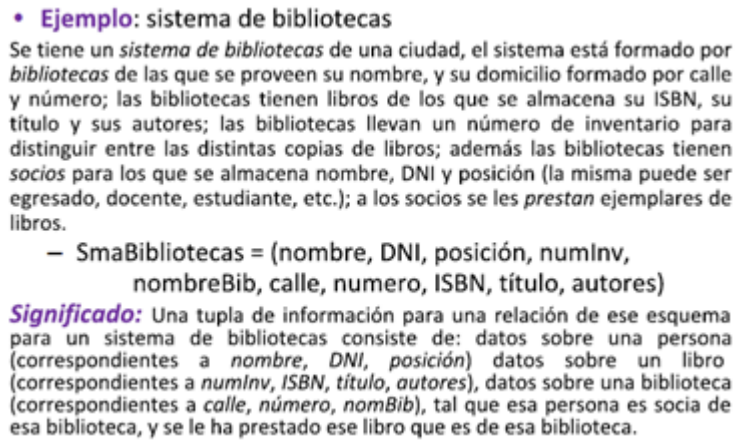
Normalización

Meta: obtener diseño de BD relacional de calidad, idealmente de manera automática

¿Qué es diseño de BD relacional?

conjunto de esquemas de relación.



- **Ejemplo**: sistema de bibliotecas

Se tiene un *sistema de bibliotecas* de una ciudad, el sistema está formado por *bibliotecas* de las que se proveen su nombre, y su domicilio formado por calle y número; las bibliotecas tienen libros de los que se almacena su ISBN, su título y sus autores; las bibliotecas llevan un número de inventario para distinguir entre las distintas copias de libros; además las bibliotecas tienen *socios* para los que se almacena nombre, DNI y posición (la misma puede ser egresado, docente, estudiante, etc.); a los socios se les *prestan* ejemplares de libros.

– SmaBibliotecas = (nombre, DNI, posición, numInv, nombreBib, calle, numero, ISBN, título, autores)

Significado: Una tupla de información para una relación de ese esquema para un sistema de bibliotecas consiste de: datos sobre una persona (correspondientes a *nombre*, *DNI*, *posición*) datos sobre un libro (correspondientes a *numInv*, *ISBN*, *título*, *autores*), datos sobre una biblioteca (correspondientes a *calle*, *número*, *nomBib*), tal que esa persona es socia de esa biblioteca, y se le ha prestado ese libro que es de esa biblioteca.

Esquema universal tiene todos los atributos del problema considerado

Una descomposición del esquema universal

R es esquema universal, una descomposición de R es R1,...,Rn

R = R1 U...U Rn

¿Qué significa que un diseño de BD relacional sea de calidad?

Respuesta: Hay ciertas propiedades indeseables de malos diseños de BD relacional. Hay que evitar esas propiedades cuando uno quiere construir un buen diseño.

¿Cómo encontrar esos problemas?

Respuesta: haciendo ejemplos de diseños y evaluándolos.

SmaBibliotecas = (nombre, DNI, posición, numInv,
nombreBib, calle, numero, ISBN, título, autores)

redundancia de información:

- el mismo usuario saca varios libros: se repite nombre y posición varias veces
- el mismo libro se saca varias veces prestado y se repite titulo y autores
- para cada persona que es socia de biblioteca se repite calle y número

trabajo con valores nulos:

- si tengo una persona solamente (no es socia, no le prestaron libros, etc.) entonces los campos van ser nulos.
- lo mismo si tengo libro solamente, los demas campos van a ser nulos.
-

Esto dice que el esquema universal no siempre es un buen diseño de BD y por lo tanto hace falta descomponerlo.

¿Cómo puedo arreglar el problema de redundancia de información en SmaBibliotecas?

Puedo descomponerlo de acuerdo a conceptos sirven para sacar atributos que ocasionan redundancia de información afuera (los demás atributos quedan dentro porque pueden hacer falta para relaciones con otros atributos).

