez** y **Riquelme**, están asignados a la misma oficina (Dirección de Cómputos).

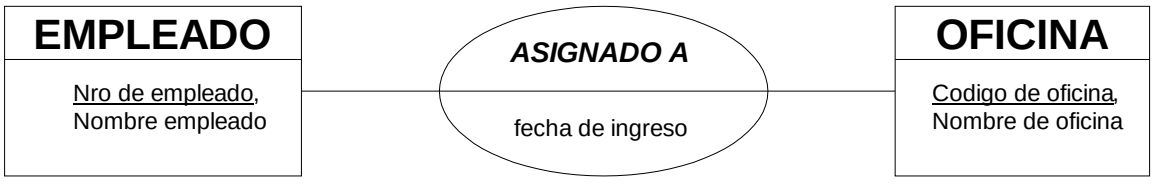
5.6 Características de una relación

5.6.1 Colección

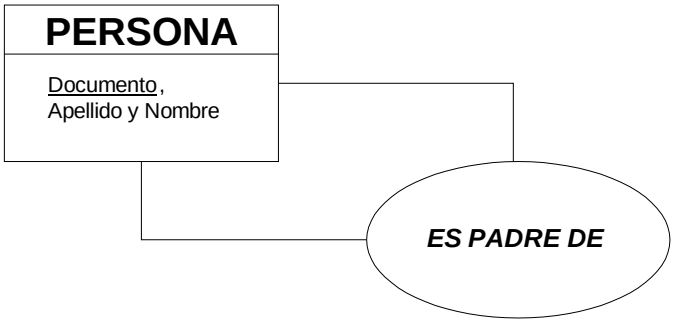
La colección de una relación-tipo, es la lista de entidades-tipo sobre las que se define la relación. En el ejemplo anterior, la colección de la relación ***ASIGNADO A*** se define sobre la colección (**EMPLEADO, OFICINA**).

5.6.2 Dimensión

La dimensión de una relación-tipo, es el número de ocurrencias de entidades alcanzadas por una única ocurrencia de la relación-tipo. Es superior o igual al número de entidades de la colección. Por ejemplo:



Colección: **EMPLEADO, OFICINA**
Dimensión: **2**



Colección: PERSONA
Dimensión: 2

Son necesarias dos ocurrencias de la relación **PERSONA** para una ocurrencia de la relación ***ES PADRE DE*** (se necesitan dos personas, una como padre y otra como hijo).

Una relación de dimensión 2, se la denomina relación **binaria.** Se hace esta acotación, ya que la mayoría de herramientas de diseño asistido CASE, trabajan con relaciones binarias. Otra característica que poseen, es que tales relaciones no pueden contener atributos, razón por la cual se deberán transformar en entidades débiles.

5.6.3 Funcionalidad

Se define la funcionalidad de una relación-tipo, como la correspondencia existente entre dos entidades-tipo **X e Y**, distinguiéndose las siguientes:

Uno a uno (1-1)

A cualquier ocurrencia de **X**, sólo le corresponde una ocurrencia de **Y** y recíprocamente.

Uno a muchos (1-n):

A toda ocurrencia de **X** corresponde 1 o varias ocurrencias de **Y**, y a toda ocurrencia de **Y**, corresponde una sola de **X**.

Muchos a muchos (m-n):

A toda ocurrencia de **X** corresponde una o varias ocurrencias de **Y** y recíprocamente.

5.6.4 Cardinalidad

La noción de cardinalidad permite expresar la totalidad o parcialidad de la participación de las entidades en la relación y su funcionalidad.

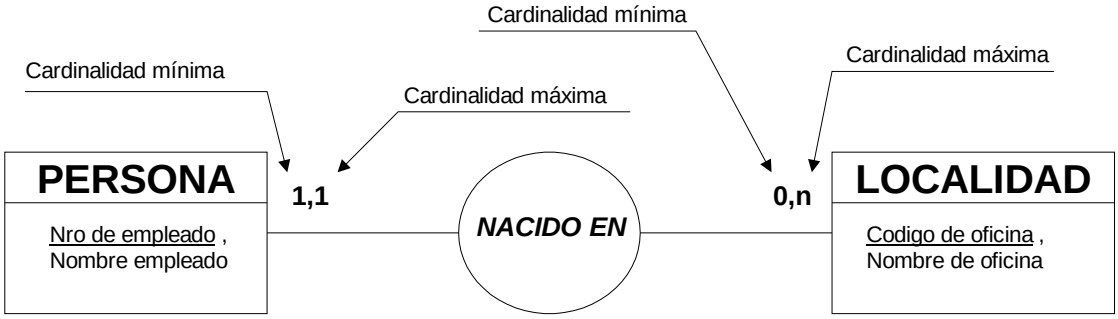
Cardinalidad Mínima:

La cardinalidad mínima de una relación, es el mínimo número de veces en la que cada ocurrencia de una entidad-tipo, participa en una relación-tipo. La cardinalidad mínima **0**, corresponde a una relación parcial. La cardinalidad mínima **1**, significa que no puede existir una ocurrencia de la entidad-tipo sin participar en una ocurrencia de la relación, vale decir que corresponde a una relación total.

Cardinalidad Máxima:

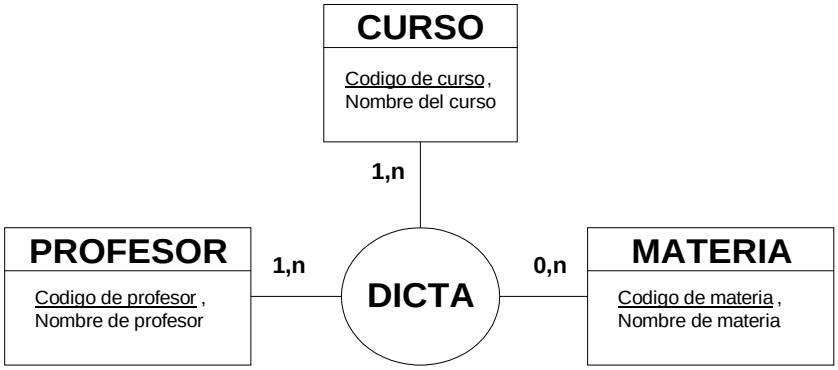
La cardinalidad máxima de una relación, es el máximo número de veces en la que cada ocurrencia de una entidad-tipo, participa en una ocurrencia de la relación-tipo. La cardinalidad máxima **1**, significa que cualquier ocurrencia de la entidad-tipo, no puede participar más que en una ocurrencia de la relación-tipo. La cardinalidad máxima **n**, significa que una ocurrencia de la entidad-tipo, puede estar implicada en un máximo de n ocurrencias de la relación.

Representación Gráfica



Representa que una **PERSONA**, ha nacido al menos en **1 LOCALIDAD** (cardinalidad mínima **1**) y a lo sumo en **1 LOCALIDAD** (cardinalidad máxima **1**), lo que es equivalente a decir que ha nacido en un único lugar. En una **LOCALIDAD** al menos pueden haber nacido **0 PERSONAS** (cardinalidad mínima **0** del subconjunto de personas considerado) y a lo sumo pueden haber nacido **n PERSONAS** (cardinalidad máxima **n**).

Ejemplo:



Un **PROFESOR** ***DICTA*** al menos una **MATERIA** y puede ***DICTAr*** varias. Una **MATERIA**, puede que no sea ***DICTAda*** y si se ***DICTA***, puede serlo muchas veces. En un **CURSO** se ***DICTA*** al menos una **MATERIA** con un **PROFESOR** pero pueden ***DICTA***rse varias **MATERIA**s siempre con un **PROFESOR**.

5.7 Reglas de gestión o reglas de negocio

Las reglas de gestión del modelo conceptual de datos, definen los condicionamientos que deben respetarse por el modelo. Por Ejemplo:

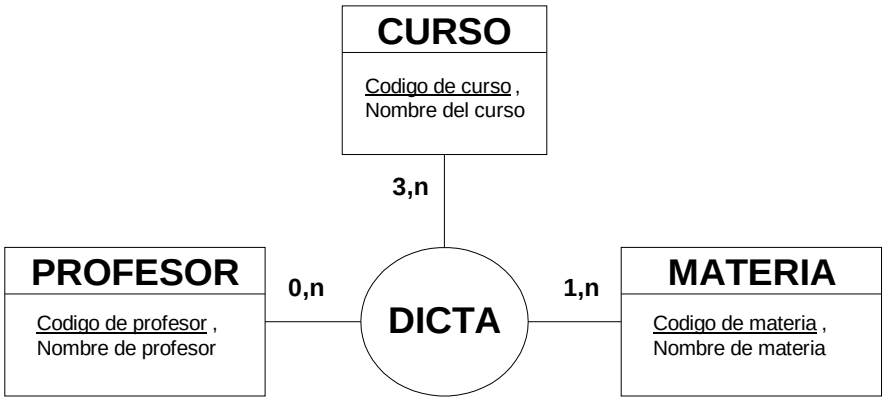
En el **MCD** de una escuela, las reglas de gestión pueden ser las siguientes:

Regla 1:
Todo **PROFESOR** ***DICTA*** en principio, al menos una **MATERIA**, pero algunos de ellos pueden tener licencia especial en razón de sus trabajos de investigación, vale decir que no ***DICTAN*** ninguna

Regla 2:
Toda **MATERIA**, se ***DICTA*** en al menos, un **CURSO**.

Regla 3:
En todo **CURSO**, se ***DICTAN*** al menos, tres **MATERIAS**.

El **MCD** sería entonces:



Es de destacar que la cardinalidad mínima **3** en curso, en realidad y desde el punto de vista de la funcionalidad de las relaciones, no tiene importancia. Simplemente es un detalle que podrá tener alguna regla de negocios posterior. En el modelo conceptual de datos, bastaría con indicar **1**, ya que precisamente lo que trata de recalcarse precisamente es si la participación de la entidad en la relación es parcial o

total (**0** ó **1**).

Las reglas de gestión, expresan los CONDICIONAMIENTOS DE INTEGRIDAD del modelo. Estos condicionamientos de integridad, representan las leyes del universo real que se modela del sistema de información. Se distinguen dos tipos de condicionamientos:

a. Condicionamientos estáticos

Se pueden referir a:

- **Una propiedad:** su formato, lista de posibles valores, entorno o rango de valores admisibles, etc.
- **Diversas propiedades de una misma relación o entidad**. Por ejemplo que la *fecha de pedido* sea menor o igual que la *fecha de entrega* y siendo éstas propiedades de la entidad **PEDIDO**.
- **Propiedades de ocurrencias distintas de una relación o entidad**. Por ejemplo que, en un histórico de precios de artículos, la *fechaInicio* del precio actual, debe ser mayor que la *fechaInicio* del último precio anterior.
- **Propiedades de entidades/relaciones diferentes.** Por ejemplo que la suma de la cantidad de productos vendidos y entregados a los clientes, debe ser igual a la suma de los productos que faltan en el depósito.
- **Cardinalidades**
- **Dependencias funcionales**

b. Condicionamientos dinámicos:

Los condicionamientos dinámicos, expresan las reglas de evolución y afectan directamente el paso del sistema de información de un estado a otro. Por ejemplo que se registre que el sueldo de un empleado en un período no puede ser inferior al sueldo del período del ejercicio anterior.

5.8 Dependencias funcionales

5.8.1 Dependencias funcionales entre propiedades

5.8.1.1 Dependencia funcional

Se dice que dos propiedades **a** y **b**, están ligadas por una dependencia funcional y se denota:

$$\mathbf{a} \longrightarrow \mathbf{df} \longrightarrow \mathbf{b}$$

si el conocimiento del valor de **a**, determina uno y sólo un valor de **b**.

p.e.:

$$\textit{Codigo\_cliente} \longrightarrow \mathbf{df} \longrightarrow \textit{Nombre\_cliente}$$

El conocimiento de *Codigo_cliente*, determina uno y sólo un *nombre_cliente.* En otras palabras, si se conoce el código del cliente, se debe poder conocer su nombre que será único.

Lo recíproco es falso. El nombre del cliente no permite conocer su código, pues varios clientes pueden tener el mismo nombre.

No necesariamente una dependencia funcional se referirá a un atributo elemental. La dependencia funcional, puede afectar a la concatenación de varias propiedades.

p.e.:

$$\textit{Nro\_pedido} + \textit{Codigo\_de\_articulo} \longrightarrow \mathbf{df} \longrightarrow \textit{Cantidad}$$

La sola referencia del *Codigo_de_articulo*, determinará solamente de qué artículo se trata pero no basta para determinar la cantidad pedida, ya que un mismo producto puede presentarse en varias boletas de pedido distintas. En consecuencia – si se admite que un artículo puede figurar a lo sumo una sola vez en una boleta de pedido – para poder determinar unívocamente el valor de una *Cantidad*, se deberá conocer el ***Nro_pedido + Codigo_de_articulo***.

5.8.1.2 Dependencia funcional elemental

Se dice que existe una ***dependencia funcional elemental*** entre **a** y **b**, y se indica como:

$$\mathbf{a} \longrightarrow \mathbf{b}$$

SI

$$\mathbf{a} \longrightarrow \mathbf{df} \longrightarrow \mathbf{b}$$

donde $\mathbf{a} = \mathbf{a}_1 + \mathbf{a}_2 + \ldots\ldots + \mathbf{a}_n$
tal que $\mathbf{a}_i$ es una propiedad elemental

y no existe ningún $\mathbf{a}_i$ que por sí solo, pueda determinar el valor de **b**.

p.e.: sea

Codigo_cliente + *Nombre_cliente* $\longrightarrow$ **df** $\longrightarrow$ *Direccion_cliente* **no es elemental**,

Donde **a** = *Codigo_cliente* + *Nombre_cliente*
entonces $\mathbf{a}_1$ = *Codigo_cliente*

$$\mathbf{a}_2 = Nombre\_cliente$$

pero para determinar la *Direccion_cliente*, basta solamente con *Codigo_cliente* ($\mathbf{a}_1$) (es más, ocurre que *Nombre_cliente* es también determinado por *Codigo_cliente*). Entonces se deduce que la expresión **1**, no constituye una *dependencia funcional elemental*, simplemente es una *dependencia funcional*.

