|  |
| --- |
|  |
| Dossier de maintenance |
|  |

|  |
| --- |
| Fink Jérôme & Seel Océane  2015-2016 |

Table des matières

[1. Introduction 5](#_Toc438210762)

[2. SYS 7](#_Toc438210763)

[2.1. Acl.sql 7](#_Toc438210764)

[2.2. CreateUsers.sql 7](#_Toc438210765)

[3. CB 8](#_Toc438210766)

[3.1. CreaCB.sql 10](#_Toc438210767)

[3.2. CreaExternalTable.sql 10](#_Toc438210768)

[3.3. CreaJobAlimCC.sql 10](#_Toc438210769)

[3.4. CreaJobBackup.sql 10](#_Toc438210770)

[3.5. CreaRapport.sql 10](#_Toc438210771)

[3.6. CreateLogTable.sql 10](#_Toc438210772)

[3.7. CreaXMLCommunicationTable.sql 11](#_Toc438210773)

[3.8. PackageAlimCC.sql 11](#_Toc438210774)

[3.8.1. PROCEDURE MOVIE\_COPY\_GENERATOR 11](#_Toc438210775)

[3.8.2. PROCEDURE JOB 11](#_Toc438210776)

[3.9. PackageCB.sql 11](#_Toc438210777)

[3.9.1. FUNCTION VERIF\_FILM\_FIELDS 11](#_Toc438210778)

[3.9.2. FUNCTION VERIF\_GENRE\_FIELDS 11](#_Toc438210779)

[3.9.3. FUNCTION VERIF\_PRODUCTEUR\_FIELDS 12](#_Toc438210780)

[3.9.4. FUNCTION VERIF\_PAYS\_FIELDS 12](#_Toc438210781)

[3.9.5. FUNCTION VERIF\_LANGUE\_FIELDS 12](#_Toc438210782)

[3.9.6. FUNCTION VERIF\_PERSONNE\_FIELDS 12](#_Toc438210783)

[3.9.7. FUNCTION VERIF\_ROLE\_FIELDS 12](#_Toc438210784)

[3.10. PackageRecherche.sql 13](#_Toc438210785)

[3.10.1. FUNCTION recherche\_id 13](#_Toc438210786)

[3.10.2. FUNCTION recherche 13](#_Toc438210787)

[3.10.3. FUNCTION getAfficheFilm 13](#_Toc438210788)

[3.10.4. FUNCTION getNoteUtilisateurFilm 13](#_Toc438210789)

[3.10.5. FUNCTION getActeursFilm 13](#_Toc438210790)

[3.10.6. FUNCTION getRealisateursFilm 14](#_Toc438210791)

[3.10.7. FUNCTION getAvisFilm 14](#_Toc438210792)

[3.11. ProcedureAlimCB.sql 14](#_Toc438210793)

[3.12. ProcedureBackup.sql 14](#_Toc438210794)

[3.13. ProcedureEvalFilm.sql 14](#_Toc438210795)

[3.14. ProcedureRetourCopie.sql 15](#_Toc438210796)

[3.15. TriggerCopieCotesAvis.sql 15](#_Toc438210797)

[3.16. TriggersCopieFilm.sql 15](#_Toc438210798)

[4. CBB 16](#_Toc438210799)

[4.1. CreaCBB.sql 18](#_Toc438210800)

[4.2. CreateLogTable.sql 18](#_Toc438210801)

[4.3. PackageRecherche.sql 18](#_Toc438210802)

[4.3.1. FUNCTION recherche\_id 18](#_Toc438210803)

[4.3.2. FUNCTION recherche 18](#_Toc438210804)

[4.3.3. FUNCTION getAfficheFilm 18](#_Toc438210805)

[4.3.4. FUNCTION getNoteUtilisateurFilm 19](#_Toc438210806)

[4.3.5. FUNCTION getActeursFilm 19](#_Toc438210807)

[4.3.6. FUNCTION getRealisateursFilm 19](#_Toc438210808)

[4.3.7. FUNCTION getAvisFilm 19](#_Toc438210809)

[4.4. ProcedureEvalFilm.sql 19](#_Toc438210810)

