UT_3 BASES DE DATOS RELACIONALES

3.1 Modelo de datos

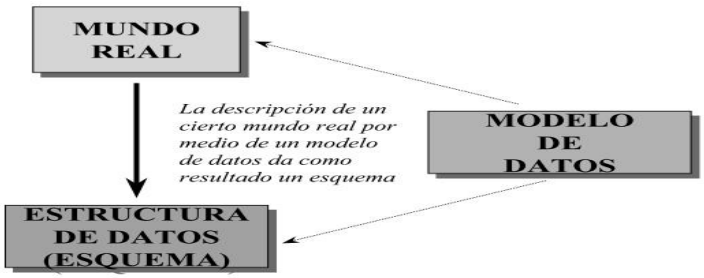
El diseño de una base de datos consiste en extraer todos los datos relevantes de un problema. Para ello hay que realizar un análisis en profundidad del problema, y así saber qué datos son importantes y cuales son descartados. Una vez extraidos los datos comienza el proceso de modelización, es decir, construir mediante herramientas de diseño un esquema que exprese con total exactitud todos los datos que el problema requiere almacenar.

Un **modelo** se puede definir como la representación de cualquier aspecto o tema extraido del mundo real. Un modelo de datos se puede definir como aquel que nos permite describir los elementos que intervienen en una realidad o en un problema dado y la forma en la que se relacionan dichos elementos entre sí.

En informática, un modelo de datos es un lenguaje utilizado para la descripción de una base de datos, nos permite describir las estructuras de los datos (tipos de datos y relaciones entre ellos), las restricciones de integridad (condiciones que deben cumplir los datos según las necesidades de nuestro modelo basado en la realidad) y las operaciones de manipulación de los datos (inserción, borrado y actualización).

Hay que distinguir entre modelo de datos y esquema:

"La descripción específica de un determinado mini-mundo en términos de un modelo de datos se denomina esquema de datos del mini-mundo. La colección de datos que representan la información acerca del mini-mundo constituye la base de datos" , Dittrich 1994.



Para clasificar los modelos tendremos en cuenta el nivel de abstracción, es decir, en lo alejado que esté del mundo real:

- **Modelo de datos conceptual,** describe las estructuras de datos y las restricciones de integridad. Se utilizan en la etapa de análisis del problema y están orientados a representar los elementos que intervienen y sus relaciones. El más utilizado es el modelo Entidad Relación.
- **Modelo de datos lógico**, se centra en las operaciones que se pueden realizar y se implementan en algún sistema gestor de base de datos. Ejemplos: el modelo Relacional, UML.
- **Modelo de datos físico**, son estructuras de datos a bajo nivel y se implementan dentro del propio SGBD.

Si trasladamos los niveles físico, conceptual y lógico a una BD relacional específica habrá un solo nivel interno y uno lógico o con

3.2 Terminología del Modelo Relacional

El modelo relacional fue propuesto por Edgar Frank Codd para IBM a finales de los 60. Es un modelo lógico que establece una estructura sobre los datos independientemente de su almacenamiento. Ejemplo: guardamos nuestra colección de libros dependiendo del número de habitaciones que tenga en casa, del tamaño y forma de nuestras estanterías, podremos disponer nuestros libros de un modo u otro para facilitar el acceso y consulta. Los libros serán los mismos pero podemos colocarlos de distinta forma. Codd se apoya en la teoría de conjuntos como base para el modelo relacional. Los datos se agrupan en **relaciones** (o tablas) y este elemento básico es el que da el nombre al modelo.

Lo que Codd pretendía era evitar que los usuarios de la BD necesitaran conocer el funcionamiento interno del sistema. Fue un enfoque revolucionario y aunque trabajaba para IBM, esta empresa no recibió de buen grado sus teorías. Fue Oracle la que empezó a implementar sus teorías y actualmente es el modelo de bases de datos más popular.

Relación o tabla. Atributos. Tuplas. Dominios.

Según el modelo relacional el elemento fundamental es la relación, también se conoce como tabla. Codd definió las relaciones utilizando un lenguaje matemático, pero se pueden asociar a la idea de tabla formada por filas y columnas. Podemos asociar los atributos a las columnas y las tuplas a las filas.

- **Atributo**, es el nombre de cada dato que se almacena en la relación. Ejemplos: DNI, nombre, apellidos, etc. El nombre que le demos al atributo debe ser significativo, es decir, indicativo de la información que representa. En una tabla con información de empleados, un atributo sueldo almacenará el valor en euros del salario que recibe cada empleado. Será necesario aclarar si se trata de bruto o neto.
- **Tupla**, representa cada elemento individual de la relación, cada fila de la tabla y se corresponde con la idea de registro. Todas las tuplas de la relación deben cumplir:
    - Cada tupla se debe corresponder con un elemento del mundo real.
    - No puede haber dos tuplas iguales (con todas las columnas iguales).
- Un atributo en una tupla no puede tomar cualquier valor. Por ejemplo, si hablamos del atributo Población, en este no se podrá guardar el dato "250€". Para evitar este tipo de situaciones

obligamos a que un atributo sólo pueda tomar valores de un conjunto previamente establecido o **dominio** de valores.

Normalmente un dominio se define mediante la declaración de un tipo para el atributo, por ejemplo la edad de una persona sería un atributo que tomaría valores del dominio formado por los números enteros, pero podría acotarse y especificar que es un número entero entre 16 y 65. O un atributo Sexo se podría asociar al dominio "M" o "F". La forma de indicar los valores mediante un intervalo se conoce como técnica por intensión y si se indican el conjunto de valores posible o una parte de ellos, estariamos definiendo el dominio por extensión.

Cada dominio tendrá un nombre y podrá estar asociado a tantos atributos como se necesite. Además tendrá una definición lógica, tipo de datos y formato. Ejemplo para el Sueldo de un empleado:

- Nombre: Sueldo.
- Definición lógica: sueldo neto del empleado.
- Tipo de datos: número entero.
- Formato: 9.999€.

El modelo relacional permite representar las relaciones por intensión, es decir, con el nombre de la tabla y a continuación entre paréntesis los atributos separados por comas. De ellos, el que corresponda a la clave primaria será subrayado.

Grado. Cardinalidad

Se llama **grado** de una relación al número de columnas (atributos) que tiene. A mayor grado, mayor complejidad para trabajar con ella.

Se llama **cardinalidad** al número de tuplas de la relación o número de filas de la tabla.

Dependiendo de la nomenclatura utilizada los conceptos anteriores tienen distintos sinónimos:

En nomenclatura relacional	En nomenclatura de tablas	En nomenclatura de ficheros
Relación	Tabla	Fichero
Tupla	Fila	registro
Atributo	Columna	Campo
Grado	Nº de columnas	Nº de campos
Cardinalidad	Nº de filas	Nº de registros

Una vez realizado el análisis del sistema y obtenidas las relaciones con las que se va a trabajar, éstas han de cumplir una serie de normas impuestas por el modelo relacional. Estas normas son las siguientes:

- Cada tabla tiene un nombre distinto.
- Cada atributo (columna) de la tabla toma un solo valor en cada tupla.
- Cada atributo tiene un nombre distinto en cada tabla pero puede repetirse en distintas tablas.

- No puede haber dos tuplas completamente iguales.
- El orden de las tuplas no importa.
- El orden de los atributos no importa
- Todos los datos de un atributo deben estar asociados al mismo dominio.

Dependiendo del uso que se vaya a dar a las relaciones éstas pueden ser:

- **Persistentes**, las que solo pueden ser borradas por los usuarios.
    - Base, independientes, se crean indicando su estructura y sus ejemplares. Contienen tanto datos como metadatos.
    - Vistas, son tablas que sólo almacenan una definición de consulta, resultado de la cual se puede obtener otra tabla con datos procedentes de otras tablas o de otras vistas. Si los datos de las tablas origen cambian, los de la vista también cambiarán.
    - Instantáneas, son vistas que sí almacenan los datos que muestran además de la consulta que las creó. Sólo modifican su resultado (actualizan los datos) cuando el sistema se refresca cada cierto tiempo.
- **Temporales**, son tablas eliminadas automáticamente por el sistema.

