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?

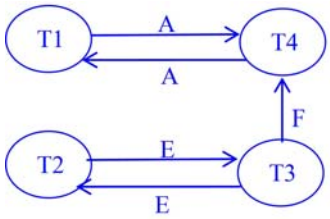
a.4. L'horari proposat, és recuperable? En cas afirmatiu doneu la definició de quan un horari és recuperable. En cas negatiu indiqueu alguna de les transaccions i operacions de l'horari que mostrin que no ho és.

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:



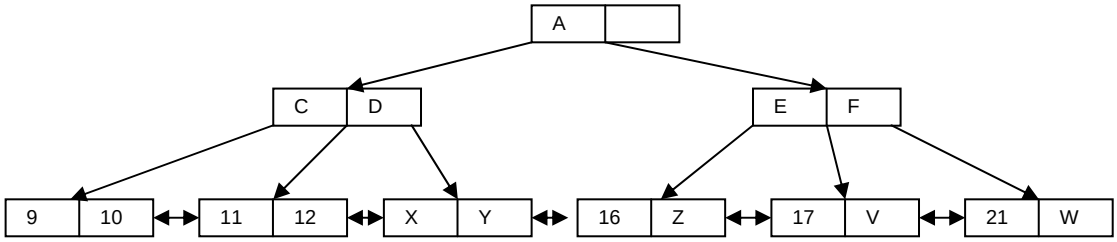
a.2) Entre T1 i T4 sobre el gràNul A es produeix una lectura no confirmada, i entre T2 i T3 sobre el gràNul E es produeix una lectura no repetible.
a.3) No existeix cap horari serial equivalent, donat que existeixen interferències.
a.4) L'horari no és recuperable. Per exemple, T4 recupera dades pendents 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.

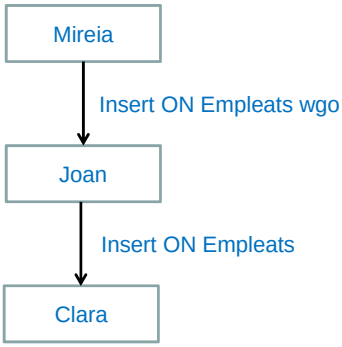
Fals, l'esquema d'informació és un esquema apart dels esquemes definits pels usuari. N´hi ha un per cada catàleg, i conté informació sobre els esquemes definits pels usuaris en el catàleg: noms i atributs de les taules, índexs, restriccions de columna, ...

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.

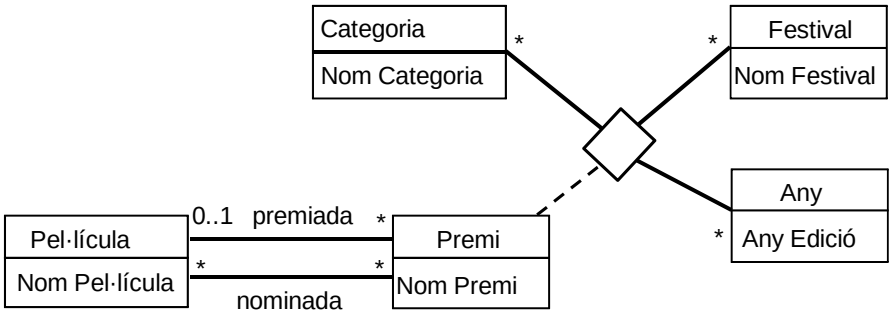
Fals. En el graf d'autoritzacions següent es pot veure que:



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.

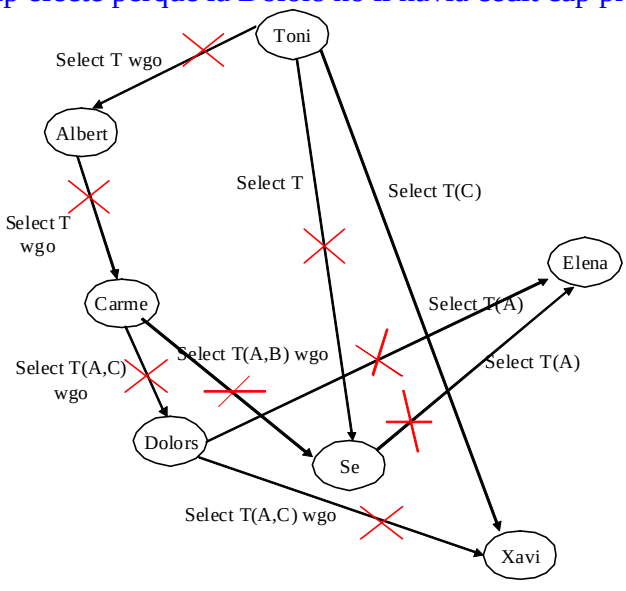
2.2) Considereu la taula professors(dni,nomProf,telefon), propietat d'en Toni. Suposeu també la seqüència de sentències següent relativa a autoritzacions sobre la taula professors. Cada sentència està numerada i s'indica el nom de l'usuari que vol executar-la.

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.

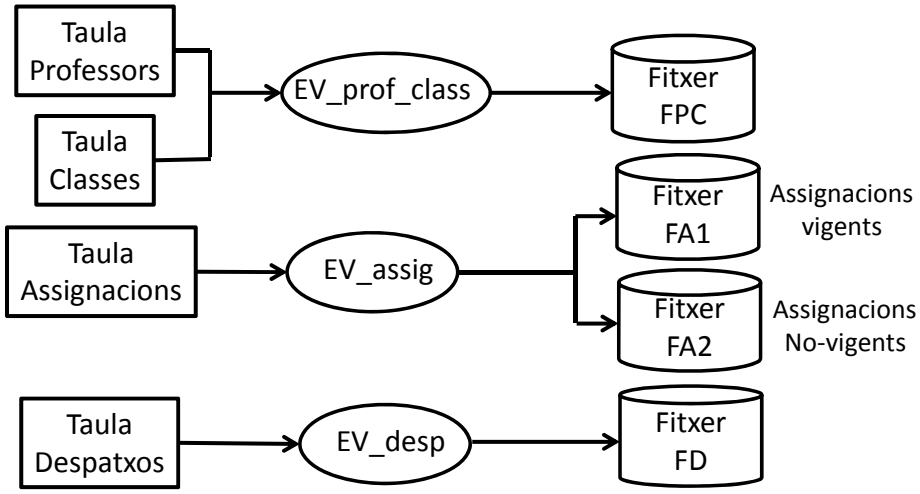
5) (2 punts) Considereu la base de dades de l'exercici 1.

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.

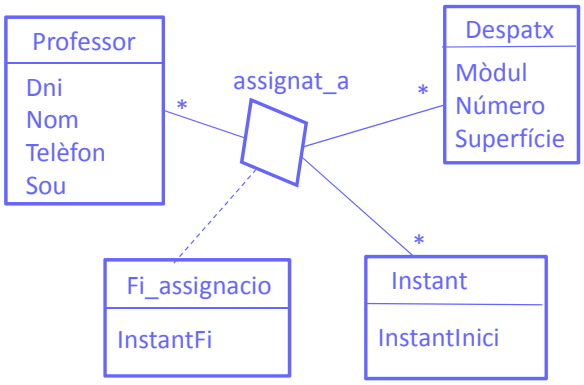
SELECT * FROM assignacions a WHERE a.dni > '444' AND a.modul='Omega'

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) = 10000/220 = 46 fulles), (46 apuntadors => 1 arrel)

$\lceil \log_{221} 10000 \rceil = 2$

Alçada de l’arbre mòdul = 2

(10000/(227*2*0,6) = 10000/273 = 37 fulles), (37 apuntadors => 1 arrel)

$\lceil \log_{274} 10000 \rceil = 2$

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 / 220) -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 / 273) -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 **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]
```

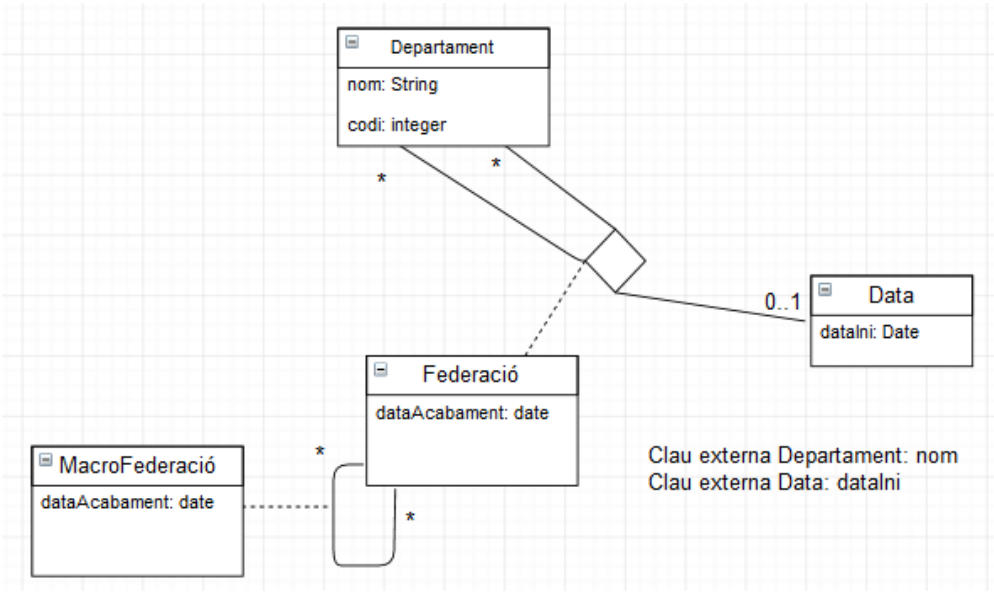
o bé

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:

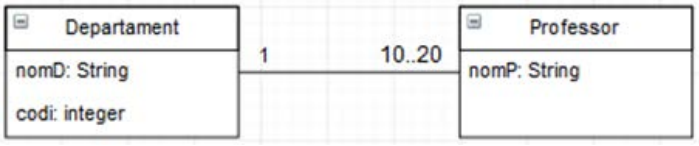


2.1 Doneu disseny lògic que s'obté de traduir el UML anterior a model relacional (cal incloure: taules, atributs, restriccions de clau primària, i restriccions de clau forana).

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.

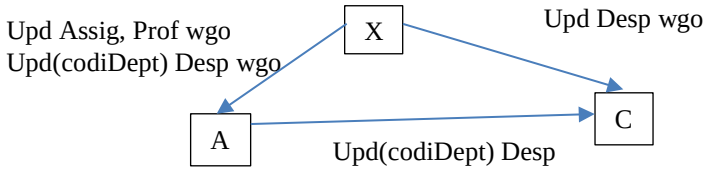
X: GRANT UPDATE ON despatxos to C WITH GRANT OPTION;

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

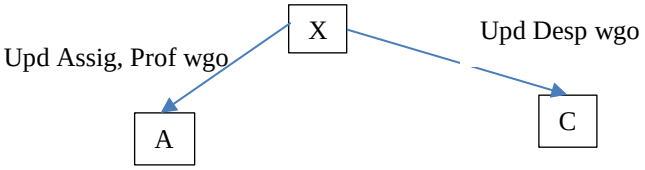
X: REVOKE UPDATE(codiDept) ON despatxos TO A RESTRICT.

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.

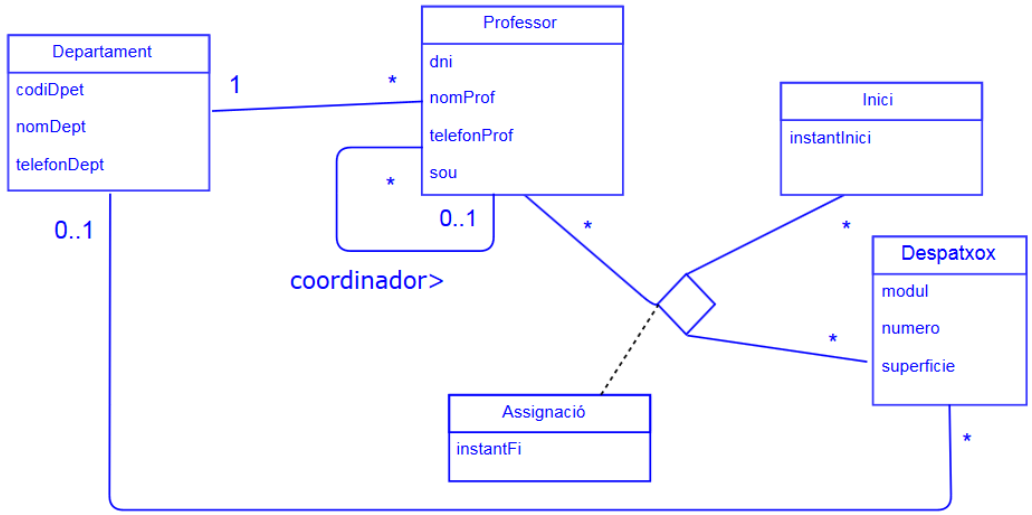
2.2 Definiu una asserció en SQL Standard per garantir el compliment de la restricció següent: "No existeix cap departament amb més de 5 professors que tinguin un coordinador que no és del departament".

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;
```

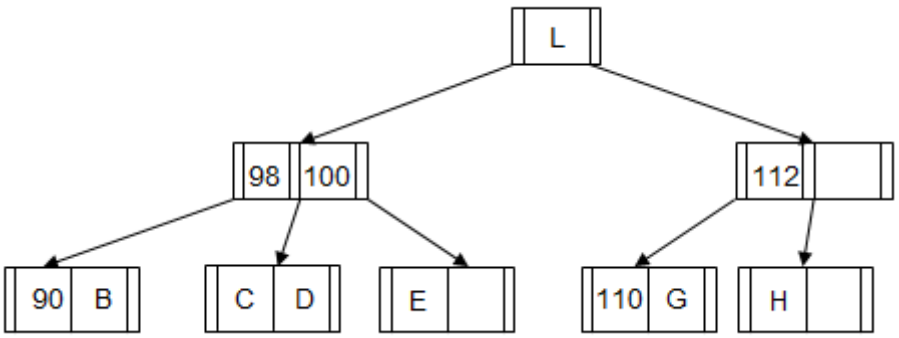
T2:

```
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 3 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'.

3 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);
```

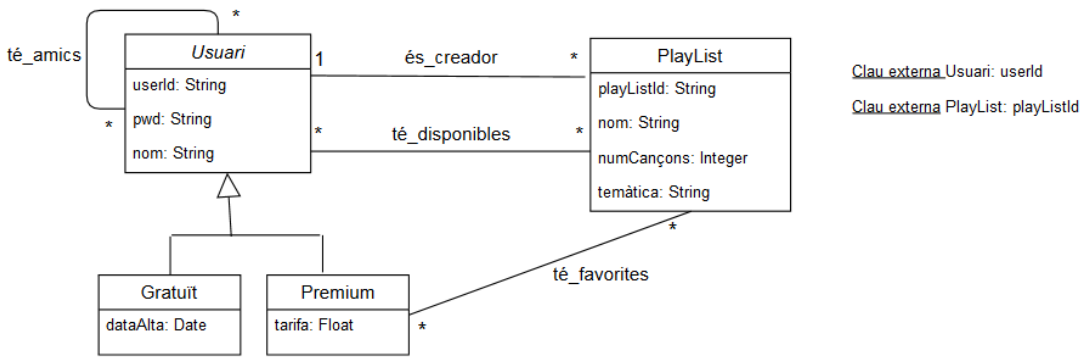
a) Correcte. Taula innecessària jugadors ju.
b) Correcte. DISTINCT innecessari.
c) Incorrecte. En una base de dades amb només una fila a la taula jugadors, la fila hauria de sortir en el resultat de la consulta, i en canvi la consulta tal com està no dóna cap resultat.

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

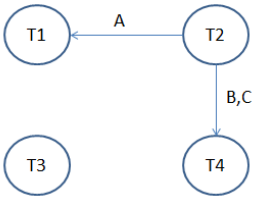
T1	T2
	RU(A)
RU(A)	
	W(A)
W(A)	
commit	
	commit

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:

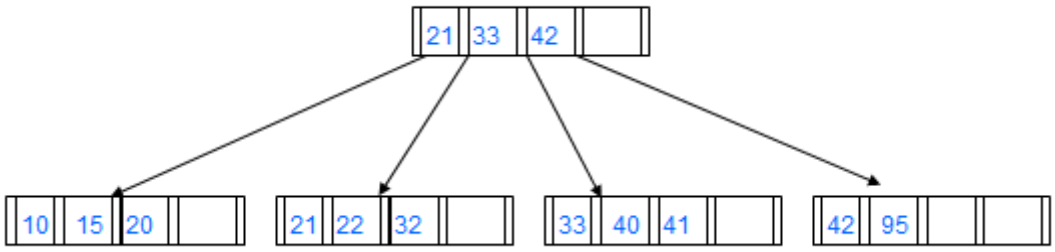
- Expliqueu com es resoldrà la consulta.
- Indiqueu el cost de la consulta en número de pàgines d'índex i número de pàgines de dades, mostrant els càlculs que feu.

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)
```

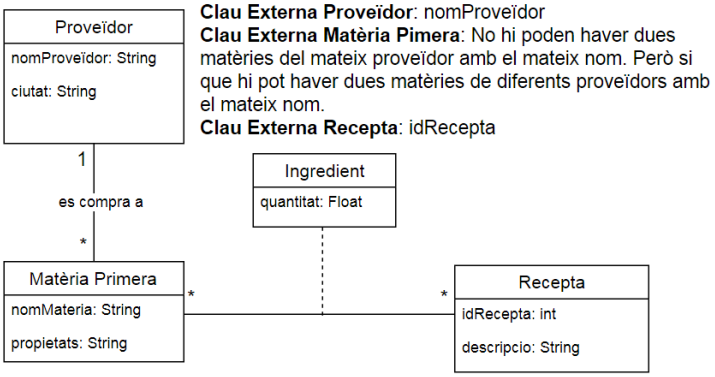
1.2. Escriu una asserció en SQL que asseguri que l'import de les comandes sigui igual a la suma de l'import de cadascun dels productes comprats en la comanda. Cal tenir en compte que l'import d'un producte comprat en una comanda és el resultat de multiplicar el preu del producte per la quantitat del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



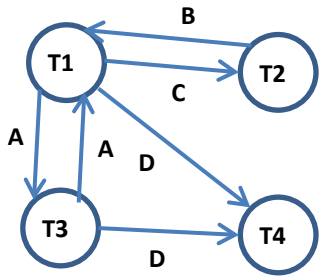

```
proveidors(nomProveidor, ciutat)
materiesPrimeres(nomProveidor, nomMateria, propietats)
        {nomProveidor} referencia proveidors
receptes(idRecepta, descripcio)
ingredients(nomProveidor, nomMateria, idRecepta, quantitat)
        {nomProveidor,nomMateria} referencia materiesPrimeres
        {idRecepta} 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 + (250/219 -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.

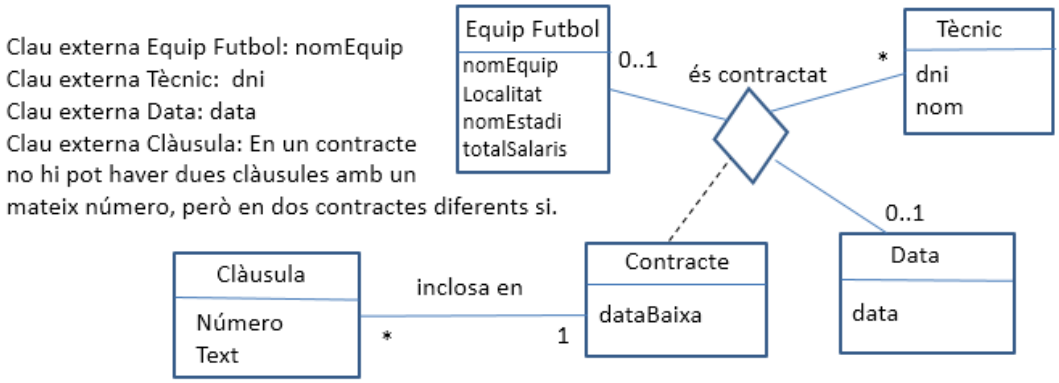
```
update partitsEmpatats
set golsEquipL = golsEquipL+2
```

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.

4.2) Suposeu que el gràNul 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 gràNul, 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

BD-GEI - Examen parcial **5.11.2020**

Temps: 90 minuts

Notes 23 novembre

Cada pregunta s'ha de lliurar en un full separat

Suposeu una base de dades amb les taules següents:

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

CREATE TABLE Partits(
idPartit integer,
nomEquipLocal char(30) NOT NULL,
dataPartit date NOT NULL,
nomEquipVisitant char(30) NOT NULL,
golsEquipL integer NOT NULL CHECK(golsEquipL >=0),
golsEquipV integer NOT NULL CHECK(golsEquipV >=0),
PRIMARY KEY (idPartit),
FOREIGN KEY (nomEquipLocal) REFERENCES EquipsFutbol,
FOREIGN KEY (nomEquipVisitant) REFERENCES EquipsFutbol,
UNIQUE(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 està contractat** el jugador

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

- el jugador dniJugador **ha estat alineat** al partit idenficat per idPartit
- els gols i targetes són les que han posat al jugador durant el partit.

1. **(1 punt)** Per cadascuna de les restriccions d'integritat següents.

- En cas que es pugui implementar com a restricció d'integritat de columna o de taula, en una sentència CREATE: Digueu la taula on s'haurien de definir, i la restricció d'integritat de columna o de taula en SQL.
- Si no es pot: Expliqueu breument perquè no es pot.

