Unidad 4: Buenas prácticas

BBDD01, Sesión 11:

Restricciones de Integridad

Formalización: Dependencias Funcionales

INDICE

- Integridad Referencial
  - Introducción.
  - Restricciones de los dominios.
  - Integridad Referencial.
  - Asertos.
  - Disparadores.
  - Seguridad y autorización.
  - Autorización en SQL.
  - Cifrado y autentificación.
- Normalización: Dependencias Funcionales
  - Introducción.
  - Dificultades en el diseño de bases de datos relacionales.
  - Dependencias funcionales.

Referencias: Silberschatz 4ª Ed. pp 140-189
Elmasri 3ª Ed. Pp 439-496

Introducción

- Restricciones de integridad ⇒ modificaciones hechas por los usuarios autorizados mantengan la consistencia de los datos.
- Dos tipos se han visto:
  - Declaración de claves.
  - Tipo de relación.
- Nuevos tipos
- Pueden sobrecargar la base de datos.

Restricciones de Dominio

- Cada atributo se asocia a un dominio ⇒ restricción de los valores.
- Son las restricciones más simples.
- Caracteres, enteros, fechas/tiempo
- El sistema de base de datos lo verifica cada vez que se produce una modificación en la base de datos.
- Equivalente a los tipos de variables en los lenguajes de programación.
- Create domain ⇒ crea nuevos dominios

create domain *Euros* **numeric**(12,2)
create domain *Dólares* **numeric**(12,2)

- Valor de tipo Euro a tipo Dólar daría error.

Restricciones de Dominio

- Se pueden convertir valores de un dominio a otro.

cast *r.A* **as** *Dólares*

- Drop domain y alter domain ⇒ borrar y modificar dominio
- Check ⇒ permite restringir un dominio más poderosamente.
- Asignar un predicado que se debe de cumplir para cualquier valor perteneciente al dominio

create domain *sueldo-por-hora* **numeric**(5,2)
constraint *comprobación-valor-sueldo*
check(**value** ≥ 4.00)

- Nombre de la restricción (constraint) es opcional

Restricciones de Dominio

- Se puede decir que no contenga nulos

```
create domain número-cuenta char(10)
    constraint comprobación-número—cuenta-nulo
        check(value not null)
```

- Conjunto concreto de valores

```
create domain tipo-cuenta char(10)
    constraint comprobación-tipo-cuenta
        check(value in ('Corriente', 'Ahorro'))
```

- Se podrían introducir subconsultas (pueden ser costosas)

```
check (nombre-sucursal in
    (select nombre-sucursal from sucursal))
```

Integridad Referencial

- Asegurar que un valor que aparece en una relación para un conjunto de atributos aparezca en otra relación ⇒ Integridad Referencial
- Conceptos básicos
- Tuplas colgantes ⇒ aquellas que no se reunen con otra tupla de otra relación.
- Pueden ser deseables o nó ⇒ Integridad referencial para permitirlas o no.
- Depende de si hay clave externa o no.
- Sean $r_1(R_1)$ y $r_2(R_2)$ con PK K_1 y K_2, se dice que un subconjunto α de R_2 es una clave externa que hace referencia a K_1 si se exige que para cada tupla t_2 haya una t_1 tal que $t_1[K_1]=t_2[\alpha]$ ⇒ Restricción de integridad (dependencia de subconjunto)

Integridad Referencial

- α generalmente es K_1
- Si el esquema viene obtenido del modelo E-R:
  - Cada relación que proceda de un conjunto de relaciones tendrá integridad referencial en la calve externa (ajena)
  - Conjuntos de entidades débiles también.
- Modificación de la base de datos
- Puede ocasionar violaciones en la integridad referencial

$$\Pi_{\alpha}(r_2) \subseteq \Pi_K(r_1)$$

- Insertar $\Rightarrow$ si se inserta una tupla t_2, el sistema debe de asegurar que hay otra tupla t_1 que cumple:

$$t_1[K] = t_2[\alpha]$$

$$t_2[\alpha] \in \Pi_K(r_1)$$

Integridad Referencial

- Borrar ⇒ Si se borra t_1 del sistema, se debe de calcular el número de tuplas de r_2 que referencian a r_1.

