SISTEMAS DE BASES DE DATOS - INGENIERÍA INFORMÁTICA

DISEÑO DE BASES DE DATOS RELACIONALES

DISEÑO:

El diseño de bases de datos tiene tres etapas:

1. DISEÑO CONCEPTUAL:

    a. Se basa en los resultados producidos en la etapa de análisis (modelo conceptual y diccionario de datos). El objetivo de la etapa de análisis es entender el dominio del problema. Está relacionado con el conocimiento.
    b. Su objetivo es satisfacer los requerimientos de información del sistema.
    c. Produce un nuevo modelo conceptual y un nuevo diccionario de datos.
    d. El modelo conceptual es:
        i. Independiente del modelo de bases de datos (relacional, orientado a objetos, etc.)
        ii. Independiente de la tecnología (DBMS)

2. DISEÑO LÓGICO:

    a. Se define el modelo de bases de datos a utilizar y se diseña un nuevo modelo. Un modelo lógico basado en el modelo de bases de datos elegido:
    b. El modelo lógico es:
        i. Dependiente del modelo de bases de datos
        ii. Independiente de la tecnología

3. DISEÑO FÍSICO o DE IMPLEMENTACIÓN:

    a. Se selecciona el DBMS a utilizar y se diseña el modelo físico para su implementación
    b. Su objetivo es lograr el mejor rendimiento posible del sistema (hardware-software) y el menor tiempo de respuesta a los usuarios:
    c. El modelo físico es:
        i. Dependiente del modelo de bases de datos
        ii. Dependiente de la tecnología

MODELIZACIÓN:

En general el proceso de diseño, fundamentalmente en el caso de nuevos productos, se basa en el proceso de modelización explicado arriba.

El resultado de la modelización se plasma en un diagrama de entidades y relaciones comúnmente conocido como DER.

En este diagrama se explicitan tres tipos de elementos:

- Entidad: Cualquier objeto material o inmaterial del cual se requiere conocer algún tipo de información.
  Cuando hablamos de entidad, nos referimos a lo que denominamos "tipo", es decir a su definición o descripción en términos generales y no de algún caso particular.
  Por ejemplo, podemos definir la entidad "Empleado" y cuando hacemos mención a ella nos referimos a las características (atributos) que describen a cualquier empleado y no a un empleado particular.

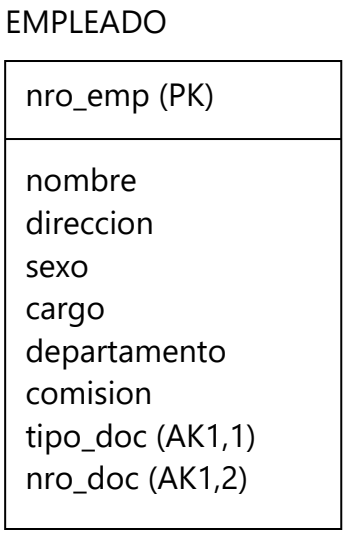
- Atributo: Permite identificar o describir cualidades, características o estados de una entidad. Por ejemplo: "Nro. de legajo" es un atributo de la entidad "Empleado" que permite identificar a cada empleado (una instancia de la entidad). El atributo "Sexo" permite describir una característica de los empleados.

- Asociación: Permite asociar o relacionar instancias de diferentes (aunque no necesariamente) entidades. Por ejemplo, se tiene una relación o asociación entre las entidades "Empleado" y "Departamento" ya que un empleado trabaja en un departamento. También podemos definir una asociación entre la entidad "Empleado" con ella misma al decir que un empleado tiene un jefe que a su vez es empleado.

Tanto el modelo conceptual como el modelo lógico, en general, dan como resultado un DER.

Los CASEs (herramientas que permiten construir modelos) actuales, en general, son utilizados solamente para construir modelos lógicos orientados a uno o varios modelos de bases de datos. Esto hace más dificultoso o, por lo menos, más engorroso, el diseño conceptual. Además, en general, se tiene conocimiento previo del modelo de bases de datos a utilizar. Esto implica que muchos modelos conceptuales se construyan con herramientas para elaborar modelos lógicos, y estén ya orientados a un modelo de bases de datos determinado.

Ejemplo:



En el ejemplo mostrado vemos una entidad con sus atributos.

NOTA: el gráfico tiene una notación particular. Cada CASE tiene una determinada. Aquí utilizaremos la de ERWIN y en particular la metodología de Ingeniería de Información.

Podemos analizar varios aspectos de la misma:

1. Nombre de la entidad: A propósito, la escribimos en singular para realzar la característica de "tipo", pero esto no es un requerimiento.

2. Hay un atributo separado de los demás: El conjunto de atributos que se ubican en la parte superior de la entidad y tienen una aclaración (PK) componen la clave primaria de la entidad. Es decir, que forman uno de los identificadores de la misma.

3. Hay atributos señalados con (AKx,y): Esto indica que forman la clave alternativa "*x*". El valor "*y*" indica el orden de los atributos en el conjunto.
   En el ejemplo: tipo_doc es el primer atributo de la clave alternativa 1 y nro_doc el segundo.
   Recordar que una relación (en este caso una entidad) puede tener más de un identificador. A estos identificadores se les denomina claves candidatas. Una de ellas se elige como clave primaria y las demás son claves alternativas.

   Recomendamos en este punto realizar los ejercicios 1 y 2 de la actividad 1 correspondiente a este módulo.

4. Hay atributos que parecen más que eso: Seguramente a más de uno no le "cayó bien" el atributo departamento, que indica el departamento en el cual trabaja el empleado. La razón es porque ese atributo parece ser más que eso. Parece ser una entidad en sí misma, ya que nos imaginamos que podríamos definir varios atributos de un departamento. Por ejemplo: el nombre, la oficina donde está ubicado, sus teléfonos, etc.

5. ¿Es necesario controlar los valores que se ingresan para cada atributo?: Los valores válidos para un atributo conforman el dominio de donde se basa dicho atributo. Esto define una regla de integridad (regla de integridad de dominio) y es necesario verificar que se cumpla.

   Recomendamos en este punto realizar los ejercicios 3 y 4 de la actividad 1 correspondiente a este módulo.

Ahora, avancemos sobre los conceptos de atributo y asociación.

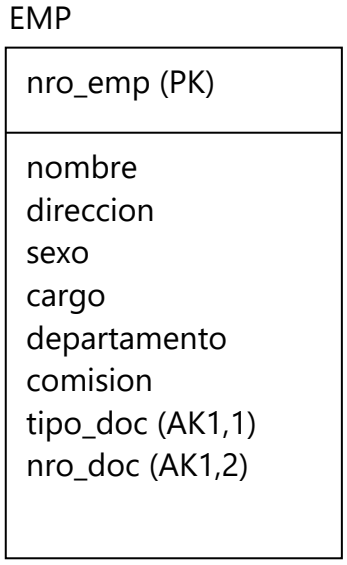
Partamos del supuesto de que en el ejercicio 3 de la actividad 1, hemos elegido la segunda alternativa, es decir, una tabla departamentos con un atributo interno que tiene la función de clave primaria y el nombre del departamento como clave alternativa.

La nueva pregunta que nos hacemos es la siguiente: ¿Será conveniente utilizar un número correlativo interno o un número o código de departamento que el usuario deberá informar?
En este caso se debe analizar lo siguiente:

- ¿El usuario tiene alguna fuente que le brinde información acerca del número o código que deberá ingresar? Por ejemplo: una regla mnemotécnica, una tabla, lista, libro o manual que contenga los valores a ingresar y que haya sido adoptada por la empresa.
- ¿Sería útil que el usuario pueda informar dicho número o código para realizar la selección de un departamento? Es decir, ¿cualquier usuario, en poco tiempo puede aprender a reconocer cada departamento por ese identificador?
- ¿Ese código es estable? (el código podría cambiar por varias razones: reestructuración, su base está relacionada con características que pueden cambiar en el tiempo, etc.)

Si las respuestas a estas preguntas son afirmativas, se puede pensar en un código o número externo. En cualquier otro caso, será conveniente utilizar un número correlativo interno que no será mostrado ni solicitado nunca al usuario.

Para finalizar con el concepto de atributo, recordemos la entidad EMPLEADO, tal como la definimos inicialmente:



Habíamos mencionado lo siguiente: *"Hay atributos que parecen más que eso. Seguramente a más de uno no le "cayó bien" el atributo departamento, que indica el departamento en el cual trabaja el empleado. La razón es porque ese atributo parece ser más que eso. Parece ser una entidad en sí misma, ya que nos imaginamos que podríamos definir varios atributos de un departamento. Por ejemplo: el nombre, la oficina donde está ubicado, sus teléfonos, etc."*

Recomendamos en este punto realizar el ejercicio 5 de la actividad 1 correspondiente a este módulo.

Asociación:

Hasta ahora hemos visto un tipo de asociación. La misma se dio entre dos entidades y se graficó con una línea que une a las mismas más algunos otros símbolos.

Para definir una asociación se deben tener en cuenta los siguientes aspectos:

- Las asociaciones en el modelo lógico siempre se establecen entre dos (y solo dos) entidades, no necesariamente diferentes. Es decir, que una asociación puede realizarse entre una entidad y ella misma. Esto es así, solo en el modelo lógico relacional que es el que nos toca estudiar aquí. Tanto en el modelo conceptual como en otros modelos lógicos basados en otros modelos de bases de datos, esto puede no ser así.

- Las asociaciones siempre se plantean desde dos puntos de vista que corresponden a las dos entidades que se asocian, estableciéndose en cada caso, cual es conceptualmente la relación entre una instancia de una de las entidades (entidad de partida) y las instancias de la otra entidad (entidad de llegada)

Por ejemplo, la asociación entre empleado y departamento vista desde el punto de vista de empleado se puede definir como:

Un empleado trabaja en un departamento

- empleado es la entidad de partida
- un empleado es una instancia de dicha entidad
- trabaja en es la relación conceptual
- departamento es la entidad de llegada
- un departamento es una instancia de dicha entidad

Vista desde el punto de vista de departamento:

Un departamento tiene cero o varios empleados

- departamento es la entidad de partida
- un departamento es una instancia de dicha entidad
- tiene es la relación conceptual
- empleado es la entidad de llegada
- un empleado es una instancia de dicha entidad

Las cardinalidades de la asociación indican cuantas instancias de la entidad de llegada como mínimo (cardinalidad mínima) y como máximo (cardinalidad máxima) están asociadas a una instancia de la entidad de partida

En el ejemplo anterior, para el primer punto de vista (empleado hacia departamento):

- Cardinalidad mínima: 1
- Cardinalidad máxima: 1

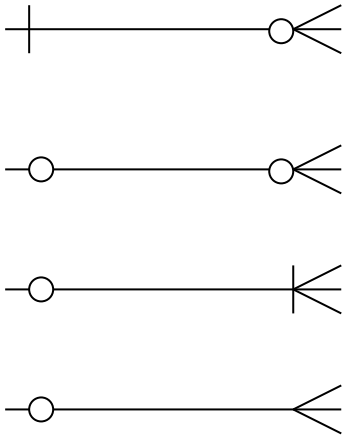
Para el segundo punto de vista (departamento hacia empleado):

- Cardinalidad mínima: 0
- Cardinalidad máxima: muchos

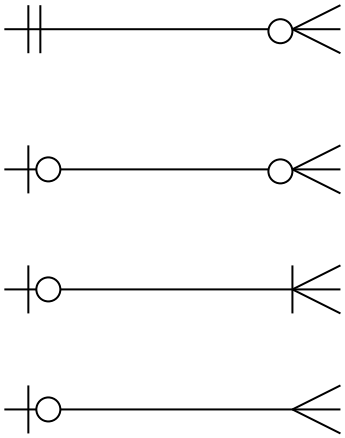
Este tipo de asociación se denomina entonces “De 1 a muchos”. Como la asociación cuya cardinalidad máxima es 1, tiene cardinalidad mínima también 1, entonces a este tipo se le denomina “De 1 a muchos, obligatoria”.

