BASES DE DADES
GRAU EN ENGINYERIA INFORMÀTICA
FIB - UPC

Proposta de Solucionari

Exàmens anys anteriors (2009-2013)

Per Josep Casado i Oriol Torrillas

Gener de 2014

Special thanks to:

Eric Álvarez, Cristian Barrientos, Pau Figueras,

Manu Maragall, Albert Martínez, Éric Mourín, Gerard Núñez,

Xavier Pidelaserra, Ivan Salfati, Josep Sanchez,

Aitor Sánchez, Wai Ling Tam,

David Tamayo, Yinghao Xu

Índex de continguts

Examen 14 de gener de 2009

1) (2.5 punts) Considereu l'esquema de la base de dades de la cadena de pizzeries que hem usat durant el curs:

```
create table productes
(idProducte char(9),
nom char(20),
mida char(20),
preu integer check(preu>0),
primary key (idProducte),
unique (nom,mida));

create table domicilis
(numTelf char(9),
nomCarrer char(20),
numCarrer integer check(numCarrer>0),
pis char(2),
porta char(2),
ciutat char(20),
primary key (numTelf));

create table comandes
(numComanda integer check(numComanda>0),
instantFeta integer not null check(instantFeta>0),
instantServida integer check(instantServida>0),
numTelf char(9),
import integer check(import>0),
primary key (numComanda),
foreign key (numTelf) references domicilis,
check (instantServida>instantFeta));

create table liniesComandes
(numComanda integer,
idProducte char(9),
quantitat integer check(quantitat>0),
primary key(numComanda,idProducte),
foreign key (numComanda) references comandes,
foreign key (idProducte) references productes);
```

a) Doneu una sentència SQL per disminuir un 10% l'import de les comandes que tenen més de 5 línies i que són d'un domicili que té comandes fetes des d'abans de l'instant 1000.

```
UPDATE comandes c
SET import = import * 0.9
WHERE EXISTS(SELECT *
                FROM comandes c
                WHERE c.instantFeta < 1000) and
                        5 > (SELECT COUNT(*)
                                FROM liniesComandes l
                                WHERE l.numComanda = c.numComanda);
```

b) Doneu una sentència SQL per obtenir els productes que s'han demanat en més de 5 comandes de domicilis diferents, i que són productes dels que no hi ha cap comanda feta amb un instant anterior a l'instant 2000.

```
SELECT l.idProducte
FROM liniesComandes l, comandes c
WHERE  l.numComanda = c.numComanda and NOT EXISTS(SELECT * FROM comandes c1, liniesComandes l2 WHERE l2.idProducte= l.idProducte and l2.numComanda = c1.numComanda and c1.instantFeta < 2000)
GROUP BY l.idProducte
HAVING COUNT(distinct c.numTelf) > 5
```

2) (1.5 punts) Considereu un arbre B+ d'ordre d

a) Quin és el nombre mínim d'apuntadors a registres que l'arbre pot contenir quan té dos nivells (l'arrel més un)? Per què?

$2d$

b) Quin és el nombre màxim d'apuntadors a registres que l'arbre pot contenir quan té dos nivells (l'arrel més un)? Per què?

$(2d+1)\cdot 2d$

c) Quin és el nombre mínim d'apuntadors a registres que l'arbre pot contenir quan té 3 nivells (l'arrel més dos)? Per què?

$2d\cdot(d+1)$

d) Quin és el nombre màxim d'apuntadors a registres que l'arbre pot contenir quan té 3 nivells (l'arrel més dos)? Per què?

$(2d+1)\cdot(2d+1)\cdot 2d$

e) Quina és l'expressió general pel mínim nombre d'apuntadors quan l'arbre té k nivells? Per què?

$2d\cdot(d+1)^{k-2}$

f) Quina és l'expressió general pel màxim nombre d'apuntadors quan l'arbre té k nivells? Per què?

$2d\cdot(2d+1)^{k-1}$

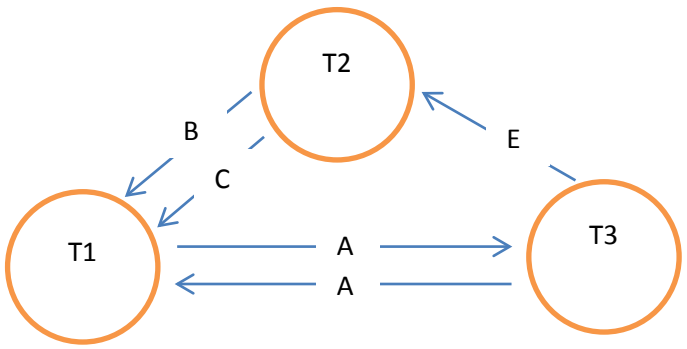
3) (2.25 punts)

a) Donat un SGBD sense cap mecanisme de control de concurrència, suposem que es produeix l'horari següent (les accions s'han numerat per facilitar la seva referència):

#Acc	T1	T2	T3
10		RU(B)	
20		W(B)	
30		R(C)	
40	RU(A)		
50	R(D)		
60	RU(B)		
70	W(B)		
80			RU(E)
90			W(E)
100			RU(A)
110			W(A)
120			RU(F)
130		R(E)	
140	W(A)		
150			W(F)
160	RU(C)		
170	W(C)		
180		COMMIT	
190			COMMIT
200	COMMIT		

Es demana:

a.1) Dibuixeu el graf de precedències associat a l'horari.



a.2) Es produeixen interferències? En cas de resposta negativa, argumenteu **breument** la vostra resposta. En cas de resposta positiva digueu quina/es interferències es produeixen (cal donar el nom de la interferència, gràNul i transaccions implicades).

Sí, es produeixen vàries interferències:

- **Actualització perduda**.
  - T1 llegeix A, T3 llegeix A i l'actualitza. Posteriorment, T1 modifica A i, per tant, es perd l'actualització feta per T3 sobre A.

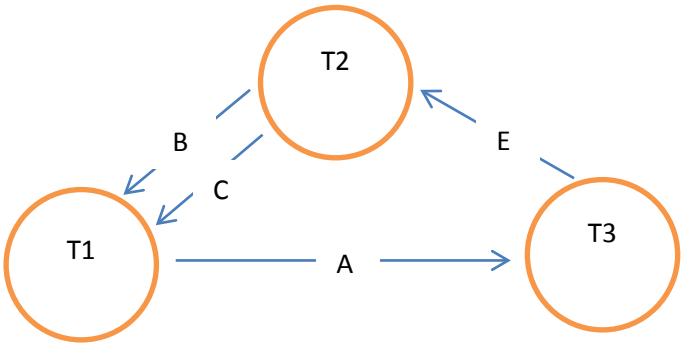
a.3) Quins horaris serials hi són equivalents?

Cap. Es produeixen vàries interferències entre les transaccions, fet que implica que no hi hagi cap horari serial equivalent.

b) Suposem ara que tenim un SGBD que treballa en mode Read Uncommited. Doneu l'horari de l'apartat a) amb les reserves i les esperes incorporades. Doneu també el graf de precedències de l'horari resuexltant d'aplicar les reserves i esperes.

#Acc	T1	T2	T3
10		LOCK(B,X)	
20		RU(B)	
30		W(B)	
40		R(C)	
50	LOCK(A,X)		
60	RU(A)		
70	R(D)		
80	LOCK(B, X)		
90			LOCK(E,X)
100			RU(E)
110			W(E)
120			LOCK(A,X)
130		R(E)	
140		UNLOCK(B)	
150		COMMIT	
160	RU(B)		
170	W(B)		
180	W(A)		
190	LOCK(C,X)		
200	RU(C)		
210	W(C)		
220	UNLOCK(A, B, C)		
230	COMMIT		
240			RU(A)
250			W(A)
260			LOCK(F,X)
270			RU(F)
280			W(F)
290			UNLOCK(A, F, E)

Graf de precedències



4) (1.75 punts)

a) Tenim una BD amb la taula Empleats. Cada fila de la taula consta de 10 camps que no accepten valors nuls i 20 camps que sí que n'accepten. En promig, dels camps que accepten valors nuls, n tenen valor diferent de nul. Quan s'emmagatzema una fila d'un empleat a una pàgina, el valor de cada un dels camps ocupa sempre 16 bytes. Considerem dues opcions per emmagatzemar les files d'empleats a la pàgina:

1) utilitzar files amb format fix, on cada camp ocupa 16 bytes independentment de si accepta o no valors nuls.
2) utilitzar files amb format variable, on tots els camps que accepten valors nuls ocupen 4 bytes per indicar si el valor del camp és nul, seguits, si el camp té valor diferent de nul, de 16 bytes amb el valor del camp.

Es demana:

i) Quin és el nombre de bytes en mitjana, que ocupa una fila d'empleats per cada una de les dues opcions d'emmagatzematge anteriors.

OPCIÓ 1

30 camps/fila · 16 bytes/camp =
480 bytes/fila

OPCIÓ 2

10camps·16bytes + (n)camps·(16+4)bytes + (20-n) camps·4bytes =
240+16n bytes/fila

ii) Per quin rang de valors de n és millor la primera opció (format fix) que la segona (format variable), només tenint en compte l'espai d'emmagatzematge ocupat.

El format fix serà millor que el format variable quan aquest ocupi menys espai. Això passa quan:

$$240 + 16n > 480 \Rightarrow 16n > 480 - 240 \Rightarrow n > \frac{480 - 240}{16} \Rightarrow \boldsymbol{n > 15}$$

iii) Donar un motiu pel qual escolliríeu l'opció de format variable per emmagatzemar les files d'empleats, encara que el valor de n fos gran.

Escolliríem el format variable per poder estendre l'esquema de la taula empleats amb més atributs.

b) Els usuaris que s'indiquen a continuació executen les sentències d'autorització següents sobre la taula Empleats, en l'ordre en què es llisten.

1	Creador:	GRANT SELECT on Empleats to Maria WITH GRANT OPTION;
2		GRANT SELECT,INSERT on Empleats to Pere;
3	Maria:	GRANT SELECT on Empleats to Joan;
4	Pere:	GRANT SELECT,INSERT on Empleats to Joan;
5	Creador:	REVOKE SELECT on Empleats FROM Maria CASCADE;

Es demana: Quins privilegis tindrà l'usuari Joan sobre la taula Empleats, després de que s'executin les sentències d'autorització anteriors? Justificar breument la resposta.

L'usuari Joan no tindrà cap privilegi sobre la taula empleats.
El privilegi de SELECT que li ha donat la Maria a la sentència 3 ha estat revocat a la sentència 5 pel creador, que ha revocat els permisos de SELECT a la Maria i **en cascada**.
Per part d'en Pere no ha rebut cap privilegi, ja que la sentència 4 no s'ha pogut realitzar de forma satisfactòria: els privilegis d'en Pere no han estat donats amb GRANT OPTION, pel que no pot donar privilegis a altres usuaris.

c) Tenim una BD amb l'esquema lògic següent:

Departaments(num_dpt,nom_dpt,pressupost)
Empleats(num_empl,nom_empl,ciutat_empl,sou,num_dpt,num_cap)
{num_dpt} Referencia Departaments, indica el departament d'un empleat.
{num_cap} Referencia Empleats, indica el num_empl del cap d'un empleat.

Un programador ha d'implementar mitjançant disparadors la restricció següent: *"la ciutat d'un empleat ha de coincidir amb la ciutat del seu cap".*

Es demana:

i) Quins esdeveniments sobre la taula empleats haurà de controlar el programador de disparadors per tal de mantenir la restricció anterior? En cas que l'esdeveniment sigui una modificació, cal indicar els atributs que afecten a la comprovació de la restricció.

- Sentències **INSERT** INTO empleats.
- Sentències **UPDATE** sobre l'atribut **ciutat_empl** o **num_cap**.

ii) Quins tipus de disparadors (row/statement i before/after) serien els més òptims per implementar la restricció anterior? Justifiqueu breument la resposta, tenint en compte les restriccions d'integritat definides sobre la BD i l'eficiència de les accions dels disparadors.

El tipus de disparador més òptim és:

- FOR EACH ROW per estalviar l'accés a tota la taula empleats, sinó que només a les que s'ha de modificar/inserir.
- Caldria decidir que sigui BEFORE si es viola amb més freqüència aquesta restricció que les regles d'integritat de la BD. Altrament, hauria de ser AFTER.

NOTA: No es demana que implementeu cap disparador, només que expliqueu breument el tipus de disparadors més òptims per implementar la restricció.

5) (2 punts) Suposem que tenim una base de dades amb les relacions següents:

X(c, a, b)
Y (e, c, d) on c és clau forana que referencia X(c)

Considerem la consulta SQL següent:

```
SELECT b, d
FROM X, Y
WHERE X.c = Y.c AND X.b = 7 AND Y.d = 5
```

a) Dona una seqüència d'operacions de l'àlgebra relacional per obtenir el mateix resultat de la consulta SQL feta amb operacions de producte cartesià, selecció i projecció i que estigui feta amb el mínim nombre d'operacions possible.

```
A = Y{c -> yc}    //reanomenem c en una de les dues taules per evitar ambigüitats en el
                  //producte cartesià
B = X x A
C = B(c = yc AND b = 7 AND d = 5)
R = C[b, d]
```

b) Transforma la seqüència d'operacions de l'àlgebra relacional de l'apartat anterior però feta amb operacions de join, selecció i projecció i el mínim nombre d'operacions possibles.

```
A = X*Y
B = A(b = 7 AND d = 5)
R = B[b, d]
```

c) Transforma la seqüència d'operacions de l'àlgebra relacional de l'apartat anterior però on les operacions de projecció es fan tant aviat com sigui possible i fent servir el mínim nombre d'operacions possibles.

A = X[c, b]
B = Y[c, d]
C = A*B
D = C(b = 7 AND d = 5)
R = D[b, d]

d) Transforma la seqüència d'operacions de l'àlgebra relacional de l'apartat b) però on les operacions de selecció es fan tant aviat com sigui possible i fent servir el mínim nombre d'operacions possibles.

A = X(b = 7)
B = Y(d = 5)
C = A * B
R = C[b, d]

Examen 18 de juny de 2009

1) (2.5 punts) Considereu l'esquema de la base de dades següent:

```
CREATE TABLE Clients(
 Nif CHAR(9) PRIMARY KEY,
 Nom CHAR(30) NOT NULL UNIQUE,
 Adresa CHAR(50),
 ciutat CHAR (20) NOT NULL,
 NumTelf CHAR(9));

CREATE TABLE Comptes(
 NumCompte CHAR(30) PRIMARY KEY,
 SaldoInicial INTEGER,
 Titular CHAR(9),
 FOREIGN KEY (Titular) REFERENCES Clients(Nif));

CREATE TABLE Transferencies(
 CompteOrigen CHAR(30),
 CompteDesti CHAR(30),
 Instant INTEGER,
 Import INTEGER,
 PRIMARY KEY (CompteOrigen,CompteDesti,Instant),
 FOREIGN KEY (CompteOrigen) REFERENCES Comptes(NumCompte),
 FOREIGN KEY (CompteOrigen) REFERENCES Comptes(NumCompte),
 Check (CompteOrigen <> CompteDesti));
```

a) Tenint en compte l'esquema de la base de dades anterior i suposant que tenim L clients, C comptes, i T transferències (L,C,T>0). Responeu les preguntes següents. Quantes files s'obtindran en el resultat de les consultes següents. Si no podeu dir el nombre exacte, indiqueu un mínim i un màxim. Justifiqueu la resposta.

1) select l.nif
from clients l, comptes c
where l.nif = c.nif

S'obtindran de 0 a C files, tantes com files com hi hagi a la taula comptes amb titular diferent de null.

2) select count(*)
from transferencies t
group by t.compteOrigen

S'obtindrà com a mínim 1 fila, en el cas que totes les transferències provinguin del mateix compte, i com a màxim C files, en el cas que s'hagi fet una transferència des de cada compte.

3) R= Comptes x Transferencies

S'obtindran C·T files, ja que el producte cartesià multiplica cada fila d'una taula per cada fila de l'altra taula.

4) R= Clients [nif * titular] Comptes

S'obtindran de 0 a C files, ja que al ser titular una clau forana de nif, no hi haurà cap compte que no tingui cap client amb qui fer el join.

b) Escriviu una sentència SQL per comptar quants clients de Manresa han fet una o més transferències entre dos comptes dels que són titulars.

```
SELECT COUNT(distinct co.titular)
FROM comptes co, clients c
WHERE co.titular = c.nif and c.ciutat = 'Manresa' and co.titular IN(SELECT c1.titular
FROM transferencies t, comptes c1, comptes c2 WHERE c1.titular = c2.titular and
t.compteOrigen = c1.numCompte and t.compteDesti = c2.numCompte);
```

c) Escriviu una sentència SQL per incrementar el saldo inicial en 1000 d'aquells comptes dels que no és titular el client amb nom 'Ot Pi' i que no han estat destí de cap transferència feta des d'un compte d'aquest titular.

```
UPDATE comptes
SET saldoInicial += 1000
WHERE comptes.titular <> (select l.nif from clients l where l.nom = 'Ot Pi') and
comptes.numCompte NOT IN(SELECT compteDesti FROM comptes co, transferencies t,
clients c1 WHERE c1.nom = 'Ot Pi' and co.titular = c1.nif and t.compteOrigen =
co.numCompte);
```

d) Contesteu les preguntes següents sobre el model relacional de bases de dades. Raoneu les respostes, posant exemples si és necessari.

1) Quantes claus primàries pot tenir com a màxim una relació?

Només es pot tenir una clau primària per relació, però es poden tenir múltiples columnes en una mateixa clau primària.

2) Expliqueu què és l'esquema, el grau i la cardinalitat d'una relació.

L'esquema d'una relació és la capçalera d'una relació i es compon del nom de la relació (ex: Assignatures) i un conjunt d'atributs (ex: grup, professor, aula).
El grau d'una relació és el nombre d'atributs(columnes) d'una taula, mentre que la cardinalitat és el nombre de tuples(files) d'una taula.

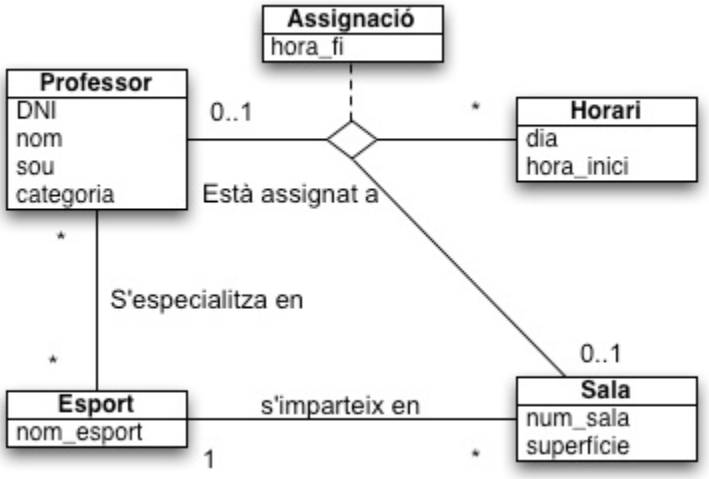
3) Expliqueu quina diferència hi ha entre una política de manteniment de la integritat referencial de restricció i de cascada.

La política de manteniment de la integritat referencial s'aplica quan es vol esborrar o modificar una clau primària referenciada per una clau forana.
La política de **restricció** produirà un error indicant que hi ha una clau forana referenciant a aquesta clau primària; en canvi, la política de **cascada** no produirà cap error, sinó que eliminarà (o actualitzarà) totes les referències.

4) Digueu quina de les dues polítiques anteriors és la que s'utilitza per defecte en PostgreSQL.

En PostgreSQL s'utilitza per defecte la política de restricció.

2) (2 punts) Considereu el disseny conceptual en UML següent:



La clau externa de la classe *Professor* és DNI, la d'*Esport* nom_esport i d'*Horari* dia i hora_inici.

Considereu també les restriccions següents:

1. Per un mateix esport el num_sala no pot estar repetit, però per esports diferents sí.
2. Tot professor que s'assigna a una sala, ha de ser especialista en l'esport que s'ha d'impartir a la sala.
3. Els professors que cobren més de 2000 euros, han de tenir obligatòriament categoria='A'.
4. A la BD, el nombre total de professors ha de ser com a mínim el doble que el nombre total d'esports.