SmaBibliotecas = (nombre, DNI, posición, numInv,
nombreBib, calle, numero, ISBN, título, autores)

Considero:

Persona = (nombre, **DNI**, posición)

SmaBibliotecas1 = (DNI, numInv, nombreBib, calle, numero, ISBN, titulo, autores)

Considero

Biblioteca = (**nombreBib**, calle, número)

SmaBibliotecas2=(DNI, numInv, nombreBib, ISBN, título, autores)

Considero:

Libro = (**ISBN**, título, autores)

SmaBibliotecas3 = (DNI, numInv, nombreBib, ISBN)

Se mitigó bastante problema de nulos pero no totalmente:

- p.ej. persona en biblioteca sin préstamo de libro: nulo en numInv, ISBN.
- que al sacar una tupla podemos perder una copia de un libro
- Libro en biblioteca pero no hubo préstamo, DNI nulo.

Eliminar estos problemas introduciendo otros conceptos con sus respectivos esquemas.

Socio = (DNI, nombreBib)

CopiaLibro = (nombreBib, numInv, ISBN)

Préstamo = (DNI, nombreBib, numInv)

Conclusión: problemas de diseño se arreglan introduciendo esquemas para conceptos varios que dependen de conocimiento del dominio y que son una descomposición del esquema universal

Recordar que modelado ER consideraba la introducción de conceptos varios y por lo tanto si se hace bien se obtendrá un buen diseño relacional luego de pasar a tablas.

La diferencia de modelado ER con lo que hicimos aquí es que hoy analizamos problemas de diseño y los fuimos eliminando uno a uno con conceptos que son introducidos.

Meta: Queremos que la descomposición salga automáticamente y no que dependa de un conocedor del dominio de las bibliotecas que hace uso inteligente del mismo. Se torna muy trabajoso si el esquema universal tiene cientos de atributos, cosa que suele pasar en la vida real.

Podemos ver que muchos de los conceptos anteriores tienen una clave primaria que no son todos los atributos del concepto.

- Además los atributos de esquema que no son de clave primaria muchas veces suelen introducir redundancia de información en el esquema universal.

DNI clave primaria de Persona: si r tabla de Persona, un valor de DNI determina una única tupla en r.

- un valor de DNI determina valores de atributos *nombre* y *posición* en r.
- Lo expresamos de otro modo mediante: DNI -> nombre, posición.

Igual podemos hacer para las otras claves primarias, o sea obtenemos las propiedades:

· nombreBib -> calle, número

· ISBN -> título, autores

· nombreBib, numInv -> ISBN

Estas propiedades valen aun cuando no tenemos los primeros 4 esquemas y solo tenemos el esquema universal.

· Este tipo de propiedades se llaman dependencias funcionales.

SmaBibliotecas = (nombre, DNI, posición, numInv,
nombreBib, calle, numero, ISBN, título, autores)

- DNI -> nombre, posición
- nombreBib -> calle, número
- ISBN -> título, autores
- nombreBib, numInv -> ISBN

DNI -> nombre, posición es usada para descomponer SmaBibliotecas:

SmaBibliotecas1 = (DNI, numInv, nombreBib, calle, numero, ISBN, titulo, autores)

R1 = (DNI, nombre, posición)

nombreBib -> calle, número es usada para descomponer SmaBibliotecas1:

SmaBibliotecas2 = (DNI, numInv, nombreBib, ISBN, titulo, autores)

R2 = (nombreBib, calle, número)

ISBN -> título, autores es usada para descomponer SmaBibliotecas2

SmaBibliotecas3 = (DNI,numInv, nombreBib, ISBN)

R3 = (ISBN, título, autores)

nombreBib, numInv -> ISBN la usamos para descomponer SmaBibliotecas3

SmaBibliotecas4 = (DNI, numInv, nombreBib)

R4 = (nombreBib, numInv, ISBN)

Luego descompusimos SmaBibliotecas en R1, R2, R3, R4, SmaBibliotecas4. Esto es automatizable.

Conclusión: las dependencias funcionales permiten descomponer esquema universal para remover mucha de la redundancia de información de SmaBibliotecas y aminoran el problema de valores nulos.

- **Formalización**:
  - Sea *R* un esquema relacional

    $\alpha \subseteq R$ y $\beta \subseteq R$
  - La dependencia funcional

    $\alpha \rightarrow \beta$

    se cumple en *R* si y solo si para todas las relaciones legales *r*(R), cada vez que dos tuplas t_1 y t_2 de *r* coinciden en los atributos α, también coinciden en los atributos β. Formalmente:

    $t_1[\alpha] = t_2[\alpha] \Rightarrow t_1[\beta] = t_2[\beta]$

Hay otros problemas de diseño de BD relacional?

