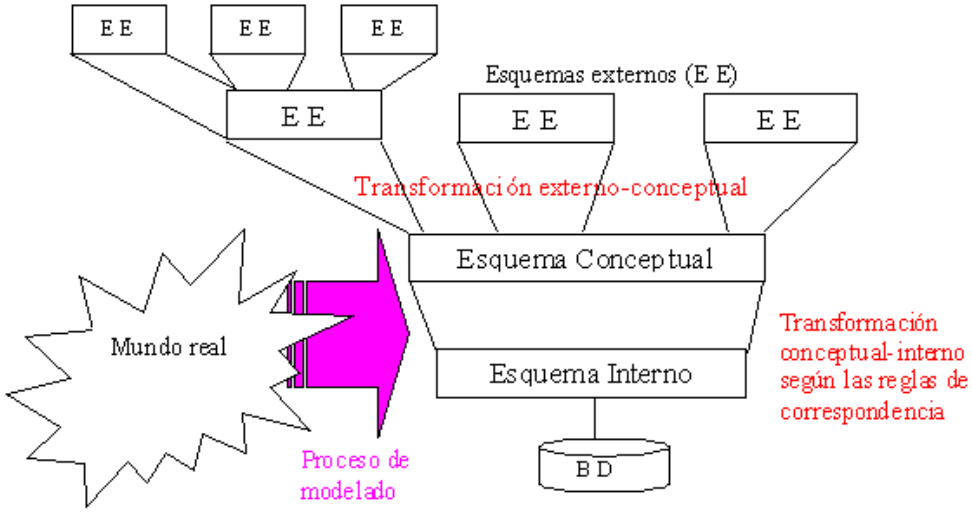


UNIVERSIDAD DE LOS ANDES

Bases de datos Unidad 3

Universidad de Los Andes
Escuela de Ingeniería de Sistemas
Departamento de Computación

Tema 1. Modelo relacional y objeto-relacional

Tema 1. Modelo relacional y objeto-relacional

- **Contenido:**
  - Conceptos básicos
  - Reglas de transformación de ERE o del diagrama de clases UML al objeto-relacional
  - Enfoque por descomposición
  - Normalización
  - Restricciones y reglas de integridad
- **Objetivo:**
  - Desarrollar habilidades en el modelado de bases de datos relacional y objeto-relacional
- **Actividades:**
  - Leer: Elmasri y Navathe, cap. 5, 7, 10, 11, 16 y 22

UNIVERSIDAD DE LOS ANDES

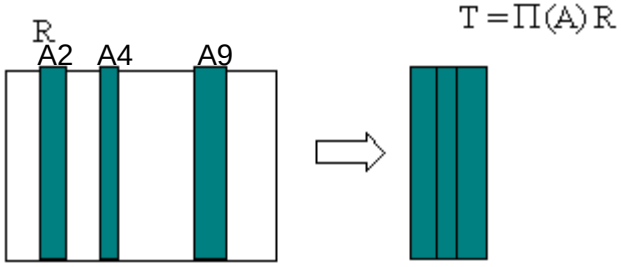
Enfoque por descomposición

- **Consiste en definir relaciones universales compuestas de todos los atributos de la base de datos y luego descomponerlas, utilizando el proceso de normalización de las relaciones en sub-relaciones que no sufren anomalías**
- **Es un proceso de refinamiento paso a paso, que lleva al aislamiento de las entidades y asociaciones del mundo real [Codd, 1979]**

Teoría de la descomposición de las relaciones

- **Se basa en el uso de las operaciones fundamentales del álgebra relacional**
- **Operaciones fundamentales:**
  - Proyección:
    - La proyección de una relación R(A1, A2, …, An) sobre los atributos $Ai_1, Ai_2, \ldots, Ai_p$, con $j \neq i_k$, es una relación R' con esquema R'($Ai_1, Ai_2, \ldots, Ai_p$) obtenida por eliminación de los valores de los atributos de R que no están en R' y la supresión de tuplas duplicadas
  - Notación: $\pi\, Ai_1, Ai_2, \ldots, Ai_p\ (R) = R'$

A=A2, A4 y A9



Proyección

- **Carro(placa, marca, modelo, color)**
  - ∏ placa, marca (Carro) = R
  - ∏ marca, color (Carro) = Q

Carro	placa	marca	modelo	color
	'MBO34L'	'Ford'	'Ka'	'verde'
	'LDA75K'	'Toyota'	'corollaXL'	'blanco'
	'ADA89A'	'Fiat'	'siena'	'gris'
	'LBF78G'	'Toyota'	'corollaXL'	'blanco'
	'XSA67D'	'Ford'	'Ka'	'rojo'

Q	marca	color
	'Ford'	'verde'
	'Toyota'	'blanco'
	'Fiat'	'gris'
	'Ford'	'rojo'

R	placa	marca
	'MBO34L'	'Ford'
	'LDA75K'	'Toyota'
	'ADA89A'	'Fiat'
	'LBF78G'	'Toyota'
	'XSA67D'	'Ford'

UNIVERSIDAD DE LOS ANDES

Reunión natural (join)

- El producto, reunión o acoplamiento de dos relaciones R y S cuyos esquemas son $R(A_1, A_2, \ldots, A_n)$ y $S(B_1, B_2, \ldots, B_p)$ es una relación T con atributos que son la unión de los atributos de R y S para las tuplas obtenidas por concatenación de las tuplas de R y S que tengan los mismos valores para los atributos de igual nombre
- Notación: $T = R \bowtie S$

Reunión natural

R	placa	marca
	‘MBO34L’	‘Ford’
	‘LDA75K’	‘Toyota’
	‘ADA89A’	‘Fiat’
	‘LBF78G’	‘Toyota’
	‘XSA67D’	‘Ford’

