Apuntes – Normalización

Nota:
El siguiente apunte constituye sólo un apoyo para las clases prácticas del tema de normalización. Los algoritmos están esquematizados a alto nivel y no tienen como objetivo una rigurosidad formal. A tal fin se deberá consultar la bibliografía de la materia. Les agradecemos informar de cualquier error encontrado o sugerencias a bddoc@dc.uba.ar, con tal de ir mejorando este apunte.

Parte 1: Algoritmos

Hallar Claves

Sea R = (A_1, A_2,…, A_n) y F un conjunto de dependencias funcionales.

Paso 1)
Obtener conjunto de atributos Siempre = {X | X no figura en ningún lado derecho en F}

Paso 2)
Calcular Siempre $^+$ con respecto a F.

Paso 3)
Si Siempre $^+$ = R, entonces Siempre es clave única.

Paso 4)
Sino (Siempre $^+$!= R) Agregar atributos (pertenecientes a R – Siempre)
GRADUALMENTE al conjunto Siempre, hasta que Siempre $^+$ = R.

Agregar gradualmente: Agregar un atributo, con todas las combinaciones. Agregar dos atributos, con todas las combinaciones. … Agregar sin obtener superclaves.

Cobertura Minimal

Sea R = (A_1, A_2,…, A_n) y F un conjunto de dependencias funcionales.

Paso 1) Abrir dependencias funcionales: Cada lado derecho contenga 1 único atributo.
Esto es, toda X->YZ se descompone en X->Y y X->Z.

Paso 2) Remover atributos a izquierda redundantes. Un conjunto de atributos B, $B \subset X$, es redundante en X->Y si $Y \in (X - \{B\})^+$ con respecto a F.
Si B es redundante, reemplazar X->Y por X – {B} ->Y.

Paso 3) Remover dependencias funcionales redundantes. X->Y es redundante en F si F – {X->Y} es equivalente a F. Esto es, si $Y \in X^+$ con respecto a **F – {X-> Y}** .
Si X->Y es redundante, removerla del conjunto.

Preservación de Dependencias Funcionales

Verificar si X->Y se preserva en una descomposición $\rho (R_1, R_2,\ldots,R_k)$

Paso 1)
Z **:=** X

Paso 2)
Repetir
  Para todo esquema $R_i \in \rho$
    $Z := Z \cup ((Z \cap R_i)^+ \cap R_i)$
Hasta que Z no cambie

Paso 3)
X->Y se preserva en $\rho (R_1, R_2,\ldots,R_k)$ si y solo si $Y \in Z$.

Tableau para Descomposición sin Pérdida

$R = (A_1, A_2,\ldots, A_n)$ y $\rho (R_1, R_2,\ldots,R_k)$

Paso 1)
Construir una matriz M de la siguiente forma:
n columnas: $A_1, A_2,\ldots, A_n$
k Filas: $R_1, R_2,\ldots,R_k$
$M[i,j] = a_j$ si $A_j \in R_i$
$M[i,j] = b_{ij}$ si $A_j \notin R_i$

Paso 2)
Repetir
  Para toda X->Y: Si dos filas coinciden en X, *igualar Símbolos* en Y
Hasta que M no cambie

Paso 3)
$\rho (R_1, R_2,\ldots,R_k)$ es SPDI (Sin pérdida de información) si existe una fila en M con todos símbolos a

Auxiliar: *Igualar Símbolos* en Y:
Si el valor en alguna fila es un a_i entonces asignarle el valor a_i a la otra fila.
Si en cambio tengo b_{ij} y b_{lj}, asignarle a ambos el valor b_{ij}.

Algoritmo Descomposición en 3FN, SPI, y SPDF.

Sea R = (A_1, A_2,…, A_n) y F un conjunto de dependencias funcionales.

Paso 1)
- Hallar un Cubrimiento Minimal CM para F.
-Inicializar ρ con el conjunto vacío.

Paso 2)
Por cada X->Y ∈ CM, agregar XY a ρ.
(Si tengo X-> Y e X->Z, agregar un único esquema XYZ)

Paso 3) Hasta el paso anterior, ρ es una descomposición en 3FN, SPDF. Para que también sea SPI se hace un último control:
Si ninguna clave está contenida en algún esquema en ρ, entonces agregar una de las claves a ρ.

