UNIDAD 3

MODELO LÓGICO. MODELO RELACIONAL
DISEÑO Y NORMALIZACIÓN DE LA BBDD RELACIONALES

1. INTRODUCCIÓN

En la unidad anterior se ha obtenido un esquema conceptual empleando el Modelo Entidad/Relación. Dicho esquema corresponde al ámbito conceptual, al mundo de las ideas. Ahora vamos a desarrollar el **Diseño de los Datos.**

La fase de análisis de los datos se centra en qué datos debe recordar la aplicación. En cambio, la fase de diseño de datos da un paso más en el acercamiento a la solución. Trata de encontrar cómo organizar la información para que pueda ser manejada por el ordenador.

El diseño de los datos consta de dos partes: diseño de estructuras en las que se almacenarán los datos y diseño de consultas, donde se recogerán los accesos que se realicen sobre los datos. Al esquema de la BD obtenido en el diseño se le denomina esquema lógico de datos. Existen diferentes modelos lógicos para desarrollar el esquema lógico de la BD. Nosotros emplearemos el más utilizado, el Modelo Relacional (MR).

La teoría del Modelo Relacional se desarrolló hacia 1970 de la mano de E. Codd, que propuso también tres lenguajes de definición y manipulación de datos basados en el Álgebra de conjuntos y el Cálculo de predicados de primer orden. Desde entonces, el Modelo Relacional se ha impuesto claramente sobre sus inmediatos predecesores, el Jerárquico y el Red, por su sencillez y por la aparición de un lenguaje de definición de datos, el SQL (Standard Query Language), de fuerte aceptación como lenguaje de manipulación de datos.

Al encontrarnos en la fase de diseño lógico, el modelo de datos que utilicemos influirá directamente en el tipo de BD y SGBD a utilizar posteriormente. Por tanto, si utilizamos durante el diseño lógico el modelo relacional será porque después utilizaremos un **SGBD relacional** (y por tanto, una **BD relacional**). Así, al tratar las principales características del modelo relacional estaremos también tratando las de las BD relacionales.

Veremos las normas a seguir para poder **transformar el modelo E/R al MR**. Este proceso también se llama Paso a Tablas.

2. CONCEPTOS BÁSICOS DEL MODELO RELACIONAL.

El modelo relacional se basa en dos ramas de las matemáticas: la teoría de conjuntos y la lógica de predicados de primer orden. El hecho de que el modelo relacional esté basado en la teoría de las matemáticas es lo que lo hace tan seguro y robusto.

Hay quien ofrece una cierta resistencia a estudiar complicados conceptos matemáticos para tan sólo llevar a cabo una tarea bastante concreta. Es habitual escuchar quejas sobre que las teorías matemáticas en las que se basa el modelo relacional y sus metodologías de diseño, no tienen relevancia en el mundo real o que no son prácticas. No es cierto: las matemáticas son básicas en el modelo relacional. Pero, por fortuna, **no hay que aprender teoría de conjuntos o lógica de predicados de primer orden para utilizar el modelo relacional**.

Sería como decir que hay que saber todos los detalles de la aerodinámica para poder conducir un automóvil...

La teoría matemática proporciona la base para el modelo relacional y, por lo tanto, hace que el modelo sea predecible, fiable y seguro. La teoría describe los elementos básicos que se utilizan para crear una base de datos relacional y proporciona las líneas a seguir para

construirla. El organizar estos elementos para conseguir el resultado deseado es lo que se denomina diseño.

2.1. TABLAS

La estructura fundamental del modelo es la **tabla o relación,** representación de la información referente a un objeto; representa una entidad genérica. Una tabla está formada por columnas y filas, cada **columna** representa un atributo de la entidad genérica y cada **fila o tupla** representa una entidad concreta, una ocurrencia o ejemplar.

Una tabla o relación está formada por dos partes principales: cabecera y cuerpo. La **cabecera** es un conjunto fijo de atributos. El **cuerpo** es un conjunto de tuplas variable en el tiempo.

El número de filas o tuplas en una tabla recibe el nombre de **cardinalidad**. El número de columnas o atributos en una tabla recibe el nombre de **grado**.

TABLA "CLIENTES"

Cabecera

Cuerpo

DNI	NOMBRE	APELLIDOS	TELÉFONO
25001223	Juan Pedro	López González	966223445
27999888	Ana	Saura Jiménez	968345555
12887637	Nuria	Pérez Andreu	965887444
25669884	Sara	Bermúdez Ríos	966885447
19000444	Antonio	Ruíz Abellán	966550987

Cardinalidad de "CLIENTES": 5

Grado de "CLIENTES": 4

Ejemplo de tupla de CLIENTES: (27999888, Ana, Saura Jiménez, 968345555)

La **Cabecera de la tabla** es invariante en el tiempo (a no ser que cambie el diseño), sin embargo, el **Cuerpo de la tabla** varía en el transcurso del tiempo, así como la **Cardinalidad**.

A la hora de describir o definir una tabla, podemos hacerlo indicando su esquema o mostrando la instancia de la tabla:

- **Esquema de una tabla:**

AUTOR (NOMBRE: texto, NACIONALIDAD: texto, INSTITUCION: texto**)**

Atributo Tipo de Dato

- **Instancia de una tabla:**

AUTOR		
NOMBRE	**NACIONALIDAD**	**INSTITUCION**
Date, C.J.	Norteamericana	Relational Ins.
De Miguel, A.	Española	FIM
Ceri,S.	Italiana	Politécnico Milán

2.2. CLAVES

Una **clave candidata** de una tabla es un conjunto no vacío de atributos que identifican unívoca y mínimamente cada tupla. Una tabla puede tener más de una clave candidata, entre las cuales se debe distinguir:

- **Clave primaria** (Primary Key o PK)**:** es aquella clave candidata que el usuario escogerá, por consideraciones ajenas al modelo relacional, para identificar las tuplas de la tabla.

- **Claves alternativas**: son aquellas claves candidatas que no han sido escogidas como clave primaria.

Se denomina **clave ajena** (clave foránea, Foreign Key o FK) de una tabla a un conjunto no vacío de atributos cuyos valores han de coincidir con los valores de la clave primaria de otra tabla. Es decir, la clave ajena de una tabla es el atributo o conjunto de atributos que forman la clave primaria de otra tabla. Cabe destacar que la clave ajena y la correspondiente clave primaria han de estar definidas sobre el mismo tipo de dato.

EDITORIAL (nombre_e: texto, direccion:texto, ciudad:texto, pais:texto **)**

PK: nombre_e

LIBRO(cidigo: numero, titulo: texto, idioma: texto,.... , nombre_e: texto **)**

PK: codigo

FK: nombre_e → EDITORIAL

Esta clave que referencia a EDITORIAL debe concordar con la clave primaria de EDITORIAL.

3. RESTRICCIONES DE INTEGRIDAD EN EL MODELO RELACIONAL.

Las **restricciones de integridad** intentan asegurar que la información contenida en la BD es correcta, es decir, que refleja fielmente la realidad. A continuación introduciremos los tipos de restricciones de integridad más importantes en el modelo relacional y algunas propiedades relacionadas con éstas.

En el modelo relacional, las tablas o relaciones tienen las siguientes propiedades y restricciones:

- Toda tabla ha de poseer una **clave primaria**, la cual tiene las siguientes restricciones:

Los atributos que la componen deben ser mínimos en el sentido de que la eliminación de cualquiera de ellos le haría perder su carácter identificador.

