NORMALIZACIÓN DE LAS BASES DE DATOS RELACIONALES

Al almacenar la información en la base de datos debemos evitar las redundancias e inconsistencias de forma que la información que obtengamos al consultarla esté libre de inconsistencias.

Para evitar inconsistencias en las bases de datos se definieron las formas normales, hay seis formas normales y dos adicionales aunque **normalmente con aplicar hasta la tercera es suficiente** ya que por las relaciones de los datos no es necesario aplicar formas normales superiores.

Cuanto mayor sea la forma normal de una tabla o una base de datos menos casos existirán de que contengan inconsistencias, una tabla que cumpla una forma normal cumple las formas normales de menor nivel. Las formas normales son las siguientes:

##### **Referencias**

* <https://picodotdev.github.io/blog-bitix/2018/02/las-6-plus-2-formas-normales-de-las-bases-de-datos-relacionales/>

# **Primera Forma Normal (1FN)**

La primera forma normal (1FN) es la usada en la normalización de bases de datos. Los criterios que tiene que satisfacer la tabla para adherirse a 1FN es que representa una relación y está libre de grupos repetitivos.

Según Date, una tabla está en 1FN si y sólo si es “isomorfa”[[1]](#footnote-0) a alguna relación. Es decir, satisface los 5 criterios que él propuso.

Estos son:

1. No hay orden de arriba-a-abajo en las filas.

2. No hay orden de izquierda-a-derecha en las columnas.

3. No hay filas duplicadas.

4. Cada intersección de fila-y-columna contiene exactamente un valor del dominio aplicable (y nada más).

5. Todas las columnas son regulares [es decir, las filas no tienen componentes como IDs de fila, IDs de objeto, o timestamps ocultos].

Si la tabla no cumple cualesquiera de estas condiciones significa que la tabla no es estrictamente relacional, por lo tanto no entra en 1FN.

En la imágen siguiente vemos como una tabla puede pasar a 1FN. La solución es crear una celda por cada número de teléfono.



# **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:**



Suponiendo que el DNI y el código 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:



##### 

##### **Referencias**

* <https://es.wikipedia.org/wiki/Primera_forma_normal>
* <https://es.wikipedia.org/wiki/Segunda_forma_normal>
* <https://www.cartagena99.com/recursos/programacion/apuntes/DisenioBBDD.pdf>

# **Tercera Forma Normal (3FN)**

Una relación se encuentra en Tercera Forma Normal si, y sólo si, se encuentra en 2FN y si los atributos no clave dependen de forma no transitiva de la clave primaria. (Reynosa E., Maldonado C., Muñoz R., Damiano L., Abrutsky M, 2012).

## Dependencia funcional transitiva

**Definición:** Supongamos que tenemos una relación con tres conjuntos de atributos: X, Y y Z, y las siguientes dependencias X -> Y, Y -> Z, Y -> |X. Es decir X determina Y e Y determina Z, pero Y no determina X. En ese caso, decimos que Z tiene dependencia transitiva con respecto a X, a través de Y.

La tercera forma normal consiste en eliminar las dependencias transitivas.

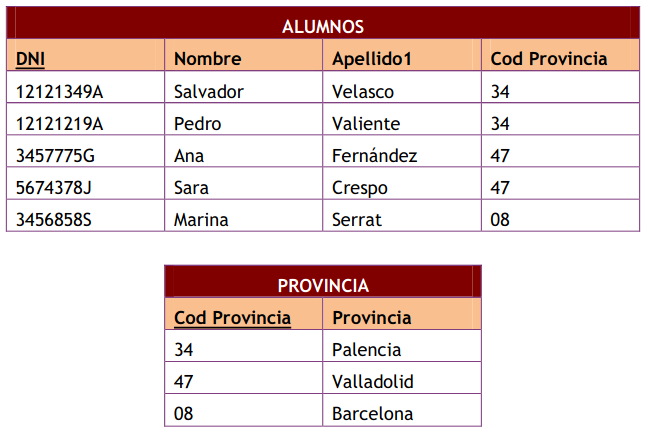
## 

## Transformación a Tercera Forma Normal (3FN)

Si se desea transformar una relación que no está en tercera forma normal, lo que debe hacerse es remover los atributos no clave que dependen de otros atributos no clave. Estos atributos removidos formarán parte de una nueva relación la cual tendrá como clave primaria al atributo del cual dependía el anterior en mención. De esta forma, cada atributo no clave de la relación es independiente al resto de atributos. La técnica implica mover los atributos en conflicto a una nueva relación.

##### Ejemplo.

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

****

La tercera forma normal resuelve los problemas causados por las dependencias funcionales entre atributos no clave. Busca que todas las claves no primarias de la relación no posean dependencias funcionales con otros atributos que tampoco son claves primarias.

