Temps: 2 hores i 30 minuts

Notes 24 gener tarda **Revisió: 25 gener tarda** **Cada pregunta en un full separat**

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
que hi ha en el instant actual en el compte bancari.

CREATE TABLE ingressos
(numCompte char(16),
instantIngres integer NOT NULL,
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 NOT NULL,
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 cb
WHERE c.dni=cb.dniClient and
    cb.saldoDisponible < 0 and
    NOT EXISTS (SELECT * FROM ingres i
          WHERE i.numCompte = cb.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 DISTINCT c.dni, c.nomClient
FROM clients c, comptesBancaris cb, ingressos i
WHERE c.dni=cb.dniClient AND
      cb.numCompte=i.numCompte AND
      i.instantIngres >=1000 AND
      i.instantIngres <= 2000 AND
      NOT EXISTS (SELECT * FROM reintegraments r
                        WHERE r.numCompte = cb.numCompte AND
                              r.instantReintegrament >=1000 AND
                              r.instantReintegrament <= 2000)
GROUP BY c.dni, c.nomClient, cb.numCompte
HAVING sum(i.quantitat) > 50000
```

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
```

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

clients (dni, nomClient, ciutatResidencia) 10 Anna Barcelona
comptesBancaris(numCompte, dniClient, saldoDisponible) 111 10 5000
ingressos(numCompte, instantIngres, quantitat) 111 1001 100 111 1002 200
reintegraments(numCompte, instantReintegrament, quantitat) 111 1003 500

El resultat hauria de ser:

saldos1000p (num, saldo) 111 5200

El resultat obtingut en la implementació donada és:

saldos1000p (num, saldo) 111 5700 Degut a que les files que es formen en el where són: numCompte dniClient saldoDisponible instIngres quantitat instReintegrament quantitat 111 10 5000 1001 100 1003 500 111 10 5000 1002 200 1003 500

Un altre error en la vista que també s'ha considerat com una solució a l'apartat és que un compte no apareix a la vista si no té ingressos o reintegraments després de l'instant 1000.

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.

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).

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".

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.

Solució:

a)

Departaments(num_dpt,pressupost)	(1,4000)	(2, 3000)	
Empleats(num_empl, sou, num_empl_cap, num_dpt)	(1,1000,null,1)	(2,3000,1,1)	(3,3000,1,1)

L'execució de l'acció del disparador primer provoca l'activació del segon disparador. L'acció d'aquest segon disparador s'executa dues vegades, una per cada fila de la taula Departaments.

b) El contingut inicial de les taules d'empleats i departaments no es veu modificat per l'execució de la sentència SQL.

Les accions del segon disparador s'han d'executar dues vegades: una per la tupla (1,3000) I una altra per la tupla (2,2000). La primera vegada que s'executa l'acció del segon disparador, no es produeix cap error, però la segona vegada es produeix una violació del check de departaments pressupost >0, donat que la sentència UPDATE departaments SET pressupost=pressupost-1000 deixa el pressupost de la segona tupla a zero. Per tant, quan es produeix aquesta violació, es desfan les accions realitzades per la transacció en curs. És a dir, es desfan les accions realitzades pel segon disparador, pel primer i per la sentència que ha disparat el disparador.

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

Es considera també correcta una solució on s'afegeixi la condició "e1.num_empl_cap is not null". Encara que es tracta d'una condició innecessària, ja que en cas que no es compleixi aquesta condició mai es pot complir la condició "e1.num_empl_cap=e2.num_empl".

d.1) Possible implementació amb disparadors:

Esdeveniments i tipus de disparadors:
INSERT on Empleats BEFORE/FOR EACH ROW
UPDATE of num_dpt on Empleats BEFORE/FOR EACH ROW
UPDATE of num_empl_cap on empleats BEFORE/FOR EACH ROW
Com que l'enunciat no especifica la freqüència amb que es viola aquesta restricció respecte a les restriccions d'integritat de la BD, el tipus de disparador podria ser BEFORE o AFTER. No obstant, recordeu en la recomanació general en aquestes situacions és comprovar la restricció del disparador el més aviat possible.
L'esdeveniment DELETE on Empleats no és rellevant:
- Considerant que en la base de dades quan s'esborra un empleat s'apliqués una política de manteniment d'integritat referencial RESTRICCIÓ, no seria rellevant. El motiu és que aquest esdeveniment no podria arribar mai a violar la restricció, ja que només s'arribaria a fer l'esborrat en cas que l'empleat no fos cap de cap altre empleat.
- Considerant que en la base de dades quan s'esborra un empleat s'apliqués una política de manteniment d'integritat referencial CASCADA, tampoc seria rellevant. El motiu és que en esborrar l'empleat de les claus foranes dels empleats que li eren subordinats, aquests empleats no podrien passar a violar la restricció, ja que serien empleats que no tindrien cap, i per tant a qui no afectaria la restricció.

d.2) Possible implementació amb procediments emmagatzemats:

Per cada procediment que actualitza dades d'empleats afegir la comprovació de la restricció. Els procediments afectats per tant tots aquells que insereixin i/o modifiquin o bé l'atribut num_dpt o el num_empl_cap d'un empleat.

d.3) Sí s'afegeix la restricció a cada procediment que actualitza dades d'empleats, aleshores la restricció queda amagada, distribuïda i replicada a diversos procediments. És difícil de localitzar i modificar.

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.

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

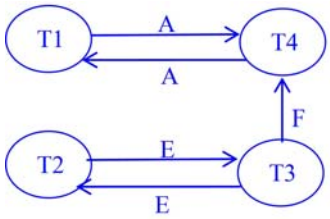
a.4. L'horari proposat, és recuperable? En cas afirmatiu doneu la definició de quan un horari és recuperable. En cas negatiu indiqueu alguna de les transaccions i operacions de l'horari que mostrin que no ho és.

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?

a.1) Graf de precedències:



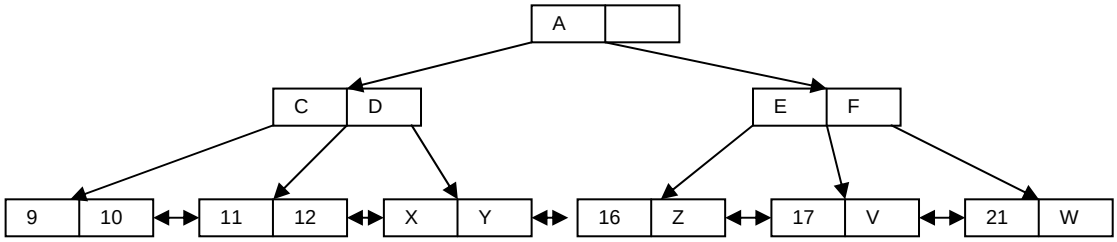
a.2) Entre T1 i T4 sobre el gràNul A es produeix una lectura no confirmada, i entre T2 i T3 sobre el gràNul E es produeix una lectura no repetible.
a.3) No existeix cap horari serial equivalent, donat que existeixen interferències.
a.4) L'horari no és recuperable. Per exemple, T4 recupera dades pendents de confirmació i confirma la seva execució abans que la transacció que ha modificat les dades (aquesta transacció és T1).

b.1) L'horari quedaria de la manera següent:

Acc#	T1	T2	T3	T4
10			L(E,S)	
20			R(E)	
30			U(E)	
40	L(A,X)			
50	RU(A)			
60	W(A)			
70			L(F,X)	
80			RU(F)	
90			W(F)	
100		L(E,X)		
110		RU(E)		
120				L(A,S)
130		W(E)		
140		L(B,S)		
150		R(B)		
160	L(B,S)			
170	R(B)			
180	U(B)			
190		L(C,S)		
200		R(C)		
210	RU(A)		L(E,S)	
220	W(A)			
230		COMMIT(U(E),U(B),U(C))		
240			R(E)	
250			U(E)	
260			COMMIT(U(F))	
270	COMMIT(U(A))			
280				R(A)
290				U(A)
300				L(F,X)
310				RU(F)
320				W(F)
330				COMMIT(U(F))

b.2) No existeix cap horari serial, la interferència de lectura no repetible entre T2 i T3 sobre el gràNul E se segueix produint, donat que T3 treballa amb nivell d'aïllament READ COMMITTED, i per evitar la interferència T3 hauria d'haver treballat amb un nivell d'aïllament de REPEATABLE READ.

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.
- **b.** Indiqueu quins són els valors possibles de C i D.
- **c.** Indiqueu quins són els valors possibles de E i F.
- **d.** Indiqueu quins són els valors possibles de A.

Justifiqueu convenientment la resposta de cada apartat.

a) Segons la definició d'arbre B+, els valors en les fulles estan ordenats i no hi ha repetits. Per tant, $12 < X < Y < 16$, i conseqüentment:
X=13, aleshores Y = 14 o 15
X=14, aleshores Y = 15
X=15, aleshores Y no podria tenir cap valor.

Z no pot contenir cap valor ja que hauria de ser $16 < Z < 17$,

V pot ser 18, 19 o 20

W>21

b) Segons la definició de node intern d'un arbre B+, C>10 i C <=11,per tant només pot ser 11. De manera similar, D > 12 i D <=X; però tenint en compte que tots els valors dels nodes interns han d'estar a les fulles D també ha d'estar a les fulles, i per tant ha de ser igual a X.
En resum, C=11 i D=X (notis, doncs, que en realitat D no pot ser < que X)

c) Els valors de E i F, estan determinats, només poden ser 17 i 21. Els argument són similars al cas C.

d) A ha de complir que ha de ser <= que tots els valors de la dreta de l'arbre i a més a més ha d'estar a les fulles. Per tant, necessàriament ha de ser 16.

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 = Pel.licula*Habilitat
B = A[nomactor,nompel]
R = Actuar-B
```

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?

Només als atributs de les taules: Actor, Habilitat i Actuar

Nivells de seguretat:

- Nivell bàsic: Nom Actor, Adreça, Telèfon, Any Naixement, Sexe, Habilitat en temes, Actuació en pel·lícules

- 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, ja que els catàlegs en PostgreSQL corresponen a les bases de dades.

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.

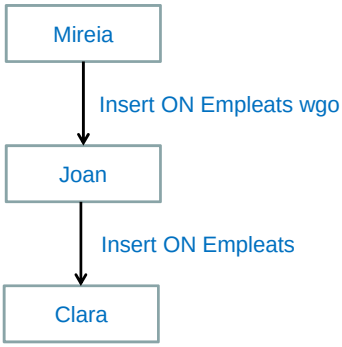
Fals, l'esquema d'informació és un esquema apart dels esquemes definits pels usuari. N´hi ha un per cada catàleg, i conté informació sobre els esquemes definits pels usuaris en el catàleg: noms i atributs de les taules, índexs, restriccions de columna, ...

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.

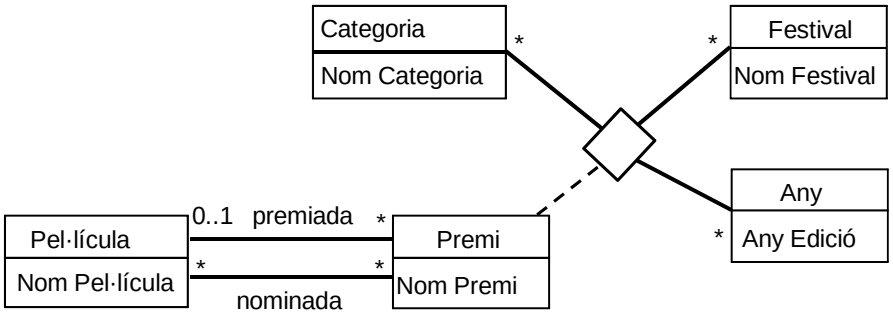
Fals. En el graf d'autoritzacions següent es pot veure que:



Si s'executa l'operació següent: Mireia: REVOKE insert ON Empleats.

Aquesta operació no es pot anul·lar tornant a la situació de partida amb una única operació 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

Categories (nom_categoria)

Festivals(nom_festival)

Any(any_edició)

Pel·lícules (nom_pel·lícula)

Premis(nom_categoria, nom_festival, any_edició, nom_premi, nom_premiada)

nom_categoria referencia Categories

nom_festival referencia Festivals

any_edició referencia Any

nom_premiada referencia Pel·lícules

Nominades(nom_categoria, nom_festival, any_edició, nom_nominada)

nom_categoria, nom_festival, any_edició referencia Premis

nom_nominada referencia Pel·lícules

També és una solució vàlida si no s'inclou la relació Any (veure transpa 306)

Temps: 3 hores

Notes 26 juny tarda **Revisió: 27 juny tarda** **Cada pregunta en un full separat**

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 tenen 2 o més assignacions a un mateix mòdul però a despatxos diferents.

```
SELECT sum(sou)
FROM professors p
WHERE    EXISTS (SELECT *
                 FROM assignacions a
                 WHERE a.instantFi is not null
                       AND a.dni=p.dni and a.modul = 'Omega')
         AND p.dni IN (SELECT a1.dni
                 FROM assignacions a1, assignacions a2
                  WHERE a1.dni=a2.dni and a1.numero <> a2.numero
                       and  a1.modul=a2.modul);

SELECT sum(sou)
FROM professors p
WHERE    EXISTS (SELECT *
                 FROM assignacions a
                 WHERE a.instantFi is not null
                       AND a.dni=p.dni and a.modul = 'Omega')
         AND EXISTS(SELECT a1.modul
                 FROM assignacions a1
                 WHERE a1.dni = p.dni
                 GROUP BY a1.modul
                 HAVING count(distinct a1.numero)>=2);
```

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 és o no correcta**. En cas de que no ho sigui **doneu una extensió de la base de dades** que faci que la consulta doni 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);
```

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);
```

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

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;
```

a) Correcte.

b) Incorrecte. Tal com està la consulta un professor sortirà o no segons si hi ha o no alguna assignació vigent a la taula assignacions. Un professor que no té assignacions vigents ha de sortir, però en aquesta consulta no sortirà quan hi hagi altres professors a la base de dades que si que en tinguin.

professors
1, Joan
2, Martí
assignacions
1, Omega, 124, 1000, null

En Martí hauria de sortir i en Joan no.

No sortiran ni en Martí ni en Joan perquè la subconsulta sempre dona resultats per tots dos.

c) Incorrecte. Que hi hagi assignacions vigents que no són del professor, no vol dir que el professor no tingui assignacions vigents. Un professor que té assignacions vigents, si a la base de dades hi ha assignacions d'altres professors amb assignacions vigents, el primer sortirà a la consulta.

professors
1, Joan
2, Martí
assignacions
1, Omega, 124, 1000, null
2, Omega, 123, 1100, null

Cap dels dos professors hauria de sortir.

Sortiran tots dos perquè cadascun es combina amb l'assignació que no és seva i compleixen la condició del where.

d) Incorrecta. Amb un group by mai no hi ha grups sense files. Els professors que no tenen assignacions vigents, desapareixen en la combinació de taules.

professors
1, Joan
2, Martí
assignacions
1, Omega, 204, 1000, null

En Martí hauria de sortir i en Joan no.

No sortirà cap dels dos. En Martí perquè no pot combinar-se amb cap assignació per tant no pot formar grup, en Joan perquè el seu grup té 1 fila.

1.3) Escriviu una seqüència d'operacions d'àlgebra relacional per obtenir els dni dels professors que han tingut alguna assignació en el mateix despatx que el professor amb dni '123'. Tingueu en compte que en el resultat de la consulta no volem que surti el professor '123'.

B=assignacions(dni='123')
C=B[dni, modul, numero]
D=C{dni ->dnip, modul->modulp, numero ->numerop}
E=assignacions[dni, modul,numero]
F=E[dni<>dnip,modul=modulp,numero=numerop]D
R=F[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.

personalactualOmega No, perquè no selecciona tota la clau.
dadespersonalactualOmega No, perquè té join i a més està definida sobre una vista no actualitzable.

b) Considereu la consulta següent:

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

b.1) Explica breument què retorna la consulta.

S'obtenen les parelles de professors amb nom diferent que tenen assignacions vigents al mateix despatx de l'edifici Omega.

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

```
select p1.nomProf, P2.nomprof
from professors p1, assignacions a1,
         professors p2, assignacions a2
where p1.dni= a1.dni and p2.dni= a2.dni
   and a1.instantFi IS NULL and a2.instantFi IS NULL
   and a1.modul='Omega' and a2.modul='Omega'
   and a1.numero=a2.numero and p1.nomprof<>p2.nomprof;
```

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

Independència lògica de dades, simplificació de consultes i programes, i control d'accés.

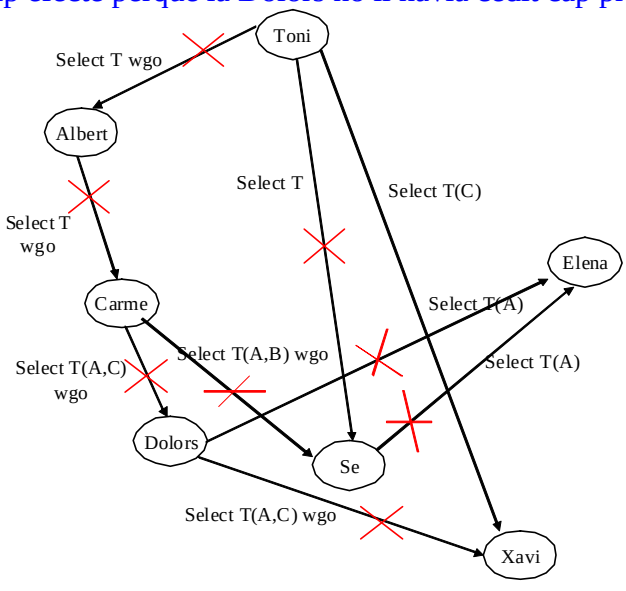
2.2) Considereu la taula professors(dni,nomProf,telefon), propietat d'en Toni. Suposeu també la seqüència de sentències següent relativa a autoritzacions sobre la taula professors. Cada sentència està numerada i s'indica el nom de l'usuari que vol executar-la.

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.

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

a) La sentència 8 no tindrà èxit perquè la Se no té els privilegis que vol passar al Xavi.
La sentència 12 no tindrà èxit perquè, a banda de la Dolors també el Xavi perdria part dels privilegis que té en el moment d'executar la sentència (RESTRICT).
La sentència 13 no té cap efecte perquè la Dolors no li havia cedit cap privilegi a la Se.



On T=Professors, A= dni, B=nomProf i C=telefon

La sentència 11 s'executarà amb èxit perquè el revocar el privilegi de la Se no provoca l'eliminació de cap altre privilegi (RESTRICT).

b) Xavi té privilegis de SELECT Professors(telefon).
Albert no té cap privilegi sobre Professors.

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;
```

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 solució en PostgreSQL seria:

```
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.

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

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

3.1) És una solució no incremental.
3.2) Solució incremental exemple 4 de les transparències de Disparadors.
3.3) Solució incremental exemple 4 de les transparències de Disparadors.

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;
T2: select * from R; select * from R;

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.

4.2) Suposem ara que la transacció T2 s'executa concurrentment amb una transacció T3, 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 gràNul 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 transaccions (R, RU, W, COMMIT), les operacions de petició i alliberament de reserves i l'ordre d'execució de totes aquestes peticions.
b) Quins horaris serials hi són equivalents?

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

a) Quines propietats de les transaccions es veuen compromeses?
b) Quines fonts d'informació necessita l'SGBD per reconstruir la BD?

4.1) Els resultats obtinguts varien en funció de l'ordre d'execució de les transaccions.
Si no hi ha encavalcament, es poden produir dos possibles resultats, en funció de l'horari serial sota consideració:

T1; T2: les dues *select* retornen el mateix resultat: {('a1',2), ('a2',4), ('a3',6)}. No es produeix cap interferència, donat que es tracta d'una execució serial.
T2; T1: les dues select retornen el mateix resultat: {('a1',1), ('a2',2)}. No es produeix cap interferència, donat que es tracta d'una execució serial.

En el cas d'execucions concurrents, les *selects* poden retornar resultats diferents, com a conseqüència de l'existència d'interferències:

T1	T2	Resultat
	select * from R	{(‘a1’,1), (‘a2’,2)}
insert into R values(‘a3’, 3)		
	select * from R	{(‘a1’,1), (‘a2’,2), (‘a3’,3)}
update R set b=b*2		
commit		
	commit	

S’està produïnt a T2 una interferència de tipus fantasma.

Un altre possible resultat seria:

T1	T2	Resultat
	select * from R	{(‘a1’,1), (‘a2’,2)}
insert into R values (‘a3’,3)		
update R set b=b*2		
	select * from R	{(‘a1’,2), (‘a2’,4), (‘a3’,6)}
commit		
	commit	

En aquest cas, a T2 s’està produïnt una interferència de lectura no repetible i una interferència de tipus fantasma.

Finalment, el darrer resultat possible seria:

T1	T2	Resultat
insert into R values (‘a3’,3)		
	select * from R	{(‘a1’,1), (‘a2’,2), (‘a3’,3)}
update R set b=b*2		
	select * from R	{(‘a1’,2), (‘a2’,4), (‘a3’,6)}
commit		
	commit	

En aquest cas, a T2 s’està produïnt una interferència de lectura no repetible

4.2) L’horari, en termes d’operacions de baix nivell i com a conseqüència d’aplicar reserves S,X amb nivell d’aïllament *repeatable read* sobre grànuls fila, quedaria tal i com s’indica a continuació:

T2	T3
L(f1,S)	
R(f1)	
L(f2,S)	
R(f2)	
	L(f1,S)
	R(f1)
	L(f1,X)
R(f1)	
R(f2)	
commit (U(f1), U(f2))	
	RU(f1)
	W(f1)
	commit (U(f1))

Les cel·les fosques denoten que T3 ha bloquejat la seva execució. L'horari no conté inferències i té associat com horari serial equivalent T2; T3.
Es considera correcte afegir les lectures de la informació de control (IC), però en cap cas es poden efectuar reserves sobre ella, degut al nivell d'aillament.

4.3) La propietat ACID de les transaccions que es veu vulnerada és la definitivitat (veure pàg 52 del pdf de material complementari), donat que s'estant perdent canvis confirmats sobre la BD.
Les fonts d'informació que necessita l'SGBD per fer la reconstrucció són una còpia de segurat de la BD que contingui un estat correcte de la BD i el dietari (en concret el seu contingut a partir del moment que es va fer la còpia de seguretat), tal i com s'explica a la pàgina 55 del pdf de material complementari.

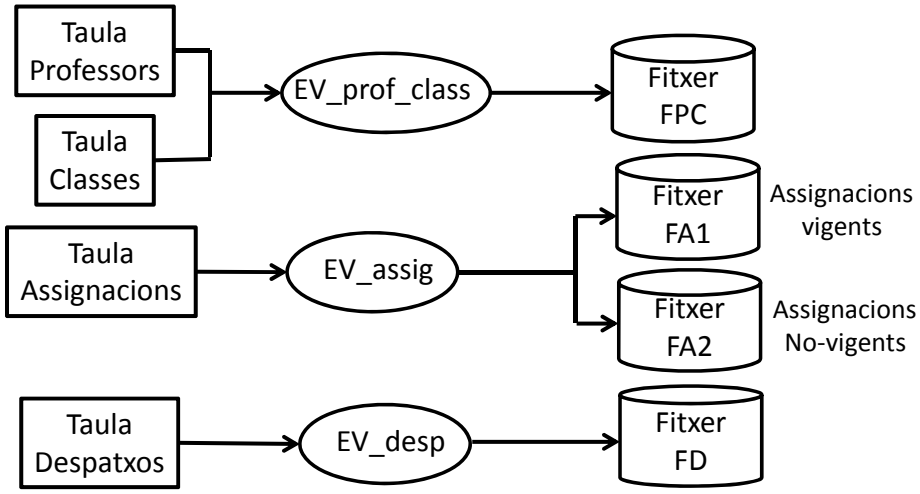
5) (2 punts) Considereu la base de dades de l'exercici 1.

