Dossier de maintenance

Fink Jérôme & Seel Océane
2015-2016

Table des matières

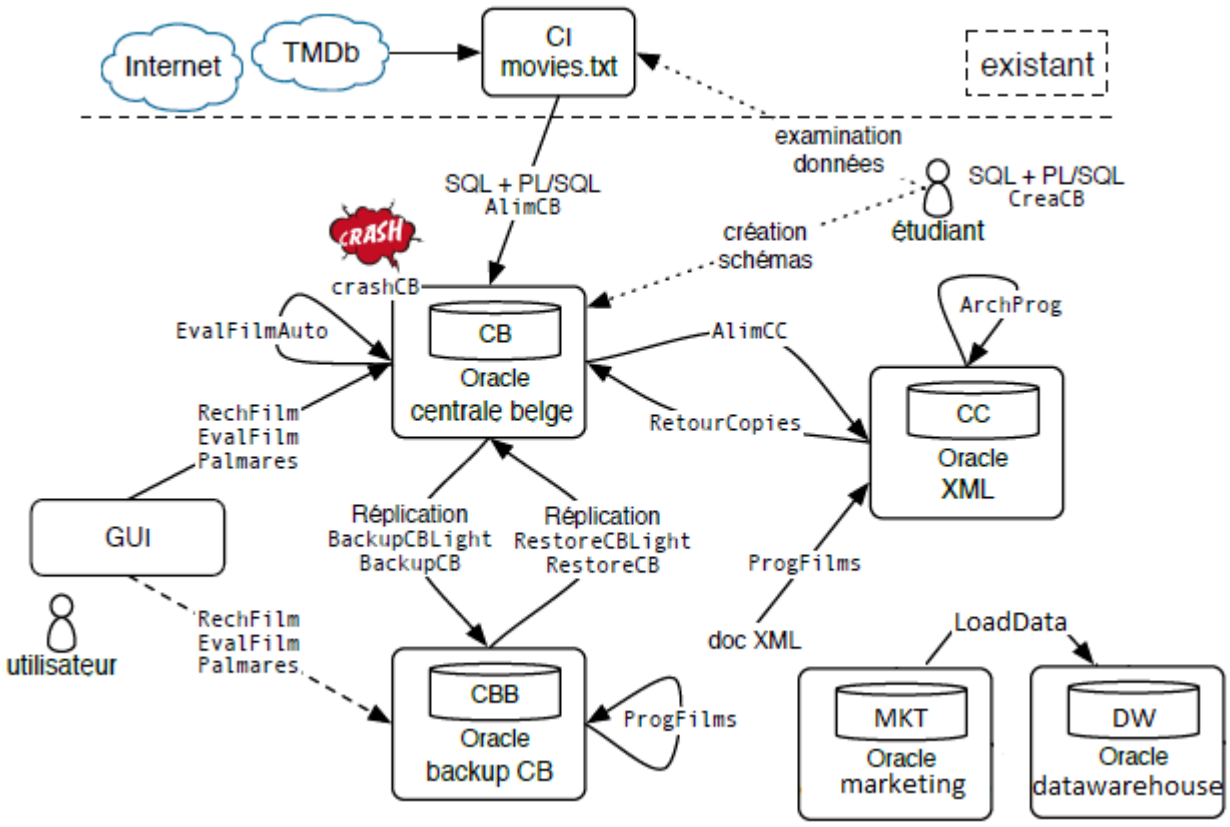
1. Introduction

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.

2. SYS

2.1. 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.

