

Módulo 4. Normalización de Datos

Definición.

Es un proceso por medio del cual se logra un nivel de eficiencia en el diseño de una base de datos, acorde con los requerimientos de procesamiento del usuario.

Hasta ahora, con las reglas de traducción vistas en clase y aplicándolas correctamente al modelo entidad relación, sabemos que llegamos a un diseño de bases de datos. Y hasta ahora sabemos que dicha base de datos a la cual se llega es correcta. Pero no sabemos el por qué.....ese es el objetivo del tema de normalización de datos. Entender por qué la base de datos resultante de la aplicación de las reglas de traducción es eficiente.

Existen varios grados de eficiencia en el diseño de una base de datos. Estos grados de eficiencia se llaman formas normales. Existen seis formas normales:

- Primera Forma Normal (1NF)
- Segunda Forma Normal (2NF)
- Tercera Forma Normal (3NF)
- Forma Normal de Boyce/Codd (BCNF)
- Cuarta Forma Normal (4NF)
- Quinta Forma Normal (5NF)

El menor grado de eficiencia es la NO NORMALIZACION, es decir, el diseño de base de datos que ni siquiera está en Primera Forma Normal. El mayor grado de eficiencia es la QUINTA FORMA NORMAL.

Cabe aclarar que el 98% de las bases de datos empresariales trabajan satisfactoriamente estando en 3NF. En otras palabras, el 98% de las veces una base de datos que está en 3NF cumple con los requerimientos del usuario.

Esa es la razón por la cual en este curso de bases de datos se ve hasta la tercera forma normal (3NF). Las formas normales superiores (BCNF, 4NF, 5NF) quedan como tema de consulta.

Posibles anomalías en el diseño de una base de datos.

Al diseñar una base de datos, existe la posibilidad de que ésta quede con tres (3) tipos de anomalías (o defectos) que existen. Las posibles anomalías que existen son las siguientes:

- Anomalía de Inserción

- Anomalía de Actualización
- Anomalía de Borrado

Cabe aclarar que las anomalías las tienen individualmente las tablas que componen la base de datos.

Anomalía de Inserción.

Es una situación anormal que se presenta en una tabla al INSERTAR nuevas tuplas en ella y que es causada por el diseño de la tabla.

La consecuencia de este tipo de anomalía es que la tabla va a tener REDUNDANCIA DE DATOS.

Anomalía de Actualización.

Es una situación anormal que se presenta en una tabla al ACTUALIZAR tuplas ya existentes y que es causada por el diseño de la tabla.

La consecuencia de este tipo de anomalía es una posible INCONSISTENCIA FUTURA DE LOS DATOS.

Anomalía de Borrado.

Es una situación anormal que se presenta en una tabla al BORRAR tuplas de ella y que es causada por el diseño de la tabla.

La consecuencia de este tipo de anomalía es una posible PERDIDA O BORRADO NO DESEADO DE DATOS.

Ejemplo.

Para entender los tipos de anomalías existentes, suponga el esquema de tabla siguiente:

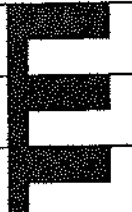
VEHICULO = { placa, marca, modelo, capacidad, codoficinatransito, ciudadoficina, nroempleados }

Esta tabla pretende almacenar los datos de los vehículos junto con los datos de la oficina de tránsito donde está matriculado.

Los atributos de la tabla VEHICULO son la placa del carro, la marca, modelo y capacidad del carro, el código de la oficina de tránsito donde está matriculado, la ciudad donde queda dicha oficina y el número de empleados que tiene la oficina de tránsito.

Para saber si una tabla tiene anomalías de inserción, la pregunta que se debe hacer es "EN QUÉ SITUACIONES SE PRESENTA REDUNDANCIA DE DATOS AL INSERTARLE NUEVAS TUPLAS A LA TABLA?"

Si analizamos bien la tabla, nos damos cuenta de que los datos de la misma pudieran ser los siguientes:

PLACA	MARCA	MODELO	CAPACIDAD	CODIFICINA	CIUDAD	NROEMP
KJH564	RENAULT	2004	5	10	BELLO	45
PEF786	MAZDA	2008	4	10	BELLO	45
DER876	BMW	2010	3	10	BELLO	45
HJN983	MAZDA	2000	4	20		
HBF909	TOYOTA	2011	6	20		
GTF501	AUDI	2011	4	20		

Como se puede ver claramente esta tabla presenta alta redundancia de datos cuando hay varios vehículos matriculados en una misma oficina de tránsito, situación ésta que es completamente normal en la vida real. Se ve que en los atributos CIUDAD y NROEMP se está presentando redundancia de datos a medida que se insertan vehículos de una misma oficina de tránsito.

Por lo tanto, la tabla **VEHICULO TIENE ANOMALÍAS DE INSERCIÓN.**

Para saber si una tabla tiene anomalías de actualización, nos hacemos la siguiente pregunta: "AL ACTUALIZAR ALGÚN DATO DE UNA TUPLA (REGISTRO), HAY POSIBILIDAD DE QUE LA TABLA QUEDE CON DATOS INCONSISTENTES?"