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

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:



b) Com sabeu, els diferents tipus d'espais virtuals poden afavorir certs tipus de consultes. Tenint en compte només els espais virtuals abans identificats, digueu quina de les dues consultes següents es veu afavorida pels espais virtuals anteriors. 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;

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';
```

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.

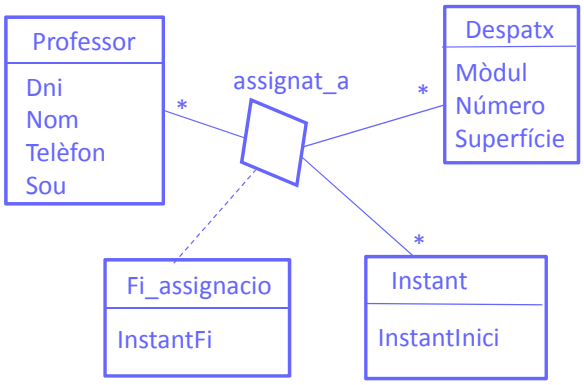
SELECT * FROM assignacions a WHERE a.dni > '444' AND a.modul='Omega'

Solució:

5.1) Clau externa Professor: Dni

Clau externa Despatx: Mòdul, Número

Clau externa Instant: InstantInici



5.2)

a) Espai virtual d'agrupació per professors i classes, espai virtual de taules per despatxos i espai virtual fragmentat per assignacions.

b) La primera consulta es veu afavorida per l'espai d'agrupació donat que es consulten les dades de les dues taules que s'emmagatzemen a l'espai. Cada pàgina de l'espai virtual emmagatzema les files que contenen les dades de cada professor i a continuació les files corresponents a les classes del professor. Per tant, per resoldre la consulta s'accedeix a cada una de les pàgines de l'espai virtual i es van llegint les files de cada pàgina. Com que les dades dels professors i classes s'emmagatzemen juntes el cost de resoldre la consulta és N, sent N el nombre de pàgines que emmagatzemen les dades dels professors més les seves classes.

b) En canvi, la segona consulta no es veu afavorida per l'espai virtual de taula i per l'espai virtual fragmentat donat que es consulten juntes les dades dels despatxos i les assignacions vigents a aquests despatxos. Per resoldre la consulta, cal accedir a cada una de les pàgines de l'espai virtual de taula que emmagatzema les dades dels despatxos, per obtenir les dades dels despatxos situats a l'edifici omega. Per cada despatx, cal recórrer les pàgines del fragment que emmagatzema les assignacions vigents per anar llegint les seves dades. El cost de resoldre la consulta és, N *M sent N el nombre de pàgines que emmagatzemen els despatxos i M el nombre de pàgines que emmagatzemen les assignacions vigents.

5.3)

-1000 assignacions de dni>’444’

-150 assignacions de modul=’Omega’

- 50 assignacions que compleixen les dues condicions

Alçada de l’arbre dni = 2

(10000/(157*2*0,7) = 10000/219 = 45 fulles), (45 apuntadors => 1 arrel)

$\lceil \log_{220} 10000 \rceil = 2$

Alçada de l’arbre mòdul = 2

(10000/(227*2*0,6) = 10000/272 = 36 fulles), (36 apuntadors => 1 arrel)

$\lceil \log_{273} 10000 \rceil = 2$

Cost en nombre de pàgines accedides = (ha+Fa) + (hb+Fb) + |RIDa>=’444’ i RIDb=’Omega’|

Cerca de RIDs per dni (ha+Fa):

ha (alçada de l’arbre) = 2

Fa (fulles de l’arbre que cal llegir) = (1000 / 219) -1 = 5 – 1 = 4

Cerca de RIDs per modul (hb+Fb):

ha (alçada de l’arbre) = 2

Fa (fulles de l’arbre que cal llegir) = (300 / 272) -1 = 2 – 1 = 1

Intersecció de RIDs = 50 RIDs

Accés a dades |RIDa>=’444’ i RIDb=’Omega’|:

número de files que compleixen les dues condicions = 50

Cost en nombre de pàgines accedides = (2 + 4) + (2 + 1) + 50 = 59 pàgines.

BD **11.1.2016**

Temps: 3 hores

Notes **25 gener tarda** **Revisió: 27 gener matí**

1. **(2 punts)** Donada la següent base de dades

```
CREATE TABLE professors
(dni char(50),
nomProf char(50) unique,
telefon char(15),
salari integer,
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(50),
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,
CHECK (instantFi > instantInici));
```

1.1 Escriviu una sentència SQL que obtingui el mòdul i el número dels despatxos amb una superfície de més de 15 metres quadrats, que han tingut exactament 6 assignacions i tal que totes les seves assignacions estan finalitzades (instantFi diferent de NULL).

```
SELECT d.modul, d.numero
FROM assignacions a natural inner join despatxos d
WHERE (d.superficie > 15)  AND
        NOT EXISTS (SELECT * FROM assignacions a1
                    WHERE d.modul = a1.modul AND
                          d.numero = a1.numero AND  a1.instantfi is null)
GROUP BY d.modul, d.numero
HAVING (COUNT(*) = 6);
```

1.2 Escriviu una seqüència d'operacions d'àlgebra relacional per obtenir els despatxos de més de X metres de superfície que han estat ocupats exclusivament per professors amb un salari inferior a Y (un despatx que sempre hagi estat buit *NO* ha de sortir en el resultat)

```
A = professors(salari >= Y)
B = assignacions * A
C = B[modul, numero]
/* a C tenim els despatxos que han estat ocupats en
algun moment per profes amb sou >= Y */
D = despatxos(superficie > X)
E = D * assignacions
F = E[modul, numero]
/* F te els despatxos amb una superficie > X i que
han estat ocupats en algun moment */
G = F - C
/* a G hem deixat els despatxos de superficie > X que han
estat ocupats en algun moment i tots els seus ocupants han
tingut un sou < Y */
```

1.3 Suposem les taules R(a,b) i S(x,b) i la consulta *SELECT * FROM R NATURAL INNER JOIN S*. Responeu de forma breu i justificada les següents qüestions:

a) És possible que en les tuples resultants de la consulta hi hagi alguna tupla en la que el valor l'atribut "a" sigui igual que el de l'atribut "x" ?. En cas afirmatiu poseu un exemple de les tuples de les taules inicials i del resultat del SELECT; en cas negatiu justifiqueu la resposta.

b) És possible que en les tuples resultants de la consulta, hi hagi alguna tupla en que el valor de l'atribut "R.b" o "S.b" sigui null ?. En cas afirmatiu poseu un exemple de les tuples de les taules inicials i del resultat del SELECT; en cas negatiu justifiqueu la resposta.

c) Suposem ara que la relació S tingui per esquema S(x,y), quantes tuples hi haurà en el resultat de la consulta en cas que R tingui nR tuples i S en tingui nS?

SOLUCIÓ:

1.3.a)

R.a	R.b
1	2

S.x	S.b
1	2

a	b	x
1	2	1

1.3.b) No és possible ja que R.b=S.b no és cert si un o tots dos tenen valor nul

1.3.c) N'hi haurà nRxnS

2. (2 punts) Donada la base de dades utilitzada en l'exercici 1. Suposeu per aquest exercici que tot professor té com a mínim una assignació. Direm que un professor està jubilat si totes les seves assignacions estan finalitzades (instantFi diferent de NULL).

2.1 Completeu el següent procediment emmagatzemat per fer que es jubili un professor (*prof*) en un instant determinat (*instantJubilació*) i retorni la quantitat de professors jubilats que queden posteriorment. És a dir, el procediment ha d'actualitzar totes les assignacions no finalitzades d'aquest professor amb l'instant que s'indica en els paràmetres d'entrada del procediment. En cas que l'instant de finalització sigui anterior o igual a l'instant d'inici d'alguna de les seves assignacions cal que el procediment llenci l'excepció 'Data incorrecta'.

```
CREATE OR REPLACE FUNCTION jubila(prof professors.dni%type,
                  instantJubilació assignacions.instantFi%type)
RETURNS  integer  AS $$
...
RETURN nb_jubilats;
...
END;
$$LANGUAGE plpgsql;
```

2.2 A diferència de l'apartat anterior, suposeu ara que per jubilar un professor s'executa una sentència update que finalitza les seves assignacions. Aquesta sentència activa un trigger que comprova la restricció d'integritat que diu que hi ha d'haver menys (estricte) professors jubilats que no jubilats, i insereix el professor a la taula següent:

```
CREATE TABLE profsJubilats(
   dni char(50),
   nomProf char(50) unique,
   telefon char(15),
   salari integer,
   instantJubilacio integer,
   primary key (dni));
```

Per implementar aquest comportament s'ha creat el trigger següent:

```
CREATE OR REPLACE FUNCTION jubilacio_cond() returns trigger as
$$
DECLARE
    nb_jubilats integer;
    nb_actius integer;
BEGIN
nb_jubilats := (select count(*) from profsJubilats);
nb_actius := (select count(*) from professors) - nb_jubilats;
IF (nb_jubilats < nb_actius)
THEN
   INSERT INTO profsJubilats
          (SELECT dni,nomProf,telefon,salari, NEW.instantFi
          FROM professors
          WHERE dni = NEW.dni AND dni NOT IN (SELECT dni
               FROM profsJubilats));
   RETURN new;
ELSE
  RAISE EXCEPTION 'Masses jubilats';
END IF;
END $$ language plpgsql;

CREATE TRIGGER jubila_si_pots
AFTER UPDATE OF instantFi ON assignacions
FOR EACH ROW EXECUTE PROCEDURE jubilacio_cond();
```

a) Donada la sentència update següent, digueu quantes vegades s'executarà el trigger. Justifiqueu la resposta.

```
UPDATE assignacions
SET instantFi=1000
WHERE dni= '111' AND instantFi is null;
```

b) Que passarà si volem jubilar a un professor ja jubilat? És a dir, que passarà la segona vegada que s'executi la sentència anterior. Justifiqueu la resposta.

c) Proposeu una nova implementació feta amb dos triggers que redueixi al màxim possible el nombre de vegades que s'executen les sentències subratllades quan s'activa el trigger.

SOLUCIÓ:

2.1)

```
CREATE OR REPLACE FUNCTION jubila(
   profe professors.dni%type,
   data assignacions.instantFi%type) RETURNS
  integer AS $$
DECLARE
 nb_jubilats integer;
BEGIN
   /*--jubilem professor---*/
   update assignacions
   set instantFi= InstantJubilació
   where dni= profe and instantFi is NULL;

   /*---comptem jubilats ----*/
```

```
   select count(*) into nb_jubilats
   from professors
   where dni not in (select dni
                from assignacions a2
                where a2.instantFi is null);
RETURN nb_jubilats;
EXCEPTION
        WHEN check_violation THEN
             raise exception ‘Data Incorrecta';
END;
$$LANGUAGE plpgsql;
```

2.2.a) S’executa tantes vegades com assignacions no finalitzades tingui el professor 111.

2.2.b) El trigger no s’activarà, ja que el primer Update haurà finalitzat totes les assignacions i, per tant, el segon update no farà cap actualització i, conseqüentment, no activarà el trigger.

2.2.c) Proposa una nova implementació que redueixi el nombre d’accessos a la base de dades.

```
CREATE OR REPLACE FUNCTION condicio() RETURNS TRIGGER AS $$
DECLARE
    nb_jubilats integer;
    nb_actius integer;
BEGIN
nb_jubilats := (select count(*) from profsJubilats)+1;
nb_actius := (select count(*) from professors) - nb_jubilats;
IF (nb_jubilats >= nb_actius) THEN
  raise exception 'Masses jubilats';
END IF;
RETURN null;
END $$ language plpgsql;

CREATE OR REPLACE FUNCTION jubilacio()RETURNS TRIGGER AS $$
BEGIN
   INSERT INTO profsJubilats
        (select dni,nomProf,telefon,NEW.instantFi
        from professors
        where dni = NEW.dni and dni not in (select dni
             from profsJubilats));
   RETURN null;
END $$ language plpgsql;

CREATE TRIGGER jubila_si_pots1
BEFORE UPDATE OF instantFi ON assignacions
FOR EACH STATEMENT EXECUTE PROCEDURE condicio();

CREATE TRIGGER jubila_si_pots2
AFTER UPDATE OF instantFi ON assignacions
FOR EACH ROW EXECUTE PROCEDURE jubilacio(); (s’admet també la
solució before row)
```

També és correcte (on per cert, s'ha pogut també suprimir el and dni not in (select dni from profsJubilats) del insert)

```
create table temp(primera boolean);

CREATE OR REPLACE FUNCTION condicio() returns trigger as $$
BEGIN
DELETE FROM temp;
INSERT INTO temp VALUES(true);
RETURN null;
END $$ language plpgsql;

CREATE OR REPLACE FUNCTION jubilacio_cond() returns trigger as $$
DECLARE
    nb_jubilats integer;
    nb_actius integer;
    primer boolean;
BEGIN
primer:= (SELECT primera FROM temp);
IF primer THEN
    nb_jubilats := (select count(*) from profsJubilats)+1;
    nb_actius := (select count(*) from professors) - nb_jubilats;
    Update temp set primera = false;
    IF (nb_jubilats < nb_actius)
    THEN
    INSERT INTO profsJubilats
        (SELECT dni,nomProf,telefon,NEW.instantFi
        FROM professors
        WHERE dni = NEW.dni );
    RETURN new;
    ELSE
    RAISE EXCEPTION 'Masses jubilats';
    END IF;

 END IF;
 RETURN new;
END $$ language plpgsql;

CREATE TRIGGER jubila_si_pots
BEFORE UPDATE OF instantFi ON assignacions
FOR EACH ROW EXECUTE PROCEDURE jubilacio_cond();

CREATE TRIGGER jubila_si_pots1
AFTER UPDATE OF instantFi ON assignacions
FOR EACH STATEMENT EXECUTE PROCEDURE condicio(); (s'admet també
before)
```

També és correcte:

```
create table numeroJubilats(nb_jubilats int, nb_actius int);

CREATE OR REPLACE FUNCTION jubilacio_stat() returns trigger as $$
DECLARE
    nb_jubilats integer;
    nb_actius integer;
BEGIN
    SELECT * into nb_jubilats, nb_actius FROM numeroJubilats;
    IF (nb_jubilats >= nb_actius) THEN
        raise exception 'Masses jubilats';
    END IF;
    RETURN null;
END $$ language plpgsql;

CREATE OR REPLACE FUNCTION jubilacio_row() returns trigger as $$
DECLARE
    nb_jubilats integer;
    nb_actius integer;
BEGIN
    IF (NEW.dni NOT IN (SELECT dni FROM profsJubilats))
    THEN
        INSERT INTO profsJubilats
             (select dni,nomProf,telefon,NEW.instantFi
             from professors
             where dni = NEW.dni);
        UPDATE numeroJubilats
        SET nb_jubilats= nb_jubilats+1, nb_actius= nb_actius-1;
    END IF;
    RETURN new;
END $$ language plpgsql;

CREATE TRIGGER jubila_si_pots1
BEFORE UPDATE OF instantFi ON assignacions
FOR EACH STATEMENT EXECUTE PROCEDURE jubilacio_stat();

CREATE TRIGGER jubila_si_pots
BEFORE UPDATE OF instantFi ON assignacions
FOR EACH ROW EXECUTE PROCEDURE jubilacio_row(); (estaria també bé
si fos after row).
```

3. **(2 punts)** Considereu el següent horari, on les operacions OpL(G) són R(G) o RU(G).

T1	T2	T3	T4	T5
		OpL(A)		
				RU(D)
		OpL(C)		
				RU(F)
		OpL(F)		
RU(A)				
	RU(C)			
R(F)				
	W(C)			
W(A)				
			RU(C)	
			R(F)	
			W(C)	
				W(D)
				W(F)
		Op		
				Commit
			Commit	
Commit				
	Commit			
		Commit		

3.1 Proposeu una operació per cadascuna de les operacions OpL i Op per tal de que es provoquin dues i només dues interferències. Indiqueu per cada interferència: el nom de la interferència, entre quines dues transaccions es produeix i quin/s gràNul/s implica.

3.2 Doneu el graf de precedències associat a l'horari una vegada instanciades les operacions.

3.3 Quin és el mínim nivell d'aïllament necessari per tal de que s'evitin les interferències descrites a l'apartat 3.1?

3.4 Escollint el nivell d'aïllament determinat a l'apartat 3.3, com quedaria l'horari? El nou horari ha d'incloure, a més de les operacions que executen les transaccions (R, RU, W, COMMIT), les operacions de petició i alliberament de reserves (LOCK, UNLOCK), i l'ordre d'execució de totes aquestes operacions. Quan més d'una transacció espera per un mateix gràNul, considereu que la política de la cua és FIFO (First In, First Out).

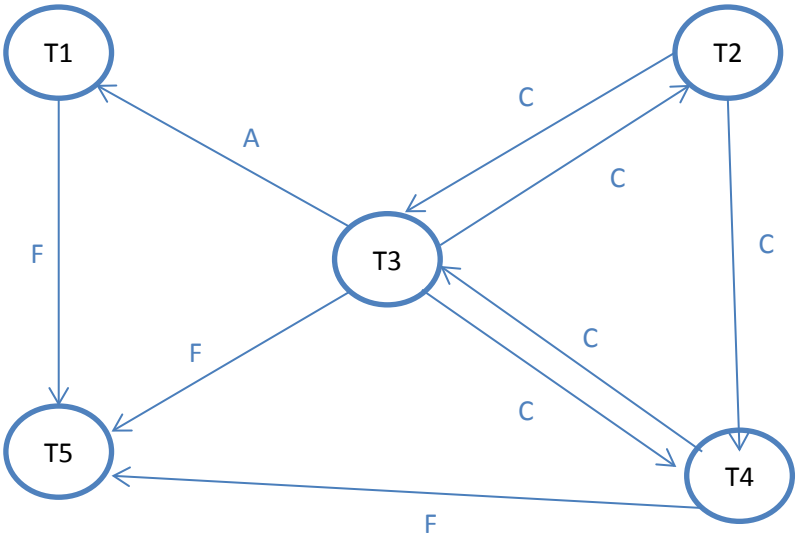
SOLUCIÓ:

2.1) OpL(A) serà R(A), OpL(F) serà R(F). Per OpL(C) hi ha dues solucions:

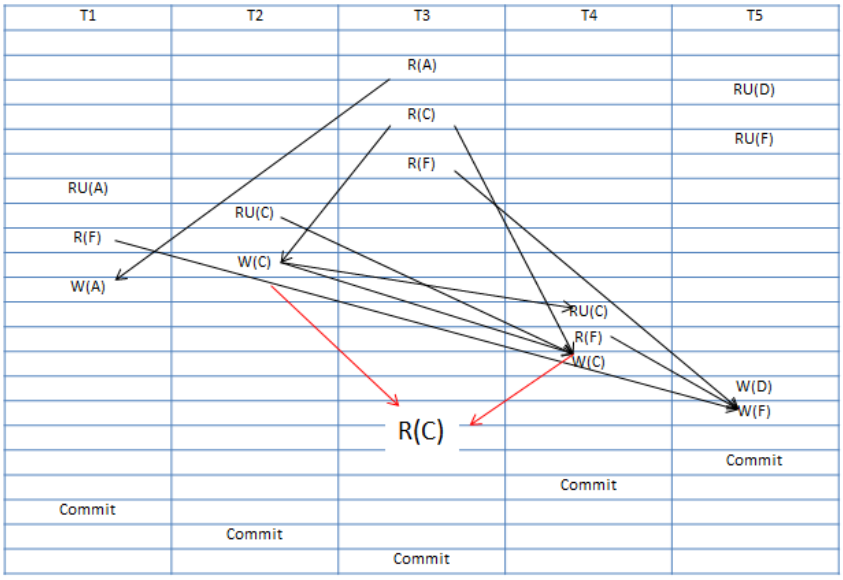
R(C) amb Op R(C) que provoquen dues interferències de lectura no repetible

RU(C) amb W(C) que provoquen dues interferències d'actualització perduda

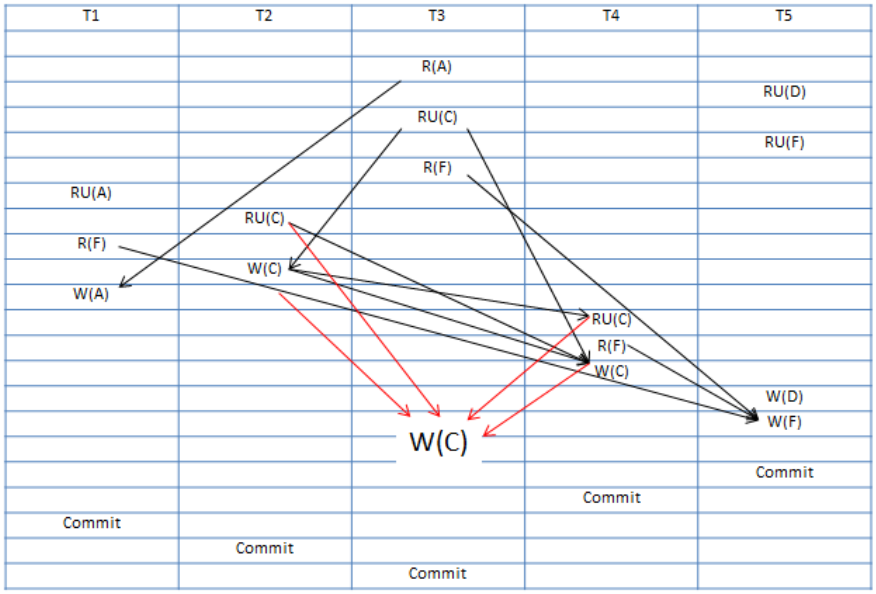
2.2) El graf en la solució de R(C) és el següent:



En el cas de que Op és R(C) surt de l'horari:



En el cas de que Op és W(C) surt de l'horari:



2.3) En la solució de R(C) és Repeteable Read, en la solució de W(C) és Read Uncommitted

2.4)

Horari per a la solució amb R(C)				
T1	T2	T3	T4	T5
		L(A,S)		
		R(A)		
				L(D,X)
				RU(D)
		L(C,S)		
		R(C)		
				L(F,X)
				RU(F)
		L(F,S)		
		xxx		
L(A,X)		xxx		
xxx	L(C,X)	xxx		
xxx	xxx	xxx	L(C,X)	
xxx	xxx	xxx	xxx	W(D)
xxx	xxx	xxx	xxx	W(F)
xxx	xxx	xxx	xxx	C+U(D,F)
xxx	xxx	R(F)	xxx	
xxx	xxx	R(C)	xxx	
xxx	xxx	C+U(C,A,F)	xxx	
RU(A)	xxx		xxx	
	RU(C)		xxx	
L(F,S)			xxx	
R(F)			xxx	
	W(C)		xxx	
W(A)			xxx	
c+U(A,F)			xxx	
	C+U(C)		xxx	
			RU(C)	
			L(F,S)	
			R(F)	
			W(C)	
			C+U(C,F)	

Horari per a la solució amb W(C)				
T1	T2	T3	T4	T5
		R(A)		
				L(D,X)
				RU(D)
		L(C,X)		
		RU(C)		
				L(F,X)
				RU(F)
		R(F)		
L(A,X)				
RU(A)				
	L(C,X)			
R(F)	xxx			
W(A)	xxx			
	xxx		L(C,X)	
	xxx		xxx	W(D)
	xxx		xxx	W(F)
	xxx	W(C)	xxx	
	xxx		xxx	C+U(D,F)
c+U(A)	xxx		xxx	
	xxx	C+U(C)	xxx	
	RU(C)		xxx	
	W(C)		xxx	
	C+U(C)		xxx	
			RU(C)	
			R(F)	
			W(C)	
			C+U(C)	

4. (2 punts) Tenim una base de dades bancària amb una taula de comptes que té una cardinalitat de 2.000.000 files. L'esquema d'aquesta taula és:

```
Comptes(numCompte, dniClient, saldo)
```

El número de compte és clau primària de la taula i els números de compte són enters repartits uniformement entre 1 i 2.000.000. Es demana el següent, justificant breument les respostes:

4.1 Suposeu que existeix un índex no agrupat amb ordre d=150 per l'atribut numCompte. Quin és el nombre mínim i màxim de pàgines que ocupa l'índex?.

4.2 Donada la consulta següent:

```
SELECT * FROM comptes WHERE saldo>300;
```

Suposant que ara es disposa d'un índex no agrupat per saldo amb d=150, que els nodes de l'índex estan plens al 85% de la seva capacitat, l'alçada de l'índex és 3 i que el factor de bloqueig del fitxer de dades és de 10 files per pàgina en mitjana. Calculeu el nombre màxim de files que poden complir la condició saldo>300 per a que sigui més útil resoldre-la emprant l'índex no agrupat per saldo en lloc de fer un recorregut seqüencial del fitxer de dades.

SOLUCIÓ:

4.1) Arbre B+ amb nombre mínim de nodes => quan els nodes de l'arbre estan plens amb 2d valors (si d=150 nodes plens amb 2d valors = 300 valors per node)

Nº de fulles= 2.000.000 de num.comptes / 300 valors per node = 6667 fulles => 6667 apuntadors a fulles

Nº nodes interns= 6667 apuntadors totals / 301 apuntadors per node= 23 nodes interns

+ 1 arrel amb 22 valors i 23 apuntadors

nombre total de nodes que ocupa l'índex= 1 arrel + 23 nodes interns + 6667 fulles =6691 nodes

Arbre B+ amb nombre màxim de nodes => quan els nodes de l'arbre estan plens amb d valors

Nº de fulles= 2.000.000 / 150 = 13333 fulles => 13.333 apuntadors a fulles

Nº nodes interns= 13.333 apuntadors totals / 151 apuntadors per node= 88 nodes interns

+ 1 arrel amb 87 valors i 88 apuntadors

En aquest cas, ens quedem en les dues divisions amb el nombre enter per sota, perquè si ens quedem amb el superior tindríem algun node per sota de 150 valors, cosa que faria que no es complís la condició que el nombre de valors ha d'estar per sobre de d.

nombre total de nodes que ocupa l'índex= 1 arrel + 88 nodes interns + 13.333 fulles =13422 nodes

4.2) h=3

nodes de l'arbre plens amb 85% de 2d=255 valors per node en mitjana.

Cost de resoldre la consulta emprant l'índex no agrupat per saldo:

h + F + Comptes(saldo>300)

Sigui x el nombre de valors que compleixen la condició saldo>300

Cost de resoldre la consulta emprant l'índex:

$3 + \lceil x/255 \rceil - 1 + x$ (

Cost de resoldre la consulta fent un recorregut seqüencial del fitxer de dades:

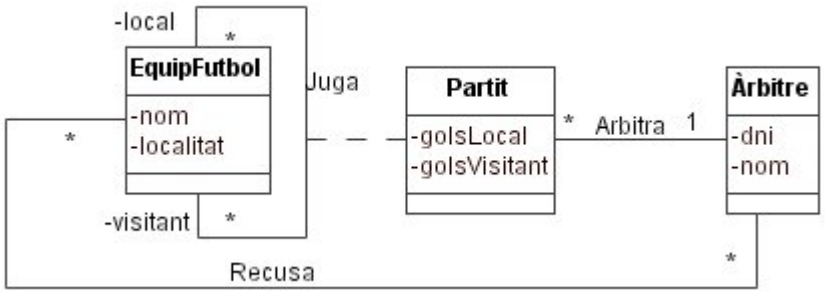
2.000.000/10

Per tal què l´índex sigui millor que el recorregut seqüencial s'ha de complir:

$3 + \lceil x/255 \rceil - 1 + x < 2.000.000/10$

x<= 199216,7 files => si hi ha menys de 199216 files amb saldo>300 és millor emprar l'índex per saldo.

5. (2 punts) Considereu el disseny conceptual en UML següent:



La clau externa de la classe *EquipFutbol* és el *nom* i la d'*Àrbitre* és el *dni*.

5.1 Transformeu el disseny conceptual anterior al model relacional.

EquipFutbol(nom, localitat)

Arbitre(dni, nom)

Partit(equipLocal, equipVisitant, golsLocal, golsVisitant, dniArbitre)

{equipLocal} referencia EquipFutbol

{equipVisitant} referencia EquipFutbol

{dniArbitre} referencia Arbitre

Recusacio(nomEquip, arbRecusat)

{nomEquip} referencia EquipFutbol

{arbRecusat} referencia Arbitre

5.2 Expliqueu com s'implementarien en l'SQL la restricció següent:

No hi pot haver dos (o més) partits que tinguin el mateix equip local i el mateix equip visitant (però s'ha de permetre que hi pugui haver dos partits amb els mateixos equips i els rols intercanviats).

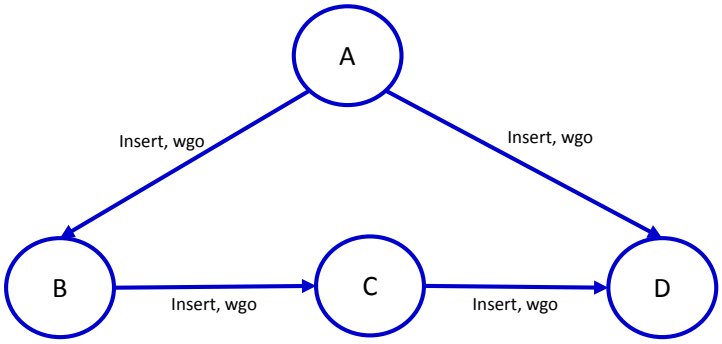
Amb la restricció de taula PRIMARY KEY(equipLocal, equipVisitant) a la taula Partit.

5.3 Considereu que l'usuari A és el propietari de la relació EquipFutbol, i cap altre usuari en té privilegis. Suposeu les sentències següents:

A: GRANT INSERT ON EquipFutbol TO B WITH GRANT OPTION;
A: GRANT INSERT ON EquipFutbol TO D WITH GRANT OPTION;
B: GRANT INSERT ON EquipFutbol TO C WITH GRANT OPTION;
C: GRANT INSERT ON EquipFutbol TO D WITH GRANT OPTION;
B: REVOKE INSERT ON EquipFutbol FROM C RESTRICT;

a) Doneu el diagrama d'autoritzacions just abans d'executar el REVOKE.

b) Digueu quin és exactament el conjunt d'usuaris que tenen el privilegi d'INSERT sobre la taula EquipFutbol després de l'execució de la sentència REVOKE.



El REVOKE s'executa normalment, perquè només C perd el privilegi, ja que D també el reb de l'usuari A.
El conjunt d'usuaris serà A, B i D.

5.4 Donades les taules amb les extensions següents:

