El modelo relacional

Este documento describe los fundamentos del modelo relacional. Comenzaremos por la estructura del modelo para pasar posteriormente al tratamiento de las restricciones y los valores nulos. Terminaremos con la normalización de relaciones.

Tabla de contenido

1 Introducción....1
2 Estructura del modelo relacional....2
2.1 Dominio y atributo....4
2.2 Definición formal de relación....4
2.3 Tipos de relación....5
2.4 Claves....6
3 Restricciones....7
3.1 Restricciones inherentes....7
3.2 Restricciones semánticas....7
4 Los valores nulos en el modelo relacional....9
4.1 Concepto de valor nulo....9
5 Normalización de relaciones....10
5.1 Dependencias funcionales....12
5.1.1 Dependencia funcional plena o completa....13
5.1.2 Dependencia funcional transitiva....14
5.2 Reglas de normalización....15
5.2.1 Primera forma normal FN1....15
5.2.2 Segunda forma normal FN2....17
5.2.3 Tercera forma normal FN3....18
5.2.4 La forma normal de Boyce-Codd (FNBC)....19
6 Bibliografía....21

El Modelo Relacional

1 Introducción

Codd, a finales de los años 60, presentó la teoría matemática de las relaciones en el campo de las bases de datos, que supuso un sólido fundamento teórico para el desarrollo, dentro de este enfoque relacional, de nuevos sistemas. El documento de Codd propone un modelo de datos basado en la teoría de las relaciones, donde los datos se estructuran lógicamente en forma de relaciones (tablas), siendo un objetivo fundamental del modelo mantener la independencia de esta estructura lógica respecto al modo de almacenamiento y a otras características de tipo físico. Los objetivos que perseguía el trabajo de Codd se pueden resumir en:

- ✔ Independencia física.- El modo en que se almacenan los datos no debe influir en su manipulación lógica y, por tanto, los usuarios que acceden a esos datos no han de modificar sus programas por cambios en el almacenamiento físico.
- ✔ Independencia lógica.- Añadir, eliminar o modificar cualquier elemento de la base de datos no debe repercutir en los programas y/o usuarios que están accediendo a subconjuntos parciales de los mismos (vistas).
- ✔ Flexibilidad.- En el sentido de poder ofrecer a cada usuario los datos de la forma más adecuada a la correspondiente aplicación.
- ✔ Uniformidad.- Las estructuras lógicas de los datos presentan un aspecto uniforme (tablas), lo que facilita la concepción y manipulación de la base de datos por parte de los usuarios.
- ✔ Sencillez.- Las características anteriores, así como unos lenguajes de usuario sencillos, producen como resultado que el modelo de datos relacional sea fácil de

comprender y de utilizar por parte del usuario final.

Para conseguir los objetivos citados, Codd introduce el concepto de relación (tabla) como estructura básica del modelo. Todos los datos de una base de datos se representan en forma de relaciones cuyo contenido varía en el tiempo. Una relación, en terminología relacional, es un conjunto de filas (tuplas) con unas determinadas características.

Con respecto a la dinámica del modelo, se propone un conjunto de operadores que se aplican a las relaciones. Algunos de estos operadores son clásicos de la teoría de conjuntos, no hay que olvidar que una relación se define matemáticamente como un conjunto, mientras que otros fueron introducidos específicamente para el modelo relacional. Todos ellos conforman el álgebra relacional.

La teoría de normalización, cuyas tres primeras formas normales fueron introducidas por Codd desde sus primeros trabajos, elimina dependencias entre atributos que originan anomalías en la actualización de la base de datos y proporciona una estructura más regular en la representación de relaciones, constituyendo el soporte para el diseño de bases de datos relacionales.

2 Estructura del modelo relacional

Como hemos señalado anteriormente, la relación es el elemento básico del modelo relacional, y se puede representar como una tabla.

Nombre de relación

Atributo 1	**Atributo 2**	**...**	**Atributo n**	
xxx	xxx	...	xxx	**tupla 1**
xxx	xxx	...	xxx	**tupla 2**
...	...	...	...	**...**
xxx	xxx	...	xxx	**tupla m**

En ella podemos distinguir su *nombre*, un conjunto de columnas, denominadas *atributos*, que representan las propiedades de la tabla y que también están caracterizadas por su nombre, y un conjunto de filas llamadas *tuplas*, que contienen los valores que toma cada uno de los atributos para cada elemento de la relación. Por ejemplo, a continuación se representa la relación AUTOR, donde aparece la estructura del modelo relacional.

AUTOR

Nombre	*Nacionalidad*	*Institución*
Date, C.J.	Norteamericana	Relational Institute
Codd, D.F.	Norteamericana	Relational Institute
Ceri, S.	Italiana	Politécnico de Milán
Saltor, F.	Española	U.P.C.

Atributos

TUPLAS

Cardinalidad 4

Grado 3

En ella podemos observar que el nombre de la relación es AUTOR, los atributos (Nombre, Nacionalidad e Institución); los dominios (de donde los atributos toman sus valores; varios atributos pueden tomar valores del mismo dominio); las tuplas (cada una de las cuales contiene los valores que toma el nombre, la nacionalidad y la institución de un determinado autor); el grado (número de atributos); y la cardinalidad (número de tuplas).

DOMINIOS

En una relación se puede distinguir una cabecera que define la estructura de la relación; es decir, sus atributos con los dominios subyacentes, y un cuerpo que está formado por un conjunto de tuplas que varían con el tiempo.

Esta representación de la relación como una tabla ha sido el origen de que los productos relacionales y los usuarios utilicen habitualmente el nombre de tabla (en principio ajeno a la teoría relacional) para denominar las relaciones y, como consecuencia de ello, se llame filas a las tuplas y columnas a los atributos, si bien, la terminología es irrelevante y un producto no es más o menos relacional por utilizar una terminología u otra.

A continuación se compara la terminología relacional con la que corresponde a las tablas y a los ficheros.

