Temps: 3 hores

Notes 23 matí **Revisió:** 23 tarda

Cada pregunta s'ha de lliurar en un full separat

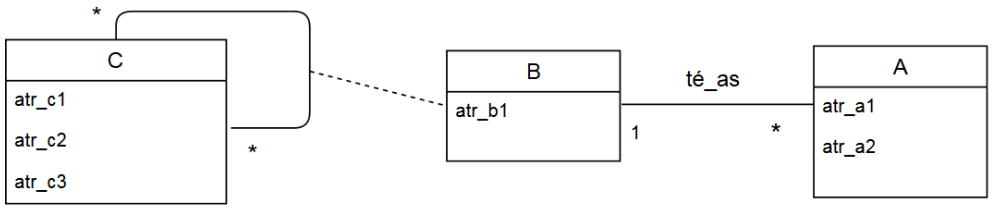
1. **(2.5 punts)** Suposeu una base de dades amb les taules següents:

```
CREATE TABLE usuaris(
  numTelefon char(9),
  nom char(50),
  estat char(100),
  instantDarreraConexio int,
PRIMARY KEY(numTelefon));

CREATE TABLE contactes(
  numTelefonUsuari char(9),
  numTelefonContacte char(9),
  instantDarrerMissatgeEnviat int,
PRIMARY KEY (numTelefonUsuari,numTelefonContacte),
FOREIGN KEY (numTelefonUsuari) REFERENCES usuaris,
FOREIGN KEY (numTelefonContacte) REFERENCES usuaris,
CHECK(numTelefonUsuari <> numTelefonContacte));
// hi ha una fila per cada contacte que té un usuari en el seu
telèfon.

CREATE TABLE missatges(
  numTelefonOrigen char(9),
  numTelefonDesti char(9),
  instantMissatge int,
  textMissatge char(500),
PRIMARY KEY (numTelefonOrigen, instantMissatge),
FOREIGN KEY (numTelefonOrigen) REFERENCES usuaris,
FOREIGN KEY (numTelefonDesti) REFERENCES usuaris,
CHECK(numTelefonOrigen <> numTelefonDesti));
// hi ha una fila per cada missatge enviat des d'un telèfon origen a
un telèfon destí.
```

1.1 Doneu una sentència SQL per obtenir els usuaris que no han enviat cap missatge a usuaris que estan entre els seus contactes i que han rebut més de 100 missatges d'usuaris que no estan als seus contactes. Es vol que surti el número de telèfon i el nom dels usuaris.

1.2 Doneu un seqüència d'operacions en àlgebra relacional per obtenir el text dels missatges que han estat enviats per usuaris que estan als contactes de l'usuari que els ha rebut.

1.3 Traduïu a model relacional el diagrama UML següent. Cal donar taules, atributs, claus primàries, claus foranes i restriccions NOT NULL i UNIQUE que es puguin derivar de l'UML.



Clau externa C: atr_c1, atr_c2

Clau externa A: atr_a1

1.4 Suposeu la base de dades següent que guarda informació sobre quantitat de robatoris.

```
comarca(idComarca, nomComarca)
ciutat(idCiutat, nomCiutat, idComarca)
   {idComarca} referencia comarca
tipus(idTipus, nomTipus)
robatoris(dia,mes,any,idCiutat,idTipus,quantsRobatoris)
   {idCiutat} referencia ciutat
   {idTipus} referencia tipus
```

Escriviu una sentència SQL equivalent a la sentència SQL següent on només s'utilitzin GROUPING SETS.

```
SELECT co.nomComarca, fet.any, t.tipus, SUM(quantsRobatoris)
FROM robatoris fet
          natural inner join ciutat ci
          natural inner join comarca co
          natural inner join tipus t
GROUP BY co.idComarca, CUBE(fet.any, t.idTipus);
```

2. (2.5 punts)

Suposeu que disposem d'una una taula de mesures que guarda les temperatures màximes de cada dia de l'any 2019 i el consum de gelats d'aquell dia en una determinada ciutat

```
CREATE TABLE mesures (
    ciutat         int,
    dia            date,
    temperatura    int,
    consum         int,
    primary key (ciutat, dia)
);
```

Suposeu també que hi ha 12 taules addicionals (mesuresgener, mesuresfebrer, ...) amb els mateixos atributs i restriccions de la taula mesures.

