PREGRADO

UNIDAD 2 | RELATIONAL DATABASE MODELING AND IMPLEMENTATION

NORMALIZATION

SI400 | Diseño de Base de Datos

Al finalizar la unidad de aprendizaje, el estudiante diseña, implementa y realiza operaciones de manipulación de datos sobre una base de datos relacional, satisfaciendo requisitos para un dominio y contexto determinados.

Logro de la semana:

Al finalizar la semana de aprendizaje, el estudiante ejecuta actividades para el modelado de datos de una base de datos bajo el enfoque relacional, considerando el cumplimento de las formas normales.

AGENDA

NORMALIZACION

Normalización

- Propuesto por E.D. Codd para las bases de datos relacionales.
- Desarrollado para obtener estructuras eficientes y rápidas.
- Expresión formal de un buen diseño de base de datos.
- Mitiga las anomalías durante una actualización de datos.
- Mejora la independencia de los datos.
- Permite un crecimiento controlado de los datos.

Primera forma normal

- Sólo se permiten valores únicos para los atributos.
- Los grupos repetitivos son removidos y separados en otra entidad.
- Se identifican las claves primarias de cada entidad.

Primera forma normal: grupos repititivos

Un grupo repetitivo es un atributo que tiene más de un valor en cada fila de una tabla. Por ejemplo, suponga que estás trabajando con una relación de empleado y necesitas almacenar los nombres y las fechas de nacimiento de los hijos de los empleados. Debido a que cada empleado puede tener más de un hijo, los nombres de los hijos y las fechas de nacimiento de los hijos forman un grupo repetitivo.

emp#	first	last	children's names	children's birthdates
1001	Jane	Doe	Mary, Sam	1/9/02, 5/15/04
1002	John	Doe	Lisa, David	1/9/00, 5/15/01
1003	Jane	Smith	John, Pat, Lee, Mary	10/5/04, 10/12/00, 6/6/2006, 8/21/04
1004	John	Smith	Michael	7/4/06
1005	Jane	Jones	Edward, Martha	10/21/05, 10/15/99

Primera forma normal: grupos repititivos

Hay dos formas de deshacerse de los grupos repetitivos para que una tabla cumpla con las reglas de la primera forma normal: una forma correcta y una forma incorrecta. Veremos primero el camino equivocado para que sepas lo que no debes hacer.

emp#	first	last	child name 1	child bdate 1	child name 2	child bdate 2	child name 3	child bdate 3
1001	Jane	Doe	Mary	1/1/02	Sam	5/15/04		
1002	John	Doe	Lisa	1/1/00	David	5/15/01		
1003	Jane	Smith	John	10/5/04	Pat	10/12/00	Lee	6/6/06
1004	John	Smith	Michael	7/4/06				
1005	Joe	Jones	Edward	10/21/05	Martha	10/15/99		

Primera forma normal: grupos repititivos

La forma correcta de manejar los grupos repetidos es crear otra tabla (otra entidad) para manejar varias instancias del grupo repetido.

employee

emp#	first	last
1001	Jane	Doe
1002	John	Doe
1003	Jane	Smith
1004	John	Smith
1005	Joe	Jones

children

emp#	c_first	c_birthdate
1001	Mary	1/1/02
1001	Sam	5/15/04
1002	Lisa	1/1/00
1002	David	5/15/01
1003	John	10/5/04
1003	Pat	10/12/00
1003	Lee	6/6/06
1003	Mary	8/21/04
1004	Michael	7/4/06
1005	Edward	10/21/05
1005	Martha	1015/99

Primera forma normal: problemas

Aunque las relaciones de primera forma normal no tienen grupos repetitivos, están llenas de otros problemas.

Considera la siguiente relación:

```
orders (customer_numb, first_name, last_name, street,
city, state, zip, phone, order_numb, order_date,
item_numb, title, price, has_shipped?)
```

Primera forma normal: anomalía de inserción

La llave primaria está formado por el número de pedido y el número de artículo, hay dos operaciones que no podemos hacer con esta relación:

- No podemos agregar datos sobre un cliente hasta que realice al menos un pedido.
- No podemos agregar datos sobre un artículo sin que se haya pedido artículo.