Relación	**Tabla**	**Fichero**
Tupla	Fila	Registro
Atributo	Columna	Campo
Grado	N.º de columnas	N.º de campos
Cardinalidad	N.º de filas	N.º de registros

2.1 Dominio y atributo

Un dominio D es un conjunto finito de valores homogéneos y atómicos V1, V2, ..., Vn caracterizado por un nombre; decimos valores homogéneos porque son todos del mismo tipo, y atómicos porque son indivisibles en lo que al modelo se refiere, es decir, si se descompusiesen, perderían la semántica a ellos asociada.

Todo dominio ha de tener un nombre, por el cual nos podemos referir a él, y un tipo de datos: así, el tipo de datos del dominio de nacionalidad es una cadena de caracteres de longitud diez. También se le puede asociar una unidad de medida, como metros, kilos, etc. y ciertas restricciones.

Los dominios pueden definirse por *intención* o por *extensión*. Por ejemplo, el dominio de las edades de las personas activas se puede definir por intención como entero de longitud dos comprendido entre 18 y 65, mientras que la definición del dominio por nacionalidades por intención sería muy pobre semánticamente, ya que permitiría toda combinación de 10 letras aun cuando no constituyesen un nombre válido de nacionalidad; por ello, sería preferible definir este dominio por extensión con los nombres de las distintas nacionalidades que admitiésemos en nuestra base de datos.

El universo del discurso de una base de datos relacional, representado por U, está compuesto por un conjunto finito y no vacío de atributos A1, A2, ..., An, estructurados en relaciones; cada atributo toma sus valores de un único dominio (dominio subyacente) y varios atributos pueden tener el mismo dominio subyacente.

Es muy usual dar el mismo nombre al atributo y al dominio subyacente. En el caso de que sean varios los atributos de una misma tabla definidos sobre el mismo dominio, habrá que darles nombres distintos, ya que una tabla no puede tener dos atributos con el mismo nombre.

Un dominio compuesto se puede definir como una combinación de dominios simples a la que se pueden aplicar ciertas restricciones de integridad. Por ejemplo, un usuario puede necesitar, además de los tres dominios *Día*, *Mes* y *Año*, un dominio compuesto por ellos denominado *Fecha*, al que podríamos aplicar las adecuadas restricciones de integridad a fin de que no aparecieran valores no válidos para la fecha.

Al igual que es posible definir dominios compuestos, existen también atributos compuestos; así, el atributo *Fecha* tomaría sus valores del dominio compuesto de igual nombre. Tanto los atributos compuestos como los dominios compuestos pueden ser tratados, si así lo precisa el usuario, como “piezas únicas” de información, es decir, como valores atómicos.

2.2 Definición formal de relación

En la noción de relación se distinguen los siguientes elementos:

- ✔ Nombre.- Las relaciones se identifican con un nombre.
- ✔ Cabecera de relación.- Conjunto de **n** pares atributo-dominio subyacente

{(Aᵢ:Dᵢ)}ⁿ, donde **n** es el grado; se corresponde con la primera fila cuando la relación se percibe como una tabla. El conjunto **A** de atributos sobre los que se define la relación se llama contexto de la misma.

- ✔ Cuerpo de la relación.- Conjunto de **m** tuplas **{t1, t2, ..., tm}**, donde cada tupla es un conjunto de **n** pares atributo-valor **{(Aᵢ:Vᵢⱼ)}**, siendo **Vᵢⱼ** el valor **j** del dominio **Dᵢ** asociado al atributo **Aᵢ**; el número de tuplas **m** es la cardinalidad. Así como la cabecera de las relación es invariante, su cuerpo varía en el transcurso del tiempo al igual que la cardinalidad.

- ✔ El esquema de la relación estará formado por el nombre **R**, si existe, y la cabecera, denotándose: **R({Aᵢ:Dᵢ}ⁿ)**. El esquema de relación representa la parte definitoria y estática y se denomina también *intención*. Se corresponde con lo que hemos llamado tipo de entidad en el ME-R.

- ✔ Estado de relación **r(R)**, al que denominaremos simplemente relación, está constituido por el esquema y el cuerpo de la relación, siendo el cuerpo el conjunto de tuplas que, en un instante dado, satisface el correspondiente esquema de la relación.

Por ejemplo, tomando la relación anterior vamos a presentar el esquema de la relación y el estado de la relación.

Esquema de la relación o Intención:

AUTOR(Nombre: *Nombre*, Nacionalidad: *Nacionalidades*, Institución: *Instituciones*)

Extensión, Estado u Ocurrencia de la relación:

AUTOR

Nombre	Nacionalidad	Institución
Date, C.J.	*Norteamericana*	*Relational Institute*
Codd, D.F.	*Norteamericana*	*Relational Institute*
Ceri, S.	*Italiana*	*Politécnico de Milán*
Saltor, F.	*Española*	*U.P.C.*

Una base de datos relacional es una base de datos percibida por los usuarios como una colección de relaciones que varían en el tiempo.

2.3 Tipos de relación

Dividiremos las relaciones en nominadas y sin nombre. Las relaciones nominadas, a su vez, pueden ser:

- ✔ Persistentes.- Aquellas cuya definición (esquema de relación) permanece en la base de datos, borrándose solamente mediante una acción explícita del usuario. Las relaciones persistentes se dividen en:

  - ➢ Relaciones base.- Existen por si mismas, no en función de otras relaciones, y se

crean especificando explícitamente su esquema de relación (nombre y conjunto de pares: atributo-dominio). Sus tuplas (extensión u ocurrencias de la relación), al igual que su definición, también se encuentran almacenadas.

- ➢ Vistas.- Son relaciones derivadas que se definen dando un nombre a una expresión de consulta. Se podría decir que son relaciones virtuales, en el sentido de que no tienen datos almacenados, sino que lo único que se almacena es su definición en términos de otras relaciones con nombre, las cuales pueden ser relaciones base, otras vistas o instantáneas.