Si miramos los datos de la tabla VEHICULO, y suponemos que la oficina de tránsito de BELLO ha aumentado su número de empleados de 45 a 50, entonces debemos reflejar este aumento en la base de datos. A cuántas tuplas de la tabla VEHICULO hay que actualizarle el número de empleados? A las que tienen la

oficina de tránsito de BELLO, es decir, se deben modificar tres (3) tuplas. Qué sucede si al tratar de actualizar esas tres (3) tuplas, sucede algo que solo deja modificar dos (2) de las tres(3) tuplas? La tabla quedaría de la siguiente manera:

PLACA	MARCA	MODEL O	CAPACIDA D	CODOFICIN A	CIUDA D	NROEM P
KJH564	RENAUL T	2004	5	10	BELLO	50
PEF786	MAZDA	2008	4	10	BELLO	50
DER876	BMW	2010	3	10	BELLO	50
HJN983	MAZDA	2000	4	20	CALDA S	33
HBF909	TOYOTA	2011	6	20	CALDA S	33
GTF501	AUDI	2011	4	20	CALDA S	33

Como se puede observar, la oficina de tránsito de Bello queda con dos números (o cantidades) distintas de empleados, lo cual no tiene sentido. En otras palabras, los datos de la tabla VEHICULO han quedado inconsistentes. Esta inconsistencia es producida por la redundancia que tiene dicha tabla. La pregunta que se hace uno es: "por qué si necesito actualizar el número de empleados de UNA sola oficina de tránsito, tengo que modificar tres(3) tuplas y no solamente una?"

Por lo tanto, la tabla **VEHICULO TIENE ANOMALÍAS DE ACTUALIZACIÓN** porque hay posibilidades de quedar los datos inconsistentes al actualizar ciertos atributos (campos) de la tabla.

Para saber si una tabla tiene anomalías de borrado, la pregunta que se debe hacer es "EN QUÉ SITUACIONES, AL BORRAR ALGUNA TUPLA DE LA TABLA, SE ME PODRÍA BORRAR DATOS QUE NO ESPERABA QUE SE BORRARAN?"

Suponga que existe una oficina de tránsito que solo tiene un vehículo matriculado (es un caso muy raro en la vida real pero se puede dar en la tabla diseñada). Es decir, suponga lo siguiente:

PLACA	MARCA	MODELO	CAPACIDAD	CODOFICINA	CIUDAD	NROEMP
KJH564	RENAULT	2004	5	10	BELLO	50
PEF78	MAZDA	2008	4	10	BELLO	50

6						
DER87 6	BMW	2010	3	10	BELLO	50
HJN98 3	MAZDA	2000	4	20	CALDAS	33
HB90 9	TOYOTA	2011	6	20	CALDAS	33
GTF50 1	AUDI	2011	4	20	CALDAS	33
JUH56 0	HYUNDAI	2009	5	30	BOGOT A	76

Como se puede ver, en la oficina de tránsito de BOGOTÁ solo está matriculado el vehículo de placas JUH560. Suponga que el vehículo en cuestión se estrella y se da pérdida total de él. Es decir, hay que retirarlo de la base de datos. Al borrar la tupla correspondiente a dicho vehículo, también se está borrando la información sobre la oficina de tránsito de Bogotá (código de oficina, nombre y número de empleados). Y de dicha oficina no quedará rastro en la base de datos (por el hecho de que en esa oficina NO HAY MAS carros matriculados). Al querer borrar el vehículo, sin querer, se me han borrado los datos de la oficina de tránsito a la que pertenece el carro. Es decir, hubo una pérdida o borrado NO DESEADO de datos. Por qué si deseo borrar los datos de un vehículo se me tienen que borrar los datos de la oficina de tránsito también?

Por lo tanto, la tabla **VEHÍCULO TIENE ANOMALÍAS DE BORRADO.**

Conclusión. La tabla VEHÍCULO tiene los tres (3) tipos de anomalías. Es decir, está MUY MAL DISEÑADA.

REGLA No.1 Para que una tabla esté bien diseñada, NO PUEDE TENER NINGÚN TIPO DE ANOMALÍA.

Ejercicio.

Suponga la siguiente tabla:

FACTURAPROD = {nrofactura, codproducto, cantidad, fechafactura, valortotalfactura,

Nombreproducto, precioproducto }

El objetivo de la tabla es almacenar el detalle de todas las facturas generadas en una empresa. Los atributos de la tabla son el número de la factura, el código del producto que está dentro de la factura, la cantidad facturada de dicho producto en

dicha factura, la fecha de generación de la factura, el valor total de la factura, el nombre del producto facturado y el precio unitario del producto facturado.

Un ejemplo de los datos de dicha tabla podrían ser los siguientes:

NROFACT	CODPRODUCTO	CANTIDAD	FECHAFACT	VALORTOTAL	NOMBREPRODUCTO	PRECIOUNITARIO
100	10	45	12/12/2005	450000	JABON	500
100	20	23	12/12/2005	450000	PERA	430
100	30	24	12/12/2005	450000	GALLETAS	760
100	40	10	12/12/2005	450000	MARGARINA	600
200	20	67	23/05/2001	760000	PERA	430
200	40	5	23/05/2001	760000	MARGARINA	600
200	70	8	23/05/2001	760000	PLATANO	440
300	10	40	31/12/2010	340000	JABON	500
300	30	11	31/12/2010	340000	GALLETAS	760

Analizar y justificar las posibles anomalías que pudiera tener la tabla anterior.

Tuplas Espurias.

Antes que nada, el sinónimo más claro de ESPURIA es BASURA, es decir, estamos hablando de TUPLAS BASURA.

Para explicar este concepto, como casi siempre lo he hecho, partiré de un ejemplo.

Suponga el siguiente par de tablas correspondientes a una serie de competiciones deportivas.

