Rennequinepolis (RQS)

Version du 20 avril 2016 17:58

Réalisé avec XeLaTeX

UE Organisation et gestion de données 1, AA Bases de données
UE Organisation et gestion de données 2, AA Bases de données avancées

1er Master en sciences de l'ingénieur industriel

Laurence Herbiet <laurence.herbiet@hepl.be>
Ludovic Kuty <ludovic.kuty@hepl.be>

2015 – 2016

Table des matières

1 Préambule

Ce document présente les deux projets à réaliser dans le cadre des laboratoires des activités d'apprentissage (AA) suivantes :

- Bases de données de l'UE Organisation et gestion de données 1
- Bases de données avancées de l'UE Organisation et gestion de données 2

Le premier projet est réalisé durant le premier quadrimestre dans le cadre de l'AA "Bases de données" (AA 1) et est individuel. Le second projet est réalisé durant le second quadrimestre dans le cadre de l'AA "Bases de données avancées" (AA 2) et est le fruit du travail d'équipe de plusieurs étudiants.

Ce document explique le contexte général du système informatique qui devra être réalisé et détaille ensuite les différentes applications qui le constituent. A la fin du document se trouvent les règles d'évaluation du cours.

Toutes les ressources sont disponibles sur la page Web du cours à `https://cours.khi.be/sgbd_m1/`. Donc lorsqu'il est fait référence dans le texte à un fichier précis en indiquant son nom mais pas son URL, vous pouvez aller le chercher sur la page Web en question.

2 Contexte général

Rennequinepolis (RQS) est une société qui distribue des films et qui gère les complexes cinématographiques qui projettent ces films. L'objectif de ce travail est d'informatiser leur système actuel et de construire les différentes plate-formes web de consultation, d'achats et de gestion qu'ils souhaitent ainsi que de réaliser les transferts d'informations nécessaires.

Le schéma de la figure 1, page 4, donne une vue d'ensemble des différentes fonctionnalités et des flux d'informations.

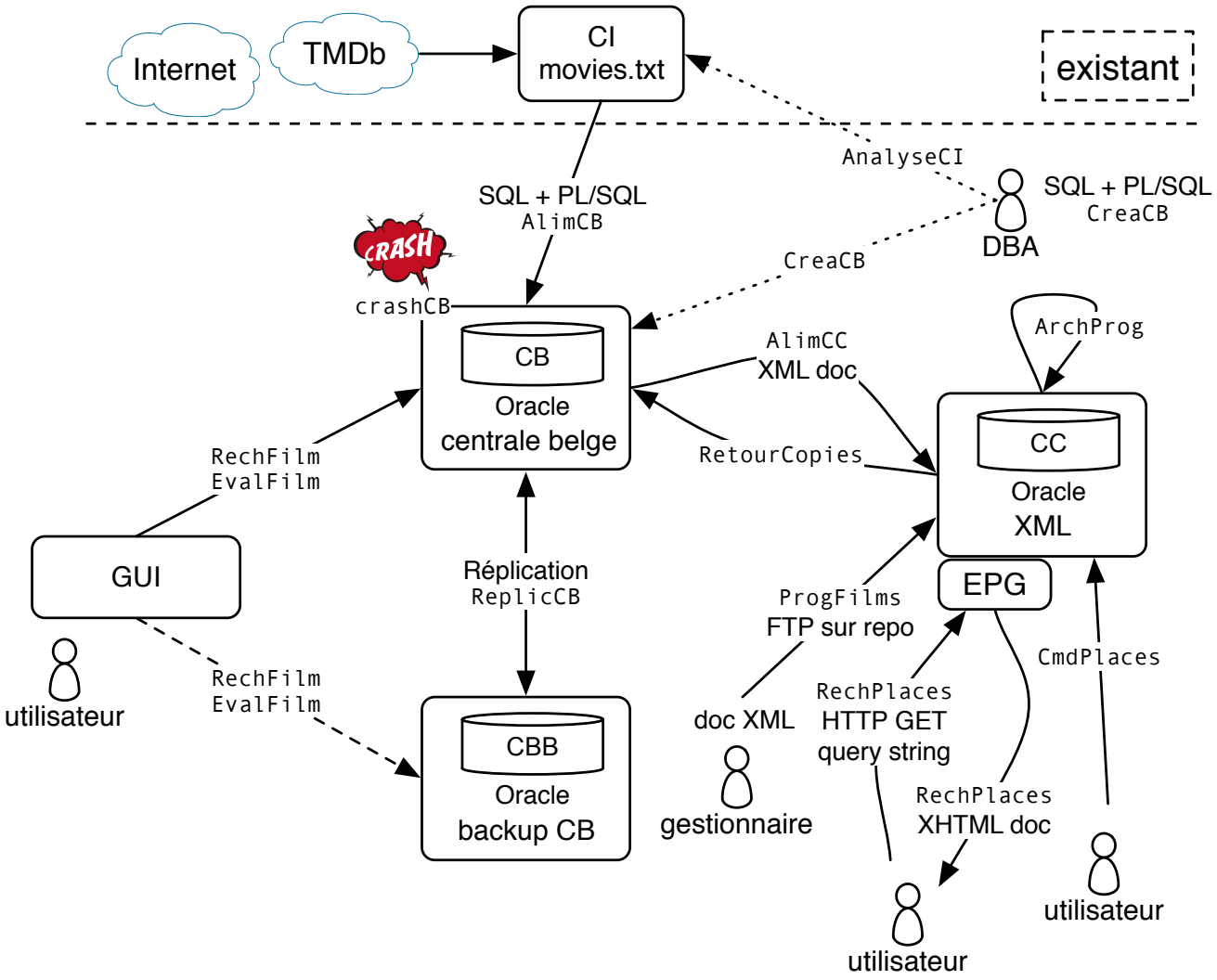
Nous insistons sur le fait que cette section permet d'établir le contexte dans lequel vous devrez réaliser les fonctionnalités qui vous sont demandées (et uniquement celles-là). Le contexte peut contenir plus d'informations que ce dont vous aurez pratiquement besoin pour réaliser les fonctionnalités demandées. Référez-vous au tableau 2 des fonctionnalités demandées, page 13, pour savoir ce que vous devez exactement réaliser et allez lire les fonctionnalités en question pour savoir ce dont vous avez besoin exactement.