2.2. CreateUsers.sql

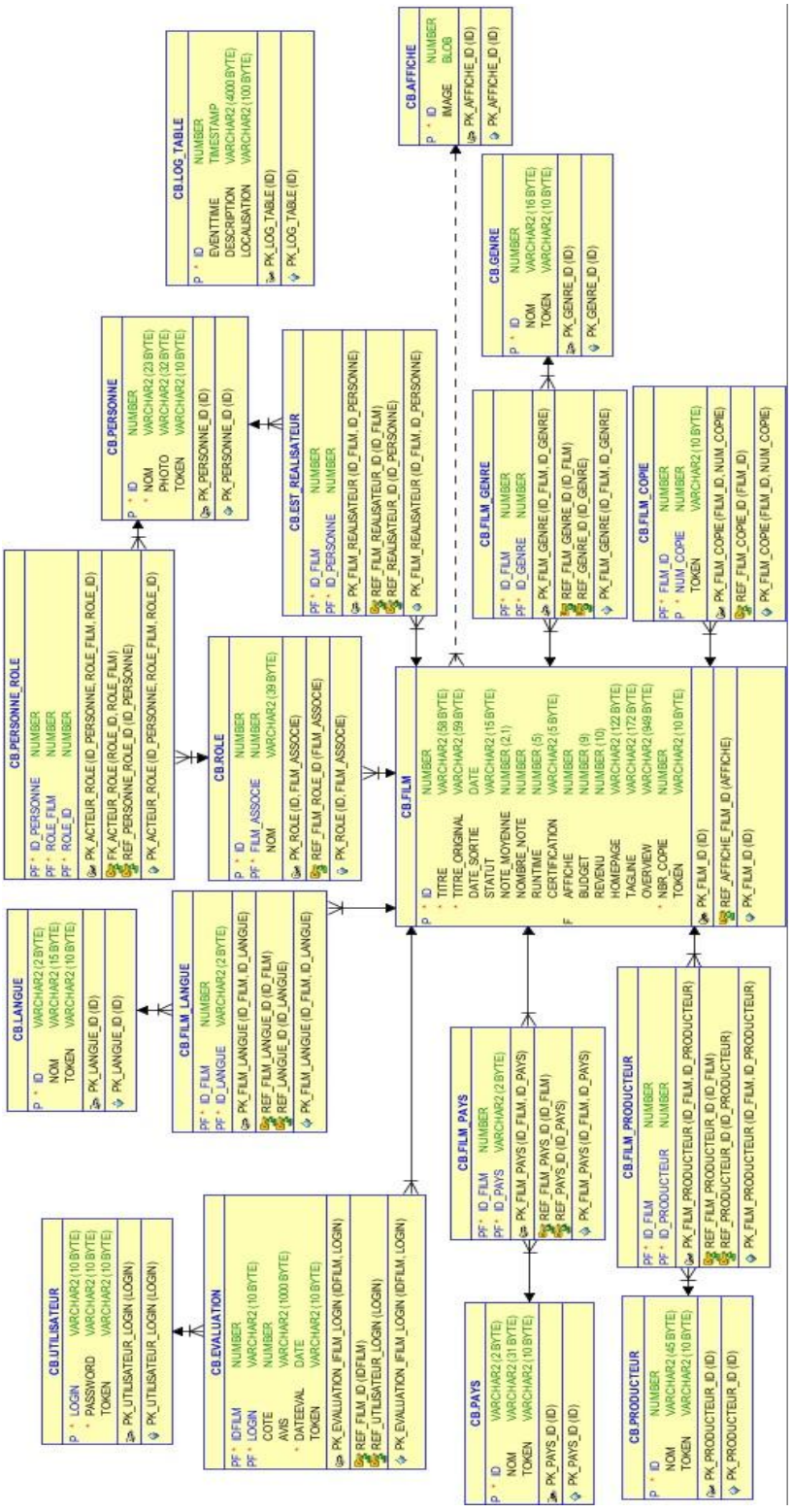
Création d'un rôle avec des privilèges.

Création des utilisateurs avec attribution du rôle prédéfini au préalable.

Autorisation d'utiliser les fonctions 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.

3. CB


CB.AFFICHE
CB.LOG_TABLE
CB.PERSONNE
CB.GENRE
CB.PERSONNE_ROLE
CB.EST_REALISATEUR
CB.FILM_GENRE
CB.FILM_COPIE
CB.LANGUE
CB.ROLE
CB.FILM
CB.FILM_LANGUE
CB.UTILISATEUR
CB.EVALUATION
CB.FILM_PAYS
CB.FILM_PRODUCTEUR
CB.PAYS
CB.PRODUCTEUR

3.1. 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).

3.2. CreaExternalTable.sql

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

3.3. 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.

3.4. 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.

3.5. 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.

3.6. 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.

3.7. CreaXMLCommunicationTable.sql

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

3.8. PackageAlimCC.sql

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

3.8.1. 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).

3.8.2. 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.

3.9. PackageCB.sql

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

3.9.1. 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.

3.9.2. FUNCTION VERIF_GENRE_FIELDS

Paramètres entrants : id et nom du genre.

Valeur de retour : tuple de genre.

Vérification sur la longueur du champ nom.

3.9.3. FUNCTION VERIF_PRODUCTEUR_FIELDS

Paramètres entrants : id et nom du producteur.

Valeur de retour : tuple de producteur.

Vérification sur la longueur du champ nom.

3.9.4. FUNCTION VERIF_PAYS_FIELDS

Paramètres entrants : id et nom du pays.

Valeur de retour : tuple de pays.

Vérification sur la longueur du champ nom.

3.9.5. 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.

3.9.6. 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.

3.9.7. 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.

3.10. 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.

3.10.1. FUNCTION recherche_id

Paramètres entrants : id du film.

