Université Libre de Bruxelles
Faculté des Sciences Appliquées
École Polytechnique

ELEC378 – Systèmes Distribués d'Information
Année Académique 2006 – 2007
Examen de première session

1 Bases de données temporelles

Une compagnie de consultance utilise la base de données temporelle suivante dans une application de gestion de ressources humaines:

- Client(#client, nom, prénom, adresse, téléphone)
- Projet(#projet, #client, nomProjet, localisation, description, fromDate, toDate)
  #client references Client.#client
- Employé(#employé, nom, prénom, adresse, téléphone)
- Travaille(#employé, #projet, pourcentage, fromDate, toDate)
  #employé references Employé.#employé
  #Projet references Projet.#projet
  $1\% < \text{pourcentage} < 100\%$
- Dirige(#employé, #projet, fromDate, toDate)
  #employé references Employé.#employé
  #projet references Projet.#projet

Ecrivez en SQL standard les requêtes suivantes :

(1) Donner le nom, prénom et adresse des employés qui dirigent actuellement un projet qui est actif et dont la fin du projet arrive avant le 31 décembre 2007.
(2) Donner les numéros des projets localisés à Liège tels que tous les employés qui y ont travaillé pendant l'année 2006 le faisaient à 100%.
(3) Donner les numéros des employés qui ont été au chômage au moins 6 mois entre deux assignations en tant que travailleur ou en tant que directeur de projet.
(4) Donner pour chaque employé le numéro d'employé et l'intervalle de temps pendant lequel il a travaillé dans un projet comme employé ou comme directeur.
(5) Donner le numéro des projets dans lesquels chaque employé qui y a travaillé n'a jamais diminué son pourcentage.

2 Bases de données actives

Considérez la base de données relationnelle suivante qui est utilisée pour l'analyse des ventes d'une société.

- Product (ProductNo, ProductName, Description, FromDate, ToDate)
- Store (StoreNo, Name, Address, FromDate, ToDate)
- SalesDistrict (DistrictName, NoEmployees, Representative, FromDate, ToDate)
- StoreSalesDistrict (StoreNo, DistrictName, FromDate, ToDate)
  StoreNo references Store(StoreNo)
  DistrictName references SalesDistrict(DistrictName)
- Sales (StoreNo, ProductNo, Date, Quantity, Amount)
  StoreNo references Store(StoreNo)
  ProductNo references Product(ProductNo)

Remarquez que les produits, les magasins et les districts ont un cycle de vie representé par les attributs FromDate et ToDate. Egalement, l'association entre les magasins et les districts est temporelle, comme l'indiquent les attributs FromDate et ToDate.

La base de données doit vérifier les contraintes suivantes :

(1) Les lignes de la table StoreSalesDistrict possédant le même StoreNo et DistrictName ne peuvent pas se chevaucher dans le temps.
(2) L'intervalle d'une instance de l'association StoreSalesDistrict doit être inclus dans le cycle de vie de vie du magasin auquelle est reliée.
(3) Un produit ne peut être vendu (ne peut apparaître dans la table Sales) que pendant son cycle de vie.
(4) Un magasin est attaché à chaque instant de son cycle de vie à exactement un district (néanmoins le district peut changer au cours du temps).
(5) Un district peut avoir plus de 10 magasins attachés (à chaque instant) que si il a au moins 5 employés.

Pour chacune de ces 5 contraintes écrire en Starburst ou en SQL Server une règle active qui se déclenche lors d'une violation et réalise les actions de réparation nécessaires. Dans les 5 règles au moins une doit se déclencher avec chacun des événements INSERT, DELETE et UPDATE. Commentez chacune des règles.