CAMPEONATO = { código-campeonato, deporte-campeonato, ciudad-campeonato }

La tabla CAMPEONATO almacena un conjunto de campeonatos hechos, cada campeonato de un solo deporte. Y cada campeonato fue realizado en una sola ciudad.

Por lo tanto, un ejemplo de los datos de esta tabla podrían ser los siguientes:

Código-campeonato	Deporte-campeonato	Ciudad-campeonato
10	Fútbol	Cali
20	Tejo	Bogotá
30	Basketbol	Bogotá
40	Voleibol	Medellín
50	Tenis	Medellín

DEPORTISTA = {cedula-dep, nombre-dep, edad-dep, ciudad-compitió }

La tabla DEPORTISTA almacena los diferentes deportistas que participaron en los campeonatos. Almacena su cédula, nombre, edad y ciudad donde compitió en un campeonato. Es de anotar que estamos suponiendo **que CADA DEPORTISTA PRACTICA UN SOLO DEPORTE, NO VARIOS.**

Los datos de esta tabla podrían ser los siguientes:

Cedula-dep	Nombre-dep	Edad-dep	Ciudad-compitió
1000	Juan Hernández	25	Bogotá
2000	Lina Carrillo	21	Cali
3000	Hugo Torres	29	Medellín
4000	Carolina Bernal	20	Medellín
5000	Tatiana Calle	22	Bogotá
6000	Víctor Lopez	28	Cali

Analicen muy bien los datos de ambas tablas para que respondan las siguientes preguntas:

- Qué deporte practicó la deportista Lina Carrillo? **Sin duda**, practicó el fútbol.
- Qué deporte practicó el deportista Víctor López? También, **sin duda**, el fútbol.

Ahora respondamos las siguientes preguntas:

- Qué deporte practicó el deportista Hugo Torres? No lo podemos afirmar con seguridad. Para la respuesta se tienen dos opciones.....Que practicó

voleibol o tenis. (Recuerde que dijimos que un deportista solo practica un deporte). Entonces, cuál deporte practicó Hugo Torres?

- Qué deporte practicó Carolina Bernal? Tenemos la misma duda que con Hugo Torres. Pudo haber sido voleibol o tenis.
- Qué deporte practicó Juan Hernández? Tenemos la duda.....pudo haber sido el tejo o el basketbol.
- Qué deporte practicó Tatiana Calle? Igual respuesta que Juan Hernández.....o tejo o basketbol.

Recuerden el concepto de INNER JOIN entre dos tablas.....el INNER JOIN me genera como resultado la concatenación de los registros relacionados por CAMPOS EN COMUN.

Cuáles son los campos en común entre estas dos tablas? El nombre de la ciudad.

Y para saber un deportista qué deporte practicó, hay que hacer un INNER JOIN entre estas dos tablas.

Al hacer el INNER JOIN entre estas dos tablas obtenemos los siguientes datos:

Cedula-dep	Nombre-dep	Edad-dep	Ciudad-compitió	Codigo-Campeonato	Deporte	Ciudad
1000	Juan Hernández	25	Bogotá	20	Tejo	Bogotá
1000	Juan Hernández	25	Bogotá	30	Basketbol	Bogotá
2000	Lina Carrillo	21	Cali	10	Fútbol	Cali
3000	Hugo Torres	29	Medellín	40	Voleibol	Medellín
3000	Hugo Torres	29	Medellín	50	Tenis	Medellín
4000	Carolina Bernal	20	Medellín	40	Voleibol	Medellín
4000	Carolina Bernal	20	Medellín	50	Tenis	Medellín
5000	Tatiana Calle	22	Bogotá	20	Tejo	Bogotá
5000	Tatiana Calle	22	Bogotá	30	Basketbol	Bogotá
6000	Víctor López	28	Cali	10	Fútbol	Cali

Teniendo en cuenta el resultado anterior, vamos a suponer que, **EN LA REALIDAD**, sucedió lo siguiente:

- Juan Hernández practicó Basketbol.
- Hugo Torres practicó Tenis.
- Carolina Bernal practicó Voleibol.
- Tatiana Calle practicó Tejo.

Por lo tanto, en la tabla resultado anterior hay unos resultados "mentirosos" que me están dando información que no es verdad. Estos resultados están dados en forma de tuplas (todos sabemos que el resultado de cualquier consulta, inclusive un INNER JOIN, es un conjunto de tuplas).

Estos resultados mentirosos están resaltados de color amarillo. Las seis tuplas que no están sombreadas son verídicas, es decir, me están relacionando correctamente a cada deportista con el deporte o campeonato donde compitió.

Las restantes cuatro (4) tuplas, las que están sombreadas de amarillo, SON LAS TUPLAS ESPURIAS.

Recuerde que dijimos que espuria es lo mismo que basura. Estas cuatro tuplas son tuplas basura ya que no me están reflejando la realidad (Carolina Bernal no practicó el tenis, Juan Hernández no practicó el tejo, Hugo Torres no practicó el voleibol ni Tatiana Calle el basketbol).

Cual fue la situación que hizo generar TUPLAS ESPURIAS al hacerle un INNER JOIN a estas dos tablas? Cuál fue el error cometido en el diseño de las tablas? Hemos dicho que dos tablas se relacionan por campos en común. Estas dos tablas tienen campos en común. Pero también hemos dicho que dos tablas se relacionan por la clave foránea de una y la clave primaria de la otra.....en otras palabras, uno de los campos en común tiene que ser clave primaria de una de las tablas.

