Normalización 2

tema 5

FN Boyce-Codd

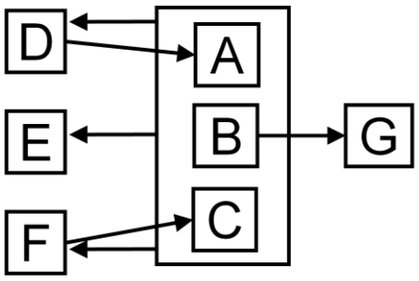
Cuarta FN

Quinta FN

Conclusión

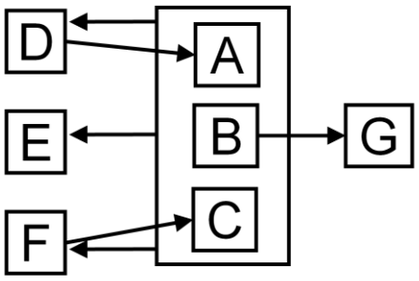
Forma Normal de Boyce-Codd

¿Cuáles son las claves candidatas?



Forma Normal de Boyce-Codd

¿Cuáles son las claves candidatas?



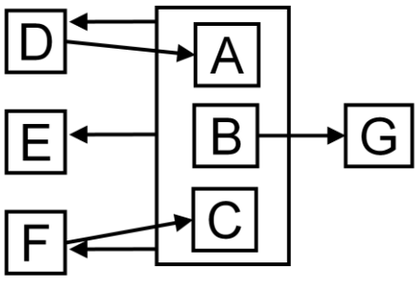
R (A,B,C,D,E,F,G)
CP (A,B,C)
CAlt (D,B,C)
CAlt (A,B,F)
CAlt (D,B,F)

Antes de entrar a definir la forma normal de Boyce-Codd, necesitamos conocer qué se entiende por determinante.

Empecemos extrayendo las claves candidatas presentes en este gráfico de dependencias funcionales. Recuérdese que tenemos que buscar conjuntos de atributos **de los que dependan todos los demás atributos**. Si elijo B, solo puedo llegar a G. Si Elijo (D,B) solo puedo llegar a G y A. Pero si elijo (D,B,C) puedo llegar a A (por D) y como "ya tengo" (A,B,C) ya puedo llegar al resto, a E y a F.

Forma Normal de Boyce-Codd

¿Cuáles son los determinantes?



R (A,B,C,D,E,F,G) (A,B,C)
CP (A,B,C) (B)
CAlt (D,B,C) (D)
CAlt (A,B,F) (F)
CAlt (D,B,F)

conjunto de atributos del que depende funcionalmente por completo algún otro atributo

Determinante es todo "de lo que salen flechas", puesto que las dependencias funcionales las estamos representando con esas flechas.

Forma Normal de Boyce-Codd

- Una relación está en forma normal de Boyce-Codd (FNBC) si y sólo si todo ***determinante*** es una clave candidata

R (A,B,C,D,E,F,G)
CP (A,B,C)
CAlt (D,B,C)
CAlt (A,B,F)
CAlt (D,B,F)

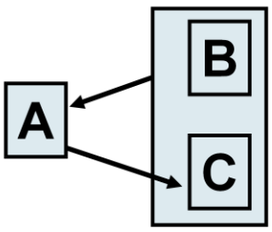
(A,B,C)
(B)
(D)
(F)

Aquí, como en ocasiones precedentes, se supone que debíamos decir que "…si la tabla está en 3FN y…". Lo que pasa es que si en una tabla todos los determinantes son claves candidatas ya estamos diciendo que cumple con la 2FN y la 3FN.

Lo que no es cierto es que una tabla en 3FN esté necesariamente en FNBC, y es lo que vamos a ver.

Forma Normal de Boyce-Codd

- en realidad, sólo nos debe preocupar si hay varias claves candidatas solapadas en atributos
  - CP(A, B) CAlt(B, C)
- el caso que resuelve se resume en

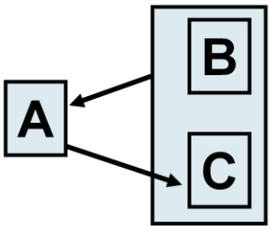


A	B	C
1	11	100
1	22	100
2	11	200

Este es el caso, en su formulación más simple, que resuelve la FNBC.

