TEMA 3: Modelo Lógico

Módulo

Bases de Datos

para los ciclos

Desarrollo de Aplicaciones Web

Desarrollo de Aplicaciones Multiplataforma

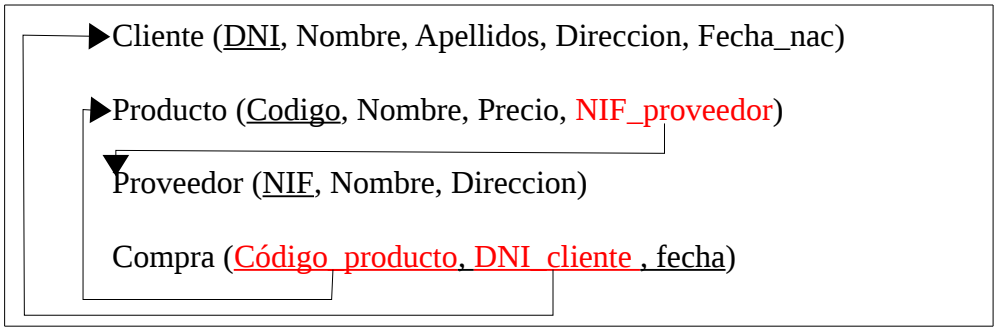
3.1 Introducción

Una vez que tenemos terminado el modelo conceptual de la base de datos, construiremos al modelo lógico. En nuestro caso eso significa pasar del diagrama Entidad/Relación al esquema relacional.

El esquema relacional se llama así porque se compone de un conjunto de relaciones (en el sentido de tablas de datos) que cumplen unas condiciones muy concretas. El resultado de la transformación será el conjunto de tablas, que representamos con dos notaciones diferentes.

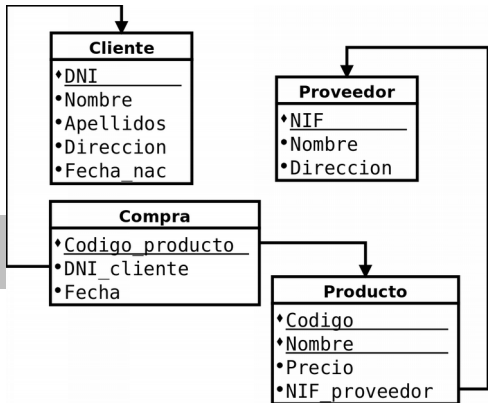
Ejemplo notación 1:

La primera de ellas con el nombre de cada tabla seguido, entre paréntesis, de todos sus campos y dominios, la clave primaria subrayada y las claves ajenas con flechas apuntando a las principales de donde vienen y con las restricciones.



Ejemplo notación 2:

La segunda notación representa cada tabla con un rectángulo. En la parte superior aparece el nombre de la tabla y debajo, uno a uno los atributos, marcando siempre la clave primaria con un rombo y subrayado. De cada una de las claves ajenas parte una flecha hasta la tabla a la que apunta.



Nota: En el programa ***Dia*** podemos usar la hoja de símbolos llamada ***Bases de datos***.

3.1.1 Conceptos básicos

Las ***tablas*** o relaciones son conjuntos de datos ***atómicos*** relativos a todos los elementos de la base de datos que son del mismo tipo. Se estructuran en filas y columnas. Se corresponden con entidades del MER casi siempre.

Las ***filas*** de la tabla o ***tuplas*** comprenden todos los datos que describen a un elemento concreto. Se corresponden con las ocurrencias del MER.

Las ***columnas*** de la tabla o ***campos*** representan una característica que comparten todos los elementos de la tabla. Se corresponden con los atributos del MER.

La ***clave*** de la tabla es el campo o conjunto de campos que identifican de manera única a cada fila. Tienen que ser mínimas y únicas. Se corresponden con los identificadores del MER. También en el modelo relacional hablamos de claves ***primarias*** y ***alternativas***, pero aparece un nuevo concepto: el de clave ***ajena***.

Se llama ***cardinalidad*** de una tabla al número de filas que tiene.

Se llama ***grado*** de una tabla al número de columnas que tiene.

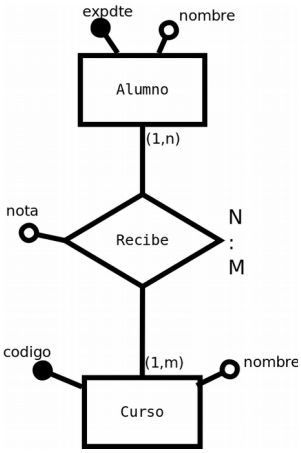
Se llama ***dato nulo*** a la ausencia de valor en una intersección fila-columna.

3.2 Transformación del diagrama E/R al esquema relacional

El paso del diagrama E/R al esquema relacional está automatizado tal y como se indica en los siguientes apartados.

3.2.1 Elementos básicos

- Por cada ***entidad*** del MER creamos una tabla en el relacional de igual nombre.
- Por cada ***atributo*** de una entidad tendremos una columna en su tabla.
- Los ***identificadores*** de la entidad serán las claves de la tabla.
- Por cada relación de cardinalidad ***N:M*** se creará una tabla que incluye las claves de las entidades que relacionan más los atributos de la relación si los hubiera. La clave de la nueva tabla será el conjunto de las dos claves.



ALUMNO (expdte, nombre)

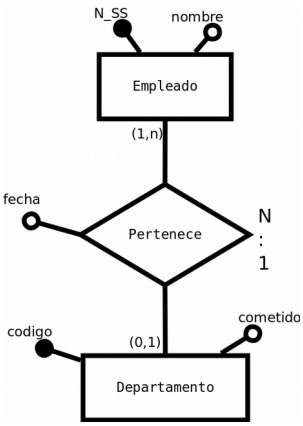
CURSO (código, nombre)

RECIBE (curso, expdte , nota)

3.2.2 Relaciones de uno a muchos

En cada relación de cardinalidad ***1:N*** estudiaremos la participación del lado 1:

- Si tenemos ***(0,1)***: La relación se transforma en una tabla, con las claves de ambas entidades y los atributos de la relación si los hubiera. La clave de esta nueva tabla será la clave de la tabla del lado N.