Si la cardinalidad mínima fuera 0, se le denomina “De 1 a muchos, no obligatoria”. Por otro lado, la asociación cuya cardinalidad máxima es muchos, también tiene variantes de acuerdo a la cardinalidad mínima (0: no obligatoria, 1: obligatoria, muchos: obligatoria) aunque no tiene nombres diferentes.
La representación gráfica en los diferentes CASEs pueden ser diferentes. Veamos dos casos:

ERWIN:



POWER DESIGNER:



La implementación de este tipo de asociaciones entre entidades se realiza a través de la propagación de la clave primaria desde la entidad donde la cardinalidad es “uno” hacia la entidad donde la cardinalidad es “muchos”. Esto permite establecer la relación entre las instancias de estas entidades y completar información de una de ellas agregando los atributos de la otra a través de una operación JOIN. Ver la asociación entre empleado y departamento.

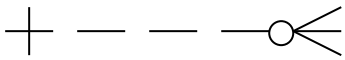
Recomendamos en este punto realizar el ejercicio 1 de la actividad 2 correspondiente a este módulo.

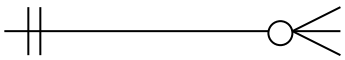
Hay una situación especial que los Case’s permiten visualizar gráficamente y es la siguiente:
Los atributos propagados pueden formar parte de la clave primaria o no en la relación que los recibe.
Mostramos la forma gráfica de diferenciar esta situación en dos CASEs disponibles en el mercado:

ERWIN:

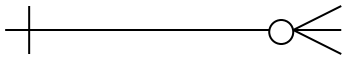
POWER DESIGNER:

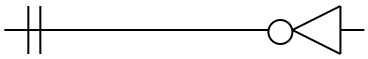
No forman parte:





Forman parte:

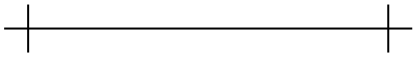




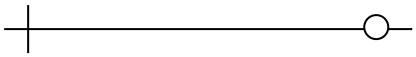
Recomendamos en este punto realizar los ejercicios 2 y 3 de la actividad 2 correspondiente a este módulo.

Continuemos con otros tipos de asociaciones:

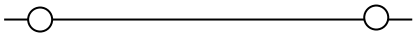
Asociaciones "uno a uno":



Una instancia de una entidad está asociada siempre a una instancia de la otra entidad

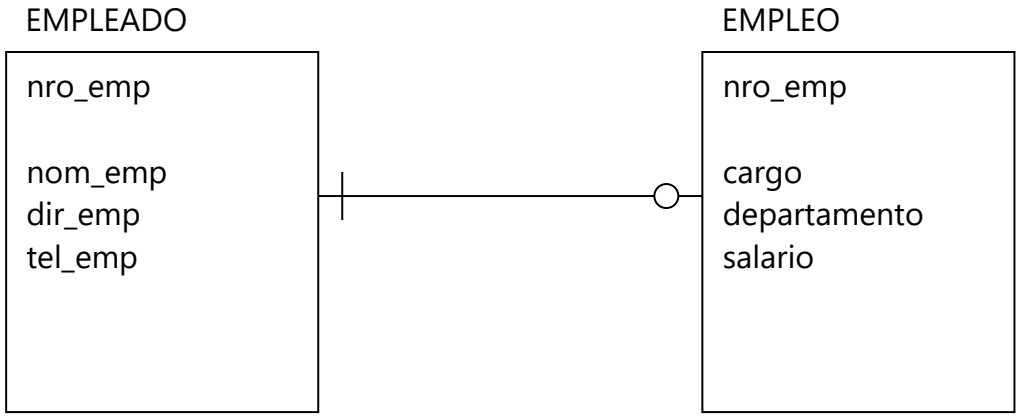


Una instancia de una entidad está asociada a cero o una instancia de la otra entidad



Una instancia de una entidad está asociada a cero o una instancia de la otra entidad y a la inversa

Ejemplo:



Esto se podría ver de la siguiente manera:

nro_emp
nom_emp dir_emp tel_emp **cargo** **departamento** **salario**

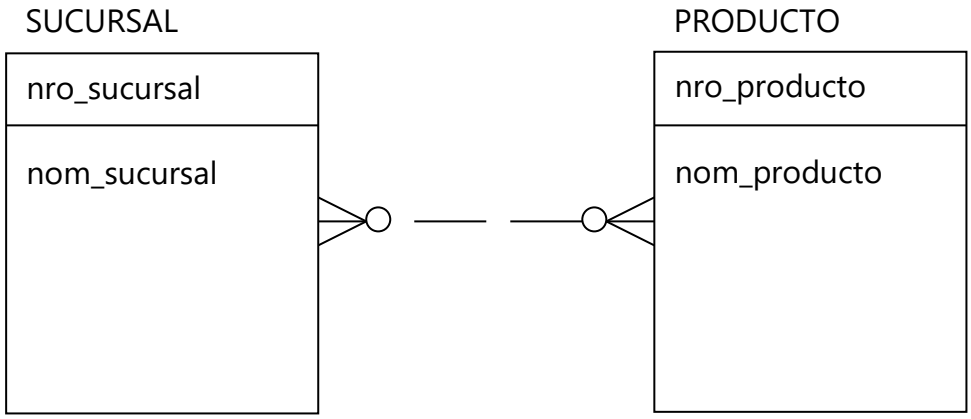
NOTA: los tres atributos en negrita permiten valores nulos.

Es decir, en principio, en el modelo lógico existen pocas asociaciones uno a uno, ya que normalmente los atributos se pueden agrupar en una única entidad. Estos atributos serán obligatorios o no dependiendo si la asociación original era obligatoria o no respectivamente.
En muchos casos, sin embargo, se prefiere establecer este tipo de asociaciones, para facilitar la construcción del modelo físico donde en líneas generales para cada entidad (modelo lógico) se genera una tabla (modelo físico), pero este modelo lógico queda entonces condicionado por el diseño físico y no debiera ser así.

Asociaciones "muchos a muchos":

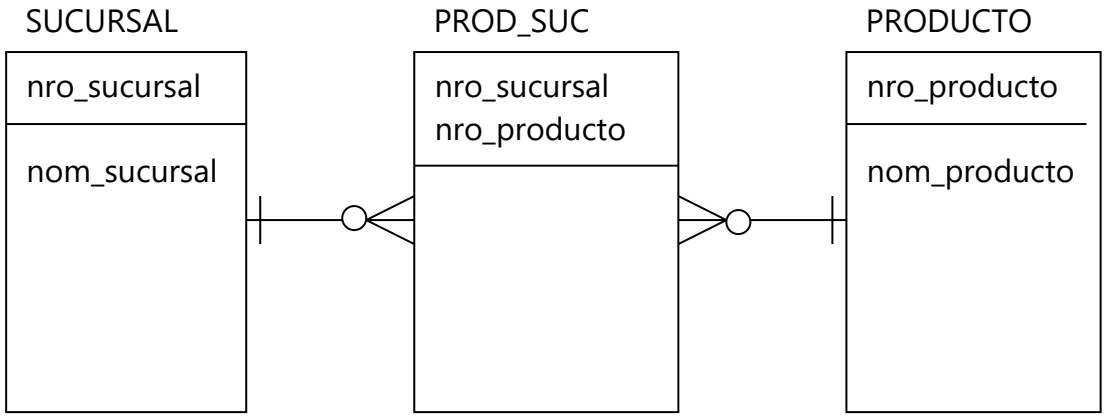
Supongamos, una empresa que tiene varias sucursales, donde, cada una de ellas comercializa varios productos, pero no todos los productos se comercializan en todas las sucursales.

Encontramos entonces que un producto es comercializado en varias sucursales y una sucursal comercializa varios productos. Esto se puede expresar de la siguiente manera:



Este tipo de asociaciones se denomina de “muchos a muchos”. En general, se expresan como asociaciones no obligatorias.
Evidentemente estas asociaciones no pueden propagar atributos a las relaciones originales, ya que estos atributos podrían contener grupos repetitivos de valores, es decir, relaciones no normalizadas.

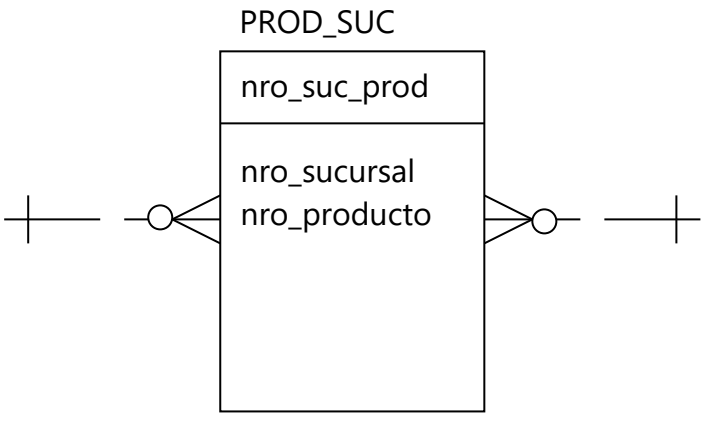
Este tipo de asociación se traslada al modelo físico generando dos tablas (una para cada entidad) y una tercera tabla (intermedia) a la cual se propagan los atributos clave de las anteriores, de la siguiente forma:



En la tabla PROD_SUC se registrarán las combinaciones producto – sucursal, indicando cuales productos se comercializan en cuales sucursales.

Este esquema también se podría utilizar directamente en el modelo lógico reemplazando a la asociación “muchos a muchos”, por un diseño similar al que finalmente quedará en el modelo físico.