2.1 Sites

Rennequinepolis est organisée en divers services, localisés à Bruxelles, en plus des complexes disséminés au travers du pays. Elle s'adresse à différents fournisseurs pour obtenir ses médias et en générer des copies qu'elle peut ensuite distribuer dans les différents cinémas.

- **La centrale internationale - CI** - Cette centrale est un entrepôt où sont disponibles toutes les oeuvres cinématographiques que pourraient souhaiter les entreprises comme RQS. Elle tient lieu de fournisseur principal de l'entreprise.
- **La centrale belge - CB** - Il s'agit de notre entreprise RQS. Située à Bruxelles, l'entreprise y dispose d'un entrepôt où elle peut stocker les copies avant qu'elles ne soient distribuées dans les cinémas. On y trouve également les serveurs principaux de RQS.
- **La centrale alternative belge - CBB** - Site de secours de RQS, il est capable de prendre le relais en cas de panne du site principal. Il propose les mêmes services que CB.
- **Le cinéma - CC** - Les différents complexes cinématographiques que gère RQS. Dans le cadre de cet énoncé il n'y aura qu'un cinéma : CC.

Fig. 1: Schéma global



2.2 Intervenants

Liste des intervenants provisoirement relevés :

- Les **gestionnaires des cinémas** - Ils sont responsables des opérations courantes des complexes dont l'alimentation en nouveaux films.
- **Les fournisseurs** - Ils sont représentés par la centrale internationale et pour certaines données par la banque de données des personnes. Il n'y a donc qu'un seul fournisseur qui est considéré comme implicite.
- Les **utilisateurs** - Les utilisateurs ont la possibilité de coter des films et de donner leur avis écrit sur un film.

2.3 Les films

La signalétique d'un film comporte toutes les informations relatives à ce film. Par exemple, son titre, sa durée, ses genres, le public concerné, les acteurs principaux, une affiche, …. Chaque film est une ligne (un

enregistrement) stocké dans un fichier texte appelé `movies.txt`[1] accessible depuis la base de données **CB** à partir d'une table externe `movies_ext`. La table externe est considérée comme étant la base de données **CI**. On retrouve 15 champs par film expliqués au tableau 1.