3.3 Tipos de datos

Una vez definidas las relaciones y en función de los datos que contenga cada atributo será necesario elegir para cada uno el tipo de dato más adecuado.

Todos los atributos se mueven dentro de un dominio. En bases de datos los dominios se corresponden con un tipo de datos. Al crear una relación decidimos qué conjunto de datos deberá ser almacenado en las filas de los atributos, es decir, tenemos que asignar un tipo de datos a cada atributo.

Cada atributo debe tener un Nombre identificativo y un tipo de dato.

Los tipos de datos más comunes son:

- Texto, almacena cadenas de caracteres (letras, símbolos o números con los que no se van a realizar operaciones).
- Numérico, guarda números con los que se realizaran operaciones matemáticas.
- Fecha/Hora.
- Si/No, datos que sólo pueden tomar dos valores Verdadero o Falso.
- Autonumérico, valor numérico secuencial que el sistema incrementa automáticamente al añadir un registro.
- Memo, almacena un texto largo.
- Moneda, es un subtipo del tipo Numérico ya que almacena números que representan cantidades de dinero.
- Objeto OLE, almacena gráficos, imágenes o textos creados por otras aplicaciones.

3.4 Claves primarias. Claves ajenas.

El modelo relacional no permite que existan dos tuplas en una relación exactamente iguales, entonces para distinguir unas tuplas de otras lo haremos mediante los valores de los atributos.

Para ello buscaremos un atributo o conjunto de atributos que identifique de modo único cada fila. A este atributo o conjunto de atributos se conoce como **superclave.**

- Claves candidatas.
- Claves primarias.
- Claves alternativas.
- Claves ajenas.

Clave candidata. Clave primaria. Clave alternativa

Una **clave candidata** es un conjunto de atributos que identifican de manera única cada tupla de la relación, es decir, las columnas cuyosvalores no se repiten en ninguna otra fila de la tabla. Una clave candidata es por tanto lo mismo que una superclave y toda tabla debe tener al menos una clave candidata.

Cuando una clave candidata está formada por más de un atributo se dice que es una **clave compuesta**.

Toda clave candidata debe cumplir los siguientes requisitos:

- **Unicidad**, es decir, no puede haber dos tuplas con los mismos valores para esos atributos.
- **Irreducibilidad**, si se elimina alguno de sus atributos deja de ser candidata.

Para identificar las claves candidatas de una relación tendremos en cuenta no sólo la situación actual, también consideraremos situaciones que se puedan producir aunque aún no se hayan producido. Para poder determinar las claves candidatas de las relaciones es fundamientasl conocer el significado real de los atributos. Por ejemplo, se puede considerar que el atributo "nombre y apellidos" es una clave candidata si nos aseguran que no existen dos clientes que tengan el mismo valor en dicho atributo, sino se descartaría como clave candidata.

La **clave primaria** de una relación es aquella clave candidata que se elige para identificar las tuplas de manera única. Siempre hay una clave candidata en una relación, por tanto siempre tendremos una clave primaria.

Una **clave alternativa** es aquella que no es elegida como primaria y pueden existir varias en una relación.

Clave externa, ajena o secundaria

Una **clave ajena** es un atributo o conjunto de atributos de una relación cuyos valores coinciden con los valores de la clave primaria de alguna otra relación (o de la misma). Las claves ajenas representan relaciones entre datos.

Ejemplo: supongamos las tablas EMPLEADOS (dni, nombre y apellidos, dirección, fecha de nacimiento, *código de departamento*) y DEPARTAMENTOS (código de departamento, nombre de departamento). Cada empleado se identifica por su dni y cada departamento por su código. Ambas tablas están relacionadas por una clave ajena que será el código de departamento, este atributo es clave ajena en la tabla EMPLEADOS y clave primaria en la tabla DEPARTAMENTOS.

Las claves ajenas no tienen las mismas restricciones que las claves primarias, por ejemplo, en la tabla EMPLEADOS un valor para el código de departamento sí puede repetirse y eso indicaría que todas las tuplas que tengan un valor común en la clave ajena pertenecen al mismo departamento.

Puede darse el caso de que un empleado aún no tenga departamento asignado y su clave ajena no tenga valor. Pero, todos los valores que en la tabla EMPLEADOS tome el código de departamento deben estar presentes en la tabla DEPARTAMENTOS.

Restricciones del modelo relacional

En todos los modelos de datos existen restricciones que a la hora de diseñar una base de datos se tienen que tener en cuenta. Los datos almacenados en la base de datos han de adaptarse a las estructuras impuestas por el modelo y deben cumplir una serie de reglas para garantizar que son correctos. El modelo relacional impone dos tipos de restricciones:

- **Restricciones impuestas por el modelo**, indican las características propias de una relación que han de cumplirse obligatoriamente y que diferencian una relación de una tabla. No hay dos tuplas iguales, el orden de las tuplas y los atributos no es relevante. Cada atributo sólo puede tomar un valor del dominio a que pertenece y ningún atributo que forme parte de una clave primaria puede ser nulo.
- **Restricciones semánticas o de usuario,** representan la semántica del mundo real. El modelo relacional proporciona una serie de mecanismos para desarrollar estas restricciones y son los siguientes:
    - la restricción de clave primaria (PRIMARY KEY) que permite declarar uno o varios atributos como clave primaria de una relación.
    - la restricción de unicidad (UNIQUE) que permite definir claves alternativas y los valores de esos atributos no pueden repetirse.
    - La restriccion de obligatoriedad (NOT NULL), que permite declara si uno o varios atributos pueden tomar valor nulo.
    - Integridad referencial o restriccion de clave ajena (FOREIGN KEY). Se utiliza para enlazar relaciones de una base de datos mediante claves ajenas. Esta norma indica que los valores de la clave ajena en la relacion hijo se corresponden con los de la clave primaria de la relación padre.

      Además de definir las claves ajenas hay que tener en cuenta las operaciones de borrado y actualización que se realizan sobre las tuplas de la relación referenciada. Las posibilidades son las siguientes:

        - Borrado y/o modificación en cascada (CASCADE). Si se borra o modifica una tupla en la relación padre (la que tiene la clave primaria) se produce un borrado o modificacion en las tuplas relacionadas en la relación hija. Si borramos un departamento, se borran los empleados de ese departamento.
        - Borrado y/o modificación restringido (RESTRICT). En este caso no es posible realizar esta operación en la relación padre si existen tuplas relacionadas en la relacion hija. Es decir, sólo se podría borrar un departamento que estuviera vacio.
        - Borrado y /o modificación con puesta a nulos (SET NULL). Esta restriccion permite poner a NULL la clave ajena en la tabla referenciada si se produce

un borrado o modificación en la tabla padre. En el ejemplo, si borramos un departamento, todos los empleados de ese departamento tomarán valor NULL en el atributo que sea la clave ajena (NUMDEP).
        - Borrado y/o modificacion con puesta a valor por defecto (SET DEFAULT). En este caso, el valor que se pone en las claves ajenas de la tabla referenciada es un valor por defecto especificado en la creación de las tablas.
    - Restricciones de verificación (CHECK). Permite especificar condiciones que deban cumplir los valores de los atributos. Cada vez que se insertan nuevos datos en la tabla o se modifican los existentes, se comprueba si los valores cumplen la condición y si no es así, se rechaza la operación

3.5 Indices. Carácterísticas y tipos

En las bases de datos, cada tabla se divide internamente en páginas de datos y se define un índice que contiene uno o varios campos y es a partir de éste desde donde comienza la búsqueda (como el índice de los capítulos de un libro).

Un índice es una estructura de datos que permite acceder a diferentes filas de una misma tabla a través de uno o más campos. Esto permite un acceso mucho más rápido a los datos. Los índices son útiles cuando se realizan consultas frecuentes a un rango de filas o una fila de una tabla. Por ejemplo, si existieran muchas consultas sobre el campo fecha de nacimiento se podría definir un índice sobre este atributo.