En cambio
Codigo_cliente ⟶ **df** ⟶ *Direccion_cliente*

es una *dependencia funcional elemental* y se puede escribir:

Codigo_cliente ⟶ *Direccion_cliente*

Es de notar que una dependencia funcional elemental, únicamente es cuestionable si el valor de **a** es una concatenación de atributos elementales. En caso contrario, si ya es una dependencia funcional, ésta será elemental.

Para el ejemplo expuesto anteriormente:

Nro_pedido + *Codigo_de_articulo* ⟶ **df** ⟶ *Cantidad*

puede verse que **a** es una concatenación de atributos elementales (puede que sea o no una dependencia funcional elemental), y que el *Nro_pedido* **no** puede determinar el valor de *Cantidad* por sí solo, y el valor de *Codigo_de_articulo* tampoco puede determinar el valor de *Cantidad* por sí solo; pero la concatenación de ambos sí lo determina unívocamente, por lo tanto constituye una ***dependencia funcional elemental***.

Nro_pedido+*Codigo_de_articulo* ⟶ *Cantidad*
(dependencia funcional elemental)

5.8.1.3 Dependencia funcional elemental directa

Se dice que la propiedad **b** depende funcionalmente de **a** mediante una ***dependencia funcional elemental directa***, si:

1. Esta dependencia es elemental:
   a ⟶ **b**
2. ***No existe*** una propiedad **c**, tal que:
   a ⟶ **df** ⟶ **c** y **c** ⟶ **df** ⟶ **b**

Dicho de otra forma, se elimina toda transitividad.

p.e.:
un **PROFESOR** (*Codigo_profesor*, *Nombre_profesor*), dicta una y solamente una
MATERIA (*Codigo_materia*, *Nombre_materia*). La siguiente proposición:

Codigo_profesor ⟶ *Nombre_materia* (**1**)

Es una *dependencia funcional elemental* debido a que un profesor dicta una y solamente una materia, entonces el conocimiento del identificativo de profesor (*Codigo_profesor*), determinará el nombre de la materia (*Nombre_materia*) que dicta. También son válidas las siguientes proposiciones:

Codigo_profesor ⟶ *Codigo_materia* (**2**)

Codigo_materia ⟶ *Nombre_materia* (**3**)

Nótese que (**1**) **NO** es una ***dependencia funcional elemental directa***, ya que las dependencias funcionales (**2**) y (**3**) determinan la existencia de una dependencia transitiva. (**2**) y (**3**) son ***dependencias funcionales elementales directas***.

5.8.1.4 Clave de identificación de una entidad

Se conoce como clave de negocios o identificativo de una entidad, a una propiedad (o concatenación de propiedades) que pertenecen a esa entidad, tal que todas las demás propiedades, dependen de ella funcionalmente y de forma tal que no sea verdadera para ninguna de sus partes (dependencia funcional elemental).

Por ejemplo, sea la entidad **CLIENTE** con sus atributos (a, b, …., n):

CLIENTE (*Cod_cliente, Nombre_cliente, Direccion_cliente***)**

Cod_cliente + Nombre_cliente **no es una clave a pesar que**

Cod_cliente + *Nombre_cliente* ⟶ **df** ⟶ *Direccion_cliente*

ya que permanece verdadero para la parte *Cod_cliente* de la concatenación *Cod_cliente* **+** *Nombre_cliente*, ya que *Cod_cliente* determina perfectamente *Direccion_cliente*.

En cambio ***Cod_cliente*** es una clave, pues:

Cod_cliente ⟶ *Nombre_cliente*
Cod_cliente ⟶ *Direccion_cliente*

Se debe tener en cuenta que una entidad puede tener **varias claves**. Se debe prever y especificar claramente en el modelo, cuál de las claves actuará como identificativo. Así, si la entidad **EMPLEADO**, tiene las propiedades *Codigo_empleado* y *CUIL* que son clave, habrá que elegir cual de las dos se desempeñará como identificativo.

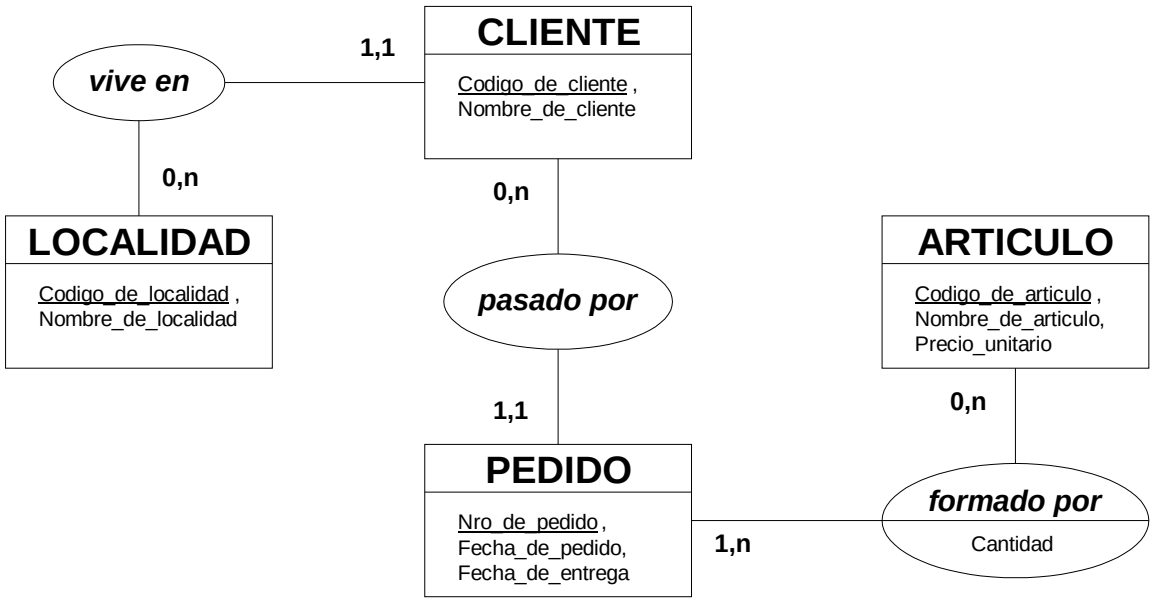
Obsérvese además que la dependencia funcional por la cual una propiedad depende de la clave, no es necesariamente elemental. Todo lo mencionado respecto a dependencias funcionales y clave de identificación de una entidad puede verse reflejado

en el siguiente ejemplo de aplicación.

Sean las siguientes reglas de gestión:

1. Un **CLIENTE** reside en una única **LOCALIDAD**.
2. Una boleta de **PEDIDO** corresponde a uno y solamente un **CLIENTE**.
3. Una boleta de **PEDIDO** estará conformada por varios **ARTÍCULOS** conociéndose la *Cantidad* de cada uno de ellos.
4. Un **ARTÍCULO** puede estar presente en varios **PEDIDOS**, pero no podrá estar repetido en el mismo.

El correspondiente modelo será de la forma:



En la entidad **ARTÍCULO**, se tiene la dependencia funcional:

$$Codigo\_de\_articulo \longrightarrow Nombre\_de\_Articulo$$

Por la relación ***formado por***, se tiene la dependencia:

$$Nro\_de\_pedido + Codigo\_de\_Articulo \longrightarrow Cantidad$$

Lo que significa que el conocimiento del *Nro_de_pedido* y del *Codigo_de_Articulo* determina la cantidad pedida.

Generalidades sobre los identificativos o clave de negocios

- Todas las ocurrencias de una entidad - tipo, deben ser distintas.

- Toda entidad - tipo, debe contar al menos, con una clave de negocios candidata.
- Una de las claves de negocio candidatas deberá ser elegida como identificativo principal de la entidad – tipo.
- El identificativo principal debe ser la mínima de las claves de negocio candidatas. Ninguna de sus partes puede a su vez, ser una clave candidata.
- Una clave de negocio candidata que no es el identificativo es una clave de negocios alternativa.

5.8.2 Dependencias funcionales entre entidades

Las dependencias funcionales entre propiedades, hay que considerarlas respecto a las entidades y relaciones.

Se dice que existe una dependencia funcional entre dos entidades **A** y **B** y se indica como:

A ⟶ **B**

si toda ocurrencia de **A**, determina una y sólo una ocurrencia de **B**.

En el ejemplo anterior, se tiene (a través de la relación ***pasado por***).

PEDIDO ⟶ **CLIENTE**

ya que el solo conocimiento del **PEDIDO** permite determinar unívocamente a qué **CLIENTE** pertenece.

Las cardinalidades **1,1** de **PEDIDO** en esta relación, expresan que todo cualquier **PEDIDO**, determina uno y sólo un cliente. Se concluye entonces que

LAS CARDINALIDADES 1,1 CORRESPONDEN SIEMPRE A UNA DEPENDENCIA FUNCIONAL

Hay que considerar las dependencias funcionales entre entidades a través de las relaciones entre esas entidades. Es posible asimilar las dependencias funcionales entre entidades a las dependencias funcionales entre los identificativos de estas entidades. Entonces con respecto a lo declarado anteriormente

PEDIDO ⟶ **CLIENTE**

puede asimilarse a:

Nro_de_pedido ⟶ *Codigo_de_cliente*

Otra ocurrencia referida al ejemplo puede ser:

CLIENTE ⟶ **LOCALIDAD**
Codigo_de_cliente ⟶ *Codigo_de_localidad*

5.8.3 Propiedades de las dependencias funcionales

Reflexividad:

a ⟶ *df* ⟶ ***a***

Proyección:

si ***a*** ⟶ *df* ⟶ ***b + c*** entonces
a ⟶ *df* ⟶ ***b*** y ***a*** ⟶ *df* ⟶ ***c***

Ejemplo: *Codigo_cliente* ⟶ *Nombre_cliente+Domicilio*
entonces
Codigo_cliente ⟶ *Nombre_cliente* y
Codigo_cliente ⟶ *Domicilio*

Ampliación:

si ***a*** ⟶ *df* ⟶ ***b*** y
a ⟶ *df* ⟶ ***c*** entonces
a + b ⟶ *df* ⟶ ***c***

Ejemplo: *Codigo_cliente* ⟶ *Nombre_cliente* y
Codigo_cliente ⟶ *Domicilio*
entonces
Codigo_cliente + Nombre_cliente ⟶ *Domicilio*

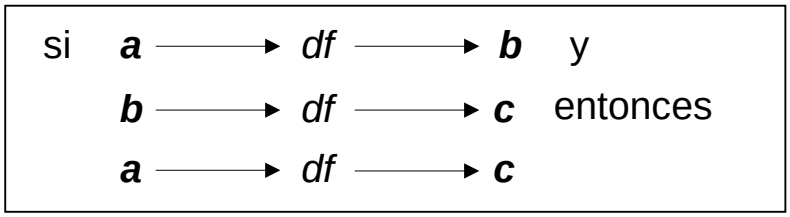
Aditividad:

si ***a*** ⟶ *df* ⟶ ***b*** y
a ⟶ *df* ⟶ ***c*** entonces
a ⟶ *df* ⟶ ***b + c***

Ejemplo: *Codigo_cliente* ⟶ *Nombre_cliente* y
Codigo_cliente ⟶ *Domicilio*
entonces

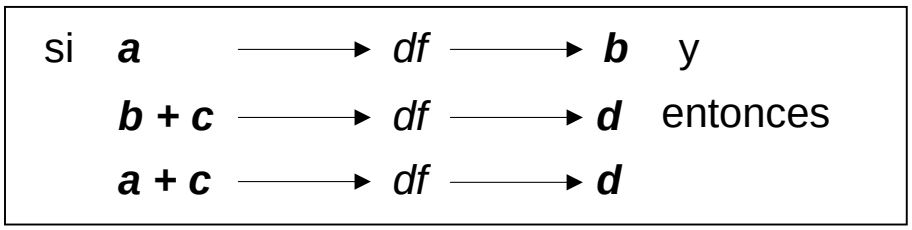
Codigo_cliente ⟶ *Nombre_cliente+Domicilio*

Transitividad:



Ejemplo: *Codigo_cliente* ⟶ *Codigo_postal* y
Codigo_postal ⟶ *Nombre_localidad*
entonces
Codigo_cliente ⟶ *Nombre_localidad*

Pseudo-Transitividad:



Ejemplo: *Codigo_cliente* ⟶ *Codigo_postal* y
Codigo_postal + Nombre_cliente ⟶ *Nombre_localidad*
entonces
Codigo_cliente + Nombre_cliente ⟶ *Nombre_localidad*

SEGUNDA PARTE
LA NORMALIZACIÓN DEL MODELO CONCEPTUAL

5.9 Introducción

En la primera parte del tema se han presentado las definiciones primitivas para la modelación conceptual de datos. En ésta, a partir de todas esas primitivas enunciadas, se desarrollarán metodologías que permitirán construir modelos conceptuales de manera consistente y con la eliminación total de redundancias. Éste último es el objetivo principal de **La Normalización**. *La normalización, es un proceso mental abstracto totalmente natural y obvio.* Consiste solamente en aplicar el sentido común y realizar las cosas de manera obvia.

5.10 La Normalización

La normalización, es una técnica, basada en reglas bien definidas que permite que todas las entidades y relaciones presentes en el sistema, tengan las redundancias eliminadas.

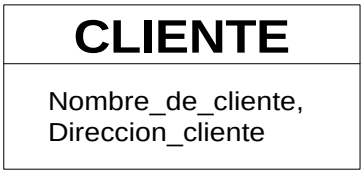
Las entidades del modelo conceptual de datos, deben verificar todas las reglas de normalización que serán desarrolladas a continuación.

La normalización, se realiza utilizando descomposición **sin pérdidas**. Ello implica que a partir de un conjunto de datos agrupados originalmente de una manera determinada, se realice la descomposición en varios subconjuntos sin perder el sentido que presentaba la información original. Los procesos que llevan a realizar la descomposición, se organizan por niveles de complejidad, yendo de lo más sencillo y general a lo de mayor detalle y particular. Tales niveles son los que conforman las denominadas Formas Normales.