Q	marca	color
	‘Ford’	‘verde’
	‘Toyota’	‘blanco’
	‘Fiat’	‘gris’
	‘Ford’	‘rojo’

$$T = R \underset{marca}{\bowtie} Q$$

T	placa	marca	color
	‘MBO34L’	‘Ford’	‘verde’
	‘MBO34L’	‘Ford’	‘rojo’
	‘LDA75K’	‘Toyota’	‘blanco’
	‘ADA89A’	‘Fiat’	‘gris’
	‘LBF78G’	‘Toyota’	‘blanco’
	‘XSA67D’	‘Ford’	‘verde’
	‘XSA67D’	‘Ford’	‘rojo’

UNIVERSIDAD DE LOS ANDES

Descomposición

- **Es el reemplazo de una relación $R(A_1, A_2, \ldots, A_n)$ por una colección de relaciones $R'_1, R'_2, \ldots, R'_n$ obtenidas de las proyecciones de R y tal que la relación resultado de las reuniones $R'_1 \bowtie R'_2 \bowtie \ldots \bowtie R'_n$ tiene el mismo esquema que R.**

R1 = Π placa, modelo, color (Carro)

R2 = Π modelo, marca (Carro)

R $\bowtie$ Q $\neq$ Carro pero R1 $\bowtie$ R2 = Carro

UNIVERSIDAD DE LOS ANDES

R	placa	marca
	‘MBO34L’	‘Ford’
	‘LDA75K’	‘Toyota’
	‘ADA89A’	‘Fiat’
	‘LBF78G’	‘Toyota’
	‘XSA67D’	‘Ford’

Q	marca	color
	‘Ford’	‘verde’
	‘Toyota’	‘blanco’
	‘Fiat’	‘gris’
	‘Ford’	‘rojo’

marca

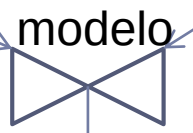
T	placa	marca	color
	‘MBO34L’	‘Ford’	‘verde’
	‘MBO34L’	‘Ford’	‘rojo’
	‘LDA75K’	‘Toyota’	‘blanco’
	‘ADA89A’	‘Fiat’	‘gris’
	‘LBF78G’	‘Toyota’	‘blanco’
	‘XSA67D’	‘Ford’	‘verde’
	‘XSA67D’	‘Ford’	‘rojo’

≠

Carro	placa	marca	modelo	color
	‘MBO34L’	‘Ford’	‘Ka’	‘verde’
	‘LDA75K’	‘Toyota’	‘corollaXL’	‘blanco’
	‘ADA89A’	‘Fiat’	‘siena’	‘gris’
	‘LBF78G’	‘Toyota’	‘corollaXL’	‘blanco’
	‘XSA67D’	‘Ford’	‘Ka’	‘rojo’

R1	placa	modelo	color
	‘MBO34L’	‘Ford’	‘verde’
	‘LDA75K’	‘Toyota’	‘blanco’
	‘ADA89A’	‘Fiat’	‘gris’
	‘LBF78G’	‘Toyota’	‘blanco’
	‘XSA67D’	‘Ford’	‘rojo’

R2	modelo	marca
	‘Ka’	‘Ford’
	‘corollaXL’	‘Toyota’
	‘siena’	‘Fiat’



Carro	placa	marca	modelo	color
	‘MBO34L’	‘Ford’	‘Ka’	‘verde’
	‘LDA75K’	‘Toyota’	‘corollaXL’	‘blanco’
	‘ADA89A’	‘Fiat’	‘siena’	‘gris’
	‘LBF78G’	‘Toyota’	‘corollaXL’	‘blanco’
	‘XSA67D’	‘Ford’	‘Ka’	‘rojo’

Descomposición sin pérdida

- **Es la descomposición de una relación R en R'_1, R'_2, …,R'_p tal que para toda extensión de R se tiene que:**

$$R = R'_1 \bowtie R'_2 \bowtie \ldots \bowtie R'_p$$

- **El problema de la concepción de bases de datos relacionales se reduce a la descomposición sin pérdida de las relaciones universales con todos sus atributos en sub-relaciones que no contengan anomalías**

UNIVERSIDAD DE LOS ANDES

Normalización

- **El esquema relacional es un modelo de la realidad bajo la forma de una colección de relaciones para:**
  1. **La creación, modificación y supresión de datos eficazmente. Es indispensable eliminar toda redundancia innecesaria. Idealmente, ante la ocurrencia de un evento se desea que éste se traduzca en el manejo de una única tupla en la extensión del esquema relacional**
  2. **La modificación del esquema relacional por la evolución de la percepción de la realidad, sea lo más simple posible**
  3. **La comprensión de la realidad sea facilitada por el esquema relacional**

UNIVERSIDAD DE LOS ANDES

Dependencias funcionales

- Sea $R(A_1, A_2, \ldots, A_n)$ y X y Z dos subconjuntos del conjunto formado por $\{A_1, A_2, \ldots, A_n\}$
- Se dice que $X \rightarrow Z$ (X determina Z o que Z depende funcionalmente de X) si para toda extensión r de R y para toda tupla t_1 y t_2 de r se tiene que $\Pi_X(t_1) = \Pi_X(t_2)$ implica que $\Pi_Z(t_1) = \Pi_Z(t_2)$

Ejemplo: placa $\rightarrow$ marca, placa $\rightarrow$ modelo, placa $\rightarrow$ color, placa $\rightarrow$ (marca, modelo), modelo $\rightarrow$ marca

- Las DF se identifican mirando atentamente el significado de los atributos, no sus valores actuales, sino todos los valores posibles de ellos

UNIVERSIDAD DE LOS ANDES

Propiedades de las DF

- **Las DF deben aparecer en el esquema conceptual de la base de datos relacional**