3 Bases de données objet

La base de données d'une fédération de football utilise le schéma suivant.

```
class Joueur (extent Joueurs key (noJoueur) ) {
   attribute String noJoueur;
   attribute String prénom;
   attribute String nom;
   attribute String adresse;
   attribute date dateNaissance;
   relationship list<Match> matchs inverse Match::joueurs;
   relationship Equipe équipe inverse Equipe::joueurs;
}
class Equipe (extent Equipes key (noEquipe) ) {
   attribute String noEquipe;
   attribute String nom;
   attribute String entraineur;
   attribute Joueur capitaine;
   attribute String stade;
   relationship list<Joueurs> joueurs inverse Joueurs::équipe;
   relationship list<Match> matchs inverse Match:équipes;
}
class Match (extent Matchs key (noMatch) ) {
   attribute String noMatch;
   attribute String stade;
   attribute Date date;
   attribut Score { unsigned short goals1,
     unsigned short goals2 } score;
   relationship list<Equipe> équipes inverse Equipe::matchs;
   relationship set<Joueur> joueurs inverse Joueur::matchs;
}
```

Soient les requêtes suivantes :

(1) Donner le prénom et le nom des joueurs qui ont joué dans tous les matchs de leur équipe.
(2) Donner le prénom et le nom des joueurs qui n'ont gagné aucun des matchs dans lesquels ils ont joué.
(3) Donner pour chaque match deux ensembles contenant le prénom et le nom des joueurs des deux équipes qui ont participé dans le match.

(4) Donner pour chaque équipe le résume de la saison en ordre croissant de date. Exemple de sortie:

Real Madrid

Equipe	Date	Bp	Bc
Barcelone	1/2/2007	3	0
Sévilla	8/2/2007	1	1
Valence	17/2/2007	2	3
Real Saragosse	24/2/2007	2	2

(5) Donner le classement des équipes. Exemple de sortie:

No	Equipe	J	G	N	P	Bp	Bc	Pts
1	Real Madrid	37	22	7	8	63	39	73
2	Barcelone	37	21	10	6	73	32	73
3	Sévilla	37	21	8	8	64	34	71
4	Valence	37	20	5	12	54	39	65
5	Real Saragosse	37	16	11	10	54	42	59
6	Villarreal	37	17	8	12	47	44	59

Pour calculer le total de points, les matchs gagnés comptent pour 3 points et les matchs nuls compent pour 1 point. Ainsi pour la première équipe $(22 \times 3) + 7 = 73$. Lorsque le nombre de points est identique pour deux équipes (comme pour les deux premières équipes ci-dessus) les autres critères dans les colonnes G, N, P, Bp et Bc sont utilisés dans cet ordre pour les départager.

4 Bases de données déductives

La STIB utilise la base de données déductive suivante pour la gestion de son réseau de transports en commun :

- Ligne (<u>numéroLigne</u>, typeLigne)
  typeLigne ∈ { 'métro', 'bus', 'tram' }
- Itinéraire (<u>numéroLigne, destination, séquence</u>, arrêt)
  numéroLigne ⊆ Ligne(numéroLigne)
  séquence ∈ $\{1, 2, \ldots\}$
- Horaires(<u>numéroLigne, destination, séquence, typeHoraire, heurePassage</u>)
  (numéroLigne, destination, séquence) ⊆ Ligne(numéroLigne, destination, séquence)
  typeHoraire ∈ { 'Lundi-Vendredi', 'Samedi', 'Dimanche-Fêtes' }
  heurePassage est de type Time (comme en SQL)

Des exemples de ces tables sont les suivantes.

Ligne

1A	métro
1B	métro
2	métro
3	tram
18	tram
19	tram
...	...

Itinéraire

2	Simonis	1	Délacroix
2	Simonis	2	Clémenceau
2	Simonis	3	Gare de Midi
2	Simonis	4	Hôtel des Monnaies
2	Simonis	5	Porte de Namur
2	Simonis	6	Arts-Loi
2	Simonis	7	Rogier
2	Simonis	8	Simonis
...	...	...	...

