Diseño lógico

J. R. Paramá

16 de febrero de 2018

Contenidos

Introducción

- En lugar de usar el modelo E/R como método de diseño, existe una técnica de diseño de BDs denominada diseño ascendente, que es una técnica más purista.
- Implica contemplar el diseño de esquemas de bases de datos utilizando la semántica de los atributos, que es capturada por una serie de relaciones entre atributos (dependencias funcionales, dependencias multivaluadas y dependencias de join).
- Esta técnica nos proporciona un método algorítmico que permitirá crear un "buen" diseño de base de datos.
- También sirve para comprobar con mayor precisión lo "bueno" o "malo" de un esquema de relación individual (que pudo ser diseñado mediante un ER).

Contenidos

Dependencias funcionales

- Una dependencia funcional (df) es una restricción entre dos conjuntos de atributos de la base de datos.
- Una *dependencia funcional*, denotada por $\{X\} \rightarrow \{Y\}$, entre dos conjuntos de atributos X e Y que son subconjuntos de R, especifica una *restricción* sobre las posibles tuplas que podrían formar un estado de relación r de R.
- Para dos tuplas cualesquiera t_1 y t_2, de r tales que $t_1[X] = t_2[X]$
  debemos tener también $t_1[Y] = t_2[Y]$

Dependencias funcionales

- Los valores del componente Y de una tupla de r dependen de los valores del componente X, o están determinados por ellos.
- O bien, los valores del componente X de una tupla determinan de manera única (o funcionalmente) los valores del componente Y.
  - Si una restricción de R dice que no puede haber más de una tupla con un valor de X dado en cualquier instancia de relación $r(R)$, es decir, X es una clave candidata de R, esto implica que $\{X\} \rightarrow \{Y\}$ para cualquier subconjunto de atributos Y de R.
  - Si $\{X\} \rightarrow \{Y\}$ en R, no nos dice si $\{Y\} \rightarrow \{X\}$ o no.

Dependencias funcionales

Supongamos que disponemos de la siguiente información, donde todos los empleados de la misma categoría tienen el mismo sueldo.

- $\{NombreCat\} \rightarrow \{Sueldo\}$
- $\{CodEmp\} \rightarrow \{Nombre, NombreCat, FechNac\}$

Empleados				
CodEmp	Nombre	FechNac	NombreCat	Sueldo
0001	Pepe	13-10-1978	Vendedor	1120
0002	Ana	21-08-1973	Analista	1800
0003	Pedro	12-05-1981	Vendedor	1120
0004	Raquel	23-07-1975	Analista	1800
0005	Raquel	23-05-1975	Gerente	2200

Dependencias funcionales

- Las dependencias funcionales son propiedades de la semántica o del significado de los atributos, no de la extensión (o estado) de una relación en un momento dado.

Dependencias funcionales

Cualquier dependencia se mantendrá en cualquier esquema de relación donde aparezcan esos atributos.

Turnos(CodEmp, Nombre, FechaNac, NombreCat, Sueldo, Fech/HTurno)

¿Dependencias funcionales?

Dependencias funcionales

Cualquier dependencia se mantendrá en cualquier esquema de relación donde aparezcan esos atributos.

Turnos(CodEmp, Nombre, FechaNac, NombreCat, Sueldo, Fech/HTurno)

- $\{NombreCat\} \rightarrow \{Sueldo\}$
- $\{CodEmp\} \rightarrow \{Nombre, NombreCat, FechNac\}$

Se mantienen, dado que estos atributos están en el esquema de relación.

Dependencias funcionales

Ahora podemos comprobar *"visualmente"* que $\{CodEmp\} \rightarrow \{Nombre, NombreCat, FechNac\}$

Turnos					
CodEmp	Nombre	FechNac	NombreCat	Sueldo	Fech/HTurno
0001	Pepe	13-10-78	Vendedor	1120	14-01-16 8:00
0001	Pepe	13-10-78	Vendedor	1120	15-1-16 16:00
0001	Pepe	13-10-78	Vendedor	1120	16-1-16 8:00
0002	Ana	21-08-73	Analista	1800	14-1-16 16:00
0002	Ana	21-08-73	Analista	1800	15-1-16 8:00
0003	Pedro	12-05-81	Vendedor	1120	17-1-16 8:00
0003	Pedro	12-05-81	Vendedor	1120	18-1-16 16:00
0004	Raquel	23-07-75	Analista	1800	18-1-16 16:00
0005	Raquel	23-05-75	Gerente	2200	18-1-16 16:00

Dependencias funcionales

$\{CodEmp\} \rightarrow \{Nombre\}$ pero NO ~~$\{Nombre\} \rightarrow \{CodEmp\}$~~