Ejemplo: (codMateria, cedulaEstudiante, semestre, año, seccion) → nota

1. **Reflexibidad: Si $Y \subseteq X \Rightarrow X \rightarrow Y$**

   Ejemplo: color → color, (marca, modelo) → marca

2. **Aumento: Si $X \rightarrow Y \Rightarrow XZ \rightarrow YZ$**

   Ejemplo: modelo → marca ⇒ (modelo, color) → (marca, color)

3. **Transitividad: Si $X \rightarrow Y$ y $Y \rightarrow Z \Rightarrow X \rightarrow Z$**

   Ejemplo: placa → modelo y modelo → marca ⇒ placa → marca

UNIVERSIDAD DE LOS ANDES

Propiedades de las DF

4. **Aditividad: Si $X \rightarrow Y$ y $Y \rightarrow Z \Rightarrow X \rightarrow YZ$**

   Ejemplo: placa $\rightarrow$ modelo y modelo $\rightarrow$ marca $\Rightarrow$ placa $\rightarrow$ (modelo, marca)

5. **Pseudo-transitividad: Si $X \rightarrow Y$ y $XWY \rightarrow Z \Rightarrow WY \rightarrow Z$**

   Ejemplo: placa $\rightarrow$ modelo y (marca, modelo) $\rightarrow$ potencia $\Rightarrow$ (placa, marca) $\rightarrow$ potencia

6. **Descomposición: Si $X \rightarrow Y$ y $Z \subseteq Y \Rightarrow X \rightarrow Z$**

   Ejemplo: placa $\rightarrow$ (modelo, marca) y modelo $\subseteq$ (modelo, marca) $\Rightarrow$ placa $\rightarrow$ modelo

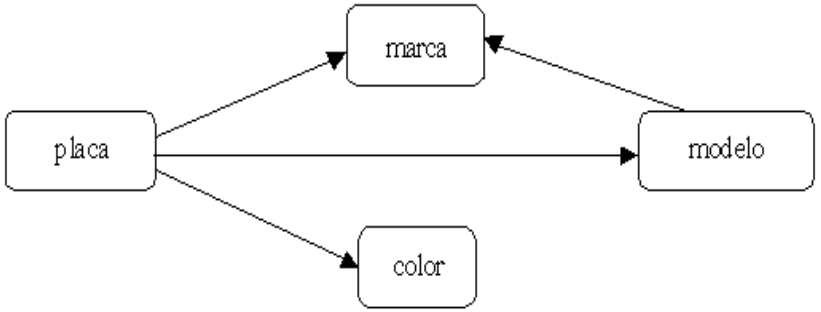
Dependencias funcionales elementales (DFE)

- **Es una DF de la forma $X \rightarrow A$, donde A es un atributo único no incluido en X y donde no existe un $X' \subseteq X$ tal que $X' \rightarrow A$**

  Ejemplo: placa $\rightarrow_{DFE}$ modelo, pero

  placa $\rightarrow_{\text{no es DFE}}$ (modelo, marca)

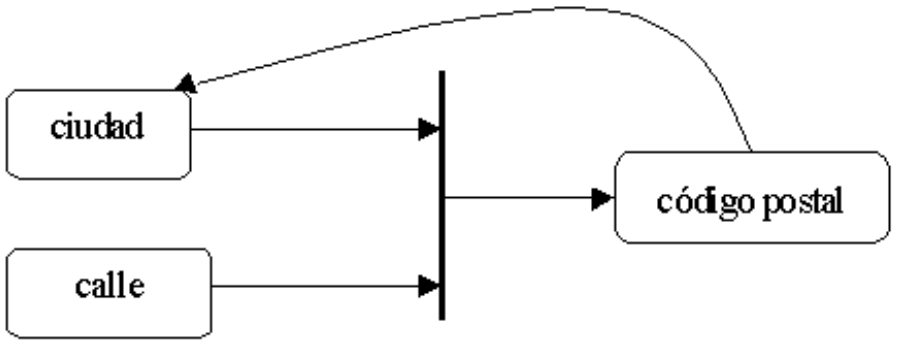
- **La regla de inferencia que se aplica a las DFE es la transitividad**
- **Grafo de DFE: Los nodos son los atributos y las aristas son las**



(a) Grafo de DFE

Redes de DFE

- **En caso de tener más de un atributo en la parte izquierda de la DFE, ésta se expresa colocando una línea que acoja las aristas de todos los atributos de la parte izquierda y de ella sale una arista al atributo de la parte derecha**



(b) Red de DFE

UNIVERSIDAD DE LOS ANDES

Cierre transitivo (C+)

- A partir de las DFE se pueden componer otras DFE utilizando la propiedad de transitividad
- **C+: Es el conjunto de las DFE consideradas, enriquecidas con todas las DFE deducidas por transitividad**
  - Ejemplo: Para la relación Carro se tiene:

  C+ = {placa $\rightarrow$ marca, placa $\rightarrow$ modelo, placa $\rightarrow$ color, modelo $\rightarrow$ marca }
- **Dos conjuntos de DFE son equivalentes si tienen el mismo C+**

Cobertura mínima (C)

- **Es el conjunto C de DFE asociado a un conjunto de atributos que verifican las propiedades siguientes:**
  a. ninguna DF es redundante en **C**, es decir para toda DF denotada **f** de **C**, **C - f** no es equivalente a **C**
  b. toda DFE de los atributos está dentro de **C+**

  Ejemplo: **C** = {placa → modelo, placa → color, modelo → marca }
- **C es esencial para la descomposición sin pérdida**

Clave de una relación

- **Es el conjunto X de atributos de una relación $R(A_1, A_2, ..., A_n)$ tal que:**
  - $X \rightarrow A_1, A_2, ..., A_n$
  - no existe un subconjunto $Y \subseteq X$ tal que $Y \rightarrow A_1, A_2, ..., A_n$
- **Pueden existir varios atributos que cumplan con esta definición dentro de una misma relación (claves candidatas), se escoge una de ellas como clave primaria.**
- **Dentro de una relación, la clave primaria se subraya**

