**NORMALIZACIÓN**

* Anomalía: Problema que surge a raíz del diseño de una relación.
  + Redundancia: información que se repite innecesariamente en diferentes tuplas.
  + Anomalías de actualización: se puede actualizar el valor en una tupla, sin actualizar los de otras tuplas.
  + Anomalías de inserción: insertar valores en ciertos atributos de una relación y no en otros me produce valores nulos.
  + Anomalías de borrado: borrar ciertos valores de una tupla, puede llevarme a perder la información de la tupla completa.
* Dependencia funcional: captura propiedades del mundo real. Es una **restricción** de una relación en una base de datos y generaliza la idea de clave de una relación.

Si dos tuplas (t1 y t2) de una relación R, coinciden en todos los atributos A1, A2,…, An, entonces deben también coincidir los atributos B1, B2,…, Bm. Para toda tupla de R.

Esto se escribe: A1, A2, .., An 🡪 B1, B2, .., Bm

Y se lee: A1, A2, .., An “determina funcionalmente a” B1, B2, .., Bm

Dicho de otra manera:

Una df de la forma X 🡪 Y se cumple en R si: para todos los pares de tuplas t1 y t2 de la relación, cuando se cumple que t1[x] = t2[x], entonces se cumple t1[y] = t2[y].

* Dependencia funcional trivial: es una df de la forma: A1, A2, .., An 🡪 B1, B2, .., Bm

Tal que: {B1, B2, .., Bm} ⊆ {A1, A2, .., An}

* Clave de una relación: los atributos {A1, A2, ..,An} son la clave de una relación R si cumplen:
  + {A1, A2, ..,An} determinan funcionalmente a todos los restantes atributos de la relación R.
  + No existe un subconjunto de {A1, A2, .., An} que determine funcionalmente a todos los atributos de R –*esto implica que una clave es un conjunto minimal-*
* Clave candidata de una relación: en caso de existir dos o más conjuntos de atributos {A1, A2, .., An}, {B1, B2, .., Bk}, … {N1, N2, .., Nm} en una relación R tales que:
  + {A1, A2, .., An} determinan funcionalmente a todos los restantes atributos de la relación R.
  + {B1, B2, .., Bk}, … y {N1, N2, .., Nm} también por si mismos determinan al resto de los atributos de R.
  + No existe un subconjunto de {A1, A2, .., An} o {B1, B2, .., Bk}, … o {N1, N2, .., Nm} que determine funcionalmente a todos los atributos de R.
  + Entonces {A1, A2, .., An}, {B1, B2, .., Bk}, … {N1, N2, .., Nm} son CLAVES CANDIDATAS para la relación R.
* Superclave de una relación: “super conjunto” de una clave. Los atributos {A1, A2, .., An} son la superclave de una relación R si cumplen:
  + {A1, A2, .., An} determina funcionalmente a todos los restantes atributos de la relación R.
  + Notar que:
    - Una clave está contenida en una superclave.
    - Una superclave no necesariamente es minimal (como lo es la clave por la segunda condición de su definición)

**Axiomas de Armstrong**

Permiten inferir nuevas df dado un conjunto base que resulto evidente.

Aplicándolos hallo un conjunto completo y seguro donde todas las df halladas son correctas.

Al generar todas las df algunas son triviales.

* Axiomas básicos:
  + Reflexión: X es un conjunto de atributos y Y ⊆ X entonces X 🡪 Y (*sabemos que a 🡪 a, luego se puede decir que a,b 🡪 a.*

Demostración: Si Y ⊆ X y existen dos tuplas diferentes de R tales que t1[x] = t2[x] por definición de df t1[y] = t2[y].

* + Aumento: Si X 🡪 Y; Z es un conjunto de atributos, entonces Z,X 🡪 Z,Y.

Demostración: Asumamos que X 🡪 Y vale pero X,Z 🡪 Y,Z no vale. Si X 🡪 Y entonces cada vez que 1) t1[x] = t2[x] implica 2) t1[y] = t2[y], por otro lado, cada vez que 3) t1[x,z] = t2[x,z] implica 4) t1[y,z] <> t2[y,z].