- **Ejemplo**: Sea el esquema universal de un banco.

  Préstamo = (numSucursal, ciudad, activo, numCliente, numPréstamo, importe)

  – Lo descomponemos en:

  SucursalCliente = (numSucursal, ciudad, activo, numCliente)

  ClientePréstamo = (numCliente, numPréstamo, importe)

  – **Evaluación de la descomposición**: Si tenemos un cliente con varios *préstamos* en distintas sucursales:
    - no se puede decir el *préstamo* que pertenece a cada sucursal en la descomposición que tenemos.
    - Luego en la descomposición que tenemos se perdió información con relación al esquema universal.

Aquí se pierde relación entre atributos del problema al descomponer.

Veremos que este problema se resuelve así:

numSucursal -> ciudad, activo la uso para descomponer préstamo:

Sucursal = (numSucursal, ciudad, activo)

prest = (numSucursal, numCliente, numPrestamo, importe)

Oservación: Se puede probar que la eliminación del problema se debe a la forma de descomponer usando dependencias funcionales (pero para probarlo hace falta bastante teoría – no va a dar el tiempo para verla).

Los problemas anteriores de diseño se pueden resolver con un algoritmo que parte de esquema universal y dependencias funcionales y calcula automáticamente un esquema de BD de calidad.

Entonces el problema de encontrar diseño de BD de calidad se reduce al problema de hallar un conjunto apropiado de dependencias funcionales (DF)

Problema: ¿cómo encontrar DF para problema del mundo real?

¿Hace falta listar todas las DF de un problema del mundo real?

La respuesta es no.

- Ejemplo: persona = (DNI, nombre, posición)
- DNI -> nombre, posición

Resulta bastante obvio que : DNI-> nombre y DNI -> posición.

- **Ejemplo**: Sea *R* = (A, B, C, G, H, I),
  - *F* = { A→ B, A→ C, B → H }
  - Resulta bastante obvio que también se cumplen:

    A→ H y A → B, C.

Estas últimas se pueden derivar o deducir de las de F. Luego no hace falta listarlas.

Conclusión 1: Si F conjunto de DF, hay otras DF fuera de F que también se cumplen y que se pueden derivar de las de F.

Conclusión 2: no necesito dar todas las DF que se cumplen en el problema del mundo real sino un subconjunto de ellas lo menor posible tal que todas las demás se puedan derivar de ese subconjunto.

- Las reglas consideradas se llaman **axiomas de Armstrong:**
  - if $\beta \subseteq \alpha$, then $\alpha \rightarrow \beta$ **(reflexividad)**
  - if $\alpha \rightarrow \beta$, then $\gamma\,\alpha \rightarrow \gamma\,\beta$ **(aumentatividad)**
  - if $\alpha \rightarrow \beta$, and $\beta \rightarrow \gamma$, then $\alpha \rightarrow \gamma$ **(transitividad)**

nombreBib -> calle

---

nombreBib, número -> calle, número

- **Ejercicio**: queremos deducir $AC \rightarrow D$ a partir de las DF $\{A \rightarrow B;\ CB \rightarrow D\}$ usando las reglas de Armstrong.
  - ➢ Bosquejar una deducción.

1) A->B
2) CB ->D
3) AC -> CB (aumentatividad a 1))
4) AC -> D (transitividad a 3) y 2))

- Podemos simplificar más las derivaciones a partir de F, usando las siguientes reglas adicionales:
  - If $\alpha \rightarrow \beta$ holds and $\alpha \rightarrow \gamma$ holds, then $\alpha \rightarrow \beta\,\gamma$ holds **(union)**
  - If $\alpha \rightarrow \beta\,\gamma$ holds, then $\alpha \rightarrow \beta$ holds and $\alpha \rightarrow \gamma$ holds **(decomposición)**
  - If $\alpha \rightarrow \beta$ holds and $\gamma\,\beta \rightarrow \delta$ holds, then $\alpha\,\gamma \rightarrow \delta$ holds **(pseudotransitividad)**

  Las reglas anteriores se pueden inferir a partir de los axiomas de Armstrong.