Los anteriores casos son **anomalías de inserción**, situación que surge cuando no se puede insertar datos en una relación porque no se dispone de una clave primaria completa. En términos prácticos, ocurren porque hay datos sobre más de una entidad en la relación

Primera forma normal: anomalía de eliminación

Las relaciones de primera forma normal también pueden darnos problemas cuando eliminamos datos. Considera, por ejemplo, lo que sucede si un cliente cancela el pedido de un solo artículo y las reglas comerciales establecen que los artículos cancelados y los pedidos deben eliminarse:

- En los casos en que el artículo eliminado era el único artículo del pedido, perderá todos los datos sobre el pedido.
- En los casos en que el pedido fue el único pedido en el que apareció el artículo, perderá los datos sobre el artículo.
- En los casos en que el artículo eliminado era el artículo pedido por un cliente, perderá todos los datos sobre el cliente.

Estas **anomalías de eliminación** ocurren porque parte de la clave principal de una fila se vuelve nula cuando se eliminan los datos del artículo, lo que lo obliga a eliminar toda la fila. El resultado de una anomalía de eliminación es la pérdida de datos que le gustaría conservar. En términos prácticos, se ve obligado a eliminar datos sobre una entidad no relacionada cuando elimina datos sobre otra entidad en la misma tabla.

Primera forma normal: anomalía de actualización

Existe un último tipo de anomalía en la relación de órdenes que no está relacionada con la clave principal: una **anomalía de modificación** o **actualización**. La relación de pedidos tiene una gran cantidad de datos duplicados innecesarios, en particular, información sobre clientes. Cuando un cliente se muda, los datos del cliente deben cambiarse en cada fila para cada artículo en cada pedido realizado por el cliente. Si no se cambia cada fila, entonces los datos que deberían ser los mismos ya no son los mismos. El potencial de estos datos inconsistentes es la anomalía de modificación.

Segunda forma normal

- No deben existir atributos no claves que **dependan** sólo de una parte de la clave primaria.

- Verificar que la entidad ya está en 1FN.

- Separar los atributos que no dependen de la totalidad de la clave primaria y formar con ellos, una entidad.

- Identificar las claves primarias de la entidad generada.

Depencias funcionales

Una dependencia funcional es una relación unidireccional entre dos atributos, de modo que en un momento dado, para cada valor único del atributo A, solo se asocia un valor del atributo B a lo largo de la relación.

Por ejemplo, supón que A es el número de cliente de la relación de pedidos. Cada número de cliente está asociado con un nombre de cliente, un apellido, una dirección, una ciudad, un estado, un código postal y un número de teléfono. Aunque los valores de esos atributos pueden cambiar en cualquier momento, solo hay uno. Por lo tanto, podemos decir que el nombre, el apellido, la calle, la ciudad, el estado, el código postal y el teléfono dependen funcionalmente del número de cliente. Esta relación a menudo se escribe:

```
customer_numb ->> first_name, last_name, street, city, state,
      zip, phone
```

Uso de dependencias funcionales para llegar a 2FN

```
customer (customer_numb, first_name, last_name, street, city,
     state, zip, phone)

item (item_numb, title, price)

order (order_numb, customer_numb, order_date)

order_items (order_numb, item_numb, has_shipped?)
```

Segunda forma normal: problemas

Aunque la segunda forma normal elimina los problemas de muchas relaciones, ocasionalmente se encontrará con relaciones que están en la segunda forma normal, pero que presentan anomalías. Considera la siguiente relación:

```
item (item_numb, title, distrib_numb, warehouse_phone_number)
```

Para cada número de artículo, solo hay un valor para el título del artículo, el distribuidor y el número de teléfono del almacén. Sin embargo, existe una **anomalía de inserción**: no puede insertar datos sobre un distribuidor hasta que tenga un elemento de ese distribuidor, y una **anomalía de eliminación**: si elimina el único elemento de un distribuidor, perderá datos sobre el distribuidor. También hay una anomalía de modificación: el número de teléfono del almacén del distribuidor se duplica para cada artículo que la empresa obtiene de ese distribuidor. La relación está en segunda forma normal, pero no en tercera.

Tercera forma normal

- La relación está en segunda forma normal.
- No hay dependencias transitivas

```
item (item_numb, distrib_numb)

distributor (distrib_numb, warehouse_phone_number)
```

Depencias transitivas

Existe una dependencia transitiva cuando tiene el siguiente patrón de dependencia funcional:

```
A ->> B and B ->> C therefore A ->> C
```

Este es precisamente el caso de la relación de artículos originales. La única razón por la que el número de teléfono del almacén depende funcionalmente del número de artículo es porque el distribuidor depende funcionalmente del número de artículo y el número de teléfono depende funcionalmente del distribuidor.