No pueden existir dos tuplas de una misma tabla con valores idénticos en su clave primaria (**Restricción de integridad de clave**).

- El orden de las tuplas es irrelevante.

- El orden de los atributos es irrelevante.

Todo atributo tiene un **tipo de dato** asociado, un conjunto de valores sobre los que toma un valor. Cuando definimos una tabla, especificaremos el tipo de dato correspondiente a cada uno de sus campos o atributos; a partir de entonces, ese campo o atributo:

- No podrá contener dos o más valores distintos para una misma ocurrencia, es decir, cada atributo puede tomar un único valor, en un momento y ocurrencia determinados. (Evidentemente, ese valor sí podrá cambiar a lo largo del tiempo).

- No podrá contener un valor perteneciente a un tipo de dato distinto del especificado.

3.1. VALORES NULOS. VNN

Cabe destacar la posibilidad de que los atributos contengan valores nulos (NULL) o bien tengan prohibida esa posibilidad. Un **valor nulo** es ausencia de información, un dato que se desconoce; podemos decir que NULO no es un valor en sí mismo, sino un indicador de ausencia de información. No tiene nada que ver con valores 0, puesto que estos son valores conocidos. Por ejemplo, de una persona podemos no conocer su teléfono. Sin embargo, esto no implica que esa persona no posea ese servicio, sino, simplemente, no disponemos de esa información. Por otro lado, el valor nulo no se puede comparar con otros valores; únicamente podemos saber si un atributo es nulo o no lo es. Así, si el sueldo de un empleado es desconocido, obviamente, no podemos interrogar a la BD sobre si esa persona es la que más cobra de toda la plantilla.

En la definición de cada tabla de la BD se indicará qué atributos pueden o no contener nulos. Si un atributo puede contener valor nulo y se inserta una tupla sin valor para tal atributo, el sistema le asignará valor nulo; en caso contrario, si un atributo no puede contener nulo y se intenta la inserción anterior, el sistema rechazaría la acción.

Ahora que conocemos el significado de los valores nulos, podríamos enunciar la **Restricción de integridad de entidad** vista anteriormente de la siguiente forma: **“Ningún atributo que pertenezca a la clave primaria puede contener un valor nulo”**.

3.2. RESTRICCIÓN DE INTEGRIDAD REFERENCIAL.

Dentro de las restricciones de integridad destaca la **Restricción de integridad referencial** que establece que "Si una tabla T1 contiene un atributo que es clave ajena de T2 (es decir, clave primaria en T2), su valor debe: o bien coincidir con el valor que tenga como clave primaria en T2, o bien ser nulo". Esta restricción se define en el esquema y el SGBD la reconoce sin necesidad de que se programe ni de que se tenga que escribir ningún procedimiento para obligar a que se cumpla.

Si el SGBD posee mecanismos de claves ajenas, debe garantizar la integridad referencial. Suponiendo que disponga de toda la información necesaria sobre claves (primarias y ajenas, al menos) se encuentra con el problema de las actualizaciones y borrados de claves primarias referenciadas por alguna clave ajena.

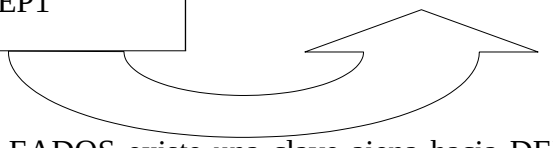
Veamos un ejemplo; consideremos el siguiente caso:

Tabla "EMPLEADOS":

Cód_emp	Nombre_emp	Depart_emp
001	Antonio	DEP1
002	José	
003	Susana	DEP2
004	Pilar	DEP1

Tabla "DEPARTAMENTOS":

Código	Nombre_dep
DEP1	Ventas
DEP2	Investigación
DEP3	Contabilidad



Observemos que en la tabla EMPLEADOS existe una clave ajena hacia DEPARTAMENTOS (campo Depart_emp).

Supongamos que se pretende borrar de la tabla DEPARTAMENTOS la tupla del "departamento de Investigación": (DEP2, Investigación). Si lo hiciéramos, la tupla de "Susana" en EMPLEADOS tendría su clave ajena apuntando a un valor de clave primaria que no existe en DEPARTAMENTOS, lo que violaría la integridad referencial.

También se puede dar el caso de que dicho departamento de Investigación cambie de clave primaria y pase a identificarse como "DEP5". Nuevamente, si no tomamos las medidas oportunas, estaríamos violando la integridad referencial.

En definitiva, para actualizaciones y borrados de tuplas referenciadas por claves ajenas en otras tablas, el SGBD debe conocer, por las especificaciones oportunas en el esquema de la BD, cuál de las siguientes estrategias debe utilizar para garantizar la restricción:

Rechazar

Anular

Propagar

Veamos cada una de esas estrategias:

RECHAZAR

El SGBD sólo permitirá la operación en caso de que no produzca problemas. En nuestro ejemplo no dejaría que realizásemos la operación.

ANULAR

El SGBD pondrá a nulos todas las claves ajenas que hagan referencia a la clave primaria que sufre la operación. En nuestro ejemplo, si pretendemos borrar la tupla de "Investigación", la tupla de "Susana" quedaría con el atributo Depart_emp con valor nulo:

Tabla "EMPLEADOS":

Cód_emp	Nombre_emp	Depart_emp
001	Antonio	DEP1
002	José	
003	Susana	
004	Pilar	DEP1

Tabla "DEPARTAMENTOS":

Código	Nombre_dep
DEP1	Ventas
DEP3	Contabilidad

Supongamos ahora, que para actualizar el departamento de Ventas a un nuevo código, cambiándolo de "DEP1" a "DEP4", adoptamos la misma estrategia. Obtendríamos:

Tabla "EMPLEADOS":

Cód_emp	Nombre_emp	Depart_emp
001	Antonio	
002	José	
003	Susana	
004	Pilar	

Tabla "DEPARTAMENTOS":

Código	Nombre_dep
DEP4	Ventas
DEP3	Contabilidad

PROPAGAR

También conocida como "en cascada", reproduce la operación sobre todas las tuplas que hagan referencia a la clave primaria. Si partimos nuevamente del ejemplo anterior (en el

estado inicial de las tablas) y borramos la tupla del departamento de Investigación, la tupla de "Susana" también sería borrada:

Tabla "EMPLEADOS":

Cód_emp	Nombre_emp	Depart_emp
001	Antonio	DEP1
002	José	
004	Pilar	DEP1

Tabla "DEPARTAMENTOS":

Código	Nombre_dep
DEP1	Ventas
DEP3	Contabilidad

Y en el caso añadido de la actualización antes mencionada del departamento de Ventas, quedaría así:

Tabla "EMPLEADOS":

Cód_emp	Nombre_emp	Depart_emp
001	Antonio	DEP4
002	José	
004	Pilar	DEP4

Tabla "DEPARTAMENTOS":

Código	Nombre_dep
DEP4	Ventas
DEP3	Contabilidad

Ya conocemos las distintas estrategias que puede seguir el SGBD para mantener la integridad referencial, pero: ¿cómo podemos indicarle la estrategia que ha de seguir en cada momento? Cada SGBD dispone de herramientas para especificar la estrategia a seguir para garantizar la restricción de integridad referencial.

En MySQL:

[CONSTRAINT símbolo] FOREIGN KEY (nombre_columna, ...) REFERENCES nombre_tabla (nombre_columna, ...)