Es demana:

a) Transformeu el disseny conceptual anterior al model relacional.

Horari(dia, hora_inici)

Esport(nom_esport)

Professor(DNI, nom, sou, categoria)

Assignació(dia, hora_inici, DNI, hora_fi, num_sala, nom_esport) on
{dia, hora_inici} referencia Horari,
{DNI} referencia Professor,
{num_sala, nom_esport} referencia Sala

Sala(num_sala, superfície, nom_esport) on
{nom_esport} referencia Esport

Especialització(DNI, nom_esport) on
{DNI} referencia Professor,
{nom_esport} referencia Esport

b) Expliqueu com s'implementarien en SQL de Postgres les restriccions 1,2,3 i sobre les taules obtingudes. Si alguna restricció s'ha d'implementar amb disparadors, indiqueu per cada restricció: esdeveniments a controlar (INSERT, DELETE, UPDATE de columnes rellevants) i tipus de disparador més òptim per implementar la restricció (BEFORE/AFTER i ROW/STATEMENT).
Considereu que les restriccions que implementeu amb disparadors es violen amb més probabilitat que les restriccions d'integritat de la BD. Justifiqueu breument la resposta.
Nota: Considereu que hi ha dos tipus d'instruccions INSERT: la que insereix una sola tupla en una taula i la que insereix un conjunt de tuples a una taula. Les vostres solucions han de funcionar eficientment pels dos tipus de sentències INSERT.

A continuació descriurem les accions que creiem que s'haurien de fer per a respectar les diferents restriccions:

1. No cal fer res, ja es respecta posant num_sala, nom_esport com a primary key.
2. S'hauria de fer un disparador que comprovi que per a cada Assignació que la combinació DNI, nom_esport existeixi a la taula Especialització.
    - Caldria controlar les sentències següents:
        - INSERT sobre la taula Assignació.
        - UPDATE de DNI sobre la taula Assignació.
        - DELETE sobre la taula Especialització.
    - Com que les restriccions amb disparadors es violen amb més freqüència, caldria decidir que el disparador fos BEFORE.
    - Caldria decidir que fos FOR EACH ROW per estalviar l'accés a tota la taula empleats, sinó que només a les que s'ha de modificar/inserir.
3. Es pot fer amb un check de taula que comprovi que (sou >= 2000) i categoria = 'A'.
4. Calen dos triggers per implementar aquesta restricció: és un model de solució incremental. Caldria crear una taula temporal on emmagatzeméssim en dues variables el número de files de professors i el d'esports, respectivament.
    - Caldria controlar les sentències següents:
        - INSERT sobre les taules Professors, Esports
        - DELETE sobre les taules Professors, Esports
    - Com que les restriccions amb disparadors es violen amb més freqüència, caldria decidir que els disparador fossin BEFORE.
    - El primer trigger caldria que fos FOR EACH ROW per anar augmentant el número de files que es van inserint (o esborrant). El segon disparador caldria que fos FOR EACH STATEMENT i que comprovés que la proporció professors-esports es segueix mantenint.

c) Si a l'apartat anterior heu proposat algun disparador amb esdeveniment activador UPDATE, doneu la seva implementació. Podeu donar la implementació del disparador utilitzant pseudo-llenguatge o pseudo-codi. Nota: cal implementar només un disparador. Si teniu més d'un disparador per l'esdeveniment UPDATE, escolliu-ne un qualsevol.

Implementarem el disparador de la restricció núm. 2.

```
CREATE FUNCTION comprovar_especialitat() RETURNS trigger AS $$
BEGIN
        IF(NOT EXISTS(SELECT * FROM assignació a WHERE new.DNI =  a.DNI and
        new.nom_esport = a.nom_esport) THEN
                RAISE EXCEPTION 'El professor no es especialista';
        END IF;
        RETURN NEW;
END

CREATE TRIGGER especialitat BEFORE DELETE ON Especialitzacio OR UPDATE OF dni OR
INSERT ON ProfessorFOR EACH ROW EXECUTE PROCEDURE comprovar_especialitat();
```

3) (2 punts)

Teniu una taula R(X,Y) on les tuples d'aquesta taula tenen valors (1,1),(2,2),(3,3), etc.

La transacció T1 és la consulta següent:

```
SELECT SUM(Y) FROM R;
COMMIT;
```

T2 és la seqüència d'actualitzacions següent:

```
UPDATE R SET Y=2*Y WHERE X=20;
UPDATE R SET Y=2*Y WHERE X=30;
UPDATE R SET Y=2*Y WHERE X=40;
COMMIT;
```

T3 és la seqüència de supressions següent:

```
DELETE FROM R WHERE x=20;
DELETE FROM R WHERE x=30;
COMMIT;
```

Abans de que cap d'aquestes transaccions s'executi la suma dels valors de la columna Y de la relació R és 2000 (aquests Ys inclouen els valors 20,30 i 40). Per simplificar considereu que les accions (reads) de la T1 s'intenten executar sempre totes seguides.

a) Si les transaccions s'executessin concurrentment en mode REPEATABLE READ quines sumes poden ser llegides per T1? Per què?

- Pot ser llegida la suma inicial (2000).
- Pot ser llegida la suma de just acabada T2 (un cop fets tots els updates) però sense executar encara T3, és a dir, 2000+20+30+40 = 2090.
- Pot ser llegida la suma un cop s'han eliminat (amb la transacció T3) les files x = 20 i x = 30 i actualitzada la fila x = 40. Per tant, la suma serà 2000-20-30+40 = 1990.

b) Quines de les sumes obtingudes a l'apartat anterior serien més correctes? Per què?

Totes són vàlides, doncs no hi ha cap interferència.

c) Quines sumes noves es podrien produir si les transaccions s'executessin en READ UNCOMMITTED? Per què?

Podrien aparèixer lectures intermèdies (per exemple una lectura després del primer UPDATE de T2, o una lectura després del primer DELETE de T3.

4) (2 punts) No està inclosa, ja no pertany al temari de l'assignatura.

5) (1.5 punts)

a) Digueu per cadascuna de les afirmacions següents si és certa o falsa. Raoneu la resposta.

- Segons l'arquitectura ANSI/SPARC, un canvi a l'esquema intern afecta sempre a l'esquema conceptual.

  Fals. L'arquitectura ANSI/SPARC proveeix tant independència lògica com física.

- Els espais virtuals d'agrupació barregen dades de més d'una taula, amb l'objectiu de resoldre eficientment les operacions de combinació (join) de les taules que comparteixen l'espai virtual.

  Cert. És precisament l'objectiu dels espais virtuals d'agrupació: agrupar taules molt relacionades on l'accés és quasi conjunt.

- El resultat d'una selecció de l'àlgebra relacional és sempre una relació de cardinalitat menor que la de la relació de partida.

  Fals. Si totes les tuples de la relació de partida compleixen la condició C, la cardinalitat del resultat serà igual que el de partida.

b) Considereu la taula T(A,B,C), propietat d'en Toni. Suposeu també la seqüència de sentències següent relativa a autoritzacions sobre la taula T. Cada sentència està numerada i s'indica el nom de l'usuari que vol executar-la.

1 - Toni:	GRANT SELECT ON T TO Albert WITH GRANT OPTION
2 - Albert:	GRANT SELECT ON T TO Carme WITH GRANT OPTION
3 - Carme:	GRANT SELECT(A,C) ON T TO Dolors WITH GRANT OPTION
4 - Carme:	GRANT SELECT(A,B) ON T TO Se WITH GRANT OPTION
5 - Toni:	GRANT SELECT ON T TO Se
6 - Toni:	GRANT SELECT(C) ON T TO Xavi
7 - Dolors:	GRANT SELECT(A,C) ON T TO Xavi WITH GRANT OPTION
8 - Se:	GRANT SELECT(A,C) ON T TO Xavi
9 - Dolors:	GRANT SELECT(A) ON T TO Elena
10 - Se:	GRANT SELECT(A) ON T TO Elena
11 - Toni:	REVOKE SELECT ON T FROM Se RESTRICT
12 - Carme:	REVOKE SELECT(A,C) ON T FROM Dolors RESTRICT
13 - Dolors:	REVOKE SELECT(A) ON T FROM Se
14 - Albert:	REVOKE SELECT ON T FROM Carme CASCADE
15 - Toni:	REVOKE SELECT ON T FROM Albert RESTRICT

1) Quines d'aquestes sentències, si n'hi ha cap, no s'executarien amb èxit seguint el llenguatge SQL estàndard? Raoneu la resposta. Assumirem que les sentències que no s'executin amb èxit no tindran cap efecte i es continuarà amb la sentència següent.

Donaran error les següents sentències:
8 – Se no pot donar permisos a Xavi sobre A i C perquè (amb GRANT OPTION) només en tenia de A i B, però no de C.
12 – Carme rep error per la clàusula RESTRICT, ja que Dolors havia donat altres permisos a altres usuaris (a Elena i a Xavi).
13 – Dolors no pot revocar uns permisos que mai ha donat a Se.

2) En acabar d'executar les sentències anteriors, quins privilegis tindran els usuaris Xavi i Albert sobre la taula T?

Xavi tindrà permisos de SELECT sobre C sense GRANT OPTION.

c) Suposem que tenim la base de dades següent, on les claus primàries són les subratllades:

T(A, B, C, D)			
1	a	4	α
2	b	5	β
3	f	6	α
5	d	7	δ

S(A, B, E)		
1	c	α
3	a	δ
4	f	α
5	a	ε

Considereu, també, la seqüència d'operacions d'àlgebra relacional següent:

R1:= T(A<6 i C<>7)
R2:= R1[A, B, D]
R3:= S(A<5)
R:= R2[B*B, D<>E]R3

1) Doneu l'esquema i l'extensió de la relació R.

R	(A,	A,	B,	D,	E)
	1	3	a	α	δ

2) Doneu una sentència en SQL que sigui equivalent a la seqüència d'operacions anterior.

SELECT T.A, S.A, T.B, T.D, S.E
FROM T, S
WHERE T.B = S.B and T.A < 6 and T.C <> 7 and S.A < 5 and T.D <> S.E

Examen 19 de gener de 2010

1) (2 punts) Considereu l'esquema de la base de dades següent:

```
create table especies
 (especie char(20) primary key,
 comestible char(1) not null check ((comestible='S') or
(comestible='N')),
 valor integer not null check (valor >= 0 and valor <= 10),
 check ((comestible = 'S') or (valor = 0)) );
-- En aquesta taula hi ha una fila per cada especie de bolets.

create table boletaires
 (nom char(20) primary key,
numLlicencia integer unique,
ciutat char(20),
edat integer check (edat >= 18) );
 -- En aquesta taula hi ha una fila per cada caçador de bolets

create table troballes
 (nom char(20),
especie char(20),
jornada integer,
quantitat integer check (quantitat > 0),
lloc char(20),  primary key (nom, especie, jornada),
foreign key (nom) references boletaires,
foreign key (especie) references especies);
 -- En aquesta taula hi ha una fila per cada espècie de bolets que
troba un boletaire en una jornada.
```

1.1) Escriviu una sentència SQL per obtenir el número de llicència dels boletaires que han trobat més de 15 espècies en una mateixa jornada.

```
SELECT distinct b.numLicencia
FROM boletaires b NATURAL INNER JOIN troballes t
GROUP BY b.numLlicencia, t.jornada
HAVING COUNT(*) > 15
```

1.2) Escriviu una sentència SQL, en la que no s'utilitzin funcions d'agregació, per obtenir les ciutats on viuen boletaires que tenen troballes de totes les especies emmagatzemades a la base de dades.

```
SELECT distinct b.ciutat
FROM boletaires b
WHERE NOT EXISTS(SELECT * FROM especies e WHERE e.especie NOT IN(SELECT
t.especie FROM troballes t WHERE b.nom = t.nom));
```

1.3) Suposem que ara algú ha creat una nova taula a la base de dades amb l'esquema següent:

```
create table classificacio
(nomBoletaire char(20),
 quantitatTotal integer not null check(quantitatTotal >=0),
primary key (nomBoletaire),
foreign key (nomBoletaire) references boletaires);
```

Doneu una (o en cas que no ho trobeu possible, més d'una) sentència d'inserció de files a la taula *Classificació* que l'ompli a partir del contingut de la resta de taules de la base de dades. Tingueu en compte el següent: Hi haurà una fila de la taula per cada boletaire de la base de dades. El valor de l'atribut quantitat total serà la suma de les quantitats de bolets trobades pel boletaire en totes les jornades. En cas que un boletaire de la base de dades no tingui troballes de cap espècie, el valor de la quantitat total ha de ser zero.

```
INSERT INTO classificacio
(SELECT t.nom, SUM(t.quantitat)
FROM troballes t NATURAL INNER JOIN boletaires b
GROUP BY t.nom);
```

2 (2 punts) Suposem que tenim la transacció T següent que fa dues consultes sobre la relació d'empleats. L'esquema de la relació d'empleats és: Empleats(idEmpleat, nomEmpleat, sou, numDept).

```
          T
Q1: SELECT avg(sou) FROM Empleats;
Q2: SELECT avg(sou) FROM Empleats;
Commit;
```

Es demana:

a) Suposem que totes les altres transaccions del sistema treballen a nivell SERIALITZABLE i que són read-only. Explicar breument quin és el mínim nivell d'aïllament al que hauria de treballar la transacció T, per evitar qualsevol interferència.

Com que totes les altres transaccions treballen en read-only, és a dir, que no modifiquen dades, no hi ha risc d'interferència pel que podem utilitzar el mínim nivell d'aïllament possible: READ UNCOMITTED.

b) Suposem que totes les altres transaccions del sistema treballen a nivell SERIALITZABLE i que només fan consultes, updates i deletes. Explicar breument quin és el mínim nivell d'aïllament al que hauria de treballar la transacció T, per evitar qualsevol interferència.

Les interferències que ens podrien sortir en aquest cas sense aïllament seria de lectura no repetible, però no hi ha cap INSERT, pel que no poden aparèixer fantasmes. Per tant, el mínim nivell d'aïllament seria REPEATABLE READ.

c) Suposem que totes les altres transaccions del sistema treballen a nivell SERIALITZABLE i que poden fer qualsevol operació. Explicar breument quin és el mínim nivell d'aïllament al que hauria de treballar la transacció T, per evitar qualsevol interferència.

Com que en aquest apartat es permeten tot tipus d'operacions, cal evitar qualsevol tipus d'interferència possibles, inclosos els fantasmes, pel que T haurà de tenir el màxim nivell d'aïllament: SERIALIZABLE.

d) Ara considereu la següent variació, on les dues consultes estan a dues diferents transaccions:

```
T1:   (Q1)   SELECT AVG(sou) FROM Empleats;
             Commit;

T2:   (Q2)   SELECT AVG(sou) FROM Empleats;
             Commit;
```

Suposeu que les dues transaccions T1 i T2 treballen a nivell READ COMMITTED. Considereu els escenaris descrits a), b) i c) per les altres transaccions. En quin d'aquests escenaris, si n'hi ha cap, s'obtindrà sempre el mateix resultat per Q1 i Q2.

En l'apartat a, ja que no es modifica cap tipus de dades, només es llegeixen.

e) Continuant amb la variació de les dues transaccions de l'apartat d), suposeu ara que T1 i T2 treballen a nivell SERIALITZABLE. En quin d'aquells escenaris, si n'hi ha cap, s'obtindrà sempre el mateix resultat per Q1 i Q2.

Com en l'apartat anterior, només s'obtindrà sempre el mateix resultat en l'escenari a, ja que encara que el nivell d'aïllament sigui màxim no ens assegura que no es modifiquen dades entre Q1 i Q2.

3 (2 punts) Una empresa gran disposa de la taula Empleats(idEmpleat,nomEmpleat,ciutat,...). S'ha creat un índex agrupat per idEmpleat i 2 índexs no agrupats, un per nomEmpleat i l'altre per ciutat.

Sabem que la taula d'empleats té aproximadament 100.000 empleats, que els idEmpleats són nombres entre 1 i 100.000 repartits més o menys uniformement, que l'empresa té uns 1.000 empleats a Barcelona. A més, se sap que els índexs són arbres B+ d'ordre 50, que les pàgines de l'índex són al 80% de la seva capacitat en mitjana, i a una pàgina de dades hi ha unes 5 files en mitjana.

Donada la consulta

```
SELECT * FROM Empleats e
WHERE e.idEmpleat > 20000 AND e.ciutat =”Barcelona”
```

estimeu (i justifiqueu breument) el nombre de pàgines (d'índex i de dades) que es llegiran si la consulta es resol:

a) sense usar els índexs

Índex: no n'hi ha.
Dades: s'han de llegir totes les files, per tant, es llegiran 100.000/5 = 20.000 pàgines

b) usant accés seqüencial per valor usant l'índex per idEmpleat

Índex: h = 3 pàgines
Dades: |R(a >= X)|/f = |R(a >= 20.000)|/f = (100.000-20.000)/5 = 16.000 pàgines

c) usant accés seqüencial per valor usant l'índex per ciutat

Índex: h+F = 3 + (1.000/80) -1 = 15 pàgines
Dades: |R(b = Y)| = |R(b = 1.000)| = 1000 pàgines

4 (2 punts)

a) Considereu la taula Empleats(id, categoria, sou) i l'asserció següent:

```
CREATE ASSERTION Limitinterns CHECK (
       NOT EXISTS ( SELECT *
                    FROM Empleats
                    WHERE categoria = ‘intern’
                       and sou > (SELECT AVG(sou) FROM Empleats)));
```

Es demana quines sentències d'actualització (inserció, esborrat i modificació) poden violar aquesta restricció. En el cas de la sentència de modificació digueu quin/s és/són els atributs rellevants. Per cadascuna d'aquestes sentències, doneu un extensió de la taula Empleats i un exemple de la sentència que produeixi la violació.

- INSERTS de nous empleats interns amb sous per sobre de la mitjana.

  INSERT INTO EMPLEATS(1,'intern',500);

ID	Categoria	Sou
1	intern	200

AVG = 200

ID	Categoria	Sou
1	intern	200
2	intern	500

AVG = 350

- DELETE d'algun empleat que faci baixar la mitjana dels sous i quedi algun altre empleat amb un sou per sobre de la mitjana.

  DELETE FROM EMPLEATS WHERE id = 1;

ID	Categoria	Sou
1	extern	500
2	intern	300
3	extern	200

AVG = 333

ID	Categoria	Sou
2	intern	300
3	extern	200

AVG = 250

- UPDATE: pujar el sou/canviar de categoria a un empleat amb un sou per sobre de la mitjana, baixar el sou a algun empleat/canviar de categoria a un empleat amb un sou baix que faci baixar la mitjana i quedin altres empleats fora de la mitjana.

  UPDATE EMPLEATS SET CATEGORIA = 'intern' WHERE id = 1;

ID	Categoria	Sou
1	extern	500
2	intern	300
3	extern	200

AVG = 333

ID	Categoria	Sou
1	intern	500
2	intern	300
3	extern	200

AVG = 333

b) Considereu la taula Empleats(id, categoria, sou) anterior i les taules següents:

```
Tipusprojectes(tipus)
Projectes(tipus, projecte, pressupost)
      on {tipus} referencia Tipusprojectes
Responsables(id, tipus, projecte, datainici, datafi)
      on {tipus, projecte} referencia Projectes,
      on {id} referencia Empleats
Especialitzacions(id, tipus)
      on {tipus} referencia Tipusprojectes,
      on {id} referencia Empleats
```

Volem garantir que tot empleat assignat com a responsable de projecte, sigui especialista en aquest tipus de projecte. Un estudiant de BD ens ha fet el disparador següent:

```
CREATE OR REPLACE FUNCTION comprova_restriccio RETURNS Trigger AS $$
BEGIN
      if (NOT EXISTS(SELECT *   FROM Especialitzacions
                                WHERE id=NEW.id AND tipus=NEW.tipus))
            then
                  RAISE EXCEPTION 'L'empleat ha de ser especialista
                  en el tipus de projecte';
      end if;
      RETURN new;
END;
$$language plpgsql;

CREATE TRIGGER Restriccio BEFORE INSERT on Responsables
FOR EACH ROW execute procedure comprova_restriccio();
```

Es demana:

b.1) Aquest disparador cobreix totes les sentències d'actualització (inserció, esborrat i modificació) i sobre totes les taules que poden violar la restricció que volem garantir? En cas afirmatiu expliqueu perquè no cal cap altre disparador més, en cas negatiu digueu sobre quines taules i esdeveniments s'haurien de definir els disparadors que falten.