Turnos					
CodEmp	Nombre	FechNac	NombreCat	Sueldo	Fech/HTurno
0001	Pepe	13-10-78	Vendedor	1120	14-01-16 8:00
0001	Pepe	13-10-78	Vendedor	1120	15-1-16 16:00
0001	Pepe	13-10-78	Vendedor	1120	16-1-16 8:00
0002	Ana	21-08-73	Analista	1800	14-1-16 16:00
0002	Ana	21-08-73	Analista	1800	15-1-16 8:00
0003	Pedro	12-05-81	Vendedor	1120	17-1-16 8:00
0003	Pedro	12-05-81	Vendedor	1120	18-1-16 16:00
0004	Raquel	23-07-75	Analista	1800	18-1-16 16:00
0005	Raquel	23-05-75	Gerente	2200	18-1-16 16:00

Contenidos

Reglas de inferencia para las dependencias funcionales

- Denotamos con F al conjunto de dependencias funcionales que se especifican sobre el esquema de relación R.
- Por lo general, el diseñador del esquema especifica las dfs que son semánticamente obvias, pero es común que muchas otras dfs se cumplan.
- Las otras dfs pueden inferirse o deducirse de las dfs de F.
- El conjunto de todas estas dependencias funcionales se denomina cierre de F y se denota con F^+

Reglas de inferencia para las dependencias funcionales

- Tomemos otra vez las dependencias funcionales del esquema Empleados $F = \{$
  $\{CodEmp\} \rightarrow \{Nombre, NombreCat, FechNac\}$
  $\{NombreCat\} \rightarrow \{Sueldo\}$ $\}$, podemos inferir (entre otras) dfs adicionales a partir de F:
  - $\{CodEmp\} \rightarrow \{NombreCat, Sueldo\}$
  - $\{Sueldo\} \rightarrow \{Sueldo\}$
  - $\{CodEmp, NombreCat\} \rightarrow \{FechaNac\}$

Reglas de inferencia para las dependencias funcionales

- Una df $\{X\} \rightarrow \{Y\}$ se infiere de un conjunto de dfs F especificado sobre R, si siempre que r cumpla F, r satisface $\{X\} \rightarrow \{Y\}$.
- El cierre F^+ de F es el conjunto de todas las dependencias funcionales que pueden inferirse de F.
- Armstrong nos dio un medio para, de forma sistemática, inferir nuevas dependencias por medio de un conjunto de reglas de inferencia.

Reglas de inferencia para las dependencias funcionales

$F \models \{X\} \rightarrow \{Y\}$ quiere decir que la dependencia funcional $\{X\} \rightarrow \{Y\}$ se infiere del conjunto de dfs F.

RI_1 Regla Reflexiva: Si $X \supseteq Y$, entonces $\{X\} \rightarrow \{Y\}$.

RI_2 Regla de Aumento: $\{X\} \rightarrow \{Y\} \models \{XZ\} \rightarrow \{YZ\}$[1].

RI_3 Regla Transitiva: $\{\{X\} \rightarrow \{Y\},\{Y\} \rightarrow \{Z\}\} \models \{X\} \rightarrow \{Z\}$.

[1] La regla de aumento también puede expresarse como $\{X\} \rightarrow \{Y\} \models \{XZ\} \rightarrow \{Y\}$, es decir, aumentar los atributos del miembro izquierdo de una df producirá otra df válida.

Reglas de inferencia para las dependencias funcionales

- Aplicando sistemáticamente estas tres reglas hasta que no se puedan aplicar más se obtiene F^+.
- Los axiomas de Armstrong son correctos porque no generan dfs incorrectas y son completos, porque para un conjunto de dfs F dado, permite generar todo F^+.
- Para simplificar más las cosas se crearon unas reglas adicionales.

Reglas adicionales de inferencia para las dependencias funcionales

Se pueden crear a partir de los axiomas de Armstrong

RI_4 Regla de descomposición o proyectiva:
$\{X\} \rightarrow \{YZ\} \models \{\{X\} \rightarrow \{Y\}, \{X\} \rightarrow \{Z\}\}$.

RI_5 Regla de unión o aditiva: $\{\{X\} \rightarrow \{Y\}, \{X\} \rightarrow \{Z\}\} \models$
$\{X\} \rightarrow \{YZ\}$.

RI_6 Regla pseudotransitiva: $\{\{X\} \rightarrow \{Y\},\{WY\} \rightarrow \{Z\}\} \models$
$\{WX\} \rightarrow \{Z\}$.

Contenidos

Dependencias funcionales

- Una dependencia funcional $\{X\} \rightarrow \{Y\}$ es trivial si $Y \subseteq X$.
- Una dependencia funcional se denomina elemental si su parte derecha está compuesta por un solo atributo.
- Se denomina dependencia funcional completa aquella df $\{X\} \rightarrow \{Y\} \in F$ si *para todo* subconjunto de X, X' $(X' \subset X)$ tenemos que $\{X'\} \rightarrow \{Y\}$ no se cumple.
- Una df $\{X\} \rightarrow \{Y\}$ es una dependencia funcional parcial si es posible eliminar un atributo $A \in X$ de X y la dependencia sigue determinando a Y, es decir, para algún $A \in X$, $\{X - \{A\}\} \rightarrow \{Y\}$.