Valeur de retour : SYS_REFCURSOR avec le film recherché.

Recherche d'un film sur son id.

3.10.2. 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.

3.10.3. 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.

3.10.4. 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.

3.10.5. 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.

3.10.6. 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.

3.10.7. 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.

3.11. 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.

3.12. 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.

3.13. ProcedureEvalFilm.sql

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

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

3.14. ProcedureRetourCopie.sql

Insertion des copies de films récupérées de CC sur CB.

Suppression de ces données sur la table XML de communication.

3.15. 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.

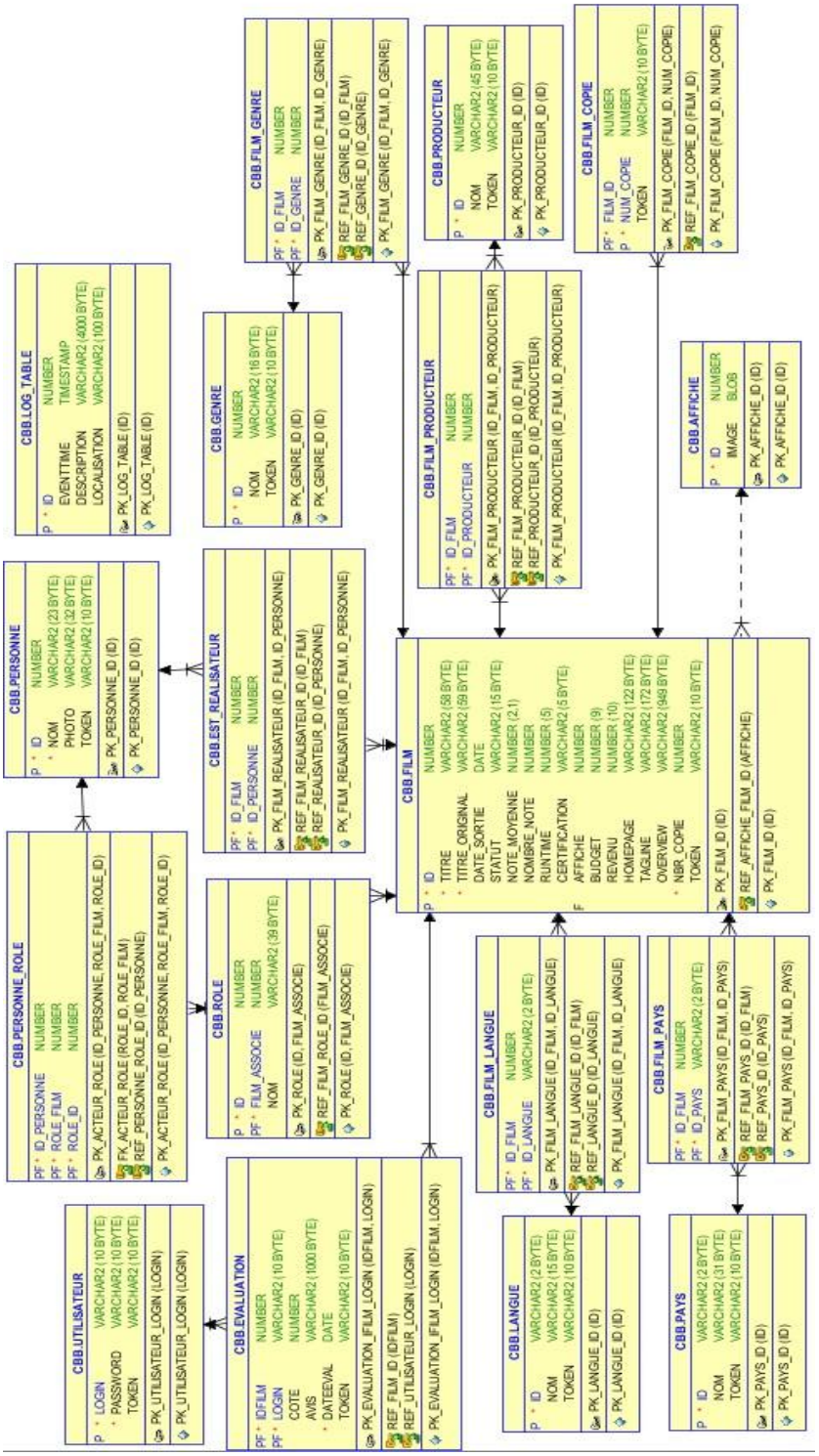
3.16. TriggersCopieFilm.sql

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.

4. CBB

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



4.1. 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).

4.2. 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.

4.3. 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.

4.3.1. FUNCTION recherche_id

Paramètres entrants : id du film.

