# *FUNDAMENTOS DE BASES DE DATOS*

## *Examen Diciembre 2007 - Solución*

**La duración del examen es de 3 horas.**
**Presentar la resolución del examen:**

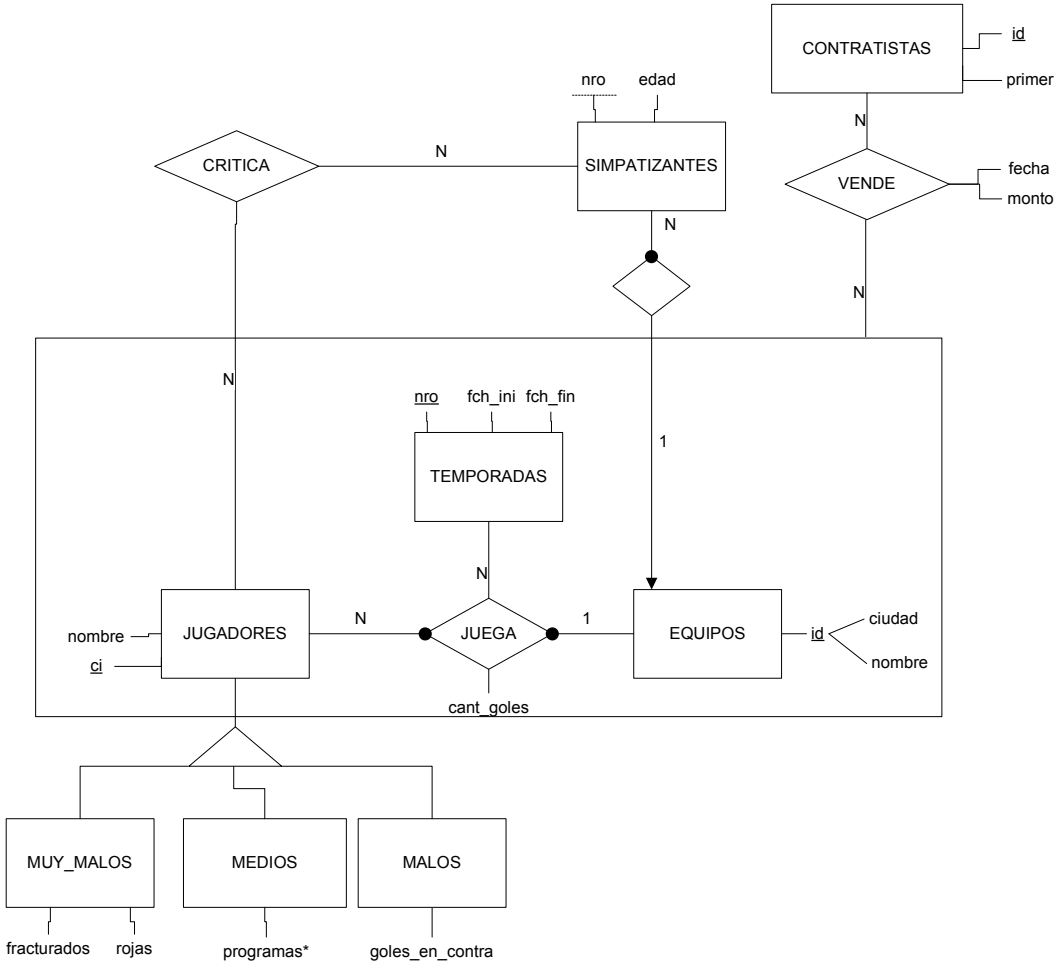
- Con las hojas numeradas y escritas de un solo lado.
- Con la cantidad de hojas entregadas en la primer hoja.
- Con cédula de identidad y nombre en cada hoja.
- Escrita a lápiz y en forma prolija.
- Comenzando cada ejercicio en una nueva hoja

