Principios sobre Bases de Datos Relacionales

Autor: Jorge Sánchez (www.jorgesanchez.net) año 2004
e-mail: mailto:info@jorgesanchez.net

índice

modelos lógicos de datos

esquema canónico

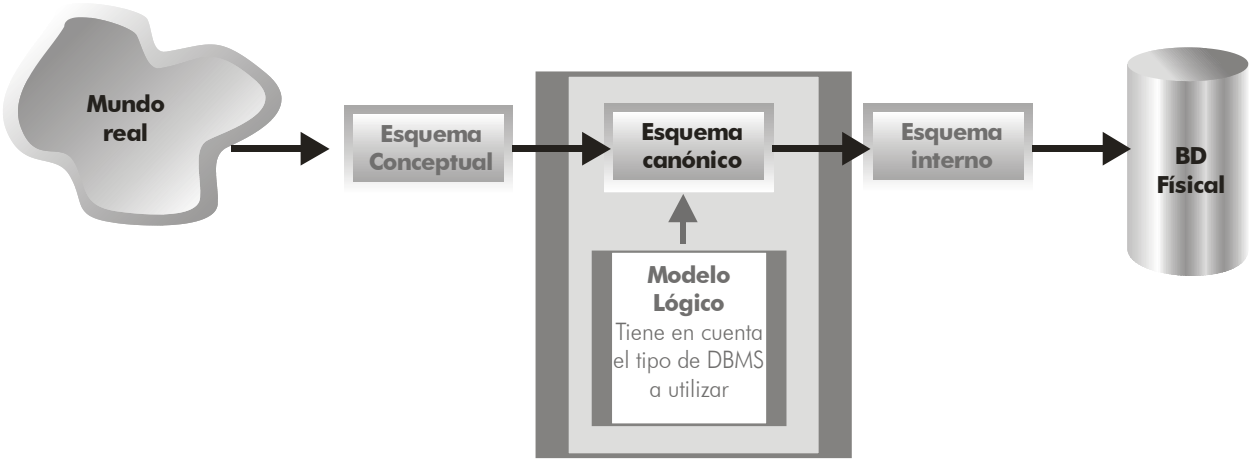


Ilustración 1, Posición de esquema canónico dentro de los esquema de creación de una base de datos

El esquema canónico o lógico global, es un esquema que presenta de forma conceptual la estructura de una base de datos. Es un esquema que depende del tipo de DBMS que vayamos a utilizar.

Se crea a partir del modelo conceptual (véase el documento *Diseño Conceptual de Bases de Datos* en **www.jorgesanchez.net/bd**). Y serviría para cualquier base de datos comercial del tipo elegido en el esquema (hay esquemas relacionales, en red, jerárquicos,...)

tipos de base de datos

jerárquicas

En ellas se organiza la información se organiza con un jerarquía en la que la relación entre las entidades de este modelo siempre es del tipo **padre / hijo**. De esta forma hay una serie de nodos que contendrán atributos y que se relacionarán con nodos hijos de forma que puede haber más de un hijo para el mismo padre (pero un hijo sólo tiene un padre).

Las entidades de este modelo se llaman **segmentos** y los atributos **campos**. La forma visual de este modelo es de árbol invertido, en la parte superior están los padres y en la inferior los hijos.

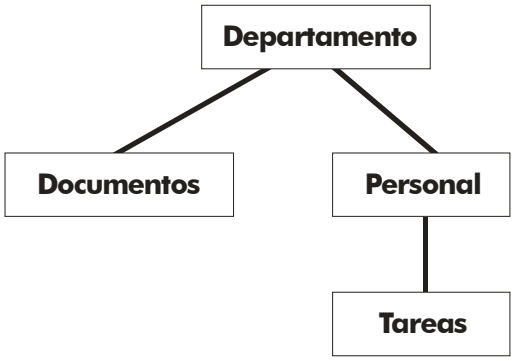


Ilustración 2, Ejemplo de esquema jerárquico

en red

Se trata de un modelo que se utilizó durante mucho tiempo. Organiza la información en **registros** y **enlaces**. Los registros representan las entidades del modelo entidad / relación. En los registros se almacenan los datos utilizando **atributos**. Los enlaces permiten relacionar los registros de la base de datos.

El modelo en red más aceptado es el llamado **codasyl**, que durante mucho tiempo se ha convertido en un estándar.

Las bases de datos en red son parecidas a las jerárquicas sólo que en ellas puede haber más de un padre. En este modelo se pueden representar perfectamente relaciones varios a varios. Pero su dificultad de manejo y complejidad hace que se estén abandonando completamente.

relacionales

Los datos se muestran en forma de tablas y relaciones. Este es el modelo que se comenta en el presente documento. De hecho es el claramente más popular.

orientadas a objetos

Desde la aparición de la programación orientada a objetos (POO u OOP) se empezó a pensar en bases de datos adaptadas a estos lenguajes. En estos lenguajes los datos y los procedimientos se almacenan juntos. Esta es la idea de las bases de datos orientadas a objetos.

A través de esta idea se intenta que estas bases de datos consiguen arreglar las limitaciones de las relacionales. Por ejemplo el problema de la herencia, tipos definidos por el usuario, disparadores almacenables en la base de datos, soporte multimedia...

Se supone que son las bases de datos de tercera generación (la primera fue las bases de datos en red y la segunda las relacionales), lo que significa que el futuro parece estar a favor de estas bases de datos. Pero siguen sin reemplazar a las relacionales (aunque cada vez hay más).

Su modelo conceptual se suele diseñar en UML y el lógico en ODMG 3.0

objeto relacionales

Tratan de ser un híbrido entre el modelo relacional y el orientado a objetos. El problema de las bases de datos orientadas a objetos es que requieren reinvertir de nuevo para convertir las bases de datos. En las bases de datos objeto relacionales se intenta conseguir una compatibilidad relacional dando la posibilidad de integrar mejoras de la orientación a objetos.

