GDINF353 - Bases de données, bases de connaissances
Examen 1ère session, décembre 2014
des solutions

Université Joseph Fourier GRENOBLE UFR IM²AG

1 A propos de théâtre

3 Expression de requêtes

Question 1 (3 points) :
Traduire en algèbre relationnelle puis en SQL les requêtes ci-dessous.

1 { r | ∃ p ∈ LesSpectacles (p.nomS = "Cirque du soleil" ∧ r.numS = p.numS) }
(LesSpectacles:nomS = "Cirque du soleil")[numS]
select numS from LesSpectacles where nomS = "Cirque du soleil"

2 { r | ∃ p ∈ LesReprésentations (∀ t ∈ LesTickets (p.numS ≠ t.numS ∨ p.dateRep ≠ t.dateRep ⟹ r.numS = p.numS ∧ r.dateRep = p.dateRep)) }
LesReprésentations[numS,dateRep] - LesTickets[numS,dateRep]

select numS, dateRep from LesReprésentations
minus
select numS, dateRep from LesTickets
Autre version :
select numS, dateRep from LesReprésentations
where (numS, dateRep) not in (select numS,dateRep from LesTickets)

3 { r | ∃ t ∈ LesTickets (∀ s ∈ LesReprésentations (s.numS = t.numS ⟹ r.numS = s.numS ∧ r.noSérie = t.noSérie)) }
LesTickets[noSérie, numS] / LesReprésentations[numS]

select noSérie from LesTickets
group by noSérie
having count(numS) in (select count(distinct numS) from LesReprésentations)
Autre version :
select noSérie
from (select noSérie, count(numS) as nbSpect from LesTickets) X1
natural join (select count(distinct numS) as nbSpect from LesReprésentations) X2

Question 2 (5 points) :
Exprimer en algèbre relationnelle puis en SQL les requêtes ci-dessous. Pour chacune, suivre avec soin les indications lorsqu'il y en a.

1 Quels sont les numéros des dossiers concernant la vente de places pour exactement deux représentations différentes.
Indication pour la requête 1 : la requête ne doit pas contenir l'opérateur group by ni having. Soit R(noDossier, numS, dateDep) telle que <n, s, d> ∈ R ⟺ le numéro de dossier n concerne un ticket pour la représentation <s, d>. L'expression qui construit R est :
R(noDossier, numS, dateDep) ⟵ LesTickets[noDossier, numS, dateDep]
En SQL :
select distinct noDossier, numS, dateDep from LesTickets

On calcule les numéros de dossier concernant la vente de places pour un moins deux représentations :

R1(n1,s1,d1) ⟵ R ; R2(n2,s2,d2) ⟵ R
AuMoins2 ⟵ (R1(n1=n2 ∧ (d1<>d2 ∨ s1<>s2))*R2)[n1]

En SQL :

```
select distinct R1.noDossier
from LesTickets R1 join LesTickets R2
     on (R1.noDossier=R2.noDossier and (R1.dateDep<>R2.dateDep or R1.numS<>R2.numS)
```

Ceux concernant la vente de places pour au moins 3 représentations :

R3(n3,s3,d3) ⟵ R
AuMoins3 ⟵ R3(n3=n2 ∧ (d1<>d2 ∧ d2<>d3 ∧ d1<>d3 ∨ s1<>s2 ∧ s2<>s3 ∧ s1<>s3))
  * (R1 (n1=n2 ∧ (d1<>d2 ∨ s1<>s2)) * R2)
Res ⟵ AuMoins3[n1]

En SQL :

```
select distinct R1.noDossier
from LesTickets R1 join LesTickets R2
     on (R1.noDossier=R2.noDossier and (R1.dateDep<>R2.dateDep or R1.numS<>R2.numS)
     join LesTickets R3
     on (R3.noDossier=R2.noDossier and (R1.dateDep<>R2.dateDep or R3.dateDep<>R2.dateDep
        or R1.dateDep<>R3.dateDep or R1.numS<>R2.numS or R3.numS<>R2.numS
        or R1.numS<>R3.numS))
```