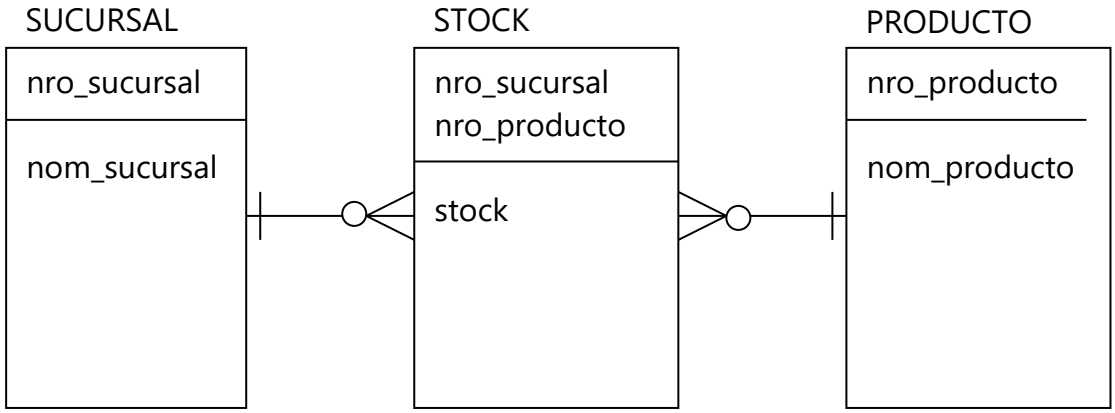
Una pregunta que surge es si será conveniente mantener como clave primaria la combinación (nro_sucursal, nro_producto) o si será conveniente agregar una clave primaria interna, por ejemplo: nro_suc_prod (un nro. interno):



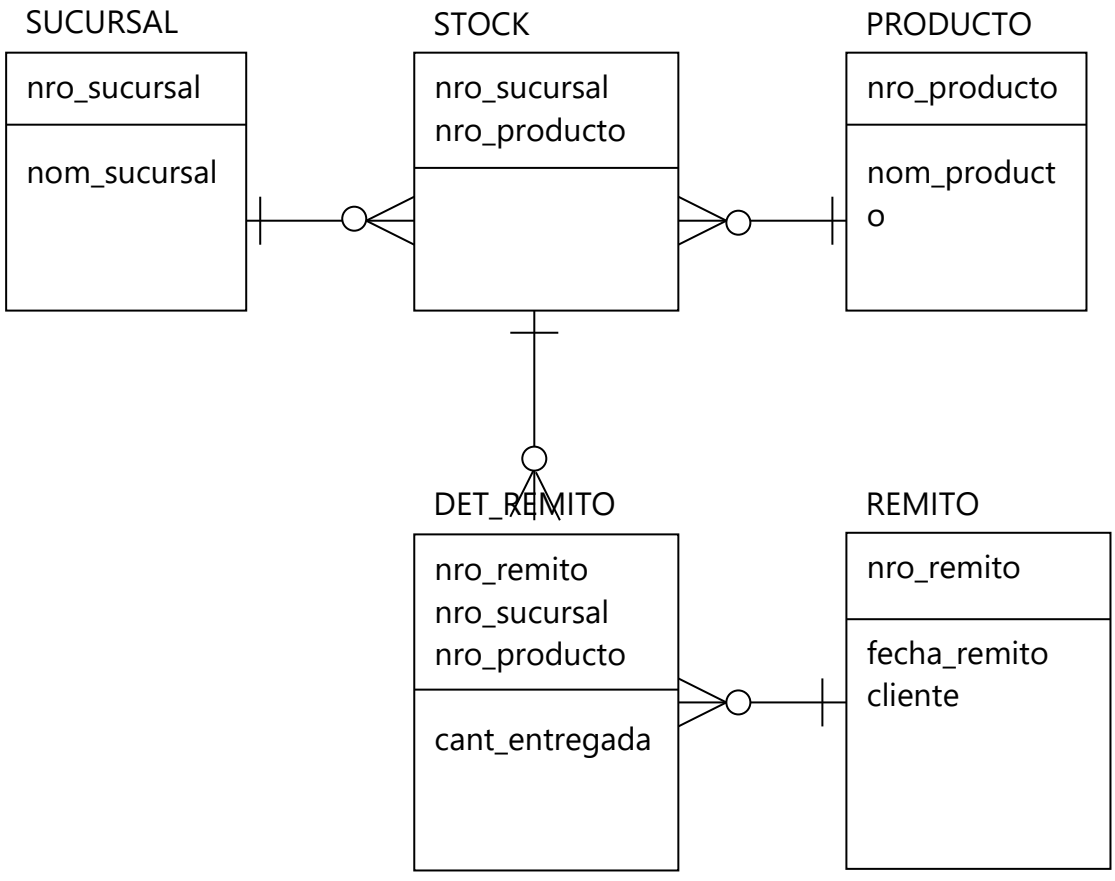
Esto no serviría de mucho, ya que la combinación (nro_sucursal, nro_producto) debe ser única y requerirá entonces definirla como clave alternativa, generando un índice adicional para esto. Con lo cual lo único que estamos haciendo es hacer que la tabla sea más grande y agregando un índice, lo cual no sería conveniente.

Ahora bien, hay casos en los cuales, en el modelo lógico necesariamente debemos resolver la asociación muchos a muchos con una entidad de intersección o intermedia. Esto ocurre cuando dicha entidad tiene atributos propios.

En el caso anterior, supongamos que además necesitamos conocer el stock de cada producto en cada sucursal. En ese caso, necesariamente debemos definir una entidad ya en el modelo lógico:



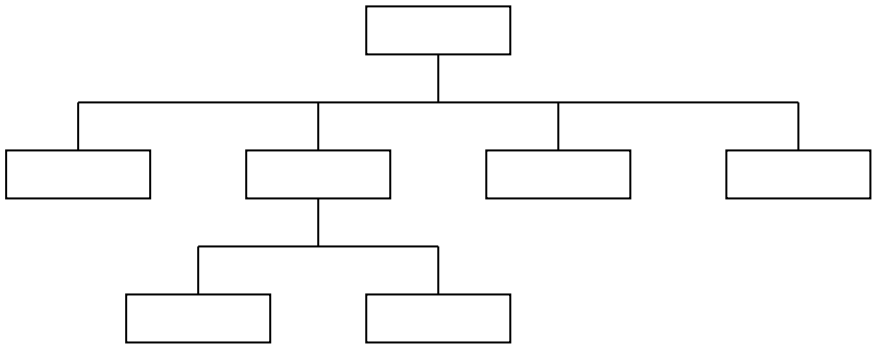
O que necesitemos que esa entidad intermedia tenga asociaciones propias. Por ejemplo:



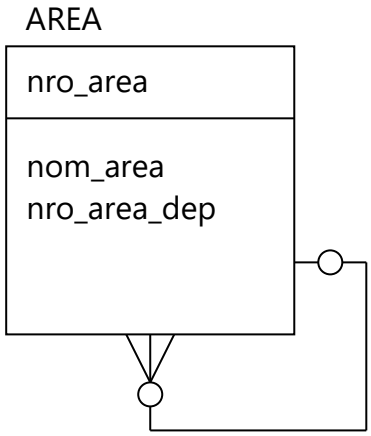
Asociaciones recursivas:

Es una asociación entre una entidad y ella misma.

Supongamos que se requiere modelizar la estructura jerárquica de las áreas de la empresa, la cual tiene una estructura de árbol, tal como la siguiente:



Una manera de modelizar esto sería:



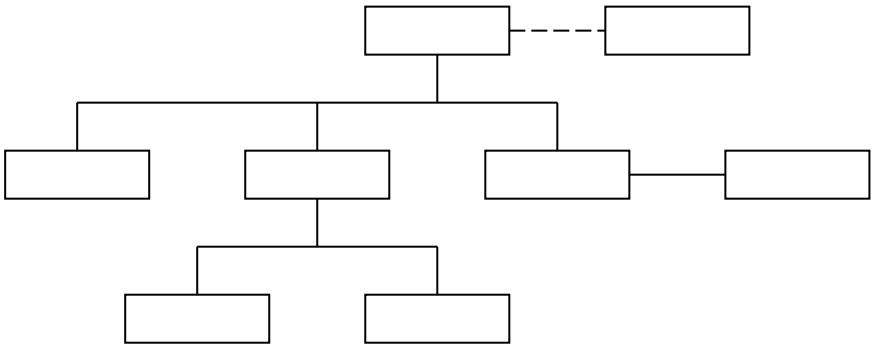
Esto se puede definir de la siguiente manera:

- Un área depende de cero o un área
- Un área tiene cero o muchos áreas dependientes

El atributo nro_area se propaga hacia la misma entidad pero con otro nombre: nro_area_dep (nro. del área de la cual depende).

Recomendamos en este punto realizar el ejercicio 4 de la actividad 2 correspondiente a este módulo.

Ahora bien, en una estructura funcional puede haber otro tipo de relaciones además de la "dependencia". Por ejemplo, puede haber un área que colabora con otra (relación de colaboración) o un área que asesora a otra (relación de asesoramiento).

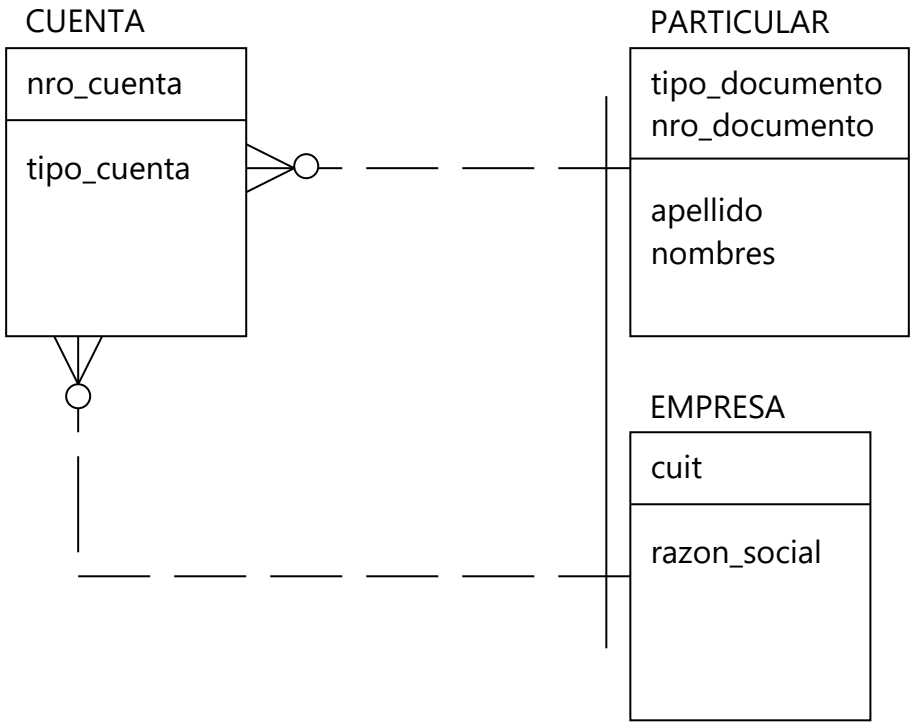


En el ejemplo, la relación con línea continua horizontal indica colaboración y con línea punteada asesoramiento.

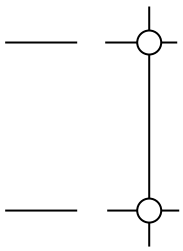
Recomendamos en este punto realizar el ejercicio 5 de la actividad 2 correspondiente a este módulo.