Horaires

2	Simonis	1	Samedi	05:28
2	Simonis	2	Samedi	05:30
2	Simonis	3	Samedi	05:31
2	Simonis	4	Samedi	05:34
2	Simonis	5	Samedi	05:37
2	Simonis	6	Samedi	05:39
2	Simonis	7	Samedi	05:43
2	Simonis	8	Samedi	05:48
2	Simonis	1	Samedi	05:45
2	Simonis	2	Samedi	05:46
2	Simonis	3	Samedi	05:47
2	Simonis	4	Samedi	05:50
2	Simonis	5	Samedi	05:52
2	Simonis	6	Samedi	05:55
2	Simonis	7	Samedi	05:58
2	Simonis	8	Samedi	06:02
...	...	...	...	...

Notez que dans les tables Itinéraire et Horaires chaque ligne est dédoublée par rapport à la destination. Par exemple, la ligne “2 Simonis” va de Délacroix à Simonis et la ligne “2 Délacroix” va de Simonis à Délacroix.

Ecrivez en Datalog les requêtes suivantes.

(1) Donner le nom des arrêts qui sont desservis par des lignes de métro, de bus et de tram.

(2) Donner par ligne et destination le nombre total d’arrêts.

(3) Donner pour chaque ligne et destination le temps de trajet minimum par type d’horaire. Vous pouvez utiliser la fonction timediff(time1,time2) qui donne comme résultat une valeur de type time résultat de time1-time2.

(4) Donner pour chaque arrêt le numéro des lignes qui ont un terminus dans l’arrêt, en ordre croissant. Exemple de sortie:

Arrêt: De Brouckère
Lignes: 29, 47, 60, 65, 66, 71
Arrêt: Bourse
Lignes: 34, 48, 95, 96

(5) Donner par ligne, par destination et par type d’horaire le nombre total de transports qui partent par heure. Vous pouvez utiliser la function hour() qui extrait l’heure d’une valeur de type Time. Exemple de sortie:

Ligne 2 Simonis	
Lundi-Vendredi	5h: 2, 6h: 13, 7h: 22, 8h: 22, 9h: 14, ...
Samedi	5h: 2, 6h: 12, 7h: 15, 8h: 12, 9h: 10, ...
Dimanche	5h: 3, 6h: 7, 7h: 8, 8h: 6, 9h: 4, ...

1 Bases de données temporelles

(1) Donner le nom, prénom et adresse des employés qui dirigent actuellement un projet qui est actif et dont la fin de projet arrive avant le 31 décembre 2007.

```
select nom, prénom, adresse
from Employé e, Dirige d, Projet p
where e.#employé = d.#employé  and d.#projet = p.#projet
and p.fromDate <= current_timestamp and current_timestamp < p.toDate
and p.toDate < 21/12/2007
```

(2) Donner les numéros des projets localisés à Liège tels que tous les employés qui y ont travaillé pendant l'année 2006 le faisaient à 100%.

```
select p.#projet
from Projet p
where p.localisation = 'Liège' and
and not exists ( select * from Travaille t
    where p.#projet = t.#projet and t.fromDate < 1/1:2007
    and 1/1/2006 < t.toDate and t.pourcentage <> 100% )
```

(3) Donner les numéros des employés qui ont été au chômage au moins 6 mois entre deux assignations en tant que travailleur ou en tant que directeur de projet.

```
create view Temp(#employé, fromDate, toDate) as
  select #employé, fromDate, toDate from Travaille
  union
  select #employé, fromDate, toDate from Dirige
/* coalescing de la vue précédente */
create view TempCoal(#employé, fromDate, toDate) as
  select distinct F.#employé, F.fromDate, L.toDate
  from Temp F, Temp L
  where F.fromDate < L.toDate
  and F.#employé = L.#employé
  and not exists ( select * from Temp M
    where M.#employé = F.#employé
    and F.fromDate < M.fromDate and M.fromDate <= L.toDate
    and not exists ( select * from Temp T1
      where T1.#employé = F.#employé
      and T1.fromDate < M.fromDate and M.fromDate <= T1.toDate ) )
  and not exists ( select * from Temp T2
    where T2.#employé = F.#employé
    and ( (T2.fromDate < F.fromDate and F.fromDate <= T2.toDate)
       or (T2.fromDate <= L.toDate and L.toDate < T2.toDate) ) )
/* recherche des périodes de 6 mois entre deux assignations */
select distinct #employé
from TempCoal T1, TempCoal T2
where T1.#employé = T2.#employé
and not exists ( select * from TempCoal T3
  where T1.#employé = T3.#employé
  and T1.toDate < T3.fromDate and T3.fromDate < T2.fromDate )
and datediff(month,T2.fromDate,T1.toDate) > 6
```