- **Generalizando**: Dado un esquema relacional R una DF f con atributos en R **se deduce** de un conjunto de DFs F con atributos en R si existe una lista de DFs $f_1, \ldots, f_n$ tales que $f_n = f$ y para todo $1 \leq i \leq n$:
  1. $f_i \in F$ ó
  2. f_i se obtiene por aplicar la regla de reflexividad ó
  3. f_i se obtiene por aplicar aumentatividad ó transitividad a pasos anteriores en la lista.
- **Notación**: Usaremos $F \vdash f$ para decir que f se deduce de F.

El cierre de un conjunto de DF F (F+) son todas las dependencias que se deducen de F.

¿Es viable calcular F+ si tenemos muchos atributos en el problema del mundo real?

En la práctica hay cientos de atributos en esquema universal y no es viable calcular F+.

La razón es que para α hay exp(2, |α|) dependencias triviales.

O si α -> β en F hay exp(2, |R|) ɣ al aplicar aumentatividad a α -> β y R esquema universal.

Como ven F+ es demasiado grande como para poder calcularlo todo.

Definición: α -> β se cumple en r(R):

para todo t1 <> t2 en r: t1[α] = t2[α] entonces t1[β] = t2[β]

para todo t1, t2 en r: t1[α] = t2[α] entonces t1[β] = t2[β]

Estas dos definiciones son equivalentes.

Vamos a usar esa definición en el siguiente

Ejercicio:

A	B	C
a1	b1	c1
a2	a1	c1
a1	b2	c2

Listar algunas DF que no se cumplen y también algunas DF que se cumplen.

Solución:

B -> C porque los valores de columna B no se repiten.

B-> A por el mismo motivo.

B -> CA por el mismo motivo

BC -> A no hay dos tuplas com mismos valores en BC

no se cumple A->C por la primera y tercera tupla no cumplen con definición

no se cumpe A-> B por la misma razón

triviales

continuar completando ...

Problema: tenemos F conjunto de DF para problema de mundo real y queremos saber si vale la pena agregar DF f a F:

idea 1) calcular F+ y ver si f está en F+

Vimos que esta idea es inviable.

idea 2) es intentar deducir f de F y si lo logramos: no se agrega f a F.

Y si no lo logramos? puede que no sepamos como derivar f de F o que f no sea derivable de F.

Necesitamos saber que f no es deducible de F, pero no sabemos hacer ese tipo de pruebas.

Por lo tanto las dos ideas anteriores son inviables.

Idea que parece viable: Quiero deducir α -> β de F. Si calculamos las dependencias de F+ con lado izquierdo α, este conjunto es mucho más chico que F+ (porque hay 2^n subconjuntos de un conjunto de n atributos y solo consideramos uno de ellos).

- **Por lo tanto**: para responder si $F \vdash \alpha \rightarrow \beta$, bastaría contestar

  $\alpha \rightarrow \beta \in \{\alpha \rightarrow \varphi \mid F \vdash \alpha \rightarrow \varphi\}$ o mejor contestar:

  $\beta \in \{\varphi \mid F \vdash \alpha \rightarrow \varphi\}$ o mejor contestar:

  $\beta \subseteq \{A \in R \mid F \vdash \alpha \rightarrow A\}$

- Llegamos así a un conjunto conocido como **cierre de un conjunto de atributos**.

Fijarse que en el segundo paso vuelvo a tener una cantidad exponencial de φ: basta tomar el φ más grande del conjunto y por la regla de descomposición todos los subconjuntos de φ también pertenecen al conjunto. Luego el conjunto del segundo paso tiene exp(2,|φ|) elementos. Nuevamente tenemos una exponenciación; la eliminamos en el tercer paso donde solo tenemos un subconjunto de atributos de R esquema universal..

Entre cada paso y el siguiente se puede ver que hay un si y sólo si.

Hacemos las cuentas con un ejemplo. Sea R = (A, B, C, H) universal

$F = \{A \rightarrow B, A \rightarrow C, B \rightarrow H\}$

queremos ver si agregar A -> H? al conjunto y

en lugar de calcular F+

ver si A -> H esta en {A -> φ: F |- A -> φ}

eso equivale a ver:

H esta en {φ: F |- A -> φ}

eso equivale a ver

H subconjunto de {M atributo de R universal: F |- A -> M } que sabemos que contiene {B, C, H } - basta mirar F.

