TP3 - M2 GIL - BD NoSQL pour le BigData
Manipulation du type XML

16 novembre 2017
Lina Soualmia

1 Préambule

L'objectif de ce TP est de manipuler des documents XML à l'aide d'un SGBD relationnel. Vous pouvez utiliser Oracle (v XE), PostgreSQL...voire MySQL.

Les commandes ci-après sont valables pour Oracle. Il conviendra de les adapter au SGBDR.
Pour plus de lisibilité des résultats des requêtes, paramétrez l'affichage avec les commandes suivantes :

```
set long 1000000;
set pagesize 100;
```

Voici une liste de liens à visiter +++ :

- Oracle® XML DB Developer's Guide :
  - http://download.oracle.com/docs/cd/B19306_01/appdev.102/b14259/toc.htm
- Introduction Oracle/XML :
  - http://www.oracle.com/technology/pub/articles/quinlan-xml.html
  - http://download.oracle.com/docs/cd/B19306_01/appdev.102/b14259/xdb05sto.htm#g1070409
- Liens généraux :
  - http://www.oracle.com/technology/tech/xml/index.html
  - http://www.oracle.com/pls/db102/portal.portal_db?selected=7
- Opérations XMLType :
  - http://download.oracle.com/docs/cd/B19306_01/appdev.102/b14258/t_xml.htm#ARPLS369
- Interroger XML avec Oracle Text :
  - http://download.oracle.com/docs/cd/B10500_01/appdev.920/a96620/xdb11sea.htm

2 Interrogation XML - Exemple

Exécutez les instructions suivantes :

2.1 Chargement de documents

Chargez le document XML `cd_catalog.xml` dans une table `catalog` ayant la structure suivante :

```
drop table catalog;

create table catalog (
    filename VARCHAR2(32) primary key,
    xml      xmltype
    );

create or replace directory xml_dir as '/home/oracle/xml';

insert into catalog values (
        'cd_catalog.xml',
        XMLType(BFILENAME('XML_DIR', 'cd_catalog.xml'),
        nls_charset_id('AL32UTF8')));
```

2.2 Requêtes SQL

Afficher le document chargé avec :

```
select xml from catalog;
```

Combien de nuplets (lignes) contient la table catalog ?

```
select count(xml) from catalog;
```

2.3 extract

Extraction de fragments XML avec XPath : `extract(XMLType, ExprXPath) : XMLType`
Extraire des éléments :

```
select extract(xml,'//ARTIST') from catalog;
```

Déterminez le nombre de lignes de la réponse précédente.

2.4 XMLSequence

Décomposition du résultat d'une expression XPath :

— "XMLSequence" transforme le résultat en séquence d'objets avec des valeurs de type XMLType (SQL3)
— "table" transforme la séquence en table d'objets (SQL3) de type XMLType
— "value(objet)" retourne la valeur de l'objet de type XMLType

Génération d'une séquence XML :

```
select XMLSequence(extract(xml,'//ARTIST')) from catalog;
```

Transformation d'une séquence XML en table d'objets qu'on peut interroger :

```
select value(artist)
from catalog, table(XMLSequence(extract(xml,'//ARTIST'))) artist;
```

ou

```
select artist.*
from catalog, table(XMLSequence(extract(xml,'//ARTIST'))) artist;
```

Déterminez le nbre de n-uplets de la réponse.
Créez une table cd avec un n-uplet par CD :

```
drop table cd;

create table cd(xml xmltype);

insert into cd(xml)
select cd.*
from catalog, table(XMLSequence(extract(xml, '//CD'))) cd;
```

Pour afficher tous les CDs :

```
select * from cd;
```

2.5 extractValue

Extraction de la valeur du résultat d'une expression XPath :

`extractValue(XMLType,ExprXpath)` : `VARCHAR2` (l'expression doit retourner une seule valeur)

Sélection avec `SQL` :

```
select *
from cd
where extractValue(xml,'/CD/COMPANY') like '%RCA%';
```

ou (avec `extract`) :

```
select *
from cd
where extract(xml,'/CD/COMPANY/text()') like '%RCA%';
```

2.6 existsNode

Test d'existence d'un nœud :

`existsNode(XMLType, ExprXpath)` : `Boolean`

Sélection avec `existsNode` :

```
select *
from cd
where existsNode(xml,'/CD[COMPANY="RCA"]')=1;
```

2.7 XQuery

Interrogation avec `XQuery` de valeurs de type `XMLType` :

```
select XMLQuery('
     for $p in //CD
     where $p/COMPANY = "Columbia"
     return <a>{$p/ARTIST/text()}</a>'
     passing xml returning content)
from catalog;
```

2.8 Vues Relationnelles sur documents XML

Création d'une vue relationnelle :

```
create or replace view cdtable(title, artist, country, company, price, yr) as
select  extractvalue(xml,'/CD/TITLE'),
        extractvalue(xml,'/CD/ARTIST'),
        extractvalue(xml,'/CD/COUNTRY'),
        extractvalue(xml,'/CD/COMPANY'),
        extractvalue(xml,'/CD/PRICE'),
        extractvalue(xml,'/CD/YEAR')
from cd;

desc cdtable;

select * from cdtable;
```