$$\sigma_{\alpha = t_1[K]}(r_2)$$

  - Si el resultado no es un conjunto vacío:
    - Se rechaza la orden de borrado
    - Se borran todas las tuplas que referencian a t_1 (borrado cascada)
- Actualizar ⇒ Dos casos
  - Actualización sobre la clave externa de r_2 ⇒ comprobación parecida a la inserción. Hay que asegurar:

$$t'_2[\alpha] \in \Pi_K(r_1)$$

  - Actualización sobre la primary key de r_1 ⇒ comprobación parecida a la del borrado. Hay que asegurar:

$$\sigma_{\alpha = t_1[K]}(r_2)$$

  - La actualización se realiza o se produce una actualización en cascada

Integridad Referencial

- Integridad referencial en SQL
- Las claves externas en SQL ⇒ foreign key en el create table
- También cláusula references

nombre-sucursal **char**(15) **references** *sucursal*

```
create table impositor
    (nombre-cliente      char(20),
    número-cuenta        char(10),
    primary key (nombre-cliente, número-cuenta),
    foreign key (nombre-cliente) references cliente,
    foreign key (número-cuenta) references cuenta)
```

- En caso de violación RI ⇒ rechazar la acción.
- Con foreign key se pueden tomar medidas

Integridad Referencial

```
create table cuenta
    (...
    foreign key (nombre-sucursal) references sucursal
                                  on delete cascade
                                  on update cascade,
    ...)
```

- Se puede:
  - Actualizar/borrar en cascada (cascade)
  - Establecer en nulo
  - Poner un valor por defecto (set default)
  - Prohibir la acción (defecto)
- Puede haber propagaciones en cascada por diferentes tablas
- Las claves externas pueden ser no nulos
- Mejor definir las claves externas como nulos

Asertos

- Es un predicado que expresa una condición que tiene que cumplir la base de datos.
- Restricciones de dominio y RI son asertos especiales.
- Hay restricciones que no se pueden definir con los anteriores
- En SQL: **create assertion** <nombre-aserto> **check** <predicado>

```
create assertion restricción-suma check
    (not exists (select * from sucursal
        where (select sum(importe) from préstamo
            where préstamo.nombre-sucursal
                = sucursal.nombre-sucursal)
        >= (select sum (importe) from cuenta
            where préstamo.nombre-sucursal
                = sucursal.nombre-sucursal)))
```

- Puede producir gran sobrecarga en la base de datos

Disparadores

- Es una orden que el sistema ejecuta como efecto secundario de la modificación sobre la base de datos.
- Dos requisitos:
  - Las condiciones del disparador ⇒ evento y condición.
  - Las acciones a realizar por el disparador.
- Modelo: evento-condición-acción.
- Se almacenan en la base de datos ⇒ persistentes
- Necesidad de disparadores:
  - En un banco cuando el saldo de una cuenta es negativo
  - En un almacén cuando el stock baja una cierta cantidad

Disparadores

- Se usan ampliamente
- Hasta SQL-99 no formaron parte de la norma.
- Pero cada sistema tiene su propia sintaxis.

```
create trigger descubierto after update on cuenta
referencing new row as nfila
for each row
when nfila.saldo < 0
begin atomic
      insert into prestatario
          (select nombre-cliente, número-cuenta
          from impositor
          where nfila.número-cuenta = impositor.número-cuenta);
      insert into préstamo values
         (nfila.número-cuenta, nfila.nombre-sucursal, – nfila.saldo)
      update cuenta set saldo = 0
          where cuenta.número-cuenta = nfila.número-cuenta
end
```