Contenidos

Cierre de un conjunto de atributos

- Una forma más sencilla y práctica de calcular F^+ consiste en determinar primero todos y cada uno de los conjuntos de atributos X que aparecen como miembro izquierdo de alguna df en F.
- Después determinar el conjunto de todos los atributos que dependen de X
- Para cada conjunto de atributos X, determinamos el conjunto X_F^+ de atributos determinados funcionalmente por X. X_F^+ se denomina cierre de X bajo F

Algoritmo cálculo cierre de un conjunto de atributos

cierre_atributos(X:atributos, F:dfs)
$X_F^+ := X$;
do{
  viejoX_F^+:=X_F^+;
  for each dependencia funcional $\{Y\} \rightarrow \{Z\}$ en F **do**
    if $X_F^+ \supseteq Y$ **then** X_F^+:=$X_F^+ \cup Z$;
until (X_F^+=viejoX_F^+);

Algoritmo cálculo cierre de un conjunto de atributos

- Con $F = \{\{CodEmp\} \rightarrow \{Nombre, FechaNac, NombreCat\}, \{NombreCat\} \rightarrow \{Sueldo\}\}$ podemos calcular
  - $\{CodEmp\}_F^+ = \{CodEmp, Nombre, FechaNac, NombreCat, Sueldo\}$.
  - $\{NombreCat\}_F^+ = \{NombreCat, Sueldo\}$.
- El resto de cierres de conjuntos de atributos de Empleados serán el propio conjunto (es decir, representarían dfs triviales), por ejemplo.
  - $\{Sueldo\}_F^+ = \{Sueldo\}$.
  - $\{FechaNac, Nombre\}_F^+ = \{FechaNac, Nombre\}$.

Algoritmo cálculo cierre de un conjunto de atributos

Así el cierre de un conjunto de atributos tiene varios usos:

- Comprobar si X es una superclave de R. Se calcula X_F^+ y se comprueba si X_F^+ contiene a todos los atributos de R.
- Se puede comprobar si se cumple una dependencia funcional $\{X\} \rightarrow \{Y\}$ (es decir si está en F^+), comprobando si $Y \subseteq X_F^+$.
- Ofrece una manera alternativa de calcular F^+. Para cada subconjunto dependencias funcionales de F que tienen como lado izquierdo el conjunto de atributos X, se calcula el X_F^+ y, para cada $Y \subseteq X_F^+$, se añade a F^+ la df $\{X\} \rightarrow \{Y\}$.

Contenidos

- Introducción
- Dependencias funcionales
  - Reglas de inferencia para las dependencias funcionales.
  - Definiciones
  - Cierre de un conjunto de atributos
  - Equivalencia de un conjunto de dependencias funcionales
- Descomposición
- Normalización
  - 1ª Forma Normal
  - 2ª Forma Normal
  - 3ª Forma Normal
  - Forma Normal Boyce Codd
- Desnormalización
- Algoritmos de diseño de BDs
  - Forma Normal Boyce Codd

Equivalencia de conjuntos de dependencias funcionales

Dos conjuntos de dfs E y F son *equivalentes* si $E^+ = F^+$

Contenidos

Descomposición

Empleados				
CodEmp	Nombre	FechNac	NombreCat	Sueldo
0001	Pepe	13-10-1978	Vendedor	1120
0002	Ana	21-08-1973	Analista	1800
0003	Pedro	12-05-1981	Vendedor	1120
0004	Raquel	23-07-1975	Analista	1800
0005	Raquel	23-05-1975	Gerente	2200

Aquí habría problemas de INSERCIÓN, ACTUALIZACIÓN y BORRADO $\Longrightarrow$ tenemos que dividir o descomponer.

Descomposición

Pero como también vimos anteriormente, esta descomposición no se puede hacer a la ligera.

Empleados				
CodEmp	Nombre	FechNac	NombreCat	Sueldo
0001	Pepe	13-10-1978	Vendedor	1120
0002	Ana	21-08-1973	Analista	1800
0003	Pedro	12-05-1981	Vendedor	1120
0004	Raquel	23-07-1975	Analista	1800
0005	Raquel	23-05-1975	Gerente	2200

`Empleados1(CodEmp, NombreCat)`
`Categoría1(Nombre, NombreCat, FechaNac, Sueldo)`

Si realizamos las proyecciones oportunas sobre la relación Empleados obtendríamos

Empleados1	
CodEmp	NombreCat
0001	Vendedor
0002	Analista
0003	Vendedor
0004	Analista
0005	Gerente

Categoría1			
Nombre	NombreCat	FechaNac	Sueldo
Pepe	Vendedor	13-10-1978	1120
Ana	Analista	21-08-1973	1800
Pedro	Vendedor	12-05-1981	1120
Raquel	Analista	23-07-1975	1800
Raquel	Gerente	23-05-1975	2200

