Lif10 – Fondements des bases de données
TP1 – Langages de définition et de manipulation de données

Licence informatique – Automne 2014–2015

`http://liris.cnrs.fr/~mplantev/doku/doku.php?id=lif10`

Résumé

Ce travail a pour objectif de pratiquer l'écriture de requêtes d'interrogation (`SELECT FROM WHERE`), de manipulation (`INSERT`, `UPDATE`, `DELETE`) et la définition de données (`CREATE / ALTER TABLE`) en SQL. Trouver toutes les informations nécessaires pour vous connecter à la base de données sur :

`http://liris.cnrs.fr/~ecoquery/dokuwiki/doku.php?id=enseignement:aide:oracle`

Exercice 1 : création de la base

Vous souhaitez créer un site d'appréciation de films. Vous avez déjà collecté les notes d'examinateurs sur différents films. Il n'y a pas beaucoup de données, mais vous pouvez toujours essayer quelques requêtes intéressantes. Le fichier SQL `TP1_moviedata.sql` [1] est donné pour mettre en place le schéma suivant et le peupler :

- *Movie*(*mID*, *title*, *year*, *director*)
- *Reviewer*(*rID*, *name*)
- *Rating*(*rID*, *mID*, *stars*, *ratingDate*)

1. A partir du fichier `TP1_moviedata.sql`, créer la base de données.

Exercice 2 : requêtes simples

1. Les films réalisés par *Steven Spielberg*.
2. Trouver toutes les années, dans l'ordre croisant, qui ont un film qui a reçu une note de 4 ou 5.
3. Trouver le nom des personnes qui ont noté le film *Gone with the Wind*.
4. Pour chaque évaluation où l'examinateur est identique au réalisateur du film (même nom), retourner le nom de l'examinateur, le titre du film, et le nombre d'étoiles.
5. Retourner l'intégralité des évaluations, en remplaçant *rID* avec les noms des examinateurs et *mID* par les titres de films. Trier le résultat, d'abord par le nom de relecteur, puis par le titre de film, et enfin par le nombre d'étoiles.
6. Retourner les titres des films non encore examinés par *Chris Jackson*.
7. Pour tous les cas où la même personne note deux fois le même film et donne une note plus élevée la seconde fois, retourner le nom de l'examinateur et le titre du film.
8. Retourner les paires d'évaluateurs qui ont noté le même film, retourner le nom de ces deux examinateurs en éliminant les duplications ((*a*, *b*) et (*b*, *a*) représentent la même chose).
9. (subsidiaire) Proposer un nom « commercial » à la requête précédente.

1. `http://liris.cnrs.fr/~mplantev/ENS/LIF10/CM/moviedata.sql`

Exercice 3 : requêtes avançées

1. Trouver les titres des films qui n'ont pas reçu d'évaluations.
2. Retourner le nom de l'examinateur, le titre du film,et le nombre d'étoiles pour tous les films qui ont actuellement la plus mauvaise note dans la base.
3. Pour chaque film, trouver la meilleure note reçue. Retourner le titre de film et le nombre d'étoiles. Trier par rapport au titre de film (ordre alphabétique).
4. (subsidiaire) Proposer un nom « commercial » à la requête précédente.

Exercice 4 : requêtes d'agrégation

1. Lister les titres de films et leur note moyenne.
2. Trouver le nom de tous les examinateurs qui ont fait au moins 3 évaluations.
3. Trouver le(s) film(s) ayant la meilleure moyenne de note. Retourner le titre de film, et la note moyenne.
4. Idem pour la plus mauvaise moyenne.
5. Trouver la différence entre la moyenne des notes des films [2] réalisés avant 1980 et ceux réalisés après.
6. Pour chaque film, retourner le titre et la différence entre la meilleure et la plus mauvaise note pour un film donné. Trier par rapport à cette amplitude puis en fonction du titre.
7. (subsidiaire) Proposer un nom « commercial » à la requête précédente.

Exercice 5 : requêtes sur les valeurs nulles

1. Trouver les noms des examinateurs qui n'ont pas daté leurs évaluations.
2. Pour chaque réalisateur, retourner leur nom, titre de film(s) dirigés et ayant reçu la meilleure note de leur carrière. Retourner également cette note. Ignorer les films dont le réalisateur n'est pas spécifié.

