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.
```

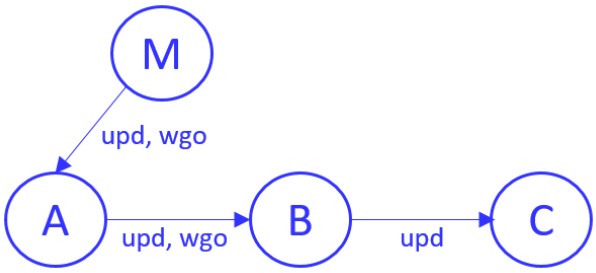
6. **(1 punt)** Suposeu que el propietari de les taules de la base de dades és M.

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.

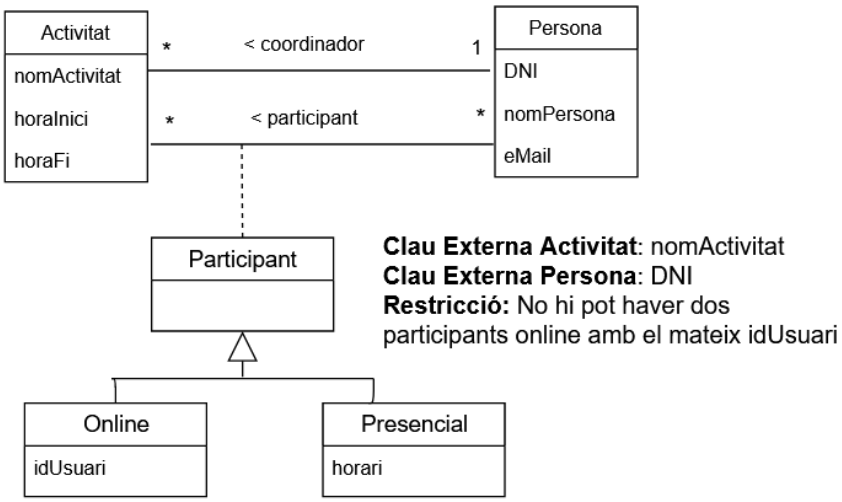
Qualsevol sentència UPDATE que per modificar necessiti consultar algun atribut de la taula partits o de qualsevol altre taula.

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 **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.

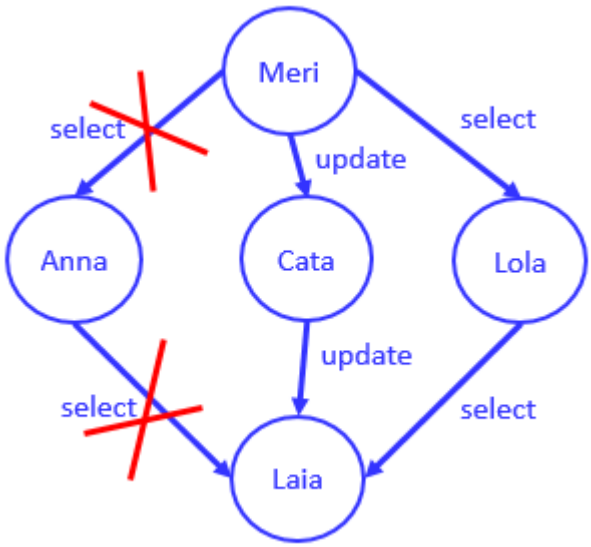
- En cas que la vista sigui actualitzable, doneu una sentència INSERT/DELETE/UPDATE que doni una excepció per al check de la vista.
- En cas que la vista no sigui actualitzable, expliqueu el motiu pel qual no ho és.

```
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.