La requête finale est :

AuMoins3 − AuMoins2

En SQL :

```
select R1.noDossier
from LesTickets R1 join LesTickets R2
     on (R1.noDossier=R2.noDossier and (R1.dateDep<>R2.dateDep or R1.numS<>R2.numS)
     join LesTickets R3
     on (R3.noDossier=R2.noDossier and (R1.dateDep<>R2.dateDep or R3.dateDep<>R2.dateDep
        or R1.dateDep<>R3.dateDep or R1.numS<>R2.numS or R3.numS<>R2.numS
        or R1.numS<>R3.numS))
minus
select R1.noDossier
from LesTickets R1 join LesTickets R2
     on (R1.noDossier=R2.noDossier and (R1.dateDep<>R2.dateDep or R1.numS<>R2.numS)
```

2 Quels sont les spectacles dont la configuration de la salle est la plus petite ?
Indication pour la requête 2 : la requête ne doit pas contenir l'opérateur in ni sa négation not in.
On introduit la relation Conf (nomS, nbSièges) telle que <s, n> ∈ Conf ⟺ la configuration de la salle pour le spectacle s contient n sièges. L'expression de Conf est impossible en algèbre relationnelle.
En SQL :

```
select nomS, count(*) as nbSièges
from (select distinct nomS, noPlace, noRang
      from LesSpectacles natural join LesConfigurations
      natural join LesZones natural join LesPlaces) X group by nomS
```

Finalement, il suffit de prendre dans Conf le nom du spectacle associé à la valeur minimale de nbSièges :

```
select nomS
from (select nomS, nbSièges from Conf) X
     join (select min(nbSièges) as minNb from Conf) Y
     – <m> ap Y <=> m est la valeur minimale de nombres de sièges toutes configuration confondues
     on (X.nbSièges = Y.minNb)
```

3 Calculer le montant total de chaque vente donnée par son numéro de dossier.

```
select noDossier, sum(prix)
from LesTickets natural join LesPlaces natural join LesZones natural join LesCatégories
group by noDossier
```

4 Donner les numéros de dossier des ventes de places pour toutes les représentations du spectacle *Le plus petit cirque du monde*.
Dans l'algèbre relationnelle :

```
LesTickets[noDossier, numS, dateRep]
   / (LesTickets * LesSpectacles:nomS = 'Le plus petit cirque du monde')[numS, dateRep]
```

```
select noDossier from LesTickets natural join LesSpectacles
where nomS = 'Le plus petit cirque du monde'
group by noDossier, numS
having count(distinct dateRep) in
           (select count(dateRep) from LesSpectacles natural join LesReprésentations
            where nomS = 'Le plus petit cirque du monde')
```

5 Calculer le nombre de sièges de la salle pour chaque nom de catégorie.
Attention : bien qu'intuitive l'expression ci-dessous est incorrecte

```
select nomC, count (noPlace, noRang) from ....
group by nomC
```

```
select nomC, count(*)
from LesZones natural join LesPlaces
group by nomC
```

6 Pour chaque spectacle et pour chacune de ses représentations, donner combien places sont encore disponibles (un entier ≥ 0).

```
select numS, dateRep, count (*) as nbPlaces
from LesRepresentations natural left outer join
     (select numS, dateRep, noPlace, noRang
      from LesRepresentations cross join LesPlaces
           natural join LesZones natural join LesConfigurations
      where (numS, dateRep, noPlace, noRang) not in
                  (select numS, dateRep, noPlace, noRang
                   from LesTickets))
group by numS, dateDep;
```

4 Normalisation

Question 3 (2 points) :
Indiquer la ou les clefs de la relation R. Tout résultat non justifié sera considéré comme faux.