El conjunto:

$\{A \in R \mid F \vdash \alpha \rightarrow A\}$ (R esquema universal del problema y F conjunto de DF del problema) es clave y se llama cierre del conjunto de atributos ***α***. Formalizando:

Sea R el esquema universal, F conjunto de DF del problema del mundo real (con atributos en el universal), sea ***α*** ⊆ R. El **cierre de *α* bajo F** (denotado por α_{F^+}) se define:

$$\alpha^+_F = \{A \in R : F \vdash \alpha \rightarrow A\}.$$

Dos resultados a guardar en la cabeza por su utilidad:

Proposición: $F \vdash \alpha \rightarrow \alpha^+_F$
Prueba: sale aplicando unión finitas veces .
Proposición: $F \vdash \alpha \rightarrow \beta$ si y solo si $\beta \subseteq \alpha^+_F$

¿Ver si α -> β se agrega a F?

¿Será que se cumple F |- α -> β ?

Usemos la proposición anterior:

Si $\beta \subseteq \alpha^+_F$: la respuesta es sí. por lo tanto no agregar a F: α -> β .

sino: α -> β no se deduce de F: por lo tanto agregamos a F: α -> β

- Algoritmo para computar α^+_F (la clausura de α bajo *F*)

```
result := α;
while (changes to result) do
    for each β → γ in F do
        if β ⊆ result then result := result ∪ γ
    end
```

Ahora aplicamos este algoritmo paso a paso en un ejercicio.

- **Ejercicio**: Dados $R = (A, B, C, G, H, I)$,

  $F = \{ A \rightarrow B, A \rightarrow C, CG \rightarrow H, CG \rightarrow I, B \rightarrow H$

  Calcular A^+_F y $(AG)^+_F$

result = A

considerando A-> B: result = {A,B}

considerando A->C: result = {A, B, C}

Las dos siguientes DF no se pueden usar para acrecentar result porque su lado izquierdo no está en result.

considerando B->H: result = {A, B, C, H}

en una nueva iteración result no cambia. Por lo tanto salimos del while.

A+ = {A,B,C,H}

Si aplican el algoritmo para AG van a encontrar:

AG+ = R

AG -> R se deduce de F?

R está en AG+ y por la proposición nuevamente la respuesta es sí.

Consideramos las siguientes definiciones:

Sea R esquema relacional y F es conjunto de DF, entonces α **superclave** de R si y solo si α -> R está en F+.

α clave candidata de R si y solo si:

- **α** superclave de R
- para todo A en **α**: **α** - {A} no es superclave de R

Entonces en el ejercicio anterior vimos que AG es superclave

Se puede probar que en el ejercicio anterior AG también es clave candidata:

A no es superclave porque A+ no da todo R

y G+ = G como no hay dependencia en F con G sola a la izquierda no se va a agregar ningun atributo a result.

Probar que de F se deduce AG -> H usando cierre de conjunto de atributos.

AG+ = R que contiene a H; entonces por la propisición AG -> H se deduce de F.

Algunas propiedades que son ciertas (para el que quiera entretenerse):

F++ = F+

α++ = α+

Una DF es **trivial** si es satisfecha por todas las relaciones de un esquema.

- **Ejemplo**: En *SmaBibliotecas*
  - *nombreBib, calle → nombreBib*
  - *calle → calle*

Una dependencia es trivial si se cumple en todas las relaciones de esquema que contiene atributos de la dependencia.

Ejercicio de la práctica:

- **Proposición**: $\alpha \rightarrow \beta$ es **trivial** si y solo si $\beta \subseteq \alpha$.

Ejercicio: sea el esquema:

Préstamo = (numSucursal, ciudad, activo, numCliente, numPréstamo, importe)

indicar dependencias funcionales para Préstamo.

numSucursal -> activo

numSucursal -> ciudad

numPrestamo -> importe

numPrestamo -> numCliente

numCliente -> numPrestamo (si el banco hace lo más un préstamo por cliente y además si el cliente no tiene prestamo se pone null en los campos del préstamo, y null en campos de prestamo significa que cliente no tiene préstamo)

numCliente, numsucursal -> numPrestamo (si el banco permite un préstamo por cada sucursal)

Completar las dependencias funcionales y eliminar si hay alguna demás (i.e.si alguna es derivable de las otras).