[4.5. ProcedureRestore.sql 20](#_Toc438210811)

[4.6. ProcedureRetourCopie.sql 20](#_Toc438210812)

[4.7. TriggerCopieCotesAvis.sql 20](#_Toc438210813)

[4.8. TriggersCopieFilm.sql 20](#_Toc438210814)

[5. CC 21](#_Toc438210815)

[5.1. CreateLogTable.sql 21](#_Toc438210816)

[5.2. CreateXMLTable.sql 21](#_Toc438210817)

[5.3. ProcedureInsertXML.sql 21](#_Toc438210818)

[5.4. ProcedureProgFilm.sql 21](#_Toc438210819)

[5.5. Fichiers .xsd 22](#_Toc438210820)

[5.6. RegisterXsd.sql 22](#_Toc438210821)

[6. MKT 23](#_Toc438210822)

[7. DW 26](#_Toc438210823)

[8. Application film 29](#_Toc438210824)

[8.1. classApplicationFilm 29](#_Toc438210825)

[8.1.1. Film.java 29](#_Toc438210826)

[8.1.2. ThreadTestConnexion.java 29](#_Toc438210827)

[8.2. GUIapplicationFilm 29](#_Toc438210828)

[8.2.1. AccueilPanel.java 29](#_Toc438210829)

[8.2.2. AffichageFilmPanel.java 29](#_Toc438210830)

[8.2.3. ConnexionPanel.java 29](#_Toc438210831)

[8.2.4. FormulaireRecherchePanel.java 29](#_Toc438210832)

[8.2.5. GUI.java 30](#_Toc438210833)

[8.2.6. NoterDialog.java 30](#_Toc438210834)

[8.2.7. RechercheResultPanel.java 30](#_Toc438210835)

# Introduction

Dans le cadre du projet RQS nous avons créé trois bases de données ainsi qu’une application.

Les bases de données sont :

* **CB (Centrale Belge)** : Base de données contenant la liste de tous les films disponibles pour les cinémas de Belgique ainsi que les utilisateurs de la plateforme d’évaluation des films, leurs notes et leurs commentaires. CB est alimenté par une Centrale Internationale déjà existante (représentée par le fichier movies.txt du dossier ressources).
* **CBB (Centrale Belge Backup)** : Le rôle de CBB est de prendre la relève en cas de panne de CB. Elle est donc capable d’effectuer les mêmes opérations que CB. Les informations sont copiées entre les deux bases de manière synchrone pour les films et les avis donnés par les utilisateurs et de manière asynchrone pour les nouveaux utilisateurs inscrits.
* **CC (Complexe cinématographique)** : Cette base de données est dédiée à un complexe cinématographique précis (celui de Verviers par exemple). Elle contient les programmations des films, l’archivage des programmations, les informations des films passés sur CC ainsi que les copies de films programmés. CB envoie régulièrement des copies de films à CC pour alimenter sa programmation.
* **MKT (Marketing)** : Base de données dédiée au schéma d’intégration préparant et sélectionnant uniquement les données utiles au datawarehouse.
* **DW (DataWarehouse)** : Base de données dédiée au datawarehouse donc à la prise de décision.

L’application écrite en java permet, quant à elle, de rechercher des films, visualiser des informations les concernant (année, réalisateur, etc.) ainsi que de les noter ou les commenter.



**Remarque** : Pour pallier au problème de réplication infinie lors des transferts de données entre deux bases nous avons utilisé un système de token (champ qui lorsqu’il vaut une certaine valeur est à répliquer, sinon non).

**ATTENTION** vous trouverez dans les différents répertoires des fichiers ScriptCreationXX.sql. Ceux-ci sont essentiels au bon fonctionnement du script d’installation. Toute modification de ceux-ci peut entrainer des erreurs lors de l’installation.

# SYS

## Acl.sql

Création d’un ACL (Access Control List) pour permettre aux différentes bases de données de récupérer des informations de sites internet. Cela sera utile pour le téléchargement des affiches de films stockés dans CB.