Descomposición

Si realizamos el join natural entre `Empleado1` y `Categoría1`:

Empleados				
CodEmp	Nombre	FechNac	NombreCat	Sueldo
0001	Pepe	13-10-1978	Vendedor	1120
0002	Ana	21-08-1973	Analista	1800
0003	Pedro	12-05-1981	Vendedor	1120
0004	Raquel	23-07-1975	Analista	1800
0005	Raquel	23-05-1975	Gerente	2200
		.....		
0001	Pedro	12-05-1981	Vendedor	1120
0002	Raquel	23-07-1975	Analista	1800
0003	Pepe	13-10-1978	Vendedor	1120
0004	Ana	21-08-1973	Analista	1800

No obtenemos la información original correcta. Las tuplas adicionales que no estaban en Empleados se denominan tuplas espurias

Descomposición

- Sea R un esquema de relación. Un conjunto de esquemas de relación $\{R_1, R_2, \ldots, R_n\}$ es una descomposición de R si:

$$R = R_1 \cup R_2 \cup \ldots \cup R_n$$

- Es decir, $\{R_1, R_2, \ldots, R_n\}$ es una descomposición de R si, para $i = 1, 2, \ldots, n$, cada R_i es un subconjunto de R y cada atributo de R aparece al menos en un R_i.
- Sea r una relación de esquema R y $r_i = \pi_{R_i}(r)$ para $i = 1, 2, \ldots, n$. Siempre se cumple que:

$$r \subseteq r_1 \bowtie r_2 \bowtie \ldots \bowtie r_n$$

Descomposición

- En tal caso, no se puede reconstruir la relación original, y por tanto se dice que es una descomposición con pérdida
- Una descomposición que no es una descomposición con pérdida es una descomposición sin pérdida.

Descomposición sin pérdida

- Una descomposición $\{R_1, R_2, \ldots, R_n\}$ de R es una descomposición sin pérdida si, para todas las extensiones r de esquema R:

$$r = \pi_{R_1}(r) \bowtie \pi_{R_2}(r) \bowtie \ldots \bowtie \pi_{R_n}(r)$$

Propiedad indispensable de toda descomposición

- Es indispensable que la descomposición sea sin pérdida.
- Sea R un esquema de relación, y sea F un conjunto de dfs de R.

  Sea R_1 y R_2 una descomposición de R. Esta descomposición es una descomposición sin pérdida si al menos una de las siguientes dependencias se halla en F^+:

  - $\{R_1 \cap R_2\} \rightarrow \{R_1\}$
  - $\{R_1 \cap R_2\} \rightarrow \{R_2\}$

Descomposición

Empleados			
CodEmp	Nombre	FechNac	NombreCat
0001	Pepe	13-10-1978	Vendedor
0002	Ana	21-08-1973	Analista
0003	Pedro	12-05-1981	Vendedor
0004	Raquel	23-07-1975	Analista
0005	Raquel	23-05-1975	Gerente

Categorías	
NombreCat	Sueldo
Vendedor	1120
Analista	1800
Gerente	2200

Descomposición del esquema Empleados sin pérdida.

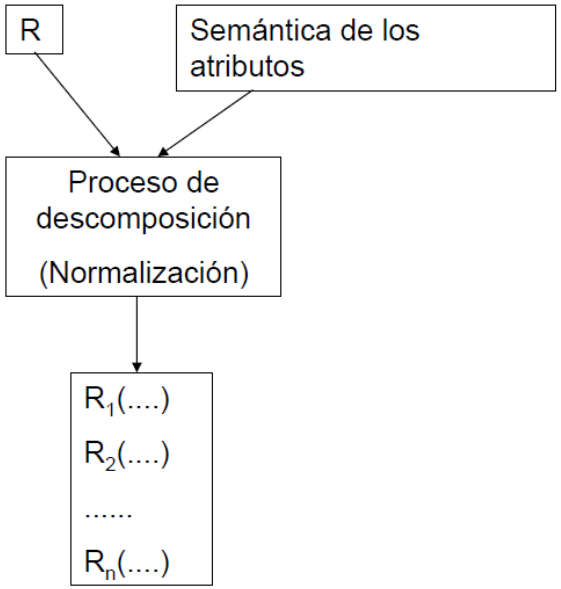
Propiedades deseables en las descomposiciones

- La propiedad de conservación de las dependencias asegura que todas las dependencias funcionales en F están presentes en algunas de las dependencias funcionales que cumplen las relaciones creadas después de la descomposición.
- Sea $\{R_1, R_2, \ldots, R_n\}$ una descomposición de R. La restricción de F a R_i es el conjunto F_i de todas las dependencias funcionales de F^+ que *sólo* incluyen atributos de R_i.
- Si $F' = \{F_1 \cup F_2 \cup \ldots \cup F_n\}$ es equivalente a F, entonces la descomposición conserva las dependencias.