- ➢ Instantáneas.-Son relaciones derivadas al igual que las vistas, es decir, se definen en términos de otras relaciones nominadas, pero tienen datos propios almacenados, los cuales son el resultado de ejecutar la consulta especificada o de guardar una relación base. Las instantáneas no se actualizan cuando cambian los datos de las relaciones sobre las que están definidas, pero se "refrescan" (es decir, se renuevan sus datos), cada cierto tiempo, de acuerdo con lo indicado por el usuario en el momento de su creación. Son, por tanto, sólo de lectura, no pudiendo ser actualizadas por el usuario, sino únicamente "refrescadas" por el sistema.

- ✔ Temporales.- A diferencia de las relaciones persistentes, una relación temporal desaparece de la base de datos en un cierto momento sin necesidad de una acción de borrado específica del usuario; por ejemplo, al terminar una sesión o una transacción.

Las relaciones sin nombre son los resultados de las consultas que no se materializan sino que se entregan al usuario que ha realizado la consulta, y pueden ser tanto resultados intermedios como finales; en consecuencia, las relaciones no nominadas son siempre temporales.

2.4 Claves

Una clave candidata de una relación es un conjunto de atributos que identifican unívoca y mínimamente cada tupla de la relación. Por la propia definición de relación siempre hay, al menos, una clave candidata, ya que al ser una relación un conjunto no existen dos tuplas iguales y, por tanto, el conjunto de todos los atributos siempre tiene que identificar unívocamente a cada tupla. Una relación puede tener más de una clave candidata, entre las cuales se debe distinguir:

- ✔ Clave primaria.- Es aquella clave candidata que el usuario escogerá, por consideraciones ajenas al modelo relacional, para identificar las tuplas de la relación.

- ✔ Claves alternativas.- Son aquellas claves candidatas que no han sido escogidas como clave primaria.

Además de las anteriores, una relación puede tener claves externas. Se denomina clave externa de una relación **R2** a un conjunto no vacío de atributos cuyos valores han de coincidir con los valores de la clave candidata de una relación **R1**. La clave externa y la

correspondiente clave candidata han de estar definidas sobre el mismo dominio.

Los conceptos de clave primaria y clave externa son muy importantes en el estudio de la integridad del modelo relacional, y volveremos a analizarlos en el siguiente epígrafe.

3 Restricciones

En el modelo relacional, al igual que en otros modelos, existen restricciones, es decir, estructuras u ocurrencias no permitidas, siendo preciso distinguir entre restricciones inherentes y restricciones semánticas (de usuario). Los datos almacenados en la base de datos han de adaptarse a las estructuras impuestas por el modelo y han de cumplir las restricciones de usuario a fin de constituir una ocurrencia válida del esquema.

3.1 Restricciones inherentes

Los modelos de datos tienen restricciones que impone el mismo modelo, el cual no admite ciertas estructuras; son las restricciones inherentes, que no son definidas por los usuarios, sino obligadas por el mismo modelo, lo que quita flexibilidad a la hora de representar el mundo real. Estas son:

- ✔ No hay dos tuplas iguales, por lo que se deduce la obligatoriedad de la clave primaria.
- ✔ El orden de las tuplas no es significativo.
- ✔ El orden de los atributos no es significativo.
- ✔ Cada atributo sólo puede tomar un único valor del dominio sobre el que está definido, no admitiéndose por tanto atributos multivaluados.

3.2 Restricciones semánticas

Dentro del contexto relacional, como en otros modelos de datos, existen restricciones semánticas o de usuario, que son facilidades que el modelo ofrece a los usuarios a fin de que éstos puedan reflejar en el esquema, lo más fielmente posible, la semántica del mundo real.

Sin embargo, estas restricciones semánticas del modelo relacional, al igual que ocurre con cualquier otro modelo, no son muchas veces suficientes para captar toda la semántica del universos del discurso que se está tratando de modelar. Por ello, algunos productos añaden ciertas facilidades que permiten programarlas o bien podrán incluirse en un programa de aplicación con sentencias de manipulación.

Las principales restricciones semánticas del modelo relacional son:

- ✔ Integridad de la clave ➔ Permite declarar un atributo o un conjunto de atributos como clave primaria de una relación, por lo que sus valores no se podrán repetir ni se admitirán valores nulos (integridad de la clave). La obligatoriedad de la clave primaria es una restricción inherente del modelo relacional; sin embargo, la declaración de un atributo como clave primaria de una relación es una restricción semántica que

responde a la necesidad del usuario de imponer que los valores del conjunto de atributos que constituyen la clave primaria no se repitan en la relación ni tampoco tomen valores nulos.

- ✔ Unicidad ➜ Mediante la cual se indica que los valores de un conjunto de atributos (uno o más) no pueden repetirse en una relación. Esta restricción permite la definición de claves alternativas.
- ✔ Obligatoriedad ➜ De uno o más atributos, con lo que se indica que el conjunto de atributos no admite valores nulos.
- ✔ Integridad referencial ➜ La integridad referencial es sobre las claves externas de una relación. Un conjunto de atributos que forman una clave externa definida en una relación solo pueden contener valores nulos o valores que se encuentran en el conjunto de atributos que son clave primaria de la relación a la que referencia. La clave externa puede ser también parte o total de la clave primaria de **R2**.

La integridad referencial es una importante restricción semántica que viene impuesta por el mundo real, siendo el usuario quien la define al describir el esquema relacional, y el modelo la reconoce sin necesidad de que se programe ni de que se tenga que escribir ningún procedimiento para obligar su cumplimiento.