Cette question peut être résolue par la requête suivante qui est plus simple, mais on a besoin du coalescing pour la question suivante.

```
create view Temp (#employé, fromDate, toDate) as (
  select #employé, fromDate, toDate from Travaille
  union
  select #employé, fromDate, toDate from Dirige)
select distinct T1.#employé
from Temp T1, Temp T2
WHERE T1.#employé = T2.#employé
and 6 > datediff(month, T1.toDate, T2.fromDate)
and not exists ( select *
  from Temp T3
  where T3.#employé = T2.#employé
  and T3.fromDate < T2.fromDate and T3.toDate > T1.toDate )
```

(4) Donner pour chaque employé le numéro d'employé et l'intervalle de temps pendant lequel il a travaillé dans un projet comme employé ou comme directeur.

```
select *
from TempCoal
```

(5) Donner le numéro des projets dans lesquels chaque employé qui y a travaillé n'a jamais diminué son pourcentage.

```
select #projet
from Projet P
where not exists ( select * from Travaille T1, Travaille T2
  where T1.#employé = T2.#employé
  and T1.#projet = P.#projet and T2.#projet = P.#projet
  and T1.toDate < T2.fromDate and T1.pourcentage > T2.pourcentage )
```

2 Bases de données actives

(1) Les lignes de la table StoreSalesDistrict possédant le même StoreNo et DistrictName ne peuvent pas se chevaucher dans le temps.

```
create trigger DisjIntervals on StoreSalesDistrict
after insert, update as
if exists ( select *
  from Inserted I, StoreSalesDistrict D
  where I.StoreNo = D.StoreNo and I.DistrictName = D.DistrictName
  and I.FromDate < D.ToDate and D.FromDate < I.ToDate )
begin
  raiserror 13000 'Constraint Violation:
    Overlapping Intervals in table StoreSalesDistrict'
  rollback
end
```

(2) L'intervalle d'une instance de l'association StoreSalesDistrict doit être inclus dans le cycle de vie de vie du magasin auquelle est reliée.

```
create trigger IntInStore on StoreSalesDistrict
after insert, update as
if exists ( select *
  from Inserted I, Store S
  where not ( S.FromDate <= I.FromDate and I.ToDate <= S.ToDate )
begin
  raiserror 13000 'Constraint Violation: Interval in StoreSalesDistrict
    must be included in the lifespan of its Store'
  rollback
end
```

(3) Un produit ne peut être vendu (ne peut apparaître dans la table Sales) que pendant son cycle de vie.

```
create trigger SalesProduct on Sales
after insert, update as
if exists ( select *
  from Inserted I, Product P
  where I.ProductNo = P.ProductNo
  and I.Date < P.FromDate or P.ToDate < I.Date )
begin
  raiserror 13000 'Constraint Violation:
    Date in Sales must be included in the lifespan of Product'
  rollback
end
```

(4) Un magasin est attaché à chaque instant de son cycle de vie à exactement un district (néanmoins le district peut changer au cours du temps).

```
/* idem que question 2 mais sans tester l'égalité de DistrictName */
create trigger DisjIntervals on StoreSalesDistrict
after insert, update as
if exists ( select *
```

```
  from Inserted I, StoreSalesDistrict D
  where I.StoreNo = D.StoreNo
  and I.FromDate < D.ToDate and D.FromDate < I.ToDate )
begin
  raiserror 13000 'Constraint Violation: At each instant
    a Store cannot be associated to more than one StoreSalesDistrict'
  rollback
end
```