Valeur de retour : SYS_REFCURSOR avec le film recherché.

Recherche d'un film sur son id.

4.3.2. 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.

4.3.3. 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.

4.3.4. 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.

4.3.5. 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.

4.3.6. 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.

4.3.7. 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.

4.4. ProcedureEvalFilm.sql

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

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

4.5. 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.

4.6. ProcedureRetourCopie.sql

Insertion des copies de films récupérées de CC sur CBB.

Suppression de ces données sur la table XML de communication.

4.7. 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.

4.8. 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.

5. CC

5.1. CreaJobArchProg.sql

Création d'un job qui va exécuter quotidiennement la procédure ArchProg.

5.2. CreaJobProgFilm.sql

Création d'un job qui va exécuter quotidiennement la procédure ProgFilm.

5.3. 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.

5.4. 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 : Contient les programmations de films.
- archives : Contient des informations utiles pour des calculs statistiques sur les programmations.

Création des DBLink permettant de communiquer avec CB et CBB.

5.5. Fichiers .xsd

Ce sont des fichiers qui définissent la grammaire des divers XML manipulés par CC. Une grammaire est le « contrat » que le fichier XML implémentant cette grammaire doit respecter. 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.

5.6. ProcedureArchProg.sql

Procédure qui va calculer des informations sur les différents films qui ont été programmés dans CC. On utilise les informations contenues dans la table programmation pour savoir, pour chaque film, combien de temps il est resté à l'affiche, combien de copies étaient présentes sur CC et combien de spectateurs ont été voir le film. Ce rapport est généré automatiquement tous les jours à minuit et est stocké dans la table archives.

5.7. 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 de CC (TMPXMLCOPY) qui va contenir les copies de films devant retourner sur CB.
2) Suppresion de ces copies retournées de la table des copies de CC.
3) Lecture de la table de CB (TMPXMLMOVIE) contenant les informations de films que CC doit insérer dans sa table FILMSCHEMA et insertion de ces données.
4) Lecture de la table de CB (TMPXMLCOPY) contenant les copies de films que CC doit insérer dans sa table COPIEFILM et insertion de ces données.
5) Suppression du contenu des deux tables de communication de CB pour ne pas réinsérer les mêmes copies au prochain appel.
6) Appel d'une méthode de CB qui lit la table TMPXMLCOPY de CC afin de récupérer les copies retournées par CC.

5.8. ProcedureProgFilm.sql

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

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

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

Un nombre de jours aléatoire est tiré pour la durée de programmation.

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

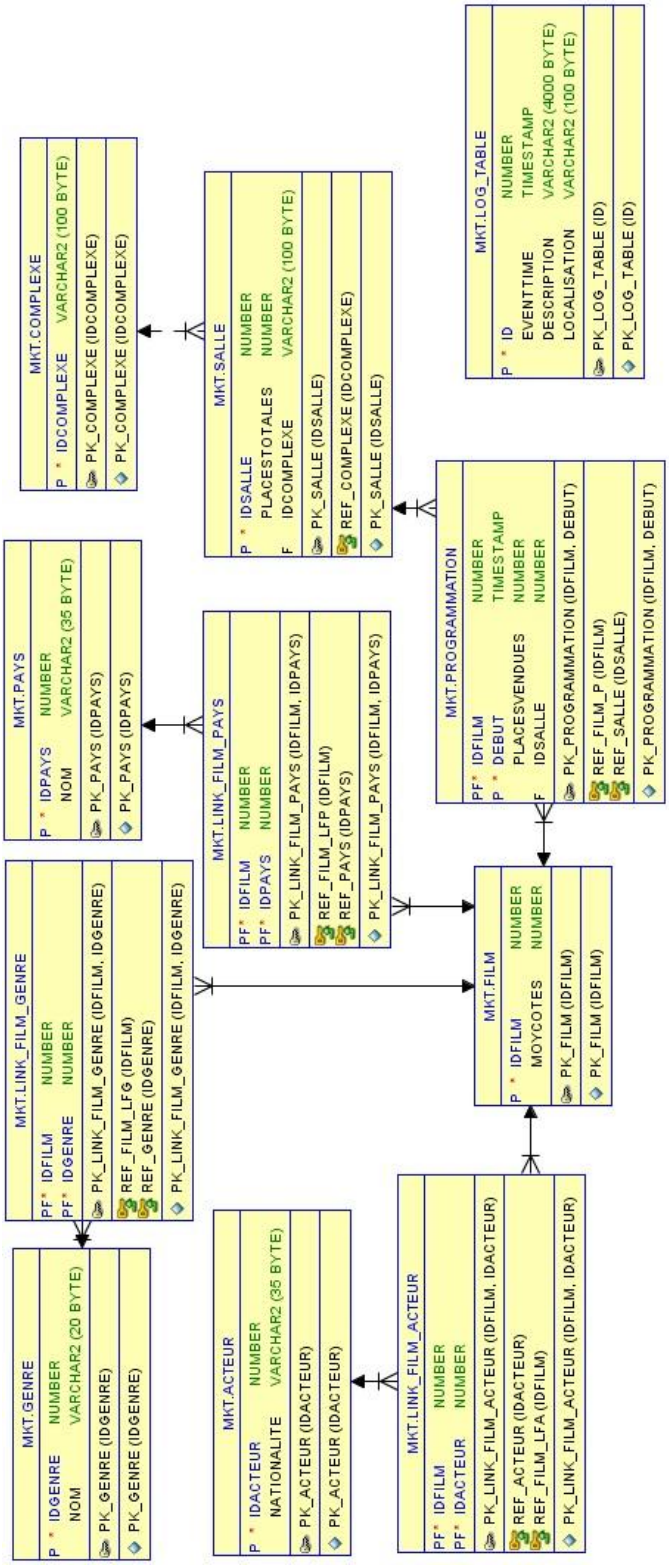
On vérifie la disponibilité d'une copie du film pour être programmée à cette heure-là.