## Ejercicio 1 (30 puntos).

Se debe implementar el registro de la realidad futbolística de un país determinado.
Se registran los contratistas, de los que se conoce un identificador y cuál de todos ellos fue el primero que comenzó a realizar esta tarea.
De los equipos se registran el nombre y la ciudad de donde provienen, pudiendo existir equipos distintos con el mismo nombre pero de distintas ciudades.
Se registran además los simpatizantes de cada equipo, los cuales se identifican dentro de cada equipo por su número, y además se registra de cada simpatizante su edad.
En cada equipo juegan varios jugadores en distintas temporadas. Las temporadas se identifican por un número y se registra su fecha de inicio y su fecha de finalización.
Los jugadores se identifican por su cédula de identidad y se registra también su nombre. Se registra además cuantos goles realizó en cada equipo, en cada temporada que jugó. Un jugador no puede jugar en distintos equipos en la misma temporada. Todos los jugadores jugaron al menos en alguna temporada en algún equipo. A su vez los jugadores se clasifican en forma excluyente en medios, malos y muy malos. Todos los jugadores pertenecen a una de estas categorías, y no pertenecen a más de una. De los muy malos se registra la cantidad de tarjetas rojas, y la cantidad de jugadores que ha fracturado. De los medios se registra los nombres de los programas de TV donde realiza declaraciones. De los malos se registra la cantidad de goles en contra que ha anotado.
Se registra a su vez las ventas de un jugador realizadas por contratistas en determinada temporada. De cada venta se registra la fecha y el monto. La fecha de venta debe estar en el rango de fechas de la temporada.
Los simpatizantes critican a los jugadores, pero critican solo a aquellos que no juegan, ni jugaron en el equipo del cual son simpatizantes.

**Se pide Modelo Entidad Relación completo.**

**Solución:**




RNE:

- Solo un contratista puede tener el atributo primer seteado a TRUE.
- La fecha de la relación vende debe estar en el rango de fechas de la entidad temporada.
- Los períodos de las temporadas no se pueden superponer.
- Los simpatizantes critican jugadores que no juegan ni jugaron en el equipo del cual son simpatizantes.
- JUGADORES = MUY_MALOS ∪ MEDIOS ∪ MALOS
- ∅ = MUY_MALOS ∩ MEDIOS = MUY_MALOS ∩ MALOS = MEDIOS ∩ MALOS

## Ejercicio 2 (25 puntos).

Sea el esquema relación R(A B C D E G) con el conjunto de dependencias funcionales
F = { AB→CEG, EG→BC, B→A, G→ D, AE→BD }.

Considere la descomposición δ = { $R_1$(ABCD), $R_2$(BCDEG) }

Se pide:

a) Decir en qué forma normal se encuentra δ. Justificar cada paso para llegar al resultado.
b) Decir si δ tiene join sin pérdida, justificando.
c) Decir si δ preserva las dependencias funcionales, justificando.
d) Llevar $R_2$ a BCNF utilizando el algoritmo dado en el curso (mostrar su aplicación).
e) Ahora suponga que se cumple E →→ G | BC. Llevar el esquema de la parte d) a 4NF y decir si se preservan las dependencias funcionales. Justificar.

**Solución:**

a)

**$R_1$(ABCD)**

$\prod_{R1}$(F) = { AB→C, B→A, AB→D }

Claves: B es clave única, ya que:
B+ = { ABCD } y
B debe pertenecer a todas las claves por no estar a la derecha en ninguna df (entonces cualquier otro conj de atrib que determine a todos los demás es superclave).

Forma Normal: BCNF, ya que los lados izquierdos de todas las dfs de F son superclave.

**$R_2$(BCDEG)**

$\prod_{R2}$(F) = { EG→BC, B→CEG, G→D, BE→D }

Claves: B y (EG), ya que:
- B+ = { BCDEG } y (EG)+ = { BCDEG }
- C y D no pertenecen a ninguna clave ya que aparecen solo del lado derecho de las dfs.
- Cualquier otra combinación de BEG es superclave.

Forma Normal: 1NF. Justificación: G→D viola 2NF, ya que D es un atributo no primo que depende parcialmente de una clave.

b) Por teorema visto en el curso:

Si R1 ∩ R2 → R1 – R2 ∈ F+ entonces la descomposición es con JSP.

Como BCD → A ∈ F+, δ tiene join sin pérdida.

c) Se pierde la dependencia funcional: AE → BD, ya que no se puede deducir del conjunto de dependencias $\prod_{R1}$(F) ∪ $\prod_{R2}$(F).

δ no preserva las dependencias funcionales.

d) Aplicando el algoritmo de BCNF, descomponemos $R_2$ en:

$R_{21}$(GD) y $R_{22}$(BCEG)

$F_{R21}$ = {G→D} => $R_{21}$ está en BCNF, ya que su clave es G.