CAMPEONATO = {código-campeonato, deporte-campeonato, nombre-deportista }

DEPORTISTA = {cedula-dep, nombre-dep, edad-dep, nombre-deportista }

Como podemos ver, ambas tablas **sí tienen campos en común** (los que están de verde) **pero ninguno de los campos en común es clave primaria de alguna tabla** (las claves primarias están en negrilla y subrayadas). Es decir, no cumple que la relación se tiene que dar por clave foránea de una tabla y clave primaria de la otra.

Cuando dos tablas se relacionan entre si violando la condición anterior, hay posibilidad de que generen TUPLAS ESPURIAS, lo cual es totalmente INDESEABLE para el usuario final. Una base de datos no le puede entregar información "mentirosa" al usuario.

Regla No. 2. Dos tablas siempre se deben relacionar por la clave foránea de una con la clave primaria de la otra. Si esto no sucede, hay posibilidad de generar TUPLAS ESPURIAS. O dicho de otra manera, el INNER JOIN entre dos tablas NUNCA puede dar TUPLAS ESPURIAS.

Ejercicio.

Escribir el esquema de dos tablas que pudieran dar TUPLAS ESPURIAS al hacerles un INNER JOIN. Coja el tema que desee (FARMACIA, BANCO, UNIVERSIDAD, etc.)

Justifique por qué dicho esquema produce tuplas espurias.

Dependencia Funcional.

Antes de iniciar el tema de dependencia funcional, cabe advertir que éste es el concepto **más importante** del capítulo. De él dependen los conceptos de formas normales que, como ya sabemos, es la razón de ser del tema de normalización.

Siempre he pensado que dar la definición formal de dependencia funcional enreda un poco al estudiante. Por lo tanto, aquí una vez más, me valdré de ejemplos para explicarles el concepto.

Pero antes de eso, necesito contextualizar el concepto como tal, dónde se utiliza, cuándo, etc. Para eso es importante entender los siguientes comentarios técnicos acerca de una dependencia funcional:

- Las dependencias funcionales se analizan en **cada tabla individual**, sin importar si la tabla tiene relación con otras tablas. En otras palabras, para analizar las dependencias funcionales en una base de datos, coja cada una de las tablas por separado y les analiza este aspecto.
- Dentro de cada tabla, las dependencias funcionales se analizan entre los atributos (o campos) de la tabla. Así que si existe el siguiente esquema de una tabla....

Tabla = { campo1, campo2, campo3, campo4 }

En dicha tabla se analizan las dependencias funcionales entre campo1 y campo2, entre campo1 y campo3, entre campo2 y campo4, etc. Inclusive, se pueden analizar dependencias funcionales entre parejas de campos (o trios, o cuartetos, etc.) con otros campos. Por ejemplo, dependencias funcionales entre campo1, campo2 con campo3.

Acá lo importante es volver a resaltar que las dependencias funcionales se analizan entre los campos de **UNA sola tabla**. No tiene sentido analizar las dependencias funcionales entre dos campos de tablas distintas.

- La forma de escribir una dependencia funcional es la siguiente:

Campo1 → Campo2

Y esta dependencia se puede leer de dos formas distintas:

Campo2 **DEPENDE FUNCIONALMENTE** de Campo1 ó
Campo1 **DETERMINA FUNCIONALMENTE** a Campo2

Cualquiera de las dos maneras que se lea, significa lo mismo y se escribe de la manera mostrada arriba.

Que es Dependencia Funcional?

Para explicar el concepto, vamos a usar dos ejemplos. Un primer ejemplo, abstracto pero muy visual, y un segundo ejemplo mas practico y cercano a la vida real.

Ejemplo 1. (Abstracto pero muy visual)

Suponga que existe una tabla que tiene cuatro (4) atributos (A, B, C, D). Vamos a suponer, para la explicación del concepto, que dicha tabla tiene ocho (8) tuplas, las cuales se muestran a continuación:

TABLA			
A	B	C	D
A1	B1	C1	D1
A1	B1	C2	D1
A1	B1	C2	D1
A2	B1	C3	D2
A2	B1	C3	D3
A3	B6	C4	D4
A4	B8	C5	D5
A4	B8	C5	D5

Vamos a analizar la siguiente dependencia funcional: $A \rightarrow B$. En otras palabras, vamos a preguntarnos lo siguiente: "El atributo B depende funcionalmente del atributo A?" o "El atributo A determina funcionalmente al atributo B?"

Para responder esta pregunta, cogemos el atributo que hay a la izquierda de la flecha (A) y armamos grupos con el mismo dato en dicho atributo, tal y como se muestra a continuación:

TABLA			
A	B	C	D
A1	B1	C1	D1
A1	B1	C2	D1
A1	B1	C2	D1
A2	B1	C3	D2
A2	B1	C3	D3
A2	B6	C4	D4
A2	B8	C5	D5
A2	B8	C5	D5

Como se puede observar se pudieron armar cuatro grupos. Ahora cogemos el atributo de

la derecha de la flecha (B) y le armamos LOS MISMOS GRUPOS que le armamos al atributo de la izquierda (A), tal y como se muestra a continuación:

TABLA			
A	B	C	D
A1	B1	C1	D1
A1	B1	C2	D1
A1	B1	C2	D1
A2	B1	C3	D2
A2	B1	C3	D3
A2	B6	C4	D4
A2	B8	C5	D5
A2	B8	C5	D5

Y en **cada grupo de datos construido** en el atributo de la izquierda de la flecha (A) hacemos el siguiente análisis: "En ese grupo, los valores del atributo B cambian?"