Création d'une vue avec `XQuery` et `XMLTable` :
http://docs.oracle.com/cd/B19306_01/appdev.102/b14259/xdb_xquery.htm#CBAHGAJI

```
create or replace view rca_artists(compagnie,artiste) as
select a.Company, a.Artiste
from catalog,
    XMLTABLE('/CATALOG/CD' PASSING catalog.xml COLUMNS
                Company VARCHAR2(10) PATH '/CD/COMPANY',
                Artiste VARCHAR2(20) PATH '/CD/ARTIST') a
where a.Company like 'R%';
```

Schéma de la vue :

```
desc rca_artists;
```

Données de la vue :

```
select * from rca_artists;
```

3 Génération XML

http://download.oracle.com/docs/cd/B19306_01/appdev.102/b14259/xdb13gen.htm#g1047191

3.1 `XMLElement`, `XMLAttributes`, `XMLForest`

`XMLElement` `XMLAttributes` : http://docs.oracle.com/cd/B19306_01/appdev.102/b14259/xdb13gen.ht
`XMLForest` : http://docs.oracle.com/cd/B19306_01/appdev.102/b14259/xdb13gen.htm#i1028160
Création d'éléments :

```
select XMLElement('SOCIETE', company) from cdtable;
```

ou

```
select XMLForest(company as 'SOCIETE') from cdtable;
```

Création d'un élément complexe :

```
select  XMLElement('SOCIETE',
        XMLElement('NOM', COMPANY),
        XMLElement('ARTISTE', XMLElement('NOM', ARTIST)))
from    cdtable;
```

Création d'attributs :

```
select  XMLElement('SOCIETE', XMLAttributes(COMPANY as 'ID'),
        XMLElement('ARTISTE', XMLAttributes(ARTIST as 'ID')))
from    cdtable;
```

Création d'une séquence d'éléments :

```
select XMLForest(TITLE as 'TITRE',
                 PRICE as 'PRIX',
                 COUNTRY as 'PAYS',
                 YR as 'ANNEE')
from    cdtable;
```

3.2 XMLConcat

Exemple de concaténation d'éléments (création de séquence) :

```
select XMLConcat(XMLElement('NOM', COMPANY),
                 XMLElement('ARTISTE',
                            XMLElement('NOM', ARTIST),
                            XMLElement('CD',
                                       XMLForest(TITLE as 'TITRE',
                                                 PRICE as 'PRIX',
                                                 COUNTRY as 'PAYS',
                                                 YR as 'ANNEE'))))
from cdtable;
```

Exécutez la requête suivante ; Quelle est la différence avec la requête précédente ?

```
select XMLElement('NOM', COMPANY),
       XMLElement('ARTISTE',
                  XMLElement('NOM', ARTIST),
                  XMLElement('CD',
                             XMLForest(TITRE as 'TITRE',
                                       PRICE as 'PRIX',
                                       COUNTRY as 'PAYS',
                                       YR as 'ANNEE')))
from cdtable;
```

3.3 XMLAgg

Pour regrouper tous les artistes par société :

```
select XMLElement('CATALOGUE',
        (select XMLAgg(XMLElement('SOCIETE',
                        XMLElement('NOM',y.COMPANY),
                       (select XMLAgg(XMLElement('ARTISTE', z.ARTIST))
                        from cdtable z where z.COMPANY=y.COMPANY)))
        from cdtable y))
from dual;
```

Créez une table avec un n-uplet par société :

```
drop table societe;

create table societe (xml XMLType);

insert into societe
select  XMLElement('SOCIETE',
                   XMLElement('NOM', y.COMPANY),
                   (select XMLAgg(XMLElement('ARTISTE', z.ARTIST))
                   from cdtable z where z.COMPANY=y.COMPANY))
from cdtable y;

select extractvalue(xml, '/SOCIETE/NOM'), count(*)
from societe
group by extractvalue(xml, '/SOCIETE/NOM')
order by count(*);
```

4 Génération XML

1. Chargez le schéma et les données fournis dans fichier `tournoi_tennis.sql` et vérifiez que les données ont bien été insérées.

2. Affichez la description des 3 tables `JOUEUR`, `RENCONTRE` et `GAIN` (rappel : commande `desc`).
3. Générez un extrait de la table `JOUEUR` qui transforme **tous** les attributs en éléments XML sauf l'attribut `NumJOUEUR` qui devient un attribut XML :

```
<!ELEMENT JOUEUR (NOM, PRENOM, ANNAISS, NATIONALITE) >
<!ATTLIST JOUEUR NumJOUEUR ID >
```

4. Ajoutez pour chaque joueur la liste des participations aux tournois :

```
<!ELEMENT JOUEUR (NOM, PRENOM, ANNAISS, NATIONALITE, PARTICIPATIONS) >
<!ATTLIST JOUEUR NUJOUEUR ID >
<!ELEMENT PARTICIPATIONS (TOURNOI*)>
<!ELEMENT TOURNOI EMPTY>
<!ATTLIST TOURNOI LIEU CDATA
                  ANNEE CDATA >
```