Bases de Datos 1

- Licenciatura e Ingeniería en Informática
- 2do. año

Diseño de Bases de Datos Relacionales

- Objetivos
  - Generar un conjunto de esquemas de relaciones con un mínimo de redundancia.
  - Facilitar la recuperación de información

- ¿Cómo lo logramos?
  - Llevando el conjunto de esquemas de relaciones a una forma normal adecuada.

- Dependencia de Datos
  - Dependencias funcionales (simples o complejas - mvds)
    - Me servirán para reconocer redundancias y actuar en consecuencia.
  - Restricciones sobre las instancias de relaciones posibles.
  - Definir como dependerán los atributos entre sí.

Problemas de Diseño

- Redundancia
  - Guardar información repetida

PROVEEDOR	DOMICILIO	ARTICULO	PRECIO
Acme Corp.	*Cuareim 1522*	*Monitor VGA*	*450*
Acme Corp.	*Cuareim 1522*	*Monitor Herc*	*120*
Computronix	*Yaguarón 9786*	*Monitor VGA*	*470*

- Inconsistencia potencial (anomalías de actualización)
  - Si tenemos redundancia, en el caso de una actualización habrá que actualizar en todos los lugares donde aparece el dato a modificar.

update por otro domicilio

PROVEEDOR	DOMICILIO	ARTICULO	PRECIO
Acme Corp.	~~*Cuareim 1522*~~	*Monitor VGA*	*450*
Acme Corp.	*Cuareim 1522*	*Monitor Herc*	*120*
Computronix	*Yaguarón 9786*	*Monitor VGA*	*470*

Problemas de Diseño

- Anomalías de inserción
  - La no independencia de ciertos datos, en caso de inserción, nos obliga a colocar valores nulos.

PROVEEDOR	DOMICILIO	ARTICULO	PRECIO
Acme Corp.	*Cuareim 1522*	*Monitor VGA*	*450*
Acme Corp.	*Cuareim 1522*	*Monitor Herc*	*120*
Computronix	*Yaguarón 9786*	*Monitor VGA*	*470*
PC_UDE	**21 de set. 2741**	**?**	**?**

- Anomalías de borrado
  - Si borramos datos de un esquema que posee información particular que debería independiente, perderemos datos importantes.

PROVEEDOR	DOMICILIO	ARTICULO	PRECIO
Acme Corp.	*Cuareim 1522*	*Monitor VGA*	*450*
Acme Corp.	*Cuareim 1522*	*Monitor Herc*	*120*
~~*Computronix*~~	~~*Yaguarón 9786*~~	~~*Monitor VGA*~~	~~*470*~~

los datos de Computronix se pierden

Dependencias Funcionales

- Definición

Sea RS = (ATR,DMN,dom,M,SC)
X, Y conjuntos de atributos tomados de ATR

X->Y sobre PRS
Para toda t1, t2 en prs / t1[X]=t2[X] -> t1[Y]=t2[Y]

X->Y
X determina Y o bien Y es determinado por X

Dependencias Funcionales

- Implicación lógica de dependencias

Sea RS = (ATR,DMN,dom,M,SC)
SC contiene dependencias funcionales y X->Y dependencia funcional

SC |= {X->Y}

SC implica lógicamente a X->Y si cada instancia r de RS que satisface las dependencias funcionales de SC también satisface X->Y

Dependencias Funcionales

- Implicación lógica de restricciones

Sea PRS=(ATR,DMN,dom) y SC su conjunto de todas las restricciones (o dependencias) que pueden definirse sobre PRS.

SC1 y SC2 subconjuntos de SC

SC1|=SC2 si cada instancia posible de PRS que satisface SC1 también satisface SC2

SC1 es equivalente a SC2, SC1<=>SC2

si SC1|=SC2 y SC2|=SC1

Dependencias Funcionales

- Conjunto clausura de restricciones

  SC* el conjunto de todas las dependencias funcionales implicadas por SC

  SC* = {X->Y | SC |= X->Y} Conjunto clausura de SC

- Teorema

  Sea RS = (ATR,DMN,dom,M,SC) y X->Y una df de SC
  Sea r una instancia de la relación RS

$$r = \pi(r,XY) * \pi(r,X(ATR - Y))$$

  - Puedo descomponer la relación original en dos relaciones que podrán ser unidas por join.