Estas bases de datos se basan en el estándar SQL 99 que dictó las normas para estas bases de datos. En ese estándar se añade a las bases relacionales la posibilidad de almacenar procedimientos de usuario, triggers, tipos definidos por el usuario, consultas recursivas, bases de datos OLAP, tipos LOB,...

Las últimas versiones de la mayoría de las grandes bases de datos relacionales (Oracle, SQL Server, Informix, ...) son objeto relacionales.

modelo relacional

introducción

Edgar Frank Codd a finales definió las bases del modelo relacional a finales de los 60. Trabajaba para IBM empresa que tardó un poco en implementar sus bases. Pocos años después el modelo se empezó a implementar cada vez más, hasta ser el modelo de bases de datos más popular.

En las bases de Codd se definían los objetivos de este modelo:

- **Independencia física**. La forma de almacenar los datos, no debe influir en su manipulación lógica
- **Independencia lógica**. Las aplicaciones que utilizan la base de datos no deben ser modificadas por que se modifiquen elementos de la base de datos.
- **Flexibilidad**. La base de datos ofrece fácilmente distintas vistas en función de los usuarios y aplicaciones.
- **Uniformidad**. Las estructuras lógicas siempre tienen una única forma conceptual (las tablas)
- **Sencillez**.

En 1978, IBM desarrolla el lenguaje QBE. Que aproximaba la idea relacional a sus ficheros VSAM. En 1979 Oracle se convierte en el primer producto comercial DBMS relacional (RDBMS). En 1980 aparece Ingres que utilizaba el lenguaje **Quel** que implementaba el cálculo relacional.

evolución del modelo relacional

Año	Hecho
1970	Codd publica las bases del modelo relacional
1971-72	Primeros desarrollos teóricos
1973-78	Primeros prototipos
1978	Aparece el lenguaje QBE
1979	Aparece Oracle
1980	Aparece Ingres
1981	Aparece SQL
1982	Aparece DB2
1986	ANSI normaliza el SQL (SQL/ANSI)
1987	SQL de ISO
1990	Versión dos del modelo relacional (RM/V2)
1992	SQL 92
1998	SQL 3

tablas

Las bases de datos relacionales se basan en el uso de tablas (también se las llama **relaciones**). Las tablas se representan gráficamente como una estructura rectangular formada por filas y columnas. Cada columna almacena información sobre una propiedad determinada de la tabla (se le llama también **atributo**), nombre, dni, apellidos, edad,.... Cada fila posee una ocurrencia o ejemplar de la instancia o relación representada por la tabla (a las filas se las llama también **tuplas**).

NOMBRE

atributo 1	**atributo 2**	**atributo 3**	**....**	**atributo n**	
valor 1,1	valor 1,2	valor 1,3	....	valor 1,n	**← tupla 1**
valor 2,1	valor 2,2	valor 2,3	....	valor 2,n	**← tupla 2**
.....	**.....**	.....	**....**	.....	....
valor m,1	valor m,2	valor m,3	....	valor m,n	**← tupla m**

Ilustración 3, Representación de una tabla en el modelo relacional

terminología relacional

- **Tupla.** Cada fila de la tabla (cada ejemplar que la tabla representa)
- **Atributo.** Cada columna de la tabla
- **Grado.** Número de atributos de la tabla
- **Cardinalidad.** Número de tuplas de una tabla
- **Dominio**. Conjunto válido de valores representables por un atributo.

tipos de tablas

- **Persistentes.** Sólo pueden ser borradas por los usuarios:
  - **Base**. Independientes, se crean indicando su estructura y sus ejemplares.
  - **Vistas**. Son tablas que sólo almacenan una definición de consulta, resultado de la cual se produce una tabla cuyos datos proceden de las bases o de otras vistas e instantáneas. Si los datos de las tablas base cambian, los de la vista que utiliza esos datos también cambia.
  - **Instantáneas**. Son vistas (creadas de la misma forma) que sí que almacenan los datos que muestra, además de la consulta que dio lugar a esa vista. Sólo modifican su resultado (actualizan los datos) siendo refrescadas por el sistema cada cierto tiempo.
- **Temporales**. Son tablas que se eliminan automáticamente por el sistema. Pueden ser de cualquiera de los tipos anterior

dominios

Los dominios suponen una gran mejora en este modelo ya que permiten especificar los posibles valores válidos para un atributo. Cada dominio incorpora su nombre y una definición del mismo. Ejemplos de dominio:

- **Dirección:** 50 caracteres
- **Nacionalidad:** Español, Francés, Italiano,...

Los dominios pueden ser también compuestos a partir de otros (año, mes y día = fecha)

claves

clave candidata

Conjunto de atributos de una tabla que identifican unívocamente cada tupla de la tabla.

clave primaria

Clave candidata que se escoge como identificador de las tuplas.

clave alternativa

Cualquier clave candidata que no sea primaria

clave externa o secundaria

Atributo de una tabla relacionado con una clave de otra tabla.

nulos

Los valores nulos indican contenidos de atributos que no tienen ningún valor. En claves secundarias indican que el registro actual no está relacionado con ninguno. En otros atributos indica que no se puede rellenar ese valor por la razón que sea.

Las bases de datos relacionales admiten utilizar ese valor en todo tipo de operaciones. Eso significa definir un tercer valor en la lógica. Además de el valor verdadero o falso, existe el valor para los nulos.