5.10.1 Primera forma normal (1FN)

En una entidad, todas las propiedades son elementales, y existe al menos una clave de negocios o identificativo definido. Cada atributo deberá tener un único valor para cualquier ocurrencia de la entidad en un momento cualquiera.

Por ejemplo, sea la entidad **CLIENTE**:

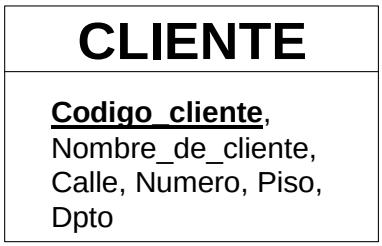


Esta entidad, no está en **1FN**, pues, no existe clave (varios clientes pueden tener el mismo nombre). Por otra parte, si *Dirección_cliente* es:

Direccion_cliente = Calle + Numero + Piso + Departamento

no constituye una propiedad elemental, entonces por más que la entidad tenga una clave de negocios, no estaría en **1FN** por la propiedad concatenada utilizada.

La clave, si es única, se tomará como identificativo. Si existen varias claves, se elegirá una de ellas. Toda entidad debe tener un identificativo. Para que esté en 1FN, debería ser:

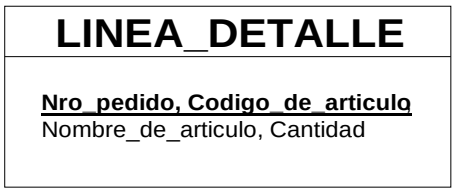


Nótese que la clave o identificativo sugerido, se destaca marcándose en letra negrita o subrayado (en el ejemplo, de ambas formas).

5.10.2 Segunda forma normal (2FN)

*Para que una entidad esté en segunda forma normal, deberá estar en **1FN** y además, toda propiedad de la entidad deberá depender de la clave mediante una dependencia funcional elemental. Dicho de otra manera, toda propiedad de la entidad deberá depender del identificativo completo (caso de tratarse de una concatenación de atributos) y no solamente de una parte de él.*

Obsérvese que, de acuerdo a la definición, el caso del ejemplo anterior de la entidad **CLIENTE**, que se encuentra en **1FN**, además está en **2FN**, ya que su clave de identificación no está conformada por una concatenación de atributos, en consecuencia todos los atributos dependen de la clave a través de una *dependencia funcional elemental*. Como regla práctica, se deberá poner en tela de juicio todas aquellas entidades en las que la clave esté conformada por dos o más atributos. Por ejemplo, sea la entidad **LINEA_DETALLE** de un **PEDIDO**:



El ejemplo presentado está en **1FN** ya que posee clave de identificación y además todos los atributos son elementales. Como la clave está conformada por la concatenación de dos atributos, cabe cuestionarse si está o no en **2FN**. De acuerdo al gráfico, tal clave de identificación está dada por *Nro_pedido + Codigo_de_Articulo*, pero la dependencia funcional:

Nro_pedido + Codigo_de_Articulo ⟶ **df** ⟶ *Nombre_de_articulo*

no es elemental, puesto que:

Codigo_de_Articulo ⟶ **df** ⟶ *Nombre_de_articulo*

En consecuencia, esta entidad no está en **2FN**. Para llevarla a tal estado, el MCD debería ser:

5.10.3 Tercera forma normal (3FN)

Para que una entidad esté en tercera forma normal, deberá estar en 2FN y además, todas las propiedades deberán depender de la clave mediante una dependencia funcional elemental directa.

Por ejemplo, sea la entidad **CLIENTE**:

Esta entidad está en **2FN** ya que la clave no es una concatenación de atributos y todas las propiedades dependen funcionalmente de esa clave (consecuentemente estará en **1FN**), pero no está en **3FN** ya que la dependencia funcional:

Codigo_cliente ⟶ **df** ⟶ *Nombre_de_localidad*

no es directa a causa de la transitividad

Codigo_cliente ⟶ **df** ⟶ *Codigo_de_localidad* ⟶ **df** ⟶ *Nombre_de_localidad*

El MCD debería ser:

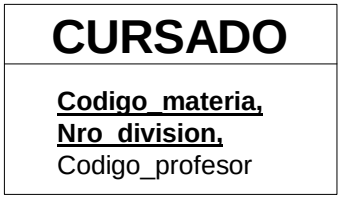
5.10.4 Forma normal de BOYCE-CODD (BCFN)

Si una entidad tiene un identificativo concatenado, ninguno de los elementos que componen este identificativo debe depender de alguna propiedad.

La **1FN, 2FN y 3FN**, se ocupan de restricciones que afectan sólo a las propiedades **no claves**. No obstante, es común referirse a la **BCFN**, como **3FN**.

Si se admiten las reglas de gestión:

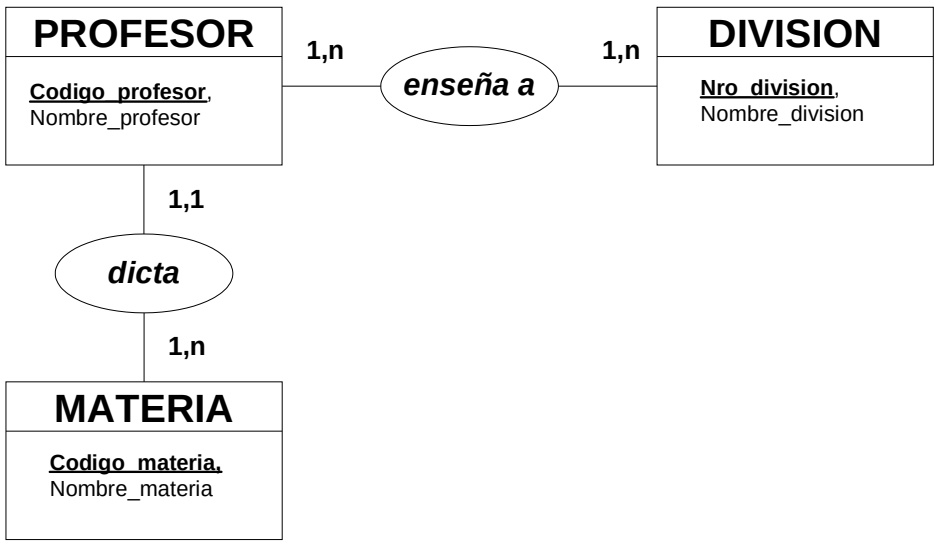
- Una **MATERIA** puede ser impartida por varios **PROFESOR**es pero en una **DIVISIÓN**, es dictada por un único profesor.
- Existen varias divisiones identificadas por número de división.



Para el caso planteado, la entidad del gráfico está normalizada ya que el profesor depende de la materia y del curso en donde la dictará. No puede determinarse de ninguna otra manera. Además ninguna de las partes de la clave determina a ningún otro atributo. Si se agrega la siguiente restricción:

- Cada **MATERIA** es impartida por un único **PROFESOR**.

no está en **BCFN**, pues, la clave de la entidad, dada por *Codigo_materia + Nro_division*, depende de la propiedad *Codigo_profesor*. Concretamente se establece la dependencia *Codigo_profesor* → *Codigo_materia*. El MCD debería ser entonces:



Las normalizaciones anteriores, tienen por objeto eliminar las redundancias (no es necesario repetir la descripción de un producto pedido, cada vez que se solicite dicho producto) y eliminar las anomalías de actualización (si se elimina un empleado, se deseará sin duda conservar la oficina a la que estaba asignado).

5.11 Cumplimiento de las condiciones de integridad – Las Relaciones

Para el desarrollo de todas las condiciones de integridad, se tomará en consideración un ejemplo de aplicación que tendrá las siguientes características:

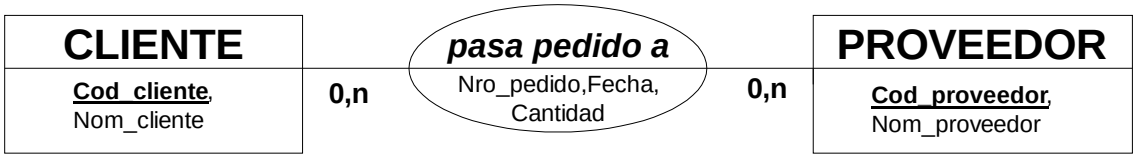
1. Un **CLIENTE** puede realizar sus **PEDIDOs** de **ARTÍCULOs** a distintos **PROVEEDORes** cada uno en una *fecha* determinada y con un *número de pedido*.

2. El **CLIENTE** al pasar el pedido al **PROVEEDOR** especifica a través del **PEDIDO** la *cantidad* de **ARTÍCULOs** que solicita.

5.11.1 Verificación

*En toda ocurrencia de entidad-tipo o de relación-tipo, no debe existir más que un único valor de cada propiedad (no repetitiva). Para las entidades esta regla procede de la **1FN**. Tal regla, deberá permanecer verdadera para las relaciones.*

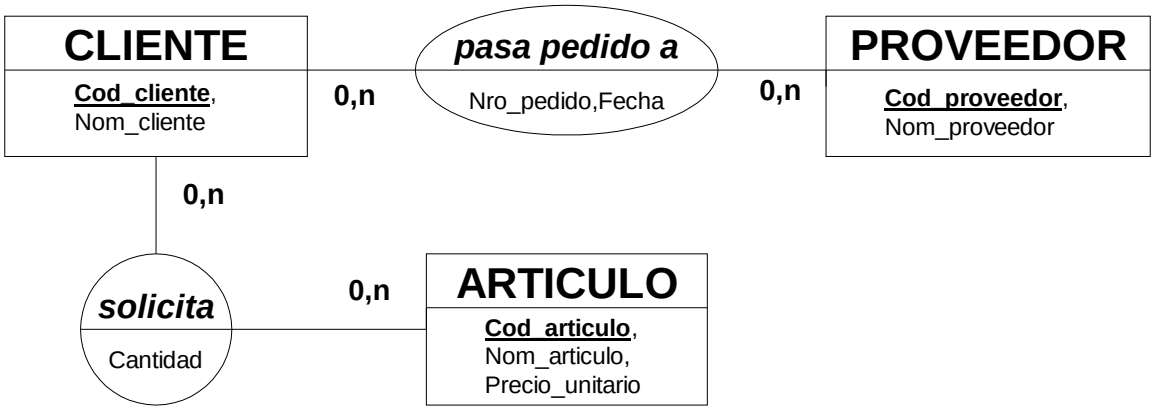
De acuerdo a la descripción anterior, considérese el siguiente MCD:



Nótese que la relación ***pasa pedido a***, no está verificada, ya que puede haber varios valores de *Cantidad* en un pedido realizado por un cliente a un proveedor (una por cada artículo solicitado). Entonces, el valor de *Cantidad*, no depende sólo del cliente y del proveedor, sino también del artículo solicitado. En otras palabras:

En una relación, las propiedades deben depender funcionalmente de las entidades que son colección de la relación, o lo que es lo mismo, de la concatenación de sus identificativos. En consecuencia, la clave de una relación es la concatenación de las claves de las entidades que son colección de ella.

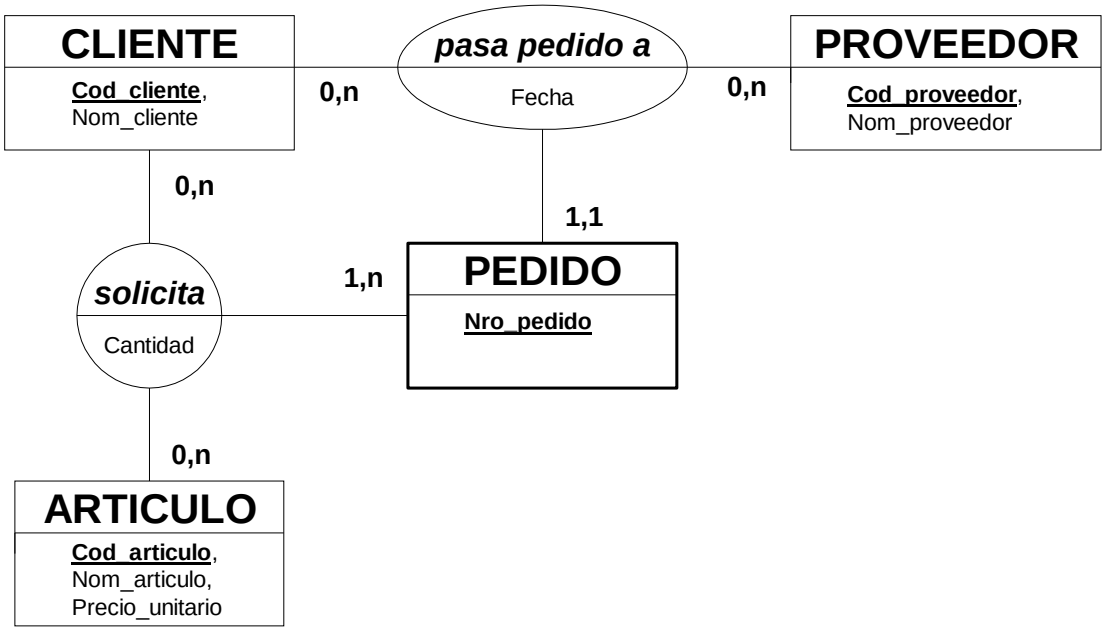
El siguiente MCD, mejora al anteriormente expuesto:



Nótese que ahora que la propiedad *Cantidad*, que figuraba en la relación ***pasa pedido a***, desaparece es ésta pues es inherente a los artículos pedidos (relación ***solicita***).

No obstante, en la relación ***solicita***, la *Cantidad*, no depende solamente del cliente y del artículo, sino también del Número de pedido, ya que un cliente puede realizar varios pedidos del mismo artículo. El Número de pedido (*Nro_pedido*), no puede deducirse aunque se conozca el CLIENTE, el ARTÍCULO solicitado y el PROVEEDOR al que fue pasado tal pedido, ya que puede haber varias boletas de pedido para un cliente dado, un representante dado y un artículo dado. En consecuencia, no se cumple la regla de verificación.