Durante la definición de una clave externa, tenemos también que indicar las consecuencias que pueden ocurrir al realizar operaciones de borrado y modificación sobre las tuplas de la relación referenciada. Para su definición asumimos que **R2** es la relación que referencia (la que contiene la clave externa) y **R1** es la relación referenciada (la que contiene la clave primaria a la que se referencia):

- ✔ Operación restringida ➜ No se puede borrar una tupla en la relación **R1** si existen tuplas en la relación **R2** que las referencia. De igual modo, no se permiten modificaciones de los valores de clave primaria en R1 si hay tuplas en la relación R2 que las referencia. Esta es la opción por defecto.
- ✔ Operación con transmisión en cascada ➜ El borrado de tuplas en la relación **R1** provoca el borrado en cascada de las tuplas de la relación **R2** que referencia a las tuplas borradas en **R1**. Si se modifica un valor de clave primaria en **R1**, implica la modificación en cascada de los valores de clave externa en **R2** que referencian a la clave primaria modificada en **R1**.
- ✔ Operación con puesta a nulos ➜ El borrado de tuplas en la relación **R1** lleva consigo poner a valor nulo las claves externas en la relación **R2** que las referencia. De igual modo, si se modifica el valor de la clave primaria en **R1**, lleva consigo poner a valor nulo los valores de las claves externas en **R2** a las que referencia.
- ✔ Operación con puesta a valor por defecto ➜ El borrado de tuplas en la relación **R1** lleva consigo asignar a las claves externas de la relación **R2** que las referencia con su valor por defecto que habría sido definido al crear la relación correspondiente. Análogamente, si se modifica un valor de clave primaria en la relación **R1**, implica asignar el valor nulo a las claves externas de la relación **R2** que las referencia.

Las opciones de borrado y modificación pueden ser distintas.

Además de las restricciones que acabamos de exponer, existen en el modelo relacional otras restricciones que podríamos llamar de rechazo, en las que el usuario formula una condición mediante un predicado definido sobre un conjunto de atributos, de tuplas o dominios, el cual debe ser verificado por los correspondientes objetos en toda operación de actualización para el nuevo estado constituya una ocurrencia válida del esquema; en caso de que la operación intente violar la condición se impide que la operación se lleve a cabo, es decir se rechaza la operación.

En el modelo relacional se pueden distinguir dos restricciones de rechazo distintas, según la condición afecte a un único elemento de la base de datos o a más de uno:

- ✔ Verificación ➜ Comprueba, en toda operación de actualización, si el predicado es cierto o falso y, en el segundo caso, rechaza la operación. La restricción de verificación se define sobre un único elemento (incluyéndose en la definición de dicho elemento) y puede o no tener nombre.
- ✔ Aserción ➜ Actúa de forma idéntica a la anterior, pero se diferencia de ella en que puede afecta a varios elementos (por ejemplo a dos relaciones distintas) y su definición; por tanto, no va unidad a la de un determinado elemento, por lo que siempre ha de tener un nombre, ya que la aserción es un elemento más del esquema que tiene vida por sí mismo.

También es posible definir restricciones de transición haciendo referencia en el predicado a los valores anteriores a la operación de actualización y a los nuevos valores.

También es importante en una restricción el momento en el que ésta se verifica dentro de una transacción. Una transacción es un conjunto de sentencias de BD, que implican actualizaciones de datos, y que deben ejecutarse todas para dejar la BD en estado consistente. En el caso de fallo en la ejecución de una o varias de ellas, la BD debe tener mecanismos para devolverla al estado consistente previo a la ejecución de la transacción. Así, el modo de verificación es inmediato, la restricción se verificará al finalizar cada sentencia, mientras que si es diferido se verificará al finalizar la transacción.

4 Los valores nulos en el modelo relacional

4.1 Concepto de valor nulo

Se puede definir el valor nulo como una señal utilizada para representar información desconocida, inaplicable, inexistente, no válida, no proporcionada, indefinida, etc. La necesidad de los valores nulos en las bases de datos es evidente por diversas razones:

- ✔ Crear tuplas (filas) con ciertos atributos desconocidos en ese momento.
- ✔ Añadir un nuevo atributo a una relación existente; atributo que, en el momento de añadirse, no tendría ningún valor para las tuplas de la relación.
- ✔ Atributos inaplicables a ciertas tuplas.

El tratamiento de valores nulos en algún operando utilizado en las operaciones exige definir:

- ✔ Operaciones de comparación ➔ Se introducen los operadores **ES_NULO**, que es cierto si el operando es nulo o falso en caso contrario, y el operador **SI_NULO**, que se aplica a dos operandos y devuelve el valor del primero, salvo que sea nulo, en cuyo caso devuelve el valor del segundo.
- ✔ Operaciones aritméticas ➔ Se considera nulo el resultado de sumar, restar, multiplicar o dividir cuando alguno de los operandos toma el valor nulo.

5 Normalización de relaciones

El diseño de una base de datos relacional se puede realizar mediante la metodología que acabamos de exponer, aplicando al mundo real, en una primera fase, un modelo semántico como el ME-R, a fin de obtener un esquema conceptual; en una segunda fase, se transforma dicho esquema al modelo relacional mediante las correspondientes reglas de transformación (las reglas de transformación del esquema conceptual al relacional se verán en el siguiente capítulo). Si bien es muy conveniente este enfoque, existe otra posibilidad, nada aconsejable, que es plasmar directamente en el modelo relacional nuestra percepción del mundo real, obteniendo el esquema relacional sin realizar ese paso previo que es el esquema conceptual.

Aunque en general, la primera aproximación produce un esquema relacional estructurado y con poca redundancia, por lo que no es imprescindible verificar la “bondad” del esquema obtenido, siempre es conveniente aplicar un conjunto de reglas, conocidas como teoría de la normalización, que nos permiten asegurar que un esquema relacional cumple unas ciertas propiedades. En el segundo enfoque, es decir, cuando no se ha aplicado la metodología de diseño anteriormente expuesta (diseñar la BD a lo bruto), la teoría de normalización resulta imprescindible.

Entre los problemas que puede presentar un esquema relacional cuando el diseño es inadecuado cabe destacar:

- ✔ Incapacidad para almacenar ciertos hechos.
- ✔ Redundancias y, por tanto, posibilidad de inconsistencias.
- ✔ Ambigüedades.
- ✔ Pérdida de información.
- ✔ Pérdida de ciertas restricciones de integridad que dan lugar a interdependencias entre los datos (dependencias funcionales).
- ✔ Aparición en la BD, como consecuencia de las redundancias, de estados que no son válidos en el mundo real; es lo que se llama anomalías de inserción, borrado y modificación.

El esquema relacional debe ser, por tanto, analizado para comprobar que no presenta

los problemas anteriormente citados, evitando así la pérdida de información y la aparición de inconsistencias.