[ON DELETE {CASCADE | SET NULL | NO ACTION| RESTRICT}]

[ON UPDATE {CASCADE | SET NULL | NO ACTION| RESTRICT}]

3.3. RESTRICCIONES DE INTEGRIDAD DE ATRIBUTO.

Sirven para indicar los valores posibles para un atributo o campo. Consiste en la especificación del tipo de dato sobre el que se define un atributo y en otras definiciones del atributo (como por ejemplo: un valor por omisión o si permite nulos o no).

No es necesaria una sentencia de creación explícita de la restricción, sino que forma parte de la definición de atributo dentro de la sentencia de creación de la tabla. Su definición puede consistir en la enumeración de valores posibles (Ej: 'rojo', 'amarillo', 'verde') o en una expresión de definición (Ej: horas > 0 AND horas< 100).

Veamos un ejemplo en el estándar SQL:

```
CREATE TABLE Vivienda (…
color_pared varchar(9) NOT NULL DEFAULT 'blanco',
color_techo varchar(9) ENUM ('rojo','amarillo','verde')
…);
```

Toda restricción de integridad de atributo es comprobada inmediatamente en cualquier intento de inserción de un nuevo valor o de modificación de ese atributo. La respuesta a una violación de la restricción siempre es el rechazo.

En Oracle,

```
CREATE TABLE vivienda (
    color_pared VARCHAR2(9) NOT NULL DEFAULT 'blanco',
        tamano VARCHAR2(10) CHECK( name IN ('small','medium','large') )
        );
```

3.4. RESTRICCIONES DE INTEGRIDAD DE TABLA.

Sirven para indicar los valores legales para cierta tabla. Se definen en el esquema de la correspondiente tabla. Se pueden referir a:

- Restricciones adicionales (además de las RI-Atributo) que tenga que cumplir algún atributo de la tabla (por ejemplo: que el salario de un empleado siempre esté entre 300 y 1200 €).

- Restricciones de clave candidata: si una tabla posee una clave candidata, ésta ha de tener valores únicos en esa tabla (que dos tuplas distintas no puedan tener el mismo valor en ese campo).

- Cualquier otro tipo de restricción que afecte a la tabla de forma general (por ejemplo: controlar que una tabla nunca esté vacía, o que su número de tuplas nunca sea mayor que 100).

Cuando se realiza una operación de modificación sobre una tabla, se chequean previamente todas las RI-Tabla de esa tabla. La respuesta a una violación de la restricción es por defecto el rechazo, aunque en la definición de la restricción se puede especificar la realización de una acción (la cual puede ser la ejecución de un determinado procedimiento).

En Oracle, las restricciones de integridad de tabla se definen con la láusula

CONSTRAINT (restricción de tabla).

3.5. RESTRICCIONES DE INTEGRIDAD DE BASE DE DATOS.

Sirven para indicar los valores legales para la base de datos en general; se trata de reglas que interrelacionan dos o más tablas distintas de la base de datos. Un ejemplo de este tipo de restricción es la Restricción de Integridad Referencial.

4. CONVERSIÓN DE UN ESQUEMA CONCEPTUAL E/R A UN ESQUEMA LÓGICO RELACIONAL.

A esta conversión de un esquema conceptual E/R a un esquema lógico relacional también se le denomina "**paso a tablas**". Este paso a tablas está basado en una serie de normas y pasos a seguir. A través de éstos se va generando el esquema relacional, es decir, el conjunto de tablas que finalmente se implementarán en el SGBD, y las restricciones o reglas de integridad que deben cumplir.

4.1. ENTIDADES.

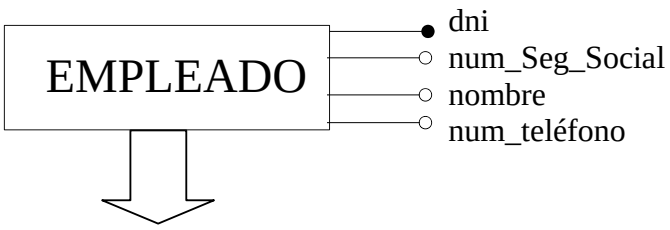
Por cada entidad del esquema E/R construiremos una tabla con su nombre y sus atributos (incluida la clave primaria).

Un posible formato para indicar el esquema de una tabla sería este:

NOMBRE_TABLA (Atributo1:Tipo_dato, Atributo2 :Tipo_dato, Atributo3: Tipo_dato)

PK: atributo1

Subrayaremos el atributo (o atributos) que formen parte de la clave primaria y abajo definiremos todas las restriciones (VNN, FK, enum, default, check).



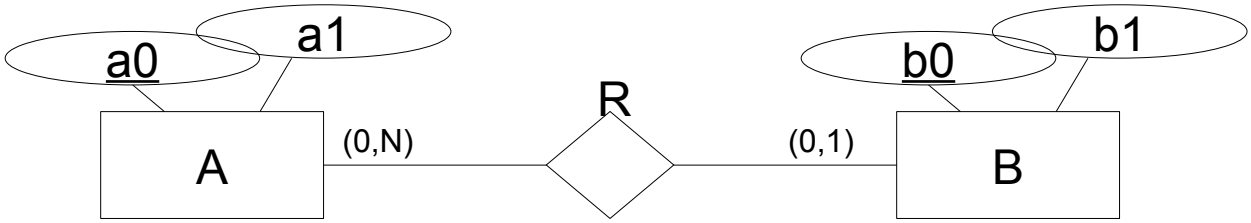
EMPLEADO(dni, num_Seg_Social, nombre, num_teléfono)

PK: dni

4.2. RELACIONES BINARIAS

Recordemos que las relaciones binarias son aquellas entre dos entidades. Este tipo de relaciones podrán o no generar una tabla, dependiendo de su cardinalidad:

a) RELACIONES CON CARDINALIDAD 1:N.



La clave primaria de la entidad 1 se incluye en la entidad N como clave ajena.

Se añade los atributos de la relación

A(a0, a1, b0)

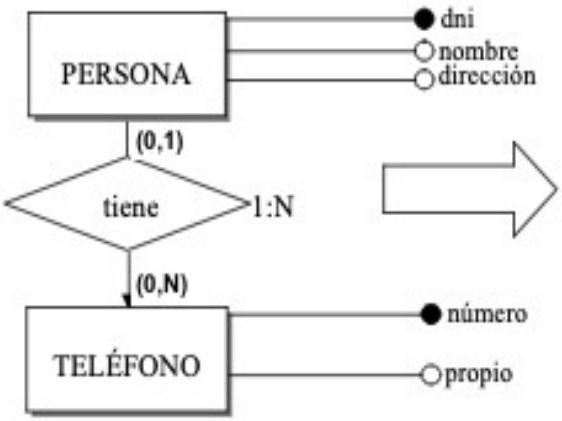
PK:a0

FK:b0 → B

B(b0, b1)