1.1 Un equip de futbol no pot jugar un partit contra ell mateix.

1.2 El número de gols d'un equip local en un partit ha de ser igual a la suma dels gols dels seus jugadors que han sigut alineats al partit.

1.3 Un equip de futbol no pot jugar dos partits en una mateixa data.

- RI1 – Si que es pot
  - A la taula partits
  - CHECK (nomEquipLocal<>nomEquipVisitant)
- RI2 – No es pot implementar
  - Perquè és una restricció que té a veure amb files de més d'una taula.
- RI3 – Es pot implementar parcialment. Amb els dos UNIQUE següents s'assegura que un equip de futbol no pot jugar com a local o com a visitant dos partits a la mateixa data. Però si que podria jugar un partit com a local i un com a visitant en una mateixa data.
  - A la taula partits
  - UNIQUE(nomEquipLocal,dataPartit), UNIQUE(nomEquipVisitant,dataPartit)

2. **(3 punts)** Doneu una sentència SQL per obtenir els equips de futbol (nomEquip) que no han jugat mai com a equip visitant en un partit empatat. A més es vol que un equip només surti si el salari total dels jugadors contractats a l'equip és més gran que la mitjana del pressupost de tots els equips.

```
select e.nomEquip
from equipsFutbol e natural inner join jugadors j
where not exists (select *
            from partits p
            where e.nomEquip = p.nomEquipVisitant and
                  p.golsEquipL=p.golsEquipV)
group by e.nomEquip
having sum(j.salari) > (select avg(ef.pressupost)
                  from equipsfutbol ef);
```

3. **(1.5 punt)** Doneu una seqüencia d'operacions en àlgebra relacional per obtenir els jugadors alineats en l'equip local que ha jugat el partit amb idPartit 333. Per cada jugador cal que surti el DNIJugador, el nom del jugador i el seu nom d'equip.

```
Una possible solució:
 A = partits(idPartit=333)
 B = A*alineacions
 C = B{DNIJugador -> DNIJugadorA}
 D = C[nomEquipLocal=nomEquip,DNIJugadorA=DNIJugador]jugadors
 R = D[DNIJugador, nom, nomEquip]
```

4. **(1 punt)** Doneu una sentència assertion per assegurar que en un partit no s'ha alineat més de 30 jugadors.

```
CREATE ASSERTION numeroJugadorsPartit CHECK
    (NOT EXISTS (SELECT p.idPartit
                 FROM alineacions a
                 GROUP BY p.idPartit
                 HAVING COUNT(*)>30));
```

5. **(1 punt)** Suposeu la vista següent:

```
CREATE VIEW massaCar (nomEquip, pressupost, costJugadors)AS
    SELECT e.nomEquip, e.pressupost, sum(j.salari)
    FROM EquipsFutbol e NATURAL INNER JOIN Jugadors j
    WHERE e.localitat in ('Barcelona','Madrid')
    GROUP BY e.nomEquip;
```

5.1 Doneu un contingut de les taules de la base de dades que tingui com a mínim 3 files a la taula jugadors i que faci que quan es consulti la vista el resultat sigui el següent:

massaCar	nomEquip	pressupost	costJugadors
	Barça	50M	70M

equipsFutbol	nomEquip	localitat	estadi
	Barça	BCN	CN

jugadors	DNIJugador	nom	nomEquip	salari
	111	Ansu	Barça	10M
	222	Messi	Barça	40M
	333	Piqué	Barça	20M

5.2 Justifiqueu si la vista massaCar és o no actualitzable.

```
No, per exemple perquè hi ha join o sum.
```

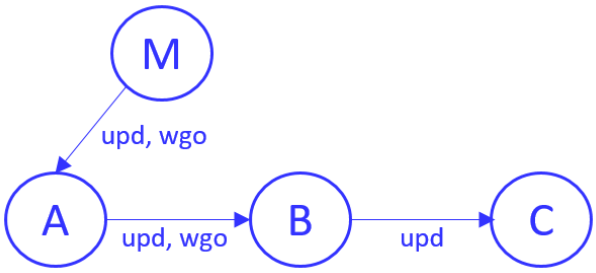
6. **(1 punt)** Suposeu que el propietari de les taules de la base de dades és M.

6.1 Doneu el diagrama d'autoritzacions, després de que s'executin les sentències següents:

M: GRANT update ON partits TO A WITH GRANT OPTION;

A: GRANT update ON partits TO B WITH GRANT OPTION;

B: GRANT update ON partits TO C;



6.2 Suposeu la situació donada pel diagrama d'autoritzacions anterior. Doneu una sentència UPDATE de la taula partits per la qual l'usuari A no tingui suficients permisos.

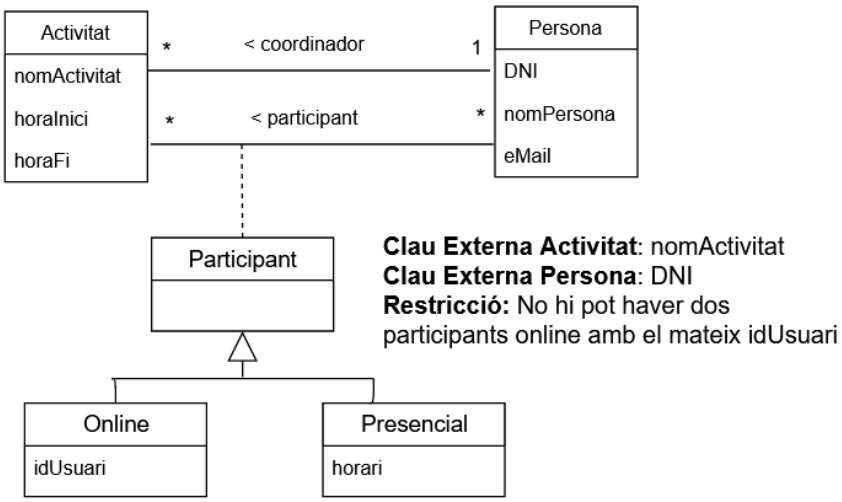
Qualsevol sentència UPDATE que per modificar necessiti consultar algun atribut de la taula partits o de qualsevol altre taula.

6.3 Suposeu la situació donada pel diagrama d'autoritzacions anterior. Doneu el nou diagrama d'autoritzacions un cop executada la sentència següent. Justifiqueu breument la resposta.

M: REVOKE GRANT OPTION FOR update FROM partits TO A RESTRICT;

El diagrama d'autoritzacions no canvia perquè la sentència no es pot executar. El motiu és que en ser RESTRICT no es pot executar perquè no només estaria perdent privilegis A, sinó també B i C. Ja que B i C no estan rebent els privilegis per cap altre camí.

7. **(1.5 punts)** 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.



persones(DNI, nomPersona, email)
activitats(NomActivitat, horaInici, horaFi, DNICoordinador)
{DNICoordinador} referencia persones
NOT NULL DNICoordinador
participants(DNI, nomActivitat)
{DNI} referencia persones
{nomActivitat} referencia activitats
participantsOnline(DNI, nomActivitat, idUsuari)
{DNI, nomActivitat} referencia participants
UNIQUE(idUsuari)
participantsPresencials(DNI, nomActivitat, horari)
{DNI, nomActivitat} referencia participants

BD-GEI - Examen final **15.1.2021**

Temps: 2 hores

Notes 27 gener Revisió: 28 gener **Cada pregunta s'ha de lliurar en un full separat**

1. **(1 punt)** Donades les tres taules següents:

```
professors(dni, nomProf, telefon, sou)
despatxos(modul, numero, superficie)
assignacions(dni, modul, numero, instantInici, instantFi)
      {dni} referencia professors
      {modul,numero} referencia despatxos
```

Considereu el següent fragment de codi Java que implementa, utilitzant JDBC, l'assignació de tots els professors que tenen un sou superior al valor de la variable sou al despatx identificat per les variables modul i numero, amb instancInici = 100 i instanFi = null. En cas que algun dels profesors ja estigui assignat al despatx s'acaba el programa mostrant un missatge d'error.

```
try {
 /* Tenim ja una connexió c establerta amb la base de dades */
 int sou = 1200;
 String modul = "omega";
 String numero = "118";

 Statement st1 = c.createStatement();
 ResultSet rs1 = st1.executeQuery("select dni from professors where sou < " + sou);


 while (rs1.next()) {
   String dniProf = rs1.getString(1);

   // Consultem si el professor ja ha estat assignat
   Statement st2 = c.createStatement();
   ResultSet rs2 = st2.executeQuery("select * from assignacions"
                     + " where dni='" + dniProf + "'"
                     + " and modul='" + modul + "'"
                     + " and numero='" + numero + "'");

   // Si rs2 retorna resultat, aleshores el professor ja està assignat
   if (rs2.next()) {
         System.out.println("Error: Un dels professors ja està assignat");
         c.rollback();
         break;    /* surt del bucle  */
   }
   // En cas contrari, procedim a crear l'assignació
   else {
         Statement st3 = c.createStatement();
         st3.executeUpdate("insert into assignacions values ("
                          + "'" + dniProf + "',"
                          + "'" + modul + "',"
                          + "'" + numero + "',"
                          + "100, null)");
         st3.close();
    }
    rs2.close(); st2.close();
    }
 st1.close(); rs1.close();c.commit(); c.close();
 }
catch (ClassNotFoundException ce) {
         System.out.println ("Error al carregar el driver");}
catch (SQLException se) {
         System.out.println ("Excepcio: "+ se.getMessage());}
```

Donat aquest fragment de codi, identifiqueu quins criteris de qualitat no es compleixen. Per cadascun d'ells, expliqueu perquè no es compleix, i expliqueu amb detall quins canvis serien necessaris en la codificació per tal de satisfer aquests criteris.

SOLUCIÓ:

No es compleixen els següents criteris de qualitat:

- 1 - Execució de sentència SQL innecessària. La comprovació feta per st2 sobre l'existència de l'assignació és innecessària, ja que aquesta comprovació es pot fer simplement mitjançant l'execució de l'operació insert, delegant la responsabilitat al propi gestor de bases de dades, i capturant/gestionant la SQLException com correspongui.
- 2 - Utilització inadequada de Statement. Tant st2 com st3 utilitzen Statement, però aquestes operacions s'executen per a cada valor dniProf obtingut a la consulta st1. Per tant, el més adequat seria utilitzar PreparedStatement establint com a variable de les operacions el valor dniProf. En el cas de st2 , i en relació al criteri de qualitat 1, aquest criteri es solucionaria també simplement suprimint aquesta comprovació.
- 3 – Execució de sentència SQL innecessària. Es podria també implementar el programa amb una única sentència SQL feta com a un Statement que faci un insert amb una subconsulta que determina els professors que s'ha d'inserir. En cas que algun dels professors ja està assignat, es genera el missatge quan es captura l'excepció afegida segons el criteri de qualitat 1.

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

```
cantants (nom, pais, anyRetir)
albums (titolAlbum, nomCantant, anyPublicacio, preu)
         {nomCantant} referencia cantants
critiques (titolAlbum, nomCantant, font, puntuacio)
         {titolAlbum, nomCantant} referencia albums
estadístiques (nomCantant, millorPuntuació)
         {nomCantant} referencia cantants
```

Indiqueu quins són els triggers necessaris (tenint en compte els criteris de qualitat establerts a l'assignatura) a definir per tal que:

- **RI1.** Es generi una excepció en cas que un esdeveniment faci que no es compleixi la següent restricció: Un cantant no pot publicar nous àlbums en els anys posteriors a l'any de retir.

- **RI2.** Es mantingui el valor de l'atribut *millorPuntuació*. Aquest atribut ha de tenir com a valor, la puntuació més alta que ha rebut un cantant en els àlbums que ha publicat.
- **RI3.** Es generi una excepció en cas que un esdeveniment faci que no es compleixi la següent restricció: No es poden afegir crítiques si la base de dades conté menys de 10 cantants.

Per cada trigger cal indicar:

- Taula per a la que es defineix
- Esdeveniment que l'activa (INSERT, DELETE, UPDATE - en cas d'UPDATE indicar atribut/s rellevants)
- BEFORE/AFTER (justificar la resposta)
- ROW/STATEMENT (justificar la resposta)

SOLUCIÓ

	Trigger RI1	Trigger RI1	Trigger RI2	Trigger RI3
Taula	cantants	albums	critiques	critiques
Esdeveniments	UPDATE any_retir	INSERT UPDATE anyPublicacio	INSERT, DELETE UPDATE puntuacio	INSERT
before/after	BEFORE (1)	BEFORE (1)	AFTER (3)	BEFORE (1)
row/statement	ROW (2)	ROW (2)	ROW (4)	STATEMENT (5)

(1) Si fossin AFTER, en cas de violar-se la restricció caldria desfer l'esdeveniment sobre la taula cantants/albums/critiques
(2) Al ser ROW poden implementar-se de manera incremental, fent les comprovacions de la restricció només per a les files inserides/modificades
(3) Si fossin BEFORE, en cas que els esdeveniments no s'acabessin executant caldria desfer els canvis de l'atribut *millorPuntuació* fets.
(4) Al ser ROW pot implementar-se de manera incremental, només aplicant canvis a l'atribut millorPuntuació dels cantants als que se'ls ha fet canvis en les crítiques.
(5) La restricció no té a veure amb quines files s'insereixen ni amb quantes files s'insereixen, per tant amb un STATEMENT n'hi ha prou.

3. **(2 punts)** Suposeu la base de dades següent, que s'ha creat en un SGBD sense cap mecanisme de control de concurrència. Considereu que el gránul és la fila.

```
autors(   codiAutor,  nom,        telefon,   poblacio)
          ca1         Autor1      111        Pobl1
          ca2         Autor2      222        Pobl2

obres(    idObra,     codiAutor,  titol,     anyEdicio)
          ob1         ca1         Bon dia    2020
          ob2         ca2         Hola       2020
          ob3         ca1         Bona nit   2020
```

Suposeu també les sentències SQL següents:

s1	UPDATE obres SET codiAutor = 'ca1' WHERE idObra= 'ob2';
s2	DELETE FROM obres WHERE idObra='ob1';
s3	INSERT INTO obres VALUES('ob4','ca1','Adeu',2020);
s4	SELECT * FROM obres WHERE codiAutor='ca1';
sX és un codi que us pot permetre fer referència a les sentències	

3.1 Doneu un horari on intervinguin dues transaccions, que poden executar una o més de les sentències SQL donades, i que presenti una interferència d'ACTULITZACIÓ PERDUDA. Una sentència es pot usar únicament una vegada en l'horari, i s'ha d'incloure a l'horari el mínim número de sentències. En cas que no sigui possible formar un horari amb els requisits indicats, justifiqueu perquè no.

3.2 Doneu un horari on intervinguin dues transaccions, que poden executar una o més de les sentències SQL donades, i que presenti una interferència de FANTASMA. S'ha d'incloure a l'horari el mínim número de sentències, però una sentència es pot usar més d'una vegada. En cas que no sigui possible formar un horari amb els requisits indicats, justifiqueu perquè no.

SOLUCIO

3.1) Si volem actualització perduda, les dues transaccions han de fer una actualització (RU,W) sobre el mateix gränul. Les tres operacions d'actualització proposades treballen sobre gränuls diferents així que NO és possible construir un horari amb aquesta interferència.

3.2) Si volem que es produeixi una interferència de tipus FANTASMA, una de les dues transaccions ha de 'canviar' el contingut de la BD. Per exemple, una transacció ha de fer l'UPDATE o l'INSERT (amb una instrucció n'hi ha prou, hem de fer servir el mínim nombre d'instruccions) i l'altra transacció ha de fer dues lectures semblants però forçant que donin resultats diferents. Per tant, ficarem l'UPDATE/INSERT en mig de les dues lectures:

T1	T2
SELECT * FROM obres WHERE codiAutor='ca1';	
	INSERT INTO obres values('ob4','ca1','Adeu',2020,'g1'); -- o bé UPDATE obres SET codiAutor = 'ca1' WHERE idObra= 'ob2';
SELECT * FROM obres WHERE codiAutor='ca1';	
	commit
commit	

4. **(2 punts)** Considereu l'horari següent:

	T1	T2
1	R(A)	
2		RU(A)
3		W(A)
4		RU(B)
5		W(B)
6	RU(B)	
7	W(B)	
8	COMMIT	
9		COMMIT

Suposeu que s'incorporen tècniques de control de concurrència amb reserves.

4.1 En cas que les transaccions treballin en nivell d'aïllament READ UNCOMMITTED.

a) Doneu l'horari aplicant reserves corresponents al nivell d'aïllament (*)

b) Doneu el graf de precedències de l'horari resultant d'aplicar reserves

c) En cas que hi hagi alguna interferència en l'horari resultant digueu quina/es són, i el/s gràNul/s implicat/s.

4.2 En cas que les transaccions treballin en nivell d'aïllament REPETEABLE READ.

a) Doneu l'horari aplicant reserves corresponents al nivell d'aïllament (*)

b) Doneu el graf de precedències de l'horari resultant d'aplicar reserves

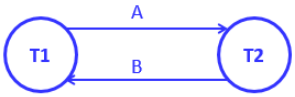
c) En cas que hi hagi alguna interferència en l'horari resultant digueu quina/es són, i el/s gràNul/s implicat/s.

(*) En els horaris heu 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

4.1)

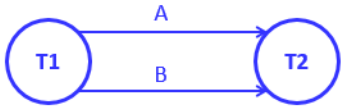
	T1	T2
1	R(A)	
2		LOCK(A,X)
3		RU(A)
4		W(A)
5		LOCK(B,X)
6		RU(B)
7		W(B)
8	LOCK(B,X)	
9	...	COMMIT+UNLOCKS (A,B)
10	RU(B)	
11	W(B)	
12	COMMIT+UNLOCK (B)	



Hi ha una interferència d'anàlisi inconsistent amb gràNuls A,B

4.2)

	T1	T2
1	LOCK(A,S)	
2	R(A)	
3		LOCK(A,X)
4	LOCK(B,X)	...
5	RU(B)	...
6	W(B)	...
7	COMMIT+UNLOCKS (A,B)	...
8		RU(A)
9		W(A)
10		LOCK(B,X)
11		RU(B)
12		W(B)
13		COMMIT+UNLOCKS (A,B)



No hi ha cap interferència

5. **(2 punts)** Suposeu la taula *albums*:

```
albums (titolAlbum, nomCantant, anyPublicacio, preu)
```

Aquesta taula té 2.000.000 files, i el factor de bloqueig, de les pàgines de dades on s'emmagatzemen aquestes files, és de 40 files per pàgina.

Sabem que el cantant `'Juli Catedrals'` ha fet 500 àlbums. També sabem que la condició `titolAlbum > 'xxx'` la compleixen 8.000 àlbums que han fet uns 1.000 cantants diferents.

Considereu la consulta següent.

```
SELECT *
FROM albums
WHERE nomcantant = 'Juli Catedrals'
     AND titolAlbum > 'xxx';
```

Doneu el cost en número de pàgines d'índex i número de pàgines de dades a les que cal accedir per resoldre la consulta:

5.1 Usant un índex agrupat definit sobre l'atribut nomCantant, amb ordre d=24, ple al 80%.

5.2 Usant un índex no agrupat definit sobre l'atribut titolAlbum, amb ordre d=16, ple al 70%

SOLUCIÓ:

5.1) Número pàgines d'índex + Número pàgines de dades = 4 + 13 = 17 pàgines

Número de pàgines d'índex = nivells de l'arbre = $\lceil \log_{40} 2.000.000 \rceil = 4$

número de valors per node = $\lceil 2d*0,8 \rceil = \lceil 2*24*0,8 \rceil = 39$ valors / node

número d'apuntadors per node = 39 + 1 = 40 apuntadors / node

Número de pàgines de dades = $\lceil$ (albums(nomcantant = 'Juli Catedrals') / factor de bloqueig $\rceil$

$= \lceil 500 / 40 \rceil = 13$

albums(nomcantant = 'Juli Catedrals') = 500

factor de bloqueig = 40

5.2) Número pàgines d'índex + Número pàgines de dades = 352 + 8.000 = 8.352 pàgines

Número de pàgines d'índex = nivells de l'arbre

$+ \lceil$ (albums(titolAlbum > 'xxx') / num valors/node $\rceil$ -1

$= \lceil \log_{24} 2.000.000 \rceil + \lceil 8.000 / 23 \rceil - 1 = 5 + 347 = 352$ pàgines

número de valors per node = $\lceil 2d*0,7 \rceil = \lceil 2*16*0,7 \rceil = 23$ valors / node

número d'apuntadors per node = 23 + 1 = 24 apuntadors / node

Número de pàgines de dades = albums(titolAlbum > 'xxx') = 8.000

6. **(1 punt)** Considereu un índex arbre B+ amb els següents paràmetres: el valor pel que es crea l'índex té V bytes, la mida dels nodes és B bytes, els RID tenen T bytes, i els apuntadors a nodes de l'índex tenen A bytes.

6.1 Sigui nf la quantitat màxima de valors que un node del nivell fulla pot contenir. Expresseu nf en funció dels paràmetres V, B, T, A

6.2 Suposa ara que 2d = 100.

a) Quin és el número màxim de files que el arbre B+ pot apuntar si té 3 nivells?

b) Quants nodes es necessiten per emmagatzemar l'índex en aquest cas?

