Fundamentos de Bases de Datos

Facultad de Ciencias, UNAM

M.I. Gerardo Avilés Rosas ✉ *gar@ciencias.unam.mx*

Normalización de Bases de Datos. Nociones

Uno de los principales problemas que se presentan cuando se convierten directamente diseños de bases de datos del **modelo E/R** al **modelo relacional** es la **REDUNDANCIA**.

Redundancia. *Consiste en que un hecho se repita en más de una tupla de* ***forma innecesaria****. Una de las causas más comunes es debido al hecho de tratar de incluir en una relación atributos* ***univaluados*** *y* ***multivaluados****.*

Anomalía. ***Error o inconsistencia*** *que puede resultar cuando un usuario intenta actualizar una tabla que contiene* ***datos redundantes****. Se trata de un problema que surge en una base de datos.*

Los principales **tipos de anomalías** son.

- **Redundancia.** La información se repite de forma innecesaria en varias tuplas.
- **Anomalía de actualización.** Podemos cambiar información en una tupla y dejarla inalterada en otra.
- **Anomalía de eliminación.** Si un conjunto de valores queda vacío podemos perder información adicional como efecto secundario.

sucursal	alcaldía	activo	cliente	numPrestamo	importe
Centro	Cuauhtémoc	$1,800 M	Santos	P-17	$200,000
Copilco	Coyoacán	$420 M	Gómez	P-23	$400,000
Viveros	Coyoacán	$340 M	López	P-15	$300,000
Centro	Cuauhtémoc	$1,800 M	Toledo	P-14	$300,000
Eugenia	Benito Juárez	$80 M	Santos	P-93	$100,000
Zapata	Benito Juárez	$1,600 M	Pérez	P-11	$180,000
San Ángel	Álvaro Obregón	$60 M	Vázquez	P-29	$240,000
San Fernando	Tlalpan	$740 M	López	P-16	$260,000
Centro	Cuauhtémoc	$1,800 M	González	P-23	$400,000
Viveros	Coyoacán	$340 M	Rodríguez	P-25	$500,000

¿Cómo acabar con las anomalías?

Una forma de **acabar con anomalías** como la redundancia es a través de la **descomposición de relaciones**. Se trata de una técnica que desarrollo **Edgar Frank Codd** (1972) y se basa en el proceso de **descomponer relaciones** con anomalías (sucesivamente) para producir **relaciones pequeñas** y **bien estructuradas**:

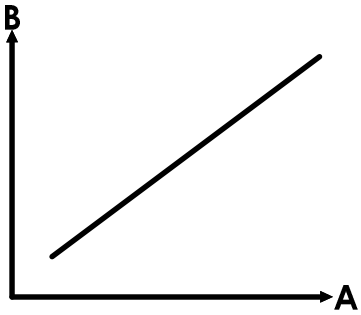
- *El proceso comprueba el cumplimiento de una* ***serie de reglas****.*
- *Cada que una* ***regla se cumple****, aumenta el* ***grado de normalización****.*
- *Cuando una* ***regla no se cumple****, la relación debe* ***descomponerse en varios esquemas*** *que si la cumplan.*

Objetivos de la normalización

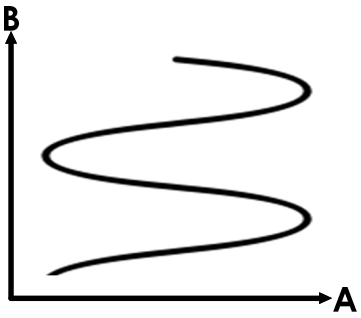
- **Minimizar la redundancia** de datos, **evitando anomalías** y **conservando espacio** de almacenamiento.
- **Simplificar** el cumplimiento de las **restricciones de integridad referencial**.
- Hacer **más fácil** el **mantenimiento** de datos (***insert, delete, update***)
- Proporcionar un **mejor diseño** (*representación mejorada del mundo real*) y una **base sólida** para el crecimiento futuro.
- Lograr que las relaciones fraccionadas tengan **JOIN sin pérdida**.
- **Conservar las dependencias funcionales**.

Dependencias Funcionales

- Ayudan a **especificar formalmente** cuándo un **diseño es correcto**.
- Se trata de una **relación unidireccional** entre dos atributos de tal forma que: *en un momento determinado, para **cada valor único** de A , solo **un valor** de B se asocia con él a través de la **relación**.*



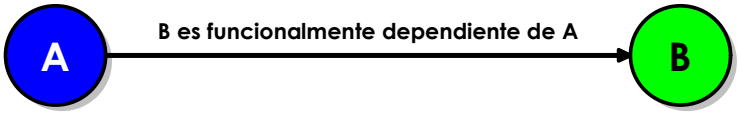
A determina funcionalmente a B. Cada valor de A corresponde con **un solo valor** de B.



A no determina funcionalmente a B. Algunos valores de A corresponden con **más de un valor** de B.