Le format d'un enregistrement de la table externe est le suivant :

```
_id(U+2059)title(U+2059)release_date(U+2059)status(U+2059)vote_average(U
+2059)vote_count(U+2059)runtime(U+2059)certification(U+2059)poster_path(U
+2059)budget(U+2059)tagline(U+2059)overview(U+2059)genres:id(U+201E)name(U
+2059)directors:id(U+201E)name(U+201E)profile_path(U+2059)actors:id(U+201E)
name(U+201E)profile_path
```

Ou encore : `_id::title::release_date::status::vote_average::vote_count::runtime::certification::poster_path::budget::tagline::overview::genres:id„name::directors:id„name„profile_path::actors:id„name„profile_path`.

Attention, les caractères séparateurs proviennent de Unicode et sont représentés par leur "code point" Unicode en hexadécimal placé entre parenthèses comme `(U+2059)`.

Certains champs peuvent être absent ce qui est représenté par deux séparateurs consécutifs `(U+2059)(U+2059)`.

Les séparateurs utilisés pour chaque enregistrement (ou ligne du fichier) texte sont les suivants :

- Le caractère `::` est utilisé pour séparer les champs qui sont au niveau le plus haut de la hiérarchie (top-level). Il s'agit du caractère Unicode appelé FIVE DOT PUNCTUATION dont le code est `U+2059`[2].
- Le nom indiqué devant les deux-points est le nom du champ contenant un tableau comme `genres`, `directors` et `actors`. Notez que le nom du champ et les deux-points ne font pas partie des données. Donc par exemple "`genres:`" n'est pas présent dans les données textuelles.
- Le caractère `‖` est utilisé comme séparateur des éléments du tableau. Il s'agit du caractère Unicode appelé DOUBLE VERTICAL LINE dont le code est `U+2016`[3].
- Le caractère `„` est utilisé pour séparer les champs d'un élément dans un tableau. Il s'agit du caractère Unicode appelé DOUBLE LOW-9 QUOTATION MARK dont le code est `U+201E`[4].

Pour le film Inception (id 27205)[5], on a l'enregistrement ci-dessous dans le fichier texte `movies.txt` :

```
27205::Inception::2010-07-16::Released::7.5::5578::148::PG-13::/tAXARVreJnWfoANIHASmgYk4SB0.jpg::160000000
::Yourmindisthesceneofthecrime.::Cobb,askilledthiefwhocommitscorporateespionagebyinfiltratingthesubco
nsciousofhistargetsisofferedachancetoregainhisoldlifeaspaymentforataskconsideredtobeimpossible:"inc
eption",theimplantationofanotherperson'sideaintoatarget'ssubconscious.::28„Action‖12„Adventure‖9648„
Mystery‖878„ScienceFiction‖53„Thriller::525„ChristopherNolan„/7OGmfDF4VHLLgbj×uEwTj3ga0uQ.jpg::6193„L
eonardoDiCaprio„/mNRMgj7K5ztvkSqrcdwEYNZIS1M.jpg‖24045„JosephGordon-Levitt„/zSuXCR6×CKIgo0gWLdp8moM
lH3I.jpg‖27578„EllenPage„/paexwBfmlVyzva7q4XgcBdqowmL.jpg‖2524„TomHardy„/aYWIZ7ywOFzyK7B1HHCQj2nq8s
4.jpg‖3899„KenWatanabe„/edm2ZInBtzjVB3p8AbkKTw2I1nt.jpg‖2037„CillianMurphy„/tEERcmKFHXypcwkP28M×Vfv
f6Ph.jpg‖8293„MarionCotillard„/m1zdHPhh×CBCSDm8b9MRWonbYrq.jpg‖3895„MichaelCaine„/nlIjMLp9zvvYM2eFm
77UhI7s1nW.jpg‖95697„DileepRao„/f9×QkvWw5guOePcC×kso1i09w12.jpg‖13022„TomBerenger„/ieHokBikODTVskOL
wQ71Az88LYK.jpg‖4935„PetePostlethwaite„/o1W1V3jUKP0HvXbduvYjXM5tnaF.jpg‖526„LukasHaas„/1J9ca3TFCzK1
ywmacqMnVBDU7X2.jpg‖66441„TalulahRiley„/cmA8krd4hpPC9kWSF9wpS×6ffMt.jpg
```