PK:b0

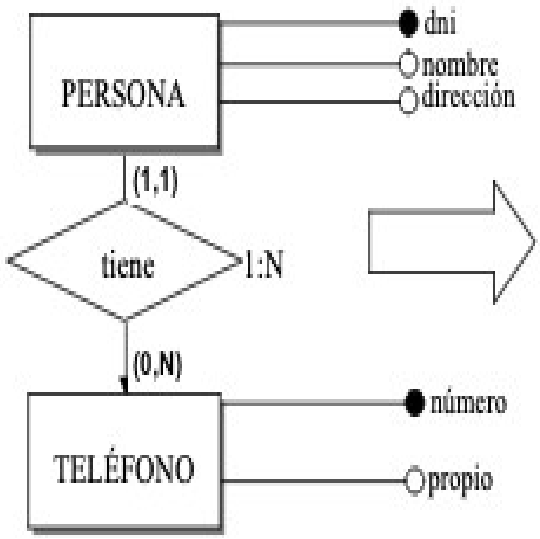
EJEMPLO 1 participación (0,1) (0,n):



PERSONA (dni, nombre, dirección)
pk: dni

TELEFONO (numero, propio, dni)
pk: numero
fk: dni → PERSONA

EJEMPLO 2 participación (1,1) (0,n):

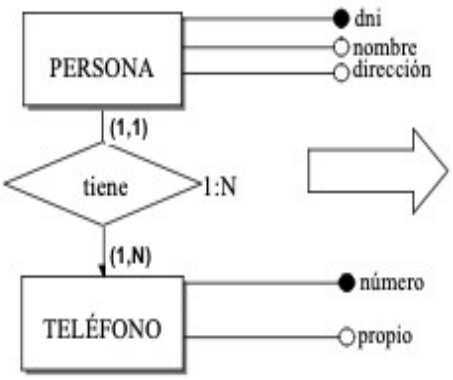


PERSONA (dni, nombre, dirección)
pk: dni

TELEFONO (numero, propio, dni)
pk: numero
VNN: dni
fk: dni → PERSONA

EJEMPLO 3 participación (x,1) (1,n):

1)



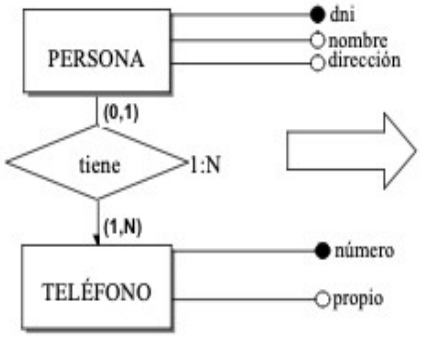
1)

PERSONA (<u>dni</u>, nombre, dirección)
pk: dni

TELEFONO (<u>numero</u>, propio, dni)
pk: numero
VNN: dni (solo sería en caso 1))
fk: dni → PERSONA

Perdida de semántica no se puede representar la cardinalidad mínima de TIENE TELEFONO

2)



2)

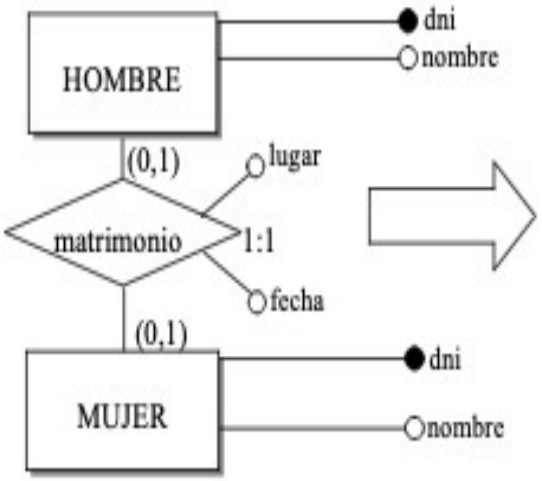
PERSONA (<u>dni</u>, nombre, dirección)
pk: dni

TELEFONO (<u>numero</u>, propio, dni)
pk: numero
fk: dni → PERSONA

Perdida de semántica no se puede representar la cardinalidad mínima de TIENE TELEFONO

b) RELACIONES CON CARDINALIDAD 1:1.

EJEMPLO 1 partición (0,1) (0,1)

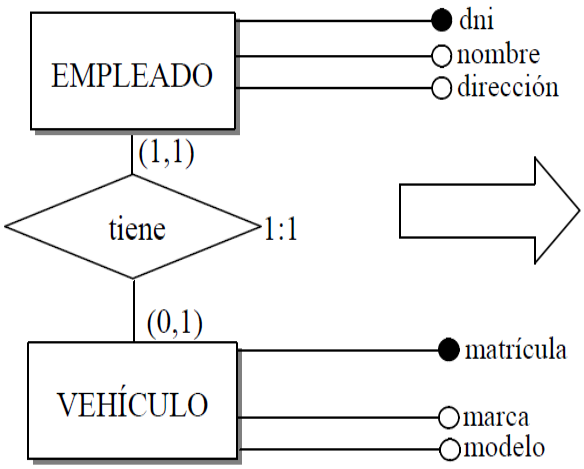


HOMBRE(<u>dni</u>, nombre)
PK: dni

MUJER(<u>dni</u>, nombre)
PK:dni

MATRIMONIO(<u>hombre</u>, mujer, lugar, fecha)
PK:hombre
AK:mujer
FK: hombre → HOMBRE
FK: mujer → MUJER

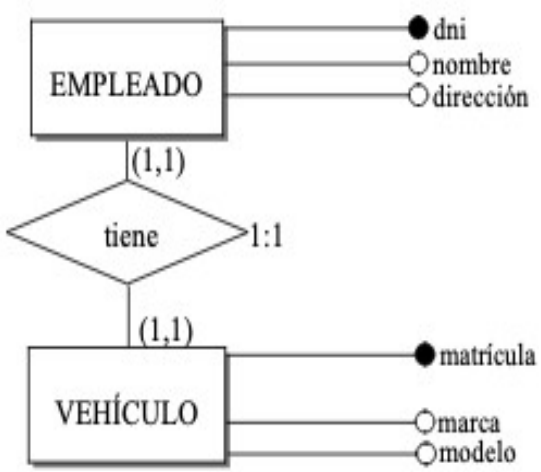
EJEMPLO 2 partición (1,1) (0,1)



EMPLEADO (dni, nombre, dirección)
PK: dni

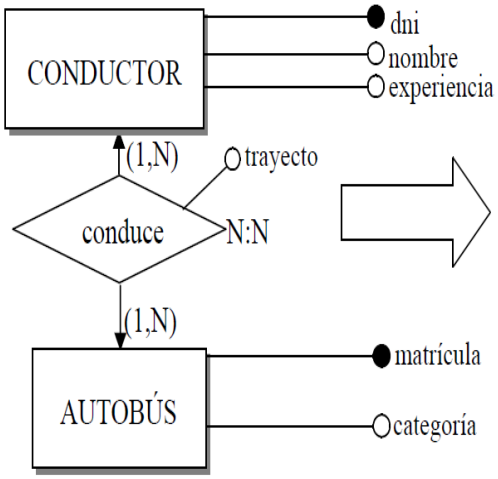
VEHÍCULO (matrícula, marca, modelo, dniEmpleado)
PK: matrícula
AK: dniEmpleado
FK: dniEmpleado → EMPLEADO

EJEMPLO 3 partición (1,1) (1,1)



TIENE (dni, nombre, dirección, matrícula, marca, modelo)
PK: dni
AK: matricula

c) RELACIONES CON CARDINALIDAD N:N.



CONDUCTOR(<u>dni,</u> nombre, experiencia)
PK: dni

AUTOBUS (<u>matricula</u>, categoria)
PK: matrícula

CONDUCIR (<u>dni*, matricula*</u>, trayecto)
PK: (dni, matricula)
FK: dni → CONDUCTOR
FK: matricula → AUTOBUS

La cardinalidad mínima no se puede representar.