Asociaciones mutuamente exclusivas:

Este tipo de asociaciones se presenta cuando una entidad está asociada a varias entidades pero en forma excluyente una de otras. Por ejemplo, supongamos que un Banco tiene una entidad cuentas cuyo titular puede ser un particular o una empresa (uno u otro y no ambos). Esto se podría representar de la siguiente manera:



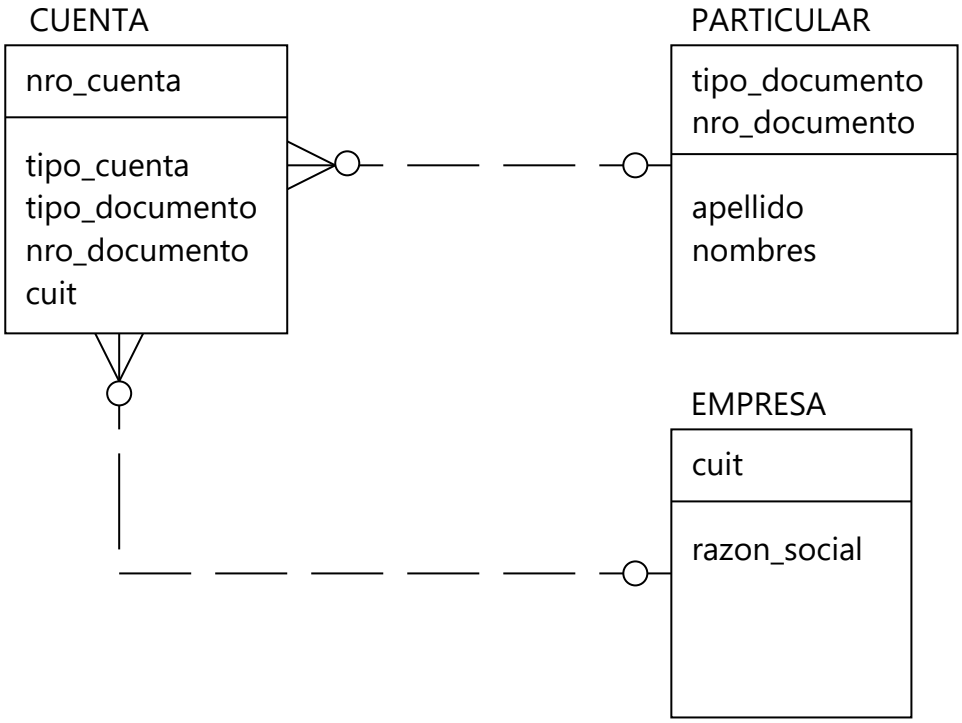
Las línea continua que une ambas asociaciones indica que la asociación es obligatoria (una cuenta SIEMPRE está asociada a una de ellas y solo a una).
Si una cuenta no estuviera obligatoriamente asociada a un particular o a una empresa (asociación no obligatoria), entonces a la línea continua que une ambas asociaciones habrá que agregarle el círculo que indica asociación no obligatoria:



La implementación de este tipo de asociaciones se realiza de una de las dos maneras siguientes:

1. Con claves foráneas (la más común)
2. Sin claves foráneas

En el primer caso se definen claves foráneas no obligatorias y se programa una regla de integridad adicional declarativa en la cual se verifica la obligatoriedad exclusiva:

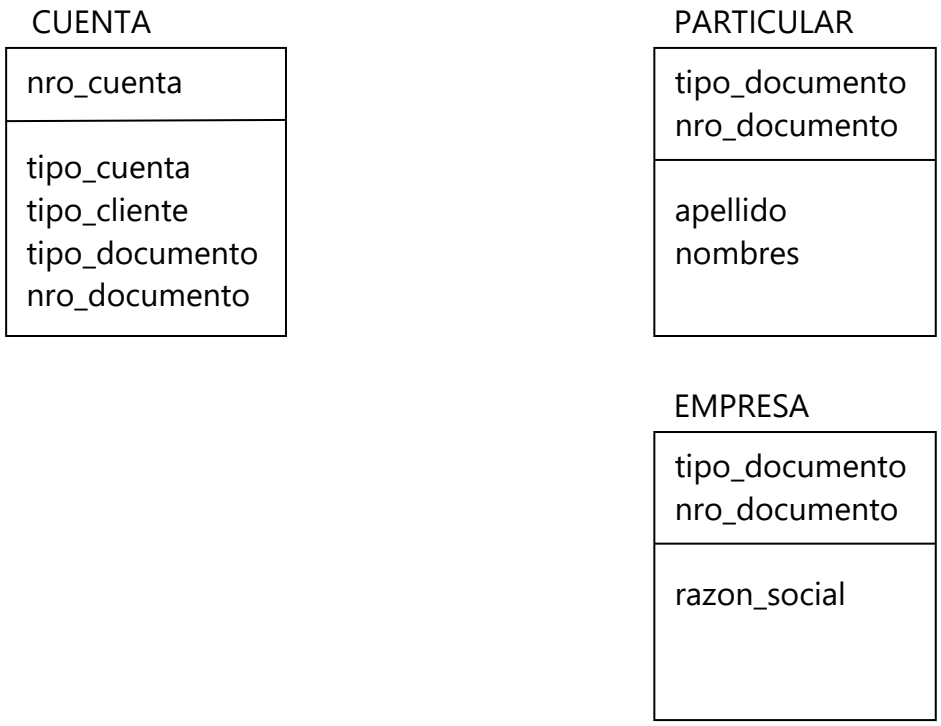


Los atributos tipo_documento, nro_documento y cuit permiten valores nulos (no son obligatorios), pero se deberá verificar que se cumpla alguna de estas dos reglas de integridad:

- tipo_nro_documento y nro_documento no nulos y cuit nulo
- tipo_nro_documento y nro_documento nulos y cuit no nulo

Esta verificación se puede implementar con una regla de integridad declarativa.

La segunda alternativa de implementación requiere que las claves primarias de las entidades PARTICULAR y EMPRESA tengan el mismo dominio (igual cantidad de atributos e igual dominio para cada uno). Supongamos que ambas entidades tienen el par tipo_documento y nro_documento como claves primarias. En ese caso la entidad CUENTAS podrá contener un único par de atributos tipo_documento y nro_documento, que recibirán los valores correspondientes de un PARTICULAR o de una EMPRESA. El problema pasa por saber a quien corresponden, si a uno o a la otra. Para esto se agrega un atributo adicional que indica a que entidad está asociada la tupla. Por ejemplo, el atributo "tipo_cliente":



En este caso, tipo_cliente tendrá una 'P' o una 'E', indicando si los valores en tipo_documento y nro_documento corresponden a un PARTICULAR o a una EMPRESA.
Además, estos últimos atributos son obligatorios ya que siempre la CUENTA está asociada o a un PARTICULAR o a una EMPRESA.

Para esta solución se deberán programar triggers que verifiquen que el par tipo_documento y nro_documento correspondan en cada caso a un PARTICULAR o a una EMPRESA.

La pregunta es, si en el caso de que se pueda implementar cualquier de las dos soluciones (recordar que no siempre se puede implementar la segunda), como elegir una de ellas.

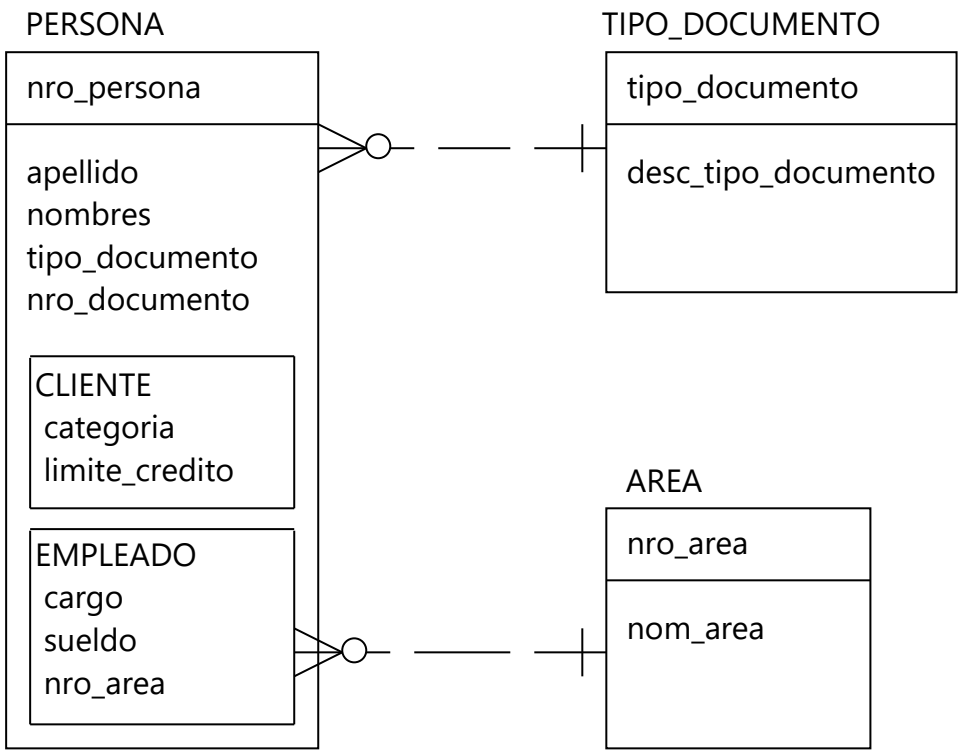
La segunda solución tiene una única ventaja que está dada por el hecho de que los atributos propagados se re-usan para mantener la referencia a cualquiera de las entidades. Esto es conveniente solo en los casos en los que se tiene más de dos entidades asociadas en forma mutuamente exclusiva o que la cantidad de atributos que conforman las claves primarias de estas asociaciones son tres o más.

De todas maneras, no hay que pensar en que esto disminuye demasiado el tamaño total de la tabla que recibe estos atributos, ya que en el primer caso los atributos que no contienen la referencia son nulos y, por lo tanto lo que aumenta el tamaño de cada fila es mínimo.
Además, el uso de tablas implementadas con la segunda solución es más complejo.

Todo esto nos indica que, salvo casos especiales, la primer solución es la más conveniente.
Cabe agregar que la mayoría de los CASEs no soporta la simbología de asociaciones mutuamente exclusivas y que, por lo tanto, en general se debe decidir la forma de implementación mientras se diseña el modelo lógico y, en ese caso, éste se verá tal como alguna de las dos implementaciones mostradas según la decisión que tomemos (por lo general la primera).

Sub-tipos:

Muchas veces se puede ver que un conjunto de entidades tienen varias características comunes (iguales atributos) y otras características particulares (atributos propios).
En esos casos, se puede considerar que se tiene un tipo de entidad (super-tipo) y casos especiales de la misma (sub-tipos).
Por ejemplo, podemos considerar las entidades CLIENTE y EMPLEADO, si suponemos que los clientes son siempre personas físicas al igual que los empleados, podemos considerar que existe un super-tipo PERSONA con atributos y asociaciones que son comunes a los dos sub-tipos y dos sub-tipos CLIENTE y EMPLEADO, cada uno de ellos con atributos y asociaciones propias.



Notar que la asociación de una instancia del sub-tipo EMPLEADO con un área es obligatoria.

Algunas consideraciones acerca de sub-tipos:

- Se debe tener en cuenta si los sub-tipos son exclusivos (excluyentes). En el ejemplo, si una persona es CLIENTE o EMPLEADO pero no puede ser ambos o si puede ser CLIENTE y a su vez EMPLEADO.

- Se debe definir si cada persona SIEMPRE pertenece a alguno de los sub-tipos, es decir, si la definición es completa o no. En este caso, toda persona o es CLIENTE o es EMPLEADO o ambos, pero no puede no ser ninguno de ellos. Siempre se debe definir en forma completa el conjunto de sub-tipos. Para esto, se podría agregar un sub-tipo OTRA PERSONA para satisfacer la propiedad de COMPLETITUD.

- Puede ser que un mismo super-tipo pueda ser visto como perteneciente a diferentes conjuntos de sub-tipos desde diferentes puntos de vista. Por ejemplo, en el caso anterior, también podemos ver a las personas como pertenecientes a dos sub-tipos de acuerdo al género (MUJER u HOMBRE). A estos dos conjuntos se les denomina sub-tipos ortogonales.

- Hay que tener en cuenta que si uno busca sub-tipos, se los encuentra muy a menudo. Lo importante para reconocer una estructura como esta es saber si,

de acuerdo a los requerimientos, se utiliza frecuentemente el super-tipo o no y si los supuestos sub-tipos son exclusivos o no.
Por ejemplo, en un caso de CLIENTES y PROVEEDORES, si los proveedores son empresas y los clientes son personas físicas, y no hay procesos que utilicen al conjunto de clientes y proveedores al mismo tiempo, a pesar de tener varios atributos comunes, no tendría sentido diseñar un modelo con super-tipo y sub-tipos.
Ahora bien, imaginemos el caso de la Universidad. Los alumnos son además, clientes, muchos de ellos se gradúan y luego son docentes, con lo cual forman parte del personal, además podrían realizar algún trabajo de asesoramiento profesional con lo cual podrán ser proveedores. Es claro que la superposición (no exclusividad) es importante, sobre todo si se piensa en el caso de que la mayoría de las personas relacionadas con la Universidad son alumnos y que cada alumno es cliente.
Además consideremos que la Universidad tiene varios procesos en los cuales se tratan instancias de varios de los sub-tipos, por ejemplo, cuando se envía información acerca de cursos tanto a alumnos como a egresados y docentes.
En este caso, se considerará el uso de un esquema de super-tipo y sub-tipos ya que se tiene la ventaja de procesamiento como de control de consistencia (evitando redundancias).