```
EquipFutbol(nom,       localitat)
            Barça      Barcelona
            R. Madrid  Madrid

Jugador    (dni, nom)
           111  Gerard
           222  Sergio

Contracte(EquipFutbol, Jugador, data_inici, data_fi,    quantitat)
           Barça          111     01-01-2010  31-12-2011    1000000
           Barça          111     01-01-2012     NULL      10000000
           R.Madrid       222     01-01-2010  31-12-2015    5000000
```

Suposant que:

- Hi ha una clau forana Contracte.Jugador que s'ha definit com "ON DELETE SET NULL" i "ON UPDATE CASCADE".
- Hi ha una clau forana Contracte.EquipFutbol que s'ha definit com "ON DELETE CASCADE" i "ON UPDATE RESTRICT".

Digues quina serà l'extensió de la taula Contracte si s'executen les sentències següents:

a) DELETE FROM Jugadors WHERE dni=111;

No es pot perquè el dni forma part de la clau primària de Contracte i no pot contenir valors nuls.

b) DELETE FROM EquipFutbol WHERE nom='Barça';

Contracte(EquipFutbol, Jugador, data_inici, data_fi, quantitat)
R. Madrid 222 1-1-2010 NULL 5000000

c) UPDATE Jugadors SET dni=333 WHERE dni=111;

Contracte(EquipFutbol, Jugador, data_inici, data_fi, quantitat)
Barça 333 1-1-2010 31-12-2011 1000000
Barça 333 1-1-2012 NULL 10000000
R. Madrid 222 1-1-2010 NULL 5000000

d) UPDATE EquipFutbol SET nom='FCB' WHERE nom='Barça';

No es pot perquè l'equip Barça està a la taula Contracte, no es podrà fer el canvi.

BD **23.6.2016**

Temps: 2 hores i 30 minuts

Notes 29 juny tarda **Revisió:** 30 juny tarda

1. **(2 punts)** Suposeu la base de dades següent

```
create table clans(
idClan int primary key,
nomClan varchar(30) unique,
instantCreacio int check(instantCreacio>0),
copes int);

create table jugadors(
idJugador int primary key,
nomJugador varchar(30),
alies char(10) unique,
copes int);

create table pertinences(
idJugador int,
idClan int,
instantInici int check(instantInici>0),
instantFi int check(instantFi>0),
posRanquing int check(posRanquing>0),
primary key(idJugador,idClan,instantInici),
foreign key(idJugador) references jugadors,
foreign key(idClan) references clans,
check (instantFi>instantInici));
// un jugador està actiu en un clan si instantFi té valor nul
// un jugador no pot pertànyer a més d'un clan en un mateix
// instant

create table tropes(
idTropa int,
idJugador int not null,
tipus char(10) check (tipus in ('arquera', 'gegant', 'bruixa',
'golem')),
primary key(idTropa),
foreign key(idJugador) references jugadors);
```

1.1 Escriviu una sentència SQL que obtingui els noms dels clans que tenen com a mínim un jugador actiu i que no en tenen cap que tingui tropes de tipus arquera.

```
select distinct c.nomClan
from clans c, pertinences p
where c.idClan = p.idClan and
     p.instantFi is null and
     not exists
          (select *
          from pertinences p, tropes t
          where c.idClan = p.idClan and
               p.idJugador = t.idJugador and
               p.instantFi is null and
               t.tipus = 'arquera')
```

1.2 Escriviu una seqüència d'operacions d'àlgebra relacional per obtenir els noms dels jugadors que mai han format part d'un clan i que no tenen tropes.

```
A = jugadors[idJugador]
B = tropes[idJugador]
C = A – B
D = pertinences[idJugador]
E = C – D
G = E * jugadors
H = G[nomJugador]
```

1.3 Indiqueu quina és la cardinalitat de la relació resultat de l'execució les consultes següents. Considereu que les taules de la base de dades tenen C clans, J jugadors, P pertinences, T tropes (C, J, P i T són enters més grans que 0). Tingueu en compte per aquest exercici que tot jugador té una o més tropes, i que cada jugador i cada clan tenen una o més pertinences. Raona breument la resposta.

<table>
<tr><td>a) select distinct alies
from jugadors</td><td rowspan="2">d) select count(*)
from clans natural join pertinences
group by idClan</td></tr>
<tr><td>b) select *
from clans, jugadors</td></tr>
<tr><td>c) select *
from jugadors natural join tropes</td><td>e) A = jugadors * pertinences
R = A * clans</td></tr>
</table>

a) J, ja que l'atribut alies és UNIQUE.

b) C*J, ja que és un producte cartesià

c) T, segons l'enunciat, no hi ha jugadors sense tropes.

d) C, en un group by hi ha un resultat per cada grup.

e) P, només hi ha un jugador i un clan per cada pertinença.

2. (2 punts) Suposeu les taules següents:

```
create table R (a int, b int, primary key(a,b));
create table S (c int, d int, a int, b int, primary key(c,d), foreign key(a,b) references R);
```

2.1 En cas de tenir definit el trigger següent:

```
create function insertaR1() returns trigger AS $$
  begin
  if (new.a*new.b)> 10
  then insert into R values(new.a-1, new.b+1);
  end if;
  return null;
  end;
$$ language plpgsql;

create trigger insertRtrig1
after insert on R
for each row execute procedure insertaR1();
```

a) Suposeu que la taula R és buida. Digueu quin serà l'estat de la taula R després d'executar la sentència següent:

```
insert into R values(11,1);
```

b) Suposeu que la taula R és buida. Digueu quin serà l'estat de la taula R després d'executar la mateixa sentència anterior en cas que el trigger fos "before":

2.2 En cas de tenir definit el trigger següent:

```
create function insertaR2() returns trigger AS $$
  begin
  insert into R (select a, NEW.b
                 from R
                 where b=NEW.a and
                    (a, new.b) not in (select a,b from R));
  return null;
  end;
$$ LANGUAGE plpgsql;

create trigger insertRtrig2
after insert on R
for each row execute procedure insertaR2();
```

Suposeu que la taula R és buida. Digueu quin serà l'estat de la taula R després d'executar les sentències següents.

```
insert into R values(1,2);
insert into R values(2,3);
insert into R values(3,4);
```

2.3 Suposeu el procediment emmagatzemat següent:

```
create function esborra_compta_R(la_a R.a%type) returns void as $$
begin
   delete from R where a=la_a;
   return;
end;
$$ language plpgsql;
```

Modifiqueu el codi del procediment per tal que generi les excepcions següents:

- Excepció 1. No es pot esborrar les files de R perquè existeixen files de S que hi referencien.
- Excepció 2. No hi ha cap fila a R on l'atribut a correspon al valor que es passa com a paràmetre d'entrada.

SOLUCIÓ

2.1.a (11,1), (10,2), (9,3), (8,4), (7,5), (6,6), (5,7), (4,8), (3,9), (2,10), (1,11), (0,12)

2.1.b No s'inseriria cap fila, per tant la taula R seria buida perquè el procediment fa "Return null"

2.2 (1,2)
(2,3), (1,3)
(3,4), (2,4), (1,4)

2.3

```
create or replace function esborra_compta_R(la_a R.a%type) returns void as $$
begin
        delete from R where a=la_a;
        if not found then raise exception 'Excepció 2'; end if;
        return;
        exception
        when raise_exception then
                raise exception '%',SQLERRM;
        when foreign_key_violation then
                raise exception 'Excepció 1';
        end;
$$ language plpgsql;
```

3. (2 punts) Suposeu la base de dades de l'exercici 2 amb l'extensió següent:

R	a	b
	1	2
	2	3
	4	5

S	c	d	a	b
	1	2	2	3
	3	4	1	2
	5	6	1	2
	7	8	2	3

Suposeu l'horari següent, on el gràNul és la fila:

	T1	T2	T3	T4
1			update S set a = 2, b=3 where c=5 and d=6;	
2				select * from S where c=5 and d=6;
3	select a from S where c=5;			
4		update S set a=4, b=5 where c=5 and d=6;		
5	select sum(a) from S where c=5;			
6				select * from S where c=3 and d=4;
7			update S set a=2, b=3 where c=3 and d=4;	
8	commit			
9		commit		
10			commit	
11				commit

3.1 Digueu quines interferències es produeixen. Per cada interferència indiqueu:

- El tipus d'interferència
- Quines dues transaccions hi estan implicades
- La transacció que es veu afectada
- Les operacions i gränuls implicats.
- El nivell mínim d'aïllament en que s'ha d'executar cada transacció implicada per evitar la interferència.

3.2 Suposeu que cada transacció s'executa en nivell d'aïllament serialitzable i que la política d'assignació de reserves és First In First Out (FIFO). Indiqueu:

- Quins són els horaris serials equivalents a l'horari resultant d'aplicar les reserves corresponents al nivell d'aïllament. NO és necessari donar l'horari amb els LOCKS i UNLOCKS, només es demana els horaris serials equivalents i que es raoni la resposta.
- Quin serà contingut de la taula S després d'executar cadascun dels horaris serials que hagueu identificat en l'apartat anterior.

SOLUCIÓ

3.1.-

- Lectura no repetible
- Entre T1 i T2, Afectada T1
- ops 3,4,5, granul (5,6)
- T1 repeteable read, T2 read uncommitted

- anàlisi inconsistent
- Entre T3 i T4, Afectada T4
- ops 1,2,6,7, granuls (3,4) i (5,6)
- T3 read uncommitted, T4 repeteable read

3.2.-

No es demanava, però l'horari resultant d'aplicar reserves és:

	T1	T2	T3	T4
			L(5,6,X)	
			RU(5,6)	
			W(5,6)	
				L(5,6,S)
	L(5,6,S)			
		L(5,6,X)		
			L(3,4,X)	
			RU(3,4)	
			W(3,4)	
			commit+U	
				R(5,6)
	R(5,6)			
	R(5,6)			
				L(3,4,S)
				R(3,4)
	commit+U			
				commit+U
		RU(5,6)		
		W(5,6)		
		commit+U		

Els horaris serials equivalents a l'horari resultant són T3;T1;T4;T2 i T3;T4;T1;T2

Els dos horaris donen el mateix resultat, sinó no serien horaris equivalents. El resultat és:

S	c	d	a	b
	1	2	2	3
	3	4	2	3
	5	6	4	5
	7	8	2	3

4. (2 punts) Considereu un índex en forma d'arbre B+ d'ordre 5 i alçada 3. Com sabreu, el nivell fulla conté tots els apuntadors a les files del fitxer de dades.

a) Suposeu que l'arbre està ple al 100% i que és no agrupat. Suposeu també que hi ha una fila del fitxer de dades per a cada valor indexat 1,2,3, ... F a on F és el número de files. Quants accessos a pàgines de l'índex caldrà fer per localitzar totes les files en el rang [95,137]? Raoneu la resposta

b) Considereu el mateix arbre B+ i les mateixes circumstàncies que a l'apartat anterior però suposeu ara que l'índex és agrupat. Com canviaria el resultat anterior? Raoneu la resposta.

c) Considereu el mateix arbre B+ i les mateixes circumstàncies que a l'apartat a) però suposeu ara que cal obtenir totes les files amb valors múltiples de 10. Digues quin és el nombre mínim d'accessos a pàgines de l'índex que caldrà fer per localitzar-les? Doneu el resultat en funció del número de files F.

d) Considereu el mateix arbre B+ i les mateixes circumstàncies que a l'apartat a) Determineu el valor F tenint en compte que l'arbre està ple al 100%. Raoneu la resposta

e) Suposeu ara que l'arbre està el més buit possible. Determineu el valor F. Raoneu la resposta

SOLUCIÓ

a) Només cal accedir a l'arrel, a un node intermedi i 5 nodes fulla (els que contenen els valors següents 91-100,101-110,111- 120,121-120,131-140). Per tant 7 accessos a pagines (nodes) de l'índex.

b) Només caldrà accedir a 3 pàgines de l'índex (arrel, node intermedi, primer node fulla), la resta de registres és localitzaran a través de les pròpies pagines de dades.

c) 1 (arrel) + 1 node intermedi + F/10 (cal accedir a tots els nodes fulla)

d) Tenint en compte que l'arbre està ple al 100%:
Arrel 2d +1 apuntadors = 11
Node intermedi 2d+1 apuntadors = 11
Node fulla 2d valors = 10
F= 1210

e) Arrel 2 apuntadors = 2
Node intermedi d+1 apuntadors = 6
Node fulla d valors = 5
F= 60

5. (2 punts) Considerant les taules següents:

departaments(num_dpt, nom_dpt)

projectes(num_proj, nom_proj, localització)

empleats(nif, nom, sou, num_dpt)

{num_dpt} referencia departaments

{num_dpt} atribut no nul

assignacions(nif, num_proj, feina)

{nif} referencia empleats

{num_proj} referencia projectes

5.1 Doneu el disseny conceptual en UML que es correspondria amb la traducció de les taules anteriors.

5.2 Privilegis

a) Escriviu les sentències SQL perquè l'usuari X (el propietari de les taules) permeti que es puguin fer les operacions següents sobre les taules anteriors (Definiu vistes si ho considereu necessari):

- L'usuari A pot modificar tots els atributs de totes les taules excepte els atributs de la taula departaments, i no pot canviar a un empleat de departament. Pot passar aquests privilegis a altres usuaris.
- L'usuari B pot modificar tots els atributs de la taula projectes, i pot modificar l'assignació d'un empleat d'un projecte a un altre. Pot passar aquests privilegis a altres usuaris.
- L'usuari C pot modificar qualsevol dels atributs de la taula projectes, però només dels projectes amb empleats del departament número 3.

b) Escriviu la sentència SQL perquè l'usuari A permeti fer l'operació següent a B:

- L'usuari B pot modificar la taula projectes. Pot passar aquests privilegis a altres usuaris.

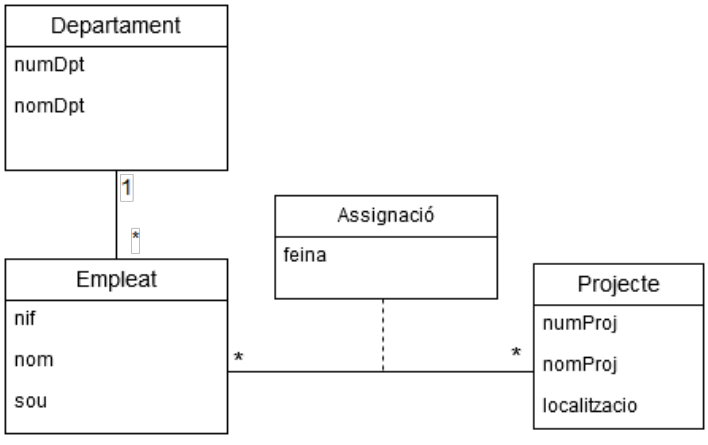
c) Escriviu una sentència en la que X desautoritza A per modificar la taula projectes en mode RESTRICT, i doneu el diagrama d'autoritzacions després d'executar tota la seqüència de sentències anteriors.

5.3 Suposeu que la taula departaments està emmagatzemada en un espai virtual de taula. Dibuixeu l'estructura física de la pàgina 1, del fitxer que conté aquesta taula, on hi ha emmagatzemades 3 files (fila1, fila2, fila3). Heu de tenir en compte que els RIDs d'aquestes files són respectivament [1,1], [1,3], [1,2].

5.4 Expliqueu breument què vol dir que el SQL és un llenguatge declaratiu.

SOLUCIÓ

5.1



5.2.a

L'usuari A pot modificar tots els atributs de totes les taules excepte els atributs de la taula departaments, i no pot canviar a un empleat de departament. Pot passar aquests privilegis a altres usuaris

X: GRANT UPDATE ON projectes, assignacions TO A WITH GRANT OPTION;
GRANT UPDATE(nif, nom, sou) ON empleats TO A WITH GRANT OPTION;

L'usuari B pot modificar tots els atributs de la taula projectes, i pot modificar l'assignació d'un empleat d'un projecte a un altre. Pot passar aquests privilegis a altres usuaris.

X: GRANT UPDATE ON projectes TO B WITH GRANT OPTION;
GRANT UPDATE(num_proj) ON assignacions TO B WITH GRANT OPTION;

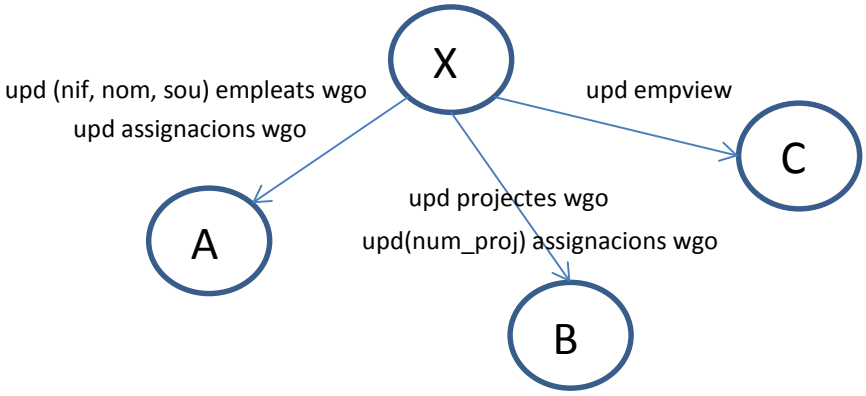
L'usuari C pot modificar qualsevol dels atributs de la taula projectes, però només dels projectes amb empleats del departament número 3.

Per poder donar aquest privilegi cal crear la vista següent i autoritzar el fer updates de la vista. El que passa és que es tracta d'una vista no actualitzable perquè té joins. En la correcció de l'examen es va considerar correcta la solució amb la definició de la vista i el grant i també la solució que diu que no és possible donar aquests privilegis pel motiu indicat abans.

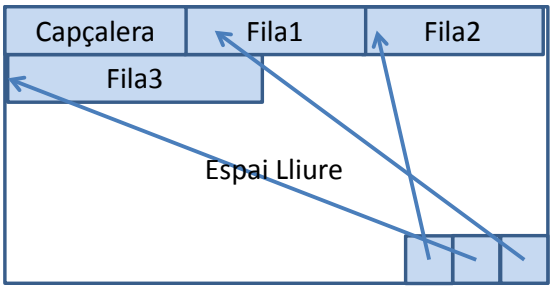
```
X: CREATE VIEW empview (num_proj, nom_proj, localitzacio) AS
        SELECT p.num_proj, p. nom_proj, p.localitzacio
        FROM projectes p, assignacions a, empleats e
        WHERE p.num_proj=a.num_proj and a.nif=e.nif and e.num_dpt=3;
   GRANT UPDATE ON empview TO C;
```

5.2.b
A: GRANT UPDATE ON projectes TO B WITH GRANT OPTION;

5.2.c
X: REVOKE UPDATE ON projectes FROM A RESTRICT;



5.3



5.4

- És un llenguatge que descriu el QUÈ es vol obtenir en el resultat, i no COM s'ha de resoldre la consulta.

Temps: 3 hores

Notes 26 gener matí **Revisió:** 27 gener matí

1. **(2 punts)** Suposeu una base de dades amb les taules següents:

```
CREATE TABLE professors (
       dni integer PRIMARY KEY,
       nom_prof char(50),
       categoria char(10),
       sou integer);

CREATE TABLE materies(
       id_mat integer PRIMARY KEY,
       nom_mat char(50) UNIQUE);

CREATE TABLE assignatures (
       id_mat integer REFERENCES materies,
       codi_assig integer,
       nom_assig char(50),
       PRIMARY KEY (id_mat, codi_assig));
       // Feu atenció que una assignatura s'identifica per la matèria de què
       tracta i un codi d'assignatura.

CREATE TABLE coordinadors (
       id_mat integer,
       codi_assig integer,
       curs char(20),
       dni integer REFERENCES professors,
       PRIMARY KEY (id_mat, codi_assig, curs),
       FOREIGN KEY (id_mat, codi_assig) REFERENCES assignatures);
       // Taula que guarda els coordinadors d'assignatura de cada curs.

CREATE TABLE especialistes (
       dni integer REFERENCES professors,
       id_mat integer REFERENCES materies,
       PRIMARY KEY (dni, id_mat));
       // Taula que guarda els especialistes de cada matèria.
```

1.1 Donar una sentència SQL per obtenir el nom dels professors que no han coordinat cap assignatura de la matèria amb nom Bases de Dades.

```
select distinct p.nom_prof
from professors p
where not exists (select *
                  from coordinadors c natural inner join materies m
                  where p.dni = c.dni and m.nom_mat='Bases de Dades')
```

1.2 Donar una sentència SQL per obtenir els noms dels professors que han coordinat totes les assignatures de la matèria amb nom Matemàtiques.