Relación N:M con dimensión temporal

Ejemplo: Relación entre los clientes que alquilan los vehículos de una empresa de alquileres. Recogemos los alquileres realizados a nuestros clientes a lo largo del tiempo.

alquila (**Cod_Cliente, Matricula, F_Inicio**, F_Fin)

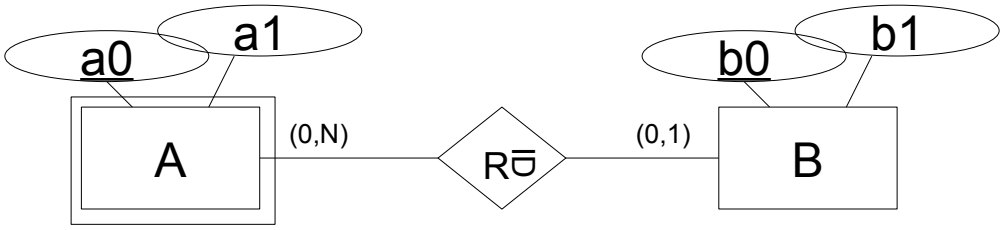
CLIENTE (**Cod_Cliente**, ……)

VEHICULO (**Matrícula**, ……)

Si la relación tiene atributos del tipo fecha será necesario incluir al menos uno en la PK.

4.3. DEPENDENCIA DE IDENTIFICACIÓN.

Cuando exista este tipo de dependencia, la clave primaria de la entidad fuerte debe introducirse en la tabla de la entidad débil y formar parte de la clave primaria de ésta, actuando además como clave ajena.



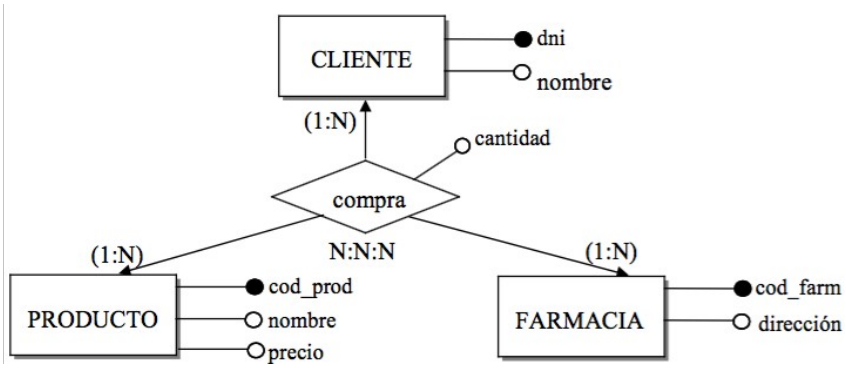
A(<u>a0</u>, <u>b0</u>, a1)
PK: (a0, b0)
FK: b0 → B

B(<u>b0</u>, b1)
PK:b0

4.4. RELACIONES DE GRADO N.

Se crea una nueva tabla con la unión de las claves primarias de las entidades relaciones. Si una de las entidades tiene cardinalidad máxima 1, se queda fuera de la PK.

Se crean tantas claves ajenas como entidades relacionadas.



CLIENTE(<u>dni</u>, nombre)

PK: dni

FARMACIA(<u>cod_farm</u>, dirección)

PK: cod_farm

PRODUCTO(<u>cod_prod</u>, nombre, precio)

PK: cod_prod

COMPRA(<u>dni, cod_prod, cod_farm</u>, cantidad)

PK: (dni, cod_prod, cod_farm)

FK: dni → CLIENTE

FK: cod_prod → PRODUCTO

FK: cod_farm → FARMACIA

4.5. RELACIONES REFLEXIVAS O RECURSIVAS.

La forma de actuar será la explicada para las relaciones binarias, dependiendo de la cardinalidad de la relación.

Si hay que añadir la clave principal como clave ajena en la misma tabla, se debe renombrar.

4.6. ATRIBUTOS COMPUESTOS Y MULTIVALUADOS.

a) ATRIBUTOS COMPUESTOS.

Se transforman en los atributos que los componen. Tenemos dos opciones:

- "Eliminar" el atributo compuesto y considerar todos sus componentes como atributos simples.
- "Eliminar" los componentes y considerar el atributo compuesto como un único atributo.

b) ATRIBUTOS MULTIVALUADOS.

Dan lugar a una nueva tabla cuya clave primaria es la concatenación de:

- La clave primaria de la entidad en la que se sitúa el atributo multivaluado, actuando también como clave ajena.
- El nombre del atributo multivaluado.

4.7. JERARQUÍAS DE GENERALIZACIÓN/ESPECIALIZACIÓN.

En lo que respecta a las jerarquías de entidades, no son objetos que se puedan representar explícitamente en el modelo relacional. Ante una jerarquía de entidades, caben varias soluciones de transformación al modelo relacional, con la consiguiente pérdida de semántica dependiendo de la estrategia elegida. Destacamos tres:

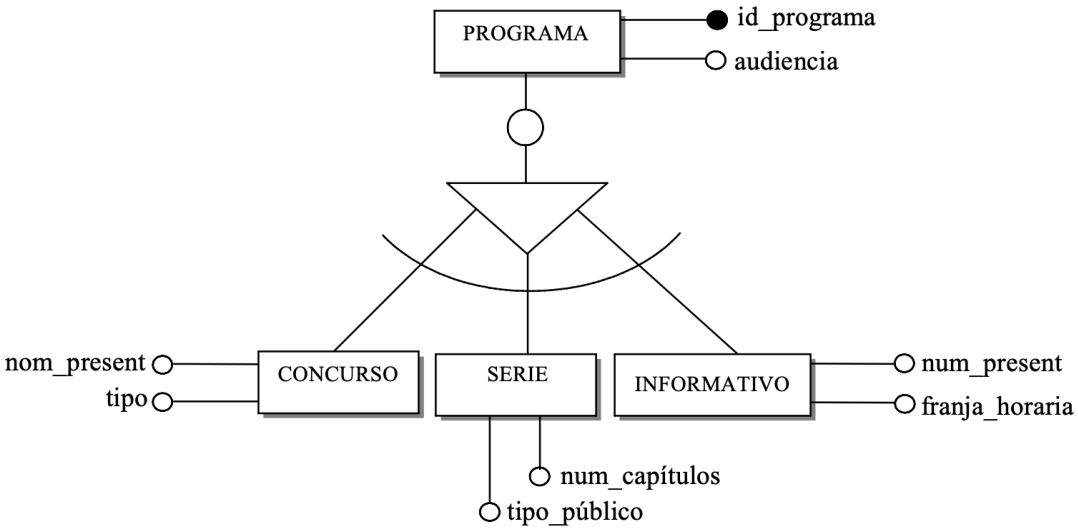
a) DIVIDIR

Consiste en:

- Crear una tabla por cada subtipo, heredando los atributos del supertipo, incluyendo la clave primaria, la cual pasará a ser la clave primaria de las nuevas tablas.
- La entidad supertipo desaparece.
- Todas las relaciones que tenía el supertipo, ahora pasan a multiplicarse como relaciones en los subtipos. Por tanto, habrá que estudiar esas nuevas relaciones y hacer las transformaciones pertinentes según lo visto en puntos anteriores.

Se realiza cuando el concepto del supertipo no se utiliza, y los subtipos son la información relevante.