En este caso, se hace necesario crear la entidad-tipo **PEDIDO**. El MCD final quedará entonces de la manera que se representa seguidamente.



En el modelo, nótese ahora que una única ocurrencia de la propiedad *Cantidad*, depende del *Nro_pedido*, del *Cod_Articulo* y del *Cod_cliente*. Esta concatenación la determina perfectamente concluyendo que el modelo está VERIFICADO.

5.11.2 Normalización de las relaciones

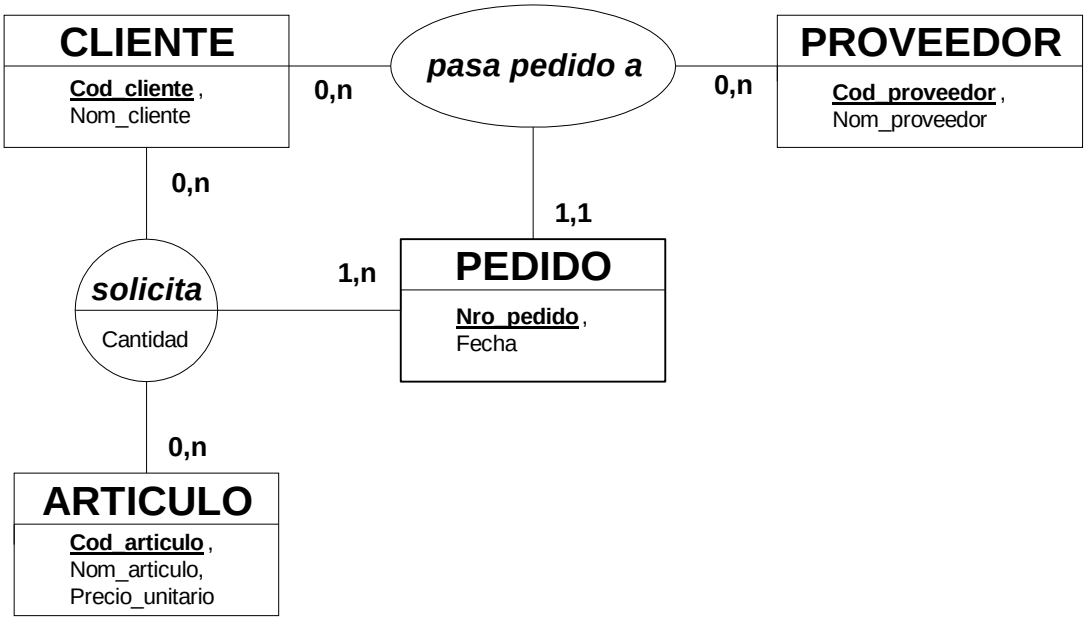
Cada propiedad de la relación, debe depender funcionalmente de la concatenación del conjunto de identificativos de las entidades colección de la relación, y no de ningún subconjunto de ellos. Esto implica que estos atributos deben tener una dependencia elemental respecto del conjunto.

Entonces, deberá haber una dependencia total de las propiedades de la relación con respecto a las entidades.

Considerando el gráfico anterior, se tiene que la propiedad *fecha* (en la relación ***pasa pedido a***), depende de ***Nro_pedido*** + ***Cod_cliente*** + ***Cod_proveedor***; y que la propiedad *Cantidad* (en la relación ***solicita***) depende de ***Cod_cliente + Nro_pedido + Cod_Articulo***.

Es de notar que la fecha en que se realiza un pedido, solamente depende del pedido, sin importar a que cliente corresponde o a qué proveedor es pasado, en consecuencia, tal dependencia es solamente de un subconjunto de la concatenación de los identificativos mencionados, no existiendo dependencia plena de las entidades (o de sus identificativos) **CLIENTE** o **PROVEEDOR** que participan en la relación ***pasa pedido a*** por lo que la relación no está normalizada. Para normalizarla, la propiedad *Fecha,* debe desaparecer de la relación ***pasa pedido a*** e incluirse en la entidad **PEDIDO**.

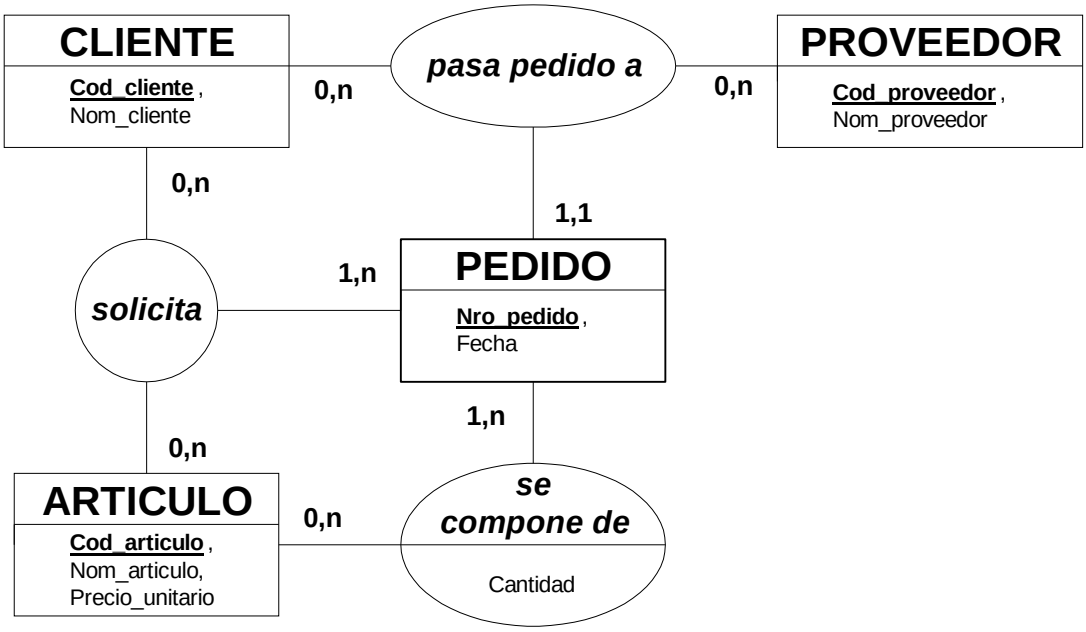
El MCD quedará:



Pero, la *Cantidad* pedida puede conocerse si se sabe al ***Nro_pedido*** y el ***Cod_Articulo*** de donde se concluye que la relación ***solicita*** no está normalizada, ya que involucra además a la entidad **CLIENTE** (***Cod_cliente***) sin depender de ésta. Se tiene entonces que:

Nro_pedido* + *Cod_Articulo ⟶ *Cantidad*

En este caso, la propiedad *Cantidad*, no puede migrarse a una entidad como se hizo con el atributo *Fecha*, entonces *Cantidad*, debe estar afectada a una relación distinta, no existente hasta el momento. Se hace necesario crear la relación-tipo ***se compone de*** en la que sólo intervendrán las entidades **PEDIDO** y **ARTÍCULO**. El MCD propuesto queda entonces:



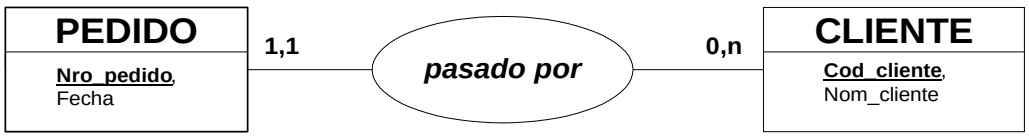
5.11.3 Descomposición de relaciones

*Consiste en reemplazar una relación de dimensión **n**, en varias relaciones de dimensiones más pequeñas, utilizando las dependencias funcionales que se pueden detectar en la relación.*

Del ejemplo, una relación pasible a ser descompuesta es ***solicita***. En la relación ***pasa pedido a*** del MCD anterior, existe la dependencia funcional:

PEDIDO ⟶ **CLIENTE**

Se puede, por consiguiente, descomponer esta relación en dos:



Y en

De lo hecho, la relación ***pasado por***, ya existía en la relación ***pasa pedido a*** (la dependencia **PEDIDO** ⟶ **CLIENTE** procede de la relación ***pasa pedido a***). La descomposición no es posible más que con dos condiciones**:**

1. *La cardinalidad mínima de las entidades de la izquierda en la dependencia funcional, debe ser 1 en la relación a descomponer (relación total para estas entidades).*

2. *Si la dependencia funcional procede de otra relación diferente a la que se desea descomponer, es necesario que se refiera a las mismas ocurrencias de entidades que la relación a descomponer.*

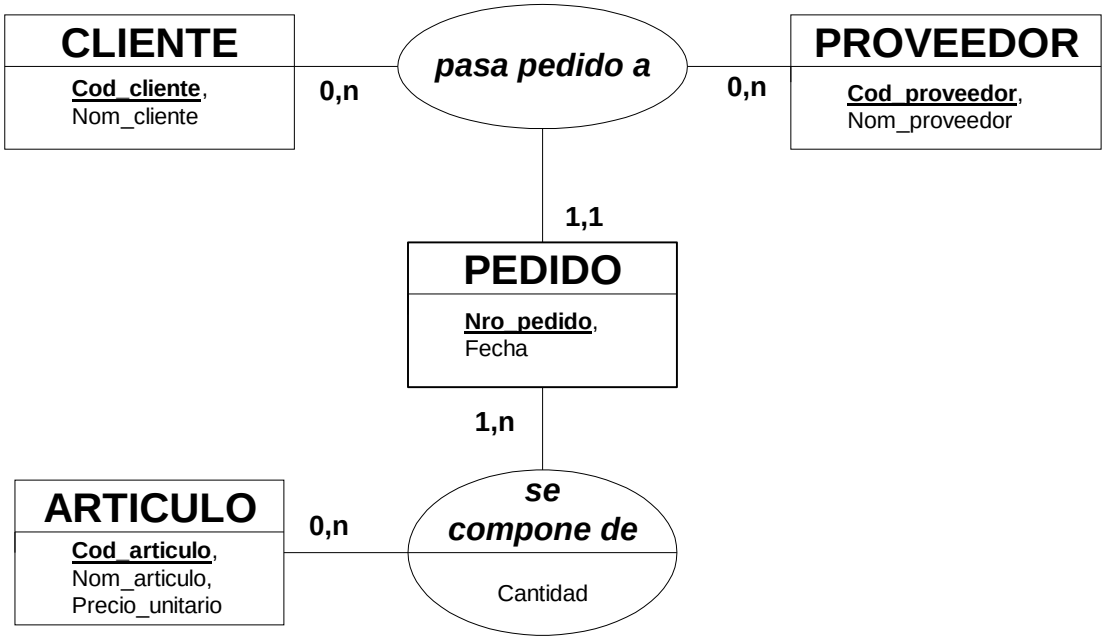
En el ejemplo, la dependencia funcional que ha permitido la descomposición es:

PEDIDO ⟶ **CLIENTE**

1. A través de esta dependencia, aplicada a la relación ***pasado por***, determina una cardinalidad mínima de **PEDIDO** es **1** (y máxima de **1** por ser una dependencia funcional). Aplicando la misma dependencia sobre la relación ***solicita***, la cardinalidad mínima de **PEDIDO** en esta relación es **1**, y en consecuencia, se asegura el cumplimiento de la regla 1.

2. La relación ***pasa pedido a***, y la relación a descomponer ***solicita***, ponen en juego las mismas ocurrencias de **CLIENTE** y de **PEDIDO**, pues son los propios clientes los que efectúan los pedidos y los que piden los artículos, y por consiguiente los artículos pedidos por los clientes corresponden a los mismos pedidos que los pedidos efectuados por los clientes. Se cumple la regla 2.

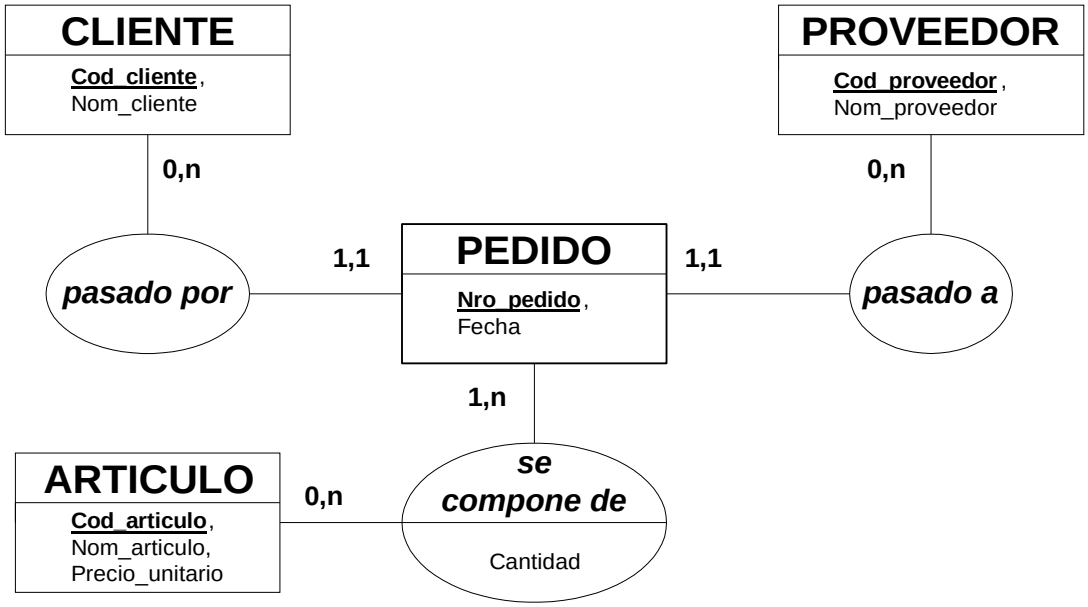
En el MCD la relación ***solicita***, se puede descomponer, por consiguiente, en ***pasa pedido a*** (que ya existía) ya que un **PEDIDO** se refiere a uno y solamente un **CLIENTE**, y en una nueva relación ***pide producto*** que vincularía al **CLIENTE** con los **ARTÍCULOS** solicitados. Como los artículos solicitados están vinculados con **PEDIDO** a través de la relación ***se compone de***, entonces puede suprimirse el doble empleo de la misma relación. Consecuentemente, la relación ***solicita*** desaparece, quedando el modelo:



De igual forma, las dependencias funcionales (debidas a las cardinalidades 1,1 de **PEDIDO** en ***pasa pedido a***):

PEDIDO ⟶ **CLIENTE** y
PEDIDO ⟶ **PROVEEDOR**

permiten descomponer ***pasa pedido a*** en dos relaciones binarias ***pasado por*** (entre **CLIENTE** y **PEDIDO**) y ***pasado a*** (entre **PROVEEDOR** y **PEDIDO**). De estas descomposiciones, se obtiene el siguiente MCD:



5.12 Aplicación de formas normales

La normalización, no es una actividad aplicable solamente a las metodologías que apuntan al diseño de bases de datos. Ésta, es necesaria siempre que se trabaje con datos (atributos) que deban ser organizados de cierta manera (entidades y relaciones) y que permitan mantener la integridad total respecto a la situación de partida. La forma en que serán posteriormente utilizadas estas agrupaciones definidas, podría ser manual, o automatizada y dentro de ésta, mediante sistemas de archivos convencionales o bien sobre bases de datos. Los ejemplos desarrollados a continuación, permiten apreciar la aplicación de las distintas instancias de evolución a través de las formas normales, logrando de esta manera reunir toda la información de partida con las redundancias eliminadas. Se tomarán para ello, atributos agrupados en forma de tablas del tipo planilla de electrónica. Demás está agregar que estos ejemplos, no apuntan a ninguna forma de utilización de datos bajo un lenguaje o base de datos específico.