Exercice 6 : modification de la base de données

1. Ajouter un nouvel examinateur *Roger Ebert* dans la base de données avec un *rID* égal à 209.
2. Pour vérifier l'insertion précédente, écrire une requête pour retourner le nombre d'évaluateurs.
3. Insérer des évaluations à 5 étoiles faites par *Roger Ebert* pour tous les films de la base. Laisser la date à `NULL`.
4. Pour vérifier l'insertion précédente, retourner le nom des examinateurs qui ont noté *tous* les films.
5. Pour tous les films qui ont une note moyenne supérieure ou égale à 4, ajouter 25 ans à la date de réalisation (mettre à jour les tuples, ne pas en créer).
6. Pour vérifier la modification précédente, retourner le nombre de films réalisés avant 1990.
7. Supprimer tous les films de la base à l'exception des films réalisés entre 2000 et 2010.
8. Maintenant, beaucoup d'évaluations réfèrent à des films qui ne sont plus dans la table *Movie*. Supprimer toutes les évaluations dont le film correspondant n'apparaît plus dans la table *Movie*.
9. Maintenant, des évaluateurs n'ont plus aucune évaluation. Supprimer les examinateurs qui n'ont pas d'évaluation dans la table *Rating*.

2. Il faut bien calculer la moyenne pour chaque film, puis la moyenne des moyennes : il ne suffit pas de calculer simplement la moyenne des notes avant et après 1980).

Corrections

2.1)

```
SELECT title
FROM Movie
WHERE director LIKE 'Steven␣Spielberg';
```

2.2)

```
SELECT DISTINCT m.year
FROM Movie m, Rating r
WHERE m.mID = r.mID AND (r.stars = 4 OR r.stars = 5)
ORDER BY m.year;
```

2.3)

```
SELECT DISTINCT w.name
FROM Movie m, Rating r, Reviewer w
WHERE m.mID = r.mID AND w.rID = r.rID AND m.title = 'Gone␣with␣the␣Wind';
```

2.4)

```
SELECT DISTINCT w.name, m.title, r.stars
FROM Movie m, Rating r, Reviewer w
WHERE m.mID = r.mID AND w.rID = r.rID AND
m.director = w.name;
```

2.5)

```
SELECT w.name, m.title, r.stars, r.ratingDate
FROM Movie m, Rating r, Reviewer w
WHERE m.mID = r.mID AND w.rID = r.rID
ORDER BY w.name, m.title, r.stars;
```

2.6)

```
SELECT m.title
FROM Movie m
WHERE m.mID NOT IN (
  SELECT r.mID
  FROM Rating r, Reviewer w
  WHERE r.rID = w.rID AND w.name = 'Chris␣Jackson')
```

2.7)

```
SELECT w.name, m.title
FROM Rating r, Rating r2, Movie m, Reviewer w
WHERE m.mID = r.mID AND w.rID = r.rID
AND r.rID = r2.rID AND r.mID = r2.mID
AND r2.ratingDate > r.ratingDate AND r2.stars > r.stars
```

2.8)

```
SELECT DISTINCT w.name n1, w2.name n2
FROM Rating r, Rating r2, Reviewer w2, Reviewer w
WHERE w2.rID = r2.rID AND w.rID = r.rID
AND r.rID > r2.rID AND r.mID = r2.mID
```

3.1)

```
SELECT m.title
```

```
FROM Movie m
WHERE m.mID NOT IN (
  SELECT DISTINCT r.mID
  FROM Rating r
)
```

3.2)

```
SELECT w.name, m.title, r.stars
FROM Movie m, Rating r, Reviewer w
WHERE m.mID = r.mID AND w.rID = r.rID
AND r.stars = (
  SELECT MIN(r.stars)
  FROM Rating r
)
```

3.3)

```
SELECT DISTINCT m.title, r.stars
FROM Movie m, Rating r
WHERE m.mID = r.mID
AND r.stars = (
  SELECT MAX(r1.stars)
  FROM Rating r1
  WHERE r1.mID = r.mID
)
ORDER BY m.title
```

