UNIDAD 4 NORMALIZACIÓN

1 Problemas del esquema relacional

2 Formas Normales

3 Conceptos previos

3.1 Dependencia funcional

3.2 Dependencia funcional completa

3.3 Conjunto de interés

3.4 Diagrama de dependencia funcionales

3.5 Clave de una relación

3.6 Atributo primo

3.7 Dependencia funcional transitiva

4 Primera Forma Normal (1FN)

5 Segunda forma normal (2FN)

6 Tercera forma normal (3FN)

7 Forma Normal de Boyce-Codd

1 Problemas del esquema relacional

Los malos diseños en una base de datos podrían ocasionar los siguientes problemas:

- **Redundancia**: Que los datos puedan repetirse continuamente e innecesariamente por las tabla de la base de datos. Si esta redundancia es excesiva tendremos que revisar el diseño de nuestra base de datos.
- **Ambigüedades**: Datos que no clarifican suficientemente el registro al que representan. Los datos de cada registro podrían referirse a más de un registro o incluso puede ser imposible saber a qué ejemplar exactamente se están refiriendo.
- **Pérdida de restricciones de integridad**: Normalmente debido a dependencias funcionales. Más adelante veremos este concepto. Se arreglan fácilmente siguiendo una serie de pasos concretos.
- **Anomalías en operaciones de modificación de datos**:
    - **Anomalías de actualización**: inconsistencia de datos a consecuencia de datos redundantes y actualizaciones parciales.
    - **Anomalías de Borrado:** pérdida no deseada de datos a consecuencia de borrado de otros datos.
    - **Anomalías de inserción**: imposibilidad de añadir nuevos datos a consecuencia de la ausencia de otros datos.

Un diseño de una Base de Datos poco meditado da lugar a problemas futuros. Esto se evita con la normalización.

La normalización es un proceso de conversión de una relación (o tabla) en una forma estándar. Nos dice si la tabla cumple con los requisitos mínimos para poder funcionar con ella.

La teoría de la normalización es un método que permite asegurar si un diseño relacional (tanto si proviene de la conversión de un diagrama entidad-relación como si se ha efectuado directamente) es más o menos correcto.

2 Formas normales

Habitualmente, el diseño de una base de datos termina en el paso del modelo entidad-relación al modelo relacional. No obstante, siempre que se diseña una base de datos, se ha de medir la calidad de la misma, y si no se cumplen determinados criterios de calidad, hay que realizar, de forma iterativa, sucesivos refinamientos en el diseño para alcanzar la calidad deseada.

La calidad de una base de datos se mide con la forma normal en la que se encuentra su diseño. Esta forma normal se puede alcanzar imponiendo ciertas restricciones a las tablas y columnas de la base de datos. Se llama **normalización** al proceso de obligar a los atributos de un diseño a cumplir ciertas 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). Es decir cumplen la relación de inclusión de la siguiente figura:

La teoría de formas normales es una teoría absolutamente matemática, peo vamos a intentar verlo de una manera más intuitiva.

La Cuarta Forma Normal y la Quinta Forma Normal se ocupan de las dependencias entre atributos multivaluados. Muchos diseñadores opinan que basta con llegar a la Forma Normal de Boyce-Codd, ya que la Cuarta Forma Normal, y sobre todo la Quinta Forma Normal son polémicas. Hay autores que opinan que hay peores bases de datos en Quinta Forma Normal que en Tercera Forma Normal.

Además, existen otras formas normales como la **Forma Normal de Domino Clave** (FNDC), que trata las restricciones y los dominios de los atributos, y la **Sexta Forma Normal,** que aborda ciertas consideraciones con las bases de datos temporales.

Estas últimas formas normales (4FN, 5FN, FNDC y 6FN) tienen una aplicación en el mundo real únicamente teórica, y por tanto, no van a ser materia de este módulo.

3 Conceptos previos

Veamos una serie de conceptos previos

3.1 Dependencia funcional

Las definiciones de las diferentes formas normales se basan en el concepto de **dependencia funcional**.

Dado dos atributos (o conjuntos de atributos) X e Y de una relación R, diremos que **Y depende funcionalmente de X** si para cada valor de X existe uno, y sólo uno, valor de Y asociado con él. También diremos que X implica Y. Lo simbolizaremos por X → Y. Al conjunto X se le denomina **determinante** o **implicante**. Al conjunto Y se le llama **implicado**.