```
 select distinct nom_prof
 from professors p natural inner join materies m natural inner join coordinadors c
where m.nom_mat ='Matemàtiques'
group by p.dni, m.id_mat
having count(distinct c.codi_assig) = (select count(*) from assignatures a
                                       where a.id_mat=m.id_mat)
```

1.3 Donar una seqüència d'operacions de l'àlgebra relacional per obtenir els els noms de les matèries tals que hi ha algun professor que és especialista d'aquesta matèria i no ha sigut coordinador de cap assignatura d'aquesta matèria.

A = coordinadors[dni, id_mat]
B = especialistes – A
C = B[id_mat]
D = C* materies
R = D[nom_mat]

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

2.1 Suposeu l'asserció:

```
CREATE assertion LimitC check (
NOT EXISTS (SELECT * FROM professors
            WHERE categoria = 'C' AND
                  sou > (SELECT avg(sou) FROM professors));
```

Digueu quines sentències d'actualització poden violar l'asserció anterior. Per cada sentència cal indicar:

- Si és una sentència insert, delete o update.
- La taula sobre la que s'aplica (i en cas de sentències update a més cal indicar el/s atributs rellevants).
- Una extensió de la taula i un exemple de sentència que produeixi la violació suposant que la taula tingui aquesta extensió.

2.2 Es vol garantir que tot professor assignat com a coordinador d'una assignatura sigui especialista en la matèria de l'assignatura. Una possible implementació d'aquesta restricció amb un disparador és la següent:

```
CREATE OR REPLACE FUNCTION restriccio() RETURNS Trigger AS $$
BEGIN
   IF (NOT EXISTS (SELECT * FROM especialistes
          WHERE dni = NEW.dni AND id_mat = NEW.id_mat)) THEN
   RAISE EXCEPTION 'Professor no especialista en la matèria';
   END IF;
   RETURN NEW;
END;
$$language plpgsql;

CREATE TRIGGER restriccio BEFORE INSERT ON coordinadors
FOR EACH ROW EXECUTE PROCEDURE restriccio();
```

Es demana:

a) Digueu si aquest disparador cobreix totes les sentències d'actualització que poden violar la restricció. En cas afirmatiu expliqueu perquè no cal cap altre disparador més, en cas negatiu digueu per quines sentències d'actualització i taules s'haurien de definir els disparadors que falten (i en cas de sentències update a més cal indicar el/s atributs rellevants).

b) Digueu si es pot o no 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 pot fer.

SOLUCIÓ:

2.1) Les sentències d'actualització que poden violar la restricció són:

Esdeveniment	Taula	Atributs	Extensió	Sentència
INSERT	professors	NA	1 A 120 2 B 100 3 C 110 On la mitjana dels sous és 110 (=330/3).	INSERT INTO TABLE professors VALUES (4, 'C', 230); El professor 4 passa a tenir un sou de 230 i la mitjana del sou dels professors passa a ser de 140 (=560/4).
DELETE	professors	NA	la mateixa que abans	DELETE FROM professors WHERE id = 1; El professor 3 té un sou de 110 i la mitjana del sou dels professors passa a ser de 105 (=110/2).
UPDATE	professors	categoria i sou	la mateixa que abans	UPDATE professors SET sou = 200 WHERE id = 3; El professor 3 passa a tenir un sou de 200 i la mitjana del sou dels professors passa a ser de 140 (=420/3).

2.2.a) No cobreix totes les sentències. Cal afegir disparadors per:

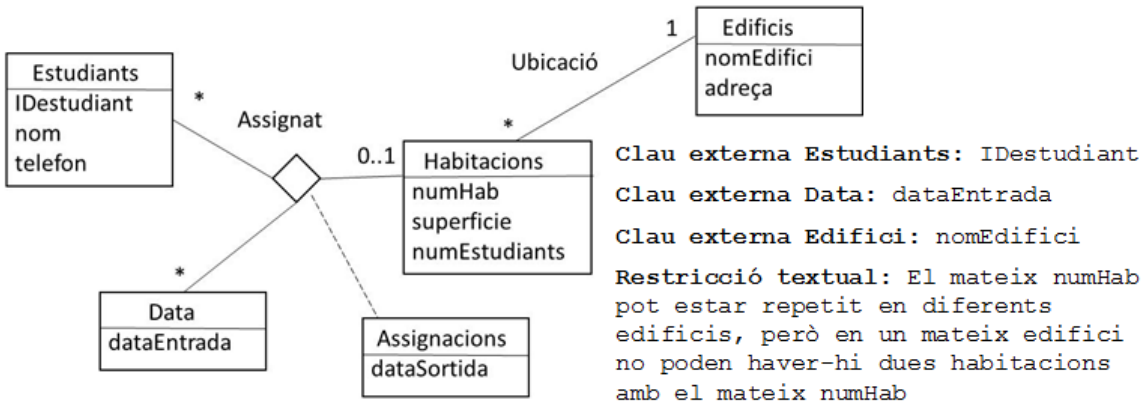
Esdeveniment	Taula	Atributs
UPDATE	coordinadors	dni, id_mat
DELETE	especialistes	NA
UPDATE	especialistes	dni, id_mat

2.2.b) Si. Es podria implementar modificant el CREATE TABLE de la taula coordinadors:

Afegint:
FOREIGN KEY (dni, id_mat) REFERENCES especialitzacions

Es possible eliminar la FOREIGN KEY següent per redundant:
REFERENCES professors

3. **(2 punts)** Considerant el disseny conceptual en UML següent, d'assignació d'habitacions en un campus universitari:



3.1 Doneu disseny lògic que s'obté de traduir el UML anterior a model relacional (cal incloure: taules, atributs, restriccions de clau primària, i restriccions de clau forana).

3.2 Suposant les taules resultants de l'apartat 3.1:

a) Doneu un exemple d'una vista que sigui actualitzable i definida com a WITH CHECK OPTION.

b) Doneu una sentència d'actualització de la vista anterior que falli degut al WITH CHECK OPTION. Expliqueu breument perquè falla.

3.3 Suposeu que el propietari de les taules resultants de l'apartat 3.1 és en Pere, Suposeu també la seqüència de sentències següent relativa a autoritzacions sobre aquestes taules. Cada sentència indica el nom de l'usuari que l'executa.

Pere: GRANT select(numHab, superfície, numEstudiants) ON habitacions TO Joan WITH GRANT OPTION;

Pere: GRANT update(superfície, numEstudiants) ON habitacions TO Ana WITH GRANT OPTION;

Pere: GRANT select(numHab, superfície, numEstudiants) ON habitacions TO Pol WITH GRANT OPTION;

Pol: GRANT select(numHab, superfície, numEstudiants) ON habitacions TO Laia;

Ana: GRANT update(superfície, numEstudiants) ON habitacions TO Laia;

Joan: GRANT select(numHab, superfície, numEstudiants) ON habitacions TO Laia;

Un cop fet això en Pere decideix fer dos REVOKES:

Pere: REVOKE select(numHab, superfície, numEstudiants) ON habitacions FROM Pol RESTRICT;

Pere: REVOKE select(numHab, superfície, numEstudiants) ON habitacions FROM Joan RESTRICT;

Es demana:

a) Dibuixar el diagrama d'autoritzacions després d'haver executat totes les sentències GRANT.

b) Dibuixar el diagrama d'autoritzacions després d'executar els dos REVOKE, i explica breument l'efecte de cadascun d'ells.

c) Justificar si la Laia podrà o no executar la sentència següent després de l'execució de totes les sentències GRANT i REVOKE.

UPDATE habitacions SET numEstudiants=numEstudiants+1 WHERE numHab=12;

SOLUCIONS:

3.1. Estudiants (IDestudiant, nom, telèfon)
Data (dataEntrada)
Edifici (nomEdifici, adreça)
Habitacions(nomEdifici, numHab, superfície, numEstudiants)
{nomEdifici} references Edifici
Assignacions(IDestudiant, dataEntrada, nomEdifici, numHab, dataSortida)
{IDestudiant} references Estudiants
{dataEntrada} references Data
{nomEdifici, numHab} references Habitacions
{nomEdifici, numHab} NOT NULL /* no penalitza el no posar-ho */

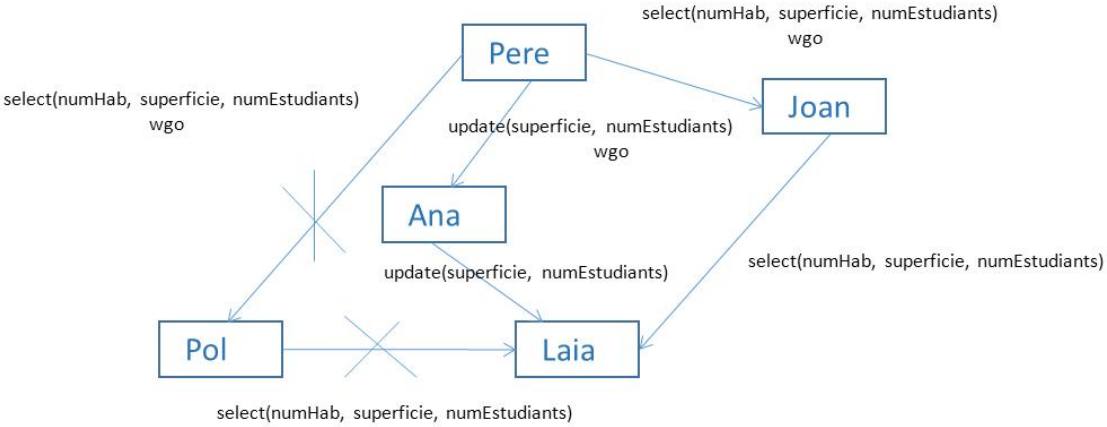
També és una solució vàlida si no s'inclou la relació Data (veure transpa 306)

3.2 Una pot ser:

```
CREATE VIEW vistaEstudiant1
AS SELECT * FROM estudiants WHERE IDestudiant=1
WITH CHECK OPTION;

INSERT INTO vistaEstudiant1 VALUES (2,'Joan',933234455);
```

Com que l'estudiant 2 no compleix la condició de la vista, no es podrà inserir.

3.3.a. i 3.3.b. En el diagrama següent es pot veure el graf després dels grants. L'efecte del primer revoke és que s'eliminen les autoritzacions taxades. El segon revoke no té cap efecte, ja que no es pot executar perquè la Laia perdria el privilegi de select que en aquest moment no li arriba per cap altra banda.



3.3.c. Si podrà executar la sentència perquè té privilegis per llegir i modificar el numEstudiants, i per llegir el numHab.

4. (2 punts) Considereu una BD amb files A,...,L, emmagatzemades en 4 pàgines tal i com es mostra a continuació:

Pàgina 1: A,B,C
Pàgina 2: D,E,F
Pàgina 3: G,H,I
Pàgina 4: J,K,L

Suposarem que només es poden adquirir reserves amb modalitat exclusiva (X), és a dir que tant per R com per RU/W cal una reserva amb modalitat X. Definirem un "comboi " com un punt en el temps en el qual una transacció T té una reserva sobre un objecte O, i com a mínim existeixen dues transaccions esperant reservar aquest mateix objecte O.

4.1 Suposant que les reserves s'adquireixen a nivell de files i que tenim les tres transaccions següents:

T1: Lock(E,X), RU(E), Lock(H,X), R(H), W(E),Unlock(E), Unlock(H)
T2: Lock(A,X), RU(A), Lock(E,X), R(E), W(A), Unlock(A), Unlock(E)
T3: Lock(G,X), R(G), Lock(D,X), RU(D), W(D), Unlock(D), Unlock(G)

existeix algun horari a on tinguem un comboi? Si la resposta es afirmativa, cal donar el graf d'espera associat. En cas negatiu, argumenteu la resposta.

4.2 Pel mateix cas que l'apartat 4.1, existeix algun horari a on tinguem una abraçada mortal? Si la resposta es afirmativa, cal donar el graf d'espera associat. En cas negatiu, argumenteu la resposta.

4.3 Suposeu ara que les reserves s'adquireixen a nivell de pàgines, i que per tant les transaccions queden definides de la manera següent,

T1: Lock(P2,X), RU(E), Lock(P3,X), R(H), W(E),Unlock(P2), Unlock(P3)
T2: Lock(P1,X), RU(A), Lock(P2,X), R(E), W(A), Unlock(P1), Unlock(P2)
T3: Lock(P3,X), R(G), Lock(P2,X), RU(D), W(D), Unlock(P2), Unlock(P3)

existeix algun horari a on tinguem un comboi? Si la resposta es afirmativa, cal donar el graf d'espera associat. En cas negatiu, argumenteu la resposta.

4.4 Pel mateix cas que l'apartat 4.3, existeix algun horari a on tinguem una abraçada mortal? Si la resposta es afirmativa, cal donar el graf d'espera associat. En cas negatiu, argumenteu la resposta.

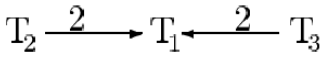
4.5 Usant les transaccions T1 i T2 de l'apartat 4.1, mostreu si és possible un horari on es vegi que el protocol de reserva en dues fases no és suficient per assegurar l'absència d'interferències. En cas de que no sigui possible trobar un horari com el que es demana, justifiqueu la resposta.

SOLUCIONS

4.1) No hi pot haver cap comboi ja que cap fila és reservada per més de dues transaccions

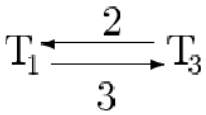
4.2) Cap abraçada mortal pot ocórrer. Només una tupla E és reservada per dues transaccions i hi ha d'haver com a mínim dues tuples reservades per dues o més transaccions.

4.3)



També són correctes els altres dos casos simètrics, on la transacció del mig és T2, o T3.

4.4)



4.5)

T1	T2
Lock(E,X)	
R(E)	
	Lock(A,X)
	R(A)
Lock(H,X)	
R(H)	
W(E)	
Unlock(E)	
Unlock(H)	
	Lock(E,X)
	R(E)
Abort	
	W(A)
	Unlock(A)
	Unlock(E)
	Commit

5. (2 punts) Suposa una base de dades on hi ha la taula següent que té 1.000.000 files, distribuïdes uniformement entre 20.000 pàgines segons un espai virtual de taula:

clients(idClient, nom, descripcio, venedor, saldo, classificacio, sucursal)

Els clients tenen identificadors que van del 1 al 1.000.000, i pertanyen a 500 sucursals. Cada sucursal té una mitjana de 2.000 clients, i una mitjana de 50 venedors. Considereu que sobre la taula hi ha definits els índexs:

- índex no agrupat per l'atribut *idClient*. Ordre d=300, ple al 90%, 2 nivells.
- índex agrupat pels atributs *sucursal*, *venedor*. Ordre d=80, ple al 75%, 3 nivells.

5.1 Indiqueu el cost en número de pàgines d'índex i de dades de resoldre les consultes següents:

a) select * from clients where idClient > 800.000 and idClient <=805.000;

b) select * from clients where classificacio = 'A';

c) select distinct(venedor) from clients where sucursal = 500;

5.2 Suposem ara que la taula *clients* està en un espai virtual d'agrupació junt amb les files d'una altra taula *sucursals(sucursal, adreça, ciutat, telefon, director).*

a) Dibuixeu quina seria l'estructura de una pàgina de dades en aquest cas.

b) Doneu una sentència SQL que es vegi afavorida per l'organització de les dues taules segons l'espai virtual d'agrupació.

c) Expliqueu com quedarien afectats els costos de les tres consultes de l'apartat 5.1 a per aquest canvi d'organització de les pàgines de dades.

SOLUCIÓ

Nombre de files per pàgina del fitxer de dades 1.000.000/20.000 = 50.
Índex per idClient
540 valors per node (300*2*90/100)
Índex per sucursal i venedor
120 valors per node (80*2*75/100)

5.1.a) S'usarà el primer índex

- cost(5.1.a) = 2 + ⌈5.000/540⌉ -1 + 5.000 = 11 pàgines índex (2 + 10 -1) + 5.000 pàgines dades = 5011 pàgines.

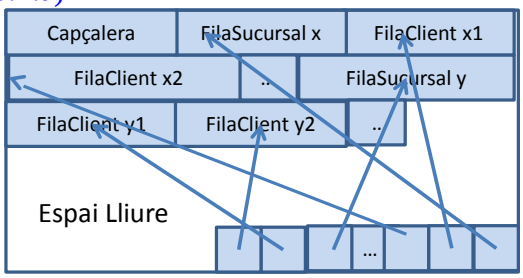
5.1.b) No hi ha cap índex que afavoreixi la consulta, cal accés seqüencial a les dades

- cost(5.1.b) = 20.000 pàgines dades.

5.1.c) S'usarà el tercer índex

- No cal accedir a dades perquè els venedors estan com a valors de l'índex.
  - cost(5.1.c) = 3 + ⌈2000/120⌉ -1 = 19 pàgines d'índex.
- Es considerava parcialment bé si s'anava a les dades, en aquest cas el cost seria:
  - cost(5.1.c) = 3 + ⌈2000/50⌉ = 3 pàgines d'índex + 40 pàgines de dades.

5.2.a)



5.2.b) select * from sucursals natural inner join clients

5.2.c) Només queden afectades les pàgines de dades, però els índexs són exactament iguals.

cost(5.2.c.1) = cost(5.1.a)	No queda afectat el cost perquè a les pàgines de dades s'hi va amb accés directe des dels RIDs que estan en l'índex. I l'índex no es veu afectat de cap manera.
cost(5.2.c.2) > cost(5.1.b)	Com que la consulta cal resoldre-la amb un accés seqüencial del fitxer de dades, el cost serà més elevat ja que el número de pàgines que ocuparan les files de clients serà més gran ja que entre mig hi haurà les files de les 50 sucursals. No es pot saber el cost exacte.
cost(5.2.c.3) = cost(5.1.c)	No queda afectat el cost perquè no cal accedir a les pàgines de dades per resoldre la consulta, i l'estructura de l'índex no es veu afectada. En el cas d'estudiants que consideraven que s'accedia a les dades el cost tampoc canvia, ja que tots els clients als que s'accedeix són de la mateixa sucursal.

BD **16.6.2017**

Temps: 3 hores

Notes 28 juny **Revisió:** 29 juny matí

1. (2 punts) Suposeu una base de dades amb les taules següents:

```
CREATE TABLE professors (
       dni integer PRIMARY KEY,
       nom_prof char(50),
       categoria char(10),
       sou integer);

CREATE TABLE materies(
       id_mat integer PRIMARY KEY,
       nom_mat char(50) UNIQUE);

CREATE TABLE assignatures (
       id_mat integer REFERENCES materies,
       codi_assig integer,
       nom_assig char(50),
       PRIMARY KEY (id_mat, codi_assig));
       // Feu atenció que una assignatura s'identifica per la matèria de què
       tracta i un codi d'assignatura.

CREATE TABLE coordinadors (
       id_mat integer,
       codi_assig integer,
       curs char(20),
       dni integer REFERENCES professors,
       PRIMARY KEY (id_mat, codi_assig, curs),
       FOREIGN KEY (id_mat, codi_assig) REFERENCES assignatures);
       // Taula que guarda els coordinadors d'assignatura de cada curs.

CREATE TABLE especialistes (
       dni integer REFERENCES professors,
       id_mat integer REFERENCES materies,
       PRIMARY KEY (dni, id_mat));
       // Taula que guarda els especialistes de cada matèria.
```

1.1 Doneu una sentència SQL per obtenir els identificadors de les assignatures per les que no hi ha hagut cap coordinador de l'assignatura que no sigui especialista de la matèria a la qual pertany l'assignatura.

```
     select a.id_mat, a.nom_assig
     from assignatures a
     where not exists (select *
                       from coordinadors c
                       where c.id_mat = a.id_mat and
                             c.codi_assig = a.codi_assig and
                             c.dni not in (select e.dni
                                           from especialistes e
                                           where e.id_mat = a.id_mat))
o bé
     select a.id_mat, a.nom_assig
     from assignatures a natural inner join coordinadors c
     group by a.id_mat, a.nom_assig
     having count(distinct c.dni) =
          (select count(distinct dni)
          from especialistes e, coordinadors c1
          where e.id_mat = c1.id_mat and e.id_mat = a.id_mat)
```

1.2 Indiqueu quina és la cardinalitat de la relació resultant de l'execució de les consultes següents. Considereu que les taules de la base de dades tenen P professors, M matèries, A assignatures. Tingueu en compte també que tota matèria té una o més assignatures, i que totes les assignatures tenen coordinador. Raona breument la resposta. Si no pots donar una única xifra, proporciona una fita inferior i una superior de la cardinalitat.

a) select count(*) from coordinadors natural inner join assignatures

group by id_mat

b) select * from matèries m

where not exists (select * from assignatures a where a.id_mat = m.id_mat)

c) select * from professors, assignatures

d) A = assignatures * coordinadors

B = A[dni]

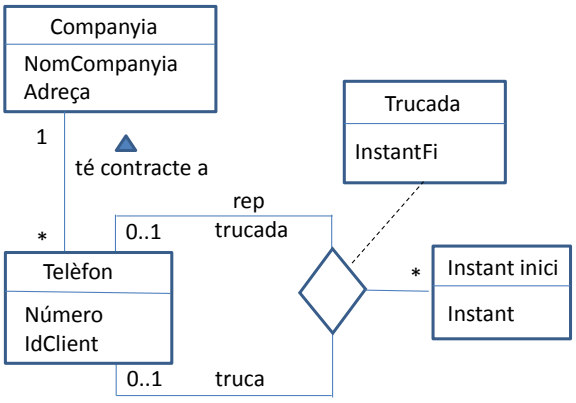
a. M (un grup per matèria)
b. 0 (tota matèria té una o més assignatures)
c. P x A (és un producte cartesià)
d. No sabem quants professors diferents són coordinadors. Fita superior P (tots els professors són coordinadors) i fita inferior 1 (un professor fa de coordinador de totes les assignatures).

1.3 Doneu una seqüència d'operacions de l'àlgebra relacional per obtenir el dni dels coordinadors que no són especialistes i que coordinen assignatures de matèries que no tenen especialistes.

```
A = coordinadors[dni]
B = especialistes[dni]
C = A – B
D = materies[id_mat]
E = especialistes[id_mat]
F = D – E
G = F * coordinadors
H = G[dni]
I = C * H
```

2. (2 punts)

2.1 Considerant el disseny conceptual en UML següent, doneu el disseny lògic que s'obté de traduir el UML anterior a model relacional (cal incloure: taules, atributs, restriccions de clau primària, restriccions de clau forana, i altres restriccions que es puguin derivar del model).



Clau externa Companyia: NomCompanyia
Clau externa Telèfon: Número
Clau externa InstantInici: instant

Companyies (nomCompanyia, adreça)
Telefons (número, idClient, nomCompanyia)
{nomCompanyia} references Companyies NOT NULL
Trucades (númeroTruca, instantInici, numTrucat, instantFi)
{númeroTruca} references Telefons
{númeroTrucat} references Telefons NOT NULL
{númeroTrucat,instantInici} UNIQUE

Es considera també correcta una solució que inclogui la taula instantInici amb una FK des de la taula Trucades.
Es considera correcta una solució que no tingui NOT NULL a númeroTrucat.

2.2 Considereu la base de dades següent:

departaments(idDept, nomDept, idEmplCap)

{idEmplCap} referencia Empleats

empleats(idEmp, nomEmpl, idDept)

{idDept} referencia departaments

Suposant que hi ha tres departaments amb identificador X, Y i Z, i que els identificadors dels seus caps són E1, E2 i E3 respectivament. Doneu les sentències SQL necessàries per tal que cada cap de departament pugui accedir per consultar únicament els empleats del seu departament.

```
create view empleatsX as
select * from empleats where idDept = 'X';
grant select on empleatsX to 'E1';

create view empleatsY as
select * from empleats where idDept = 'Y';
grant select on empleatsY to 'E2';

create view empleatsZ as
select * from empleats where idDept = 'Z';
grant select on empleatsZ to 'E3';
```

es considera també correcta la solució:

```
create view empleatsDept as
select e.idEmpl,nomEmpl from empleats e, departaments d
where e.idDept = d.idDept and d.idEmplCap = CURRENT_USER;
grant select on empleatsDept to 'E1', 'E2', 'E3';
```

2.3 L'arquitectura ANSI/SPARC d'un sistema de gestió de base de dades té tres nivells: nivell intern, nivell conceptual i nivell extern. Contesteu les preguntes següents:

a) Poseu exemples de tres components que podria tenir la base de dades de l'apartat 2.2 i que podrien formar part respectivament de l'esquema intern, l'esquema conceptual i l'esquema extern d'aquesta base de dades.

b) Explica què s'entén per independència física de dades en el marc de l'arquitectura ANSI/SPARC. Explica-ho usant els components que et facin falta dels que has posat com a exemple en l'apartat 2.3.a.

c) Explica què s'entén per independència lògica de dades en el marc de l'arquitectura ANSI/SPARC. Explica-ho usant els components que et facin falta dels que has posat com a exemple en l'apartat 2.3.a.

2.3.a esquema intern – un índex, p.ex. sobre la taula empleats per idEmpl
esquema conceptual – una taula, p.ex. la taula Empleats
esquema extern – una vista, p.ex. una de les vistes de l'apartat 2.2

2.3.b Canvis en l'esquema intern no afecten a l'esquema conceptual ni extern
Si eliminem el índex per idEmpl, i en posem un per nomEmpl, no canvien de cap manera l'esquema conceptual ni l'extern

2.3.c Canvis en l'esquema conceptual no afecten a l'esquema extern que no fa referència a la part de l'esquema conceptual modificada.
Si afegim un atribut a la taula departaments, les vistes de l'apartat 2.2 de l'esquema extern no es veuen afectades de cap manera
Canvis en l'esquema extern no afecten als altres esquemes externs ni esquema conceptual ni esquema intern
Si afegim o eliminem vistes no queden afectades les altres vistes que no depenen de les vistes definides, tampoc no queden afectades les taules (esq. conceptual), ni índexs (esq. intern)

3. (2 punts) Considereu l'esquema de la base de dades següent:

```
CREATE TABLE ProductoresTV (
   id_prod int primary key,
   descripcio char(20),
   pressupost_prod int not null check (pressupost_prod > 0),
   vendes real);

CREATE TABLE Equips (
   id_equip int primary key,
   descripcio char(20),
   pressupost_eq int not null check (pressupost_eq >= 0),
   id_prod int not null references ProductoresTV);

CREATE or REPLACE FUNCTION spPrimer() RETURNS trigger AS $$
BEGIN
   IF (new.vendes > old.vendes) THEN
      UPDATE Equips
      SET pressupost_eq = pressupost_eq + 0.001*new.vendes
      WHERE id_prod = old.id_prod;
   END IF;
   RETURN null;
END;
$$LANGUAGE plpgsql;

CREATE TRIGGER primer
AFTER UPDATE OF vendes ON ProductoresTV
FOR EACH ROW EXECUTE PROCEDURE spPrimer();

CREATE or REPLACE FUNCTION spSegon() RETURNS trigger AS $$
BEGIN
   UPDATE ProductoresTV
   SET pressupost_prod = pressupost_prod + new.pressupost_eq
   WHERE id_prod = old.id_prod;
   RETURN null;
END;
$$LANGUAGE plpgsql;

CREATE TRIGGER segon
AFTER UPDATE OF pressupost_eq ON Equips
FOR EACH ROW EXECUTE PROCEDURE spSegon();
```

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

```
INSERT INTO ProductoresTV VALUES(1, 'ImageTV', 20000, 500000);
INSERT INTO ProductoresTV VALUES(2, 'KRTV', 15000, 250000);
INSERT INTO Equips VALUES (11, 'So', 10000, 1);
INSERT INTO Equips VALUES (12, 'Camares', 10000, 1);
INSERT INTO Equips VALUES (13, 'Marqueting', 20000, 1);
INSERT INTO Equips VALUES (14, 'Documentacio', 5000, 1);
INSERT INTO Equips VALUES (21, 'Localitzacions', 10000, 2);
INSERT INTO Equips VALUES (22, 'Reporters', 10000, 2);
```

3.1 Digueu quin és el contingut final de la base de dades, després de l'execució de la sentència SQL següent i justifiqueu breument la resposta, explicant les accions que segueix el SGBD a conseqüència d'aquest update:

```
UPDATE ProductoresTV SET vendes = 560000 WHERE id_prod = 1;
```

3.2 Quina seria la diferència respecte l'apartat 3.1 suposant que els tipus dels triggers canvien de FOR EACH ROW a FOR EACH STATEMENT?

3.3 Definiu una asserció en SQL Standard per garantir el compliment de la restricció següent: "Tota productora que tingui un pressupost més gran o igual que 20000 ha de tenir al menys 3 equips que treballen per ella"

3.1) El contingut final de la taula ProductoresTV és:

1 'ImageTV' 67240 560000
2 'KRTV' 15000 250000

El contingut de la taula Equips és:

11 'So' 10560 1
12 'Camares' 10560 1
13 'Marqueting' 20560 1
14 'Documentacio' 5560 1
21 'Localitzacions' 10000 2
22 'Reporters' 10000 2

Un cop fet l'update de la productora 1, el trigger primer (after) executa el procediment spPrimer. Si la nova quantitat de vendes es superior a la antiga (com és el cas) llavors actualitza el pressupost del equips associats a la productora 1, és a dir, aquells que tenen el id_prod igual a 1, que són els equips 11, 12, 13 i 14. Donat que 0.001*560000 és igual a 560, els pressupostos d'aquests equips s'incrementa en 560.

Aquestes modificacións disparen el trigger segon (after) i s'executa el procediment spSegon per cada fila modificada Les dos execucions del procediment fan que s'augmenti en un total de 47240 (10560+10560+20560+5560) el pressupost de la productora 1 que passa a ser doncs 67240.

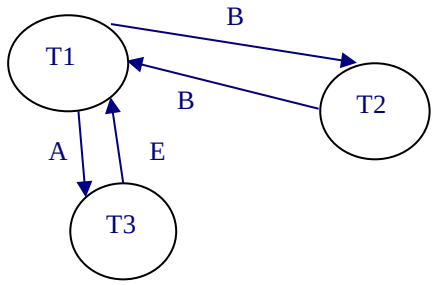
3.2) Els triggers i procediments es crearan però donaran error d'execució perquè les variables NEW i OLD no tenen valor per triggers FOR EACH STATEMENT.

3.3)
```
CREATE ASSERTION assercio CHECK (
        NOT EXISTS (SELECT * FROM ProductoresTV p
                WHERE p.pressupost_prod >= 20000 AND
                        (SELECT count(*) FROM equips e
                        WHERE e.id_prod = p.id_prod) < 3))
```

4. (2 punts) Donat un SGBD sense cap mecanisme de control de concurrència, suposem que es produeix l'horari següent:

Temps	T1	T2	T3
10		RU(B)	
20		W(B)	
30			
40	RU(A)		
50	W(A)		
60	R(D)		
70	R(B)		
80			R(E)
100			RU(A)
120			W(A)
			RU(F)
140		RU(B)	
150		W(B)	
160			W(F)
170	RU(E)		
180	W(E)		
190		COMMIT	
200			COMMIT
210	COMMIT		

4.1 Dibuixeu el graf de precedències associat a l'horari.



4.2 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). Quins horaris serials hi són equivalents? Justifica la resposta.

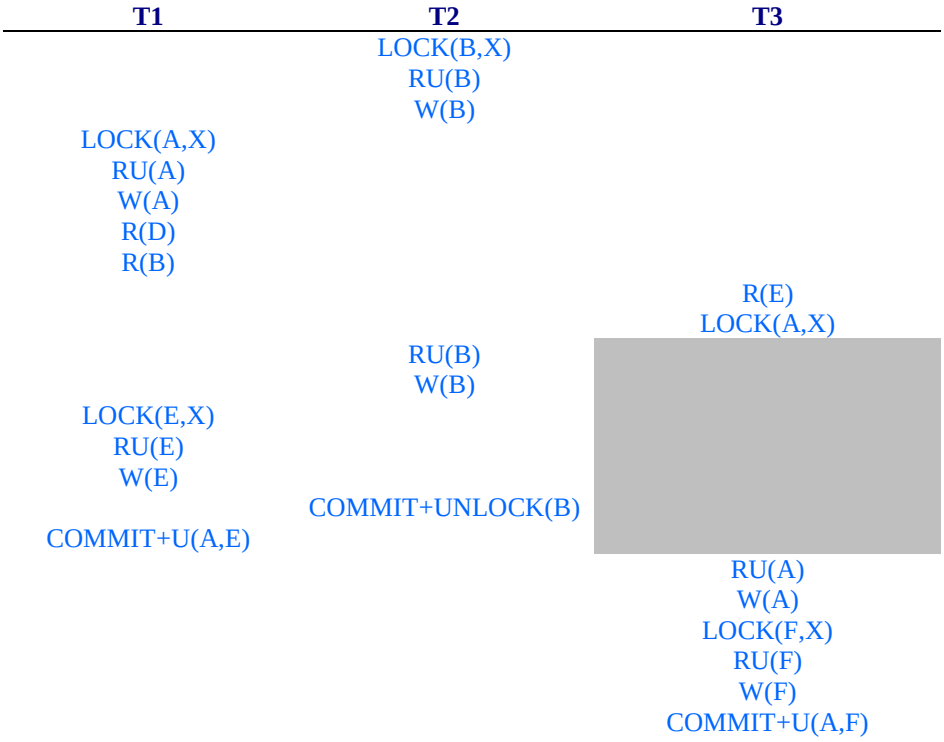
Es produeix una lectura no confirmada entre T1 i T2 sobre el grànul B.
Un anàlisi inconsistent entre T1 i T3 sobre els grànuls A i E.

No hi ha cap horari serial equivalent perquè hi ha interferències.

4.3 Suposem que s'incorporen tècniques de reserva. Les transaccions volen treballar amb SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED.

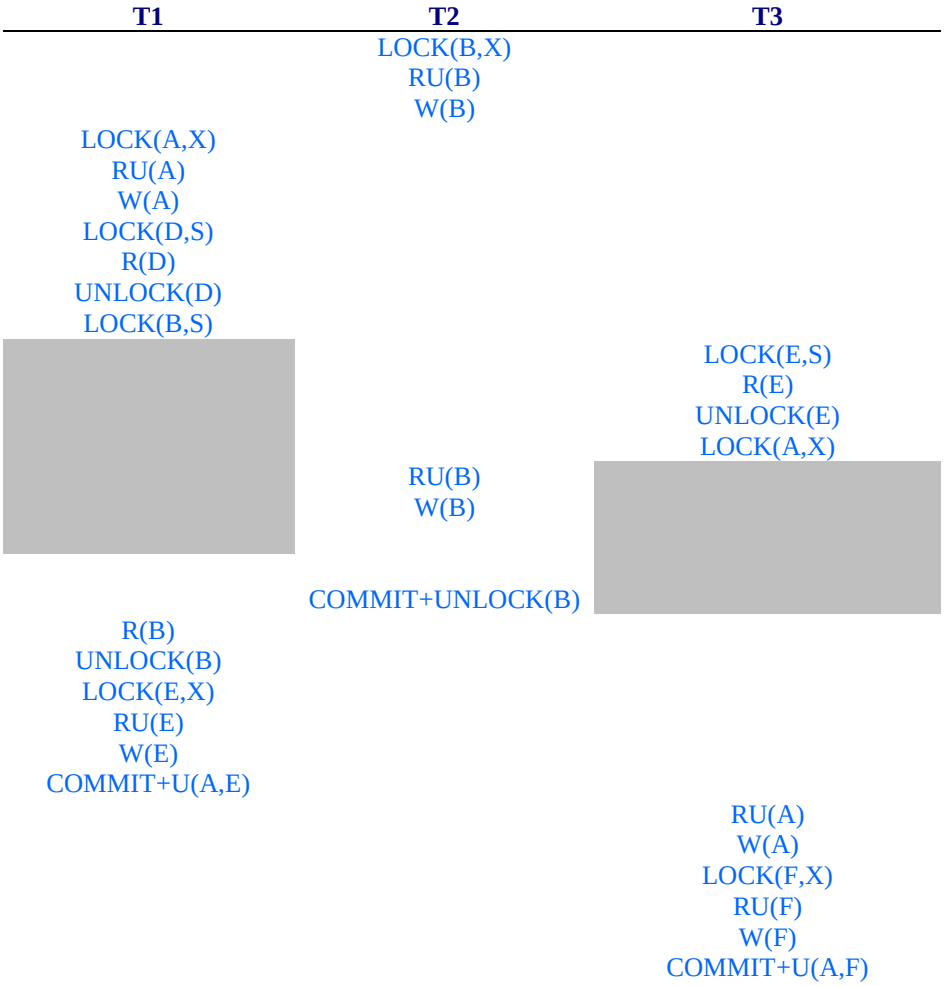
a) Determinar i executar la seqüència d'operacions read i write que es generen, el seu ordre i les peticions de lock i unlock necessàries mostrant les esperes que s'hi produeixen. En cas que cap acció de l'horari pugui executar-se en un moment donat, no continueu i justifiqueu el motiu.

b) Si ha alguna interferència en l'horari resultat de l'apartat 4.3.a digues quina/es és/són i entre quines transaccions es produeix/en. Sinó n'hi ha cap justifica la resposta.



Les dues interferències se segueixen produint. De fet, el nivell d'aïllament no les evita.

4.4 Responeu les mateixes preguntes de l'apartat 4.3 en cas de que les transaccions estan definides com SET TRANSACTION ISOLATION LEVEL READ COMMITTED.



La lectura no confirmada s'ha evitat, com era d'esperar pel nivell d'aïllament. En canvi, l'altra encara queda ja que aquest nivell no l'impedeix

5. Suposa una base de dades on hi ha una taula T que té 300.000 files. Hi ha un índex no agrupat sobre aquesta taula definit sobre l'atribut ATR que és clau primària i és de tipus char i ocupa 6 bytes. Els apuntadors a les pàgines de l'índex ocupen 4 bytes i els RID ocupen també 4 bytes. Les pàgines són de 1004 bytes i i el factor de bloqueig de les pàgines de dades és 25.

5.1 Determineu la d òptima d'aquest índex

SOLUCIÓ:

Nodes interns (2d +1)*4 + 2d 6 = 1004 => d = 50
Fulles (2d) (6+4) + 8 = 1004 => d= 49,8 => d = 49

D òptima 49

5.2 Considereu ara que la d òptima és 50 i que l'arbre està ple al 69%. Es vol que la consulta SELECT * FROM T where ATR = constant tingui un cost de 2 accessos a pàgines de disc, independentment del valor concret de la constant. Tenint en compte que es pot tenir més d'un nivell de l'arbre a memòria, és això possible? Si la resposta és afirmativa, determineu el nombre pàgines de l'índex que caldria tenir a memòria per tal de que això fos possible. En cas de no ho sigui, justifiqueu la resposta.

SOLUCIÓ:

El cost en aquest cas està determinat per la *fòrmula h + 1*.
Calculem h: $\log_{70} 300000 = 3$.
Per tant, el cost serà de 4 accessos a disc. Per tal de que el cost sigui sempre de 2, caldria que els dos nivells superior estiguessin a memòria.

Calculem el nombre pàgines de l'arrel i nivell pare de les fulles:
Nivell fulla : 300000 / 69 = 4347,82 4348 nodes
Nivell pare de les fulles: 4348/ 70 = 62,11 63 nodes
Arrel: 1 node

Per tant, necessitaríem 64 pàgines de l'índex a memòria

5.3 Suposeu ara que l'índex està definit sobre un atribut que no és clau primària. L'atribut té en mitjana 5 valors repetits. Responeu a la mateixa pregunta que a l'apartat 5.2.

SOLUCIÓ:

En aquest cas el cost està definit com (h + F -1) + |T(a=cte)|
Donat que |T(a=cte)| = 5, seria impossible que el nombre d'accessos fos 2, ja que encara que tinguéssim tots els nivells de l'arbre a memòria, requeria com a mínim 5(cas pitjor) accessos per localitzar els repetits.

BD **17.1.2018**

Temps: 3 hores

Notes **31 gener matí** **Revisió:** **1 febrer tarda**

Cada pregunta s'ha de lliurar en un full separat, excepte la pregunta 1, que s'ha de lliurar els apartats 1.1 i 1.2 en un full, i els 1.3 i 1.4 en un altre

1. (2.5 punts) Suposeu una base de dades amb les taules següents:

```
CREATE TABLE persones
   (nif CHAR(9),
    nom CHAR(30) NOT NULL,
    ciutatRes CHAR(20),
    PRIMARY KEY (nif));
-- ciutatRes és on viu la persona

CREATE TABLE tipus
   (tipusId CHAR(20),
    numMaxInscrits INTEGER,
    PRIMARY KEY (tipusId));

CREATE TABLE activitats
   (tipusId CHAR(20),
    instIni INTEGER,
    descripcio CHAR(30),
    nifP CHAR(9) NOT NULL,
    preu INTEGER,
    ciutat CHAR(20),
    PRIMARY KEY (tipusId, instIni),
    FOREIGN KEY (tipusId) REFERENCES tipus(tipusId),
    FOREIGN KEY (nifP) REFERENCES persones(nif));
    -- el nifP és el de la persona que és professor de l'activitat
    -- ciutat és on es fa l'activitat

CREATE TABLE inscripcions
   (tipusId CHAR(20),
    instIni INTEGER,
    nifI CHAR(9), -- persona inscrita
    instPagament INTEGER,
    PRIMARY KEY (tipusId, instIni, nifI),
    FOREIGN KEY (tipusId, instIni)
            REFERENCES activitats(tipusId,instIni),
    FOREIGN KEY (nifI) REFERENCES persones);
    -- el nifI és el de la persona que s'inscriu a l'activitat
```

Amb les sentències de càrrega de les taules:

```
INSERT INTO persones VALUES ('111', 'Anna', 'Barcelona'), ('222',
'Alicia', 'Sabadell'), ('333', 'Rosa', 'Gava');

INSERT INTO tipus VALUES ('body_pump', 15),('zumba', 20),('aerobic',25);

INSERT INTO activitats VALUES ('body_pump', 212, null, '111', 25,
'Barcelona'), ('zumba', 318, null, '222', 25, 'Sabadell'),('aerobic',
118, null, '222', 20, 'Sabadell');

INSERT INTO inscripcions VALUES ('zumba', 318, '333', 10), ('aerobic',
118, '333', 10);
```

1.1 Per cadascuna de les situacions que es plantegen a continuació indica si és o no possible, tenint en compte les restriccions definides a la base de dades. En cas que no sigui possible indica la restricció que ho evita, en cas que sigui possible dóna els inserts per tal que passi.

a) És possible que hi hagi dues activitats del mateix tipus que comencen al mateix instant impartides per professors diferents?

No, tipusId i instIni formen la clau primària de la taula activitats i, per tant, no és possible que hagi dues tuples amb la mateixa clau.

b) És possible que una persona s'inscrigui en una activitat de la qual ell mateix és professor?

Res ho evita, només cal afegir en els inserts anteriors el següent:

```
INSERT INTO inscripcions VALUES ('aerobic', 118, '222', 10);
```

c) És possible que una persona s'inscrigui en dues activitats del mateix tipus?

Res ho evita, ja que la clau primària és tipusId, instantIni, nifI, es pot repetir el tipus i el nif, i no l'instant. Només cal afegir en els inserts anteriors el següent:

```
INSERT INTO activitats VALUES
('aerobic', 312, null, '111', 25, 'Barcelona'),
INSERT INTO inscripcions VALUES ('aerobic', 312, '333', 10);
```

1.2 Donada la vista:

```
CREATE VIEW nose(identI, nomI, identP, nomP, quantitat) AS
SELECT pI.nif, pI.nom, pP.nif, pP.nom, COUNT(*)
FROM persones PI, persones pP,
     inscripcions i NATURAL INNER JOIN activitats a
WHERE pI.nif = i.nifI AND a.nifP = pP.nif
GROUP BY pI.nif, pP.nif
```

a) Explica breument quines dades s'obtenen quan es fa un select de la vista.

Retorna la quantitat d'inscripcions d'una persona a activitats que fa un professor. Concretament dóna el nif i nom de la persona inscrita, el nif i nom del professor, i la quantitat d'inscripcions de la persona inscrita a activitats del professor.

b) Quin és el contingut de la vista després d'executar els inserts que us donem en l'enunciat?

identi	nomi	identp	nomp	quantitat
333	Rosa	222	Alicia	2

c) És actualitzable? Per què?

No. Motius diversos: múltiples taules, un group by, funció d'agregació.

1.3 Doneu una sentència SQL per obtenir el de les persones que són professors d'alguna activitat en la que no hi ha cap inscrit que visqui en la ciutat en la que es fa l'activitat.

```
SELECT DISTINCT pr.nom
FROM persones pr, activitats a
WHERE  pr.nif = a.nifP and
        NOT EXISTS (SELECT *
                    FROM inscripcions i, persones p
                    WHERE a.tipusId = i.tipusId AND
                          a.instIni = i.instIni AND
                          i.nifI = p.nif AND
                          p.ciutatRes = a.ciutat)
```

1.4 Doneu un conjunt de sentències d'àlgebra relacional que obtenir el nif de les persones que són professors de dues o més activitats que es fan en una mateixa ciutat.

```
A = activitats[tipusId, instantIni, nifP, ciutat]
B = A{tipusId -> tipusId2, instantIni -> instantIni2, nifP -> nifP2, ciutat -> ciutat2}
C = A[nifP = nifP2, ciutat = ciutat2]B
D = C(tipusId <> tipusId2 OR instantIni <> instantIni2)
R = D[nifP]
```

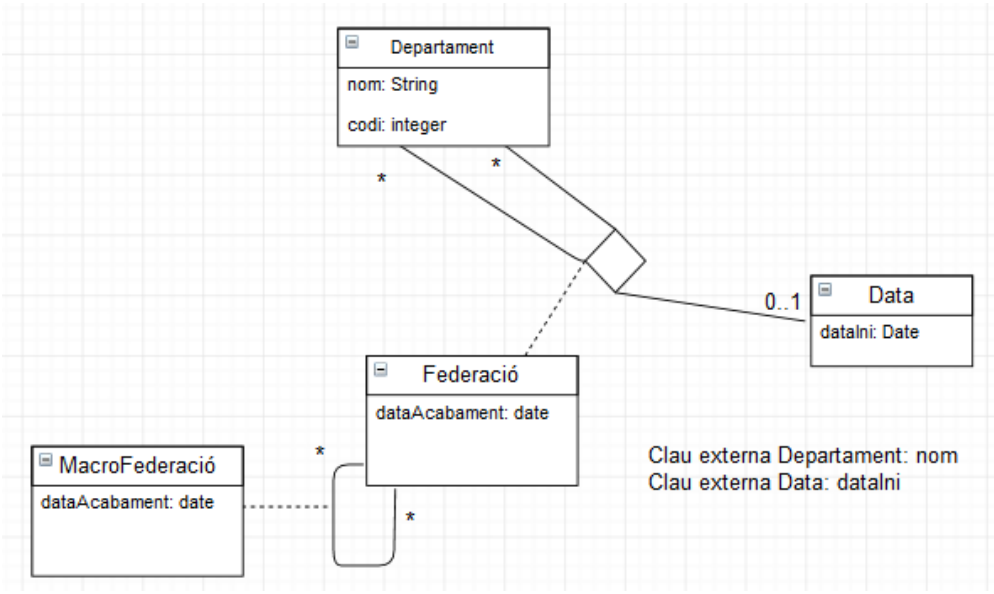
o bé

A = activitats[tipusId, instantIni, nifP, ciutat]
B = A{tipusId -> tipusId2, instantIni -> instantIni2, nifP -> nifP2, ciutat -> ciutat2}
C = A x B
D = C(nifP = nifP2 AND ciutat = ciutat2 AND (tipusId <> tipusId2 OR instantIni <> instantIni2))
R = D[nifP]

o bé

A = activitats[tipusId, instantIni, nifP, ciutat]
B = A{tipusId -> tipusId2, instantIni -> instantIni2, nifP -> nifP2, ciutat -> ciutat2}
C = A [nifP = nifP2 AND ciutat = ciutat2 AND tipusId <> tipusId2] B
D = A [nifP = nifP2 AND ciutat = ciutat2 AND instantIni <> instantIni2] B
E = C u D
R = D[nifP]

2. (2.5 punts) Considerant el disseny conceptual en UML següent:

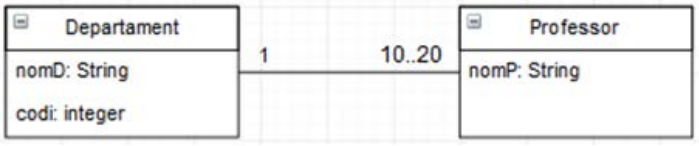


2.1 Doneu disseny lògic que s'obté de traduir el UML anterior a model relacional (cal incloure: taules, atributs, restriccions de clau primària, i restriccions de clau forana).

Departament(nom,codi)
Federació(nomd1,nomd2,dataIni,dataAcabament)
nomd1 references Departament(nom),
nomd2 references Departament(nom)
MacroFederació(nomd1,nomd2,nomd3,nomd4,dataAcabament)
nomd1,nomd2 references Federació(nomd1,nomd2),
nomd3,nomd4 references Federació(nomd1,nomd2),

Tal com s'indica a les transpes les taules corresponents a classes data, hora, instant, no cal posar-les.

2.2 Considereu ara el següent disseny conceptual:



que ha donat lloc a les taules:

```
Departament(nomD, codi)
Professor(nomP, nomD)
    nomD és clau forana que apunta a Departament
    nomD és NOT NULL
```

a) Seria possible expressar en PostgreSQL la cardinalitat 10..20 fent servir alguna restricció de columna o de taula sobre les taules anteriorment definides? En cas de que sigui possible, doneu la sentència. En cas contrari, justifiqueu la resposta.

No seria possible, ja que no es poden tenir subconsultes en els checks

b) Seria possible expressar en SQL estàndard la cardinalitat 10..20 fent servir una asserció. En cas de sigui possible, doneu la sentència. En cas contrari, justifiqueu la resposta.

```
CREATE ASSERTION cardinalitat CHECK(
            NOT EXISTS (SELECT *
                        FROM Professor P
                        GROUP BY P. NOMD
                         HAVING COUNT(*)>20 OR COUNT(*)<10)
          AND NOT EXISTS (SELECT *
                          FROM Departament D
                          WHERE NOT EXISTS (select * from professors p
                                            Where p.nomd=d.nomd)))
```

O bé

```
CREATE ASSERTION cardinalitat CHECK(
            NOT EXISTS (SELECT *
                        FROM Departament D
                          WHERE (SELECT COUNT(*)
                                 FROM professor
                                 WHERE p.nomd=d.nomd) > 20
                              OR ((SELECT COUNT(*)
                                 FROM professor
                                 WHERE p.nomd=d.nomd) < 10)
```

c) Doneu els triggers necessaris sobre Professor per tal de assegurar el compliment de les cardinalitats. Podeu fer servir la plantilla de triggers de PostgreSQL:

```
CREATE TRIGGER nom [BEFORE/AFTER] [UPDATE (of column)/DELETE/INSERT] ON taula
    [FOR EACH ROW/FOR EACH STATEMENT] execute procedure nomp();
CREATE FUNCTION nomp() RETURNS trigger AS $$
    BEGIN
    END; $$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER insercio BEFORE INSERT ON professor
FOR EACH ROW execute procedure insercio();
CREATE or replace FUNCTION insercio() RETURNS trigger AS $$
BEGIN
   IF (select count(*) from professor where nomd=new.nomd) =20 THEN
     RAISE EXCEPTION 'error ins';
   END IF;
   RETURN NEW;
END; $$ LANGUAGE plpgsql;

CREATE TRIGGER supressió BEFORE DELETE ON professor
FOR EACH ROW execute procedure supressió();
CREATE or replace FUNCTION supressió() RETURNS trigger AS $$
BEGIN
   IF (select count(*) from professor where nomd=old.nomd) =10 THEN
     RAISE EXCEPTION 'error del'; END IF;
   RETURN OLD;
END; $$ LANGUAGE plpgsql;

CREATE TRIGGER modificació BEFORE UPDATE of nomD ON professor
FOR EACH ROW execute procedure modificació();
CREATE or replace FUNCTION modificació() RETURNS trigger AS $$
BEGIN
 IF (new.nomd<>old.nomd) THEN
      IF (select count(*) from professor where nomd=new.nomd) =20 THEN
```