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

5.9. RegisterXsd.sql

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

6. MKT


MKT.COMPLEXE
P * IDCOMPLEXE VARCHAR2 (100 BYTE)
PK_COMPLEXE (IDCOMPLEXE)
PK_COMPLEXE (IDCOMPLEXE)
MKT.SALLE
P * IDSALLE NUMBER
PLACESTOTALES NUMBER
F IDCOMPLEXE VARCHAR2 (100 BYTE)
PK_SALLE (IDSALLE)
REF_COMPLEXE (IDCOMPLEXE)
PK_SALLE (IDSALLE)
MKT.LOG_TABLE
P * ID NUMBER
EVENTTIME TIMESTAMP
DESCRIPTION VARCHAR2 (4000 BYTE)
LOCALISATION VARCHAR2 (100 BYTE)
PK_LOG_TABLE (ID)
PK_LOG_TABLE (ID)
MKT.PAYS
P * IDPAYS NUMBER
NOM VARCHAR2 (35 BYTE)
PK_PAYS (IDPAYS)
PK_PAYS (IDPAYS)
MKT.LINK_FILM_PAYS
PF* IDFILM NUMBER
PF* IDPAYS NUMBER
PK_LINK_FILM_PAYS (IDFILM, IDPAYS)
REF_FILM_LFP (IDFILM)
REF_PAYS (IDPAYS)
PK_LINK_FILM_PAYS (IDFILM, IDPAYS)
MKT.PROGRAMMATION
PF* IDFILM NUMBER
P * DEBUT TIMESTAMP
PLACESVENDUES NUMBER
F IDSALLE NUMBER
PK_PROGRAMMATION (IDFILM, DEBUT)
REF_FILM_P (IDFILM)
REF_SALLE (IDSALLE)
PK_PROGRAMMATION (IDFILM, DEBUT)
MKT.LINK_FILM_GENRE
PF* IDFILM NUMBER
PF* IDGENRE NUMBER
PK_LINK_FILM_GENRE (IDFILM, IDGENRE)
REF_FILM_LFG (IDFILM)
REF_GENRE (IDGENRE)
PK_LINK_FILM_GENRE (IDFILM, IDGENRE)
MKT.FILM
P * IDFILM NUMBER
MOYCOTES NUMBER
PK_FILM (IDFILM)
PK_FILM (IDFILM)
MKT.GENRE
P * IDGENRE NUMBER
NOM VARCHAR2 (20 BYTE)
PK_GENRE (IDGENRE)
PK_GENRE (IDGENRE)
MKT.ACTEUR
P * IDACTEUR NUMBER
NATIONALITE VARCHAR2 (35 BYTE)
PK_ACTEUR (IDACTEUR)
PK_ACTEUR (IDACTEUR)
MKT.LINK_FILM_ACTEUR
PF* IDFILM NUMBER
PF* IDACTEUR NUMBER
PK_LINK_FILM_ACTEUR (IDFILM, IDACTEUR)
REF_ACTEUR (IDACTEUR)
REF_FILM_LFA (IDFILM)
PK_LINK_FILM_ACTEUR (IDFILM, IDACTEUR)

6.1. AlimMKT.sql

Script d'insertion des données soit en dur, soit à partir de données aléatoires. Utilisation de MERGE pour pouvoir relancer le script (un tel script est destiné à être lancé par un job à intervalle de temps régulier).