**Debemos recordar** que en una relación en 3FN, **cada atributo que no es una clave debe depender de la clave y solamente de la clave**.

# **Forma Normal de Boyce-Codd (FNBC)**

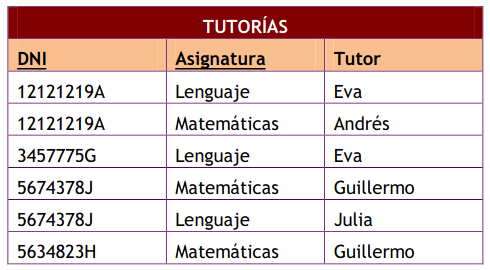
La **Forma Normal Boyce-Codd** (Denominada por sus siglas en inglés como BCNF o **FNBC**) es una forma normal utilizada en la normalización de bases de datos. Es una adaptación vagamente más segura de lo establecido en la Tercera Forma Normal (3FN).

Es más estricta que la 3FN, lo que significa que toda relación que esté en FNBC también está en 3FN; sin embargo, una relación 3FN no está necesariamente en FNBC.

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

Un determinante en una tabla de base de datos es un atributo que se puede utilizar para determinar los valores asignados a otros atributos en la misma línea. Según esta definición, cualquier clave primaria o clave candidata es un determinante, pero puede haber determinantes que no sean claves primarias o candidatas.

##### Ejemplo:



Esa tabla está en tercera forma normal (no hay dependencias transitivas), pero no en forma de Boyce -Codd, ya que (DNI, Asignatura) Tutory TutorAsignaturayTutor-/(DNI,Asignatura).

En este caso la redundancia ocurre por mala selección de clave. La redundancia de la asignatura es completamente evitable.

La solución sería:



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

## Clave candidata

Las claves **candidatas** serán aquel conjunto de atributos que identifiquen de manera única cada tupla (fila) de la relación (tabla). Es decir, las columnas cuyos valores no se repiten en ninguna otra fila de la tabla. Por tanto, cada tabla debe tener **al menos una clave candidata** aunque puede haber más de una.

Las claves candidatas pueden estar formadas por más de un atributo, siempre y cuando estos identifiquen de forma única a la fila. Cuando una clave candidata está formada por más de un atributo, se dice que es una **clave compuesta**.

**Requisitos:**

Una clave **candidata** debe cumplir los siguientes requisitos:

* **Unicidad**: no puede haber dos tuplas (filas) con los mismos valores para esos atributos.
* **Irreducibilidad**: si se elimina alguno de los atributos deja de ser única.

#### Ejemplo

Si elegimos como clave candidata **Nombre, Apellidos y F\_nacimiento**, cumple con la unicidad puesto que es muy difícil encontrarnos con dos personas que tengan el mismo nombre, apellidos y fecha de nacimiento iguales. Es irreducible puesto que sería posible encontrar dos personas con el mismo nombre y apellidos o con el mismo nombre y fecha de nacimiento, por lo que son necesarios los tres atributos (campos) para formar la clave.

La **clave primaria** de una relación es aquella clave candidata que se escoge para identificar sus tuplas de modo único.

**Las claves candidatas que no son escogidas como clave primaria** son denominadas **claves alternativas**.

##### **Referencias**

Oppel A. (2010). **Fundamentos de Bases de Datos**. Ed. McGraw-Hill. México.

Reynosa E., Maldonado C., Muñoz R., Damiano L., Abrutsky M. (2012). **Bases de datos**. Ed. Alfaomega.

Soler, J., Prados, F., Boada, I., & Poch, J. (2006). [Utilización de una plataforma de e-learning en la docencia de Bases de Datos](http://bioinfo.uib.es/~joemiro/aenui/procJenui/Jen2006/prDef0109_e369853df7.pdf). *Proc. of JENUI*, 581-588.

* <http://normalizacion-bd.blogspot.com/2012/08/5-tercera-forma-normal-3fn.html>
* <https://naps.com.mx/blog/tercera-forma-normal-en-el-modelo-relacional-3fn/>
* <https://www.cartagena99.com/recursos/programacion/apuntes/DisenioBBDD.pdf>
* <http://mysql.conclase.net/curso/?cap=004b>
* <https://ikastaroak.birt.eus/edu/argitalpen/backupa/20200331/1920k/es/DAMDAW/BD/BD01/es_DAMDAW_BD01_Contenidos/website_851_clave_candidata_clave_primaria_clave_alternativa.html>
* <https://tecnonautas.net/definicion-de-un-determinante-en-una-base-de-datos/>
* <https://sites.google.com/site/modelamientodebasesdedatos/clase-4>

# **Cuarta Forma Normal (4FN)**

La cuarta forma normal en una tabla tiene como requisito el cumplir con su predecesora la 3FN, si se cumple esta condición debemos observar que nuestra tabla no posea dependencias multivaluadas, es decir, donde la existencia de dos o más relaciones independientes muchos a muchos del modelo de datos causa redundancia; y es esta redundancia, la que es suprimida para cumplir con la cuarta forma.