No. Aquest disparador no cobreix les sentències UPDATE sobre la taula Responsables, ni tampoc les sentències DELETE sobre la taula Especialitzacions, que podrien violar les restriccions.

b.2) Es podria implementar aquesta restricció directament amb restriccions de columna i/o de taula? En cas afirmatiu digueu com ho faríeu, en cas negatiu digueu perquè no es podria fer.

No. Es tracta d'una restricció de més d'una taula (Especialitzacions, Responsables).

5 (2 punts)

a) Considereu les relacions R(A,B) i S(A,B). Es vol obtenir la intersecció de R i S, però només disposem de 5 operadors relacionals: selecció, projecció, producte cartesià, natural join i reanomenament. És possible calcular la intersecció de R i S fent servir només aquests operadors? En cas afirmatiu, doneu l'expressió més simple equivalent a la intersecció. En cas negatiu, raoneu la resposta.

C = R * S

b) Per a cadascuna de les polítiques d'actuació en cas d'esborrat d'una clau primària amb claus foranes que hi fan referència, doneu els seus noms (utilitzant notació SQL estàndard) i una breu explicació del seu funcionament.

RESTRICT: no deixa esborrar la clau primària en qüestió.
SET NULL: posa a NULL totes les claus foranes que referenciaven aquella clau primària.
CASCADE: esborra totes les tuples que referencien la clau primària a suprimir

c) Considerant la taula i la inserció següents:

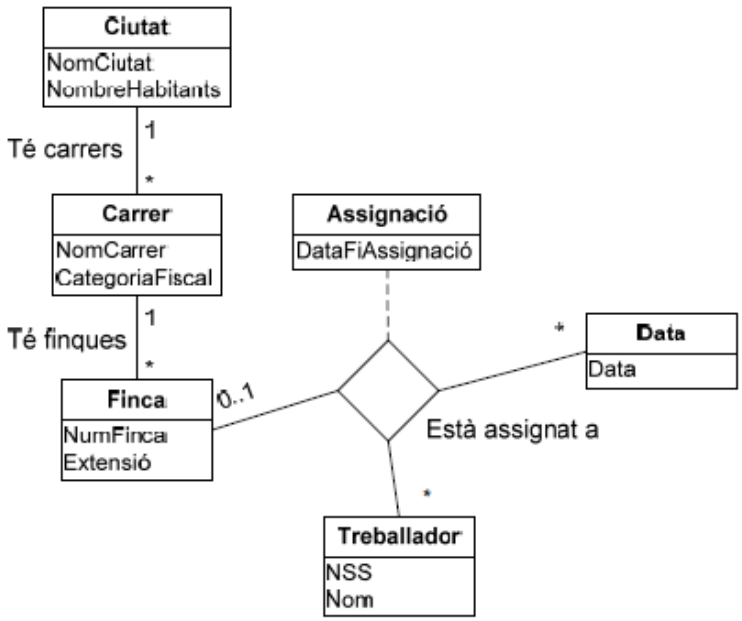
```
CREATE TABLE Empleats (
Id INTEGER,
Nom VARCHAR(50),
Categoria VARCHAR(20),
Sou FLOAT NOT NULL);
INSERT INTO Empleats (102, 'Pere', NULL, 1500);
```

Doneu la capçalera i el contingut de cadascun dels camps de la fila inserida. I doneu també la longitud total de la fila.

Camp1 5 bytes	Camp2 6 bytes		Camp3 1 byte	Camp4 8bytes
4	102	4	Pere	0	1500

Longitud total fila: 21 bytes (1 byte de núm de pàgina + 20 bytes camp)

d) Considerant el disseny conceptual en UML següent:



Les claus externes i restriccions textuals del disseny anterior són les següents:

Clau externa Ciutat: NomCiutat
Clau externa Finca: NumFinca
Clau externa Data: Data
Clau externa Treballador: NSS
Restricció1: Dins d'una mateixa ciutat, no poden haver carrers amb el mateix NomCarrer, però en ciutats diferents sí.
Restricció2: Dins d'un mateix carrer, no poden haver finques amb el mateix NumFinca, però en carrers diferents sí.

Transformeu el disseny conceptual anterior al model relacional.

Ciutat(NomCiutat, NombreHabitants)
Carrer(NomCiutat, NomCarrer, CategoriaFiscal)
Finca(NumFinca, NomCiutat, NomCarrer, Extensio)
Treballador(NSS, Nom)
Assignació(Data, NSS, numFinca, nomCiutat, NomCarrer, DataFiAssignacio)

Examen 7 de juny de 2010

1) (2 punts) Considereu l'esquema de la base de dades de mobles que hem usat durant el curs:

```
create table mobles
(idMoble integer,
descripcio char(50) unique,
preuVenda integer not null check(preuVenda>0),
estocMoble integer not null check(estocMoble>0),
primary key (idMoble));

create table factures (numFactura integer,
idMoble integer,
nomClient char(30),
quantitat integer not null check (quantitat>0),
instant integer not null check (instant>0),
unique (idMoble, nomClient, instant),
primary key (numFactura),
foreign key (idMoble) references mobles(idMoble));

create table peces
(idPeca integer,
descripcio char(50) unique,
preuPeca integer not null check(preuPeca>0),
estocPeca integer not null check(estocPeca>0),
primary key (idPeca));

create table composicio
(idMoble integer,
idPeca integer,
quantitat integer not null check(quantitat>0),
primary key (idMoble, idPeca),
foreign key (idMoble) references mobles,
foreign key (idPeca) references peces);
```

a) Doneu una sentència SQL per disminuir un 25% el preu de venda dels mobles que tenen més de 5 peces diferent i que tenen factures fetes abans de l'instant 666.

```
UPDATE Mobles
SET preuVenda = preuVenda * 0.75
WHERE mobles.idMoble IN (SELECT idMoble FROM composició co GROUP BY idMoble
                                        HAVING COUNT (*) > 5)
      and EXISTS(SELECT * FROM factures f WHERE f.idMoble = mobles.idMoble and
                 f.instant < 666);
```

b) Doneu una sentència SQL per obtenir els instants en que s'hagi facturat a exactament 5 clients diferents una unitat del mateix moble i aquest moble no estigui composat per peces amb estoc insuficient per construir els cinc mobles.

```
SELECT f.instant
FROM factures f
WHERE NOT EXISTS(SELECT * FROM composició c, peces p WHERE f.idMoble = c.idMoble and c.idPeca = p.idPeca and p.idPeca < 5 * c.quantitat)
GROUP BY f.instant, f.idMoble
HAVING COUNT(distinct nomClient) = 5
```

c) Tenint en compte l'esquema de la base de dades anterior i suposant que tenim |M| mobles, |F| factures, |P| peces i |C| composicions (|M|, |F|, |P| i |D|>0). Responeu les preguntes següents. Quantes files es poden obtenir com a màxim en el resultat de les consultes següents? Justifiqueu la resposta.

c.1) select count(*)
from composicions
group by idPeca;
S'obtindrà com a màxim P, en el cas que s'utilitzin totes les peces per a fer composicions.

c.2) R= Factures[idMoble]
Es poden obtenir com a màxim M files, ja que en àlgebra relacional no hi ha files repetides.

c.3) R= Mobles * Peces
S'obtindrà el max(M, P) ja que

2) (2 punts) Donades les transaccions següents (per cada transacció indiquem les operacions que volem executar):

T1	RU(A)	W(A)	RU(B)	W(B)	COMMIT
T2	RU(A)	W(A)	COMMIT		
T3	R(B)	R(C)	R(B)	COMMIT	

Es demana:

a) Proposar un horari que inclogui les 3 transaccions i que tingui els dos horaris serials equivalents següents: T1;T2;T3 i T1;T3;T2.

#Acc	T1	T2	T3
10	RU(A)		
20	W(A)		
30		RU(A)	
40			R(B)
50			R(C)
60			R(B)
70			COMMIT

80		W(A)
90		COMMIT
100	RU(B)	
110	W(B)	
120	COMMIT	

b) Proposar un horari que inclogui les 3 transaccions i que tingui dues interferències. Cal indicar el nom de les interferències, els grànuls i les transaccions implicades.

#Acc	T1	T2	T3
10	RU(A)		
20		RU(A)	
30	W(A)		
40			R(B)
50			R(C)
60	RU(B)		
70	W(B)		
80		W(A)	
90		COMMIT	
100			R(B)
110			COMMIT
120	COMMIT		

Interferències

- **Actualització perduda**. A és llegida per T1 i per T2 en el mateix estat, T1 modifica A i T2 ho fa més tard, però amb les dades d'A abans de ser modificada per T1: es perden els canvis de T1.
- **Lectura no repetible**. B és llegida per T3, posteriorment B és modificada per T1 i és tornada a llegir per T3, aquest cop amb un valor diferent a l'anterior.

c) Considereu ara només T1 i T3. Per a cada nivell d'aïllament proposa un horari de transaccions amb els locks corresponents de tal manera que es produeixi una interferència. En cas de que no sigui possible, justifiqueu-ho.

READ UNCOMMITED

#Acc	T1	T3
10	LOCK(A, X)	
20	RU(A)	
30	W(A)	
40		R(B)
50		R(C)
60	LOCK(B, X)	
70	RU(B)	
80	W(B)	

90	UNLOCK(A, B)	
100	COMMIT	
110		R(B)
120		COMMIT

Es segueix produint la interferència de lectura no repetible sobre B.

READ COMMITED

#Acc	T1	T3
10	LOCK(A, X)	
20	RU(A)	
30	W(A)	
40		LOCK(B, S)
50		R(B)
60		UNLOCK(B)
70		R(C)
80	LOCK(B, X)	
90	RU(B)	
100	W(B)	
110	UNLOCK(A, B)	
120	COMMIT	
130		LOCK(B, S)
140		R(B)
150		UNLOCK(B)
160		COMMIT

Es segueix produint la interferència de lectura no repetible sobre B.

REPEATABLE READ

#Acc	T1	T3
10	LOCK(A, X)	
20	RU(A)	
30	W(A)	
40		LOCK(B, S)
50		R(B)
60		R(C)
70	LOCK(B, X)	
80		R(B)
90		UNLOCK(B)
100		COMMIT
110	RU(B)	
120	W(B)	
130	UNLOCK(A, B)	
140	COMMIT	

La interferència s'evita, ja que es fan reserves tant per lectura(reserves S) com per escriptura(reserves X) fins al final de la transacció, fet que evita la lectura no repetible (tot i que no els fantasmes, malgrat que en aquest horari no n'hi ha).

SERIALITZABLE

#Acc	T1	T3
10	LOCK(A, X)	
20	RU(A)	
30	W(A)	
40		LOCK(B, X)
50		R(B)
60		R(C)
70	LOCK(B, X)	
80		R(B)
90		UNLOCK(B)
100		COMMIT
110	RU(B)	
120	W(B)	
130	UNLOCK(A, B)	
140	COMMIT	

S'eviten totes les interferències, l'aïllament serialitzable és el màxim nivell d'aïllament i es fan reserves X fins al final de la transacció, tant per lectura com per escriptura.

3) (2 punts) Donades les taules següents:

```
create table empleats1
(nemp1 integer primary key,
nom1 char(25),
 ciutat1 char(10) not null);

create table empleats2
(nemp2 integer primary key,
nom2 char(25),
ciutat2 char(10) not null);
```

i la restricció:

- Els valors de l'atribut ciutat1 de la taula empleats1 han d'estar inclosos en els valors de l'atribut ciutat2 de la taula empleats2

Suposeu que les restriccions d'integritat de la BD es violen amb menys freqüència que aquesta restricció

Es demana:

a) Digueu quines sentències d'actualització (inserció, esborrat i modificació) poden violar aquesta restricció. En el cas de les sentències de modificació digueu quin/s és/són els atributs rellevants.

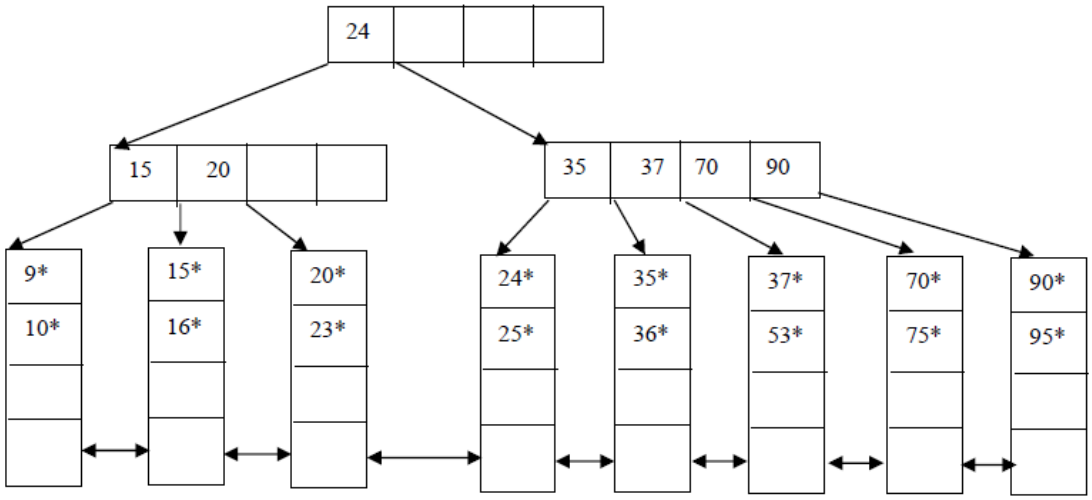
Poden violar la restricció les sentències següents:

- INSERT sobre la taula empleats1
- UPDATE de ciutat1 sobre la taula empleats1
- UPDATE de ciutat2 sobre la taula empleats2
- DELETE sobre la taula empleats2

b) Doneu la implementació del disparador associat a l'esdeveniment d'update de ciutat2 de la taula empleats2. Si considereu que aquest esdeveniment no pot violar la restricció, justifiqueu-ho raonadament.

```
CREATE FUNCTION comprova_ciutat2()  RETURNS trigger AS $$
BEGIN
        IF(NOT EXISTS(SELECT * FROM empleats1 e1, empleats2 e2 WHERE old.ciutat2
        =   e1.ciutat1 and e2.nemp2 <> old.nemp and e2.ciutat2 = e1.ciutat1) THEN
                RAISE EXCEPTION 'ciutat1 no esta inclosa a ciutat2';
        END IF;
        RETURN NEW;
END

CREATE TRIGGER especialitat BEFORE UPDATE OF ciutat2 ON empleats2 ProfessorFOR
EACH ROW EXECUTE PROCEDURE comprova_ciutat2();
```

4) (2 punts) Donat l'arbre B+ d'ordre 2 (d=2) següent:



Es demana (justifiqueu **breument** les vostres respostes):

L'arbre B+ de la figura inicial emmagatzema 16 entrades. Suposant que només voldrem inserir aquestes 16 entrades, es demana:

c.1) Quin seria l'ordre d més gran per tal d'obtenir un arbre B+ que tingui 2 nivells?

$Log_d\ 16 = 2$; $2^d = 16$; **d = 4**

c.2) Quin seria l'ordre d més gran per tal d'obtenir un arbre B+ que tingui 4 nivells?

$Log_d\ 16 = 4$; $4^d = 16$; **d = 2**

5) (2 punts)

a) Responeu i justifiqueu breument l'afirmació següent: Donat el graf d'autoritzacions següent, és possible eliminar totes les autoritzacions (fletxes) fent servir només l'operació REVOKE amb el mode RESTRICT ?

b) Responeu i justifiqueu breument l'afirmació següent: Donada una BD on existeixen restriccions sobre el nombre d'índexs a crear, per a decidir si sobre un atribut A d'una taula T creem o no un índex, l'únic factor a considerar és la grandària de la taula T.

c) Considereu la relació velocitat (llançador, rapid, lent) que recull la velocitat més ràpida (la màxima) i la velocitat més lenta (la mínima) de llançament dels llançadors.
Considereu l'expressió d'àlgebra relacional següent:

```
A = Velocitat[llançador, rapid]
B = A {rapid ->r}
C = Velocitat[llançador, lent]
D = C {lent ->l}
E = Velocitat {llançador -> ll}
F = B[r < rapid]E
```

```
G = F[llançador]
H = D[l > lent]E
I = H[llançador]
J = Velocitat[llançador]
K = J – G
L = J –  I
Resposta = K ∪ L
```

Escriviu una sentència en narrativa que expressi quines són les dades que s'obtindrien a partir de l'execució de la seqüència d'operacions d'àlgebra relacional anterior. Aquesta sentència podria ser usada com a enunciat d'un exercici on el resultat fos el de la sentència d'operacions que us donem.

d) Considereu la relació Empleats (num-empl, nom-empl, sou) amb les dades següents:

Empleats(num-empl, nom-empl, sou)

123	Pere	1000
321	Maria	2000
213	Joan	3000

d.1) Hi ha tres usuaris (Pere, Maria i Joan) treballant amb la BD on es troba la relació Empleats i cap d'ells hi té accés a aquesta relació. Doneu les sentències SQL que calguin per a que la Maria pugui consultar i modificar les dades dels empleats amb un sou més petit de 2500 euros.

d.2) Un cop la Maria tingui l'accés que li heu donat a l'apartat anterior, justifiqueu si podrà o no executar la sentència SQL següent:

```
UPDATE  Empleats
SET sou = sou + 1000
WHERE num_empl = 321;
```

Examen 20 de gener 2011

1) (2.5 punts) Considereu l'esquema de la base de dades següent. Emmagatzema dades de metges i malalts. Cada visita representa una visita que un malalt fa a un metge en una data determinada (instant-visita). Cada recepta representa una recepta d'un medicament feta en una determinada visita.

```
create table METGES (
num_met integer primary key,
nom_met  char(30),
especialitat char(30),
adreca char(30),
tel integer,
sou integer);

create table MALALTS (
num_mal integer primary key,
nom_mal char(30),
adreca char(30),
dni integer unique);

create table VISITES (
num_met integer references METGES(num_met),
num_mal integer references MALALTS(num_mal),
instant_visita integer,
import  integer,
primary key  (num_met, num_mal, instant_visita));

create table RECEPTES (
num_recepta integer,
medicament char(50),
num_met integer,
num_mal integer,
instant_visita integer,
primary key (num_recepta),
foreign key  (num_met, num_mal, instant_visita) references VISITES
(num_met, num_mal, instant_visita));
```

1.1) Escriviu una sentència SQL per obtenir el número dels metges que han receptat alguna vegada el medicament "Omeprazol" però mai li han receptat aquest medicament a un malalt que es diu "Pere Pons".

