Modelo entidad-relación (Parte II)

Normalización

Competencias

- Implementar técnicas de normalización para reducir redundancia entre tablas
- Conocer las primeras tres formas normales
- Conocer la importancia de la desnormalización

Introducción

La normalización es un proceso que elimina la redundancia de una base de datos. El proceso consta de una serie de pasos en las cuales se van eliminando distintos tipos de redundancias con el propósito de prevenir inconsistencias. Estas etapas son muy importantes y reciben el nombre de formas normales. En este capítulo aprenderemos a llevar una base de datos a tercera forma normal.

¿Para qué sirve la normalización?

La normalización es una fórmula que podemos aplicar a una base de datos, a un modelo lógico o incluso a una tabla de excel o archivo con formato .csv que nos entreguen para crear finalmente una nueva base de datos sin redundancia.

Comúnmente se entiende por "normalizar una base de datos" pasarla a tercera forma normal. Sin embargo en cada etapa lo que estamos haciendo es normalizar.

En resumen, la normalización busca:

- Evitar la redundancia de datos
- Simplificar la actualización de datos
- Garantizar integridad referencial (prevenir inconsistencia en las relaciones)

Para pasar una base de datos de un etapa a la otra tenemos que aplicar un conjunto de reglas, por lo cual definiremos términos inicialmente para luego entrar a normalizar.

Notación

Utilizaremos la siguiente notación para detallar la normalización de una tabla:

> nombre_tabla(atributo_1, atributo_2, ..., atributo_n)

- Fuera de los paréntesis tenemos el nombre de la tabla con la que trabajaremos.
- Dentro de los paréntesis se encuentran los atributos existentes.
- Hay casos en que dentro de una tabla pueden existir atributos repetidos. Para denotarlos de forma explícita, los encapsularemos entre llaves {}
  de la siguiente forma:

> nombre_tabla(atributo_1, atributo_2, ..., atributo_n, {atributo_repetido_1, ..., atributo_repetido_j})

¿Qué es un grupo repetitivo?

Lo más fácil es verlo con un ejemplo, tenemos la siguiente tabla Cliente.

N Cliente	Nombre	Teléfonos
1	Fernanda	123-456 123-457
2	Felipe	234-412
3	Francisco	123-411

En este caso, puede haber más de un teléfono asociado a cada cliente y no solo uno o dos, pueden haber cientos, no sabemos. Por eso es un atributo (o grupo de atributos) repetitivo, lo anterior es muy similar a:

N Cliente	Nombre	Teléfono1	Teléfono2
1	Fernanda	123-456	123-457
2	Felipe	234-412	
3	Francisco	123-411	

En ambos casos los atributos se repiten. Ahora si supieramos que solo existe la posibilidad que el Cliente tenga 2 teléfonos, no sería un grupo repetitivo. El problema es cuando no conocemos realmente la cantidad de columnas como para poder almacenar los datos.

Utilizando la notación aprendida podríamos representar nuestra tabla como:

> Cliente(N Cliente, Nombre, {Teléfono})

Identificando las claves primarias y foráneas

Para identificar la **clave primaria** dentro de una tabla, utilizaremos el símbolo **#** y letras en negrita.

Para representar **claves foráneas** dentro de una tabla, las identificaremos con letras cursivas.

Cuando estemos frente a claves primarias compuestas, todos los atributos correspondientes a claves serán encapsulados entre negritas.

Ejemplo de clave primaria en una tabla:

> Tabla1(#**clavePrimaria**, atributo_1, ..., atributo_n)

Ejemplo de clave primaria y foránea en una tabla:

> Tabla2(#**clavePrimaria**, *claveForánea*, ..., atributo_n)

Ejemplo de clave primaria compuesta:

> Tabla3(#**clavePrimaria**, #**clavePrimariayForánea**, ..., atributo_n)

Ahora que conocemos las notaciones, podemos conocer las formas normales.

Implementación de formas normales

Para ejemplificar el uso de las formas normales, utilizaremos el siguiente ejemplo, que corresponde a la factura de un paciente en un hospital, donde se muestran los datos del paciente y los items que consumió.

Factura		# 23464 20/04/2019
Nombre paciente: Marta Fuentes	#Paciente: 4724	
Dirección: Monjitas 245	Comuna: Santiago	
COD-SISTEMA-SALUD: 10	Ciudad: Santiago	
ITEM	NOMBRE	VALOR
200	Pieza semiprivada	150
204	Televisión	10
245	Rayos X	25
414	Exámenes	35
	SUBTOTAL	220
	%IMPUESTO	22
	TOTAL	242

Imagen 1.Ejemplo factura.

