UNIDAD DE TRABAJO 2: BASES DE DATOS RELACIONALES

TEMA 5: EL MODELO RELACIONAL. NORMALIZACIÓN

5.1 - INTRODUCCIÓN

En el diseño lógico de datos vamos a distinguir dos fases: una de alto nivel independiente del modelo de base de datos que utilicemos para la implementación y otra fase dependiente del modelo, es decir, de su estructura de almacenamiento (jerárquico, red o relacional). El modelo que utilizaremos nosotros será el relacional.

En primer lugar se obtiene el modelo lógico de alto nivel, independiente del modelo de base de datos y los objetivos a conseguir son:

a) Decisiones sobre los datos que tenemos en la BD

b) Simplificar el esquema conceptual de nuestro modelo (diagrama E/R).

5.2. – DISEÑO LÓGICO ALTO NIVEL

5.2.1 - DECISIONES SOBRE DATOS DERIVADOS

Mantener datos derivados en el esquema lógico tiene una ventaja y es que no hace falta calcularlos durante la ejecución de la aplicación, pero tiene otras desventajas como son: el mantenimiento de los datos necesita tiempo de ejecución para actualizarlos cada vez que varían los datos de los que depende y se necesita mayor espacio de almacenamiento.

Por tanto para decidir si se almacenan o no los datos derivados debemos evaluar los siguientes puntos:

a) las operaciones de recuperación de datos en las que intervienen datos derivados son más rápidas si tenemos almacenados dichos datos (ya que no tendríamos que calcularlos)
b) Las operaciones que escritura sobre los datos a partir de los cuales se obtienen datos derivados necesitan un tiempo de procesamiento adicional para actualizar los datos derivados.

Por tanto tendremos que ver la frecuencia y el número de operaciones de lectura y escritura que se realizan sobre dichos datos para decidir si conviene almacenarlos o no.

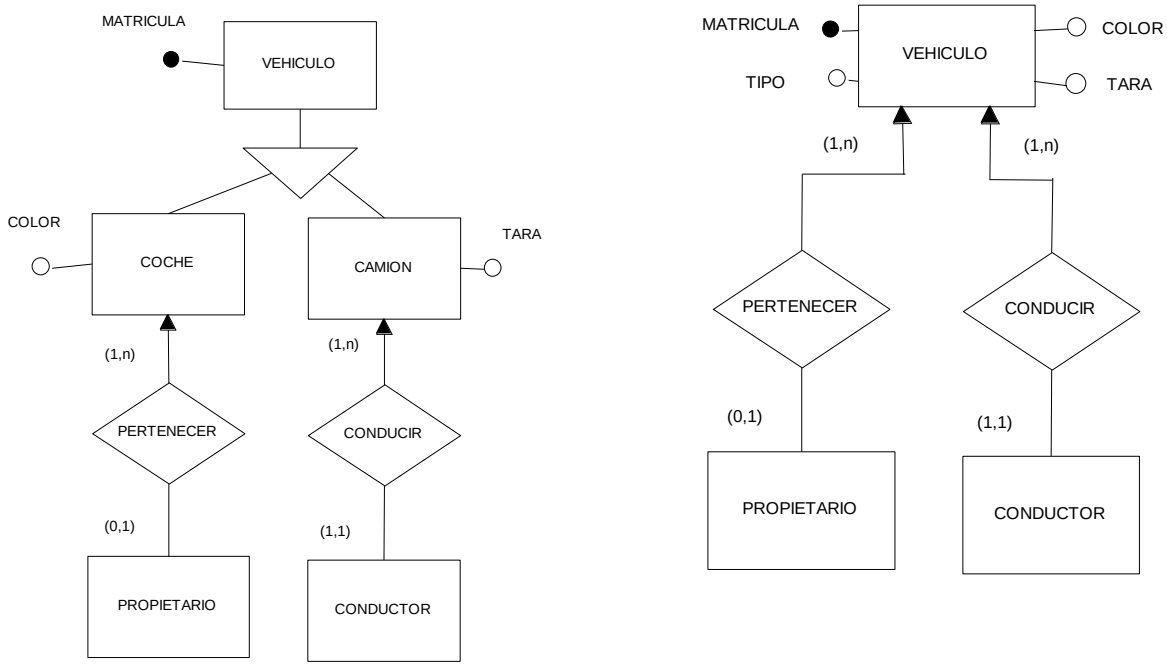
Ejemplo: en el caso anterior el atributo NOTA-MED es un atributo derivado ya que se puede obtener sumando las notas de un determinado alumno en cada asignatura y dividiendo por el número de asignaturas.

Para decidir si debe formar parte de la base de datos o no, debemos estudiar las ventajas que tiene conservarlo, en este caso sería que una vez al mes realizamos el listado de alumnos con su nota media en un proceso por lotes. En cuanto a las desventajas sería que tenemos que actualizarlo cada vez que cambia alguna nota por tanto 99 veces al día. De estas dos situaciones decidimos que es más ventajoso no almacenarlo y se calculará cuando una vez al mes hagamos el listado de alumnos con su nota media.