## CreateUsers.sql

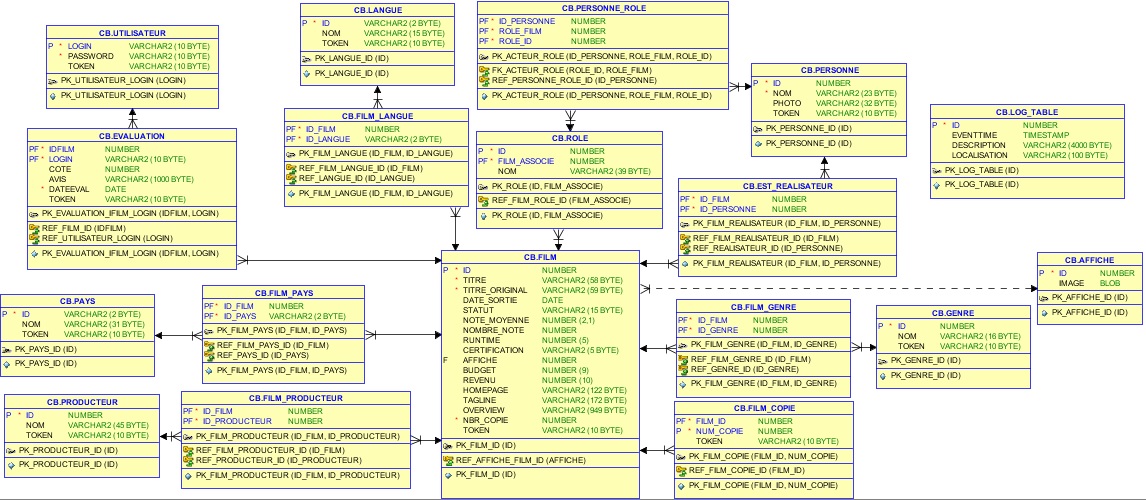
Création d’un rôle avec des privilèges.

Création des utilisateurs avec attribution du rôle prédéfini au préalable.

Autorisation d’utiliser les fonction du package SYS.UTL\_HTTP permettant l’interrogation de site internet.

Création du directory avec autorisation, pour les utilisateurs en ayant besoin, d’aller le lire ou écrire.

# CB



## CreaCB.sql

Création du schéma définitif de CB. La taille de certains champs a été imposée sur base des données fournies par le rapport (CreaRapport) calculé sur des fonctions statistiques.

Création des DB Link (permet la communication entre bases de données).

## CreaExternalTable.sql

Création d’une table à partir d’un fichier situé dans le directory.

## CreaJobAlimCC.sql

Job lancé toutes les semaines à minuit qui effectuera la procédure JOB du package ALIMCC. Un job est un travail s’effectuant à intervalle régulier et défini.

## CreaJobBackup.sql

Job lancé tous les jours à minuit qui effectuera la procédure BACKUP. Un job est un travail s’effectuant à intervalle régulier et défini.

## CreaRapport.sql

Script permettant de générer un rapport statistique sur la longueur des champs extraits du fichier movies.txt . Le rapport sera enregistré dans le directory MOVIEDIRECTORY sous le nom rapport.txt.

Le script parcourt la table externe (voir CreaExternalTable.sql) et calcule la longueur maximale, minimale, l’écart-type et la médiane de la longueur des champs, le nombre de valeurs uniques pour chaque champ, le nombre de valeurs NULL, le nombre de valeurs vides (0 ou chaîne vide), le 100-quantile et le 10000-quantile.

## CreateLogTable.sql

Création de la table LOG\_TABLE (table servant à tracer le passage à travers les différentes fonctions, procédures etc. afin de savoir si l’action a réussi ou échoué).

Création de la procédure LogEvent qui insère dans la table LOG\_TABLE.  
Paramètres entrants : emplacement et message.

Création d’un déclencheur logID pour augmenter le numéro de séquence de l’id de la table LOG\_TABLE.

## CreaXMLCommunicationTable.sql

Création de tables XML servant à envoyer ou recevoir du XML entre CB et CC.

## PackageAlimCC.sql

Ce package rassemble des procédures permettant d’envoyer des copies de film sur CC.

### PROCEDURE MOVIE\_COPY\_GENERATOR

Paramètre entrant : id d’un film.