#### Ejemplo

En esta tabla podemos observar que existe una dependencia multivaluada donde el atributo restaurante y el atributo variedad de pizza se repite desde dos a tres veces. por lo que este modelo tiene una tabla que guarda varias relaciones muchos a muchos lo que causaría algunas situaciones indeseables,

Cada vez que la pizzería añada una nueva variedad de pizza tendrá que generar varios registros, uno por cada área de envío.

Si añadimos una nueva área de envio, tendremos que añadir un registro por cada variedad de pizza que se venda

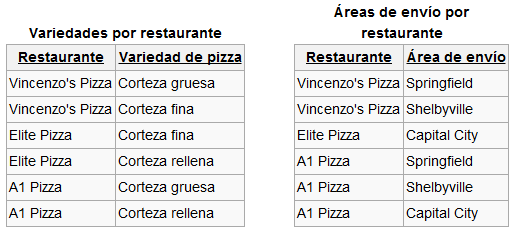
Como el caso de **A1 pizza y Corteza gruesa**

Para evitar esta redundancia de datos debemos realizar otra tabla.

Aunque **Restaurante, Variedad de pizza y Área de envío** no sean atributos primarios y podamos repetir datos en cada columna de igual forma, la cuarta regla nos dice que debemos evitar la redundancia.

Así que crearemos una nueva tabla, ambas con clave compuesta para este caso y sería una para variedad por restaurante y otra para el área de envío.

Esto facilita el que un restaurante añada una nueva variedad de pizza, únicamente añadiremos un nuevo dato a la primera tabla , y en caso de que tuviera una nueva área para enviar solo añadimos un registro a la segunda tabla, evitando de esta manera la redundancia. A pesar de tener que añadir una nueva tabla conseguimos que la consulta de los datos sea más clara.



**Referencias**

* <http://basededatos-jonathan-delatorre.blogspot.com/2012/04/la-cuarta-forma-normal-4fn.htm>
* l<http://dbadixit.com/cuarta-forma-normal-4fn/>
* <https://basedat0s.wordpress.com/2016/05/14/cuarta-forma-normal4fn/>

# **Quinta Forma Normal (5FN)**

También conocida como **Forma Normal de Proyección-Unión (PJ/NF)**, es un nivel de normalización de bases de datos diseñado para reducir redundancias en las bases de datos relacionales que guardan hechos multi-valores, aislando semánticamente relaciones múltiples relacionadas. Una tabla se dice que está en 5FN, si y sólo si, está en 4FN y cada dependencia de unión (join) en ella, es implicada por las claves candidatas.

La **5FN** se emplea cuando existe mucha información redundante en una tabla, o cuando se hace inmanejable, debido a la existencia de muchos atributos.

Por lo tanto una tabla está en 5FN

- Cuando satisface las condiciones de la 4FN

- Se aplica principalmente en la Proyección y en la Unión.

- No existen relaciones de dependencias de reunión (join) no triviales que no se generen desde las claves. Por lo que si se aplicara una consulta entre al menos tres relaciones independientes entre sí dentro de la 4FN y se obtuvieran tuplas espurias[[2]](#footnote-1), entonces no estaría dentro de la 5FN.

- La tabla no puede fragmentarse más sin que se pierda información

- En ocasiones, puede generar que se creen muchas tablas, lo que complica el manejo de las mismas

Solamente en contadas ocasiones una tabla 4NF no se corresponde con una 5NF. Estas son situaciones en las cuales una restricción compleja del mundo real, que limita las combinaciones válidas de los valores de atributos en la tabla 4NF, no está implícita en la estructura de esa tabla. Si esa tabla no se normaliza a 5NF, la tarea de mantener la consistencia lógica de los datos dentro de la tabla debe ser llevada en parte por la aplicación responsable de inserciones, borrados, y actualizaciones a ella; y hay un riesgo elevado de que los datos dentro de la tabla se vuelvan inconsistentes.

El diseño 5NF excluye la posibilidad de tales inconsistencias.

#### Ejemplo

Se indican los 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.