5.2.2 - ELIMINACIÓN DE LAS JERARQUÍAS DE GENERALIZACIÓN

Ninguno de los modelos lógicos de bases de datos(jerárquico, en red y relacional), permite representar las jerarquías de generalización, por tanto debemos eliminarlas de nuestro esquema conceptual para poder obtener el modelo lógico, sin embargo de alguna manera debemos conservar la propiedad de la herencia de atributos de la entidad supertipo a los subtipos. Existen tres formas de eliminar las jerarquías de generalización:

1.- **Jerarquía de generalización modelada por una entidad supertipo**:



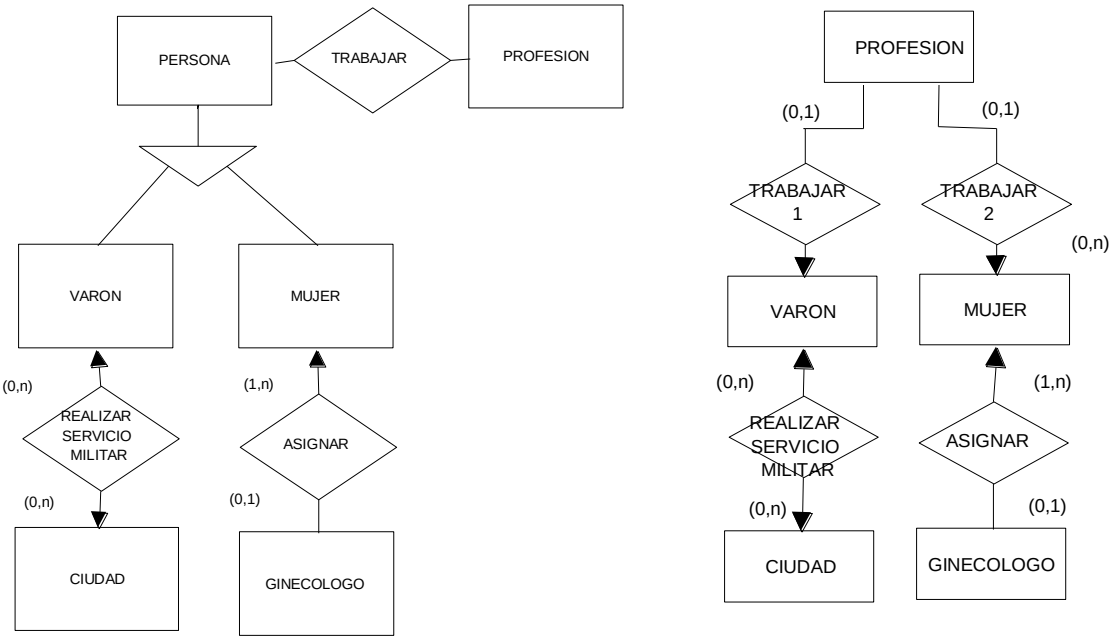
En este caso nos quedamos solamente con la entidad supertipo a la que se añaden los atributos de las entidades subtipos que serán opcionales y un nuevo atributo TIPO. Como ventajas de esta solución podemos decir que:

- Se puede aplicar a cualquier tipo de generalización (total, parcial, superpuesta o exclusiva).
- Es una solución muy fácil ya que no intervienen interrelaciones.

Las desventajas son las siguientes:

- Podemos generar un gran número de valores nulos para los atributos de los subtipos.
- Las operaciones que antes solo se realizaban sobre los subtipos ahora se realizarán sobre el conjunto completo, sobre el supertipo.

2. - Jerarquía de generalización modelada por entidades subtipo:

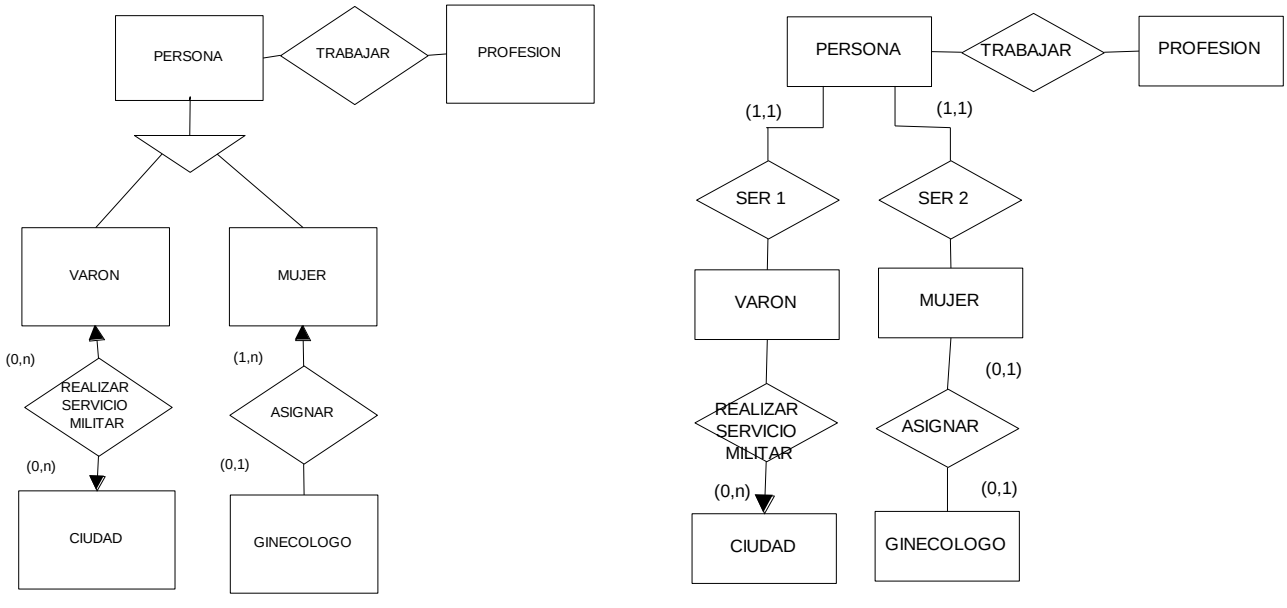


En este caso desaparece la entidad supertipo y nos quedamos solamente con los subtipos que heredan los atributos del supertipo. Como desventajas podemos apuntar:

- Solamente es práctico utilizarlo en jerarquías totales y exclusivas.
- Las operaciones que antes solo se realizaban sobre el supertipo deben recorrer ahora todos los subtipos.

3. - Jerarquía de generalización modelada por interrelaciones:

Se puede utilizar para modelar cualquier tipo de interrelación, pero es esquema resultante es bastante complejo. Consiste en eliminar la jerarquía sustituyéndola por interrelaciones ES_UN:



Como resumen podemos indicar que:

- La alternativa 1 es conveniente cuando en nuestro sistema hay muchas operaciones que utilizan a la vez tanto atributos de la entidad supertipo como atributos de los subtipos.