De 1) y 3) se deduce t1[z] = t2[z]

De 2) y 4) se deduce que t1[y,z] = t2[y,z]

* + Transitividad: si X 🡪 Y; Y 🡪 Z, entonces X 🡪 Z.

Demostración: 1) X 🡪 Y. 2) Y 🡪 Z. t1[x] = t2[x] implica por 1) y t1[y] = t2[y] implica por 2). y t1[z] = t2[z] entonces X 🡪 Z

* Axiomas que se deducen a partir de los básicos:
  + Unión: si X 🡪 Y; X 🡪 Z, entonces X 🡪 Y,Z

Demostración: 1) X 🡪 Y. 2) X 🡪 Y. Si X 🡪 Y, por aumentación vale que X 🡪 XY. Si X 🡪 Z, por aumentación vale que X,Y 🡪 Y,Z. Luego por transitividad, X 🡪 Y,Z

* + Descomposición: si X 🡪 Y,Z, entonces X 🡪 Y , X 🡪 Z

Demostración: X 🡪 Y,Z por reflexividad vale que Y,Z 🡪 Y. Luego, por transitividad X 🡪 Z. Por flexibilidad también vale que Y,Z 🡪 Z. Luego por transitividad, también vale que X 🡪 Z

* + Pseudotransitividad: Si X 🡪 Y; Y,Z 🡪 W entonces X,Z 🡪 W

Demostración: X 🡪 Y, por aumento vale que X,Z 🡪 Y,Z. Por otro lado se sabe que Y,Z 🡪 W. Luego por transitividad, vale que X,Z 🡪 W

* Clausura de un conjunto de atributos (X+):

Sea F un conjunto de df sobre un esquema R y sea X un subconjunto de R.

La clausura de X respecto de F, se denota X+ y es el conjunto de atributos A tal que la dependencia X 🡪 A puede deducirse a partir de F, por los axiomas de Armstrong.

Es decir, X+ son todos los atributos determinados por X en R.

* + Algoritmo para encontrar X+:
    - Result := X
    - While (hay cambios en result) do
    - For (cada dependencia funcional Y->Z en F ) do
    - if (Y ⊆ result) then
    - result := result U Z
* Como generar relaciones que cumplan condiciones de un buen diseño:
  + Descomposición:
    - Es una forma aceptada de eliminar las anomalías de una relación.
    - Consiste en separar los atributos de una relación en dos nuevas relaciones.
    - No se debe perder información, ni df.
  + Perdida de información: si un esquema R, se lo particiona en dos subesquemas R1 y R2 se debe cumplir algunas de las siguientes condiciones.
    - R1 intersección R2 es clave en el esquema R1
    - R1 intersección R2 es clave en el esquema R2
  + Perdida de df:
    - Verificar que cada una de las df que valían en el esquema R, sigan valiendo en alguna de las particiones Ri.
    - Cuando se chequean las df pueden ocurrir dos cosas:
      * Los atributos de la df original quedaron todos incluidos en alguna de las particiones generadas.
      * Los atributos de la df original quedaron distribuidos en más de una partición.
    - Algoritmo para analizar la perdida de df:
      * Res = x

Mientras Res cambia

Para i= 1 to cant\_de\_particiones\_realizadas

Res = Res U ((Res intersección Ri)+ intersección Ri)

* + - * Donde Ri es el conjunto de atributos de la división representada por Ri

X es el determinante de la dependencia funcional que quiero analizar.

((Res intersección Ri)+ intersección Ri), asegura que quedan sólo los atributos que pertenecen a la partición que se está tratando.