SOLUCIÓ:

6.1) nf = $\lfloor (B-2A) / (V + T) \rfloor$

6.2.a) files = 101*101*100 = 1,020,100

6.2.b) nodes = 1+ 101 + 101*101 =10303

BD-GEI **19.4.2021**

Temps: 90 minuts

Notes 6 maig **Cada pregunta s'ha de lliurar en un full separat**

Suposeu una base de dades amb les taules següents:

```
create table departaments(
codiDept integer,
nomDept char(50) unique not null,
pressupost integer not null check(pressupost>=0),
primary key (codiDept));

create table professors(
idProf integer,
nomProf char(50) not null,
codiDept integer not null,
sou real,
primary key (idProf),
foreign key (codiDept) references departaments);
-- codiDept és del departament en el que treballa el professor

create table assignatures(
idAssig char(5),
nomAssig char(50) not null,
codiDept integer not null,
profResponsable integer not null,
primary key (idAssig),
foreign key (codiDept) references departaments,
foreign key (profResponsable) references professors);
-- codiDept és del departament al que està assignada l'assignatura
-- profResponsable és el professor responsable de l'assignatura

create table assignacions(
idProf integer,
idAssig char(5),
quadrimestre char(6),
horesAssig integer not null check (horesAssig>0),
primary key (idProf, idAssig, quadrimestre),
foreign key (idProf) references professors,
foreign key (idAssig) references assignatures);
-- hi ha una fila si el professor idProf ha donat o dóna
-- classes de l'assignatura idAssig en el quadrimestre
```

1. (1.75 punts) Considereu la base de dades donada a l'inici de l'examen. Doneu una sentència SQL per obtenir les assignatures (nomAssig) per a les que no hi ha cap assignació d'un professor de més de 3 hores (horesAssig) en el quadrimestre '2020-2'.

```
SELECT DISTINCT a.nomAssig
FROM assignatures a
WHERE NOT EXISTS (SELECT *
                  FROM assignacions ass
                  WHERE ass.idAssig = a.idAssig AND
                        ass.quadrimestre = '2020-2' AND
                        ass.horesAssig > 3);
```

2. (1.75 punt) Considereu la base de dades donada a l'inici de l'examen. Doneu una seqüència d'operacions d'àlgebra relacional per obtenir l'identificador i el nom dels professors que han estat o estan assignats amb menys de 5 hores de docència (horesAssig) a assignatures d'un departament que no és el seu.

```
A = assignacions(hores<5)
B = A * professors
C = assignatures{codiDept -> cd, idAssig → ia}
D = B[idAssig = ia, codiDept <> cd]C
E = D[idProf,nomProf]
```

3. (3 punts) Considereu la base de dades donada a l'inici de l'examen.

3.1 Per cadascuna de les restriccions següents:

- Si es pot implementar com a restricció en la definició de les taules, doneu el codi SQL que cal afegir als CREATE TABLE de la BD per a implementar-la, i en quina taula s'ha d'afegir.
- Si no es pot, explicar per què.

a) Un professor no pot ser responsable de més d'una assignatura.

Afegim un UNIQUE(profResponsable) a la taula assignatures.

b) No s'ha de poder executar sentències UPDATE de l'atribut pressupost de la taula departaments.

No es pot implementar com a restricció de taula, caldria fer un grant

c) Un professor ha de tenir sou positiu, a no ser que estigui de baixa, cosa que s'indicarà quan l'atribut sou tingui valor NULL.

Afegim la condició CHECK (sou > 0 OR sou IS NULL) a la taula professors

3.2 Donar una sentència de creació d'una asserció per tal que a la base de dades es compleixi que: "Tots els professors assignats a una assignatura en el mateix quadrimestre, han de ser del mateix departament."

```
CREATE ASSERTION totsMateixDept
   CHECK (NOT EXISTS (SELECT a.idAssig, a.quadrimestre
           FROM assignacions a NATURAL JOIN professors p
           GROUP BY a.idAssig, a.quadrimestre
           HAVING COUNT (DISTINCT p.codiDept)>1))
```

```
CREATE ASSERTION totsMateixDept
   CHECK (NOT EXISTS (SELECT *
           FROM assignacions a1 NATURAL JOIN professors p1,
             assignacions a2 NATURAL JOIN professors p2
           WHERE a1.idAssig = a2.idAssig AND
                 a1.quadrimestre = a2.quadrimestre AND
                 p1.codiDept <> p2.codiDept))
```

4. (2 punts)

4.1 Considereu la vista següent definida sobre la base de dades donada a l'inici de l'examen.

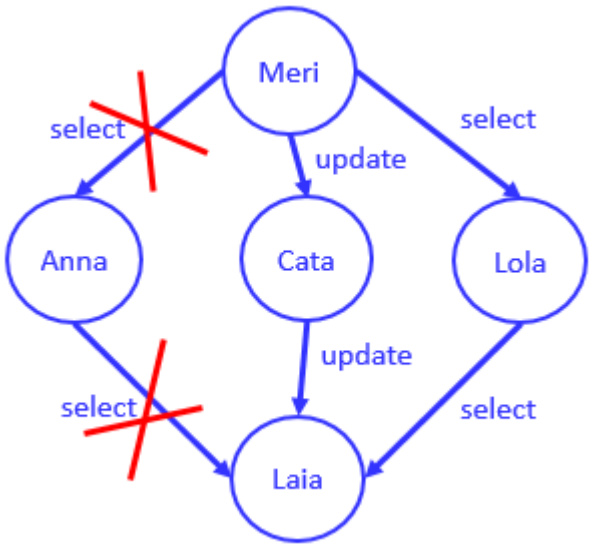
- En cas que la vista sigui actualitzable, doneu una sentència INSERT/DELETE/UPDATE que doni una excepció per al check de la vista.
- En cas que la vista no sigui actualitzable, expliqueu el motiu pel qual no ho és.

```
create view professorsDept5 as
   select p.idProf, p.nomProf
   from professors p
   where p.codiDept = 5
   with check option;
```

La vista no és actualitzable perquè, encara que està definida sobre una única taula, no inclou tots els atributs NOT NULL de la taula que no tenen valor per defecte. En concret, perquè no inclou l'atribut codiDept.

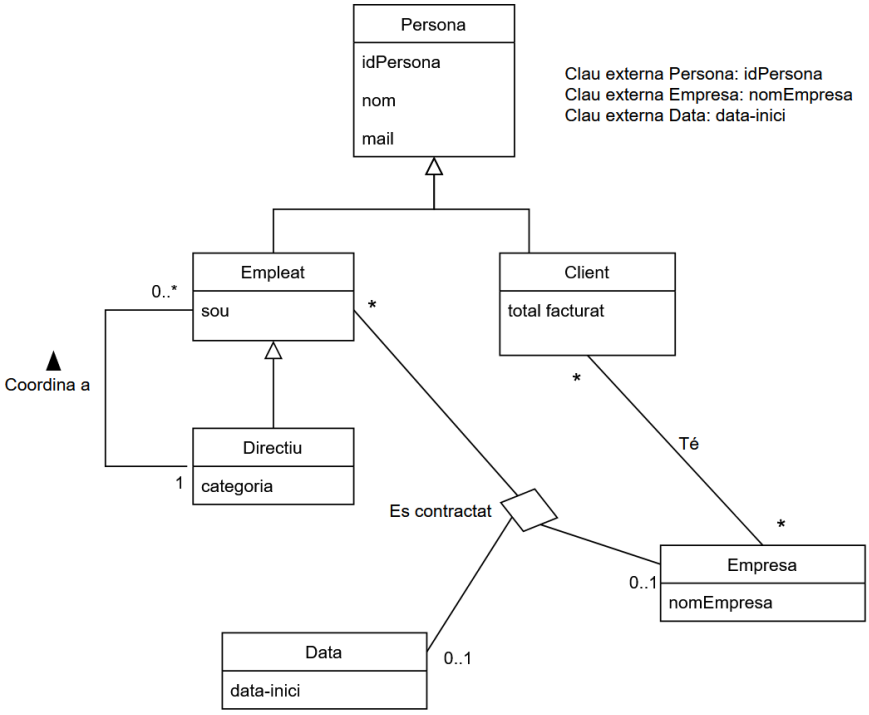
4.2 Suposem que les taules de la base de dades són propietat de la Meri. Considereu les sentències GRANT i REVOKE següents. Doneu els diagrames d'autoritzacions corresponents a l'instant abans d'executar les sentències REVOKE, i a l'instant després d'executar les sentències REVOKE. Justifica el diagrama d'autoritzacions de després dels REVOKE.

```
Meri: GRANT select ON professors TO Anna WITH GRANT OPTION;
Meri: GRANT update ON professors TO Cata WITH GRANT OPTION;
Meri: GRANT select ON professors TO Lola WITH GRANT OPTION;
Anna: GRANT select ON professors TO Laia;
Cata: GRANT update ON professors TO Laia;
Lola: GRANT select ON professors TO Laia;
Meri: REVOKE select ON professors FROM Anna RESTRICT;
Meri: REVOKE update ON professors FROM Cata RESTRICT;
```



El REVOKE de la Meri a l'Anna s'executa, perquè ningú més, a part de l'Anna perd cap privilegi. En canvi, el REVOKE de la Meri a la Cata no s'executa amb èxit, perquè a part de la Cata, la Laia perdria el privilegi.

5. (1.5 punts) 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 foranes i restriccions NOT NULL i UNIQUE que siguin necessàries.



Persona(idPersona,nom,mail)

Empleat(idPersona, sou, directiu)

{idPersona} referencia Persona

{directiu} referencia Directiu NOT NULL

Directiu(idPersona,categoria)

{idPersona} referencia Empleat

Client(idPersona, totalfacturat)

{idPersona} referencia Persona

Empresa(nomEmpresa)

Data(data-inici)

Te(idPersona,nomEmpresa)

{idPersona} referencia Client

{nomEmpresa} referencia Empresa

EsContratctat(idPersona,nomEmpresa,data-inici)

[data-inici} referencia Data NOT NULL

UNIQUE (idPersona,data-inici)

{idPersona} referencia Empleat

{nomEmpresa} referencia Empresa

Temps: 2,5 hores

Notes 22 juny Revisió: 28 juny **Cada pregunta s'ha de lliurar en un full separat**

1. **(1 punt)** Donades les base de dades següent:

```
create table professors
(dni char(50),
nomProf char(50) unique,
telefon char(15),
primary key (dni));

create table despatxos
(modul char(5),
numero char(5),
superficie integer not null check(superficie <=25),
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 (instantInici < instantFi));
```

Considereu el següent fragment de codi Java/JDBC. Implementeu el cos del **while** per tal que s'incrementi en 5 unitats la superfície dels despatxos de cadascun dels mòduls obtinguts en la variable varModul. Definiu també els Statements o PreparedStatements necessaris on creieu convenient. Cal que el programa tregui un missatge d'excepció si la superfície d'algun despatx passa a ser superior a 25. En cas que no hi hagi cap excepció el programa indicarà el número de despatxos modificats de cada mòdul.

```
try {
 /* Tenim ja una connexió c establerta amb la base de dades */
 /* Tenim un variable in que permet llegir el que escriu l'usuari */

 System.out.print("Escriu el nom d'un mòdul: ");
 String varModul = in.nextLine();

 while (!varModul.equals("acabar")) {

     //codi a implementar

     System.out.print("Escriu el nom d'un mòdul: ");
     varModul = in.nextLine();
     }
 c.commit();
 }
catch (ClassNotFoundException ce) {
        System.out.println ("Error al carregar el driver");}
catch (SQLException se) {
        System.out.println ("Excepcio: "+ se.getMessage());}
```

Recordeu que podeu usar els mètodes/codis d'excepció de JDBC estudiats a classe:

Statements

- Statement createStatement();
- ResultSet executeQuery(String sql);
- int executeUpdate(String sql);

PreparedStatement

- PreparedStatement prepareStatement(String sql);
- ResultSet executeQuery();
- int executeUpdate();
- void setXXX(int posicioParametre, XXX valor);

ResultSet

- boolean next();
- XXX getXXX(String nomColumna)

SQLException

- // 23502 -not_null_violation, 23503 -foreign_key_violation
- // 23505 -unique_violation, 23514 -check_violation
- String getSQLState();

SOLUCIÓ

```
try {
    /* Tenim ja una connexió c establerta amb la base de dades */

    System.out.println("Escriu el nom d'un mòdul: ");
    String varModul = System.console().readLine();

    PreparedStatement ps = c.prepareStatement("update despatxos d "+
                                    "set superficie = superficie + 5 "+
                                    "where d.modul = ?");


    while (!varModul.equals("acabar")) {

        //codi a implementar
        ps.setString(1,varModul);
        int numFilesActualitzades = ps.executeUpdate();
        System.out.println (varModul+"  "+numFilesActualitzades);
        System.out.println("Escriu el nom d'un mòdul: ");
        varModul = System.console().readLine();
        }
    c.commit();
    }
   catch (ClassNotFoundException ce) {
             System.out.println ("Error al carregar el driver");}
   catch (SQLException se) {
            if(se.getSQLState().equals("23514"))
                  System.out.println("La superficie no pot ser més gran que 25");
            else
                  System.out.println ("Excepcio: "+ se.getMessage());}
```

2. (1 punt) Suposeu la base de dades següent:

```
empleats1 (numEmp1, nomEmp1, ciutatEmp1);
empleats2 (numEmp2, nomEmp2, ciutatEmp2);
```

Suposeu que es vol implementar la restricció següent mitjançant disparadors:

Els valors de l'atribut `ciutatEmp1` de la taula `empleats1` han d'estar inclosos en els valors de l'atribut `ciutatEmp2` de la taula `empleats2`

La idea és llançar una excepció en cas que s'intenti executar una sentència que violi la restricció.

2.1 Digueu quins són els esdeveniments rellevants (taula/es i esdeveniment/s).

empleats1	Insert	
empleats1	Update	ciutatEmp1
empleats2	Delete	
empleats2	Update	ciutatEmpl2

2.2 Digueu i justifiqueu de quin tipus han de ser el disparadors (ROW / STATEMENT, BEFORE/AFTER).

Tots els disparadors han de ser before, for each row.

Before perquè com que es tracta de que salti una excepció quan es violi una restricció d'integritat, com més aviat es descarti que la restricció es viola millor, per tal d'evitar fer feina innecessària.

For each row perquè és millor una solució incremental. Suposant que a cada taula hi ha una gran quantitat d'empleats, i que els esdeveniments afecten a poques files, una solució incremental sempre comportarà menys accessos a la base de dades, perquè només requereix fer comprovacions per a les tuples que s'han inserit/esborrat/modificat.

2.3 Implementeu un dels disparadors que hagueu identificat per a la taula empleats1. En cas de que no n'hi hagi cap justifiqueu perquè. Podeu fer servir la plantilla següent.

```
CREATE FUNCTION comprovarCiutat() RETURNS trigger AS $$
    BEGIN
     //implementació
    END;
$$LANGUAGE plpgsql;

CREATE TRIGGER ex3_3
    BEFORE/AFTER <esdeveniment>
    FOR EACH ROW/STATEMENT EXECUTE PROCEDURE comprovarCiutat();
```

```
CREATE FUNCTION comprovarCiutEmpl1() RETURNS trigger AS $$
    BEGIN
     IF  not exists(select* from empleats2
         where  ciutatEmp2=new. ciutatEmp1) THEN
         RAISE EXCEPTION 'Error en inserir o modificar empleats1';
     END IF;
     RETURN NEW;
    END;
$$LANGUAGE plpgsql;

CREATE TRIGGER ex3_3
    BEFORE INSERT OR UPDATE of ciutatEmp1 ON empleats1
    FOR EACH ROW EXECUTE PROCEDURE comprovarCiutEmpl1();
```

3. **(2 punt)** Considereu la base de dades de l'exercici 1 amb el contingut següent:

Professors	dni	nomProf	telefon	sou
	111	Joana	97743121	10000
	222	Martí	93818903	3000
	333	Lourdes	97261201	50000

Despatxos	modul	numero	superficie
	C6	101	13
	Omega	300	8
	Omega	301	8

Assignacions	dni	modul	numero	instantIni	instantFi
	333	C6	101	5	9
	111	C6	101	3	NULL
	222	Omega	300	10	NULL

Considereu també el procediment emmagatzemat següent:

```
CREATE TYPE despatx AS (d_modul char(5), d_numero char(5));

CREATE FUNCTION assignaProfessor(dni_p char(50), modul_d char(5),
                                 numero_d char(5), inici_a integer)
            RETURNS SETOF despatx AS $$
DECLARE desp despatx;
BEGIN
  IF (NOT EXISTS(SELECT * FROM Professors WHERE dni=dni_p)
       OR NOT EXISTS(SELECT * FROM Despatxos
                     WHERE modul=modul_d AND numero=numero_d)) THEN
       RAISE EXCEPTION 'El professor o el despatx no existeix';
     ELSIF (EXISTS (SELECT * FROM Assignacions a
                    WHERE a.dni=dni_p AND a.instantFi IS NULL)) THEN
          UPDATE Assignacions SET instantFi = inici_a - 1
          WHERE dni=dni_p AND instantFi IS NULL;
   END IF;
   INSERT INTO Assignacions VALUES (dni_p,modul_d,numero_d,inici_a,NULL);
   FOR desp IN SELECT d.modul,d.numero
                  FROM Professors p NATURAL JOIN Assignacions a
                                    NATURAL JOIN Despatxos d
               WHERE a.instantFi IS NULL
               GROUP BY d.modul, d.numero
               HAVING COUNT(*) >= 2
    LOOP
      RETURN NEXT desp;
    END LOOP;
   EXCEPTION
       WHEN raise_exception THEN raise exception '%',sqlerrm;
 END;
 $$LANGUAGE plpgsql;
```

3.1 Indiqueu i justifiqueu quin és l'estat de la base de dades i el valor de retorn després de l'execució de la consulta: `SELECT * FROM assignaProfessor('111','Omega','300',15);`

Les taules Professors i Despatxos no es modifiquen, el contingut de la taula Assignacions és:

Dni	modul	numero	instantIni	instantFi
333	C6	101	5	9
111	C6	101	3	14
222	Omega	300	10	NULL
111	Omega	300	15	NULL

El procediment ha finalitzat totes les assignacions actives del professor amb dni_p (posant com a instant fi el d'inici menys 1), i n'ha creat una de nova per al despatx donat com a paràmetre. L'execució de la consulta retorna:

d_modul	d_numero
Omega	300

els quals corresponen als despatxos compartits (més d'un professor assignat) amb assignacions actives (instantFi amb valor NULL).

3.2 Digueu quins són els criteris de qualitat que no es compleixen en la implementació del procediment donat i com caldria canviar el codi per tal que es complíssin.

- Accessos innecessari a la BD.
  - El primer IF amb els seus SELECTs no cal. Per tant els SELECTs són innecessaris perquè l'excepció es pot capturar mitjançant la violació de la condició de foreign key que saltarà en fer el INSERT de l'assignació nova.
  - El ELEIF amb els seu SELECT no calen. El UPDATE ja només modifica les assignacions amb instantFi NULL si és que n'hi ha alguna.
- TAULES+JOINs innecessàries
  - En el SELECT que serveix per buscar el resultat que ha de donar el procediment, les taules Professors i Desspatxos no són necessàries.

4. **(3 punts)** Transaccions:

4.1 Suposeu ara l'horari següent.

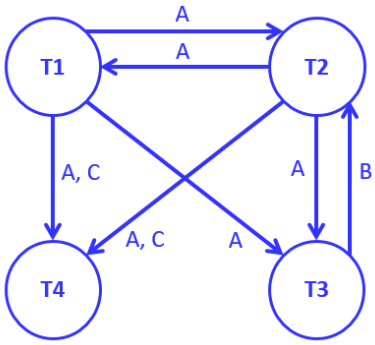
a) Doneu el graf de precedències per aquest horari.

b) En cas que hi hagi alguna interferència en l'horari resultant digueu quina/es són, les transaccions i el/s gràNul/s implicat/s.

c) És serialitzable? En cas afirmatiu doneu el horari serial equivalent. En cas negatiu perquè no.

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

a) Graf de precedències.



b) Hi ha una actualització perduda entre T1 i T2 i un anàlisi inconsistent entre T2 i T3
c) No és serialitzable perquè hi ha interferències.

4.2 En cas que les transaccions treballin en nivell d'aïllament READ COMMITTED. Doneu l'horari un cop aplicades les reserves corresponents al nivell d'aïllament. En l'horari heu 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).

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

4.3 Considereu ara un SGBD sense cap mecanisme de control de concurrència, on el gràNul és la fila. I considereu l'horari següent que té una interferència d'anàlisi inconsistent. Expliqueu la interferència d'anàlisi inconsistent en base a l'horari i al contingut de la base de dades donat en l'exercici 3.

	T1	T2
1	select * from despatxos where modul='Omega' and numero=300;	
2		update despatxos set superficie = superficie + 8 where modul='Omega' and numero=300;
3		select * from despatxos where modul='Omega' and numero=301;
4	update despatxos set superficie = superficie + 5 where modul='Omega' and numero=301;	
5	Commit	
6		Commit

La interferència d'anàlisi inconsistent té a veure amb la visió que dues transaccionstenen d'un conjunt de dades. En aquest horari el conjunt de dades són les files 'Omega', 300 i 'Omega', 301.

En aquest horari:

- T1 llegeix les files amb valors 'Omega', 300, 8 i 'Omega', 301, 8.
- T2 llegeix les files amb valors 'Omega', 300, 8 i 'Omega', 301, 8

L'horari no és correcte perquè no dóna el mateix resultat que cap dels dos horaris serial.

En cas de l'horari seria T1;T2:

- T1 llegeix les files amb valors 'Omega', 300, 8 i 'Omega', 301, 8
- T2 llegeix les files amb valors 'Omega', 300, 8 i 'Omega', 301, 13

En cas de l'horari seria T2;T1:

- T2 llegeix les files amb valors 'Omega', 300, 8 i 'Omega', 301, 8
- T1 llegeix les files amb valors 'Omega', 300, 16 i 'Omega', 301, 8

5. **(3 punt)** Considereu la taula R(a,b,c,d). Aquesta taula té 100.000 files, i el factor de bloqueig, de les pàgines de dades on s'emmagatzemen aquestes files, és de 10 files per pàgina.