La razón de este tercer valor ambiguo es que comparar dos atributos con valor nulo, no puede resultar ni verdadero, ni falso. De hecho necesitamos definir la lógica con este valor:

- **verdadero Y (AND) nulo** da como resultado, **nulo**
- **falso Y (AND) nulo** da como resultado, **falso**
- **verdadero O (OR) nulo** da como resultado, **verdadero**
- **falso O nulo** da como resultado **nulo**
- **la negación de nulo,** da como resultado **nulo**

restricciones

Se trata de unas condiciones de obligado cumplimiento por los datos de la base de datos. Las hay de varios tipos.

inherentes

Son aquellas que no son determinadas por los usuarios, sino que son definidas por el hecho de que la base de datos sea relacional. Por ejemplo:

- **No puede haber dos tuplas iguales**
- **El orden de las tuplas no importa**
- **El orden de los atributos no importa**
- **Cada atributo sólo puede tomar un valor en el dominio en el que está inscrito**

semánticas

El modelo relacional permite a los usuario incorporar restricciones personales a los datos. Las principales son:

- **Clave primaria**. Hace que los atributos marcados como clave primaria no puedan repetir valores.
- **Unicidad**. Impide que los valores de los atributos marcados de esa forma, puedan repetirse.
- **Obligatoriedad**. Prohíbe que el atributo marcado de esta forma no tenga ningún valor
- **Integridad referencial**. Prohíbe colocar valores en una clave externa que no estén reflejados en la tabla donde ese atributo es clave primaria.
- **Regla de validación.** Condición que debe de cumplir un dato concreto para que sea actualizado.

las 12 reglas de Codd

Preocupado por los productos que decían ser sistemas gestores de bases de datos relacionales (RDBMS) sin serlo, Codd publica las 12 reglas que debe cumplir todo DBMS para ser considerado relacional. Estas reglas en la práctica las cumplen pocos sistemas relacionales. Las reglas son:

1> **Información**. Toda la información de la base de datos debe estar representada explícitamente en el esquema lógico. Es decir, todos los datos están en las tablas.

2> **Acceso garantizado**. Todo dato es accesible sabiendo el valor de su clave y el nombre de la columna o atributo que contiene el dato.

3> **Tratamiento sistemático de los valores nulos**. El DBMS debe permitir el tratamiento adecuado de estos valores

4> **Catálogo en línea basado en el modelo relacional.** Los metadatos deben de ser accesibles usando un esquema relacional.

5> **Sublenguaje de datos completo**. Al menos debe de existir un lenguaje que permita el manejo completo de la base de datos. Este lenguaje, por lo tanto, debe permitir realizar cualquier operación.

6> **Actualización de vistas.** El DBMS debe encargarse de que las vistas muestren la última información

7> **Inserciones, modificaciones y eliminaciones de dato nivel.** Cualquier operación de modificación debe actuar sobre conjuntos de filas, nunca deben actuar registro a registro.

8> **Independencia física.** Los datos deben de ser accesibles desde la lógica de la base de datos aún cuando se modifique el almacenamiento.

9> **Independencia lógica.** Los programas no deben verse afectados por cambios en las tablas

10> **Independencia de integridad.** Las reglas de integridad deben almacenarse en la base de datos (en el diccionario de datos), no en los programas de aplicación.

11> **Independencia de la distribución.** El sublenguaje de datos debe permitir que sus instrucciones funciones igualmente en una base de datos distribuida que en una que no lo es.

12> **No subversión**. Si el DBMS posee un lenguaje que permite el recorrido registro a registro, éste no puede utilizarse para incumplir las reglas relacionales.

paso del esquema ER al modelo relacional

transformaciones de entidades fuertes

En principio las entidades fuertes del modelo Entidad Relación son transformados al modelo relacional siguiendo estas instrucciones:

- **Entidades.** Las entidades pasan a ser tablas
- **Atributos**. Los atributos pasan a ser columnas.
- **Identificadores principales**. Pasan a ser claves primarias
- **Identificadores candidatos**. Pasan a ser claves candidatas.

Esto hace que la transformación sea de esta forma:

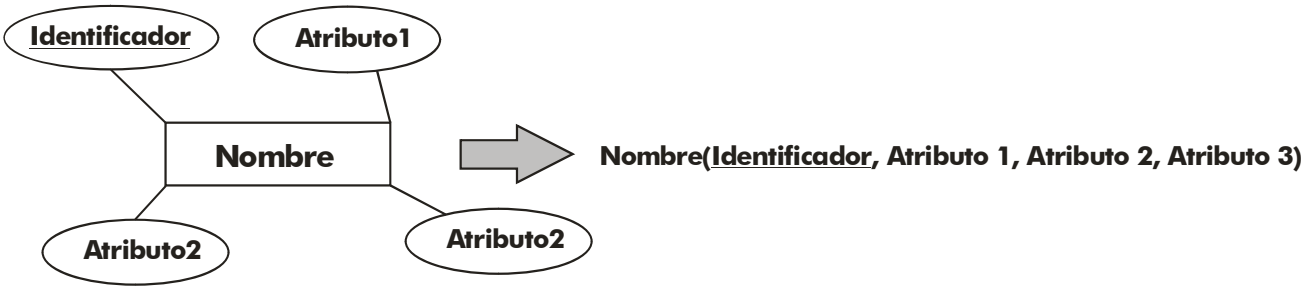


***Ilustración 4*,Transformación de una entidad fuerte al esquema relacional**

transformación de relaciones

La idea inicial es transformar a cada relación en una tabla en el modelo relacional. Pero hay que distinguir según el tipo de relación.

relaciones varios a varios

En las relaciones varios a varios, la relación se transforma en una tabla cuyos atributos son: los atributos de la relación y las claves de las entidades relacionadas (que pasarán a ser claves externas). La clave de la tabla la forman todas las claves externas:

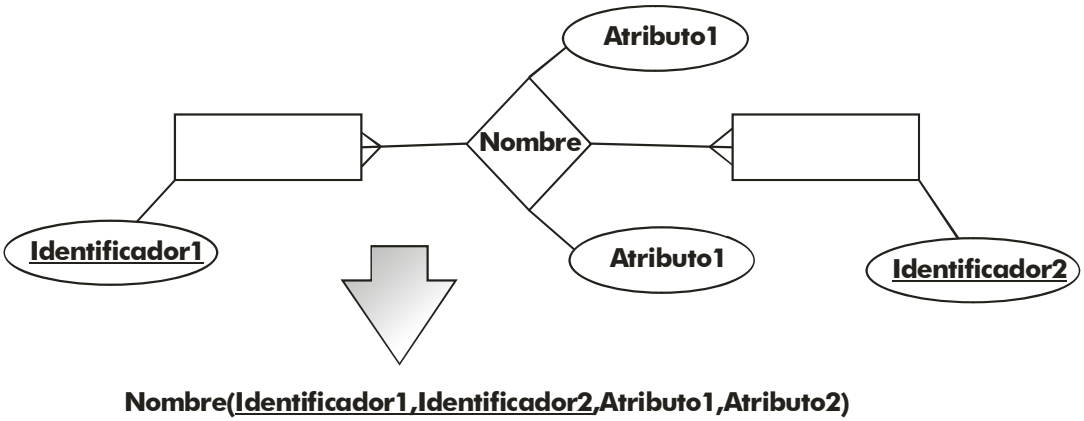
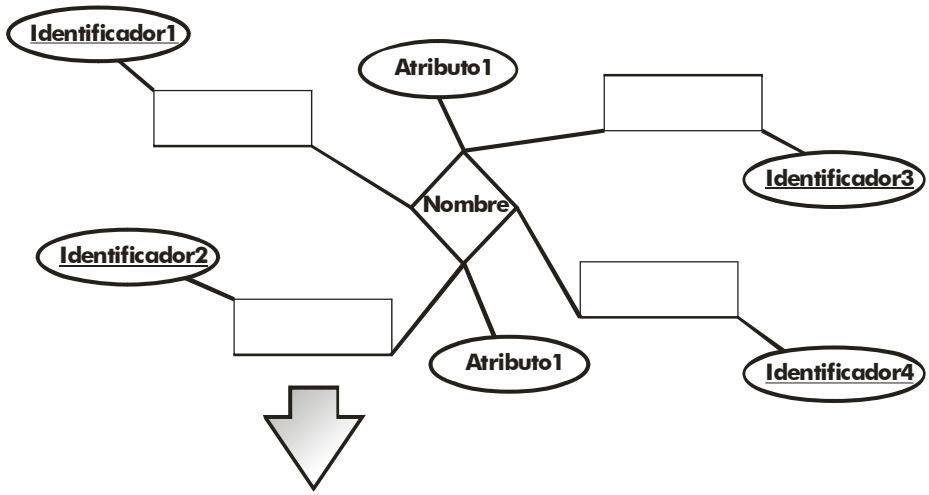


Ilustración 5, Transformación de una relación varios a varios

relaciones de orden *n*

Las relaciones ternarias, cuaternarias y *n-arias* que unen más de dos relaciones se transforman en una tabla que contiene los atributos de la relación más los identificadores de las entidades relacionadas. La clave la forman todas las claves externas:

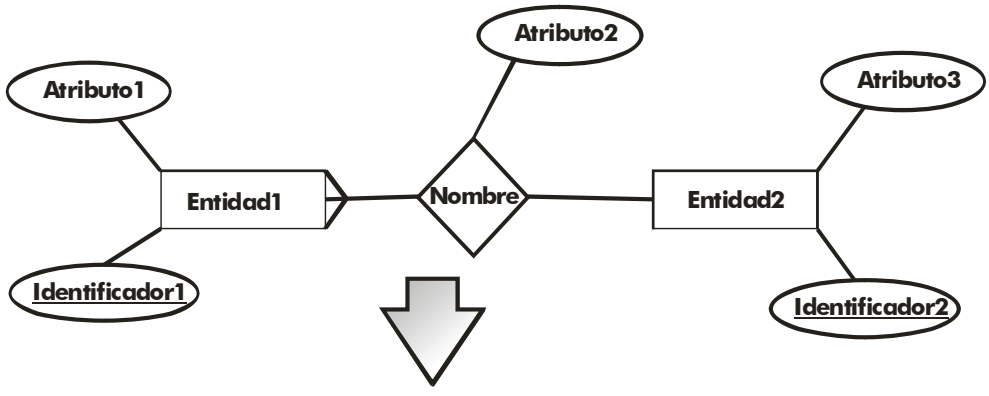


Nombre(Identificador1,Identificador2,,Identificador3,Identificador4,Atributo1,Atributo2)

Ilustración 6, Transformación en el modelo relacional de una entidad n-aria

relaciones uno a varios y uno a uno

Las relaciones binarios de tipo uno a varios no requieren ser transformadas en una tabla en el modelo relacional. En su lugar la tabla del lado *varios* (**tabla** relacionada) incluye como clave externa[1] el identificador de la entidad del lado *uno* (**tabla principal**):



Entidad2(Identificador2,Atributo3)
Entidad1(Identificador1,Atributo1,Identificador2,Atributo2)

Ilustración 7, Transformación de una relación uno a varios

Así en el dibujo, el *identificador2* en la tabla E*ntidad1* pasa a ser una clave externa. En el caso de que el número mínimo de la relación sea de *cero* (puede haber ejemplares de la entidad uno sin relacionar), se deberá permitir valores nulos en la clave externa

[1] Clave externa, clave ajena, clave foránea, clave secundaria y *foreign key* son sinónimos

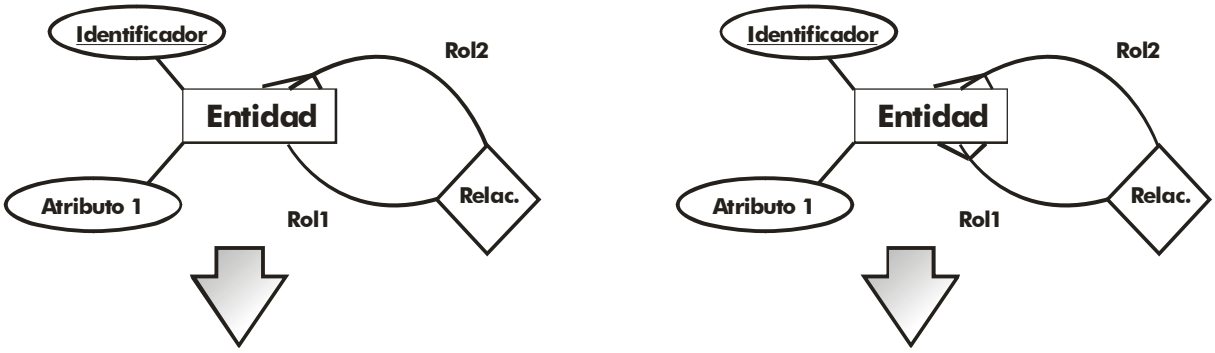
identificador2. En otro caso no se podrán permitir (ya que siempre habrá un valor relacionado).