UNIVERSIDAD DE LOS ANDES

Normalización: 1FN

- **Objetivo de las tres primeras formas normales**
  - Permitir la descomposición de relaciones sin pérdida de información, a partir de las DFE y obtener el esquema conceptual relacional normalizado
- **Primera forma normal (1FN): Una relación está en 1FN si todo atributo contiene un valor atómico**

Persona(cedula, nombre, apellido, sexo, telefono, direccion)

- los primeros cinco atributos son atómicos y el atributo **direccion** puede ser considerado atómico en aquellas aplicaciones donde esta columna no va a ser utilizada como un atributo de búsqueda, lo que implica que la relación Persona está en 1FN

Estudiante(cedula, apellido, nombre, escuela, materias, notas)

- los primeros cuatro atributos son atómicos, pero los dos últimos no lo están, la relación no está en 1FN.
- Para convertirla a 1FN se proyecta en dos relaciones, obteniendo:

Estudiante(cedula, apellido, nombre, escuela)

Cursa(cedula, materia, nota)

2FN

- **Segunda forma normal (2FN): Una relación está en 2FN si y solo si:**
  1. la relación está en 1FN
  2. todo atributo que no pertenece a una clave no puede depender de una parte de esa clave

  Ejemplo:

  Proveedor(codProv, codArt, dirProv, precio)

  - Ella está en 1FN considerando la dirProv como una columna atómica, pero dadas las DFE siguientes: (codProv, codArt) → precio y codProv → dirProv, ella no está en 2FN
  - Para normalizarla se proyecta en dos relaciones:

    Proveedor(codProv, dirProv)

    ProveeArticulos(codProv, codArt, precio)

  Carro(placa, marca, modelo, color) está en 2FN

- **La segunda forma normal permite eliminar las redundancias para que ningún atributo está determinado por una parte de una clave**

3FN

- **Tercera forma normal (3FN): Una relación está en 3FN si y solo si:**
  1. la relación está en 2FN
  2. todo atributo que no pertenece a la clave no depende de un atributo que no es clave

  Ejemplo:

  Carro(placa, marca, modelo, color) está en 2FN, pero no en 3FN ya que se tiene la DFE modelo $\rightarrow$ marca

  Para normalizarla se proyecta en dos relaciones:

  Carro(placa, modelo, color)

  ModelosDeCarros(modelo, marca)

- **3FN permite asegurar la eliminación de redundancias debidas a las dependencias transitivas**

Descomposición que preserva las DFs

- **La descomposición $\{R_1, R_2, ..., R_n\}$ de una relación R preserva las DF de R, si C+ de R es la misma que la de la unión de las DF de $\{R_1, R_2, ..., R_n\}$**
- **Toda relación R tiene al menos una descomposición en 3FN tal que:**
  1. la descomposición preserve las DF
  2. la descomposición sea sin pérdida

Algoritmo de descomposición en 3FN

- **Propuesto por Bernstein en 1976 se basa en el principio siguiente:**
  - Se construye la cobertura mínima **C** y se editan los atributos aislados, considerándolos como claves,
  - luego se busca el conjunto más grande X de atributos que determine a otros $A_1, A_2, ..., A_n$ con $n \geq 1$ y como salida se genera la relación $(X, A_1, A_2, ..., A_n)$
  - Las DFE utilizadas en la formación de esa relación se eliminan de **C** y todos los atributos aislados que no están en las DFE que quedaron en **C**

Algoritmo de descomposición en 3FN

Procedimiento Normalizar3FN(DFE)

1. C = cobertura mínima de las DFE
2. At = Obtener los atributos aislados que pertenecen a C
3. reducir(C, At)
4. formar una R con los atributos restantes en At, si los hay
5. fin del procedimiento

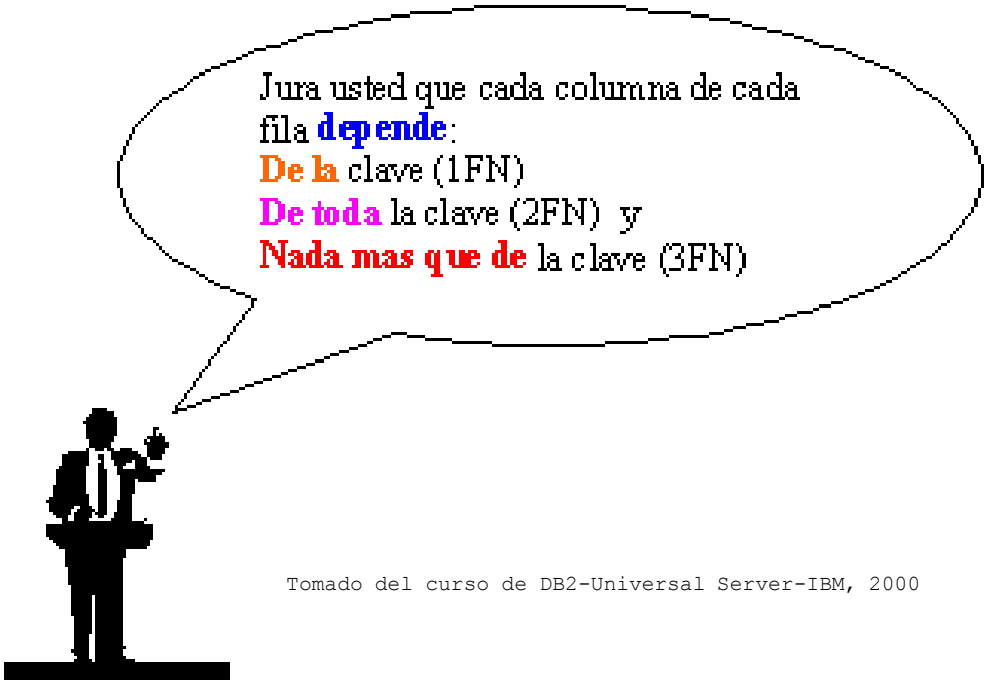
Procedimiento reducir(C, At)