En el siguiente ejemplo, la relación LIBRO almacena datos sobre los libros **(ISBN, Título, Editorial, Año)** y sobre los autores que los han escrito **(Autor y Nacionalidad)**. Si observamos esta relación, vemos que presenta varios de los problemas enumerados anteriormente.

LIBRO

AUTOR	*NACIONALIDAD*	*ISBN*	*TÍTULO*	*EDITORIAL*	*AÑO*
Date, C.	*Norteamericana*	*23433*	*Databases*	*Addisson-W*	*1990*
Date, C.	*Norteamericana*	*55654*	*SQL Standard*	*Addisson-W*	*1986*
Date, C.	*Norteamericana*	*53235*	*Guide to Ingres*	*Addisson-W*	*1988*
Codd, E.	*Norteamericana*	*97875*	*Relational M.*	*Adisson-W*	*1990*
Gardarin	*Francesa*	*32245*	*Base de Datos*	*Paraninfo*	*1986*
Gardarin	*Francesa*	*55366*	*Comparación BD*	*Eyrolles*	*1984*
Valduriez	*Francesa*	*86754*	*Comparación BD*	*Eyrolles*	*1984*
Kim, W.	*Norteamericana*	*32176*	*OO Databases*	*ACM Press*	*1989*
Lochovsky	*Canadiense*	*23456*	*OO Databases*	*ACM Press*	*1989*

Los principales problemas de esta relación se derivan de la gran cantidad de redundancia que presenta. Por ejemplo, la nacionalidad del autor se repite por cada libro que ha escrito; y algo análogo sucede, cuando un libro tiene más de un autor, con la editorial y el año de publicación. Esta redundancia produce a su vez:

- ✔ Anomalías de inserción, ya que dar de alta un libro obliga a insertar en la base de datos tantas tuplas como autores tenga el libro.
- ✔ Anomalías de modificación, ya que cambiar la editorial de un libro obliga a modificar todas las tuplas que corresponden a ese libro.
- ✔ Anomalías de borrado, ya que el borrado de un libro obliga a borrar varias tuplas, tantas como autores tenga ese libro y, viceversa, el borrado de un autor nos lleva a borrar tantas tuplas como libros ha escrito ese 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 de la base de datos en manos del usuario. Al riesgo de la posible pérdida de integridad hay que añadir la falta de eficiencia asociada a estas múltiples actualizaciones.

Además de estas anomalías de inserción, borrado y modificación, existen otros problemas adicionales, como la imposibilidad de almacenar ciertos hechos o la desaparición de información que desearíamos mantener en la base de datos. Por ejemplo, si se quisiera incluir información sobre un autor del que no existiera ningún libro en la base de datos, no sería posible, ya que el atributo **ISBN** forma parte de la clave primaria de la relación; ni tampoco podríamos introducir obras anónimas (recuérdese la regla de integridad de entidad

que no permite los nulos en ningún atributo que forme parte de la clave primaria). Por otro lado, al dar de baja un libro, se pierden también los datos de sus autores (si éstos sólo tuviesen ese libro en la base de datos) y, viceversa, si borramos un autor desaparecen de nuestra base de datos los libros escritos por él (a no ser que el libro tuviera más de un autor).

Esta relación presenta todos estos problemas debido a que atenta contra un principio básico en todo diseño: Hechos distintos se deben almacenar en objetos distintos, en este caso, en relaciones distintas, con lo que se habrían evitado redundancias y, por tanto, los problemas que acabamos de escribir.

Si se hubiera llevado a cabo un diseño riguroso no se habría presentado una relación de este tipo. El problema es que a menudo no se llega a comprender completamente o a representar de forma precisa el problema, debido a una excesiva premura al realizar el análisis o a carecer el analista de conocimientos sobre metodologías de diseño de bases de datos o de experiencia para aplicarlas adecuadamente. Si se siguiera la metodología de diseño propuesta, realizando un buen diseño conceptual en el ME-R, seguido de una cuidadosa transformación al modelo relacional, se evitarían en gran parte estas anomalías, obteniéndose en general un esquema exento de errores. Sin embargo, ante las posibles dudas respecto a si un determinado esquema relacional es o no correcto, será preferible aplicar siempre a dicho esquema un método formal de análisis que determine lo que pueda estar equivocado en el mismo y nos permita llegar a otro esquema en el que se asegure el cumplimiento de ciertos requisitos; este método formal es la teoría de la normalización.

5.1 Dependencias funcionales

La teoría de la normalización se basa en el concepto de dependencias, hasta el punto de que también se la conoce como la teoría de las dependencias.

Una dependencia funcional consiste en: sea el esquema de relación **R**, definido sobre el conjunto de atributos **A** y sean **X** e **Y** subconjuntos de **A**. Se dice que **Y** dependen funcionalmente de **X**, o lo que es igual **X** determina **Y**, si y sólo si, cada valor de **X** tiene asociado en todo momento un único valor de **Y**. Representamos esta dependencia funcional de la siguiente forma:

X → Y

Se denomina *determinante* o *implicante* al descriptor que se encuentra a la izquierda del símbolo de implicación, e *implicado* al descriptor que se encuentra a la derecha. Por ejemplo, sea la relación:

LIBRO(ISBN, Título, Idioma, ...)

Podemos decir que el código de un libro determina el título del mismo

ISBN → Título

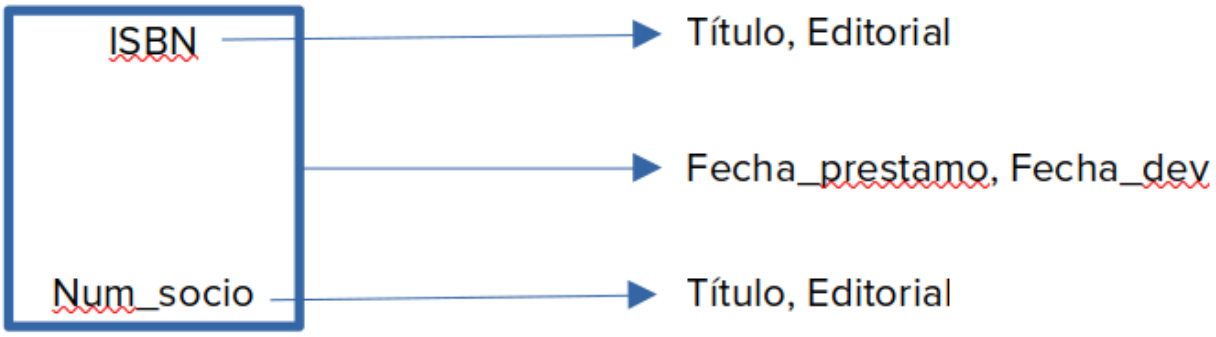
El ISBN es el determinante y título es el implicado. Esta dependencia nos dice que el título es una información acerca del libro.