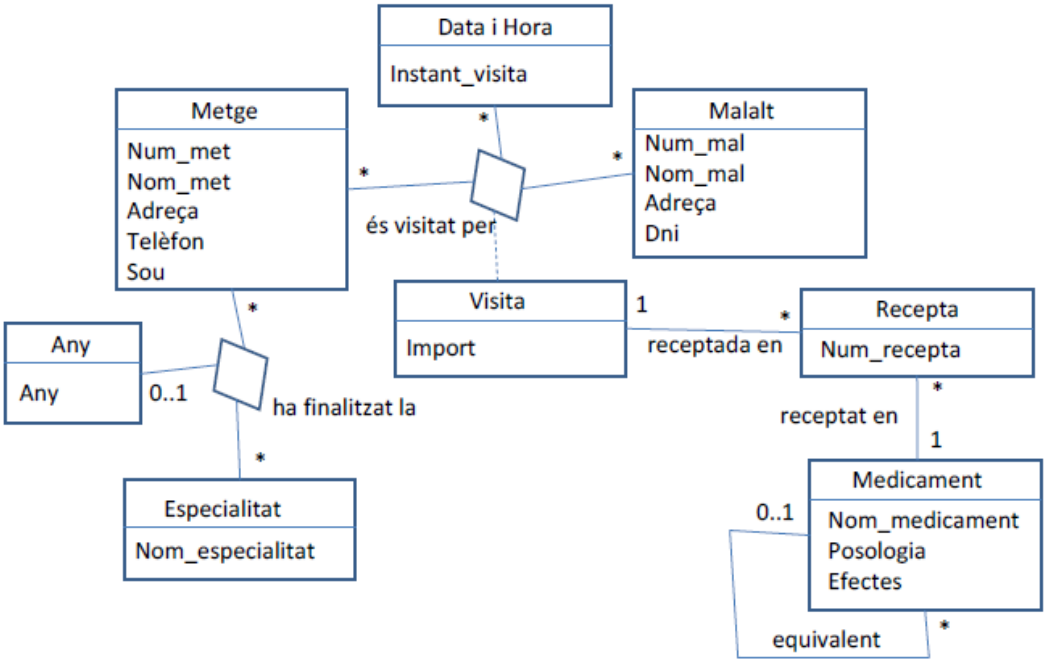
```
SELECT me.num_met
FROM metges me, receptes re, malalts ma
WHERE me.num_met = re.num_met AND re.num_mal = ma.num_mal
        AND EXISTS(SELECT *
                        FROM metges me, receptes re, malalts ma
                        WHERE re.medicament = 'Omeprazol') AND
                                NOT EXISTS (SELECT * FROM metges me, receptes re,
                                                malalts ma
                                        WHERE re.medicament = 'Omeprazol'
                                        AND ma.nom_mal = 'Pere Pons');
```

1.2) Escriviu una sentència SQL per obtenir el nom i especialitat dels metges amb qui s'ha visitat més de 5 vegades el malalt "Joan Grau" i que han receptat alguna vegada a aquest malalt el medicament 'ProCIM".

```
SELECT me.nom_met, me.especialitat
FROM metges me, receptes re, malalts ma
WHERE me.num_met = re.num_met AND re.num_mal = ma.num_mal
        AND ma.nom_mal = 'Joan Grau'
GROUP BY me.num_met, ma.num_mal
HAVING COUNT(*) > 5 AND EXISTS(SELECT *
                               FROM metges me, receptes re, malalts ma
                               WHERE re.medicament = 'ProCIM');
```

1.3) Suposeu que un dissenyador ha dissenyat la base de dades anterior a partir del diagrama UML següent. Indiqueu què falta i/o sobra a l'esquema. Concretament per cada part de l'esquema que falti o sobri cal indicar: les taules, les columnes, i les restriccions que cal afegir o eliminar.

```
Clau externa Metge: Num_met
Clau externa Malalt: Num_mal
Clau externa Especialitat:  Nom_especialitat
Clau externa Any: Any
Clau externa Data i Hora: Instant_visita
Clau externa Recepta: Num_recepta
Clau externa Medicament: Nom_medicament
```



Sobra l'associació "ha finalitzat la" entre metge, especialitat i any, ja que això implicaria una nova taula a la BD "FINALITZACIÓ"; llavors, a banda de l'associació, cal eliminar la classe "Any" (aquest atribut no hi és a la BD) i hem d'inserir "Nom_especialitat" dins de "Metge" com un atribut més (i eliminar la classe "Especialitat").

També sobra la classe “Medicament”, i òbviament l'associació “equivalent”. Hem d'afegir “Nom_medicament” a la classe “Recepta” com a nou atribut i eliminar a la vegada l'associació “receptat en”.

2) (2 punts) Donat l'esquema de la BD següent que emmagatzema els socis (*Socis*) d'un club social. En aquest club s'organitzen activitats de diferents tipus (*Tipus_activitats*). Els socis poden organitzar activitats (*Activitats*) d'un cert tipus que comencen en un cert instant (*instant-inici*). Els socis que ho desitgen s'inscriuen (Inscripcions) per participar en les activitats.

```
Socis(nif, nom, telèfon, ciutat_residència)
Tipus_activitats(tipus_activitat, num_màxim_inscripcions)
Activitats(tipus_activitat, instant_inici, descripció,
nif_organitzador,
preu, ciutat_activitat)
{tipus_activitat} clau forana referència Tipus_activitats.
{nif_organitzador} clau forana referència Socis.
Inscripcions(tipus_activitat, instant_inici, nif, forma_pagament,
instant_pagament)
{tipus_activitat, instant_inici} clau forana referència Activitats.
{nif} clau forana referència Socis.
```

2.1) Digueu quin mecanisme creieu millor per a implementar cada una de les restriccions següents en PostgreSQL. Justifiqueu breument la vostra resposta. Per les situacions en que decidiu usar disparadors, i indiqueu els esdeveniments que els activaran (INSERT/UPDATE/DELETE sobre nom_taula). Si un d'aquests esdeveniments és l'UPDATE indiqueu quin/s atribut/s són rellevants per al compliment de la restricció.

a) El nombre d'inscripcions a una activitat no pot superar el num_màxim d'inscripcions del tipus al que correspon l'activitat.

Mitjançant triggers sobre inscripcions mirem si les inscripcions a l'activitat en qüestió son més grans que el màxim del tipus d'activitat. Quan:

- INSERT INTO INSCRIPCIONS (BEFORE/FOR EACH ROW): Per cada nova fila recalculem el total de l'activitat i comparem amb el màxim. Si és més gran no executem (return null).
- UPDATE INSCRIPCIONS(tipus_activitat, instant_inici) (BEFORE/FOR EACH ROW): Si canviem l'activitat a la que s'ha inscrit algú, hem de mirar que no es traspassi a una activitat que tingui el màxim del tipus d'activitat permès; si es així es fa return old.

b) Si la forma_pagament és "EFECTIU" l'instant de pagament ha de ser diferent de NULL.

Amb un check de taula al es pot solucionar.

c) Els socis només es poden inscriure en activitats que es realitzen a la mateixa ciutat on resideixen.

Mitjançant triggers sobre inscripcions mirem que la ciutat de l'activitat on s'inscriu un soci sigui igual a la seva ciutat. Quan:

- INSERT INTO INSCRIPCIONS (BEFORE/FOR EACH ROW): Per cada nova inscripció comprovem. Si no és així return null.
- UPDATE INSCRIPCIONS(tipus_activitat, instant_inici) (BEFORE/FOR EACH ROW): Abans de modificar una una activitat d'un soci hem de mirar que sigui de la seva ciutat. Sinó return old.

- UPDATE ACTIVITAT(ciutat_activitat) (BEFORE/FOR EACH ROW): No podem permetre updates sobre activitats amb socis inscrits, perquè trencaríem la norma. Si la activitat té socis, return old. També es podrien eliminar totes les inscripcions i llavors fer l'update.
- UPDATE SOCIS(ciutat_residència) (BEFORE/FOR EACH ROW): No podem permetre updates sobre ciutats de socis que tinguin inscripcions, perquè trencaríem la norma. Si el soci té activitats, return old. També es podrien eliminar les inscripcions dels socis que es canviïn de ciutat.

2.2) Doneu la implementació que faríeu per a la restricció b de l'apartat anterior.

```
CHECK ((forma_pagament='EFECTIU' AND instant_pagament<>NULL) OR (forma_pagament <> 'EFECTIU') );
```

2.3) Suposeu ara la restricció: "Només es poden esborrar activitats de la base de dades si tots els socis de la taula socis han organitzat ja alguna activitat". Doneu la implementació que faríeu d'aquesta restricció mitjançant disparadors. En cas que algun esdeveniment intenti violar la restricció cal que la implementació ho eviti i mostri el missatge següent: 'No es poden esborrar activitats. Encara hi ha socis que no han organitzat cap activitat'. Tingueu en compte que les restriccions d'integritat definides a la BD (primary key, foreign key, check...) es violen amb menys freqüència que la restricció comprovada per aquest disparador.

```
CREATE FUNCTION myfunc() RETURNS trigger AS $$
BEGIN
        IF ( (SELECT COUNT(DISTINCT nif_organitzador) FROM inscripcions) != (SELECT COUNT(*) FROM socis) ) THEN
                RAISE EXCEPTION ''No es poden esborrar activitats. Encara hi ha socis que no han organitzat cap activitat.';
RETURN NULL;
END;

CREATE TRIGGER supertrigger BEFORE DELETE FOR EACH ROW ON inscripcions
EXECUTE PROCEDURE myfunc();
```

3) (1.5 punts)

3.1) Considereu que les relacions R(A, B) i S(A, B) tenen exactament el mateix esquema. Per cadascuna de les següents igualtats: digueu si són certes o falses. Si són certes poseu un exemple en què es compleix. Si són falses doneu un contraexemple.

a) $R \cap S = R - (R - S)$ b) $R \cap S = R * S$
c) $(R \cap S)[A] = R[A] \cap S[A]$ d) $(R \cup S)[A] = R[A] \cup S[A]$

a) Cert. Suposem R = {(1,1),(2,2),(3,3)} y S={(1,1),(3,3)}.
La diferència R-S és {(2,2)}, i R^S={(1,1),(2,2)} = {(1,1),(2,2),(3,3)} – {(2,2)} = R – (R-S).

b) Fals. Suposem que la clau primària és A (hauria de ser o A o B). Suposem R = {(1,1),(2,2)} y S={(1,2),(2,3)}. La join implícita seria R*S={(1,1,2),(2,2,3)}, mentre que R^S={}.

c) Fals. Suposem R={(1,2),(2,1)} i S={(2,2),(1,1)}.
Llavors (R^S)[A] = {}, perquè no hi ha cap tupla idèntica entre elles, i R[A]^S[A]={(1),(2)}, perquè si hi han As iguals entre elles.

d) Cert. Suposem R={(1,1),(2,2)} i S={(3,3),(4,4)}. Llavors (RVS)[A]={(1),(2),(3),(4)} i R[A]VS[A] = {(1),(2),(3),(4)}.

3.2) Considereu que, inicialment, l'usuari A és el propietari de la relació R, i cap altre usuari en té privilegis. Digueu quin és exactament el conjunt d'usuaris que tenen el privilegi d'INSERT sobre R després de l'execució de les instruccions que podeu trobar a continuació

```
A: GRANT INSERT ON R TO B WITH GRANT OPTION;
B: GRANT INSERT ON R TO C WITH GRANT OPTION;
C: GRANT INSERT ON R TO D WITH GRANT OPTION;
D: GRANT INSERT ON R TO B WITH GRANT OPTION;
B: REVOKE INSERT ON R FROM C CASCADE;
```

A i B (no s'ha eliminat el privilegi concedit per A sobre B).

3.3) Donades les mateixes taules de l'exercici 2 amb les següents extensions:

Socis(nif, nom, telèfon)

nif	nom	telèfon
1	Joan	4003
2	Pere	4004

Tipus_activitats(tipus_activitat, num_màxim_inscripcions)

tipus_activitat	num_màxim_inscripcions
Pàdle	10
Tennis	15

Activitats(tipus_activitat, instant_inici, descripció, nif_organitzador, preu)

tipus_activitat	instant_inici	descripció	nif_organitzador	preu
Pàdle	1	curs	1	100
Pàdle	10	torneig	1	50
Pàdle	15	torneig	2	50

Inscripcions(tipus_activitat,instant_inici,nif, forma_pagament, instant_pagament)

tipus_activitat	instant_inici	nif	forma_pagament	instant_pagament
Pàdle	1	1	visa	10
Pàdle	1	2	visa	10

Suposant que *Activitats.nif_organitzador* s'ha definit com "ON DELETE SET NULL" i *Inscripcions.nif* s'ha definit com "ON DELETE CASCADE", i suposant que s'executa una sentència d'esborrat del soci 1 de la taula *Socis*. Digues:

a) Quina serà l'extensió de la taula SOCIS després de l'execució de la sentència?

Socis(nif, nom, telefon)
2 Pere 4004

b) Quina serà l'extensió de la taula ACTIVITATS després de l'execució de la sentència?

Activitats(tipus_activitat, instant_inici, descripció, nif_organitzador, preu)

Padle	1	curs	NULL	100
Padle	10	torneig	NULL	50
Padle	15	torneig	2	50

c) Quina serà l'extensió de la taula INSCRIPCIONS després de l'execució de la sentència?

Inscripcions(tipus_activitat,instant_inici, nif, forma_pagament, instant_pagament)

Padle	1	2	visa	10

d) Quina seria la diferència si Inscripcions.nif s'hagués definit "ON DELETE SET NULL"?

Violació de les regles d'integritat (nif és part de la clau primària d'inscripcions).

4) (2 punts) Donades les dues taules següents que guarden, respectivament, dades de departaments i dades d'empleats d'aquests departaments:

Departaments		
codi_dept	**nom_dept**	**pressupost**
1	RRHH	100000
2	Marketing	40000
3	Finances	20000

Empleats			
codi_empl	**nom_empl**	**sou**	**codi_dept (referencia Departaments)**
1	Joan Roca	1500	1
2	Maria Puig	2000	2
3	Anna Rius	2500	2
4	Marc Barà	2000	3
5	Lluís Pérez	1700	3

4.1) Suposeu que el gränul és la fila, que no existeix cap mecanisme per al control de concurrència i que es produeix l'horari següent (les operacions s'han numerat per facilitar la seva referència):

num. op.	T1	T2	T3
1		select * from empleats where codi_dept=2	
2	update empleats set codi_dept=2 where codi_empl=4		
3		select sum(sou) from empleats where codi_dept=2	
4			select * from departaments where codi_dept in (1,3)
5	update departaments set pressupost=pressupost*1.10 where codi_dept=1		
6			select * from departaments where codi_dept=1
7		select sum(pressupost) from departaments	
8			update departaments set pressupost=pressupost*1.10 where codi_dept=3
9			commit
10		commit	
11	commit		

a) Digueu si existeixen interferències en l'horari. Si la resposta és afirmativa, doneu el nom de la/es interferència/es, gràNul i transaccions implicades.

- **Fantasma:** Entre T1 y T2. A 1, T2 llegeix tots els empleats del dpt 2, després T1 modifica aquesta quantitat d'empleats afegint un nou empleat al dpt2, i després T2, amb una informació llegida prèviament incorrecta, suma els sous dels empleats del dpt2 (hi ha un més ara i T2 no ho sap).
- **Lectura no repetible:** Entre T1 y T3. A 4, T3 llegeix la informació dels departaments 1 y 3, a 5 T1 modifica la informació del departament1 y a 6 T3 torna a llegir la informació de 1, que ha canviat.
- **Lectura no confirmada:** Entre T2 y T3. A 7, T2 llegeix la suma dels pressupostos a departaments, i sense abans de confirmar, T3 modifica aquesta xifra modificant el pressupost del departament3.

b) Digueu quins horaris serials donen resultats equivalents a l'horari proposat.

Cap. L'horari no és seriable.

c) Indiqueu, en cas que hi hagi interferències, quin és el nivell mínim d'aïllament necessari per evitar cadascuna d'elles.

T2 hauria de tenir mínim SERIALIZABLE i T3 REPETEABLE READ.

d) Digueu si l'horari proposat, és recuperable. Justifiqueu breument la vostra resposta.

No, perquè per exemple, T3, que sobreescriu el que va llegir T2, confirma abans que T2.

4.2) Suposeu que el gràNul és la fila, que tenim un mecanisme de control de concurrència basat en reserves S, X i que totes les transaccions treballen a un nivell d'aïllament de SERIALIZABLE, digueu quines sumes poden ser produïdes per la transacció T2 (ens referim a la segona i tercera sentència SQL que executa T2).

Primera suma: 2000+2500=4500 (ens protegim contra el fantasma de T1).
Segona suma: 110000+40000+20000=170000 (tenim en compte la modificació de departaments de T1).

5) (2 punts) Tenim una taula amb 200000 empleats, que està emmagatzemada en pàgines en les que hi ha un promig de 10 tuples per pàgina. Els números d'empleat van del 1 al 200000. Hi ha un índex definit sobre l'atribut numEmpl (12 bytes) d'aquesta taula que està implementat com un arbre B+. Un apuntador a node de l'arbre ocupa 3 bytes. Un RID ocupa 4 bytes. Les pàgines de l'índex estan plenes a un 80%. Les pàgines tant de l'índex com de dades són de 4K.

5.1) Determineu l'ordre òptim de l'arbre B+ descrit.

Nodes interns: 2d*12+(2d+1)*3<=4096 → d<=136
Nodes fulla: 2*3+2d*(12+4) <= 4096 → d<=127
Ordre òptim: 127.

5.2) Suposeu per aquest apartat que l'ordre de l'arbre B+ és d=100. Determineu la quantitat de pàgines necessàries per a emmagatzemar l'índex.

(200000/160)+(2500/161)+1 = 1267 pàgines.

5.3) Suposeu per aquest apartat que l'ordre de l'arbre B+ és d=100. Indiqueu quina quantitat de pàgines (d'índex i de dades) cal accedir per resoldre la consulta següent, suposant primer que l'índex és agrupat, i després que és no agrupat. SELECT * FROM empleats WHERE numEmpl BETWEEN 10000 AND 20000.

- **Agrupat:**
  Accedim a 10002 elements. Baixem fins a l'últim nivell (2 accessos), i desde la tupla 10000 accedim a totes les altres seqüencialment directament desde disc. 2+10000/10= 1002 accessos.

- **No agrupat:**
  Baixem fins a l'últim nivell (3 accessos), però ara iterem desde els nodes fulla. Per tant 3+10002/80(nodes a recòrrer)+10002(accessos a fitxer)= 10131 accessos.

5.4) Suposeu per aquest apartat que l'ordre de l'arbre B+ és d=100. Indiqueu quina quantitat de pàgines (d'índex i de dades) cal accedir per resoldre la consulta següent, suposant primer que l'índex és agrupat, i després que és no agrupat. SELECT count(*) FROM empleats WHERE numEmpl BETWEEN 10000 AND 20000.

Igual que a 5.3, hem de fer un recorregut seqüencial igualment.

Examen 8 de juny de 2011

1) (2 punts) Considereu l'esquema de la base de dades següent:

```
create table especies  (
especie char(20) primary key,
comestible char(1) not null check ((comestible='S') or
(comestible='N')),  valor integer not null check (valor >= 0 and valor
<= 10),
check ((comestible = 'S') or (valor = 0)));
 -- En aquesta taula hi ha una fila per cada espècie de bolets.

create table boletaires  (
nom char(20) primary key,
numLlicencia integer unique,
ciutat char(20),
edat integer);
-- En aquesta taula hi ha una fila per cada caçador de bolets

create table troballes  (
nom char(20),
especie char(20),
jornada integer,
quantitat integer check (quantitat > 0),
lloc char(20),
primary key (nom, especie, jornada),
foreign key (nom) references boletaires,
foreign key (especie) references especies);
-- En aquesta taula hi ha una fila per cada espècie de bolets que
troba un boletaire en una jornada.
```

1.1) Escriviu una sentència SQL per obtenir el número de llicència dels boletaires que han trobat més de 15 espècies en una mateixa jornada i que entre totes les seves troballes hi ha troballes de més de 2 espècies diferents amb quantitat trobada més gran que 10.

```
SELECT DISTINCT b.numLlicencia
FROM boletaires b, troballes t
WHERE b.nom = t.nom AND 2 < (SELECT COUNT(DISTINCT t.especie)
                             FROM boletaires b, troballes t
                             WHERE t.quantitat > 10)
GROUP BY b.nom, t.jornada
HAVING COUNT(DISTINCT t.especie) > 15;
```

1.2) Escriviu una seqüència d'operacions d'àlgebra relacional per obtenir els noms dels boletaires que no han participat a la jornada 14 i que no han fet mai cap troballa d'una espècie de bolets amb valor més gran que 8.

```
A = troballes(jornada=14)
B = A[nom]
C = boletaires[nom]
D = C - B
E = especies(valor > 8)
F = E * troballes
G = F[nom]
R = F – D
```

1.3) Escriviu una sentència SQL per definir una vista que doni els noms dels boletaires que no tenen cap troballa a la taula troballes. El nom de la vista ha de ser: 'boletairessensetroballes'.

```
CREATE VIEW boletairessensetroballes AS
SELECT b.nom
FROM boletaires b
WHERE NOT EXISTS (SELECT * FROM troballes t
                  WHERE b.nom = t.nom);
```

a) Aquesta vista és actualitzable? Justifiqueu la resposta.

És sempre actualizable perquè no hem afegit WITH CHECK OPTION.

b) Independentment que la vista sigui o no actualitzable, suposem la sentència SQL següent:

```
INSERT INTO boletairessensetroballes VALUES ('Pere');
```

Digues quina/es són la/es traduccions possibles de la sentència anterior sobre la vista a sentències a aplicar sobre taules.