- Se trata de una **restricción entre dos atributos** (o *subconjuntos* de atributos) en la cual, el **valor** de **un atributo** es **determinado** por **el valor** de **otro atributo**:

  *Para una relación R, un atributo B es **funcionalmente dependiente** de un atributo A, si para **cada instancia** válida de A, el valor de A **únicamente determina** el valor de B.*



- Una **dependencia funcional** la denotaremos por $X \rightarrow Y$, y sucede entre **dos conjuntos** de atributos X e Y que son **subconjuntos** de R.

Utilidad de las dependencias funcionales

- Especificar **restricciones** sobre el conjunto de relaciones.
- Examinar las relaciones y determinar **si son legales** bajo un conjunto de **dependencias funcionales** dado.

Características

- Si X es una llave de $R \Longrightarrow X \rightarrow Y \quad \forall \quad Y \subset R$.
- Si $X \rightarrow Y$ no implica que $Y \rightarrow X$.
- Se deben cumplir para la **extensión de una relación**.
- Las **extensiones** que satisfacen las **dependencias funcionales** se denominan **estados legales**.

Definición formal de llave. Conjunto de atributos $A_1, A_2, A_3, \ldots, A_n$ tales que:

i. Determinan funcionalmente cualquier otro atributo de R (**dos tuplas** distintas **no pueden coincidir** en todos los atributos $A_1, A_2, A_3, \ldots, A_n$).
ii. Ningún **subconjunto propio** de $A_1, A_2, A_3, \ldots, A_n$ **determina funcionalmente** a otros atributos de R, es decir, debe ser **mínimo**.

Llave candidata. Se trata de un **atributo** o **combinación de atributos**, que **identifica de manera única** una tupla en una relación. Una **llave candidata** debe cumplir las siguientes propiedades (**Dutka y Hanson**, 1989):

1. **Identificación única.** Para cada tupla, el valor de la llave **debe identificar** de **forma única** esa tupla. Esta propiedad implica que **cada atributo no llave** depende funcionalmente de esa llave.
2. **No redundancia**. Ningún atributo en la llave se puede **eliminar** sin destruir la propiedad de **identificación única**.

Superllave. Conjunto de atributos que **contiene una llave**.

Reglas de inferencia de Armstrong

Desarrolladas por **William W. Armstrong**(1974) (*Dependecy Structures of Data base relationships*). Se trata de un **conjunto de reglas** que permiten **deducir todas las dependencias funcionales** que tienen lugar en un conjunto de atributos dado, como consecuencia de aquellas que se **asumen como ciertas**. Permiten establecer **algoritmos** para:

- **Encontrar la cerradura de un conjunto de dependencias funcionales**
- **Encontrar equivalencia lógica de esquemas**
- **Deducir dependencias**
- **Calcular las llaves de un esquema**

Reglas:

1. **Reflexividad:** Sí $Y \subseteq X \Longrightarrow X \to Y$
2. **Aumento:** $\{X \to Y\} \Longrightarrow XZ \to YZ$
3. **Transitividad:** $\{X \to Y, Y \to Z\} \Longrightarrow X \to Z$
4. **Descomposición:** $\{X \to YZ\} \Longrightarrow X \to Y \; y \; X \to Z$
5. **Unión:** $\{X \to Y, X \to Z\} \Longrightarrow X \to YZ$
6. **Pseudo-transitividad:** $\{X \to Y, W\,Y \to Z\} \Longrightarrow WX \to Z$

Formas Normales

Relación bien estructurada. Se trata de una relación que contiene **redundancia mínima** y permite a los usuarios insertar y modificar tuplas de la tabla sin provocar errores o inconsistencias.

Forma normal. **Estados de una relación** que requiere que ciertas normas relativas a las relaciones entre los atributos (o dependencias funcionales) se satisfagan. Pasos en la normalización:

1. **Primera forma normal.** Cualquier atributo multivaluado (incluso grupos repetidos) han sido removidos. Solo se permiten valores atómicos y posiblemente nulos.
2. **Segunda forma normal.** Cualquier dependencia funcional parcial se han removido (los atributos que no son llave se identifican por toda la llave primaria).
3. **Tercera forma normal.** Cualquier dependencia transitiva se ha removido (atributos que no son llave son identificados solo por la llave primaria).
4. **Forma normal de Boyce-Codd.** Anomalías restantes, resultados de las dependencias funcionales se han removido (puede haber más de una llave primaria para los mismos atributos).
5. **Cuarta forma normal.** Cualquier dependencia multivaluada se han removido.
6. **Quinta forma normal.** Anomalías que no se pudieron retirar por las anteriores formas normales, se han removido.

Forma Normal de Boyce-Codd (BCNF)

Una relación $\boldsymbol{R}$ *está en* $BCNF$ *si y sólo si en toda* ***DF no trivial*** $\boldsymbol{A_1, A_2, A_3, \dots, A_n \to B}$ *para* $\boldsymbol{R}$*, se tiene que* $\{\boldsymbol{A_1, A_2, A_3, \dots, A_n}\}$ *es* ***superllave*** *de* $\boldsymbol{R}$.