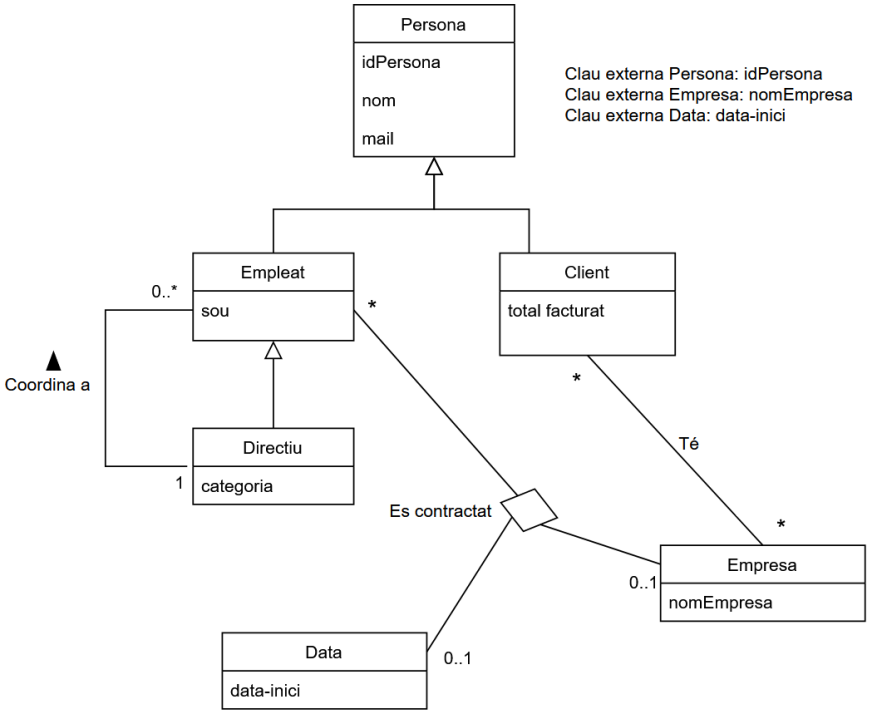
4.2 Suposem que les taules de la base de dades són propietat de la Meri. Considereu les sentències GRANT i REVOKE següents. Doneu els diagrames d'autoritzacions corresponents a l'instant abans d'executar les sentències REVOKE, i a l'instant després d'executar les sentències REVOKE. Justifica el diagrama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

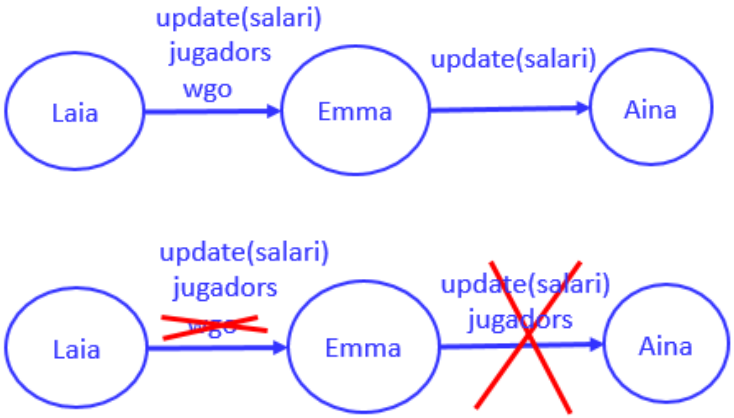
```
Laia: GRANT update(salari) ON jugadors TO Emma WITH GRANT OPTION
Emma: GRANT update(salari) ON jugadors TO Aina
```

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:

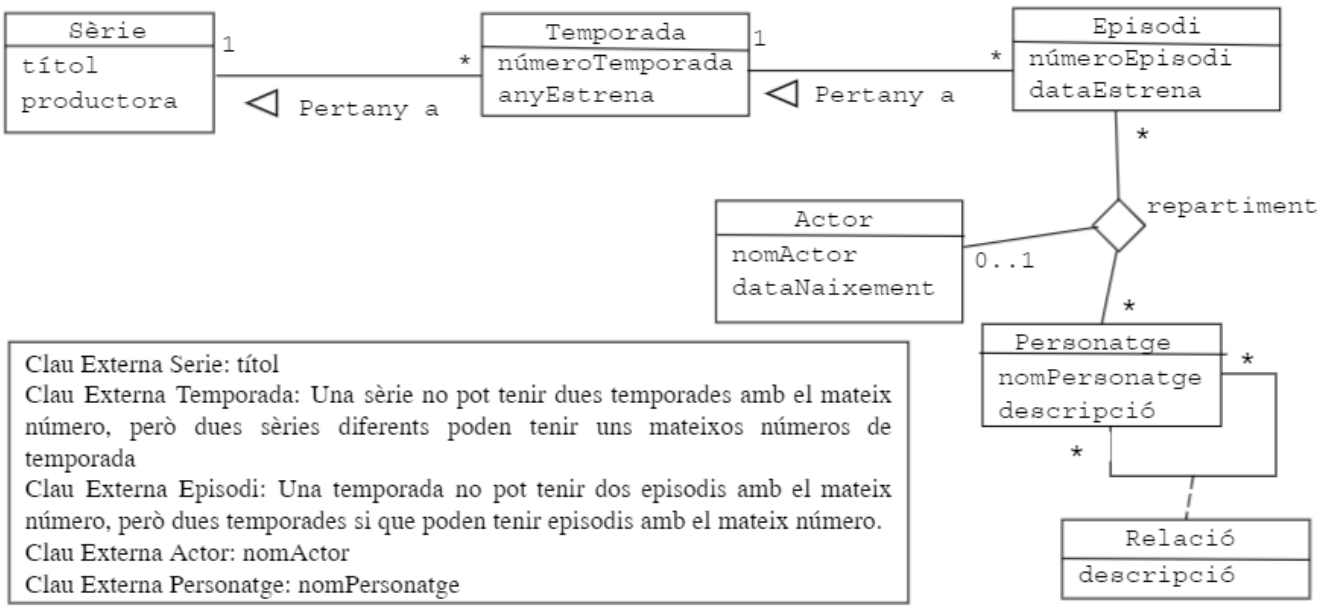
select(dniJugador, salari), update(salari) – jugadors

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

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.

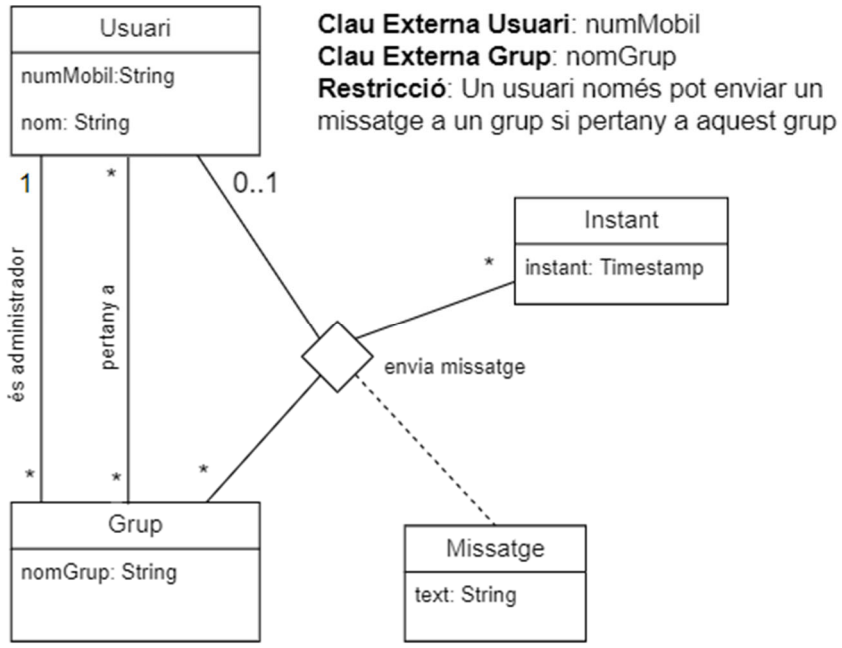
b) Si no es pot, expliqueu per què, i indiqueu com es podria implementar.