Ya dijimos que se construyeron cuatro (4) grupos. Cojamos el primer grupo (amarillo) y nos damos cuenta de que en esas tres (3) tuplas el valor de B no cambia, se mantiene constante en B1.

Segundo grupo (verde), los valores de B se mantienen constante: B1.

Tercer grupo (azul), los valores de B no cambian: B6 (Sobra decir que en los grupos que contienen una sola tupla, no es necesario hacer el análisis ya que es obvio que el valor del atributo de la derecha de la flecha no varía).

Cuarto grupo (rosado), los valores de B se mantienen inalterables: B8.

Si en **TODA** la tabla se da la propiedad de que en **cada grupo construido**, los valores del atributo de la derecha se mantienen fijos, inalterables, son iguales, entonces podemos concluir que el atributo de la derecha de la flecha **DEPENDE FUNCIONALMENTE** del atributo que está a la izquierda de la flecha.

Por lo tanto, para este caso podemos afirmar que $A \rightarrow B$

Ejercicios:

En la tabla anterior, analizar las siguientes posibles dependencias funcionales:

- $A \rightarrow C$
- $A \rightarrow D$
- $B \rightarrow A$
- $B \rightarrow C$
- $B \rightarrow D$
- $C \rightarrow A$
- $C \rightarrow B$
- $C \rightarrow D$
- $D \rightarrow A$
- $D \rightarrow B$
- $D \rightarrow C$
- $CD \rightarrow B$ (Aca se concatenan los valores de C y D y se procede igual)
- $AB \rightarrow C$

Ejemplo 2. (Práctico y cercano a la realidad)

Suponga la siguiente tabla de empleados:

EMPLEADO = { cedula, nombre, edad, teléfono, sueldo, dirección }

Analicemos la siguiente dependencia funcional:

nombre \rightarrow edad

Antes de analizar esta dependencia, cabe aclarar que se podría pensar que para darle respuesta a dicha dependencia tendríamos que tener a la mano los datos de las 5000 posibles tuplas que pudiera tener dicha tabla. Que tendríamos que escribir los datos de las 5000 tuplas de dicha tabla, tal y como lo hicimos con las ocho (8) tuplas del ejemplo anterior. Sería muy engorroso. No es necesario. **Lo importante es comprender muy bien que significan cada uno de los datos que hay en cada uno de los atributos de la tabla.** Con eso es suficiente.

Retomemos la dependencia a analizar. Debemos armar grupos de tuplas con el mismo valor del atributo de la izquierda de la flecha (nombre). Ya armamos los mismos grupos de tuplas del atributo de la derecha (edad). En cada grupo los valores de edad se conservan? No cambian?

Mire el siguiente caso:

EMPLEADO					
CEDULA	NOMBRE	EDAD	TELEFONO	SUELDO	DIRECCION
23090	JUAN PEREZ	45	6739078	4,500,000	CALLE 16
45978	JUAN PEREZ	23	3985643	3,400,000	CRA 56
72366	JUAN PEREZ	19	2546754	1,200,000	CALLE 10
78562	ANA ZEA	34	3985643	3,900,000	CRA 56
45876	LINA PAEZ	19	7908567	3,400,000	CALLE 16

En el grupo representado de amarillo, el grupo de JUAN PEREZ, la edad CAMBIA! Rompe con la propiedad que venimos diciendo. Por lo tanto, la dependencia funcional

nombre → edad no se cumple, es decir, la edad NO depende funcionalmente del nombre del empleado. (Tenga en cuenta que pueden haber varios empleados con el mismo nombre y, obvio, diferente edad).

Ahora analicemos **telefono → sueldo**. Sera cierta esta dependencia?

Cuantos grupos de teléfonos distintos podemos armar? Cuatro. Armandos esos mismos cuatro (4) grupos en sueldo, nos damos cuenta que para el grupo del teléfono 3985643 hay dos (2) tuplas y en esas tuplas se tienen sueldos distintos (3,400,000 y 3,900,000). **Por lo tanto, el atributo sueldo NO DEPENDE FUNCIONALMENTE del atributo teléfono.** En otras palabras, podemos concluir que pueden existir dos empleados con el mismo teléfono pero diferente sueldo...y esta es razón suficiente para concluir la no existencia de dicha dependencia.

Ejercicios:

Analizar las siguientes dependencias:

- cedula → sueldo
- sueldo → direccion
- direccion → nombre
- nombre → direccion
- cedula → edad

En resumen, escriba las UNICAS dependencias funcionales que se dan en dicha tabla y escriba una conclusión. (Esta conclusión se convertirá en la regla No. 3 de normalización de datos).

Formas Normales.**Definición.**

Son grados de eficiencia del diseño de las tablas de una base de datos. Dichos grados de eficiencia se toman en base a los requerimientos del usuario que es, en primera instancia, el determinante de cómo debe quedar construida la base de datos.

Existen seis (6) formas normales:

- 1NF (Primera Forma Normal)
- 2NF (Segunda Forma Normal)
- 3NF (Tercera Forma Normal)
- BCNF (Forma Normal de Boyce / Codd)
- 4NF (Cuarta Forma Normal)
- 5NF (Quinta Forma Normal)

Es de aclarar que la base de datos con menor grado de eficiencia es la que NO ESTÁ NORMALIZADA. Es decir, la que no se encuentra ni siquiera en 1NF. La base de datos de mayor grado de eficiencia es la que está en 5NF.