- La alternativa 2 solo se utiliza para jerarquías de generalización totales y exclusivas.

- La alternativa 3 es conveniente cuando las operaciones del sistema no utilizan a la vez atributos del supertipo y de los subtipos.

5.2.3 – SELECCIÓN DE CLAVES PRIMARIAS

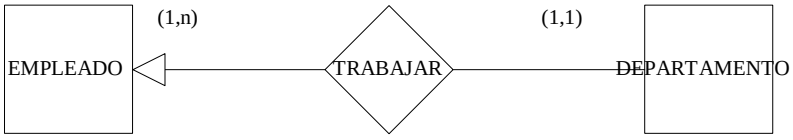
La selección de la clave primaria es importante porque facilita el acceso a las ocurrencias en el diseño físico, por tanto a la hora de elegirla debemos tener en cuenta las siguientes cuestiones:

- Si una entidad tiene varios atributos candidatos, elegiremos el atributo por el que se realicen el mayor número de accesos a la entidad.

- Es mejor elegir los identificadores simples a los compuestos.

- Es mejor elegir identificadores internos a externos, es decir procedentes de otra entidad.

Por ejemplo, si tenemos el caso siguiente:



Los atributos de la entidad EMPLEADO son:

NOMBRE: nombre del empleado.
FECHA: fecha de nacimiento, junto a NOMBRE son únicos en la empresa.
DNI: documento nacional de identidad.
NUM_SEG_SOC: número de la seguridad social.
NUM_EMP_EMPRESA: a cada empleado de la empresa se le asigna un número diferente.
NUM_EMP_DEPARTEMENTO: a cada empleado se le asigna un número diferente dentro del departamento al que pertenece.

Los atributos de la entidad DEPARTAMENTO son:
NUM_DEPARTAMENTO: número de departamento.

Para identificar la entidad EMPLEADO tenemos las siguientes posibilidades:
NOMBRE + FECHA
DNI
NUM_SEG_SOC
NUM_EMP_EMPRESA
NUM_EMP_DEPARTAMENTO + NUM_DEPARTAMENTO

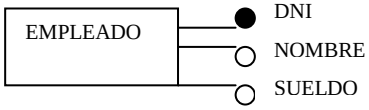
Aplicando los criterios vistos vemos que las mejores opciones están entre DNI, NUM_SEG_SOC y NUM_EMP_EMPRESA, entre ellos elegiremos el más corto y resultan más fáciles si son de tipo carácter o aquél por el que se acceda el mayor número de veces

5.3.- TRANSFORMACIÓN DE UN ESQUEMA E/R AL MODELO RELACIONAL

Vamos a partir de un esquema lógico E/R y obtendremos un esquema relacional, es decir el esquema de cada una de las relaciones (tablas) de nuestro sistema, cada una de las relaciones tendrá su clave primaria. Los pasos a seguir para la transformación son los siguientes:

1.- Transformación de entidades:

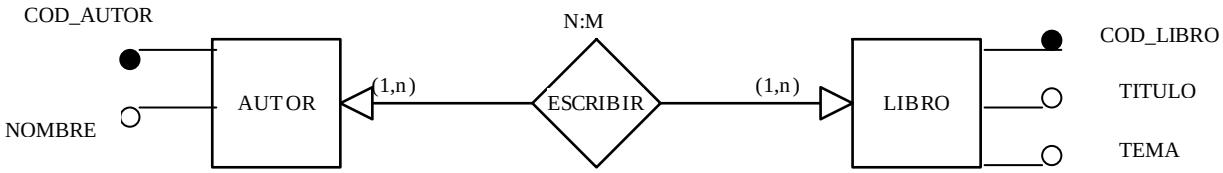
Cada entidad se convierte en una relación en la cual los atributos se convierten en los atributos de la relación y el atributo identificador principal se convierte en la clave primaria de la relación. Por ejemplo:



Se convierte en la relación: EMPLEADO (DNI, NOMBRE, SUELDO)

2.- Transformación de interrelaciones N:M

Las interrelaciones N:M se transforman en una relación cuya clave primaria es la concatenación de las claves primarias de cada una de las relaciones a que dan lugar las entidades que intervienen en la interrelación y tendrán como atributos los atributos de la interrelación.



ESCRIBIR(COD_AUTOR, COD_LIBRO)

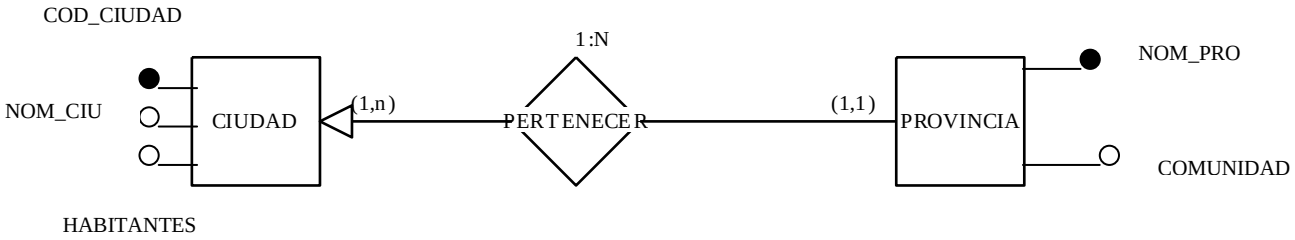
AUTOR(COD_AUTOR, NOMBRE)

LIBRO(COD_LIBRO, TITULO, TEMA)

Cada uno de los atributos que forman parte de la clave primaria de la relación ESCRIBIR, son clave ajena respecto de cada una de las relaciones donde ese atributo es clave primaria.

3.- Transformación de interrelaciones uno a muchos 1:N

Las interrelaciones 1:N se transforman mediante propagación de clave o bien creando una nueva relación.