Los índices son independientes lógica y físicamente de los datos y por ello pueden ser creados y eliminados en cualquier momento sin afectar a otras tablas u otros índices. En una tabla no hay un límite de columnas a indexar, se podría crear uno para cada columna pero no sería operativo.

Al crear índices, las operaciones de modificar o agregar datos se ralentizan al ser necesaria la actualización tanto de la tabla como del índice. Si se elimina un índice, el acceso a los datos puede empezar a ser más lento a partir de ese momento.

Se aconseja crear índices en campos que contengan una gran cantidad de valores y que intervengan habitualmente en condiciones de las consultas (where, group by u order by).

No se aconseja crear índices en tablas pequeñas o sobre campos que no intervengan en consultas a menudo o sobre tablas que se actualizan frecuentemente.

3.6 El valor NULL

En los lenguajes de programación se utiliza el valor nulo para reflejar que un identificador (una variable, un objeto,…) no tiene ningún contenido y se trata de un valor que no permite utilizarse en operaciones aritméticas o lógicas.

Las bases de datos relacionales permiten más posibilidades para el valor nulo (null) y se puede utilizar para diversos fines aunque su significado no cambia: valor vacío o ausencia de valor.

En las claves ajenas de una relación el valor NULL indica que ese registro aún no está asociado con ninguno de la tabla padre. En otros atributos indica que la tupla carece de dicho atributo. Por

ejemplo en nuestra tabla EMPLEADOS si aún no conocemos la fecha de nacimiento podríamos darle valor NULL.

Es importante diferenciar el valor NULL del carácter ' ' o del valor numérico 0. No representa ningún valor.

El valor nulo se utiliza frecuentemente en bases de datos. En la lógica booleana existen dos valores lógicos: verdadero y falso y el valor nulo no es ninguno de estos. Existen una serie de operadores lógicos: and, or y not. Al operar el valor nulo con valores lógicos mediante operadores lógicos tendremos los siguientes resultados:

- Verdadero AND nulo => nulo
- Falso AND nulo=> falso
- Verdadero OR nulo =>verdadero
- Falso OR nulo => nulo
- NOT nulo = nulo

En todas las bases de datos relacionales existe un operador especial "IS NULL" que devuelve verdadero si el valor que se compara es nulo y falso en caso contrario.

3.7 Vistas

Una vista se define como una tabla virtual cuyas filas y columnas se obtienen a partir de una o varias tablas. Lo que se almacena no es la tabla en sí (los datos) sino su definición, por eso se dice que es virtual. Toda vista viene definida por una consulta que se realiza sobre una o varias tablas, o sobre otras vistas o sobre otras bases de datos.

Una vez creada, la vista se puede utilizar como otra tabla más.

Existen dos razones fundamentales para utilizar vistas:

- Por seguridad, nos permite dar acceso a los usuarios a parte de la información que hay en una tabla y no a la tabla completa.
- Por comodidad, en la definición de una vista se puede guardar una sentencia de consulta compleja que luego será muy fácil de utilizar con la vista.

Las vistas no tienen una copia física de los datos, por tanto, si actualizamos los datos de una vista, estamos actualizando realmente la tabla física. Igualmente si se actualizan los datos de la tabla, estos cambios serán visibles desde la vista ya que no tiene contenido estático.

3.8 Gestión de la seguridad. Usuarios, roles y privilegios

A la hora de conectarnos a la base de datos necesitamos disponer de una cuenta de usuario, esta cuenta además de nombre de usuario y contraseña lleva asociado un modo de acceso donde quedan descritos los permisos del usuario durante la conexión.

Un **usuario** es un conjunto de permisos que se aplican a una conexión de base de datos. Cada usuario será propietario de los objetos que se crean en su esquema y será el administrador de la base de datos el que conceda los permisos a los distintos usuarios.

Un **privilegio** es un permiso que se concede a un usuario para que pueda realizar ciertas operaciones sobre la base de datos. Los privilegios pueden ser:

- **De sistema**, le permitirá determinado permiso sobre el sistema.
- **De objeto**, dispondrá de un permiso sobre determinado objeto.

Los permisos, tanto de sistema como de objeto, se pueden agrupar en lo que se denomina **rol**.

Supongamos un grupo de 10 personas, todas con permiso de consulta sobre una tabla. Si queremos añadir el permiso de actualización sobre la tabla sería necesario ir usuario a usuario, si lo hacemos mediante un rol, sólo sería necesario añadirle el permiso al rol.

3.9 Paso del modelo Entidad Relación al Modelo Relacional

Una vez planteado el problema a modelar, hemos diseñado el correspondiente **diseño conceptual** que se materializa en el esquema Entidad/Relación. Este esquema conceptual habrá sido revisado para comprobar que se cumplen adecuadamente todos y cada uno de los requerimientos del problema a modelar. Entonces, este esquema será el punto de partida para la siguiente fase, el **diseño lógico** de la base de datos.

El diseño lógico consistirá en la construcción de un esquema de la información relativa al problema, basado en un modelo de datos concreto. El esquema conceptual se transformará en un esquema lógico que utilizará los elementos y características del modelo de datos elegido. Podremos optar por: modelo en red, modelo jerárquico y el más usado, el modelo relacional.

El esquema E/R obtenido será simplificado y transformado para que el paso al modelo lógico elegido sea más sencillo. Esta transformación se hará aplicando determinadas reglas y condiciones que garanticen la equivalencia entre el esquema conceptual y el lógico.

A continuación, se realiza la **normalización** del esquema lógico obtenido aplicando para ello un conjunto de técnicas que permiten validar esquemas lógicos basados en el modelo relacional.

Una vez llegados a este punto, el esquema lógico obtenido se podrá implementar físicamente en cualquier SGBD basado en el modelo relacional y esta implementación sí será dependiente de las características del SGBD elegido.

Simplificación previa del diagrama

Si aplicamos un conjunto de normas a nuestro diagrama E/R para transformarlo en el modelo lógico relacional, el proceso será casi automático, será fácil y fiable. Estas transformaciones son las siguientes:

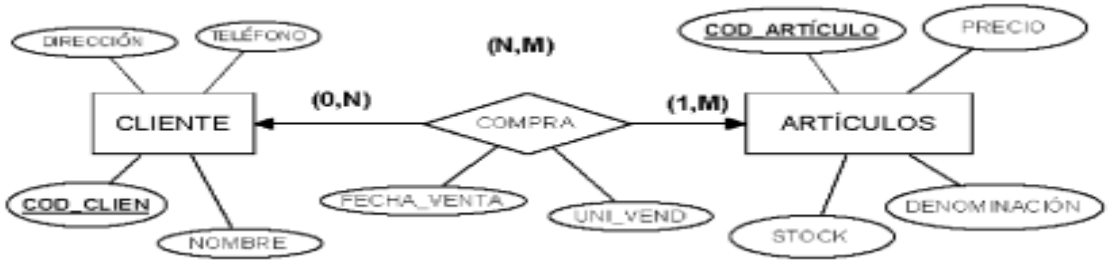
- Transformación de atributos compuestos en los atributos simples que los forman.
- Transformación de atributos multivaluados. Si se da este caso, dicho atributo debe convertirse en una entidad relacionada con la entidad de la que procede. Esta entidad tendrá un solo atributo, el antiguo atributo múltiple, que es posible que funcione correctamente como clave primaria de la entidad o puede ser que la nueva entidad sea débil.

Paso del diagrama E/R al modelo Relacional

Si hemos aplicado las simplificaciones anteriores, tendremos un esquema E/R que sólo contiene entidades fuertes y con sus atributos no compuestos ni multivaluados. Ahora la transformación es casi automática y tendremos en cuenta las siguientes características:

- Toda entidad se transforma en una tabla.
- Todo atributo se transforma en columna dentro de la tabla.
- El atributo clave de la entidad se convierte en clave primaria de la tabla y se representa subrayado.
- Cada entidad débil generará una tabla que incluirá todos sus atributos, añadiéndose a esta los atributos que son clave primaria de la entidad fuerte con la que está relacionada. Estos atributos serán clave ajena que referencia a la entidad fuerte.
- Toda relación N: M se transforma en una tabla que tendrá como clave primaria la concatenación de los atributos clave de las entidades que asocia.