6.2. CreaMKT.sql

Création du schéma d'intégration de MKT. Un schéma d'intégration est un MRD réduit contenant uniquement les informations nécessaires aux requêtes de l'utilisateur auxquelles le datewarehouse répondra ainsi que le nécessaire pour respecter la structure du MRD.

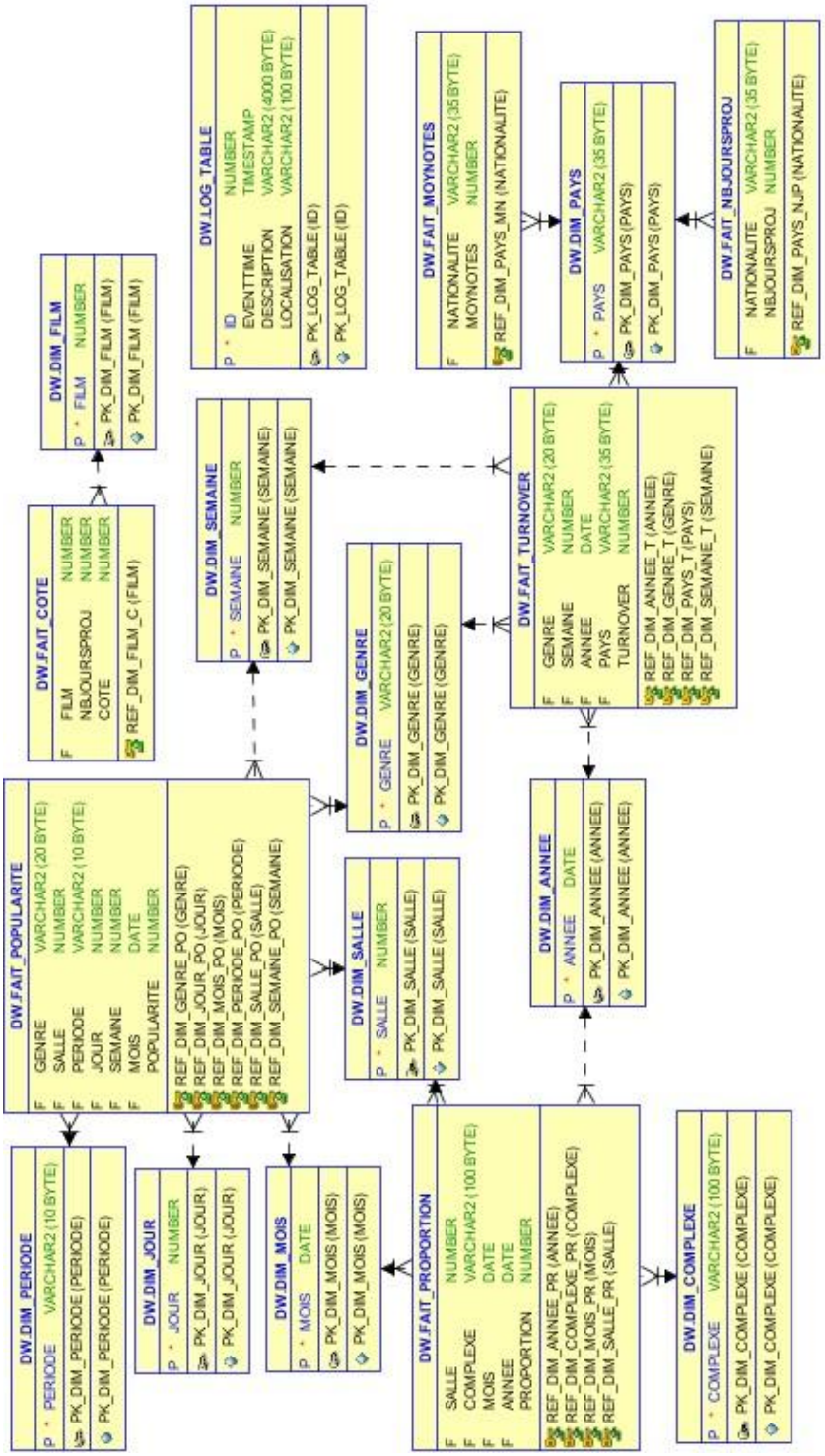