También se detecta porque las relaciones y las operaciones que se realizan en la base de datos son siempre sobre los subtipos.



CONCURSO(id_concurso, nom_present, tipo, audiencia)
SERIE(id_serie, num_capítulos, tipo_público, audiencia)
INFORMATIVO(id_inf, num_present, franja_horaria, audiencia)

b) COLAPSAR

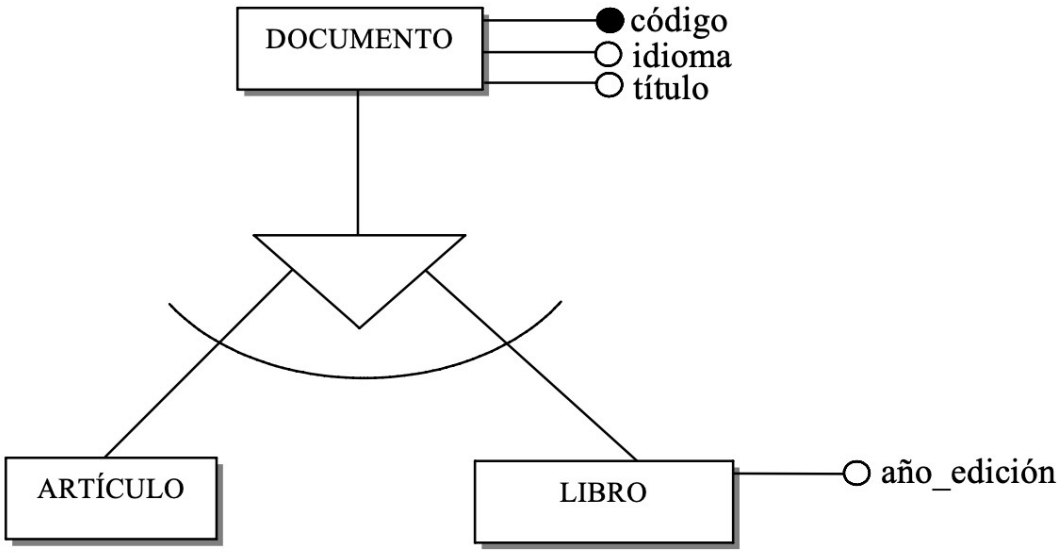
Consiste en integrar todas las entidades (supertipo y subtipos) en una tabla, incluyendo en ella los atributos de la entidad padre y los atributos de todos los hijos como opcionales.

En general, adoptaremos esta solución cuando los subtipos se diferencien en muy pocos atributos y las relaciones que los asocian con el resto de entidades del esquema sean las mismas para todos (o casi todos) los subtipos.

Para saber a qué subtipo pertenece ahora cada instancia de la **tabla podríamos añadir un atributo discriminante** (el atributo discriminante de la jeraquía) que indique el caso al cual pertenece la entidad en consideración. Si la jerarquía es solapada, el atributo discriminante será multivaluado (en ese caso es importante recordar que ese atributo requerirá una transformación posterior). Si es total será obligatorio.

Las relaciones que había sobre los subtipos, ahora son sobre el supertipo. Por tanto, habrá que estudiar esas relaciones y hacer las transformaciones pertinentes según lo visto en puntos anteriores.

Una desventaja es que ahora **habrá muchos valores nulos posibles**.



DOCUMENTO(<u>código</u>, idioma, título, tipo, año_edición)
Nota: tipo (por ser una jerarquía parcial) y año_edición (por provenir de un subtipo) serán opcionales.

c) EXPLICITAR

Consiste en crear una tabla para el supertipo y tantas tablas como subtipos haya, con sus atributos correspondientes.

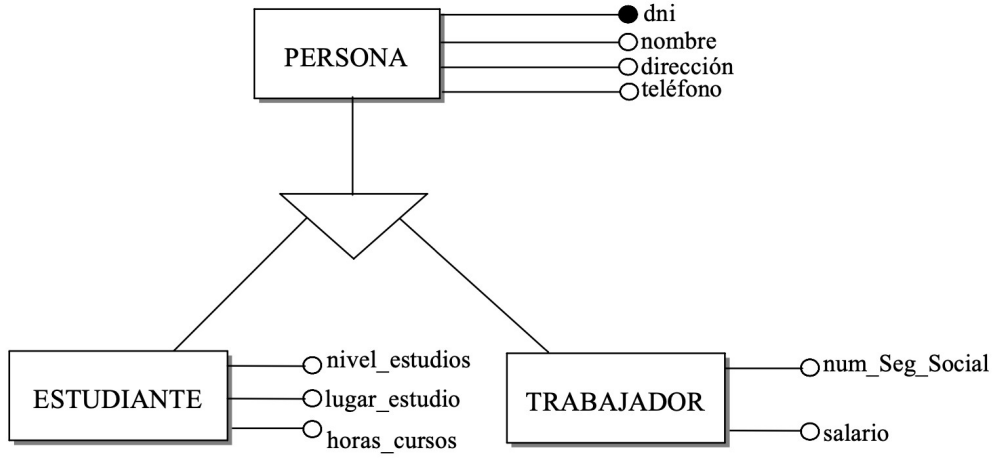
Si uno de los subtipos no posee ningún atributo ni ninguna relación propios, ese subtipo desaparece y no se le asigna tabla. En ese caso, si se trata de una jerarquía disjunta, el atributo discriminante de la jerarquía se incorpora en la tabla de la entidad supertipo.

Las tablas de los subtipos heredan como clave primaria la del supertipo. Por lo tanto, la clave primaria de las tablas de los subtipos es también una clave ajena a la del supertipo.

Adoptaremos esta opción cuando existe distinción entre el supertipo y los subtipos, y unas veces se actúa sobre el supertipo y otras veces sobre los subtipos, existiendo también relaciones distintas de otras entidades con el supertipo y con los subtipos.

Es la solución adecuada cuando existen muchos atributos distintos entre los subtipos y se quieren mantener de todas maneras los atributos comunes a todos ellos en una tabla.

Normalmente esto ocurre en jerarquías parciales, aunque esta opción sirve para cualquier tipo de jerarquía.



PERSONA(<u>dni</u>, nombre, dirección, teléfono)
ESTUDIANTE(<u>dni*</u>, nivel_estudios, lugar_estudio, horas_cursos)
TRABAJADOR(<u>dni*</u>, num_Seg_Social, salario)

5. NORMALIZACIÓN.

5.1. INTRODUCCIÓN.

¿Crees que tu base de datos ya podría construirse directamente sobre el SGBD relacional que hayas elegido? La respuesta podría ser afirmativa, pero si queremos que nuestra base de datos funcione con plena fiabilidad, es necesario antes llevar a cabo un proceso de normalización de las tablas que la componen. Así pues, una vez que se ha obtenido el Esquema Relacional del sistema, este debe ser analizado para comprobar que no presenta pérdida de información e inconsistencias. Es la Teoría de la Normalización la que nos permite identificar esos casos y nos muestra la forma de convertir esas tablas a una forma más deseable.

Veamos un ejemplo: en la figura siguiente se muestra una tabla denominada ESCRIBE que almacena datos sobre autores de libros y sobre los propios libros, con una clave primaria formada por: Autor y Cod_libro.