Reglas de gestión:

1. La organización objeto de estudio, cuenta con varios departamentos en que está subdivida.
2. Todos los empleados de la organización, deben estar asignados a uno y sólo un departamento.
3. Cada departamento tiene un sólo jefe.
4. Existen varios proyectos de distinto tipo.
5. Cada empleado, tiene asignado uno o varios proyectos distintos, y una cantidad de horas asignadas a cada uno.

La primera tabla que se planteará, contendrá toda la información volcada. En ella puede verse la estructura (columnas que la componen) y las ocurrencias presentadas (filas).

EMPLEADO		DEPARTAMENTO				PROYECTO			
		Descripción		Jefe					
Código	Nombre	Código	Nombre	Código	Nombre	Código	Nombre	Inicio	Horas
COEMP	*NOEMP*	*CODEP*	*NODEP*	*COJEF*	*NOJEF*	*COPRO*	*NOPRO*	*FEIN*	*HS*
908	SÁNCHEZ	10	VENTAS	988	PÉREZ	10 30 40	FACTIBILIDAD DESARROLLO MERCADO	12/10/99 05/01/00 02/02/00	150 260 350
562	GÓMEZ	20	MARKETING	387	LOYARTE	20 50	ANÁLISIS SOFTWARE	20/12/99 23/11/99	400 600
988	PÉREZ	10	VENTAS	988	PEREZ	10 20 40	FACTIBILIDAD ANÁLISIS MERCADO	03/11/99 11/12/99 28/01/00	230 450 480
921	MÉNDEZ	15	INSUMOS	919	ROLDAN	20	ANÁLISIS	15/02/00	630

Una forma directa de construir la tabla, es generar el llamado registro compuesto. El registro compuesto, tiene toda la información con respecto a cada conjunto existente de formas de relación que afecten a todas las variables. La cuestión es hacer esto de modo de obtener un agrupamiento significativo. Para lograr un registro compuesto, se necesita una forma de relación compuesta como la propuesta. La tabla propuesta, tiene toda la información que se puede necesitar para muchas aplicaciones. Y éste es justamente el problema. MUCHA INFORMACIÓN ES REPETITIVA. Por ejemplo, el nombre del proyecto, está repetido varias veces; el nombre del departamento también entre otras cosas y puede llevar a la inconsistencia de la información repetida, además de ocupar espacio innecesario.

Partiendo de un registro compuesto, se puede ahorrar mucho espacio conservando los datos sobre empleados, proyectos y departamentos en conjuntos separados. Si se procede de este modo, es necesario encontrar una forma de recolectar los datos más tarde para obtener el registro compuesto o un registro menor si es todo lo que se necesita para una aplicación en particular. A esto apunta la Normalización.

1FN

Mediante la aplicación de la teoría especificada para la primera forma normal y considerando la entidad **EMPLEADO** en relación al proyecto asignado que tiene, se obtendrían las siguientes tablas:

EMPLEADO

COEMP
NOEMP
CODEP
NODEP
COJEF
NOJEF

COEMP	NOEMP	CODEP	NODEP	COJEF	NOJEF
908	SÁNCHEZ	10	VENTAS	988	PÉREZ
562	GÓMEZ	20	MARKETING	387	LOYARTE
988	PÉREZ	10	VENTAS	988	PÉREZ
921	MÉNDEZ	15	INSUMOS	919	ROLDÁN
919	ROLDÁN	15	INSUMOS	919	ROLDÁN

En donde la clave es el código de empleado **COEMP**.

ASIGNACION

COEMP COPRO
NOPRO
FEIN
HS

COEMP	COPRO	NOPRO	FEIN	HS
908	10	FACTIBILIDAD	12/10/99	150
908	30	DESARROLLO	05/01/00	260
908	40	MERCADO	02/02/00	350
562	20	ANÁLISIS	20/12/99	400
562	50	SOFTWARE	23/11/99	600
988	10	FACTIBILIDAD	03/11/99	230
988	20	ANÁLISIS	11/12/99	450
988	40	MERCADO	28/01/00	480
921	20	ANÁLISIS	15/02/00	630

En donde la clave es la concatenación de las propiedades código de empleado y código de proyecto: **COEMP + COPRO**.

2FN

De acuerdo a la segunda forma normal, en la tabla de asignaciones, el nombre del proyecto (**NOPRO**), depende solamente del código de proyecto (**COPRO**) y no de toda la clave concatenada, por lo que se para llevarla a 2FN se debe descompone:

ASIGNACION

COEMP COPRO
FEIN
HS

COEMP	COPRO	FEIN	HS
908	10	12/10/99	150
908	30	05/01/99	260
908	40	02/02/00	350
562	20	20/12/99	400
562	50	23/11/99	600
988	10	03/11/99	230
988	20	11/12/99	450
988	40	28/01/00	480
921	20	15/02/00	630

Consecuentemente, la tabla proyecto, quedará de la forma:

PROYECTO

COPRO
NOPRO

COPRO	NOPRO
10	FACTIBILIDAD
20	ANÁLISIS
30	DESARROLLO
40	MERCADO
50	SOFTWARE

3FN

Según los principios de la tercera forma normal, en la tabla **EMPLEADO**, existen dos dependencias funcionales no directas. En uno de los casos:

CODEP ⟶ **NODEP**

que se resuelve descomponiendo. En el otro:

COJEF ⟶ **NOJEF**

Puede ser absorbida en la misma tabla, pues un jefe también es un empleado. Entonces:

EMPLEADO

COEMP
NOEMP
CODEP
COJEF

COEMP	NOEMP	CODEP	COJEF
908	SÁNCHEZ	10	988
562	GÓMEZ	20	387
988	PÉREZ	10	988
921	MÉNDEZ	15	908
919	ROLDÁN	15	919

DEPARTAMENTO

CODEP
NODEP

CODEP	NODEP
10	VENTAS
15	INSUMOS
20	MARKETING

BCFN (Forma normal de Boyce Codd)

Para ejemplificar esta forma normal, se recurrirá a la siguiente tabla:

DEPARTAMENTO	PROYECTO	RESPONSABLE	HORAS-HOMBRE
10	P1	362	1.000
10	P3	486	500
15	P1	298	2.500
15	P2	320	1.700
15	P3	486	800

En la que la clave es una concatenación de las propiedades **DEPARTAMENTO** y **PROYECTO**.

Si se agregan las siguientes reglas de gestión:

1. Ningún empleado, puede ser responsable de más de un proyecto y todos los proyectos tienen al menos un responsable

2. Cada departamento no tiene más que un responsable por proyecto

La tabla, no está en **BCFN**, pues la propiedad **PROYECTO** de la clave concatenada, depende de la propiedad no clave **RESPONSABLE**. Se obtiene entonces la **BCFN** descomponiendo:

DEPARTAMENTO	RESPONSABLE	HORAS-HOMBRE
10	382	1.000
10	486	500
15	298	2.500
15	320	1.700
15	486	800

RESPONSABLE	PROYECTO
382	P1
486	P3
298	P1
320	P2

En donde la clave es **DEPARTAMENTO + RESPONSABLE**, y **RESPONSABLE** respectivamente.

5.13 Formas normales de orden superior

Ciertos autores, consideran la existencia de otras formas normales que son casos muy particulares. Si se trabaja ordenadamente, normalizando entidades (hasta 3FN), verificando, normalizando y descomponiendo relaciones, no será necesario plantearse si se cumple o no con la 4FN o 5FN. Se verá además que si bien siempre se nombran las entidades, la 4FN y la 5FN hacen referencia a la forma en que se pueden o no descomponer relaciones.

5.13.1 Cuarta forma normal (4FN)

Para estar en **4FN**, la entidad no debe contener dependencias de valores múltiples. Por ejemplo:

Sean las entidades: **EMPLEADO, LENGUAJE** y **HARDWARE**
Sean las relaciones:

EMPLEADO conoce LENGUAJE
EMPLEADO conoce HARDWARE

y considérese que un empleado, puede ser diestro en uno o más tipos de lenguajes y hardware, y estos conocimientos son independientes entre sí. Tómese como ejemplo la ocurrencia del empleado **328** que posee conocimientos de:

LENGUAJE:	1 – PASCAL	**HARDWARE**:	100 – AS400
	2 – COBOL		200 – VAX
	3 – BASIC		300 – PC

La representación de esta información podría realizarse de las siguientes formas:

DISYUNTO

Nº Empleado	Lenguaje	Hardware
328	1	--
328	2	--
328	3	--
328	--	100
328	--	200
328	--	300

PRODUCTO

Nº Empleado	Lenguaje	Hardware
328	1	100
328	1	200
328	1	300
328	2	100
328	2	200
328	2	300
328	3	100
328	3	200
328	3	300

COMPRIMIDO

Nº Empleado	Lenguaje	Hardware
328	1	100
328	2	200
328	3	300

Otra manera de resolver el problema, es a través de la creación de un atributo no existente hasta el momento que indique el tipo de conocimiento.

Nº Empleado	Tipo	Conocimiento
328	L	1
328	L	2
328	L	3
328	H	100
328	H	200
328	H	300

En los ejemplos planteados, la clave de la entidad, será la concatenación de **EMPLEADO**, **LENGUAJE** y **HARDWARE**, y para el último caso, será **EMPLEADO**, ***Tipo***, ***Conocimiento*** **(LENGUAJE ó HARDWARE)**. Nótese que para el último caso, el atributo conocimiento tiene dos significados distintos, dependiendo éste del valor que toma otro atributo (*tipo*).

Si el conocimiento que se pretende representar, no está relacionado (el del software con el del hardware), para normalizar se impone la descomposición.

Nº Empleado	Lenguaje
328	1
328	2
328	3

Nº Empleado	Hardware
328	100
328	200
328	300

En donde la clave, estará dada para cada uno por **EMPLEADO**, **LENGUAJE** y **EMPLEADO**, **HARDWARE** respectivamente.

Pero, si se desea mantener reflejado el conocimiento de un empleado sobre un determinado software que corre sobre determinado hardware, la descomposición anterior no permite reorganizar la información inicial ya que existe una dependencia de valor múltiple (no está en 4FN). Refiriéndose al ejemplo anterior, si el empleado 328 conoce el lenguaje 1 que corre sobre el hardware 200, y el lenguaje 3 que corre también sobre el lenguaje 200, resulta claro que la descomposición anterior no permite mostrar tal situación, es por ello que en estos casos, para llegar a la 4FN se hace necesario:

Nº Empleado	Lenguaje	Hardware
328	1	200
328	3	200

En donde la clave estará dada por la concatenación de **EMPLEADO, LENGUAJE** y **HARDWARE** sin poderlo descomponer, ya que está en 4FN.