$F_{R22}$ = {EG→BC, B→CEG} => $R_{22}$ está en BCNF, ya que sus únicas claves son B y (EG).

e) Se cumple la dependencia multivaluada E ->> G en el esquema $R_{22}$(EGBC).

$R_{21}$ ya esta en 4NF porque no tiene ninguna dependencia multivaluada no trivial.
Llevamos $R_{22}$ a 4NF aplicando el algoritmo:

$R_{221}$(EG), $F_{R221}$ = { } => Esta en 4NF.
$R_{222}$(EBC), $F_{R222}$ = { B→CE } => B es clave, esta en 4NF.

## Ejercicio 3 (25 puntos).

Dada la realidad de una base de datos para el manejo de la logística del stock de una empresa química que almacena productos líquidos, con los siguientes esquemas de relación:

DEPOSITOS (idDeposito, nomDeposito, esPeligroso)

Representa información de los depósitos, su identificador, nombre y si admite productos peligrosos.

PRODUCTOS (idProducto, nomProducto, esPeligroso)

Representa información de los productos, su identificador, nombre y si es peligroso o no.

STOCK (idDeposito, idProducto, cantidad)

Representa la información de la cantidad en stock de cada producto en cada depósito. Si no hay stock del producto en un depósito, no existe el registro, por lo que el atributo cantidad es mayor a cero siempre.

MOVIMIENTOS (nroMovimiento, idDepositoOrigen, idDepositoDestino, idProducto, cantidad, fecha)

Representa la información de los movimientos de stock entre distintos depósitos, qué producto se movió, qué cantidad y en qué fecha.

Además se cumplen las siguientes dependencias de inclusión:

$\prod_{idDeposito}$ (STOCK) $\subseteq$ $\prod_{idDeposito}$ (DEPOSITOS)
$\prod_{idProducto}$ (STOCK) $\subseteq$ $\prod_{idProducto}$ (PRODUCTOS)
$\prod_{idDepositoOrigen}$ (MOVIMIENTOS) $\subseteq$ $\prod_{idDeposito}$ (DEPOSITOS)
$\prod_{idDepositoDestino}$ (MOVIMIENTOS) $\subseteq$ $\prod_{idDeposito}$ (DEPOSITOS)
$\prod_{idProducto}$ (MOVIMIENTOS) $\subseteq$ $\prod_{idProducto}$ (PRODUCTOS)

Resolver la siguiente consulta en álgebra relacional:

1. Devolver los nombres de aquellos depósitos que admiten productos peligrosos, y que además tienen stock de todos los productos peligrosos.

Resolver la siguiente consulta en cálculo relacional:

2. Devolver los nombres de los productos de los cuales se realizaron movimientos el último día registrado, pero que no tienen stock en un depósito que admite productos peligrosos.

Resolver la siguiente consulta en SQL:

3. Devolver los identificadores de los depósitos que fueron origen de al menos diez movimientos de stock y su cantidad promedio de unidades de productos es al menos cien.

**Solución:**

1. Nombre de los depósitos que admiten productos peligrosos, que tienen stock de todos los productos peligrosos.

A = $\sigma_{esPeligroso=TRUE}$ (**DEPOSITOS**)
B = $\Pi_{idDeposito,\ idProducto}$ (**STOCK)**
C = $\Pi_{idProducto}$ ($\sigma_{esPeligroso=TRUE}$ (**PRODUCTOS**))
S = $\Pi_{nomDeposito}$ (**A * (B % C))**

2. Nombre de los productos de los cuales se le realizaron movimientos el último día registrado, pero que no tienen stock en un depósito que admite productos peligrosos.