Com afegim un nou boletaire a boletairessensetroballes, vol dir que aquest no ha de tenir troballes, és a dir, la traducció a sentències sobre taules serà:
DELETE FROM troballes WHERE nom = 'Pere'; (si es troba a la BD)
INSERT INTO boletaires values('Pere',...); (si no es troba a la BD)

2) (2 punts) Donades les taules de la base de dades de l'exercici 1:

a) Considereu que la taula especies té un atribut addicional quantitat_boletaires que indica la quantitat total de boletaires de Barcelona que han trobat aquella espècie. Digueu quines sentències d'actualització (inserció, esborrat i modificació) de les taules troballes i boletaires són rellevants pel manteniment automàtic d'aquest atribut derivat. En el cas de les sentències de modificació digueu quin/s és/són els atributs rellevants.

Troballes:
- Insert d'una troballa d'un boletaire de Barcelona que no tenia troballes d'aquella espècie.
- Delete de la última troballa d'una espècie d'un boletaire de Barcelona.
- Update de l'espècie d'una troballa.

Boletaires:
- Delete d'un boletaire de Barcelona amb troballes.
- Update de la ciutat d'un Boletaire amb troballes.

b) Doneu la implementació d'un trigger associat a l'esdeveniment update de l'atribut edat de la taula *boletaires*, de tal manera que si es vol abaixar el valor d'aquest atribut es generi una excepció. Considereu que les restriccions d'integritat definides a la BD (primary key, foreign key...) es violen amb menys freqüència que la restricció comprovada per aquest trigger.

```
CREATE or REPLACE FUNCTION func() RETURNS trigger AS $$
BEGIN
        IF (NEW.edat < OLD.edat) THEN RAISE EXCEPTION 'No es pot abaixar l'edat';
        RETURN NEW;
END;

CREATE TRIGGER mi_trigger BEFORE UPDATE OF edat ON boletaires FOR EACH ROW
EXECUTE PROCEDURE func();
```

c) Suposeu que heu d'implementar amb triggers la restricció d'integritat: "Només es poden esborrar troballes durant la tardor".

c.1) Digueu si aquest trigger ha de ser FOR EACH STATEMENT o FOR EACH ROW. Justifiqueu la resposta.

FOR EACH STATEMENT. Si és tardor farem el/s delete/s, sinó no. No cal comprovar-ho tupla per tupla.

c.2) Digueu si aquest trigger ha de ser BEFORE o AFTER. Justifiqueu la resposta.

BEFORE. Abans d'eliminar res hem de mirar si és tardor o no.

d) Suposeu que heu d'implementar un trigger per tal que cada vegada que s'executi una sentència update que modifiqui l'atribut valor d'una o més especies, s'insereixi una única fila a la taula auditoria_valor. Els atributs de la taula auditoria_valor són: usuari que ha fet les modificacions dels valors, instant en què s'han fet.

d.1) Digueu si aquest trigger ha de ser FOR EACH STATEMENT o FOR EACH ROW. Justifiqueu la resposta.

FOR EACH STATEMENT, s'insereix una única fila a la taula auditoria_valor.

d.2) Digueu si aquest trigger ha de ser BEFORE o AFTER. Justifiqueu la resposta.

AFTER, perquè en cas que hi hagués algun error amb l'update no s'afegiria la nova fila.

3) (2 punts) Donada la taula següent de la BD de boletaires:

create table boletaires(

```
nom char(20) primary key,
numLlicencia integer unique,
ciutat char(20),
edat integer check (edat>=18));
```

El SGBD que conté la BD té un mecanisme de control de concurrència amb reserves on el grànul és la fila. Es vol executar una transacció, anomenada T1. T1 incorpora les sentències SQL que es mostren a continuació. També es mostra el resultat de l'execució d'aquestes sentències.

T1	Resultat execució

`select *`
`from boletaires`
`where nom='Anna Cases'`
	S'obté les dades de l'Anna Cases:
	`(Anna Cases,125,Blanes,20)`

`update boletaires`
`set ciutat='Mataró'`
`where nom='Anna Cases'`
	Es canvia la ciutat de residència de l'Anna Cases de Blanes a Mataró

`COMMIT`	

a) Suposeu que la resta de transaccions que s'executen concurrentment amb T1 són transaccions READ ONLY, digueu quin és el nivell mínim d'aïllament amb el que ha de treballar T1 per tal de garantir que la resta de transaccions no puguin causar interferències en T1. Cal argumentar la resposta.

Si les altres transaccions només llegeixen, mai causaran interferències sobre T1, per tant no cal cap nivell d'aïllament.

b) Suposeu ara que la transacció T1 treballa amb un nivell d'aïllament SERIALIZABLE i que totes les transaccions READ ONLY treballen amb un nivell d'aïllament de READ COMMITTED. Quines interferències es poden produir? Per cada possible interferència, cal donar: a) El nom de la interferència i els grànuls implicats; b) El nivell mínim d'aïllament per evitar la interferència; c) Un exemple d'horari amb 2 transaccions (la transacció T1 i un exemple de transacció READ ONLY) expressat en termes de sentències SQL que il·lustri la interferència que es produeix.

- LECTURA NO REPETIBLE sobre la tupla "Anna Cases". Per evitar-la caldria que les transaccions READ ONLY tinguessin nivell d'aïllament REPETEABLE READ.

T1	Tx
select * from boletaires where nom='Anna Cases'	
	select * from boletaires where nom='Anna Cases'
update boletaires set ciutat='Mataró' where nom='Anna Cases'	
COMMIT	
	select * from boletaires where nom='Anna Cases'
	COMMIT

- ELEMENT FANTASMA. Si llegim informació sobre tuples amb ciutat='Mataró', després de canviar la ciutat de 'Anna Cases', aquest seria el nostre element fantasma. El nivell d'aïllament de les transaccions READ ONLY hauria de ser SERIALIZABLE.

T1	Tx
select * from boletaires where nom=’Anna Cases’	
	Select SUM(edat) from boletaires where ciutat = 'Mataró'
update boletaires set ciutat=’Mataró’ where nom=’Anna Cases’	
COMMIT	
	Select SUM(edat) from boletaires where ciutat = 'Mataró'
	COMMIT

c) Suposeu ara que la transacció T1 treballa amb un nivell d’aïllament SERIALIZABLE, però en comptes d’acabar la seva execució amb COMMIT, l’acaba amb un ROLLBACK. Totes les transaccions READ ONLY treballen amb un nivell de READ COMMITTED. En aquest cas, es pot produir alguna interferència addicional, a més de les interferències indicades a l’apartat anterior? En cas de resposta negativa cal que argumenteu la resposta. En cas de resposta afirmativa, cal que doneu un horari d’exemple.

No es pot produir cap més interferència, perquè les altres transaccions treballen amb READ COMMITTED i això les protegeix de Lectura No Confirmada (i actualització perduda, però com que no actualitzen no es té en compte). En READ COMMITTED, abans de llegir cap gränul, es fa un lock d'aquest, i com que els locks d'actualització d'altres transaccions (en aquest cas de T1) es mantenen fins al final de la transacció, la transacció Tx no podrà llegir la dada fins que T1 acabi (en aquest cas amb ROLLBACK, i Tx llegiria el que hi havia al gränul abans de T1).

d) Suposeu ara que es produeix un canvi en la política d'accés a la BD de boletaires, de tal manera que només es permet l'execució de transaccions READ ONLY. En aquestes circumstàncies: a) Cal disposar d'un dietari (o log) de la BD? b) Cal fer còpies de seguretat (bolcats o backups) de la BD? Argumenteu les respostes.

No cal cap log de la BD, ja que aquest només té sentit quan es modifica algun gràNul de la BD. Un backup sí, perquè es podrien perdre les dades i s'haurien de poder recuperar d'alguna manera.

4) (2 punts) Considereu una taula de productes (esquema: Prod(pid, categoria, preu,...)) amb 600000 productes, que està emmagatzemada en pàgines en les que hi ha un promig de 10 tuples per pàgina. Els números de productes (pids) estan distribuïts uniformement entre 1 i 600000. El pid del producte és la clau primària de la taula de productes i s'ha definit un índex B+ no agrupat sobre aquest atribut. Les pàgines tant de l'índex com de dades són de 2K.

a) Calculeu el nombre mínim i màxim de pàgines que ocupa l'índex suposant que l'ordre de l'índex és d=100. Justifiqueu breument la resposta.

Màxim: Si cada node té 100 RIDS.
(600000/100)+(6000/101)+(60/101)= 6061

Mínim: Si cada node té 200 RIDS.
(600000/200)+(3000/201)+(15/201)= 3016

b) Indiqueu a quina quantitat de pàgines (d'índex i de dades) cal accedir per resoldre les consultes següents, suposant que els nodes de l'arbre estan plens al 50% de la seva capacitat. Justifiqueu breument la resposta.

```
SELECT * FROM prod WHERE pid BETWEEN 20000 AND 30000
```

Hem de recórrer els RIDS de l'últim nivell de 20000 a 30000 seqüencialment, ja que no està agrupat.

3(baixar a últim nivell) + 10002/100 (nombre de nodes a recórrer) + 10002 (accedir a les 10002 tuples) = 10106.

```
SELECT * FROM prod WHERE pid<>20000
```

Seleccionem totes les tuples excepte una (pid=20000). Tornem a recórrer seqüencialment l'últim nivell, però aquest cop sencer.

3(baixar a últim nivell) + 6000/100(nodes a recórrer) + 59999(accedir a les tuples en qüestió) = 66002

c) Expliqueu breument en quines de les consultes de l'apartat anterior és útil l'ús de l'índex no agrupat sobre el pid per resoldre la consulta.

En la primera consulta l'índex ens facilita l'accés a la tupla 20000 i per tant no hem de recórrer tota la taula fins a trobar-la. En canvi, en la segona, hem de recórrer tota la taula sempre, per tant ens és indiferent utilitzar índex o no.

5) (2 punts)

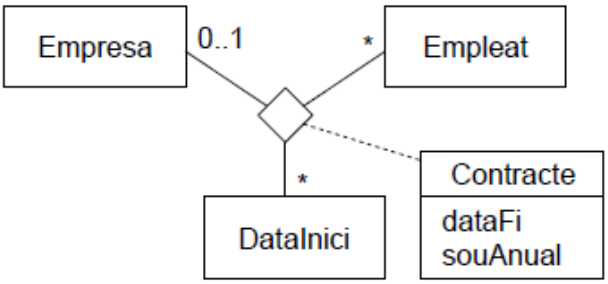
a) Suposeu que sou propietaris de la base de dades de l'exercici 1, i que heu de donar privilegis a l'usuari Anna per tal que l'Anna pugui executar les sentències següents i pugui passar aquests privilegis a altres usuaris. Doneu les sentències SQL necessàries per fer-ho.

```
update especies set valor=valor+5; //necessitem select,update de valor
select especie from especies; //necessitem select d'especies
```

GRANT select(especie,valor) ON especies TO Anna WITH GRANT OPTION;
GRANT update(valor) ON especies TO Anna WITH GRANT OPTION;

b) Feu la traducció a model relacional de la classe associativa Contracte, donant el disseny lògic de la/es taula/es resultant/s.

Clau externa Empresa: NIF
Clau externa Empleat: numEmpl
Clau externa DataInici: dataIni



Empresa(NIF);
Empleat(numEmpl);
DataInici(dataIni);
Contracte(dataIni,numEmpl,NIF,dataFi,souAnual)
dataIni referencia DataInici, numEmpl referencia Empleat, NIF referencia Empresa.

c) Tingueu en compte el mètode consulta JDBC (sobre la base de dades de l'exercici 1) inclòs més endavant. Aquest mètode:

- Implementa l'aplicació d'una penalització de 5 sobre la quantitat de bolets trobada en les troballes d'una determinada espècie i una determinada jornada (passats com a paràmetres d'entrada).
- Retorna la quantitat de troballes que han estat penalitzades, o bé una excepció en cas que alguna de les troballes penalitzades quedi amb una quantitat trobada inferior o igual a 0, o en cas que es produeixi un error intern.
- Utilitza de la manera habitual els objectes d'entrada/sortida que s'han vist a les sessions de laboratori.

Es demana:

c.1) Per cada sentència SQL que s'executa digueu si s'usen adequadament els objectes *Statement* i *PreparedStatement*. Justifica la resposta.

> El primer Statement si s'usa correctament, perquè només s'executa una vegada amb un executeQuery i no caldria cap PreparedStatement.
>
> El segon, encara que un PreparedStatement és l'objecte adequat per la consulta, al codi s'està sobreescrivint l'objecte a cada iteració, i perd tot el sentit utilitzar el preparedStatement; s'hauria de crear abans del bucle i simplement anar executant un executeUpdate() sobre l'objecte a cada iteració.

c.2) Penseu en una implementació alternativa que doni els mateixos resultats però que faci menys accessos a la base de dades, i responeu als apartats següents:

- Doneu les sentències SQL que s'hi executarien
- Expliqueu com es calcularia el número de troballes penalitzades.
- Expliqueu com s'identificaria l'excepció de que alguna de les troballes penalitzades queda amb quantitat inferior a 0.

```
public ConjuntTuples consulta(Connection c, Tuple params) throws BDException {
try {
  String especiePenalitzada = params.consulta(1);
  String jornadaPenalitzada = params.consulta(2);
  Statement s = c.createStatement();
  ResultSet troballes = s.executeQuery("select tr.nom,tr.quantitat "+
                                "from troballes tr "+
                                "where tr.especie='"+especiePenalitzada+"' "+
                                "and tr.jornada ="+jornadaPenalitzada+";");
  String boletaire = null;
  int quantitatSensePenalitzacio = 0;
  int numPenalitzades = 0;
  while (troballes.next())
  { boletaire=troballes.getString("nom");
    quantitatSensePenalitzacio=troballes.getInt("quantitat");
    PreparedStatement penal = c.prepareStatement("update troballes tr "+
                                "set quantitat = ? - 5 "+
                                "where tr.especie = ? and tr.jornada =? "+
                                "and tr.nom = ?");
    if (quantitatSensePenalitzacio <= 5) throw new BDException(10);
    penal.setInt(1, quantitatSensePenalitzacio);
    penal.setString(2, especiePenalitzada);
    penal.setInt(3, Integer.parseInt(jornadaPenalitzada));
    penal.setString(4, boletaire);
    penal.executeUpdate();
    numPenalitzades++;
  }
  ConjuntTuples ct = new ConjuntTuples();
  Tuple t = new Tuple();
  t.afegir(String.valueOf(numPenalitzades));
  ct.afegir(t);
  return ct;
  }
  catch(SQLException e) {throw new BDException(11);}
}
```

Simplement es pot fer l'update directament amb una única sentència:

```
UPDATE troballes tr
SET quantitat = quantitat – 5
WHERE tr.especie = especiePenalitzada AND
      tr.jornada = jornadaPenalitzada;
```

Per saber el nombre de troballes penalitzades, després del update s'executa la següent sentència amb un Statement:

```
SELECT COUNT(*)
FROM troballes
WHERE especie = especiePenalitzada AND
      jornada = jornadaPenalitzada;
```

Com quantitat té un check que verifica que sigui >0, un update que deixés quantitat=<0 provocaria una excepció de la BD; per tant, només caldria controlar aquesta excepció en concret per saber si alguna quantitat ha quedat amb =<0.

Examen 18 de gener de 2012

1) (2 punts) Considereu l'esquema de la base de dades següent:

```
CREATE TABLE clients (
dni char(9),
nomClient char(50) UNIQUE NOT NULL,
ciutatResidencia char(15),
PRIMARY KEY (dni));
-- Hi ha una fila per cada client d'una entitat bancària.

CREATE TABLE comptesBancaris (
numCompte char(16),
dniClient char(9) NOT NULL,
saldoDisponible real,
PRIMARY KEY (numCompte),
FOREIGN KEY (dniClient) REFERENCES clients(dni));
-- Hi ha una fila per cada compte bancari d'una entitat. El
saldoDisponible és el saldo del compte bancari un cop aplicats els
ingressos i reintegraments.

CREATE TABLE ingressos (
numCompte char(16),
 instantIngres integer,
quantitat integer NOT NULL CHECK (quantitat>0),
PRIMARY KEY (numCompte, instantIngres),
FOREIGN KEY (numCompte) REFERENCES comptesBancaris(numCompte));
 -- Hi ha una fila per cada ingrés fet al compte bancari.

CREATE TABLE reintegraments (
numCompte char(16),
instantReintegrament integer,
quantitat integer NOT NULL CHECK (quantitat>0),
PRIMARY KEY (numCompte, instantReintegrament),
FOREIGN KEY (numCompte) REFERENCES comptesBancaris(numCompte));
-- Hi ha una fila per cada reintegrament fet al compte bancari.
```

1.1) Escriviu una sentència SQL per obtenir el nom dels clients que tenen un o més comptes amb saldo disponible negatiu i cap ingrés.

```
SELECT DISTINCT c.nomClient
FROM clients c, comptesBancaris co
WHERE c.dni = co.dniClient AND co.saldoDisponible < 0 AND
        NOT EXISTS (SELECT *
                        FROM ingressos i
                        WHERE i.numCompte = co.numCompte);
```

1.2) Escriviu una sentència SQL per obtenir el dni i el nom dels clients que tenen algun compte bancari en el que s'han ingressat més de 50000 euros entre l'instant 1000 i el 2000, i en el que no s'ha fet cap reintegrament en el mateix període.

```
SELECT c.dni, c.nomClient
FROM clients c, comptesBancaris co
WHERE c.dni = co.dniClient AND 50000 < (SELECT SUM(i.quantitat)
                                        FROM ingressos i
                                        WHERE i.numCompte = co.numCompte AND
                                        i.instantIngres BETWEEN 1000 AND 2000)
      AND NOT EXISTS (SELECT *
                      FROM reintegraments r
                      WHERE r.numCompte = co.numCompte
                            AND r.instantReintegrament BETWEEN 1000
                            AND 2000);
```

1.3) Es vol obtenir els saldos dels comptes bancaris a l'instant 1000. Concretament es vol per cada número de compte, el saldo disponible a l'instant 1000. Per fer-ho algú ha implementat aquesta vista:

```
CREATE VIEW saldos1000p (num,saldo) AS SELECT cb.numCompte,
cb.saldoDisponible –(sum(i.quantitat) - sum(r.quantitat))   FROM
comptesBancaris cb, ingressos i, reintegraments r WHERE
cb.numCompte=i.numCompte and i.instantIngres>1000 and     cb.numCompte
= r.numCompte and r.instantReintegrament>1000 GROUP BY cb.numCompte,
cb.saldoDisponible
```

Doneu una extensió de les taules de la base de dades i de la vista que demostrin que la implementació és incorrecta. Raoneu la resposta.

Hi ha error perquè en el cas que un compte no hagi fet un reintegrament, aquest no s'afegirà a la vista.

2) (2 punts) Considereu l'esquema de la base de dades següent:

```
CREATE TABLE Departaments (
num_dpt int primary key,
pressupost int not null check (pressupost>0));
CREATE TABLE Empleats (
num_empl int primary key,
sou int not null check (sou>=0),
num_empl_cap int references Empleats,
num_dpt int not null references Departaments);

CREATE or REPLACE FUNCTION pr_primer() RETURNS trigger AS $$
BEGIN
      UPDATE Empleats SET sou=3000 WHERE sou=new.pressupost;
      RETURN null;
END;
$$LANGUAGE plpgsql;

CREATE TRIGGER primer   AFTER INSERT on Departaments  FOR EACH ROW
EXECUTE PROCEDURE pr_primer();
```

```
CREATE or REPLACE FUNCTION pr_segon() RETURNS trigger AS $$
BEGIN
      UPDATE departaments SET pressupost=pressupost+500;
      RETURN null;
END;
$$LANGUAGE plpgsql;

CREATE TRIGGER segon  AFTER UPDATE on Empleats  FOR EACH ROW EXECUTE
PROCEDURE pr_segon();
```

Es demana:

a) Suposeu que el contingut inicial de la base de dades és el següent:

Departaments(num_dpt,pressupost)	(1,3000)		
Empleats(num_empl, sou, num_empl_cap, num_dpt)	(1,1000,null,1)	(2,2000,1,1)	(3,2000,1,1)

Digueu quin és el contingut final de la base de dades, després de l'execució de la sentència SQL:

```
INSERT INTO Departaments VALUES (2,2000).
```

Justifiqueu breument la resposta, explicant les accions que segueix el SGBD a conseqüència d'aquesta inserció.