4.1)

```
SELECT m.title, AVG(r.stars)
FROM Movie m, Rating r
WHERE m.mID = r.mID
GROUP BY m.mID, m.title
```

4.2)

```
SELECT w.name, COUNT(r.mID)
FROM Reviewer w, Rating r
WHERE w.rID = r.rID
GROUP BY r.rID, w.name
HAVING  COUNT(r.mID) >= 3
```

4.3)

```
SELECT m.title, AVG(r.stars)
FROM Movie m, Rating r
WHERE m.mID = r.mID
GROUP BY m.mID, m.title
HAVING AVG(r.stars) = (
  SELECT MAX(avgstars)
  FROM (
    SELECT AVG(r2.stars) AS avgstars
    FROM Rating r2
    GROUP BY r2.mID
  )
)
```

4.4)

```
SELECT m.title, AVG(r.stars)
FROM Movie m, Rating r
WHERE m.mID = r.mID
```

```
GROUP BY m.mID, m.title
HAVING AVG(r.stars) = (
  SELECT MIN(avgstars)
  FROM (
    SELECT AVG(r2.stars) AS avgstars
    FROM Rating r2
    GROUP BY r2.mID
  )
)
```

4.5)

```
SELECT (avg1-avg2) FROM
(SELECT AVG(avgstars1) AS avg1
FROM (
  SELECT AVG(r.stars) AS avgstars1
  FROM Rating r, Movie m
  WHERE m.mID = r.mID AND m.year < 1980
  GROUP BY r.mID
)),
(SELECT AVG(avgstars2) AS avg2
FROM (
  SELECT AVG(r1.stars) AS avgstars2
  FROM Rating r1, Movie m1
  WHERE m1.mID = r1.mID AND m1.year >= 1980
  GROUP BY r1.mID
))
```

4.6)

```
SELECT m.title, (MAX(r.stars) - MIN(r.stars)) AS spread
FROM Rating r, Movie m
WHERE m.mID = r.mID
GROUP BY r.mID, m.title
ORDER BY spread, title
```

5.1)

```
SELECT DISTINCT w.name
FROM Rating r, Reviewer w
WHERE w.rID = r.rID AND
r.ratingDate IS NULL
```

5.2)

```
SELECT DISTINCT m.director, m.title, r.stars
FROM Movie m, Rating r
WHERE m.mID = r.mID AND (m.director, r.stars) IN (
  SELECT m2.director, MAX(r2.stars)
  FROM Movie m2, Rating r2
  WHERE m2.mID = r2.mID AND m2.director IS NOT NULL
  GROUP BY m2.director
)
```

6.1)

```
insert into Reviewer values(209, 'Roger␣Ebert');
```

6.2)

```
select count(*) from reviewer;
```

6.3)

```
insert into Rating values(209, 101, 5, null);
insert into Rating values(209, 102, 5, null);
insert into Rating values(209, 103, 5, null);
insert into Rating values(209, 104, 5, null);
insert into Rating values(209, 105, 5, null);
insert into Rating values(209, 106, 5, null);
insert into Rating values(209, 107, 5, null);
insert into Rating values(209, 108, 5, null);
```

6.4)

```
SELECT w2.name
FROM Reviewer w2
WHERE w2.rID NOT IN (
  SELECT DISTINCT w1.rID
  FROM Movie m1, Reviewer w1
  WHERE (w1.rID, m1.mID) NOT IN (
    SELECT r.rID, r.mID
    FROM Rating r
  )
)
```

6.5)

```
UPDATE Movie
SET year = year+25
WHERE mID IN (
  SELECT mID
  FROM Rating r
  GROUP BY mID
  HAVING AVG(stars) >= 4
)
```

6.6)

```
SELECT COUNT(*)
FROM Movie
WHERE year < 1990
```

6.7)

```
DELETE
FROM Movie
WHERE year < 2000 OR year > 2010
```

6.8)

```
DELETE
FROM Rating
WHERE mID NOT IN (
  SELECT mID
  FROM Movie
)
```

6.9)

```
DELETE
FROM Reviewer
WHERE rID NOT IN (
  SELECT DISTINCT rID
  FROM Rating
)
```