Síntesis de la Teoría de Normalización

Soraya Abad Mota

Primera Edición: Marzo 2013
Versión 2

Se presenta un resumen de los tópicos de la Teoría de Normalización como se cubren en el curso CI-3311. Se sigue el texto básico del curso: "Fundamentals of Database Systems"de Elmasri y Navathe, 5ta edición, Addison-Wesley 2007, pero se modifica el orden de algunos temas y se agregan ejemplos propios.

Se construyó este documento en base a las láminas escritas por la Prof. Abad en las diferentes oportunidades que ha dictado la asignatura. Las definiciones y algoritmos de la teoría de normalización se tradujeron al castellano a partir de los contenidos del libro de texto.

Son bienvenidas las sugerencias de cambio y la detección de errores en esta breve síntesis.

Índice

1. Introducción **3**

2. Problemas de diseño de las relaciones **3**

3. Dependencias funcionales y otras definiciones **4**

3.1. Superclaves y claves 5

3.2. Dependencia Funcional 5

3.2.1. Definiciones adicionales 6

3.2.2. Reglas de Inferencia de dependencias funcionales 6

3.2.3. Algoritmo de clausura de atributos 6

3.2.4. Algoritmo 11.4(a) 7

4. Teoría de Normalización **7**

4.1. Las primeras formas normales 7

4.1.1. Definiciones generales de 2NF y 3NF 8

4.2. Definiciones y algoritmos sobre conjuntos de dependencias funcionales 9

4.2.1. Cobertura Mínima: definición y algoritmo 10.2 9

4.2.2. Algoritmo 11.4: Descomposición en relaciones en 3FN que preservan las dependencias y tienen joins no aditivos. 10

5. La forma normal BCNF **10**

5.1. Algoritmo 11.3 11

6. Criterios de bondad de las descomposiciones **11**

6.1. Proyección de atributos 12

6.2. Preservación de dependencias funcionales 12

6.3. Propiedad de joins no aditivos 12

6.4. Ejemplo Patológico 13

6.5. Conclusión sobre los algoritmos existentes 13

1. Introducción

La teoría de normalización provee mecanismos para garantizar que las relaciones de una base de datos relacional tengan un buen diseño.

En este documento se sintetizan los aspectos de normalización cubiertos en las clases de CI3311, en el orden en el cual fueron tratados. La fuente principal para cubrir estos aspectos es el libro de texto de Navathe y Elmasri, "Fundamentals of Database Systems", en su quinta edición, Addison-Wesley 2007, varios ejemplos son propios de la Prof. Abad Mota.

2. Problemas de diseño de las relaciones

Hay cuatro problemas que se pueden presentar en una base de datos relacional, si las relaciones no están bien diseñadas, esos problemas son:

1. Mala semántica de una relación.
2. Información redundante en las tuplas de una relación. Aparte de ocupar espacio adicional, la redundancia puede producir problemas de actualización; cada una de las tres operaciones de actualización, a saber: inserción, modificación y eliminación, puede producir anomalías indeseables.
3. Valores Nulos. Un esquema de relación que se haya diseñado con nulos sitemáticos es muy inconveniente, además del desperdicio de espacio, todas las operaciones del álgebra relacional se complican ante su presencia, por ejemplo, el join no toma en cuenta las tuplas que tienen un valor nulo en el atributo por el cual se hace join. Los manejadores de base de datos relacionales enfrentan de forma diferente y no uniforme, la presencia de nulos, las funciones de agregación tienen semántica variada o indefinida cuando el atributo sobre el cual se aplican, puede tomar el valor nulo. En la discusón de esta teoría, vamos a suponer que los atributos no pueden tomar el valor nulo.
4. Generación de tuplas ficticias. Una relación que se trata de descomponer en varias relaciones con los atributos de la relación original, puede proveer información falsa, al hacer el join de las relaciones más pequeñas, generando tuplas que no existían en la relación original.