Considereu la consulta següent.

```
SELECT *
FROM R
WHERE b=5 and c >=50
```

Se sap que hi ha 70 files que compleixen la condició b=5, 100 files la condició c>=50 i només 1 fila les dues condicions alhora.

IMPORTANT: En tots els apartats cal detallar tots els càlculs que feu per calcular els costos.

5.1 Suposant que <u>només podem definir un únic índex, arbre B+, per un únic atribut</u>, i que aquest índex tindrà una ocupació del 80% i un ordre d=50, Quin tipus d´índex (agrupat o no agrupat) escolliríeu i per quin atribut hauria d'estar definit per tal de fer la consulta el més eficient possible? Justifiqueu les respostes en base els costos de totes les opcions possibles.

SOLUCIÓ:

Les opcions possibles són definir un únic índex agrupat o no agrupat pels atributs b o c, i aplicar la resta de condicions a mida que es van obtenint les files de les pàgines de dades. Qualsevol altre índex no augmentaria l'eficiència de la consulta.

Índex agrupat R.b: Cost = h +D = 3 +7 = 10

Valors per node = $\lceil 2d*0,8 \rceil = \lceil 2*50*0,8 \rceil$ = 80 valors / node

h = $\lceil \log_{81} 100000 \rceil$ = 3

D= $\lceil$ card(R.b=5) / 10 $\rceil = \lceil 70 / 10 \rceil = 7$

Índex no agrupat R.b: Cost = h + F + card(R.b=5) = 3 +0 + 70 = 73

h = $\lceil \log_{81} 100000 \rceil$ = 3

F= $\lceil 70 / 80 \rceil$ - 1= 0

card(R.b=5) = 70

Índex agrupat R.c: Cost = h +D = 3 +10 = 13

Valors per node = $\lceil 2d*0,8 \rceil = \lceil 2*50*0,8 \rceil$ = 80 valors / node

h = $\lceil \log_{81} 100000 \rceil$ = 3

D= $\lceil$ card(R.c>=50) / 10 $\rceil = \lceil 100 / 10 \rceil = 10$

Índex no agrupat R.c: Cost = h + F + card(R.c > 50) = 3 +1 + 100 = 104

h = $\lceil \log_{81} 100000 \rceil$ = 3

F= $\lceil 100/ 80 \rceil$ - 1= 1

card(R.c > =50) = 100

La millor opció serà definir un índex agrupat per l'atribut R.b, amb un cost de 10 accessos a disc.

5.2 Suposant ara que en lloc d'un únic índex per un únic atribut, en podem definir 2 també per un únic atribut i del mateix tipus que abans (una ocupació del 80% i un ordre d=50) quin seria el cost fent servir l'estratègia de intersecció de RIDs?

Cost índex per R.b = h + F = 3

Cost índex per R.c = h + F = 4

Cost per acedir a dades = Card(R.b=5 and R.c >50) = 1

El cost seria 8 accesos a disc

5.3 Suposant ara que podem definit un índex no agrupat multiatribut pels atributs b, c. Quin seria el cost fent servir aquest índex en cas de tenir ordre d=25 i ocupació 80%?

el cost seria: h +F +Card(R.b=5 and R.c >=50) = 4 + 0 +1 = 5

Valors per node = $\lceil 2d*0,8 \rceil = \lceil 2*25*0,8 \rceil$ = 40 valors / node

h = $\lceil \log_{41} 100000 \rceil$ = 4

F = F= $\lceil 1/ 80 \rceil$ - 1= 0

Temps: 90 minuts

Notes 25 novembre

Cada pregunta s'ha de lliurar en un full separat

Suposeu una base de dades amb les taules següents:

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

CREATE TABLE Partits(
idPartit integer,
nomEquipLocal char(30) NOT NULL,
dataPartit date NOT NULL,
nomEquipVisitant char(30) NOT NULL,
golsEquipL integer NOT NULL CHECK(golsEquipL >=0),
golsEquipV integer NOT NULL CHECK(golsEquipV >=0),
PRIMARY KEY (idPartit),
FOREIGN KEY (nomEquipLocal) REFERENCES EquipsFutbol,
FOREIGN KEY (nomEquipVisitant) REFERENCES EquipsFutbol,
UNIQUE(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 està contractat** el jugador

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

- el jugador dniJugador **ha estat alineat** al partit idenficat per idPartit
- els gols i targetes són les que han posat al jugador durant el partit.

1. **(2.5 punts)** Doneu una sentència SQL per obtenir els partits en els que ha guanyat un equip visitant, i en els que, a més, no s'ha alineat cap jugador a l'equip visitant que tingui un salari inferior a la mitjana dels salaris de tots els jugadors de la base de dades. Es vol que en el resultat hi hagi, la data del partit, el nom de l'equip local, el nom de l'equip visitant i l'estadi on es van jugar. Considereu que els partits es juguen a l'estadi de l'equip local.

```
select p.dataPartit, p.nomEquipLocal,
          p.nomEquipVisitant, e.nomEstadi
from partits p, equipsfutbol e
where p.golsEquipV > p.golsEquipL
     and p.nomEquipLocal = e.nomEquip
     and not exists (select *
               from alineacions a, jugadors j
               where a.dniJugador = j.dniJugador
                    and a.idPartit = p.idPartit
                    and j.nomEquip = p.nomEquipVisitant
                    and j.salari < (select avg(j2.salari)
                         from jugadors j2));
```

2. **(1.5 punt)** Doneu una asserció que no permeti que els equips de futbol tinguin un pressupost inferior a la suma dels salaris dels seus jugadors.

```
CREATE ASSERTION sostreSalarial CHECK (
NOT EXISTS (
     SELECT ef.nomEquip
     FROM EquipsFutbol ef NATURAL INNER JOIN Jugadors j
     GROUP BY ef.nomEquip
     HAVING ef.pressupost < sum(j.salari)))
```

3. **(1 punt)** Suposeu la vista `pichichi` que té per objectiu mostrar la classificació dels golejadors del campionat.

```
CREATE VIEW pichichi (dniJugador, totalGols) AS
        SELECT dniJugador, SUM(gols)
        FROM alineacions
        GROUP BY dniJugador;
```

Abans de l'inici del campionat es van insertar tots els equips i tots els jugadors de cada equip a les taules corresponents. Per registrar el primer partit es van fer els següents inserts:

```
INSERT INTO PARTITS VALUES (1,'FCB','2021/08/15','RSO',4,2);
INSERT INTO ALINEACIONS VALUES (1,'33333333C', 3,0);
INSERT INTO ALINEACIONS VALUES (1,'11111111A', 0,0);
INSERT INTO ALINEACIONS VALUES (1,'22222222B', 2,0);
```

a) Quin és el resultat d'executar la sentència següent, un cop executats els inserts:

```
SELECT * FROM pichichi ORDER BY totalGols DESC
```

b) És actualitzable la vista? Per què?

c) Modifiqueu la definició de la vista per tal que mostri només els jugadors que han marcat en total més de 2 gols.

a)

dniJugador	totalGols
'33333333C'	3
'22222222B'	2
'11111111A'	0

b) No és actualitzable perquè té funcions d'agregació.

c) Cal afegir a la definició el HAVING SUM(gols) > 2

4. (1 punt) Suposeu que la `Laia` és la propietària de la taula `jugadors`.

a) Doneu el diagrama d'autoritzacions que hi haurà després de l'execució de les sentències següents:

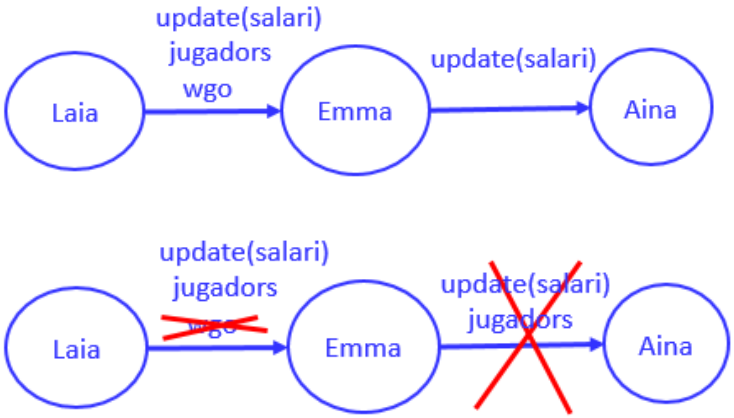
```
Laia: GRANT update(salari) ON jugadors TO Emma WITH GRANT OPTION
Emma: GRANT update(salari) ON jugadors TO Aina
```

b) Partint del diagrama d'autoritzacions resultant de l'apartat anterior, doneu el diagrama d'autoritzacions després de l'execució de la sentència següent:

```
Laia: REVOKE GRANT OPTION FOR update(salari) ON jugadors FROM Emma CASCADE
```

c) Indiqueu quins privilegis mínims li farien falta al `Josep` per tal de poder executar la sentència següent:

```
UPDATE jugadors SET salari= salari +100
WHERE NOT EXISTS (SELECT a.idPartit FROM alineacions a
                  WHERE a.dniJugador=jugadors.dniJugador
                        AND a.gols=0);
```



c) Hauria de tenir privilegis de:

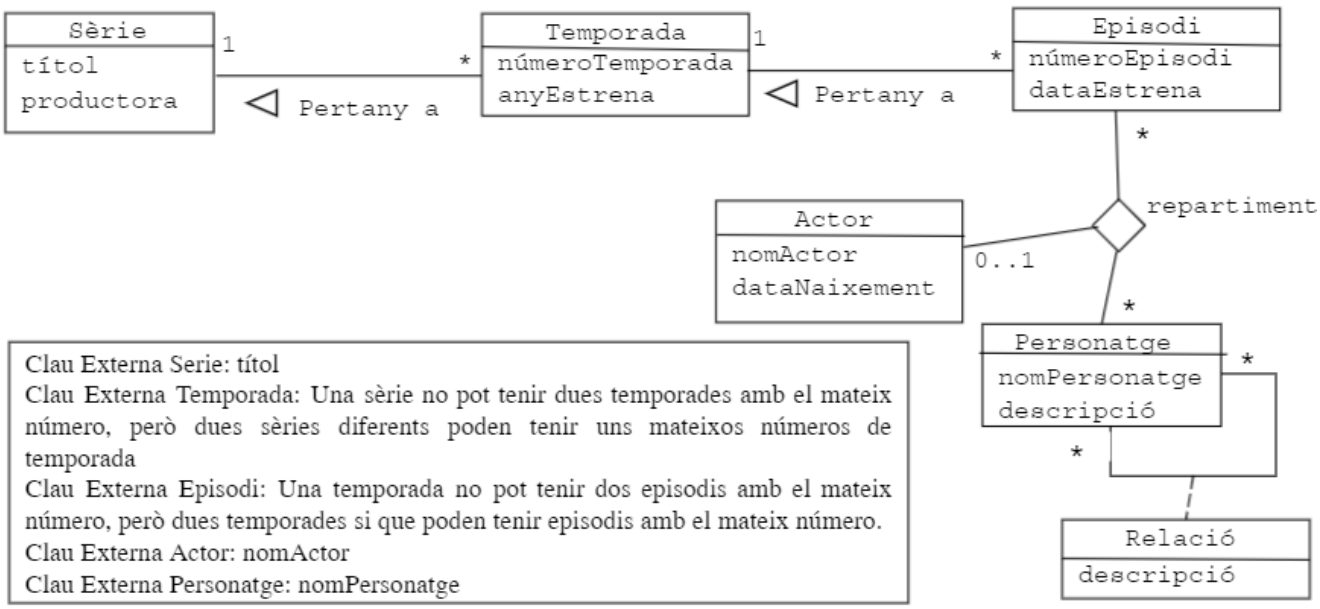
select(dniJugador, salari), update(salari) – jugadors

select(dniJugador, gols, idPartit) – alineacions

5. (1.5 punt) Doneu una seqüencia d'operacions en àlgebra relacional per obtenir els noms dels equips que han marcat fora de casa (han fet algun gol com a visitants) en un únic partit.

```
A=partits(golsEquipV>0)
B=A[nomEquipVisitant, idPartit]
C=B{nomEquipVisitant->nomEquipVisitant2, idPartit->idPartit2}
D=B[nomEquipVisitant=nomEquipVisitant2, idPartit<>idPartit2]C
E=D[nomEquipVisitant]
F=A[nomEquipVisitant]
R=F-E
```

6. (1.5 punts) 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 foràanies i restriccions NOT NULL i UNIQUE que siguin necessàries.



sèries(títol, productora)
temporades(títol, númeroTemporada, anyEstrena)
{títol} referencia sèries
episodis(títol, númeroTemporada, númeroEpisodi, dataEstrena)
{títol, númeroTemporada} referencia temporades
actors(nomActor, dataNaixement)
personatges(nomPersonatge, descripció)
repartiments(títol, númeroTemporada, númeroEpisodi, nomPersonatge, nomActor)
{títol, númeroTemporada, númeroEpisodi} referencia episodis
{nomPersonatge} referencia personatges
{nomActor} referencia actors NOT NULL
relacions(nomPersonatge, nomPersonatgeRelacionat, descripció)
{nomPersonatge} referencia personatges
{nomPersonatgeRelacionat} referencia personatges

7. (1 punt)

7.1 L'actualització incorrecta de dades redundants, una avaria en un disc on estan els fitxers d'una BD o transaccions que no poden acabar per un tall de subministrament elèctric són exemples de situacions que comprometen la fiabilitat d'un SGBD. Expliqueu breument un dels mecanismes que tingui un SGBD per garantir l'objectiu de fiabilitat.

7.2 Un altre objectiu fonamental dels SGBDs és que permetin a molts usuaris accedir concurrentment a una BD. Expliqueu breument algun problema que es pot produir com a conseqüència de l'accés simultani de molts usuaris a una BD (en cas de no disposar dels mecanismes que ofereixen els SGBD per evitar aquest problema).

Veure material del tema 1 de l'assignatura

1. (1 punt) Donades les taules següents.

```
assignatures(idAssignatura,nomAssignatura,credits)

estudiants(idEstudiant,nomEstudiant,username)

assignaturaMatriculada(idEstudiant,idAssignatura,quadrimestre,nota)
   {idEstudiant} referencia estudiants
   {idAssignatura} referencia assignatures
```

Considereu el següent fragment de codi Java/JDBC. El programa té dos parts. En la primera part es mostra en quantes assignatures han estat matriculats cadascun dels estudiants indicats en la variable *ids*. En la segona part s'esborra de la base de dades una assignatura identificada en la variable *id* si no hi ha cap estudiant que s'hi hagi matriculat; si a l'assignatura s'hi ha matriculat algun estudiant, aleshores es dóna un missatge d'error.

```
01  /* Tenim ja una connexió c establerta amb la base de dades */
02
03  /* consultar assignatures */
04  ResultSet rs = null;
05  int[] ids = {1, 2, 3};
06  PreparedStatement ps = c.prepareStatement("select distinct idAssignatura "+
07                                            "from assignaturaMatriculada "+
08                                            "where idEstudiant = ?");
09  for(int i=0; i<ids.length;i++){
10     ps.setInt(1,ids[i]);
11     rs = ps.executeQuery();
12     int quantesAssignatures=0;
13     while(rs.next()){quantesAssignatures = quantesAssignatures + 1;}
14     System.out.println(ids[i] + " ha estat matriculat en "
15                               + quantesAssignatures + " assignatures!");
16     }
17
18  /* esborrar assignatura */
19  int id = 4;
20  Statement stCons = c.createStatement();
21  Statement stDel = c.createStatement();
22  rs = stCons.executeQuery("select count(*) as quants "+
23                           "from assignaturaMatriculada "+
24                           "where idAssignatura = "+id);
25  if (rs.next() && rs.getInt("quants")>0)
26      {System.out.println("Error: no es pot esborrar assignatura "+
27                          "perquè hi ha estudiants matriculats a"+id);}
28    else {stDel.executeUpdate("delete from assignatures where idAssignatura ="+id);}
29
30 /* Commit i desconnexio de la base de dades */
```

Donat aquest fragment de codi, identifiqueu quins dels criteris de qualitat estudiats a l'assignatura no es compleixen. Per cadascun d'ells:

- Expliqueu perquè no es compleix
- Expliqueu en detall quins canvis seria necessari fer en el codi donat per tal de que es complexi.

Solució

- Primera part - Part del codi és redundant
  - No és necessari el WHILE que calcula la quantitat d'assignatures matriculades d'un estudiant. Cal canviar el SELECT per buscar directament aquest total en el mateix SELECT.
  - El nou SELECT, que substitueix l'existent en línies 05..07:

```
select count(distinct idAssignatura) as total
from students_marks where student_id=?
```

  - S'eliminarà el WHILE de la línia 12, i la línia 11 canviarà de la manera indicada a continuació:

```
rs.next();
int quantesAssignatures = rs.getInt("total");
```

- Segona part – Select innecessari
  - Es pot saber si hi ha l'error simplement executant el DELETE, i recullint en cas que es produeixi l'error de clau forana. Per tant, no es necessari fer el SELECT, i s'ha de recollir l'excepció amb un CATCH.
  - S'eliminarà les línies de codi 19, 21-23 i 24
  - El DELETE de la línia 27 es farà just després de crear el Statement de la línia 20

```
stDel.executeUpdate("delete from assignatures where idAssignatura ="+id);
```

  - Les línies 18, 19 i la nova línia amb el DELETE estaràn en try, i s'afegirà un catch que contidrà les línies 25 i 25 dins de un IF.

```
catch (SQLException se)
        { if (se.getSQLState().equals("23503")
          {System.out.print("Error: ....  ");}
```

2. (2.5 punts) Considereu la base de dades següent.

```
tipusProductes(idTipus, nomTipus, preuMaxim)

productes(nomProducte, idTipus, preu)
    {idTipus} references tipusProductes

comandes(numComanda, import)

productesComanda(numComanda, nomProducte, quantitat)
    {numComanda} references comandes
    {nomProducte} references productes
```

2.1 Suposeu que les taules de la base de dades estan inicialment buides i que executem una crida al següent procediment emmagatzemat *afegir_producte()*. Quin és l'estat de la taula *productes* després de l'execució? Justifiqueu la resposta.

```
CREATE or replace FUNCTION afegir_producte() RETURNS SETOF char(20)
AS $$
    DECLARE nom char(20);
    BEGIN
        FOR nom IN SELECT nomProducte FROM productes
            LOOP RETURN NEXT nom;
            END LOOP;
        IF NOT FOUND THEN
            RAISE EXCEPTION 'Error inesperat';
        END IF;
        INSERT INTO tipusProductes VALUES (1, 'aperitius', 10);
        INSERT INTO productes VALUES ('patates', 1, 4);
    RETURN;
END; $$LANGUAGE plpgsql;
```

En no haver-hi files a productes, el programa no entra en el FOR i per tant FOUND val fals. S'entra a l'IF i salta l'excepció, aturant l'execució del procediment. No s'arriba a fer l'insert i per tant la taula segueix estant buida.

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

- Esdeveniment que l'activa (cal indicar esdeveniment, taula i atributs rellevants)
- *Before* o *After* (justifiqueu la resposta)
- *Row* o *Statement* (justifiqueu la resposta)

a) REGLA1: Hi ha d'haver una fila a la taula `logs(tipusEsdeveniment, usuari, instant)` per cada vegada que un usuari de la base de dades faci insercions o modificacions de comandes. Es vol mantenir amb triggers el contingut de la taula `logs`.

b) REGLA2: L'`import` d'una comanda ha de ser igual a la suma dels resultats de multiplicar el `preu` de cada producte comprat en la comanda per la `quantitat` d'aquest producte que s'ha comprat. Es vol mantenir correcte el valor de l'atribut `import` quan quedi afectat per esdeveniments sobre la taula `productes`.

c) REGLA3: No pot existir cap producte a la base dades amb un `preu` superior al `preuMaxim` del tipus del producte. Es vol generar una excepció en cas que no es compleixi aquesta regla.

Regla 1

esdeveniments:
- INSERT comandes
- UPDATE comandes

before/after: AFTER. Per no fer feina innecessària en cas que es violi alguna RI de la BD
row/statement: STATEMENT. només ens importa l'esdeveniment i no les files

Regla 2

esdeveniments:
- UPDATE of preu on productes

before/after: AFTER. Per no fer feina innecessària en cas que es violi alguna RI de la BD
row/statement: ROW. Es pot fer statement però implicaria recalcular tots els imports de totes les comandes de la base de dades -> millor incremental

Regla 3

esdeveniment:

- UPDATE of idTipus,preu on productes
- UPDATE preuMaxim on tipusProductes
- INSERT on productes

before/after: BEFORE. Ja que és millor comprovar el més aviat possible el que es compleixen les restriccions d'integritat.

row/statement: ROW. Es pot fer statement però implicaria verificar que es compleix la restricció per tots els productes de la base de dades -> millor incremental

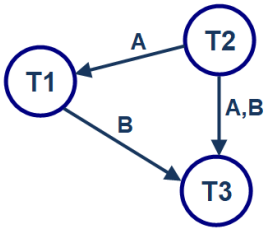
2.3 Implementeu un trigger que mantingui correcte el valor de l'atribut `import` de la taula `comandes` quan s'executi una sentència INSERT sobre la taula `productesComanda`. Tingueu en compte la REGLA2 de l'apartat anterior que defineix com es calcula l'import d'una comanda.

