# LIF10 – FONDEMENTS DES BASES DE DONNÉES
## *TP2 – Contraintes et dictionnaire*

Licence informatique – Automne 2014–2015

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

**Résumé**

Ce TP poursuit le précédent avec les contraintes de clef, de valeurs non-nulles, sur les tuples et d'intégrité référentielle, qui sont les contreparties pratiques des dépendances fonctionnelles et d'inclusion vues en cours et en TD. Pour finir, une prise en main des curseurs PL/SQL est proposé.

# 1 Contraintes de clé

### Exercice 1 : re-création de la base avec contraintes

Supprimer les tables créées dans le TP1. Modifier ensuite les déclarations de création de tables afin de prendre en compte les contraintes suivantes.

1. Ajouter les contraintes de clés (`PRIMARY KEY, UNIQUE`) suivantes dans la déclaration des tables
   – `mID` est une clé de `Movie`.
   – Le couple `(title, year)` est également une clé de `Movie`.
   – `rID` est une clé de `Reviewer`.
   – `(rID, mID, ratingDate)` est une clé de `Rating`, mais avec des valeurs nulles autorisées.
2. Exprimer les contraintes des questions précédentes dans le formalisme des dépendances fonctionnelles.
3. Ajouter les contraintes de valeurs non-nulles (`NOT NULL`) suivantes dans la déclaration des tables
   – `Reviewer.name` doit être non nul.
   – `Rating.stars` doit être non nul.
4. Ajouter les contraintes suivantes sur les tuples (`CHECK`) dans la déclaration des tables
   – Les films doivent être réalisés après 1900.
   – `Rating.stars` est défini sur l'ensemble $\{1, 2, 3, 4, 5\}$.
   – La date `Rating.ratingDate` doit être postérieure au 01/01/2000.
   – Les films réalisés par *Steven Spielberg* sont antérieurs à 1990 et ceux de *James Cameron* sont postérieurs à 1990. Pour l'expression de la contrainte booléenne, penser à l'équivalence logique $A \Rightarrow B \equiv \neg A \vee B$.
5. Reprendre les questions précédentes en modifiant la structure des tables sans les supprimer (commande `ALTER TABLE`).

### Exercice 2 : vérification des contraintes

1. Importer les données du TP1 dans les tables nouvellement crées. Il ne doit pas y avoir d'erreur car toutes les contraintes sont satisfaites.
2. Vérifier que chacune des commandes suivante génère une erreur. Préciser la contrainte violée en relevant le message d'erreur obtenu.
   1. `insert into Movie values (109, 'Titanic', 1997, 'JC');`
   2. `insert into Reviewer values (201, 'Ted Codd');`
   3. `update Rating set rID = 205, mID=104;`
   4. `insert into Reviewer values (209, null);`
   5. `update Rating set stars = null where rID = 208;`
   6. `update Movie set year = year - 40;`

7. `update Rating set stars = stars + 1;`
8. `insert into Rating VALUES (201, 101, 1, to_date('01-01-1999','dd-mm-yyyy'));`
9. `insert into Movie values (109, 'Jurassic Park', 1993, 'Steven Spielberg');`
10. `update Movie set year = year-10 where title = 'Titanic';`

3. Vérifier que chacune des commandes suivantes ne génère pas d'erreur.
   1. `update Movie set mID = mID +1;`
   2. `insert into Movie values (109, 'Titanic', 2001, null);`
   3. `update Rating set mID = 109;`
   4. `update Movie set year = 1901 where director <> 'James Cameron';`
   5. `update Rating set stars = stars - 1;`
4. Avec la requête vue dans l'exercice 3 du TD2, vérifier que les dépendances de la question I.5 de l'exercice 2 de ce TP sont bien satisfaites.
5. Videz les trois tables puis réimportez les données du TP1 avant de passer à la suite.

# 2 Contraintes d'intégrité référentielle

### Exercice 3 : contraintes d'intégrité référentielle

Oracle permet plusieurs types d'application des contraintes d'intégrité référentielle qu'il faut préciser au moment de la déclaration de la contrainte via `FOREIGN KEY`. Dans les exemples suivants `DEPT` est la *table parent* et `EMP` est la *table enfant*, c'est-à-dire respectivement la table *référencée* et celle sur laquelle *porte* la contrainte.

– Pour empêcher toute mise à jour ou suppression d'une clé de la table parent :

```
CREATE TABLE EMP (
  empno number(4),
  deptno number(2),
  FOREIGN KEY (deptno) REFERENCES DEPT (deptno));
```

– Quand une clé de la table parent est supprimée, tous les tuples de la table enfant qui dépendent de la valeur supprimée sont également supprimés :

```
CREATE TABLE EMP (
  empno number(4),
  deptno number(2),
  FOREIGN KEY (deptno) REFERENCES DEPT (deptno)
  ON DELETE CASCADE);
```

– Ne pas supprimer les tuples de la table enfant quand une clé de la table parent est supprimé. Les valeurs de la table enfant sont alors mises à `NULL` :

```
CREATE TABLE EMP (
  empno number(4),
  deptno number(2),
  FOREIGN KEY (deptno) REFERENCES DEPT (deptno)
  ON DELETE SET NULL);
```

1. Modifiez la définition de la table `Rating` pour prendre en compte les contraintes suivantes :
   1. Intégrité référentielle entre `Rating.rID` et `Reviewer.rID` (suppression sur `Reviewers` : `SET NULL`).
   2. Intégrité référentielle entre `Rating.mID` et `Movie.mID` (suppression sur `Movies` : `CASCADE`).
2. Exprimer les contraintes des questions précédentes dans le formalisme des dépendances d'inclusion.

## Exercice 4 : vérification des contraintes (suites)

1. Est-ce que les instructions suivantes génèrent des erreurs ou non ? Si oui, relever les messages.
    1. `INSERT INTO Rating VALUES(333, 104, 3, to_date('02-01-2011', 'dd-mm-yyyy'));`
    2. `INSERT INTO Rating VALUES(208, 111, 3, to_date('02-01-2011', 'dd-mm-yyyy'));`
    3. `update Rating set rID = 209 where rID = 208;`
    4. `update Rating set mID = mID + 1;`
    5. `update Movie set mID = 109 where mID = 108;`
    6. `update Reviewer set rID = rID + 10;`
2. Exécutez les requêtes suivantes en vérifiant que la table `Rating` a été modifiée en conséquence.
    1. `DELETE FROM Reviewer WHERE rID = 208;`
    2. `DELETE FROM MOVIE WHERE mID = 101;`
3. Expliquer ce que fait la requête suivante (recherche de la documentation)

```
SELECT table_name,
       constraint_name,
       column_name
FROM USER_CONSTRAINTS NATURAL JOIN USER_CONS_COLUMNS
WHERE table_name in ('MOVIE', 'RATING', 'REVIEWER');
```