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

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

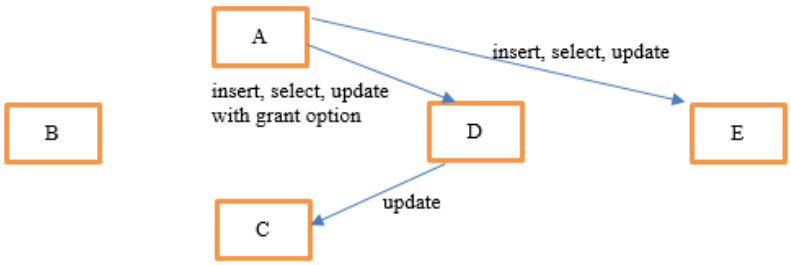
2.2.4. Suposeu ara que la sentència 7 queda substituïda per la que es dóna a continuació:

```
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.

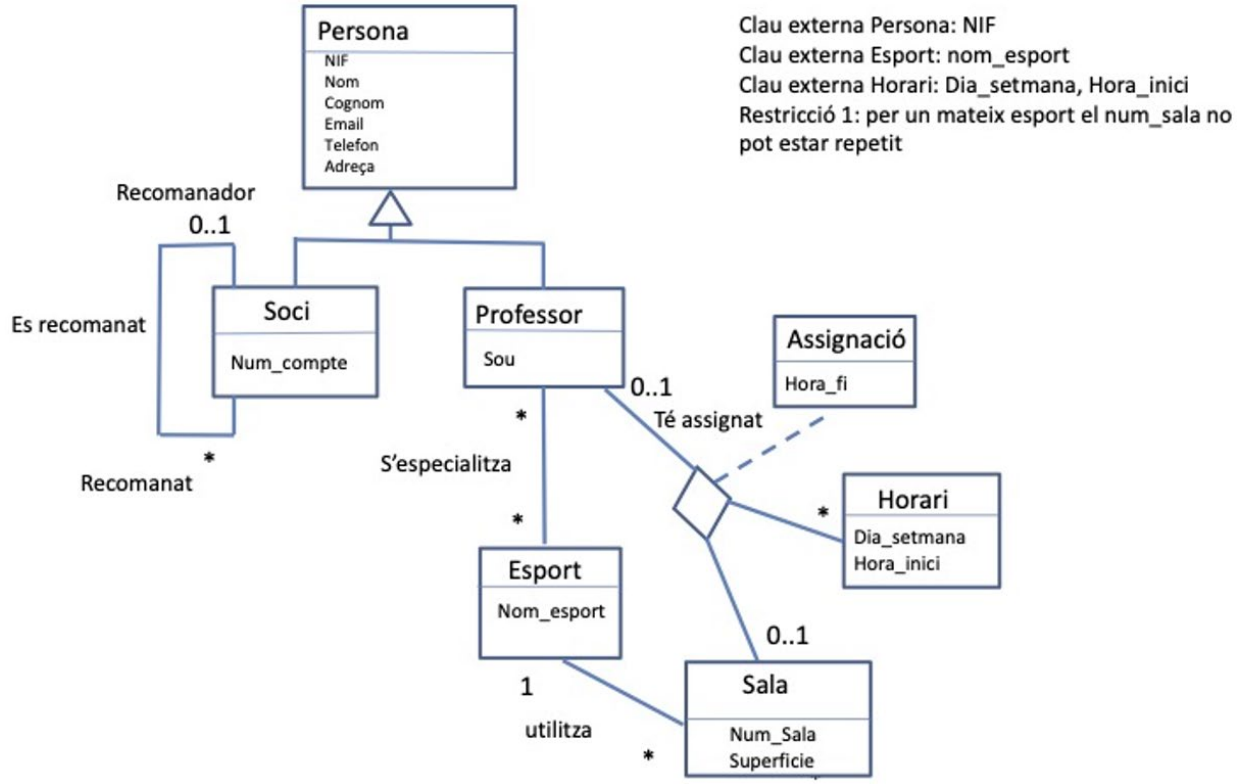
4.2 Doneu una asserció que comprovi que tots els professors de fora de Barcelona fan en total menys de 20 hores de classe per quadrimestre.

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)
```

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.