Un nombre aléatoire de copies du film (maximum la moitié du nombre de copies disponibles pour ce film) est tiré. Si ce nombre est strictement supérieur à 0, on enregistre dans une table binary XML les données du film envoyé et dans une autre table binary XML les copies envoyées de ce film (copies supprimées du coup de CB).

### PROCEDURE JOB

Le but est d’envoyer des copies de chaque film sur CC toutes les semaines (procédure appelée par JOB\_ALIMCC). Pour ca, elle appelle la procédure MOVIE\_COPY\_GENERATOR du même package pour chaque film puis la procédure RECEPTION\_FILM sur CC.

## PackageCB.sql

Ce package rassemble des fonctions permettant la validation des données à insérer.

### FUNCTION VERIF\_FILM\_FIELDS

Paramètres entrants : id, titre, titre original, date de sortie, statut, note moyenne, nombre de notes, durée, certification, lien poster, budget, revenus, homepage, tagline, overview et nombre de copies du film.

Valeur de retour : tuple de film.

Vérification sur la longueur des champs titre, titre original, certification, homepage, tagline, overview, note moyenne, budget et revenus.

Vérification sur la valeur des champs certification, nombre de notes et durée.

### FUNCTION VERIF\_GENRE\_FIELDS

Paramètres entrants : id et nom du genre.

Valeur de retour : tuple de genre.

Vérification sur la longueur du champ nom.

### FUNCTION VERIF\_PRODUCTEUR\_FIELDS

Paramètres entrants : id et nom du producteur.

Valeur de retour : tuple de producteur.

Vérification sur la longueur du champ nom.

### FUNCTION VERIF\_PAYS\_FIELDS

Paramètres entrants : id et nom du pays.

Valeur de retour : tuple de pays.

Vérification sur la longueur du champ nom.

### FUNCTION VERIF\_LANGUE\_FIELDS

Paramètres entrants : id et nom de la langue.

Valeur de retour : tuple de langue.

Vérification sur la longueur du champ nom.

### FUNCTION VERIF\_PERSONNE\_FIELDS

Paramètres entrants : id, nom et URL de l’image de la personne.

Valeur de retour : tuple de personne.

Vérification sur la longueur des champs nom et image.

### FUNCTION VERIF\_ROLE\_FIELDS

Paramètres entrants : id et nom du rôle, id du film.

Valeur de retour : tuple de rôle.

Vérification sur la longueur du champ nom.

## PackageRecherche.sql

Ce package rassemble des fonctions permettant la recherche de divers données selon des critères.

Les fonctions du package renvoient un SYS\_REFCURSOR. C’est un curseur contenant des tuples de résultat lisible également en dehors du SGBD.

### FUNCTION recherche\_id

Paramètres entrants : id du film.

Valeur de retour : SYS\_REFCURSOR avec le film recherché.

Recherche d’un film sur son id.

### FUNCTION recherche

Paramètres entrants : titre, acteurs, réalisateurs, année de sortie, année de sortie minimale, année de sortie maximale du film.

Valeur de retour : SYS\_REFCURSOR avec les films correspondants aux critères fournis.

Recherche des films correspondants à un ou plusieurs critères fournis en paramètre.

### FUNCTION getAfficheFilm

Paramètres entrants : id du film.

Valeur de retour : SYS\_REFCURSOR avec l’affiche du film demandé.

Obtention d’une affiche d’un film sur son id.

### FUNCTION getNoteUtilisateurFilm

Paramètres entrants : id du film.

Valeur de retour : SYS\_REFCURSOR avec la moyenne et le nombre de notes des utilisateurs du film demandé.

Obtention de la moyenne et du nombre de notes des utilisateurs d’un film sur son id.

### FUNCTION getActeursFilm

Paramètres entrants : id du film.

Valeur de retour : SYS\_REFCURSOR avec les acteurs ayant joué dans le film demandé.

Obtention des acteurs jouant dans un film sur son id.

### FUNCTION getRealisateursFilm

Paramètres entrants : id du film.

Valeur de retour : SYS\_REFCURSOR avec les réalisateurs ayant réalisé le film demandé.

Obtention des réalisateurs réalisant un film sur son id.