Forma Normal de Boyce-Codd

- atributo primo
  - el que está dentro de alguna cc
- atributo no primo
  - el que NO pertenece a ninguna cc

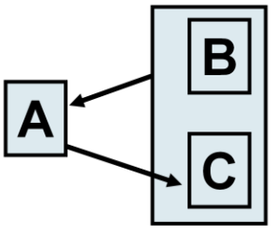


T(A,B,C)
CP(A,B)
CAlt(B,C)

Algo que ya se ha utilizado anteriormente, o se debería haber hecho, es el concepto de atributo primo. Es por eso que ciertas dependencias que parecen transitivas, en realidad, no lo son.

Forma Normal de Boyce-Codd

- ¿hasta 3FN o FNBC?
  - 1FN sí
  - 2FN sí
    - DF completa = todo atributo NO PRIMO depende de forma completa de TODA clave candidata
      - todos los atributos son primos
  - 3FN sí
    - DF transitiva = un atributo NO primo depende de otro atributo NO primo (o de un conjunto de atributos que no contiene ninguna CC)
      - todos los atributos son primos



T(A,B,C)

CP(A,B)

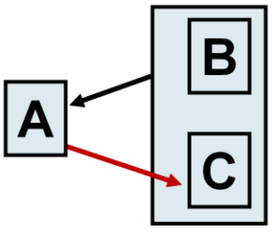
CAlt(B,C)

En el caso que nos ocupa ni hay dependencias funcionales incompletas ni transitivas. Para que haya DDFF incompletas debe existir un atributo no primo que dependa de forma parcial de alguna clave candidata. Para que haya dependencias transitivas, ha de haber un atributo no primo que dependa de otro atributo no primo.

Y en esta relación **todos los atributos son primos**, nadie puede incumplir esas formas normales.

Forma Normal de Boyce-Codd

- ¿hasta 3FN o FNBC?
  - 1FN sí
  - 2FN sí
  - 3FN sí
  - **FNBC no**
    - determinantes (B,C) y **(A)**



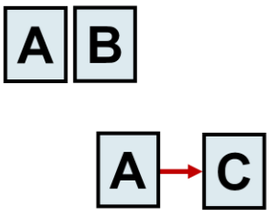
T(A,B,C)
CP(A,B)
CAlt(B,C)

A	B	C
1	11	100
1	22	100
2	11	200

Lo que ya falla es la correspondencia entre determinantes y claves candidatas. De hecho el determinante A no es CC: la dependencia A>C no es correcta.

Forma Normal de Boyce-Codd

- ¿hasta 3FN o FNBC?
  - 1FN sí
  - 2FN sí
  - 3FN sí
  - **FNBC no**
    - determinantes (B,C) y **(A)**



T(A,B)
CP(A,B)
CAj(A)→T2

T2 (A,C)
CP(A)

A	B
1	11
1	22
2	11

A	C
1	100
2	200

Esto se soluciona como siempre: la dependencia errónea forma una nueva tabla y se eliminan de la tabla inicial todos los atributos no primos de esa nueva tabla.

Aquí ya tenemos algún problema derivado de qué hemos elegido, al principio, como clave primaria y qué como alternativa. Pensemos en que hubiéramos intercambiado en la tabla inicial primaria y alternativa. Evidentemente, si C desaparece de la tabla inicial, ya no tiene sentido ninguna definición de clave candidata que la contenga, y simplemente se elimina esa restricción. Si se elimina, precisamente, la clave primaria, se cambia la alternativa a primaria y ya está.

Forma Normal de Boyce-Codd

- ¿hasta 3FN o FNBC? “truco”
  - Nos preocupa FNBC si:
    1. hay varias claves candidatas
    2. se solapan en atributos
  - Si alguna de esas condiciones es falsa entonces

3FN = FNBC

cuarta forma normal

- Transporte (conductor, tipoVeh, tipoCarga)
  CP(conductor, tipoVeh, tipoCarga)
  - no hay DD.FF. “normales”, por lo tanto está en FNBC
  - pero sí DD.FF. **MULTIVALOR**
    - conductor >> tipoVehículo
    - carga >> tipoVehículo

Conductor	Tipo Veh	Carga
Juan	Furgoneta	Perecederos
Marcos	Furgoneta	Perecederos
Juan	Camión	Muebles
Marcos	Camión	Muebles

T(conductor, tipoVeh, carga) CP(conductor, tipoVeh, carga)

Este ejemplo muestra una tabla que no presenta dependencias funcionales ya que todos los atributos pertenecen a la clave primaria. Sin embargo, hay ciertas dependencias especiales, o no vistas hasta ahora: las **dependencias multivalor**. Con el ejemplo se verá más claro.