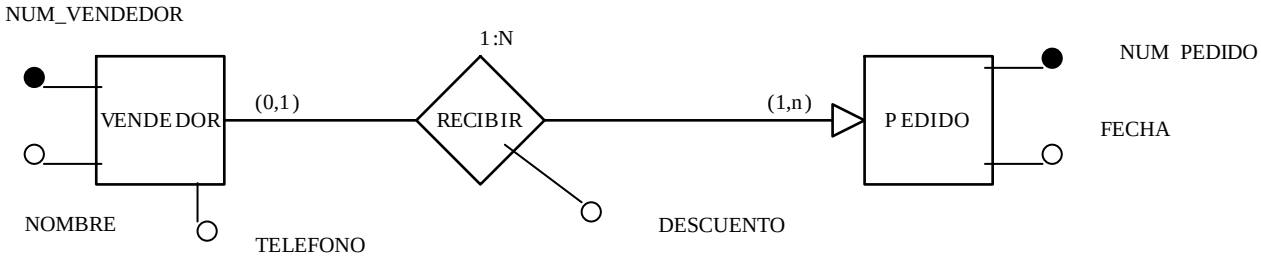
Si la cardinalidad mínima de la entidad que se propaga la clave es uno es mejor propagar la clave:

CIUDAD (COD_CIUDAD, NOM_CIU, HABITANTES, NOM_PRO)

PROVINCIA (NOM_PRO, COMUNIDAD)

Transformar en una relación es conveniente cuando la relación tiene atributos o cuando la cardinalidad mínima de la entidad que deberíamos propagar la clave es cero:

Suponemos el caso de un mayorista que admite pedidos a través de sus vendedores y en este caso aplica un descuento, pero también admite pedidos realizados directamente a la empresa sin haber contactado con un vendedor en cuyo caso no se realizan descuentos:



En este caso si utilizamos la propagación de clave tendríamos:

VENDEDOR (NUM_VENDEDOR, NOMBRE, TELEFONO)

PEDIDO(NUM_PEDIDO, FECHA, NUM_VENDEDOR)

Perderíamos el atributo DESCUENTO que es de la interrelación RECIBIR y podemos generar muchos valores nulos para NUM_VENDEDOR en la relación PEDIDO puesto que hay pedidos que no se han realizado a través de vendedores.

La mejor solución es transformar la interrelación en una relación de la siguiente forma:

VENDEDOR (NUM_VENDEDOR, NOMBRE, TELEFONO)

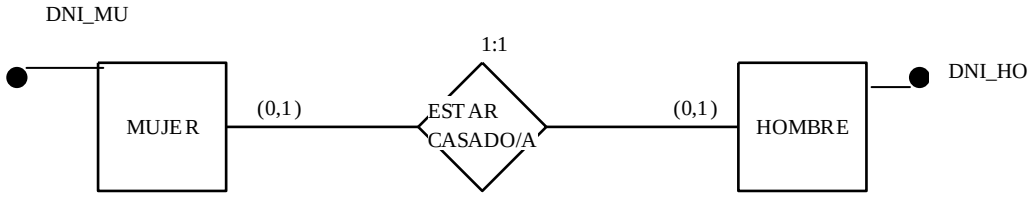
PEDIDO(NUM_PEDIDO, FECHA)

RECIBIR (NUM_VENDEDOR, NUM_PEDIDO, DESCUENTO)

4.- Transformación de interrelaciones uno a uno (1:1)

Cada entidad se convierte en su relación correspondiente, y para la interrelación tenemos los siguientes casos:

4.1.- Cuando la participación de las dos entidades en la interrelación es opcional y por tanto sus cardinalidades mínimas son cero, la interrelación se transforma en una relación:

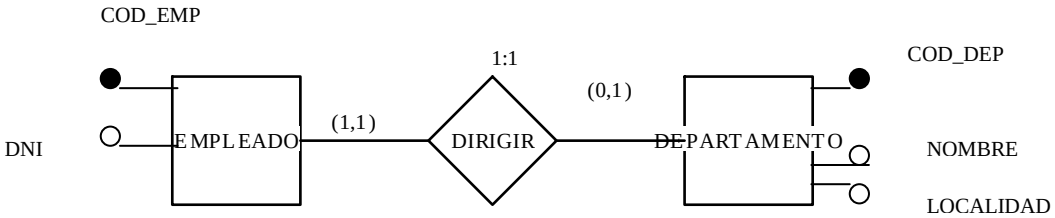


MUJER(DNI_MU,.....)

HOMBRE(DNI_HO,....)

ESTAR CASADO/A (DNI_MU,DNI_HO)

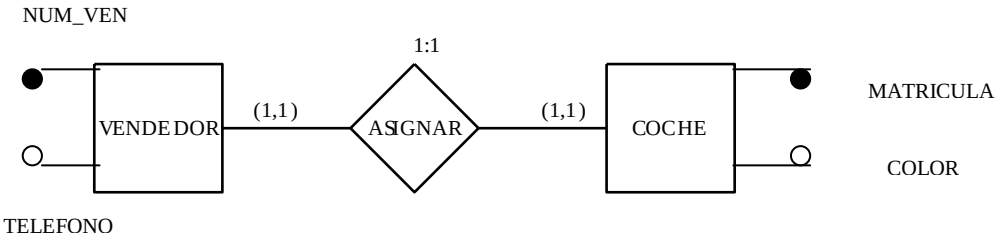
4.2.- Si una entidad tiene participación opcional y la otra participación obligatoria, se propaga la clave de la entidad que tiene participación obligatoria a la opcional:



EMPLEADO(COD_EMP,DNI,...)

DEPARTAMENTO(COD_DEP, NOMBRE, LOCALIDAD,...,COD_EMP)

4.3.- Cuando la participación de las dos entidades en la interrelación es obligatoria y por tanto sus cardinalidades mínimas son 1, las dos entidades se transforman en una sola relación que tiene tanto las claves primarias como los atributos de las dos entidades. Una de las claves primarias se elige como clave primaria de la relación, generalmente aquella por la cual se realicen la mayoría de los accesos a la relación.