Primer es fa el insert a la taula departaments, i després s'executa el primer trigger una vegada (una fila només). El trigger buscarà els empleats amb sou 2000 i els hi modificarà a 3000. Després de cada update, s'executa el segon trigger; per tant s'executarà dues vegades, augmentant el pressupost de l'únic departament en 1000.

Departaments	(1;4000)	(2;2000)	
Empleats	(1;1000;null;1)	(2;3000;1;1)	(3;3000;1;1)

b) Repetiu l'apartat a) suposant que se substitueix la sentència SQL dins el procediment pr_segon per: UPDATE departaments SET pressupost=pressupost-1000. Agafeu com a contingut inicial, de la base de dades, el contingut inicial de l'apartat a).

Ara serà semblant, pero en comptes de sumar 1000 al departament, se li restarà 2000.

Departaments	(1;1000)		
Empleats	(1;1000;null;1)	(2;3000;1;1)	(3;3000;1;1)

c) Definiu una asserció en SQL Standard per garantir el compliment de la restricció següent: "Tot empleat que té un cap, ha de tenir un cap que pertany al departament de l'empleat".

```
CREATE ASSERTION assercio CHECK
(NOT EXISTS(SELECT *
            FROM empleats e1, empleats e2
            WHERE e1.num_empl_cap IS NOT NULL
                  AND e1.num_empl_cap = e2.num_empl
                  AND e2.num_dpt <> e1.num_dpt);
```

d) Expliqueu com implementaríeu l'asserció anterior en PostgreSQL mitjançant disparadors i mitjançant procediments emmagatzemats:

d.1) Per cada disparador cal que indiqueu: l'esdeveniment activador, la taula i el tipus de disparador. En cas de l'esdeveniment UPDATE cal també les columnes rellevants. Per cada disparador, cal també que justifiqueu breument el tipus de disparador escollit.

d.2) Pels procediments emmagatzemats, expliqueu breument la solució proposada.

d.3) Expliqueu un possible avantatge d'implementació de l'asserció mitjançant disparadors, respecte a la seva implementació mitjançant procediments emmagatzemats.

- Insert into Empleats: BEFORE/FOR EACH ROW. Per cada fila que es vol inserir cal veure si el seu cap és del seu mateix departament.
- Update(num_cap_empl) on Empleats: BEFORE/FOR EACH ROW. Per cada fila que es vol modificar cal veure si el nou valor compleix l'asserció.
- Update(num_dpt) on Empleats: BEFORE/FOR EACH ROW. Per cada empleat en el que es vulgui modificar el seu departament, cal veure si es compleix l'asserció amb el nou valor perquè podria ser el cap d'algú.

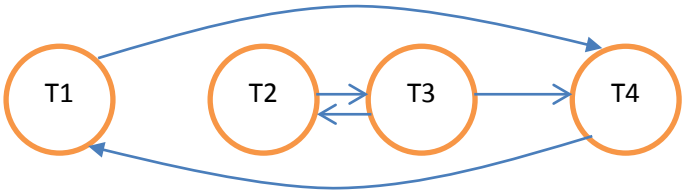
3) (2,5 punts)

a) Sigui un SGBD sense cap mecanisme de control de concurrència, i suposem que es produeix l'horari següent (R= Read, RU= Read for Update, W= Write; les accions s'han numerat per facilitar fer-hi referència):

Acc#	T1	T2	T3	T4
10			R(E)	
20	RU(A)			
30	W(A)			
40			RU(F)	
50			W(F)	
60		RU(E)		
70				R(A)
80				RU(F)
90				W(F)
100		W(E)		
110		R(B)		
120	R(B)			
130		R(C)		
140				COMMIT
150			R(E)	
160	RU(A)			
170	W(A)			
180			COMMIT	
190		COMMIT		
200	COMMIT			

Contesteu, **argumentant les respostes**, a les preguntes següents:

a.1. Quin és el graf de precedències associat a l'horari donat?



a.2. Quines interferències es produeixen? per cada interferència cal que doneu: nom de la interferència, transaccions i grànuls implicats.

Lectura no repetible d'E entre T2 i T3.

a.3. Quins horaris serials donen resultats equivalents a l'horari proposat?

No hi ha horari serial equivalent perquè no és un horari serialitzable (el graf de precedències conté cicles).

a.4. L'horari proposat, és recuperable? En cas afirmatiu doneu la definició de quan un horari és recuperable. En cas negatiu indiqueu alguna de les transaccions i operacions de l'horari que mostrin que no ho és.

L'horari no és recuperable, perquè T4 llegeix A, que prèviament ha modificat T1, i confirma abans de que ho faci T1, trencant el criteri de recuperabilitat.

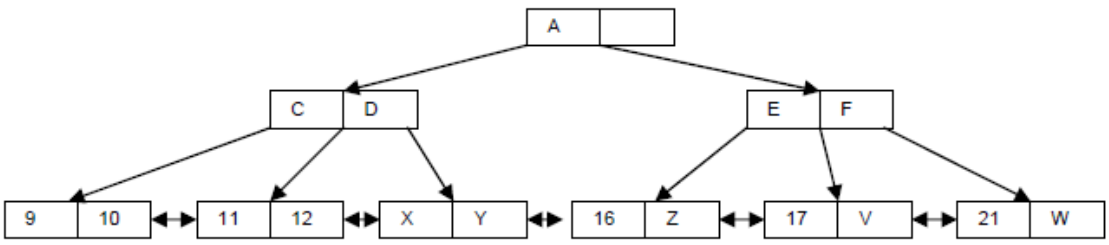
b) Suposeu ara que tenim un mecanisme de control de concurrència basat en reserves S, X i que les transaccions T1, T3 i T4 treballen a un nivell d'aïllament de READ COMMITTED i que T2 treballa amb un nivell d'aïllament de REPEATABLE READ. Contesteu a les preguntes següents:

b.1. Com quedaria l'horari? L'horari ha d'incloure, a més de les peticions que executen les transaccions (R, RU, W, COMMIT), les operacions de petició i alliberament de reserves i l'ordre d'execució de totes aquestes peticions.

b.2. Quins horaris serials hi són equivalents?

No hi ha horari serial equivalent, perquè segueix havent-hi una lectura no repetible a T3.

4) (1,5 punts) Donat l'arbre B+ d'ordre 1 (d=1) que correspon a un índex definit sobre una taula T per a un atribut que és clau primària de la taula:



Es demana:

a) Indiqueu quins són els valors possibles de X, Y, Z, V i W.

- X=13, Y=14 // X=14, Y=15 // X=13,Y=15.
- No té valor.
- V=18//19//20
- 22 o més.

b) Indiqueu quins són els valors possibles de C i D.

- C = 11.
- D = 13//14.

c) Indiqueu quins són els valors possibles de E i F.

- E = 17.
- F = 21.

d) Indiqueu quins són els valors possibles de A.

- A = 17.

Justifiqueu convenientment la resposta de cada apartat.

5) (2 punts) Considereu l'esquema de la base de dades format per les taules següents:

```
Actor(nomactor, adreça, telèfon, anynaix, sexe, religió, caché)
                                        on caché és el que cobra un
                                        actor per fer una pel·lícula
Tema(nomtema)
Pel·lícula(nompel, nomtema)             on nomtema referencia Tema
Habilitat(nomactor, nomtema)            on nomtema referencia Tema
                                        on nomactor referencia Actor
Actuar(nomactor, nompel)                on nomactor referencia Actor
                                        on nompel referencia Pel·lícula
```

a) Escriviu una seqüència d'operacions d'àlgebra relacional per obtenir per cadascun dels actors, els noms de totes les pel·lícules on ha actuat però que no eren d'un tema en que l'actor tenia habilitat. Concretament es demana aquesta informació en parelles: nom actor, nom pel·lícula.

A=Actor[nomActor]
B=A*Actuar //cada actor con sus peliculas
C=B*Pel·licula //Ahora tenemos el tema de cada pelicula
D=C[nomActor=nomActor, nomTema!=nomTema]Habilitat //nos quedamos con las que no tenia habilidad en el tema.
R=D[nomActor,nomPel]

b) A quins atributs afecta la Llei orgànica de protecció de dades personals (LOPD), i en quin/s dels seus tres nivells classificaríeu aquests atributs?

Els atributs de la taula Actor (nomactor, adreça, telèfon, anynaix, sexe, religió, caché).

Nivell bàsic: nomactor, adreça, telèfon, anynaix, sexe.
Nivell mitjà: caché.
Nivell alt: religió.

c) Digues si són certes o falses les sentències següents. Justifica la resposta.

i. Un Servidor en un Entorn SQL agrupa un conjunt de Catàlegs, que en PostgreSQL s'anomenen Esquemes.

Fals. Un catàleg és un conjunt d'esquemes.

ii. L'esquema d'informació és l'esquema per defecte al que es connecta un usuari al fer la connexió amb la base de dades.

Cert.

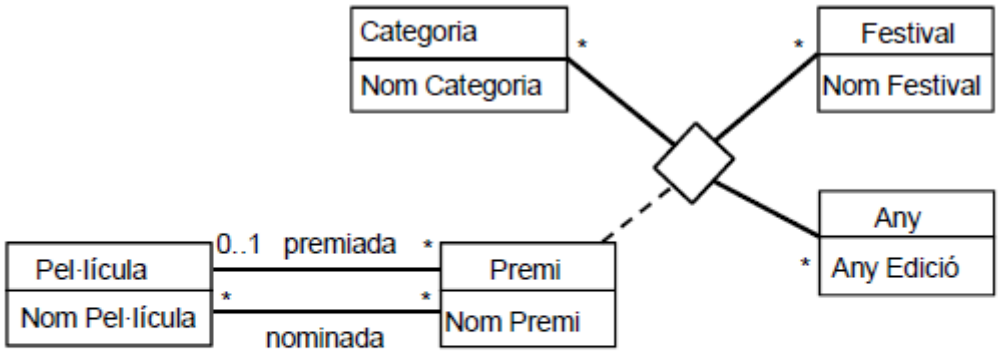
iii. L'esquema de la base de dades que us hem donat és un esquema conceptual segons l'arquitectura de tres nivells ANSI/SPARC.

Cert.

iv. Donat un cert diagrama d'autoritzacions, l'efecte d'una operació REVOKE sempre es pot anul·lar amb una única operació GRANT.

Fals, un revoke amb cascade pot provocar una reacció en cadena fent revoke d'altres permisos donats per l'inicial, per tant no és recuperable amb un únic grant.

d) Suposant el model conceptual en UML següent, dóna el disseny lògic que s'obté fent la traducció a model relacional.



Clau externa Categoria: Nom Categoria
Clau externa Festival: Nom Festival
Clau externa Any: Any Edició
Clau externa Pel·lícula: Nom Pel·lícula

Categoria(Nom Categoria)
Festival(Nom Festival)
Any(Any Edició)
Pel·lícula(Nom Pel·lícula)
Nominada(Nom Categoria, Nom Festival, Any Edicio, Nom Pel·lícula, Nom premi)
Premiada(Nom Categoria, Nom Festival, Any Edicio, Nom Pel·lícula, Nom premi)

on Nom Categoria referencia Categoria,
Nom Festival referencia Festival,
Any Edicio referencia Any,
Nom Pel·lícula referencia Pel·lícula

Examen 26 de juny de 2012

1) (2 punts) Considereu l'esquema de la base de dades següent:

```
create table professors (
dni char(9),
nomProf char(50) unique,
telefon char(15),
sou integer not null check(sou>0),
primary key (dni));

create table despatxos (
modul char(5),
numero char(5),
superficie integer not null check(superficie>12),
primary key (modul,numero));

create table assignacions (
dni char(9),
modul char(5),
numero char(5),
instantInici integer,
instantFi integer,
primary key (dni, modul, numero, instantInici),
foreign key (dni) references professors,
foreign key (modul,numero) references despatxos);
-- assignació d'un professor a un despatx.
-- instantFi te valor null quan una assignacio es encara vigent.
```

1.1) Escriviu una sentència SQL per obtenir la suma del sou dels professors que tenen alguna assignació no vigent a despatxos del mòdul 'Omega' i que te nen 2 o m és assignacions a un mateix mòdul però a despatxos diferents.

```
SELECT SUM(p.sou)
FROM professors p
WHERE EXISTS(SELECT *
                FROM assignacions a
                WHERE a.dni = p.dni AND a.instantFi<>NULL AND a.modul='Omega')
        AND 2<=(SELECT COUNT(DISTINCT a.numero)
                        FROM assignacions a
                        WHERE a.dni = p.dni
                        GROUP BY a.modul);
```

1.2) Suposeu que volem una sentència SQL per trobar els professors (dni, nomProf) que no tenen cap assignació vigent. Digueu, per cadascuna de les sentències següents **si permet obtenir els professors indicats o no**. En cas de que no, **doneu una extensió de la base de dades** i mostreu que la consulta dóna un resultat incorrecte.

a)
```
SELECT p.dni, p.nomProf
FROM professors p
WHERE p.dni NOT IN (SELECT a.dni FROM assignacions a
                    WHERE a.instantFi IS NULL);
```

És vàlida.

b)
```
SELECT p.dni, p.nomProf
FROM professors p
WHERE not exists (SELECT *  FROM assignacions a, professors p1
            WHERE a.dni = p1.dni AND a.instantFi IS NULL);
```

No és vàlida. Suposem que tenim un professor amb una assignació vigent, i uns quants amb no vigent. Llavors la subconsulta trobaria que una tupla a tota la taula (que és el que s'està mirant, perquè s'està fent la join amb una altra taula de professors p2) no compleix la condició, i el not exists donaria fals, i no es mostraria cap professor.

c)
```
SELECT p.dni, p.nomProf
FROM professors p, assignacions a
WHERE p.dni <> a.dni AND a.instantFi IS NULL;
```

No és vàlida. Si tinguéssim dos professors sense assignacions vigents, però algun d'ells tingués una assignació no vigent, amb la sentència miraríem les assignacions a professors diferents (p.dni<>a.dni) i veuríem que instantFi <> NULL, per tant no es mostraria el resultat correcta per algun professor.

d)
```
SELECT p.dni, p.nomProf
FROM professors p, assignacions a
WHERE p.dni = a.dni AND a.instantFi IS NULL
GROUP BY p.dni, p.nomProf
HAVING count(*) = 0;
```

No és vàlida. Si tenim només una assignació a un professor vigent i algun professor amb no vigent, el count ja no donaria 0 i no es retornaria el desitjat.

1.3) Escriviu una seqüència d'operacions d'àlgebra relacional per obtenir els dni dels professors que han estat assignats a un despatx on alguna vegada ha estat assignat el professor amb dni '123'. Tingueu en compte que en el resultat de la consulta no volem que surti el professor '123'.

A = assignacions(dni=123) //totes les assignacions de 123
B = A*despatxos //despatxos d'aquestes assignacions
C = B[modul,numero] //ens quedem amb els despatxos unicament
D = assignacions(dni!=123) //assignacions de tots menys el 123
E = C*D //assignacions a despatxos de 123
R = E[dni]

2) (2 punts)

2.1) Considereu les taules de la base de dades de l'exercici 1, i les vistes següents definides sobre elles:

```
CREATE VIEW personalactualOmega AS
select  dni, modul, numero
from assignacions
where instantFi IS NULL and modul='Omega';

CREATE VIEW dadespersonalactualOmega (nomProf, modul, numero, telefon)
AS
select  p.nomProf, pa.modul, pa.numero, p.telefon
from professors p, personalactualOmega pa
where p.dni=pa.dni;
```

a) Són actualitzables aquestes vistes segons l'estàndard SQL? Justifiqueu la resposta.

No són actualitzables cap de les dues.
En la primera vista, per tal que fos actualitzable, els atributs que formen la primary key han de ser part de la vista; altrament es podrien tenir tuples repetides.
En la segona vista, no és actualitzable perquè no estan inclosos tots els atributs que formen la primary key ni els que tenen la restricció NULL.

b) Considereu la consulta següent:

```
SELECT d1.nomprof, d2.nomprof
FROM dadespersonalactualOmega d1, dadespersonalactualOmega d2
WHERE  d1.modul=d2.modul and
       d1.numero = d2.numero and
       d1.nomprof <> d2.nomprof;
```

b.1) Expliqueu breument què retorna la consulta.

Retorna les parelles de professors que tenen assignada un mateix despatx a l'edifici Omega.

b.2) Doneu una sentència SQL sobre taules, que compleixi els criteris de qualitat establerts a l'assignatura, i que retorni el mateix resultat que la consulta anterior sobre vistes.

SELECT p1.nomProf, p2.nomProf FROM professors p1, professors p2, assignacions a1, assignacions a2 WHERE p1.dni <> p2.dni AND p1.dni = a1.dni AND p2.dni = a2.dni AND a1.modul = 'Omega' AND a2.modul = 'Omega' AND a1.instantFi IS NULL AND a2.instantFi IS NULL AND a1.numero = a2.numero;

c) Si es pot fer una solució que compleix els criteris de qualitat i que permet accedir a les mateixes dades consultant les taules, digueu quines avantatges pot aportar el fer-ho amb vistes.

Fer-ho amb vistes ens permet no haver de destriar a cada consulta tots els empleats de l'edifici Omega que tinguin assignacions vigents, ja que ja estan emmagatzemats a la vista en qüestió.

2.2) Considereu la taula professors(dni,nomProf,telefon), propietat d'en Toni. Suposeu també la seqüència de sentències següent relativa a autoritzacions sobre la taula professors. Cada sentència està numerada i s'indica el nom de l'usuari que vol executar-la.

1 - Toni: GRANT SELECT ON professors TO Albert WITH GRANT OPTION
2 - Albert: GRANT SELECT ON professors TO Carme WITH GRANT OPTION
3 - Carme: GRANT SELECT(dni,telefon) ON professors TO Dolors WITH GRANT OPTION
4 - Carme: GRANT SELECT(dni,nomProf) ON professors TO Se WITH GRANT OPTION
5 - Toni: GRANT SELECT ON professors TO Se
6 - Toni: GRANT SELECT(telefon) ON professors TO Xavi
7 - Dolors: GRANT SELECT(dni,telefon) ON professors TO Xavi WITH GRANT OPTION
8 - Se: GRANT SELECT(dni,telefon) ON professors TO Xavi
9 - Dolors: GRANT SELECT(dni) ON professors TO Elena
10 - Se: GRANT SELECT(dni) ON professors TO Elena
11 - Toni: REVOKE SELECT ON professors FROM Se RESTRICT
12 - Carme: REVOKE SELECT(dni,telefon) ON professors FROM Dolors RESTRICT
13 - Dolors: REVOKE SELECT(dni) ON professors FROM Se CASCADE
14 - Albert: REVOKE SELECT ON professors FROM Carme CASCADE
15 - Toni: REVOKE SELECT ON professors FROM Albert RESTRICT

a) Quines d'aquestes sentències, si n'hi ha cap, no s'executaran amb èxit o no tindran cap efecte sobre la base de dades? Raoneu la resposta. Assumirem que les sentències que no s'executin amb èxit, no tindran cap efecte i es continuarà amb la sentència següent.

8 - Se només té GRANT OPTION de Select(dni,nom), no de telèfon.
11 - Se ha donat permisos a Elena, per tant no se li poden revocar amb RESTRICT.
12 - Dolors ha donat permisos a Xavi i Elena, i no se li poden revocar amb RESTRICT.
13 - No té efecte, Dolors no ha donat privilegis a Se.

b) En acabar d'executar les sentències anteriors, quins privilegis tindran els usuaris Xavi i Albert sobre la taula professors?