Otra manera de decirlo: **Y depende funcionalmente de X si cada valor de X tiene asociado siempre el mismo valor de Y** en una relación R que contiene a X e Y como atributos.

Veamos varios ejemplos.

Ejemplo 1. Sea el siguiente esquema de una relación:

ALUMNO (cod_alumno:dom_cod, alumno:dom_alumno, promedio:dom_promedio)

CP {cod_alumno}

{cod_alumno} → {alumno}

es decir, para un código_alumno concreto solo hay un alumno posible.

Ejemplo 2. Sea el siguiente esquema de relación:

PRODUCTO (código:dom_cod, nombre:dom_nombre, precio:dom_precio, descripción:dom_descripción)

CP {código}

{código} → {nombre}

Es decir, un código de producto solo puede tener asociado un único nombre, dicho de otro modo, a través del código del producto se localiza un único nombre.

Ejemplo 3. Sea el siguiente esquema de relación:

HA_ESCRITO(dni:dom_dni, nombre:dom_nombre, ISBN:dom_cod, título:dom_titulo, euros:dom_euros)

CP {dni,ISBN}

cada tupla representaría la siguiente información: "el escritor de dni, *dni*, que se llama *nombre* ha escrito, solo o en colaboración, el libro de código *ISBN* y de título *título* y le han pagado la cantidad de *euros* euros"

En los atributos de esta relación, se pueden encontrar las siguientes dependencias funcionales:

1. {dni} → {nombre} (es decir, para un valor de dni sólo existe asociado un posible valor del atributo nombre)
2. {dni, ISBN} → {nombre}
3. {dni, título} → {nombre}
4. {dni, euros} → {nombre}
5. {dni, ISBN, título} → {nombre}
6. {dni, ISBN, euros} → {nombre}
7. {dni, título, euros} → {nombre}
8. {dni, ISBN, título, euros} → {nombre}
9. {ISBN} → {título}
10. {ISBN, dni} → {título}
11. {ISBN, nombre} → {título}
12. {ISBN, euros} → {título}
13. {ISBN, dni, nombre} → {título}
14. {ISBN, dni, euros} → {título}
15. {ISBN, nombre, euros} → {título}
16. {ISBN, dni, nombre, euros} → {título}

17. {ISBN, dni} → {euros}
18. {ISBN, dni, nombre} → {euros}
19. {ISBN, dni, título} → {euros}
20. {ISBN, dni, nombre, título} → {euros}
21. {dni, título} → {nombre, título}

Y alguna más, que no ponemos.

3.2 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,** pero no depende funcionalmente de ningún subconjunto de X.

Ejemplo 1: Dada la siguiente tabla:

CLIENTE (DNI, nombre, apellidos, empresa, sueldo)

CP {DNI}

- ✔ El conjunto atributos formado por nombre y DNI produce una dependencia funcional sobre el atributo apellidos {DNI, nombre} → {apellidos}
- ✔ Pero no es completa , ya que el DNI individualmente también produce una dependencia funcional sobre apellidos. El DNI sí que produce una dependencia funcional completa sobre el campo apellidos.

Así, del ejemplo 3 anterior, las dependencias funcionales completas son:

1. {dni} → {nombre}
2. {ISBN} → {título}
3. {ISBN, dni} → {euros}
4. {dni, título} → {nombre, título}

3.3 Conjunto de interés

Un conjunto de interés de una relación es un conjunto de dependencias funcionales que basta considerar para normalizar esa relación eliminado problemas de redundancias. Este conjunto cumple las siguientes condiciones:

- Y consta de un solo elemento.
- Y depende completamente de X.

El conjunto de interés del ejemplo es:

{{dni} → {nombre} , {ISBN} → {título}, {ISBN, dni} → {euros}}

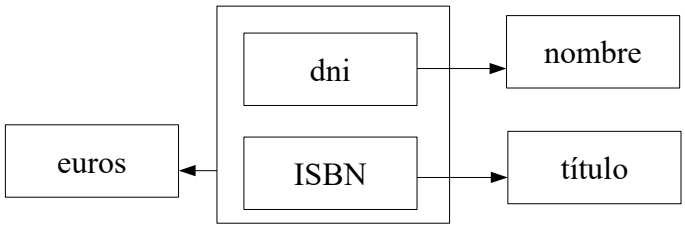
Para la normalización de las relaciones, sólo se considerarán las dependencias funcionales de este conjunto.