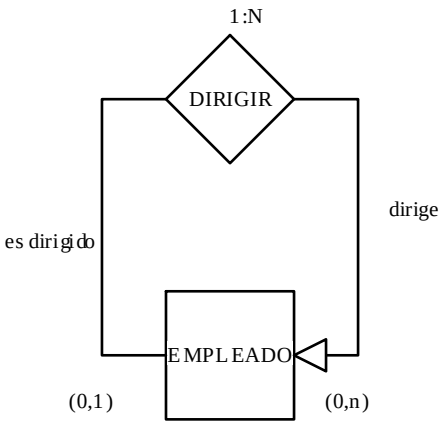


VENDEDOR-COCHE (NUM_VEN, TELEFONO, MATRICULA, COLOR)

O bien:

COCHE-VENDEDOR(MATRICULA,COLOR, NUM_VEN,TELEFONO)

5.- Transformación de interrelaciones reflexivas:



En este caso tendríamos lo siguiente:

EMPLEADO(NOMBRE, FECHA_NACIMIENTO, CATEGORIA)

DIRIGIR(NOMBRE_SUBORDINADO, NOMBRE_DIRECTOR)

Otra forma sería:

EMPLEADO (NOMBRE, FECHA_NACIMIENTO, NOMBRE_DIRECTOR)

5.4.- RESTRICCIONES EN EL MODELO RELACIONAL

En el modelo relacional existen restricciones u ocurrencias no permitidas, es decir los datos que almacenamos en la base de datos deben adaptarse a la estructura marcada por la relación (restricción impuesta por el modelo de base de datos adoptada) y deben cumplir las restricciones impuestas por el usuario para que sean una ocurrencia válida del esquema.

Como restricciones impuestas por el modelo tenemos las características de una relación:

- No puede haber dos tuplas iguales.
- El orden de las tuplas no es significativo.
- El orden de los atributos no es significativo.
- Los atributos en las tuplas pueden tomar un solo valor de su dominio.
- **Regla de integridad de la entidad**: ningún atributo que forme parte de la clave primaria puede tomar un valor nulo.

Restricciones impuestas por el usuario:

- **Regla de integridad referencial**: los valores que puede tomar una clave ajena en una relación, o bien coinciden con los valores de la clave primaria que referencian o son nulos.

Por ejemplo si tenemos las relaciones:

EDITORIAL (NOMBRE_E, DIRECCION, CIUDAD, PAIS)

LIBRO (CODIGO, TITULO, IDIOMA,...,NOMBRE_E)

Los valores que puede tomar NOMBRE_E en la relación LIBRO será el de una editorial existente y por tanto formará parte de la relación EDITORIAL o bien lo desconocemos y por tanto tendrá un valor nulo.

Si tenemos el caso siguiente en el que la clave ajena forma parte de la clave primaria de la relación ESCRIBIR, como las relaciones deben cumplir la regla de integridad de entidad, los valores de las claves ajenas no pueden ser nulos.

ESCRIBIR(COD_AUTOR, COD_LIBRO)

AUTOR(COD_AUTOR, NOMBRE)

LIBRO(COD_LIBRO, TITULO, TEMA)

Cuando se define el modelo relacional, además de establecer las claves ajenas hay que determinar las consecuencias que tienen las operaciones de BORRADO y MODIFICACIÓN realizadas sobre la relación referenciada, los casos que se pueden dar son los siguientes:

- **Operación restringida (RESTRICT)**: el borrado o modificación de la tupla que contienen la clave primaria referenciada, solo se permite si no existen tuplas con ese valor en la relación que contiene la clave ajena. Es decir para poder borrar una editorial no debería existir ningún libro editado en esa editorial, sino el sistema impediría el borrado.

  POR EJEMPLO, ORACLE, que es un SGBD de los más potentes y más utilizados, no permite otra restricción. No hace falta ni ponerlo, ya que es lo que en su motor funciona por defecto.

  En otros SGBD sí se permite. Pero… ¿cuándo se pone?, ¿dónde?. Se pone en la relación "hija" (llamamos relación "padre" a la relación cuya clave primaria se propaga, y relación "hija" a la relación a la que se propaga la clave de la padre, es decir, la hija siempre será la que tiene clave ajena).

  En nuestro ejemplo: EDITORIAL sería padre, y LIBRO sería hija.

  EDITORIAL (NOMBRE_E, DIRECCION, CIUDAD, PAIS)

  LIBRO (CODIGO, TITULO, IDIOMA,...,NOMBRE_E)

  CREAMOS LAS DOS TABLAS:

```
CREATE TABLE editorial (
        id INT NOT NULL,
        PRIMARY KEY (id)
);
CREATE TABLE libro(
        id INT NOT NULL,
        libro_id INT,
        INDEX libro_ind (libro_id),
        FOREIGN KEY (libro_id)
                REFERENCES libro (id)
                [RESTRICT|ON DELETE CASCADE…]
);
```

En ORACLE, cuando queremos poner restricciones de algún tipo, se pone la palabra clave por defecto "constraint".

Por ejemplo:

```
CREATE TABLE libro(..
…..
….
CONSTRAINT FOREIGN KEY id
REFERENCES editorial
);
```

- **Operación con transmisión en cascada (CASCADE)**: el borrado o modificación de la tupla que contiene la clave primaria referenciada, implica el borrado o modificación de las tuplas que contienen ese valor como clave ajena. Es decir la modificación del nombre de una EDITORIAL implica que se modifiquen los nombres de todas las tuplas de LIBRO que tienen el valor como clave ajena.