1. repita mientras que una DFE en C no incluya todos los atributos o C está vacío
   1. buscar el conjunto más grande de atributos X tal que $X \rightarrow A_1, ..., X \rightarrow A_k$
   2. formar la relación $R(X, A_1, A_2, ..., A_k)$
   3. eliminar de C las DFE utilizadas en R
   4. eliminar de At los atributos que no pertenezcan ya a C
   5. reducir(C, At)

   fin del repita mientras
2. regresar

Juramento

- **Un esquema normalizado hasta 3FN debe cumplir con el juramento siguiente:**



Tomado del curso de DB2-Universal Server-IBM, 2000

Forma normal de Boyce-Codd (FNBC)

- **Una relación está en FNBC si y solo si las solas DFE son aquellas dentro de las cuales una clave determina un atributo**

Ejemplo:

Examen(<u>cedEst, codMat</u>, cedProf, nota)

(cedEst, codMat) -> cedProf

cedProf -> codMat

(cedEst, codMat) -> nota

Está en 3FN

No está en FNBC si cada profesor dicta una única materia

Forma normal de Boyce-Codd (FNBC)

- **Para resolver el problema se proyecta en 2 relaciones**

  Examen(<u>cedEst, codMat</u>, nota)

  Dicta(<u>codMat</u>, cedProf)

  No se preserva la DFE (cedEst, codMat) -> cedProf

- **En general, la descomposición en FNBC es sin pérdida pero NO preserva las DFE, después ellas pueden obtenerse por reunión o producto**

Dependencias multivaluadas (DM)

- Sea $R(A_1, A_2, ..., A_n)$ y X e Y dos subconjuntos de atributos de $\{A_1, A_2, ..., A_n\}$. Se dice que X ->> Y, si dados los valores de X hay un conjunto de valores Y asociados y este conjunto es independiente de otros atributos Z = R – X – Y de R
- Las DM caracterizan la independencia entre Y y Z correlacionadas por X
- Las DF son un caso particular de las DM, por lo cual X -> Y => X ->> Y

Dependencias multivaluadas elementales (DME)

- **Una DME es una DM X ->>Y de una relación R tal que:**
  1. Y no es vacío y es disjunto de X
  2. R no contiene otra DM del tipo X' **->>** Y' tal que X' $\subset$ X y Y' $\subset$ Y

Ejemplo: EstMatDeporte (nroEst, codMat, deporte)

Un estudiante puede cursar varias materias y puede practicar varios deportes. codMat es independiente de deporte, solo están correlacionados a través de nroEst

EstMatDeporte	nroEst	codMat	Deporte
	105	'PR1'	'tennis'
	105	'PR1'	'natacion'
	145	'AL10'	'tennis'
	145	'FI20'	'futbol'

4FN

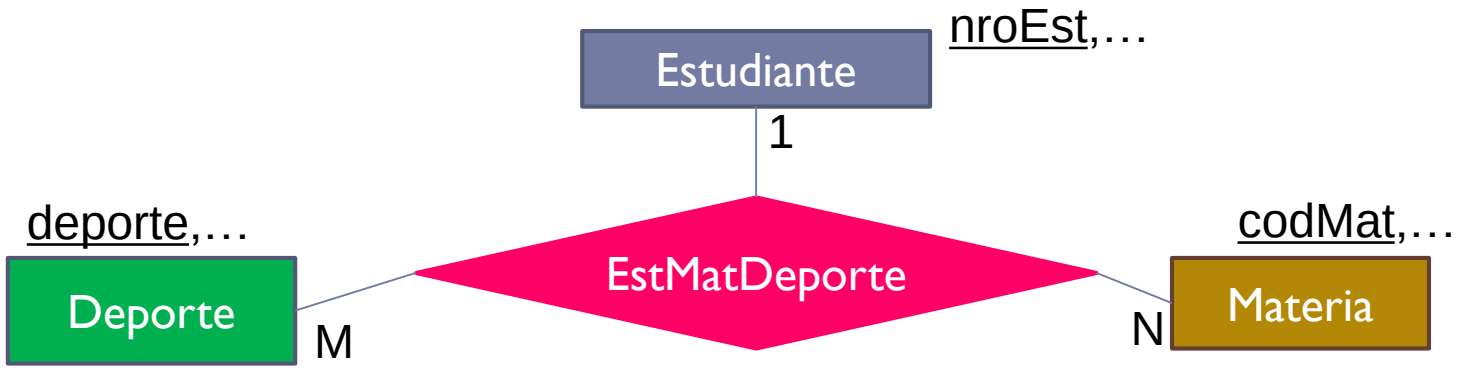
- **Una R está en 4FN si y solo si las solas DME son aquellas donde una clave determina un atributo. Una R en 4FN está en 3FN y en FNBC**

Ejemplo:

EstMatDeporte (nroEst, codMat, deporte) no está en 4FN, por lo que se proyecta según sus DME como:

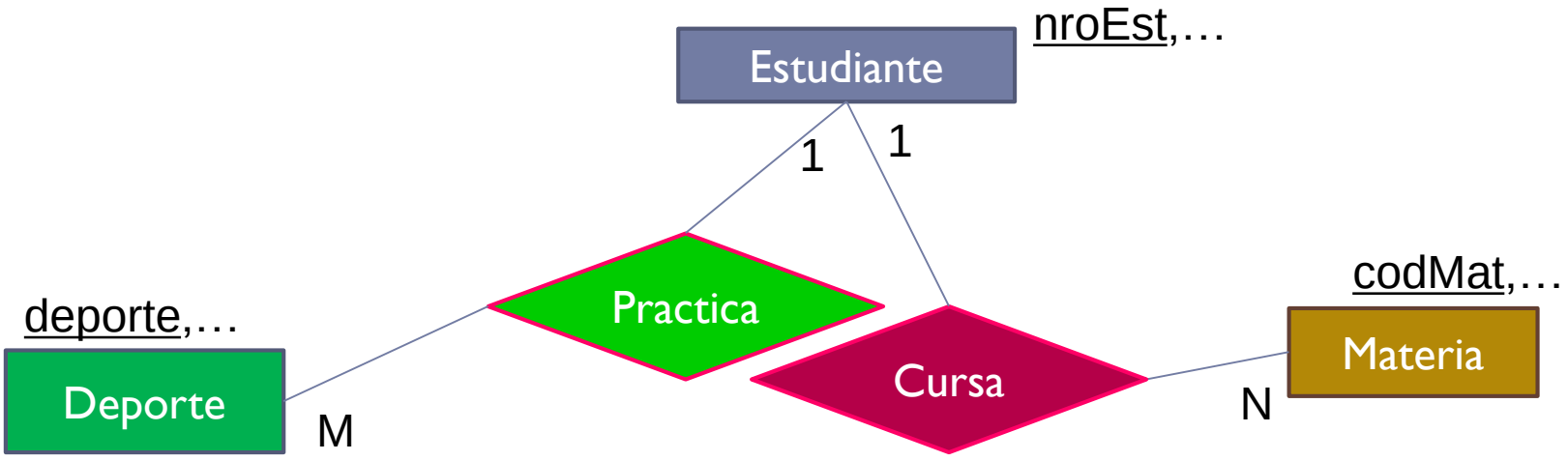
Cursa(nroEst, codMat)

Practica(nroEst, deporte)



Teorema de Fagin (1979)

- **R(A, B, C) se puede descomponer sin pérdida en R1(A, B) y R2(A, C) si y solo si se cumplen en R las DM A ->> B | C**
- **Demuestra que toda R tiene una descomposición (no siempre única) en 4FN sin pérdida de información**



Teorema de Fagin

Curso	nomCur	prof	Texto
	'Estadística'	'Perez'	'Estadística I'
	'Estadística'	'Perez'	'Introducción a la Estadística'
	'Estadística'	'Mendez'	'Estadística I'
	'Estadística'	'Mendez'	'Introducción a la Estadística'

Ejemplo:

Curso(nomCur, prof, texto)

nomCur ->> prof

nomCur ->> texto

Se proyecta como:

TextoMateria(nomCur, texto)

Dicta(nomCur, prof)

• Existen relaciones que no es posible descomponer-las en 2 relaciones, pero si en 3, 4 o más relaciones

Dependencias de producto (DP)

- **Sea $R(A_1, A_2, ..., A_n)$ y $X_1, X_2, ..., X_m$ subconjuntos de $\{A_1, A_2, ..., A_n\}$. Se dice que existe una DP simbolizada por $*\{X_1, X_2, ..., X_m\}$ si R es el producto de sus proyecciones sobre $X_1, X_2, ..., X_m$, es decir si**

$$R = \prod X_1(R) \prod X_2(R) \ldots \prod X_m(R)$$

Ejemplo:

Si el proveedor #E suministra la pieza #P y en el proyecto #J se usan piezas #P y el proveedor #E suministra piezas al proyecto #J, entonces #E suministra #P al proyecto #J

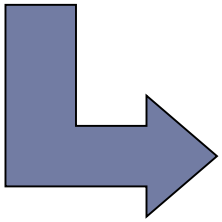
Suministro(#E, #P, #J) está en 4FN

No está en 5FN

pues #E **->>** #P, #P **->>** #J, #J **->>** #E, no es posible descomponerla en 2 relaciones, pero si es posible en 3 relaciones

DP

Suministro	#E	#P	#J
	E1	P1	J2
	E1	P2	J1
	E2	P1	J1
	E1	P1	J1



R1	#E	#P
	E1	P1
	E1	P2
	E2	P1

R2	#P	#J
	P1	J2
	P2	J1
	P1	J1

R3	#E	#J
	E1	J2
	E1	J1
	E2	J1

Suministro ≠ R1 ⋈ R2,

Suministro ≠ R1 ⋈ R3,

Suministro ≠ R2 ⋈ R3,

Suministro = R1 ⋈ R2 ⋈ R3

UNIVERSIDAD DE LOS ANDES

5FN

- **Una relación R está en 5FN si y solo si toda DP está implicada por las claves candidatas de R**
- **En la realidad no es común tener DP y es muy difícil darse cuenta de su existencia**
- **Fagin, 1979 presenta un algoritmo para probar si una DP está implicada por un conjunto de claves en R**

UNIVERSIDAD DE LOS ANDES

Restricciones

- **Son parte del esquema relacional**
- **Una vez que una restricción está declarada, las actualizaciones en la base de datos que violan la restricción no están permitidas**
- **Álgebra Relacional (AR) como un lenguaje de restricciones**
  - Si R es una expresión en AR, R=∅ es una restricción que dice "no hay tuplas en la extensión de R"
  - Si R y S son expresiones en AR, entonces R⊆S es una restricción que dice "cada tupla de R debe estar en S"

UNIVERSIDAD DE LOS ANDES

Restricciones de integridad

- **Son aserciones que deben verificar los datos en instantes determinados**
- **La integridad de los datos en bases de datos accedidas por procesos concurrentes debe ser asegurada, mediante la aplicación de restricciones y reglas que aseguren la concordancia de los datos que la base de datos modela con los del mundo real**
- **Bases de datos coherentes: Son bases de datos donde el conjunto de restricciones de integridad (explícitas o implícitas) se respeta a todo lo largo de la vida útil de la BD**