Contenidos

Normalización

Codd propuso un método para obtener buenos diseños de bases de datos basándose en un proceso que somete a los esquema de relación a una serie de pruebas para "certificar" si pertenece o no a una cierta forma normal



Normalización

- La *normalización de los datos* puede considerarse como un proceso de análisis de los esquemas de relación basado en sus dfs para alcanzar las propiedades deseables de
  - (1) minimizar la redundancia y
  - (2) minimizar la anomalías de inserción, eliminación y actualización.
- Los esquemas de relación insatisfactorios que no cumplan determinadas condiciones, se descomponen en esquemas de relación más pequeños que satisfagan dichas pruebas.

Normalización

- La forma normal es la que está una relación es la forma normal más alta que cumple. Así podemos normalizar las relaciones hasta llegar un grado de normalización (forma normal) determinado.
- Pero no basta con que las relaciones de nuestra base de datos cumplan determinada forma normal, es necesario que la descomposición sea una descomposición sin pérdida.
- Un atributo de un esquema de relación R se denomina atributo primo de R si es miembro de alguna clave candidata de R.

Contenidos

Normalización: 1FN

- La primera forma normal (1FN) se considera ahora parte de la definición formal de relación en el modelo relacional básico (o plano).
- Establece que el dominio de un atributo debe incluir valores atómicos (simples o indivisibles) y que el valor de cualquier atributo en una tupla debe ser un valor individual proveniente del dominio de ese atributo.

Normalización: 1FN

DEPT1

DEPTNO	DNAME	LOCALIZACIONES
10	ACCOUNTING	NEW YORK, A CORUÑA, NOVOSIBIRSK
20	RESEARCH	DALLAS, LAS CRUCES
30	SALES	CHICAGO, EL PASO
40	OPERATIONS	BOSTON, CARDIFF

Normalización: 1FN

DEPT2

DEPTNO	DNAME	LOCALIZACIÓN
10	ACCOUNTING	NEW YORK
10	ACCOUNTING	A CORUÑA
10	ACCOUNTING	NOVOSIBIRSK
20	RESEARCH	DALLAS
20	RESEARCH	LAS CRUCES
30	SALES	CHICAGO
30	SALES	EL PASO
40	OPERATIONS	BOSTON
40	OPERATIONS	CARDIFF

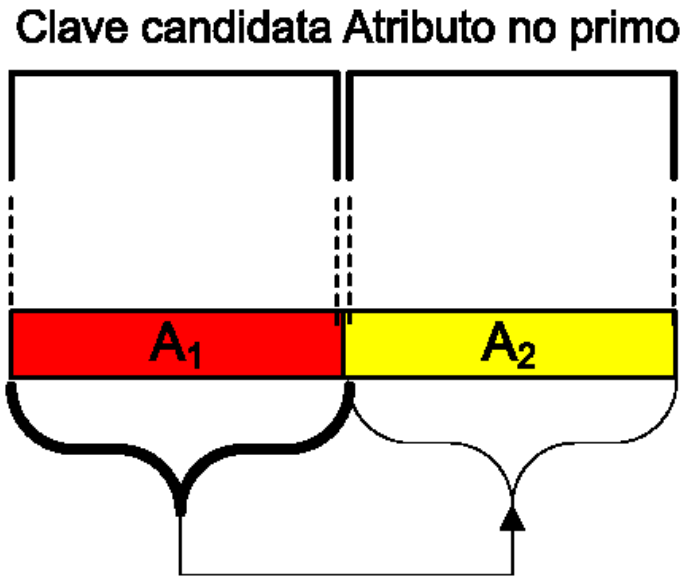
Contenidos

Normalización: 2FN

- La **segunda forma normal (2FN)** se basa en el concepto de dependencia funcional completa.
- Un esquema R está en 2FN si todo atributo no primo A en R depende funcionalmente de de todas las claves candidatas con dfs completas.

Normalización: 2FN
Ejemplos de situaciones CORRECTAS

$\{A_1\} \rightarrow \{A_2\}$



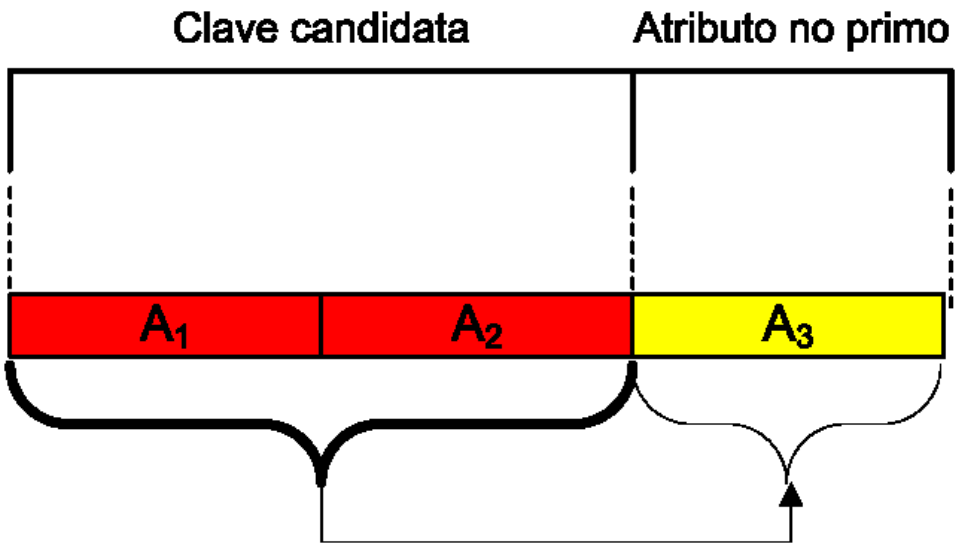
Normalización: 2FN
Ejemplos de situaciones CORRECTAS