AUTOR	NACIONALIDAD	COD_LIBRO	TITULO	EDITORIAL	AÑO
DATE, C	NORTE AMERICANA	98987	DATABASES	ADDISON-W	1990
DATE, C.	NORTE AMERICANA	97777	SQL STANDARD	ADDISON-W	1986
DATE, C.	NORTE AMERICANA	98988	A GUIDE TO INGRES	ADDISON-W	1988
CODD, E.	NORTE AMERICANA	7980	RELATIONAL MOD.	ADDISON-W	1990
GARDARIN	FRANCESA	12345	BASES DE DATOS	PARANINFO	1986
GARDARIN	FRANCESA	67890	COMPARACION BD	EYROLLES	1984
VALDURIEZ	FRANCESA	67890	COMPARACION BD	EYROLLES	1984
KIM, W.	NORTE AMERICANA	11223	0-0 DATABASES	ACM PRESS	1989
LOCHOVSKY	CANADIENSE	11223	0-0 DATABASES	ACM PRESS	1989

Si observamos con atención esta tabla, vemos que presenta varios problemas:

Gran cantidad de **redundancia**, ya que la nacionalidad del autor se repite por cada ocurrencia del mismo, y algo análogo sucede, cuando un libro tiene más de un autor, con la editorial y el año de publicación.

Anomalías de modificación, ya que, inadvertidamente podemos, por ejemplo, cambiar el nombre de la editorial en una tupla sin modificarlo en el resto de las que corresponden al mismo libro, lo que da lugar a incoherencias.

Anomalías de inserción, ya que si se quisiera incluir información sobre algún autor del que no hubiera ningún libro en la base de datos, no sería posible, al formar el atributo cod_libro parte de la clave primaria de la tabla, ni tampoco podríamos introducir obras anónimas (la regla de integridad de entidad no permite nulos en ningún atributo que forme parte de una clave primaria). Además, la inserción de un libro que tuviera dos autores obligaría a incluir dos tuplas en la base de datos.

Anomalías de borrado, ya que si quisiéramos dar de baja un libro, también se perderían datos sobre sus autores (si estos no habían escrito nada más que un libro) y, viceversa, si borramos un autor desaparecerían de nuestra base de datos los libros escritos por él (a no ser que tuviera el libro más de un autor).

Vemos, por tanto, que la actualización (alta, baja o modificación) de un solo libro o de un solo autor nos puede obligar a actualizar más de una tupla, dejándose la integridad en manos del usuario; además de la falta de eficiencia asociada a estas múltiples actualizaciones.

Esta tabla presenta todos estos problemas, y alguno más, debido a que atenta contra un principio básicoen todo diseño: “hechos distintos se deben almacenar en objetos distintos”, en este caso, en tablas distintas, con lo que se habrían evitado redundancias y, por tanto, anomalías de actualización.

Si se hubiera seguido la metodología adecuada, realizando un diseño conceptual en el modelo E/R, seguido de una cuidadosa transformación al modelo relacional, se evitarían estos problemas y

se obtendría un esquema relacional exento de errores. Sin embargo, ante las posibles dudas respecto a si un determinado esquema relacional es correcto, será preferible aplicar a dicho esquema un método formal de análisis que determine lo que pueda estar equivocado en el mismo y nos permita deducir otro que nos asegure el cumplimiento de ciertos requisitos. Ese método formal es la teoría de la normalización.

5.2. FORMAS NORMALES DE CODD.

La **teoría de la normalización** trata de evitar las redundancias y las anomalías de actualización, obteniendo tablas más estructuradas que no presenten los problemas que comentábamos anteriormente. Así, en lugar de la tabla del ejemplo anterior, se podría haber diseñado el siguiente esquema relacional:

LIBRO (Cod_libro, título, editorial, año)

AUTOR (Nombre, nacionalidad)

ESCRIBE (Cod_libro*, Nombre*)

Uno de los conceptos fundamentales en la normalización es el de **dependencia funcional**. Una dependencia funcional es una relación entre atributos de una misma tabla. Veamos:

Si x e y son atributos de la tabla T, se dice que **y es funcionalmente dependiente de x** (se denota por: x→ y) si cada valor de x tiene asociado un sólo valor de y. Dicho de otra forma, para un determinado valor de x, siempre se dará el mismo y único valor de y.

x e y pueden constar de uno o varios atributos.

A x se le denomina **determinante**, ya que x determina el valor de y.

Se dice que el atributo y es **completamente dependiente** de x si depende funcionalmente de x y no depende de ningún subconjunto de x.

Por ejemplo, podemos decir que el código de un libro determina el título del mismo:

cod_libro → título

Ejemplo:

EMPLEADO (NIF, Nombre, Dirección, Fecha-Nac)

En la tabla Empleado, los atributos nombre, dirección y fecha de nacimiento dependen funcionalmente del atributo NIF, porque dado un valor específico de NIF existe sólo un valor correspondiente para cada uno. Esto se representa:

NIF → Nombre

NIF → Dirección

NIF → Fecha-Nac

De forma reducida se puede indicar:

NIF → Nombre, Dirección, Fecha-Nac

Sin embargo, dada una dirección no ocurre que sólo haya un empleado con esa dirección, puede que más de un empleado viva en la misma casa, así pues **NIF** no depende de dirección:

~~Dirección → NIF~~

La dependencia funcional es una noción semántica. Si hay o no dependencias funcionales entre atributos no lo determina una serie abstracta de reglas, sino los modelos mentales del usuario y las reglas de negocio de la organización o empresa para la que se desarrolla el sistema de información.

En el proceso de normalización se debe ir comprobando que cada tabla cumple una serie de reglas que se basan en la clave primaria y las dependencias funcionales. Cada regla que se cumple aumenta el grado de normalización. Si una regla no se cumple, la tabla se debe descomponer en varias tablas que sí la cumplan (la tabla original se transforma).

Al ser las dependencias funcionales restricciones que recogen la semántica de nuestro mundo real, deben estar reflejadas en el esquema de nuestra base de datos. En consecuencia, la transformación del esquema origen debe llevarse a cabo sin pérdida de estas dependencias o, lo que es lo mismo, el conjunto de dependencias funcionales de partida debe ser equivalente al conjunto de dependencias funcionales de las tablas resultantes después de las transformaciones.

La normalización se lleva a cabo en una serie pasos. Cada paso corresponde a una forma normal que tiene unas propiedades. Conforme se va avanzando en la normalización, las tablas tienen un formato más estricto (más fuerte) y, por lo tanto, son menos vulnerables a las anomalías de actualización. **El modelo relacional sólo requiere un conjunto de tablas en primera forma normal. Las restantes formas normales son opcionales. Sin embargo, para evitar las anomalías de actualización, es recomendable llegar al menos a la tercera forma normal.**

Otra ventaja de la normalización de base de datos es el consumo de espacio. Una base de datos normalizada ocupa menos espacio en disco que una no normalizada. Hay menos repetición de datos, lo que tiene como consecuencia un mucho menor uso de espacio en disco.

a) PRIMERA FORMA NORMAL.

Una tabla está en primera forma normal, si y sólo si, todos los campos de la misma contienen valores atómicos, es decir, no hay atributos multivaluados o repetitivos. Si se ve la tabla gráficamente, estará en 1FN si tiene un sólo valor en la intersección de cada fila con cada columna.

Si una tabla no está en 1FN, hay que eliminar de ella los campos multivaluados. Una solución consiste en ponerlos en una tabla aparte, eliminándose de la tabla base. La clave principal de la nueva tabla estará compuesta por la clave principal de la tabla base (actuando también como clave ajena) y alguno o algunos de los atributos de la nueva tabla.