Considere la relación $LIBRO(\underline{isbn}, titulo, autor, fechaNacA, sexoA, emailA, genero, idioma)$ los atributos titulo, genero e idioma, son propios del libro, pero fechaNacA, sexoA, emailA son propios del autor del libro. Si suponemos que cada libro tiene un único autor (por lo cual isbn puede ser clave de LIBRO), hay redundancia en los atributos de autor, pues todos aquellos autores que hayan escrito varios libros, tendrán repetidos todos sus atributos. Aquellos autores que hayan escrito muchos libros tendrán mucha redundancia pues para cada tupla de LIBRO, se repetirán los cuatro atributos de autor.

Esta redundancia puede producir las tres anomalías de actualización, a saber: inserción, si se quiere insertar los datos de un nuevo autor que todavía no ha escrito libro alguno, no se puede guardar esa información en LIBRO, pues no se tendría un valor para la clave primaria de LIBRO; si se elimina el único libro de un autor, se pierden los datos personales de ese autor, y si se modifican el email de un autor que ha escrito varios libros, puede que "se olvide" cambiar el email de ese autor en alguna de las tuplas de sus libros, con lo cual se generaría una inconsistencia en los datos de los libros de ese autor.

Para ilustrar el problema 4 de los enumerados inicialmente, considere la relación ELECCION(CIcandidato, siglasEstado, fechaEleccion), un ejemplo de instancia de esta relación se muestra en el cuadro 3.

Cuadro 1: Instancia Ejemplo de ELECCION

CIcandidato	siglasEstado	fechaEleccion
5	MI	7-oct-2012
1	AR	7-oct-2012
2	MI	16-dic-2012

Si los atributos de esta relación se agrupan de manera distinta en las relaciones CE(CIcandidato, siglasEstado) y EE(siglasEstado, fechaEleccion). Al proyectar la instancia de ELECCION mostrada en el cuadro 3 se muestran en el cuadro. 2.

Cuadro 2: Instancias de ELECCION proyectadas sobre CE y EE

CIcandidato	siglasEstado
5	MI
1	AR
2	MI

siglasEstado	fechaEleccion
MI	7-oct-2012
AR	7-oct-2012
MI	16-dic-2012

Al hacer el join de las relaciones CE y EE, deberíamos obtener la relación ELECCION original, sin embargo, la instancia que se obtiene con el join es la que se muestra en el cuadro 3. Las tuplas marcadas con x son tuplas ficticias, generadas por una mala descomposición de la relación original.

Cuadro 3: Instancia resultante de hacer CE * EE

	CIcandidato	siglasEstado	fechaEleccion
	5	MI	7-oct-2012
x	5	MI	16-dic-2012
	1	AR	7-oct-2012
x	2	MI	7-oct-2012
	2	MI	16-dic-2012

Todos estos problemas se pueden evitar siguiendo los preceptos de la teoría de normalización que se cubre en las próximas secciones.

3. Dependencias funcionales y otras definiciones

En esta sección se dan las definiciones de los conceptos básicos en los cuales se apoya la teoría de normalización. Los primeros conceptos ya se definieron antes, en el modelo relacional.

En todas las definiciones de las secciones que siguen, se supone que existe una relación $R(A_1,\ A_2,\ ...,\ A_n)$, con $r\ =\ instancia\ de\ R$ y $A = A_1,\ A_2,\ ...,\ A_n$.

3.1. Superclaves y claves

Superclave: $S \subseteq A$ es una *superclave sii* $\forall\ t_1, t_2\ in\ r,\ t_1[S] \neq t_2[S]$

Clave: Sea $K \subseteq A$ una superclave, K es una *clave* si además es *minimal*, es decir, si al eliminar cualquier atributo de K, el conjunto de atributos resultante ya no es más una superclave.

Candidato a clave: si R tiene más de una clave, a cada una de ellas se le llama *candidato a clave*.

Atributo primo: es un atributo A_i que es miembro de algún candidato a clave de R.

Atributo no primo: es un atributo A_j que *no es miembro* de candidato a clave alguno de R.

3.2. Dependencia Funcional

Sean X y Y dos conjuntos de atributos que existen en una base de datos, en el esquema R.
Si para todo par de tuplas t_1 y t_2 en *cualquier instancia posible* r de R, se cumple la siguiente restricción:

$$t_1[X] = t_2[X] \implies t_1[Y] = t_2[Y],$$

en otras palabras, si que las dos tuplas tengan el mismo valor en los atributos de X implica que tienen los mismos valores en los atributos de Y, (si t_1 y t_2 coinciden en X implica que también coinciden en Y)
entonces,
existe una dependencia funcional entre X y Y, la cual se denota como

$$X \longrightarrow Y$$

Si $X \longrightarrow Y$ decimos:

1. Los valores de X determinan unívocamente los valores de Y.
2. X *determina funcionalmente a* Y.
3. Hay una dependencia funcional desde X hasta Y.
4. Y es *funcionalmente dependiente de* X.