Como ya se dijo anteriormente, el 98 % de las bases de datos empresariales se encuentran, como máximo, en 3NF. En otras, palabras el 98 % de las veces, una base de datos que está en 3NF cumple con los requerimientos funcionales del usuario. Por lo tanto, en este curso se dicta hasta 3NF. Las formas normales superiores (BCNF, 4NF, 5NF) son para situaciones muy específicas y poco comunes y quedan como consulta.

Para dar comienzo a la explicación de las formas normales, no sobra recordar que es fundamental tener muy claro el concepto de Dependencia Funcional visto en el documento anterior.

Durante la explicación de las formas normales, vamos a partir del siguiente esquema de tablas:

Deportista = { cédula, nombre, edad, ciudad-competencia, numero-medallas }

EmpProy = { cod-empleado, cod-proyecto, numero-horas-semanales, nombre-empleado,

Titulo-proyecto, valor-proyecto, edad-empleado }

Ciudad = { cod-ciudad, nombre-ciudad, nro-habitantes, cod-pais, nombre-pais, área-pais }

Cabe aclarar que cada una de las tablas anteriores no tienen nada que ver la una con la otra. Son tres (3) tablas totalmente independientes entre sí.

Sí analizamos las tres (3) tablas anteriores, todas tienen anomalías, es decir, están mal diseñadas. Por lo tanto, es bueno recalcar que la explicación siguiente, parte de tablas mal diseñadas, explica por qué están mal diseñadas y sus implicaciones, define la forma normal y convierte el diseño de la tabla a dicha forma normal.

PRIMERA FORMA NORMAL (1NF)

Trabajemos con la tabla Deportista enunciada anteriormente. Dicha tabla podría tener los siguientes datos:

DEPORTISTA				
Cédula	Nombre	Edad	Ciudad-Compet.	Número-Medalla
2500	Juan Villa	25	Bogotá Medellín Cali	12
3600	Lina Jaramillo	19	Medellín Bello	6
5800	Gregorio Torres	22	Bogotá Fusagasugá	3
4800	Ana Vélez	18	Cali	1
8500	Patricia Calle	26	Bogotá Medellín	3

La tabla Deportista contiene la cédula del deportista, su nombre, edad, número total de medallas obtenidas y las ciudades donde ha competido el deportista.

Hay que concluir que la tabla Deportista **NO ESTÁ EN 1NF** ya que el campo Ciudad-Competencia es **MULTIVALORADO**. Es decir, la tabla Deportista se dice que **NO ESTÁ NORMALIZADA**, no tiene el mínimo grado de eficiencia en su diseño.

Definición 1NF.

Una tabla está en 1NF **si TODOS sus atributos son univalorados (atómicos)**. En otras palabras, una tabla está en 1NF si **NO TIENE atributos multivalorados**.

Como se puede ver, en la tabla Deportista, el atributo Ciudad-Competencia es multivalorado, es decir, puede tener varias ciudades dentro de una misma tupla. Por lo tanto, no está en 1NF.

La pregunta que surge es: **“Qué tiene de malo tener atributos multivalorados en una tabla?”**

Suponga que una de las consultas más usada por el usuario es: **“Listar los nombres de deportistas que han competido en Medellín.”** Para este caso, la consulta se dificulta mucho porque habría que recorrer TODAS las tuplas y en cada una de ellas DESCOMPONER los valores del atributo Ciudad-Competencia hasta obtener las distintas ciudades que hay. Y de ahí preguntar si una de ellas es Medellín. Esta descomposición de la que se habla acá debe ser hecha a través de programación, es decir, se hace desde la aplicación que está accediendo a la base de datos. Como se puede observar, este campo multivalorado genera una dependencia no deseable entre la base de datos y la aplicación. Esa es la razón por la cual no es deseable tener atributos multivalorados en las tablas.

Importante.

Solo es deseable convertir un atributo multivalorado en univalorado en el caso de que se vayan a hacer consultas sobre la tabla por ese atributo. Si, de antemano, se sabe que sobre un atributo multivalorado nunca se van a hacer consultas, podría optar por dejarse la tabla **NO NORMALIZADA**.

Para normalizar la tabla Deportista, existen dos posibilidades:

- Con redundancia de datos
- Sin redundancia de datos

Con redundancia de datos.

Suponga los mismos datos de la tabla Deportista pero grabados de la siguiente manera:

DEPORTISTA				
Cédula	Nombre	Edad	Ciudad-Compet.	Número-Medalla
2500	Juan Villa	25	Bogotá	12
2500	Juan Villa	25	Medellín	12
2500	Juan Villa	25	Cali	12
3600	Lina Jaramillo	19	Medellín	6
3600	Lina Jaramillo	19	Bello	6
5800	Gregorio Torres	22	Bogotá	3
5800	Gregorio Torres	22	Fusagasugá	3
4800	Ana Vélez	18	Cali	1
8500	Patricia Calle	26	Bogotá	3
8500	Patricia Calle	26	Medellín	3

La estructura de la tabla se conserva pero hay una tupla por cada ciudad donde haya participado el deportista. Esta tabla está en 1NF CON REDUNDANCIA de información, es decir, tiene el primer grado de eficiencia, pero todavía tiene problemas de diseño.

Sin redundancia de datos.

Para entender este caso, que es el que mejor resuelve el problema, recuerde la regla de traducción vista para el atributo multivalorado. Algo le va a parecer familiar en esta explicación.