Podeu fer servir la plantilla de triggers de PostgreSQL següent:

```
CREATE TRIGGER nomTrigger
[BEFORE/AFTER] [UPDATE (of column)/DELETE/INSERT]ON taula
[FOR EACH ROW/FOR EACH STATEMENT] execute procedure nomp();

CREATE FUNCTION nomp() RETURNS trigger AS $$
    DECLARE
        ...
    BEGIN
        ...
    END;
$$ LANGUAGE plpgsql;
```

```
create function actualitza_import() returns trigger as $$
     declare
          preu integer;
     begin
          select p.preu into preu from productes p
          where nomProducte = new.nomProducte;

          update comandes
          set import = (import + (preu * new.quantitat))
          where numComanda = new.numComanda;

          return null;
        end;
$$ language plpgsql;

create trigger mante_import after insert on productesComanda
for each row execute procedure actualitza_import();
```

3. (2 punt) Donat el graf de precedències següent:



3.1 Digueu per cadascuna de les interferències que es produeixen: entre quines transaccions es produeixen, i els grànuls implicats.

No es produeix cap interferència donat que no hi ha cap cicle.

3.2 Doneu un horari que inclogui les mínimes operacions (R, RU, W, COMMIT) possibles i que es correspongui amb el graf de precedències.

Una possible solució és:

	T1	T2	T3
10		RU(A)	
20		W(A)	
30	R(A)		
40			R(A)
50	R(B)		
60			RU(B)
70		R(B)	
80			W(B)
90		commit	
100	commit		
110			commit

3.3 Digueu si l'horari que heu donat en l'apartat anterior és o no recuperable. Justifiqueu la resposta.

L'horari donat en la solució anterior és recuperable. Veure transparència.

3.4 Modifiqueu l'horari que heu donat, afegint una única operació (R,RU,W), per tal que s'afegeixi una única interferència de lectura no repetible entre T2 i T3. Si no és possible, justifiqueu la resposta.

Una possible solució és:

	T1	T2	T3
			R(A)
10		RU(A)	
20		W(A)	
30	R(A)		
40			R(A)
50	R(B)		
60			RU(B)
70		R(B)	
80			W(B)
90		commit	
100	commit		
110			commit

4. (2 punt) 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)	
40	RU(A)		
50	W(A)		
60	R(D)		
70	R(B)		
80			R(E)
100			RU(A)
110			W(A)
120			RU(F)
130		R(F)	
140		RU(D)	
150		W(D)	
160			W(F)
170	RU(F)		
180	W(F)		
190		COMMIT	
200			COMMIT
210	COMMIT		

4.1 Suposeu que s'incorpora un mecanisme per al control de la concurrència basat en reserves S, X, on les transaccions T1 i T3 treballa amb SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED, i T2 amb SET TRANSACTION ISOLATION LEVEL READ COMMITTED. Doneu l'horari aplicant reserves 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 esperi per un mateix gràNul, considereu que la política de la cua és FIFO (First In, First Out). En cas que cap acció de l'horari pugui executar-se en un moment donat, no continueu i justifiqueu el motiu.

T1	T2	T3
	LOCK(B,X)	
	RU(B)	
	W(B)	
LOCK(A,X)		
RU(A)		
W(A)		
R(D)		
R(B)		
		R(E)
		LOCK(A,X)
	L(F,S)	
	R(F)	
	U(F)	
	LOCK(D,X)	
	RU(D)	
	W(D)	
LOCK(F,X)		
RU(F)		
W(F)		
	COMMIT+U(B,D)	
COMMIT+U(A,F)		
		RU(A)
		W(A)
		LOCK(F,X)
		RU(F)
		W(F)
		COMMIT+U(A,F)

4.2 Suposeu que s'incorpora un mecanisme per al control de la concurrència basat en reserves S, X, on les transaccions T1, T2 i T3 treballen amb SET TRANSACTION ISOLATION LEVEL SERIALIZABLE. Doneu l'horari aplicant reserves 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 esperi per un mateix gränul, considereu que la política de la cua és FIFO (First In, First Out). En cas que cap acció de l'horari pugui executar-se en un moment donat, no continueu i justifiqueu el motiu.

T1	T2	T3
	LOCK(B,X)	
	RU(B)	
	W(B)	
LOCK(A,X)		
RU(A)		
W(A)		
LOCK(D,S)		
R(D)		
LOCK(B,S)		
		LOCK(E,S)
		R(E)
		LOCK(A,X)
	LOCK(F,S)	
	R(F)	
	LOCK(D,X)	

5. (2.5 punts) Considereu la taula T(a,b). La taula T té 40000 tuples i està emmagatzemada en un fitxer on a cada pàgina hi caben en promig 10 files.

5.1 En la taula no hi ha cap índex definit. Doneu el cost de la consulta següent i mostreu el detall de com es calcula.

```
SELECT * FROM T where a = valor
```

5.2 Es defineix ara un índex B+ no agrupat per l'atribut `a` de T. L'índex té d=50 i els nodes estan plens al 70% d'ocupació. Doneu el cost de la consulta de l'apartat anterior i mostreu el detall de com es calcula.

5.3 Es defineix ara un índex B+ no agrupat per l'atribut `b` de T. L'índex té d=50 i els nodes estan plens al 70% d'ocupació. Doneu el cost de la consulta següent i mostreu el detall de com es calcula.

```
SELECT SUM(b) FROM T
```

5.4 Suposeu ara que només hi ha un índex B+ agrupat per l'atribut `a` de T. L'índex té d=50 i els nodes estan plens al 70% d'ocupació. No es defineix cap índex sobre l'atribut `b`. Suposeu que els valors de l'atribut `a` estan distribuïts uniformement entre 1 i 40000. Doneu el cost de la consulta següent i mostreu el detall de com es calcula.

```
SELECT * FROM T WHERE a>=33000 AND b<=1000
```

Solució

5.1) Com que no hi ha cap índex definit a T i el fitxer no està ordenat, caldrà fer un recorregut seqüencial del fitxer de dades.

Card(T)=40000

f=factor de bloqueig del fitxer de dades=10

Cost = $\lceil Card(T) / f \rceil = \lceil 40000 / 10 \rceil$ = 4000 accessos a pàgines de dades.

5.2) Com que hi ha un índex definit per a l'atribut a, es baixarà per l'arbre buscant el valor (a=valor), i després un cop localitzat s'accedirà a la única fila (clau primària) que pot tenir aquest valor.

Al tenir una d=50, i ocupació del 70%, el número de valors per node és de $\lceil 2*50*0,7 \rceil$, és a dir, 70 valors per node, i 71 apuntadors en nodes interns.

Al baixar per l'arbre s'accedirà a tantes pàgines com nivells (h) té l'arbre.

$h=\lceil \log_{71} 40000 \rceil = 3$

El cost de localitzar la fila on es compleix a=valor, tenint en compte que "a" és clau primària, és:

Cost = h + 1 = 4

5.3) Caldrà accedir a totes les fulles de l'arbre per obtenir els valors de b i anar-los sumant. No és necessari accedir a les pàgines de dades, perquè els valors de b estan a l'índex. Per tant, el cost de resoldre la consulta és:

h és la mateixa que per l'índex sobre l'atribut a, ja que la d i la ocupació són les mateixes.

h = 3

F = nombre de fulles que cal llegir.

F = $\lceil 40000/70 \rceil$= 572

Cost = h + F - 1 = 3 +572 -1 =3 + 572 -1 =575, és resta 1 perquè hi ha una fulla que s'ha comptat dues vegades.

5.4) Com que només tenim un índex agrupat per l'atribut "a". Cal baixar per l'índex buscant el valor 33000. Tenint en compte que l'índex és agrupat, després es va a les pàgines de dades per buscar les files on es compleixi a>=33000. En accedir a les files dins de les pàgines de dades es podrà saber per cada fila si es compleix la condició b<=1000.

h és la mateixa que per l'índex sobre l'atribut a, ja que la d i la ocupació són les mateixes.

h = 3

D = nombre de pàgines de dades que cal llegir

D = $\lceil 7001 / 10 \rceil$ = 701

Cost = h + D = 3 + 701 = 704

Temps: 2 hores

Notes 20 d'abril

Revisió 21 d'abril (a les 13h, presencial)

Cada pregunta s'ha de lliurar en un full separat

Suposeu una base de dades amb les taules següents:

```
create table departaments(
codiDept integer,
nomDept char(50) unique not null,
pressupost integer not null check(pressupost>=0),
primary key (codiDept));

create table professors(
idProf integer,
nomProf char(50) not null,
codiDept integer not null,
sou real,
primary key (idProf),
foreign key (codiDept) references departaments);
-- codiDept és del departament en el que treballa el professor

create table assignatures(
idAssig char(5),
nomAssig char(50) not null,
codiDept integer not null,
profResponsable integer not null,
primary key (idAssig),
foreign key (codiDept) references departaments,
foreign key (profResponsable) references professors);
-- codiDept és del departament al que pertany l'assignatura
-- profResponsable és el professor responsable de l'assignatura

create table assignacions(
idProf integer,
idAssig char(5),
quadrimestre char(6),
horesAssig integer not null check (horesAssig>0),
primary key (idProf, idAssig, quadrimestre),
foreign key (idProf) references professors,
foreign key (idAssig) references assignatures);
-- hi ha una fila si el professor idProf ha impartit o imparteix
-- classes de l'assignatura idAssig en el quadrimestre
```

1. **(2 punts)** Considereu la base de dades donada a l'inici de l'examen. Doneu una sentència SQL per obtenir noms de les assignatures que o bé són l'única assignatura del departament al que pertanyen, o bé no han estat mai impartides pels seus professors responsables.

```
select distinct nomassig
from assignatures a
where not exists (
        select * from assignatures a2
        where a2.idAssig<>a.idAssig and
              a2.codiDpt=a.codiDpt)
    or not exists (
        select * from assignacions aP
        where aP.idAssig = a.idAssig and
              aP.idProf = a.profResponsable)
```

2. (2.5 punts) Tenint en compte l'esquema de la base de dades donat a l'inici de l'examen.

2.1. Per cadascuna de les situacions que es plantegen a continuació indiqueu si és o no possible que succeeixi.

- En cas que no sigui possible, indiqueu quina restricció ho evita.
- En cas que sigui possible, doneu un conjunt d' inserts per tal que passi (parteix de la BD buida).

a) És possible que existeixin dues o més assignatures amb el mateix professor responsable?

b) És possible que existeixin dues assignacions d'un mateix professor, a una mateixa assignatura en un mateix quadrimestre?

c) És possible que quan s'esborri algun professor que és responsable d'alguna assignatura, es posi a null el valor de l'atribut profResponsable per les assignatures afectades per l'esborrat?

2.2. Doneu una sentència de creació d'una asserció que garanteixi que cap professor que té un sou inferior a 1500 euros, doni classes a més de tres assignatures diferents en un mateix quadrimestre.

SOLUCIÓ

2.1

a) Sí que és possible que passi.

```
insert into departaments values (1,'ESSI', 10000);
insert into professors values (11,'Pep',1,1000);
insert into assignatures  values ('bd','bases de dades',1,11);
insert into assignatures  values ('so','sistemes opertius',1,11);
```

b) No és possible que passi, la PK d'assignacions ho evita.

c) No és possible que passi, l'atribut ProfResponsable té restricció not null.

2.2

```
CREATE ASSERTION maxim_assig CHECK (
NOT EXISTS (SELECT a.idprof, a.quadrimestre
        FROM assignacions a, professors p
        WHERE a.idprof=p.idprof and p.sou<1500
        GROUP BY a.idprof,a.quadrimestre
        HAVING count (*)>3));
```

3. **(1 punts)** Considereu les taules R, S, i T, i la vista Tot. Com es pot observar, d'acord amb els criteris vistos a classe, la vista Tot no és actualitzable i per aquest motiu els SGBD impediran que s'hi faci qualsevol operació d'actualització, tot i que en realitat podria ser que algunes operacions sí que es poguessin fer sense ambigüitat.

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

create view tot(a1,a2,a3,a4,a5,a6) as select T.a,T.b,S.b,S.c,R.c,R.d
                                      from T natural inner join S
                                             natural inner join R;
```

3.1 Indiqueu una operació (Delete o Update) sobre la vista Tot i un contingut de les taules R, S, i T que mostrin que la operació NO es pot realitzar sense ambigüitat. En cas de que no sigui possible, justifiqueu la resposta.

Una possible solució: Si tenim R(3,4) S(2,3) T(1,2) , la vista tindrà la tupla tot(1,2,2,3,3,4). Un Delete d'aquesta tupla no es podria fer de manera no ambigua ja que hi ha múltiples solucions.

3.2 Indiqueu una operació (Delete o Update) sobre la vista Tot i un contingut de les taules R, S, i T que mostrin que la operació es pot realitzar sense ambigüitat. En cas de que no sigui possible, justifiqueu la resposta.

Una possible solució: Si tenim R(3,4) S(2,3) T(1,2), la vista tindrà la tupla tot(1,2,2,3,3,4). I una operació d'update del valor 4 d'aquesta tupla es podria realitzar sense cap problema.

4. (2.5 punts)

4.1 Doneu una seqüència d'operacions d'àlgebra relacional per obtenir el codi i nom dels departaments que no tenen cap professor que hagi impartit classes el quadrimestre '1819q2'

```
A = assignacions(quadrimestre='1819q2')
B = A*professors
C = B[codiDept]
D = departaments[codiDept]
E = D-C
F = E * departaments
R = F[codiDept,nomDept]
```

4.2 Tenint en compte l'esquema de la base de dades de l'inici de l'examen i suposant que tenim:

a) **3 departaments,** amb codis de departament 1, 2, i 3.

b) **25 professors,** 10 del departament 1, i 15 professors del departament 2.

c) **5 assignatures** que pertanyen al departament 1, i el seu professor responsable és del departament 2.

Indiqueu quantes files s'obtindran en el resultat de les consultes següents. Justifiqueu la resposta. Si no podeu dir el nombre exacte, indiqueu un mínim i un màxim.

1) R = Assignatures * Departaments

   5 files: 1 per cada assignatura (assignada al departament 1).

2) P = Professors { codiDept -> codiDeptProf }

   R = Assignatures [profResponsable = idProf, codiDept = codiDeptProf] P

   0 files: perquè el codiDept de la assignatura (sempre 1) mai és igual al codiDept del professor responsable (sempre 2)

3) Q = Professors * Departaments

   R = Q[codiDept,nomDept]

   2 files: perquè en fer la join, només queden 25 files dels professors dels departaments 1 i 2, quan projectem només queden les dades dels 2 departaments de la taula professors

5. (2 punts)

5.1 Considerant la base de dades donada a l'inici de l'examen. Per cada una de les restriccions següents:

a) Si es pot implementar com a restricció en la definició de taules, doneu el codi SQL que cal afegir a la sentència CREATE TABLE de la BD per implementar-la, indicant en quina taula s'ha d'afegir.

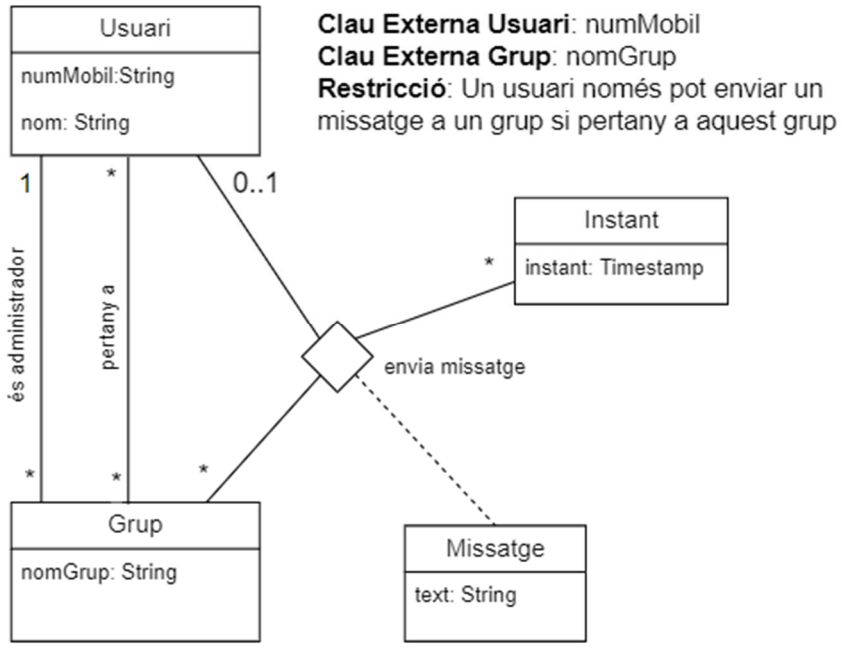
b) Si no es pot, expliqueu per què, i indiqueu com es podria implementar.

R1. El departament al que està assignada una assignatura ha de coincidir amb el departament del professor responsable d'aquella assignatura.

R2. Un professor no pot impartir més de 6 hores en una mateixa assignatura en un mateix quadrimestre.
R3. Quan s'esborren departaments que tenen assignatures o professors assignats, també caldrà esborrar les assignatures o professors d'aquests departaments. També caldrà esborrar les assignacions d'aquests professors a aquestes assignatures.

- R1: No es pot implementar com a restricció de taula o columna, perquè el codi de departament d'un professor i el departament al que està assignada una assignatura estan a taules diferents.
    - En SQL estàndard R1 es podria implementar amb una asserció.
- R2: Sí que és pot implementar com a restricció de columna a la columna horesAssig.
    - Afegir a la taula assignacions CHECK (horesAssig<=6)
- R3: Si que és pot implementar. Cal indicar que la política per mantenir la integritat referencial en cas d'esborrats és en cascada.
    - Afegir a Assignatures, foreign key (codiDept) references departaments ON DELETE CASCADE
    - Afegir a Professors, foreign key (codiDept) references departaments ON DELETE CASCADE
    - Afegir a Assignacions, foreign key (idProf) references professors ON DELETE CASCADE i foreign key (idAssig) references assignatures ON DELETE CASCADE

5.2 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 foranes i restriccions NOT NULL i UNIQUE que siguin necessàries.



Usuari (numMobil, nom)

Grup (nomGrup, administrador)

{administrador} referencia Usuari NOT NULL

Pertinença(nomGrup, numMobil)

{nomGrup} referencia Grup {numMobil} referencia Usuari

Missatge (instant, nomGrup, numMobil, text)

{numMobil] referencia Usuari NOT NULL

{nomGrup,numMobil} referencia Pertinença /* per implementar la restricció*/

text NOT NULL

Temps: 2,5 hores

Notes 20 juny - Revisió d'examen: 21 juny (presencial)

Cada pregunta s'ha de lliurar en un full separat

1. (1 punts) Donada la base de dades següent:

```
create table professors
(dni char(50),
nomProf char(50) unique,
sou integer,
primary key (dni));

create table despatxos
(modul char(5),
numero char(5),
superficie integer not null check(superficie <=25),
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 (instantInici < instantFi));
/*instantFI té valor nul quan una assignació és vigent */
```

Considereu el següent fragment de codi Java/JDBC. Implementeu el cos del programa per tal que per cada professor de la base de dades que tingui alguna assignació vigent al despatx número 124 del mòdul 'Omega', es decrementi el sou del professor en 20 euros. Cal que el programa tregui un missatge d'error si no hi ha cap professor que compleixi la condició indicada.