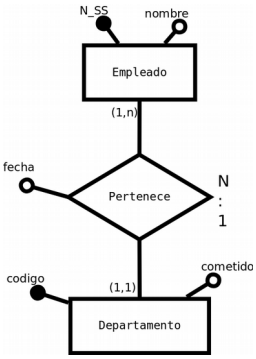


EMPLEADO (N_SS, nombre)

DEPARTAMENTO (código, cometido)

PERTENECE (empleado , dpto, fecha)

- Si tenemos ***(1,1)***: Se hace una propagación de clave, es decir, la clave de la entidad del lado 1 se incluye en la tabla del lado N (y los atributos de la relación si hubiera).



EMPLEADO (N_SS, nombre, dpto, fecha)

DEPARTAMENTO (código, cometido)

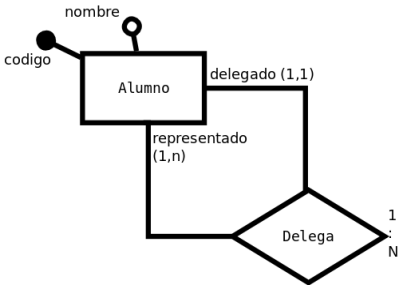
3.2.3 Relaciones de uno a uno

En cada relación de cardinalidad 1:1 estudiaremos las participaciones:

- Si tenemos ***(0,1)*** y ***(1,1)***: Se hace una propagación de clave desde el lado ***(1,1)*** hacia el lado ***(0,1)***.
- Si tenemos ***(0,1)*** y ***(0,1)***: Se crea una nueva tabla, con las claves de ambos lados y los atributos de la relación si los hubiera. La clave de esta nueva tabla puede ser cualquiera de las claves (solo una) de las entidades que relaciona.
- Si tenemos ***(1,1)*** y ***(1,1)***: Se hace una propagación de clave en cualquiera de los sentidos (o se unifican ambas tablas en una sola).

3.2.4 Relaciones reflexivas

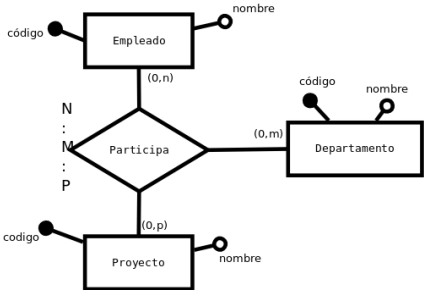
Las ***relaciones reflexivas*** igual que las binarias, según sus cardinalidad y participaciones.



ALUMNO (código, nombre, delegado)

3.2.5 Relaciones n-arias

Para las relaciones de grado mayor a 2, independientemente de la cardinalidad, se creará una nueva tabla con las claves de todas las entidades involucradas en la relación y los atributos propios.



EMPLEADO (código, nombre)

PROYECTO (código, nombre)

DPTO (código, nombre)

PARTICIPA (dpto, proy, emp)

3.2.6 Las claves ajenas y su tratamiento (integridad referencial)

Definición: Una ***clave ajena*** (también llamada clave externa o foránea) es el campo o conjunto de campos que ejerce como clave primaria en otra tabla. Son extremadamente importantes en el modelo lógico porque gracias a las claves ajenas se conectan todas la tablas unas con otras.

En los ejemplos anteriores de este tema las claves ajenas están señaladas con texto en color rojo y con una flecha que apuntan al campo donde ejercen como clave primaria.

En este otro ejemplo vemos la forma en que los datos de la clave ajena repiten los de la clave primaria, permitiendo la conexión entre las tablas:

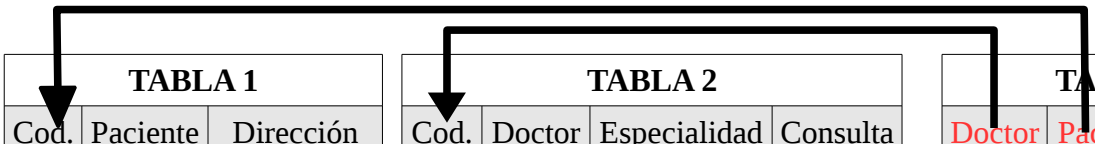


TABLA 1		
Cod.	Paciente	Dirección
401	Pedro	Luna, 10
402	Luís	Arroyo, 15
403	Ana	Estrella, 22

TABLA 2			
Cod.	Doctor	Especialidad	Consulta
100	López	General	310
101	Alonso	Otorrino	402
102	Sanz	Digestivo	120

TABLA 3		
Doctor	Pac.	Diag.
100	401	Sarampión
101	402	Sinusitis
102	401	Diabetes
100	402	Fractura
102	403	Colitis

Con las claves ajenas surge la necesidad de controlar que la conexión entre tablas sigue existiendo a pesar de los cambios (esto se conoce como ***integridad referencial***), por eso, para cada clave ajena tendremos que establecer el comportamiento que tendrá cada vez que un valor de la clave principal se modifica o desaparece.

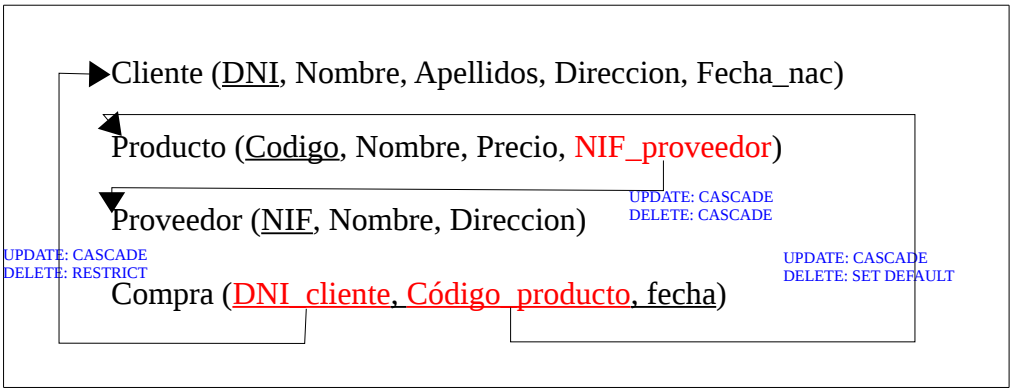
Las posibilidades son 4:

- ***RESTRICT***: (prohibido) Significa que no se permitirán cambios en una clave principal que esté asociada con una clave ajena.
- ***CASCADE***: (en cascada) Significa que todos los cambios que se produzcan en la clave principal se propagarán en cascada hacia la clave ajena.
- ***SET NULL***: (establecer como nulo) Significa que los cambios que se produzcan en la clave principal provocarán que la clave ajena tome el valor ***NULL*** (nulo). Recuerda que el valor nulo significa ausencia de dato.
- ***SET DEFAULT***: (establecer un valor por defecto) Significa que los cambios que se produzcan en la clave principal provocarán que la clave ajena tome un valor por defecto preestablecido previamente.