**Formas Normales – BCNF**

* Forma normal: propiedad sobre la relación.
  + BCNF (Forma normal de Boyce Codd): provee un mecanismo para asegurar que:
    - * Las anomalías dejan de estar en un particionamiento,
      * que no se pierda información y,
      * en algunos casos, asegura que no se pierdan df.
    - Un esquema esta en BCNF si, siempre que una df de la forma X 🡪 A es valida en R, entonces se cumple que:
      * X es superclave de R, o bien,
      * X 🡪 A es una df trivial.
    - Como llevar un esquema R a BCNF: de manera esquemática y simplificada, una vez halladas las df y las claves candidatas:
      * 1-analizar si en el esquema R existe alguna dependencia funcional que no cumple con la definición de BCNF
        + 1.1) si existe tal dependencia funcional, particionar el esquema en dos nuevos esquemas Ri, Ri+1, contemplando la dependencia funcional en cuestión. Analizar las 2 particiones generadas

1.1.1) Se pierde información?

1.1.1.1: NO, entonces sigo a 1.1.2

1.1.1.2: SI. La partición es errónea. Reanalizar

1.1.2) Se pierden Dependencias funcionales?

1.1.2.1 NO, entonces sigo a 1.1.3

1.1.2.2 Si. Entonces no es posible llevar a BCNF. Cambia la forma normal analizada.

1.1.3) Determinar en qué forma normal esta Ri, Ri+1, si no están en BCNF, reiniciar desde 1, sino pasar a 1.2

* + - * + 1.2) Si no existe, el esquema está en BCNF
  + 3FN (tercera forma normal): se utiliza cuando no se puede llevar a BCNF porque se pierden df, y asegura que no se pierda información, no se pierdan df, pero no siempre se quitan las anomalías.
    - Un esquema de relación R esta en 3FN si para toda dependencia de la forma X 🡪 A, se cumple que:
      * X 🡪 A es trivial, o bien,
      * X es superclave, o bien,
      * A es primo *(atributo primo: atributo que forma parte de alguna clave candidata)*
    - Como llevar un esquema R a 3FN: de manera esquemática y simplificada, una vez halladas las df y las claves candidatas y habiendo detectado que no se puede llevar a BCNF.
      * Se construye una tabla por cada df.
      * Si la clave de la tabla original, no esta incluida en ninguna de las tablas del punto anterior, se construye una tabla con la clave.
  + 1FN (primera forma normal): los atributos de la relación son simples y atómicos.
  + 2FN (segunda forma normal): un esquema de relación R esta en 2FN si para toda dependencia de la forma X 🡪 A, se cumple que: A depende de manera total de la clave.
* Dependencia Multivaluada:

Una dependencia multivaluada, afirma que dos o mas atributos son independientes del resto. Como consecuencia de la independencia, se tiene redundancia. Esta redundancia no se elimina con las df.

* + Se puede decir que: X -->> Y si dado un valor de X, hay un conjunto de valores de Y asociados y este conjunto de valores de Y NO esta relacionado (ni funcional ni multifuncionalmente) con los valores de R – X – Y (donde R es el esquema), es decir Y es independiente de los atributos de R – X – Y.
  + Sea R un esquema de relación
  + Otra forma de definirla:
    - La Dependencia Multivaluada X->>Y vale en R si para todos los pares de tuplas t1 y t2 en R, tal que:
      * t1[X] = t2[X] existen las tuplas t3 y t4 en R tales que:
        + t1[X] = t2[X] = t3[X] = t4[X]
        + t3[Y] = t1[Y]
        + t3[R-X-Y] = t2[R-X-Y]
        + t4[Y] = t2[Y]
        + t4[R-X-Y] = t1[R-X-Y]
* Dependencia Multivaluada trivial:

Sea R un esquema de relación, una dependencia multivaluada de la forma X ->> Y que vale en R es trivial si el conjunto de atributos X, Y son todos los atributos del esquema.

* Como proceder cuando se hallan dependencias multivaluada:
  + 4FN (cuarta forma normal): un esquema R esta en 4FN con respecto a un conjunto de dependencias multivaluada D, si para toda dependencia multivaluada de la forma x ->> Y se cumple que X ->> Y es una dependencia multivaluada trivial.
    - En otras palabras, un esquema esta en 4FN cuando no tiene dependencias multivaluada o bien, las dependencias multivaluada que en el valen, son triviales.