El proveedor número 1 nos suministró el material número 1 para el proyecto 2.

|  |  |  |
| --- | --- | --- |
| Proveedor | Material | Proyecto |
| 1 | 1 | 2 |
| 1 | 2 | 1 |
| 2 | 1 | 1 |

La dependencia que produce esta restricción es lejana y se denomina de reunión **(join).** Para esa restricción esta división en tablas sería válida.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Proveedor | Material |  | Material | Proyecto |
| 1 | 1 | 1 | 2 |
| 1 | 2 | 2 | 1 |
| 2 | 1 | 1 | 1 |

Eso ocurre en los datos como:

El proveedor número 1 nos suministró el material número 1 para el proyecto 2

En el proyecto 1 utilizamos el material 1

Por lo tanto aparecerá la tupla proveedor 1, material 1 y proyecto 1.

Esa descomposición no pierde valores, sabiendo que proveedor nos suministra un material podremos relacionarlo con todos los proyectos que utilizan ese material.

Resumiendo, una tabla no está en 5FN si hay una descomposición de esa tabla que muestre la misma información que la original.

Normalmente se crean tablas en 5FN cuando en la misma tabla hay muchos atributos y es casi inmanejable o cuando hay muchos registros y pocos atributos. En el caso de que haya muchos atributos se divide la tabla en dos, donde la clave es la misma en ambas tablas.

##### 

##### **Referencias**

* <https://es.wikipedia.org/wiki/Quinta_forma_normal>
* <http://academicos.azc.uam.mx/jfg/diapositivas/bases_datos/Unidad_4.pdf>
* [https://prezi.com\*La tabla está en 4FN](https://prezi.com/eov4nvponzzt/quinta-forma-normal-5fn/?frame=5abf51318cc8e3fa6785b6352528b6a46708e25e)
* <https://iutablog.wordpress.com/2016/04/26/cuarta-y-quinta-forma-normal-4fn-5fn/>
* <https://www.cartagena99.com/recursos/programacion/apuntes/DisenioBBDD.pdf>
* <http://elmerfloresxia123.blogspot.com/2017/08/normalizacion-boyce-codd4-y-5-fn.html>

# **Sexta Forma Normal (6FN)**

La sexta forma de normalización es muy reciente. Fue presentada a finales de la década de los años 90 por Christopher J. Date. En esta normalización la variable de relación se descompone hasta componentes irreductibles. Una base de datos cumplirá con la sexta forma de normalización si satisface los siguientes criterios:

1. Que cumpla con la 5NF (Quinta Forma Normal)

2. Que cada dependencia de la relación sea trivial, una dependencia funcional A -> B es trivial cuando B es parte de A. Esto sucede cuando A es un conjunto de atributos, y B es a su vez un subconjunto de A.

La sexta forma normal (6NF), en pocas palabras, se basa en el principio de que si se tiene más de dos claves candidatas en una tabla, se tendrán que crear otras tablas con estas.

La sexta forma normal no es muy utilizada porque genera más tablas cuando tenemos pequeñas bases de datos.

Entonces, de una tabla en 5NF, 6NF produce dos tablas.

#### Ejemplo

#### **TABLA 1 (2FN)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Nombre médico** | **Ocupación** | **Tipo** | **Práctica en años** |
| Smith James | ortopédico | especialista | 23 |
| Miller Michael | ortopédico | persona a prueba | 4 |
| Thomas Linda | neurólogo | persona a prueba | 5 |
| Scott Nancy | ortopédico | residente | 1 |
| Allen Brian | neurólogo | especialista | 12 |
| Turner Steven | oftalmólogo | persona a prueba | 3 |
| Collins Kevin | oftalmólogo | especialista | 7 |
| Rey donald | neurólogo | residente | 1 |
| Harris Sarah | oftalmólogo | residente | 2 |

Las dependencias de unión de la tabla son {nombre médico, ocupación}, {nombre médico, práctica en años} y {nombre médico, tipo}.

Por lo tanto, pudimos ver que dicha tabla es 2NF (debido a la aparición de dependencia transitiva). Las siguientes tablas intentan llevarlo a 6NF:

##### **TABLA 2.1**

|  |  |
| --- | --- |
| **Nombre médico** | **Ocupación** |
| Smith James | ortopédico |
| Miller Michael | ortopédico |
| Thomas Linda | neurólogo |
| Scott Nancy | ortopédico |
| Allen Brian | neurólogo |
| Turner Steven | oftalmólogo |
| Collins Kevin | oftalmólogo |
| Rey donald | neurólogo |
| Harris Sarah | oftalmólogo |

##### **TABLA 2.2**

|  |  |
| --- | --- |
| **Nombre médico** | **Practica en años** |
| Smith James | 23 |
| Miller Michael | 4 |
| Thomas Linda | 5 |
| Scott Nancy | 1 |
| Allen Brian | 12 |
| Turner Steven | 3 |
| Collins Kevin | 7 |
| Rey donald | 1 |
| Harris Sarah | 2 |