### FUNCTION getAvisFilm

Paramètres entrants : id du film, numéro de page.

Valeur de retour : SYS\_REFCURSOR avec 5 avis du film demandé.

Obtention de 5 avis d’un film sur son id. Ces avis sont sélectionnés par rapport au numéro de page courant dans l’application cliente car il faut les retourner 5 par 5 selon ce numéro.

## ProcedureAlimCB.sql

Paramètre entrant : Nombre de films à ajouter

Ajout d’un nombre donné de films dans le schéma de CB en utilisant les tuples brutes choisis aléatoirement dans la table externe Movie.

La procédure génère aussi un nombre aléatoire de copies de films suivant une loi normale de moyenne 5 et d’écart-type 2.

Si le film choisi aléatoirement est déjà présent dans le schéma CB le nombre de copies généré est ajouté aux copies déjà existantes.

## ProcedureBackup.sql

Insertion dans CBB des tuples de toutes les tables dont le token indique qu’il n’est pas encore sur l’autre base.

Mise à jour des token.

## ProcedureEvalFilm.sql

Paramètres entrants : id du film, login du user, note, commentaire.

Procédure permettant d’insérer ou de mettre à jour une évaluation.

## ProcedureRetourCopie.sql

Insertion des copies de films récupérées de CC sur CB.

Suppression de ces données sur la table XML de communication.

## TriggerCopieCotesAvis.sql

Déclencheur COPIECOTESAVIS appelé lorsqu’une insertion ou mise à jour est faite sur la table EVALUATION. Il va insérer ou mettre à jour l’évaluation.

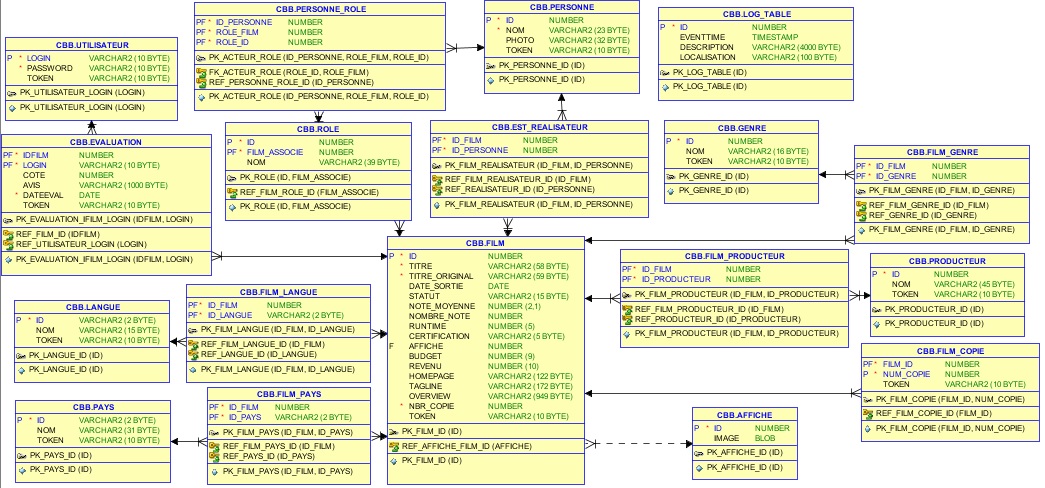
## TriggersCopieFilm.sql

Déclencheur COPIEFILM appelé lorsqu’une copie de film est insérée sur CB. Il va la recopier sur CBB.

Déclencheur DELETECOPIE appelé lorsqu’une copie de film est supprimée sur CB. Il va la supprimer également sur CBB.

# CBB

Copie de la base de données CB destinée à pouvoir prendre la relève si CB est en panne.



## CreaCBB.sql

Création du schéma définitif de CBB. La taille de certains champs a été imposée sur base des données fournies par le rapport (CreaRapport) calculé sur des fonctions statistiques.

Création des DB Link (permet la communication entre bases de données).

## CreateLogTable.sql

Création de la table LOG\_TABLE (table servant à tracer le passage à travers les différentes fonctions, procédures etc. afin de savoir si l’action a réussi ou échoué).

