UNIVERSITE LIBRE DE BRUXELLES 23 juin 2008
Faculté des Sciences Appliqués

INFO-H-511 Systèmes Distribués d'Information
Examen de première session

---

1 XQuery

Soit une base de données XML musicale suivant l'exemple ci-dessous.

```
<music>
  <song>
    <artist>Eumir Deodato</artist>
    <title>Carly and Carole</title>
    <album>Prelude</album>
  </song>
  <song>
    <artist>Serge Gainsbourg</artist>
    <artist>Jane Birkin</artist>
    <title>Je t'aime... Moi non plus</title>
    <album>Jane Birkin/Serge Gainsbourg</album>
  </song>
  ...
  <album>
    <artist>Eumir Deodato</artist>
    <title>Prelude</title>
    <year>1973</year>
    <genre>Jazz</genre>
  </album>
  <album>
    <artist>Serge Gainsbourg</artist>
    <artist>Jane Birkin</artist>
    <title>Jane Birkin/Serge Gainsbourg</title>
    <year>1969</year>
    <genre>Rock</genre>
  </album>
  ...
  <genre>
    <name>Classical</name>
    <genre>
      <name>Chamber music</name>
    </genre>
    <genre>
      <name>Orchestral music</name>
    </genre>
  </genre>
  <genre>
    <name>Non-classical</name>
    <genre>
      <name>Rock</name>
    </genre>
    <genre>
      <name>Pop</name>
    </genre>
    <genre>
      <name>Jazz</name>
    </genre>
  </genre>
  ...
</music>
```

On vous demande d'écrire en XQuery les requêtes suivantes :

(1) La liste des noms des artistes de la base de données.
(2) Les chansons chantées par au moins deux personnes.
(3) La liste des chansons sorties entre 1998 et 2008.
(4) Le nombre de chansons groupées par super-genre (Classical, Non-Classical, ...)
(5) Le ou les albums qui contiennent le plus de chansons.

2 Bases de données temporelles

Une compagnie d'assurances utilise la base de données temporelle suivante :

- Client(NoClient, Prénom, Nom, Adresse, NoTel)
- Voiture(NoPlaque, Modèle, Année, NoClient, DateDébut, DateFin)
  NoClient references Client.NoClient
- Police(NoPolice, TypePolice, NoPlaque, Montant, DateDébut, DateFin)
  NoPlaque references Voiture.NoPlaque
- Accident(NoAcc, NoPlaque, Localisation, Description, DateHeure)
  NoPlaque references Voiture.NoPlaque
- Réparation(NoAcc, Description, Coût, DateDébut, DateFin)
  NoAcc references Accident.NoAcc

où

- La table Client contient les personnes ayant une voiture couverte par une police d'assurance
- La table Voiture contient l'information décrivant les voitures. La temporalité décrit la période pendant laquelle un client est propriétaire d'une voiture.
- La table Police contient l'information sur les polices d'assurance des voitures. La temporalité décrit la période pendant laquelle une police est valide sur une voiture.
- La table Accident contient les accidents survenus sur les voitures. La temporalité décrit l'instant dans lequel l'accident est survenu.
- La table Réparation contient les réparations qui ont été faites sur les voitures après un accident. La temporalité décrit la période de temps pendant laquelle la réparation a été réalisée. Remarquez qu'une réparation peut être réalisée plusieurs mois après l'accident, même si la police d'assurance a expiré entretemps.

On vous demande d'écrire en SQL standard les requêtes suivantes :

(1) Le numéro des accidents qui n'ont pas été couverts par une police d'assurance.
(2) Le prénom, nom et adresse des clients dont une de ses polices d'assurance a expiré le mois dernier (Mai 2008).
(3) Le prénom, nom et adresse des clients dont toutes les voitures qu'ils possèdent actuellement sont couvertes par une police d'assurance.
(4) Pour chaque police d'assurance donner le numéro et le coût total de toutes ses réparations.
(5) Pour les accidents survenus pendant l'année 2007 le temps moyen que les clients ont attendu pour la réparation correspondante.