Cuando hay muchas DF:

- Pueden tener dudas si añadir alguna y ahí viene bien la proposición.
- O tienen dudas si eliminar alguna y ahí viene bien la proposición.

meta de esta clase: definir el algoritmo de normalización (automático)

Vimos que si R esquema con redundancia de información en un conjunto de atributos β y la DF α -> β se cumple en R:

- Si α no determina todos los atributos de R,
- esa dependencia puede ser usada para eliminar redundancia de información por medio de la descomposición de R:
    - se saca β de R y se crea un esquema con los atributos de α y β.

Sea R, F (conjunto de DF). Que α no determina todos los atributos de R es lo mismo que decir:

- que α -> R no se deduce de F
- que α no es superclave de R

Queremos caracterizar entonces un esquema R que no tiene redundancia de información "proveniente de dependencias funcionales". Un esquema que cumple esa propiedad diremos que está en la forma normal Boyce-Codd (FNBC).

¿Qué tenemos que pedir entonces para R?

para toda α -> β en F+, α y β en R :

1) α -> R o α superclave de R
2) β subconjunto de α (dependencia trivial)

F viene dado por:

DNI -> nombre, posición

nombreBib -> calle, número

ISBN -> título, autores

nombreBib, numInv -> ISBN

Libro = (ISBN, título, autores)

Se puede probar que Libro está en FNBC. Hay que analizar dependencias de F+ con atributos en Libro y para cada una chequear que vale al menos una de las condiciones 1) y 2).

- **Definición**: Un esquema R está en **forma normal de Boyce-Codd (FNBC)** con respecto a un conjunto F de DFs si para todas las DFs en F^+ de la forma $\alpha \rightarrow \beta$, donde $\alpha \subseteq R$ y $\beta \subseteq R$, al menos una de las siguientes propiedades se cumple:
  - $\alpha \rightarrow \beta$ es trivial (i.e., $\beta \subseteq \alpha$)
  - α es una superclave de R (i.e. $\alpha \rightarrow R \in F^+$).

- **Definición**: Sea R esquema universal, F conjunto de DFs. Una descomposición $\{R1,\ldots,Rn\}$ de R está en Forma normal de Boyce-Codd (FNBC) con respecto a F si y solo si cada Ri está en FNBC con respecto a F.

¿Cómo comprobar que un esquema R con respecto a F no está en FNBC?

encontrar dependencia que cumple:

$\alpha \rightarrow \beta$ no trivial en F^+ tal que $\alpha \rightarrow R \notin F^+$

se las llama dependencias testigo (o violaciones de FNBC).

- **Ejemplo**: Sea $R = (A, B, C)$ esquema con DFs:
  $F = \{A \rightarrow B, B \rightarrow C\}$.

¿Está R en FNBC?

B->C es testigo. No trivial (obvio). A no está en B+ = {B, C}, por lo tanto por la proposición B no es superclave de R. En consecuencia R no está en FNBC.

- **Ejemplo**: Sea el esquema relacional $R = (A, C, D)$ con DFs: $F = \{A \rightarrow B, B \rightarrow C\}$.

¿Está R en FNBC?

A->B: no se puede usar porque B no está en R. Cuidado con este tipo de error común!

A->C está en F+ (por transitividad de las DF de F)

no es trivial (obvio), A no superclave de R: D no está en A+ (el algoritmo que calcula el cierre de un conjunto de atributos solo toma atributos del F y el F no contiene el D), entonces por la proposición A no es superclave de R. En consecuencia R no está en FNBC.

Proposición: si los atributos de F están contenidos en R: basta con buscar testigos en F.

(si R no está en FNBC, entonces F tiene un testigo).

Consecuencia: para descomponer el esquema universal basta con buscar testigos en F.

- **Ejemplo**: Sea $R = (A, B, C)$ esquema con DFs:
  $F = \{A \rightarrow B, B \rightarrow C, C \rightarrow A\}$.

¿está R en FNBC?

Atributos(F) contenidos en R: por proposición anterior basta con chequear dependencias del F: A, B y C son superclaves (justificarlo). Entonces ningún miembro de F es testigo. Por lo tanto R está en FNBC.

La proposición anterior dice cuando es fácil chequear si un esquema está en FNBC.