Dependencia funcional se abrevia como *DF*. En $X \longrightarrow Y$, X es el *lado izquierdo de la DF* y Y es el *lado derecho de la DF*.

NOTAS:

- Sea $R(K, Y)$ un esquema de relación, donde K es un candidato a clave de R y Y es el conjunto de todos los otros atributos de R, entonces se cumple que: $K \longrightarrow Y$.
- El hecho de que $X \longrightarrow Y$ *no dice nada acerca de* $Y \longrightarrow X$.
- Si K es candidato a clave de R, esto implica que $K \longrightarrow Z$, donde Z es cualquier subconjunto de los atributos de R.

Significado de las dependencias funcionales:
Una dependencia funcional es una propiedad de la semántica de los atributos. Las dd.ff. dicen cómo los atributos están relacionados unos con los otros, semánticamente.

¿Cómo determinamos las dependencias funcionales? (elija una)

1. ¿Analizando la semántica de los datos? SI
2. ¿Observando una extensión (instancia) específica de la relación y viendo qué pasa en las tuplas? NO

Esto es porque:

- Las dd.ff. dicen cuáles son las *extensiones legales* de las relaciones. Esas extensiones en las cuales las dd.ff. no se cumplen, no son legales.
- Las dd.ff. especifican restricciones en los atributos de una relación que deben cumplirse *siempre*.
- Las dd.ff. restringen los posibles valores de los atributos.

3.2.1. Definiciones adicionales

- Dependencia funcional trivial: una dependencia funcional $X \longrightarrow Y$ es trivial si $Y \subset X$ de lo contrario (Si Y no es subconjunto de X) es no trivial. Sea $R(A_1, A_2, A_3, A_4)$, $A_1 \longrightarrow A2$ y $A_1 \longrightarrow A3$ son dependencias funcionales no triviales, pero $A_2 \longrightarrow A2$ y $A_1A_2 \longrightarrow A2$ son dependencias funcionales triviales.
- Dependencia funcional completa: una DF $X \longrightarrow Y$ es *completa* si al quitar cualquier atributo de X, la dependencia ya no se cumple. Por ejemplo, en $TRIMESTRE(ciE, carnetE, nombreE, codigoA, descA, carrera, periodo, calificacion)$, la DF $carnetE, periodo, codigoA \longrightarrow calificacion$ es completa.
- Dependencia funcional parcial: si en la DF $X \longrightarrow Y$, al quitar algún atributo de X, la DF todavía se cumple, entonces se dice que la DF es parcial. En el ejemplo anterior, la DF $carnetE, periodo, codigoA \longrightarrow descA$ es parcial pues si se quitan los atributos $carnetE$ y $periodo$ la DF todavía se cumple, es decir $codigoA \longrightarrow descA$.
- Dependencia Funcional Transitiva: $X \longrightarrow Y$ es una DF transitiva si existe un conjunto de atributos Z, que no es un subconjunto de la clave primaria de R y se cumplen las siguientes dos dependencias funcionales: $X \longrightarrow Z$ y $Z \longrightarrow Y$. Por ejemplo, la DF $isbn \longrightarrow emailA$ es transitiva porque $isbn \longrightarrow autor$ y $autor \longrightarrow emailA$.

3.2.2. Reglas de Inferencia de dependencias funcionales

Son seis reglas para deducir nuevas dependencias funcionales, las tres de Armstrong que constituyen un conjunto *sound and complete* de reglas, a saber: la reflexiva, la aumentativa y la transitiva y otras tres, convenientes para los algoritmos que utilizamos en normalización, la de descomposición, la aditiva y la pseudo transitiva. (ver las reglas completas en la 5ta edición del libro de texto).

3.2.3. Algoritmo de clausura de atributos

Algoritmo 10.1 de cálculo de X^+ (ver el algoritmo en la 5ta edición del libro de texto).