Disparadores

- El evento disparador:
  - INSERT, DELETE
  - UPDATE: se pueden realizar sobre un campo

create trigger *descubierto* **after update of** *saldo* **on** *cuenta*

- Referencing old row as ⇒ valor de la fila actualizada o borrada
- Referencing new row as ⇒ valor nuevo de la fila a actualizar o insertar
- Activación:
  - AFTER: después del evento
  - BEFORE: antes del evento. Se pueden usar como restricciones extras

create trigger *poner-nulo* **before update on** *r*
referencing new row as *nfila*
for each row
when *nfila.número-teléfono* = ' '
set *nfila.número-teléfono* = **null**

Disparadores

- En vez de cada fila, se puede hacer una vez por cada instrucción completa que provocó el disparador ⇒ for each statement

```
create trigger nuevopedido after update of cantidad on inventario
referencing old row as ofila, new row as nfila
for each row
when nfila.nivel <= (select nivel
                     from nivelmínimo
                     where nivelmínimo.producto = ofila.producto)
and ofila.nivel <= (select nivel
                    from nivelmínimo
                    where nivelmínimo.producto = ofila.producto)
begin
    insert into pedidos
        (select producto, cantidad
         from nuevopedido
         where nuevopedido.producto= ofila.producto);
end
```

Disparadores

- Hay que escribirlos con mucho cuidado
- Puede provocar el fallo de una modificación de la base de datos y desencadenar una secuencia de disparadores infinita ⇒ se suelen limitar las secuencias consecutivas
- Antiguamente:
  - se usaban para hacer copias o replicación.
  - Para hacer resúmenes de una base de datos. Hoy con vistas materializadas es más fácil

Seguridad y Autorización

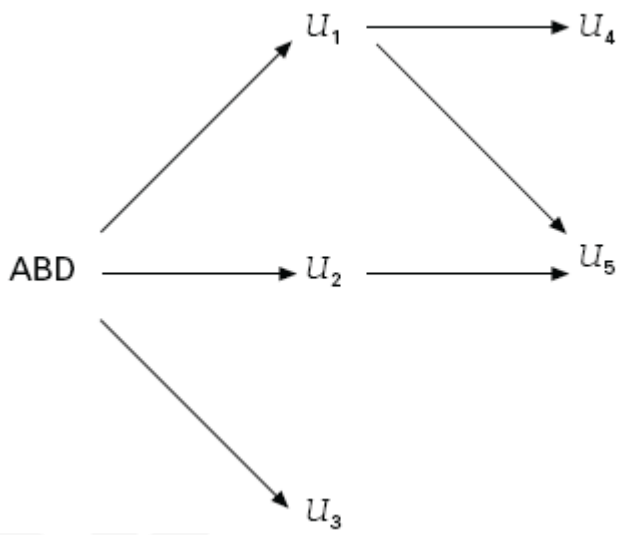
- Los datos deben estar protegidos contra accesos no autorizados que pueden producir :
  - Destrucción
  - Alteraciones malintencionadas
  - Inconsistencias de los datos
- Violaciones de seguridad
  - Lectura no autorizada
  - Modificación no autorizada
  - Destrucción no autorizada
- Protección frente a usuarios no autorizados:
  - Sistema Gestor de Base de Datos
  - Sistema Operativo
  - Conexiones de Red
  - Sitios protegidos
  - Personas con autorizaciones

Seguridad y Autorización

- Autorizaciones
- Varios tipos sobre datos:
  - Lectura
  - Inserción
  - Actualización
  - Borrado
- Varios tipos sobre el esquema:
  - Indices
  - Recursos: nuevas relaciones
  - Alteración: nuevos atributos a las relaciones
  - Eliminación de las relaciones
  - Vistas: como una forma de ocultar detalles de la base de datos a ciertos usuarios.