El hecho de que el **ISBN** determina el título no quiere decir que conocido el código de

un libro podamos deducir, a partir de él, cuál es su título. Solamente nos limitaremos a afirmar que para dos tuplas que tengan el mismo **ISBN**, tienen que tener el mismo título.

Una herramienta muy útil a la hora de explicitar las dependencias funcionales es el grafo o diagrama de dependencias funcionales, mediante el cual se representa un conjunto de atributos y las dependencias funcionales existentes entre ellos. En el grafo aparecen los nombres de los atributos unidos por flechas, las cuales indican las dependencias funcionales y parten del implicante hacia el implicado. Cuando el implicante de una dependencia no es un único atributo, es decir, se trata de un implicante compuesto, los atributos que lo componen se encierran en un recuadro y la flecha parte de éste, no de cada atributo.

En la siguiente figura se presenta un ejemplo de cómo se visualizan las dependencias; podemos observar que **ISBN** determina funcionalmente el **Título** del libro y la **Editorial**, como indica la correspondiente flecha; de forma análoga, **Num_socio** determina el **Nombre**, **Domicilio** y **Teléfono** del socio, mientras que ambos atributos en conjunto **ISBN** y **Num_socio** (lo que se indica haciendo que la flecha parta del recuadro que los incluye) determinan **Fecha_prestamo** y **Fecha_dev**.



Existen otros conceptos, como el de dependencia plena o completa y el de dependencia transitiva, fundamentales en la teoría de la normalización, que veremos a continuación.

5.1.1 Dependencia funcional plena o completa

Sea **X(x_1, x_2)** un descriptor compuesto de dos atributos. Se dice que el atributo **Y** tiene dependencia funcional completa o plena de **X** si dependen funcionalmente de **X** pero no depende de ningún subconjunto del mismo, es decir, **Y** dependen de todos, y exclusivamente de todos, los atributos que forman **X**. Se representa por **X ==> Y**.

Por ejemplo, supongamos la siguiente relación de la cual dibujamos el grafo anterior:

PRESTAMO(ISBN, Título, Editorial, Num_socio, Nombre, Domicilio, Teléfono, Fecha_préstamo, Fecha_dev)

La dependencia funcional

ISBN, Num_socio → Fecha_prestamo

indica que, dado un determinado ISBN y un número de socio, existe una única fecha de préstamo. Ni ISBN, ni tampoco número de socio, implican, por sí solos, la fecha del préstamo, ya que tanto un libro se puede prestar en varias fechas como un socio puede recibir libros prestados en varias fechas. Por tanto, la dependencia funcional anterior es completa, lo que se representa:

ISBN, Num_socio ==> Fecha_prestamo

Esto quiere decir que la fecha del préstamo constituye una información sobre el conjunto de libro y socio, pero esta información no atañe a un libro o a un socio por separado.

Por el contrario, la dependencia:

ISBN, Num_socio ==> Editorial

No es plena, ya que:

ISBN → Editorial

Se dice entonces que, en esa dependencia, **Num_socio** es un atributo redundante o ajeno a la dependencia (también llamado extraño).

5.1.2 Dependencia funcional transitiva

Sea la relación **R(X,Y,Z)** en la que existen las siguientes dependencias funcionales:

X → Y

Y → Z

Y ↛ X

Se dice entonces que **Z** tiene una dependencia transitiva respecto de **X** a través de **Y**, lo que se representa por:

X --> Z

Por ejemplo, consideremos la siguiente relación:

LIBROS(ISBN, Editorial, País)

donde tenemos, para cada libro, su código, la editorial que lo publica y el país de la editorial (suponemos que una editorial tiene su sede en un único país), se tendrán las siguientes dependencias:

ISBN → Editorial

Editorial → País

Además, **Editorial ↛ ISBN**, al depender **País** de **Editorial** y ésta de **ISBN**, entonces **País** depende transitivamente de **ISBN**.

ISBN --> País.

5.2 Reglas de normalización

La teoría de la normalización está basada en la aplicación de una serie de reglas a las que se les denomina *Reglas de normalización*. Se dice que una relación está en una determinada forma normal si satisface un cierto conjunto específico de restricciones impuestas por la regla de normalización correspondiente. La aplicación de una regla de normalización es una operación que toma una relación como argumento de entrada y da como resultado dos o más relaciones, y:

- ✔ La relación a la que se aplica la regla es desestimada en el nuevo esquema relacional obtenido.
- ✔ No se introducen nuevos atributos en el esquema relacional resultante de la normalización.
- ✔ Los atributos de la relación a la que se aplica la regla pasan a formar parte de la intención de una o más de las relaciones resultantes.
- ✔ En la aplicación de la regla de normalización se ha debido eliminar, al menos, una dependencia existente entre los atributos de la relación a la que se aplica la regla.

De esta forma, la sucesiva aplicación de las reglas de normalización va a dar lugar a la generación de un número mayor de relaciones que formen parte del esquema relacional y, desde un punto de vista sólo lógico, una redundancia de los atributos considerados en el esquema.

La aplicación sucesiva de las reglas de normalización restringe, por tanto el número de relaciones que las satisfacen. Por regla general, se dice que un esquema relacional es consistente si las relaciones satisfacen al menos la forma normal de Boyce-Codd

5.2.1 Primera forma normal FN1

Para que una relación esté en FN1 tiene que cumplir los siguientes requisitos:

- ✔ El orden de las tuplas y las columnas no es significativo.
- ✔ Los valores de los atributos son atómicos, es decir, solo toman un valor del dominio subyacente.
- ✔ No existen filas duplicadas, es decir, es obligatorio la existencia de la clave primaria.