Implementación de sub-tipos:

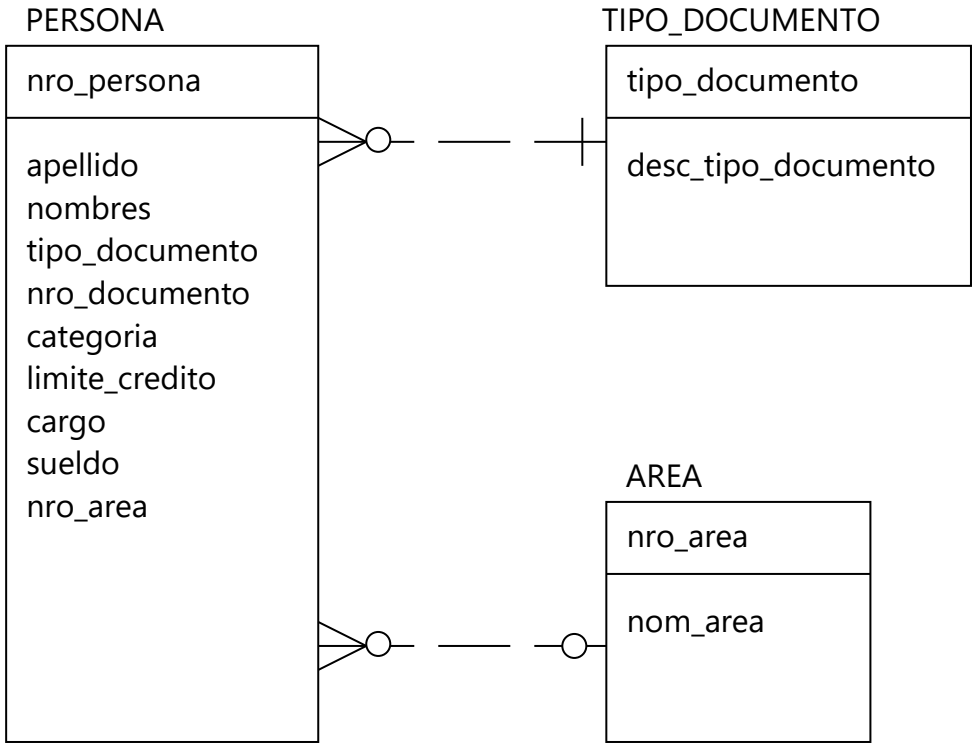
Existen 3 maneras de implementar el esquema de super-tipo y sub-tipos:

1. Todo en una tabla (super-tipo)
2. Una tabla por sub-tipo
3. Combinación de las dos anteriores (una tabla para el super-tipo y una para cada sub-tipo)

1. Todo en una tabla.

- Todas las instancias se registran en una única tabla
- Los atributos propios de los sub-tipos son no obligatorios (aunque sean obligatorios para el sub-tipo) y se deben agregar reglas de integridad declarativas para analizar la obligatoriedad o no de los atributos propios de los sub-tipos)
- Se puede agregar un atributo adicional para facilitar la evaluación de la pertenencia de una instancia a un sub-tipo o a otro (o a más de uno si no fuera excluyente)
- Los sub-tipos se implementan con vistas del tipo restricción (selección de tuplas)

- Es conveniente en el caso de superposición importante de sub-tipos (todos los alumnos son clientes) para evitar redundancia y disminuir el costo de almacenamiento
- También es muy útil cuando la frecuencia de uso del conjunto completo de instancias (super-tipo) es mayor que el procesamiento de los sub-tipos individualmente

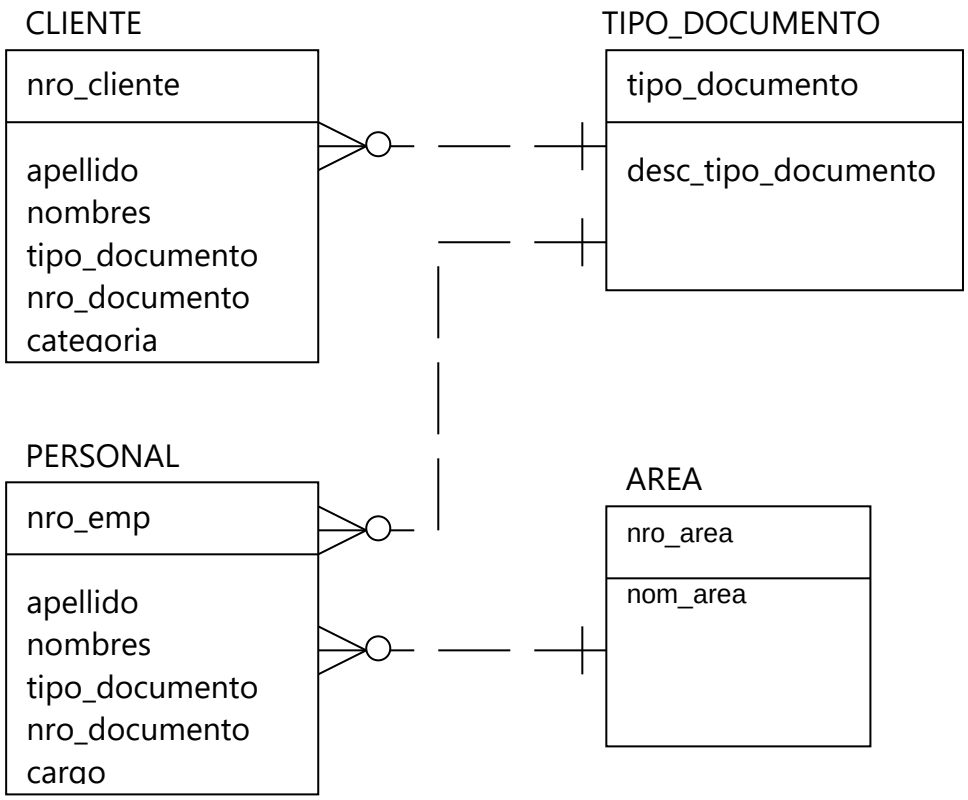


Notar que la asociación del super-tipo PERSONA con AREA pasa a ser opcional (no obligatorio) ya que solo en el caso de que la persona sea EMPLEADO estará asociado a un área.

2. Una tabla para cada sub-tipo.

- Los atributos del super-tipo se propagan (copian) a cada sub-tipo manteniendo su obligatoriedad o no.
- Los atributos de los sub-tipos también se mantienen con su característica de obligatoriedad.
- Cada instancia se registra en la tabla correspondiente. Podría ser que una misma instancia se registre en más de una tabla si pertenece a más de un sub-tipo, lo que implica redundancia, la cual habrá que controlar para que no existan inconsistencias

- El super-tipo se implementa con una vista del tipo unión, la cual suele implicar una caída en la performance si requiere la eliminación de duplicados.
- Es conveniente siempre que los sub-tipos sean exclusivos
- También en el caso que el uso de los sub-tipos individuales sea mucho más frecuente que el uso del conjunto completo de instancias (super-tipo)

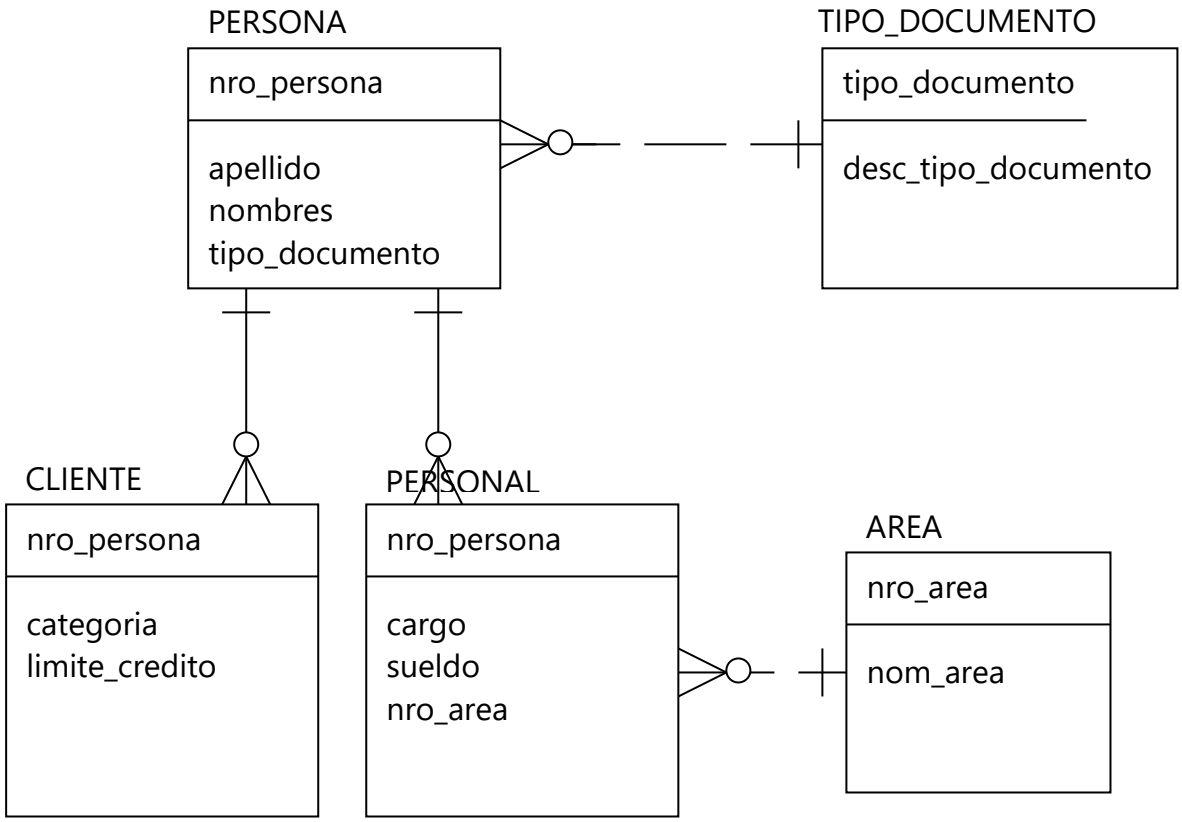


Notar que se mantiene la obligatoriedad de la asociación entre el sub-tipo PERSONAL y AREA.

3. Combinación

- Se propaga la clave primaria del super-tipo a los sub-tipos.
- Los atributos y asociaciones mantienen su característica de obligatoriedad
- Es conveniente en el caso de un uso frecuente de las instancias del super-tipo pero sin requerir el uso de atributos propios de los sub-tipos (no hay necesidad de realizar JOINS entre la tabla del super-tipo y las de los sub-tipos)
- Normalmente el uso de un sub-tipo requerirá el JOIN con la tabla correspondiente al super-tipo. Esto puede implicar una caída de performance, aunque las tablas individuales tendrán menor tamaño.

- Requiere más tablas, lo que implica un mayor problema a la hora de administrar seguridad.
- Se pueden definir vistas del tipo JOIN para el super-tipo y los sub-tipos, con el fin de facilitar el uso de los mismos



El proceso de modelización de estas bases de datos se complementa con el proceso de normalización de las mismas, tema que será tratado a continuación.