$\{A_1, A_2\} \rightarrow \{A_3\}$, y que NO se de:

$\{A_1\} \not\rightarrow \{A_3\}$

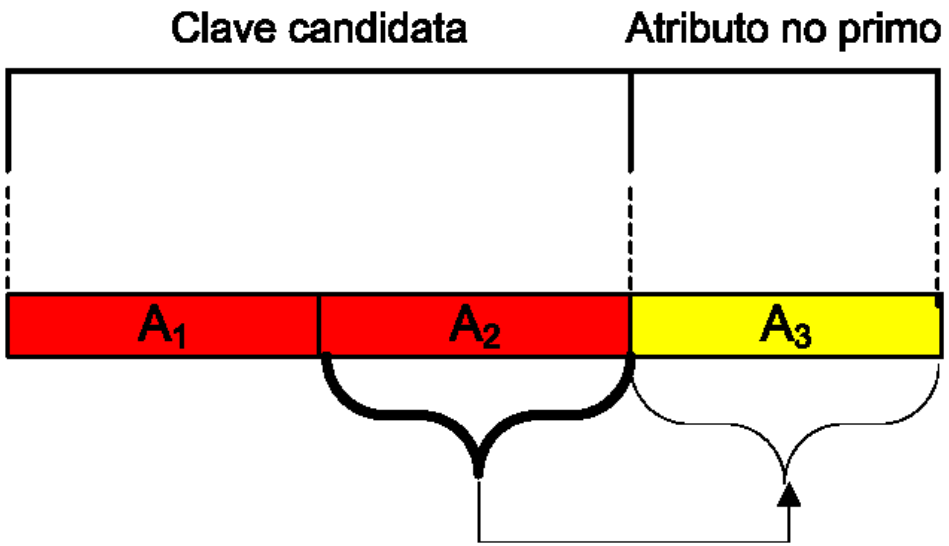
o

$\{A_2\} \not\rightarrow \{A_3\}$



Normalización: 2FN
Ejemplo de situación INCORRECTA

$\{A_1, A_2\} \rightarrow \{A_3\}$, y que se de:
$\{A_1\} \rightarrow \{A_3\}$
o
$\{A_2\} \rightarrow \{A_3\}$



Normalización: 2FN

DEPT2

DEPTNO	DNAME	LOCALIZACIÓN
10	ACCOUNTING	NEW YORK
10	ACCOUNTING	A CORUÑA
10	ACCOUNTING	NOVOSIBIRSK
20	RESEARCH	DALLAS
20	RESEARCH	LAS CRUCES

...

La única df que se pueden extraer de esa relación es $\{DEPTNO\} \rightarrow \{DNAME\}$.
Por otro lado, la única clave candidata es {DEPTNO, LOCALIZACIÓN}.
Como vemos el único atributo no primo de la relación (DNAME) no tiene una df completa con respecto a la clave candidata

Tiene una dependencia parcial con la clave candidata, ya que depende de parte (en concreto de DEPTNO) de la clave candidata

Normalización: 2FN

DEPT3

DEPTNO	DNAME
10	ACCOUNTING
20	RESEARCH
30	SALES
40	OPERATIONS

LOCALIZACIONES

DEPTNO	LOCALIZACIÓN
10	NEW YORK
10	A CORUÑA
10	NOVOSIBIRSK
20	DALLAS
20	LAS CRUCES
...	

En DEPT3 se cumple una única df $\{DEPTNO\} \rightarrow \{DNAME\}$
Pero en este caso, dado que la (única) clave candidata es DEPTNO, DNAME (el atributo no primo) tiene una df completa con la clave candidata.

En la otra relación, LOCALIZACIONES, no hay ninguna dependencia funcional ni atributos no primos (la clave candidata es {DEPTNO, LOCALIZACIÓN}), por lo tanto está en 2FN.