[1] `https://cours.khi.be/sgbd3/labo/movies.rar`

[2] `http://unicode-table.com/en/2059/` ou `http://www.fileformat.info/info/unicode/char/2059/index.htm`

[3] `http://unicode-table.com/en/2016/` ou `http://www.fileformat.info/info/unicode/char/2016/index.htm`

[4] `http://unicode-table.com/en/201E/` ou `http://www.fileformat.info/info/unicode/char/201E/index.htm`

[5] L'enregistrement peut être obtenu à l'aide d'un grep avec expression régulière, option -P pour choisir les expressions régulières PCRE à la Perl : `grep -P "^27205::" movies.txt`.

Tab. 1: Informations à propos d'un film

Nom	Explications
`actors`	Un tableau d'acteurs. Chaque acteur contient les informations suivantes: `id`, `name` et `profile_path` qui contient un chemin permettant de construire l'URL vers une image de la personne.
`budget`	Le coût du film en dollars.
`certification`	Le rating qui indique le public cible du film. Les abréviations sont tirées de la notation proposée par l'association MPAA[6] et sont les seules autorisés.
`directors`	Un tableau de réalisateurs. Chaque réalisateur contient les informations suivantes: `id`, `name` et `profile_path` qui contient un chemin permettant de construire l'URL vers une image de la personne.
`genres`	Les genres dont fait partie le film. Un tableau d'objets dont chacun possède les informations suivantes: `id` qui est l'identifiant du genre sous forme d'un nombre entier et `name` qui est le nom du genre comme "Action", "Comedy" ou "Thriller".
`id`	L'identifiant TMDb du film.
`poster_path`	Un chemin permettant de construire l'URL vers une image de type poster pour le film. La manière de procéder est indiquée à la section 2.4, page 7.
`release_date`	La date de sortie du film (dans le fichier texte elles sont au format `YYYY-MM-DD`).
`runtime`	La durée du film en minutes.
`status`	L'état de production du film.
`tagline`	Le texte qui accompagne le film et qui est souvent écrit sur la pochette du film.
`title`	Le titre du film. Pour rappel, les données sont encodées en UTF-8.
`vote_average`	La cote qu'a obtenu le film suite aux votes des membres TMDb.
`vote_count`	Le nombre de votes TMDb qui a servi à établir le `vote_average`.

[6] `http://www.mpaa.org/film-ratings/`

2.4 Les images

Les images des films et des personnes sont accessibles par HTTP en vous basant sur les informations disponibles ci-dessous. La propriété `images` de l'objet JSON que l'on peut reprendre[7] en accédant à `http://api.themoviedb.org/3/configuration` contient un objet avec différentes propriétés :

- `base_url` qui est l'URL de base à utiliser pour reprendre l'image d'un film ou d'une personne.
- `poster_sizes` pour disposer des tailles possibles des images de type poster de film. Vous prendrez une image par film dont la taille est `w185`.
- `backdrop_sizes` que vous n'utiliserez pas.
- `profile_sizes` pour disposer des tailles possibles des images de type photo de personne. Vous prendrez une image par personne dont la taille est `w185`.

L'URL pour reprendre une image est construite en concaténant dans l'ordre :

1. La `base_url`
2. La taille
3. Le chemin renseigné dans le document

Par exemple, pour le film 348, "Alien", on concatènera les chaînes de caractères suivantes :

- `http://image.tmdb.org/t/p/`
- `w185`
- `/uU9R1byS3USozpzWJ5oz7YAkXyk.jpg`

pour avoir `http://image.tmdb.org/t/p/w185/uU9R1byS3USozpzWJ5oz7YAkXyk.jpg`

Listing 1: Configuration


```
{"images":{"base_url":"http://image.tmdb.org/t/p/","secure_base_url":"https
://image.tmdb.org/t/p/","backdrop_sizes":["w300","w780","w1280","original"],"
logo_sizes":["w45","w92","w154","w185","w300","w500","original"],"
poster_sizes":["w92","w154","w185","w342","w500","w780","original"],"
profile_sizes":["w45","w185","h632","original"],"still_sizes":["w92","w185","
w300","original"]},"change_keys":[...]}
```


3 Bases de données

3.1 Infrastructure au sein de l'école

Si vous avez besoin d'utiliser les bases de données de l'école, contactez Ludovic Kuty qui vous créera les schémas appropriés.