Si los atributos de F no están contenidos en R necesitamos otra manera de probar que un esquema está en FNBC.

- **Comprobación de FNBC (caso 2)**: Sea R_U universal, con DFs F y sea R_i que forma parte de descomposición de R_U; para probar que R_i está en FNBC se puede hacer la siguiente comprobación:
  - $\forall\ \alpha \subseteq R_i : \alpha^+ \cap (R_i - \alpha) = \varnothing \vee R_i \subseteq \alpha^+$

la primera condición del V significa: todas las DF con lado izquierdo α son triviales.

Chequear todos los α -> β de F+ con atributos en Ri es costoso!

Ri en FNBC y α -> β de F+ con atributos en Ri: si α no superclave de Ri entonces todas las dependencias con lado izquierdo igual a α son triviales.

Que todas las dependencias con lado izquierdo igual a α son triviales es equivalente a:

....

$\alpha^+ \cap (R_i - \alpha) = \varnothing$'

Si la comprobación anterior (del V) falla para un α esto significa que la siguiente DF es testigo:

$$\alpha \rightarrow \alpha^+ \cap (R_i - \alpha)$$

y usamos esa dependencia para descomponer Ri.

Ahora estamos en condiciones para presentar el algoritmo de normalización.

- **Problema 3:** Sea *R*, *F* conjunto de DFs. ¿Cómo hallar una descomposicion de *R* que está en FNBC?
- **Solución**: Algoritmo de normalización en FNBC.
- *result* := {*R*};
  while (there is a schema R_i in *result* that is not in BCNF) **do**
  begin
  let $\alpha \rightarrow \beta$ DF witness of R_i and $\alpha \cap \beta = \varnothing$;
  result := (*result* – R_i) ∪ (R_i – β) ∪ (α, β);
  end

.

¿cómo encontrar un esquema que no está en FNBC?

¿Cómo buscar un testigo para un miembro de la descomposición?

- **Ejercicio**: Sea el esquema universal:
  BibLibs = (nomBib, calle, número, numInv, ISBN, título, editorial, autores, edición)
  Sea *F* dado por:
  – nomBib → calle, número
  – calle, número → nomBib
  – ISBN → título, editorial, autores, edición
  – nomBib, numInv → ISBN
  Aplicar el algoritmo de normalización en FNBC.

Como comenzamos con esquema universal BibLibs y atributos(F) contenido en BibLibs por proposición buscamos testigos en F por proposición.

nomBib -> calle, numero es testigo?

no porque no es trivial y

nomBib+ = {nomBib, calle, numero} <> R

Luego nomBib no es superclave de R.

Luego usamos esta dependencia para descomponer BibLibs2:

BibLibs2= (nomBib, numInv, ISBN, titulo, editoral, autores, edicion)

R1 = (nomBib, calle, numero)

Anora vamos a chequear que R1 está en FNBC

- $\forall\ \alpha \subseteq R_i : \alpha^+ \cap (R_i - \alpha) = \varnothing \vee R_i \subseteq \alpha^+$

nomBib es superclave, porque nomBib+ = R1; luego todos los superconjuntos de nomBib

están chequeados.

{calle, numero}+ = R1, luego {calle, numero} superclave, todos los superconjuntos están chequeados.

calle + = calle Esto significa que dependencias con calle a la izquierda son triviales

numero + = numero. Idem.

R1 está en FNBC.

Ver BibLibs2

BibLibs2 = (nomBib, numInv, ISBN, titulo, editoral, autores, edicion)

- $\dot{\forall}\ \alpha \subseteq R_i : \alpha^+ \cap (R_i - \alpha) = \varnothing \vee R_i \subseteq \alpha^+$

comienzo con ISBN

ISBN+ = {ISBN, titulo, editoral, edición, autores} que es menor que BibLib2

ISBN+ ∩ BibLib2 - ISBN

= {ISBN, titulo, editoral, edición, autores} ∩ {nomBib, numInv, titulo, editoral, autores, edicion}

= {titulo, editorial, edicion, autores } <> ∅

Por lo tanto la testigo es

$$\alpha \rightarrow \alpha^+ \cap (R_i - \alpha)$$

ISBN -> titulo, editorial, edicion, autores

La uso para descomponer BibLibs2:

BibLibs3 = (nomBib, numInv, ISBN)

R2 = (ISBN, titulo, editorial, edicion, autores)

R2 esta en FNBC?

