IT Bootcamp

Normalización, Tablas Temporales e Índices

DigitalHouse>

SCHOOL OF INNOVATION

TECH ACADEMY

Índice

1. Normalización
2. Tablas temporales
3. Índices

1 | Normalización

¿Qué es la Normalización?

La normalización es un proceso de **estandarización** y **validación de datos** que consiste en ***eliminar*** las **redundancias** o **inconsistencias**, completando datos mediante una serie de reglas que actualizan la información, **protegiendo** su **integridad** y favoreciendo la **interpretación**, para que así sea más simple de consultar y más eficiente para quien la gestiona.

Existen **niveles de normalización** como reglas para minimizar problemas de lógica y mejorar la organización de los datos. Veremos a continuación:

- 1FN: Primera Forma Normal
- 2FN: Segunda Forma Normal
- 3FN: Tercera Forma Normal

1FN: Primera Forma Normal

En la **Primera Forma Normal**, se *eliminan datos duplicados en atributos* y se crean registros independientes.

La regla de la Primera Forma Normal establece que las columnas repetidas deben eliminarse y colocarse en tablas separadas, respetando la atomicidad de los datos.

Una relación está en primera forma normal si, y sólo si, satisface que sus dominios simples sólo tienen valores atómicos, es decir, si todos sus atributos son atómicos.

Ejemplo - 1FN: Primera Forma Normal

Por ejemplo: si se tiene una tabla ***cliente***, permitiendo tener varios números de teléfono por registro, algunos malos diseños serían:

id_cliente	nombre	apellido	teléfono
1	Zlatan	Ibrahimovic	011-1212, 011-2323, 0376-00000
2	Paolo	Maldini	0376-11111
3	Andriy	Shevchenko	011-7733

id_cliente	nombre	apellido	teléfono1	teléfono2	teléfono3
1	Zlatan	Ibrahimovic	011-1212	011-2323	0376-00000
2	Paolo	Maldini	0376-11111		
3	Andriy	Shevchenko	011-7733		

Ejemplo - 1FN: Primera Forma Normal

Un diseño se encontraría en 1FN en este caso, haciendo uso de dos tablas:

Una tabla de ***cliente*** y una tabla de ***teléfono del cliente***.

id_cliente	nombre	apellido
1	Zlatan	Ibrahimovic
2	Paolo	Maldini
3	Andriy	Shevchenko

id_cliente	teléfono
1	011-1212
1	011-2323
1	0376-00000
2	0376-11111
3	011-7733

2FN: Segunda Forma Normal

En la **Segunda Forma Normal**, se *eliminan columnas que no dependen de la clave principal*. La regla de la Segunda Forma Normal establece que todas las dependencias parciales se deben eliminar y separar dentro de sus propias tablas.

Una relación se encuentra en 2FN si, y sólo si, se encuentra en 1FN y si todos los atributos no clave dependen por completo de la clave.

Ejemplo - 2FN: Segunda Forma Normal

Por ejemplo, si se tiene la siguiente tabla con los datos de **cursos** y **alumnos**:

cod_curso	cod_alumno	nombre	apellido	nombre_curso
C001	A11	Sandro	Tonalli	Álgebra
C002	A14	Theo	Hernández	Estadística
C003	A19	Fikayo	Tomori	Contabilidad

Ejemplo - 2FN: Segunda Forma Normal

Para cumplir con la 2FN, se podría separar en este caso, en 2 (dos) tablas:

Una tabla para **alumnos** y otra tabla para **cursos**.

cod_alumno	nombre	apellido
A11	Sandro	Tonalli
A14	Theo	Hernández
A19	Fikayo	Tomori

cod_curso	nombre_curso
C001	Álgebra
C002	Estadística
C003	Contabilidad

3FN: Tercera Forma Normal

En la **Tercera Forma Normal**, se elimina subgrupos de datos en múltiples columnas de una tabla y crea tablas nuevas, con relaciones entre ellas.

Una relación está en 3FN si, y sólo si, está en 2FN y además, cada atributo no clave depende de la clave primaria de modo no transitivo.

Cuando las tablas están en la Tercera Forma Normal se previenen errores de lógica cuando se insertan o borran registros. Cada columna en una tabla está identificada de manera única por la llave primaria y no deben haber datos repetidos. Esto provee un esquema limpio y elegante, que es fácil de trabajar y expandir.

