

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 représenté 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 auquel 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 Matches 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::matches;
    relationship set<Joueur> joueurs inverse Joueur::matches;
}
```

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 résumé de la saison en ordre croissant de date. Exemple de sortie:

Real Madrid				
Equipe	Date	Bp	Bc	
Barcelone	1/2/2007	3	0	
Séville	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	Séville	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 comptent 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 (numéroLigne, typeLigne)
typeLigne \in { 'métro', 'bus', 'tram' }
- Itinéraire (numéroLigne, destination, séquence, arrêt)
numéroLigne \subseteq Ligne(numéroLigne)
séquence \in {1, 2, ...}
- Horaires(numéroLigne, destination, séquence, typeHoraire, heurePassage)
(numéroLigne, destination, séquence) \subseteq Ligne(numéroLigne, destination, séquence)
typeHoraire \in { 'Lundi-Vendredi', 'Samedi', 'Dimanche-Fêtes' }
heurePassage est de type Time (comme en SQL)

Des exemples de ces tables sont les suivantes.

Ligne				Horaires				
1A	métro			2	Simonis	1	Samedi	05:28
1B	métro			2	Simonis	2	Samedi	05:30
2	métro			2	Simonis	3	Samedi	05:31
3	tram			2	Simonis	4	Samedi	05:34
18	tram			2	Simonis	5	Samedi	05:37
19	tram			2	Simonis	6	Samedi	05:39
...	...			2	Simonis	7	Samedi	05:43
				2	Simonis	8	Samedi	05:48
Itinéraire				2	Simonis	1	Samedi	05:45
2	Simonis	1	Délaacroix	2	Simonis	2	Samedi	05:46
2	Simonis	2	Clémenceau	2	Simonis	3	Samedi	05:47
2	Simonis	3	Gare de Midi	2	Simonis	4	Samedi	05:50
2	Simonis	4	Hôtel des Monnaies	2	Simonis	5	Samedi	05:52
2	Simonis	5	Porte de Namur	2	Simonis	6	Samedi	05:55
2	Simonis	6	Arts-Loi	2	Simonis	7	Samedi	05:58
2	Simonis	7	Rogier	2	Simonis	8	Samedi	06:02
2	Simonis	8	Simonis
...					

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 Délaacroix à Simonis et la ligne “2 Délaacroix” va de Simonis à Délaacroix.

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 fonction `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, ...

```