```
try {
 /* Tenim ja una connexió c establerta amb la base de dades */

 //codi a implementar

 c.commit();
 }
catch (ClassNotFoundException ce) {
        System.out.println ("Error al carregar el driver");}
catch (SQLException se) {
        System.out.println ("Excepcio: "+ se.getMessage());}
```

Recordeu que podeu usar els mètodes/codis d'excepció de JDBC estudiats a classe:

Statements

- Statement createStatement();
- ResultSet executeQuery(String sql);
- int executeUpdate(String sql);

PreparedStatement

- `PreparedStatement prepareStatement(String sql);`
- `ResultSet executeQuery();`
- `int executeUpdate();`
- `void setXXX(int posicioParametre, XXX valor);`

ResultSet

- `boolean next();`
- `XXX getXXX(String nomColumna)`

SQLException

- `// 23502 -not_null_violation`
- `// 23503 -foreign_key_violation`
- `// 23505 -unique_violation`
- `// 23514 -check_violation`
- `String getSQLState();`

Solució

```
try {
 /* Tenim ja una connexió c establerta amb la base de dades */
  Statement stDel = c.createStatement();
  int numDcr = stDel.executeUpdate("update professors p "+
                 "set sou = sou - 20 "+
                 "where exists (select * from assignacions a "+
                               "where a.dni = p.dni and a.modul='Omega' "+
                               "and a.numero='124' "+
                                "and a.instantFi is null)");
  if (numDcr==0)
     {System.out.println("Error");}
     else {System.out.println("S'han decrementat ="+numD);}

 c.commit();}
 catch (ClassNotFoundException ce)
        {System.out.println     ("Error     al     carregar     el     driver");}
 catch (SQLException se)
        {System.out.println ("Excepcio: "+ se.getMessage());}
```

2. **(3.5 punts)** Donades les transaccions següents (per cada transacció s'indica les operacions que vol fer i l'ordre en que la transacció realitza aquestes operacions):

T1: RU(A), W(A), RU(A), W(A), Commit

T2: R(A), RU(B), W(B), Commit

T3: R(B), R(C), R(B), Commit

Es demana:

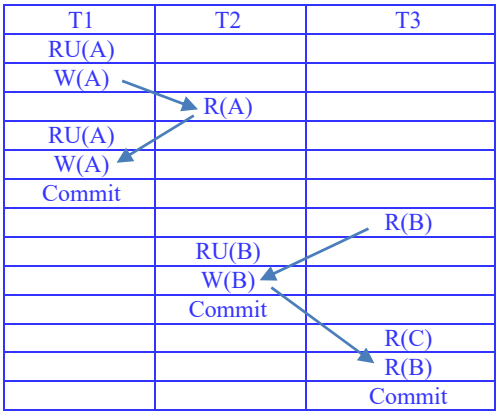
2.1 Proposeu un horari que inclogui les tres transaccions i que tingui dues interferències. Cal indicar el nom de les interferències, els grànuls implicats i les transaccions implicades.

2.2 Proposeu un horari, que inclogui encavalcaments entre les tres transaccions, i que tingui els dos horaris serials equivalents següents: T2;T1;T3 i T2; T3;T1.

2.3 Considereu ara només T1 i T2 i suposeu que tenim control de concurrència basat en reserves. Per cadascun dels quatre nivells d'aïllament, proposeu un horari amb aquestes dues transaccions de tal manera que es produeixi una interferència. En cas que no sigui possible produir la interferència, expliqueu breument perquè. 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. També ha d'incloure, en cas que es produeixin, les esperes de les transaccions.

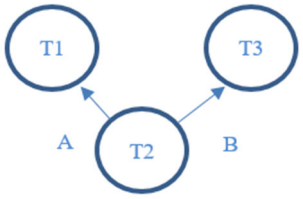
2.4 Considereu ara només T2 i suposeu que tenim control de concurrència basat en reserves. Per al nivell d'aïllament serialitzable proposeu un horari format per T2 i una altra transacció de tal manera que es produeixi una abraçada mortal. L'horari ha d'incloure, a més de les peticions que executen les transaccions (R, RU, W, COMMIT), les operacions de petició i alliberament de reserves i l'ordre d'execució de totes aquestes peticions. També ha d'incloure, en cas que es produeixin, les esperes de les transaccions. Doneu també el graf d'espera just després de que es produeixi l'abraçada mortal.

2.1.



Es produeix una lectura no confirmada entre T1 i T2 pel gränul A i una lectura no repetible entre T2 i T3 pel gränul B.

2.2. Per tenir els dos horaris serials equivalents demanats cal tenir el següent graf de precedències.



Per tant, cal generar un horari on les operacions no commutables entre T2 i T1 segueixin l'ordre següent: primer s'executen les de T2 i després les de T1. Entre T2 i T3 s'ha de produir un cas similar. Evidentment, l'horari ha d'estar lliure d'interferències.

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

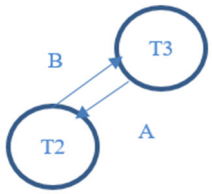
2.3. En el nivell Read Uncommitted no es fan reserves per lectura (modalitat S) i es fan reserves en modalitat X fins a l'acabament de la transacció. Per generar la interferència de lectura no confirmada hem d'assegurar-nos que la lectura del gränul A que fa T2 es faci entre les dues escriptures que fa T1.

T1	T2
Lock (A,X)	
RU(A)	
W(A)	
	R(A)
RU(A)	
W(A)	
Commit (+ unlock (A))	
	Lock (B,X)
	RU(B)
	W(B)
	Commit (+ unlock (B))

Aplicant les reserves del nivell Read Committed s'evita la lectura no confirmada. Per tan, no és possible produir un horari amb aquesta interferència.
Amb la resta de nivells d'aïllament també s'evita aquesta interferència. Per tant, no és possible generar un horari amb aquesta interferència.

2.4. A l'horari següent es pot veure que a partir de l'instant t=8 les dues transaccions estan suspeses sense possibilitat de continuar. T2 està esperant per reservar B (aquest gränul el té reservat T3) i T3 està esperant per reservar A (aquest gränul el té reservat T2).

Temps	T2	T3
1	Lock(A,S)	
2	R(A)	
3		
4		Lock (B,S)
5		R(B)
6	Lock(B,X)	
7		Lock (A,X)
8		



Graf d'espera en temps=8:

3. **(2 punts)** Considereu la base de dades següent.

```
competicions(idCompetició, nomCompetició)
equips(nomEquip, ciutat)
atletes(idAtleta, nomAtleta, dataNaixement, nomEquip)
    {nomEquip} referencia equips
arribadaAtletes(idCompetició, idAtleta, temps)
    {idCompetició} referencia competicions
    {idAtleta} referencia atletes
    -- el valor de l'atribut temps correspon al temps que
    ha trigat l'atleta a arribar a la meta en la competició
arribadaEquips(idCompetició, nomEquip, numAtletes)
    {idCompetició} referencia competicions
    {nomEquip} referencia equips
```

3.1 Es vol mantenir amb disparadors el contingut de la taula *arribadaEquips*.

- Hi ha d'haver una fila en aquesta taula per una competició i equip a partir de que es registra l'arribada a la meta del primer atleta de l'equip que participa en la competició.
- L'atribut *numAtletes* de la taula *arribadaEquips* ha de tenir per valor el número d'atletes d'un equip pels que s'ha registrat la seva arribada a la meta per una determinada competició.

Implementeu el(s) disparador(s) necessari(s) per al cas de l'esdeveniment d'inserció de files a la taula *arribadaAtletes*. Podeu fer servir la plantilla de triggers de PostgreSQL inclosa.

3.2 Suposeu ara que a la base de dades, en lloc d'haver-hi la taula *arribadaEquips*, hi ha la taula

```
ranquing(idCompetició, idAtleta, posicio, temps)
    {idCompetició} referencia competicions
    {idAtleta} referencia atletes
```

Es vol mantenir amb disparadors el contingut de la taula *ranquing*.

- Hi ha d'haver una fila en aquesta taula per cada atleta pel que s'ha registrat la seva arridada a la meta en una competició.
- El valor de l'atribut *temps* correspon al temps que ha trigat l'atleta a arribar a la meta en la competició.
- El valor de l'atribut *posicio* correspon al lloc on l'atleta ha quedat en la competició. La posició es calcula en funció del temps que han trigat els atletes a arribar a la meta. Dos atletes NO poden trigar exactament el mateix temps en arribar a la meta.

Implementeu el(s) disparador(s) necessari(s) per al cas de l'esdeveniment d'inserció de files a la taula *arribadaAtletes*. Aquestes insercions poden NO fer-se en l'ordre d'arribada a la meta dels atletes. Podeu fer servir la plantilla de triggers de PostgreSQL inclosa.

```
CREATE TRIGGER nomTrigger
[BEFORE/AFTER] [UPDATE (of column)/DELETE/INSERT] ON taula
[FOR EACH ROW/FOR EACH STATEMENT] execute procedure nomp();

CREATE FUNCTION nomp() RETURNS trigger AS $$
    DECLARE
        ...
    BEGIN
        ...
    END;
$$ LANGUAGE plpgsql;
```

2.1

```
CREATE or replace FUNCTION fn_novaArribadaAtleta_21() RETURNS trigger AS $$
   DECLARE
     nomEquipAtleta equips.nomEquip%type;
   begin
        -- INSERTAR O ACTUALITZAR LA TAULA D'arribadaEquips

        -- Cal recuperar el nom de l'equip de l'atleta
        select nomEquip into nomEquipAtleta
        from atletes
        where idatleta = new.idatleta;

        -- Si tenim valor fem UPDATE incrementant en 1 el numAtletes,
        -- sinó fem INSERT amb 1 atleta
        update arribadaEquips set numAtletes = numAtletes+1
        where idCompeticio = new.idCompeticio
              and nomEquip = nomEquipAtleta;

        if not found then
            insert into arribadaEquips
               values (new.idCompeticio, nomEquipAtleta, 1);
        end if;
       return null;
   END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER tg_novaArribadaAtleta_21
AFTER INSERT ON arribadaatletes
      FOR EACH ROW execute procedure fn_novaArribadaAtleta_21();
```

2.2

```
CREATE or replace FUNCTION fn_novaArribadaAtleta_22() RETURNS trigger AS $$
   DECLARE
     maxpos ranquing.posicio%type;

   begin
        -- INSERTAR O ACTUALITZAR A LA TAULA DE ranquing
        select max(r.posicio) into maxpos
        from ranquing r
        where r.idcompeticio = new.idcompeticio and r.temps <= new.temps;

        -- no hi ha temps inferior o igual, la posició ha de ser 1
        if maxpos is null then
          insert into ranquing
           values (new.idCompeticio, new.idAtleta, 1, new.temps);
        else
          --temps màxim inferior, la posició ha de ser màxima +1
          insert into ranquing
            values (new.idCompeticio, new.idAtleta, maxPos+1,new.temps);
        end if;

        -- desplaçem el ranquing dels arribats en temps superiors
        update ranquing set posicio = posicio+1
        where idcompeticio = new.idCompeticio and temps > new.temps;

        return null;
   END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER tg_novaArribadaAtleta_22
AFTER INSERT ON arribadaatletes
      FOR EACH ROW execute procedure fn_novaArribadaAtleta_22();
```

4. (3.5 punts)

4.1 Suposeu les dues situacions següents per a la base de dades de l'exercici 3:

- Les taules *equips* i *atletes* estan cadascuna en un espai virtual de taula.
- Les taules *equips* i *atletes* estan en un espai virtual d'agrupació.

a) Expliqueu quines diferències hi ha a nivell físic entre les dues situacions anteriors.

b) Doneu un exemple de consulta SQL que es vegi afavorida per cadascuna de les situacions anteriors.

4.2 Suposeu ara que la taula *atletes* està en un espai virtual de taula i té esquema:

atletes(idAtleta, nomAtleta, dataNaixement, nomEquip)

A la taula hi ha 120.000 atletes, i les pàgines de dades tenen un factor de bloqueig de 20. Hi ha 1000 atletes amb una data de naixement anterior a l'any 2000. Sobre la taula hi ha definits:

- un índex arbre B+ agrupat per *idAtleta* d'ordre d=255 i ple al 75%.
- un índex arbre B+ per *dataNaixement*. Es tracta d'un índex no agrupat amb ordre d=146, i ple al 80%.

a) Expliqueu com el SGBD resoldrà la consulta següent, doneu el cost en número de pàgines accedides, i mostreu el detall de com es calcula aquest cost.

SELECT * FROM atletes WHERE idAtleta = 34567 OR nomAtleta = 'Joan Pi'

b) Expliqueu com el SGBD resoldrà la consulta següent, doneu el cost en número de pàgines accedides, i mostreu el detall de com es calcula aquest cost.

SELECT * FROM atletes WHERE dataNaixement <= '31-12-1999'

4.1.a) En el primer cas, cada taula estarà emmagatzemada en un fitxer, les pàgines de cada fitxer contindran files únicament de la taula corresponent.

En el segon cas, les dues taules estaran emmagatzemades en el mateix fitxer, les pàgines de la taula estaran combinades les files de les dues taules tenint en compte el nom de l'equip (és a dir, hi haurà una fila corresponent a un equip, seguida de les files corresponents als atletes de l'equip, i a continuació un altre equip,...).

4.1.b) En el primer cas,

SELECT * FROM equips

En el segon cas,

SELECT * FROM equips natural inner join atletes WHERE nomEquip='BCN'

4.2.a) En aquest cas l'índex definit sobre idAtleta no ajuda en res a la resolució de la consulta, i per aquest motiu el SGBD farà una accés seqüencial a les pàgines de dades

cost = 120.000 files / 20 files per pàgina = 6.000 pàgines

4.2.b) En aquest cas s'usarà l'índex per dataNaixement, baixant per l'arbre B+, recorrent les fulles corresponents a valors on dataNaixement <= '31-12-1999', i per cada valor s'usarà el RID per buscar la fila corresponent al valor.

El cost de resoldre la consulta usant el índex seria:

n = ⌈2*d*ocupació⌉ = ⌈2*146*0,8⌉ = ⌈ 233,6 ⌉ = 234 valors per node

ap = n+1 = 234+1 = 235 apuntadors per node intern

h = ⌈ log235 120.000 ⌉ = ⌈ 2,14 ⌉ = 3 nodes = 3 pàgines

F = ⌈ valors a llegir en els nodes fulla / 234 valors per node⌉ =

⌈ 1000/234 nodes fulla⌉ = ⌈ 4,27 ⌉ = 5 pàgines

D = tantes pàgines com número de files que compleixen la condició

dataNaixement <= '31-12-1999' = 1000 pàgines

cost = 3+(5-1)+1000 = 1007 pàgines

BD-GEI **2.11.2022**

Temps: 2 hores

Notes 21 de novembre

Revisió 24 de novembre (a les 18h, presencial)

Cada pregunta s'ha de lliurar en un full separat

Suposeu una base de dades amb les taules següents:

```
create table departaments(
codiDept integer,
nomDept char(50) unique not null,
pressupost integer not null check(pressupost>=0),
primary key (codiDept));

create table professors(
idProf integer,
nomProf char(50) not null,
codiDept integer not null,
sou real,
ciutatRes char(100) not null,
primary key (idProf),
foreign key (codiDept) references departaments);
-- codiDept és del departament en el que treballa el professor

create table assignatures(
idAssig char(5),
nomAssig char(50) not null,
codiDept integer not null,
profResponsable integer not null,
primary key (idAssig),
foreign key (codiDept) references departaments,
foreign key (profResponsable) references professors);
-- codiDept és del departament al que pertany l'assignatura
-- profResponsable és el professor responsable de l'assignatura

create table assignacions(
idProf integer,
idAssig char(5),
quadrimestre char(6),
horesAssig integer not null check (horesAssig>0),
primary key (idProf, idAssig, quadrimestre),
foreign key (idProf) references professors,
foreign key (idAssig) references assignatures);
-- hi ha una fila si el professor idProf ha impartit o imparteix
-- classes de l'assignatura idAssig en el quadrimestre
```

1. (2,5 punts) Considereu la base de dades donada a l'inici de l'examen.

1.1 Doneu una sentència SQL per obtenir els noms dels professors que en el quadrimestre Q122 estan assignats a assignatures que al mateix quadrimestre Q122 tenen assignats professors de departaments diferents.

```
select distinct p.nomProf
from assignacions a, professors p, professors p1, assignacions
a1
where p.idprof=a.idprof and a.quadrimestre='Q122' and
      p1.idprof=a1.idprof and a1.idAssig=a.idAssig and
      a1.quadrimestre='Q122' and
      p1.codiDept<>p.codiDept
```

1.2 Doneu una sentència SQL per obtenir els codis dels departaments on algun dels seus professors no és responsable d'assignatura.

```
select distinct p.codidept
from professors p
where not exists (select *
                  from assignatures a
                  where p.idprof=a.profresponsable);
```

2. (2 punts) Donada la següent definició de vista:

```
CREATE VIEW AssignacionsAssignatura AS
      SELECT idAssig, quadrimestre, horesAssig
      FROM Assignacions
      WHERE quadrimestre = 'Q122'
WITH CHECK OPTION;
```

2.1 Digueu si la vista és o no actualitzable segons l'estàndard SQL. Raoneu la resposta**.**

SOLUCIÓ:

Per ser actualitzable cal que la vista:

- Faci SELECT sobre una única relació sense agregats ni distinct → OK
- No faci agrupacions → OK
- Els atributs seleccionats incloguin tots els atributs amb restricció NOT NULL → NO OK, donat que falta idProf que explícitament no és NOT NULL, però ho és implícitament pel fet de formar part de la clau primària.

Per tant NO ÉS ACTUALITZABLE

2.2 Suposeu que l'usuari A és el propietari de la taula `professors` i que s'executa la seqüència de sentències següent:

```
1- A: GRANT Insert, Select, Update ON Professors TO B WITH GRANT OPTION
2- B: GRANT Insert, Select, Update ON Professors TO C
3- A: GRANT Insert, Select, Update ON Professors TO D WITH GRANT OPTION
4- D: GRANT Update ON Professors TO C
5- A: GRANT Insert, Select, Update ON Professors TO E
6- A: REVOKE Insert, Select, Update ON Professors  FROM B CASCADE
7- A: REVOKE Insert, Select, Update ON Professors FROM D RESTRICT
8- E: GRANT Select, Update ON Professors TO C
```

2.2.1. Doneu el diagrama d'autoritzacions resultant després d'executar les sentències anteriors.

2.2.2. Suposat les autoritzacions representades en el diagrama anterior, digueu si l'usuari C podrà executar la sentència següent. Raoneu la resposta.

```
UPDATE Professors
SET sou = 1000
WHERE sou < 1000;
```

2.2.3. Suposant les autoritzacions representades en el diagrama anterior, digueu si l'usuari E podrà executar la sentència de l'apartat 2.2.2. Raoneu la resposta

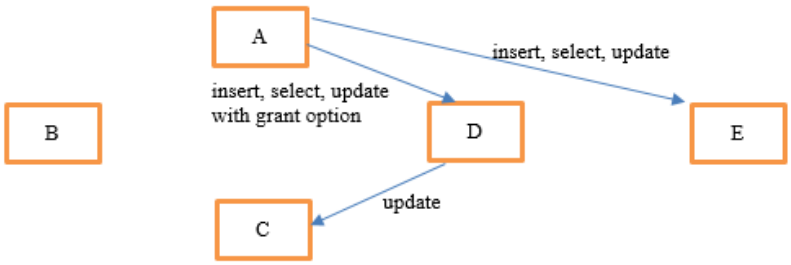
2.2.4. Suposeu ara que la sentència 7 queda substituïda per la que es dóna a continuació:

```
7- A: REVOKE GRANT OPTION ON Professors FROM D CASCADE
```

Digueu si amb aquest canvi l'usuari C podrà executar la sentència de l'apartat 2.2.2.

- En cas que pugui, doneu el nou diagrama d'autoritzacions tenint en compte la sentència substituïda, i raoneu la resposta.
- En cas que no pugui, indiqueu les sentències GRANT mínimes necessàries per a que l'usuari C la pugui executar.

2.2.1 El diagrama és:



2.2.2 L'usuari C no pot executar la sentència UPDATE perquè no té autorització a llegir el sou, i el necessita per la verificar la condició. La sentència 8 no té cap efecte perquè E no té autorització per atorgar permisos.

2.2.3 L'usuari E pot executar la sentència UPDATE perquè té permisos per SELECT i UPDATE.

2.2.4 L'usuari C no pot executar la sentència UPDATE perquè en substituir la sentència 7, el nou REVOKE fa que a l'usuari C li falta el privilegi de SELECT i, a més a més, la de UPDATE. L'únic usuari que li pot donar permisos és l'usuari A, per tant la sentència GRANT que faria falta:

```
A: GRANT Update, Select ON Professors TO C
```

3. (2 punts) Doneu una seqüència d'operacions en àlgebra relacional per obtenir l'identificador i nom dels professors que tenen un sou superior a 3000 i són responsables d'assignatures que han tingut 2 ó més professors diferents assignats en quadrimestres diferents.

```
A = assignacions[idProf, idAssig, quadrimestre]
B = A{idProf->idProfB, idAssig->idAssigB, quadrimestre->quadrimestreB}
C = A[idAssig = idAssigB, idProf <> idProfB,
      quadrimestre <> quadrimestreB]B
D = assignatures * C   // no fa join per professor, perquè els atributs
                       // tenen noms diferents
E = D[profResponsable]
F = professors(sou > 3000)
G = E[profResponsable = idProf]F
R = G[idProf, nomProf]
```

4. (2 punts)

4.1 Per cada una de les restriccions següents:

a) R1. Tots els professors que tenen assignacions a una mateixa assignatura han de pertànyer al mateix departament.

b) R2. Un professor només podrà consultar les seves assignacions.

c) R3. Al quadrimestre de primavera del curs 2022-2033 (quadrimestre='Q222') els professors han d'impartir entre 4 i 6 hores de docència per cada assignatura a la que estiguin assignats.

- Si es pot implementar com a restricció en la definició de taules, doneu el codi SQL que cal afegir a la sentència CREATE TABLE de la BD per implementar-la, indicant en quina taula s'ha d'afegir.
- Si no es pot, expliqueu per què, i indiqueu com es podria implementar.

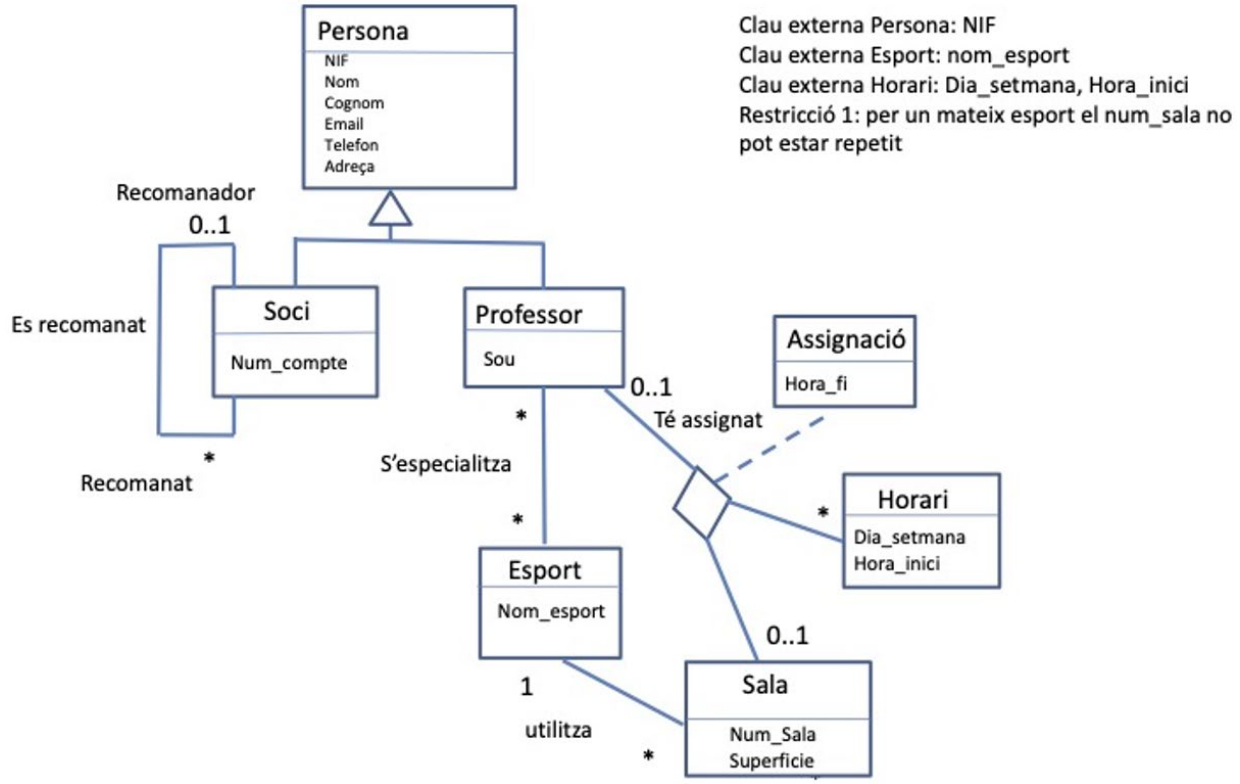
4.2 Doneu una asserció que comprovi que tots els professors de fora de Barcelona fan en total menys de 20 hores de classe per quadrimestre.

2.1

- R1. No es pot implementar com a restricció de taula o columna perquè el departament al que pertanyen els professors està a una altra taula. En SQL estàndard es podria fer una asserció i en Postgres aquesta restricció s'implementaria amb disparadors.
- R2.No es pot implementar com a restricció de taula o columna. S'haurien de crear vistes per a que cada professor només pugui consultar les seves assignacions i després l'administrador atorgar privilegis de consulta sobre la vista corresponent a cada professor.
- R3. Sí que es podria implementar com a restricció de taula. A la taula assignacions hauríem d'afegir un CHECK (quadrimestre<>'Q222' or (quadrimestre='Q222' and hores>=4 and hores<=6))

2.2 – CREATE ASSERTION sous_valids CHECK (
NOT EXISTS (SELECT *
FROM professors p natural inner join assignacions a
WHERE p.ciutatRes <> 'Barcelona'
GROUP BY a.idprof, a.quadrimestre
HAVING SUM(a.horesAssig) > 20))

5. (1.5 punts) 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 foranes i restriccions NOT NULL i UNIQUE que siguin necessàries.