Antes de empezar a normalizar debemos analizar la información que se nos presenta e identificar cuales son los atributos y grupos repetitivos en caso de exisitir

> Factura(numero_factura, fecha_factura, nombre_paciente, numero_paciente, direccion_paciente, comuna_paciente, ciudad_paciente, codigo_sistema_salud, {numero_item, nombre_item,valor_item}, ~~subtotal~~, ~~impuesto~~, ~~total~~)

Subtotal, impuesto y total podemos eliminarlos ya que estos atributos son derivables, es decir, los podemos calcular.

Primera Forma Normal (1FN)

Para que una tabla se encuentre normalizada acorde a la Primera Forma Normal (1FN), la tabla debe cumplir las siguientes condiciones:

- Cada campo debe ser atómico, es decir, debe contener un único valor.
- No pueden haber grupos repetitivos.

Como observamos en nuestro ejemplo, tenemos un grupo repetitivo de {numero_item, nombre_item,valor_item}

Por lo que a partir de acá crearemos una nueva tabla, resultando:

> FacturaItem(#numero_factura,#*numero_item*, nombre_item, valor_item)

donde la nueva tabla, llamada FacturaItem contiene a todos los atributos del grupo repetitivo, y se crea una clave primaria compuesta con el numero_factura y numero_item.

A la tabla factura se le elimina el grupo repetitivo, quedando:

> Factura(#numero_factura, fecha_factura, nombre_paciente, numero_paciente, direccion_paciente, comuna_paciente, ciudad_paciente, codigo_sistema_salud)

El resultado de esta operación la podemos ver gráficamente de la siguiente manera

Factura	# 23464 20/04/2019
Nombre paciente: Marta Fuentes	#Paciente: 4724
Dirección: Monjitas 245	Comuna: Santiago
COD-SISTEMA-SALUD: 10	Ciudad: Santiago

#Factura	ITEM	NOMBRE	VALOR
23464	200	Pieza semiprivada	150
23464	204	Televisión	10
23464	245	Rayos X	25
23464	414	Exámenes	35

Imagen 2. Eliminación de grupos repetitivos.

Cada grupo repetitivo se deja como una nueva tabla, manteniendo la clave de la cual provienen. Normalmente esta tabla tendrá una clave primaria compuesta por la **clave primaria de la tabla original y el atributo del cual dependen los demás atributos** del grupo repetitivo.

Segunda Forma Normal (2FN)

Esta forma debe cumplir las siguientes condiciones:

- Debe satisfacer la 1FN
- Cada atributo debe depender de la clave primaria, y no solo una parte de ella.
- Los atributos que dependan de manera parcial de la clave primaria deben ser eliminados o almacenados en una nueva entidad.

Para pasar una tabla a esta forma debemos fijarnos en las tablas que tengan claves primarias compuestas, porque solo en ellas se pueden dar dependencias parciales. En otras palabras, si una entidad depende de sólo una parte de la clave primaria compuesta, se deberá eliminar de esa tabla y llevarla a una nueva con la clave primaria que le corresponde.

Aplicando la Segunda Forma Normal en nuestro ejemplo:

Tenemos la tabla factura que cumple la primera forma normal, por lo que tenemos la siguiente anotación:

> Factura(#numero_factura, fecha_factura, nombre_paciente, numero_paciente, direccion_paciente, comuna_paciente, ciudad_paciente, codigo_sistema_salud)

Aca no tenemos ninguna clave primaria compuesta, ni claves foraneas, por lo que la tabla Factura ya se encuentra en segunda forma normal.

En la tabla FacturaItem tenemos dos claves, el número de factura y el número de item.

> FacturaItem(#numero_factura,#*numero_item*, nombre_item, valor_item)

Cuando ocurre algo de esta naturaleza, hay que determinar si los otros atributos dependen de una clave, o bien de ambas claves.

- nombre_item depende del numero_item
- valor_item: pueden ocurrir dos casos
    - depende de la relación numero_factura y numero_item, ya que es el valor obtenido para la factura, no unitario.
    - es el valor unitario y depende de numero_item

Para solucionar este conflicto, deberemos hablar con nuestro cliente para definir cual de las dos opciones debemos seguir. Para efectos de este ejercicio, optaremos por la segunda opción.

Como tenemos atributos que dependen de ambas claves y otros de una sola clave, tenemos que separarlas en 2 tablas nuevamente

> FacturaItem(#numero_factura,#*numero_item*)
> Item(#numero_item, nombre_item, valor_item)

Resultando las siguientes Tablas

#Factura	ITEM
23464	200
23464	204
23464	245
23464	414

ITEM	NOMBRE	VALOR
200	Pieza semiprivada	150
204	Televisión	10
245	Rayos X	25
414	Exámenes	35

Imagen 3. Segunda Forma Normal.

Tercera Forma Normal (3FN)

Esta forma debe cumplir las siguientes condiciones:

- Debe satisfacer 2FN
- Toda entidad debe depender directamente de la clave primaria

Si observamos las tablas que tenemos hasta el momento, nos preguntamos ¿el paciente tiene alguna relación directa con la factura en si?