3 Bases de données objet

Une société spécialisée dans des produits bio utilise la base de données suivante.

```
struct AddressType {
   attribute String Address;
   attribute String City;
   attribute String PostalCode;
   attribute String Country;
}
class Order (extent Orders key (OrderID) ) {
   attribute String OrderID;
   attribute Date OrderDate;
   attribute Float Freight;
   attribute String ShipName;
   attribute AddressType ShipAddress;
   relationship Employee OrderEmployee inverse Employee::Orders;
   relationship Customer OrderCustomer inverse Customer::Orders;
   relationship list<OrderLine> OrderLines inverse OrderLines::Orders;
}
class Employee (extent Employees key (EmployeeID) ) {
   attribute String EmployeeID;
   attribute String FirstName;
   attribute String LastName;
   attribute Float Salary;
   attribute AddressType Address;
   relationship list<Order> Orders inverse Order:OrderEmployee;
}
class Customer (extent Customers key (CustomerID) ) {
   attribute String CustomerID;
   attribute String CompanyName;
   attribute String ContactName;
   attribute AddressType Address;
   relationship list<Order> Orders inverse Order:OrderCustomer;
}
class OrderLine (extent OrderLines ) {
   attribute Float UnitPrice;
   attribute Int Quantity;
   attribute Float Discount;
   relationship Order RelatedOrder inverse Order::OrderLines;
   relationship Product ProductSold inverse Product::OrderLines;
}
class Product (extent Products key (ProductID) ) {
   attribute String ProductID;
   attribute String ProductName;
   attribute String Supplier;
   attribute Float UnitPrice;
   relationship list<OrderLine> OrderLines inverse OrderLine::ProductSold;
}
```

On vous demande d'écrire en OQL les requêtes suivantes :

(1) Le nom des employés qui gagnent plus que n'importe quel employé qui habite à Londres.
(2) Le nom des clients qui ont acheté tous les produits achetés par la société dont l'identificateur est 'LAZYK'
(3) Le nom des employés qui vendent les produits de plus de 7 fournisseurs.
(4) Spécifier pour chaque employé, l'identificateur, le nom ainsi que le total des ventes réalisées, ordonné par identificateur d'employé. Le total des ventes est calculé en additionant le total des lignes des commandes. Le total d'une ligne de commande est calculé à partir du prix unitaire, la quantité et la remise, cette dernière est exprimée en pourcentage.
(5) Le nom des clients et le nom des produits, pour les clients qui ont acheté de ce produit 5 fois plus que la moyenne des ventes de ce produit parmi tous les clients.

4 Bases de données actives

Une compagnie de consultance utilise la base de données relationnelle suivante dans la gestion de ses projets:

- Projet(CodeProjet, NomProjet, Description, Budget, Département)
- Tâche(CodeTâche, NomTâche, NbJours, Description)
- Employé(NoEmployé, Nom, Prénom, Adresse)
- ProjetTâche(CodeProjet, NoTâche, CodeTâche, DateDébut, Responsable)
  CodeProjet references Projet.CodeProjet
  CodeTâche references Tâche.CodeTâche
  Responsable references Employé.NoEmployé
  NoTâche $\in \{1, 2, \ldots\}$ donne le numéro de séquence de la tâche dans le projet.
- Ressource(NoRessource, CodeRessource, CoûtRessource)
- TâcheRessource(CodeTâche, NoRessource)
  CodeTâche references Tâche.CodeTâche
  NoRessource references Ressource.NoRessource

La base de données doit vérifier les contraintes suivantes :

(1) Un employé ne peut être responsable de toutes les tâches d'un projet.
(2) Les tâches assignées à un projet ne peuvent pas se chevaucher dans les dates.
(3) Les numéros des tâches d'un projet doivent constituer une séquence continue {1, 2, ...} sans trous.
(4) La somme des coûts des ressources utilisées dans les tâches d'un projet ne peut pas dépasser le budget du projet.
(5) Le nombre de jours qu'un employé est responsable des tâches d'un projet ne peut pas dépasser le 50% de la somme totale des jours du projet.

Pour chacune de ces 5 contraintes écrire en SQL Server un trigger 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.