{ t.nomProducto / PRODUCTOS(t) ^
(∃m1) (MOVIMIENTOS(m1) ^ m1.idProducto = t.idProducto ^
(∀m2) (MOVIMIENTOS(m2) → m1.fecha >= m2.fecha)) ^
(¬(∃v)) (STOCK(v) ^ v.idProducto = t.idProducto ^
(∃w) (DEPOSITOS(w) ^ w.esPeligroso = TRUE ^
v.idDeposito = w.idDeposito))}

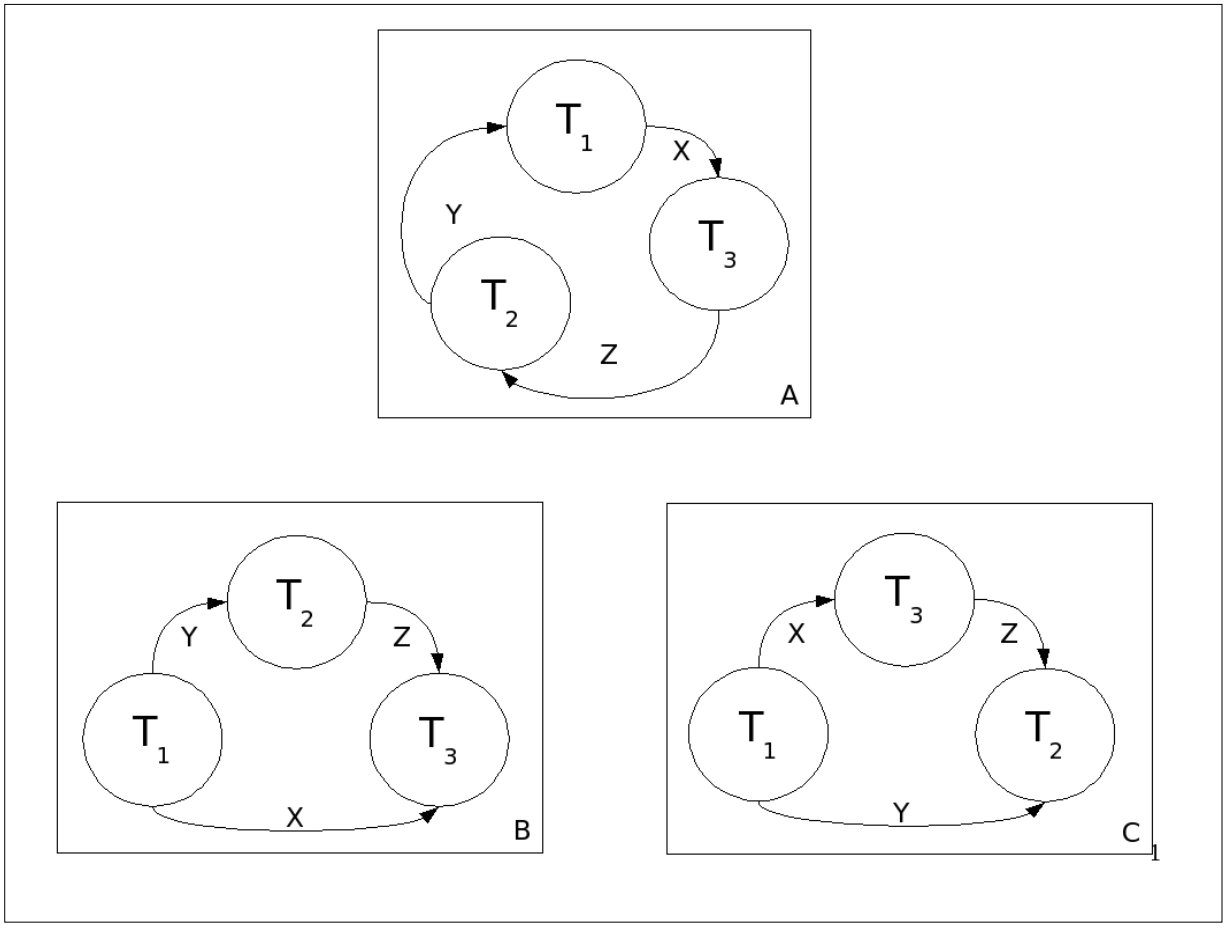
3. Identificador de los depósitos que fueron origen de al menos diez movimientos de stock y tienen al menos cien unidades en promedio para cualquiera de sus productos en stock.

```
SELECT idDeposito
FROM STOCK
WHERE idDeposito IN (SELECT idDepositoOrigen
                     FROM MOVIMIENTOS
                     GROUP BY idDepositoOrigen
                     HAVING COUNT(*) >= 10)
GROUP BY idDeposito
HAVING AVG(cantidad) >= 100
```

## Ejercicio 4 (20 puntos).

Considere las siguientes transacciones y grafos de seriabilidad:

$T_1$: $w_1(x),w_1(y),c_1$
$T_2$: $r_2(z),w_2(z),w_2(y),c_2$
$T_3$: $w_3(v),w_3(z),w_3(x),c_3$




**Se Pide**:

Para cada uno de los grafos anteriores, construya una historia entrelazada (si es posible) tal que ese sea su grafo de seriabilidad y tal que cada historia cumpla:

i. Para el grafo A, la historia tiene que evitar abortos en cascada
ii. Para el grafo B, la historia tiene que ser estricta.
iii. Para el grafo C, la historia tiene que ser recuperable pero no evitar abortos en cascada.