- **Operación con puesta a nulos (SET NULL)**: el borrado o modificación de la tupla que contiene la clave primaria referenciada, implica poner a nulos los valores de la clave ajena en la relación que referencia. En nuestro ejemplo si borramos una editorial, todos los libros que estaban publicados en esa editorial pasarán a tener nulo el valor de la clave ajena.

- **Operación con puesta a valor por defecto (SET DEFAULT)**: el borrado o modificación de la tupla que contiene la clave primaria implica que todas las claves ajenas que contenían ese valor en la relación que referencia pasan a tener un valor por defecto que se definió cuando se creó la tabla.

- **Operación que desencadena un procedimiento de usuario**: en este caso el borrado o modificación de la tupla que contiene la clave primaria pone en marcha un procedimiento que habrá sido definido por el usuario.

Existen otras restricciones impuestas por el usuario sobre el dominio que corresponde a cada uno de los datos utilizados, el tipo de datos, interrelaciones que pueden no estar permitidas, etc.

5.5.- NORMALIZACIÓN. INTRODUCCIÓN

En el año 1970 Codd definió la teoría de la Normalización, se trata de una serie de reglas o normas a seguir en el diseño de los datos que va a utilizar un sistema, para conseguir que los datos sean independientes de las aplicaciones que los vayan a utilizar.

Se habían realizado estudios sobre el rendimiento de los datos teniendo en cuenta las aplicaciones que los utilizan y lo que ocurría cuando cambian los requisitos, los resultados obtenidos demostraban que:

- Diferentes aplicaciones necesitaban diferentes diseños de datos.

- La información en muchos casos estaba duplicada.

- Era difícil y costoso mantener los datos actualizados.

Las ventajas que aportan el modelo relacional y la normalización son:

- Facilidad de uso: los datos se agrupan en tablas que identifican claramente cada entidad o interrelación.

- Flexibilidad: la información que precisan los usuarios se obtiene facilmente (uniendo tablas, seleccionando unos valores, etc.).

- Seguridad: los controles de acceso resultan sencillos de implementar tanto a nivel de relaciones como de atributos.

- Fáciles de implementar: las tablas se almacenan físicamente como archivos planos.

- Independencia de datos: los programas no están unidos a la estructura, y se pueden añadir nuevas tablas o nuevos atributos sin que afecte a los programas que los utilizan.

- Claridad: la representación de la información es clara y sencilla.

- Facilidad de gestión: los lenguajes manipulan la información de forma sencilla ya que los datos se basan en el álgebra y cálculo relacional.

- Mínima redundancia: la información no se repite de forma innecesaria.

- Máximo rendimiento de las aplicaciones: cada aplicación trata solamente los datos que le son de utilidad.

El diseño lógico de datos debe obtener una estructura óptima y esto se consigue minimizando el espacio que ocupan y consiguiendo que la gestión de la información sea rápida y eficiente.

5.6.- PRIMERA FORMA NORMAL (1FN)

Una relación está en primera forma normal 1FN si y solo si los valores que componen los atributos de una tupla son atómicos. En un atributo no pueden aparecer valores repetitivos, tienen que ser elementales y únicos.

Una tabla que no está en 1FN plantea los siguientes problemas:

- En cuanto al espacio que ocupa: desaprovechamiento del espacio de almacenamiento cuando hay pocos valores para ese atributo, o falta de espacio si aparecen valores nuevos del atributo.

- Dificultades en el tratamiento de ese atributo (actualizaciones, consultas o búsquedas) de

un valor determinado.

EMPLEADOS:

COD_EMP	NOMBRE	IDIOMA
001	M. FERNANDEZ	ESPAÑOL FRANCES
002	M. RODRÍGUEZ	ESPAÑOL

Una solución podría ser poner una columna para cada idioma posible:

COD_EMP	NOMBRE	IDIOMA1	IDIOMA2	IDIOMA3
001	M.FERNANDEZ	ESPAÑOL	FRANCES	
002	M.RODRIGUEZ	ESPAÑOL		

Esta solución supone una gran ocupación de espacio de almacenamiento y puede generar muchos valores nulos.

Para pasar una tabla a la 1FN se trabaja de la siguiente manera:

1.- Localizar la clave primaria.

2.- Descomponer la tabla realizando proyecciones:

2.1.- Se forma una tabla con la clave primaria y los atributos que tienen valores únicos, esta tabla recibe el nombre de la tabla original:

T_EMPLEADOS

COD_EMP	NOMBRE
001	M. FERNANDEZ
002	M. RODRIGUEZ

2.2.-Se forma otra tabla con la clave y los atributos multivaluados pero distribuyendo cada valor en una tupla, existiendo un solo valor por cada fila. La tabla toma el nombre de los atributos que la forman:

T_IDIOM_EMP1:

COD_EMP	IDIOMA
001	ESPAÑOL
001	FRANCES
002	ESPAÑOL

La clave estaría formada por los dos atributos que forman la clave. Una clave compuesta siempre es más difícil de manejar (menos eficiente) que una clave primaria formada por un solo atributo, por esta razón en muchos casos, sobre todo cuando los dominios de los atributos son muy diferentes se forma otra tabla:

T_IDIOMAS

COD_IDIOMA	IDIOMA
01	ESPAÑOL
02	FRANCES
03	INGLES

T_IDIOM_EMP2:

COD_EMP	COD_IDIOMA
001	01
001	02
002	01

De esta forma para interrelacionar la tabla T_EMP con la tabla T_IDIOM_EMP2 se utilizan los tres primeros caracteres de COD_EMP_IDIM, para interrelacionar la tabla T_IDIOMA con T_IDIOM_EMP2 se utilizan los dos últimos caracteres.

Conocer cuantos empleados hablan un idioma supone manejar una sola clave y por tanto un acceso más rápido.

5.7.- DEPENDENCIA FUNCIONAL:

Se dice que un atributo o conjunto de atributos B depende funcionalmente del atributo o conjunto de atributos A (A $\longrightarrow$ B) si y solo si cada valor de A determina un único valor de B.