Partimos de un sistema de información que tiene los siguientes requisitos:

Los conductores conducen siempre el mismo conjunto de vehículos.

Cada carga se transporta siempre en el mismo conjunto de vehículos.

De hecho, lo que estamos diciendo es que **un conductor determina un conjunto de tipos de vehículos**, o que carga determina un conjunto de tipos de vehículos. Ahí radica la diferencia con las dependencias funcionales vistas con anterioridad. También estamos expresando que **una carga se transporta en un conjunto de vehículos**. Podemos resumirlo en

CONDUCTOR >> TIPOVEH << CARGA

En la tabla anterior venimos a decir que Juan conduce furgonetas y camiones, lo mismo que Marcos. También estamos diciendo que los perecederos se llevan en furgoneta y los muebles en camión.

cuarta forma normal

- muebles>>{furgoneta,camión}
  - ¿Juan transporta muebles?

Conductor	Tipo Veh	Carga
Juan	Furgoneta	Perecederos
Marcos	Furgoneta	Perecederos
Juan	Furgoneta	Muebles
Marcos	Furgoneta	Muebles
Juan	Camión	Muebles
Marcos	Camión	Muebles

Pero en nuestro sistema de información los muebles se pueden llevar en furgoneta o en camión. Puesto que hemos establecido esas dependencias multivalor, cada vez que insertemos, digamos, muebles, estamos obligados a insertar tantas filas como vehículos se usan para ellos, y por cada vehículo, tantas filas como conductores. **Pero no estamos diciendo que Juan está transportando muebles**, tan solo que **puede** transportar muebles.

Queremos almacenar que los muebles también se transportan en furgoneta. **Estamos obligados**, aparte de que la clave primaria nos obliga a introducir un valor, **a insertar 2 filas puesto que Juan y Marcos conducen furgonetas**. Es evidente que estamos introduciendo redundancia en la tabla.

La anomalía reside en que **con las formas normales habituales esta tabla no presenta ningún problema**, ni hay dependencias funcionales incompletas ni transitividades, y todos los determinantes son claves candidatas.

cuarta forma normal

- Una relación está en 4NF si y solo si esta en 3FN o en FNBC y no posee dependencias multivalor no triviales
  - **CONDUCTOR**(conductor, tipoVeh) CP(conductor,tipoVeh)
  - **CARGA**(carga,tipoVeh) CP(carga,tipoVeh)

conductor	tipoVeh
Juan	Furgoneta
Marcos	Furgoneta
Juan	Camión
Marcos	Camión

tipoVeh	carga
Furgoneta	Perecederos
Furgoneta	Muebles
Camión	Muebles

Con esta descomposición se ve mejor qué es lo que realmente queremos almacenar, y es, simplemente, quién conduce qué, y qué se transporta en qué.

La tabla inicial: T(conductor, tipoVeh, carga) CP(conductor, tipoVeh, carga)

La descomposición por la 4FN:

T1(conductor, tipoVeh) CP(conductor, tipoVeh)

T2(tipoVeh, carga) CP(tipoVeh, carga)

Una dependencia multivalor trivial se da en cada una de las tablas ya que si bien conductor >> tipoVeh (conductor determina un conjunto de tipoVeh), son los únicos atributos de la tabla (ya que hemos descompuesto la original).

Sin embargo, fijémonos en que no se declara ninguna clave ajena: no se pretendía almacenar que una persona concreta transportaba una carga determinada. La carga es independiente del conductor. Se habla de "descomponer" la tabla original en dos, pero no de que se tengan que relacionar con claves ajenas.

No siempre es deseable esta descomposición.

quinta forma normal

- Dependencia de combinación
  - se trata de una dependencia "cíclica" impuesta por los requisitos del sistema
    - "si una propiedad P requiere un mueble M, y existe un proveedor S que ya provee a P y, además, suministra M, **entonces S suministra M en P**"
      - P4 necesita **camas** y sillas
      - P4 se provee de S1 y **S2**
      - S2 suministra sillas y **camas**

propiedad	mueble	suministrador
P4	cama	S1
P4	silla	S2
P16	cama	S2

Hay que entender que partimos de unos requisitos del sistema que, en principio, no está del todo claro cuál va a ser la forma más eficiente de reflejarlos en tablas.