(5) Un district peut avoir plus de 10 magasins attachés (à chaque instant) que si il a au moins 5 employés.

```
/* It is necessary to first compute the temporal count in a view */
create view AssChanges(DistrictName,Day) as
  select distinct DistrictName,FromDate from StoreSalesDistrict
  union
  select distinct DistrictName,ToDate from StoreSalesDistrict
create view AssPeriods(DistrictName,FromDate,ToDate) as
  select DistrictName,P1.Day,P2.Day
  from AssChanges P1, AssChanges P2
  where P1.DistrictName=P2.DistrictName
  and P1.Day<P2.Day and not exists (
    select * from AssChanges P3
    where P1.DistrictName=P3.DistrictName
    and P1.Day<P3.Day and P3.Day<P2.Day )
create view TempCount(DistrictName,NbStores,FromDate,ToDate) as
  select DistrictName,count(*),P.FromDate,P.ToDate
  from StoreSalesDistrict S, AssPeriods P
  where S.DistrictName=P.DistrictName
  and S.FromDate<=P.FromDate and P.ToDate<=S.ToDate
  group by P.FromDate, P.ToDate
create view TempCountCoal(DistrictName,NbStores,FromDate,ToDate) as
  /* coalesce previous view */

create trigger DistrictNbStores on SalesDistrict
after insert, update as
if exists ( select *
  from Inserted I, TempCountCoal T
  where I.DistrictName = T.DistrictName
  and I.NoEmployees < 5 and T.NbStores > 10
)
begin
  raiserror 13000 'Constraint Violation: A district must have at
    least 5 employees to have more than 10 associated stores'
  rollback
end
```

3 Bases de données objet

(1) Donner le prénom et le nom des joueurs qui ont joué dans tous les matchs de leur équipe.

```
select prénom, nom
from Joueur j
where forall m in j.équipe.matchs : j in m.joueurs
```

(2) Donner le prénom et le nom des joueurs qui n'ont gagné aucun des matchs dans lesquels ils ont joué.

```
select prénom, nom
from Joueur j
where forall m in j.équipe.matchs :
( j.équipe = m.match.équipes[0] and goals1 < goals2 ) or
( j.équipe <> m.match.équipes[0] and goals1 > goals2 )
```

(3) Donner pour chaque match deux ensembles contenant le prénom et le nom des joueurs des deux équipes qui ont participé dans le match.

```
select JoueursDomicile: ( select nom, prénom from m.joueurs j
    where j.équipe = équipes[0] ),
  JoueursExtérieur: ( select nom, prénom from m.joueurs j
    where j.équipe = équipes[1] )
from Match m
```

(4) Donner pour chaque équipe le résume de la saison en ordre croissant de date.

```
select e.nom, résumé: (
  select
    case m.équipes[0] = e
      m.équipes[1].nom, m.date, m.goals1, m.goals2
    else
      m.équipes[0].nom, m.date, m.goals2, m.goals1
    end
  from e.matchs m
  order by résumé.date )
from Equipe e
```

Autre solution sans case

```
create view classement as
  select e.nom, résumé: (
    select m.équipes[1].nom, m.date, m.goals1, m.goals2
    from e.matchs m
    where m.équipes[0] = e
    union
    select m.équipes[0].nom, m.date, m.goals2, m.goals1
    from e.matchs m
    where m.équipes[1] = e )
  from Equipe e
select c.nom, ( select * from c.résumé order by c.résumé.date )
from classement c
```

(5) Donner le classement des équipes.

```
select e.nom,
  J: count(e.matchs),
  G: count( select * from e.matchs m
            where (e=m.équipes[0] and goals1 > goals2)
               or (e=m.équipes[1] and goals1 < goals2) ),
  N: count( select * from e.matchs m
            where goals1 = goals2 ),
  P: count( select * from e.matchs m
            where (e=m.équipes[0] and goals1 < goals2)
               or (e=m.équipes[1] and goals1 > goals2) ),
  Bp:  sum( select goals1 from e.matchs m
            where e=m.équipes[0]
            union
            select goals2 from e.matchs m
            where e=m.équipes[1] ),
  Bc:  sum( select goals2 from e.matchs m
            where e=m.équipes[0]
            union
            select goals1 from e.matchs m
            where e=m.équipes[1] ),
  Pts: sum( select 3 from e.matchs m
            where (e=m.équipes[0] and goals1 > goals2)
               or (e=m.équipes[1] and goals1 < goals2)
             union
             select 1 from e.matchs m
             where goals1 = goals2 ),
from Equipe e
order by J, G, N, P, Bp, Bc, Pts
```