Justifique adecuadamente su respuesta en cada caso.

**Solución:**

**Para el grafo A:**
Los abortos en cascada son producidos por las lecturas pero no por las escrituras. Dadas las transacciones actuales, una sola lee un item. Por lo tanto es el único que deberíamos cuidar que terminara la transacción que lo graba antes que la se realice la lectura de ese item por la otra transacción.

Por otro lado, como el grafo tiene un ciclo, indica que la historia no debe ser serializable. Además debe tener en cuenta el sentido de los conflictos directos entre las transacciones por lo que en el conflicto por X de $T_1$ y $T_3$ la instrucción de $T_1$ debe aparecer antes que la de $T_3$ .

De esta forma, una solución es la siguiente:

$H_a$: $w_1(x),w_3(v),w_3(z),w_3(x),c_3,r_2(z),w_2(z),w_2(y),w_1(y),c_1,c_2$

Esta historia es no serializabl, dado que tenemos un conflicto sobre X de $T_1$ a $T_3$ , dos conflictos en Z de $T_3$ a $T_2$ y un conflicto en Y de $T_2$ a $T_1$ .
Sin embargo, la historia evita abortos en cascada dado que no hay transacciones que no lean de transacciones confirmadas previamente, dado que $T_2$ lee Z después del commit de $T_3$ .

**Para el grafo B:**
El grafo corresponde a una historia serializable en donde, con respecto a los conflictos, $T_3$ debe ir como última transacción. Como la historia debe ser estricta, entonces todas las operaciones en conflicto (sin importar el tipo- w o r ) deben ser realizadas después del commit de las que escribieron esos items.
Dado que $T_3$ maneja un ítem que no tiene conflicto con ninguna otra, podría comenzar en primer lugar.
Una solución posible es la siguiente:

$H_b$: $w_3(v),w_1(x),r_2(z),w_2(z),w_1(y),c_1 ,w_2(y),c_2,w_3(z),w_3(x),c_3$

Esta historia es serializable, ya que tenemos un conflicto sobre X de $T_1$ a $T_3$, un conflicto en Y de $T_1$ a $T_2$ y dos conflictos sobre Z de $T_2$ a $T_3$ lo que no forma ciclos. Además es estricta porque las operaciones destino del conflicto (las segundas en el orden de la historia), siempre están después del commit de la transacción origen del conflicto.

**Para el grafo c:**
El grafo corresponde a una historia serializable en donde, con respecto a los conflictos, $T_2$ debe ir como última transacción.Como la historia debe ser recuperable pero no evitar abortos en cascada, la misma debe tener los commits en el orden de los conflictos pero cuando hay lecturas, el commit de la transacción que lee debe estar **después**de la lectura de la siguiente transacción. Pero como debe ser recuperable, los commit deben estar en el orden del flujo de datos en la lectura. De esta forma, lo fundamental es grantizar que el commit de $T_3$ debe estar necesariamente después de la lectura de $T_2$ pero antes que el commit de $T_2$ . Dado que $T_1$ no tiene invlucradas lecturas en sus conflictos, se puede confirmar en cualquier lugar.

Una solución posible es la siguiente:

$H_c$: $w_3(v),w_3(z),w_1(x),w_1(y),w_3(x),r_2(z),w_2(z),c_3,w_2(y),c_2 , c_1$

Esta historia es serializable porque tenemos dos conflictos sobre Z que va de $T_3$ a $T_2$ , un conflicto sobre X que va de $T_1$ a $T_3$ y un conflicto sobre Y que va de $T_1$ a $T_2$ lo que no forma ciclos. Por otro lado, el commit de $T_3$ está después de la lectura de $T_2$ por lo que la historia no evita abortos en cascada pero como el commit de $T_3$ está antes que el de $T_2$ entonces la historia es recuperable.