Ejemplo: compras de artículos que hacen los clientes



Cada entidad se convierte en una tabla y los atributos de las entidades se convierten en columnas.

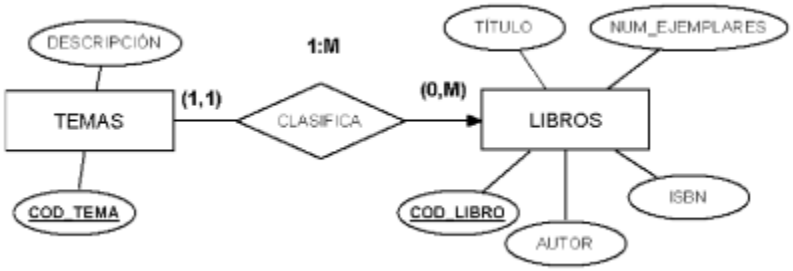
CLIENTE (CodCliente, Nombre, Dirección, Tfno)
ARTICULOS (CodArticulo, Precio, Stock, Descripción)

La relación N:M se convierte a tabla cuyo nombre será el de la relación, su clave primaria será la concatenación de las claves de las dos entidades que pasarán a ser claves ajenas y referencian a las tablas CLIENTE y ARTICULO. Además se añaden los atributos que intervienen en la relación.

COMPRA (CodCliente (FK), CodArticulo (FK), FechaVenta, UniVendidas)

- En la transformación de relaciones 1:N existen dos soluciones:
    - Transformar la relacion en una tabla como si se tratara de una N:M. Esta solucion se toma si la relación tiene atributos propios o si la cardinalidad es opcional, es decir, (0,1) y (0,n). La clave de esta nueva tabla será la de la entidad del lado N.
    - Propagar la clave, este caso se aplicac cuando la cardinalidad es obligatoria, es decir, cuando tenemos (1,1) y (0,n) o (1, n). Se propaga el atributo principal de la entidad que tiene la cardinalidad máxima 1 a la que tiene cardinalidad N. si existen atributos propios de la relación, éstos también se propagan.

Ejemplo: supongamos una clasificacion de los libros en temas.



Las dos soluciones serían:

- Se convierten a tabla las dos entidades:

TEMAS (CodTema, Descripción)
LIBROS (CodLibro, Autor, Isbn, Título, NumEjemplares)

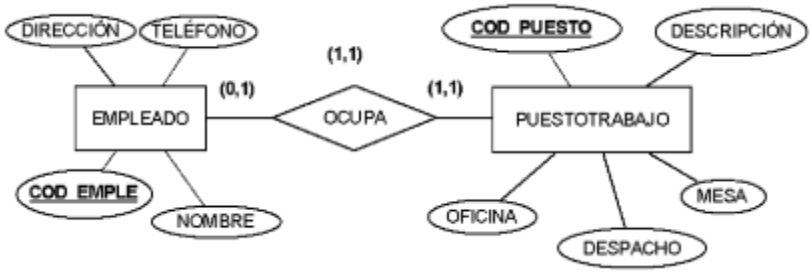
- Se propaga la clave de la entidad TEMAS a la entidad LIBROS y esta quedaría:

LIBROS (CodLibro, Autor, Isbn, Título, NumEjemlares, CodTema (FK))

- En las relaciones 1:1 se tienen en cuenta las cardinalidades de las entidades que participan. Existen dos soluciones:

  - Transformar la relación en una tabla si las entidades poseen cardinalidades (0,1).
  - Propagar la clave. Si una de las entidades posee cardinalidad (0,1) y la otra (1,1), conviene propagar la clave de la entidad que participa con (1,1) a la otra. Si ambas entidades poseen cardinalidad (1,1), se puede propagar la clave de cualquiera de ellas a la tabla resultante de la otra.

Supongamos la relación que representa EMPLEADO-ocupa-PUESTOTRABAJO. Un empleado solo ocupa un puesto de trabajo, y un puesto de trabajo es ocupado por un solo empleado o por ninguno.



En este caso, la clave se propaga desde la entidad PUESTOTRABAJO, con cardinalidad (1,1) a la entidad EMPLEADO y las tablas nos quedarán:

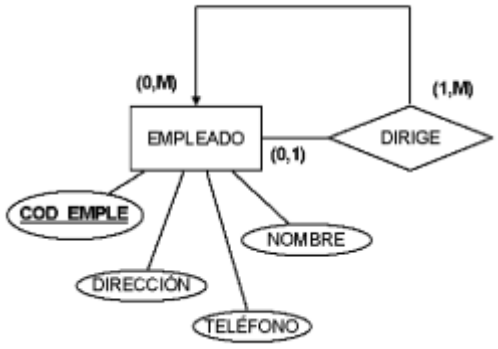
EMPLEADO (CodEmple, Nombre, Direccion, Tfno, CodPuesto(FK))
PUESTOTRABAJO (CodPuesto, Descripción, Oficina, Despacho, Mesa)

- En las relaciones reflexivas o recursivas tendremos en cuenta la cardinalidad. Normalmente es que toda relación reflexiva se convierta en dos tablas, una para la entidad y otra para la relación. Se pueden presentar los siguientes casos:

    - Si la relación es 1:1, la clave de la entidad se repite, por tanto, la tabla resultante tendrá dos veces ese atributo, una como clave primaria y otra como clave ajena a ella misma. No se crea una segunda tabla.
    - Si la relación es 1:N, podemos tener dos casos:

        - Si la entidad muchos es obligatoria, se procede como en el caso 1:1.
        - Si no es obligatoria, se crea una nueva tabla cuya clave será la de la entidad del lado muchos, y además se propaga la clave a la nueva tabla como clave ajena.

Supongamos la relación EMPLEADO-dirige-EMPLEADO en la que un empleado puede dirigir a muchos empleados o a ninguno y un empleado es dirigido por un director o por ninguno si es el que dirige. En este caso no hay obligatoriedad en la entidad muchos.

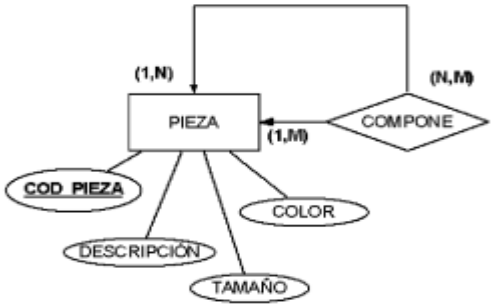


La tabla DIRIGE tiene como clave primaria el código del empleado, que a su vez será clave ajena. Además se le añade el código de empleado, pero en este caso tendrá papel de director, que a su vez será clave ajena a la tabla EMPLEADO. El resultado será el siguiente:

EMPLEADO (CodEmple, Nombre, Dirección, Tfno)
DIRIGE (CodEmple (FK), CodDirec (FK))

- Si es N:M, se trata igual que en las relaciones binarias. La tabla resultante de la relacion contendrá dos veces la clave primaria de la entidad del lado muchos, más los atributos de la relación si los hubiera. La clave de esta nueva tabla será la combinación de las dos.

Supongamos la siguiente relación en la que una pieza se compone de muchas piezas que a su vez están compuestas de otras piezas.



En este caso, obtenemos la segunda tabla COMPONE_PIEZA en la que aparecerá repetido el código de pieza y formará la clave de la tabla. El atributo CodPiezaCom representa la pieza que se compone de otras y CodPieza tiene el papel de pieza que compone a otras piezas. Ambos atributos son claves ajenas de la tabla PIEZA. Las tablas resultantes serían:

PIEZA (CodPieza, Descripción, Tamaño, Color)
COMPONE_PIEZA (CodPiezaCom (FK), CodPieza (FK))

- En las generalizaciones como el modelo relacional no dispone de mecanismos para la representación de relaciones jerárquicas, éstas han de eliminarse. Para ello hay que tener en cuenta:
  - La especialización que los subtipos tienen respecto a los supertipos, es decir, los atributos diferentes que tenga cada subtipo.
  - El tipo de especialización: total o parcial exclusiva y total o parcial solapada.
  - Otros tipos de relación que mantengan tanto el supertipo como los subtipos.