##### **TABLA 2.3**

|  |  |
| --- | --- |
| **Nombre médico** | **Tipo** |
| Smith James | especialista |
| Miller Michael | persona a prueba |
| Thomas Linda | persona a prueba |
| Scott Nancy | residente |
| Allen Brian | especialista |
| Turner Steven | persona a prueba |
| Collins Kevin | especialista |
| Rey donald | residente |
| Harris Sarah | residente |

##### 

##### **Referencias**

* https://en.wikipedia.org/wiki/Sixth\_normal\_form
* https://cristiansalinas08.blogspot.com/2019/09/normalizaicon-de-las-bases-de-datos-lahtml

# **Forma Normal de Dominio/Clave DNFN**

La forma normal de dominio/clave (DKNF) es una forma normal usada en normalización de bases de datos que requiere que la base de datos contenga restricciones de dominios y de claves.

Una restricción del dominio especifica los valores permitidos para un atributo dado, mientras que una restricción clave especifica los atributos que identifican únicamente una fila en una tabla dada.

Es mucho más fácil construir una base de datos en forma normal de dominio/clave que convertir pequeñas bases de datos que puedan contener numerosas anomalías. Sin embargo, construir con éxito una base de datos en forma normal de dominio/clave sigue siendo una tarea difícil, incluso para programadores experimentados de bases de datos. Así, mientras que la forma normal de dominio/clave elimina los problemas encontrados en la mayoría de las bases de datos, tiende a ser la forma normal más costosa de alcanzar. Sin embargo, el no poder alcanzar la forma normal de dominio/clave puede llevar costos ocultos a largo plazo, debido a anomalías que aparecen con el tiempo en las bases de datos que solamente se adhieren a formas normales más bajas.

Una violación de DKNF ocurre en la siguiente tabla:

****

Asuma que el dominio para la DNI Persona rica consiste en los DNI's de toda la gente rica en una muestra predefinida de gente rica; el dominio para el Tipo de persona rica consiste de los valores 'Millonario excéntrico', 'Multimillonario excéntrico', 'Millonario malvado', y 'Multimillonario malvado'; y el dominio para el Valor neto en dólares consiste de todos los números enteros mayor que o igual a 1.000.000.

Hay una restricción que liga el Tipo de persona rica al Valor neto en dólares, incluso aunque no podamos deducir uno del otro. La restricción dicta que un Millonario excéntrico o Millonario malvado tendrá un valor neto de 1.000.000 a 999.999.999 inclusive, mientras que un Multimillonario excéntrico o un Multimillonario malvado tendrá un valor neto de 1.000.000.000 o más. Esta restricción no es ni una restricción de dominio ni una restricción de clave; por lo tanto no podemos confiar en las restricciones de dominio y las de clave para garantizar que una combinación de anómala de Tipo de persona rica / Valor neto en dólares no tenga cabida en la base de datos.

La violación de la DKNF podría ser eliminada alterando dominio Tipo de persona rica para hacer que sea consistente con solo dos valores, 'Malvado' y 'Excéntrico' (el estatus de persona rica como un millonario o un multimillonario es implícito en su Valor neto en dólares, así que no se pierde ninguna información útil).

**Desnormalización**

La desnormalización es una estrategia utilizada en una base de datos previamente normalizada para aumentar su rendimiento. La idea detrás de ella es agregar datos redundantes donde pensamos que nos ayudarán más. Podemos utilizar atributos adicionales en una tabla existente, agregar nuevas tablas o incluso crear instancias de tablas existentes. El objetivo habitual es disminuir el tiempo de ejecución de las consultas haciendo que los datos sean más accesibles generando informes resumidos en tablas separadas.

Normalizar datos es el punto de partida para el proceso de desnormalización. Es importante diferenciar una base datos que no ha sido normalizada de la base de datos que se normaliza primero y luego se desnormaliza. La segunda está bien y la primera es a menudo el resultado del mal diseño de una base de datos o de una falta de conocimiento.

Es necesario estar seguro de por qué se desea aplicar la desnormalización. Necesitas también estar seguro de que el beneficio de usarlo supera cualquier daño. Hay algunas situaciones en las que se debería pensar en desnormalización:

* Mantenimiento de un histórico: los datos pueden cambiar con el tiempo y tenemos que almacenar valores que eran válidos cuando se creó un registro. El nombre y apellido de una persona puede cambiar, un cliente también puede cambiar su nombre comercial o cualquier otro dato. Los detalles de ciertas consultas deben contener valores que eran reales en el momento en que se generaron. No seríamos capaces de recrear los datos del pasado correctamente si esto no es así. Podríamos resolver este problema agregando una tabla que contiene el historial de estos cambios. Pero en este caso, una consulta podría ser muy complicada.
* Mejorar el rendimiento de las consultas: algunas consultas pueden utilizar varias tablas para acceder a los datos que con frecuencia necesitamos. Piensa en una situación en la que tuviéramos que unir 10 tablas para devolver el nombre de un cliente y los productos que se vendieron. Algunas tablas también podían contener grandes cantidades de datos.
* Aceleración de presentación de informes: si necesitamos algunas estadísticas con mucha frecuencia, crearlas a partir de datos en vivo requiere mucho tiempo y puede afectar al rendimiento general del sistema.
* Tener precalculados valores que utilizamos con frecuencia: algunos valores es posible que queramos tenerlos listos para no tener que generarlos con datos en vivo.