Por lo tanto, en el grafo relacional, cada vez que localicemos una clave ajena tendremos que:

- Indicar mediante una flecha la clave principal de la que procede.
- Indicar cómo reacciona ante modificaciones (***UPDATE***) de la clave principal.
- Indicar cómo reacciona ante borrados (***DELETE***) de la clave principal.

Ejemplo:



En este ejemplo, vemos que ***todas las modificaciones*** se propagan ***en cascada***, es decir, si se modifica el NIF de un proveedor, se trasladará esa modificación al campo NIF_proveedor de la tabla Producto y lo mismo haremos con las otras dos claves ajenas.

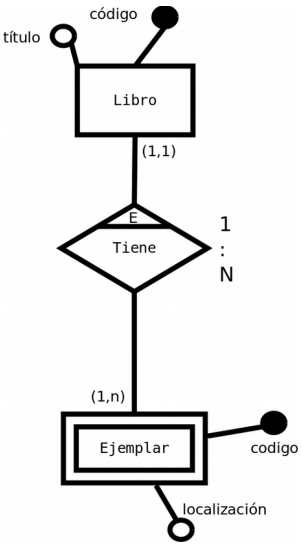
El ***borrado del NIF*** del proveedor también se transmite ***en cascada***, por tanto, si se borra de la tabla Proveedor una fila de un proveedor concreto, habrá que borrar todas las filas de la tabla producto que tengan ese NIF. Esta elección se corresponde con la siguiente semántica "***No se guardarán en la base de datos los productos cuyos proveedores ya no trabajan con la empresa***".

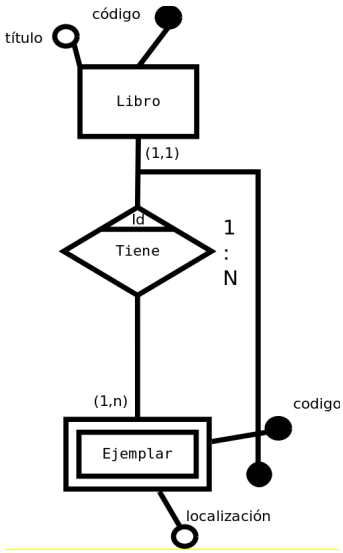
El campo ***DNI_cliente*** es una clave ajena con la restricción ***DELETE: RESTRICT***. Eso significa que no se permitirá borrar la fila de un cliente (en la tabla Cliente) que haya realizado una compra. Esto se corresponde con una empresa que desea guardar registro de todas las compras y eso obliga a guardar el registro del cliente implicado, porque de lo contrario el campo DNI_cliente de la tabla Compra haría referencia a un DNI que no existe en la base de datos.

El campo Código_producto tiene un ***SET DEFAULT*** para los borrados. Eso significa que cuando un producto se borra en la tabla de Productos, todos los registros de compra de ese producto tendrán un valor por defecto (por ejemplo "descatalogado").

Recuerda: que las restricciones de claves ajenas para el borrado y modificación de sus claves principales las determina el cliente en función de cómo funciona su empresa.

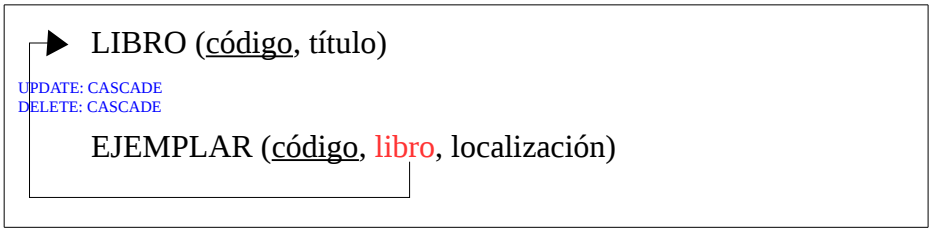
3.2.7 Entidades débiles



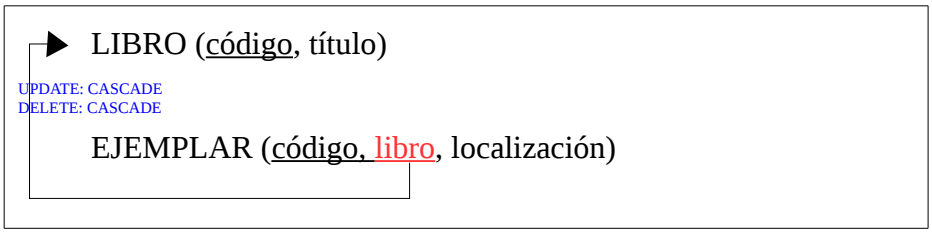


Recordemos que hay dos tipos de entidades débiles:

- Las que sólo tienen ***dependencia de existencia***: (imagen de la izquierda) En este caso siempre se tratará de una relación ***1:N*** con participación ***(1,1)*** en la entidad fuerte, así que la resolvemos propagando la clave de la entidad fuerte a la débil:



- Las que además tienen ***dependencia de identificación***: (imagen de la derecha) Se resuelven de igual manera, pero hay un matiz importante: ***la clave principal de la entidad débil estará formada por su propia clave junto a la clave de la fuerte***:



En cualquiera de los dos casos la clave ajena tendrá como restricciones: ***NOT NULL***, ***DELETE CASCADE***, ***UPDATE CASCADE***.

3.2.8 Atributos derivados

Para los atributos derivados tenemos dos opciones:

- ***Opción A***: Se incluyen en la tabla como el resto de atributos y se construye un ***disparador*** que calcule su valor cada vez que se inserten, borren o modifiquen valores de los campos implicados en el cálculo. Es lo que llamamos un ***dato real***.
- ***Opción B***: No se almacenan y se crean procedimientos para calcularlos cada vez que sean solicitados (***dato virtual***). Esta opción crea problemas de semántica y eficiencia.