Dependencias Funcionales

- Teorema

Sea RS = (ATR,DMN,dom,M,SC) y SC un conjunto de dfs.
Sean X e Y conjuntos de atributos de ATR.

Para toda instancia r de la relación RS tenemos

$$r = \pi(r,XY) * \pi(r,X(ATR - Y)).$$

Entonces

X->Y Є SC* o X->(ATR-Y) Є SC*

Dependencias Funcionales

- Axiomas de Armstrong
  - Son axiomas que nos permiten deducir nuevas dependencias funcionales.
  - Con las tres primeras reglas podemos hacer todas las inferencias válidas.

Dependencias triviales:

- $X \rightarrow Y$ siempre que $Y \subset X$
- $X \rightarrow \varnothing$

Dependencias Funcionales

- Axiomas de Armstrong
  - *Regla 1* - **REFLEXIVA**

    $\{\} \models X \rightarrow Y$, si $Y \subset X$

  - *Regla 2* - **AUMENTACIÓN**

    $\{ X \rightarrow Y \} \models XZ \rightarrow YZ$

  - *Regla 3* – **TRANSITIVA**

    $\{ X \rightarrow Y , Y \rightarrow Z \} \models X \rightarrow Z$

  - *Regla 4* – **UNION**

    $\{ X \rightarrow Y , X \rightarrow Z \} \models X \rightarrow YZ$

  - *Regla 5* - **INTERSECCION**

    $\{ X \rightarrow Y , X \rightarrow Z \} \models X \rightarrow Y \cap Z$

Dependencias Funcionales

- Axiomas de Armstrong

  - *Regla 6* - **REDUCCION**

    $$\{X \rightarrow Y\} \models X \rightarrow Y - X$$

  - *Regla 7* - **AUMENTACIÓN GENERALIZADA**

    $$\{X \rightarrow Y\} \models W \rightarrow V \text{ , si } X \subset W \text{ y } V \subset XY$$

  - *Regla 8* - **FRAGMENTACIÓN**

    $$\{X \rightarrow Y\} \models X \rightarrow A \text{ , si } A \subset Y$$

  - *Regla 9* - **TRANSITIVA GENERALIZADA**

    $$\{X \rightarrow Y, U \rightarrow V\} \models W \rightarrow Z \text{ , si } U \subset XY,\ X \subset W \text{ y } Z \subset VW$$

Dependencias Funcionales

- Determinantes (claves)

  Sea RS un esquema de relación
  ATR = {A1,A2,…AN}
  SC un conjunto de restricciones con dfs

  El conjunto X es determinante o clave si:

  - X -> ATR está en SC*
  - Para ningún Y incluido en X ocurre que Y->ATR está en SC* (condición de minimalidad).

- Clave candidata
  - Conjunto mínimo de atributos que determinan funcionalmente a todos los atributos del esquema.
  - Designaremos a una de ellas como *clave primaria*.

Dependencias Funcionales

- Superclave
  - Cualquier superconjunto propio de una clave
  - Toda clave es superclave

- Lema
  - Si se deduce X->Y a partir de SC usando axiomas de Armstrong, entonces X->Y se cumple para cualquier instancia posible de RS en las cuales las dfs de SC sean verdaderas.

Clausura de Atributos X+

Sea SC un conjunto de dfs sobre el conjunto de atributos ATR y X un conjunto de atributos de ATR

X+ será el conjunto de atributos A tales que SC|=X->A utilizando las reglas de inferencia.

X->Y pertenece a SC* si y sólo si Y está incluido en X+

Algoritmo de Cálculo de X+:

ENTRADA : Conjunto de atributos X y conjunto de dependencias SC.
SALIDA : X+ (Clausura de X respecto a SC)

```
Resultado := X
Mientras (Resultado cambió)
        Para cada dependencia Y → Z en SC
                Si Y ⊂ Resultado
                        Resultado := Resultado ∪ Z
                FinSi
        FinPara
FinMientras
```

Clausura de Atributos X+

- Un conjunto de atributos X es superclave si su clausura X+ contiene a todos los atributos del esquema.