3.2.4. Algoritmo 11.4(a)

Es muy útil tener un mecanismo sistemático de encontrar claves para R, en el texto hay un algoritmo, el 11.4(a) que recibe como entrada el esquema de la relacin $R(X)$ y el conjunto F de dependencias funcionales que se cumplen en R y encuentra una clave K. (ver el algoritmo en la 5ta edición del libro de texto).

4. Teoría de Normalización

Desarrollada dentro del modelo relacional para:

1. Verificar si alguna de las situaciones patológicas se presenta en las relaciones de una BD relacional.
2. Descomponer las relaciones *mal diseñadas* en dos o más relaciones *buenas*.

Utiliza las definiciones relativas a las *claves* y las *dependencias funcionales* para establecer el concepto de *Forma Normal (NF)*. Una forma normal define condiciones que debe cumplir una relación para estar en esa forma normal. Las formas normales que vamos a estudiar son: 1NF, 2NF, 3NF y BCNF. En una primera aproximación se cubren las tres primeras formas normales (1NF, 2NF y 3NF), luego se definen algunos conceptos necesarios de los conjuntos de depencias funcionales y algunos algoritmos útiles y finalmente se aborda la forma normal BCNF.

4.1. Las primeras formas normales

En esta subsección se comienza con la definición básica de las primeras tres formas normales. Las definiciones básicas de 2NF y 3NF suponen que la relación tiene un solo candidato a clave; luego se generalizan esas definiciones, al caso de una relación con más de un candidato a clave.

Primera Forma Normal (1NF): una relación está en 1NF si todos sus atributos sólo pueden tomar valores atómicos. En el modelo relacional, todas las relaciones, por definición están en 1NF.

Se considera la relación $LIBRO1(isbn, titulo, autor, fechaNacA, sexoA, emailA, genero, idioma)$ donde se han establecido las siguientes dependencias funcionales:
$F1 = \{isbn \longrightarrow titulo, genero, idioma;\quad autor \longrightarrow fechaNacA, sexoA, emailA\}$

¿Cuál es la clave primaria de la relación $LIBRO1$? la combinación de los atributos $\{isbn, autor\}$
Se puede ver que $(isbn, autor)^+ = \{isbn, autor, titulo, fechaNacA, sexoA, emailA, genero, idioma\}$

Segunda Forma Normal (2NF): una relación R está en 2NF si todo atributo no primo de R es dependiente funcional y completamente de la clave primaria.

Si R no está en 2NF, se descompone en varias relaciones que sí están en 2NF, pero esta descomposición debe cumplir ciertas condiciones para que sea buena, pronto veremos cuáles son esas condiciones.

$LIBRO1$ no está en 2NF porque la dependencia $isbn, autor \longrightarrow fechaNacA, sexoA, emailA$ no es completa. Se puede eliminar el isbn y todavía se cumpliría la DF. De la misma forma, se puede eliminar *autor* de ls DF $isbn, autor \longrightarrow titulo, genero, idioma$ y todavía queda una DF válida para LIBRO. (Bastaba con verificar una sola de estas dos situaciones para afirmar que no está en 2NF.)

Tercera Forma Normal (3NF): una relación R está en 3NF si:

- está en 2NF y
- ningún atributo no primo de R es transitivamente dependiente de la clave primaria.

Si consideramos ahora la relación $LIBRO(isbn, titulo, autor, fechaNacA, sexoA, emailA, genero, idioma)$ con los mismos atributos pero un conjunto de dependencias F ligeramente distinto, a saber: $F = \{isbn \longrightarrow titulo, autor, genero, idioma;\quad autor \longrightarrow fechaNacA, sexoA, emailA\}$
En este caso, $LIBRO$ tiene como clave solamente el $isbn$ y está en 2NF.

¿está en 3NF?
No, porque la dependencia $isbn \longrightarrow fechaNacA, sexoA, emailA$ de la clave primaria, es transitiva, ninguno de los atributos del lado derecho de esta dependencia son primos, Z es autor y se cumplen las siguientes dd.ff: $isbn \longrightarrow autor$ y $autor \longrightarrow fechaNacA, sexoA, emailA$.