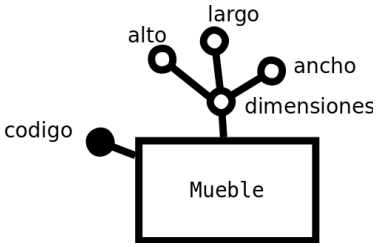
3.2.9 Atributos multivaluados

Para cada atributo multivaluado crearemos una tabla con la clave primaria de la entidad donde se encuentra y el atributo multivaluado. En esa nueva tabla, la clave primaria de la tabla original actuará como clave ajena y la clave principal será el conjunto de ambos atributos. El atributo multivaluado desaparecerá de la tabla original.

Nota: Si el nº de valores posibles se concreta (por ej, 2 teléfonos máximo por Fabricante) podemos incluirlos en la tabla original: ***FABRICANTE (nombre, dirección, tfno1, tfno2)***

3.2.10 Atributos compuestos

El modelo relacional no contempla los atributos compuestos, así que habrá que tratar a cada uno de sus componentes como atributos simples.



MUEBLE (código, alto, ancho, largo)

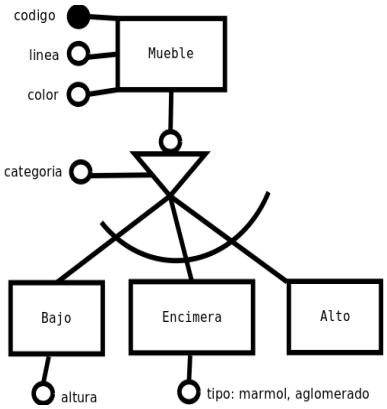
3.2.11 Especialización/Generalización

Las jerarquías no tienen una transformación única. Hay hasta tres opciones distintas para obtener el modelo relacional, que serán más o menos adecuadas en función de que sea total o parcial, exclusiva o solapada. También puede influir el número de atributos comunes (en el supertipo) o de atributos específicos (en el subtipo).

La opción que deberíamos contemplar en primer lugar, porque es la que mejor refleja la semántica y no genera valores nulos, es la siguiente:

Creamos una tabla para el supertipo que incluye los campos comunes y una tabla para cada uno de los subtipos (aunque no tenga atributos propios) que incluye la clave del supertipo y los campos específicos de ese subtipo. Desaparece el atributo discriminante (porque queda implícito en los nombres de las tablas de los subtipos).

Ejemplo:



MUEBLE (código, línea, color)

UPDATE: CASCADE
DELETE: CASCADE

BAJO (código, altura)

ENCIMERA (código, tipo)

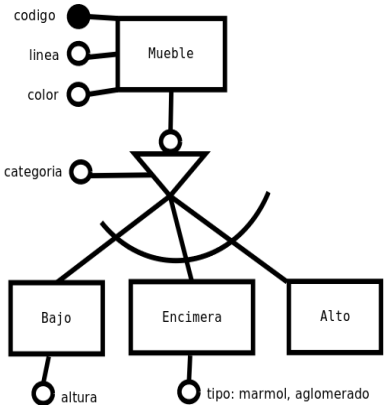
ALTO (código)

Restricciones:

- Para cada tupla del supertipo deberá existir una tupla en el subtipo correspondiente (según discriminante) con la misma clave.
- La clave primaria de los subtipos también será clave ajena con respecto a la tabla del supertipo y con ***DELETE CASCADE***, ***UPDATE CASCADE***.
- Si la jerarquía es EXCLUSIVA no se repite la clave en los subtipos, si fuese SOLAPADA puede repetirse la clave en distintos subtipos.

OPCIÓN ALTERNATIVA 1

Creamos una tabla para cada subtipo que incluye también los atributos comunes. No se crea tabla para el supertipo y desaparece el atributo discriminante.



BAJO (<u>código</u>, línea, color, altura)

ENCIMERA (<u>código</u>, línea, color, tipo)

ALTO (<u>código</u>, línea, color)

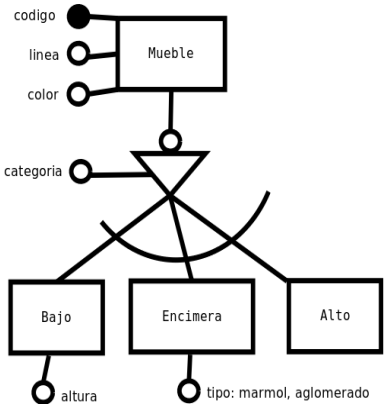
Esta opción será adecuada cuando los subtipos tengan muchos atributos y hará que el acceso sea más eficiente, pero sólo es posible si es jerarquía TOTAL.

Restricciones:

- Si la jerarquía es SOLAPADA hay que controlar la redundancia de los atributos comunes o, mejor, ***no se debería usar esta opción*** para evitar esa redundancia.

OPCIÓN ALTERNATIVA 2

Creamos una sola tabla con todos los atributos: los del supertipo, el discriminante y los de todos los subtipos.



MUEBLE (<u>código</u>, línea, color, categoría, altura, tipo)

Esta opción será adecuada cuando los subtipos se diferencien en pocos atributos y las relaciones sean las mismas para todos o casi todos los subtipos, pero genera muchos valores nulos y hace necesario controlarlos.

Restricciones:

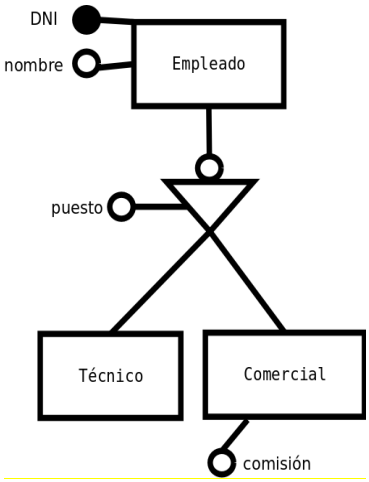
- Para cada valor del discriminante habrá que chequear que los campos de ese subtipo no sean nulos y los campos del resto de los subtipos si sean nulos.

En el ejemplo tenemos que chequear que en cada tupla en la que el valor del campo ***categoría*** sea ***bajo*** haya un valor no nulo para ***altura*** y un valor nulo para ***tipo***, que en cada tupla en la que el valor del campo ***categoría*** sea ***encimera*** justo al contrario y que en cada tupla en la que el valor de ***categoría*** sea ***alto***, los valores de ***altura*** y ***tipo*** sean nulos.