5.13.2 Quinta forma normal (5FN)

Valores múltiplemente dependientes, no admiten descomposición. Por ejemplo:

- **EMPRESA *fabrica* PRODUCTO**
- **AGENTE *representa* EMPRESAS**
- **AGENTE *venden determinados productos de* EMPRESA**

Considérese el siguiente ejemplo:

Agente	Empresa	Producto
Sánchez	FORD	Auto
Sánchez	FIAT	Camión
García	FORD	Camión

Si se intenta descomponer esta tabla de relación, quedará**:**

Agente	Empresa
Sánchez	FORD
Sánchez	FIAT
García	FORD

Agente	Producto
Sánchez	Auto
Sánchez	Camión
García	Camión

Empresa	Producto
FORD	Auto
FIAT	Camión
FORD	Camión

Combinando nuevamente para obtener la información original, quedará:

Fila incorrectamente creada →

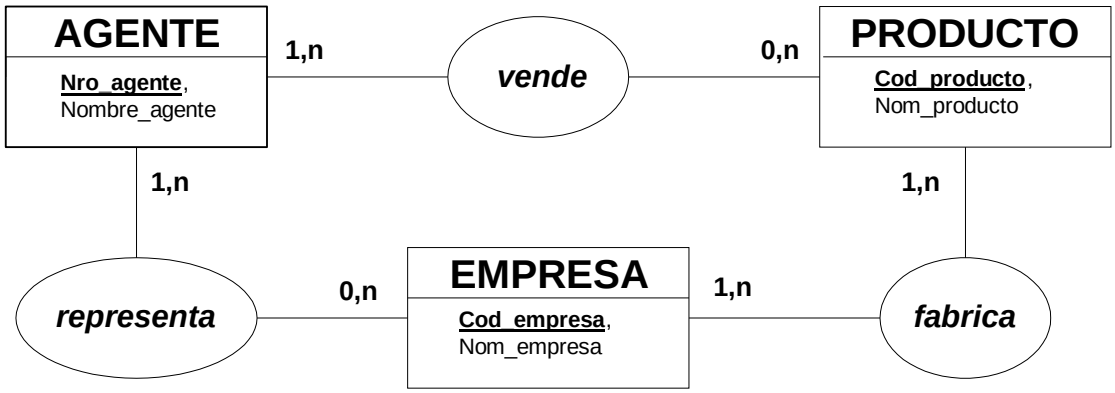
Agente	Empresa	Producto
Sánchez	FORD	Auto
Sánchez	**FORD**	**Camión**
Sánchez	FIAT	Camión
García	FORD	Camión
García	FORD	Auto

Se concluye entonces, que para descomponer, debe respetarse la restricción SIMÉTRICA:

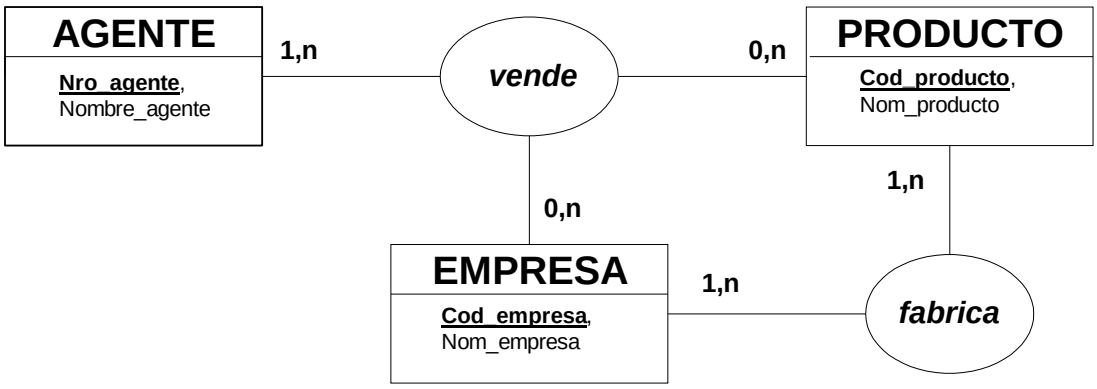
el AGENTE **A** VENDE EL PRODUCTO **P**
y el AGENTE **A** REPRESENTA A LA EMPRESA **E**
y la EMPRESA **E** PRODUCE EL PRODUCTO **P**

restricción:

el AGENTE **A** VENDE EL PRODUCTO **P** PARA LA EMPRESA **E**



Nótese que en el MCD graficado, no hay manera de poder determinar los productos de qué empresa son los que vende el agente. Precisamente el problema radica en la existencia de una dependencia de valores múltiples. Consecuentemente, éste MCD no está en 5FN. El MCD quedará de la siguiente forma:



El nudo del problema se centra en la relación ***VENDE*** que es donde la dependencia múltiple no admite descomposición (nuevamente se ve que la forma normal se refiere a una relación). Concluyendo, no es posible descomponer una relación cuya DIMENSIÓN ES MAYOR O IGUAL QUE TRES.

5.14 Construcción de modelos conceptuales de datos

Se explicará el método con la ayuda de un ejemplo. El sistema de información contiene esencialmente las propiedades que figuran en las boletas de pedido. La apariencia del formulario de pedido es del tipo:

MCD Intermediarios					
			Boleta de pedido Nº **0000-0000125683**		
			Fecha: **…. / …. / 20…**		
	Cliente:	….. - ……..……......			
F	***Domicilio:***	…… Nº …..	***Localidad:***	……. - ……	
I					
C	***Proveedor:***	….. - ……..……......			
H	***Domicilio:***	…… Nº …..	***Localidad:***	……. - ……	
unl					

Cód.	***Descripción***	***Cantidad***	***Precio Unitario***	***Importe***
			TOTAL	

Recopilación de la información

Después de la recopilación de información relativa al sistema de información existente, se reúnen todos los documentos utilizados, así como las descripciones de los diversos ficheros actualmente en uso. El siguiente paso, consistirá en la determinación y especificación clara de

las reglas de gestión:

Regla 1: Un cliente puede tener ninguna o muchas boletas de pedido.
Regla 2: Una boleta corresponde a un único cliente.
Regla 3: Un pedido tiene uno o varios artículos. Al menos tendrá uno.
Regla 4: Un artículo solamente puede estar una vez en un pedido.
Regla 5: Una boleta de pedido se pasa a solamente un proveedor (que no siempre es el mismo para un cliente dado).

En caso de tratarse de un sistema manual, no existirá todavía una gran utilización de codificaciones. Se supondrá en estos casos que existen códigos para identificar las entidades evidentes. Por ejemplo el Código de Cliente para **CLIENTE**, y el Código de Proveedor para **PROVEEDOR**, el Código de Localidad para **LOCALIDAD**.

Construcción del diccionario de datos

A partir de todos los ítemes de datos que se encuentran, se construye el diccionario de datos, en el que se deberá consignar:

- **Nombre del atributo**: es la etiqueta con la que se identificará a la propiedad representada.
- **Significado**: descripción relativa al uso del atributo.
- **Tipo**: clasificación del ítem de dato según sea numérico, carácter o fecha (en el marco de un compromiso asumido al trabajar con un lenguaje de programación que use sistemas de archivos o de una base de datos en particular, aparecerán nuevos tipos de datos). Los tipos de datos que tengan las mismas características tales como limitantes y se considere que pertenecen a una misma clase, pueden conformar tipos de datos de usuario conocidos además como dominios.
- **Longitud**: es la cantidad de caracteres o espacios que ocupa.
- **Naturaleza**: permite indicar en principio si el atributo es elemental, concatenado o calculado. En caso de ser concatenado, debe especificarse cómo está conformado, y si es calculado, especificar la forma de obtenerlo. Por otra parte, permite indicar si se trata de un atributo de tipo movimiento, filiación o situación.
- **Regla de cálculo**: permite indicar las validaciones que tiene cada atributo, valores admisibles, lista de valores posibles, condiciones de integridad, forma de calcularlo, formato, etc.

Seguidamente se presenta el diccionario de datos que surge del análisis del formulario de boleta de pedido.

Nombre	Significado	Tipo	Long	Naturaleza		Regla de Cálculo
Nro_pedido	Es la numeración que permite identificar una boleta de pedido.	N	10	E	M	Entero no nulo y mayor que cero. Formato: ##########
Fecha_pedido	Es la fecha en que se confecciona un pedido.	F	10	E	M	No nulo. Formato dd/mm/aaaa
Cod_cliente	Es la clave de identificación de un cliente.	N	5	E	FI	***A crear. Entero no nulo y mayor que cero.***
Nom_cliente	Es el nombre o denominación de un cliente.	A	30	E	FI	No nulo.
Domi_cliente	Es el domicilio de residencia de un cliente.	A	45	CO	FI	Calle_cliente + Nro_domicli
Calle_cliente	Es la calle del domicilio de residencia de un cliente.	A	35	E	FI	
Nro_domicli	Es el número del domicilio de residencia de un cliente.	A	10	E	FI	
Cod_locali_cliente	Es el código postal de la localidad de residencia de un cliente.	N	5	E	FI	Entero no nulo y mayor que cero.
Nom_locali_cli	Es el nombre de la localidad de residencia de un cliente.	A	30	E	FI	No nulo.
Cod_provee	Es el código de identificación de un proveedor.	N	5	E	FI	***A crear. Entero no nulo y mayor que cero.***
Nom_provee	Es el nombre o denominación de un proveedor.	A	30	E	FI	No nulo.
Domi_provee	Es el domicilio de residencia de un proveedor.	A	45	CO	FI	Calle_provee+ Nro_domiprovee
Calle_provee	Es la calle del domicilio de residencia de un proveedor.	A	35	E	FI	
Nro_domiprovee	Es el número del domicilio de residencia de un proveedor.	A	10	E	FI	
Cod_locali_provee	Es el código postal de la localidad de residencia de un proveedor.	N	5	E	FI	Entero no nulo y mayor que cero.
Nom_locali_provee	Es el nombre de la localidad de residencia de un proveedor.	A	30	E	FI	No nulo.
Cod_Articulo	Es el código de identificación de un artículo.	A	5	E	FI	No nulo. Formato: un caracter + 4 dígitos
Nom_Articulo	Es el nombre o denominación de un artículo.	A	30	E	FI	No nulo.
Cantidad	Es la cantidad de artículos solicitada en una línea de la boleta de pedido.	N	5	E	M	Entero no nulo y mayor que cero.
Precio_unitario	Es el precio unitario de venta de un artículo.	N	10,2	E	FI	No nulo y mayor que cero. Formato: ocho dígitos para la parte entera y dos para la decimal.
Importe_linea	Es el costo de un artículo con relación a la cantidad solicitada que figura en cada línea de detalle de la boleta de pedido.	N	10,2	CA	M	***Precio_unitario** * **Cantidad***. No nulo y mayor que cero. Formato: ocho dígitos para la parte entera y dos para la decimal.

Nombre	Significado	Tipo	Long	Naturaleza		Regla de Cálculo
Total_pedido	Es el importe total que figura en la boleta de pedido.	N	10,2	CA	M	Sumatoria de ***Importe_linea*** No nulo y mayor que cero. Formato: ocho dígitos para la parte entera y dos para la decimal.

Referencias

Tipo:
A – Alfabético
F – Fecha
N – Numérico

Naturaleza:
E – Elemental
CO – Concatenado
CA – Calculado
FI – Filiación
SI – Situación
M – Movimiento

Si una propiedad se emplea en diferentes utilizaciones, habrá que considerar en principio, que se trata de propiedades distintas. Una vez establecidas las entidades corresponderá determinar si se deben reunir en una sola o tienen significado distinto. En el diccionario, figura el código postal y en nombre de localidad en dos oportunidades, una haciendo referencia al cliente y la otra al proveedor.

Depuración del diccionario de datos

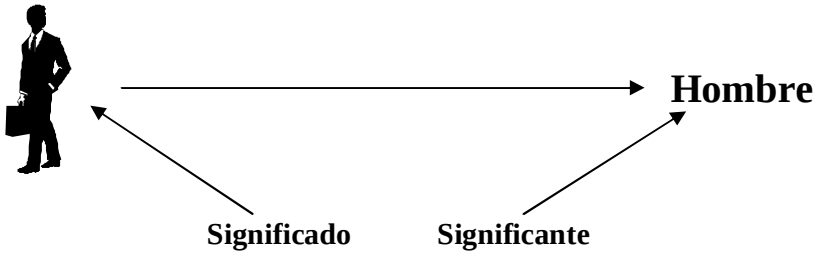
Eliminar los atributos calculados que se puede determinar a partir de otros mediante alguna regla de transformación explicitada, como ***Importe_linea*** **y** ***Total_pedido*** ya que el primero surge del producto entre la ***cantidad * precio_unitario*** y el segundo de la sumatoria de éstos últimos.

Eliminar los atributos concatenados como ***Domi_cliente*** **y** ***Domi_provee*** ya que se utilizan en su lugar los que se especificaron como elementales.

La confusión en la definición de los datos, puede surgir de la relación que se establece entre los significados y sus significantes.

El significado es la conceptualización que se tiene del objeto abstracto o concreto que se quiere representar o cualificar.

El significante es la palabra o signo empleado para representar un significado.



Los sinónimos, son dos significantes distintos para un mismo significado. Por ejemplo, las propiedades *Dirección_cliente* y *Domicilio_cliente* representan el mismo significado pero tienen dos nombres de atributos (significantes) distintos.

La polisemia, es un mismo significante para dos significados distintos (homónimos). Por ejemplo, si se tiene la propiedad *Dirección* que se refiere a la dirección del cliente dentro de la entidad cliente, a la dirección del proveedor dentro de la entidad proveedor, etc.

Para depurar la lista de datos, deben en primer lugar ser eliminados los sinónimos y las polisemias. Puede ocurrir que cuando se comienza a trabajar, no se detecten diferencias y algunos atributos estén mal definidos. Conforme se avanza en el diseño del modelo, tales diferencias se ponen de manifiesto rápidamente.

En el diccionario de datos, se dará un nombre para todos y cada uno de los tipos de datos y se eliminarán los sinónimos y polisemias. Para el caso del ejemplo, el atributo ***Cod_locali_cliente*** es sinónimo de ***Cod_locali_provee*** y consecuentemente ***Nom_locali_cli*** es sinónimo de ***Nom_locali_provee***; por hacer referencia simplemente a **LOCALIDAD**. Para obtener un nombre más acorde, estos grupos serán reemplazados como ***Cod_locali*** y ***Nom_locali*** siendo las características, las mismas a las ya definidas.

Determinación de las entidades y las relaciones

Una manera natural y semi intuitiva, basada en el conocimiento de la realidad que se modela, puede ordenarse según los siguientes pasos:

1. Se buscan las propiedades de la lista que puedan ser identificativos de las entidades. Se realizan los muestreos necesarios hasta obtener las entidades que van a configurar el MCD.

2. Se describen esas entidades, asignando las propiedades que forman el resto de la descripción de las mismas.

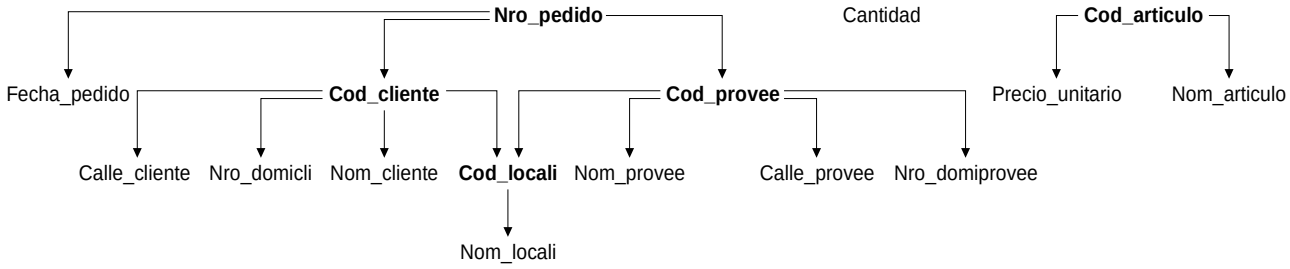
3. Las propiedades que queden en la lista, y que no se han atribuido a ninguna entidad, pertenecerán a relaciones (con su correspondiente colección).