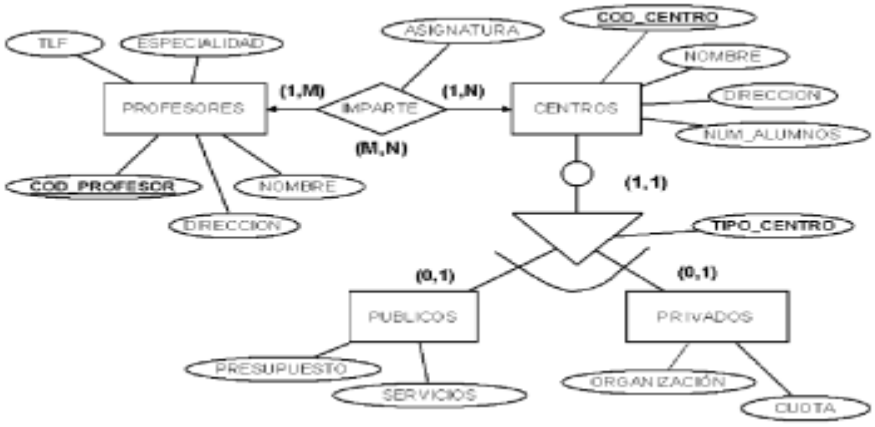
Teniendo en cuenta lo anterior, para realizar el paso de estas relaciones al modelo relacional se aplicará una de las siguientes reglas:

A. Integrar todas las entidades en una sola eliminando los subtipos. Esta nueva relación tendrá todos los atributos del supertipo, todos los de los subtipos y los discriminantes que permiten distinguir a que subtipo pertenece cada atributo. Todas las relaciones se mantienen con la nueva entidad. Esta norma puede aplicarse a cualquier jerarquía. Su ventaja es que todo se reduce a una sola entidad, pero el gran inconveniente es la cantidad de valores nulos en los atributos opcionales de cada entidad.

B. Eliminación del supertipo, transfiriendo los atributos del supertipo a cada uno de los subtipos. Las relaciones del supertipo se consideran para cada uno de los subtipos. La clave genérica del supertipo pasa a cada subtipo. Esta regla sólo puede ser aplicada a jerarquías totales y exclusivas. Los inconvenientes de esta transformación son: se crea redundancia de información ya que los atributos del supertipo se repiten en cada uno de los subtipos y el número de relaciones aumenta.

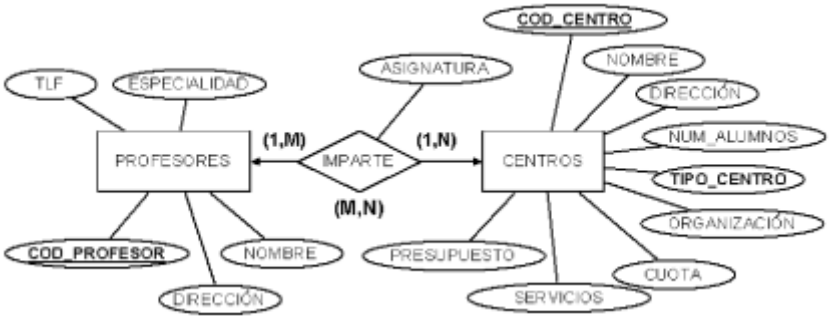
C. Insertar una relación 1:1 entre el supertipo y cada uno de los subtipos. Los atributos se mantienen y cada subtipo se identificará con la clave ajena del supertipo. El supertipo mantendrá una relación 1:1 con cada subtipo. Los subtipos mantendrán la cardinalidad mínima a 0 si es la relación exclusiva o a 0 o 1 si es solapada.

Supongamos un ejemplo en el que hay profesores que imparten clase en dos tipos de centros educativos: públicos y privados. Un profesor puede impartir clase en varios centros, ya sean públicos o privados. Consideremos la asignatura como atributo de la relación entre profesor imparte en centro. Los centros sólo pueden ser de esos dos tipos. Un centro público no puede ser a la vez privado. Los atributos específicos para los centros públicos son el presupuesto y los servicios, para los privados son la organización y la cuota. La jerarquía es total y exclusiva. Gráficamente quedaría:



A la entidad CENTROS se añade el atributo TIPO_CENTRO como discriminante que identifique el subtipo. Al ser jerarquía total y exclusiva, podemos aplicar cualquiera de las tres reglas anteriores:

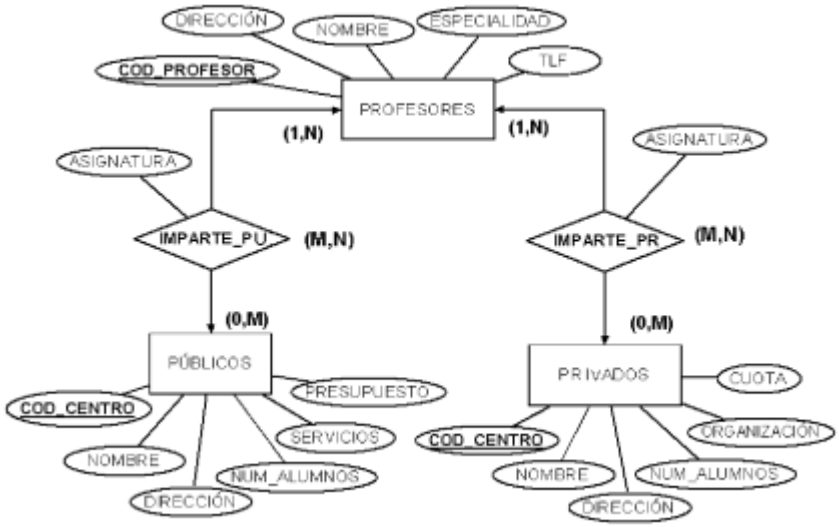
A. Integramos los subtipos en el supertipo.



El modelo relacional contará con las siguientes tablas:

PROFESORES (CodProfesor, Dirección, Nombre, Teléfono, Especialidad)
CENTROS (CodCentro, Nombre, Dirección, NumAlumnos, TipoCentro, Organización, Cuota, Servicios, Presupuesto)
IMPARTE (CodProfesor, CodCentro, Asignatura)

B. Eliminación del supertipo.



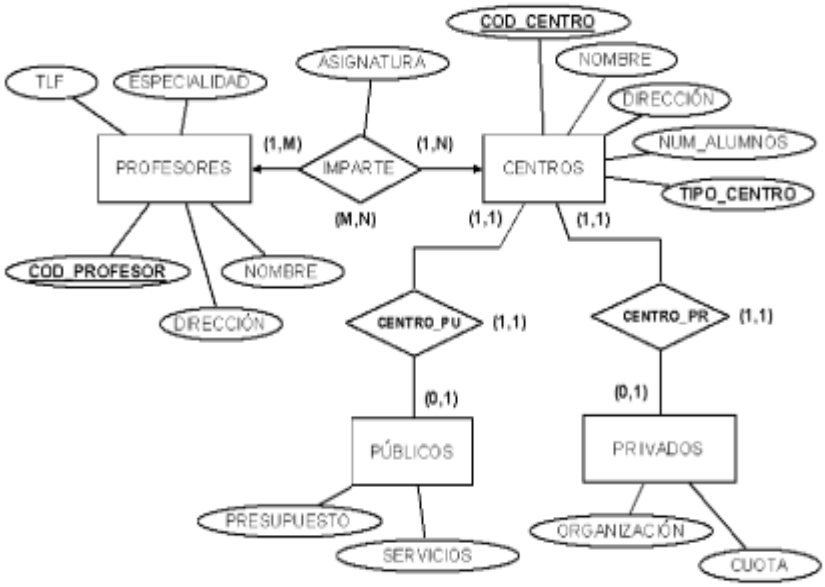
El atributo TipoCentro se elimina y el modelo queda con las siguientes tablas:

PROFESORES (CodProfesor, Dirección, Nombre, Teléfono, Especialidad)
PUBLICOS (CodCentro, Nombre, Dirección, NumAlumnos, Servicios, Presupuesto)
PRIVADOS (CodCentro, Nombre, Dirección, NumAlumnos, Organización, Cuota)
IMPARTE_PR (CodProfesor (FK), CodCentro (FK), Asignatura)
IMPARTE_PU (CodProfesor (FK), CodCentro (FK), Asignatura)

En este esquema relacional podría darse el caso de que un centro pueda estar en las tablas PUBLICOS y PRIVADOS, este supuesto va en contra del enunciado. Sería necesario controlar esta situación con algún mecanismo, por ejemplo, un disparador asociado a cada tabla.