ISBN+ = R2 cualquier superconjunto no necesita ser chequeado.

cualquier subconjunto de {título, editorial, edición, autores} no es superclave.

subconj { ‘título, editorial, edición, autores} + ∩ R2 - subconj { ‘título, editorial, edición, autores}

= subconj { ‘título, editorial, edición, autores} ∩ R2 - subconj { ‘título, editorial, edición, autores} = ∅

R2 está en FNBC.

está BibLibs3 = (nomBib, numInv, ISBN) en FNBC?

nomBib + = {nomBib, calle, numero}

nomBib + ∩ BibLibs3 - nomBib = {nomBib, calle, numero} ∩ {numInv, ISBN} = ∅.

nombib, numinv es superclave no hace falta chequear superconjuntos

numinv + = numinv

ISBN + = R2 luego ISBN no es superclave de BibLib3

ISBN + ∩ {nomBib, numInv} = ∅

{ISBN, numInv} + = R2 U {numInv}

{ISBN, numInv}+ ∩ BibLibs3 - {ISBN, numInv} = R2 ∩ nomBib = ∅

{ISBN, nomBib} + = R2 U R1

{ISBN, nomBib} + ∩ BibLibs3 - {ISBN, nombib} = (R2 U R1) ∩ {numInv} = ∅

Luego BibLib3 está en FNBC.

Terminó el algoritmo.

Vamos a detallar un poco más el algoritmo de normalización en FNBC.

Considero el tipo decomposition que además de tener una descomposición del esquema universal contiene anotaciones en los esquemas diciendo si sabemos que están en FNBC (valor True), si hay que averiguarlo (valor unknown), o si sabemos que no está en FNBC (valor False).

function testigo(D: decomposition) : decomposition x ((DF x int) U {FNBC})

La función testigo busca testigo en la descomposición y retorna:

- Descomposición D actualizada: o sea, con anotaciones actualizadas de esquemas de acuerdo a lo que se descubrió sobre ellos..
- Testigo de esquema de D con lados izquierdo y derecho disjuntos y en qué esquema está ese testigo.
- Valor FNBC si la descomposición está en FNBC.

Entonces el algoritmo de normalización en FNBC cambia a:

```
result :={(R, unknown)};

X = testigo(result);

while (X[2] <> FNBC) do

  α -> β = X[2][1]

  S  = SchemaWithFalse(X[1])

  // S es el esquema anotado con False y que tiene la dependencia α -> β

  result := (result - S) U (S - β) U (α U β)

  X = testigo(result)

od

print (result)
```

Ahora describo testigo(D):

- Si D tiene un esquema R solamente: Buscar testigo en F.
    - Si α -> β testigo de F: retorna: ({(R, False)}, (α -> (β - (α ∩ β)), 1))
    - sino retorna: ({(R, true)}, FNBC)
- L: Sino buscar en D esquema Ri con anotación unknown y chequear:

$$\dot{\forall}\ \alpha \subseteq R_i : \alpha^+ \cap (R_i - \alpha) = \varnothing \vee R_i \subseteq \alpha^+$$

    - Si esta condición es válida, marcar en D: Ri con True
        - Si hay siguiente esquema en D con anotación unknown: ejecutar L
        - sino: retorna: (D, FNBC)
    - Sino: retorna: (D, ($\alpha \rightarrow \alpha^+ \cap (R_i - \alpha)$, i))

El chequeo de $\dot{\forall}\ \alpha \subseteq R_i : \alpha^+ \cap (R_i - \alpha) = \varnothing \vee R_i \subseteq \alpha^+$ puede ser inteligente:

- Si Ri tiene dos atributos, entonces está en FNBC.
- Puedo comenzar con los conjuntos unitarios, luego pasar a los binarios y a así…
- Si encuentro superclave la anoto así no hace falta chequear superconjuntos de esta.
- Puedo buscar el conjunto X de atributos de Ri que no aparece en el lado izquierdo de ninguna DF de F; X y sus subconjuntos no necesitan ser chequeados (porque van a dar DF triviales).

Estas son las cosas que usamos en el ejemplo de la clase pasada.