3.4 Diagrama de dependencias funcionales

Permite una representación gráfica de las dependencias funcionales facilitando su estudio. Se

utilizan cuadros para enmarcar los atributos o conjuntos de atributos y flechas para denotar la dependencia funcionales. Normalmente sólo se representan las dependencias funcionales del conjunto de interés.

El diagrama de dependencias funcionales de nuestro ejemplo es:



3.5 Clave de una relación

Sea R un esquema de relación cuyo conjunto de atributos es A = {A_1, A_2, …, A_n}, y sea C un subconjunto de ese esquema (C ⊂ A); se dice que C es una clave de R, si C es la clave primaria de R, o bien si C tiene una restricción de unicidad (es única)

En el ejemplo sólo hay una clave que es {dni, ISBN}.

3.6 Atributo primo

Sea R un esquema de relación cuyo conjunto de atributos es A = {A_1, A_2, …, A_n}, un atributo A se dice que es primo si forma parte de alguna clave de R.

En ele ejemplo serían atributos primos dni e ISBN.

3.7 Dependencia funcional transitiva

Se produce cuando tenemos tres conjuntos de atributos **X**, **Y**, **Z**, los cuales tienen las siguientes dependencias funcionales:

X → Y (**Y** depende funcionalmente de **X**)

Y → Z (**Z** depende funcionalmente de **Y**)

Y - /→ X (**X** no depende funcionalmente de **Y**)

Entonces se dice que **Z** tiene una dependencia funcional transitiva respecto a **X** a través de **Y**. Se denota como **X - → Z.**

Ejemplo 1. Dada la tabla:

PRODUCTO (Código, Nombre, Fabricante, País)

CP {Código}

Se observa que:

Código → Fabricante

Fabricante → Pais.

Por tanto:, Código - → País, es decir, el país depende transitivamente del Código de Producto

Ejemplo 2, Si X el atributo Número_de_Clase de un instituto, Y el atributo Código_Tutor y **Z** es el Código_Departamento. Entonces:

- **{número_de_clase} → {código_tutor}** (el tutor depende funcionalmente del número de clase, es decir, para un número de clase concreto sólo hay un tutor)
- **{código tutor} → {código_departamento}** (el código del departamento depende funcionalmente del código del tutor, es decir, un tutor sólo puede estar en un departamento)
- **{código_tutor} -/-> {número_de_clase}** (el código de clase no depende funcionalmente del código del tutor, es decir, un código de tutor se puede corresponder con varios números de clase, un mismo tutor puede serlo de varias clases)

Entonces **{número_de_clase} - → {código_departamento}** (el código del departamento depende transitivamente del código de la clase)

4 Primera forma normal (1FN)

Una relación o tabla está en primera forma normal (1FN) si todos los valores de sus atributos son atómicos, es decir, sus elementos se pueden considerar como unidades indivisibles. En ninguna fila puede haber ningún atributo con valores múltiples.

Sea el siguiente esquema relacional:

ALUMNO (dni:dom_dni, nombre: dom_nombre, asignatura: dom_conj_asig)

Clave Primaria {dni}

Donde los conjuntos de dominios son los siguientes:

dom_dni: cad(9)

dom_nombre: cad(30)

dom_conj_asig: conjunto de dom_asign

Una extensión de este esquema podría ser la siguiente:

DNI	NOMBRE	ASIGNATURA
18455978	Juan	Pascal, C
24789123	Eva	Visual Basic, C
12568741	Luis	Java, C++

Si observamos el atributo Asignatura de la siguiente tabla vemos que no cumple con la primera forma normal.

Paso a 1FN

La técnica para transformar una relación que no está en 1FN a una relación que está en 1FN es muy simple. La razón para que no este en 1FN es que posee un atributo A multivaluado, por lo que la solución está en extraer de la relación R el atributo A y crear una nueva relación R' que contiene

la clave primaria de R y el atributo A.

En nuestro ejemplo, tendríamos una nueva relación ALUMNOS_MATRICULA cuyo esquema sería (dni: dom_dni, asignatura: dom_asig).

La extensión de las relaciones quedaría:

ALUMNO

DNI	NOMBRE
18455978	Juan
24789123	Eva
12568741	Luis

ALUMNO_MATRICULA

DNI	ASIGNATURA
18455978	Pascal
18455978	C
24789123	Visual Basic
24789123	C
12568741	Java
12568741	C++

¿Cuál es la clave primaria?

La nueva clave está formada por ambos campos.