Création de la procédure LogEvent qui insère dans la table LOG\_TABLE.  
Paramètres entrants : emplacement et message.

Création d’un déclencheur logID pour augmenter le numéro de séquence de l’id de la table LOG\_TABLE.

## PackageRecherche.sql

Ce package rassemble des fonctions permettant la recherche de divers données selon des critères.

Les fonctions du package renvoient un SYS\_REFCURSOR. C’est un curseur contenant des tuples de résultat lisible également en dehors du SGBD.

### FUNCTION recherche\_id

Paramètres entrants : id du film.

Valeur de retour : SYS\_REFCURSOR avec le film recherché.

Recherche d’un film sur son id.

### FUNCTION recherche

Paramètres entrants : titre, acteurs, réalisateurs, année de sortie, année de sortie minimale, année de sortie maximale du film.

Valeur de retour : SYS\_REFCURSOR avec les films correspondants aux critères fournis.

Recherche des films correspondants à un ou plusieurs critères fournis en paramètre.

### FUNCTION getAfficheFilm

Paramètres entrants : id du film.

Valeur de retour : SYS\_REFCURSOR avec l’affiche du film demandé.

Obtention d’une affiche d’un film sur son id.

### FUNCTION getNoteUtilisateurFilm

Paramètres entrants : id du film.

Valeur de retour : SYS\_REFCURSOR avec la moyenne et le nombre de notes des utilisateurs du film demandé.

Obtention de la moyenne et du nombre de notes des utilisateurs d’un film sur son id.

### FUNCTION getActeursFilm

Paramètres entrants : id du film.

Valeur de retour : SYS\_REFCURSOR avec les acteurs ayant joué dans le film demandé.

Obtention des acteurs jouant dans un film sur son id.

### FUNCTION getRealisateursFilm

Paramètres entrants : id du film.

Valeur de retour : SYS\_REFCURSOR avec les réalisateurs ayant réalisé le film demandé.

Obtention des réalisateurs réalisant un film sur son id.

### FUNCTION getAvisFilm

Paramètres entrants : id du film, numéro de page.

Valeur de retour : SYS\_REFCURSOR avec 5 avis du film demandé.

Obtention de 5 avis d’un film sur son id. Ces avis sont sélectionnés par rapport au numéro de page courant dans l’application cliente car il faut les retourner 5 par 5 selon ce numéro.

## ProcedureEvalFilm.sql

Paramètres entrants : id du film, login du user, note, commentaire.

Procédure permettant d’insérer ou de mettre à jour une évaluation.

## ProcedureRestore.sql

Insertion dans CB des tuples des tables des utilisateurs, des copies de film et des évaluations dont le token indique qu’il n’est qu’il n’est pas encore sur l’autre base. Les évaluations sont copiées en synchrone par trigger, excepté dans le cas où l’utilisateur a été ajouté durant la journée puisqu’il n’existe pas encore sur l’autre base (copie asynchrone par appel de cette procédure dans ce cas).

Mise à jour des token.

## ProcedureRetourCopie.sql

Insertion des copies de films récupérées de CC sur CBB.

Suppression de ces données sur la table XML de communication.

## TriggerCopieCotesAvis.sql

Déclencheur COPIECOTESAVIS appelé lorsqu’une insertion ou mise à jour est faite sur la table EVALUATION. Il va insérer ou mettre à jour l’évaluation.

## TriggersCopieFilm.sql

Déclencheur COPIEFILM appelé lorsqu’une copie de film est insérée sur CBB. Il va la recopier sur CB.

Déclencheur DELETECOPIE appelé lorsqu’une copie de film est supprimée sur CBB. Il va la supprimer également sur CB.

# CC

## CreateLogTable.sql

Création de la table LOG\_TABLE (table servant à tracer le passage à travers les différentes fonctions, procédures etc. afin de savoir si l’action a réussi ou échoué).

Création de la procédure LogEvent qui insère dans la table LOG\_TABLE.  
Paramètres entrants : emplacement et message.

Création d’un déclencheur logID pour augmenter le numéro de séquence de l’id de la table LOG\_TABLE.