Contenidos

Normalización: 3FN

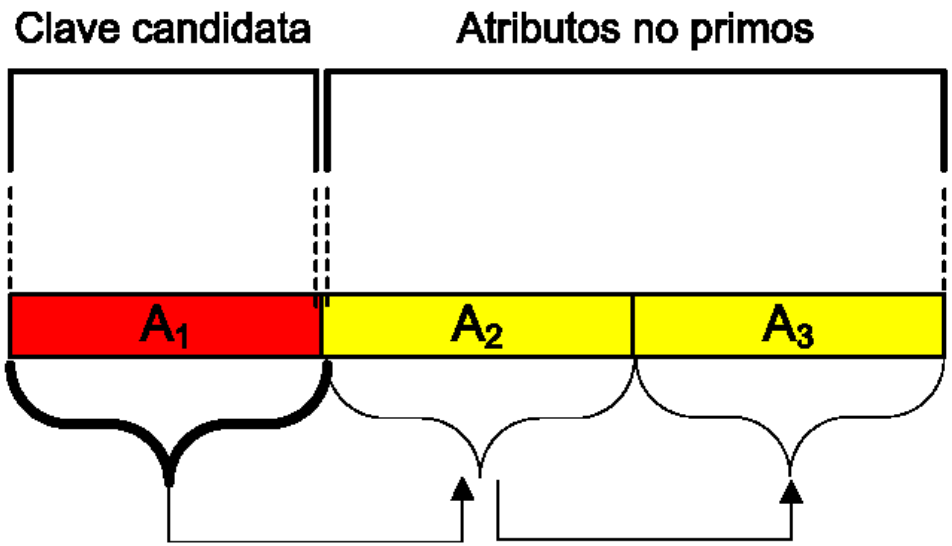
- La **Tercera forma normal (3FN)** se basa en el concepto de dependencia transitiva. Una df $\{X\} \rightarrow \{Y\}$ en un esquema de relación R es transitiva si existe un conjunto de atributos Z que no sea un subconjunto de cualquier clave candidata de R y se cumple tanto $\{X\} \rightarrow \{Z\}$ como $\{Z\} \rightarrow \{Y\}$.
- Un esquema de relación R está en 3FN si está en 2FN y ningún atributo no primo de R depende transitivamente de una clave candidata.

Normalización: 3FN

- Otra forma de comprobar que un esquema de relación está en 3FN es comprobar que para todas las dfs $\{X\} \rightarrow \{Y\}$ que se cumplen en R, se cumple al menos una de las siguientes condiciones:
  - $\{X\} \rightarrow \{Y\}$ es una df trivial.
  - X es una superclave de R.
  - Cada atributo A de $Y - X$ está contenido en alguna clave candidata.

Normalización: 3FN
Ejemplo de situación INCORRECTA

$\{A_1\} \rightarrow \{A_2\}$ y $\{A_2\} \rightarrow \{A_3\}$:



Normalización: 3FN

- Analicemos las dfs de la relación Empleados:
  $F = \{\{CodEmp\} \rightarrow \{Nombre, FechaNac, NombreCat\}, \{NombreCat\} \rightarrow \{Sueldo\}\}$.
  Consideremos el atributo no primo Sueldo, existen estas dos fds en F^+:
  - $\{CodEmp\} \rightarrow \{NombreCat\}$
  - $\{NombreCat\} \rightarrow \{Sueldo\}$
- Por lo tanto Sueldo tiene una dependencia transitiva con la clave candidata (a través de NombreCat).

Contenidos

Normalización: Forma normal Boyce-Codd (FNBC)

- La **forma normal Boyce-Codd** fue propuesta porque había casos en que relaciones en 3FN todavía contenían redundancia, eso sí, en casos muy puntuales y raros.
- Una relación en FNBC está en 3FN, pero (evidentemente) una relación en 3FN no tiene porque estar en FNBC.

Normalización: Forma normal Boyce-Codd (FNBC)

- Para ver la necesidad de una forma normal superior a la 3FN consideremos el siguiente esquema de relación:

  `Empleado_Asignado(ID_Cli, ID_Suc, ID_Emp)`

- Un empleado trabaja en una sucursal. Un cliente puede tener cuentas en varias sucursales, pero en cada sucursal tiene asignado un "empleado gestor".
- Las dependencias funcionales en este esquema son:
  $F = \{\{ID\_Cli, ID\_suc\} \rightarrow \{ID\_Emp\},$
  $\{ID\_Emp\} \rightarrow \{ID\_Suc\}\}$.
- Todos los atributos son primos, dado que las claves candidatas son: {ID_Cli, ID_Suc} y {ID_Cli, ID_Emp}

Normalización: Forma normal Boyce-Codd (FNBC)

- Para ver si está en 3FN:
  - $\{ID\_Cli, ID\_suc\} \rightarrow \{ID\_Emp\}$: el antecedente es superclave (de hecho, clave candidata).
  - $\{ID\_Emp\} \rightarrow \{ID\_Suc\}$: el consecuente es un atributo primo.
- Por lo tanto el esquema Empleado_Asignado está en 3FN.

Normalización: Forma normal Boyce-Codd (FNBC)

- Sin embargo, observemos una posible extensión de este esquema.