5 Segunda forma normal (2FN)

Una tabla está en segunda forma normal (2FN) si se encuentra en 1FN y todo atributo no primo tiene una dependencia funcional completa de la clave primaria y no de parte de ella. Toda la clave principal debe hacer dependientes el resto de atributos, si hay atributos que dependen sólo de parte de la clave, entonces esa parte de la clave y esos atributos formarán otra tabla.

La comprobación solo hay que hacerla cuando la clave primaria está compuesta por varios atributos.

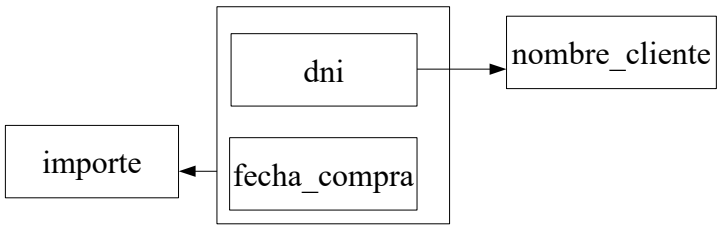
Veamos un ejemplo:

Sea el siguiente esquema relacional:

CLIENTE (dni: dom_dni, fecha_compra: dom_fecha, importe: dom_importe,
nombre_cliente: dom_nombre)

Tiene como clave primaria el conjunto de atributos DNI, Fecha_Compra.

Veamos el diagrama de dependencias funcionales



Está claro que Nombre_Cliente no tiene dependencia funcional de Fecha _Compra, sino sólo de DNI, sólo el importe tiene dependencia funcional completa y, por lo tanto, esta tabla no está en 2FN.

La extensión de la relación sería:

CLIENTES

DNI	FECHA_COMPRA	IMPORTE	NOMBRE_CLIENTE
35879421	11/10/08	100	Val Pérez, J.
71456789	12/10/08	2000	Martínez López A.

Paso a 2FN

Si tenemos una relación R cuya clave primaria es CP está en 1FN pero no en 2FN. De este hecho y de la definición de 2FN se puede deducir que la clave CP consta de más de un atributo y que existe al menos un atributo (o conjunto de atributos) no-primo A de R que no depende completamente de CP. Por tanto existe un subconjunto propio de CP, llámese S, tal que A depende completamente de S. Por otra parte, es posible que exista un atributo B de R que sí dependa completamente de CP.

La técnica consiste en obtener un conjunto de relaciones proyectando la relación R adecuadamente, teniendo en cuenta cuáles son las dependencias no completas que aparecen en R. En el supuesto anterior se deben hacer las siguientes proyecciones: R1=R[S, A] y R2=R[CP, B]. La clave primaria de R1 es S y la clave primaria de R2 es CP.

Además R2 tiene una clave ajena que hace referencia a R1. Así, las dos relaciones resultantes quedan en 2FN.

En nuestro ejemplo:

Para solucionarlo tendríamos que construir dos relaciones:

COMPRAS (dni: dom_dni, fecha_compra:dom_fecha, importe: dom_importe)

CP {dni, fecha_compra}

CLIENTES (dni:dom_dni, nombre_cliente:dom_nombre)

CP {dni}

La extensión del esquema es:

COMPRAS

DNI	FECHA_COMPRA	IMPORTE
35879421	11/10/08	100
71456789	12/10/08	2000

CLIENTES

DNI	NOMBRE_CLIENTE
35879421	Val Pérez, J.
71456789	Martínez López A.

Podemos verlo de otra manera más resumida:

- ✔ Se comprueba que está en 1FN
- ✔ Se crea una primera tabla con la clave primaria de la tabla inicial y todos los atributos que tienen una dependencia funcional total con ella.
- ✔ Se crea una segunda tabla para los atributos que no dependan de la totalidad de la clave compuesta. El identificador de esta segunda tabla será la parte de la clave compuesta de que dependen el atributo o atributos seleccionados.

6 Tercera forma normal (3FN)

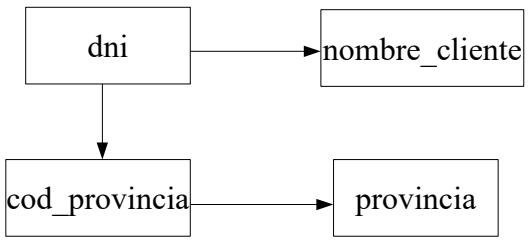
Una relación o tabla está en tercera forma normal si se encuentra en 2FN y no contiene dependencias funcionales transitivas, es decir, un atributo que no sea clave depende transitivamente de las claves de la tabla. Es decir no está en 3FN cuando algún atributo depende funcionalmente de atributos que no son clave.