Por ejemplo entre DNI y NOMBRE existe una dependencia funcional ya que un DNI determina un y solo un NOMBRE.

DNI $\longrightarrow$ NOMBRE (DNI determina NOMBRE)

Si suponemos que los nombres no se repiten tendríamos que:

NOMBRE $\longrightarrow$ DNI (NOMBRE determina DNI)

en este caso tenemos que la dependencia funcional se produce de forma biunívoca, pero no se da siempre esta situación.

Supongamos que una persona vive en una sola casa y los atributos DNI, CALLE:

DNI $\longrightarrow$ CALLE (DNI determina CALLE)

pero CALLE $\longrightarrow$ DNI (CALLE no determina DNI) porque en una misma calle viven muchas personas.

Hay atributos entre los que no existe ninguna dependencia funcional, si consideramos DNI_AUTOR y TITULO_LIBRO, tendremos:

DNI_AUTOR $\not\longrightarrow$ TITULO_LIBRO ya que un autor puede haber escrito muchos libros.

TITULO_LIBRO $\not\longrightarrow$ DNI_AUTOR ya que hay libros escritos por varios autores y por tanto:

DNI_AUTOR $\not\longleftrightarrow$ TITULO_LIBRO

Hay situaciones en las que no basta un atributo para determinar un único valor de otro atributo. Consideremos la situación (DNI, EMPRESA, SUELDO) en la que una misma persona puede trabajar en varias empresas y dependiendo de la empresa percibe diferente sueldo:

DNI,EMPRESA $\longrightarrow$ SUELDO

También se da la situación contraria, un único atributo determina varios atributos:

DNI $\longrightarrow$ NOMBRE|CALLE

El proceso de normalización se basa en distintas variantes de la dependencia funcional. Por tanto es importante conocer las dependencias funcionales sobre todo las que relacionan atributos de distintas entidades.

5.8.- DEPENDENCIA FUNCIONAL TOTAL:

Un atributo Y tiene una dependencia funcional total o completa con el atributo X si tiene una dependencia funcional con X y no depende funcionalmente de ningún subconjunto de X.

Por ejemplo si tenemos un sistema formado por las personas que trabajan, donde una misma persona puede trabajar en varias empresas y dependiendo de la empresa percibe diferentes sueldos, tendremos:

DNI,EMPRESA $\longrightarrow$ NOMBRE

esta dependencia no es total ya que nombre queda determinado si solamente damos DNI, pero

DNI,EMPRESA ⟶ SUELDO

si es una dependencia funcional total ya que para conocer el sueldo necesitamos determinar la persona y la empresa.

5.9.- SEGUNDA FORMA NORMAL (2FN)

Una relación (tabla) se dice que está en segunda forma normal si cumple las siguientes condiciones:

. Está en 1FN
. Todo atributo secundario, es decir, que no forma parte de la clave, depende totalmente de la clave completa y no de una parte de ella. Es decir, tiene una dependencia funcional completa respecto de la clave.

Si la clave primaria está formada por un solo atributo, ya se encuentra en segunda forma normal.

Si consideramos el caso anterior de los empleados que trabajaban en varias empresas, tendríamos que nuestro sistema maneja los siguientes atributos:

(DNI, NOMBRE, EMPRESA, SUELDO)

Elegimos la clave (por definición, todos los atributos deben depender funcionalmente de la clave, con lo cual, la clave será DNI, EMPRESA), ya que si elegimos como clave DNI no podríamos sacar el sueldo.

Las dependencias funcionales son las siguientes:

DNI ⟶ NOMBRE
DNI,EMPRESA ⟶ SUELDO

Vemos que NOMBRE depende de sólo una parte de la clave, por lo tanto, no se encuentra en segunda forma normal, para ello podemos dividir la relación en dos:

EMPLEADO (DNI, NOMBRE)
EMPR_EMP (DNI, EMPRESA, SUELDO)

EJEMPLO 1: ALMACEN(COD_ALM, DIR, COD_PIEZA, CANTIDAD)

COD_ALM	DIR	COD_PIEZA	CANTIDAD
A1	SAN PABLO	P1	100
A1	SAN PABLO	P2	500
A2	MADRID	P1	300

1.- Buscamos la clave primaria: COD_ALM,COD_PIEZA

Está en Primera Forma Normal

2.- ¿Todo atributo secundario tiene una dependencia funcional completa de la clave principal?, es decir ¿está en 2ª FN?

No ya que la dirección sólo depende del código de almacén. Debemos, por tanto, dividir la relación:

ALMACEN(COD_ALM,DIR)

ALMACEN_PIEZA(COD_ALM,COD_PIEZA,CANTIDAD)

Ahora tendríamos dos tablas en segunda forma normal

EJEMPLO 2: Consideramos un sistema de préstamos de libros a socios, donde no se permite que el mismo socio coja el mismo libro más de una vez, la información se almacena de la forma siguiente:

PRESTAMOS(NUM_SOCIO,NOMBRE_SOCIO,COD_LIBRO,FECHA_PRESTAMO, EDITORIAL, PAIS)

1.- Clave primaria: NUM_SOCIO,COD_LIBRO

2.- Está en primera forma normal.

3.- Dependencias funcionales:

NUM_SOCIO ⟶ NOMBRE_SOCIO
COD_LIBRO ⟶ EDITORIAL
COD_LIBRO ⟶ PAIS
EDITORIAL ⟶ PAIS
NUM_SOCIO,COD_LIBRO ⟶ FECHA_PRÉSTAMO

No está en 2ªFN, por lo tanto, dividimos em tablas:

Nos quedarían tres tablas:

SOCIOS(NUM_SOCIO, NOMBRE_SOCIO)

LIBROS(COD_LIBRO,EDITORIAL,PAIS)

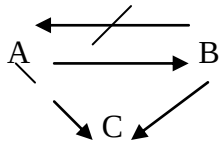
PRESTAMOS(NUM_SOCIO,CODIGO_LIBRO,FECHA_PRESTAMO)

5.10.- DEPENDENCIA FUNCIONAL TRANSITIVA:

Si tenemos tres subconjuntos distintos de atributos A, B y C, pertenecientes a la misma relación (tabla) T, de forma que se cumplen las siguientes condiciones:

$A \rightarrow B$ y $B \not\rightarrow A$

se dice que C tiene una dependencia funcional transitiva con A si se cumple que $B \rightarrow C$.