2.1 Tenint en compte que els usuaris només actualitzaran la taula mesures, és vol implementar mitjançant disparadors, el manteniment automàtic de les 12 taules addicionals. Indiqueu i justifiqueu quins triggers caldria definir. Per cada trigger cal indicar:

a) esdeveniment que l'activa (cal indicar esdeveniment, taula i atributs rellevants)

a) before/after

b) row/statement

2.2 Considereu ara una nova situació. Per evitar duplicar la informació només és vol guardar les dades a les taules mensuals, encara que continua existint la taula mesures buida. Considereu que mai és viola la primary key i que disposeu d'una funció mes(dia) que retorna el mes del dia que se li passa per paràmetre. Implementeu el trigger d'inserció associat a la taula mesures, completant i justificant l'esquelet següent:

```
CREATE TRIGGER insert_mesures_trigger
 ... INSERT ON mesures
 FOR EACH ... EXECUTE PROCEDURE mesures_insert_trigger();


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

2.3 Suposeu que s'ha definit la taula `ciutats(ciutat, habitants)` i que l'atribut `mesures.ciutat` és clau forana de `ciutats`. Considereu que la taula mesures té, a l'igual que l'apartat 2.1, totes les tuples de mesures per dia i ciutat. Definiu una vista com una join entre `ciutats i mesures`. És actualitzable aquesta vista? En cas de que ho sigui, justifiqueu la resposta. En cas de que no ho sigui, doneu una extensió de la vista i una operació que mostri que no és actualitzable.

2.4 Suposeu ara que la taula mesures està buida, a l'igual que a l'apartat 2.2, i que només hi ha tuples a les taules mensuals. Redefiniu la vista creada a l'apartat anterior per tal de que estigui definida sobre les taules mensuals i la taula ciutats, i assegurar d'aquesta manera la independència lògica de dades.

3. (2.5 punts)

Calculeu el cost òptim de la consulta següent:

```
SELECT *
FROM R, S
WHERE R.c = 2
      AND R.a = S.a
```

Tenint en compte:

R(a,b,c)
Índex cluster per R.a; Índex hash per R.c; d= 75; ocupació dels arbre: 2/3
Min(c) = 0; Max(c) = 100; NDIST(c) = 1000
Card(R) = 10000; 3 tuples per pàgina; B_R= 3334

S(c, a ,d)
Índex hash per S.c; Índex btree per S.a; Índex cluster per S.d d= 75 pels dos arbres; ocupació dels dos arbres: 2/3
NDIST(a) = 10000
Card(S) = 100000; 3 tuples per pàgina; B_s= 33334

4. (2.5 punts)

Sigui un SGBD sense cap mecanisme de control de concurrència, i suposem que es produeix l'horari següent (les accions s'han numerat per facilitar fer-hi referència):

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

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

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

4.2.1 Quines interferències es produeixen? per cada interferència cal que doneu: nom de la interferència, les dues transaccions implicades i grànuls implicats.

4.2. Suposem ara un mecanisme de control de concurrència basat en reserves S, X, i que les transaccions treballen amb SET TRANSACTION ISOLATION LEVEL READ COMMITED. A més, suposem també que quan dues o més transaccions s'esperen per adquirir una reserva s'atorga a la que porta més temps esperant.

4.2.1. 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, així com també assenyalades les esperes.

4.2.2. Quins horaris serials hi són equivalents?

4.2.3. Quin seria el graf d'espera tot just abans de l'execució del *commit* de T4?