Factura	# 23464 20/04/2019
Nombre paciente: Marta Fuentes	#Paciente: 4724
Dirección: Monjitas 245	Comuna: Santiago
COD-SISTEMA-SALUD: 10	Ciudad: Santiago

#Factura	ITEM
23464	200
23464	204
23464	245
23464	414

ITEM	NOMBRE	VALOR
200	Pieza semiprivada	150
204	Televisión	10
245	Rayos X	25
414	Exámenes	35

Imagen 4. Tercera Forma Normal.

La respuesta es no.

A esta dependencia se le denomina dependencia funcional transitiva.

En esta fase, si aún quedan atributos que no dependen directamente de la clave primaria, queda llevarlas a una nueva tabla y se puede relacionar a través de una clave foránea con la tabla proveniente. Para pasar a esta forma, debemos encargarnos de eliminar toda dependencia funcional transitiva.

Aplicando la Tercera Forma Normal:

Tenemos la tabla factura, en la cual nombre_paciente, numero_paciente, direccion_paciente, comuna_paciente, ciudad_paciente y codigo_sistema_salud dependen de paciente y no de factura.

> Factura(#numero_factura, fecha_factura, nombre_paciente, numero_paciente, direccion_paciente, comuna_paciente, ciudad_paciente, codigo_sistema_salud)

Por lo que separaremos estos atributos a una nueva tablas

> Factura(#numero_factura, fecha_factura, *numero_paciente*)
> Paciente(*#numero_paciente*, nombre_paciente, direccion_paciente, comuna_paciente, ciudad_paciente, codigo_sistema_salud)

Resultando las siguientes Tablas

#Factura	Fecha factura	#paciente
23464	20/04/19	4724

#Paciente	Nombre	Dirección	Comuna	Ciudad	Cod-sis-salud
4724	Marta Fuentes	Monjitas 245	Santiago	Santiago	10

#Factura	ITEM
23464	200
23464	204
23464	245
23464	414

ITEM	NOMBRE	VALOR
200	Pieza semiprivada	150
204	Televisión	10
245	Rayos X	25
414	Exámenes	35

Imagen 5. Tercera Forma Normal.

Que bien en un modelo de bases de datos quedaría representado de la siguiente forma:

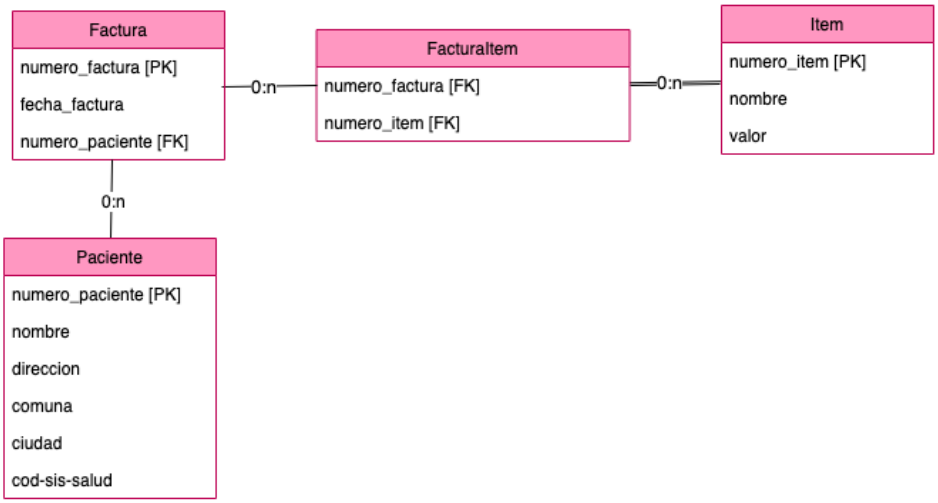

Imagen 6. Modelo de base de datos.

Resumen de la normalización

Como podemos ver, el proceso de normal conlleva una serie de pasos ordenados que nos permiten reducir la cantidad de datos que debemos almacenar en una base de datos, evitando así la redundancia de datos, simplifica la actualización de los datos y garantiza integridad referencial, previniendo la inconsistencia en las relaciones.

Como podemos ver en la siguiente imágen, en cada paso de la normalización se va reduciendo la cantidad de datos que tenemos en nuestro universo de la base de datos.

Imagen 7. Resumen de Normalización.

¿Qué pasaría si no hubiésemos normalizado nuestra base de datos?