- Si la jerarquía es TOTAL el discriminante será NOT NULL y si la jerarquía es PARCIAL el discriminante admitirá valores nulos.

El ejemplo es una jerarquía TOTAL (véase el circulito sobre el triángulo) y eso se traduce en que el campo ***categoría*** no puede admitir valores nulos.

- Si la jerarquía es SOLAPADA, habrá una nueva tabla con el discriminante y la clave del supertipo (como clave ajena); la clave primaria serán los dos juntos (consiste en tratar el discriminante como un atributo multivaluado).



EMPLEADO (DNI, nombre, comisión)

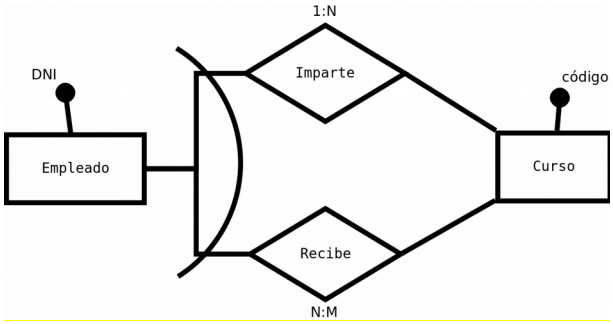
OCUPA (DNI, puesto)

Nota: También podemos usar códigos para los valores del discriminante que contemplen los posibles solapamientos (por ejemplo: ***técnico***, ***comercial*** y ***tecnico-comercial***). En ese caso no hará falta la segunda tabla.

3.2.12 Restricciones de Exclusividad, Inclusión, Inclusividad e Inclusión

Las restricciones de exclusividad, exclusión, inclusividad e inclusión se trasladan al modelo relacional en forma de reglas de verificación o chequeo que controlen la inserción de datos respetando esas restricciones.

EXCLUSIVIDAD



EMPLEADO (DNI, ...)

CURSO (código, …, empleado)

RECIBE (curso, empleado)

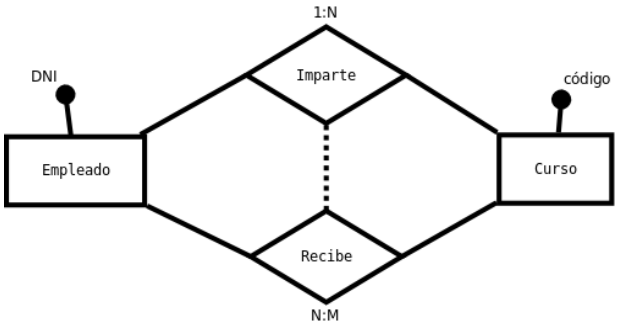
"un empleado que imparte no recibe y viceversa"

Deberíamos incluir unas reglas como las siguientes:

Para todo registro de ***CURSO*** con un valor de ***empleado***, no se permite un registro de ***RECIBE*** con el mismo valor de ***empleado***.

Para todo registro de ***RECIBE*** con un valor de ***empleado***, no se permite un registro de ***CURSO*** con el mismo valor de ***empleado***.

EXCLUSIÓN



EMPLEADO (DNI, ...)

CURSO (código, …, empleado)

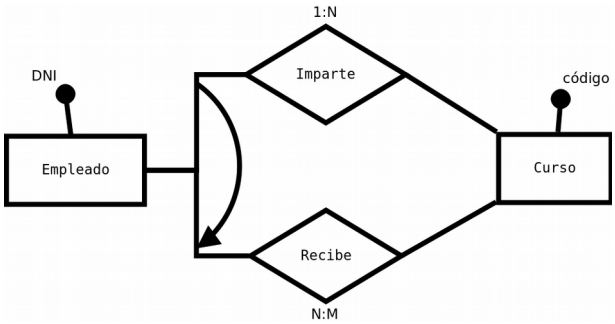
RECIBE (curso, empleado)

"un empleado no puede recibir e impartir el mismo curso"

Deberíamos incluir unas reglas como las siguientes:

Para toda pareja de valores ***código***, ***empleado*** en la tabla ***CURSO***, no puede existir la misma pareja en los valores ***curso***, ***empleado*** de la tabla ***RECIBE***.

INCLUSIVIDAD



EMPLEADO (DNI, ...)

CURSO (código, …, empleado)

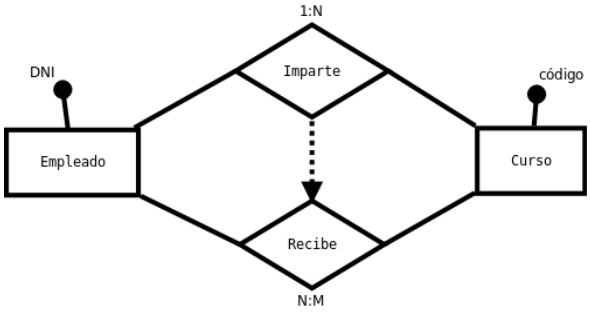
RECIBE (curso, empleado)

"sólo los empleados que reciben pueden impartir"

Deberíamos incluir unas reglas como las siguientes:

Para todo registro de ***CURSO*** con un valor de ***empleado***, tiene que existir un registro en ***RECIBE*** con el mismo valor de ***empleado***.

INCLUSIÓN



EMPLEADO (DNI, ...)

CURSO (código, …, empleado)

RECIBE (curso, empleado)

"sólo los empleados que reciben un determinado curso pueden impartirlo"

Deberíamos incluir unas reglas como las siguientes:

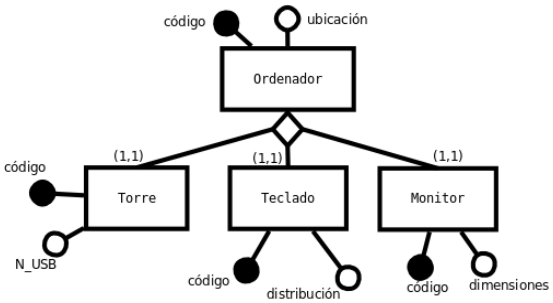
Para todo registro de ***CURSO*** con una pareja de valores ***código***, ***empleado*** tiene que existir un registro en ***RECIBE*** con los mismos valores de la pareja ***curso***, ***empleado***.

3.2.13 Agregación

Recordemos que existen tres tipos de agregación:

- ***Compuesto/componente***: En este caso deberíamos incluir las claves de todos los componentes en el compuesto que sería la solución más cercana al concepto que representa.