Tipos de restricciones de integridad

1. **Restricciones de dominio o integridad de dominio:**

   Están referidas al tipo de dato del atributo o columna

   El valor que se puede asignar a una columna debe estar en el dominio especificado para dicha columna

   Se permite a un dato estar marcado para contener un valor especial definido por el diseñador de la BD (NoDefinido), no contener valor alguno o contener el valor **nulo** si:

   1. Existe la posibilidad de desconocer la información (nulo aplicable)
   2. No tiene sentido asignar un valor del dominio (nulo inaplicable)

Ejemplo: *cant* es de tipo Entero siempre positivo

***cedIdent* es de tipo Entero > 0 y no puede ser nulo**

Tipos de restricciones de integridad

2. **Restricciones de rango o integridad de columna:**
   Se refiere al intervalo de variación de los valores del dominio del atributo y de los tipos de datos definidos en el SMBD
   Ejemplo: *edad* es de tipo Entero siempre positivo entre 0 y 120
3. **Integridad de entidad o de dependencias funcionales:**
   Se refiere al hecho de tener un atributo que está determinado por uno o varios atributos
   Estas restricciones están aseguradas con la normalización de las tablas de la BD
   Ningún componente de una clave primaria puede contener valores nulos
   Ningún componente de una clave foránea debe permitir un valor nulo por inaplicable, aunque si puede permitir valor nulo por desconocimiento de información
   Ejemplo: cedula determina edad

Tipos de restricciones de integridad

4. **Dependencias multi-valuadas:**
   - Son aquellas donde uno o varios atributos multi-determinan un atributo
   - Estas están aseguradas con la normalización de las tablas de la BD
   - Ejemplo: cedulaEstudiante multi-determina deportePractica
5. **Integridad referencial:**
   - Son las dependencias de inclusión en varias tablas o de claves foráneas
   - Para cada clave foránea debe existir un valor equivalente de una clave primaria y en el mismo dominio
   - Ejemplo: Se tienen las tablas *Carro(placa, modelo, color)* y *ModeloMarca(modelo, marca)*, en ellas se observa que el atributo modelo es clave en la tabla ModeloMarca y está incluida en la tabla Carro, por tanto el atributo modelo es una clave foránea en la relación Carro

Integridad referencial

Producto(nroPro, nombrePro, cantidad, color)

Venta(nroVen, fechaVen, nombreCli, *nroProVen*, cantVen)

nroProVen es foránea en Ventas y refiere a nroPro de Producto, por lo tanto

$\Pi_{\text{nroProVen}}$ **(Ventas)** $\subseteq \Pi_{\text{nroPro}}$ **(Producto)**

También se puede expresar como

$\Pi_{\text{nroProVen}}$ **(Ventas) -** Π_{nroPro} **(Producto)** $= \varnothing$

Tipos de restricciones de integridad

6. **Restricciones aritméticas:**

Son las expresiones aritméticas que deben cumplir algunos atributos de una tabla o que involucra a varias tablas de la BD

Ejemplo: En la BD formada por las tablas siguientes:

Producto(codPro, nomPro, cantExistencia, color)

Venta(codVen, nomCli, *codProVen*, cantVen, fechaVen)

Compra(codCom, fechaCom, *codProCom*, cantCom, nomProveedor)

- Para todo producto identificado con su código codPro de la tabla Producto, la cantExistencia debe ser mayor que la cantidad vendida cantVen para el producto codProVen, ya que no se puede vender una cantidad de producto mayor que la que se tiene en existencia

Tipos de restricciones de integridad

7. **Valores invariantes que no son posibles de expresar en el esquema:**

   Ejemplo:

   Tomando la BD descrita anteriormente, se tiene que en todo momento la cantidad comprada menos la cantidad vendida debe ser igual a la cantidad en existencia (cantCom - cantVen = cantExistencia), para cada producto presente en la BD

8. **Restricciones temporales:**

   Son aquellas aserciones que deben ser cumplidas periódicamente o en momentos específicos

   Ejemplo:

   En una BD de transacciones bancarias al finalizar cada mes, el saldo de cada cuenta debe ser igual a la suma de depósitos en la cuenta menos la suma de los retiros de la cuenta

UNIVERSIDAD DE LOS ANDES

Bases de Datos Activas

- **Las BD convencionales son pasivas, solo ejecutan consultas a petición de un usuario o de un programa de aplicación**
- **Una BD activa es una BD que tiene la capacidad de monitorear situaciones de interés y cuando ellas ocurren disparan una respuesta adecuada**
- **El comportamiento deseado se expresa por medio de reglas de producción o reglas evento-condición-acción (ECA), las cuales pueden ser definidas y almacenadas en la BD**

on evento	**Una regla se *dispara* cuando el evento ocurre**
if condición	**se *considera* cuando su condición se evalúa**
then acción	**y se *efectúa* cuando la acción se ejecuta**

- **Las reglas serán disparadas por eventos de la BD como: un estado particular de la BD y cambios de estado de la misma**

UNIVERSIDAD DE LOS ANDES

Reglas

- **Ellas pueden:**
  - asegurar la integridad de datos,
  - implementar gatillos y alertadores,
  - mantener datos derivados,
  - asegurar las restricciones de acceso,
  - implementar las políticas de control de versiones,
  - mantener las estadísticas de acceso para la optimización de consultas,
  - etc.