La otra manera de convertir la tabla Deportista a 1NF, sugiere partir la tabla original en las siguientes dos tablas:

Deportista1 = { cédula, nombre, edad, número-medallas }

CiudadDep = { ced-deportista , ciudad-competencia }

Ced-deportista referencia a Deportista1(cédula)

Los datos correspondientes a estas dos tablas corresponden a los datos que se quieren almacenar originalmente en la tabla Deportista pero cumpliendo con la 1NF, es decir, ningún atributo es multivalorado.

Los datos de ambas tablas serían los siguientes:

DEPORTISTA1			
Cédula	Nombre	Edad	Número-Medallas
2500	Juan Villa	25	12
3600	Lina Jaramillo	19	6
5800	Gregorio Torres	22	3
4800	Ana Vélez	18	1
8500	Patricia Calle	26	3

CIUDADDEP	
Ced-deportista	Ciudad-Competencia
2500	Bogotá
2500	Medellín
2500	Cali
3600	Medellín
3600	Bello
5800	Bogotá
5800	Fusagasugá
4800	Cali
8500	Bogotá
8500	Medellín

Como se puede observar, con las tablas anteriores sabremos los datos de cada deportista y en qué ciudades ha competido. Pero la diferencia es que los atributos de las tablas son univalorados. Por lo tanto, las dos tablas (Deportista1 y CiudadDep) están en 1NF. (De hecho, también están en 2NF y 3NF).

SEGUNDA FORMA NORMAL (2NF)

Para explicar la segunda forma normal, mostremos los posibles datos de la tabla EmpProy mostrada anteriormente:

EMPPROY						
Cod-emp	Cod-proy	Nro-horas	Nom-empl	Titulo-pro	Valor-pro	Edad-emp
10	1	52	Jorge Zea	ABD	520000	33
10	2	33	Jorge Zea	NHY	360000	33
10	3	5	Jorge	LOK	690000	33

			Zea			
20	2	12	Lina Torres	NHY	360000	20
20	4	2	Lina Torres	HIL	500000	20
30	3	11	Juan Lopez	LOK	690000	44
40	1	12	Ana Perez	ABD	520000	19
40	4	15	Ana Perez	HIL	500000	19

La tabla EmpProy tiene la intención de almacenar la información de los diferentes proyectos en los cuales está trabajando cada empleado de una empresa. Un empleado puede trabajar en varios proyectos y en un proyecto pueden estar trabajando varios empleados. Cabe aclarar que en el tercer atributo (numero-horas-semanales) se graba el número de horas semanales que trabaja un empleado en un proyecto dado.

Es indudable que la tabla EmpProy tiene anomalías. Eso se nota por la alta redundancia que tiene la tabla.

Por la definición de la primera forma normal (1NF) podemos concluir que la tabla **EmpProy SÍ ESTÁ EN 1NF**, ya que no tiene atributos multivalorados. Ahora la pregunta que cabe hacerse es si la tabla está en 2NF. Para responder a dicha pregunta, debemos analizar las dependencias funcionales que existen en dicha tabla:

DF1 : cod-empleado → nombre-empleado

DF2 : cod-empleado → edad-empleado

DF3 : cod-proyecto → título-proyecto

DF4 : cod-proyecto → valor-proyecto

DF5: cod-empleado, cod-proyecto → numero-horas-semanales

(Las dependencias funcionales anteriores ya deben comprenderse en su totalidad. Si no es así, estudie de nuevo el documento anterior sobre Dependencia Funcional).

Hay que recalcar que la clave primaria de la tabla EmpProy es compuesta por los atributos cod-empleado y cod-proyecto.

Analizando las cinco (5) dependencias funcionales que tiene la tabla, qué característica tiene la dependencia DF5 que no tengan las cuatro (4) anteriores (DF1, DF2, DF3 y DF4)? En DF5 el atributo de la derecha de la flecha depende funcionalmente de **TODA LA CLAVE PRIMARIA DE LA TABLA**. En cambio, en DF1, DF2, DF3 y DF4 el atributo del lado derecho de la flecha depende funcionalmente de **UNA PARTE DE LA CLAVE PRIMARIA**.

Por lo tanto, podemos afirmar que la tabla EmpProy **NO ESTÁ EN 2NF** ya que hay dependencias funcionales que incluyen PARTE de la clave primaria.

Definición 2NF.

Una tabla está en 2NF si cumple las siguientes dos (2) condiciones:

- Está en 1NF
- Todo atributo NO CLAVE depende funcionalmente, **EN FORMA TOTAL, NO PARCIAL**, de la clave primaria

Para explicar cómo se normaliza la tabla EmpProy, tenga muy en mente la regla de traducción de la relación muchos a muchos y de la agregación.

Si pensamos de qué porción de diagrama entidad relación vienen los atributos de la tabla EmpProy, nos damos cuenta que vienen de dos entidades relacionadas a través de una relación MUCHOS A MUCHOS: Empleado y Proyecto.

Por lo tanto, la normalización de la tabla EmpProy a 2NF se hace generando tres (3) tablas que son las siguientes:

Empleado = { cod-empleado, nombre-empleado, edad-empleado }

Proyecto = { cod-proyecto, título-proyecto, valor-proyecto }

ProyEmp = { cod-emp, cod-proy, numero-horas-semanales }

Cod-emp referencia a Empleado(cod-empleado)

Cod-proy referencia a Proyecto(cod-proyecto)

Las tres tablas anteriores están en 1NF y también en 2NF, ya que todo atributo no clave depende funcionalmente de la clave primaria **COMPLETA**, no de parte de la clave. (De hecho, también están en 3NF).