Algoritmo Descomposición en FNBC, SPI

Sea R = (A_1, A_2,…, A_n) y F un conjunto de dependencias funcionales.

Paso 1)
Tomar una X->Y tal que no respeta FNBC. Particionar R en XY y R-Y.
Paso 2)
Proyectar F sobre los esquemas XY y R-Y.
Paso 3)
Descomponer recursivamente XY y R-Y.

Parte 2: Ejemplos

Hallar Claves

Sea R=(A,B,C,D,E) y F = {A->BC, CD->E, B->D, E->AD}

Paso 1)
Obtener conjunto Siempre ={∅} (conjunto vacío)
Paso 2) Agregar gradualmente atributos en R-Siempre:

Probamos con combinaciones de un único atributo:
A^+=ABCDE (es clave)
B^+=BD
C^+=C
D^+=D
E^+= **ABCDE (es clave)**

Probamos todas las combinaciones de dos atributos (sin A y E, ya que utilizando cualquiera de ellos obtendría superclaves):
BC^+ = **BCDEA (es clave).**
CD^+=**CDEAB (es clave).**
BD^+=BD

No probamos combinaciones de 3 atributos, ya que serían todas superclaves para este caso.

Luego, las claves son **{A,E,BC,BD}.**

Cobertura Minimal

Sea R=(A,B,C,D,E) y F = {A->B,CE->B,C->A,D->CA,B->C}

Paso 1) Descomponer dependencias funcionales a derecha:
Reemplazo D->CA por D->C y D->A.
CubMin={A->B,CE->B,C->A,D->C,D->A,B->C}

Paso 2) Eliminar elementos a izquierda redundantes:
Lo aplico a las reglas que tengan más de un atributo del lado izquierdo. En este caso, la única es CE->B.

Veamos si C es redundante en CE->B.
Esto es, veamos si $B \in E^+$ con respecto a CubMin.
E^+=E. Luego, C no es redundante.

Veamos ahora si E es redundante. Veamos si $B \in C^+$ con respecto a CubMin.
C^+ =CAB. Luego, E es redundante. Reemplazamos CE->B por C->B.
Mi nuevo CubMin = {A->B,C->B,C->A,D->C,D->A,B->C}

(Siempre se realizan las clausuras con respecto al último CubMin obtenido.)

No hay más casos para ver atributos redundantes a izquierda.

Paso 3) Eliminar dependencias funcionales redundantes.
Para ver si A->B es redundante, debo ver que $B \in A^+$ con respecto a CubMin – {A->B}
A^+ con respecto a CubMin – {A->B}= A.
Luego, como B no pertenece a esa clausura, la dependencia no es redundante.

Veamos si C->B es redundante.
C^+ con respecto a CubMin – {C->B}= CAB.
Como B pertenece esta clausura, la dependencia es redundante.
Luego la eliminamos de CubMin.
Mi nuevo CubMin={A->B,C->A,D->C,D->A,B->C}

Veamos ahora si C->A es redundante.
C^+ con respecto a CubMin – {C->A}=C. Luego, no es redundante.

Veamos ahora si D->C es redundante.
D^+ con respecto a CubMin – {D->C}=DABC. Luego, es redundante, y la eliminamos.

Mi nuevo CubMin={A->B,C->A,D->A,B->C}

Continuando, veremos que ya no hay más dependencias redundantes, luego
CubMin={A->B,C->A,D->A,B->C} es un cubrimiento minimal para F.

Preservación de Dependencias Funcionales

Sea R=(A,B,C,D,E), F=(AB->C,B->D,C->E,D->C,E->B), y ρ (ABC,CD,BDE).

El algoritmo tiene como entrada a ρ y a una dependencia funcional, por lo que deberíamos correrlo con cada dependencia funcional en F, y si para todas se preserva afirmar que no pierde dependencias. Si alguna falla, entonces F no es SPDF.