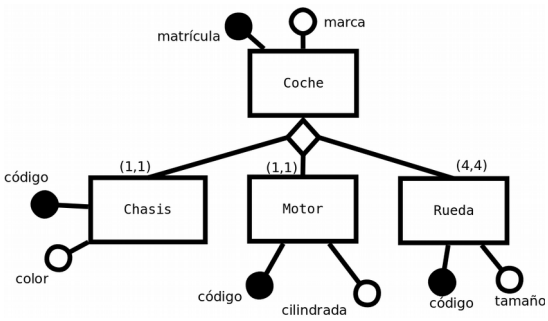
  En el siguiente ejemplo un ordenador está formado por una torre, un teclado y un monitor. Puesto que la participación en todos los componentes es ***(1,1)*** podemos aplicar la solución sugerida sin modificar la filosofía expuesta en todo el tema.



ORDENADORES (<u>código</u>, ubicación, cód_monitor, cód_teclado, cód_torre)

- TORRES (<u>código</u>, N_USB)
- TECLADOS (<u>código</u>, distribución)
- MONITORES (<u>código</u>, dimensiones)

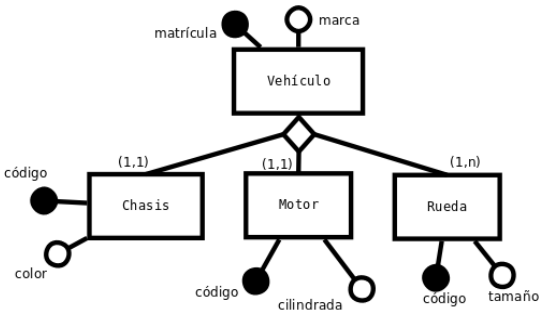
En el siguiente ejemplo un coche está formado por un número concreto de cada uno de los componentes (1 motor, 1 chasis y 4 ruedas):



COCHE (<u>matrícula</u>, marca, cód_chasis, cód_motor, rueda1, rueda2, rueda3, rueda4)

- CHASIS (<u>código</u>, color)
- MOTORES (<u>código</u>, cilindrada)
- RUEDAS (<u>código</u>, tamaño)

Podemos encontrarnos un compuesto que tenga un número indeterminado de alguno de los componentes. En el siguiente ejemplo un vehículo tiene un 1 motor y un chasis, pero tiene un número indeterminado de ruedas porque estamos considerando desde monociclos (1 rueda) hasta grandes camiones:

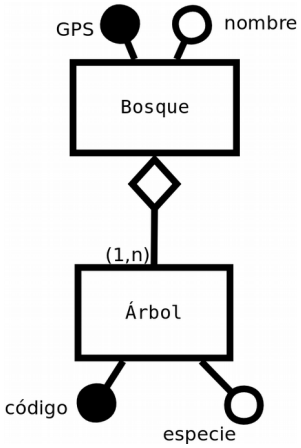


VEHÍCULOS (<u>matrícula</u>, marca, cód_chasis, cód_motor)

- CHASIS (<u>código</u>, color)
- MOTORES (<u>código</u>, cilindrada)
- RUEDAS (<u>código</u>, tamaño, vehículo)

En este caso hemos recurrido (sólo en el caso de las ruedas) a propagar la clave del vehículo hasta la entidad RUEDAS. Recuerda que la participación del compuesto siempre es ***(1,1)*** aunque no se ponga explícitamente.

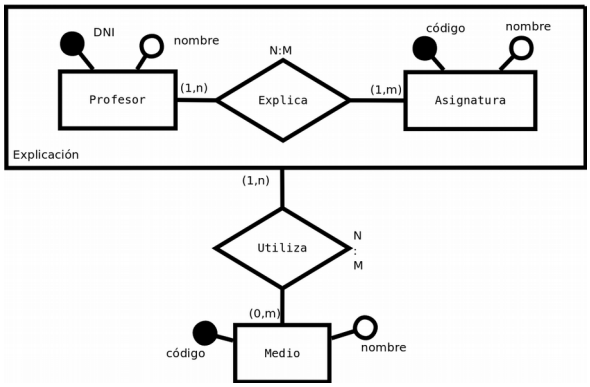
- ***Miembro/colección***: En este tipo de agregaciones siempre se da una relación ***1:N*** con una participación total ***(1,1)*** implícita del compuesto:



BOSQUES (GPS, nombre)

ÁRBOLES (código, especie, bosque)

- ***Relaciones complejas***: La tercera forma de agregación responde a un concepto muy distinto a las otras dos y aparece como solución a la imposibilidad de relacionar una entidad con una relación en el ***MER***. En este caso, se resuelve el interior de la agregación de la forma habitual y posteriormente es tratada como una entidad cuya clave es la de la relación en la que se centra la agregación:



PROFESORES (DNI, nombre)

ASIGNATURAS (código, nombre)

EXPLICA (DNI_profe, cód_asignatura)

MEDIO (código, nombre)

UTILIZA (DNI_profe, cód_asignatura, cód_medio)

3.2.14 Completando el esquema relacional

Junto con el grafo relacional, el modelo final debería de incluir esta otra información:

- Las claves primarias se suelen escribir con la abreviatura ***PK (PRIMARY KEY)*** tras el nombre del campo. Cuando una clave primaria está formada por más de un campo se suele poner bajo la definición de la relación:

```
Tabla1 (campo1, campo2, campo3, campo4)
PK {campo1, campo2}

Tabla2 (campo5 (PK), campo6, campo7)
```

En el ejemplo anterior la clave primaria de ***Tabla1*** está formada por los campos ***campo1*** y ***campo2*** conjuntamente. La clave primaria de la ***Tabla2*** la ostenta ***campo5***.

Nota: La nomenclatura que hemos usado en este tema (subrayado la clave principal) también es válida y más cómoda.