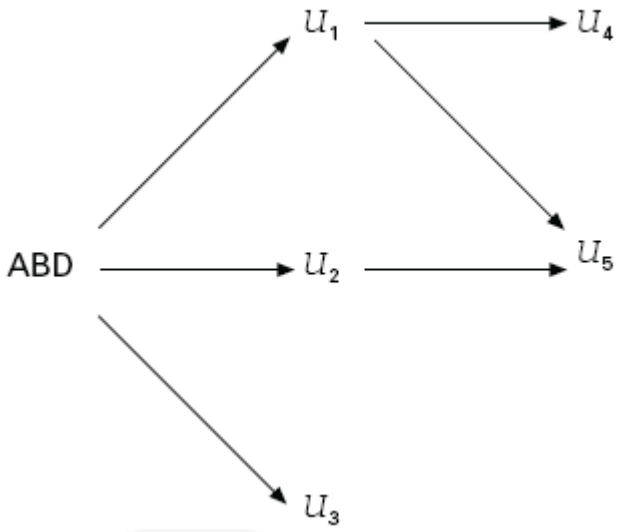
Seguridad y Autorización

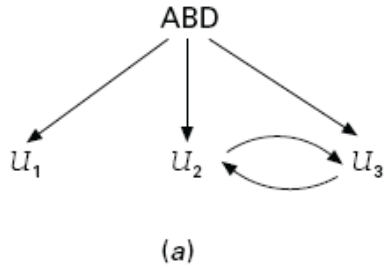
- Concesión de privilegios
- Un usuario que tiene concedido autorizaciones puede transmitir esas autorizaciones a otros usuarios.
- Grafo de autorización:



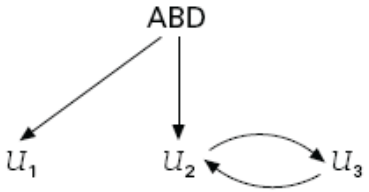
Seguridad y Autorización

- Intento de eludir autorización

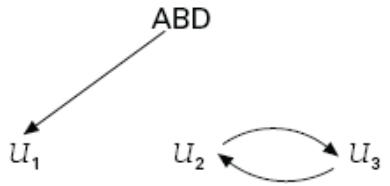




(a)



(b)



(c)

Seguridad y Autorización

- Concepto de papel (Role)
- Hay usuarios que tienen los mismos privilegios y autorizaciones.
- En vez de asignarlo a cada usuario que se crea del mismo tipo ⇒ mejor crear un esquema de autorización para conjuntos de usuarios del mismo tipo ⇒ papel
- Asignar el papel de ese usuario en la base de datos.
- Asignar un identificador propio para cada usuario
- Trazas de auditorías
- Registro histórico donde se guarda todos los cambios que se han producido junto con el usuario que lo ha realizado y cuando.

Autorización en SQL

- Privilegios en SQL
- Delete, insert, update y select
- Grant ⇒ conceder privilegios

grant <lista de privilegios> **on** <nombre de relación o de lista> **to** <lista de usuarios/papeles>

- Con update y insert se pueden establecer a campos concretos

grant select on *sucursal* **to** U_1, U_2, U_3

- También references ⇒ declarar claves externas en relaciones

grant update (*importe*) **on** *préstamo* **to** U_1, U_2, U_3

- All privileges ⇒ todos los privilegios son concedidos
- Nombre usuario public ⇒ referencia a todos los usuarios y los que puedan venir.

grant references (*nombre-sucursal*) **on** *sucursal* **to** U_1

Autorización en SQL

- Papeles
- Crear: **create role** *cajero*
- Conceder privilegios: **grant select on** *cuenta* **to** *cajero*
- Asignar papeles a otros usuarios o papeles:

grant *cajero* **to** juan
create role *gestor*
grant *cajero* **to** *gestor*
grant *gestor* **to** maría

- Privilegios de un usuario o papel constan:
  - Privilegios concedidos al usuario o papel
  - Privilegios concedidos a papeles que se hayan concedido al papel o usuario