Sin embargo, algunas dependencias se preservan trivialmente porque se proyectan de manera directa. Por ejemplo, AB->C se proyecta en el esquema ABC, ya que tanto sus atributos en el lado izquierdo, como los del derecho, están incluidos en el esquema. Lo mismo acontece con B->D (en el esquema BD),D->C (en el esquema CD), y E->B (en el esquema BDE). Luego la única dependencia que nos queda comprobar es C->E.
Lo comprobamos mediante el algoritmo.
Z:=C.
Iteración 1:
$Z := C \cup ((C \cap ABC)^+ \cap ABC)$
$C \cup (C^+ \cap ABC)$
$C \cup (C^+ \cap ABC)$
$C \cup (CEBD \cap ABC)$
$C \cup CB$
CB
$Z := CB \cup ((CB \cap CD)^+ \cap CD)$
$CB \cup (C^+ \cap CD)$
$CB \cup (CEBD \cap CD)$
$CB \cup (CD)$
CBD

$Z := CBD \cup ((CBD \cap BDE)^+ \cap BDE)$
$CBD \cup (BD^+ \cap BDE)$
$CBD \cup (BDCE \cap BDE))$
$CBD \cup BDE$
CBD**E**
Fin Iteración 1.

Como $E \in Z$, podemos afirmar que C->E se preserva en ρ.
Luego, ρ es SPDF.

Nota: Recordar que para terminar el algoritmo y afirmar que la dependencia no se preserva es necesario pasar toda una iteración sin cambios en Z. En este caso, nuestro Z al comenzar la primera iteración era C y al finalizar era CBDE. Si hubiésemos estado comprobando la regla C->G, como G todavía no está en Z, deberíamos iterar una vez más por todos los esquemas.

Tableau para Descomposición sin Pérdida

Sea R=(A,B,C,D,E), F=(B->C,C->D,D->A,B->E), y ρ (AB,BCD,DE).
Verificar si ρ es SPDI.

Como ρ tiene más de dos esquemas, utilizamos el tableau. (Con dos esquemas R_1 y R_2 es más sencillo verificar si se cumple si $R_1 \cap R_2 \rightarrow R_1$. R2 o $R_1 \cap R_2 \rightarrow R_2$-R1).
Construimos la matriz inicial:

	A	B	C	D	E
AB	a_1	a_2	b_{13}	b_{14}	b_{15}
BCD	b_{21}	a_2	a_3	a_4	b_{25}
DE	b_{31}	b_{32}	b_{33}	a_4	a_5

Iteración 1:
Aplico B->C.
Busco columnas que tengan el valor B, para iguarlas en C. *Es decir, busco las filas donde la dependencia B->C no se cumple.*
Se da para las dos primeras filas. Al igualar b_{13} con a_3 reemplazamos b_{13} con a_3:

	A	B	C	D	E
AB	a_1	a_2	**a_3**	b_{14}	b_{15}
BCD	b_{21}	a_2	a_3	a_4	b_{25}
DE	b_{31}	b_{32}	b_{33}	a_4	a_5

Aplico C->D.

	A	B	C	D	E
AB	a_1	a_2	a_3	**a_4**	b_{15}
BCD	b_{21}	a_2	a_3	a_4	b_{25}
DE	b_{31}	b_{32}	b_{33}	a_4	a_5

Aplico D->A.

	A	B	C	D	E
AB	a_1	a_2	a_3	a_4	b_{15}
BCD	**a_1**	a_2	a_3	a_4	b_{25}
DE	**a_1**	b_{32}	b_{33}	a_4	a_5

Aplico B->E.

	A	B	C	D	E
AB	a_1	a_2	a_3	a_4	b_{15}
BCD	a_1	a_2	a_3	a_4	**b_{15}**
DE	a_1	b_{32}	b_{33}	a_4	a_5

Terminé la primera iteración (pasé por todas las reglas), y todavía no encontré una fila entera con variables distinguidas (las ai). Sin embargo, la matriz ha cambiado desde que empecé la iteración, por lo que debo volver a iterar por todas las reglas.

Comprobaremos que aplicando nuevamente todas las reglas ya no se producen cambios en la matriz. Como no hemos obtenido una fila con todas variables distinguidas, concluimos que ρ no es SPDI.

Nota: Apenas descubrimos toda una fila con variables distinguidas podemos terminar el algoritmo afirmando que si es SPDI.