```
            RAISE EXCEPTION 'error mod'; END IF;
        IF (select count(*) from professor where nomd=old.nomd) =10 THEN
           RAISE EXCEPTION 'error mod'; END IF;
     END IF;
    RETURN NEW;
    END; $$ LANGUAGE plpgsql;
```

3. **(2.5 punts)** Considera la base de dades i la transacció següent:

Esquema	*Extensió*
oficines(numOficina, adreça) comptes(numOficina, numCompte, saldo) {numOficina} referencia oficines	Oficines 12 Barcelona 24 Vic comptes 12 450 60 24 670 200

T1
UPDATE comptes SET saldo = saldo + 50 WHERE numOficina = 12 AND numCompte = 450;
UPDATE comptes SET saldo = saldo - 50 WHERE numOficina = 24 AND numCompte = 670;
commit;

3.1 Interferències de lectura no repetible

a) Escriu un horari en que participi la transacció T1 i on hi hagi únicament una interferència de lectura no repetible. En el nou horari s'ha d'afegir una transacció T2 que només pot utilitzar sentències `select`.

T1	*T2*
UPDATE comptes SET saldo = saldo + 50 WHERE numOficina = 12 AND numCompte = 450;	
	SELECT * FROM comptes WHERE numOficina = 24 AND numCompte = 670;
UPDATE comptes SET saldo = saldo - 50 WHERE numOficina = 24 AND numCompte = 670;	
	SELECT * FROM comptes WHERE numOficina = 24 AND numCompte = 670;
commit;	
	commit;

És una solució correcta qualsevol T2 on hi hagi dos selects que cadascun consulti una única fila (o bé 12, 450 o bé 24, 670), i que aquesta fila sigui la mateixa. Els dos selects han d'estar sempre abans i després de l'update del mateix gränul que consulten. Els selects han de consultar com a mínim l'atribut saldo del compte.

b) Indica quin és el nivell d'aïllament mínim que evita aquest tipus d'interferències.

Repeatable Read

3.2 Interferències d'anàlisi inconsistent

a) Escriu un horari en que participi la transacció T1 i on hi hagi únicament una interferència d'anàlisi inconsistent. En el nou horari s'ha d'afegir una transacció T2 que només pot utilitzar sentències `select`.

T1	*T2*
UPDATE comptes SET saldo = saldo + 50 WHERE numOficina = 12 AND numCompte = 450;	

	`SELECT * FROM comptes WHERE numOficina = 12 AND numCompte = 450;`
	`SELECT * FROM comptes WHERE numOficina = 24 AND numCompte = 670;`
`UPDATE comptes SET saldo = saldo - 50 WHERE numOficina = 24 AND numCompte = 670;`	
`commit;`	
	`commit;`

És una solució correcta qualsevol T2 on hi hagi dos selects que consultin cadascun una fila de la taula (o bé 12, 450 o bé 24, 670). Els dos selects poden estar tots dos entremig dels dos updates (en qualsevol ordre), o poden estar un abans i un després dels updates (en qualsevol ordre).

És també una solució correcta una T2 on hi hagi un únic select que consulti les dues files de la taula, o totes les files de la taula (que són les dues files). En aquest cas el select ha d'estar necessàriament entremig dels dos updates.

Els selects han de consultar com a mínim l'atribut saldo del compte.

b) Indica quin és el nivell d'aïllament mínim que evita aquest tipus d'interferències.

Repeatable Read

3.3 Interferències de fantasmes

a) Escriu un horari en que participi la transacció T1 i on hi hagi únicament una interferència de fantasmes. En el nou horari s'ha d'afegir una transacció T2 que només pot utilitzar sentències `select`.

T1	***T2***
`UPDATE comptes SET saldo = saldo + 50 WHERE numOficina = 12 AND numCompte = 450;`	
	`SELECT * FROM comptes WHERE saldo<200;`
`UPDATE comptes SET saldo = saldo - 50 WHERE numOficina = 24 AND numCompte = 670;`	
	`SELECT count(*) FROM comptes WHERE saldo<200;`
`commit;`	
	`commit;`

En cas de posar una T2 amb dos selects que consultin els comptes amb un saldo superior a 70 al voltant del primer update, a part de un fantasma hi hauria un anàlisi inconsistent.

En el cas de posar una T2 amb dos selects un abans dels dos updates i l'altre després a part d'un fantasma es pot produir una interferència de lectura no repetible.

b) Tradueix l'horari a operacions R, RU i W tenint en compte que el gràNul és la fila.

Suposant que la fila 12, 450 és el gràNul a, i que la fila 24, 670 és el gràNul b.

	T1	T2
40	RU(a)	
50	W(a)	
60	RU(IC)	
70	W(IC)	
80		R(IC)
90		R(a)
110	RU(b)	
120	W(b)	

130	RU(IC)	
140	W(IC)	
150		R(IC)
160		R(a)
179		R(b)
190	c	
200		c

c) Indica quin és el nivell d'aïllament mínim que evita aquest tipus d'interferències.

SERIALIZABLE

d) Dona l'horari de l'apartat b amb les reserves i les esperes incorporades segons el nivell d'aïllament indicat.

	T1	T2
40	L(a,X)	
50	RU(a)	
60	W(a)	
70	L(IC,X)	
80	RU(IC)	
90	W(IC)	
100		L(IC,S)
120	L(b,X)	
140	RU(b)	
150	W(b)	
160	RU(IC)	
170	W(IC)	
180	c+U(a,b,IC)	
190		R(IC)
200		L(a,S)
210		R(a)
220		L(b,S)
230		R(b)
240		c+U(a,b,IC)

4. (2.5 punts) Considerant la base de dades de l'exercici anterior. Suposa que a la taula oficines hi ha 3.000 oficines, i que a la taula comptes hi ha 15.000.000 comptes.

4.1 Indica si com a resultat de cadascuna de les consultes anteriors es poden obtenir o no resultats repetits. En cas que sí, dóna un contingut de les taules que faci que s'obtinguin resultats repetits. En cas que no, explica perquè no pot passar.

a) SELECT COUNT(*) FROM comptes WHERE saldo>200 GROUP BY numOficina;

b) SELECT numCompte FROM comptes WHERE numOficina=555;

c) SELECT numOficina FROM oficines NATURAL INNER JOIN comptes;

4.2 Suposa que sobre la taula comptes hi ha únicament definit un índex agrupat (arbre B+) multi-atribut definit pels atributs numOficina, numCompte. La d de l'arbre és 70, i està ple al 80%. Suposa que cada oficina té una mitjana de 5.000 comptes, i que el factor de bloqueig de les pàgines de dades és 100. Explica quina és la manera òptima de resoldre cadascuna de les consultes següents. Calcula els cost de cada consulta, indicant el que significa cada factor que intervé en el càlcul.

a) SELECT * FROM comptes ORDER BY numOficina, numCompte;

b) SELECT * FROM comptes WHERE numOficina=350 AND numCompte = 18;

c) SELECT numOficina, numCompte FROM comptes WHERE numOficina=480;

4.3 Suposa que tenim un índex no agrupat (arbre B+) definit sobre l'atribut saldo que té 50000 nodes fulla, i que l'índex està ple als 70%.

a) Indica quina és la d de l'arbre. Explica com l'has obtingut.

b) Explica on surten els RIDs en un índex arbre B+. Indica quants RIDs hi haurà en total en els nodes de l'índex indicat. Explica com has fet els càlculs.

4.1.a Hi haurà resultats repetits si dos oficines tenen un mateix número de comptes amb saldo superior a 200.

oficines (1, 'Barcelona), (2, 'Vic)

comptes (1, 1, 250), (1, 2, 350), (2, 1, 450), (2, 2, 500)

en aquest cas el resultat seria (2), (2)

4.1.b No hi pot haver repetits perquè surt una fila per cada compte de l'oficina 555, i en l'oficina 555 no hi pot haver números de compte repetits, tenint en compte la clau primària de la taula comptes

4.1.c Hi haurà repetits en cas que hi hagi una oficina amb més d'un compte.

oficines (1, 'Barcelona), (2, 'Vic)

comptes (1, 1, 250), (1, 2, 350), (2, 1, 450), (2, 2, 500)

en aquest cas el resultat seria (1), (1), (2), (2)

4.2.a Com que es tracta d'un índex agrupat en que les dades estan ordenades com l'índex, i l'accés seqüencial per valor és en el mateix ordre de l'ordenació n'hi haurà prou en accedir al fitxer de dades per resoldre la consulta, així doncs els cost serà número de files de la taula, dividides pel factor de bloqueig de les pàgines de dades.

cost = 15000000 files/100 files/pàgina = 150000 pàgines

4.2.b En aquest cas la consulta és un accés directe per valor que només retorna una única fila, ja que s'accedeix per la clau primària, així doncs els cost serà el cost de baixar per l'arbre més un accés més a la pàgina de dades que conté la fila.

valors per node 70 x 2 x 0.8 = 112

alçada de l'arbre = $\log_{113}15000000 = 3.5 \Rightarrow 4$

cost = 4 + 1

4.2.c En aquest cas la consulta és un accés seqüencial per valor a tots els comptes d'una oficina. Cal tenir en compte que només es mostra els números dels comptes. Cal també tenir en compte que aquests números estan al propi índex, no cal accedir a les dades per buscar aquests números. Així doncs els cost serà el cost de baixar per l'arbre més un accés a les pàgines fulles de l'índex que contenen números de compte de l'oficina.

alçada de l'arbre 4

número de nodes fulla de l'índex que conté números de compte d'oficines

5000 comptes/oficina / 112 valors/node = 45 nodes = 45 pàgines

cal restar 1 perquè la primera fulla es compta dues vegades

cost = 4 + 45 -1

4.3.a 15000000 valors en els nodes fulla

50000 nodes fulla

15000000 valors / 50000 nodes fulla = 300 valors per node

nodes plens al 70%

2 x d x 0.7 = 300, d= 300/1.4, d= 214

4.3.b RIDs <u>només</u> n'hi han als nodes fulla d'un arbre B+, i n'hi ha tants com valors hi ha als nodes fulla. Com que als nodes fulla hi ha d'haver 15000000 valors, hi haurà 15000000 de RIDs.

BD **28.6.2018**

Temps: 3 hores

Notes 4 juliol Revisió: 5 juliol matí

Cada pregunta s'ha de lliurar en un full separat, excepte la pregunta 1, que s'ha de lliurar els apartats 1.1 i 1.2 en un full, i els 1.3 i 1.4 en un altre

1. (2.5 punts) Suposeu una base de dades amb les taules següents:

```
create table departaments
(codiDept integer,
nomDept char(50) unique,
telefonDept char(15),
primary key (codiDept));

create table professors
(dni char(50),
nomProf char(50) unique,
telefonProf char(15),
sou integer not null check(sou>0),
dniCoord char(50),
codiDept integer not null,
primary key (dni),
foreign key (dnicoord) references professors,
foreign key (codiDept) references departaments);
-- dniCoord correspon al dni del coordinador del professor
-- codiDept correspon al departament al que pertany el professor

create table despatxos
(modul char(5),
numero char(5),
superficie integer not null check(superficie>12),
codiDept integer,
primary key (modul,numero),
foreign key (codiDept) references departaments);
-- codiDept correspon al departament al que pertany el despatx

create table assignacions
(dni char(50),
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);
-- instantFi te valor null quan una assignacio es encara vigent.
```

1.1 Doneu una sentència SQL per obtenir els departaments tals que totes les assignacions als seus despatxos han sigut assignacions de professors del departament. En la consulta han d'aparèixer també els departaments pels que no hi ha hagut cap assignació a despatxos del departament.

```
select dpt.codiDept
from departaments dpt
where not exists (select *
                  from assignacions a, despatxos d, professors p
                  where a.modul = d.modul and
                        a.numero = d.numero and
                        d.codiDept = dpt.codiDept and
                        a.dni = p.dni and
                        p.codidept != dpt.codiDept)
```

1.2 Escriviu les sentències SQL per tal que l'usuari X (el propietari de les 4 taules) permeti que es puguin fer les operacions següents sobre les taules anteriors:

a) L'usuari A pot modificar tots els atributs de la taula assignacions i de la taula professors, i també modificar el departament dels despatxos. Pot passar aquests privilegis a altres usuaris.

X: GRANT UPDATE ON assignacions to A WITH GRANT OPTION ;
GRANT UPDATE ON professors to A WITH GRANT OPTION ;
GRANT UPDATE(codiDept) ON despatxos to A WITH GRANT OPTION;

b) L'usuari A permet que l'usuari C pugui modificar el departament dels despatxos.

A: GRANT UPDATE(codiDept) ON despatxos to C;

c) L'usuari X permet que l'usuari C modifiqui la taula despatxos. L'usuari C pot passar aquests privilegis a altres usuaris.

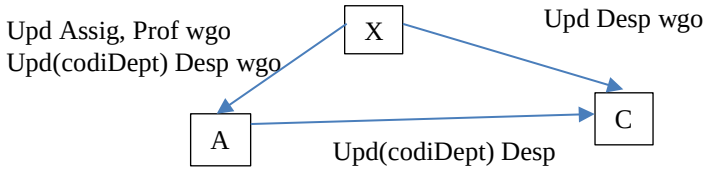
X: GRANT UPDATE ON despatxos to C WITH GRANT OPTION;

d) L'usuari X desautoritza l'usuari A per modificar l'atribut codiDept de la taula despatxos en mode RESTRICT.

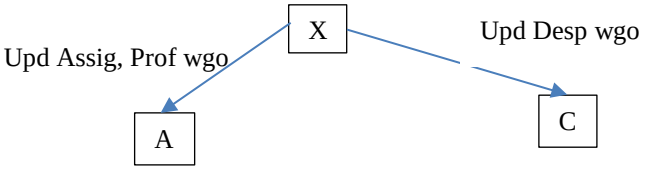
X: REVOKE UPDATE(codiDept) ON despatxos TO A RESTRICT.

e) Dibuixa el diagrama d'autoritzacions després d'executar les sentències de a), b), c) i abans de d) i el diagrama d'autoritzacions resultant després de la sentència d).

Abans del REVOKE



Després del REVOKE



1.3 Doneu una seqüencia d'operacions d'àlgebra relacional per obtenir el codi i nom dels departaments que han tingut o tenen professors assignats a despatxos d'un altre departament.

```
A = assignacions * professors
B = despatxos{modul -> mod, numero -> num, codiDept → cd}
C = A[modul = mod, numero = num, codiDept <> cd]B
D = C[codiDept]
E = D*departaments
F = E[codiDept, nomDept]
```

1.4 Suposant que la quantitat de departaments és DP, la quantitat de professors és P, la quantitat de despatxos és DX, i la quantitat d'assignacions és A, digueu i justifiqueu quin serà l'esquema i la cardinalitat mínima i màxima de la relació R que s'obté de cadascuna de les seqüències d'operacions d'àlgebra relacional següents:

a) R = departaments*professors

R(codiDept, nomDept, telefonDept, dni, nomProf, telefonProf, sou, dniCoord)

Card mínima = 0 Tots els professors tenen el codiDept a NULL o no n'hi ha cap

Card màxima = P Tots els professors tenen codiDept diferent de NULL

b) A = professors{codiDept → cd}

R = departaments[codiDept * cd]A

R(codiDept, nomDept, telefonDept, dni, nomProf, telefonProf, sou, dniCoord)

Card mínima = 0 Tots els professors tenen el codiDept a NULL o no n'hi ha cap

Card màxima = P Tots els professors tenen codiDept diferent de NULL

c) A = professors{codiDept → cd}

R = departaments[codiDept < cd]A

R(codiDept, nomDept, telefonDept, dni, nomProf, telefonProf, sou, cd)

Card mínima = 0 Tots els professors estan al mateix departament o no hi ha cap

Card màxima = (DP-1)*P Tots els professors estan assignats al departament més gran i ha altres departaments

2. (2.5 punts) Considereu la base de dades de l'exercici 1.

2.1 Doneu el disseny conceptual en UML tal que al traduir a model relacional donaria com a resultat l'esquema de la base de dades de l'exercici 1.

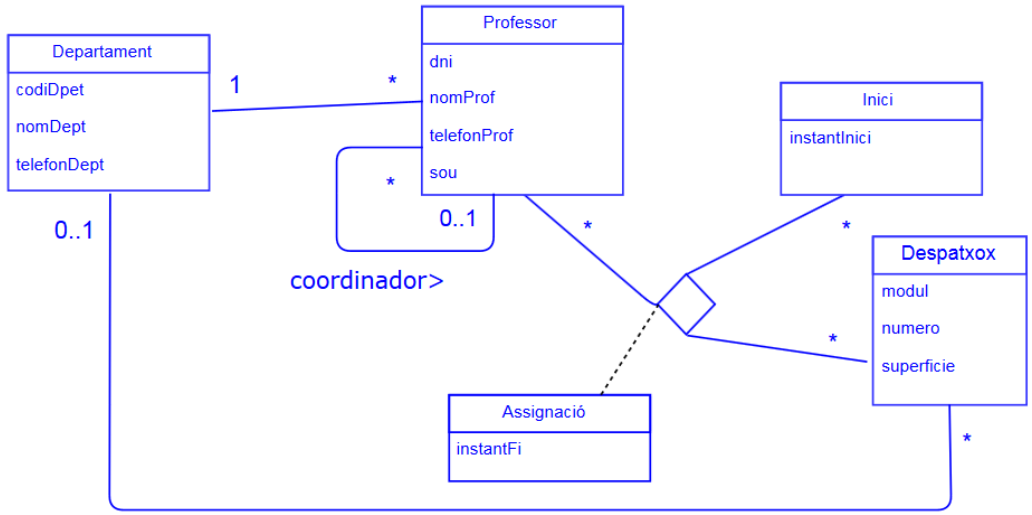
2.2 Definiu una asserció en SQL Standard per garantir el compliment de la restricció següent: "No existeix cap departament amb més de 5 professors que tinguin un coordinador que no és del departament".

2.3 Indiqueu els disparadors que caldria per implementar l'asserció definida a l'apartat 2.2 en PostgreSQL.

Per a cada disparador cal que indiqueu: l'esdeveniment activador, la taula, el tipus de disparador, i la justificació del tipus de disparador escollit. En cas que l'esdeveniment activador sigui un UPDATE cal també explicitar els atributs rellevants.

Tingueu en compte que les restriccions d'integritat definides a la BD (primary key, foreign key,...) es violen amb menys freqüència que la restricció comprovada per aquests disparadors.

2.1)



2.2)

```
CREATE ASSERTION assercioSol1 CHECK (not exists(select p.codiDept
            from professors p,professors p1
            where p.profCoord=p1.dni and
                  p.codiDept<>p1.codiDept
            group by p.codiDept
            having count(*)>5)
```

Es considera també correcta una solució on s'afegeixi la condició "p.dni.coord is not null". Encara que es tracta d'una condició innecessària, ja que en cas que no es compleixi aquesta condició mai es pot complir la condició "p.dnicoord=p1.dni".

La solució següent també funciona, encara que té una subconsulta innecessària.

```
CREATE ASSERTION assercioSol2 CHECK (not extists(select d.codiDept
            from departaments d
            where 5 < (select count(distinct p1.dni)
                        from professors p, professors p1
                        where p.codiDept = d.codiDept and
                              p.profCoord=p1.dni and
                              p.codiDept<>p1.codiDept))
```

2.3)

Professors	INSERT BEFORE FOR EACH ROW	UPDATE, CodiDept BEFORE FOR EACH ROW	UPDATE, dniCoord BEFORE FOR EACH ROW

Com que l'enunciat indica que la restricció es viola amb menys freqüencia que les restriccions d'integritat de la BD, el tipus de disparador ha de ser BEFORE.
L'esdeveniment DELETE on Professors no és rellevant:
- Considerant que en la base de dades quan s'esborra un professor s'apliqués una política de manteniment d'integritat referencial RESTRICCIÓ, no seria rellevant. El motiu és que aquest esdeveniment no podria arribar mai a violar la restricció, ja que només s'arribaria a fer l'esborrat en cas que el professor no fos cap de cap altre professor.
- Considerant que en la base de dades quan s'esborra un professor s'apliqués una política de manteniment d'integritat referencial CASCADA, tampoc seria rellevant. El motiu és que en esborrar el professor de les claus foranes dels professors que li eren coordinats, aquests professors no podrien passar a violar la restricció, ja que serien professors que no tindrien coordinador, i per tant a qui no afectaria la restricció.

3. (2.5 punts) Considereu les taules R(A,B) i S(C). Suposeu que les taules estan buides inicialment i que el grànul de concurrència és la tupla. Per a cadascun de les parelles de transaccions següents determineu tots els possibles resultats [X,Y], on X és el resultat del primer count de T1 i Y el resultat del segon.

Per a cada resultat és imprescindible justificar la resposta en base als nivells d'aïllament de les transaccions, i les potencials esperes degudes a locks. Justifiqueu també perquè no hi ha cap altre resultat possible a part dels que proposeu.

1.1 T1:

```
Set Transaction Isolation Level Read Committed;
Select count(*) From R;
Select count(*) From S;
Commit;
```

T2:

```
Set Transaction Isolation Level Serializable;
Insert Into R Values (1,2);
Insert Into S Values (3);
Commit;
```

[1,1] T1 s'executa després de T2. No hi ha esperes.
[0,0] T2 s'executa després de T1. No hi ha esperes.
[0,1] Tota la transacció T2 s'executa entre el primer i el segon count de T1. Això és possible ja que el mode read Committed allibera immediatament després de la lectura.
Cap altre resultat és possible ja que el nivell read committed farà que T1 esperi fins al commit de T2 abans de llegir les tuples insertades de T2.

1.2 T1:

```
Set Transaction Isolation Level Read Committed;
Select count(*) From R;
Select count(*) From S;
Commit;
```

T2:

```
Set Transaction Isolation Level Serializable;
Insert Into R Values (1,2);
Insert Into R Values (3,4);
Commit;
```

[0,0] Tot t2 s'executa després de t1
[2,0] Tot t1 s'executa després de t2
Cap altre resultat és possible ja que d'una banda el nivell read committed farà que T1 esperi fins al commit de T2 abans de llegir la tuple insertada de T2.
A més, T2 no afecta al segon count.

1.3 T1:

```
Set Transaction Isolation Level Repeatable Read;
Select count(*) From R;
Select count(*) From R;
Commit;
```

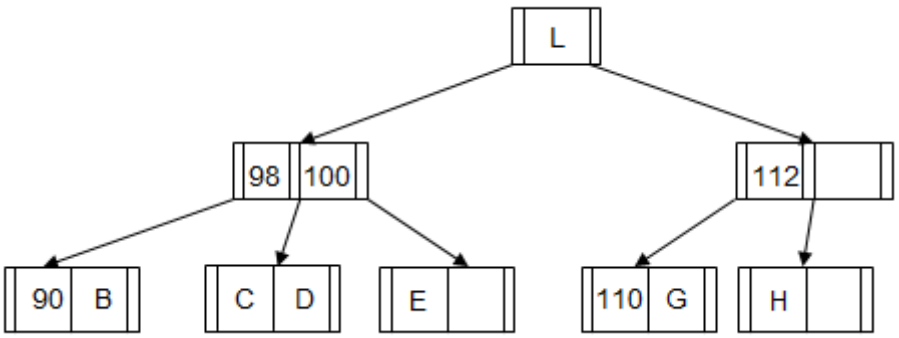
T2:

```
Set Transaction Isolation Level Serializable;
Insert Into R Values (1,2);
Commit;
```

[1,1] T1 s'executa després de T2.
[0,0] T2 s'executa després de T1.
[0,1] Tota la transacció T2 s'executa entre el primer i el segon count de T1. Això és possible ja que el mode repeteable read no reserva la informació de control, i per tant T2 no queda bloquejada. Es produeix un fantasma.

4. (2.5 punts)

4.1 Donat l'arbre B+ següent, d'ordre d=1, i definit per a l'atribut a de la taula T(a,b), indiqueu quins són els valors que poden prendre B, C, D, E, G, H, L.



Cal recordar que tal com s'ha explicat durant el curs:
- Al node arrel i nodes interns NO hi pot haver valors que NO estiguin als nodes fulla. És a dir tots els valors que estan a nodes arrel o interns han d'estar necessàriament a nodes fulla.
- L'apuntador de l'esquerra porta a valors més petits, l'apuntador de la dreta porta a valors superiors o iguals.

B pot no tenir valor, o un valor entre 91 i 97.
C ha de tenir valor 98
D pot no tenir valor, o valor 99.
E ha de tenir valor 100.
G pot no tenir valor, o valor 111.
H ha de tenir valor 112
L ha de tenir valor 110

4.2 Considereu la taula assignacions de la base de dades de la pregunta 1. Suposeu que la taula té 100.000 de files, i un factor de bloqueig de 10 files per pàgina. Suposeu també que: hi ha 18 assignacions del professor 123 a la taula assignacions, de les quals només 2 són assignacions vigents (instantFi null); hi ha 3000 assignacions al mòdul A en despatxos amb número superior a 100, de les quals només 450 són assignacions vigents.

Expliqueu com es resol i mostreu com es calcula el cost de la consulta següent en els casos indicats a sota la consulta.

```
select * from assignacions
where modul = 'A'
      and numero >= '100'
      and dni = '123'
      and instantFi is null
```

a) En cas de que no s'usi cap índex.

b) En cas que s'utilitzi un índex arbre B+ agrupat multiatribut definit per als atributs modul, numero amb 2 nivells, una d de 146 i una ocupació del 80%.

c) En cas que s'utilitzi un índex arbre B+ no agrupat definit per l'atribut dni amb 2 nivells, una d de 292 i una ocupació del 70%.

a) Si no hi ha índex, no hi ha més remei que accedir a totes les pàgines de dades.

(100.000 files):(10 files/pàg)= 10.000 pàg

b) Al ser agrupat el SGBD baixa per l'índex i després va directament a les pàgines de dades (que estaran ordenades igual que l'índex pel motiu de estar definit com a índex agrupat).

A través de l'índex l'únic que es pot trobar són les files del modul 'A' i numero és 100, però no se sap res de la resta de condicions, que caldrà comprovar quan es llegeixi les files en el fitxer de dades. Recordeu que en l'índex només hi ha els valors, que en aquest cas són parelles (modul, numero).

Hi ha 3000 files a la taula assignacions que compleixen les condicions modul = 'A' and numero >= '100'.

2 pàg (baixar per l'índex)
+ $\lceil$(3000 files):(10 files/pàg)$\rceil$ = 303 pàgines

c) Al ser NO agrupat el SGBD baixa per l'índex i després ha d'usar les fulles de l'índex per trobar les assignacions del mateix dni '123'. Per cada valor '123' en les fulles de l'índex el SGBD usa el RID corresponent per trobar la fila al fitxer de dades.

A través de l'índex l'únic que es pot trobar són les files del dni '123', però no se sap res de la resta de condicions, que caldrà comprovar quan es llegeixi les files en el fitxer de dades. Recordeu que en l'índex només hi ha els valors, que en aquest cas són dni.

Hi ha 3000 files a la taula assignacions que compleixen la condició associada als atributs modul, numero.

2 pàg (baixar per l'índex)
+ $\lceil$(18 valors):$\lceil$(292*2*0,7)$\rceil\rceil$ - 1
+ 18 pàgines = 20 pàgines

El -1 és per no comptar dues vegades el primer node fulla de l'arbre (al baixar per l'arbre i al comptar els nodes fulla).

1.5 Sabent que la taula professors de la base de dades de la pregunta 1 té 6 atributs. Indica quants índexs (arbres B+) agrupats i quants índexs no agrupats, sobre un únic atribut, poden existir de manera simultània per a aquesta taula. Justifica la resposta.

Una taula només pot tenir definit un índex agrupat. El motiu és que quan un índex és agrupat les pàgines de dades tenen les files ordenades com l'índex. Per tant, és impossible tenir les files ordenades a l'hora físicament de dues maneres diferents.

Hi poden haver, doncs, índex no agrupats definits per a la resta dels atributs, ja que pels índexs no agrupats no es hi ha un ordre físic establert de les files de la taula.

Temps: 3 hores

1. (2 punts) Suposeu una base de dades amb les taules següents:

```
CREATE TABLE EquipsFutbol(
   nomEquip char(30),
   localitat char(50),
   nomEstadi char(100) UNIQUE NOT NULL,
PRIMARY KEY(nomEquip));

CREATE TABLE Partits(
   nomEquipLocal char(30),
   dataPartit date,
   nomEquipVisitant char(30) NOT NULL,
   golsEquipL integer NOT NULL CHECK(golsEquipL >=0),
   golsEquipV integer NOT NULL CHECK(golsEquipV >=0),
PRIMARY KEY (nomEquipLocal,dataPartit),
FOREIGN KEY (nomEquipLocal) REFERENCES EquipsFutbol,
FOREIGN KEY (nomEquipVisitant) REFERENCES EquipsFutbol,
UNIQUE (nomEquipVisitant, dataPartit),
CHECK(nomEquipLocal <> nomEquipVisitant));

CREATE TABLE Jugadors(
   dniJugador char(9),
   nom char(50) NOT NULL,
   nomEquip char(30) NOT NULL,
PRIMARY KEY (dniJugador),
FOREIGN KEY (nomEquip) REFERENCES EquipsFutbol);

CREATE TABLE Alineacions(
   nomEquipLocal char(30),
   dataPartit date,
   dniJugador char(9),
   gols integer NOT NULL CHECK(gols>=0),
   numTargetes integer NOT NULL CHECK(numTargetes>=0),
PRIMARY KEY (nomEquipLocal,dataPartit,dniJugador),
FOREIGN KEY (nomEquipLocal,dataPartit) REFERENCES Partits,
FOREIGN KEY (dniJugador) REFERENCES Jugadors);
```

1.1 Escriviu una sentència SQL per obtenir els equips que no han perdut cap partit jugat a fora i que han guanyat tots els partits jugats a casa. Es diu que un equip juga un partit a casa quan és l'equip local del partit, i que un equip juga a fora quan és l'equip visitant del partit. Per cadascun dels equips que surtin en el resultat de la consulta es vol el nom de l'equip i la quantitat total de gols que ha fet en els partits jugats a casa.