Autorización en SQL

- Privilegio de conceder privilegios
- Un usuario no está predeterminado a conceder un privilegio que se le ha concedido a otro usuario.
- Si se desea ⇒ with grant option

grant select on *sucursal* **to** U_1 **with grant option**

- Retirar autorización ⇒ revoke

revoke <lista de privilegios> **on** <nombre de relación o de vista>
from <lista de usuarios o papeles> [**restrict** | **cascade**]

revoke select on *sucursal* **from** U_1, U_2, U_3
revoke update (*importe*) **on** *préstamo* **from** U_1, U_2, U_3
revoke references (*nombre-sucursal*) **on** *sucursal* **from** U_1

revoke grant option for select on *sucursal* **from** U_1

Autorización en SQL

- El creador de un objeto ⇒ obtiene los privilegios de ese objeto y además puede concederlos a otros.
- Sólo el propietario del esquema puede ejecutar cualquier modificación del mismo

Cifrado y Autenticación.

- Proteger datos extremadamente delicados ⇒ cifrado
- Técnicas de cifrado:
  - Sencillas
  - Complicadas:
    - DES: Data Encription Standard: Sustitución y ordenación de los caracteres en base a una clave de cifrado
    - AES: Advanced Encription Stantdard
    - Cifrado de clave asimétrica: clave pública y clave privada
- Autenticación ⇒ tarea de verificar la identidad de una persona o software que se conecte a la base de datos.
  - Usuario – contraseña
  - Sistemas desafío-respuesta + cifrado con clave pública
  - Firmas digitales

Introducción

- Estudio de los problemas de bases de datos relacionales.
- Objetivo :
  - generación de un conjunto de esquemas relacionales que no almacenen redundancia.
  - Permitir la recuperación de información fácilmente.
- Diseño de esquemas que se hallen en una forma normal adecuada ⇒ normalización.
- Hace falta información adicional de la lógica de negocio que se está almacenando.
- Dependencias funcionales
- Formas normales en función de las dependencias funcionales y otros tipos de dependencias.

Dificultades en el diseño de BD relacionales

- Mal diseño de BD:
  - Repetición de la información.
  - Imposible representar determinada información.
- Ejemplo:

Esquema-empréstito = (nombre-sucursal, ciudad-sucursal, activo, nombre-cliente, número-préstamo, importe)

nombre-sucursal	*ciudad-sucursal*	*activo*	*nombre-cliente*	*número-préstamo*	*importe*
Centro	Arganzuela	9.000.000	Santos	P-17	1.000
Moralzarzal	La Granja	2.100.000	Gómez	P-23	2.000
Navacerrada	Aluche	1.700.000	López	P-15	1.500
Centro	Arganzuela	9.000.000	Sotoca	P-14	1.500
Becerril	Aluche	400.000	Santos	P-93	500
Collado Mediano	Aluche	8.000.000	Abril	P-11	900
Navas de la Asunción	Alcalá de Henares	300.000	Valdivieso	P-29	1.200
Segovia	Cerceda	3.700.000	López	P-16	1.300
Centro	Arganzuela	9.000.000	González	P-18	2.000
Navacerrada	Aluche	1.700.000	Rodríguez	P-25	2.500
Galapagar	Arganzuela	7.100.000	Amo	P-10	2.200

- Se desea añadir un nuevo préstamo

Dificultades en el diseño de BD relacionales

- Repetir información: activo y sucursal.
- Ocupa más espacio en disco.
- Complica actualización de la base de datos ⇒ si cambia el activo, hay que modificarlo en todas las tuplas.
- Este diseño es malo: cada sucursal identifica un activo.
- Dado una sucursal no se puede identificar a un préstamo.
- Se cumple la dependencia funcional: $nombre\text{-}sucursal \rightarrow activo$

- Es independiente el hecho de tener activo y de conceder préstamos ⇒ relaciones independientes.
- Para introducir una sucursal tiene que haber un préstamo concedido. Si no, introducir valores nulos