4 Bases de données déductives

(1) Donner le nom des arrêts qui sont desservis par des lignes de métro, de bus et de tram.

```
MBT(A) :- Itinéraire(L1,_,_,A), Itinéraire(L2,_,_,A),
  Itinéraire(L3,_,_,A), Ligne(L1,_,'métro'),
  Ligne(L2,_,'bus'), Ligne(L3,_,'tram')).
```

(2) Donner par ligne et destination le nombre total d'arrêts.

```
NbArrêts(L,D,N) :- Itinéraire(L,D,N,D).
```

(3) Donner pour chaque ligne et destination le temps de trajet minimum par type d'horaire.

```
Between(L,D,N,T,HD,HA) :- Horaires(L,D,N,T,HA1), HD<HA1, HA1<HA.
DuréeTrajet(L,D,T,Dur) :- Horaires(L,D,1,T,HD), Horaires(L,D,N,T,HA),
  NbArrêts(L,D,N), ~Between(L,D,N,T,HD,HA), Dur=timediff(HA,HD).
Smaller(L,D,T,Dur) :- DuréeTrajet(L,D,T,Dur1), Dur1<Dur.
MinTrajet(L,D,T,Dur) :- DuréeTrajet(L,D,T,Dur), ~Smaller(L,D,T,Dur).
```

(4) Donner pour chaque arrêt le numéro des lignes qui ont un terminus dans l'arrêt, en ordre croissant.

```
Terminus(L,A) :- Itinéraire(L,A,_,A).
GreaterTerm(L,A) :- Terminus(L1,A), L1>L.
BetweenTerm(A,L1,L2) :- Terminus(L3,A), L1<L3, L3<L2.
ArrêtTerminus1(A,[L]) :- Terminus(L,A), ~GreaterTerm(L,A).
ArrêtTerminus1(A,[L1|[L2|_]]) :- ArrêtTerminus1(A,[L2|_]),
  Terminus(L1,A), L1<L2, ~BetweenTerm(A,L1,L2).
ArrêtTerminus(A,L) :- ArrêtTerminus1(A,L), ~ArrêtTerminus1(A,[_|L]).
```

(5) Donner par ligne, par destination et par type d'horaire le nombre total de transports qui partent par heure.

```
DépartsHeure(L,D,T,H,HM) :- Horaires(L,D,1,T,HM), H=hour(HM).
SmallerDepHeure(L,D,T,H,HM) :- DépartsHeure(L,D,T,H,HM1), HM1<HM.
BetweenDepHeure(L,D,T,H,HM1,HM2) :- DépartsHeure(L,D,T,H,HM3),
  HM1<HM3, HM3<HM2.
NbDépartsHeure(L,D,T,H,HM,1) :- DépartsHeure(L,D,T,H,HM),
  ~SmallerDepHeure(L,D,T,H,HM).
NbDépartsHeure(L,D,T,H,HM,N) :- NbDépartsHeure(L,D,T,H,HM1,N1),
  N=N1+1, DépartsHeure(L,D,T,H,HM), HM>HM1,
  ~BetweenDepHeure(L,D,T,H,HM,HM1).
GreaterNbDépartsHeure(L,D,T,H,N) :- NbDépartsHeure(L,D,T,H,_,N1),
  N1>N.
NbDépartsHeure(L,D,T,H,N) :- NbDépartsHeure(L,D,T,H,_,N),
  ~GreaterNbDépartsHeure(L,D,T,H,_,N).
```