## CreateXMLTable.sql

Création des tables sur base des schémas XSD préalablement enregistrés (voir RegisterXsd.sql).  
Les principales tables créées sont les suivantes :

* filmSchema : Contient toutes les informations relatives à un film
* copieFilm : Contient les copies des films
* programmation : Les programmations de films
* archives : des informations utiles pour des calculs statistique sur les programmations

Création des DBLink permettant de communiquer avec CB et CBB

## ProcedureInsertXML.sql

Cette procédure est exécutée sur demande de CB (voir CB/ProcedureAlimCB et CB/PackageAlimCC)  
La procédure effectue les actions suivantes :

1. Remplissage d'une table (TMPXMLCOPY) qui va contenir les copies de film devant retourner à CB.
2. Lecture de la table de CB contenant les informations de films que CC doit insérer dans sa table filmSchema
3. Lecture de la table de CB contenant les XML copies que CC dans insérer dans sa table COPIEFILM
4. Une fois les tables de CB vue CC efface leur contenu pour ne pas réinsérer les mêmes copies au prochain appel
5. CC appel une méthode de CB qui lire la table TMPXMLCOPY afin de récupérer les copies retournées par CC.

## ProcedureProgFilm.sql

Cette procédure va lire un fichier XML de demande de programmations se trouvant dans la directory MOVIEDIRECTORY. Cette demande de programmation contient une id de film à programmer, le numéro de la salle et une heure de début de séance.

Le dossier XML a insérer doit porter le nom « progra.xml » pour être reconnus par la procédure

La procédure vérifie la validité du XML reçus et va calculer la durée de la séance sur base de la durée du film + 30 min (pub + rangement).

On va ensuite tirer un nombre de jour aléatoire pour avoir la durée de programmation.

On vérifie que la salle est libre dans la tranche horaire demandée et ce pour tous les jours de programmation

On regarde si on a une copie libre pour être programmée à cette heure-là.

Si toutes ces conditions sont remplies on insert la programmation sinon on génère un message de feedback précisant la nature l'erreur. Ce fichier feedback sera généré dans le directory

## Fichiers .xsd

Ce sont des fichiers qui définissent la grammaire des divers XML manipulés par CC. Ces documents seront enregistrés dans le SGBD (voir RegisterXsd.sql) afin de pouvoir vérifier la grammaire des fichiers XML entrants.

**Film.xsd** : Grammaire pour les informations concernant un film (utilisé dans PackageAlimCC)

**Archivages.xsd** : Grammaire pour les informations des programmations archivées

**Copie.xsd** : Grammaire des informations d’une copie de film sur CC

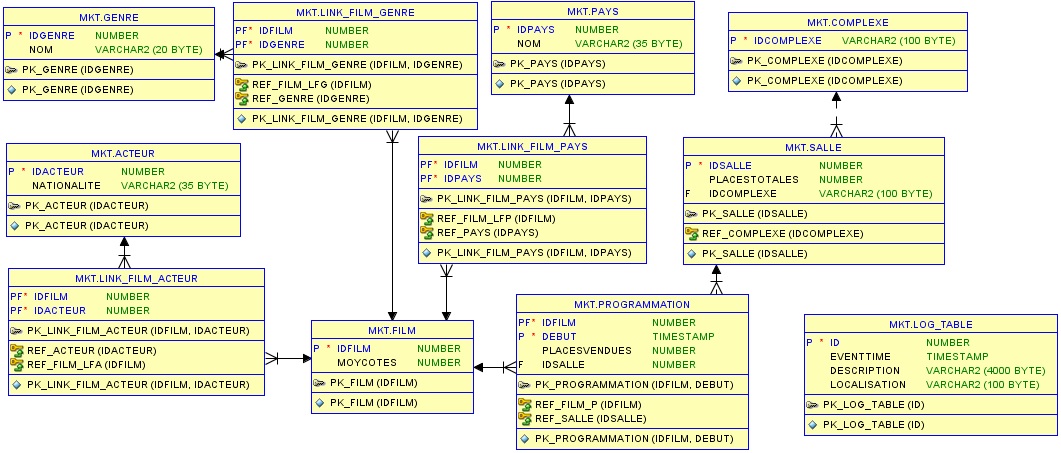
**porgraDemandee.xsd** : Grammaire des programmations créée par un opérateur et placée dans MOVIEDIRECTORY (voir ProcedureProgFilm.sql)