Dependencias Funcionales

- Papel importante entre buenos y malos diseños.
- Tipo de restricción ⇒ generalización del concepto clave
- Representa hechos de la lógica de negocio real.
- Superclave ⇒ Subconjunto K de R donde r(R) donde t_1 y t_2 de r se cumple que si $t_1 \neq t_2$ entonces $t_1[K] \neq t_2[K]$
- Sea $\alpha \subseteq R$ y $\beta \subseteq R$., se define la dependencia funcional $\alpha \rightarrow \beta$
- En cualquier relación r(R) , para los pares de tuplas t_1 y t_2 de r tales que $t_1[\alpha]=t_2[\alpha]$, se cumple $t_1[\beta] = t_2[\beta]$.
- K es una superclave de R si K ⟶ R. Es decir, siempre que $t_1[K]=t_2[K]$, también se produce que $t_1[R]=t_2[R]$. ($t_1=t_2$)
- Permiten expresar restricciones que no se pueden expresar con las superclaves.

Dependencias Funcionales

- Ejemplo:

Esquema-info-préstamo = (*número-préstamo*, *nombre-sucursal*, *nombre-cliente*, *importe*)

- Se necesita cumplir:

número-préstamo → *importe*
número-préstamo → *nombre-sucursal*

- Pero no:

número-préstamo → *nombre-cliente*

- Se utilizan para:
  - Para probar las relaciones y ver si son legales. Conjunto F de DF. r satisface F
  - Para especificar las restricciones del conjunto de relaciones legales. F se cumple en R.

Dependencias Funcionales

- Ejemplo:

A	B	C	D
a_1	b_1	c_1	d_1
a_1	b_2	c_1	d_2
a_2	b_2	c_2	d_2
a_2	b_3	c_2	d_3
a_3	b_3	c_2	d_4

- Se cumple: A $\rightarrow$ C (dos tuplas de a_1 tienen c_1)
- No se cumple C $\rightarrow$ A (dos tuplas que para C no tiene el mismo valor de A)
- Se cumple: AB$\rightarrow$D (todos valores diferentes. Sólo una tupla)
- Triviales: A$\rightarrow$A
- Dependencia trivial $\alpha \rightarrow \beta$, si $\beta \subseteq \alpha$

Dependencias Funcionales

- Ejemplo: relación cliente

nombre-cliente	*calle-cliente*	*ciudad-cliente*
Santos	Mayor	Peguerinos
Gómez	Carretas	Cerceda
López	Mayor	Peguerinos
Pérez	Carretas	Cerceda
Rupérez	Ramblas	León
Abril	Preciados	Valsaín
Valdivieso	Goya	Vigo
Fernández	Jazmín	León
González	Arenal	La Granja
Rodríguez	Yeserías	Cádiz
Amo	Embajadores	Arganzuela
Badorrey	Delicias	Valsaín

- Calle-cliente→ciudad-cliente ?. No se cumple generalmente

Dependencias Funcionales

- Ejemplo: relación préstamo

número-préstamo	*nombre-sucursal*	*importe*
P-17	Centro	1.000
P-23	Moralzarzal	2.000
P-15	Navacerrada	1.500
P-14	Centro	1.500
P-93	Becerril	500
P-11	Collado Mediano	900
P-29	Navas de la Asunción	1.200
P-16	Segovia	1.300
P-18	Centro	2.000
P-25	Navacerrada	2.500
P-10	Galapagar	2.200

- Numero-prestamo→importe. Se cumple y se debe de exigir ya que un número de préstamo tiene un importe asociado.

Dependencias Funcionales

- Ejemplo: relación sucursal

nombre-sucursal	*ciudad-sucursal*	*activo*
Centro	Arganzuela	9.000.000
Moralzarzal	La Granja	2.100.000
Navacerrada	Aluche	1.700.000
Becerril	Aluche	400.000
Collado Mediano	Aluche	8.000.000
Navas de la Asunción	Alcalá de Henares	300.000
Segovia	Cerceda	3.700.000
Galapagar	Arganzuela	7.100.000

- Nombre-sucursal→activo. Se cumple y se debe de exigir. Al revés no es cierto.