6.3. CreateLogTable.sql

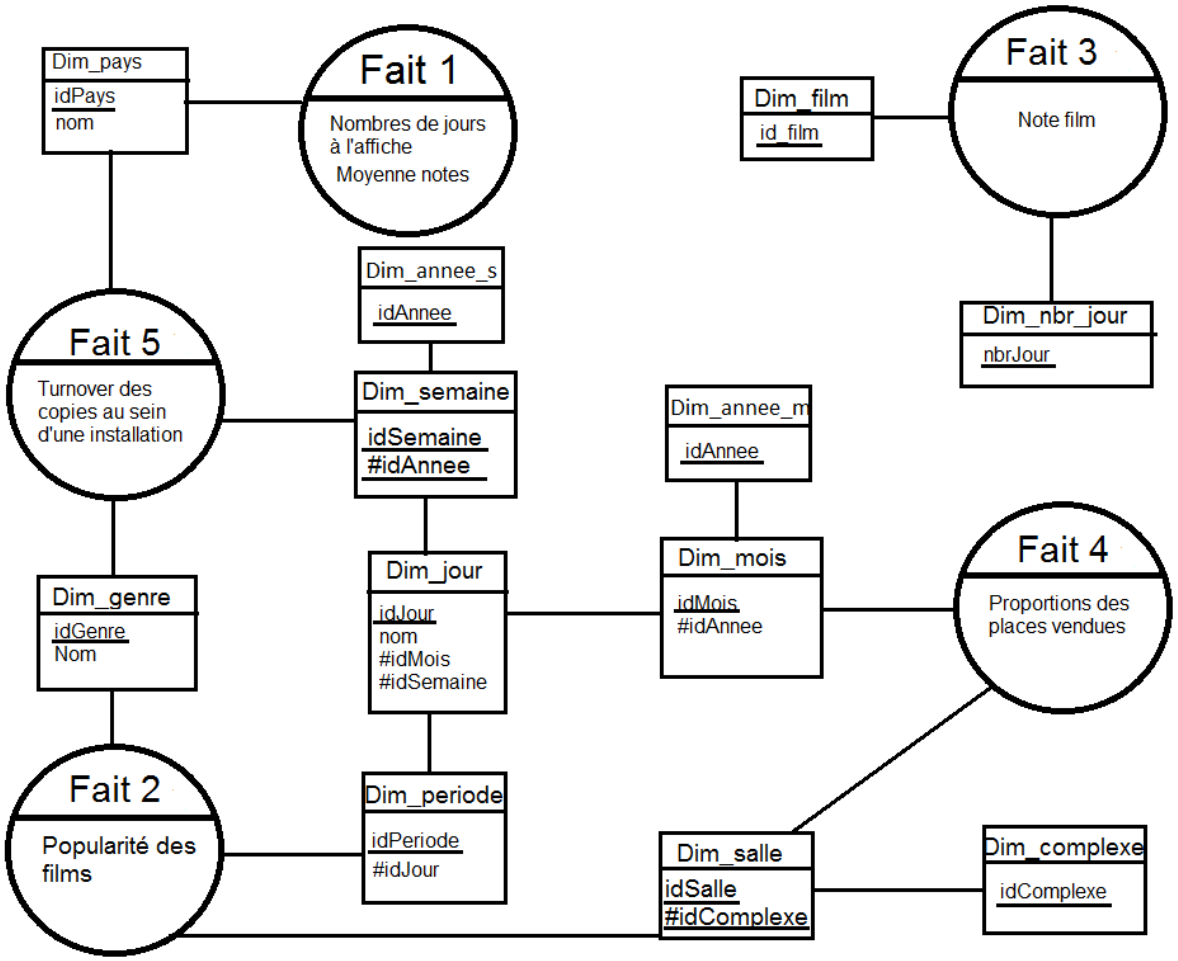
Création de la table LOG_TABLE (table servant à tracer le passage à travers les différentes fonctions, procédures etc. afin de savoir si l'action a réussi ou échoué).

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.