Las tablas IMPARTE_PR e IMPARTE_PU no se pueden agrupar en una sola ya que el código de centro es clave ajena en distintas tablas.

C. Insertar una relación 1:1 entre el supertipo y cada uno de los subtipos.



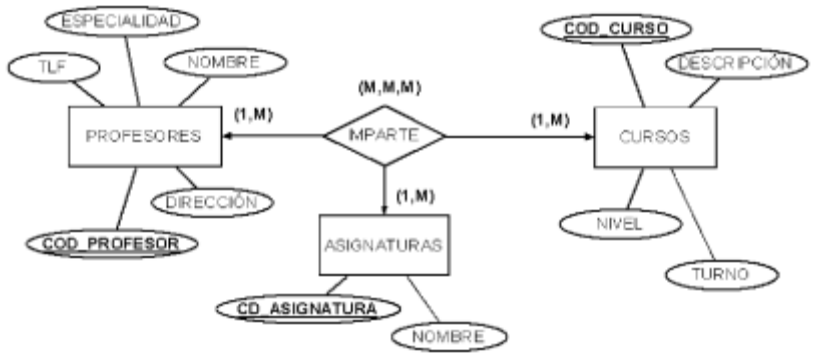
El atributo TipoCentro se incluye en el supertipo. El resultado es el siguiente:

PROFESORES (CodProfesor, Dirección, Nombre, Teléfono, Especialidad)
CENTROS (CodCentro, Nombre, Dirección, NumAlumnos, TipoCentro)
PUBLICOS (CodCentro (FK), Servicios, Presupuesto)
PRIVADOS (CodCentro (FK), Organización, Cuota)
IMPARTE (CodProfesor (FK), CodCentro (FK), Asignatura)

- Para las relaciones N-arias que agrupan 3 o más entidades, cada entidad se convierte en tabla y también la relación que contendrá sus atributos propios más las claves de todas las entidades. La clave principal será la concatenación de las claves de las entidades. Pueden darse dos casos dependiendo de las cardinalidades:

  - Si la relación es N:M:N, es decir todas las entidades participan con cardinalidad máxima M, la clave de la tabla resultante es la unión de las claves de las entidades que relaciona

Supongamos una relación ternaria entre las entidades PROFESORES-CURSOS-ASIGNATURAS, en la que un profesor imparte en varios cursos varias asignaturas y además las asignaturas son impartidas por varios profesores en varios cursos.

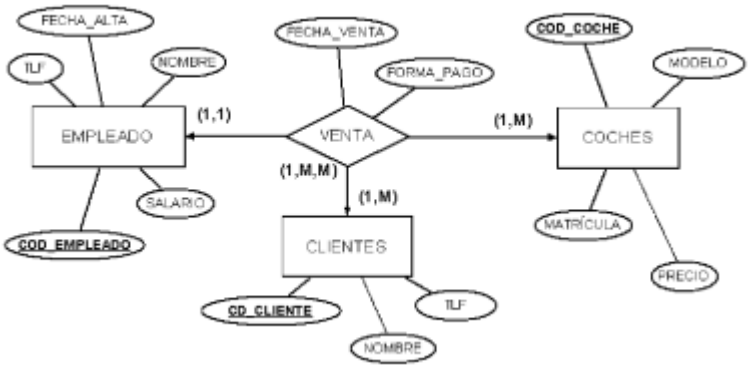


El resultado en el modelo relacional será:

PROFESORES (CodProfesor, Dirección, Nombre, Teléfono, Especialidad)
CURSOS (CodCurso, Descripción, Nivel, Turno)
ASIGNATURAS (CodAsignatura, Nombre)
IMPARTE (CodProfesor (FK), CodCurso (FK), CodAsignatura (FK))

  - Si la relación es 1:N:M, es decir una de las entidades participa con cardinalidad máxima 1, la clave de esta entidad no pasa a formar parte de la clave de la tabla resultante, pero forma parte de la relación como un atributo más.

Supongamos el caso de una tienda de venta de coches en la que un empleado vende muchos coches a muchos clientes y los coches son vendidos por un solo empleado. En la venta hay que tener en cuenta la forma de pago y la fecha de venta.



El resultado en el modelo relacional es:

CLIENTES (CodCliente, Nombre, Tfno)
EMPLEADO (CodEmpleado, Nombre, Tfno, Salario, FechaAlta)
COCHES (CodCoche, Matrícula, Modelo, Precio)
VENTA (CodCoche (FK), CodCliente (FK), CodEmpleado (FK), FormaPago, FechaVenta)

3.10 Normalización del modelo relacional

La normalización es una técnica para diseñar la estructura lógica de los datos de un sistema de información en el modelo relacional. Fue desarrollada por Codd en 1972 y es una estrategia de diseño de abajo arriba: se parte de los atributos y éstos se van agrupando en tablas según su afinidad. Vamos a utilizar esta técnica para eliminar las dependencias no deseadas entre atributos. Este proceso conseguirá los siguientes objetivos:

- Suprimir dependencias erróneas entre atributos.
- Optimizar los procesos de inserción, borrado y modificación en la base de datos.

El proceso de normalización se basa en el análisis de las dependencias entre atributos. Para ello tendremos en cuenta los conceptos de: **dependencia funcional**, **dependencia funcional completa** y **dependencia transitiva**.

La normalización se realiza en varias etapas secuenciales, cada una asociada a una **forma normal** que establece unos requisitos a cumplir por la tabla sobre la que se aplica. Existen varias formas normales: **Primera, Segunda, Tercera, Boyce-Codd, Cuarta, Quinta y Dominio-Clave.** El proceso es secuencial y si una tabla no satisface una determinada forma normal no se pasa al análisis de la siguiente. Con cada forma normal que se cumple, el modelo se hace más robusto.

Es recomendable aplicar el proceso hasta Tercera Forma Normal o hasta Boyce-Codd.

Tipos de dependencias

Los tipos de dependencias entre atributos son los siguientes:

- **Dependencia funcional,** dados los atributos A y B de una relación, se dice que B depende funcionalmente de A o que A determina a B si y solo si, para cada valor de A sólo puede existir un valor de B. La dependencia funcional siempre se establece entre atributos de la

misma tabla. El atributo A se llama determinante y esta dependencia se representa como **A -> B.** A y B podrían ser un atributo o un conjunto de ellos.

- **Dependencia funcional completa,** dados los atributos A1, A2, …Ak y B, se dice que B depende funcionalmente de forma completa de A1, A2, …Ak si y solo si B depende funcionalmente del conjunto de atributos A1, A2, …Ak, pero no de ninguno de sus posibles subconjuntos.
- **Dependencia transitiva,** dados tres atributos A, B y C, se dice que existe una dependencia transitiva entre A y C, si B depende funcionalmente de A y C depende funcionalmente de B. A, B y C podrían ser un solo atributo o un conjunto de ellos.

