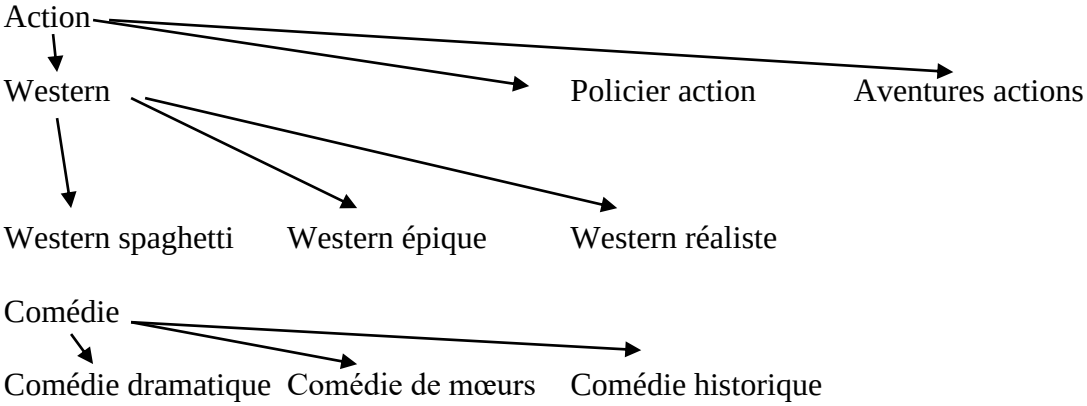
L3 INFORMATIQUE - L3 MIASHS – M1 MSID
-=-=-=-=-
BASES DE DONNEES
-=-=-=-=-=-=-=-=-=-=-

INTERROGATION DE LA BASE DE DONNEES CINEMA-DECLENCHEURS

PARTIE I :

On doit intégrer dans la base le fait que les catégories sont liées par des liens hiérarchique Père-Fils. Modifiez la structure de la table catégorie afin de prendre en compte cette hiérarchie. Pour les données actuellement contenues dans la base mettre à jour les valeurs afin de prendre en compte les structures arborescentes suivantes :



Donnez les requêtes SQL qui permettent :

- ✓ D'afficher toutes les catégories (code et nom) qui dépendent de la catégorie Action.
- ✓ D'afficher le code et le titre des films associés à la catégorie Action et à toutes les catégories qui en dépendent.

PARTIE II : UTILISATION DES DECLENCHEURS (TRIGGERS)

Afin de faciliter les opérations de mises à jour, vous avez créé des déclencheurs. Le déclencheur permet de lancer l'exécution d'une ou plusieurs commandes sur d'autres tables lorsqu'une opération survient sur la table à laquelle il est associé. Le corps du déclencheur est constitué par un programme. Dans les commandes SQL, on peut faire référence aux valeurs contenues dans le n-uplet concerné par l'événement déclencheur en utilisant les préfixes :old ou :new pour désigner le n-uplet en modification. Le préfixe :old peut être utilisé lors d'une suppression ou d'une modification de valeur, le préfixe :new peut être utilisé lors d'une insertion ou une modification de valeur.

La description du déclencheur contient :

- ✓ Le nom du déclencheur
- ✓ L'ordre d'exécution des commandes (avant ou après l'événement déclencheur)
- ✓ L'événement déclencheur (Insertion, Modification, Suppression)
- ✓ Le nom de la table sur laquelle se produit l'événement
- ✓ Exécution des opérations déclenchées pour chaque ligne modifiée ou bien une seule exécution pour l'événement
- ✓ Opération(s) à exécuter (corps du programme)
- ✓ En cas d'anomalie, on peut arrêter le programme, l'opération de déclenchement (événement) et les modifications réalisées dans le corps du déclencheur ne sont pas réalisées.

Plus généralement des programmes PL/SQL peuvent être créés, ces programmes peuvent être stockés dans le dictionnaire de données sous forme de procédures et fonctions. Ces programmes ne sont pas déclenchés automatiquement, ils sont exécutés à la demande.