El procedimiento a seguir al descubrir una relación que no está en 3NF, es descomponerla en varias relaciones que sí están en 3NF.

$LIBRO$ se puede descomponer en:
$L1(\underline{isbn}, titulo, genero, idioma, autor)$ y $L2(\underline{autor}, fechaNacA, sexoA, emailA)$ ambas relaciones: L1 y L2 están en 3NF. El atributo $autor$ en $L1$ es clave foránea que referencia a $L2$.
En la subsección 4.2.2 se presenta un algoritmo que produce descomposiciones en 3NF con buenas propiedades.

4.1.1. Definiciones generales de 2NF y 3NF

Las definiciones anteriores no consideran si la relación tiene *más de un candidato a clave.*
Consideremos la siguiente relación:
ESTUDIANTE (Carnet, nombre, CI, carrera, dir, tel, fechaNac), donde se cumplen las siguentes dd.ff.
$F_{ESTUDIANTE} = \{Carnet \longrightarrow nombre, CI, carrera, dir, tel, fechaNac;\quad CI \longrightarrow Carnet\}$

¿está en 2NF?
Si, todo atributo no primo (nombre, CI, carrera, dir, tel, fechaNac) es funcional y completamente dependiente de la clave primaria.

¿está en 3NF?
está en 2FN. ¿CI es un atributo primo o no primo?
CI es primo, $Carnet \longrightarrow CI$ y $CI \longrightarrow nombre$
en consecuencia, $Carnet \longrightarrow nombre$ es una dependencia funcional transitiva. No está en 3FN.
Pero en realidad no hay problema, pues CI también es una clave.

Cuando hay varios candidatos a clave en R, es necesario usar definiciones más generales de las formas normales que los tome en cuenta a todos.

DEFINICIONES GENERALES
Segunda Forma Normal (2NF): un esquema de relación, R, está en 2NF si todo atributo no primo A de R es dependiente funcional y completamente de todas los candidatos a clave de R.
Tercera Forma Normal (3NF): un esquema de relación R está en 3NF si para toda dependencia funcional, $X \longrightarrow A$ que se da en R, se cumple alguna de las dos condiciones siguientes: X es una superclave de R, o A es un atributo primo de R.

De acuerdo con estas definiciones generales, la relación $ESTUDIANTE$ presentada anteriormente si está en 3NF, porque las dos dependencias funcionales en $F_{ESTUDIANTE}$ cumplen con que el lado izquierdo de cada DF es una superclave, en particular, es una clave.

4.2. Definiciones y algoritmos sobre conjuntos de dependencias funcionales

- Cobertura de conjuntos de dependencias funcionales. Sean E y F conjuntos de dependencias funcionales, se dice que E es cubierto por F (o F cubre a E) si toda dependencia funcional en E también está en F^+. (En otras palabras, si toda dependencia en E puede inferirse de las dependencias en F, usando las reglas de inferencia.)
- ¿Cómo se determina si F cubre a E?
  - **(a)** Para toda dependencia funcional $X \longrightarrow Y$ en E, se calcula X^+ con respecto a F.
  - **(b)** Si para cada dependencia funcional considerada en (a), se cumple que X^+ incluye todos los atributos de Y, entonces F cubre a E.
- Equivalencia de conjuntos de dependencias funcionales.E y F son equivalentes si $E^+ = F^+$, en otras palabras, si E cubre a F y F cubre a E.

4.2.1. Cobertura Mínima: definición y algoritmo 10.2

- Conjunto "minimal" de dependencias funcionales. F es "minimal" si satisface estas condiciones:
  1. Toda DF tiene un único atributo del lado derecho.
  2. No podemos quitar ninguna DF de F y todavía tener un conjunto de dependencias equivalente a F.
  3. No podemos reemplazar ninguna DF de la forma $X \longrightarrow A$ en F, por otra $Y \longrightarrow A$ donde $Y \subset X$ y todavía tener un conjunto de dependencias equivalente a F. (Es decir, no sobra atributo alguno del lado izquierdo de la DF.)
- Cobertura mínima de F. Es un conjunto "minimal" de dependencias funcionales que es equivalente a F. Existen varias coberturas mínimas de F, hay un algoritmo para conseguir una, el cual se muestra a continuación.)

