1 Optimisation de requêtes

On considère la base de données dont le schéma est le suivant.

```
CREATE TABLE Pays (code   VARCHAR(4)                    NOT NULL,
                   nom    VARCHAR(30) DEFAULT 'Inconnu' NOT NULL,
                   langue VARCHAR(30)                   NOT NULL,
                   CONSTRAINT PKPays PRIMARY KEY (code));

CREATE TABLE Artiste  (idArtiste  INTEGER     NOT NULL,
                       nom        VARCHAR(30) NOT NULL,
                       prenom     VARCHAR(30) NOT NULL,
                       anneeNaiss INTEGER,
                       CONSTRAINT PKArtiste PRIMARY KEY (idArtiste),
                       CONSTRAINT UniqueNomArtiste UNIQUE (nom, prenom));

CREATE SEQUENCE seq_film;

CREATE TABLE Film  (idFilm   INTEGER     NOT NULL,
                    titre    VARCHAR(50) NOT NULL,
                    annee    INTEGER NOT NULL,
                    idMES    INTEGER,
                    genre    VARCHAR(20) NOT NULL,
                    resume   LONG,
                    codePays VARCHAR(4),
                    CONSTRAINT PKFilm PRIMARY KEY (idFilm),
                    FOREIGN KEY (idMES) REFERENCES Artiste,
                    FOREIGN KEY (codePays) REFERENCES Pays);

CREATE TABLE Role (titreFilm VARCHAR(50) NOT NULL,
                   idActeur  INTEGER     NOT NULL,
                   nomRole   VARCHAR(30),
                   CONSTRAINT PKRole PRIMARY KEY (idActeur,titreFilm),
                   FOREIGN KEY (idActeur) REFERENCES Artiste);
```

1.1 Préparation

1. Créez une base de données standard : regardez et exécutez le script *opti.sql*.
2. Utilisation des plans de requêtes.
   (a) Créez la table pour stocker les plans de requêtes, en exécutant le script (que vous aurez regardé avant) : *$ORACLE_HOME/rdbms/admin/utlxplan.sql*.
   (b) On détermine le plan d'une requête *q* à l'aide de l'ordre *EXPLAIN PLAN SET STATEMENT_ID = 'nom de ma requete' FOR q*, qui va stocker le plan dans la table *PLAN_TABLE* (c'est vous qui choisissez l'identifiant *nom de ma requete*). Il peut s'afficher ensuite à l'aide du script *explain.sql*.[1] Faites-le pour *select * from pays*. Avant de restocker le plan d'une requête, détruisez l'ancien en le sélectionnant par son *statement_id*.

1.2 Plans de requêtes

Le but de ces petits exercices est de se familiariser avec les plans d'exécution, en consultant ceux d'Oracle pour un petit panorama de requêtes, impliquant ou non des index et des jointures. On rappelle celles des opérations utilisées par Oracle que nous allons considérer ici.

- Parcours séquentiel d'une table : `FULL TABLE SCAN`

[1] Vous pouvez aussi utiliser le script *utlxpls*.

- Parcours d'index : `INDEX SCAN`
- Accès par addresse : `ACCESS BY ROWID`
- Jointure par boucles imbriquées : `NESTED LOOP`
- Jointure par Tri/Fusion: `SORT` et `MERGE`

1. Affichez les plans de requêtes pour les requêtes suivantes, et tentez de déterminer l'interaction entre index et plans d'exécution (exécutez la requuête une fois avant pour la visualiser, même si c'est évident).
   - `SELECT * FROM Film WHERE titre = 'Vertigo';`
   - `SELECT * FROM Film WHERE idFilm = 21;`
   - `SELECT titre, nom, prenom FROM Film f, Artiste a`
     `WHERE idMES = idArtiste;`
   - `SELECT nomRole FROM Role r, Artiste a`
     `WHERE r.idActeur = a.idArtiste AND nom = 'Pacino';`
2. On considère la requête suivante, qui donne tous les films avec James Stewart, parus en 1958 :
   ```
   SELECT titre FROM Film f, Role r, Artiste a
   WHERE a.nom = 'Stewart' AND a.prenom='James'
   AND f.titre = r.titreFilm AND r.idActeur = a.idArtiste
   AND f.annee = 1958;
   ```
   - Quels sont les indexs créés automatiquement par Oracle ? Comment le plan d'execution varie-t-il en fonction de la présence d'indexs ? On pourra créer/supprimer des indexs avec :
     `CREATE INDEX monIndex ON maTable(maColonne[,...])`
     `DROP INDEX monIndex`, ou modifier les contraintes existant sur les tables.
   - Peut-on forcer un plan d'execution différent en modifiant la requête ? Penser à utiliser des requêtes imbriquées, avec des `IN`, des `EXISTS`. Les requêtes que vous proposez vous paraissent elle efficaces ?

Remerciements

à Omar Benjelloun pour avoir rédigé cet énoncé, et à Philippe Rigaux pour avoir autorisé l'utilisation du matériel dont provient quasiment tout cet énoncé.[2]

[2] Faites *Google* pour en savoir plus.

1.3 Correction

Afficher les plans de requêtes pour les requêtes suivantes, et tenter de determiner l'interaction entre indexs et plans d'execution.

- `SELECT * FROM Film WHERE titre = 'Vertigo';`

  ```
  explain plan set statement_id = 'selectVertigo' for
  SELECT * FROM Film WHERE titre = 'Vertigo';
  @explain selectVertigo
  ```

  Full scan de la table car il n'y a pas d'index, pour trouver le Film dont le titre est Vertigo.