Empleado_Asignado		
ID_Cli	ID_Emp	ID_Suc
C1	E100	PrincipalAC
C1	E200	JuanFlorezAC
C8	E100	PrincipalAC
C8	E300	RealAC
C10	E100	PrincipalAC
C12	E400	PrincipalAC

Existe redundancia, siempre que aparece el empleado E100, la sucursal es "PrincipalAC".

Normalización: Forma normal Boyce-Codd (FNBC)

- La definición formal de la FNBC difiere un poco de la definición de 3FN.
- Un esquema de relación está en FNBC si, siempre que una df no trivial $\{X\} \rightarrow \{Y\}$ es válida en R, entonces X es una superclave de R.
- La única diferencia entre FNBC y 3FN es en la condición (ausente) de que los atributos $Y - X$ podían ser atributos primos si X no era superclave y, la relación seguía estando en 3FN.

Normalización: Forma normal Boyce-Codd (FNBC)

- En nuestro ejemplo inspeccionamos una a una otra vez las dfs para ver si la relación está en FNBC:
  - $\{ID\_Cli, ID\_suc\} \rightarrow \{ID\_Emp\}$: el antecedente es superclave.
  - $\{ID\_Emp\} \rightarrow \{ID\_Suc\}$: el antecedente NO es una superclave, por lo tanto la relación no está en FNBC.
- Una vez más para conseguir una base de datos libre de anomalías hay que descomponer.

Normalización: Forma normal Boyce-Codd (FNBC)

Empleado_Suc(ID_Emp,Id_Suc)
Clave candidata:ID_Emp
F=$\{ID\_Emp\} \rightarrow \{ID\_Suc\}$

Empleado_Asignado1(ID_Cli,ID_Emp)
Clave candidata:{ID_Cli, ID_Emp}
F=$\emptyset$

- Las dos relaciones están es FNBC.
- En el esquema Empleado_Suc la única df que hay ($\{ID\_Emp\} \rightarrow \{ID\_Suc\}$) tiene como antecedente la clave candidata,
- mientras que en Empleado_Asignado1 no hay dfs (salvo las triviales), y por tanto está en FNBC.

Normalización: Forma normal Boyce-Codd (FNBC)

- Esta es una descomposición sin pérdida:

$Empleado\_Suc \cap Empleado\_Asignado1 = ID\_Emp$
$\{ID\_Emp\} \rightarrow \{ID\_Suc\}$

- Sin embargo, obsérvese que no se conservan las dependencias
- La df $\{ID\_Cli, ID\_suc\} \rightarrow \{ID\_Emp\}$ se ha perdido
- En general siempre es posible encontrar una descomposición de una relación en 3FN en la que se conserven las dependencias funcionales, mientras que tal vez no sea posible encontrar una descomposición en FNBC en la que se conserven las dependencias.

Contenidos

Desnormalización

- La descomposición de relaciones en relaciones más pequeñas tiene un coste en el rendimiento debido a la necesidad de realizar joins para "reconstruir" la información
- Las relaciones pueden dejarse en formas normales inferiores por razones de rendimiento.
- El proceso de almacenar relaciones en formas normales inferiores (sin descomponer lo suficiente) se conoce como desnormalización

Contenidos

Algoritmos de diseño de bases de datos

- Diseño descendente, una técnica que se emplea actualmente en el diseño de aplicaciones de bases de datos comerciales.
- Diseñar un esquema conceptual (modelo ER) y luego transformar el esquema conceptual en un conjunto de relaciones empleando el procedimiento de transformación E/R a MR.
- A continuación, se analiza cada una de las relaciones usando las dependencias funcionales y se les asignan claves primarias y se descomponen aquellas relaciones que no estén en la forma normal deseada.

Algoritmos de diseño de bases de datos

- Diseño ascendente, una técnica más purista que implica contemplar el diseño de esquemas de bases de datos relacionales estrictamente en términos de dependencias funcionales y dependencias de otros tipos.
- En un diseño ascendente puro, una vez que el diseñador de bases de datos especifica las dependencias, se aplica un algoritmo de normalización con el fin de sintetizar los esquemas de relación.
- Suelen comenzar sintetizando un esquema de relación gigante, llamado *relación universal*, que incluye todos los atributos de la base de datos.
- A continuación, se realiza repetidamente una descomposición hasta que ya no sea factible o deseable seguir.

Contenidos

Algoritmos de diseño de bases de datos: FNBC

decomposicion_FNBC (R,F)

ENTRADA: Una relación universal R y un conjunto de dependencias funcionales F sobre los atributos de R.

1. Asignar $D := \{R\}$.
2. Mientras haya un esquema de relación Q en D que no esté en FNBC:
   {
   escoger un esquema de relación Q en D que no esté en FNBC
   buscar una df $\{X\} \rightarrow \{Y\}$ tal que X no sea clave candidata de Q
   reemplazar Q en D por dos esquemas $(Q - Y)$ y $(X \cup Y)$
   }