```
SELECT p.nomEquipLocal, sum(golsEquipL)
FROM partits p
WHERE NOT EXISTS(SELECT *
                 FROM partits pV
                 WHERE p.nomEquipLocal = pV.nomEquipVisitant
                      AND pV.golsEquipV<=pV.golsEquipL)
      AND NOT EXISTS (SELECT *
                 FROM partits pL
                 WHERE p.nomEquipLocal = pL.nomEquipLocal
                      AND pL.golsEquipL<=pL.golsEquipV)
GROUP BY p.nomEquipLocal
```

1.2 Suposeu que volem una sentència SQL per trobar els jugadors (dniJugador, nom) que no han estat en cap alineació. Digueu, per cadascuna de les sentències següents **si dóna el resultat correcte o incorrecte**.

- En cas de que el resultat que doni sigui **correcte,** digueu si hi ha alguna part de la consulta que no passa els criteris de qualitat de l'assignatura, i quina és aquesta part.
- En cas que el resultat que doni sigui **incorrecte**, donar un contingut de les taules, el resultat que dóna la consulta tal com està per aquest contingut i el resultat que hauria de donar si fos correcte.

a)
```
SELECT j.dniJugador, j.nom
FROM jugadors j
WHERE j.dniJugador NOT IN (SELECT a.dniJugador
                  FROM alineacions a, jugadors ju
                  WHERE a.dniJugador=ju.dniJugador);
```

b)
```
SELECT DISTINCT j.dniJugador, j.nom
FROM jugadors j
WHERE NOT EXISTS (SELECT * FROM alineacions a
                   WHERE a.dniJugador=j.dniJugador);
```

c)
```
SELECT DISTINCT j.dniJugador, j.nom
FROM jugadors j, alineacions a
WHERE j.dniJugador = a.dniJugador and
      j.dniJugador NOT IN (SELECT al.dniJugador
                           FROM alineacions al);
```

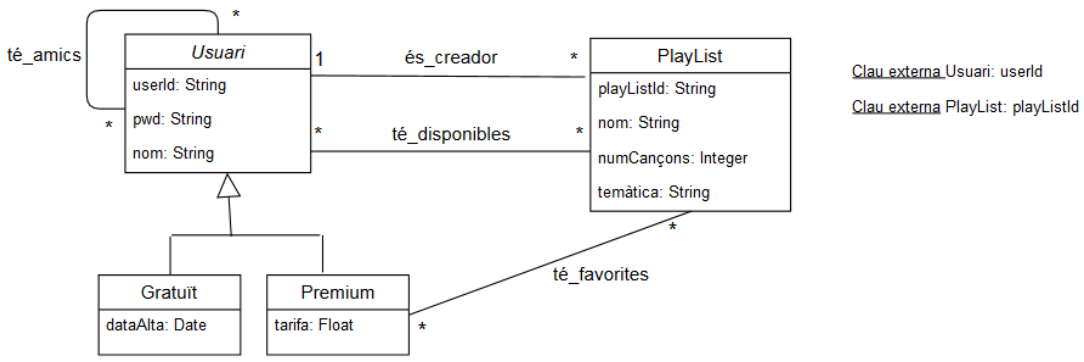
a) Correcte. Taula innecessària jugadors ju.
b) Correcte. DISTINCT innecessari.
c) Incorrecte. En una base de dades amb només una fila a la taula jugadors, la fila hauria de sortir en el resultat de la consulta, i en canvi la consulta tal com està no dóna cap resultat.

1.3 Escriviu una seqüència d'operacions d'àlgebra relacional per obtenir els dni dels jugadors que han tingut alguna alineació en un partit on també ha estat alineat el jugador amb dni '999'. Tingueu en compte que en el resultat de la consulta no volem que surti el jugador '999'.

B=alineacions(dniJugador='999')
C=B[dniJugador, nomEquipLocal,dataPartit]
D=C{dniJugador ->dniJ, nomEquipLocal ->nomE,dataPartit –>dataP}
E=alineacions[dniJugador, nomEquipLocal,dataPartit]
F=E[dniJugador<>dniJ,nomEquipLocal=nomE,dataPartit =dataP]D
R=F[dniJugador]

2. (2 punts)

2.1 Doneu la traducció a model relacional del diagrama de classes UML següent. Cal incloure: taules, atributs, claus primàries, claus foranes, i restriccions NOT NULL i UNIQUE derivades del diagrama.



2.2 Tenint en compte l'esquema de la base de dades que heu obtingut a l'apartat 2.1. Indiqueu si es pot implementar, en PostgreSQL, cadascuna de les restriccions textuals següents **com a restricció de columna o de taula.**

- En cas que es pugui, indica quina serà la restricció i en quina taula es definirà.
- En cas que no es pugui, explica perquè.

RI1 – Un usuari no es pot tenir com a amic a sí mateix.

RI2 – Un usuari només pot crear un màxim de 20 playlists.

RI3 – Les playlists favorites d'un usuari premium han de ser playlists que l'usuari tingui disponibles.

RI4 – No poden existir dues playlists amb el mateix nom i temàtica.

RI5 – El número de cançons d'una playlist ha de tenir valor i aquest valor ha de ser superior a 0.

2.3 Suposeu per aquest exercici una base de dades que té una taula amb els moviments realitzats amb les targetes de crèdit d'una entitat bancària. La taula `moviments` és propietat de l'usuari A. A continuació podeu veure l'esquema de la taula i un exemple del seu contingut.

moviments(idTargeta, instant, botiga, import)

'1111222244445555'	7777	'Zerka'	35
'2222333344445555'	8888	'Sous'	900
'2222333344445555'	9999	'Tnac'	330

a) Escriu les sentències SQL necessàries per tal que l'usuari B pugui consultar i modificar únicament els moviments que tenen un import inferior a 1000.

b) Un cop l'usuari B tingui l'accés que li heu donat a l'apartat anterior, justifiqueu si podrà o no executar la sentència SQL següent:

```
UPDATE moviments
SET import = import + 200
WHERE idTargeta = '2222333344445555' and instant = 8888;
```

2.1 usuari(userId, pwd, nom)

gratuït(userId,dataAlta)
{userId} referencia usuari

premium(userId,tarifa)
{userId} referencia usuari

playlist(playListId,nom,numCançons,tematica,userIdCreador)
{userIdCreador} referencia usuari NOT NULL

disponibles(userId,playListId)
{userId} referencia usuari
{playListId} referencia playList

favorites(userId,playListId)
{userId} referencia premium
{playListId} referencia playList

amics(userId, userIdAmic)
{userId} referencia usuari
{userIdAmic} referencia usuari

2.2 RI1. check (userId <> userIdAmic), taula amics.

RI2. En SQL Standard es podria posar com un check amb una expresió SQL que comptés el número de playlist creades. En PostgreSQL no es pot perquè és un check que afecta a vàries taules.

RI3. {userId, playListId} references disponibles, taula favorites

RI4. UNIQUE(nom, tematica), taula playlist

RI5. NOT NULL CHECK (numCançons>0), taula playlist

2.3

a)
A: create view movs1000 as select * from moviments
where import > 1000

A: grant select, update on movs1000 to B

b) No la pot executar perquè no té privilegis sobre la taula moviments, només sobre la vista.
Si fos la vista podria o no depenent de si la vista s'hagués definit com a "with check option".

3. (2 punts) Considereu una base de dades amb les taules següents:

```
departaments(idD, nomD, pressupost)
empleats(numE, nomE, sou, cap, idD)
     {cap} references empleats(numE)
     {idD} references departaments(idD)
edificis(nomEd, adreça)
assignacions(nomEd, idD, planta)
     {nomEd} references edificis(nomEd)
     {idD} references departaments(idD)
```

3.1 Suposeu que cadascuna de les `regles` que s'indiquen a continuació es vol implementar mitjançant triggers. Indiqueu i justifiqueu per cada regla quins triggers caldria definir, tenint en compte els criteris de qualitat establerts a l'assignatura. Per cada trigger cal indicar:

- esdeveniment que l'activa (cal indicar esdeveniment, taula i atributs rellevants)
- before/after
- row/statement

R1 – No hi pot haver cap empleat que tingui un sou superior al sou del seu cap. En cas que aquesta regla no es compleixi cal que salti una excepció.

R2 – El pressupost d'un departament s'ha de mantenir igual a la suma dels sous dels empleats del departament.

R3 – Només es pot afegir empleats si la suma dels pressupostos de tots els departaments és superior a 100.000. En cas que aquesta regla no es compleixi cal que salti una excepció.

3.2 A les taules `empleats` i `assignacions` es vol que quan s'esborri un departament s'apliquin les polítiques següents:

- taula empleats

```
FOREIGN KEY (idD) REFERENCES departaments(idD)
ON DELETE SET NULL
```

- taula assignacions

```
FOREIGN KEY (idD) REFERENCES departaments(idD)
ON DELETE CASCADE
```

Suposem que en comptes d'indicar aquestes polítiques al crear les taules, es vol implementar aquestes polítiques en el procediment emmagatzemat `tancaDepartament` que serveix per esborrar un departament que es passa com a paràmetre. A més, el procediment ha de llançar l'excepció "Departament no existeix" si el departament passat com a paràmetre no està a la base de dades. Completa la implementació del procediment seguint els criteris de qualitat establerts a l'assignatura.

```
CREATE FUNCTION tancaDepartament(prof departaments.idD%type)
RETURNS void AS $$
...
END;
$$LANGUAGE plpgsql;
```

3.1

	insert	update	delete
RI1	taula empleats before/row	atribut cap taula empleats before/row atribut sou taula empleats before/row	-
RI2	taula empleats after/row	atribut sou o idD taula empleats after/row atribut pressupost (*) taula departaments after/row	delete empleats after/row
RI3	insert empleats before/statement	-	-

(*) Cal tenir en compte que en cas de update de pressupost a la taula departaments s'hauria de definir com propagar el canvi a la taula empleats.

3.2

```
create function tancaDepartament(dept departaments.id%type)
returns void as $$
begin
     update empleats set idD = null where idD = dept;
     delete from assignacions where idD = dept;
     delete from departaments where idD = dept;
     if not found then
          raise exception '%', 'Departament no existeix'; end if;
     return;
end;
$$language plpgsql;
```

4. (2 punts)

4.1 Disposeu de 3 operacions: R(A), RU(A) i W(A).

a) Doneu un horari amb 2 transaccions que facin servir les operacions anteriors que necessiteu (una operació pot aparèixer més d'una vegada si és necessari), i on hi hagi una única interferència d'ACTUALITZACIÓ PERDUDA.

b) Doneu la resta d'horaris possibles que presentin la mateixa interferència i que facin servir el mínim nombre d'operacions donades (igualment, una operació pot aparèixer més d'una vegada si és necessari).

c) Justifica que l'horari que has proposat en l'apartat a) no té un horari serial equivalent.

4.2 Suposeu ara l'horari següent:

T1	T2	T3	T4
		OPX	
	R(A)		
			RU(B)
			RU(C)
RU(A)			
		R (F)	
	R(B)		
	R(C)		
W(A)			
R(A)			
		OPY	
			W(C)
			W(B)
			commit
commit			
	commit		
		commit	

a) Sense tenir en compte OPX i OPY, doneu el graf de precedències corresponent a l'horari donat. En cas que l'horari tingui interferències, digues quines són (nom de la interferència, i transaccions i grànuls implicats).

b) Tingueu en compte que OPX i OPY poden ser operacions simples (R, RU, W) o complexes (RU+W). Doneu tots els parells de valors de OPX i OPY que generin una interferència entre T1 i T3. Indiqueu quina interferència es produeix per a cada parell donat? Justifiqueu breument perquè es produeix cada interferència.

c) Doneu l'horari amb les reserves i les esperes incorporades segons el nivell d'aïllament REPETEABLE READ. Per a aquest horari suposeu que OPX = R(A) i OPY = R(A).

4.1.a

T1	T2
RU(A)	
	RU(A)
W(A)	
	W(A)
commit	
	commit

4.1.b

T1	T2
	RU(A)
RU(A)	
W(A)	
	W(A)
commit	
	commit

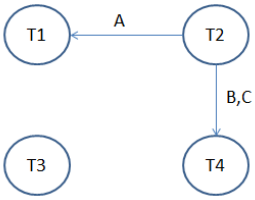
T1	T2
	RU(A)
RU(A)	
	W(A)
W(A)	
commit	
	commit

T1	T2
RU(A)	
	RU(A)
	W(A)
W(A)	
commit	
	commit

4.1.c

No hi ha cap horari serial equivalent perquè hi ha un cicle al graf de precedències que correspon a una interferència de actualització perduda.

4.2.a.



El graf de precedències no té cicles i per tant, podem assegurar que no hi ha cap interferència.

4.2.b.

- OPX = R(A), OPY = R(A), lectura no repetible
- OPX = R(A), OPY = RU(A)+W(A), lectura no repetible
- OPX = RU(A), OPY=W(A), actualització perduda
- OPX = RU(A)+W(A), OPY = R(A), lectura no repetible
- OPX = RU(A)+W(A), OPY = RU(A)+W(A), lectura no confirmada

4.2.c

T1	T2	T3	T4
		L(A,S)	
		R(A)	
	L(A,S)		
	R(A)		
			L(B,X)
			RU(B)
			L(C,X)
			RU(C)
L(A,X)			
		L(F,S)	
		R(F)	
	L(B,S)		
		R(A)	
			W(B)
			c+U(B,C)
	R(B)		
	L(C,S)		
	R(C)		
	c+U(A,B,C)		
		c+U(A,F)	
RU(A)			
W(A)			
R(A)			
c+U(A)			

5. **(2 punts)** Suposeu la taula `logs` següent, amb registres d'accés a un servei web:

IP	usuari	data	acció	durada
10.0.0.2	Joan	22/12/2018 11:30	accés	40
10.0.16.1	Carla	22/12/2018 11:40	compra	20
10.0.14.1	Jordi	22/12/2018 11:41	venda	22
10.0.1.2	Pere	23/12/2018 16:50	accés	10

10.0.0.4	Maria	23/12/2018 17:00	consulta	95
10.0.2.1	Carla	23/12/2018 17:14	venda	21
10.0.0.2	Pere	28/12/2018 09:00	compra	42
10.0.0.2	Pere	28/12/2018 09:01	compra	15
10.0.0.2	Pere	28/12/2018 09:02	compra	41
10.0.2.2	Joan	28/12/2018 20:00	consulta	32
10.0.3.2	Maria	30/12/2018 14:16	accés	33

5.1 Creeu un arbre B+ d'alçada 2 (h=2) i d'ordre 2 (d=2) sobre l'atribut `durada`, amb un 75% d'ocupació.

5.2 La taula logs ha crescut al llarg de l'any, i el servei web disposa ara de 300 usuaris únics, on cadascun d'ells ha fet en mitjana 50 accessos, 50 compres, 50 vendes, i 50 consultes. Les files de la taula es distribueixen uniformement entre 800 pàgines. Considereu que s'ha definit un índex agrupat multiatribut pels atributs `acció,usuari` d'ordre 100, ple al 75%. Per cadascuna de les consultes següents:

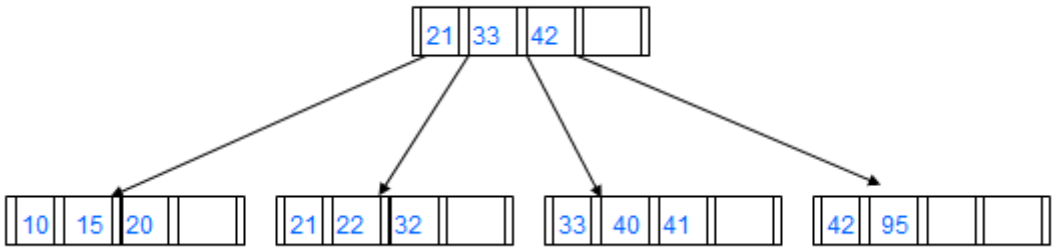
- Expliqueu com es resoldrà la consulta.
- Indiqueu el cost de la consulta en número de pàgines d'índex i número de pàgines de dades, mostrant els càlculs que feu.

a) SELECT * FROM log WHERE usuari = 'maria'

b) SELECT DISTINCT(usuari) FROM log WHERE accio = 'venda'

SOLUCIÓ

5.1



5.2.a

L'índex no afavoreix la consulta, pel que les dades s'obtindran amb un sequential access.

Cost accés índex = 0

Cost accés dades = 800 pàgines de dades

5.2.b

L'índex acció, usuari permet obtenir el resultat accedint únicament a l'índex:

n = d * 2 * ocupació = (100 * 2 * 75) / 100 = 150 valors/node

h = $\log_{151}$ 60000 = 2.1 => 3 nivells

50 vendes * 300 usuaris = 15000 vendes

Cost = 3 + 15000 / 150 – 1 = 3 + 100 – 1 = 102 pàgines d'index.

Temps: 3 h

1. (3 punts) Considereu l'esquema de la base de dades següent:

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

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

create table producteComprat
(numComanda integer,
idProducte char(9),
quantitat integer check(quantitat>0),
primary key(idProducte, numComanda),
foreign key (idProducte) references producte ,
foreign key (numComanda) references comanda);
// Hi ha una fila a la taula per cada producte comprat
// en una comanda.
```

1.1. Doneu una sentència SQL per obtenir els productes comprats que no s'han demanat en comandes servides després de l'instant 333.

```
select pc.idproducte
from comanda c natural inner join producteComprat pc
group by pc.idproducte
having max(c.instantServida) <=333

select distinct pc.idProducte
from productescomprats pc
where not exists (select * from productescomprats pc2, comandes c
        where pc2. idProducte = pc.idProducte and
              pc2.numComanda = c.numComanda    and
              c.instantServida >333)
```

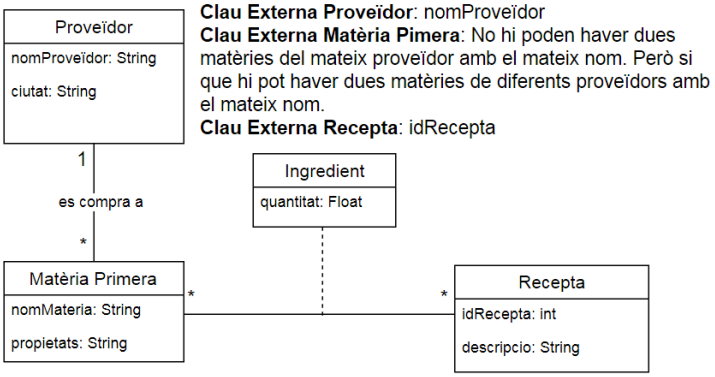
1.2. Escriu una asserció en SQL que asseguri que l'import de les comandes sigui igual a la suma de l'import de cadascun dels productes comprats en la comanda. Cal tenir en compte que l'import d'un producte comprat en una comanda és el resultat de multiplicar el preu del producte per la quantitat del producte que s'ha comprat en la comanda.

```
create assertion control_import check
(not exists (select *
        from comanda c
        where c.import != (select sum(p.preu*pc.quantitat)
                from producteComprat pc natural inner join producte p
                where pc.numComanda = c.numComanda))

create assertion control_import check
(not exists (select *
        from comanda c natural inner join
                producteComprat pc natural inner join producte p
        group by c.numComanda
        having c.import != sum(p.preu*pc.quantitat))
```

1.3. Escriu una seqüència d'operacions d'àlgebra relacional per obtenir les comandes en les que s'ha comprat un únic producte.

```
A= productecomprat[numComanda, idProducte]
B = A{numComanda -> nc, idProducte ->idp}
C = A[numComanda = nc, idProducte != idp]B
D= C[numComanda] /* pedidos con más de un producto */
E = A[numComanda] /* todos los pedidos */
F = E – D              /* pedidos con 1 solo producto */
```

1.4. Suposem el diagrama UML següent. Feu la traducció de UML a model relacional, indicant taules, atributs, claus primàries, claus foranes i restriccions de UNIQUE i NOT NULL que es deriven del diagrama.



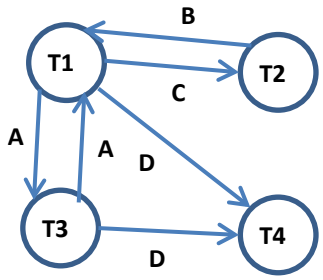

```
proveidors(nomProveidor, ciutat)
materiesPrimeres(nomProveidor, nomMateria, propietats)
        {nomProveidor} referencia proveidors
receptes(idRecepta, descripcio)
ingredients(nomProveidor, nomMateria, idRecepta, quantitat)
        {nomProveidor,nomMateria} referencia materiesPrimeres
        {idRecepta} referencia receptes
```

2. (2.5 punts)

2.1. 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	T4
10		RU(B)		
20		W(B)		
30			R(D)	
40	RUA)			
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				RU(D)
140				W(D)
145	W(A)			
150			W(F)	
160	RU(C)			
170	W(C)			
180		RU(C)		
190		W(C)		
200				COMMIT
210		COMMIT		
220			COMMIT	
230	COMMIT			

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



3.1-b) 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).

Es produeix una actualització perduda entre T1 i T3 sobre el gràNul A.
Anàlisi Inconsistent entre T1,T2

3.1-c) Quins horaris serials hi són equivalents? Justificar la resposta.

No existeix cap horari serial equivalent, atès que hi ha interferències.

3.1-d) És recuperable aquest horari? Justificar la resposta.

No, ja que per exemple T2 llegeix C i confirma abans que T1 que també ha llegit i escrit C

2.2. Suposem que s'incorporen tècniques de reserva. Les transaccions estan definides com SET TRANSACTION ISOLATION LEVEL REPEATABLE READ.

- Donar l'horari resultant d'aplicar les reserves per aquest nivell d'aïllament. El nou horari ha d'incloure, a més de les operacions que executen les transaccions (R, RU, W, COMMIT), les operacions de petició i alliberament de reserves (LOCK, UNLOCK), i l'ordre d'execució de totes aquestes operacions.
- En cas que en algun instant de l'horari no es pugui executar cap operació, no continueu i justifiqueu el motiu.
- Si hi ha alguna interferència en l'horari resultat digues quina/es és/són i entre quines transaccions es produeix/en. Altrament, digues quin és l'horari serial equivalent.