##### 

##### 

##### 

##### **Referencias**

* <https://www.informaticaparatunegocio.com/blog/normalizar-datos-algunas-veces-se-desnormalizan/>
* <https://es.wikipedia.org/wiki/Denormalizaci%C3%B3n_(base_de_datos)>
* ttps://picodotdev.github.io/blog-bitix/2018/02/las-6-plus-2-formas-normales-de-las-bases-de-datos-relacionales/
* <http://mysql.conclase.net/curso/?cap=004a>

# RESUMEN Y PRÁCTICA

La Normalización nos va a servir para optimizar las bases de datos, ocupar el menor espacio posible, eliminar datos repetidos, eliminar errores lógicos y mantener los datos ordenados.

Tenemos: Primera Forma Normal (1FN), Segunda Forma Normal (2FN), Tercera Forma Normal (3FN), Forma Normal Boyce Codd, Cuarta Forma Normal (4FN) y Quinta Forma Normal (5FN). Las Bases de Datos suelen normalizarse hasta la 3FN, porque en esta forma ya tenemos una base de datos bien estructurada, sin datos repetidos , con el espacio optimizado y sin pérdida de información.

#### **PRIMERA FORMA NORMAL (1FN)**

Una tabla está en Primera Forma Normal si:

* Todos los atributos son atómicos. (Un atributo es atómico si los elementos del dominio son indivisibles, mínimos).
* La tabla contiene una clave primaria única.
* La clave primaria no contiene atributos nulos.
* No debe existir variación en el número de columnas.
* Los Campos no clave deben identificarse por la clave (Dependencia Funcional)
* Debe Existir una independencia del orden tanto de las filas como de las columnas, es decir, si los datos cambian de orden no deben cambiar sus significados
* Una tabla no puede tener múltiples valores en cada columna.

La tabla “Estudiantes” no estaría en 1FN ya que tiene campos multivaluados.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tabla Estudiantes** | | | | |
| **DNI** | **Cod\_curso** | **Nombre** | **Apellido** | **Nota** |
| 33473259I | UF1473 | Luis | Perez | 10 |
| 4525938K | MF0225  CT0310 | Angel | Martínez | 5  6 |
| 72354982S | UC0225 | Angela | Saavedra | 7 |
| 6594327G | UF1473  CT0310 | Imelda | Sanchez | 8  7 |
| 35728294T | MF0225 | Pedro | Garcia | 5 |
| 94287658V | CT0310 | Jose | Rodriguez | 6 |
| 65879431A | MF0225  UF1473 | Pedro | Picapiedra | 10  8 |

Para pasarla a 1FN tendríamos que eliminar esos campos creando nuevas filas:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **1FN** | | | | |
| **DNI** | **Cod\_curso** | **Nombre** | **Apellido** | **Nota** |
| 33473259I | UF1473 | Luis | Perez | 10 |
| 4525938K | CT0310 | Angel | Martínez | 6 |
| 4525938K | MF0225 | Angel | Martínez | 5 |
| 72354982S | UC0225 | Angela | Saavedra | 7 |
| 6594327G | CT0310 | Imelda | Sanchez | 7 |
| 6594327G | UF1473 | Imelda | Sanchez | 8 |
| 35728294T | MF0225 | Pedro | Garcia | 5 |
| 94287658V | CT0310 | Jose | Rodriguez | 6 |
| 65879431A | UF1473 | Pedro | Picapiedra | 8 |
| 65879431A | MF0225 | Pedro | Picapiedra | 10 |

#### **SEGUNDA FORMA NORMAL (2FN)**

Una relación está en 2FN si está en 1FN y si los atributos que no forman parte de ninguna clave dependen de forma completa de la clave principal. Es decir que no existen dependencias parciales. (Todos los atributos que no son clave principal deben depender únicamente de la clave principal).