UNIVERSIDAD DE LOS ANDES

Gatillos

- **Gatillo: pieza de código que espera que un evento ocurra (inserción, eliminación o modificación) y cuando ocurre se dispara una secuencia dada de acciones**
- **En los SMBDR como DB2, Sybase, PostgreSQL y Oracle, las reglas se definen como metadatos en el esquema a través de los gatillos**
- **Cada gatillo tiene un mecanismo que la asocia a las tuplas insertadas, eliminadas o modificadas que referencia en sus condiciones y acciones**

```
create trigger DptoEliminación
before delete on departamento
when departamento.presupuesto < 100.000.oo
delete empleado where empleado.dptoNro = departamento.dptoNro
```

PostgreSQL

- **El procesamiento de los gatillos se invoca inmediatamente que ha ocurrido la modificación (procesamiento orientado a la tupla), es recursivo y sincrónico**
- **El algoritmo de procesamiento es:**
  1. Algún usuario o gatillo modifica una tupla
  2. La modificación dispara los gatillos G1, G2, ..., Gn
  3. Para cada gatillo Gi ejecutar la acción de Gi
- **No hay mecanismo de resolución de conflictos**
- **Los gatillos se ejecutan en un orden arbitrario**

- **A partir de MySQL 5.0.2 se incorporó el soporte básico para disparadores (triggers)**

```
mysql> CREATE TRIGGER ins_sum BEFORE INSERT ON account
    -> FOR EACH ROW SET @sum = @sum + NEW.amount;
```

- **MySQL gestiona los errores ocurridos durante la ejecución de disparadores de esta manera:**
  - Si lo que falla es un disparador BEFORE, no se ejecuta la operación en el correspondiente registro
  - Un disparador AFTER se ejecuta solamente si el disparador BEFORE (de existir) y la operación se ejecutaron exitosamente
  - Un error durante la ejecución de un disparador BEFORE o AFTER deriva en la falla de toda la sentencia que provocó la invocación del disparador
  - En tablas transaccionales, la falla de un disparador (y por lo tanto de toda la sentencia) debería causar la cancelación (*rollback*) de todos los cambios realizados por esa sentencia
  - En tablas no transaccionales, cualquier cambio realizado antes del error no se ve afectado

Según el manual en http://dev.mysql.com/doc/refman/5.0/es/using-triggers.html

Oracle

- **Granularidad: a nivel de tupla (orientada a la instancia) o de instrucción (al conjunto)**
- **El gatillo se considera y se efectúa antes y después de la operación disparable**
- **Combinaciones:**
  - tupla-gatillo-antes
  - instrucción-gatillo-antes
  - tupla-gatillo-después
  - instrucción-gatillo-después

Algoritmo de procesamiento de los gatillos:

1. Ejecute los gatillos instrucción-gatillo-antes
2. Para cada tupla de la tabla
   a. Ejecute los gatillos tupla-gatillo-antes
   b. Efectúe las modificaciones de la tupla y las tuplas asociadas por la integridad referencial verificando las aserciones
   c. Ejecute los gatillos tupla-gatillo-después
3. Efectúe las modificaciones asociadas a la instrucción por integridad referencial verificando las aserciones
4. Ejecute los gatillos instrucción-gatillo-después

UNIVERSIDAD DE LOS ANDES

Oracle

- El número máximo de gatillos en cascada es 32
- Cuando el sistema alcanza este número, que indica la posibilidad de no finalización, se suspende la ejecución y se da una condición de excepción
- Si una excepción es alcanzada u ocurre un error, entonces todos los cambios efectuados en la BD, por la instrucción SQL original y todas las acciones disparadas subsecuentes son deshechas

UNIVERSIDAD DE LOS ANDES

DB2

- **Manejo de los gatillos parecido a Oracle**
- **Varios gatillos pueden monitorear el mismo evento, en base al orden total impuesto por DB2, que toma en cuenta el tiempo de creación del gatillo**
- **Si una instrucción de modificación S en la acción A causa el evento E, entonces:**

Algoritmo de procesamiento de los gatillos:

1. Suspender A y guardar su estado en el área de trabajo en la pila
2. Calcular los valores de transición (OLD, NEW) relativos al evento E
3. Considerar y ejecutar todos los gatillos-antes relativos a E, posiblemente cambiando los valores de transición NEW
4. Aplicar los valores de transición NEW a la BD haciendo el cambio de estado de E

5. Considerar y ejecutar todos los gatillos-después relativos a E. Si cualquiera de sus acciones Ai activa otros gatillos, iniciar este proceso recursivo para Ai
6. Sacar de la pila el área de trabajo de A y continuar su evaluación

- **El paso 4 del procedimiento anterior se modifica si se viola alguna restricción en SQL-92**

4. Aplicar los valores de transición NEW a la BD haciendo el cambio de estado de E. Para cada restricción de integridad violada, considerar la acción Aj que la compensa

a. Calcular los valores de transición (OLD, NEW) relativos a Aj
b. Ejecute los gatillos-antes relativos a Aj cambiando posiblemente los valores de transición NEW
c. Aplicar los valores de transición NEW a la BD haciendo el cambio de estado asociado a Aj
d. Meter en la cola de los gatillos suspendidos todos los gatillos-después relativos a Aj

UNIVERSIDAD DE LOS ANDES

Autoevaluación

1. ¿Qué se entiende por el enfoque por descomposición en el modelo relacional?
2. ¿Cuáles son las operaciones fundamentales de la teoría de la descomposición?
3. ¿Qué es una descomposición sin pérdida?
4. ¿Cómo se definen las cinco formas normales?
5. ¿Qué son dependencias funcionales, multi-valuadas y las de producto y cuál es la diferencia entre ellas?
6. ¿Qué se entiende por restricciones de integridad y cuáles son sus tipos?
7. ¿Qué son BD activas, qué se entiende por reglas y cómo se soportan en los SGBDR?

UNIVERSIDAD DE LOS ANDES

Ejercicios

Para los resultados obtenidos en los ejercicios de la clase 8 realice:

1. La verificación que el modelo relacional obtenido está en 3FN, encontrando las DFE, construyendo el cierre transitivo y corriendo el algoritmo de descomposición en 3FN
2. La verificación de las DME y las DP, si ellas están presentes
3. Clasifique todas las restricciones de integridad por tipo
4. Proponga al menos 2 gatillos para cada modelo relacional en forma descriptiva (no en SQL)