Pensemos en que unos suministradores proveen de ciertos muebles a un conjunto de inmuebles o propiedades nuestras. Cada propiedad (supongamos, locales de oficinas, viviendas, tiendas, etc.) tiene unas características propias que hacen que se necesite un determinado mobiliario, y cada mobiliario tiene un conjunto de posibles suministradores.

En esta tabla se está almacenando **T**(propiedad, mueble, suministrador) **CP**(propiedad, mueble, suministrador):

1. Cada propiedad, el mobiliario que necesita.
2. Cada suministrador, el mueble que provee.
3. Pero es que, además, **cuando un proveedor suministra algo para una propiedad concreta, suministra lo mismo para cualquier otra propiedad que necesite ese tipo de mueble**.

En la tabla:

S2 suministra sillas para P4.

S2 suministra camas para P16.

P4 necesita sillas y camas.

Luego S2 debe suministrar camas a P4.

Obsérvese que S2 suministra sillas y camas pero P16 solo necesita camas: S2 NO tiene que suministrar sillas a P16. También que S1 solo suministra camas.

quinta forma normal

- Dependencia de combinación
  - se trata de una dependencia "cíclica" impuesta por los requisitos del sistema
    - "si una propiedad P requiere un mueble M, y existe un proveedor S que ya provee a P y, además, suministra M, **entonces S suministra M en P**"
      - esto nos obliga a insertar la que se ve como última fila:
        - **S2 suministra camas a P4**

propiedad	mueble	suministrador
P4	cama	S1
P4	silla	S2
P16	cama	S2
P4	cama	S2

En la tabla:

P4 necesita sillas y camas.

P4 se provee de S1 y S2.

S2 suministra sillas para P4.

S2 suministra camas para P16.

Luego S2 debe suministrar camas a P4.

La redundancia consiste en que decimos **varias veces** que S2 suministra camas y que P4 necesita camas.

quinta forma normal

- Una relación está en 5FN si no tiene dependencias de combinación
  - *forma normal de proyección-combinación*
    - "...como S2 ya suministra a P4 y suministra CAMAS, es obligado que suministre CAMAS a P4."

propiedad	mueble
P4	cama
P4	silla
P16	cama

mueble	suministrador
cama	S1
silla	S2
cama	S2

propiedad	suministrador
P4	S1
P4	S2
P16	S2

"...como S2 ya suministra a P4 y suministra CAMAS, es obligado que suministre CAMAS a P4."

•P16 solo necesita camas y tiene un suministrador, S2.

•S1 solo suministra camas.

•S2 suministra sillas y camas a P4 y P16.

Como en el caso de la 4FN, no es que la información repartida entre estas 3 tablas, resultado de la descomposición de esa dependencia circular, esté relacionada. Simplemente,

1. cada propiedad tiene sus necesidades de mobiliario,
2. cada mueble es suministrado por un conjunto de proveedores, y
3. cada propiedad es atendida por un conjunto de proveedores.

Dicho de otra forma, todas estas tablas tienen todos sus atributos dentro de la clave primaria y no hay claves ajenas. El caso es que si hacemos un JOIN de todas estas tablas obtenemos la tabla original, pero con esta descomposición hemos eliminado redundancia (que P4 necesita camas, lo decimos una sola vez, por ejemplo).

La tabla inicial: **T**(propiedad, mueble, suministrador) **CP**(propiedad, mueble, suministrador)

La descomposición por 5FN:

T1(propiedad, mueble) **CP**(propiedad, mueble)

T2(mueble, suministrador) **CP**(mueble, suministrador)

T3(propiedad, suministrador) **CP**(propiedad, suministrador)

conclusión

- formas normales
  - requisitos a cumplir por tablas
  - eliminar las anomalías de actualización
  - 1FN → 2FN → 3FN (1 cc)
  - 1FN → 2FN → 3FN → FNBC (>1 cc)
  - proceso reversible
  - 4FN, 5FN, ... son casos especiales
  - no siempre se descompone
    - tiempos de respuesta, por ejemplo
    - incluso se desnormaliza

Curiosamente, no siempre es deseable tener las tablas normalizadas, en ocasiones se introduce cierta redundancia en el sistema porque ayuda a mejorar el rendimiento del motor de base de datos o para sistemas de información especiales como son los de los denominados almacenes de datos (*data warehouse*) y la minería de datos (*data mining*).

conclusión: referencias

http://fbddocs.dlsi.ua.es/lecturas

licencias