Algoritmo 10.2: Cobertura Mínima, F, de un conjunto de dependencias funcionales E

1. Coloque en F todas las dependencias de E: $F := E$
2. Reemplace cada DF de la forma $X \longrightarrow \{A_1, A_2, ..., A_n\}$ en F por las n DF's siguientes $X \longrightarrow A_1,\ X \longrightarrow A_2, ..., X \longrightarrow A_n$.
3. Para cada DF $X \longrightarrow A$ en F,
   para cada atributo B que es un elemento de X,
   Si $\{F - \{X \longrightarrow A\}\} \cup \{(X - \{B\}) \longrightarrow A\}$ es equivalente a F
   entonces reemplace $X \longrightarrow A$ por $(X - \{B\}) \longrightarrow A$ en F.
4. Para cada DF $X \longrightarrow A$ remanente en F
   si $\{F - \{X \longrightarrow A\}\}$ es equivalente a F, entonces elimine $X \longrightarrow A$ de F.

4.2.2. Algoritmo 11.4: Descomposición en relaciones en 3FN que preservan las dependencias y tienen joins no aditivos.

Entrada: R y F (dependencias funcionales sobre R)

Salida: D descomposición de R

1. Consiga una cobertura mínima G de F.
2. Para cada lado izquierdo X de una DF de G construya un esquema de relación en D que tenga los atributos $\{X \cup \{A_1\}... \cup \{A_k\}\}$ donde $X \longrightarrow A_1,\ X \longrightarrow A_2,\ ..., X \longrightarrow A_k$ son las únicas DFs en G que tienen a X como lado izquierdo. (X es la clave de esta relación.)
3. Si no existe un esquema de relación en D que contenga un clave de R, cree una relación más en D que contenga todos los atributos que forman una clave de R.
4. Elimine las relaciones redundantes en D. Una relación es redundante si se obtiene al hacer un project sobre otra relación del mismo esquema relacional.

Ejercicio: aplique este algoritmo a la relación $LIBRO1$ presentada en la subsección 4.1. ¿Cuáles relaciones obtiene? Compárelas con las relaciones $L1$ y $L2$.

5. La forma normal BCNF

Consideremos la siguiente relación:
ALCALDIA (idPropiedad, nombreMunicipio, nroTerreno, area)

Las dependencias funcionales que se cumplen en esta relación son:

$$DF1:\ idPropiedad \longrightarrow nombreMunicipio,\ nroTerreno,\ area$$

$$DF2:\ nombreMunicipio,\ nroTerreno \longrightarrow idPropiedad,\ area$$

$$DF3:\ area \longrightarrow nombreMunicipio$$

¿está en 3NF?
Usando la definición general de 3NF, debemos verificar para cada DF, que el lado izquierdo sea superclave o que el lado derecho sea primo.

Si está en 3NF.
En DF1 y en DF2, el lado izquierdo es superclave.
En DF3, el lado derecho es primo.

¿Hay algún problema con esta relación ALCALDIA?
Si, hay redundancia en el nombre del Municipio, pues está determinado por el área y ¿si sólo hay dos áreas posibles y muchos municipios?

Boyce-Codd Normal Form (BCNF): un esquema de relación R está en BCNF si para toda dependencia funcional, $X \longrightarrow A$ que se da en R, se cumple que:

- X es una superclave de R.

BCNF es más fuerte que 3NF. Pues la forma normal 3NF permite que haya una dependencia funcional donde el lado izquierdo no sea una superclave, si el lado derecho es primo.

El problema con la relación $ALCALDIA$ es que no está en BCNF. Una SOLUCIÓN sería descomponer la relación en:
ALCALDIA1 (idPropiedad, nroTerreno, area)
AREA_MUN (area, nombreMunicipio)
El atributo *area* en $ALCALDIA1$ es clave foránea que referencia a $AREA\_MUN$.

El problema con esta descomposición es que se perdió la DF2 original; habrá que implementar un mecanismo para que se controle la integridad que provee esa dependencia, pero es preferible usar la descomposición y no la relación original, pues la original contiene redundancia, la cual es peligrosa por la posibilidad de generar inconsistencias.