- Las claves ajenas se suelen escribir con la abreviatura ***FK (FOREIGN KEY)*** tras el nombre del campo, aunque en este tema las hemos señalado con color rojo. Estas claves siempre deberían recoger el modo en que se gestionan los borrados y modificaciones (de forma independiente) y que pueden ser: ***NO ACTION CASCADE***, ***SET NULL***, ***SET DEFAULT***.
- Los campos que no admitan valores nulos deberían recoger la restricción ***NOT NULL***, aunque en el caso de las claves primarias no es necesario porque se sobreentiende.
- Las claves alternativas deberían recoger la restricción ***UNIQUE*** (significa que no pueden repetirse los mismos valores para ese campo, en registros distintos).
- Los dominios de los campos se detallan tras el nombre del campo separado con el símbolo dos puntos ***(:)***, aunque solo lo haremos cuando no se trate de un dominio obvio.

```
Tabla1 (campo1:dominio1, campo2:dominio2 campo3:dominio3)
```

- Habría que especificar otras restricciones (por ejemplo, en la jerarquía que un determinado campo no tenga valores nulos en función del valor del discriminante), o las que corresponden a la exclusvidad, etc.

  Para estas restricciones puede usarse lenguaje natural o matemático cuidando que se entienda claramente. Otra opción muy habitual es usar el mismo lenguaje con el que se va a implementar posteriormente (por ejemplo, SQL).

3.3 Resumen de las condiciones del modelo relacional

Condiciones propias del modelo:

- Todas las tuplas de una tabla son distintas.
- El orden de las filas y columnas es irrelevante.
- Todo campo en cada tupla tiene un solo valor (dato atómico).
- Los campos clave no pueden ser nulos.

Condiciones semánticas:

- Toda tabla tiene clave primaria (***PRIMARY KEY***).
- Las claves no se repiten (***UNIQUE***).
- Se pueden declarar campos no nulos (***NOT NULL***), pero para la clave no es necesario.
- ***Integridad referencial***: los valores de una clave ajena se corresponden con los de su primaria:
  - En ***cascada***: borrar o modificar la clave primaria implica borrar las ocurrencias de la ajena.
  - ***Restrict***: no se podrá borrar una tupla o modificar su clave primaria si existe en otra tabla como ajena.

- ***Set null***: borrar una tupla o modificar su clave primaria implica asignar el valor ***NULL*** en las tablas donde existe como ajenas.
- ***Set default***: borrar una tupla o modificar su clave primaria implica asignar un valor por defecto en las tablas donde existe como ajena.

- Verificación (***check***): esta restricción permite especificar condiciones que deben cumplir los campos (se comprobarán en cada inserción o modificación) por ejemplo edad entre 0 y 125.

3.4 Normalización

Cuando se diseña una base de datos mediante el modelo relacional, al igual que ocurre con otros modelos de datos, tenemos distintas opciones, es decir, podemos obtener diferentes esquemas relacionales, y no todos ellos son equivalentes, ya que unos van a representar la realidad mejor que otros.

Con la teoría de la normalización se consigue formalizar el diseño lógico de bases de datos relacionales para disponer de instrumentos algorítmicos de ayuda al diseño y, de esta forma, poder desarrollar programas que automaticen el diseño en el modelo relacional.

Con el conocimiento de la normalización aprenderemos qué propiedades debe tener un esquema relacional para representar adecuadamente la realidad y cuáles son los problemas que se pueden derivar de un diseño inadecuado.

Nota: el esquema relacional debe ser siempre analizado para comprobar que no presenta problemas ambigüedad, pérdida de información, redundancia, valores nulos inaplicables y la aparición de estados que no son válidos en el mundo real.

3.4.1 Primera Forma Normal

DEFINICIÓN

Decimos que una tabla se encuentra en primera forma normal (***1ª FN***) si no existen valores multivaluados, es decir, existe una clave principal formada por uno o varios atributos de modo que representa de forma única a cada una de las tuplas posibles.

Otra forma de decir esto es que para cada valor de la clave ningún atributo puede tener más de un valor posible.

Nota: En nuestro caso, la 1ª FN la conseguimos con el procedimiento que hemos establecido para transformar los multivaluados del ER al relacional.

PROCEDIMIENTO

Cuando una tabla no se encuentra en 1ª FN podemos solucionarlo de dos modos:

- Ampliando la clave
- Descomponiendo la tabla

En cada supuesto será más adecuada una u otra opción e incluso es posible que necesitemos aplicar la ampliación de clave para unos grupos repetitivos y la descomposición de tabla para otros.

Si nuestra tabla no está en 1ª FN significa que hay atributos o grupos de atributos que constituyen grupos repetitivos (o sea, que para un mismo valor de la clave, pueden ofrecer valores distintos)

En este caso tendremos que detectar los atributos repetitivos, teniendo en cuenta si se repite un atributo individualmente o si se repiten grupos de atributos conjuntamente.

Por ejemplo, si tenemos la tabla:

SOCIO (NIF, Nombre, Poblacion, Provincia, Telefono, NombreHijo, EdadHijo)

podemos ver que no está en ***1ª FN*** porque para un mismo valor de la clave elegida (***NIF***) se pueden dar varios valores de los atributos ***Telefono***, ***NombreHijo*** y ***EdadHijo***.

Analizando estos atributos nos damos cuenta de que ***NombreHijo*** y ***EdadHijo*** son un grupo que se repite de forma conjunta (es decir, un socio puede tener varios hijos, cada uno con una edad). Sin embargo el atributo ***Telefono*** es un atributo que se repite para cada ***NIF***, pero de forma independiente de ***NombreHijo*** y de ***EdadHijo***. Tenemos por tanto dos grupos repetitivos:

Primer grupo repetitivo: Telefono

Segundo grupo repetitivo: NombreHijo, EdadHijo

AMPLIAR LA CLAVE

Se trata de añadir a la clave inicial que ya teníamos, un atributo o grupo de atributos del grupo repetitivo, de forma el resto de los atributos del grupo dejen de ser multivaluados para esta nueva clave.

Así, en el segundo grupo del ejemplo anterior

(NombreHijo, EdadHijo)

podemos extender la clave inicial (***NIF***) al atributo ***NombreHijo***, de modo que nos quedaría la siguiente tabla:

SOCIO (NIF, Nombre, Poblacion, Provincia Telefono, NombreHijo, EdadHijo)

Ahora, el resto de atributos del grupo (***EdadHijo***) deja de ser multivaluado puesto que para la nueva clave (***NIF, NombreHijo***) no se puede dar más que un valor de ***EdadHijo***.

Sin embargo esta tabla no está aún en ***1ª FN*** porque sigue habiendo un atributo multivaluado (***Telefono***), del que tendremos que ocuparnos a continuación.