Ejemplo - 3FN: Tercera Forma Normal

Por ejemplo, si se tiene la siguiente tabla con los datos de empleados y a qué provincia pertenecen:

legajo	dni	nombre	apellido	cod_provincia	provincia
L001	00000111	Alessio	Romagnoli	33	Milano
L002	11222333	Matteo	Gabbia	33	Milano
L003	33222111	Davide	Calabria	45	Brescia

Ejemplo - 3FN: Tercera Forma Normal

Para cumplir con la 3FN, se podría separar en este caso, en 2 (dos) tablas:

Una tabla para los **empleados** y otra para las **provincias**.

legajo	dni	nombre	apellido	cod_provincia
L001	00000111	Alessio	Romagnoli	33
L002	11222333	Matteo	Gabbia	33
L003	33222111	Davide	Calabria	45

cod_provincia	provincia
33	Milano
33	Milano
45	Brescia

Más niveles de normalización

Existen seis niveles más de normalización, pero normalizar más allá de estos niveles vistos podría resultar exagerado, ya que estas reglas simplemente son una *guía para determinar la organización y simplificación de la base de datos*, pero resulta ser una ciencia subjetiva a la hora de la implementación.

En resumen, la normalización es una técnica que se utiliza para crear relaciones lógicas apropiadas entre tablas de una base de datos. Ayuda a prevenir errores lógicos en la manipulación de datos y facilita también, agregar nuevas columnas sin romper el esquema actual ni las relaciones.

2 Tablas temporales

¿Qué son las Tablas temporales?

Las tablas temporales o tablas de sesión, son tablas que generalmente se utilizan para hacer pruebas, consultas, análisis, etc.

- Se pueden realizar operaciones **SELECT**, **INSERT**, **UPDATE**, **DELETE**.
- La tabla y sus datos se eliminan al finalizar la sesión.

Se utilizan muchas veces para *evitar el uso de múltiples joins* en una misma consulta.

Tablas temporales: Algunas características

- No se pueden compartir a otros usuarios.
- Se pueden crear sin necesidad de permisos especiales.
- Se pueden crear hasta 1000 tablas volátiles en una sesión.
- Los datos *no quedan guardados* de manera permanente.

Cuando la cantidad de resultados que se obtenga de una consulta es demasiado pesada, en ocasiones se vuelve más conveniente tenerlas almacenadas en una tabla temporal.

Tablas temporales: Sintaxis

Para crear una tabla temporal, se usa la sintaxis:

SQL

```
CREATE TEMPORARY TABLE <table_name>
//query para obtener los datos a insertar;
```

SQL

```
CREATE TEMPORARY TABLE <table_name>
(column1 dataType [not null] [primary key],
 column2 dataType [not null] [primary key],
 ...,
 columnN dataType [not null] [primary key]);
```

3 | Índices

¿Qué son los Índices?

Son un mecanismo para optimizar consultas en SQL.

- Mejoran los tiempos de respuesta en consultas complejas.
- Mejoran el acceso a los datos al proporcionar una ruta más directa a los registros.

Evitan realizar escaneos (barridas) completas o lineales de los datos en una tabla.

Los índices en bases de datos tienen un funcionamiento similar al índice de un libro.

Tipos de índice

- Índice de clave primaria
    - No admite duplicados
- Índice ordinario
    - Admite duplicados
- Índice único
    - Son como los índices ordinarios pero no admiten duplicados

¿Cómo determinar un índice?

Algunas recomendaciones para tener en cuenta a la hora de crear un índice:

- **Comprender** bien tanto la base de datos como el negocio.
- Entender los "**WHERE**" y los "**JOIN**" de las consultas que se realizan para poder "predecir" próximas consultas recurrentes.
- Es habitual seleccionar como **Primary Index** el **id de la tabla** así como también fechas o id de País/Región.
- Puede involucrar uno o más campos.
- Evitar índices en tablas que se actualizan con mucha frecuencia.

Detectar índice de una tabla

Para ver el índice de una tabla específica, se puede utilizar:

```sql
SHOW INDEX FROM <table_name>;
```

O se puede validar índices también utilizando:

```sql
HELP INDEX <schema.table_name>;
```

Crear índices

Con la siguiente sintaxis, podemos crear un índice en el script de creación de una tabla:

```sql
CREATE TABLE Tabla (Columna1 int, Columna2
varchar(255),
      ...
ColumnaN DATA_TYPE
)
UNIQUE PRIMARY INDEX (Columna1, ...,ColumnaN);
```

Se podría usar sin UNIQUE, para determinar que es no único.

Crear índices

Con la siguiente sintaxis, podríamos crear índices en tablas de una base de datos:

```sql
CREATE [UNIQUE] INDEX <index_name>
ON <tablename> (column1, column2, ..., columnN);
```

```sql
ALTER TABLE <table_name>
ADD INDEX nombre_indice (column1, column2, ..., columnN);
```

Ejemplo: Índices

Aquí podemos observar un ejemplo de creación de un índice con la BD movies.

```sql
CREATE INDEX movies_idx
ON movies (id);
```

DigitalHouse>