Si repasamos los requisitos anteriores vemos que consisten en las restricciones inherentes del modelo relacional. Por tanto, podemos decir que la primera forma normal verifica que una relación cumple los requisitos anteriores.

Por ejemplo observemos la siguiente tabla:

ALUMNO(Nif, Nombre, Apellidos, Teléfono, Grupo)

Nif	Nombre	Apellidos	Teléfono	Grupo
30000001A	*José*	*González Gómez*	*999-292929*	*1ºCFGS*

			888-303030	
30000002B	*Sara*	*Martínez Marín*	*999-797979* *888-808080*	*2ºCFGS*
30000003C	*Javier*	*Sánchez Gil*	*999-494949* *888-505050* *666-515151*	*1ºCFGS*

Se observa que para cada fila, existen varios valores para el atributo *Teléfono,* es decir, hay alumnos que tienen más de un número de teléfono. Para pasar esta tabla (no es relación por que no está en FN1) a FN1 tendríamos dos opciones:

- ✔ Crear más atributos para albergar cada uno de los posibles valores del atributo repetido. Esta opción no es recomendable y solo es viable si se conoce a priori cuantos valores como máximo podría tomar el atributo multivaluado. Si no es así no podría implementarse y, en caso de que si lo fuera, dificultaría ciertas operaciones como las búsquedas.
- ✔ Cambiar el dominio del atributo multivaluado aumentando su longitud para acomodar varios números de teléfono. Esta opción tampoco se recomienda, ya que el atributo **`Teléfono`** se vuelve ambiguo, pudiendo representar un número de teléfono, o una lista de números de teléfono, o de hecho cualquier cosa. Además, no podrían hacerse consultas en las que se buscaran alumnos con el mismo número de teléfono.
- ✔ Añadir nuevas tuplas en las que se repetirían los valores de los otros atributos.

Si aplicamos la última opción, tendremos que repetir el resto de atributos de la tupla para cada uno de los valores del atributo **`Teléfono`**, tal y como aparece a continuación.

ALUMNO(Nif, Nombre, Apellidos, Teléfono, Grupo)

Nif	**Nombre**	**Apellidos**	**Teléfono**	**Grupo**
30000001A	*José*	*González Gómez*	*999-292929*	*1ºCFGS*
30000001A	*José*	*González Gómez*	*888-303030*	*1ºCFGS*
30000002B	*Sara*	*Martínez Marín*	*999-797979*	*2ºCFGS*
30000002B	*Sara*	*Martínez Marín*	*888-808080*	*2ºCFGS*
30000003C	*Javier*	*Sánchez Gil*	*999-494949*	*1ºCFGS*
30000003C	*Javier*	*Sánchez Gil*	*888-505050*	*1ºCFGS*
30000003C	*Javier*	*Sánchez Gil*	*666-515151*	*1ºCFGS*

Al repetir el resto de atributos por cada valor del atributo **`Teléfono`** hemos creado grupos repetitivos. Esta tabla todavía no está en FN1 ya que, como se puede observar, el valor de clave primaria se repite, lo que viola una de las restricciones inherentes del modelo.

Para evitarlo, no hay más remedio que crear dos relaciones: una con los alumnos y otra con los números de teléfono. Serían las siguientes:

ALUMNO(Nif, Nombre, Apellidos, Grupo)

Nif	**Nombre**	**Apellidos**	**Grupo**

30000001A	*José*	*González Gómez*	*1ºCFGS*
30000002B	*Sara*	*Martínez Marín*	*2ºCFGS*
30000003C	*Javier*	*Sánchez Gil*	*1ºCFGS*

TELEFONO(Teléfono, Nif)

Teléfono	Nif
999-292929	*30000001A*
888-303030	*30000001A*
999-797979	*30000002B*
888-808080	*30000002B*
999-494949	*30000003C*
888-505050	*30000003C*
666-515151	*30000003C*

Además, el atributo **Nif** de la relación **Teléfono** se define como clave externa a la relación **Alumno**.

Ahora si, ambas tablas pueden considerarse relaciones ya que cumplen las restricciones del modelo relacional y, por ende, están en FN1.

5.2.2 Segunda forma normal FN2

Una relación R satisface la segunda forma normal (FN2) si, y sólo si, satisface la primera forma normal y cada atributo de la relación depende funcionalmente de forma completa de la clave primaria de esa relación.

Por ejemplo, consideremos la siguiente relación:

PRESTAMO(ISBN, Num_socio, Nombre, Domicilio, Teléfono, Fecha_préstamo, Fecha_dev)

En la que el conjunto de ISBN y Num_socio forman la clave principal de la relación, y en la que existen las siguientes dependencias funcionales:

Num_socio → Nombre

Num_socio → Domicilio

Num_socio → Teléfono

ISBN, Num_socio → Fecha_préstamo

ISBN, Num_socio → Fecha_dev

Podemos observar que solo hay dos atributos que dependen de la clave completa: **Fecha_préstamo** y **Fecha_dev**. El resto depende de parte de la clave. Los atributos **Nombre**, **Domicilio** y **Teléfono** depende de **Num_socio**.

Por tanto esta relación no se encuentra en FN2 al haber atributos que sólo dependen de parte de la clave y no de la clave completa. Esta relación presenta los siguientes inconvenientes al no estar en FN2:

- ✔ Inserción de tuplas.- No se pueden conocer los socios que hay en el sistema hasta que no exista un préstamo, ya que si se introduce una tupla sólo con los valores del socio, habría valores nulos en parte de la clave (**ISBN**) con lo que se viola la regla de integridad de la clave.
- ✔ Borrado de tuplas.- Si se eliminan las tuplas correspondientes a los libros prestados a un socio, se pierde la información que representa ese socio.
- ✔ Actualización de tuplas.- Si se cambia el número de teléfono de un socio, hay que actualizar todas las tuplas donde aparezca ese número de socio.