Esto quiere decir que un atributo (C) se puede conocer a través de diferentes vías y por tanto se produce una redundancia de información.

Consideremos el siguiente ejemplo: son datos asociados a un alumno: NUMMAT (número de matrícula), GRUASI (grupo asignado), AULGRU (aula correspondiente al grupo). Se cumplen los siguientes requisitos:

- Un alumno solo tiene asignado un grupo.
- A un grupo le corresponde un solo aula.

en este caso nos encontramos ante las siguientes dependencias funcionales:

NUMMAT $\longrightarrow$ GRUASI|ALULGRU
GRUASI $\longrightarrow$ AULGRU

el atributo AULGRU depende transitivamente de NUMMAT ya que se puede conocer por medio del atributo GRUASI

5.11.- TERCERA FORMA NORMAL:

Una relación (tabla) se dice que está en tercera forma normal si y solo si se cumplen las siguientes condiciones:

- está en 2FN
- ningún atributo secundario depende transitivamente de la clave de la relación.

Esto quiere decir que un atributo secundario (los que no son posibles claves) solo se puede conocer a través de las claves candidatas nunca mediante otro atributo secundario.

EJEMPLO 1: Cogemos el ejemplo anterior, el sistema de préstamos de libros a socios, donde no se guardan archivos históricos, y los nombres y números de socios no se repiten.

PRESTAMOS (NUM_SOCIO, NOMBRE_SOCIO, COD_LIBRO, FECHA_PRÉSTAMO, EDITORIAL, PAIS)

No está en 2ªFN, la dividimos:

SOCIOS(NUM_SOCIO, NOMBRE_SOCIO)

LIBROS(COD_LIBRO,EDITORIAL,PAIS)

PRESTAMOS(NUM_SOCIO,CODIGO_LIBRO,FECHA_PRESTAMO)

Si observamos la 2ª tabla, tenemos:

COD_LIBRO	EDITORIAL	PAIS
L1	RAMA	ESPAÑA
L2	RAMA	ESPAÑA
L3	MASSON	ITALIA

Vemos que está en 2ªFN, pero ¿está en 3ª FN?, pues no, ya que el atributo PAIS depende transitivamente de COD_LIBRO, ya que se puede conocer por medio del atributo EDITORIAL.

COD_LIBRO ⟶ EDITORIAL
COD_LIBRO ⟶ PAIS
EDITORIAL ⟶ PAIS

Por tanto, tendríamos que dejar esta relación en 3ªFN, para ello, dividimos la relación en dos realciones a su vez:

LIBRO (COD_LIBRO, EDITORIAL)
EDITORIAL (EDITORIAL, PAIS)

EJEMPLO 2: Si tenemos la relación:

EMPLEADOS(COD_EMPLEADO,COD_DEPARTAMENTO,NOMBRE_DEPARTAMENTO)

la clave sería COD_EMPLEADO, veamos cual es su Forma Normal:

1.- Está en primera forma normal ya que los atributos de todas las tuplas son monovalentes.

2.- También está en segunda forma normal ya que los atributos no claves tienen una dependencia funcional completa de la clave.

3.- No está en tercera forma normal porque el atributo NOMBRE_DEPARTAMENTO tiene una dependecia transitiva respecto de la clave.

Por tanto partiremos la relación en la siguiente forma:

EMPLEADOS(COD_EMPLEADO, COD_DEPARTAMENTO)

DEPARTAMENTOS(COD_DEPARTAMENTO, NOMBRE_DEPARTAMENTO)

5.12.- FORMA NORMAL DE BOYCE-CODD (FNBC)

Esta forma normal se creó para evitar anomalías que aparecen cuando la clave es compuesta y un atributo que no pertenece a la clave determina una parte de ésta. Si la clave es simple, y la tabla está en 3ªFN también esté en Forma Normal de Boyce-Codd. La definición formal sería:

Una tabla está en Forma Normal de Boyce-Codd si y sólo si las únicas dependencias funcionales elementales son las que la clave principal determina los atributos.

Supongamos la siguiente tabla:

CALLEJERO:

DIREC	CIUDAD	CODPOS
Pez, 2	Getafe	09094
Luz, 5	Getafe	09094
Mar, 4	Madrid	09001
Sol, 4	Madrid	09006
Sal, 7	Madrid	09001
Mar, 6	Getafe	09094

La clave primaria para esta tabla sería DIREC, CIUDAD, ya que una misma calle puede existir en distintas ciudades. Nos encontramos con las siguientes dependencias funcionales:

DIREC + CIUDAD → CODPOS
CODPOS → CIUDAD

Hay un atributo que no forma parte de la clave que determina parte de ella, es decir, parte de la clave depende funcionalmente de un atributo de la relación. Para que quedara en Forma Normal de Óbice Codd, dividiríamos la relación en dos relaciones:

CALLEJERO(DIREC, CODPOS)
CODIGOS(CODPOS, CIUDAD)

Ejemplo 2: Dado un curso, queremos almacenar la siguiente información: asignatura, número de clase y código del profesor. La clave está formada por ASIG + NUMCLA (un número de clase se puede repetir), pues una clase la pueden usar varios profesores. Sabemos también que un profesor imparte una única asignatura, por tanto, las dependencias funcionales serían:

ASIG + NUMCLA → CODPROF
CODPROF → ASIG

No está en Forma Normal de Boyce Codd, solución:

AULA (NUMCLA, CODPROF)
PROFESOR (CODPROF, ASIG)