Ejemplo:

Dadas las siguientes tablas:

EMPLEADO (DNI, Nombre, Dirección, Localidad, CodLocalidad, NombreHijo, EdadHijo)
LIBROS (TituloLibro, NumEjemplar, Autor, Editorial, Precio)

Se pide resolver las siguientes cuestiones:

a) Qué atributos presentan una dependencia funcional de la clave primaria de la tabla EMPLEADO.
b) Qué atributos presentan una DF completa en la tabla LIBROS.
c) Qué atributos presentan una DF transitiva en la tabla EMPLEADOS.

Solución:

a) Los atributos Nombre, Dirección, Localidad y CodLocalidad dependen funcionalmente de DNI, ya que para un DNI concreto sólo puede haber un Nombre, una Dirección, una Localidad y un CodLocalidad. Los atributos NombreHijo y EdadHijo no dependen funcionalmente de DNI ya que par un valor de DNI podríamos tener varios valores diferentes de estos atributos.

DNI->Nombre, DNI->Dirección, DNI->Localidad, DNI->CodLocalidad

b) Los atributos Editorial y Precio dependen funcionalmente del conjunto de atributos que forman la clave primaria de la tabla, pero no dependen de TituloLibro o de NumEjemplar por separado, por tanto es una DF completa. El atributo Autor depende funcionalmente sólo y exclusivamente de TituloLibro por lo que no presenta una DF completa de los atributos que forman la clave.
c) Los atributos CodLocalidad y Localidad dependen funcionalmente de DNI, pero entre estos atributos existe otra DF, por tanto Localidad depende de CodLocalidad y a su vez CodLocalidad depende de DNI. Entonces, existe una dependencia transitiva entre Localidad y DNI. Se representará como:

DNI -> CodLocalidad ->Localidad

Formas Normales

El proceso de Normalización supone llevar a cabo una serie de etapas consecutivas en las que se aplicarán las propiedades de cada forma normal definida por Codd. Se dice que una relación está en una forma normal si satisface un cierto conjunto específico de restricciones impuestas por la

regla de normalización correspondiente. La aplicación de una norma es una operación que toma como entrada una relación y da como resultado una o más relaciones.

1ª Forma Normal

Una relación está en 1ª Forma Normal (1FN) si y solo si, todos sus atributos contienen valores atómicos, es decir, cada atributo de la relación toma un único valor del dominio correspondiente, por tanto no hay grupos repetitivos. De otra manera, la tabla sólo tiene un valor en cada intersección de fila y columna.

Habrá que eliminar los grupos repetitivos y hay dos formas de hacerlo:

- Repetir los atributos con un solo valor para cada valor del grupo repetitivo. Se introducen redundancias al duplicar valores y serán eliminadas mediante las restantes formas normales.
- Poner cada grupo repetitivo en una relación aparte, heredando la clave primaria de la tabla inicial.

Si en la fase de análisis se ha elegido bien la clave primaria la tabla ya se encontrará en 1FN.

Supongamos la tabla ALUMNO, su clave primaria CodAlumno en la que el atributo Tfno puede tomar varios valores:

COD_ALUMNO	NOMBRE	APELLIDO	TLF	DIRECCION
1111	PEPE	GARCÍA	678-900800 91-2233441 91-1231232	C/Las cañas 45
2222	MARÍA	SUÁREZ	91-7008001	C/Mayor 12
3333	JUAN	GIL	91-7562324 660-111222	C/La plaza
4444	FRANCISCO	MONTOYA	678-556443	C/La arboleda

Esta tabla no está en 1FN, para que lo esté podemos elegir dos caminos:

- Definir como clave primaria de la tabla CodAlumno y Tfno, así no habrá tuplas repetidas.

COD_ALUMNO	TLF	NOMBRE	APELLIDO	DIRECCIÓN
1111	678-900800	PEPE	GARCÍA	C/Las cañas 45
1111	91-2233441	PEPE	GARCÍA	C/Las cañas 45
1111	91-1231232	PEPE	GARCÍA	C/Las cañas 45
2222	91-7008001	MARÍA	SUÁREZ	C/Mayor 12
3333	91-7562324	JUAN	GIL	C/La plaza
3333	660-111222	JUAN	GIL	C/La plaza
4444	678-556443	FRANCISCO	MONTOYA	C/La arboleda

- Eliminar los grupos repetitivos, Tlfno, y crear una tabla nueva con este atributo junto con la clave inicial.

COD_ALUMNO	NOMBRE	APELLIDO	DIRECCION
1111	PEPE	GARCÍA	C/Las cañas 45
2222	MARÍA	SUÁREZ	C/Mayor 12
3333	JUAN	GIL	C/La plaza
4444	FRANCISCO	MONTOYA	C/La arboleda

La segunda tabla será:

COD_ALUMNO(FK)	TLF
1111	678-900800
1111	91-2233441
1111	91-1231232
2222	91-7008001

2ª Forma Normal

Se dice que una relación se encuentra en 2FN si y solo si, está en 1FN y cada atributo de la relación que no está en la clave depende de manera completa de la clave primaria de la relación. La 2FN sólo se aplica a las relaciones que tienen las claves primarias compuestas por más de un atributo, cualquier tabla que esté en 1FN y cuya clave primaria sólo contenga un atributo, estará en 2FN. Para normalizar a 2FN:

- Se crea, a partir de la tabla inicial, una tabla con los atributos que dependen completamente de la clave primaria. La clave primaria será la misma de la tabla inicial y ésta ya estará en 2FN.
- Con los atributos restantes, se crea otra tabla que tendrá por clave el subconjunto de atributos de la clave inicial de los que dependen de forma completa. Si esta tabla está en 2FN el proceso termina, si no la tomaremos como tabla inicial y continuaremos el proceso.

Dada la relación ALUMNO en la que representamos los datos de los alumnos y las notas en cada asignatura en la que está matriculado. La clave primaria es el CodAlumno y la Asignatura.

COD_ALUMNO	NOM_ALUM	APE_ALUM	ASIGNATURA	NOTA	CURSO	AULA
1111	PEPE	GARCÍA	LENGUA I	5	1	15
1111	PEPE	GARCÍA	IDIOMA	5	2	16
2222	MARÍA	SUAREZ	IDIOMA	7	2	16
2222	MARÍA	SUAREZ	CIENCIAS	7	2	14
3333	JUAN	GIL	PLÁSTICA	6	1	18
3333	JUAN	GIL	MATEMÁTICAS I	6	1	12
4444	FRANCISCO	MONTOYA	LENGUA II	4	2	11
4444	FRANCISCO	MONTOYA	MATEMÁTICAS I	6	1	12
4444	FRANCISCO	MONTOYA	CIENCIAS	8	1	14

Todos los atributos no dependen de la clave completa (CodAlumno, Asignatura). Las dependencias son las siguientes:

- CodAlumno -> NomAlumno, ApeAlumno
- Asignatura - > Curso, Aula, una asignatura pertenece a un curso y se imparte en un aula.
- (CodAlumno, Asignatura) - > Nota, para que exista una nota debe haber un alumno y una asignatura.

Por tanto, para que la relación ALUMNO esté en 2FN, debemos crear tres relaciones: ALUMNO, ASIGNATURA y NOTAS.

ALUMNO

COD_ALUMNO	NOM_ALUM	APE_ALUM
1111	PEPE	GARCÍA
2222	MARÍA	SUAREZ
3333	JUAN	GIL
4444	FRANCISCO	MONTOYA

NOTAS

COD_ALUMNO (FK)	ASIGNATURA (FK)	NOTA
1111	LENGUA I	5
1111	IDIOMA	5
2222	IDIOMA	7
2222	CIENCIAS	7
3333	PLÁSTICA	6
3333	MATEMÁTICAS I	6
4444	LENGUA II	4
4444	MATEMÁTICAS I	6
4444	CIENCIAS	8

ASIGNATURA

ASIGNATURA	CURSO	AULA
LENGUA I	1	15
IDIOMA	2	16
CIENCIAS	2	14
PLÁSTICA	1	18
MATEMÁTICAS I	1	12
LENGUA II	2	11

3ª Forma Normal

Una tabla está en 3FN si y solo si, está en 2FN y además cada atributo que no está en la clave primaria no depende transitivamente de la clave primaria. Es decir, los atributos de la relación no dependen unos de otros, dependen sólo de la clave primaria. Para normalizar a 3FN:

- Se crea, a partir de la tabla inicial, una nueva tabla con los atributos que no poseen dependencias transitivas de la clave primaria y ésta ya estará en 3FN.
- Con los atributos restantes, se crea otra tabla con los dos atributos no clave que intervienen en la dependencia transitiva, y se elige el determinante como clave primaria. Se comprueba si la tabla resultante ya está en 3FN o no y se repite el proceso si fuera necesario.