Los datos presentados en la tabla EmpProy quedarían de la siguiente manera en las tablas ya normalizadas:

EMPLEADO		
Cod-empleado	Nombre-empleado	Edad-empleado
10	Jorge Zea	33
20	Lina Torres	20
30	Juan Lopez	44
40	Ana Perez	19

PROYECTO		
Cod-proyecto	Título-proyecto	Valor-proyecto
1	ABD	520000
2	NHY	360000
3	LOK	690000
4	HIL	500000

PROYEMP		
Cod-emp	Cod-proy	Numero-horas-semanales
10	1	52
10	2	33
10	3	5
20	2	12
20	4	2
30	3	11
40	1	12
40	4	15

Como se puede apreciar, al normalizar la tabla EmpProy convirtiéndola en las tres (3) tablas anteriores, ya no existen anomalías de diseño. Y si no se ha percatado todavía de algo, nos podemos dar cuenta que lo que se hizo fue aplicar la regla de traducción de la relación muchos a muchos y de la agregación vistas en el capítulo pasado.

TERCERA FORMA NORMAL (3NF)

Para la explicación de la tercera forma normal (3NF), miremos los posibles datos de la tabla Ciudad enunciada al principio de este documento:

CIUDAD					
Cod-ciudad	Nombre-ciu	Nro-habitan.	Cod-pais	Nombre-pais	Área-pais
10	Medellín	2500000	1	Colombia	36000000
20	Cali	1800000	1	Colombia	36000000
30	Pereira	520000	1	Colombia	36000000
40	New York	52000000	2	EEUU	56000000
50	Boston	25600000	2	EEUU	56000000
60	Quito	4500000	3	Ecuador	5280000
70	Guayaquil	3600000	3	Ecuador	5280000

La tabla se refiere a las ciudades del mundo junto con el país donde se encuentran.

Vamos a analizar las dependencias funcionales que existen en dicha tabla:

- DF1 : cod-ciudad → nombre-ciu
- DF2 : cod-ciudad → nro-habitantes
- DF3 : cod-ciudad → cod-pais
- DF4 : cod- país → nombre-pais
- DF5 : cod-ciudad → nombre-pais
- DF6 : cod-pais → área-pais
- DF7 : cod-ciudad → área-pais

(Las dependencias funcionales anteriores ya deben comprenderse en su totalidad. Si no es así, estudie de nuevo el documento anterior sobre Dependencia Funcional).

Si se entendió la segunda forma normal (2NF), es claro que la tabla Ciudad **está en 2NF** (De hecho, también está en 1NF, ya que no tiene atributos multivalorados). Pero **NO ESTÁ EN 3NF**. Expliquemos por qué no está en 3NF.

De las dependencias anteriores, escojamos DF3, DF4 y DF5. Si se analizan bien esas tres (3) dependencias funcionales, no damos cuenta de que existe **TRANSITIVIDAD**. Es decir, nombre-pais depende de cod-ciudad porque cod-pais depende de cod-ciudad y nombre-pais depende de cod-pais. (Recuerden lo que alguna vez aprendimos en matemáticas: Si A se relaciona con B; y B se relaciona con C; entonces A se relaciona con C. Propiedad transitiva. Lo mismo está sucediendo en estas tres (3) dependencias funcionales. El atributo nombre-pais depende funcionalmente de cod-ciudad en forma transitiva).

Por esta transitividad y por la que hay entre DF3, DF6 y DF7, la tabla Ciudad NO ESTÁ EN 3NF.

Para poder corregir el diseño de esta tabla, pensemos en la regla de traducción de la relación uno a muchos (de hecho, entre Ciudad y Pais, en el modelo entidad relación, hay una relación uno a muchos).

La tabla Ciudad la vamos a convertir a 3NF, separándola en las siguientes dos (2) tablas:

Definición 3NF.

Una tabla está en 3NF cuando cumple las siguientes dos (2) condiciones:

- Está en 2NF
- Todo atributo no clave depende funcionalmente, en forma directa, no transitiva, de la clave primaria. (En otras palabras, cuando las únicas dependencias funcionales que existen son entre los atributos no claves y la clave primaria).

Ciudad1 = { Codigo-ciu, Nombre-ciu, Nro-habitantes, Cod-pais }

Cod-pais referencia a Pais(cod-pais)

Pais = { cod-pais, nombre-pais, área-pais }

Los datos quedan así:

CIUDAD1			
Codigo-ciu	Nombre-ciu	Nro-habitantes	Cod-pais
10	Medellín	2500000	1
20	Cali	1800000	1
30	Pereira	520000	1
40	New York	52000000	2
50	Boston	25600000	2
60	Quito	4500000	3
70	Guayaquil	3600000	3

PAIS		
Cod-pais	Nombre-pais	Area-pais
1	Colombia	36000000
2	EEUU	56000000
3	Ecuador	5280000

Las tablas Ciudad1 y Pais reflejan la misma información que se quería almacenar en la tabla original Ciudad. La diferencia es que estas dos últimas tablas NO tienen anomalías de ningún tipo. Ambas tablas están en 3NF porque, en cada tabla, cada atributo no clave depende funcionalmente solamente de la clave primaria.

Conclusión importante.

En la explicación dada en este documento se justifica claramente por qué cuando a un modelo entidad relación se le aplican las reglas de traducción vistas en clase, se llega a una base de datos en 3NF, es decir, bien diseñada, sin ningún tipo de anomalía.