Algoritmo Descomposición en 3FN, SPDI, y SPDF

Sea R=(A,B,C,D,E) y F = {A->B,CE->B,C->A,D->CA,B->C}

Calculamos las claves y obtenemos que DE es la única. Luego los atributos primos son D y E, y los no primos A, B, y C.

¿Ya está en 3FN? No, porque por ejemplo, en A->B tenemos que ni A es superclave ni B es primo. (Si no está en 3FN, tampoco está en FNBC. Tampoco está en 2FN: en D->C tenemos un atributo no primo que depende parcialmente de la clave DE, ya que depende de D.).

En primer paso consiste en hallar un CubMin para F.
Según lo resuelto en el segundo ejemplo de este apunte,
CubMin ={A->B,C->A,D->A,B->C} es un cubrimiento minimal de F.

El segundo paso nos dice agregar un esquema por cada dependencia en CubMin.
Luego ρ=(AB,CA,DA,BC).

El tercer paso es comprobar si algunas de las claves está contenida en algún esquema de ρ. En este caso, DE no está en ningún esquema, por lo que la agregamos:
ρ=(AB,CA,DA,BC,DE).
Por construcción, podemos afirmar que ρ es una descomposición en 3FN, SPI y SPDF.

Algoritmo Descomposición en FNBC, SPDI

Sea R=(A,B,C,D,E,F) y F = {A->D, D->AFE, BF->C}.

Las claves del esquema según F son: BD y BA. Luego el esquema no respeta la FNBC. (ninguna dependencia la respeta).[1]

Para descomponer, "conviene" (es solo una heurística) elegir reglas que del lado derecho no tengan atributos que pertenezcan a una clave. Es decir, intentar mantener las claves.

Elegimos para descomponer a la dependencia BF->C, por lo que obtenemos dos esquemas: BFC y ABDEF.

[1] El esquema está en 1FN, ya que D->E no respeta la 2FN.

En el esquema BFC se proyecta únicamente la regla BF->C. La clave de este esquema es BF, y la única regla respeta FNBC. Luego **BFC ya está en FNBC.**

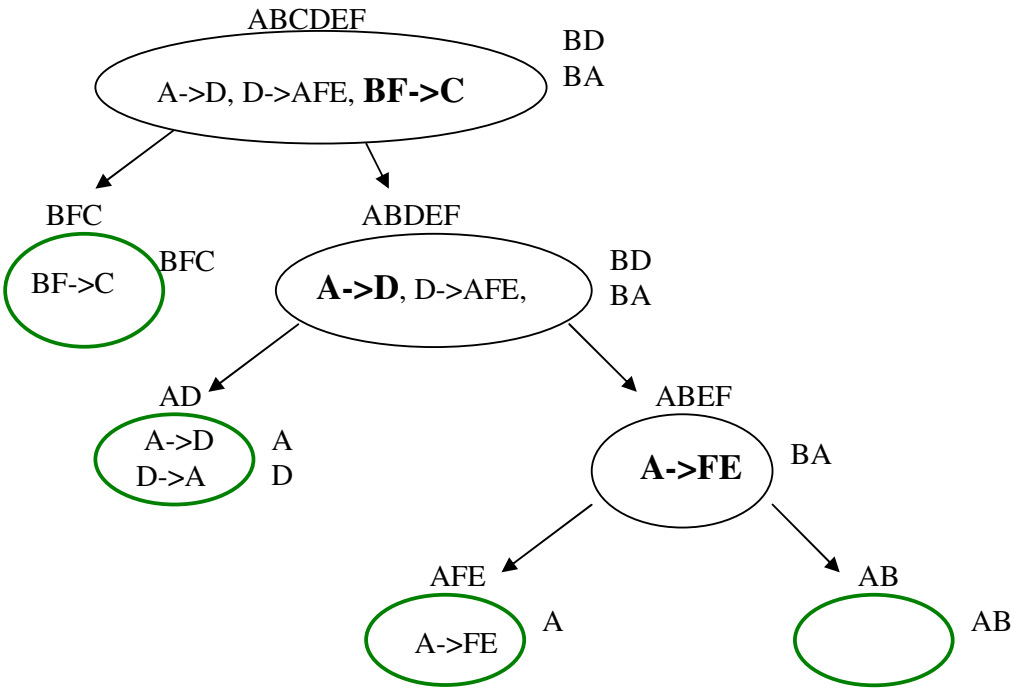
En el esquema ABDEF nos quedan las reglas A->D y D->AFE, con claves BD y BA. Todavía no está en FNBC, por lo que descomponemos nuevamente.