Para finalizar con el estudio del tema modelización, recomendamos en este punto realizar los ejercicios 1 y 2 de la actividad 3 correspondiente a este módulo.

NORMALIZACIÓN:

Habiendo finalizado la etapa de elaboración del modelo conceptual, se debe decidir cual modelo de base de datos se utilizará.
Si la elección recae en el modelo relacional, entonces, la etapa de diseño lógico tendrá como resultado final un diagrama de entidades y relaciones (DER) normalizado.
Para realizar esta actividad se cuenta en el mercado con una gran variedad de CASEs orientados a la construcción de estos diagramas basados en el modelo relacional,

que facilitan su construcción, la ejecución de la siguiente etapa (diseño físico) y finalmente su implementación.

Este modelo lógico es el que se utilizará como base para la implementación del esquema conceptual (nivel conceptual) de la base de datos.
NOTA: no confundir el esquema conceptual de la base de datos con el modelo de datos conceptual (revisar estos conceptos).

NOTA: A partir de este momento, todo el desarrollo se basará en el supuesto de la elección del modelo relacional.

Ahora bien, en esta etapa es crucial lograr un buen diseño lógico que permita extraer de la base de datos, la información necesaria y debe asegurar que esta información es confiable, independientemente de la performance y los tiempos de respuesta.

¿Qué significa entonces, un buen diseño lógico? ¿Qué distingue un buen diseño de un mal diseño?

Estas son preguntas que se responden con las siguientes tres palabras: integridad, redundancia y consistencia.

- Un buen diseño debe contemplar todas las reglas de integridad de los datos.
- Un buen diseño debe evitar al máximo posible la redundancia (en principio, no debiera haber redundancia como resultado del modelo lógico, luego, si fuera necesario, se puede agregar redundancia en el modelo físico)
- Un buen diseño, aún con redundancia, debiera prever los procesos de aseguramiento de la consistencia en los datos almacenados.

Recomendamos en este punto realizar el ejercicio 1 de la actividad 4 correspondiente a este módulo.

Suponiendo que el modelo conceptual y el diccionario de datos, nos brindan toda la información que necesitamos acerca de los datos que debe contener la base de datos y que el diseño de los procesos y procedimientos nos ha permitido conocer en profundidad la gestión de los mismos, ahora podemos comenzar con el diseño lógico.
Para esto se podrán aplicar las técnicas de modelización ya revisadas, pero además deberemos contemplar algo más para lograr el buen diseño que esperamos.

El proceso por el cual mejoraremos nuestro diseño lógico es aquel conocido como "normalización adicional".

Caben dos preguntas hasta aquí:

1. ¿Este proceso se debe realizar en la etapa final del diseño lógico?, es decir, ¿primero se modeliza y luego se aplica el proceso de normalización adicional?.

   No necesariamente, se puede ir aplicando mientras se modeliza y al final solo se verifica. Esto es más común en profesionales con experiencia en diseño relacional.

2. ¿Por qué se llama "normalización adicional"?, ¿hay un proceso previo de normalización?

   Si volvemos a la definición de relación y a sus propiedades vamos a notar que una esas propiedades expresa que "en cada tupla, todo atributo simple contiene (tiene asociado) un único valor atómico", esto quiere decir que no puede haber grupos repetitivos, es decir, un atributo no puede contener un conjunto de valores.

   Esta es una regla de integridad intrínseca del modelo (meta-regla), todas las relaciones la deben cumplir. A esta primera regla se le denomina "1ª Forma Normal".

   Esto implica que todas las relaciones cumplen con esa regla y por lo tanto se dice que "toda relación está normalizada". Por lo tanto, una base de datos relacional es "un conjunto de relaciones normalizadas", aunque podría decirse que es "un conjunto de relaciones" (ya que estamos hablando de eliminar redundancias).

   Por lo tanto, el proceso de "normalización adicional", entonces aplica nuevas reglas que continúan normalizando las relaciones y de ahí su nombre.

Normalización adicional:

Repasemos algunas ideas:

Forma Normal: restricción a partir de la cual se mejora el diseño de una base de datos relacional. Hay varias formas normales:

Normalización:

1FN: 1ª Forma Normal

Normalización adicional:

2FN: 2ª Forma Normal
3FN: 3ª Forma Normal
FNBC: Forma Normal de Boyce-Codd
4FN: 4ª Forma Normal
5FN: 5ª Forma Normal

En ese orden, cada una de ellas agrega una nueva restricción a las anteriores. A medida que avanzamos en el proceso de normalización, haciendo que se cumplan las sucesivas formas normales, se van eliminando diferentes tipos de redundancias, mejorando el diseño.

NOTA: A partir de este momento dejaremos de distinguir entre el proceso de normalización y el de normalización adicional.

El proceso de normalización debe realizarse teniendo en cuenta la siguiente condición: Cada paso en el proceso de normalización debe obtener como resultado una base de datos equivalente a la original, pero con un mejor diseño. Es decir que el proceso de normalización no debe "perder" información.

¿Cómo se realiza el proceso de normalización?.

La normalización de una relación se realiza mediante la descomposición de la misma en sub-relaciones, aplicando la operación "PROYECCIÓN" del álgebra relacional. Este proceso se conoce con el nombre de "descomposición sin pérdida de información". Las sub-relaciones se denominan "proyecciones".

En la práctica, una "descomposición sin pérdida de información", significa que se puede volver a la relación original aplicando la operación inversa sobre las proyecciones obtenidas. La operación inversa es la reunión natural (JOIN NATURAL).

Por ejemplo: sea R una relación donde a es la clave primaria y b,c,d y e son atributos no clave.

R (a, b, c, d, e)

a	b	c	d	e
1	1	2	3	4
2	1	2	3	4
3	4	6	8	7
4	0	9	1	2

Se puede descomponer R en:

R1 (a, b, c) y R2 (d, e)

a	b	c
1	1	2
2	1	2
3	4	6
4	0	9

d	e
3	4
8	7
1	2

(notar la eliminación de duplicados en esta última proyección)

o en:

R1 (a, b, c) y R2 (c, d, e)

a	b	c
1	1	2
2	1	2
3	4	6
4	0	9

c	d	e
2	3	4
6	8	7
9	1	2

(notar la eliminación de duplicados en esta última proyección)

Pero esta descomposición en ambos casos produce pérdida de información ya que el resultado del JOIN entre las relaciones proyectadas no produce la relación original R. En el primer caso porque no hay atributos comunes, por lo tanto la relación resultado del join es una relación vacía, y en el segundo caso porque las tuplas obtenidas como

resultado del join no son las correspondientes a la relación original (puede haber más tuplas, menos tuplas y/o tuplas diferentes).

La descomposición de R en:

R1 (a, b, c) y R2 (a, d, e)

a	b	c
1	1	2
2	1	2
3	4	6
4	0	9

a	d	e
1	3	4
2	3	4
3	8	7
4	1	2

sí produce la relación original R y se considera una descomposición sin pérdida de información.

Resumiendo:

- Las formas normales son restricciones que toda relación debe cumplir para evitar esquemas indeseables (con redundancias)
- Cada forma normal agrega restricciones a las anteriores
- El proceso de normalización se practica aplicando sucesivas “proyecciones” a la relación original para lograr una descomposición sin pérdida de información de la misma, con el objetivo de llevarla a la máxima forma normal y lograr un mejor diseño (diseño sin redundancias).

Notas:

- Durante el proceso de normalización (descomposición), como en todo proceso de diseño se deben tomar decisiones de compromiso, es decir, decidir por dos o más alternativas, cada una de las cuales presenta ventajas y desventajas. Dos situaciones se presentan aquí:

1. Una base de datos totalmente normalizada no tiene redundancias, pero en algunos casos, cierto tipo de redundancia, facilita el tratamiento de los datos y/o permite una mejora en la performance del procesamiento de los mismos.

   Esta redundancia se debiera agregar al momento del diseño físico, pero en la práctica muchos diseñadores deciden agregarla ya en el modelo lógico. Es importante, en este último caso, realizar las aclaraciones correspondientes, de tal manera que se implementen los procesos de "propagación de actualizaciones" que permitirán asegurar la consistencia.

2. Las proyecciones (sin pérdida de información) pueden tener problemas de dependencia. Es decir, la actualización de una de las proyecciones puede requerir la revisión de una regla de integridad en la otra (proyecciones dependientes).

   Siempre se debe tratar de elegir (de entre las proyecciones sin pérdida) aquellas que producen proyecciones independientes, pero no siempre es posible.
   En este caso se debe decidir si se lleva las relaciones a una forma normal superior, produciendo una mejora desde el punto de vista de la redundancia, aunque las mismas sean dependientes y requiriendo un proceso de verificación adicional; o si se la deja en la forma normal actual, aunque se mantenga algo de redundancia y requiriendo de esta manera un proceso de propagación de actualizaciones, pero no necesitando un proceso de verificación adicional.
   NOTA: Esto se verá en mayor detalle más adelante.

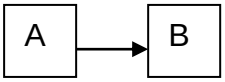
Las formas normales, a partir de la segunda, se basan en el concepto de dependencia. En particular, las formas normales 2ª, 3ª y Boyce-Codd, se basan en el concepto de dependencia funcional.

Dependencia funcional: En una relación R, el atributo B depende funcionalmente del atributo A, si para cada valor de A existe un único valor de B. También se dice que A determina funcionalmente a B.
A se denomina "determinante".
B se denomina "dependiente".
A y B pueden ser compuestos y es por eso que se los escribe en mayúsculas.

Esta dependencia funcional se denota como:

A → B

Gráficamente:



Dependencia funcional total: En una relación R, el atributo B depende funcionalmente en forma total del atributo A, si B depende funcionalmente de A y no depende funcionalmente de un sub-conjunto de A.
La dependencia funcional total queda asegurada si A es un atributo simple.

Teniendo en claro estos conceptos, pasemos a la definición de las formas normales y a las técnicas de normalización.
Inicialmente, nos concentraremos en las primeras tres formas normales, luego completaremos las demás.

Definiciones:

1ª Forma Normal: Una relación está 1ª Forma Normal (1FN) si todos sus atributos simples contienen valores atómicos. Es decir si no contiene grupos repetitivos.

NOTA: Una relación, por definición, está siempre en la primera forma normal, aunque en algunos ejemplos partiremos de "relaciones no normalizadas" para la aplicación de alguna técnica.

2ª Forma Normal: Una relación está en 2ª Forma Normal (2FN) si está en 1FN y todos los atributos no clave (no participan de la clave primaria) dependen funcionalmente en forma total de la clave primaria.

Recomendamos en este punto realizar el ejercicio 2 de la actividad 4 correspondiente a este módulo.

3ª Forma Normal: Una relación está en 3ª Forma Normal (3FN) si está en 2FN y todos los atributos no clave (no participan de la clave primaria) son independientes entre sí. No existe una dependencia funcional transitiva.

Recomendamos en este punto realizar el ejercicio 3 de la actividad 4 correspondiente a este módulo.

Técnicas de normalización:

1ª FORMA NORMAL:

¿Cómo pasamos de una "relación no normalizada" a una normalizada?.

Grupos repetitivos: atributo o conjunto de atributos que contienen más de un valor atómico.
Los grupos repetitivos pueden ser dependientes o independientes entre sí.