En el caso de las relaciones uno a uno, ocurre lo mismo: la relación no se convierte en tabla, sino que se coloca en una de las tablas (en principio daría igual cuál) el identificador de la entidad relacionada como clave externa.

En el caso de que una entidad participe opcionalmente en la relación, entonces es el identificador de ésta el que se colocará como clave externa en la tabla que representa a la otra entidad.

relaciones recursivas

Las relaciones recursivas se tratan de la misma forma que las otras, sólo que un mismo atributo puede figurar dos veces en una tabla como resultado de la transformación:



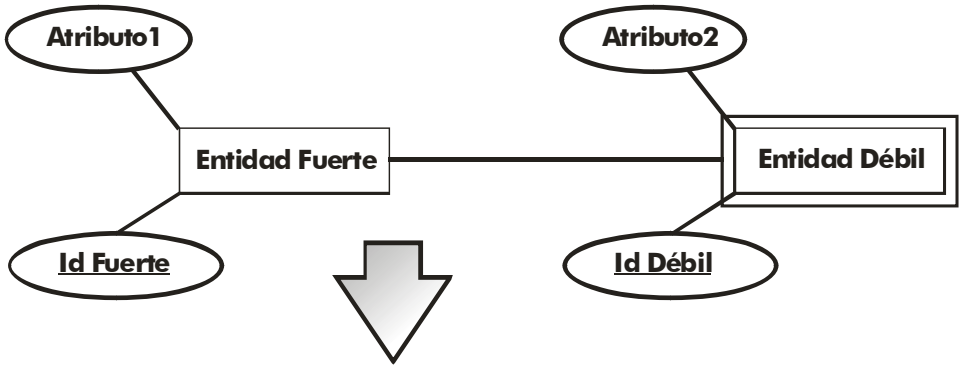
Entidad(Identificador,Atributo1,Identificador Rol 1)

Entidad(Identificador,Atributo1)
Relac(Identificador Rol 1, Identificador Rol 2,Atributo1)

Ilustración 8, Transformación de relaciones recursivas en el modelo relacional

entidades débiles

Toda entidad débil incorpora una relación implícita con una entidad fuerte. Esta relación no necesita incorporarse como tabla en el modelo relacional. Sí se necesita incorporar la clave de la entidad fuerte como clave externa en la entidad débil. Es más, normalmente esa clave externa forma parte de la clave principal de la tabla que representa a la entidad débil. El proceso es:



Entidad Fuerte(Id Fuerte, Atributo 1)
Entidad1(Id Débil, Id Fuerte, Atributo2)

Ilustración 9, transformación de una entidad débil en el modelo relacional

En ocasiones el identificador de la entidad débil es suficiente para identificar los ejemplares de dicha entidad, entonces ese identificador quedaría como clave principal, pero el identificador de la entidad fuerte seguiría figurando como clave externa en la entidad débil.

generalizaciones y especificaciones

Las generalizaciones y/o especificaciones se convierten al modelo relacional de esta forma:

1> Las subentidades pasan a ser tablas.

2> Si la clave de la superentidad es distinta de las subentidades, entonces se coloca el identificador de la superentidad en cada subentidad como clave externa:

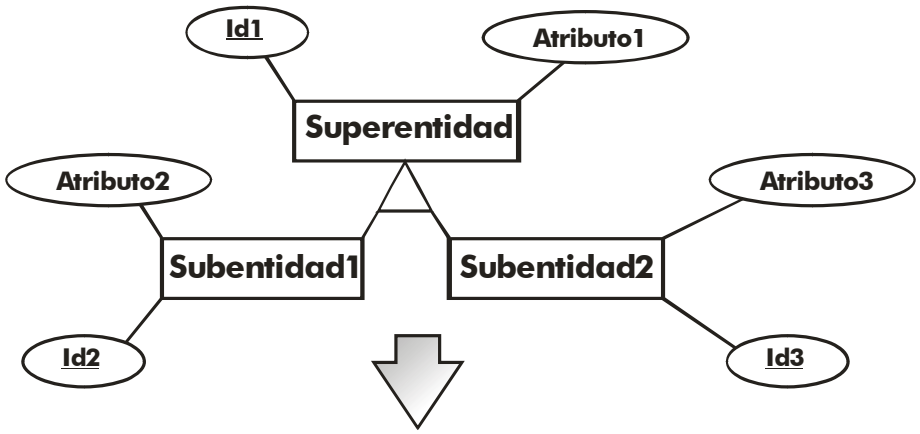


Ilustración 10, Proceso de transformación de relaciones ISA con clave propia

3> Si la clave es la misma, entonces todas las entidades tendrán la misma columna como identificador:

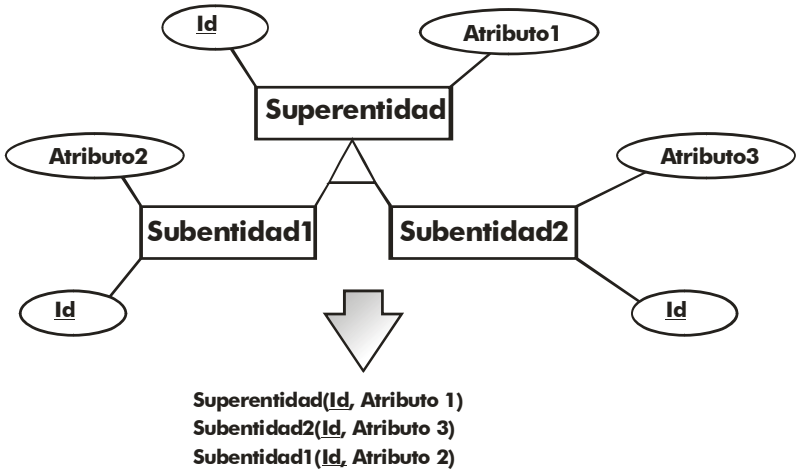


Ilustración 11, Proceso de transformación de relaciones ISA en el modelo relacional si tienen la misma clave