Elegimos la regla A->D y obtenemos dos esquemas: AD y ABEF.
En AD se nos proyectan A->D y D->A (por descomposición D->AFE se puede pensar como D->A, y D->FE). En este esquema A y D son claves, y ambas dependencias respetan FNBC, luego **AD ya está en FNBC.**

En el esquema ABEF, pareciera inicialmente que no se proyecta ninguna dependencia. Sin embargo, A->FE pertenece a F^+, ya que teníamos las reglas A->D y D->AFE(por transitiva, A->FE). Como para la proyección tenemos que tener en cuenta F^+ y no F, tenemos que proyectar A->FE sobre el esquema ABEF.
A->FE es la única dependencia en el esquema ABEF. La clave es BA. Luego, como no respeta FNBC volvemos a descomponer.

Obtenemos un esquema **AFE** y otro **AB**. En el primero se proyecta A->FE y en el segundo no se proyecta ninguna dependencia. **Es fácil verificar que ambos respetan FNBC.** Luego, la descomposición final que respeta FNBC y es SPI es: (BFC,AD,AFE,AB). En este caso, la descomposición tampoco pierde dependencias funcionales, pero esta situación no siempre pasa.

Gráficamente:



Determinar Forma Normal de un Esquema

Para determinar la forma normal de un esquema es fundamental obtener todas las claves del mismo, para así obtener el conjunto de atributos Primos y los No Primos. Luego, analizando las dependencias funcionales determinamos que forma normal respeta el esquema.

FNBC: Para determinar si un esquema cumple FNBC sólo es necesario observar los lados izquierdos de las dependencias.
Todos deben ser de la forma: **SuperClave -> Atributos**.
Ejemplo:
En un esquema R (ABCDE) con claves A y BC, las reglas A->D, BC->E, BCE->D y AE->DC respetan FNBC. En cambio, la regla BD->C ya no la cumple.

3FN: Esta forma normal es más flexible que la FNBC. Exige que todo lado izquierdo sea superclave, o que el lado derecho pertenezca al conjunto de Primos. Siguiendo con el ejemplo anterior, ahora tenemos que la regla BD->C respeta la 3FN, ya que si bien BD no es superclave, tenemos que C pertenece a los atributos Primos. Una regla que no respete 3FN sería por ejemplo BD->DE, ya que ninguna de las dos condiciones se cumple: ni BA es superclave ni DE son primos.

2FN: La segunda forma normal es quizás la más difícil de determinar. La regla nos pide que ningún atributo no primo dependa parcialmente de alguna clave. Una manera simple de buscar dependencias que **no** respeten 2FN es buscar el siguiente patrón:
Subconjunto Estricto de Alguna Clave -> Atributo(s) No primo(s).
Siguiendo con el mismo ejemplo, tenemos que B->D no respeta la 2FN. (B es subconjunto de BC, y D no es primo). En cambio, B->C si la respeta, ya que C es primo, y no puedo aplicar el patrón mencionado. De la misma forma, A->D o BC->D también cumplen 2FN ya que en los lados izquierdos no tengo subconjuntos estrictos de una clave.

Todo esquema en FNBC está en 3FN, y todo esquema en 3FN está en 2FN. Asimismo, un esquema que no está en 2FN, no estará en 3FN, y un esquema que no respeta 3FN, tampoco respetará FNBC. Todos los esquemas que vemos en la materia asumimos que cumplen la 1FN.

Importante: Siempre se espera obtener como respuesta la máxima categoría a la cual corresponde un esquema (donde FNBC es la máxima y 1FN la mínima). Por ejemplo, para responder que está en 3FN tenemos que mostrar que cumple las reglas de 3FN, y que al menos existe una dependencia que no cumple la FNBC. Igualmente, si decimos que un esquema está en 2FN, debemos mostrar que se satisface las condiciones para 2FN, y que existe una dependencia que no respeta 3FN. Finalmente, si decimos que un esquema está en 1FN, debemos mostrar una dependencia que no respete la 2FN.