Ejemplo: Se tiene la siguiente "relación no normalizada":

R (nro_area, nom_area, {nro_emp, nom_emp, {cargo, sueldo}}, {año, presupuesto})

Aclaraciones:

- La relación contiene información de las áreas de la organización, sus empleados y el presupuesto de cada año.
- Las llaves encierran atributos que corresponden a grupos repetitivos.
- Los atributos subrayados corresponden a identificadores:

    - Nro_area: identificador de la relación (identificador de área – nunca se repite)
    - Nro_emp: identificador de cada empleado (es único – nunca se repite)
    - Cargo: identificador de cada cargo ocupado por el empleado en su historia dentro de la organización (único por empleado – se puede repetir para empleados diferentes)
    - Año: identificador del presupuesto (único por área – se puede repetir en áreas diferentes)

- El grupo "cargos de cada empleado" es dependiente del grupo "empleado".
- Los grupos "empleado" y "presupuesto" son independientes.

Los pasos para pasar de una "relación no normalizada" a una "relación normalizada" (1FN) son los siguientes:

1. Determinar el identificador de la relación.
2. Proyectar los grupos repetitivos independientes a relaciones independientes (quedando como resultado la relación original sin grupos repetitivos y una relación por cada grupo repetitivo independiente).
3. Determinar el identificador de cada proyección.

4. Aplicar los mismos pasos a cada relación que contenga grupos repetitivos hasta que todas las relaciones estén normalizadas.

Apliquemos estos pasos a R:

1. El identificador de la relación es: nro_area.

2. Cada grupo repetitivo independiente se propaga junto con el identificador de la relación que lo contiene, quedando la relación original sin estos grupos.

   Emp (nro_area, nro_emp, nom_emp, {cargo, sueldo})

   Presupuestos (nro_area, año, presupuesto)

   Areas (nro_area, nom_area)

3. Determinar el identificador de cada proyección. Tener en cuenta el identificador de cada grupo repetitivo y su relación con el atributo de la relación original.

   Emp (nro_area, nro_emp, nom_emp, {cargo, sueldo})

   Presupuestos (nro_area, año, presupuesto)

   Areas (nro_area, nom_area)

4. Aplicar los mismos pasos a la relación emp:

   Cargos_Emp (nro_emp, cargo, sueldo)

   Emp (nro_area, nro_emp, nom_emp)

Finalmente el conjunto de relaciones normalizadas (1FN) queda:

   Areas (nro_area, nom_area)

   Presupuestos (nro_area, año, presupuesto)

   Emp (nro_area, nro_emp, nom_emp)

   Cargos_Emp (nro_emp, cargo, sueldo)

Ahora quedará por verificar si estas relaciones están totalmente normalizadas y, si no es así, llevarlas a la forma normal más elevada (5FN).

2ª FORMA NORMAL:

¿Cómo pasar de una relación en 1FN a un conjunto de relaciones en 2FN?

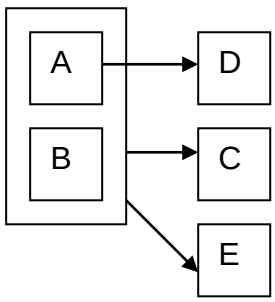
Los pasos son los siguientes:

1. Determinar claramente la clave primaria de la relación
2. Determinar las dependencias funcionales no totales (aquellas dependencias que nos traen problemas)
3. Proyectar las dependencias funcionales no totales
4. Proyectar los atributos de la relación original no proyectados en el paso 3 junto a los determinantes de las proyecciones del paso 3.

Ejemplo: Sea una relación R (A, B, C, D, E), con las siguientes dependencias funcionales:

A,B → C
A → D
A,B → E

Gráficamente:



1. Encontrar las claves candidatas de R.

   NOTA: Hacemos un paréntesis aquí, para revisar conceptos y técnicas utilizadas para encontrar las claves candidatas de una relación a partir de los axiomas de Armstrong.

   Para encontrar las claves candidatas de una relación se debe encontrar el conjunto mínimo (irreducible) de dependencias funcionales que sea equivalente al conjunto de todas las dependencias funcionales de la relación.
   Para obtener este conjunto se utilizan los axiomas de Armstrong.

   Axiomas de Armstrong:

   1. Reflexividad: Si B es un sub-conjunto de A, entonces A → B
   2. Aumento: Si A → B, entonces AC → BC
   3. Transitividad: Si A → B y B → C, entonces A → C

   De éstos se desprenden:

   4. Determinación propia: A → A
   5. Descomposición: Si A → BC, entonces A → B y A → C
   6. Unión: Si A → B y A → C, entonces A → BC
   7. Composición: Si A → B y C → D, entonces AC → BD

   Superclave: Una superclave es un sub-conjunto K de R que tiene la propiedad de unicidad (no hay dos tuplas con el mismo valor de K), pero no tiene la propiedad de irreducible (por eso no es una clave candidata). Todos los atributos de R dependen funcionalmente de K.

   Clave candidata: Es una superclave irreducible.

   Para obtener las claves candidatas, encontrar las superclaves y luego reducirlas.

   Dado un conjunto de dependencias funcionales, siempre hay un conjunto equivalente irreducible. Ejemplo:

   A → BC
   B → C
   A → B
   AB → C
   AC → D

   1. Reducir la parte derecha de la dependencia funcional:

A → B
A → C
B → C
A → B
AB → C
AC → D

2. Reducir la parte izquierda:

A → B
A → C
B → C
A → B
AB → C ➔ *A → C (por aumento: A → B ➔ A → AB - por transitividad: A → AB y AB → C ➔ A → C)*
AC → D ➔ *A → D (por aumento: A → C ➔ A → AC - por transitividad: A → AC y AC → D ➔ A → D)*

3. Eliminando las duplicadas:

A → B
B → C
A → D

queda un conjunto irreducible de dependencias funcionales.

De este conjunto se puede deducir las claves candidatas de la relación.

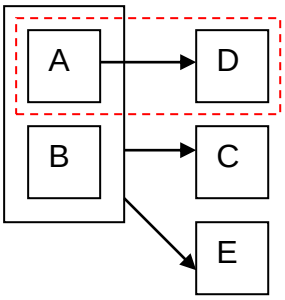
Recomendamos en este punto realizar los ejercicios 4 a 8 de la actividad 4 correspondiente a este módulo.

Volviendo al problema planteado, siendo el conjunto de dependencias funcionales dadas el conjunto mínimo:

A,B	→	C
A	→	D
A,B	→	E

La clave candidata y clave primaria es: AB

2. Encontramos que esta relación está en 1FN, ya que existe una dependencia funcional no total. Para llevar la relación a 2FN debemos eliminar esa dependencia.



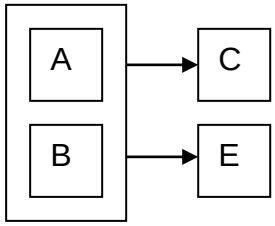
Proyectamos los atributos de la dependencia con problemas (A,D) a una relación y el resto de los atributos (B,C,E) y el determinante de la proyección anterior (A) a otra. Este último atributo queda como clave foránea a la proyección anterior.

R1 (A, D)
R2 (A, B, C, E) FK: A referencia a R1

Gráficamente:

Para R1:

Para R2:



3ª FORMA NORMAL:

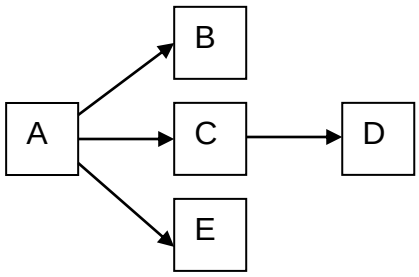
¿Cómo pasar de una relación en 2FN a un conjunto de relaciones en 3FN?

Los pasos a seguir son los mismos que para pasar a 2FN.

Ejemplo: Sea una relación R (A, B, C, D, E), con las siguientes dependencias funcionales:

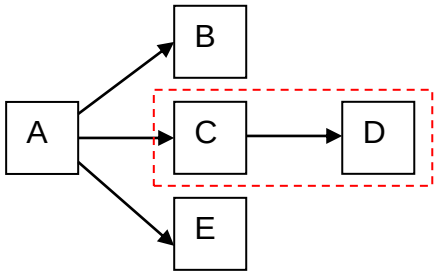
A → B,C,E
C → D

Gráficamente:



Aplicando la misma técnica que en el caso anterior:

1. La clave primaria de la relación es A.
2. Encontramos que la dependencia funcional C → D, es una dependencia funcional transitiva que produce anomalías de actualización.

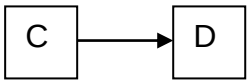


3. Proyectamos los atributos de la dependencia funcional problemática (C,D) a una nueva relación y el resto de los atributos (A,B,E) más el determinante de la proyección anterior (C) a otra. Quedando:

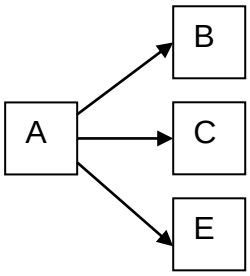
   R1 (C, D)
   R2 (A, B, C, E) FK: C referencia a R1

   Las dependencias funcionales para las dos relaciones son las siguientes:

Para la relación R1:



Para la relación R2:



Recomendamos en este punto realizar los ejercicios 9 y 10 de la actividad 4 correspondiente a este módulo.

Continuemos con las siguientes formas normales.

Para esto se debe realizar previamente una aclaración.
Hasta la 3FN la teoría solo analiza las anomalías la relación suponiendo la existencia de una clave primaria, sin evaluar la existencia de otras claves candidatas (claves alternativas) de la relación.

La forma normal de Boyce-Codd considera esta situación.

FORMA NORMAL DE BOYCE-CODD (FNBC):

Ampliamos aquí un poco más el concepto de dependencias funcionales, superclaves y claves candidatas.

Superclave: Una superclave K de R es un sub-conjunto de atributos de R que cumple con la dependencia funcional:

$$K \rightarrow A$$

para todo atributo A de R.
Obviamente esto implica dependencias funcionales triviales (no hay manera de que no se cumplan) tales como K → X, siendo X un atributo de K.

R puede tener varias superclaves.

Como mencionábamos más arriba, el primer paso del proceso de normalización adicional (partiendo de una relación normalizada), es encontrar las clave primaria de la relación. Ampliamos esto, diciendo ahora que el primer paso será determinar todas las claves candidatas de la relación.

Clave candidata: Una clave candidata C es una superclave K irreducible. Es decir que no se puede eliminar ningún atributo de K y mantener la dependencia funcional K → A, para todo atributo A de R.

Dado un conjunto F de dependencias funcionales para una relación R; para determinar todas las claves candidatas de R, será necesario encontrar previamente un conjunto F' irreducible (mínimo) de dependencias funcionales equivalente a F.

Conjuntos equivalentes: El conjunto de todas las dependencias funcionales implicadas por F se denomina "Clausura de F" y se denota como F+.
Entonces F' será equivalente a F si sus clausuras son iguales (F+ = F'+)

F' no necesariamente es único.

Para encontrar la clausura (F+) de un conjunto de dependencias funcionales (F) se utilizan los axiomas de Armstrong.