Teorema de la 3FN, sea una relación formada por los atributos A, B y C, con clave primaria A. Si se cumple que B->C, entonces la relación puede descomponerse en otras dos R1 y R2, de la siguiente forma R1(A, B) y R2 (B, C).

Supongamos la relación LIBROS en la que representamos los datos de sus editoriales:

COD_LIBRO	TÍTULO	EDITORIAL	PAÍS
12345	DISEÑO DE BD RELACIONALES	RAMA	ESPAÑA
34562	INSTALACIÓN y MANTENIMIENTO DE EQUIPOS	MCGRAW-HILL	ESPAÑA
72224	FUNDAMENTOS DE PROGRAMACIÓN	SANTILLANA	ESPAÑA
34522	BASE DE DATOS OO	ADDISON	EEUU

Las dependencias respecto a la clave son:

- Título y Editorial dependen directamente del código del libro.
- El País depende del libro, pero está más ligado a la Editorial. Por esta razón, la relación no está en 3FN.

La solución será:

Tabla LIBROS en 3FN

COD_LIBRO	TÍTULO	EDITORIAL (FK)
12345	DISEÑO DE BD RELACIONALES	RAMA
34562	INSTALACIÓN y MANTENIMIENTO DE EQUIPOS	MCGRAW-HILL
72224	FUNDAMENTOS DE PROGRAMACIÓN	SANTILLANA
34522	BASE DE DATOS OO	ADDISON

Tabla EDITORIAL

EDITORIAL	PAÍS
RAMA	ESPAÑA
MCGRAW-HILL	ESPAÑA
SANTILLANA	ESPAÑA
ADDISON	EEUU

Actividad propuesta: dada la siguiente tabla, normalizarla hasta 3FN

DNI	NOMBRE	APELLIDOS	DIRECCIÓN	C_POST	POBLACIÓN	PROVINCIA
413245-B	JUAN	RAMOS	C/Las cañas 59 C/Pilón 12	19005 45589	GUADALAJARA CALERUELA	GUADALAJARA TOLEDO
23456-J	PEDRO	PÉREZ	C/Vitoria 3 C/El altozano	28804 10392	ALCALÁ DE HENARES BERROCALEJO	MADRID CÁCERES
34561-B	MARÍA	RODRÍGUEZ	C/Sanz Vázquez 2	19004	GUADALAJARA	GUADALAJARA
222346-J	JUAN	CABELLO	C/El ensanche 3 C/Los abedules 10	28802 10300	ALCALÁ DE HENARES NAVALMORAL DE LA MATA	MADRID CÁCERES

Forma Normal de Boyce Codd

Una tabla está en Forma Normal de Boyce Codd (FNBC) si y solo si, está en 3FN y todo determinante es una clave candidata. Un determinante será todo atributo simple o compuesto del que depende funcionalmente de forma completa algún otro atributo de la tabla. Si encontramos un determinante que no es clave candidata, la tabla no estará en FNBC. Una tabla en la que todos los atributos forman parte de la clave primaria estará en FNBC.

La 2FN y la 3FN eliminan las dependencias parciales y las dependencias transitivas de la clave primaria. Pero este tipo de dependencias pueden existir sobre otras claves candidata si las hubiera. Hay que comprobar si una relación incumple la FNBC si tiene dos o más claves candidatas compuestas solapadas, es decir, con al menos un atributo en común.

Teorema Boyce – Codd: sea una relación R formada por los atributos A, B, C y D, con claves candidatas compuestas (A, B) y (B, C) tal que A ->C, entonces la relación puede descomponerse en cualquiera de las dos siguientes maneras:

- R1 (A, C) y R2 (B, C, D)
- R1 (A, C) y R2 (A, B, D)

Supongamos una relación EMPLEADOS que guarda información de los empleados de una fábrica:

DNI	NÚM_SEG_SOC	NOMBRE	APELLIDOS	DEPARTAMENTO	PUESTO	SALARIO
413245-B	28-1234566	JUAN	RAMOS	COMPRAS	GERENTE	2.300
23456-J	28-2345686	PEDRO	PÉREZ	NÓMINAS	AUXILIAR	1.200
123123-C	19-458766	MARÍA	GIL	ALMACÉN	CONSERJE	1.530
1234556-B	45-223344	ANTONIO	SANZ	COMPRAS	GESTIÓN	2.200

En esta relación DNI, NumSegSocial y el grupo Nombre y Apellidos son claves candidatas. Esta relación se puede descomponer en otras dos: EMPLEADOS, con la clave junto con las claves candidatas y EMPLETRABAJO, con la clave primaria y el resto de los campos.

EMPLEADOS

DNI	NUM_SEG_SOC	NOMBRE	APELLIDOS
413245-B	28-1234566	JUAN	RAMOS
23456-J	28-2345686	PEDRO	PÉREZ
123123-C	19-458766	MARÍA	GIL
1234556-B	45-223344	ANTONIO	SANZ

EMPLETRABAJO

DNI (FK)	DEPARTAMENTO	PUESTO	SALARIO
413245-B	COMPRAS	GERENTE	2.300
23456-J	NÓMINAS	AUXILIAR	1.200
123123-C	ALMACÉN	CONSERJE	1.530
1234556-B	COMPRAS	GESTIÓN	2.200

Otras formas normales

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

Ejercicio resuelto:

Dada la siguiente tabla:

COMPRAS (CodCompra, CodProd, NomProd, fecha, Cantidad, Precio, FechaRec, CodProv, NomPro, Tfno)

Normalizarla hasta FNBC.

Comprobamos 1FN, la tabla ya está en 1FN, sus atributos son atómicos y todos los atributos no clave dependen funcionalmente de la clave.

Comprobamos 2FN, ¿todo atributo depende de todo el conjunto de atributos que forman la clave primaria o sólo de parte?, la tabla no está en 2FN dadas las dependencias:

CodProd -> nomProd y CodProd es parte de la clave primaria, un subconjunto.

Al no estar en 2FN, descomponemos la tabla COMPRAS EN

COMPRAS1 (CodCompra, CodProd, Fecha, Cantidad, Precio, fechaRec, CodProv, NomProv, Tfno)
PRODUCTO (CodProd, NomProd)

Ahora las dos tablas están en 2FN. Todos los atributos no clave dependen de toda la clave primaria.

Comprobamos 3FN, la tabla PRODUCTO está en 3FN ya que no puede tener dependencias transitivas al tener sólo dos atributos. La tabla COMPRAS1 no está en 3FN ya que sí hay dependencias transitivas entre atributos no clave: CodProv -> NomProv y CodProv -> Tfno, por tanto hay que descomponer:

COMPRAS2 (CodCompra, CodProd, Fecha, Cantidad, Precio, fechaRec, CodProv)
PROVEEDOR (CodProv, NomProv, Tfno)

Comprobamos FNBC

PRODUCTO está en FNBC ya que está en 3FN y todo determinante es clave candidata.
COMPRAS2 está en FNBC ya que está en 3FN y todo determinante es clave candidata.
PROVEEDOR está en FNBC ya que está en 3FN y todo determinante es clave candidata.

Por tanto la tabla inicial COMPRAS queda normalizada hasta FNBC del siguiente modo:

COMPRAS2 (CodCompra, CodProd, Fecha, Cantidad, Precio, fechaRec, CodProv)
PROVEEDOR (CodProv, NomProv, Tfno)
PRODUCTO (CodProd, NomProd))