1) Gestion de la répartition horizontale avec copie :

Lors de l'implantation du lien d'héritage par une représentation horizontale avec copie cela implique que l'insertion dans l'une des tables FILMS ou FILMS_ANIM doit être **précédée** par l'insertion dans la table PRODUCT_CINE (Deux commandes d'insertion réalisées).
Vous avez créé dans le TP n°1 des déclencheurs associés aux tables FILMS et FILMS_ANIM dans le cas d'insertion dans ces tables. Dans le cas de l'insertion d'une production cinématographique comme « Qui veut la peau de Roger Rabbit » (film et film d'animation), on doit insérer une ligne dans FILM et FILM_ANIMATION. Les déclencheurs que vous avez créés risquent de poser problème. Modifiez ces déclencheurs afin que ces insertions se passent correctement.

2) Contraintes d'intégrité :

Lors de la création de la base la contrainte un film n'a pas plus de 5 réalisateurs n'a pas été prise en compte. Mettez en place cette contrainte.

3) Recherche dans une arborescence :

On souhaite rechercher à partir d'un nom de catégorie la liste des productions cinématographiques (code, titre, année de sortie) correspondantes**.** Une première requête peut être écrite afin d'obtenir toutes les productions qui sont directement rattachées à la catégorie mentionnée. On souhaite si cette requête n'aboutit pas rechercher les productions rattachées à les sous-catégories de la catégorie donnée. Donnez la requête SQL permettant de rechercher dans toutes les sous-catégories.

ANNEXES :

SYNTAXE GENERALE DE LA REQUETE SQL :

SELECT [ALL | DISTINCT] { * | < liste d'expressions > }
FROM < liste de tables >
[**WHERE** < condition >]
[**GROUP BY** < liste d'expressions > [**HAVING** < condition >]]
[{UNION | INTERSECT | MINUS} SELECT .....]
[**ORDER BY** expr1 [ASC|DESC], expr2 [ASC|DESC],....]

PARCOURS D'UNE STRUCTURE ARBORESCENTE :

SELECT [LEVEL] Liste_résultat
FROM Nom_Relation
[WHERE Condition**]**
[START WITH condition]
CONNECT BY PRIOR Condition;

START WITH : Point de départ du parcours de l'arbre

CONNECT BY PRIOR : Introduit la condition portant sur les colonnes de jointures (Colonne_sup, Colonne_inf)

-Parcours du bas vers le haut
CONNECT BY PRIOR colonne_sup = colonne_inf;
-Parcours du haut vers le bas
CONNECT BY PRIOR colonne_inf = colonne_sup;

FONCTIONS SUR LES DATES :

ADD_MONTHS (d,i)	Ajout de mois à une date
CURRENT_DATE	Date courante
EXTRACT ({YEAR|MONTH|DAY|HOUR|MINUTE|SECOND|} FROM {d|i})	Extraction d'une partie d'une date
LAST_DAY(d)	Retourne le dernier jour du mois
MONTHS_BETWEEN(d1,d2)	Nombre de mois entre deux dates
NEXT_DAY(d,jour)	Date du prochain jour ouvrable
ROUND(d,format)	Arrondi la date selon le format
TRUNC(d,format)	Tronque la date selon le format

DECLENCHEURS :

La syntaxe générale en SQL est donnée ci-dessous :

```
CREATE [OR REPLACE ]TRIGGER Nom_déclencheur
        {AFTER | BEFORE}{ DELETE | INSERT | UPDATE [OF col1 [,col2]…]}ON
        [Nom_de_table | Nom_de_vue ]
        [FOR EACH ROW]
        [DECLARE Variables_locales ;]
        BEGIN
                Corps_du_déclencheur;
        END ;
```

Dans le corps du déclencheur la détection d'erreurs peut être gérée :

Utilisation dans un déclencheur sans section EXCEPTION :

```
CREATE OR REPLACE TRIGGER Nom_déclencheur
…….
DECLARE
Variables_locales ;
BEGIN
…..
..      RAISE_APPLICATION_ERROR (-20100, 'erreur utilisateur');
END ;
```

Utilisation dans un déclencheur avec section EXCEPTION :

```
CREATE OR REPLACE TRIGGER Nom_déclencheur
…….
DECLARE
Variables_locales ;
BEGIN
…..
..      RAISE_APPLICATION_ERROR (-20100, 'erreur utilisateur');
EXCEPTION
        WHEN …..
        …
        WHEN OTHERS THEN
                RAISE;

END ;
```