- `SELECT * FROM Film WHERE idFilm = 21;`

  ```
  explain plan set statement_id = 'selectVertigoIndex' for
  SELECT * FROM Film WHERE idFilm = 21;
  @explain selectVertigoIndex

  Plan d'execution
  --------------------------------------------------------------------------------
  SELECT STATEMENT
    TABLE ACCESS BY INDEX ROWID FILM
      INDEX UNIQUE SCAN PKFILM
  ```

  Scan de l'index, pour trouver le rowid correspondant, puis accès direct à la table.

- `SELECT titre, nom, prenom FROM Film f, Artiste a`
  `WHERE idMES = idArtiste;`

  ```
  explain plan set statement_id = 'join1' for
  SELECT titre, nom, prenom FROM Film f, Artiste a
  WHERE idMES = idArtiste;
  @explain join1

  Plan d'execution
  --------------------------------------------------------------------------------
  SELECT STATEMENT
    NESTED LOOPS
      TABLE ACCESS FULL FILM
      TABLE ACCESS BY INDEX ROWID ARTISTE
        INDEX UNIQUE SCAN PKARTISTE
  ```

  Utilisation de l'index pour accès direct à la table Artiste, Full scan pour la table film. La jointure se fait par boucle imbriquées indexées: balayage de la table non indexée, et accès direct sur l'index de l'autre table par l'attribut de jointure. C'est très efficace et produit des résultat partiels rapidement. C'est la stratégie utilisée par Oracle à chaque fois qu'il a un index. Quelle table met-il en mémoire : on voudrait la plus petite, mais en l'absence de statistiques il prend la dernière de la clause *from*... Celle qu'il met en mémoire est la première qu'on voit dans le plan.

- `SELECT nomRole FROM Role r, Artiste a`
  `WHERE r.idActeur = a.idArtiste AND nom = 'Pacino';`

  ```
  explain plan set statement_id = 'join2' for
  SELECT nomRole FROM Role r, Artiste a
  WHERE r.idActeur = a.idArtiste AND nom = 'Pacino';
  @explain join2

  Plan d'execution
  ```

```
--------------------------------------------------------------------------------
SELECT STATEMENT
  NESTED LOOPS
    TABLE ACCESS BY INDEX ROWID ARTISTE
      INDEX RANGE SCAN UNIQUENOMARTISTE
    TABLE ACCESS BY INDEX ROWID ROLE
      INDEX RANGE SCAN PKROLE
```

Même technique que ci-dessus. Il utilise l'index sur le nom de l'artiste (en range car plusieurs prénoms différents possibles) qui ramène donc un tout petit nombre d'identifiants d'artistes. Muni de l'identifiant de l'artiste, il le cherche avec l'index (toujours en range, même raison) dans role.

On considère la requête suivante, qui donne tous les films avec James Stewart, parus en 1958 :

```
SELECT titre FROM Film f, Role r, Artiste a
WHERE a.nom = 'Stewart' AND a.prenom='James'
AND f.titre = r.titreFilm AND r.idActeur = a.idArtiste
AND f.annee = 1958
```

- Quels sont les index créés automatiquement par Oracle ? Comment le plan d'execution varie-t-il en fonction de la présence d'indexs ? On pourra créer/supprimer des index avec :
  `CREATE INDEX monIndex ON maTable(maColonne[,...])`
  `DROP INDEX monIndex`, ou modifier les contraintes existant sur les tables.

  Encore des boucles imbriquées indexées, avec un full scan de la table Film, car pas d'index sur le titre du film, et des accès directs par index aux tables ROLE et ARTISTE.

  On peut rajouter un index sur le titre du film:

  `CREATE INDEX IDXFilm ON FILM(TITRE);`

  Cette fois l'accès à la table FILM se fait par l'index que l'on vient de créer. Accès par intervalle, car il n'y a pas de contraintes d'unicité sur le titre d'un film.

  On peut aussi supprimer les contraintes sur la table ROLE:

  `ALTER TABLE ROLE DROP CONSTRAINT PKROLE;`

  Boucle imbriquées indexées entre ROLE et ARTISTE, car il existe un index sur artiste, puis tri/fusion avec la table Film, car il n'y a pas d'index à utiliser. Cet algorithme est plus efficace que des boucles imbriquées sans index.

  Enfin, on peut supprimer la clé primaire sur idArtiste:

  `ALTER TABLE ARTISTE DROP CONSTRAINT PKARTISTE CASCADE;`

  Cette fois-ci, les tables sont accédées en full scan, et les jointures sont faites par tri/fusion.

- Peut-on forcer un plan d'execution différent en modifiant la requête ? Penser à utiliser des requêtes imbriquées, avec des `IN`, des `EXISTS`. Les requêtes que vous proposez vous paraissent elle efficaces ?

  On pourra essayer les requêtes suivantes:

```
SELECT titre
FROM Film f, Role r
WHERE f.titre = r.titreFilm
AND f.annee = 1958
AND r.idActeur IN (SELECT idArtiste
                   FROM Artiste
                   WHERE nom='Stewart'
                   AND prenom='James');

SELECT titre
FROM Film f, Role r
WHERE f.titre = r.titreFilm
AND f.annee = 1958
```

```
AND EXISTS (SELECT 'x'
            FROM Artiste a
            WHERE nom='Stewart'
            AND prenom='James'
            AND r.idActeur = a.idArtiste);


SELECT titre FROM Film
WHERE annee = 1958
AND idFilm IN
    (SELECT idFilm FROM Role
     WHERE idActeur IN (SELECT idArtiste
                        FROM Artiste
                        WHERE nom='Stewart'
                        AND prenom='James'));
```

Globalement les plans d'execution sont tous moins bons.