- Una clave de será el conjunto de atributos que sus miembros no aparecen a la derecha de las dfs.

- Ej: R(A,B,C,D,E,F)

  SC = {A->D,B->(E,F),(A,B)->C}

  A+ = {A,D}
  B+ = {B,E,F}
  (A,B)+ = {A,B,D,E,F,C}

Obs. (A,B)+ no es la unión de las clausuras de los elementos del conjunto.

Cubrimientos

- Equivalencia de conjuntos de dfs

  Sean F y G conjuntos de dependencias funcionales.

  F es equivalente a G si F* = G* (F cubre a G - y G cubre a F)

```
Algoritmo
ENTRADA:  F y G (conjuntos de dfs)
SALIDA :   VERDADERO si cada fd de F está en G*
PARA CADA Y-->Z en F
   Calcular Y+ según G
   SI  Z no es subconjunto de Y+
      RETORNAR (FALSO)
   FINSI
FINPARA
RETORNAR (VERDADERO)
```

  – Con el algoritmo probaré si G cubre a F. Debo repetirlo para saber si F cubre a G.

  – <u>Lema</u>: Existirá un conjunto G que cubra a F que sus dfs contienen un único atributo a la derecha.

Cubrimiento Minimal

- Un conjunto F de dependencias funcionales es minimal si:
  - El lado derecho de cada df es un único atributo
    - Convertimos todas las dfs a dependencias con un único atributo a la derecha
  - Para ninguna X->Y en F y siendo Z < X
    se cumple que F - {X->Y} U {Z->Y} es equivalente a F
    - Trato de eliminar atributos redundantes a la izquierda.

  - Para ninguna X->Y en F se cumple que F - {X->Y} es equivalente a F

    Verificar para cada df si puedo encontrar un conjunto equivalente sin ella, entonces la elimino.

- Teorema
  - Todo conjunto SC de dfs es equivalente a un conjunto SC' minimal.

Cubrimiento Minimal

- SC = {AB->C, B->D,D->GC,CG->H}

Paso 1: Pasar las dependencias con varios atributos a la derecha a un sólo atributo.

SCpaso1 = {AB->C,B->D,D->G,D->C,CG->H}

Paso2: Atributos redundantes (Verifico AB->C y CG->H)

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

C+ = {C} G+ = {G} no redundantes (no llego sin el otro a H)

A sólo no determina a C, por lo que B no es redundante

B sólo determina a C, no necesita de A.

SCpaso2 = {B->C,B->D,D->G,D->C,CG->H}

Paso3: Dependencias redundantes

Debo verificar para cada dependencia si no es redundante.

B->C es redundante? SCpaso3'={B->D,D->G,D->C,CG->H}

B+(sobre SCpaso3') ={B,D,G,C,H} llego a C, por lo tanto ES redundante. Sigo con SCpaso3'.

B->D es redundante? SCpaso3''={D->G,D->C,CG->H}

B+(sobre SCpaso3'')={B} no es redundante. Sigo con SCpaso3'.

D->G es redundante? Scpaso3'''={B->D,D->C,CG->H}

D+(sobre SCpaso3''')={D,C} no es redundante. Sigo con Scpaso3'

D->C es redundante? SCpaso3''''={B->D,D->G,CG->H}

D+(sobre SCpaso3'''')={D,G} no es redundante. Sigo con Scpaso3'

CG->H es redundante? SCpaso3'''''={B->D,D->G,D->C}

CG+(sobre SCpaso3''''') = {C,G} no es redundante. Sigo con Scpaso3'

SCpaso3 = {B->D,D->G,D->C,CG->H}

Descomposición de esquemas

Sea un esquema de relación RS = (ATR,DMN,dom,M,SC)
Sea un esquema de base de datos DS=(PDS,DM,SDC)
con PDS = {RSi=(ATRi,DMNi,domi,Mi,SCi) | i=1..k}

Descomposición con Join Sin Pérdida

- ATRi incluido o igual a ATR, i=1..k
- ATR1 U ATR2 U … U ATRk = ATR
- DM expresa que DS es una representación de la relación RS con significado M
- SDC $= \emptyset$
- Para cada instancia r de RS el join natural de las proyecciones sobre los ATRi es r.