Las dependencias funcionales son realmente:

```
item_numb ->> distrib_numb
distrib_numb ->> warehouse_phone_number
```

Forma normal Boyce-Codd

Para la mayoría de las relaciones, la tercera forma normal es un buen objetivo de diseño. Las relaciones en ese estado están libres de la mayoría de las anomalías. Sin embargo, de vez en cuando te encuentras con relaciones que muestran características especiales donde aún ocurren anomalías. La forma normal de Boyce-Codd (BCNF), la cuarta forma normal (4NF) y la quinta forma normal (5NF) se crearon para manejar este tipo de situaciones especiales.

Forma normal Boyce-Codd

La forma más fácil de entender BCNF es comenzar con un ejemplo. Suponga que la empresa ***Antique Opticals*** decide agregar una relación a su base de datos para manejar la programación del trabajo de los empleados. Cada empleado trabaja uno o dos turnos de cuatro (4) horas al día en la tienda. Durante cada turno, se asigna un empleado a una estación (un lugar en la tienda, como la recepción o el almacén). Solo un empleado trabaja en una estación durante el turno dado.

Una relación para manejar la programación podría diseñarse de la siguiente manera:

```
schedule (employee_id, date, shift, station, worked_shift?)
```

Dadas las reglas para la programación (una persona por estación por turno), hay dos posibles claves primarias para esta relación: employee_id + date + shift + station. Las dependencias funcionales en la relación son:

```
employee_id + date + shift ->> station, worked_shift?

date + shift + station ->> employee_id, worked_shift?
```

Forma normal Boyce-Codd

Ten en cuenta que lo anterior es cierto solo porque solo hay una persona trabajando en cada estación durante cada turno.

Esta relación de programación exhibe claves candidatas superpuestas (ambas claves candidatas tienen date y shift en común.) BCNF fue diseñado para manejar relaciones que exhiben esta característica.

Para estar en BCNF, una relación debe cumplir con las siguientes reglas:

- La relación está en tercera forma normal.
- Todos los determinantes son claves candidatas.

Se considera que BCNF es una forma más general de ver 3NF porque incluye esas relaciones con las claves candidatas superpuestas. La relación de programación de muestra que hemos estado considerando cumple con los criterios para BCNF porque los dos determinantes son claves candidatas.

RESUMEN

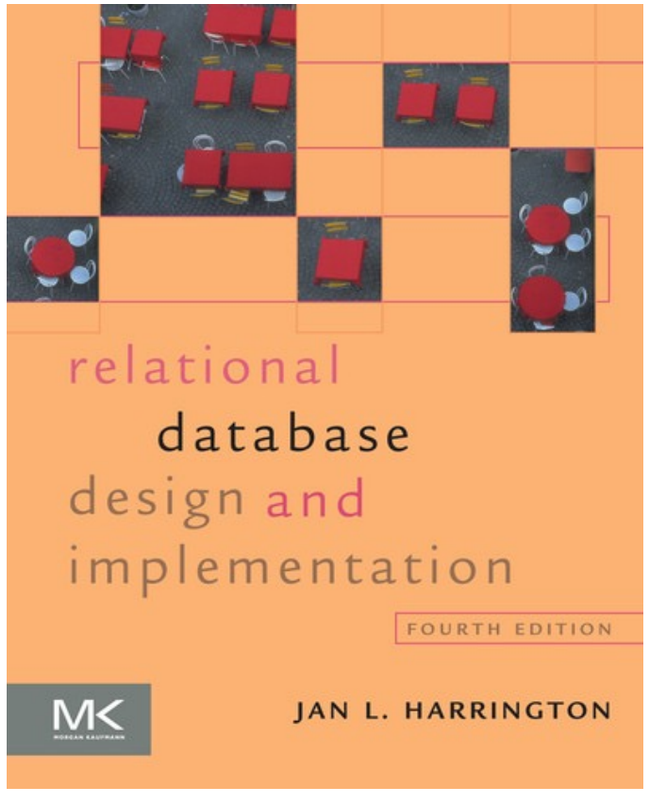
Recordemos

Normalización

REFERENCIAS

Para profundizar

Relational Database Design and Implementation



PREGRADO

Ingeniería de Software

Escuela de Ingeniería de Sistemas y Computación | Facultad de Ingeniería

UPC
Universidad Peruana
de Ciencias Aplicadas

Prolongación Primavera 2390,
Monterrico, Santiago de Surco
Lima 33 - Perú
T 511 313 3333
https://www.upc.edu.pe

exígete, innova