Diseño de Bases de Datos

Refinamiento de esquemas y Normalización

El problema de la redundancia

» el punto de partida es el esquema relacional, que incluye las restricciones de integridad

- Redundancia: almacenar información en más de un sitio
  - Almacenamiento redundante
  - Posibilidad de inconsistencia
  - Anomalías
    - actualización
    - inserción
    - borrado

...

Empleado

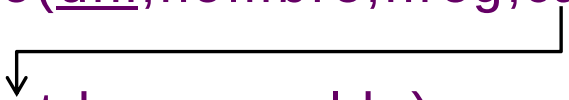
dni	nombre	nreg	cat	horas_sem	base_sueldo
2354	García	48	8	40	10
9625	Aragón	22	8	30	10
5557	Pozo	35	5	35	7
2121	Sainz	35	5	40	7

- la combinación 8 -> 10 y 5 -> 7 es redundante
- actualización de base_sueldo: en todas las tuplas
- inserción de empleado: sólo si se sabe la base de su categoría —dejar *null* ?
- borrado de todas las tuplas de una categoría: se pierde la base_sueldo

Descomposición

» sustituir una relación por otras dos (o más) cada una con un subconjunto de campos y, conjuntamente, incluyendo todos los originales

Empleado(dni,nombre,nreg,cat,horas_sem)



Sueldo(cat,base_sueldo)

- Requisitos descomposición
  - Reunión sin pérdida
  - Conservación de las dependencias
  - Consultas pueden obligar a la reunión de las relaciones descompuestas
    - penalización en el rendimiento
    - ¿y si la eficiencia resultante no fuera aceptable?

Dependencias funcionales

» generaliza el concepto de clave

DF, $X \rightarrow Y$

$\forall$ t1,t2 : R | t1.X=t2.X $\Rightarrow$ t1.Y=t2.Y

DF, AB $\rightarrow$ C

- y sin embargo AB no es clave

A	B	C	D
1	1	1	1
1	1	1	2
1	2	2	1
2	1	3	2

DF, clave $\rightarrow$ todos los atributos —*definición relacional*

- la dependencia no exige que la clave sea mínima

DF, X $\rightarrow$ todos los atributos

- si X no es mínimo, entonces es una *superclave*

Formas Normales

» si el esquema se encuentra en una de estas formas normales, ciertos tipos de problemas serán eliminados / minimizados

- Basadas en DF
  - 1NF, 2NF, 3NF, BCNF
  - 3NF y BCNF son importantes en el diseño de bases de datos
- Basadas en otros tipos de dependencias
  - multivaluadas (DM): 4NF, reunión (DR): 5NF

BCNF

» —*intuitivamente*— las únicas dependencias son aquellas en las que una clave determina algún atributo; no se pueden inferir valores

- Rel R, DFs F, subconj. atributos X, atrib. A
  R está en BCNF si, para cada DF $X \rightarrow A$,
  - $A \in X$, DF *trivial*, o
  - X contiene una clave de R
- Ejemplo: Rel XYA y DF $X \rightarrow A$
  - (DF) la segunda tupla también tiene 'a' (redundancia?)
  - (si BCNF, $A \neq X$) X tiene que ser clave
  - (X clave) y1=y2, y las tuplas son idénticas
  - (relacional) sólo se almacena una vez

X	Y	A
x	y1	a
x	y2	?

BCNF y 3NF

- Problema *técnico* con BCNF
  - no se puede garantizar cualquier esquema en BCNF que conserve las propiedades de (1) reunión sin pérdida, (2) mantenimiento de dependencias
- Cuando eso no es posible, 3NF
  - rebaja las condiciones sólo lo necesario para garantizarlo
  - es posible cierta redundancia
- Similar a BCNF, con una tercera posibilidad
  - A es parte de alguna clave de R
    - la minimalidad de la clave es crucial aquí

¿Qué conseguimos con 3NF?

- Casos posibles de que X → A no cumpla 3NF.
  - X es un subconj. propio de alguna clave
    - dependencia parcial
    - ej. reservas: se anota la tarjeta de pago (dni,matr,fecha,tarjeta), con DF DNI → TARJETA
    - se guardan redundantemente los pares (dni,tarjeta)
  - X no es subconj. propio de ninguna clave
    - dependencia transitiva
    - ej. empleados: con DF CAT → SUELDO, con transitividad DNI → CAT → SUELDO
    - anomalías de inserción, actualización y borrado

Redundancia en 3NF, y 2/1NF

- Permitimos que A sea parte de una clave
  - Ej. reservas: con DF DNI $\rightarrow$ TARJETA, *pero* sabiendo que la tarjeta identifica a los clientes, DF TARJETA $\rightarrow$ DNI
    - tarjeta,matr,fecha también es clave
    - mantenemos la redundancia de pares (dni,tarjeta) para capturar todas las dependencias originales
    - se cumple 3NF
- 2NF: no se permiten depend. parciales
- 1NF: no se permiten grupos repetitivos (condición relacional)

Descomposición de una relación

» en dos (o más) que se *reparten* los atributos, garantizando *reunión sin pérdida* y *conservación de dependencias*

- Algoritmo de descomposición en BCNF
  1. R no en BCNF, $X \subset R$, A atributo, $X \rightarrow A$ que provoca el no cumplimiento
  2. Descomponer en R-A y XA
  3. Si R-A o XA no en BCNF, aplicar recursivamente
- Ej. Contratos con atributos CPYDRNV (idC,idProve,idProy,idDepto,idRepuesto,cantidad,valor) y DFs PD $\rightarrow$ R, Y $\rightarrow$ P, con claves en el lado izquierdo

...

1. guiado por PD $\rightarrow$ R
   - PDR y CPYDNV, ahora aplicar para Y $\rightarrow$ P
   - PDR, YP y CYDNV
2. guiado por Y $\rightarrow$ P
   - YP y CYDRNV
   - la otra dependencia ya no desnormaliza

- Distintas alternativas, cuya elección se hará respecto a la semántica de la aplicación

ER -> Rel -> Rel Normalizado

- ¿Se puede directamente producir un ER libre de problemas de redundancia?
  - ER es un proceso complejo y subjetivo (piénsese en esquemas con más de 100 tablas)
  - Determinadas restricciones / dependencias no se pueden expresar en ER (al menos fácilmente)
  - Técnica formal para tratar un diseño original que da como resultado un diseño mejor