**prograFinale.xsd** : Grammaire des programmations validées et stockées dans CC

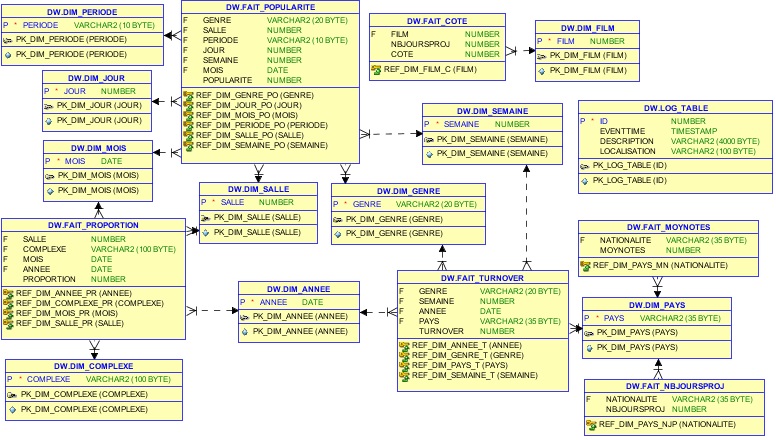
## RegisterXsd.sql

Enregistrement de la grammaire XML pour chaque document XML manipulé par CC.

# MKT



# DW



# Application film

## classApplicationFilm

### Film.java

Classe contenant les informations des films ainsi les getters et setters adéquats.

### ThreadTestConnexion.java

Thread qui appelle la fonction setConnextion du GUI à intervalle régulier.

## GUIapplicationFilm

### AccueilPanel.java

Modification du panel vers FormulaireRecherchePanel lors du click sur le bouton de recherche.

Modification du panel vers ConnexionPanel lors du click sur le bouton de logout, ce qui équivaut à une déconnexion.

### AffichageFilmPanel.java

Remplissage des champs d’affiche du film par appel des fonctions du package PACKAGERECHERCHE.

Modification du panel vers AccueilPanel lors du click sur le bouton d’accueil.

Affichage des avis selon le numéro de page (min 1) par appel de la fonction getAvisFilm du package PACKAGERECHERHE. Lors du click sur précédent, on diminue le numéro de page ; lors du click sur suivant, on l’augmente. Ce numéro ainsi que l’id du film seront passés en paramètres à la fonction.

Ouverture de NoterDialog lors du click sur le bouton d’évaluation.

### ConnexionPanel.java

Récupération du login et mot de passe entrés par l’utilisateur. Comparaison entre ces données et celles disponibles dans la base de données. Si ca correspond, modification du panel vers AccueilPanel.

### FormulaireRecherchePanel.java

Recherche d’un film sur son id par appel de la méthode recherche\_id du package PACKAGERECHERCHE.

Recherche d’un film sur un ou plusieurs critères (titre, acteurs, réalisateurs, année de sortie, année de sortie minimale, année de sortie maximale) par appel de la méthode recherche du package PACKAGERECHERCHE. Remplissage de la liste de résultats avec les résultats retournés de la fonction.

### GUI.java

C’est la fenêtre principale de l’application.

Changement de panel.

Setters et getters (bean d’accès à la base de données, user, résultat).

Setter de la connexion à la base de données pour savoir si on peut se connecter à CB ou bien si on doit basculer sur CBB.

### NoterDialog.java

Récupération de la note et du commentaire entrés par l’utilisateur. Au moins un des deux champs doit être rempli.

Appel de la procédure dans la base de données EVALFILM qui ajoutera ou mettra à jour l’évaluation.

### RechercheResultPanel.java

Modification du panel vers AccueilPanel lors du click sur le bouton d’accueil.

Modification du panel vers AffichageFilmPanel lors du click sur le bouton d’affichage du film sélectionné dans la liste des films trouvés sur la recherche.

Modification du panel vers FormulaireRecherchePanel lors du click sur le bouton de nouvelle recherche.

Remplissage de la jList de résultat des films.