#Acc	T1	T2	T3	T4
10		L(B,X)		
20		RU(B)		
30		W(B)		
40			L(D,S)	
50			R(D)	
60	L(A,X)			
70	RU(A)			
80	L(D,S)			
90	R(D)			
100	L(B,X)			
110			L(E,X)	
120			RU (E)	
130			W(E)	
140			L(A,X)	
170				L(D,X)
250		L(C,X)		
260		RU(C)		
270		W(C)		
280		c+u		
290	RU(B)			
300	W(B)			
310	W(A)			
320	L(C,X)			
330	RU(C)			
340	W(C)			
350	c+u			
360			RU(A)	
370			W(A)	
380			L(F,X)	
390			RU(F)	
400			W(F)	
410			c+u	
420				RU(D)
430				W(D)
440				c+u

Horari serial equivalent: t2;t1;t3;t4

3. **(2.5 punts**) Suposeu la base de dades de l'exercici 1. Suposeu que hi ha definit el procediment emmagatzemat `nouProducteComprat` que s'invoca per cadascun dels productes comprats en una comanda, passant com a paràmetre el número de comanda, el producte comprat i la quantitat del producte comprat en la comanda. En la mateixa base de dades hi ha definit el disparador `disparadorBDComandes`.

```
10    create or replace function nouProducteComprat(numCom integer, numProd
11    char(9), qtt int)
12    returns void language plpgsql as $$
13    begin
14    if((not exists(select * from comanda
15                where numComanda = numCom) or
16      (not exists(select * from producte
17                where idproducte = numProd)))) then
18                raise exception 'La comanda o el producte no existeixen';
19    elseif(exists(select * from producteComprat
20                where idproducte = numProd and numComanda = numCom)) then
21                raise exception 'El producte comprat ja es a la comanda';
22    else insert into productecomprat values(numCom,numProd,qtt);
23    end if;
24    return;
25    exception
26          when raise_exception then raise exception '%',SQLERRM;
27          when others then raise exception 'Error inern';
28    end; $$

29    create or replace function procDisparadorBDComandes()
30    returns trigger language plpgsql as $$
31    declare varNumComanda integer;
32           varImport integer;
33    begin
34    for varNumComanda in select numComanda from comanda
35    loop
36          varImport := (select sum(p.preu*pc.quantitat)
37                               from producte p, productecomprat pc
38                               where p.idproducte = pc.idProducte and
39                                     pc.numcomanda = varNumComanda);
40          update comanda
41          set import = varImport
42          where numComanda = varNumComanda;
43    end loop;
44    return null;
45    end; $$

46    create trigger disparadorBDComandes
47    after insert on producteComprat
48    for each row execute procedure procDisparadorBDComandes();
```

3.1. Partint del contingut de la base de dades que s'indica a continuació. Digueu quin és el contingut de les taules de la base de dades després de l'execució de cadascuna de les seqüències de sentències següents:

a)
```
select * from nouProducteComprat(333,'300',1);
select * from nouProducteComprat(333,'200',3);
```

b)
```
select * from nouProducteComprat(222,'100',1);
select * from nouProducteComprat(222,'300',1);
```

taula comanda

numComanda	instantFeta	instantServida	numTelf	import
111	1	10	123	8
222	11	20	123	12
333	21	30	123	0

taula producte

idProducte	nom	mida	preu
100	dodotus	gran	5
200	kloonex	petit	3
300	tollolets	normal	6

taula domicili

numTelf	nomCarrer	numCarrer	pis	porta	ciutat
123	Pont de Baix	24	1	2	Manlleu

taula producteComprat

numComanda	idProducte	quantitat
111	100	1
111	200	1
222	300	2

SOLUCIÓ

3.1.a) Les taules que canvien són:

taula comanda

numComanda	instantFeta	instantServida	numTelf	import
111	1	10	123	8
222	11	20	123	12
333	21	30	123	15

taula producteComprats

numComanda	idProducte	quantitat
111	100	1
111	200	1
222	300	2
333	300	1
333	200	3

3.1.b) Les taules que canvien són les següents. La segona sentència fa saltar una excepció perquè intenta afegir un producte comprat que ja està a la taula.

taula comanda

numComanda	instantFeta	instantServida	numTelf	import
111	1	10	123	8
222	11	20	123	17
333	21	30	123	0

taula producteComprats

numComanda	idProducte	quantitat
111	100	1
111	200	1
222	300	2
222	100	1

Es considera correcta una solució on diguin que l'estat de la base de dades és el mateix que l'inicial en cas que les dues sentències de l'apartat b formen una transacció.

3.2. Trobeu les sentències del procediment emmagatzemat que no compleixen els criteris de qualitat treballats a l'assignatura. Per cada sentència/es que no compleixin els criteris de qualitat:

- Indiqueu el número/s de la/es sentències que no compleixen el criteri
- Indiqueu quin és el criteri de qualitat que no compleixen
- Justifiqueu perquè no el compleixen
- Doneu una nova versió del procediment emmagatzemat `nouProducteComprat` que sí que compleixi el criteri

14 a 18: selects innecessaris perquè l'excepció es pot capturar mitjançant la violació de la condició de foreign key

19 a 21: select innecessari perquè l'excepció es pot capturar mitjançant la violació de la restricció de unique/primary key

```
create function nouProducteComprat(numCom integer, numProd char(9), qtt int)
returns void language plpgsql as $$
begin   insert into producteComprat values(numCom, numProd, qtt);
return;
exception when foreign_key_violation then
            raise exception 'La comanda o el producte no existeixen';
          when unique_violation then
            raise exception 'El producte comprat ja es a la comanda';
          when others then raise exception 'Error intern';
end; $$
```

3.3. El tipus del disparador que s'ha usat en la implementació que us donem és el millor segons els criteris de qualitat treballats a l'assignatura, però la implementació del procediment `procDisparadorBDComandes` que activa el disparador no és la millor perquè és una implementació NO incremental. Doneu una implementació alternativa del procediment que faci el mateix però que sigui incremental.

```
create or replace function incrementPreuComanda()
returns trigger language plpgsql as $$
    declare pr integer;
    begin
        pr := (select preu from producte
               where idProducte = NEW.idProducte);
        update comanda
        set import = import + NEW.quantitat*pr
        where numComanda = NEW.numComanda;
    return null;
end; $$
```

4. (2 punts) Suposem la taula `comanda` de l'exercici 1.

`comanda(numComanda, instantFeta, instantServida, numTelf, import)`

Hi ha 5.000.000 files a la taula `comanda` distribuïdes uniformement en 50.000 pàgines de dades. Les pàgines tenen una mida de 4096 bytes. S'ha decidit crear dos índexs, implementats com un arbre B+, sobre la taula.

- Un índex agrupat definit sobre l'atribut `numComanda`. L'atribut `numComanda` ocupa 6 bytes. Els apuntadors a nodes de l'índex ocupen 3 bytes. Els apuntadors a files de la taula ocupen 4 bytes.
- Un índex no agrupat definit sobre els atributs `numTelf`, `instantFeta`. L'ordre d de l'índex és 146. L'arbre està ple al 75%. L'arbre té 3 nivells. Les comandes de la base de dades s'han fet des de 20.000 telèfons diferents, i cada telèfon ha fet una mitjana de 250 comandes.

Mostra en detall com has fet els càlculs que es demanen a continuació:

4.1. La consulta següent no es veu afavorida per usar cap dels dos índexs definits. Calcula el cost (número de pàgines accedides) de resoldre-la.

- select * from comanda where import = 50

4.2. Índex per numComanda.

4.1-a) Calcula l'ordre d òptim de l'índex

4.1-b) Suposant l'índex té una ocupació del 80%, i suposant l'ordre que has trobat en l'apartat anterior:

- Calcula quants valors hi haurà a cada node de l'arbre.
- Calcula quantes pàgines ocuparan els nodes fulla de l'arbre.
- Calcula quants nivells tindrà l'arbre B+.

4.1-c) Suposant les dades calculades en els apartats anteriors, calcula el cost (número de pàgines accedides) de resoldre la consulta següent usant l'índex

- select * from comanda where numComanda = 55555

4.3. Índex per numTelf, instantFeta. Calcula el cost (número de pàgines accedides) de resoldre la consulta següent usant aquest índex.

- select * from comanda where numTelf = '123'

SOLUCIÓ:

4.1) com que no usem índex, s'ha de fer accés seqüencial a totes les pàgines de dades per tant el cost és 50.000

4.2.a)

4096 >= 6*2d+(3*(2d+1)) => d<= 227

4096 >= 6*2d + 4*2d + 2*3 => d<= 204

d òptima 204

4.2.b) n = 2*204*0,8 = 327

num pàgines = num nodes fulla

5000000 valors / 327 valors/nodesfulla = 15291 pàgines

$\log_{328}500000 = 2,66$ => 3 nivells

4.2.c) Com que numComanda és PK, només busquem una fila

h + 1 = 3 + 1

4.3) num de valors per node = 2*146 * 0,75 = 219

busquem en mitjana 250 comandes

h + F + D(numTelf = '123') = 3 + (250/220 -1) + 250 = 254

Temps: 3 h
Notes 21 gener tarda **Revisió: 21 gener tarda**
Cada pregunta en un full separat

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

```
CREATE TABLE EquipsFutbol(
nomEquip char(30),
localitat char(50),
nomEstadi char(100) UNIQUE NOT NULL,
totalSalaris real not null check(totalSalaris>=0),
PRIMARY KEY(nomEquip));

CREATE TABLE Partits(
nomEquipLocal char(30),
dataPartit date,
nomEquipVisitant char(30),
golsEquipL integer NOT NULL CHECK(golsEquipL >=0),
golsEquipV integer NOT NULL CHECK(golsEquipV >=0),
PRIMARY KEY (nomEquipLocal,dataPartit,nomEquipVisitant),
FOREIGN KEY (nomEquipLocal) REFERENCES EquipsFutbol,
FOREIGN KEY (nomEquipVisitant) REFERENCES EquipsFutbol,
CHECK(nomEquipLocal <> nomEquipVisitant));

CREATE TABLE Jugadors(
dniJugador char(9),
nom char(50) NOT NULL,
nomEquip char(30) NOT NULL,
salari real not null check(salari>=0),
PRIMARY KEY (dniJugador),
FOREIGN KEY (nomEquip) REFERENCES EquipsFutbol);
```

- nomEquip és l'equip on juga el jugador

```
CREATE TABLE Alineacions(
nomEquipLocal char(30),
dataPartit date,
nomEquipVisitant char(30),
dniJugador char(9),
gols integer NOT NULL CHECK(gols>=0),
numTargetes integer NOT NULL CHECK(numTargetes>=0),
PRIMARY KEY (nomEquipLocal,dataPartit,
               nomEquipVisitant, dniJugador),
FOREIGN KEY (nomEquipLocal,dataPartit,nomEquipVisitant)
                          REFERENCES Partits,
FOREIGN KEY (dniJugador) REFERENCES Jugadors);
```

- el jugador dniJugador ha estat alineat al partit idenficat per nomEquipLocal, dataPartit, nomEquipVisitant
- els gols i targetes són les que han posat al jugador durant el partit.

1.1) Escriviu una sentència SQL per obtenir els equips de futbol que han jugat, com a equip local, en partits contra més de 3 equips diferents i que, a la vegada, en aquests equips locals, no hi juga cap jugador que tingui targetes. En el resultat de la consulta hi ha d'haver el nom dels equips locals i la quantitat de partits que han jugat a casa amb equips diferents.

```
Select p.nomequiplocal, count(distinct p.nomequipvisitant)
From partits p
Where not exists (select *
                  From jugadors j, alineacions a
                  Where j.nomEquip=p.nomEquipLocal and
                        j.dniJugador=a.dniJugador and
                         a.numtargetes>0)
Group by p.nomequiplocal
Having count(distinct p.nomequipvisitant) >3;
```

1.2) Vistes.

a. Definiu una vista amb els partits empatats (és a dir, els que tant l'equip local com el visitant han fet el mateix número de gols). La vista ha de incloure tots els atributs de la taula partits i la clausula WITH CHECK OPTION.

```
CREATE VIEW partitsEmpatats AS
      SELECT *
      FROM Partits p
      WHERE p.golsEquipL=p.golsEquipV
      WITH CHECK OPTION;
```

b. Doneu una sentència update de la vista, que inclogui la modificació de l'atribut golsEquipL, i que NO violi la restricció WITH CHECK OPTION.

```
update partitsEmpatats
set golsEquipL = golsEquipL+2,
    golsEquipV = golsEquipLV+2
```

c. Doneu una sentència update de la vista, que inclogui la modificació de l'atribut golsEquipL, i que SI que violi la restricció WITH CHECK OPTION.

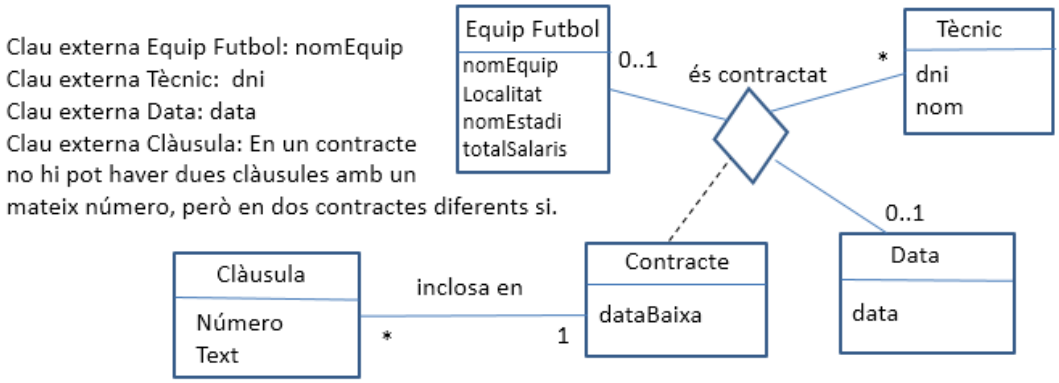
```
update partitsEmpatats
set golsEquipL = golsEquipL+2
```

1.3) La cardinalitat de les taules de la base de dades és respectivament EF, P, J, A (amb EF,P,J,A > 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. Justifiqueu la resposta.

```
a. R = Partits * Alineacions

b. R= Partits x Alineacions

c. A= Partits[golsEquipPL]
   B= Partits[golsEquipPV]
   R= A U B

(iv) A = Alineacions[nomEquipLocal,dataPartit,nomEquipVisitant]
     B = Partits[nomEquipLocal,dataPartit,nomEquipVisitant]
     R = A ∩ B
```

```
i.   card(R)= A
ii.  card(R)= P x A
iii. card(R)màxima = 2P, card(R) mínima = 1
iv.  card(R)màxima P, card(R) mínima = 1
```

1.4) Suposeu el model conceptual en UML següent. Doneu el disseny lògic que s'obté fent la traducció a model relacional. Cal incloure, taules, atributs, claus primàries, claus foranies i restriccions NOT NULL i UNIQUE que siguin necessàries.




```
equipsFutbol(nomEquip, localitat, nomEstadi, totalSalaris)
tècnics(dni, nom)
contractes(dni, dataAlta, nomEquip, dataBaixa)
      {dni} referencia tècnics
      {nomEquip} referencia equipsFutbol NOT NULL
      UNIQUE(nomEquip, dataAlta)
       --és també correcte posar com a PK nomEquip, dataAlta
       --i posar com a UNIQUE dni, dataAlta
clàusules(dni, dataAlta, número, text)
      {dni, dataAlta} referencia contractes
```

2) **(2.5 punts)** Suposeu la base de dades de l'exercici 1. Suposeu que a la base de dades s'afegeixen les taules següents. S'inclou a continuació algunes de les files que contenen les taules.

```
t_files(nomTaula, quantesFiles)
            -- conté per cada taula quantes files hi ha a la taula,
            -- en cas que no n'hi hagi cap contindrà un 0
    EquipsFutbol 22
    Jugadors    420

t_sentencies(nomSentencia, nomTaula, quantesSentencies)
            -- conté per cada taula i per cada tipus de sentència, quantes vegades s'ha
                  executat una sentència del tipus sobre la taula
            --en cas que no n'hi hagi cap, contindrà un 0
        insert          EquipsFutbol      2
        insert          Jugadors         10
        insert          Partits           6
        delete          Partits           1
        insert          Alineacions      80
        update          Alineacions      10
```

2.1) Considereu les restriccions següents:

a. Es mantingui correcte el valor de la columna *quantesFiles* de fila *Jugadors* de la taula *t_files*.

b. Es mantingui correcte el valor de la columna *quantesSentencies* de les files on apareix *Jugadors* a la taula *t_sentencies*.

c. Es mantingui correcte la columna *totalSalaris* de la taula *EquipsFutbol*, que ha de correspondre a la suma dels salaris dels jugadors de l'equip.

Indiqueu quins són els triggers que cal definir sobre la taula *Jugadors*. Per cada trigger cal indicar:

- Esdeveniment (Insert, Delete, Update – en cas d'update indicar atribut/s rellevants),
- Before/After
- Row/Statement

insert	after	statement
delete	after	statement
update	after	statement
insert	after	Row
delete	after	Row
update of nomEquip, salaris	after	Row

2.2) Explicar breument el què caldrà que facin els procediments a definir per al/s trigger/s *delete*. En cas de que no hagin estat necessaris triggers *delete*, explicar perquè no fan falta.

- After, statement
  - Incrementa en 1 la columna quantesSentencies de la a fila 'delete', 'jugadors' de la taula t_sentencies
- After, row
  - Decrementa en 1 la columna quantesFiles de la fila 'jugadors' de la taula t_files
  - Decrementa l'atribut salarisTotals de la taula EquipsFutbol, de l'equip OLD.nomEquip amb el valor de OLD.salari

2.3) Implementar els procediments necessaris per al/s trigger/s *update*. En cas de que no hagin estat necessaris triggers *update*, explicar perquè no fan falta.

```
CREATE FUNCTION upJugadorsStatement() RETURNS trigger AS $$
BEGIN
    UPDATE t_sentencies
    SET quantesSentencies = quantesSentencies + 1
    WHERE nomSentencia='update' and
        nomTaula='jugadors';
    RETURN NULL;
END; $$ LANGUAGE plpgsql;

CREATE FUNCTION upJugadorsRow() RETURNS trigger AS $$
BEGIN
    UPDATE equipsFutbol
    SET totalSalaris = totalSalaris + NEW.salari
    WHERE nomEquip=NEW.nomEquip;
    UPDATE equipsFutbol
    SET totalSalaris = totalSalaris -OLD.salari
    WHERE nomEquip=OLD.nomEquip;
    RETURN NEW;
END; $$ LANGUAGE plpgsql;
```

3) **(2.5 punts)** S'ha creat una taula empleats(numEmpl, nom, sou, ciutat, categoria). La taula empleats té 100.000 tuples i els valors de numEmpl estan repartits uniformement entre 1 i 100.000. El factor de bloqueig del fitxer de dades és de 20 files per pàgina. Es defineixen dos índexs sobre la taula empleats:

- un índex B+ agrupat per l'atribut `numEmpl`, amb ordre d=60 i amb nodes plens al 80% de la seva capacitat.
- un índex B+ no agrupat per l'atribut `sou`, amb ordre d=80 i amb nodes plens al 75% de la seva capacitat.

3.1) Expliqueu quina diferència hi ha entre un índex agrupat i un índex no agrupat.

3.2) Digueu quants índexs agrupats hi pot haver definits com a màxim sobre una taula. Justifiqueu la resposta.

3.3) Es vol obtenir totes les files que compleixen les condicions `numEmpl>95.000 AND sou>2000`. Sabent que hi ha 400 files que compleixen sou>2000, i hi ha 100 files que compleixen les dues condicions. Mostreu com calculeu el cost de la consulta en cadascun dels casos següents.

a. Emprant recorregut seqüencial de la taula *empleats*.
b. Emprant l'índex B+ agrupat per *numEmpl*.
c. Emprant l'índex B+ no agrupat per *sou*.

3.4) Digueu si hi ha algun índex que es pogués afegir a la taula empleats i que millorés l'eficiència de la consulta. Justifiqueu la resposta en funció dels costos.

3.1. Veure transparències assignatura
3.2. Veure transparències assignatura
3.3.

a. Recorregut seqüencial = 100.000 files / 20 files-pàgina = 5000 pàgines
b. índex numEmpl
   núm valors per node = 2d*ocupació = 2*60*0,8 = 96 valors
   núm apuntadors per node = 96+1 = 97 apuntadors
   cost = h + D
   h = alçada de l'arbre = $\log_{97}100.000 = 2.51 \Rightarrow 3$
   D = número de pàgines de dades =
   5000 files compleixen la condició / 20 files-pàgina = 250
   cost = 3 + 250 = 253 pàgines
c. índex sou
   núm valors per node = 2d*ocupació = 2*80*0,75 = 120 valors
   núm apuntadors per node = 120+1 = 121 apuntadors
   cost = h + (F -1) + D
   h = alçada de l'arbre = $\log_{121}100.000 = 2.4 \Rightarrow 3$
   F = número de pàgines, nodes fulla = 400 valors sou>2000 / 120 valors =
   4 pàgines, nodes fulla
   D = número de pàgines de dades = 400 pàgines en el cas pitjor, una per fila.
   cost = 3 + (4 - 1) + 400 = 406 pàgines

3.4.

Un índex no agrupat, multiatribut, concretament pels atributs sou, numEmpl.
Hauria de ser no agrupat, perquè ja n'hi ha un d'agrupat.
El cost al ser no agrupat hauria de ser h+F+D

- h=3
  - Tindrà entre 3 o 4 nivells segons el tamany del valor.
  - Com que el tamany del valor serà més gran que en els altres dos arbres, la d serà més petita (no hi cabran tants valors per node). I per tant, podria ser que l'arbre passés a ser de 4 nivells.
  - Considerarem 3 nivells.
- F = 400 / n
  - hi ha 400 valors que cal considerar, ja que es fa accés a les fulles de l'arbre a partir del valor "2.000, 95.000", i cal buscar valors superiors a 2.000 i superiors també a 95.000. Cal entendre que els 100 valors que compleixen les dues condicions no estaran seguits. Només cal pensar en el valor "2.100, 85.000". Aquest valor estarà entre els 400 però no serà un dels 100 que ens interessen. I després d'aquest si que trobarem valors que tornen a complir les condicions com el valor "2.100, 96.000".
  - Per altra banda el valor n el desconeixem. Desconeixem la d, encara que sabem que serà inferior a la dels altres arbres, i desconeixem com estarà de ple el arbre. Posem-nos en un cas dolent de d=30 i 60% d'ocupació. Aleshores n = 30*2*0,7 = 42 valors/node.
- D = 100 en el cas pitjor.

El cost seria 3 + (10 -1) + 100 = 112, considerablement millor que els altres dos índexos.

4) **(2.5 punts)** Donades les dues taules següents:

```
Clients(numClient, nom, instantNaixement, població, saldoTotal)
Comptes(numCompte, titular, vigent, saldo)
    {titular} referencia Clients
```

El contingut en un cert instant de les taules és:

numClient	nom	instantNaixement	poblacio	saldoTotal
1	Anna Puig	422	Barcelona	28.900
2	Rosa Carrasco	145	Barcelona	33.250

numCompte	titular	vigent	saldo
11	1	N	28.000
21	1	S	-1.500
31	1	S	2.400
42	2	S	13.250
52	2	N	20.000

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

#op	T1	T2	T3

1	`select saldoTotal`
	`from clients`
	`where numClient =1`		

2		`update clients`
		`set saldoTotal=`
		`saldoTotal * 1.1`
		`where numClient = 1`	

3			`select saldoTotal`
			`from clients`
			`where numClient = 1`

4	`select max(saldoTotal)` `from clients`		

5			`select *`
			`from comptes`
			`where titular = 2`

6		`update comptes`
		`set titular = 2`
		`where titular = 1`
		`and vigent = 'N'`	

7			`select saldo`
			`from comptes`
			`where titular =2`

8	`insert into clients`
	`values(3, 'Marta López',`
	`323, 'Barcelona', 50.000)`		

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.
b. Digueu quins horaris serials donen resultats equivalents a l'horari proposat.
c. Indiqueu, en cas que hi hagi interferències, quin és el nivell mínim d'aïllament necessari per evitar cadascuna d'elles.
d. Digueu si l'horari proposat és recuperable. Justifiqueu breument la resposta.

a. Sí, hi ha interferències.
- Entre T1 i T2 una lectura no repetible a T1 sobre la taula clients, granul 1. L'update de T2 entre els 2 select de T1 canvia el contingut de la taula.
- Entre T2 i T3 hi ha un fantasma que produeix T2 canviant el titular d'una fila de la taula comptes. Quan T3 fa el segon select sobre aquesta taula, el contingut ha canviat i apareix una nova fila. Granul, el fantasma és el compte 11 i la informació de control IC.

b. Com que hi ha interferències, és impossible que hi hagi un horari serial que doni un resultat equivalent.
c. Per evitar la lectura no repetible entre T1 i T2 necessitem nivell REPETEABLE READ. Per evitar el fantasma SERIALIZABLE.
d. No, no és recuperable. Per exemple, T3 fa lectures sobre clients i comptes i fa el commit abans que T2 que és la que escriu sobre clients i comptes. Si T2, en comptes de fer un commit, fes un rollback, T3 hauria confirmat la lectura d'uns continguts que no són veritat. També passa el mateix entre T1 i T3, T3 confirma quan encara no ho ha fet T1 que és la que escriu una de les taules que llegeix T3.

4.2) Suposeu que el gränul és la fila, que tenim un mecanisme per al control de la concurrència basat en reserves S, X i que la transacció T1 treballa a nivell d'aïllament de READ COMMITTED i T2 i T3 a nivell SERIALITZABLE.

a. Doneu l'horari que ha d'incloure, a més de les operacions que executen les transaccions (R, RU, W, COMMIT), les operacions de petició i alliberament de reserves (LOCK, UNLOCK), i l'ordre d'execució de totes aquestes operacions. Quan més d'una transacció espera per un mateix gränul, considereu que la política de la cua és FIFO (First In, First Out).
b. Indiqueu els horaris serials equivalents, en cas que n'hi hagi. Si no n'hi ha, justifiqueu la resposta.

T1	T2	T3
L(client,1,S)		
R(client,1)		
U(client,1)		
	L(client,1,X)	
	RU(client,1)	
	W(client,1)	
		L(client,1,S)
L(client,1,S)		
	L(compte,11,X)	
	RU(compte,11)	
	W(compte,11)	
	L(IC,X)	
	RU(IC)	
	W(IC)	
	C+U((client,1), (compte,11))	
		R(client,1)
R(client,1)		
U(client,1)		
L(client,2,S)		
R(client,2)		
U(client,2)		
		L(IC,S)
		R(IC)
		L(comptes,42,S)
		R(comptes,42)
		L(comptes,52,S)
		R(comptes,52)
		L(comptes,11,S)
		R(comptes,11)
		R(comptes,42)
		R(comptes,52)
		R(comptes,11)
L(client,3,X)		
RU(client,3)		
W(client,3)		
		C+U((client,1), (comptes,42), (comptes,52), (comptes,11))
C+U(client,3)		

b. No hi ha horaris serials equivalents perquè hi continua havent la interferència de lectura no repetible