El proceso descrito, se puede formalizar a través de lo que se conoce como grafo de dependencias funcionales, o su correspondiente matriz.

Grafo de dependencias funcionales

Se extrae del diccionario de datos la lista de las propiedades que no están ni concatenadas ni son calculadas. En el ejemplo propuesto, se rechazan las propiedades: ***Domi_cliente***, ***Domi_provee*** (*concatenadas*), ***Importe_linea*** y ***Total_pedido*** (*calculadas*).

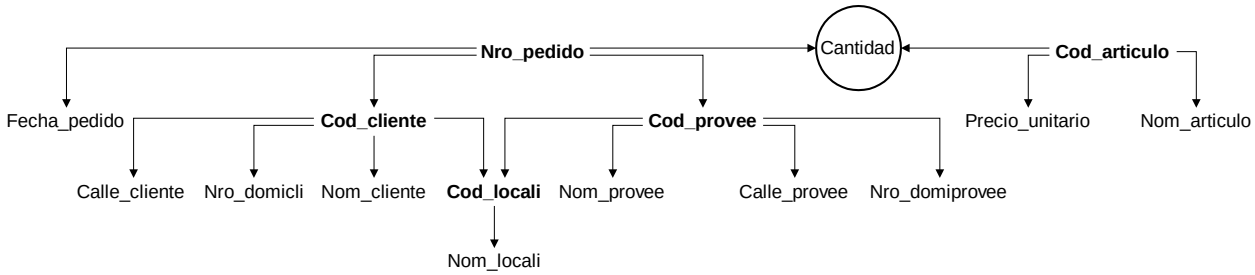
Se establece luego la Lista de Dependencias Funcionales cuyo dominio de partida no contiene más que una sola propiedad no concatenada a partir del examen de los documentos y de los identificativos propuestos. Esta lista de **df** se puede visualizar mediante un grafo como el siguiente:



Nótese que la propiedad ***Cantidad*** ha quedado aislada.

Si quedan propiedades aisladas, se buscan dependencias funcionales que conduzcan a estas propiedades a partir de la concatenación de propiedades. Si no se encuentra ninguna, tal propiedad se continúa dejando aislada. En este caso, se utiliza la dependencia:

Nro_pedido + Cod_articulo ⟶ ***Cantidad***



Se intentará asegurar, siempre, que las propiedades aisladas no se correspondan a entidades aisladas (casos de tablas propias del sistema de aplicación), para que las que habría que suponer una clave de identificación que permita añadir las dependencias que faltan. Por ejemplo, si existiese la propiedad ***Cod_articulo***, se estaría entonces en presencia de dos propiedades aisladas: ***Precio_unitario*** y ***Nom_articulo***, claramente afectadas a la entidad **ARTÍCULO**. Se establecería entonces el identificativo ***Cod_articulo*** (u otro nombre), que aportaría nuevas dependencias lo que permitiría concretamente llegar a ***Precio_unitario*** y ***Nom_articulo***. Sería preciso entonces, volver a comenzar a partir del grafo inicial.

Si el grafo obtenido conduce a ciclos, se elimina esta anomalía suprimiendo una dependencia funcional. En caso de existir transitividades, éstas deben ser eliminadas.

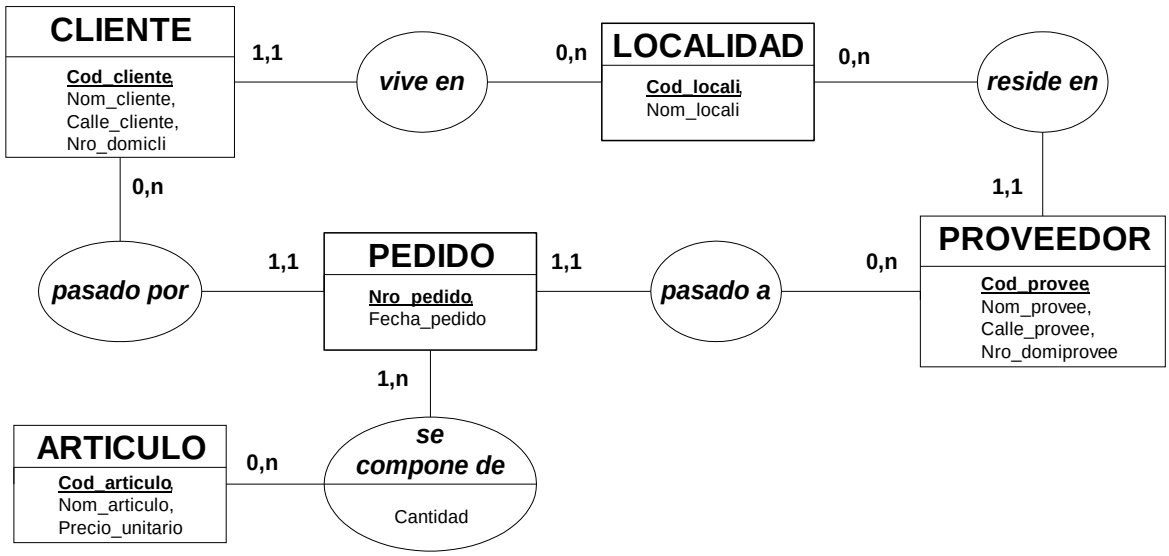
Establecimiento del MCD

Los arcos terminales obtenidos a partir de las propiedades elementales, definen las ENTIDADES. Los atributos que están en el origen de cada uno de esos arcos, serán los identificativos de tales entidades. Aplicando estas definiciones al grafo, se tiene que:

ENTIDAD	Atributo	Atributo	Atributo	Atributo
PEDIDO	***Nro_pedido***	Fecha_pedido		
CLIENTE		***Cod_cliente***	Nom_cliente	
			Calle_cliente	
			Nro_domicli	
LOCALIDAD			***Cod_locali***	Nom_locali
Relación	*Cantidad*	***Cod_provee***	Nom_provee	
			Calle_provee	
			Nro_domiprovee	
LOCALIDAD (R)			***Cod_locali* (R)**	
ARTÍCULO	***Cod_articulo***	Nom_articulo		
		Precio_unitario		

Los arcos restantes, establecen las relaciones. Las propiedades no aisladas restantes se afectan a las relaciones. Aquellas que no pueden ser asignadas a relaciones, constituirán entidades aisladas.

Las reglas de gestión, deben permitir encontrar las cardinalidades. De esta manera, el modelo quedará:



Es necesario, por último, asegurar que se cumplen las reglas de verificación, normalización y descomposición. Todo ello, ocurre en el ejemplo anterior.

Cuantificación del MCD

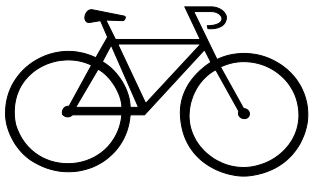
Finalmente, se completa la información del modelo, adicionando fichas con datos de filiación de las entidades incluyendo la cuantificación de los elementos recogidos en las entrevistas. Esto dará una idea del tamaño de modelo completo. Se realizan separadamente tablas de cuantificación para las entidades y para las relaciones.

5.15 MDC extendido

Hasta el momento se han trabajado con elementos conceptuales de los diagramas entidad-relación que resultan genéricos para todos los autores que se explayan sobre el tema. A continuación se agregarán nuevos conceptos primitivos que permitirán agregar expresividad a los modelos conceptuales.

Abstracciones en el diseño del modelo conceptual de datos

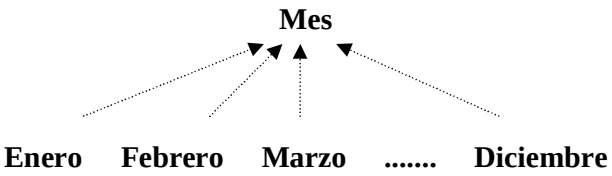
La abstracción es un proceso mental que se aplica al seleccionar algunas características y propiedades de un conjunto de objetos y excluir otras no pertinentes, vale decir que se hace una abstracción al fijar la atención en las propiedades consideradas esenciales de un conjunto de cosas y desechar sus diferencias. Si se considera por ejemplo el siguiente gráfico:



El concepto de bicicleta puede verse como resultado de un proceso de abstracción, lo que hace excluir todos los elementos o detalles de la estructura de la bicicleta (cadena, pedales, frenos, manubrio, etc.) y todas las posibles diferencias entre bicicletas. Comúnmente se asocia un nombre con cada abstracción. El dibujo es una representación de esta abstracción. Otra representación sería una descripción en castellano del mismo dibujo. En el diseño conceptual de datos, se usan tres tipos de abstracciones: clasificación, agregación y generalización.

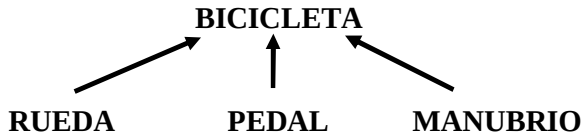
5.15.1Abstracción de clasificación

La abstracción de clasificación, se usa para definir un concepto como una *clase* de objetos de la realidad, caracterizados por propiedades comunes. Por ejemplo, se tiene que el concepto *bicicleta* es la clase cuyos miembros son todas bicicletas (la bicicleta roja, la bicicleta de Claudio, etc.). De igual manera el concepto *mes* es la clase cuyos miembros son *Enero, Febrero, ..., Diciembre* como se muestra en la figura. Se representa gráficamente la clasificación como un árbol de un nivel, que tiene como raíz la clase y como hojas los elementos de la clase. Las ramas del árbol se representan por líneas discontinuas. Cada rama del árbol indica que un nodo hoja es un miembro de la clase que representa la raíz.

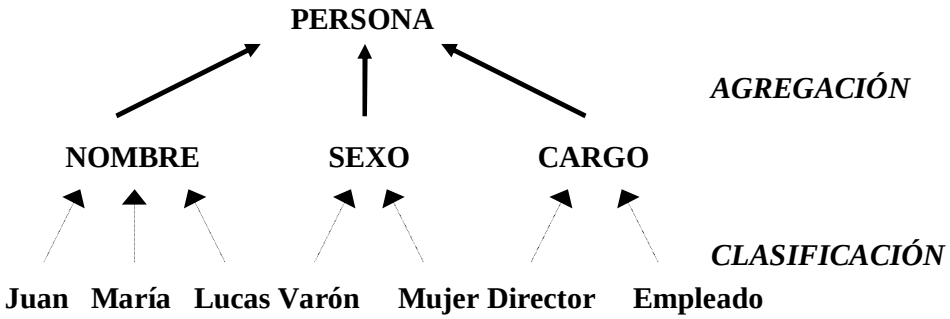


5.15.2 Abstracción de agregación

Una abstracción de agregación define una nueva clase a partir de un conjunto de (otras) clases que representan sus partes componentes. Se aplica esta abstracción cuando, partiendo de las clases rueda, pedal, manubrio, etc., se forma la clase BICICLETA. La abstracción por agregación se representa por un árbol de un nivel en el cual todos los nodos son clases; la raíz representa la clase creada por agregación de las clases representadas por las hojas. Cada rama del árbol indica que una clase hoja es una parte de la clase representada por la raíz. Para distinguir la agregación de la clasificación las ramas dirigidas están representadas por líneas gruesas que van de los componentes a los objetos agregados.

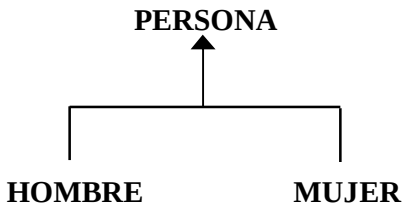


La clasificación y la agregación son las dos abstracciones básicas utilizadas para construir estructuras de datos dentro de los modelos y de muchos lenguajes convencionales de programación. La clasificación es el procedimiento utilizado cuando, partiendo de elementos individuales de información, se identifican tipos de atributos. La agregación es el procedimiento mediante el cual se reúnen tipos de campos relacionados en grupos como por ejemplo tipos de registros.



5.15.3 Abstracción de generalización

Una abstracción de generalización define una relación de subconjunto entre los elementos de dos o más clases. Por ejemplo, la clase **VEHICULO** es una generalización de la clase **BICICLETA**, ya que todas las bicicletas son vehículos. Asimismo, la clase **PERSONA** es una generalización de las clases **HOMBRE y MUJER**.



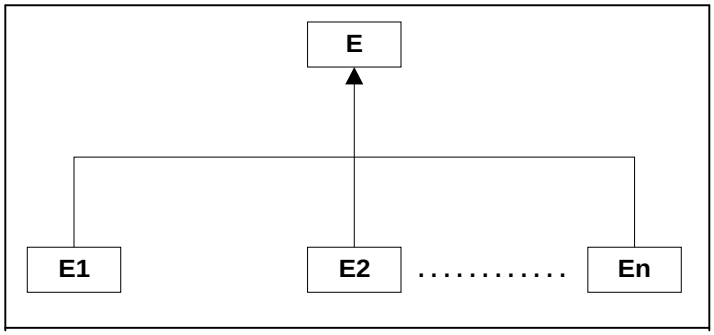
Cada generalización se representa con árbol de un nivel, en el que todos los nodos son clases, con la clase genérica como raíz y las clases subconjunto como hojas; cada rama del

árbol expresa que una clase hoja es un subconjunto de la clase raíz. Para distinguir la generalización de otras abstracciones, se usa una flecha sencilla apuntando hacia la raíz. La abstracción de generalización, a pesar de ser muy común e intuitiva, no se usa en muchos modelos de datos. Sin embargo, es muy útil por su cualidad fundamental de herencia: en una generalización, todas las abstracciones definidas para la clase genérica son heredadas por las clases subconjunto. Dentro del concepto de abstracción de generalización, se encuentra embebido el de cobertura. La cobertura se clasifica en función de las cardinalidades mínima y máxima, determinando si es total o parcial o si es exclusiva o superpuesta. Es total (t), si cada elemento de la clase genérica corresponde al menos a un elemento de las clases subconjunto; es parcial (p) si existe algún elemento de la clase genérica que no corresponde a ningún elemento de las clases subconjunto. Es exclusiva (e) si cada elemento de la clase genérica corresponde, a lo sumo, a un elemento de las clases subconjunto; es superpuesta (s) si, al contrario, existe algún elemento de la clase genérica que corresponde a elementos de dos o más clases subconjunto diferentes.