5.1. Algoritmo 11.3

En el libro de texto hay un algoritmo, el 11.3, que permite descomponer una relación en relaciones que están en BCNF con la garantía de que la descomposición resultante tiene la propiedad de joins no aditivos. No existe un algoritmo como el de 3NF que garantice las dos propiedades, la de joins no aditivos y la de preservación de dependencias. De modo que al llevar las relaciones a BCNF en algunos casos pueden perderse dependencias, pero el algoritmo 11.3 garantiza que no habr joins aditivos en la descomposición resultante.

Entrada: R y F (dependencias funcionales sobre los atributos de R)

Salida: D descomposición de R

1. Asigne a D la relación R, $D := \{R\}$
2. Mientras exista un esquema de relación Q en D que no esté en BCNF, haga lo siguiente:
   - escoja una relación Q en D que no está en BCNF;
   - encuentre una dependencia funcional de la forma $X \longrightarrow Y$ que viola la definición de BCNF;
   - reemplace Q en D por los siguientes dos esquemas de relación $Q1(Q - Y)$ y $Q2(X \cup Y)$.

Este fue el procedimiento aplicado para producir la descomposición de ALCALDIA, al principio de la sección 5 para producir las relaciones ALCALDIA1 y AREA_MUN.

6. Criterios de bondad de las descomposiciones

Los principales criterios para determinar la bondad de una descomposición de una relación en varias relaciones son:

- Preservación de atributos.
- Preservación de dependencias funcionales.
- Descomposiciones que producen joins no aditivos (*lossless join property*).

En las próximas subsecciones, se cubren las definiciones necesarias para verificar estos criterios en una descomposición. En toda la sección se considera el esquema de relación $R(A_1, A_2, ..., A_n)$ que será descompuesto en m relaciones, la descomposición de R se llama D y consiste de $D = \{R_1, R_2, ..., R_m\}$

6.1. Proyección de atributos

Se dice que la descomposición D preserva los atributos de R si todo atributo A_j de R aparece en al menos una R_i de D. La unión de todos los esquemas de relación en D, da un esquema de relación igual a R. Es decir, $\bigcup_{i=1}^{m} R_i = R$

6.2. Preservación de dependencias funcionales

Al descomponer R para llevarla a formas normales deseables para evitar la redundancia en los datos, se debe cuidar que las dd.ff. que se cumplían en R, se sigan cumpliendo en la descomposición y no se pierda ninguna. Las dependencias funcionales actúan como restricciones de integridad de los datos, es bueno mantenerlas, aún al descomponer una relación, pero no siempre es posible.

Para poder verificar si se mantienen las dependencias funcionales en la descomposición de una relación, se utiliza la noción de *proyección de dependencias* y se verifica si con estas proyecciones se mantienen todas las dependencias originales.

Proyección de dependencias funcionales
F es el conjunto de dd. ff. que se cumplen en R y $D = \{R_1, R_2, ..., R_m\}$ es una descomposición de R.

La Proyección de F sobre R_i, $\pi_{R_i}(F)$ es el conjunto de dependencias funcionales de la forma $X \longrightarrow Y$ en F^+ tales que los atributos en $X \cup Y \subset R_i$.

Descomposición que preserva las dependencias. Una descomposición D de R preserva las dependencias funcionales con respecto a F si la unión de las proyeccioes de F sobre cada $R_i \in D$ es equivalente a F, es decir $(\pi_{R_1}(F) \cup \pi_{R_2}(F) \cup ... \cup \pi_{R_m}(F))^+ = F^+$.

6.3. Propiedad de joins no aditivos

A la característica de NO generar tuplas ficticias que puede tener una descomposición, se le llama *descomposición con joins no aditivos* o "lossless join property".