7. DW


DW.LOG_TABLE
P * ID NUMBER
EVENTTIME TIMESTAMP
DESCRIPTION VARCHAR2 (4000 BYTE)
LOCALISATION VARCHAR2 (100 BYTE)
PK_LOG_TABLE (ID)
PK_LOG_TABLE (ID)
DW.FAIT_MOYNOTES
F NATIONALITE VARCHAR2 (35 BYTE)
MOYNOTES NUMBER
REF_DIM_PAYS_MN (NATIONALITE)
DW.DIM_PAYS
P * PAYS VARCHAR2 (35 BYTE)
PK_DIM_PAYS (PAYS)
PK_DIM_PAYS (PAYS)
DW.FAIT_NBJOURSPROJ
F NATIONALITE VARCHAR2 (35 BYTE)
NBJOURSPROJ NUMBER
REF_DIM_PAYS_NJP (NATIONALITE)
DW.DIM_FILM
P * FILM NUMBER
PK_DIM_FILM (FILM)
PK_DIM_FILM (FILM)
DW.FAIT_COTE
F FILM NUMBER
NBJOURSPROJ NUMBER
COTE NUMBER
REF_DIM_FILM_C (FILM)
DW.DIM_SEMAINE
P * SEMAINE NUMBER
PK_DIM_SEMAINE (SEMAINE)
PK_DIM_SEMAINE (SEMAINE)
DW.FAIT_TURNOVER
F GENRE VARCHAR2 (20 BYTE)
F SEMAINE NUMBER
F ANNEE DATE
F PAYS VARCHAR2 (35 BYTE)
F TURNOVER NUMBER
REF_DIM_ANNEE_T (ANNEE)
REF_DIM_GENRE_T (GENRE)
REF_DIM_PAYS_T (PAYS)
REF_DIM_SEMAINE_T (SEMAINE)
DW.DIM_GENRE
P * GENRE VARCHAR2 (20 BYTE)
PK_DIM_GENRE (GENRE)
PK_DIM_GENRE (GENRE)
DW.FAIT_POPULARITE
F GENRE VARCHAR2 (20 BYTE)
F SALLE NUMBER
F PERIODE VARCHAR2 (10 BYTE)
F JOUR NUMBER
F SEMAINE NUMBER
F MOIS DATE
F POPULARITE NUMBER
REF_DIM_GENRE_PO (GENRE)
REF_DIM_JOUR_PO (JOUR)
REF_DIM_MOIS_PO (MOIS)
REF_DIM_PERIODE_PO (PERIODE)
REF_DIM_SALLE_PO (SALLE)
REF_DIM_SEMAINE_PO (SEMAINE)
DW.DIM_SALLE
P * SALLE NUMBER
PK_DIM_SALLE (SALLE)
PK_DIM_SALLE (SALLE)
DW.DIM_ANNEE
P * ANNEE DATE
PK_DIM_ANNEE (ANNEE)
PK_DIM_ANNEE (ANNEE)
DW.DIM_PERIODE
P * PERIODE VARCHAR2 (10 BYTE)
PK_DIM_PERIODE (PERIODE)
PK_DIM_PERIODE (PERIODE)
DW.DIM_JOUR
P * JOUR NUMBER
PK_DIM_JOUR (JOUR)
PK_DIM_JOUR (JOUR)
DW.DIM_MOIS
P * MOIS DATE
PK_DIM_MOIS (MOIS)
PK_DIM_MOIS (MOIS)
DW.FAIT_PROPORTION
F SALLE NUMBER
F COMPLEXE VARCHAR2 (100 BYTE)
F MOIS DATE
F ANNEE DATE
PROPORTION NUMBER
REF_DIM_ANNEE_PR (ANNEE)
REF_DIM_COMPLEXE_PR (COMPLEXE)
REF_DIM_MOIS_PR (MOIS)
REF_DIM_SALLE_PR (SALLE)
DW.DIM_COMPLEXE
P * COMPLEXE VARCHAR2 (100 BYTE)
PK_DIM_COMPLEXE (COMPLEXE)
PK_DIM_COMPLEXE (COMPLEXE)



7.1. CreaDW.sql

Création du schéma de diffusion de DW. Un schéma de diffusion est une structure informationnelle mettant l'information à la disposition des utilisateurs afin de répondre à leurs questions.

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

7.2. 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.

7.3. PackageAlimDW.sql

Package regroupant les procédures pour alimenter les faits et les dimensions

7.3.1. PROCEDURE LOAD_DIMENSIONS

Procédure privée appelée par la procédure LOAD. Elle alimente les dimensions à partir des données se trouvant sur MKT.

7.3.2. PROCEDURE LOAD_FAITS

Procédure privée appelée par la procédure LOAD. Elle alimente les faits à partir des données se trouvant sur MKT.

ATTENTION : Par manque de temps, cette procédure est en commentaire et seulement la structure existe, il serait donc bon de la compléter.

7.3.3. PROCEDURE LOAD

Procédure appelant les procédures LOAD_DIMENSIONS et LOAD_FAITS (actuellement en commentaire et non finie) qui elles-mêmes alimentent les dimensions et les faits de DW. Procédure idéale pour être appelée par un job à intervalle régulier.

8. Application film

8.1. classApplicationFilm

8.1.1. Film.java

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

8.1.2. ThreadTestConnexion.java

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

8.2. GUIapplicationFilm

8.2.1. AccueilPanel.java

Modification du panel vers FormulaireRechercheP anel 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.

8.2.2. 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.

8.2.3. 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.

8.2.4. FormulaireRechercheP anel.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.

8.2.5. 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.

8.2.6. 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.

8.2.7. 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 FormulaireRechercheP anel lors du click sur le bouton de nouvelle recherche.

Remplissage de la jList de résultat des films.