Conjunto irreducible: Un conjunto F de dependencias funcionales es irreducible o mínimo si cumple con las siguientes reglas:

1. Los atributos dependientes (a la derecha) son atributos simples
2. Los determinantes (a la izquierda) son conjuntos irreducibles de atributos (ningún atributo puede ser descartado de dicho conjunto sin que cambie F+)
3. Ninguna dependencia funcional del conjunto puede ser descartada sin que cambie F+

Ejemplo:

Supongamos la relación R (A,B,C,D) y el siguiente conjunto F de dependencias funcionales:

A → BC

B → C
A → B
AB → C
AC → D

La primera dependencia funcional A→ BC no cumple con la primer regla. Aplicando los axiomas de Armstrong se puede encontrar el siguiente conjunto equivalente que sí la cumple:

A → B
A → C
B → C
A → B
AB → C
AC → D

Las dependencias funcionales AB → C y AC → D, pueden ser reducidas (no cumplen con la segunda regla. Aplicando nuevamente los axiomas de Armstrong se puede encontrar el siguiente conjunto equivalente:

A → B
A → C
B → C
A → B
A → C
A → D

Como se ve, quedan dependencias funcionales duplicadas, y por lo tanto este conjunto se puede reducir a:

A → B
A → C
B → C
A → D

La segunda dependencia funcional se puede eliminar ya que se deriva de la primera y la tercera. Finalmente nos queda:

A → B
B → C
A → D

Este conjunto F' es irreducible y equivalente a F.

Recomendamos en este punto realizar el ejercicio 11 de la actividad 4 correspondiente a este módulo.

Forma normal de Boyce-Codd (FNBC): Una relación está en la Forma Normal de Boyce-Codd (FNBC), si y solo si cada conjunto irreducible de dependencias funcionales tiene como determinantes a claves candidatas.

Informalmente: Una relación está en FNBC si y solo si cada determinante es una clave candidata.
Gráficamente: Las únicas flechas en un diagrama de dependencias funcionales salen de claves candidatas.

Para una relación que está en 3FN, la verificación de FNBC solo se justifica (la relación podrá estar en 3FN y no en FNBC) si se cumplen las tres reglas siguientes:

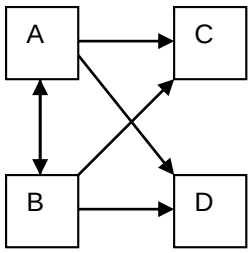
1. La relación tiene dos o más claves candidatas
2. Por lo menos dos de ellas son compuestas (contienen más de un atributo)
3. Ellas se solapan (tienen algún atributo en común)

En cualquier otra situación, si una relación está en 3FN, también está en FNBC.

Ejemplo 1:

Dada la relación R (A,B,C,D) y las dependencias funcionales:

A → CD
B → CD
A → B
B → A



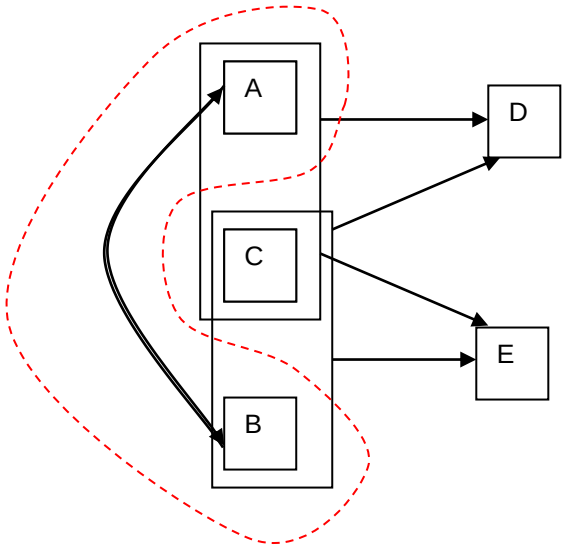
Suponiendo que A es la clave primaria, esta relación está en 3FN ya que todos los atributos no clave (C, D) dependen en forma total de la clave primaria y son independientes entre sí.
Además, esta relación está en FNBC ya que todos los determinantes son claves candidatas.

Ejemplo 2:

Supongamos la relación R (A,B,C,D,E), con claves candidatas (A,C) y (B,C) y las siguientes dependencias funcionales:

AC → D
AC → E
BC → D
BC → E
A → B

Gráficamente:



Suponiendo (AC) la clave primaria, la relación está en 3FN ya que los atributos no clave (D,E) dependen completamente de ella y son independientes entre sí.
NOTA: En la definición original de la 3FN no se considera que los atributos de una clave alternativa sean dependientes en forma total de la clave primaria.

La relación no está en la FNBC porque algunos determinantes (A y B), no son claves candidatas.
Nuevamente, aplicando la técnica ya vista, eliminamos las dependencias funcionales que nos traen problemas (A → B y B → A), llevándolas a nuevas relaciones.
Así, proyectamos:

R1 (A , B)
R1' (B , A)

Como ambas tienen los mismos atributos, solo dejamos una de ellas (cualquiera):

R1 (<u>A</u> , B) donde A: clave primaria y B: clave alternativa

NOTA: También se podría haber usado:

R1' (A , <u>B</u>) donde A: clave alternativa y B: clave primaria

Para completar el proceso, proyectamos:

R2 (<u>A, C</u>, D, E)

O

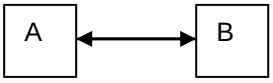
R2' (<u>B, C</u>, D, E)

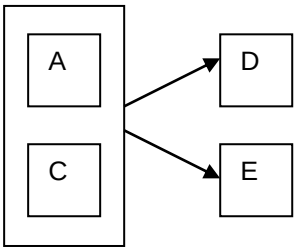
La elección de una u otra combinación de R1s y R2s, deberá basarse en el uso y tamaño de almacenamiento requerido para los atributos A y B.

En este caso, elegiremos:

R1 (<u>A</u>, B)	A: clave primaria, B: clave alternativa
R2 (<u>A, C</u>, D, E)	AC: clave primaria, A: clave foránea

Las dependencias funcionales de R1 y R2 son las siguientes:





Estas dos relaciones están en FNBC.

Proyecciones independientes:

Durante el proceso de normalización se debe tratar de que la descomposición efectuada produzca proyecciones independientes.

Dos proyecciones son independientes si se cumplen las 2 reglas siguientes:

1. Cada dependencia funcional en R (relación original) se puede deducir de aquellas dependencias funcionales en R1 y R2 (proyecciones)
2. Los atributos comunes de R1 y R2 forman una clave candidata en, al menos, una de las relaciones proyectadas (R1 y R2).

Una relación que no puede ser descompuesta en proyecciones independientes se denomina atómica.

La idea del proceso de normalización es producir un conjunto de relaciones normalizadas e independientes. Pero, en algunos casos, ambos objetivos no pueden ser logrados simultáneamente y se deberá optar por una solución de compromiso.

Veamos el siguiente ejemplo:

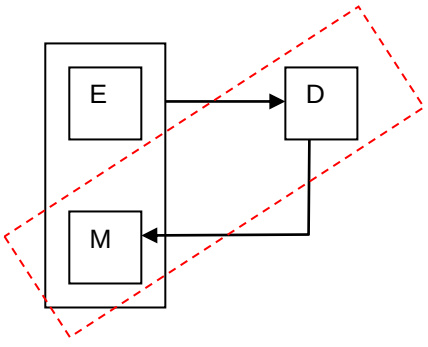
R (estudiante, materia, docente) o, en forma simplificada R (E, M, D).

El significado de cada tupla (E:e, M:m, D:d) de la relación R es que el estudiante s cursa la materia m con el docente d.

Además:

- Para cada materia, cada estudiante de la misma tiene un único docente
- Cada docente dicta solo una materia
- Cada materia puede ser dictada por varios docentes

Veamos el diagrama de dependencias funcionales:



Vemos que la dependencia D → M implica que la relación no esté en FNBC, por lo tanto con la técnica vista la proyectamos a una nueva relación:

R1 (D, M)

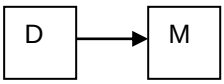
Quedando:

R2 (E, D)

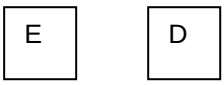
NOTA: como el atributo E no determina a D, entonces ambos conforman la clave primaria.

Veamos las dependencias funcionales de ambas relaciones:

R1:



R2:



Notar que la relación R2 no tiene dependencias funcionales no triviales.

Si se analizan estas proyecciones, se notará que no son independientes entre sí. Veamos:

- La regla 2 se cumple ya que el atributo en común D, es clave candidata en la relación R1.
- La regla 1 no se cumple ya que la dependencia funcional EM → D, no se puede deducir de las dependencias funcionales de las relaciones R1 y R2 (D → M).

Como resultado de esto, las dos proyecciones no pueden ser actualizadas en forma independiente.
Por ejemplo, el intento de insertar una tupla para el estudiante Velazquez y el docente Ramirez en la relación R2 debe ser rechazada si el docente Ramirez dicta la materia Física y el estudiante Velazquez ya cursa la materia Física con el docente Garrido, ya que no cumple la dependencia funcional EM → D.
Será necesario entonces un control adicional para esta verificación.

En conclusión, no siempre se puede lograr que las relaciones estén en FNBC y sean independientes, por lo que se deberá optar por una u otra solución.

4ª FORMA NORMAL (4FN):

Para definir la cuarta forma normal (4FN), debemos previamente definir el concepto de dependencia multivaluada.

Dependencia multivaluada (MVD): Dada una relación R, y sean A, B y C subconjuntos arbitrarios de atributos de R, se dice que B es multidependiente de A o que A multidetermina a B (A ->> B) si para cada valor de A se tiene un conjunto de valores de B.
De otra forma, el conjunto de valores de B depende solo de A y es independiente de C.
Dada esta definición se puede deducir que también se da que A ->> C, por eso es común representar esto como: A ->> B / C.

Se puede decir que la dependencia funcional es un caso especial de la dependencia multivaluada donde el conjunto de valores de B para un valor de A es tiene un único valor.

4ª Forma normal (4FN): Se puede definir de la siguiente manera: Una relación R está en 4FN si y solo si el conjunto de todas sus dependencias (funcionales y multivaluadas) son dependencias funcionales del tipo K → X, donde K es una clave candidata y X cualquier atributo de R.

Ejemplo:

Dada la relación R (materia, docente, libro) o en forma simplificada R(M, D, L), donde:

M ->> D
M ->> L
o
M ->> D / L

Es decir, una materia es dictada por varios docentes independientemente de los libros utilizados y, para una materia se utilizan varios libros independientemente de los docentes que la dictan.

Esta es una relación del tipo “todo clave”, es decir su clave primaria está compuesta por todos sus atributos:

R (M, D, L)

El proceso de llevar una relación en FNBC a 4FN se realiza proyectando el determinante con cada dependiente a relaciones separadas.

R1 (M, D)
R2 (M, L)

Una alternativa a esto es pensar a la relación como no normalizada (no está en 1FN), con clave primaria igual al determinante M. Esto se puede representar como:

R (M, {D},{L})

Los grupos repetitivos {D} y {L} son independientes. Aplicando la técnica de normalización vista para el caso de relaciones "no normalizadas", nos quedan las relaciones:

R1 (M, D)
R2 (M, L)

Las cuales están en 4FN.

Como conclusión, podemos adelantarnos al problema separando los grupos repetitivos independientes al comienzo del proceso de normalización, eliminando las dependencias multivaluadas.

5ª FORMA NORMAL (5FN):

Para definir la quinta forma normal (5FN), debemos previamente definir el concepto de dependencia de reunión.

Dependencia de reunión (JD): Dada una relación R, y sean A, B, ..., Z subconjuntos arbitrarios de conjuntos de atributos de R, se dice que R satisface la dependencia de reunión JD(A, B, ..., Z) si y solo si R es igual al join (reunión) de estas proyecciones (A, B, ..., Z).

5ª Forma normal (5FN): Una relación R está en quinta forma normal (5FN) también llamada Forma normal de proyección-reunión (FNPR o PJNF), si y solo si toda dependencia de reunión es implicada por las claves candidatas de R.

No es simple encontrar todas las dependencias de reunión de una relación. Casos en los cuales una relación está en 4FN y no en 5FN, se consideran casos patológicos y raros en la práctica.