Definición: una descomposición $D = \{R_1, R_2, ..., R_m\}$ tiene la propiedad de *joins no aditivos* con respecto al conjunto de dependencias funcionales F de R, si para toda instancia r de R que satisface F, se cumple que el join natural de todas las relaciones en D es igual a r, es decir $*(\Pi_{R_1}(r), ..., \pi_{R_m}(r)) = r$

Dos propiedades especiales tienen las descomposiciones con joins no aditivos, a saber:

LJ1: Una descomposición $D = \{R_1, R_2\}$ de R tiene joins no aditivos con respecto al conjunto de dependencias F que se cumplen en R, si y solo si se cumple una de estas condiciones:

- la DF $(R_1 \cap R_2) \longrightarrow (R_1 - R_2)$ está en F^+, o
- la DF $(R_1 \cap R_2) \longrightarrow (R_2 - R_1)$ está en F^+

LJ2: Si $D = \{R_1, R_2, ..., R_m\}$ tiene jons no aditivos con respecto a F de R y si $D_i = \{Q_1, ..., Q_n\}$ es una descomposición de R_i que también tiene joins no aditivos, entonces la descomposición D_2 de R siguiente $D_2 = \{R_1, R_2, ..., R_{i-1}, Q_1, ..., Q_n, R_{i+1}, ..., R_m\}$, también tiene joins no aditivos con respecto a F.

Al tratar de comprobar esta propiedad en el ejemplo de la ALCALDIA de la sección 5 se puede ver que el atributo común a ALCALDIA1 y AREA_MUN es *area* y ese atributo es clave para AREA_MUN por lo tanto todos los atributos de AREA_MUN dependen funcionalmente del atributo *area*. ¿Qué pasa si los atributos que pertenecen a $(R_1 \cap R_2)$ son una superclave para R_1 o para R_2, se cumple la propiedad? (Verifique en el ejercicio 4 de los ejercicios publicados en la página del curso.)

6.4. Ejemplo Patológico

Hay un ejemplo patológico de relación que lo presentó originalmente Jeffrey D. Ullman [1], que es útil para ilustrar el problema de tratar de llevar una relación a la forma normal BCNF. A continuación se presenta esa relación y una discusión acerca de cómo llevarla a BCNF.

$TEACH(Instructor, Curso, Estudiante)$, con las siguientes dependencias funcionales
$F = \{Estudiante, Curso \longrightarrow Instructor;\ Instructor \longrightarrow Curso\}$

$TEACH$ si está en 3NF (verificar), pero no se encuentra en BCNF porque en la DF $Instructor \longrightarrow Curso$ el lado izquierdo no constituye una superclave de la relación.

Las posibles descomposiciones de TEACH son las siguientes:
$D1 = T1(Estudiante, Instructor)$ y $T2(Estudiante, Curso)$
$D2 = T1^a(Instructor, Curso)$ y $T2^a(Curso, Estudiante)$
$D3 = T1^b(Instructor, Curso)$ y $T2^b(Instructor, Estudiante,)$

Evalúe las bondades de cada descomposición, es decir, verifique cuáles pierden dependencias funcionales y cuáles son esas dependencias perdidas, verifique cuáles generan tuplas ficticias, ¿hay alguna descomposición que no genere tuplas ficticias? Responda a estas preguntas formalmente, utilizando los conceptos descritos en esta sección 6.

6.5. Conclusión sobre los algoritmos existentes

Hay un algoritmo (el 11.1 del libro de texto) que permite verificar si una descomposición D tiene la propiedad de joins no aditivos. En el curso de CI-3311 no se cubre ese algoritmo, en su lugar, se verifican las dos propiedades LJ1 y LJ2, de las descomposiciones.

Existen varios algoritmos para llegar a descomposiciones en 3NF o BCNF con buenas propiedades.

En el caso de la 3NF hay un algoritmo, el 11.4 presentado en la subsección 4.2.2 que descompone la relación original en relaciones que están en 3NF y tiene las tres características deseables, a saber: preserva todos los atributos y todas las dependencias funcionales y tiene la propiedad de joins no aditivos. De modo que vale la pena llevar todas las relaciones a 3NF pues garantiza que no existen los problemas descritos en la sección 2 y tiene todas las bondades de la descomposición indicadas en esta sección.

Para BCNF, el algoritmo 11.3 presentado en la sección 5.1, descompone la relación original en varias relaciones que están en BCNF con la propiedad de tener joins no aditivos, pero no se puede garantizar que

[1] en el libro "Principles of Database Systems" de J. Ullman, Computer Science Press 1980

preserva todas las dependencias funcionales. Sin embargo, las dependencias funcionales que no se cumplan con la descomposición se pueden hacer cumplir con algún procedimiento explícito de control de integridad, como los triggers o algún otro mecanismo que provea el manejador de base de datos.