Albert no tindrà privilegis, i Xavi Select(telèfon)

c) Definiu els rols "rol-Xavi" i "rol-Albert" de tal manera que aquests rols tinguin els privilegis del Xavi i l'Albert després d'executar les sentències.

```
CREATE ROLE rol-Xavi;
GRANT SELECT(telefon) ON professors TO rol-Xavi;
CREATE ROLE rol-Albert; //no té privilegis
```

3) (2 punts) Donada la taula següent:

```
CREATE TABLE items (
item integer primary key,
name char(25),
qtt integer,
preu_total decimal(9,2));
```

i la regla de negoci: "*Una única sentència de modificació (update) no pot augmentar la quantitat total en estoc dels productes en més d'un 50%*", ens demanen implementar amb triggers aquesta regla de negoci. Una possible so lució en PostgreSQL seria (cada part de la solució està assenyalada amb una lletra per facilitar fer-hi referència):

```
a) CREATE TABLE temp(old_qtt integer);

b) CREATE FUNCTION update_items_before()RETURNS trigger AS $$
    BEGIN
    DELETE FROM temp;
    INSERT INTO temp SELECT sum(qtt) FROM items;
    RETURN NULL;
   END $$ LANGUAGE plpgsql;

c) CREATE FUNCTION update_items_after()RETURNS trigger AS $$
    DECLARE
    oldqtt integer default 0;
    newqtt integer default 0;
      BEGIN
    SELECT old_qtt into oldqtt FROM temp;
    SELECT sum(qtt) into newqtt FROM items;
    IF (newqtt>oldqtt*1.5) THEN
      RAISE EXCEPTION 'Violació regla de negoci';
    END IF;
    RETURN NULL;
   END $$ LANGUAGE plpgsql;

d) CREATE TRIGGER regla_negociBS BEFORE UPDATE ON items
   FOR EACH STATEMENT EXECUTE PROCEDURE update_items_before();

e) CREATE TRIGGER regla_negociAS AFTER UPDATE ON items
   FOR EACH STATEMENT EXECUTE PROCEDURE update_items_after();
```

3.1) Expliqueu quin és el principal problema d'aquesta solució, atenent als criteris de qualitat de triggers vistos a classe.

El problema és que a cada update hem de sumar totes les quantitats de cada tupla de la taula items, és a dir, accedim a totes les tuples de items en comptes de només a les que modifiquem.

3.2) Quins canvis faríeu a d) i e) per tal superar aquest inconvenient? Per què?

Canviar la crida a FOR EACH ROW en comptes de FOR EACH STATEMENT, per així accedir únicament a les files modificades i no a tota la taula.

3.3) Codifiqueu la nova funció c) que elimini els inconvenients de la solució donada quan es combini amb els canvis introduïts a 3.2.

```
CREATE FUNCTION update_items_after() RETURNS trigger AS $$
DECLARE
        oldqtt integer default 0;
        suma_incr integer default 0;
BEGIN
        UPDATE temp SET incr = incr+(NEW.qtt-OLD.qtt);
        SELECT old_qtt, incr INTO oldqtt, suma_incr FROM temp;
        IF (suma_incr > old_qtt*0.5) THEN
                RAISE EXCEPTION 'Violació de regla de negoci';
        END IF;
        RETURN NULL;
END $$ LANGUAGE plpgsql;
```

4) (2 punts)

4.1) Donada la taula R(a,b) (clau primària subratllada) que conté les files {('a1',1), ('a2',2)} i les dues transaccions següents:

```
T1: insert into R values ('a3',3); update R set b=b*2; commit;
T2: select * from R; select * from R; commit;
```

Suposant que el SGBD no implementa cap mecanisme de control de concurrència, digueu quins serien els possibles resultats de les dues sentències select de T2, en funció dels possibles ordres d'execució de les transaccions. En cas que alguns d'aquests resultats siguin conseqüència de l'existència d'interferències, digueu quines serien aquestes interferències. Argumenteu breument les vostres respostes.

Considereu que les accions (R, RU, W) corresponents a una mateixa sentència SQL s'executen sempre totes seguides.

Si les dues transaccions fossin serials, els dos selects llegirien el mateix, l'estat inicial si T2 ho fa abans que T1, o les modificacions de T1 si T2 va després.
Si, en canvi, T2 fes el primer select, després T1 inserís l'element 'a3' y més tard T2 tornés a llegir R, tindríem la interferència d'un element fantasma (en aquest cas no importa la situació de l'update de T1).
Si primer T1 fes el insert, després T2 llegís amb el primer select, T1 fés el update i T2 tornés a llegir, es produiria una lectura no repetible.

4.2) Suposeu ara que la transacció T2 s'executa concurrentment amb una transacció T 3, en l'ordre que tot seguit s'indica:

T2	T3
select * from R	
	select * from R where a='a1'
	update R set b=b*2 where a='a1'
	commit
select * from R	
commit	

Suposant que les dues transaccions treballen amb nivell de *repeatable read*, que l'SGBD fa servir reserves S, X, que el gränul és la fila, i que la taula R conté les files {('a1',1), ('a2',2)}, contesteu a les preguntes següents:

a) Com quedaria l'horari en termes d'operacions de baix nivell? L'horari ha d'incloure, a més de les peticions que executen les trans accions (R, RU, W, COMMIT), les operacions de petició i alliberament de reserves i l'ordre d'execució de totes aquestes peticions.

T2	T3
Lock(a1,S)	
R(a1)	
Lock(a2,S)	
R(a2)	
	Lock(a1,S)
	R(a1)
	Lock(a1,X)
	-espera-
R(a1)	
R(a2)	
COMMIT(Unlock(a1,a2))	
	RU(a1)
	W(a1)
	COMMIT(Unlock(a1,a2))

b) Quins horaris serials hi són equivalents?

T2;T3

4.3) Ens informen que la BD que conté la taula R, a causa d'un desastre, passa a estar inaccessible. En aquest escenari contesteu, argumentant les vostres respostes, a les preguntes següents:

a) Quines propietats de les transaccions es veuen compromeses?

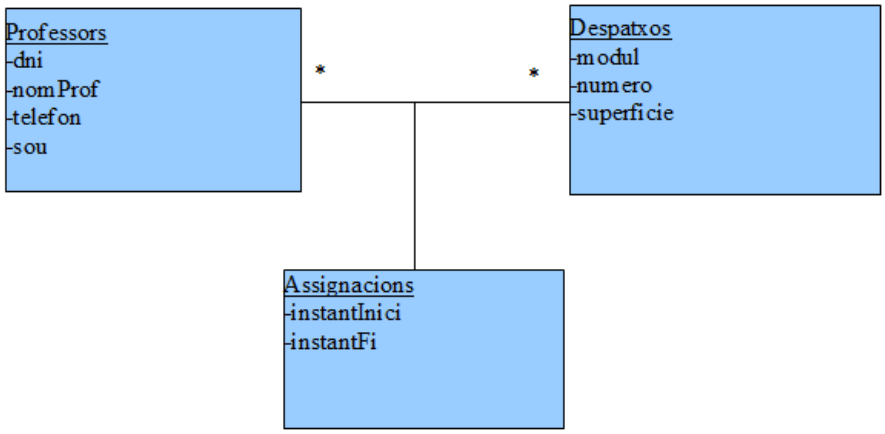
> Tenint en compte les propietats ACID, es veu compromesa la propietat de la Definitivitat, ja que no s'assegura que les dades fossin definitives a la BD.

b) Quines fonts d'informació necessita l'SGBD per reconstruir la BD?

- Una còpia de seguretat de la BD.
- Un dietari de les accions a la BD des de que es va fer la còpia de seguretat.

5) (2 punts) Considereu la base de dades de l'exercici 1.

5.1) Doneu un possible model conceptual en UML que generi, quan es fa la traducció a m odel relacional, les taules anteriors. En el UML hi ha d'haver: classes, atributs, associacions amb les seves multiplicitats, i les claus externes.

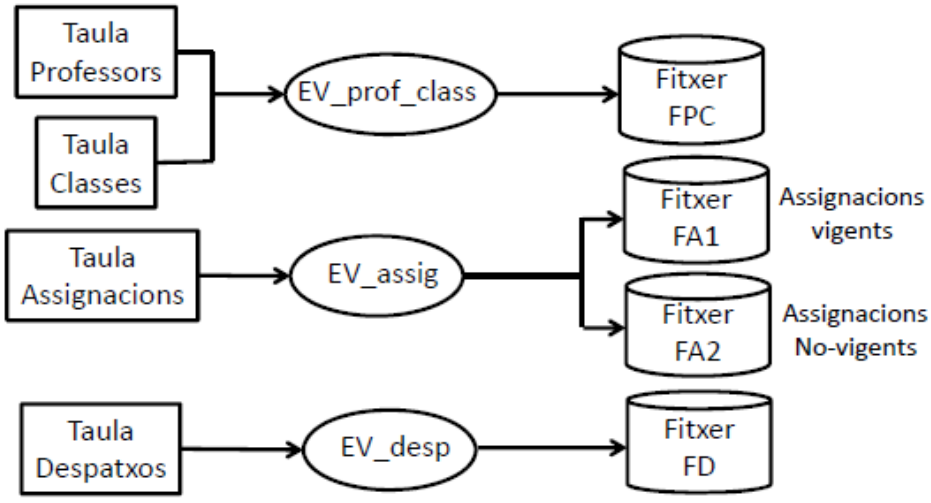


Clau externa professors: dni.
Clau externa Despatxos: (modul,numero).

5.2) Suposeu ara, que a la base de dades de l'exercici 1 s'hi afegeix la taula classes, amb l'esquema següent:

```
classes (aula, horari, grup, dni)
 -- on dni és clau forana que referencia professors.
```

a) Digueu a quin tipus correspon cadascun dels espais virtuals indicats a continuació:



1. Espai virtual d'agrupació.
2. Espai virtual fragmentat.
3. Espai virtual de taules.

b) Com sabeu, els diferents tipus d'espais virtuals afavoreixen o penalitzen certs tipus de consultes. Tenint en compte nom és els espais virtuals abans identificats, digueu quines de les consultes següents es veuen afavorides o penalitzades. Justifiqueu breument la resposta.

```
SELECT p.dni,p.nom,c.aula,c.grup,c.horari
FROM professors p, classes c
WHERE p.dni=c.dni;
```

Aquesta consulta es veu afavorida, perquè s'està fent una join de dos taules que estan dins d'un mateix espai virtual d'agrupació.

```
SELECT d.modul, d.numero, d.superficie
FROM despatxos d, assignacions a
WHERE d.modul=a.modul AND d.numero=a.numero AND a.instantFI is NULL
      AND d.modul='Omega';
```

Aquesta consulta no es veu gens afavorida, perquè ha de consultar dues a taules a dos espais virtuals diferents (i per tant, a dos fitxers diferents a la vegada) i costa més de fer la join.

5.3) Suposeu ara que la taula d'assignacions s'emmagatzema emprant un espai virtual de taula i que té aproximadament 10.000 tuples, que estan emmagatzemades en pàgines amb una mitjana de 10 tuples per pàgina. Sabem també que hi ha aproximadament 1000 assignacions amb dni>'444' i que hi ha 300 assignacions al mòdul 'Omega'. De les assignacions amb dni>'444' n'hi ha 50 que són al mòdul 'Omega'. A més, se sap que s'han definit dos índexs B+ un per dni i un per mòdul. L'arbre B+ per dni és d'ordre d=157, és no agrupat, i té una ocupació de les pàgines de l'índex del 70% en mitjana. L'arbre B+ per mòdul és d'ordre d=227, és no agrupat, i té una ocupació de les pàgines de l'índex del 60% en mitjana.

Donada la consulta següent, estimeu (i justifiqueu breument) el nombre de pàgines (d'índex i de dades) que es llegiran si la consulta es resol emprant els dos índexs amb estratègia d'intersecció de RIDs.

```
SELECT * FROM assignacions a WHERE a.dni > '444' AND a.modul='Omega'
```

Per la intersecció de RIDs, s'han de trobar els RIDs de les dues bandes del AND, per després trobar la intersecció entre ells.

6 accessos per dni + 3 accessos per mòdul + 50 accessos pel RID = 2 + (1000/219) – 1 = **59**

Examen 14 gener de 2013

1) (2 punts) Considereu l'esquema de la base de dades següent:

```
create table professors (
dni char(9),
nomProf char(50) unique,
telefon char(15),
sou integer not null check(sou>0),
primary key (dni));

create table despatxos (
modul char(5),
numero char(5),
superficie integer not null check(superficie>12),
primary key (modul,numero));

create table assignacions (
dni char(9),
modul char(5),
numero char(5),
instantInici integer,
instantFi integer,
primary key (dni, modul, numero, instantInici),
foreign key (dni) references professors,
foreign key (modul,numero) references despatxos);
-- assignació d'un professor a un despatx.
-- instantFi te valor null quan una assignacio es encara vigent.
```

1.1) Escriviu una sentència SQL per obtenir el s mòduls que tenen m és de 100 despatxos amb superfície >15 i amb totes les seves assignacions vigents.

```
SELECT d.modul
FROM despatxos d NATURAL INNER JOIN assignacions a
WHERE d.superficie > 15 AND a.instantFi IS NULL
GROUP BY d.modul
HAVING COUNT (*) > 100;
```

1.2) Escriviu una sentència SQL per augmentar 10 metres quadrats la superfície dels despatxos que no han estat mai assignats a més d'un professor.

```
UPDATE despatxos SET superficie = superficie+10
WHERE 1 > (SELECT COUNT(DISTINCT a.dni)
                FROM assignacions a
                WHERE a.modul = despatxos.modul
                        AND a.numero = despatxos.numero);
```

1.3) Doneu una seqüència que contingui el mínim nombre de sentències SQL d'actualització de la base de dades anterior completament buida, que violi la integritat referencial de la clau forana de la taula Assignacions que referencia la taula Despatxos, amb l'intent d'inserció o modificació d'una tupla d'assignacions amb modul='Omega' i numero='119'. Les instruccions NOMÉS han de violar aquesta restricció.

INSERT INTO professors
VALUES ('47813',Manolo','628111111',2000);
INSERT INTO assignacions
VALUES ('1111','Omega','119',100,NULL);

2) (2 punts) Considereu l'esquema de la base de dades següent:

```
nens(idNen, nomNen)
botigues(idBotiga, nomBotiga, ciutat)
caramels(idCaramel, nomCaramel)
compren(idNen, idBotiga)
       idNen referencia la taula nens
       idBotiga referencia la taula botigues
       Hi ha una fila per cada nen i botiga on va a comprar.
venen(idBotiga, idCaramel)
       idBotiga referencia la taula botigues
       idCaramel referencia la taula caramels
       Hi ha una fila per cada botiga i caramel que hi venen.
agraden(idNen, idCaramel)
       idNen referencia la taula nens
       idCaramel referencia la taula caramels
```

2.1) Descriviu amb una frase o enunciat les dades que s'obtenen com a resultat de la consulta en àlgebra relacional següent. La frase o enunciat hauria de començar amb: "Donar una sentència d'operacions d'àlgebra relacional per obtenir...".

```
A = caramels(nomCaramel = 'SUGUS')
B = agraden * A
C = B[idNen]
D = nens*C
E = nens-D
```

Donar una sentència d'operacions d'àlgebra relacional per obtenir la id i el nom dels nens als quals no els agraden els sugus.

2.2) Suposeu l'enunciat i seqüència de sentències d'àlgebra relacional següents. Digues si les sentències són una solució correcta per a l'enunciat donat. En cas afirmatiu, doneu un contingut de la base de dades i la solució que s'obtindria en fer la consulta. En cas negatiu, digueu quins són els errors que existeixen en la seqüència de sentències.

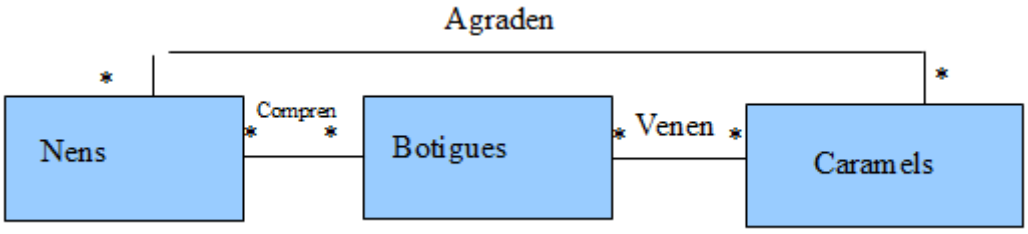
Enunciat: Donar una sentència d'operacions en àlgebra relacional per obtenir els caramels que agraden o al Joan o a la Queralt i que no els venen a cap botiga de Barcelona.

Solució:

A = nens * agraden
B = A(nomNen='Joan')
C = A(nomNen='Queralt')
D = B ∩ C
E = D[idCaramel]
F = botigues(ciutat <> 'Barcelona')
G = venen * F
H = G[idCaramel]
R = E-H

Hi ha diversos errors. Primer, a la quarta sentència, hauria de ser una intersecció i no una unió, perquè sinó s'estan agafant els que li agraden a un I a l'altre. Després, a l'hora de destriar els que no es venen a Barcelona (a partir de la línia 6), el que s'hauria de fer és agafar els que es venen a Barcelona (F = botigues(ciutat='Barcelona'); G=venen*F), i treure-li al total de caramels (H=G[idCaramel], I=caramels-H). Ara tan sols queda fer la join amb l'anterior (R=I*E).

2.3) Doneu un possible model conceptual en UML que generi, quan es fa la traducció a model relacional, les taules anteriors. En el UML hi ha d'haver: classes, atributs, associacions amb les seves multiplicitats, i les claus externes.



2.4) Suposant que la base de dades anterior és propietat de l'usuari Mikel, doneu les sentències SQL necessàries per tal d'autoritzar a l'usuari Ton per a que pugui consultar (però no modificar) les dades de les botigues que venen caramels a més de 10 nens, i només les d'aquestes botigues.

Mikel:
```
CREATE VIEW vista AS SELECT * FROM botigues WHERE 10 < (SELECT COUNT(DISTINCT
c.idNen) FROM compres c WHERE c.idBotiga = botigues.idBotiga);
GRANT SELECT ON vista TO Ton;
```

3. (2 punts) Donades les taules següents:

```
CREATE TABLE Departaments (
num_dpt integer primary key,
pressupost integer not null);

CREATE TABLE Empleats (
num_empl integer primary key,
sou integer not null check (sou>0),
num_dpt integer not null references Departaments);
```

i l'asserció següent expressada en SQL:

```
CREATE ASSERTION menys_de_10 CHECK
NOT EXISTS (SELECT d.num_dpt
            FROM departaments d NATURAL INNER JOIN empleats e
            WHERE d.pressupost<50000
            GROUP BY d.num_dpt
            HAVING count(*) >10)
```

3.1) Considerant que es vol implementar aquesta asserció en PostgreSQL mitjançant disparadors, indiqueu quin són els esdeveniment rellevants. Per a cada esdeveniment rellevant i taula, indiqueu també el tipus de disparador escollit. En cas de que l'esdeveniment sigui un UPDATE cal també indicar les columnes rellevants. Doneu una breu explicació de l tipus de disparador escollit.

- Update departaments (pressupost): trigger BEFORE/FOR EACH ROW. Si el departament passarà a tenir pressupost<50000 i te més de 10 empleats, s'ha d'anul·lar (return OLD).
- Update Empleats: trigger BEFORE/FOR EACH ROW. Si l'empleat es vol passar a un departament amb 10 treballadors i pressupost<50000 s'ha d'anular (return OLD).
- Insert Empleats: trigger BEFORE/FOR EACH ROW. Si es vol afegir un empleat a un departament amb 10 treballadors i pressupost<50000 s'ha d'anular (return NULL).

3.2) Suposeu ara que afegim un atribut derivat a la taula de Departaments anomenat quants_empleats. Aquest atribut contindrà el número d'empleats del departament. Indiqueu quin són els esdeveniments rellevants i els tipus de disparadors que escollíreu per mantenir de manera automàtica aquest atribut derivat. Per cada disparador, cal també que justifiqueu breument el tipus de disparador escollit. Suposeu un usuari no farà mai un UPDATE sobre l'atribut quants_empleats de la taula departament.

- Insert Empleats: trigger AFTER/FOR EACH ROW. Per cada empleat que inserim, augmentem en 1 el contador quants_empleats del departament corresponent.
- Update Empleats(num_dpt): trigger AFTER/FOR EACH ROW. Si canviem el número de departament d'un empleat, hem de disminuir en 1 quants_empleats del antic i augmentar 1 el del nou.
- Delete Empleats: trigger AFTER/FOR EACH ROW. Després d'eliminar l'empleat, disminuim en 1 el comptador del departament on era.

3.3) Tenint en compte aquest nou atribut derivat, és possible usar un check de taula per implementar l'asserció? En cas afirmatiu, indiqueu el check. En cas negatiu, indiqueu la raó.