Si pasamos nuestra vista inicial de factura a una tabla, sin normalizar, tenemos la siguiente tabla donde solo estamos mostrando los registros correspondientes a Marta Fuentes.

numero factura	fecha factura	nombre paciente	numero paciente	direccion paciente	comuna paciente	ciudad paciente	Cod-sis-salud	numero item	nombre item	valor item
23464	20/04/2019	Marta Fuentes	4724	Monjitas 245	Santiago	Santiago	10	200	Pieza semiprivada	150
23464	20/04/2019	Marta Fuentes	4724	Monjitas 245	Santiago	Santiago	10	204	Televisión	10
23464	20/04/2019	Marta Fuentes	4724	Monjitas 245	Santiago	Santiago	10	245	Rayos X	25
23464	20/04/2019	Marta Fuentes	4724	Monjitas 245	Santiago	Santiago	10	414	Exámenes	35

Imagen 8. Tabla sin normalizar.

Imaginemos que queremos cambiar la dirección de nuestro paciente. Al tener todo desnormalizado, antes hubiésemos tenido que ir a buscar en cada uno de los registros de las facturas y cambiar la dirección del paciente correspondiente. En cambio, al tener normalizado, basta que consultemos la tabla de Pacientes y realizar la actualización en dicha tabla.

Lo bueno de tener la base de datos completamente normalizada es que minimizamos el espacio necesario para almacenar los datos y reducimos las anomalías de mantención.

Desnormalización y sus usos

Por desnormalización, entendemos el proceso de añadir redundancia en las tablas de manera deliberada. Una de las motivaciones para desnormalizar tablas en una base de datos es el reducir el tiempo de consulta al implementar joins en múltiples tablas.

Un elemento importante a considerar es el hecho que desnormalizar no significa ignorar el proceso de normalización. Más bien, es un paso posterior a la normalización donde buscamos maximizar la eficiencia y representación de los datos a expensas de hacer más compleja la mantención de una tabla específica

En una base de datos tradicional buscamos modularizar las entidades asociadas a un fenómeno en distintas tablas para reducir la redundancia y problemas lógicos en éstas. En una base de datos desnormalizada, preferimos aumentar la redundancia de nuestra base para reducir la cantidad de consultas mediante joins y ganar eficiencia en las consultas.

Ejemplo de desnormalización

Tenemos una base de datos sobre pacientes en un hospital, la cual está compuesta por tres tablas.

- Una tabla Paciente, que registra el nombre del paciente, el tipo de tratamiento y el doctor con el que se atendió, éstas últimas dos columnas están registradas como claves foráneas.

Paciente	Tratamiento_id	Doctor_id
María Zenteno	1	1
Fernando Sánchez	2	2
Jose Artigas	1	2
Emilia Tapia	1	1
Felipe Zamorano	2	1
César Oyarce	2	1
Catalina Maillard	1	2

Imagen 9. Ejemplo 1 de desnormalización.

- Una tabla Doctores que vincula la clave primaria con el nombre del doctor tratante.

Id	Nombre
1	Astorquiza
2	Feris

Imagen 10. Ejemplo 2 de desnormalización.

- Una tabla Tratamiento que vincula la clave primara con el nombre del tratamiento asignado al paciente.

Id	Nombre
1	Control Médico
2	Biopsia

Imagen 11. Ejemplo 3 de desnormalización.

Resulta que por si sola, la tabla Paciente no nos informa con quién se trataron ni qué tratamiento tuvieron. Para generar sentido sobre estos números, es necesario complementar la información existente en la tabla Paciente con las tablas Doctores y Tratamiento.

Se observa que la tabla Doctores presenta una cantidad menor de datos registrados, lo cual nos permite hacer uso eficiente de la memoria en nuestra base de datos. Lo mismo se aplica para el caso de la tabla Tratamientos.

Para este ejemplo práctico, observamos que la aplicación de consultas mediante join no es lo suficientemente compleja, dada la baja cantidad de casos y la simpleza de las consultas utilizadas.

Una tabla desnormalizada implica el duplicar la información de manera deliberada en una nueva representación de todas las tablas solicitadas.

Tabla desnormalizada que permite asociar cada paciente con un doctor y un tratamiento específico

Paciente	Tratamiento_id	Tratamiento_non	Doctor_id	Doctor_nombre
María Zenteno	1	Control Médico	1	Astorquiza
Fernando Sánchez	2	Biopsia	2	Feris
Jose Artigas	1	Control Médico	2	Feris
Emilia Tapia	1	Control Médico	1	Astorquiza
Felipe Zamorano	2	Biopsia	1	Astorquiza
César Oyarce	2	Biopsia	1	Astorquiza
Catalina Maillard	1	Control Médico	2	Feris

Imagen 12. Tabla desnormalizada.

¿Y por qué debemos desnormalizar?

La desnormalización busca como objetivo optimizar el funcionamiento de una base de datos con el proceso de agregar datos redundantes. El costo de agregar información adicional en una tabla puede generar que a la larga cueste menos con respecto a las consultas que se deba estar haciendo de manera reiterada.