Algoritmo para obtener BCNF

1. **Buscar una dependencia funcional no trivial** $\boldsymbol{X \to B}$ **que viole BCNF.**
2. **Calcular** $\boldsymbol{X+}$.
3. **Fraccionar** $\boldsymbol{R}$ **en** $\boldsymbol{R_1(X+) \cup R_2\big((R - X+) \cup X\big)}$.
4. **Encontrar las dependencias funcionales para las nuevas relaciones.**

Tercera Forma Normal

Una relación $\boldsymbol{R}$ *está en* ***tercera forma normal*** *(*$\boldsymbol{3NF}$*) con respecto a* $\boldsymbol{F}$ *, si para toda* ***dependencia funcional no trivial*** $\boldsymbol{A_1, A_2, A_3, \dots, A_n \to B}$*, se tiene que:*

1. *El lado izquierdo* $\{\boldsymbol{A_1, A_2, A_3, \dots, A_n}\}$ *es una* ***superllave*** *o bien,*
2. *El lado derecho* $\boldsymbol{B}$*, es miembro de alguna* ***llave candidata*** *de* $\boldsymbol{R}$.

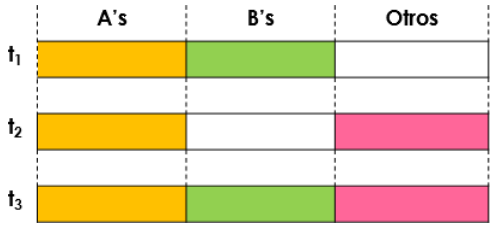
Algoritmo para obtener 3NF

1. Hacer F **mínimo**
2. Para toda **dependencia funcional** en F **mínimo:**
   a. **Crear una relación** que contenga **sólo los atributos** de cada **dependencia funcional**.
   b. **Eliminar un esquema** si es **subconjunto** de otro.
3. **Si no existen esquemas que contengan llaves candidatas, crear una relación con esos atributos.**

Dependencia Multivaluada

Existe una ***dependencia multivaluada*** *(**DMV**)* $A_1, A_2, \dots, A_n \twoheadrightarrow B_1, B_2, \dots, B_m$ *sí para cada par de tuplas* t_1 *y* t_2 *de la relación* R*, que coinciden en todos los valores de las* A*'s, podemos encontrar una tupla* t_3 *tal que coincida con:*

1. t_1 *y* t_2 *en las* A*'s.*
2. t_1 *en las* B*'s y,*
3. t_2 *en todos los atributos de* R *que no están ni en* A *ni en* B*.*



Cuarta Forma Normal

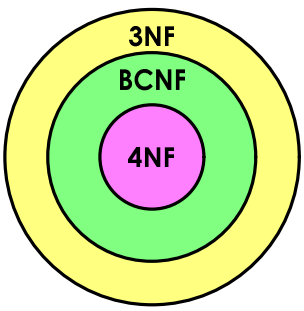
- *Una relación* R *está en* $4FN$ *si toda* DMV ***no trivial*** $A_1, A_2, \dots, A_n \twoheadrightarrow B_1, B_2, \dots, B_m$*, tiene que* $\{A_1, A_2, \dots, A_n\}$ *es una* ***superllave****.*
- *Es importante hacer notar que las nociones de* ***llave*** *y* ***superllave*** *dependen sólo de las* ***dependencias funcionales****. La* $4NF$ *es una generalización de la* $BCNF$ *debido a que toda* DF *es una* DMV*.*
- *Por lo tanto, toda violación a la* $BCNF$ *es una violación a la* $4NF$*, pero al revés* ***no es cierto****.*

Algoritmo para obtener 4NF

Objetivo: *Eliminar la* ***redundancia*** *debida al* ***efecto multiplicativo*** *de las* $DMVs$*.*

1. *Encontrar una violación a la* $4NF$*, es decir,* $A_1, A_2, \dots, A_n \twoheadrightarrow B_1, B_2, \dots, B_m$ *donde el conjunto de las* A*'s no forma una* ***superllave****.*
2. ***Dividir la relación*** *en* ***dos esquemas****:*
   - *El que contiene las* A*'s y las* B*'s*
   - *El que contiene las* A*'s y los atributos de* R *que no están entre las* B*'s.*
3. *Si alguno de los esquemas* ***no estuviera*** *en* $4NF$*, regresar al* ***paso 1****.*

Nota: *En este caso,* ***no hay pruebas análogas*** *a las de la* ***cerradura de atributos (DFs)*** *para* $DMVs$*.*



Propiedad	3NF	BCNF	4NF
Elimina la redundancia por dependencias funcionales	La mayor parte	Sí	Sí
Elimina la redundancia debida a dependencias multivaluadas	No	No	Sí
Conserva las dependencias funcionales	Sí	Quizá	Quizá
Conserva las dependencias multivaluadas	Quizá	Quizá	Quizá