$$\pi(r,ATR1) * \pi(r,ATR2) * \ldots * \pi(r,ATRk) = r$$

Algoritmo JSP
Join Sin Pérdida

- Algoritmo
  - Crear una matriz S con una fila i por cada relación Ri en la descomposición y una columna j por cada atributo Aj en Ri
  - Hacer S(i,j) = bij para todas las entradas de la matriz
  - Para cada fila i que represente el esquema de relación Ri

    Para cada columna j que representa al atributo Aj

    Si Ri incluye a Aj entonces S(i,j)=aj
  - Repetir hasta no modificar más S

    Para cada df X->Y en F

    Igualar los símbolos en los atributos de Y para aquellas filas que coinciden en los atributos de X
  - Si una fila tiene todos los símbolos a, la descomposición es JSP, en caso contrario no lo es.

Ej: Algoritmo JSP
Join Sin Pérdida

D = {R1,R2,R3} SC = {A->B,C->(D,E),(A,C)->F}

R = {A,B,C,D,E,F}

R1 = {A,B}

R2 = {C,D,E}

R3 = {A,C,F}

Paso1 -

	A	B	C	D	E	F
R1	b11	b12	b13	b14	b15	b16
R2	b21	b22	b23	b24	b25	b26
R3	b31	b32	b33	b34	b35	b36

Paso 2 -

	A	B	C	D	E	F
R1	a1	a2	b13	b14	b15	b16
R2	b21	b22	a3	a4	a5	b26
R3	a1	b32	a3	b34	b35	a6

Paso 3 -

A->B

	A	B	C	D	E	F
R1	a1	a2	b13	b14	b15	b16
R2	b21	b22	a3	a4	a5	b26
R3	a1	~~b32~~ a2	a3	b34	b35	a6

Paso 4 -

C->(D,E)

	A	B	C	D	E	F
R1	a1	a2	b13	b14	b15	b16
R2	b21	b22	a3	a4	a5	b26
R3	a1	a2	a3	~~b34~~ a4	~~b35~~ a5	a6

Join Sin Pérdida

- Teorema

  Sea {RS1,RS2} una descomposición del esquema RS = (ATR,DMN,dom,M,SC).

  La descomposición es join sin pérdida con respecto a SC si y sólo si, o bien se cumple

  $R1 \cap R2 \dashrightarrow R1 - R2$

  o bien se cumple

  $R1 \cap R2 \dashrightarrow R2 - R1$

Preservación de dependencias

- Además es deseable que no se pierdan dependencias luego de las descomposiciones.

- Proyección de dfs

  Sea RS=(ATR,DMN,dom,M,SC) donde SC es un conjunto de dfs.

  Sea ATRi incluido o igual a ATR.

  Una proyección del conjunto de dfs SC sobre ATRi es un conjunto Sci que es lógicamente equivalente a {X->Y | X->Y pertenece a SC* y XY está incluido o es igual a ATRi}

$$\pi fd(SC,ATRi)$$

Preservación de dependencias

Sea RS=(ATR,DMN,dom,M,SC)
Una descomposición d = {RS1,RS2,…,RSk}

La descomposición d preserva dependencias de SC si la unión de todas las dependencias en $\pi fd(SC,ATRi)$

con i k $\subseteq$
implica lógicamente a todas las dfs de SC

Preservación de dependencias

Algoritmo [Test de preservación]
ENTRADA: RS = (ATR,DMN,dom,M,SC) y {RS1,...,RSk} (descomposición de RS)
SALIDA : VERDADERO si la descomposición preserva fds
PARA CADA X-->Y en SC
  Z := X
  MIENTRAS Z cambió
  / Agregar a Z los atributos comunes de RSi con la clausura de los atributos comunes de Z y RSi (i=1..k) */*
    Z := Z U ((Z ∩ RSi)+ ∩ RSi) ,i=1..k
  FINMIENTRAS
  SI Y >= Z */* ¿Y no es subconjunto propio de Z? */*
    RETORNAR (FALSO)
  FINSI
FINPARA
RETORNAR (VERDADERO)