DESCOMPONER LA TABLA

Consiste en separar la tabla inicial en varias tablas: una de ellas estará formada por la clave elegida hasta este momento y todos aquellos atributos que no son multivaluados. Las otras tablas (una por cada grupo que aún sea multivaluado) estarán formadas por el atributo o grupo de atributos repetitivos y la ***parte de la clave*** de la que depende.

En el ejemplo anterior tenemos la tabla:

SOCIO (NIF, Nombre, Poblacion, Provincia, Telefono, NombreHijo, EdadHijo)

que quedará descompuesta en otras dos:

SOCIO1 (NIF, Nombre, Poblacion, Provincia, NombreHijo, EdadHijo)

con la clave más los atributos que no se repiten, y

SOCIO2 (NIF, Telefono)

que contiene el atributo repetitivo (***Telefono***) y la parte de la clave (***NIF***) de la que depende.

Obsérvese que hemos obviado parte de la clave (***NombreHijo***) porque el ***Telefono*** no tiene ninguna relación con el atributo ***NombreHijo***.

Una vez realizado este proceso nos tendremos que preguntar si la tabla o las tablas resultantes ya están en ***1ª FN*** y, de no ser así, tendremos que volver a aplicarlo para cada una de ellas.

En nuestro ejemplo podemos ver que la primera tabla:

SOCIO1 (NIF, Nombre, Poblacion, Provincia, NombreHijo, EdadHijo)

está en ***1ª FN***, sin embargo la segunda:

SOCIO2 (NIF, Telefono)

sigue teniendo el ***Telefono*** como atributo multivaluado y si aplicamos de nuevo el proceso, vemos que al ampliar la clave queda definitivamente en ***1ª FN***:

SOCIO2 (NIF, Telefono)

Esto no significa que las tablas sean correctas, aún están por conseguir la ***2ª FN*** y la ***3ª FN***.

3.4.2 Segunda Forma Normal

DEFINICIÓN

Decimos que una tabla se encuentra en segunda forma normal (***2ª FN***) si está en ***1ª FN*** y además todos los atributos no principales dependen de la clave de forma completa (es decir, no dependen de un subconjunto de la clave)

Nota: Atributos principales son los que forman parte de alguna clave (principal o alternativa). Atributos NO principales son, por tanto, los que no forman parte de ninguna clave.

PROCEDIMIENTO

Lo primero que tenemos que comprobar es si la tabla se encuentra en ***1ª FN***. De no ser así tendremos que transformarla según el apartado anterior. Partimos pues de la suposición de que ya se encuentra en ***1ª FN***.

A continuación comprobamos si ya se encuentra en ***2ª FN*** y si este es el caso no tenemos que hacer nada. Si de la comprobación deducimos que no lo está, para llegar a la ***2ª FN*** descomponemos la tabla en varias tablas:

- Una tabla con la clave y aquellos atributos que dependen de forma completa de ella.
- Para cada subconjunto de la clave del que dependan atributos: una tabla con el subconjunto de la clave y los atributos que dependen de él.

En nuestro ejemplo, después de haber conseguido la ***1ª FN***, teníamos dos tablas:

SOCIO1 (NIF, Nombre, Poblacion, Provincia, NombreHijo, EdadHijo)

SOCIO2 (NIF, Telefono)

La tabla ***SOCIO2*** se encuentra ya en ***2ª FN***, puesto que no tiene atributos no principales.

La tabla ***SOCIO1*** no está en ***2ª FN*** porque existen atributos no principales (***Nombre, Poblacion, Provincia***) que no dependen en forma completa de la clave (***NIF, EdadHijo***) ya que nos basta el ***NIF*** para implicarlos. Por tanto descomponemos la tabla en otras dos:

SOCIO11 (NIF, NombreHijo, EdadHijo)

que tiene la clave y todos los atributos que dependen de ella en forma completa (***EdadHijo***), y

SOCIO12 (NIF, Nombre, Poblacion, Provincia)

que tiene los atributos que no dependen de forma completa de la clave (***Nombre, Poblacion, Provincia***) y la parte de la clave de la que dependen (***NIF***)

Esto no significa que las tablas sean correctas, aún están por conseguir la ***3ª FN***.

3.4.3 Tercera Forma Normal

DEFINICIÓN

Decimos que una tabla se encuentra en tercera forma normal (***3ª FN***) si está en ***2ª FN*** y no existen dependencias transitivas. Dicho de otro modo si no existen dependencias entre los atributos NO principales.

Nota: Atributos principales son los que forman parte de alguna clave (principal o alternativa). Atributos NO principales son, por tanto, los que no forman parte de ninguna clave.

PROCEDIMIENTO

Lo primero que tenemos que comprobar es si la tabla se encuentra en ***2ª FN***. De no ser así tendremos que transformarla según el apartado anterior. Partimos pues de la suposición de que ya se encuentra en ***2ª FN***.

A continuación comprobamos si ya está en ***3ª FN*** y en ese caso no tenemos que hacer nada. Si de la comprobación deducimos que no lo está, para llegar a la ***3ª FN*** descomponemos la tabla en varias tablas:

- Una con la clave y los atributos que NO dependen de otros atributos NO principales.
- Para cada atributo NO principal del que dependan otros atributos: una tabla con el atributo no principal y los atributos que dependen de él.

En nuestro ejemplo, después de haber conseguido la 2ª ***FN***, teníamos tres tablas:

SOCIO11 (NIF, NombreHijo, EdadHijo)

SOCIO12 (NIF, Nombre, Poblacion, Provincia)

SOCIO2 (NIF, Telefono)

La tabla ***SOCIO11*** ya está en ***3ª FN***, puesto al tener un sólo atributo NO principal (***EdadHijo***) no existe la posibilidad de dependencias entre atributos NO principales.

La tabla ***SOCIO2*** ya está en ***3ª FN***, puesto que al no tener atributos NO principales no existe la posibilidad de dependencias entre atributos NO principales.

La tabla ***SOCIO12*** no está en ***3ª FN***, porque para un mismo valor de ***Población*** siempre se dará el mismo valor de ***Provincia***, por tanto ***Provincia*** depende funcionalmente de ***Poblacion*** y eso significa que tenemos dependencias entre atributos NO principales.

Esta tabla la tenemos que descomponer en otras dos:

SOCIO121 (NIF, Nombre, Población)

que contiene la clave y los atributos que NO dependen de otros atributos NO principales, y

SOCIO122 (Poblacion, Provincia)

que contiene el atributo NO principal ***Poblacion*** y los atributos que depende de él (***Provincia***)