On effectue la fermeture des dépendances :

login $\longrightarrow$ login, nom, prénom, courriel, mdp
courriel $\longrightarrow$ login, nom, prénom, mdp, courriel

noDossier $\longrightarrow$ login, nom, prénom, mdp, courriel, noDossier

On en déduit :

{ noDossier }$^+$ = { login, nom, prénom, mdp, courriel, noDossier }
{ login }$^+$ = { login, nom, prénom, mdp, courriel }
{ courriel }$^+$ = { login, nom, prénom, mdp, courriel }

de plus, noDossier $\longrightarrow$ login, nom, prénom, mdp, courriel, noDossier est élémentaire. Par suite, noDossier est clef dans la relation.

Question 4 (3 points) :
En quelle forme normale est la relation R ? Si nécessaire, générer un schéma en 3e forme normale en appliquant l'algorithme de décomposition. Tout résultat non justifié sera considéré comme faux.

La relation est en 1ère forme normale (par construction). Elle est en 2e FN, car tous les attributs non clef (login, nom, prénom, mdp, courriel) sont en DFE avec la clef. Elle n'est pas en 3e FN car nom (entre autre) est transitivement dépendant de noDossier :

noDossier $\longrightarrow$ login
login $\longrightarrow$ nom
login $\not\longrightarrow$ noDossier

On choisit de décomposer R, selon la DF login $\longrightarrow$ login, nom, prénom, courriel, mdp. On obtient, R1 (noDossier, login) et R2 (login, nom, prénom, courriel, mdp). noDossier est la clef de R1, R1 est en 3e FN. D'autre part, login et courriel sont deux clefs candidates de R2, R2 est 3e FN.

5 Datalog

Question 5 (1 point) :
Ecrire les requêtes suivantes, en calcul prédicatif puis en Datalog, si et seulement si elles sont exprimables en calcul prédicatif :

1 Le nom de tous les professeurs de la troisième C.

{r | ∃ p ∈ Professeur (r.nomProf=P.nomProf ∧ p.classe='3eC') }
Prof3eC (X) :- Professeur (X, Y, 3eC)

2 Le nom de tous les cours donnés le Lundi à 8 heures.

/ Le nom du cours est la matière : */*
{ m | ∃ c ∈ Cours (m.matière=c.matière ∧ c.Jour ='lundi' ∧ c.heure = 8) }
NomCours8H (X) :- Cours (lundi, 8, Y, Z, X)
/ Le nom du cours n'existe pas, on retourne toutes les informations associées : */*
{ c | c ∈ Cours (c.Jour ='lundi' ∧ c.heure = 8 }
NomCours8H (X, Y, Z) :- Cours (lundi, 8, Y, Z, X)

3 Le nom de tous les professeurs de Mathématiques.

{ r | ∃ p ∈ Professeur (r.nomProf = p.nomProf ∧ p.matière = "Mathématiques") }
ProfMaths(X) :- Professeur(X, Mathématiques, Y)

Question 6 (2 points) :
On dispose des règles suivantes, pour s'assurer que la base est cohérente avec les besoins du collèges. Ces règles sont censées ne renvoyer aucun tuple si la base de données est cohérente. Indiquer les contraintes vérifiées par ces règles, si les règles sont sûres, et correctement écrites, et, le cas échéant, comment les modifier pour qu'elles le soient.

```
mystere1(X,Y,Z) :- Cours(X,Y,Z,A,B), Cours(X,Y,Z,C,D), not(A=C).
mystere2(Y) :- not(ProfesseurPrincipal(X,Y)).
mystere3(X,Y,Z) :- Cours(X,Y,A,B,C), Cours(X,Y,D,E,F),
   Professeur(Z,C,B), Professeur(Z,F,E), not(A=D).
mystere4(A,B) :- Professeur(X,A,B), Professeur(Y,A,B).
```

- Mystère1 : règle sûre. Vérifie que deux classes différentes n'ont pas cours au même moment dans la même salle.
- Mystère2 : règle non sûre. Pour vérifier que toutes les classes ont un professeur principal, on écrira par exemple : mystere2(Y) :- Eleve(E, Y), not(ProfesseurPrincipal(X,Y)).
- Mystère 3 : règle sûre. Vérifie qu'un même professeur n'est pas attendu dans deux salles différentes au même moment.
- Mystère 4 : règle sûre. mais renvoie toujours quelque chose. Pour vérifier qu'une classe n'a pas deux professeurs différents dans la même matière, il faut rajouter (not X = Y).