```
persona(nif, nom, cognom, email, adreça, telèfon)
soci(nif, num_compte, nif_soci_recomanador)
      {nif} referencia persona
      {nif_soci_recomanador} referencia soci
professor(nif, sou)
      {nif} referencia persona
esport(nom esport)
especialització (nif professor,nom esport)
      {nif_professor} referencia professor
      {nom_esport} referencia esport
horari(dia setmana,hora inici)
sala(nom esport,num sala, superficie)
      {nom_esport} referencia esport
assignacions(dia setmana,hora inici,nom esport,num sala,
                  nif_professor,hora_fi)
      {dia_setmana,hora_inici} referencia horari
      {nom_esport,num_sala} referencia Sala
      {nif_professor} referencia Professor
      {nif_professor} not null
      Unique (dia_setmana,hora_inici, nif_professor)
```

Temps: 3 hores

Notes 25 de gener. Revisió d'examen: 26 de gener a la tarda (presencial)

Cada pregunta s'ha de lliurar en un full separat

1. (1 punts) Donada la base de dades següent:

```
artistes (nom, genere, anyDebut, anyRetirada)
albums(titol, anyAlbum, artista)
     {artista} referencia artistes
```

Considereu el codi Java/JDBC següent que esborra artistes que compleixen certes condicions:

```
try {
   PreparedStatement s, s2;
   ResultSet r, r2;
   int num = 0;
   // c és la connexió a la BD
   // Seleccionem els artistes que es van retirar abans del 1960
   s = c.prepareStatement("select distinct nom from artistes "+
                          "where anyRetirada < 1960");
   r = s.executeQuery();
   // Preparem un statement s2 per comprovar si un artista té àlbums
   s2 = c.prepareStatement("select * from albums where artista = ?");
   while (r.next()) {
      // Per cada artista resultant de s, executem s2 per comprovar si té àlbums
      s2.setString(1, r.getString("nom"));
      r2 = s2.executeQuery();
      if (r2.next()) {
          // Té àlbums, no l'esborrem, fem rollback i acabem
          System.out.println("Error: Algun dels artistes a esborrar té àlbums");
          c.rollback();
          num = 0;
          break; }
      else {
          // No té àlbums, executem s3 per a esborrar-lo i seguim
          Statement s3 = c.createStatement();
          num = s3.executeUpdate("delete from artistes where nom = '" +
                                 r.getString("nom") + "'"); }
   }
   if (num > 0) System.out.println("Esborrats els artistes antics i sense àlbums");
   // TANCAMENT DE LES ESTRUCTURES, COMMIT I DESCONNEXIÓ
}
catch (SQLException se) {
      System.out.println("Excepció: " + se.getSQLState() + ":" + se.getMessage());
}
```

Proposeu una implementació alternativa a aquest fragment de codi, de manera **que tingui el mateix comportament**, i **que compleixi els criteris de qualitat** de l'assignatura. **Justifiqueu** els canvis que heu fet.

Recordeu que podeu usar els mètodes/codis d'excepció de JDBC estudiats a classe i que estan inclosos a continuació:

Statement

- `Statement createStatement();`
- `ResultSet executeQuery(String sql);`
- `int executeUpdate(String sql);`

PreparedStatement

- `PreparedStatement prepareStatement(String sql);`
- `ResultSet executeQuery();`
- `int executeUpdate();`
- `void setXXX(int posicio, XXX valor);`

ResultSet

- `boolean next();`
- `XXX getXXX(String nomColumna)`

SQLException

- `// 23502 -not_null_violation`
- `// 23503 -foreign_key_violation`
- `// 23505 -unique_violation`
- `// 23514 -check_violation`
- `String getSQLState();`

SOLUCIÓ:

Una solució que compleix tots els criteris seria:

```
try {
      // c és la connexió a la BD
      Statement s = c.createStatement();
      // Intentem esborrar els artistes que es van retirar abans del 1960
      // Si algún d'ells té un àlbum a la BD, saltarà una excepció de clau forana
      int num = s.executeUpdate("delete from artistes where anyRetirada < 1960");
      if (num>0) System.out.println("Els artistes antics i sense àlbums han estat
esborrats ");
      // TANCAMENT DE LES ESTRUCTURES, COMMIT I DESCONNEXIÓ
}
catch (SQLException se) {
      if(se.getSQLState().equals("23503"))
            System.out.println("Error: Algun dels artistes a esborrar té àlbums");
         else {
            System.out.println("Excepció: " + se.getSQLState() + ":"
                                          + se.getMessage());
      }
}
```

El que s'ha fet és:

- Eliminar el primer select (PreparedStatement s) perquè és innecessari: es pot afegir com a condició del delete. A més, té un distinct innecessari, i s'hauria de fer amb un Statement.
- Eliminar el segon select (PreparedStatement s2) perquè és innecessari: es pot detectar l'error capturant una excepció de violació de clau forana produïda pel delete.
- Com que es pot fer tot amb una sentència de delete que s'executarà una sola vegada (ja no tenim cap bucle), hem utilitzat un Statement. A la solució inicial, l'opció correcta hauria estat un PreparedStatement (si el bucle fos necessari).

2.(3 punts) Suposeu una base de dades amb les taules i contingut següents.

```
articles (idArticle, nom,    marca,      stock)
            (42,    'ps5', 'sony',      1)
            (43,    'ps4', 'nintendo',  100)

cistelles (idClient, idArticle, quantitat)
             (31,    42,         null)
             (55,    42,         null)
```

Considereu l'horari que apareix a continuació.

T1	T2	T3
	select * from articles where marca = 'sony';	
		select * from articles where idArticle = 42;
	update cistelles set quantitat = 1 where idClient = 31 and idArticle = 42;	
		update cistelles set quantitat = 1 where idClient = 55 and idArticle = 42;
	update articles set stock = stock - 1 where idArticle = 42;	
		select * from articles where idArticle = 42;
		commit
update articles set marca = 'sony';		
commit		
	abort	

2.1 Tenint en compte que el grànul és la fila (per identificar un grànul, utilitzeu la clau primària de la fila). Es demana:

a) Doneu l'horari en termes d'operacions de baix nivell sobre grànuls (no poseu operacions sobre el grànul IC, a no ser que hagueu vist que hi ha alguna interferència Fantasma a l'horari donat).

b) En cas que hi hagi interferències a l'horari, indiqueu, per cada interferència: nom de la interferència, transaccions implicades, grànuls afectats, i el nivell mínim d'aïllament per evitar-la. En cas que no n'hi hagi, indiqueu els horaris serials equivalents.

2.2 Suposeu ara que T1 treballa amb nivell d'aïllament REPEATEABLE READ, i T2 i T3 treballen amb READ COMMITTED.

a) Doneu l'horari anterior aplicant els nivell d'aïllament indicats. L'horari ha d'incloure, a més de les operacions que executen les transaccions (R, RU, W, COMMIT, ABORT), 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).

b) Doneu el graf d'espera just abans de l'abort de T2.

2.3 En cas que hi hagi interferències a l'horari donat a l'apartat 2.2, cal que indiqueu per cada interferència: nom de la interferència, transaccions implicades, grànuls afectats. En cas que no n'hi hagi, indiqueu els horaris serials equivalents.

SOLUCIÓ 2.1

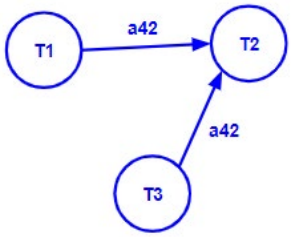
T1	T2	T3
	R(a42)	
		R(a42)
	RU(c3142)	
	W(c3142)	
		RU(c5542)
		W(c5542)
	RU(a42)	
	W(a42)	
		R(a42)
		commit
RU(a42)		
W(a42)		
RU(a43)		
W(a43)		
commit		
	abort	

Sí, existeix tres interferències:

- Entre T1 i T2 lectura no confirmada sobre a42. Nivell d'aïllament: Read Committed.
- Entre T2 i T3 lectura no repetible sobre a42. Nivell d'aïllament: Repeteable Read.
- Entre T2 i T3 lectura no confirmada sobre a42. Nivell d'aïllament: Read Committed.

SOLUCIÓ 2.2

T1 Repeateable Read	T2 Read Committed	T3 Read Committed
	L(a42, S)	
	R(a42)	
	U(a42)	
		L(a42, S)
		R(a42)
		U(a42)
	L(c3142, X)	
	RU(c3142)	
	W(c3142)	
		L(c5542, X)
		RU(c5542)
		W(c5542)
	L(a42, X)	
	RU(a42)	
	W(a42)	
		L(a42, S)
LOCK(a42, X)		
	ABORT+U(c3142, a42)	
		R(a42)
		U(a42)
		COMMIT+U(c5542, a42)
RU(a42)		
W(a42)		
L(a43, X)		
RU(a43)		
W(a43)		
COMMIT+U(a42, a43)		



SOLUCIÓ 2.3

No hi ha cap interferència.

- Les de lectura no confirmada s'han evitat pels nivells d'aïllament usats en les transaccions.
- La de lectura no repetible no s'evita amb Read Committed, però s'evita circumstancialment degut a l'Abort.

Els horaris serials equivalents són: T3;T1

3.(3 punts) Considereu la taula `Empleats(id, nom, ciutat)`. Hi ha un índex arbre B+ agrupat per `id` i dos índexs arbre B+ no agrupats, un per `nom` i un per `ciutat`. La taula té 100.000 tuples, i els id dels empleats estan repartits uniformement entre 1 i 100.000. Hi ha 1.000 empleats de Barcelona, i d'aquests 200 tenen el id més gran que 20.000. Els índexs són tots d'ordre 50, estan ocupats en mitjana al 80%, i a cada pàgina de dades hi ha 5 tuples.

3.1 Donada la consulta següent, calculeu (mostreu i justifiqueu clarament els càlculs) el cost en nombre de pàgines (d'índex i de dades) que es llegiran si la consulta es resol:

a) usant l'índex per `id`
b) usant l'índex per `ciutat`
c) usant els 2 índexs alhora

```
SELECT *
FROM Empleats e
WHERE e.id > 20000 AND e.ciutat ='Barcelona'
```

3.2 Considereu ara la consulta següent. Quin hauria de ser el valor de X per tal de que la consulta resolta usant l'índex per `id` tingui un cost menor que resolent la consulta amb un recorregut seqüencial? Mostreu i justifiqueu clarament els càlculs que heu fet per obtenir el valor de X.

```
SELECT *
FROM Empleats e
WHERE e.id > X AND e.ciutat ='Barcelona'
```

SOLUCIÓ:

3.1)

a) $n = 2*d*\text{ocupació} = 2*50*0,8 = 80$ valors per node
$h = \lceil \log_{81} 100000 \rceil = 3$
$D = 80000/5 = 16000$

Cost total = 3 + 16000

b)

n = 2*d*ocupació = 2*50*0,8 = 80 valors per node
h = ⌈ $\log_{81}$ 100000 ⌉ = 3
F = ⌈ valors a llegir en els nodes fulla / 80⌉ = ⌈ 1000/80 ⌉ nodes fulla = 13

Cost total = 3 + 12+ 1000 = 1015 pàgines

c)

n = 2*d*ocupació = 2*50*0,8 = 80 valors per node
h = ⌈ $\log_{81}$ 100000 ⌉ = 3
F = id a llegir en els nodes fulla / 80 = 80000/80 nodes fulla = 1000

Cost índex per id = h + (F-1) = 3 + 999 = 1002
Cost índex per ciutat = h + (F-1) = 3 + 12 = 15 (aquesta F s'ha calculat a 3.1.b)

Cost total: 15 + 1002 +200 pàgines

3.2)

Cost Rec. Seq: 100000/5 = 20000 pàgines

Índex per id: 3 + ⌈ (100000- X)/5 ⌉

$$3 + \lceil (100000 - X)/5 \rceil < 20000$$

Qualsevol X més gran que 20 farà que la consulta amb índex tingui un cost menor (es considera també correcta la solució X més gran que 15 a la que s'arriba si no es considera l'arrodoniment).

4.(3 punts) Considereu una base de dades amb les taules següents. Per simplificar, suposeu que les dates estan implementades com a `integer`.

```
clients (dni, nom, telèfon, mail)
tipusHabitacions (idHotel, idTipus, descripció)
        -- hi ha una fila a la taula per cada tipus d'habitació
        que hi ha en un hotel
reserves (dni, dataIni, dataFi, idHotel, idTipus)
        {idHotel, idTipus} referencia tipusHabitacions
        {dni} referencia clients
        {dataFi} is not null
        -- les reserves es fan d'un tipus d'habitació d'un hotel
        des d'una dataIni fins a la dataFi. Per exemple hi pot
        haver una reserva d'una habitació doble a l'hotel Ritz
        entre les dates 4 i 8 feta pel client amb dni 333.
        -- per simplificar suposarem que les dates estan
        implementades com un número enter. Per exemple, una
```

```
        reserva entre les dates 4 i 8, ocuparà una habitació en
        les dates 4,5,6,7 i 8.
ocupacioHotels (idHotel, idTipus, data, reservades)
        {idHotel, idTipus} referencia tipusHabitacions
        check (reservades>=1)
        -- l'atribut reservades indica quantes habitacions del
        tipus idTipus de l'hotel idHotel estan reservades en una
        data concreta.
        -- hi ha una fila a la taula ocupacioHotels quan hi ha
        una o més reserves d'habitacions d'un tipus en un hotel
        i una data. Per exemple, hi haurà una fila si en la data
        7 hi ha 5 habitacions dobles reservades a l'hotel Ritz.
        Però no hi haurà cap fila corresponent a l'hotel Ritz,
        habitació individual, i data 10, si per aquest hotel,
        tipus d'habitació i data no hi ha cap habitació
        reservada.
```

Implementeu amb disparadors el comportament següent que **s'ha de fer complir quan s'insereix una reserva a la taula reserves**:

- No hi pot haver dues reserves solapades en dates fetes per un mateix client. En cas que no es compleixi caldrà generar una excepció. Dues reserves estan solapades si en els intervals entre la seva data d'inici i data fi, s'inclou una mateixa data (per exemple, la reserva ('dni1', 10, 20, 'hotel1', 'individual') està solapada amb la reserva ('dni1', 18, 25, 'hotel2', 'individual') concretament en les dates 18, 19 i 20).
- Hi ha d'haver una fila a la taula ocupacióHotels per un hotel, tipus d'habitació i data a partir de que hi hagi hi ha una o més reserves fetes d'una habitació del tipus, a l'hotel, en la data.
- El valor de l'atribut derivat *reservades*, de la taula *ocupacioHotels*, ha de correspondre a la quantitat d'habitacions del tipus que hi ha reservades en l'hotel en aquella data.

Podeu fer servir la plantilla de triggers següent.

```
CREATE OR REPLACE FUNCTION actualitza_lloger() RETURNS TRIGGER AS $$
DECLARE
  ... ...
BEGIN
  ... ...
EXCEPTION
  WHEN { raise_exception | unique_violation | foreign_key_violation | check_violation | others }
  THEN
   SELECT texte INTO missatge FROM missatgesExcepcions WHERE num=?;
```

```
    RAISE EXCEPTION '%', missatge;
   ...
END;
$$ LANGUAGE PLPGSQL;

CREATE TRIGGER disparador {BEFORE | AFTER} {esdeveniment[OR ...]} ON taula
FOR [EACH] {ROW | STATEMENT} EXECUTE PROCEDURE actualitza_lloger();
```

SOLUCIÓ:

```
CREATE OR REPLACE FUNCTION novaReservaBef() RETURNS TRIGGER AS $$
BEGIN
    if (exists(select * from reserves
               where new.dni = dni and
                    ((new.dataIni >=dataini and new.dataIni<=dataFi)
                  or (new.dataFi >=dataIni and new.dataFi<=dataFi)
                  or (new.dataIni <=dataIni and new.dataFi>=dataFi))))
      then raise exception 'reserva solapada';
     end if;
     return new;
END;
$$ LANGUAGE PLPGSQL;

CREATE TRIGGER disparadorBef BEFORE INSERT ON reserves
FOR EACH ROW EXECUTE PROCEDURE novaReservaBef();

CREATE OR REPLACE FUNCTION novaReservaAft() RETURNS TRIGGER AS $$
DECLARE
    dataA integer;
BEGIN
    for dataA in new.dataIni..new.dataFi
      loop
            update ocupacioHotels
            set ocupades = ocupades + 1
            where idHotel = new.idHotel
                  and idTipus = new.idTipus
                  and data = dataA;
            if not found then
                  insert into ocupacioHotels
                  values (new.idHotel,new.idTipus,dataA,1);
            end if;
      end loop;
    return null;
END;
$$ LANGUAGE PLPGSQL;

CREATE TRIGGER disparadorAft AFTER INSERT ON reserves
FOR EACH ROW EXECUTE PROCEDURE novaReservaAft();
```

BD **24.4.2023**

Temps: 2 hores

Notes 9 maig tarda **Revisió: 10 maig al mat** **Cada pregunta en un full separat**

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 en un periode que va des de
l'instantiInici a l'instantFi
-- instantFi te valor null quan una assignacio es encara vigent.
```

1. **(2,5 punts)** Escriviu una sentència SQL per obtenir els despatxos que només han tingut un únic professor assignat amb un sou superior a 1000 i que, a més, el mateix professor ha estat assignat al despatx en 3 o més períodes diferents.
La sentència ha de mostrar l'identificador del despatx, el nom del professor i la quantitat de vegades que el professor ha estat assignat al despatx.

```
select a.modul, a.numero, p.nomProf, count(*) as numVegades
from assignacions a join professors p on a.dni = p.dni
where p.sou > 1000 and
not exists (
           select *
           from assignacions a2 join professors p2 on a2.dni = p2.dni
           where p2.sou > 1000 and
                 a2.modul = a.modul and a2.numero = a.numero and
                 a2.dni != a.dni
           )
group by a.modul, a.numero, p.nomProf
having count(*) >= 3
```

2. **(2,5 punts)** Escriviu una seqüència d'operacions d'àlgebra relacional per obtenir el dni dels professors que tenen dues o més assignacions en despatxos que estan al mateix mòdul però tenen diferent número, o no tenen assignacions a despatxos més grans de 15 metres quadrats.

```
A = assignacions[dni, modul, numero]
B = assignacions[dni, modul, numero]
C = B{ dni-> dni2, modul-> modul2, numero->numero2}
D = C[dni = dni2, modul=modul2, numero !=numero2]A
E=D[dni]
F=despatxos(superfície>15)
G=F*assignacions
H=G[dni]
I=professors[dni]
J=I-H
R=E_u_J
```

3. **(2,5 punts)**

3.1 Definir en SQL estàndard una asserció per garantir que no hi ha cap mòdul que tingui 5 o més assignacions d'un mateix professor a despatxos diferents.

```
CREATE ASSERTION no-mes-de-dos CHECK(
       NOT EXISTS(SELECT a.modul
             FROM assignacions a
             GROUP BY a.dni, a.modul
             HAVING count(distinct a.numero)>=5));
```

3.2 Considereu les vistes i la consulta SQL següents:

```
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;

SELECT d1.nomprof, d2.nomprof
FROM dadesPersonalActualOmega d1,
           dadesPersonalActualOmega d2
WHERE d1.modul=d2.modul AND
      d1.numero = d2.numero AND
      d1.nomprof <> d2.nomprof;
```

a) Explica breument què retorna la consulta.