Las tres abstracciones son independientes: ninguna de ellas puede describirse en función de las otras, y cada una de ellas proporciona un mecanismo diferenciado en el proceso de estructuración de la información.

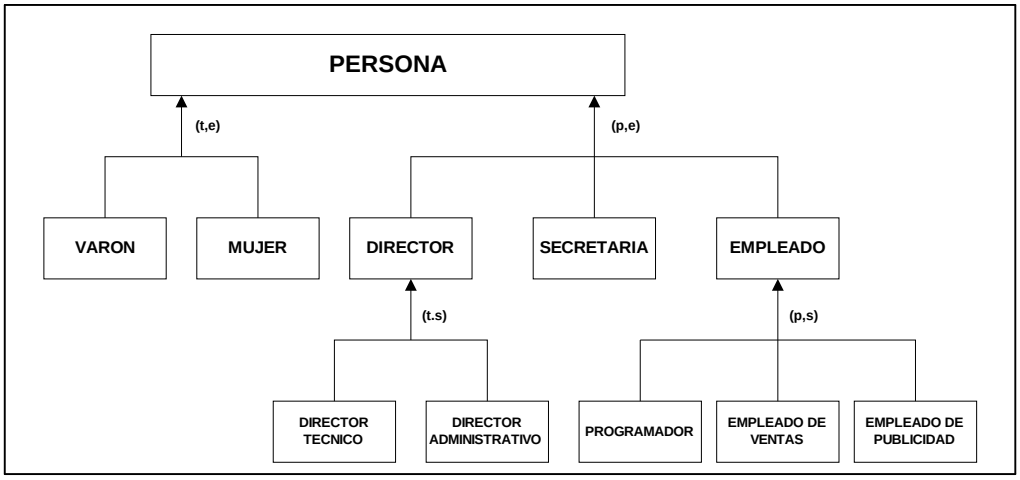
5.15.4 Jerarquía de generalización

En el modelo entidad–relación es posible establecer jerarquías de *generalización* entre las entidades. Una entidad **E**, es una generalización de un grupo de entidades $\mathbf{E_1, E_2, ..., E_n}$, si cada objeto de las clases $\mathbf{E_1, E_2, ..., E_n}$, es también un objeto de la clase **E**. Una generalización en el modelo entidad – relación, expresa la abstracción de generalización expuesta anteriormente. Nótese que solamente se mostrará en la representación gráfica, el bloque de entidad y el nombre de ella. La representación esquemática será la siguiente:

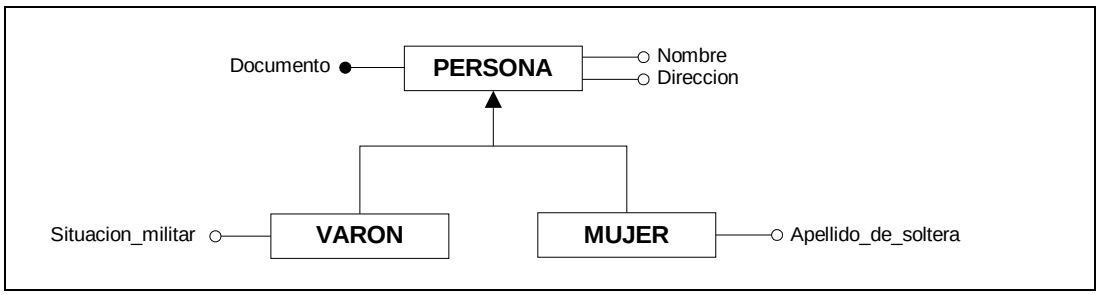


En la figura, la flecha indica la entidad generalizada.

Cada entidad puede participar en múltiples generalizaciones, posiblemente en el papel de entidad genérica con respecto a una generalización y en el papel de entidad subconjunto con respecto a otra generalización. La siguiente figura, presenta una jerarquía de generalización compleja para la entidad **PERSONA**. Lo opuesto a la generalización se denomina especialización.



La propiedad fundamental de la abstracción de generalización es que todas las propiedades de la entidad genérica son heredadas por las entidades subconjunto. En términos del modelo entidad - relación, esto significa que cada atributo, relación o generalización definido para la entidad genérica, será heredado automáticamente por todas las entidades subconjunto de la generalización. Esta propiedad es importante, porque permite construir jerarquías de generalización estructuradas. Considerando el siguiente gráfico, la propiedad de herencia establece que los atributos *nombre* y *dirección* de **PERSONA** son también atributos de **VARON** y **MUJER**; luego, pueden ser eliminados de las entidades subconjunto, simplificando el esquema.

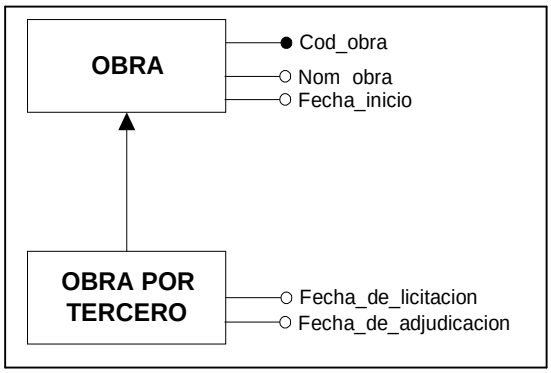


Si bien la simbología utilizada difiere de la propuesta anteriormente, su significado no cambia en absoluto.

5.15.5 Subconjuntos

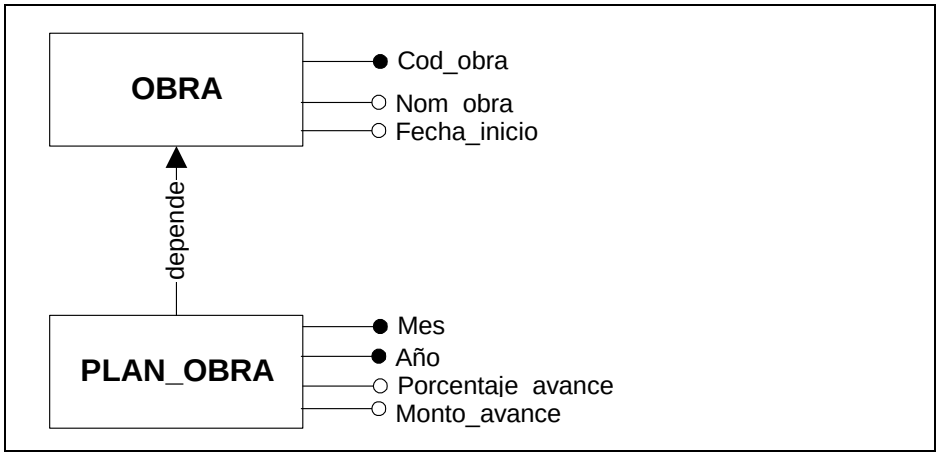
Un subconjunto es un caso particular de jerarquía de generalización, con una sola entidad subconjunto. Se tratarán por separado los subconjuntos porque la cobertura de un subconjunto, es claramente parcial y exclusiva y no necesita definirse. Se representan los subconjuntos con una flecha que une la entidad genérica a la entidad subconjunto y apunta hacia la entidad genérica, como lo indica la figura. Suponga por ejemplo la existencia de la entidad **OBRA** que tendrá como atributos (*Cod_obra, Nom_obra, Fecha_inicio*), pero hay un grupo de obras que son realizadas por empresas privadas y adjudicadas a través de un proceso de licitación que necesariamente requieren registrar además de los atributos comunes, la *Fecha_de_licitación* y la *Fecha_de_adjudicación*. Surge entonces la necesidad de crear una especialización (**OBRA POR TERCERO**) de la entidad **OBRA** que contenga tales

propiedades.



5.15.6 Entidades dependientes y entidades débiles

Existen casos de características similares a las de subconjuntos, con la diferencia que las entidades no son parte de otra de jerarquía superior y de la que heredan sus atributos, sino que, poseen atributos propios pero no identidad propia única (identificativo único) y para ello, requieren heredarlo de la entidad de nivel superior de la que son totalmente dependientes. Por ejemplo, la planificación de ejecución (*Porcentaje_avance, Monto_avance*) de una obra, es totalmente dependiente de la obra en cuestión, siendo el identificativo para cada ocurrencia, el *Cod_obra*, *Mes* y *Año*. De acuerdo a la teoría desarrollada durante el curso, se ha establecido que puede haber atributos asociados a entidades y a relaciones. Existe gran cantidad de software de diseño asistido en los que no resulta posible asociar atributos a una relación, y además, se limitan a trabajar pura y exclusivamente con relaciones binarias. Para salvar esta situación, se suelen crear entidades totalmente dependientes también denominadas entidades débiles que no poseen un atributo identificativo, sino que el identificativo estará dado por la concatenación de las claves de las entidades que se vinculan con él (normalización de relaciones). En estos casos, esta entidad será totalmente dependiente de al menos dos entidades "fuertes".



5.16 Estrategias de diseño para los modelos conceptuales de datos

La creación de un modelo conceptual de datos, es un proceso incremental: la percepción de la realidad se refina y enriquece de forma progresiva, y el esquema conceptual se desarrolla gradualmente. Las distintas formas de encarar el estudio y comenzar el diseño, son las que dan el nombre a las estrategias que se mencionan a continuación:

5.16.1 Estrategia descendente

Se parte de un esquema que contiene abstracciones de alto nivel y luego se aplican refinaciones descendentes sucesivas. Se puede comenzar por especificar unos cuantos tipos de entidades de alto nivel; luego al especificar sus atributos, se dividen en tipos de entidades y de relaciones de menor nivel. El proceso de especialización para refinar un tipo de entidades convirtiéndolo en subclases, es otro ejemplo de estrategia de diseño descendente.

5.16.2 Estrategia ascendente

Se parte de un esquema que contiene abstracciones básicas, y luego se combinan o se les agregan otras abstracciones. Por ejemplo se puede comenzar con los atributos y agruparlos en tipos de entidades y relaciones. Conforme avanza el diseño se podrían agregar nuevas relaciones entre tipos de entidades. El proceso de generalizar subclases para obtener clases generalizadas de más alto nivel es otro ejemplo de estrategia de diseño ascendente.

5.16.3 Estrategia centrífuga

Este es un caso especial de estrategia ascendente, en la que la atención se concentra en un conjunto principal o núcleo de conceptos que son los más evidentes. A continuación el modelado se extiende hacia afuera al considerar conceptos nuevos en las cercanías de los ya existentes. Por ejemplo se podría especificar en el esquema unos cuantos tipos de entidades obvias y continuar agregando otros tipos de entidades y de relaciones vinculadas con ellos.

5.16.4 Estrategia mixta

En vez de seguir una estrategia específica durante todo el diseño, los requerimientos se dividirán según una estrategia descendente, y se diseñará una parte del modelo para cada partición de acuerdo con una estrategia ascendente. Por último, se combinarán las diferentes partes del modelo.

5.17 Conclusión

Los modelos de datos son elementos que permiten describir la realidad. El bloque de construcción elemental común a todos los modelos de datos, es una pequeña colección de mecanismos de abstracción primitivos. Las siguientes consideraciones justifican acabadamente la importancia del enfoque conceptual:

- El diseño conceptual no se ayuda mucho de herramientas automáticas.

- El diseñador asume total responsabilidad sobre el proceso de entender y transformar los requerimientos en esquemas conceptuales.

- A partir de la primera conceptualización, muchos sistemas de bases de datos ofrecen herramientas para la creación rápida de prototipos, usando lenguajes de cuarta generación de aplicaciones, formatos de pantallas e informes.

- Estas herramientas pueden ser usadas directamente por no profesionales para desarrollar bases de datos simples y facilitan el trabajo a los creadores profesionales de bases de datos.

- El diseño conceptual, es la fase más crucial del diseño de bases de datos, y el desarrollo posterior de la tecnología de bases de datos, no cambiará esta situación.

- Aun si se supone que el diseño conceptual está dirigido por un profesional, se alcanzan resultados satisfactorios sólo mediante la cooperación con los usuarios de las bases de datos, quienes tienen que describir las necesidades y de explicar el significado de los datos.

- Las características básicas del diseño conceptual y de los modelos conceptuales de datos, son relativamente simples y su entendimiento no requiere mayor conocimiento técnico previo sobre sistemas de bases de datos. De este modo, los usuarios pueden aprender fácilmente lo suficiente sobre diseño conceptual para orientar a los diseñadores en sus decisiones e incluso para diseñar bases de datos simples.

- Una influencia fuerte del usuario final sobre las decisiones de diseño tiene muchas consecuencias positivas:

    - mejora la calidad del esquema conceptual

    - eleva la probabilidad de que el proyecto converja hacia el resultado esperado

    - reduce los costos de desarrollo

- Un gran argumento de fuerza, es su independencia de un fabricante de base de datos en particular. Esta característica genera las siguientes ventajas:

    - La elección del software de base de datos se puede posponer, y el esquema conceptual puede sobrevivir a una decisión tardía de cambiar el software.

    - Si el software de base de datos o los requerimientos de la aplicación cambian, el esquema conceptual puede aún usarse como punto de partida de la nueva actividad de diseño.

- Las diferentes bases de datos, descritas mediante su esquema conceptual, se pueden comparar en un marco homogéneo de trabajo. Esta característica facilita la construcción de sistemas consolidados a partir de varias bases de datos ya existentes y la creación de un diccionario de datos integrado.