Dependencias Funcionales

- Cierre de un conjunto de DF.
- Hay que considerar todas las DF: las definidas y las implícitas
- Una dependencia funcional f de R está implicada lógicamente por un conjunto de DF F de R, si cada ejemplar r(R) que satisface F, satisface f también.
- Ejemplo: R(A,B,C,G,H,I) y el conjunto de DF

$$A \rightarrow B$$
$$A \rightarrow C$$
$$CG \rightarrow H$$
$$CG \rightarrow I$$
$$B \rightarrow H$$

- La DF A⟶H está implicada lógicamente.

Dependencias Funcionales

- Cierre de F $\Rightarrow$ F^+ , conjunto de todas las DF implicadas lógicamente en F.
- Los axiomas o reglas de inferencia se pueden aplicar repetidamente para hallar F^+ (axiomas de Armstrong)

- **Regla de la reflexividad.** Si α es un conjunto de atributos y $\beta \subseteq \alpha$, entonces se cumple que $\alpha \rightarrow \beta$.
- **Regla de la aumentatividad.** Si se cumple que $\alpha \rightarrow \beta$ y γ es un conjunto de atributos, entonces se cumple que $\gamma\alpha \rightarrow \gamma\beta$.
- **Regla de la transitividad.** Si se cumple que $\alpha \rightarrow \beta$ y también se cumple que $\beta \rightarrow \gamma$, entonces se cumple que $\alpha \rightarrow \gamma$.

- Son correctos $\Rightarrow$ generan DF correctas.
- Son completos $\Rightarrow$ permiten generar todo F^+

Dependencias Funcionales

- Para simplificar, se puede emplear:

- **Regla de la unión.** Si se cumple que $\alpha \rightarrow \beta$ y que $\alpha \rightarrow \gamma$, entonces se cumple que $\alpha \rightarrow \beta\gamma$.
- **Regla de la descomposición.** Si se cumple que $\alpha \rightarrow \beta\gamma$, entonces se cumple que $\alpha \rightarrow \beta$ y que $\alpha \rightarrow \gamma$.
- **Regla de la pseudotransitividad.** Si se cumple que $\alpha \rightarrow \beta$ y que $\gamma\beta \rightarrow \delta$, entonces se cumple que $\alpha\gamma \rightarrow \delta$.

- Ejemplo: R=(A,B,C,G,H,I) y F

$$\{A \rightarrow B, A \rightarrow C, CG \rightarrow H, CG \rightarrow I, B \rightarrow H\}$$

- A⟶H ,dado que A⟶B y B⟶H (Transitividad)
- CG⟶HI ,dado que CG⟶H y CG⟶I (Unión)
- AG⟶I , dado que A⟶C y CG⟶I (pseoudotransitividad)

Dependencias Funcionales

- Cierre de un conjunto de atributos
- Sea α conjunto de atributos
- Conjunto de todos los atributos determinados funcionalmente por α bajo un conjunto F de DF$\Rightarrow$ cierre de α bajo F (α^+)
- Si α es superclave debe implicar todas las DF de F
- Algoritmo:

```
resultado := α;
while (cambios en resultado) do
        for each dependencia funcional β → γ in F do
          begin
              if β ⊆ resultado then resultado := resultado ∪ γ;
          end
```

- Si α^+ contiene todos los atributos de R $\Rightarrow$ es superclave

Dependencias Funcionales

- Ejemplo: Calcular $(AG)^+$ con las DF $\{A \rightarrow B, A \rightarrow C, CG \rightarrow H, CG \rightarrow I, B \rightarrow H\}$
- Primero se añade AG
- A⟶B , B se añade al resultado: ABG
- A⟶C , C se añade a resultado: ABCG
- CG⟶H, H se añade: ABCGH
- CG⟶I, I se añade: ABCGHI

- Usos del algoritmo:
  - Comprobar si α es superclave.
  - Comprobar si se cumple DF $\alpha \rightarrow \beta$, comprobando si $\beta \subseteq \alpha^+$
  - Otra manera de calcular F^+