La clave principal (PK) sería DNI y los demás atributos dependerían de la clave principal.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **2FN** | | |  | **2FN** | | |
| **DNI** | **Nombre** | **Apellido** |  | **DNI** | **Cod\_curso** | **Nota** |
| 33473259I | Luis | Perez |  | 33473259I | UF1473 | 10 |
| 4525938K | Angel | Martínez |  | 4525938K | CT0310 | 6 |
| 72354982S | Angela | Saavedra |  | 4525938K | MF0225 | 5 |
| 6594327G | Imelda | Sanchez |  | 72354982S | UC0225 | 7 |
| 35728294T | Pedro | Garcia |  | 6594327G | CT0310 | 7 |
| 94287658V | Jose | Rodriguez |  | 6594327G | UF1473 | 8 |
| 65879431A | Pedro | Picapiedra |  | 35728294T | MF0225 | 5 |
|  | | |  | 94287658V | CT0310 | 6 |
|  | 65879431A | UF1473 | 8 |
|  | 65879431A | MF0225 | 10 |

#### **TERCERA FORMA NORMAL(3FN)**

La tabla se encuentra en 3FN si es 2FN y si no existe ninguna dependencia funcional transitiva entre los atributos que no son clave.

En la tabla “Estudiantes II”, vemos que existe una dependencia funcional entre **Nombre\_curso** y **código\_curso**, por lo tanto no estaría en 3FN.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Tabla Estudiantes II** | | | | | | |
| **DNI** | **Cod\_curso** | **Nombre\_curso** | | | **Nombre** | **Apellido** |
| 33473259I | UF1473 | Salvaguarda y seguridad de los datos | | | Luis | Perez |
| 4525938K | CT0310 | Administración de bases de datos | | | Angel | Martínez |
| 72354982S | UC0225 | Configuración de bases de datos | | | Angela | Saavedra |
| 6594327G | CT0310 | Administración de bases de datos | | | Imelda | Sanchez |
| 35728294T | MF0225 | Gestión de bases de datos | | | Pedro | Garcia |
| 94287658V | CT0310 | Administración de bases de datos | | | Jose | Rodriguez |
| 65879431A | UF1473 | Salvaguarda y seguridad de los datos | | | Pedro | Picapiedra |

Para pasarlo a 3FN, lo arreglariamos creando una nueva tabla. No quedando así, ninguna dependencia funcional entre atributos que no son clave.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Tabla Estudiantes II** | | | |  | **Tabla Cursos** | | | |
| **DNI** | **Cod\_curso** | **Nombre** | **Apellido** |  | **Cod\_curso** | **Nombre\_curso** | | |
| 33473259I | UF1473 | Luis | Perez |  | CT0310 | Administración de bases de datos | | |
| 4525938K | CT0310 | Angel | Martínez |  | MF0225 | Gestión de bases de datos | | |
| 72354982S | UC0225 | Angela | Saavedra |  | UC0225 | Configuración de bases de datos | | |
| 6594327G | CT0310 | Imelda | Sanchez |  | UF1473 | Salvaguarda y seguridad de los datos | | |
| 35728294T | MF0225 | Pedro | Garcia |  |  |  |  |  |
| 94287658V | CT0310 | Jose | Rodriguez |  |  |  |  |  |
| 65879431A | UF1473 | Pedro | Picapiedra |  |  |  |  |  |

#### **FORMA NORMAL DE BOYCE-CODD (FNBC)**

Una vez que llegamos a la 3FN, en algunos casos **muy inusuales** aún puede haber redundancias. En casos donde hay más de una llave candidata, y estas llaves tienen atributos en común

La tabla se encuentra en FNBC si cada determinante, atributo que determina completamente a otro, es clave candidata.

|  |  |  |  |
| --- | --- | --- | --- |
| **Tabla Estudiantes III** | | | |
| **Alumno** | **Nombre\_curso** | | **Profesor** |
| 35728294T | Salvaguarda y seguridad de los datos | | Carlos |
| 65879431A | Administración de bases de datos | | Angeles |
| 35728294T | Configuración de bases de datos | | Gerardo |
| 6594327G | Administración de bases de datos | | Angeles |
| 35728294T | Gestión de bases de datos | | Juan |
| 94287658V | Administración de bases de datos | | Gerardo |
| 65879431A | Salvaguarda y seguridad de los datos | | Carlos |

En la tabla superior vemos que en el mismo curso se inscriben diferentes alumnos, y de la misma manera el mismo curso puede tener distintos profesores. Entonces, necesitamos el alumno y el curso para poder determinar el profesor. Entonces tendríamos como llaves candidatas Alumno\_Curso y Alumno\_Profesor.

El problema surge si por ejemplo el alumno 35728294T se retira del curso “Gestión de bases de datos”, entonces perdemos la información del profesor “Juan”. También podría suceder que llegará un nuevo profesor, por ejemplo “María”, a dar el curso de “Programación”, sólo podríamos ingresarla cuando tuviéramos un alumno matriculado para ese curso.