Ejemplo:

LIBRO (Código, Título, Autor)

CÓDIGO	**TÍTULO**	**AUTOR**
010101010	DATA MODELS	TSICHIRITZIS LOCHOVSKY
090909090	A GUIDE TO DB2	DATE
010203040	BASES DE DATOS	GARDARIN VALDURIEZ

NO ESTÁ EN 1FN

HAY CAMPOS MULTIVALUADOS

Solución:

LIBRO (Código, Título):

CÓDIGO	**TÍTULO**
010101010	DATA MODELS
090909090	A GUIDE TO DB2
010203040	BASES DE DATOS

AUTOR_LIB (Código_lib*, Nombre_aut)

CÓDIGO_LIB	**NOMBRE_AUT**
010101010	TSICHIRITZIS
010101010	LOCHOVSKY
090909090	DATE
010203040	GARDARIN
010203040	VALDURIEZ

b) SEGUNDA FORMA NORMAL.

Una tabla está en segunda forma normal si, y sólo si, **está en 1FN y, además, cada atributo que no está en la clave primaria es completamente dependiente de la clave primaria.**

Es decir, cada atributo que no está en la clave primaria depende de toda la clave primaria y no de ningún subconjunto de ella. Se pretende garantizar una correcta elección de claves y eliminar redundancias.

La 2FN se aplica a las tablas que tienen claves primarias compuestas por dos o más atributos. Si una tabla está en 1FN y su clave primaria tiene un sólo atributo, entonces también está en 2FN. Las tablas que no están en 2FN pueden sufrir anomalías de actualización.

Para pasar una tabla en 1FN a 2FN se eliminan los atributos que no dependen funcionalmente de forma completa de la clave principal y se ponen en una nueva tabla con una copia de su determinante (los atributos de la clave primaria de los que dependen). La clave principal de la nueva tabla estará compuesta por el determinante, que será clave ajena en la tabla original.

Ejemplo:

PUBLICA (Artículo, Revista, Número, Página, Editorial)

ARTICULO	REVISTA	NUMERO	PAGINA	EDITORIAL
Arritmias	Ciencia Moderna	4	10	Planeta
Colesterol	Ciencia Moderna	4	14	Planeta
Arritmias	Vivir en salud	6	3	Salvat
Listas de espera	Ciencia Moderna	5	11	Planeta

En la tabla anterior refleja en qué número de qué revista se publica un artículo, en qué página comienza y cuál es la editorial.

Tenemos las siguientes dependencias:

Artículo, Revista, Número → Página
Revista → Editorial

La clave primaria está compuesta por (Artículo, Revista, Número). Por tanto, el esquema no está en 2FN al venir la Editorial determinada sólo por la Revista.

Tiene además redundancias como repetir la Editorial por cada Artículo que se publica en una Revista.

Solución :

REVISTA_ED (Revista, Editorial)

REVISTA	EDITORIAL
Vivir en salud	Salvat
Ciencia Moderna	Planeta

PUBLICA (Artículo, Revista*, Número, Página)

ARTICULO	REVISTA	NUMERO	PAGINA
Arritmias	Ciencia Moderna	4	10
Colesterol	Ciencia Moderna	4	14
Arritmias	Vivir en salud	6	3
Listas de espera	Ciencia Moderna	5	11

c) TERCERA FORMA NORMAL.

Una tabla está en tercera forma normal, si y sólo si, **está en 2FN y además, cada atributo que no está en la clave primaria no depende funcionalmente de forma transitiva de la clave primaria**. La dependencia x → z es transitiva si existen las dependencias x → y , y → z, siendo x, y, z atributos o conjuntos de atributos de una misma tabla.

Aunque las tablas en 2FN tienen menos redundancias que las tablas en 1FN, todavía pueden sufrir anomalías frente a las actualizaciones.

Para pasar una tabla de 2FN a 3FN hay que eliminar las dependencias transitivas: se eliminan los atributos que dependen transitivamente de la clave principal y se ponen en una nueva tabla con una copia de su determinante (el atributo o atributos no clave de los que dependen). La clave principal de la nueva tabla estará compuesta por el determinante, que será clave ajena en la tabla original.

Ejemplo:

SOCIO (DNI, Ciudad, País)

DNI	CIUDAD	PAÍS
22334778	Almería	ESPAÑA
23099555	Glasgow	ESCOCIA
22333111	Almería	ESPAÑA
77888999	Edimburgo	ESCOCIA
11222333	Glasgow	ESCOCIA

Presenta las siguientes dependencias funcionales:

DNI → Ciudad
Ciudad → País
DNI → País (dependencia transitiva de la clave primaria)

Dicha tabla no se encuentra en 3FN, ya que País no es conocido por la clave primaria sino que es conocido por la ciudad que no es la PK..

Solución:

CIUDAD_P (Ciudad, País)

CIUDAD	PAÍS
Almería	ESPAÑA
Glasgow	ESCOCIA
Edimburgo	ESCOCIA

SOCIO (DNI, Ciudad*)

DNI	CIUDAD
22334778	Almería
23099555	Glasgow
22333111	Almería
77888999	Edimburgo
11222333	Glasgow

Otro Ejemplo:

EMPLEADOS(DNI, Cod_Dpto, Nombre_Dpto)

DNI	Cod_Dpto	Nombre_Dpto
22334778	C01	Contabilidad
23099555	M05	Marketing
22333111	I02	Investigación
77888999	C01	Contabilidad
11222333	M05	Marketing

Presenta las siguientes dependencias funcionales:

DNI → Cod_Dpto
Cod_Dpto → Nombre_Dpto
DNI → Nombre_Dpto (dependencia transitiva de la clave primaria)

Dicha tabla no se encuentra en 3FN, ya que Nombre_Dpto es un atributo no es conocido por la PK sino que es conocido por Cod_Dpto que no es PK.

Solución:

DEPARTAMENTOS (Cod_Dpto, Nombre_Dpto)

Cod_Dpto	**Nombre_Dpto**
C01	Contabilidad
M05	Marketing
I02	Investigación

EMPLEADOS (DNI, Cod_Dpto*)

DNI	**Cod_Dpto**
22334778	C01
23099555	M05
22333111	I02
77888999	C01
11222333	M05

Y otro ejemplo más:

COCHES(Matrícula, Modelo, Marca, Color)

Matrícula	**Modelo**	**Marca**	**Color**
1234BCC	Ibiza	Seat	Azul
4321CFR	Focus	Ford	Blanco
2314BRF	Laguna	Renault	Rojo
5122DCC	Ibiza	Seat	Gris
7763CGT	Avensis	Toyota	Gris

Presenta entre otras, las siguientes dependencias funcionales:

Matrícula → Modelo

Modelo → Marca

Matrícula → Marca (dependencia transitiva de la clave primaria)

Dicha tabla no se encuentra en 3FN, ya que Marca es un atributo que se conoce por Modelo que no esta en la PK y entonces depende transitivamente de la PK.

Solución:

MODELOS(Modelo, Marca)

Modelo	**Marca**
Ibiza	Seat
Focus	Ford
Laguna	Renault
Avensis	Toyota

COCHES(Matrícula, Modelo*, Color)

Matrícula	**Modelo**	**Color**
1234BCC	Ibiza	Azul
4321CFR	Focus	Blanco
2314BRF	Laguna	Rojo
5122DCC	Ibiza	Gris
7763CGT	Avensis	Gris