Por ejemplo sea el siguiente esquema de relación:

CLIENTES (dni: dom_dni, nombre_cliente: dom_nombre, cod_provincia: dom cod, provincia: dom_provincia)

CP {dni}

El diagrama de dependencias funcionales es:



Se observa que existe una dependencia funcional entre dos atributos no-primos, cod_provincia y provincia, ya que las provincias tienen códigos únicos.

La extensión del esquema es:

CLIENTES

DNI	NOMBRE_CLIENTE	COD_PROVINCIA	PROVINCIA
35879421	Val Pérez, J.	50	Zaragoza
71456789	Martínez López A.	22	Huesca
12789456	Sanz Serrano E.	40	Teruel

Paso a 3FN

Suponer que R es una relación con clave primaria CP que está en 2FN y no en 3FN. Supóngase que B es un atributo que depende funcionalmente de otro atributo A, que no es clave.
Las relaciones que se obtienen como proyección de R son R1=R[CP,C,A] (donde C es el resto de atributos no-primos de R) y R2=R[A,B]. La clave primaria de R1 es CP y además A es clave ajena que hace referencia a R2. La clave primaria de R2 es A.

En nuestro ejemplo:

Para ponerlo en 3FN obtendríamos dos relaciones:

CLIENTES (dni: dom_dni, nombre_cliente: dom_nombre, cod_provincia:dom:cod)

CP {dni}

Clave ajena {cod_provincia} hace referencia a PROVINCIA.

PROVINCIAS (cod_provincia: dom_cod, nombre: dom_nombre)

CP {cod_provincia}

La extensión del esquema sería:

CLIENTES

DNI	NOMBRE_CLIENTE	COD_PROVINCIA
35879421	Val Pérez, J.	50
71456789	Martínez López A.	22
12789456	Sanz Serrano E.	40

PROVINCIAS

COD_PROVINCIA	PROVINCIA
50	Zaragoza
22	Huesca
40	Teruel

Podemos ver el paso a 3FN de otra manera más resumida:

- ✔ Se comprueba que está en 2FN
- ✔ Por cada atributo que dependa de otro que no sea clave, se crea una nueva tabla 8si no existe) a la que se le pasan esos atributos, con clave del atributo del cual dependían. Este atributo se queda en la primera tabla como clave ajena. Si la nueva tabla sigue sin estar en 3FN se repite el proceso de nuevo hasta que obtengamos una 3FN.

7 Forma Normal de Boyce-Codd. FNBC o BCFN

Una tabla se encuentra en Forma Normal de Boyce-Codd (FNBC) si está en 3FN y además todo determinante o implicante de la tabla es una clave candidata.

Veamos un ejemplo: En una empresa, un trabajador puede trabajar en varios departamentos. En cada departamento hay varios responsables, pero cada trabajador sólo tiene asignado uno.

El esquema de la relación sería:

ORGANIZACIÓN (trabajador, departamento, responsable)

La única clave candidata es (trabajador, departamentos, como solo hay una será la clave primaria.

La extensión del esquema sería:

ORGANIZACIÓN

TRABAJADOR	DEPARTAMENTO	RESPONSABLE
1	Producción	10
2	Producción	11
3	Ventas	12
3	Producción	10
4	Producción	13
5	Ventas	14
5	Producción	11
6	Ventas	12
7	Ventas	14

Si añadimos la limitación de que el trabajador que sea responsable de un departamento solo puede serlo de uno, este detalle produce una dependencia funcional ya que Responsable → Departamento.

Por lo tanto hemos encontrado un determinante (Responsable) que sin embargo no es clave candidata. Por ello, esta tabla no está en FNBC.

La solución sería:

PERSONAL

TRABAJADOR	RESPONSABLE
1	10
2	11
3	12
3	10
4	13
5	14
5	11
6	12
7	14

Clave Primaria {Trabajador, Responsable}

RESPONSABLES

RESPONSABLE	DEPARTAMENTO
10	Producción
11	Producción
12	Ventas
13	Producción
14	Ventas

Clave Primaria {Responsable}

En las formas de Boyce-Codd hay que tener cuidado al descomponer la tabla de partida en nuevas tablas ya que se podría perder información si no se hace bien.