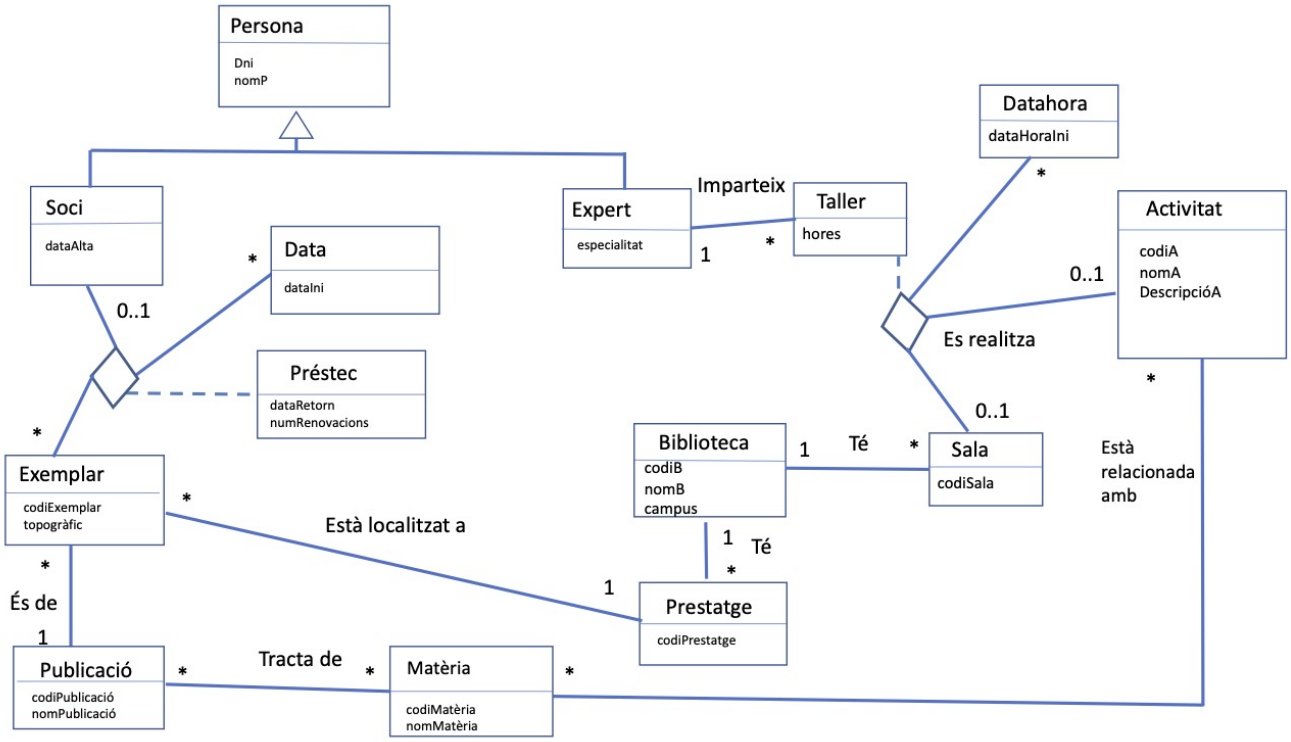
b) Doneu una sentència SQL definida sobre les taules de la base de dades, que compleixi els criteris de qualitat establerts a l'assignatura, i que retorni el mateix resultat que la consulta donada.

```
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
```