3.2 Utilisation d'Oracle sur votre machine

Si vous ne souhaitez pas travailler sur les machines de l'école, nous vous invitons néanmoins à garder la même façon de nommer les schémas de manière à rester le plus clair possible et permettre aussi un portage aisé de votre sur l'infrastructure de l'école si cela s'avérait nécessaire.

Vous pouvez procéder de différentes manières :

[7] Il est nécessaire de disposer d'une clé pour utiliser leur API. Une fois qu'on a cette clé, il suffit d'ajouter le paramètre `api_key` dans l'URL.

- Utiliser une VM pour Oracle.
- Installer directement Oracle sur votre OS.

Si vous souhaitez utiliser Oracle en VM, nous vous invitons à la demander à votre professeur de laboratoire. C'est une VM de type Oracle VM VirtualBox. Par conséquent il sera nécessaire d'installer VirtualBox sur votre machine. Celui-ci peut coexister avec les produits de VMWare. Concrètement vous allez prendre une image nommée `Oracle Developer Day.ova` de 4.3GB.

Le nom d'utilisateur et le mot de passe de la VM sont tous deux `oracle`. Le mot de passe de `root` est également `oracle`.

Une fois la VM installée, démarrez-la pour déterminer son adresse IP. Lancez un terminal (Applications/Accessories/Terminal) et l'affichage de l'IP aura lieu automatiquement. Sinon vous pouvez toujours devenir `root` et demander l'IP :

```
[oracle@localhost ~]$ su -
Password:
[root@localhost ~]# ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:E7:8F:A3
          inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0
...
```

Ensuite, il est nécessaire de fournir un accès aux ports 1521 et 1158 en écrivant deux règles de port forwarding dans votre VM. Il faut choisir la VM, menu Machine/Settings (ou bouton Settings), onglet Network, Adapter 1, partie Advanced, Port Forwarding, et installer les deux règles suivantes :

- Rule 1 – TCP – 127.0.0.1 – 1521 – <IP de la VM> – 1521
- Rule 2 – TCP – 127.0.0.1 – 1158 – <IP de la VM> – 1158

Ensuite en vous connectant avec SQL Developer à partir de votre machine hôte sur `127.0.0.1:1521` vous aurez accès à votre BD Oracle. Notez que vous pouvez remplacer `127.0.0.1` par votre IP ou `0.0.0.0` dans les deux règles ci-dessus et ainsi permettre à d'autres de se connecter à votre BD. On vous conseille de travailler à partir de votre machine hôte sur votre BD. Configurez SQL Developer avec les informations suivantes :

- Username : SYS
- MDP : oracle
- Role : SYSDBA
- Hostname : 127.0.0.1
- Port : 1521
- SID : orcl

Si vous avez déjà une installation d'Oracle présente sur votre machine, il est souhaitable d'utiliser un autre port que 1521 (voire que 1158 également) sur votre machine physique. Vous pouvez choisir ce que vous souhaitez.

4 Consignes générales de développement

Cette section vous donne des indications générales sur ce que vous pouvez faire, ce que vous ne pouvez pas faire, ce à quoi il faut apporter une attention particulière, etc. lorsque vous réaliserez les différentes fonctionnalités demandées à la section.

4.1 Personnalisation des fonctionnalités

Un certain nombre de points ont été simplifiés par rapport à la réalité. Certains points ont également été volontairement maintenus dans le vague ou l'obscur. Dans tous les cas, l'imagination est la bienvenue pour donner à vos solutions un cachet plus personnel, plus réaliste, plus professionnel. Cependant, pour vous éviter de vous fourvoyer dans une impasse ou perdre votre temps à réaliser des options présentant peu d'intérêt, il vous est vivement recommandé de prendre conseil auprès de l'un de des professeurs concernés par le projet.

4.2 Gestion des erreurs

On vous demande d'utiliser les mécanismes de gestions des erreurs/exceptions mis à votre disposition dans le ou les langages de programmation utilisés pour le développement de vos applications ainsi que les erreurs générées par la base de données quel que soit le contexte (PL/SQL, JDBC, …).