No és possible, perquè estem parlant d'una condició que combina els valors de dues columnes de la taula, no només d'una. Podem fer un check que miri si el pressupost es <50000 o que quants_empl>10, però no els dos a la vegada.

4) (2 punts) Donada la taula T(x,y) (clau primària subratllada) que conté les files { ('x1',1), ('x2',2)} i les dues transaccions següents:

```
T1: select * from T; select * from T; commit;
T2: update T set y=y*2; delete from T where x='x1'; commit;
```

4.1) Suposant que el SGBD no implementa cap mecanisme de control de concurrència proposeu, si és possible, horaris d'exemple (i els resultats de les dues sentències select de T1) que verifiquin les característiques següents:

a) Un horari que no contingui cap interferència.

T1	T2
Select * from T	
Select * from T	
COMMIT	
	update T set y=y*2
	delete from T where x='x1'
	COMMIT

El resultat dels Select és l'estat inicial de la taula.

b) Un horari que només contingui una interferència de tipus lectura no repetible.

T1	T2
Select * from T	
	update T set y=y*2
Select * from T	
COMMIT	
	delete from T where x='x1'
	COMMIT

El primer select llegeix l'estat inicial de la taula, el següent, després de l'update, un estat diferent (lectura no repetible).

c) Un horari que només contingui una interferència de tipus fantasma.

No pot haver-hi un elements fantasma perquè no s'està inserint cap element nou a la taula.

Important: Argumenteu breument les vostres respostes. Considereu que les accions (R, RU, W) corresponents a una mateixa sentència SQL s'executen sempre totes seguides.

4.2) Suposem ara que la transacció T1 s'executa concurrentment amb una transacció T3, en l'ordre que tot seguit s'indica:

T1	T3
select * from T	
	update T set y=y*2 where x='x1'
select * from T	
	select * from T where x='x2'
	commit
commit	

Suposant que les dues transaccions treballen amb nivell de *read committed*, que l'SGBD fa servir reserves S, X, que el gràNul és la fila, i que la taula T conté les files {('x1',1), ('x2',2)}, contesteu a les preguntes següents:

a) Com quedaria l'horari en termes d'operacions de baix nivell? L'horari ha d'incloure, a més de les peticions que executen les transaccions (R, RU, W, COMMIT), les operacions de petició i alliberament de reserves i l'ordre d'execució de totes aquestes peticions.

T1	T2
Lock(x1,S)	
R(x1)	
Unlock(x1)	
Lock(x2,S)	
R(x2)	
Unlock(x2)	
	Lock(x1,X)
	RU(x1)
	W(x1)
Lock(x1,S)	
-espera-	
	Lock(x2)
	R(x2)
	Unlock(x2)
	COMMIT(Unlock(x1))
R(x1)	
Unlock(x1)	
Lock(x2,S)	
R(x2)	
Unlock(x2)	
COMMIT	

b) Quins horaris serials hi són equivalents?

No té horari serial equivalent, perquè no es serialitzable (té una lectura no repetible).

4.3) Expliqueu què signifiquen les propietats ACID que tota transacció ha de complir.

Atomicitat: Les operacions s'han de fer senceres, no es pot interrumpir una operació per la meitat sense acabar-la.
Consistència: La BD ha d'estar programada per fer el que l'usuari espera.
Aïllament: Les accions fetes a una transacció no han d'afectar a els resultats esperats a una altra transacció.
Definitivitat: Els canvis produïts a una transacció s'han de preservar a la base de dades i no esborrar-se.

5) (2 punts) S'ha creat una taula T1 (amb esquema: T1(a integer, b integer)). L'atribut a és clau primària a de la taula, però no s'ha definit ca p índex a T1. La taula T1 té 4000 tuples i està emmagatzemada en un fitxer no ordenat, on a cada pàgina hi caben en promig 10 files. Sobre T1, s'executa la consulta C1 = {SELECT * FROM T1 where a = valor}. Es demana:

5.1) Calcular el cost (en nombre de pàgines accedides) per resoldre C1. Justifiqueu breument la resposta.

Si la taula té 4000 files, i a cada pàgina hi caben 10 files, accedirem en el pitjor dels casos a 400 pàgines fins a trobar la fila que tingui aquell valor.

5.2) Es defineix ara un índex B+ no agrupat per l' atribut a de T1. L'índex té d=50 i els nodes estan plens al 70% d'ocupació. Es torna a executar la consulta C1. Calcular el cost (en nombre de pàgines accedides) per resoldre C1. Justifiqueu breument la resposta.

Només caldria accedir a 2 pàgines d'índexs per baixar fins a l'últim nivell, més un accés a la fila trobada en qüestió. En total 3 accessos.

5.3) Es defineix ara un índex B+ no agrupat per l' atribut b de T1. L'índex té d=50 i els nodes estan plens al 70% d'ocupació. S'executa la consulta C2 ={SELECT sum(b) FROM T1 }. Calcular el cost (en nombre de pàgines accedides) per resoldre C2. Justifiqueu breument la resposta.

Hem de baixar fins a l'últim nivell (2 accessos), anar al primer element al que apunti l'índex, i iterar fins al final els nodes fulla. És a dir, accedirem a tots els nodes fulles, que són 4000/70=58 pàgines. En total, 61 accessos.

5.4) Es defineix ara un índex B+ agrupat per l'atribut a de T1. L'índex té d=50 i els nodes estan plens al 70% d'ocupació. No es de fineix cap índex sobre l'atribut b. Suposeu que els valors de l'atribut a estan distribuïts uniformement entre 1 i 4000. S' executa la consulta C3={SELECT * FROM T1 WHERE a>=3300 OR b<=100}. Calcular el cost (en nombre de pàgines accedides) per resoldre C3. Justifiqueu breument la resposta.

Cost per a = 2 +(4000-3300)/10 = 72 pàgines
Cost per b = 4000/factor bloqueig = 4000/f = 400
Cost total: 72 + 400 = 472 pàgines.

Examen 7 de juny de 2013

1) (2 punts) Considereu l'esquema de la base de dades següent:

```
create table METGES (
num_met integer,
nom_met  char(30),
especialitat char(30),
carrer char(30),
telefon integer,
sou integer,
primary key (num_met));

create table MALALTS (
num_mal integer,
nom_mal char(30),
carrer char(30),
dni integer unique,
primary key (num_mal) );

create table VISITES (
num_met integer,
num_mal integer,
instant_visita integer,
import integer,
primary key  (num_met, num_mal, instant_visita),
foreign key (num_met)  references METGES(num_met),
foreign key (num_mal)  references MALALTS(num_mal) );
```

1.1) Escriviu una sentència SQL per obtenir el nom dels metges que no tenen cap visita a malalts que viuen al carrer 'Muntaner' i que tenen una especialitat que no té cap altre metge.

```
SELECT me.nom_met
FROM metges me
WHERE 1 = (SELECT COUNT(*)
                FROM metges
                WHERE especialitat = me.especialitat)
GROUP BY me.num_met
HAVING NOT EXISTS (SELECT ma.carrer
                        FROM malalts ma, visites v
                        WHERE ma.num_mal = v.num_mal
                                AND me.num_met = v.num_met
                                AND ma.carrer = 'Muntaner');
```

1.2) Escriviu una sentència SQL per obtenir els metges que han fet més de 10 visites a un mateix malalt i que cobren un sou superior a la mitjana del sou de tots els metges. En el resultat de la consulta ha de sortir el número del metge, el nom del metge, el número del malalt i l'import total de totes les visites del metge al malalt.

```
SELECT me.num_met, me.nom_met, ma.num_mal, SUM(v.import)
FROM metges me, malalts ma, visites v
WHERE me.num_met = v.num_met AND ma.num_mal = v.num_mal
GROUP BY me.num_met, me.nom_met, ma.num_mal HAVING COUNT(*)>10 AND
me.sou>(SELECT AVG(sou) FROM metges);
```

1.3) Proposeu una sentència SQL de creació de la taula següent:

```
proves-programades(num_met, num_mal, instant_visita, nom_prova,
prioritat)
```

Cada fila de la taula proves-programades representa una prova que el metge li demana al malalt (durant una visita) que es faci.

Per a la creació de la taula, cal tenir en compte que:

- Quan un metge visita a un malalt, el metge pot demanar que es faci una o més proves (per exemple, durant una visita el metge pot demanar al malalt que quan torni en la propera visita s'hagi fet dues proves: una ecografia i una anàlisi de sang). El que no pot passar mai és que en una mateixa visita a un malalt se li demani dues vegades el mateix nom de prova.
- Una prova és sempre d'una visita que existeix a la base de dades.
- El nom de prova és un atribut de tipus varchar(30).
- Prioritat ha de ser un atribut de tipus varchar(6), que ha de tenir obligatòriament (no pot tenir valors nuls) un dels valors següents: 'URGENT', 'MITJA', 'BAIXA'.

```
CREATE TABLE proves_programades(
num_met INTEGER,
num_mal INTEGER,
instant_visita INTEGER NOT NULL,
nom_prova varchar(30),
prioritat varchar(6) NOT NULL CHECK(prioritat IN ('URGENT','MITJA','BAIXA')),
PRIMARY KEY (num_met, num_mal, instant_visita, nom_prova),
FOREIGN KEY (num_met, num_mal, instant_visita) REFERENCES VISITES
);
```

2) (2 punts)

2.1) Suposeu l'esquema de la base de dades següent:

```
nens(idNen, nomNen)
botigues(idBotiga, nomBotiga, ciutat)
caramels(idCaramel, nomCaramel)
compren(idNen, idBotiga)  idNen referencia la taula nens
idBotiga referencia la taula botigues
Hi ha una fila per cada nen i botiga on va a comprar.
venen(idBotiga, idCaramel)  idBotiga referencia la taula botigues
```

```
idCaramel referencia la taula caramels
Hi ha una fila per cada botiga i caramel que hi venen.
agraden(idNen, idCaramel, grau)  idNen referencia la taula nens
idCaramel referencia la taula caramels
Hi ha una fila per cada nen i caramel que li agrada  al nen en un cert
grau (mesurat com un enter).
```

Digueu quins privilegis li farien falta a un usuari sobre aquestes taules per poder executar les sentències següents. Concretament cal dir, per cada taula on calguin privilegis, el nom de la taula, les operacions per a les que calen privilegis, i els atributs rellevants per a cada operació.

```
Select Avg(a2.grau)
From agraden a2, nens n
Where a2.idNen = n.idNen and n.nom='Joan';

Update agraden a1
Set grau = 20
Where a1.idCaramel In (Select v.idCaramel
                       From venen v, botigues b
                       Where v.idBotiga = b.idBotiga and
                             b.ciutat ='Ripoll')
```

agraden:
- select(grau, idNen, idCaramel)
- update(grau)

nens:
- select(idNen,nom)

venen:
- select(idBotiga, idCaramel)

botigues:
- select(idBotiga,ciutat)

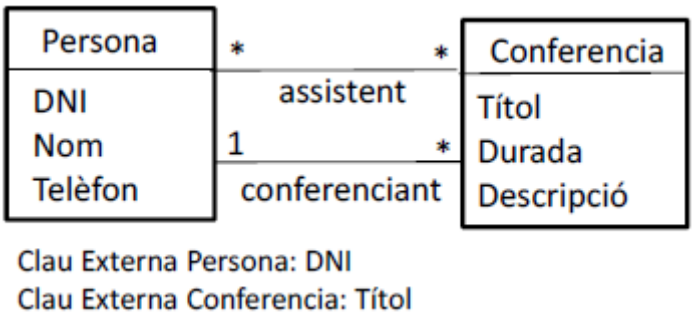
2.2) Suposeu la base de dades de l'apartat 2.1 on la cardinalitat de les taules és respectivament NE, BO, CA, CO, VE, AG (amb NE,BO,CA,CO,VE,AG > 0). Digueu quina serà la cardinalitat de la relació R per cadascuna de les seqüències d'operacions d'àlgebra relacional següents. En cas de no poder dir exactament quina serà dóna una cardinalitat mínima i una màxima.

```
(i)     R = Nens * Agraden

(ii)    R=  Nens x Agraden

(iii)   A= Compren[idBotiga]
        B= Venen[idBotiga]
        R= A U B

(iv)    A= Agraden[grau]
        B = A{grau -> gr}
        C = A[grau > gr]B
        D = C[gr]
        R= A-D
```

i. Min:0, Max:max{NE,AG}
ii. NE*AG

iii. Min: 0 (si les dues taules són buides), Max: CO+VE (si no hi ha cap repetida)
iv. AG

2.3) Suposeu el model conceptual en UML següent. Doneu el disseny lògic que s'obté fent la traducció a model relacional.



Clau Externa Persona: DNI
Clau Externa Conferencia: Títol

Persona(DNI,Nom,Telefon)
Conferencia(Titol,Durada,Descripcio,dni_conferenciant)
{dni_conferenciant} referencia Persona.
Assistent(dni-assistent,titol-conferencia)
{dni-assistent} referencia Persona.
{titol-conferencia} referencia Conferencia.

2.4) Expliqueu què és la redundància de les dades, per què es desitjable evitar aquesta redundància en la funcionalitat d'un SGBD i en quins casos podria ser útil no evitar-la.

La redundància de dades es repetir (o inserir) dades calculables a la pròpia BD. És desitjable evitar-les perquè ocupen espai i, especialment, per les possibles inconsistències que poden sorgir si aquesta còpia a memòria no es manté actualitzada.

Els únics casos en que és útil no evitar-la es per millorar significativament el rendiment de les consultes.

3) (2 punts) Considereu que hem definit una taula amb un únic atribut T1(A) que conté inicialment dues tuples {(11),(12)} i una altra taula T2(B) amb tuples {(3),(4),(5),(6)}.

3.1) Suposeu que hem definit en PostgreSQL el trigger següent:

```
CREATE or replace FUNCTION actualitzar() RETURNS trigger AS $$
   DECLARE
       BEGIN
             UPDATE T1 set A=A+(select count(*) from T1);
             RETURN NEW;
       END;
$$LANGUAGE plpgsql;

CREATE TRIGGER ex3.1 BEFORE INSERT ON T1
FOR EACH STATEMENT EXECUTE PROCEDURE actualitzar();
```

Indiqueu quines tuples hi haurà a la taula T1 després de l'execució de la sentència

```
Insert Into T1 (Select * From T2)
```

Raoneu breument la resposta.

T1 = {(13),(14),(3),(4),(5),(6)}

3.2) Quin seria el resultat si el trigger fos:

```
CREATE TRIGGER ex3.2 AFTER INSERT ON T1     FOR EACH STATEMENT EXECUTE
PROCEDURE actualitzar();
```

Raoneu breument la resposta.

T1 = {(17),(18),(9),(10),(11),(12)}

3.3) I si fos

```
CREATE TRIGGER ex3.3 BEFORE INSERT ON T1     FOR EACH ROW EXECUTE
PROCEDURE actualitzar();
```

Raoneu breument la resposta

T1 = {(25),(26),(15),(13),(10),(6)}

4) (2 punts) Donada la taula T(x,y) amb les files {('x1',1), ('x2',2)} (clau primària subratllada) i les tres transaccions següents:

```
T1:   set transaction isolation level repeatable read;
      select * from T;
      select * from T;
      commit;
T2:   set transaction isolation level serializable;
      insert into T values ('x3',3);
      commit;
T3:   set transaction isolation level read committed;
      update T set y=y*2;
      commit;
```

4.1) Suposant que el SGBD implementa reserves S, X i que el gràNul és la fila proposeu, si és possible, horaris d'exemple amb les tres transaccions que incloguin totes les seves sentències i que verifiquin les característiques següents:

a) Un horari que contingui una interferència de tipus lectura no repetible.

No és possible obtenir un horari amb lectura no repetible, ja que la única transacció que llegeix dues vegades és T1, que té un nivell d'aïllament Repeatable Read.

b) Un horari que contingui una interferència de tipus fantasma.

Si T1 llegeix primer la taula T, bloqueja totes les files actuals, i després T2 n'hi afegeix una altra, T1 tornarà a llegir després i es trobarà amb un resultat diferent, amb una tupla de més.

T1	T2	T3
	Lock(x1,X)	
	RU(x1)	
	W(x1)	
	Lock(x2,X)	
	RU(x2)	
	W(x2)	
	COMMIT(Unlock(x1,x2))	
Lock(x1,S)		
R(x1)		
Lock(x2,S)		
R(x2)		
		Lock(x3,X)
		W(x3)
		COMMIT(Unlock(x3))
R(x1)		
R(x2)		
Lock(x3,S)		
R(x3)		
COMMIT(Unlock(x1,x2,x3))		

Important: Argumenteu breument les vostres respostes. Considereu que les accions (R, RU, W) corresponents a una mateixa sentència SQL s'executen sempre totes seguides.

4.2) Suposem ara que la transacció T1 s'executa concurrentment amb una transacció T4, en l'ordre que tot seguit s'indica:

T1	T4
select * from T	
	select * from T where x='x1'
select * from T	
	update T set y=y*2 where x='x2'
	commit
commit	

Suposant que les dues transaccions treballen amb nivell de repeatable read, que l'SGBD fa servir reserves S, X, que el gràNul és la fila, i que la taula T conté les files {('x1',1), ('x2',2)}, contesteu a les preguntes següents:

a) Com quedaria l'horari en termes d'operacions de baix nivell? L'horari ha d'incloure, a més de les peticions que executen les transaccions (R, RU, W, COMMIT), les operacions de petició i alliberament de reserves i l'ordre d'execució de totes aquestes peticions.

T1	T2
Lock(x1,S)	
R(x1)	
Lock(x2,S)	
R(x2)	
	Lock(x1,S)
	R(x1)
R(x1)	
R(x2)	
	Lock(x2,X)
	-espera a commit de T1-
COMMIT(Unlock(x1,x2))	
	RU(x2)
	W(x2)
	COMMIT(Unlock(x1,x2))

b) Quins horaris serials hi són equivalents? Important: Argumenteu breument les vostres respostes. Considereu que les accions (R, RU, W) corresponents a una mateixa sentència SQL s'executen sempre totes seguides.

Horari serial equivalent: T1;T4.

4.3) Expliqueu què signifiquen les propietats ACID que tota transacció ha de complir.

Atomicitat: Les transaccions s'han de fer senceres sempre, si quelcom les atura, es fa un rollback.
Consistència: La programació de la BD ha de ser consistent i portar a terme el que l'usuari espera.
Aïllament: Les accions fetes a una transacció concurrent a una altra, no poden afectar als resultats de la segona.
Definitivitat: S'ha d'evitar la pèrdua d'informació tant antiga con nova, per mitjà de backups.

5) (2 punts) S'ha creat una taula T (amb esquema: T(a integer, b integer)). L'atribut a és clau primària de la taula. La taula T té 4.000.000 de tuples i els valors de a estan repartits uniformement entre 1 i 4.000.000. El factor de bloqueig del fitxer de dades és de 20 files per pàgina. Es defineixen dos índexs sobre T: un índex B+ agrupat per l'atribut a i un índex B+ no agrupat per l'atribut b. Els dos índexs tenen ordre d=50 i els nodes dels índexs estan plens al 80% de la seva capacitat.

Es demana:

5.1) Determineu el nombre de pàgines que ocupa l'índex agrupat per l'atribut a.

(4000000/80)+(50000/81)+(617/81)+(8/81) = 50000+617+8+1 = 50626 pàgines

5.2) Volem obtenir totes les files que compleixen la condició a<=10.000 AND b>5000. Sabent que hi ha 4000 files que compleixen b>5000, dubtem entre els 3 mètodes següents:

5.2.1) Recorregut seqüencial del fitxer de dades on està emmagatzemada T.

En el pitjor dels casos, 4.000.000/20 = 200.000 accessos, perquè no està ordenat.

5.2.2) Emprant l'índex B+ agrupat per a.

4+10.000/20 (trobem tots els de a, quedant-nos només amb els que compleixen la condició de b, suposant 3 nivells) = 504 accessos.

5.2.3) Emprant l'índex B+ no agrupat per b.

4+4000/80+4000 = 4054 accessos.

Calculeu el nombre de pàgines accedides (d'índex + dades) per cada un dels mètodes anteriors. Finalment, digueu quin triaríeu per resoldre la consulta, justificant breument la resposta.

Ens quedem amb l'índex agrupat per a.