UNIVERSITE LIBRE DE BRUXELLES 2 juin 2009
Faculté des Sciences Appliqués

INFO-H-415 Systèmes Distribués d'Information
Examen de première session

---

1 XQuery

Soit la base de données IMDB (Internet Movie Database) suivant l'exemple ci-dessous.

```
<imdb>
  <movies>
    <film>
      <director>Marco Tullio Giordana</director>
      <title>Sanguepazzo</title>
      <year>2008</year>
      <writer>Leone Colonna</writer>
      <writer>Marco Tullio Giordana</writer>
      <cast>
        <artist>Monica Bellucci</artist>
        <character>Luisa Ferida</character>
      </cast>
      <cast>
        <artist>Luca Zingaretti</artist>
        <character>Osvaldo Valenti</character>
      </cast>
      ...
    </film>
    <film>
      <director>Florian Henckel von Donnersmarck</director>
      <title>Das Leben der Anderen</title>
      <year>2006</year>
      ...
      <prize>
        <year>2007</year>
        <result>Winner</result>
        <award>Oscar</award>
        <category>Best Foreign Language Film of the Year</category>
        <recipient>Germany</recipient>
      </prize>
      <prize>
        <year>2008</year>
        <result>Nominated</result>
        <award>BAFTA Film Award</award>
        <category>Best Director</category>
        <recipient>Florian Henckel von Donnersmarck</recipient>
      </prize>
      ...
      <userRating>
        <user>joeblack@gmail.com</user>
        <rating>10</rating>
      </userRating>
    </film>
  </movies>
  <user>
    <email>joeblack@gmail.com</email>
    <sex>male</sex>
    <yearOfBirth>1975</yearOfBirth>
    <postalCode>6969</postalCode>
    <country>USA</country>
  </user>
  ...
</imdb>
```

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

1. La liste des noms des réalisateurs de la base de données.
2. La liste des films où le réalisateur est également acteur.
3. Pour chaque acteur, donner sa filmographie en ordre décroissant de l'année.
4. Pour les BAFTA Film Award, donner par année la liste des films récompensés et nominés, pour toutes les catégories.
5. Pour chaque film, classifier les avis des utilisateurs par tranche d'âge: moins que 18 ans, de 18 à 29 ans, de 30 à 44 ans, 45 ans ou plus. Pour chaque catégorie d'utilisateur, donner la moyenne des avis.

2 Bases de données temporelles

Une société de ventes utilise la base de données temporelle suivante pour son entrepôt de données :

- Product(ProductNo, ProductName, Description, CategoryNo, FromDate, ToDate)
  CategoryNo references Category(CategoryNo)
- Category(CategoryNo, CategoryName, Description)
- Store(StoreNo, StoreName, Address, FromDate, ToDate)
- District(DistrictNo, DistrictName, Representative, FromDate, ToDate)
- StoreDistrict(StoreNo,DistrictNo,FromDate, ToDate)
  StoreNo references Store(StoreNo)
  DistrictNo references District(DistrictNo)
- Client(ClientNo, ClientName, BirthDate, SalaryRange)
- Sales(ProductNo,ClientNo,StoreNo,Date, Quantity, Amount)
  ProductNo references Product(ProductNo)
  ClientNo references Client(ClientNo)
  StoreNo references Store(StoreNo)

où

- La table Product contient les données des produits, leur catégorie, ainsi que l'interval de temps pendant lequel le produit a été vendu
- La table Category contient les données des catégories
- La table Store contient les données des magasins, ainsi que l'interval de temps pendant lequel le magasin est ou était opérationnel
- La table District contient les données des districts, où un district regroupe géographiquement un ensemble de magasins
- La table StoreDistrict contient les assignations d'un magasin à un district, cette assignation peut varier au cours du temps
- La table Client contient les données des clients
- La table Sales contient les données des ventes réalisés par produit, client, magasin et date.

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

1. Le nombre de produits distincts vendus dans le district 'Ixelles' en Mai 2009.
2. Donner le nom des magasins qui ont changé de district pendant l'année 2008. On suppose qu'un magasin peut ne pas être attaché à un district pendant un laps de temps.
3. Donnez le nom des magasins dont une des associations à un district n'est pas couverte par les cycles des vies du magasin et du district.
4. Donnez pour chaque magasin les intervals pendant lesquels il n'est pas lié à un district.
5. Agréger la table Sales par district, en tenant compte que l'assignation d'un magasin à un district est temporelle. Le résultat est une table dont la structure est la suivante

   Sales(ProductNo,ClientNo,**DistrictNo**,Date, Quantity, Amount).

   N.B. Ceci qui correspond à l'opération de roll-up dans les entrepôts de données.