Toute erreur liée à la base de données doit obligatoirement être enregistrée de manière détaillée dans une table. Une idée intéressante lorsque cela est possible est de produire deux types de message en cas d'erreur : un message généraliste mais clair destiné à l'utilisateur lambda et un message stocké précis destiné au développeur ou à l'administrateur de la base de données indiquant exactement quelle est l'erreur et à quel endroit elle a eu lieu.

Il est vivement conseillé d'utiliser un package ou une procédure indépendante pour gérer ces insertions dans une table de log des erreurs et les différentes exceptions pouvant se produire. De même, tous les messages d'exception transmis au programme utilisateur ou au serveur web proviendront de messages prédéfinis, stockés dans la base de données et associés à un code d'exception.

On s'attend à ce que les données enregistrées relatives à l'erreur contiennent au minimum la date complète (date + heure), l'endroit précis où a eu lieu l'erreur (package / procédure / déclencheur / …), le numéro de l'erreur et le message d'erreur correspondant. Ces informations sont destinées à l'administrateur de la base.

Attention, qu'il est nécessaire de différencier les messages informationnels non liés à une erreur des messages liés à une erreur. Vous pouvez utiliser une colonne pour cela ou même une table différente.

Techniques attendues : transactions autonomes, séquences.

4.3 Couche ORM ou pas ?

Quel que soit le langage de programmation choisi ou le framework web choisi, nous vous demandons de n'utiliser aucun intermédiaire de type ORM (Object Relational Mapping) entre votre code et les bases de données Oracle. Par conséquent, il est nécessaire que vous écriviez vous-même les requêtes SQL que vous soumettrez à votre BD.

Néanmoins on demande à ce que vous encapsuliez proprement le code d'accès aux données dans vos programmes de manière à bien séparer la partie "modèle" (le M de MVC) du reste du code. Dans cet esprit, on vous demande de veiller à privilégier l'utilisation de procédures stockées.

Le but de ces restrictions est de favoriser l'utilisation du langage SQL au lieu du langage ou système d'interrogation de votre ORM dont l'expressivité est nettement moindre. De plus l'utilisation d'une couche aussi générique ajoute inutilement son lot de complexité dans le cadre de ce projet. Cela se justifie pleinement dans un cours de base de données et non de programmation orientée objet.

Vous pouvez lire le court article[8] intitulé "What ORMs have taught me: just learn SQL" si vous désirez un complément d'information.

[8] `http://wozniak.ca/what-orms-have-taught-me-just-learn-sql`

A vous de voir ce que propose votre langage de programmation pour accéder à une base de données Oracle. Nous en discutons dans la section suivante.

4.4 Choix du langage de programmation

Le choix du langage de programmation à utiliser se pose lors de la réalisation de certaines fonctionnalités.

Ce choix est libre mais nous attirons votre attention sur les points suivants :

- Le choix supporté par le cours est d'utiliser JDBC pour accéder à votre BD Oracle. Cela implique d'utiliser la JVM. Le langage supporté par le cours est Java. Notez cependant qu'il existe d'autres langages qui tournent sur la JVM.
- Un autre choix est d'utiliser ODP.NET avec le langage C#. Cette solution n'est pas supportée par le cours, c'est-à-dire qu'en cas de problèmes nous ne pourrons vous apportez qu'une aide limitée.
- Les autres choix ne sont pas supportés par le cours. Attention aux problèmes liés au manque de mises à jour et de documentation des implémentations d'accès à Oracle. Parfois elles n'existent tout simplement pas et il faut passer par un pont ODBC. Bien que ce soit une très bonne idée d'explorer d'autres langages que ceux enseignés au sein de l'école, la perte de temps qui peut résulter de ces choix nous pousse fortement à déconseiller leur utilisation. Néanmoins, nous désirons laisser le choix du langage libre.

  Attention, pour certaines fonctionnalités, nous imposons l'appel de procédures stockées. On vous demande de prêter attention à l'existence d'un support pour l'utilisation des BLOBs qui sont obligatoires et des types tableau et des variables de type curseur (REF CURSOR).