S'obtenen les parelles de professors que tenen assignacions vigents a un mateix despatx del mòdul Omega.

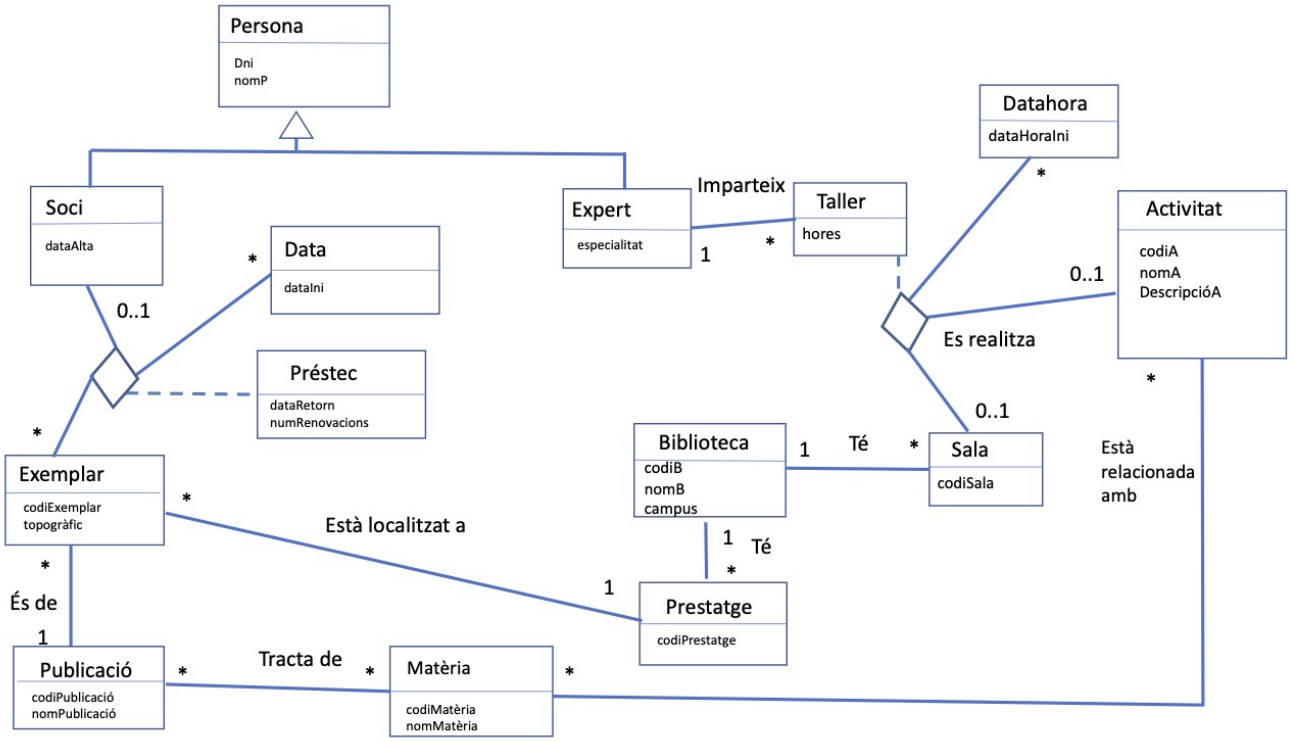
b) Doneu una sentència SQL definida sobre les taules de la base de dades, que compleixi els criteris de qualitat establerts a l'assignatura, i que retorni el mateix resultat que la consulta donada.

```
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) Expliqueu breument quins avantatges pot aportar el fet de fer definir la consulta sobre les vistes, en lloc de definir-la sobre les taules bàsiques.

Independència lògica de dades, simplificació de consultes i programes.

4. (2,5 punts) Donat el model conceptual següent en UML:



Clau externa **Biblioteca:** codiB
Clau externa **Sala:** No poden existir dues sales amb el mateix codiSala a la mateixa biblioteca, però sí a biblioteques diferents.
Clau externa **Prestatge:** No poden existir dos prestatges amb el mateix codiPrestatge a una mateixa biblioteca, però sí a biblioteques diferents.
Clau externa **Publicació:** codiPublicació
Clau externa **Exemplar:** codiExemplar
Clau externa **Matèria:** codiMatèria
Clau externa **Persona:** dni
Clau externa **Data:** dataIni
Clau externa **DataHora:** dataHoraIni
Clau externa **Activitat:** codiA

Es demana fer la traducció a model relacional donant el disseny lògic de la base de dades resultant. Concretament cal indicar:

- taules
- atributs
- claus primàries
- claus foranes
- restriccions NOT NULL per a les claus foranes que ho requereixin
- restriccions UNIQUE en cas d'existir claus alternatives

Solució:

```
Biblioteca (codiB, nomB, campus)
Sala(codiSala,codiB)
    {codiB} Referencia Biblioteca
Prestatge(codiPrestatge, codiB)
    {codiB} Refernencia Biblioteca
Publicació(codiPublicació, numPublicació)
Exemplar(codiE, topogràfic, codiPublicació, codiB,codiPrestatge)
    {codiPublicació} Referencia Publicació NOT NULL
    {codiB,codiPrestatge} Referencia Prestatge NOT NULL
Matèria(codiMatèria, nomMatèria)
TractaDe(codiPublicació,codiMatèria)
    {codiPublicació} Referencia Publicació
    {codiMatèria} Referencia Matèria
Persona(dni,nomP)
Soci(dni, dataAlta)
    {dni} Referencia Persona
Expert(dni, especialitat)
    {dni} Referencia Persona
Prèstec(codiE,dniSoci,dataIni,dataRetorn,numRenovacions)
    {codiE} Referencia Exemplar
    {dniSoci}Referencia Soci NOT NULL
Activitat(codiA,nomA,descripcióA)
RelacionadaAmb(codiA,codiMateria)
    {codiA} Referencia Activitat
    {codiMatèria} Referencia Matèria
RealitzaTaller(codiB,codiSala,codiA,dataHoraIni,hores, dniExpert)
    {codiB,codiSala} Referencia Sala
    {codiA} Referencia Activitat NOT NULL
    (dataHoraIni,codiA) UNIQUE NOT NULL
    {dniExpert} Referencia Expert NOT NULL
```

Temps: 3 hores

Notes 27 de juny. Revisió d'examen: 29 de juny a la tarda (presencial)

Cada pregunta s'ha de lliurar en un full separat

1. (1 punts) Donada la base de dades següent:

```
create table professors
(dni char(50),
nomProf char(50) unique,
sou integer check (sou<=2000),
primary key (dni));

create table despatxos
(modul char(5),
numero char(5),
superficie integer not null check(superficie <=25),
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 (instantInici < instantFi));
```

Considereu el següent fragment de codi Java/JDBC. Implementeu el cos del programa per tal que per cada professor de la base de dades que tingui assignacions a un despatx amb una superfície més gran que 20 que no sigui del mòdul Omega, s'augmenti el seu sou en 100. Cal que el programa tregui un missatge d'excepció si el sou d'algun professor passa a ser superior a 2000. En cas que no hi hagi cap excepció el programa indicarà el número de professors modificats i si no se n'ha modificat cap, es mostrarà un text que ho indiqui.

```
try {
 /* Tenim ja una connexió c establerta amb la base de dades */

 //codi a implementar

 c.commit();
 }
catch (ClassNotFoundException ce) {
        System.out.println ("Error al carregar el driver");}
catch (SQLException se) {
        System.out.println ("Excepcio: "+ se.getMessage());}
```

Recordeu que podeu usar els mètodes/codis d'excepció de JDBC estudiats a classe:

Statements

- `Statement createStatement();`
- `ResultSet executeQuery(String sql);`
- `int executeUpdate(String sql);`

PreparedStatement

- PreparedStatement prepareStatement(String sql);
- ResultSet executeQuery();
- int executeUpdate();
- void setXXX(int posicioParametre, XXX valor);

ResultSet

- boolean next();
- XXX getXXX(String nomColumna)

SQLException

- // 23502 -not_null_violation
- // 23503 -foreign_key_violation
- // 23505 -unique_violation
- // 23514 -check_violation
- String getSQLState();

SOLUCIÓ:

```
try {
    /* Tenim ja una connexió c establerta amb la base de dades */
    Statement st = c.createStatement();
    int numP = st.executeUpdate("UPDATE professors "+
      "SET sou = sou + 100 "+
      "WHERE exists (SELECT *"+
            "FROM assignacions a, despatxos d "+
            "WHERE a.dni = professors.dni "+
                 "AND a.modul = d.modul AND a.numero = d.numero "+
                 "AND d.superficie > 20 AND d.modul != 'Omega')");
    if (numP==0)
      {System.out.println("Error");}
    else
      {System.out.println("S'han incrementat "+numP)+" sous;}
    c.commit();
    }
catch (ClassNotFoundException ce) {
    System.out.println ("Error al carregar el driver");}
catch (SQLException se) {
   if(se.getSQLState().equals("23514"))
        System.out.println("La sou no pot ser més gran de 2000");
    else
        System.out.println ("Excepcio: "+ se.getMessage());} }
```

2. (3 punts)

2.1 Considereu la transacció següent: T1: RU(A);W(A);R(A);COMMIT;

a) Justifiqueu si pot existir un horari on intervingui només aquesta transacció i sigui NO serialitzable

b) Justifiqueu si pot existir un horari on intervingui només aquesta transacció i sigui NO recuperable

SOLUCIÓ

No poden existir. Veure definicions de serialitzable i recuperable en les transparències.

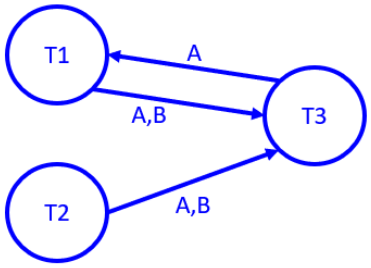
2.2 Considereu l'horari següent:

a) Indiqueu si és serialitzable, justificant breument la resposta a partir del graf de precedències corresponent a l'horari.

b) Indiqueu si es produeix alguna interferència. En cas afirmatiu, digueu quines són, sobre quins gränul i quin és el nivell mínim d'aïllament que podria evitar cadascuna d'elles. En cas negatiu, justifiqueu la resposta.

Temps	T1	T2	T3
10		R(E)	
20	RU(A)		
30		R(A)	
40			RU(A)
50	R(B)		
60			RU(B)
70	W(A)		
80			W(A)
90		R(B)	
100			W(B)
110		RU(C)	
120		W(C)	
130		COMMIT	
140	RU(A)		
150	W(A)		
160	COMMIT		
170			COMMIT

a) No és serialitzable atès que es produeixen cicles en el graf de precedències.



b) Es produeixen tres interferències:

- Entre T1 i T3 es produeix una interferència d'actualització perduda sobre el gränul A (op. 70). El nivell mínim d'aïllament que l'evitaria seria READ UNCOMITTED
- Entre T1 i T3 es produeix una interferència de lectura no confirmada sobre el gränul A (op. 140). El nivell mínim d'aïllament que l'evitaria seria READ COMITTED
- Entre T1 i T3 es produeix un interferència d'anàlisi inconsistent sobre els gränuls A i B (op. 50, 80, 100, 140). El nivell mínim d'aïllament que l'evitaria seria REPETEABLE READ.

2.3 Suposeu ara que sobre l'horari de l'apartat anterior, s'incorpora un mecanisme de control de concurrència basat en reserves S, X i que les transaccions T1 i T3 treballen a un nivell d'aillament REPEATABLE READ i T2 ho fa a un nivell d'aïllament READ COMMITED.

a) Doneu l'horari resultant

b) És serialitzable l'horari resultant? Justifiqueu la resposta.

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

b) Hi ha un anàlisi inconsistent entre T2 i T3 pels grànuls A i B, l'horari no és serialitzable.

2.4 Suposeu l'horari resultant de l'apartat anterior. Considereu si és o no possible afegir una transacció T4 que provoqui una lectura no confirmada sobre el grànul A amb la transacció T1. Si es pot produir la interferència, indiqueu el nivell d'aïllament de la nova transacció T4 i doneu l'horari resultant. Si no és possible justifiqueu la resposta.

L'únic nivell d'aïllament possible per T4 seria el READ UNCOMMITTED. Com que el READ UNCOMMITED no fa LOCKS per les operacions de lectura, es pot provocar la interferència: fent un R(A) a T4 en qualsevol moment després de que T1 actualitzi el grànul A però abans que el torni a actualitzar per segona vegada; o bé fent un R(A) a T4 abans que T1 faci la primera actualització, i fent un abort de T4 desrprés de que T1 hagi fet el commit.

Temps	T1	T2	T3	T4
10		LOCK(E,S)		
20		R(E)		
30		UNLOCK(E)		
40	LOCK(A,X)			
50	RU(A)			
60		LOCK(A,S)		
70			LOCK(A,X)	
80	LOCK(B,S)			
90	R(B)			
100	W(A)			
105				R(A)
106				COMMIT
110	RU(A)			
120	W(A)			
130	COMMIT + UNLOCK(A,B)			
	La resta de files ja no canvien			

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

```
CREATE TABLE lots
       (idLot INT primary key,
       preuLot REAL,
       quantsProductes INT,
       check (preuLot < 100 or quantsProductes > 3));

CREATE TABLE productes
       (idProd INT primary key,
       preuProd REAL NOT NULL,
       idLot INT NOT NULL references lots);

CREATE FUNCTION incrPreuPrBef() RETURNS trigger AS $$
 BEGIN
   if (TG_OP = 'INSERT' and NEW.preuprod<50) then
            NEW.preuprod := NEW.preuprod+5;
            RETURN NEW;
   end if;
   RETURN NULL;
 END; $$ LANGUAGE plpgsql;

 CREATE FUNCTION incrPreuPrAft() RETURNS trigger AS $$
 BEGIN
   if (TG_OP = 'INSERT') then
       update lots
       set preuLot = preuLot + NEW.preuProd,
            quantsProductes = quantsProductes + 1
       where idLot = NEW.idLot;
  else
       update lots
       set preuLot = preuLot + (NEW.preuProd-OLD.preuProd)
       where idLot = NEW.idLot;
   end if;
 RETURN NULL;
 END; $$ LANGUAGE plpgsql;

 CREATE TRIGGER incrPreuTrBef
 BEFORE INSERT OR UPDATE OF preuProd ON productes
 FOR EACH ROW EXECUTE procedure incrPreuPrBef();

 CREATE TRIGGER incrPreuTrAft
 AFTER INSERT OR UPDATE OF preuProd ON productes
 FOR EACH ROW EXECUTE procedure incrPreuPrAft();
```

3.1 Suposeu que s'executen una a una les sentències SQL següents (en cada apartat - a i b - partint de la base de dades buida). Indiqueu quin serà el resultat de les sentències ***select * from lots; select * from productes;*** en el cas de ser afegides i executades just després de cadascun dels inserts de cadascun dels apartats. Justifiqueu breument la resposta, explicant les accions que fa el SGBD a conseqüència de cada inserció.

a)

```
insert into lots values (1,0,0);
insert into productes values(1,30,1);
insert into productes values(2,50,1);
insert into productes values(3,40,1);
insert into productes values(4,35,1);
```

b)

```
begin transaction;
insert into lots values (1,0,0);
insert into productes values(1,30,1);
insert into productes values(2,50,1);
insert into productes values(3,40,1);
insert into productes values(4,35,1);
commit;
```

3.2 Doneu una implementació equivalent al trigger after anterior, amb la mateixa funcionalitat, i que sigui For Each Statement. Justifiqueu quina de les dues maneres d'implementar el trigger s'adequa més als criteris de qualitat de l'assignatura.

3.1.a

Sentència que s'executa	Lots	Productes	Accions del SGBD
insert into lots values (1,0,0);	(1,0,0)		Només s'executa el insert
insert into productes values(1,30,1);	(1,35,1)	(1,35,1)	El trigger before canvia el preu de producte que s'insereix. S'executa l'insert del producte. El trigger after causa que es modifiqui el preu del lot al que pertany el producte.
insert into productes values(2,50,1);	(1,35,1)	(1,35,1)	El trigger before retorna null, i per tant no s'acaba fent la inserció del producte ni tampoc s'executa el trigger after amb la modificació del preu del lot
insert into productes values(3,40,1);	(1,80,2)	(1,35,1) (1,45,1)	El trigger before canvia el preu de producte que s'insereix. S'executa l'insert del producte. El trigger after causa que es modifiqui el preu del lot al que pertany el producte.
insert into productes values(4,35,1);	(1,80,2)	(1,35,1) (1,45,1)	El trigger before canvia el preu de producte que s'insereix. S'executa l'insert del producte. El trigger after causa que es modifiqui el preu En modificar el preu del lot es dóna error de c perquè la fila 1 de lots tindrà el preu més gra no tindrà més de 3 productes.

3.1.b

Sentència que s'executa	Lots	Productes	Accions del SGBD
begin transaction; insert into lots values (1,0,0);	(1,0,0)		S'inicia la transacció i només s'executa el insert de lot.
insert into productes values(1,30,1);	(1,35,1)	(1,35,1)	El trigger before canvia el preu de producte que s'insereix. S'executa l'insert del producte. El trigger after causa que es modifiqui el preu al que pertany el producte.
insert into productes values(2,50,1);	(1,35,2)	(1,35,1)	El trigger before retorna null, i per tant no s'acaba fent la inserció del producte ni tampoc s'executa el trigger after amb la modificació del preu del lot
insert into productes values(3,40,1);	(1,80,2)	(1,35,1) (1,45,1)	El trigger before canvia el preu de producte que s'insereix. S'executa l'insert del producte. El trigger after causa que es modifiqui el preu del lot al que pertany el producte
insert into productes values(3,40,1);			El trigger before canvia el preu de producte que s'insereix. S'executa l'insert del producte. El trigger after causa que es modifiqui el preu del lot, En modificar el preu del lot es dóna error de check perquè la fila 1 de lots tindrà el preu més gran que 100 i no tindrà més de 3 productes. Al ser dins d'una transacció es torna a l'estat abans de l'inici de la transacció.

3.2

```
CREATE FUNCTION incrPreuPrAft() RETURNS trigger AS $$
 BEGIN
       update lots
       set quantsProductes = select count(*)
                             from productes p
                             where p.idLot = lots.idLot,
            preuLot = select sum(p.preuProd)
                      from productes p
                      where p.idLot = lots.idLot;
       RETURN NULL;
 END; $$ LANGUAGE plpgsql;

 CREATE TRIGGER incrPreuTr
 AFTER INSERT OR UPDATE OF preuProd ON productes
 FOR EACH STATEMENT EXECUTE procedure incrPreuPrAft();
```

- Aquesta solució no és incremental, ja que ha de modificar el preu i la quantitat de productes **de tots els lots**, per tant és pitjor que la For Each Row que només modifica el preu dels lots que tenen productes que han canviat de preu.

4. (3 punts) Considereu la taula `R(a,b,c,d)`. L'atribut `a` és clau primària i sobre l'atribut `c` hi ha definida una restricció unique. La taula té 500.000 tuples. Hi ha un índex definit sobre l'atribut `a`, d'ordre 125, 80% ocupació en mitjana, i a cada pàgina de dades hi ha 5 tuples.

Mostreu clarament com heu fet els càlculs que es demanen als apartats següents.

4.1 Quina és la quantitat de pàgines necessàries per emmagatzemar l'índex i la taula?

Taula: 500000/5 = 100000 pàgines de dades

Índex: 2d* 0.8 = 200

fulles: 500000/200 = 2500

Nivell 1: 2500/201 = 13

Nivell 2: 13/201 = 1

2514 pagines d'índex

4.2 Suposeu que hi ha 4000 tuples que compleixen la condició de select següent. Quin seria el cost de resoldre la consulta si l'índex fos no agrupat? I si fos agrupat?

```
SELECT *
FROM R
WHERE a >= 800000
```

Cost no agrupat = h + (F-1) + D

h = $\lceil \log_{201} 500000 \rceil = 3$

F – 1= 4000/200 -1 = 19

Cost = 3 + 19+ 4000

Cost agrupat = h + D

D= 4000/5 = 800

Cost= 3 + 800

4.3 Considereu ara la següent expressió SQL, i calculeu el cost en els dos casos anteriors, és a dir, si l'índex fos agrupat o no agrupat. Justifiqueu la resposta.

```
Exists (SELECT a
        FROM R
        WHERE a >= 800000 )
```

No agrupat : h =3

Agrupat: h = 3

No és necessari accedir a la taula per resoldre la consulta, n'hi ha prou amb saber si hi ha algun valor que compleix la condició.

4.4 Considereu ara la següent expressió SQL:

```
Exists (SELECT a,c
        FROM R
        WHERE a >= 800000 and c = 10)
```

I suposeu que hi ha un índex agrupat per l'atribut `a`, un índex no agrupat per l'atribut `c`, que tots dos índexs tenen ordre d=125 i 80% d'ocupació.

Quin mètode usaríeu per resoldre la consulta? Justifiqueu la resposta mostrant que el mètode escollit és el de menor cost.

El mètode millor seria accedir per l'índex de c, recuperar la única tupla que compliria la condició (hi ha una restricció unique per aquest atribut) i accedir al fitxer de dades per comprovar la condició sobre l'atribut a.

- $Cost_c$: h +1

Els altres mètodes tindrien sempre un cost igual o superior. En el cas pitjor:

- índex per a = h + 19 + 4000
- intersecció RIDs = h + 19 + h + (1-1)