Question 7 (5 points) :
Ecrire les règles suivantes, en calcul prédicatif puis en Datalog, si et seulement si elles sont exprimables en calcul prédicatif. Les règles de Datalog doivent être des règles sûres.

1 Tous les cours donnés par le professeur Fernier.
$\{ c \mid \exists p \in Professeur\ (p.nomProf = "Fernier" \wedge p.classe = c.classe \wedge p.matière = c.matière) \}$
```
CoursFernier(X,Y,Z,A,B) :- Cours(X,Y,Z,A,B), Professeur('Fernier', A, B).
```

2 Le nom de tous les professeurs n'étant pas professeur principal.
$\{ r \mid \exists p \in Professeur\ (r.nomProf = p.nomProf \wedge not\ \exists pp \in ProfPrincipal\ (r.nomProf = pp.nomProf) \}$
```
ProfNonPrincipal(X) :- Professeur(X, A, B), not(ProfPrincipal(X, C)).
```

3 La matière enseignée par le moins de professeurs. Impossible.

4 Le nom des classes qui ont cours tous les matins à 8 heures.
$\{ r \mid \forall c1 \in Cours\ (\exists c2 \in Cours\ (\ c1.jour = c2.jour \wedge r.classe = c2.classe \wedge c2.heure\ )) \}$
```
PasTousLesMatins(A) :- Cours(X,Y,Z,A,B), not (Cours(X,'8.00',Z,A,B) ).
TousLesMatins(A) :- Cours(X,Y,Z,A,B), not (PasTousLesMatins(A)).
```

5 Le nom de l'élève de 4èmeB qui a eu la meilleure note en Mathématiques.
$\{ r \mid \exists\ e \in Eleve, \exists n \in NoteTrimestre1,$
$(r.nomEleve = n.nomEleve \wedge n.nomEleve = e.nomEleve \wedge e.classe = "4emeB" \wedge n.matière = "Maths" \wedge$
$(\ \nexists e2 \in Eleve, \exists\ n2 \in NoteTrimestre1,$
$(n2.nomEleve = e.nomEleve \wedge e2.classe = "4emeB" \wedge n2.note > n.note))) \}$
```
AUneMeilleureNoteEn4emeB(A) :-
   Eleve(A,'4emeB'), Eleve(B, '4emeB'), NoteTrimestre1(A, N1, 'Maths', X),
   NoteTrimestre2(B, N2, 'Maths', Y), N1 < N2.
Maximum4emeB(A) :- Eleve(A, '4emeB'), not(AUneMeilleureNoteEn4emeB(A)).
```

6 La moyenne du deuxième trimestre de l'élève Chambonnier. Inexprimable en calcul prédicatif.

Question 8 (1 point) :
On veut créer la règle NoteTrimestre(NomEleve, NumeroTrimestre, Matière, Note, Appreciation). Où NuméroTrimestre vaut 1, 2 ou 3.

Ecrire cette règle puis écrire son graphe de dépendance (avec les + et les -).

```
NoteTrimestre(E, 1, M, N, A) :- NoteTrimestre1(E, M, N, A).
NoteTrimestre(E, 2, M, N, A) :- NoteTrimestre2(E, M, N, A).
NoteTrimestre(E, 3, M, N, A) :- NoteTrimestre3(E, M, N, A).
```

Graphe de dépendance :

NoteTrimestre1 ——–>
NoteTrimestre2 ——–> NoteTrimestre (les arcs ont tous des +)
NoteTrimestre3 ——–>