Esto violaría las leyes de FNBC, ya que el atributo “profesor” no sería una llave de la relación.

Cómo lo solucionamos:

Aquí no perderíamos información en caso de modificaciones de inserción o eliminación.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Tutorías** | |  | **CursoTutor** | | |
| **Alumno** | **Profesor** |  | **Nombre\_curso** | | **Profesor** |
| 35728294T | Carlos |  | Salvaguarda y seguridad de los datos | | Carlos |
| 65879431A | Angeles |  | Administración de bases de datos | | Angeles |
| 35728294T | Gerardo |  | Configuración de bases de datos | | Gerardo |
| 6594327G | Angeles |  | Gestión de bases de datos | | Juan |
| 35728294T | Juan |  | Administración de bases de datos | | Gerardo |
| 94287658V | Gerardo |  |  |  |  |
| 65879431A | Carlos |  |  |  |  |

#### **CUARTA FORMA NORMAL(4FN)**

Una tabla está en 4FN si: se encuentra en FNBC y para cualquier combinación clave -campo no existen valores duplicados.

Tenemos la tabla:

|  |  |  |
| --- | --- | --- |
| **Profesor** | **Curso** | **Centro** |
| Gerardo | Salvaguarda y seguridad de los datos | GNueve\_Santiago |
| Gerardo | Administración de bases de datos | GNueve\_Santiago |
| Gerardo | Configuración de bases de datos | GNueve\_Santiago |
| Juan | Salvaguarda y seguridad de los datos | GDoce\_Santiago |
| Gerardo | Gestión de bases de datos | GDoce\_Santiago |
| Angeles | Gestión de bases de datos | GSiete\_Santiago |
| Juan | Configuración de bases de datos | GDoce\_Santiago |
| Juan | Configuración de bases de datos | GNueve\_Santiago |
| Carlos | Salvaguarda y seguridad de los datos | GDoce\_Santiago |
| Angeles | Administración de bases de datos | GCinco\_Santiago |

Aunque **Profesor, Curso y Centro** no sean atributos primarios y podamos repetir datos en cada columna de igual forma, la cuarta regla nos dice que debemos evitar la redundancia.

Esto facilita que si a un profesor se le añade un nuevo curso, únicamente añadiremos un nuevo dato a la primera tabla , y en caso de que tuviera un nuevo centro para dar cursos solo añadimos un registro a la segunda tabla, evitando de esta manera la redundancia.

A pesar de tener que añadir una nueva tabla conseguimos que la consulta de los datos sea más clara.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Profesor** | **Centro** |  | **Profesor** | **Curso** |
| Gerardo | GNueve\_Santiago |  | Gerardo | Salvaguarda y seguridad de los datos |
| Juan | GDoce\_Santiago |  | Gerardo | Administración de bases de datos |
| Gerardo | GDoce\_Santiago |  | Gerardo | Configuración de bases de datos |
| Angeles | GSiete\_Santiago |  | Juan | Salvaguarda y seguridad de los datos |
| Juan | GNueve\_Santiago |  | Gerardo | Gestión de bases de datos |
| Carlos | GDoce\_Santiago |  | Angeles | Gestión de bases de datos |
| Angeles | GCinco\_Santiago |  | Juan | Configuración de bases de datos |
|  |  |  | Carlos | Salvaguarda y seguridad de los datos |
|  |  |  | Angeles | Administración de bases de datos |

#### **DESNORMALIZACIÓN**

Tenemos la siguiente tabla que estaría normalizada.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| idEstudiante | Nombre | Apellido | eMail | Telefono |
| 1 | Juan | Perez | juan@gmail.com | 666999333 |
| 2 | Pepe | Martinez | pepe@gmail.com | 636555222 |
| 3 | Maria | López | maria@gmail.com | 656123321 |

Pero ahora tenemos que añadir a cada estudiante un nuevo número de teléfono:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| idEstudiante | Nombre | Apellido | eMail | Telefono | Telefono2 |
| 1 | Juan | Perez | juan@gmail.com | 666999333 | 981565656 |
| 2 | Pepe | Martinez | pepe@gmail.com | 636555222 | 988656565 |
| 3 | Maria | López | maria@gmail.com | 656123321 | 982363636 |

Así romperemos con la primera forma normal, tendríamos dos columnas para “Teléfono”. Pero en la práctica, si sabemos que no vamos a añadir nuevos campos para teléfono, puede resultar más cómodo que crear una nueva tabla. Estamos desnormalizando conscientemente.

1. El concepto matemático de isomorfismo pretende captar la idea de tener la misma estructura. [↑](#footnote-ref-0)
2. Tuplas espurias (erroneas) [↑](#footnote-ref-1)