4> La superentidad debe generar una tabla sólo en el caso de que haya posibilidad de que exista un ejemplar de dicha entidad que no sea ejemplar de las subentidades. De otro modo basta con generar las tablas de las subentidades e incluir los atributos de la entidad superior:

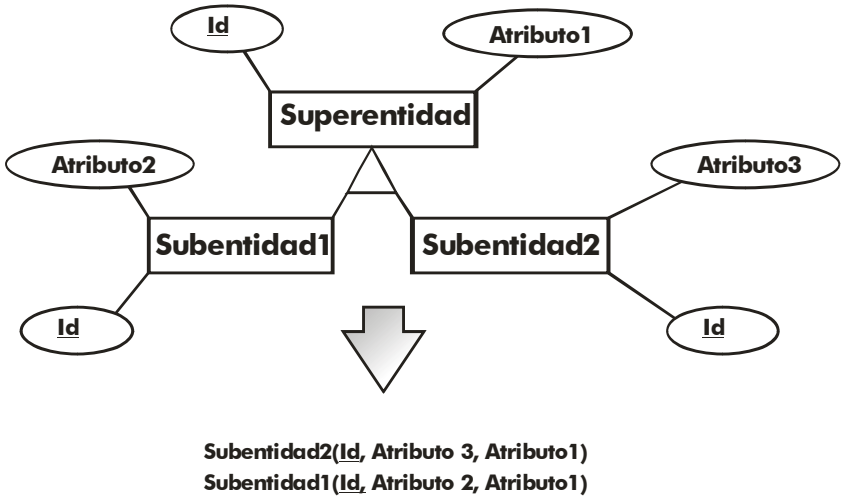


Ilustración 12, Paso de relaciones ISA al modelo relacional cuando toda superentidad figura como subentidad

normalización del esquema relacional

problemas del esquema relacional

Una vez obtenido el esquema relacional resultantes del modelo entidad relación que representaba la base de datos, normalmente tendremos una buena base de datos. Pero otras veces, debido a fallos en el diseño o a problemas indetectables en esta fase del diseño, tendremos un esquema que puede producir una base de datos que incorpore estos problemas:

- **Redundancia.** Se llama así a los datos que se repiten continua e innecesariamente por las tablas de las bases de datos.
- **Ambigüedades.** Datos que no clarifican suficientemente el registro al que representan.
- **Pérdida de restricciones de integridad.**
- **Anomalías en operaciones de modificación de datos.** El hecho de que al insertar un solo elemento haya que repetir tuplas en una tabla para variar unos pocos datos. O que eliminar un elemento suponga eliminar varias tuplas.

El principio fundamental reside en que las tablas deben referirse a objetos o situaciones muy concretas. Lo que ocurre es que conceptualmente es difícil obtener ese problema.

La solución suele ser dividir la tabla con problemas en otras tablas más adecuadas.

formas normales

Las formas normales se corresponde a una teoría de normalización iniciada por el propio Codd y continuada por otros autores (entre los que destacan Boyce y Fagin). Codd definió en 1970 la primera forma normal, desde ese momento aparecieron la segunda, tercera, la Boyce-Codd, la cuarta y la quinta forma normal.

Una tabla puede encontrarse en primera forma normal y no en segunda forma normal, pero no al contrario. Es decir los números altos de formas normales son más restrictivos (la quinta forma normal cumple todas las anteriores).

La teoría de formas normales es una teoría absolutamente matemática, pero en el presente manual se describen de forma intuitiva.

primera forma normal (1FN)

Una tabla se encuentra en primera forma normal si impide que un atributo de una tupla pueda tomar más de un valor. La tabla:

TRABAJADOR		
DNI	**Nombre**	**Departamento**
12121212A	Andrés	Mantenimiento
12345345G	Andrea	Dirección Gestión

Visualmente es un tabla, pero no una tabla relacional (lo que en terminología de bases de datos relacionales se llama **relación**). No cumple la primera forma normal. Lo cumpliría si:

TRABAJADOR		
DNI	**Nombre**	**Departamento**
12121212A	Andrés	Mantenimiento
12345345G	Andrea	Dirección
12345345G	Andrea	Gestión

Esa tabla sí esta en primera forma normal.

dependencias funcionales

Se dice que un conjunto de atributos (Y) depende funcionalmente de otro conjunto de atributos (X) si para cada valor de X hay un único valor posible para Y. Simbólicamente se denota por X→Y.

Por ejemplo el nombre de una persona depende funcionalmente del DNI, para un DNI concreto sólo hay un nombre posible. En la tabla ejemplo anterior, el departamento no tiene dependencia funcional, ya que para un mismo DNO puede haber más de un departamento posible.

Al conjunto X del que depende funcionalmente el conjunto Y se le llama **determinante**. Al conjunto Y se le llama **implicado.**

dependencia funcional completa

Un conjunto de atributos (Y) tiene una dependencia funcional completa sobre otro conjunto de atributos (X) si Y tiene dependencia funcional de X y además no se puede obtener de X un conjunto de atributos más pequeño que consiga una dependencia funcional de Y.

Por ejemplo en una tabla de clientes, el conjunto de atributos formado por el **nombre** y el **dni** producen una dependencia funcional sobre el atributo **apellidos**. Pero no es plena ya que el **dni** sólo también produce una dependencia funcional sobre **apellidos.** El dni sí produce una dependencia funcional completa sobre el campo apellidos.