Dependencias Funcionales

- Recubrimiento canónico
- Si se produce una modificación en la BD, se debe asegurar que la modificación no viole ninguna DF de F.
- Se puede reducir el esfuerzo comprobando un conjunto de DF simplificado que tenga el mismo cierre que el conjunto dado.
- Atributo raro de una DF $\Rightarrow$ si se puede eliminar sin modificar el cierre de F. Considerar $\alpha \rightarrow \beta$
  - A es raro en α, si $A \in \alpha$ y F implica lógicamente a $(F-\{\alpha \rightarrow \beta\}) \cup \{(\alpha-A) \rightarrow \beta\}$
  - A es raro en β, si $A \in \beta$ y el conjunto de DF $(F-\{\alpha \rightarrow \beta\}) \cup \{\alpha \rightarrow (\beta-A)\}$ implica lógicamente a F
- Ejemplo: $AB \rightarrow C$ y $A \rightarrow C$, entonces B es raro en $AB \rightarrow C$
- $AB \rightarrow CD$ y $A \rightarrow C$, C es raro en el lado derecho de $AB \rightarrow CD$

Dependencias Funcionales

- Recubrimiento canónico
- Para comprobar eficientemente si un atributo A es raro en la DF $\alpha \rightarrow \beta$:
  - Si $A \in \beta$, hay que considerar el conjunto

    F'= (F-{$\alpha \rightarrow \beta$}) U {($\alpha \rightarrow$(β-A)} y comprobar si $\alpha \rightarrow$A puede inferirse a partir de F'. Se calcula α+ bajo F' y si incluye A, entonces A es raro en β
  - Si $A \in \alpha$, sea $\gamma = \alpha$-{A} hay que comprobar que si $\gamma \rightarrow \beta$ puede inferirse a partir de F. Se calcula γ+ bajo F y si incluye todos atributos de β, entonces A es raro en α
- Ejemplo: F contiene AB$\rightarrow$CD, A$\rightarrow$E y E$\rightarrow$C. Para comprobar si C es raro en AB$\rightarrow$CD hay que calcular el cierre de AB bajo F'={AB$\rightarrow$D, A$\rightarrow$E y E$\rightarrow$C}. El cierre es ABCDE incluye a CD por lo que C es raro.

Dependencias Funcionales

- Recubrimiento canónico F_c de F es un conjunto de las dependencias tales que F implica lógicamente todas las dependencias de F_c y F_c implica todas las dependencias de F.
- Posee dos propiedades:
  - Ninguna dependencia funcional F_c contiene atributos raros.
  - Lado izquierdo de cada DF de F_c es único. No hay dos DF $\alpha_1 \rightarrow \beta_1$ y $\alpha_2 \rightarrow \beta_2$ de F_c tales que $\alpha_1 = \alpha_2$.
- Algoritmo

Fc = *F*
repeat
  Utilizar la regla de unión para sustituir las dependencias de F_c de la forma
    $\alpha_1 \rightarrow \beta_1$ y $\alpha_1 \rightarrow \beta_2$ con $\alpha_1 \rightarrow \beta_1\ \beta_2$.
  Hallar una dependencia funcional $\alpha \rightarrow \beta$ de F_c con un atributo raro en α o en β.
    /* Nota: la comprobación de los atributos raros se lleva a cabo empleando F_c, no *F* */
  Si se halla algún atributo raro, hay que eliminarlo de $\alpha \rightarrow \beta$.
until F_c ya no cambie.

Recordando: Dependencias Funcionales

- Ejemplo: Dado el conjunto de DF para el esquema (A,B,C)

  A→BC , B→C , A→ B y AB→C

- Calcular el recubrimiento canónico de F.
- Hay dos DF con el mismo conjunto de atributos a la izquierda:

  A→BC

  A→ B

- Se transforman en A→ BC
- A es raro en AB→ C porque F implica lógicamente a (F-{AB→C}) U {(B→C}
- C es raro en A→ BC , ya que A→ BC está implicada lógicamente en A→ B y B→ C
- El recubrimiento canónico es:

  A→ B

  B→ C