3 Bases de données actives

Une ligue de football amateur utilise la base de données relationnelle suivante:

- Player(PlayerNo,FirstName,LastName,Address,TeamNo,NbMatchsPlayed)
  TeamNo references Team(TeamNo)
- Team(TeamNo,TeamName,Captain)
  Captain references Player(PlayerNo)
- Practices(TeamNo,Stadium)
  TeamNo references Team(TeamNo)
- Match(MatchNo,Stadium,LocalTeamNo,VisitorTeamNo,Goals1,Goals2)
  LocalTeamNo references Team(TeamNo)
  VisitorTeamNo references Team(TeamNo)
- Plays(PlayerNo,MatchNo)
  PlayerNo references Player(PlayerNo)
  MatchNo references Match(MatchNo)

La base de données doit vérifier les contraintes suivantes :

1. Le capitaine d'une équipe doit être joueur de l'équipe
2. Un joueur ne joue que dans des matchs de son équipe
3. L'équipe locale d'un match doit s'entraîner dans le stade du match
4. Une équipe a au plus 22 joueurs inscrits
5. L'attribut NbMatchsPlayed est un attribut dérivé à partir de la relation Plays

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.

4 Bases de données objet

Une université utilise la base de données suivante pour décrire les catalogue de cours.

```
class AcadProgram (extent AcadPrograms key (AcadProgramID) ) {
   attribute String AcadProgramID;
   attribute String AcadProgramName;
   attribute int ECTS;
   relationship list<AcadYear> Years inverse AcadYear::Program;
}
class AcadYear (extent AcadYears key (AcadYearID) ) {
   attribute String AcadYearID;
   attribute String AcadYearName;
   attribute int ECTS;
   relationship AcadProgram Program inverse AcadProgram::Years;
   relationship list<CourseYear> Courses inverse CourseYear::Year;
   relationship list<Student> Students inverse Student::Year;
}
class Course (extent Courses key (CourseID) ) {
   attribute String CourseID;
   attribute String CourseName;
   attribute String ProfessorName;
   attribute int ECTS;
   relationship list<CourseYear> Years inverse CourseYear::Course;
   relationship list<StudentCourse> Students inverse StudentCourse::Course;
}
class CourseYear (extent CourseYears) {
   attribute Boolean Mandatory;
   relationship Course Course inverse Course::Years;
   relationship AcadYear Year inverse AcadYear::Courses;
}
class Student (extent Students key (StudentID) ) {
   attribute String StudentID;
   attribute String FirstName;
   attribute String LastName;
   attribute BirthDate Address;
   attribute String Address;
   relationship AcadYear Year inverse AcadYear:Students;
   relationship list<StudentCourse> Courses inverse StudentCourse:Student;
}
class StudentCourse (extent StudentCourses) {
   attribute int Grade;
   relationship Course Course inverse Course::Years;
   relationship Student Student inverse Student::Courses;
}
```

où

1. Le nom des années d'études dans lequel le cours 'Swarm Intelligence' est obligatoire.

```
select distinct A.AcadYearName
from Courses C, C.Year CY, CY.AcadYear A
where Course.CourseName = 'Swarm Intelligence'
and CY.Mandatory = True
```

2. Le nom des étudiants qui ont suivi tous les cours donnés par le professeur 'Hugues Bersini'

```
select StudentName
from Students S
where forall C in ( select * from Courses where ProfessorName =
  'Hugues Bersini' ) : exists SC in S.Courses : SC.Course = C
```

3. Le nom des programmes qui ont au moins 10 professeurs.

```
select AcadProgramName
from AcadPrograms A
where count(
  select distinct C.ProfessorName
  from A.Years AY, AY.Courses CY, CY.Course C ) >= 10
```

4. Spécifier pour chaque année d'études, la liste des cours ainsi que la moyenne des côtes que les étudiants ont obtenu pour le cours.

```
select AcadYearName, courses: (
  select C.CourseName, avg: (
    select avg(Grade)
    from C.Students )
  from AY.Courses CY, CY.Course C )
from AcadYears AY
```

5. Pour chaque année d'études, grouper les étudiants par grade (LPGD, GD, D, S, AJ) par rapport à la moyenne des côtes qu'ils ont obtenus.

```
select AcadYearName, deliberation: (
  select S.StudentName, avg: ( select avg(SC.Grade) from S.Courses SC )
  from AY.Students S
  group by
    LPGD : avg >= 18
    GD : avg >= 16 and avg < 18
    D : avg >= 14 and avg < 16
    S : avg >= 10 and avg < 14
    AJ : avg < 10 )
from AcadYears AY
```