Una dependencia funcional completa se denota como X⇒Y

dependencia funcional elemental

Se produce cuando X e Y forman una dependencia funcional completa y además Y es un único atributo.

dependencia funcional transitiva

Es más compleja de explicar, pero tiene también utilidad. Se produce cuando tenemos tres conjuntos de atributos X, Y y Z. Y depende funcionalmente de X (X→Y), Z depende funcionalmente de Y (Y→Z). Además X no depende funcionalmente de Y. Entonces ocurre que X produce una dependencia funcional transitiva sobre Z. Esto se denota como:

(X—→Z)

Por ejemplo si X es el atributo **Número de Clase** de un instituto, e Y es el atributo **Código Tutor**. Entonces X→Y (el tutor depende funcionalmente del número de clase). Si Z representa el **Código del departamento**, entonces Y→Z (el código del departamento depende funcionalmente del código tutor, cada tutor sólo puede estar en un

departamento). Como no ocurre que Y→X (el código de la clase no depende funcionalmente del código tutor, un código tutor se puede corresponder con varios códigos de clase). Entonces X— →Z (el código del departamento depende transitivamente del código de la clase).

segunda forma normal (2FN)

Ocurre si una tabla está en primera forma normal y además cada atributo que no sea clave, depende de forma funcional completa respecto de cualquiera de las claves. Toda la clave principal debe hacer dependientes al resto de atributos, si hay atributos que depende sólo de parte de la clave, entonces esa parte de la clave y esos atributos formarán otra tabla. Ejemplo:

ALUMNOS				
DNI	**Cod Curso**	**Nombre**	**Apellido1**	**Nota**
12121219A	34	Pedro	Valiente	9
12121219A	25	Pedro	Valiente	8
3457775G	34	Ana	Fernández	6
5674378J	25	Sara	Crespo	7
5674378J	34	Sara	Crespo	6

Suponiendo que el DNI y el número de curso formen una clave principal para esta tabla, sólo la nota tiene dependencia funcional completa. El nombre y los apellidos dependen de forma completa del DNI. La tabla no es 2FN, para arreglarlo:

ALUMNOS		
DNI	**Nombre**	**Apellido1**
12121219A	Pedro	Valiente
3457775G	Ana	Fernández
5674378J	Sara	Crespo

ASISTENCIA		
DNI	**Cod Curso**	**Nota**
12121219A	34	9
12121219A	25	8
3457775G	34	6
5674378J	25	7
5674378J	34	6

tercera forma normal (3FN)

Ocurre cuando una tabla está en 2FN y además ningún atributo que no sea clave depende transitivamente de las claves de la tabla. Es decir no ocurre cuando algún atributo depende funcionalmente de atributos que no son clave.

Ejemplo:

ALUMNOS				
DNI	**Nombre**	**Apellido1**	**Cod Provincia**	**Provincia**
12121349A	Salvador	Velasco	34	Palencia
12121219A	Pedro	Valiente	34	Palencia
3457775G	Ana	Fernández	47	Valladolid
5674378J	Sara	Crespo	47	Valladolid
3456858S	Marina	Serrat	08	Barcelona

La Provincia depende funcionalmente del código de provincia, lo que hace que no esté en 3FN. El arreglo sería:

ALUMNOS			
DNI	**Nombre**	**Apellido1**	**Cod Provincia**
12121349A	Salvador	Velasco	34
12121219A	Pedro	Valiente	34
3457775G	Ana	Fernández	47
5674378J	Sara	Crespo	47
3456858S	Marina	Serrat	08

PROVINCIA	
Cod Provincia	**Provincia**
34	Palencia
47	Valladolid
08	Barcelona

forma normal de Boyce-Codd (FNBC o BCFN)

Ocurre si una tabla está en tercera forma normal y además todo determinante es una clave candidata. Ejemplo:

TUTORÍAS		
DNI	**Asignatura**	**Tutor**
12121219A	Lenguaje	Eva
12121219A	Matemáticas	Andrés
3457775G	Lenguaje	Eva
5674378J	Matemáticas	Guillermo
5674378J	Lenguaje	Julia
5634823H	Matemáticas	Guillermo

Esa tabla está en tercera forma normal (no hay dependencias transitivas), pero no en forma de Boyce - Codd, ya que (DNI, Asignatura) →Tutor y Tutor→Asignatura. En este caso la redundancia ocurre por mala selección de clave. La redundancia de la asignatura es completamente evitable. La solución sería:

TUTORÍAS	
DNI	**Tutor**
12121219A	Eva
12121219A	Andrés
3457775G	Eva
5674378J	Guillermo
5674378J	Julia
5634823H	Guillermo

ASIGNATURASTUTOR	
Asignatura	**Tutor**
Lenguaje	Eva
Matemáticas	Andrés
Matemáticas	Guillermo
Lenguaje	Julia

En las formas de Boyce-Codd hay que tener cuidado al descomponer ya que se podría perder información por una mala descomposición

dependencia multivaluada

Para el resto de formas normales (las diseñadas por Fagin, mucho más complejas), es importante definir este tipo de dependencia, que es distinta de las funcionales. Si las funcionales eran la base de la segunda y tercera forma normal (y de la de Boyce-Codd), éstas son la base de la cuarta forma normal.

Una dependencia multivaluada de una tabla con atributos X, Y, Z de X sobre Z (es decir X->>Z) ocurre cuando los posibles valores de Y sobre cualquier par de valores X y Z dependen sólo del valor de X y son independientes de Z.

Ejemplo:

Nº Curso	Profesor	Material
17	Eva	1
17	Eva	2
17	Julia	1
17	Julia	2
25	Eva	1
25	Eva	2
25	Eva	3

La tabla cursos, profesores y materiales del curso. La tabla está en FNBC ya que no hay dependencias transitivas y todos los atributos son clave sin dependencia funcional hacia ellos. Sin embargo hay redundancia. Los materiales se van a repetir para cualquier profesor dando cualquier curso, ya que los profesores van a utilizar todos los materiales del curso (de no ser así no habría ninguna redundancia).