Para eliminar todos estos inconvenientes es necesario realizar un proceso de descomposición de la relación *PRESTAMO* en dos nuevas relaciones que satisfacen la FN2. Estas relaciones quedarían de la siguiente manera:

PRESTAMO(ISBN, Num_socio, Fecha_préstamo, Fecha_dev)

SOCIO(Num_socio, Nombre, Domicilio, Teléfono)

Vemos que los atributos que dependían del atributo **Num_socio** se han eliminado de la relación original, formando, junto con éste, una nueva relación, *SOCIO*. En esta nueva relación el atributo del que dependían será clave primaria y el atributo **Num_socio** de la relación *PRESTAMO* será definido como clave externa con la relación *SOCIO*, para que sólo pueda tomar valores nulos o bien igual a valores existentes en alguna tupla de la relación *SOCIO*, de esta forma no habrá pérdida de información.

5.2.3 Tercera forma normal FN3

Una relación R satisface la tercera forma normal (FN3) si, y sólo si, satisface la segunda forma normal y cada atributo no principal de la relación no depende transitivamente de la clave principal.

Por ejemplo, consideremos la siguiente relación:

MATRICULA(nif, asignatura, nota, aula, ubicación)

Esta relación representa la docencia que recibe un alumno de una asignatura en un aula que se encuentra situada en una ubicación, por la cual será evaluado con una nota. La clave principal son los atributos nif y asignatura. Se observan las siguientes dependencias:

nif, asignatura → nota

nif, asignatura → aula

aula → ubicación

Como se puede apreciar, la relación está en FN2, ya que los atributos no principales que dependen de la clave lo hacen de forma completa, pero el atributo **ubicación** depende del atributo **aula**, y como el atributo **aula** depende de la clave tenemos una dependencia transitiva de **nif, asignatura --> ubicación**. Es decir el atributo ubicación depende completamente de la clave a través del atributo aula, que no forma parte de la clave.

La existencia de estas dependencias entre los atributos no principales ocasiona problemas en la manipulación de la relación en:

- ✔ Inserción de tuplas.- No se pueden conocer las aulas de cada ubicación utilizadas para la docencia de asignaturas hasta que no haya algún alumno matriculado en esa asignatura.
- ✔ Borrado de tuplas.- Una vez borrados todos los alumnos matriculados en una asignatura, se pierde la información correspondiente a las aulas de cada ubicación utilizadas para la docencia.
- ✔ Modificación de tuplas.- Existe una gran redundancia de información puesto que se almacena la ubicación en el que se encuentra situada un aula para cada uno de los alumnos que cursan una asignatura que se imparte en dicha aula.

Para eliminar los problemas que ocasiona en la relación la existencia de esta dependencia transitiva, hay que descomponerla en dos relaciones, quedando de la siguiente manera:

MATRICULA(nif, asignatura, aula, nota)

AULA(aula, ubicación)

Donde se observa que la relación **MATRICULA** está en FN3, ya que todas las dependencias que existen son completas entre atributos no principales y la clave de la relación. Se ha generado una nueva relación, **AULA**, para la cual el atributo **aula** es clave, y tiene un único atributo no principal, **ubicación**, el cual depende funcionalmente de la clave, por lo que también se encuentra en FN3. El atributo **aula** de la relación **MATRICULA** deberá definirse como clave externa de la relación **AULA**, de forma que para cualquier tupla de la relación **MATRICULA** este atributo sólo pueda tomar valores nulos o bien igual a valores existentes en alguna tupla de la relación **AULA**.

5.2.4 La forma normal de Boyce-Codd (FNBC)

Una relación **R** satisface la forma normal de Boyce-Codd (FNBC) si, y sólo si, se encuentra en FN1, y cada determinante funcional es una clave candidata de la relación R.

Observemos la siguiente relación

MATRICULA(dni, asignatura, nie, nota, aula)

Se observa que hay dos claves candidatas: **dni, asignatura** y **asignatura, nie** (Número de Identificación Escolar). En esta relación hay un dependencia funcional mutua entre el atributo **dni** y el atributo **nie**. Una dependencia funcional es mutua cuando

X → Y

Y → X

Y se representa por **X <--> Y**. En el ejemplo en cuestión nos encontramos la dependencia funcional mutua:

nif <--> nie

Además, de la dependencia funcional mutua descrita, existen las siguientes dependencias:

nif, asignatura → nota

nif, asignatura → aula

asignatura, nie → nota

asignatura, nie → aula

Vemos que los atributos no principales mantienen dependencia funcional completa con las claves candidatas.

En las dos claves candidatas está presente un atributo común, el atributo asignatura, el cual forma parte de los dos determinantes funcionales, por lo que ambas claves se encuentran traslapadas. Dos claves candidatas se dice que están traslapadas si cada una de ellas está formada por dos o más atributos y alguno de ellos es común a ambas. La presencia de claves traslapadas pueden ocasionar serios problemas en el mantenimiento de la información que hay que resolver. La relación **MATRICULA** sería conveniente descomponerla en dos relaciones que satisfagan la FNBC, quedando de la siguiente forma:

MATRICULA(nif, asignatura, nota, aula)

ALUMNO(nif, nie)

La descomposición realizada ha sido: se crea una nueva relación (**MATRICULA**) que tendrá como clave principal una de las claves candidatas de la relación original (**nif, asignatura**) y los atributos que no forman parte de ninguna clave candidata (**nota, aula**). Con los atributos que mantienen dependencia funcional mutua (**nif <--> nie**) se crea otra relación (**ALUMNO**), en la que un determinante es clave principal (**nif**), y el otro es clave alterna (**nie**), de esta forma sigue existiendo la dependencia funcional mutua que mantenían. El atributo de esta relación (**ALUMNO**) elegido como clave principal (**nif**), será, en la otra relación (**MATRICULA**) clave externa con esta.

6 Bibliografía

GÓMEZ NIETO, Miguel Ángel y LUQUE RUIZ, Irene, *Diseño y uso de Bases de Datos Relacionales*. 1997 RA-MA ISBN 9788478972791