Los materiales del curso dependen del curso y no del profesor en una dependencia multivaluada. Para el par Nº de curso y profeso podemos saber los materiales, pero por el curso y no por el profesor.

cuarta forma normal (4FN)

Ocurre esta forma normal cuando una tabla está en forma normal de Boyce Codd y toda dependencia multivaluada es una dependencia funcional. Para la tabla anterior la solución serían dos tablas:

Nº Curso	**Material**
17	1
17	2
25	1
25	2
25	3

Nº Curso	**Profesor**
17	Eva
17	Julia
25	Eva

Un teorema de Fagin indica cuando hay tres pares de conjuntos de atributos X, Y y Z si ocurre X->>Y|Z (Y y Z tienen dependencia multivaluada sobre X), entonces las tablas X,Y y X,Z reproducen sin perder información lo que poseía la tabla original. Este teorema marca la forma de dividir las tablas hacia una 4FN

quinta forma normal (5FN)

Es la más compleja y polémica de todas. Polémica pues no está claro en muchas ocasiones que sea una solución mejor que el no llegar a este nivel de normalización. Fue definida también por Fagin.

Es raro encontrarse este tipo de problemas cuando la normalización llega a 4FN. Se deben a restricciones muy concretas. Ejemplo:

Proveedor	**Material**	**Proyecto**
1	1	2
1	2	1
2	1	1
1	1	1

Indican códigos de material suministrado por un proveedor y utilizado en un determinado proyecto.

Si ocurre una restricción especial como por ejemplo: Cuando un proveedor nos ha suministrado alguna vez un determinado material, si ese material aparece en otro proyecto, haremos que el proveedor nos suministre también ese material para ese proyecto.

Eso ocurre en los datos como el proveedor número 1 nos suministró el material número 1 para el proyecto 2 y en el proyecto 1 utilizamos el material 1, aparecerá la tupla proveedor 1, material 1 y proyecto 1.

La dependencia que produce esta restricción es lejana y se la llama de reunión. Para esa restricción esta división en tablas sería válida:

Proveedor	**Material**
1	1
1	2
2	1

Material	**Proyecto**
1	2
2	1
1	1

Esa descomposición no pierde valores en este caso, sabiendo que si el proveedor nos suministra un material podremos relacionarle con todos los proyectos que utilizan ese material.

Resumiendo, una tabla no está en quinta forma normal si hay una descomposición de esa tabla que muestre la misma información que la original.

apéndice: términos técnicos

1FN Abreviatura de *Primera Forma Normal*. Normalización estándar de las tablas relacionales.

2FN Abreviatura de *Segunda Forma Normal*. Normalización estándar de las tablas relacionales.

3FN Abreviatura de *Tercera Forma Normal*. Normalización estándar de las tablas relacionales.

4FN Abreviatura de *Cuarta Forma Normal*. Normalización estándar de las tablas relacionales.

5FN Abreviatura de *Quinta Forma Normal*. Normalización estándar de las tablas relacionales.

ANSI *American National Standards Institute,* Instituto de estándares de Estados Unidos. Uno de los organismos de estandarización más importantes.

ATU Área de trabajo de usuario. Parte de la memoria que utilizan los procesos de usuario para almacenar los datos recibidos de una base de datos.

BCNF *Véase FNBC*

BD Abreviatura de *Base de Datos*.

Buffer Zona de la memoria que se utiliza para almacenar temporalmente algunos datos.

Codasyl *Conference on Data System Languages, Data Base Task Group*. Nombre que se da al modelo de bases de datos en red que resultó de una conferencia en el año 1971 y que provocó su aceptación como estándar.

DB Abreviatura de *Data Base,* base de datos

DBA *Data Base Administrator*, nombre que recibe el administrador de la base de datos

DBMS *Data Base Management System*, Sistema gestor de bases de datos. El software encargado de administrar y producir bases de datos.

DCL *Data Control Language*, lenguaje de control de datos. Lenguaje que proporcionan las DBMS para controlar los usuarios de la base de datos.

DDL *Data Definition Language*, lenguaje de definición de datos. Lenguaje que proporcionan las DBMS para definir la base de datos.

DML *Data Modification Language*, lenguaje de modificación de datos. Lenguaje que proporcionan las DBMS para realizar operaciones de búsqueda y modificación de datos.

ERE Modelo entidad relación extendido

FNBC Abreviatura de *Forma Normal de Boyce Codd*. Normalización estándar de las tablas relacionales.

LOB *Large Object Binary*, objeto binario largo. Tipo de datos de muchas bases de datos que admiten almacenar grandes cantidades de información en formato binario.

ODMG *Object Data Management Group*, grupo de administración de objetos de datos. Estándar utilizado para definir modelos lógicos de bases de datos de objetos.

OLAP *On Line Analytical Process*, Proceso analítico en línea. Nombre que reciben las
OOP Programación orientada a objetos

OS *Véase SO*

POO Programación orientada a objetos

QBE *Query by Example,* consultas mediante ejemplos. Lenguaje relacional utilizado en algunas de las primeras bases de datos relacionales.

RM/V2 *Relational Model Version 2*, Modelo relacional, versión 2. Modelo desarrollado por Codd, considerado como la segunda versión del modelo relacional.

RDBMS *Relational Data Base Management System*, Sistema gestor de bases de datos relacionales. El software encargado de administrar y producir bases de datos relacionales

SGBD *Véase DBMS*

SGBDR *Véase RDBMS*

SO Sistema operativo

SPARC *System Planing and Repairments Comitte*, comité de planificación de sistemas y reparaciones, subsección de ANSI.

UML *Uniform Modeling Language,* Lenguaje de modelado universal, utilizado para realizar modelos conceptuales de información orientada al objeto.

X3 Sección de ANSI encargada de los estándares de ordenadores y m