5 Réalisations attendues AA 1

5.1 CreaCBLight - Script SQL initial de création de CB et CBB

Dans un premier temps, vous allez réaliser les schémas relationnels de CB et CBB sans disposer des tables nécessaires pour stocker les informations des films. On désire simplement pouvoir stocker les informations des utilisateurs et leurs cotes et avis sur les films qui seront identifiés par un nombre entier positif. Il n'y aura donc pas de FK définie à ce stade vers la table contenant les films puisque celle-ci n'existe pas.

On souhaite disposer de la date-heure de la cote ou de l'avis qu'un utilisateur a donné pour un film.

Notez que le schéma de CBB est la copie conforme de celui de CB. On vous demande de produire un script SQL propre disposant d'un maximum de contraintes pour garantir sa cohérence.

5.2 Opérations sur les utilisateurs

Vous allez écrire un package PL/SQL qui permet d'effectuer les quatres opérations de base sur les utilisateurs :

- L'ajout d'un nouvel utilisateur : create (C).
- La recherche et l'affichage des données d'un utilisateur : read (R).
- La mise à jour des données d'un utilisateur : update (U).
- L'effacement d'un utilisateur : delete (D).

On souhaite que les sous-programmes (procédures et fonctions) de notre package n'exposent que ce qui est nécessaire à travers leur interface. On vous demande également de prévoir des scripts PL/SQL pour tester chacune des fonctionnalités avec les différents cas de figure imaginables et une bonne gestion des exceptions.

Veillez aussi à logger les erreurs dans une table. Cela a déjà été précisé à la section 4, page 8 mais nous préférons insister sur ce point.

5.3 UserAdd - Ajout d'un utilisateur

Lorsqu'on crée un nouvel utilisateur on doit pouvoir fournir les informations suivantes, certaines étant facultatives : nom, sexe, email (unique), date de naissance, adresse (facultatif). L'adresse servira d'adresse de facturation et de livraison par défaut. Notez que les clients peuvent disposer de plusieurs adresses de livraison. Les adresses additionnelles sont fournies lors de la validation des commandes de tickets.

La clé primaire utilisée pour identifier un utilisateur est un nombre généré automatiquement et non un nombre introduit par l'utilisateur. On vous demande d'utiliser une séquence Oracle (instruction `CREATE SEQUENCE`). On garde les dates-heures de création et de dernière modification d'un client (pensez à utiliser des colonnes dont le nom est similaire à `created_at` et `updated_at`).

Les contraintes ci-dessous doivent être respectées pour que l'utilisateur puisse être ajouté à la base de données. Notez que la vérification de ces contraintes doit être réalisée de la manière la plus proche possible des données. En effet, on peut imaginer que la création d'un utilisateur puisse également se faire par l'exécution directe d'une instruction `INSERT` au lieu d'utiliser le package prévu à cet effet.

- Le client doit avoir au moins 18 ans.
- Le sexe est `M` ou `F`.
- Le nom doit commencer par une majuscule.
- La date de modification ne peut pas être antérieure à la date de création.

5.4 UserRead - Lecture des informations d'un utilisateur

On vous demande de permettre d'implémenter quelques opérations :

- (UserReadId) Afficher toutes les informations d'un utilisateur particulier lorsqu'on possède son identifiant.
- (UserReadList) Afficher la liste de tous les utilisateurs sans donner les détails. On se contentera de l'identifiant, du nom et de la date de naissance. Utilisez la technique des `BULK COLLECT`.
- (UserReadSearch) Afficher la liste des utilisateurs selon des critères de recherche combinables :
  - Sur tout ou une partie du nom.
  - Sur le sexe.
  - Sur la date de naissance. On peut faire une recherche $\leq$, $\geq$ ou $=$ basée sur l'année, le mois et l'année ou le jour, le mois et l'année.

  Utilisez la technique des `BULK COLLECT`.

5.5 UserUpdate - Modification d'un utilisateur

On peut modifier le nom, l'email et la date de naissance d'un utilisateur. Ecrivez des sous-programmes pour réaliser cela. Les contraintes doivent toujours être vérifiées.

5.6 UserDel - Effacement d'un utilisateur

Vous allez réaliser deux types d'effacement distincts : physique et logique. Pour l'effacement physique, on vous demander d'utiliser deux techniques différentes : à l'aide de contraintes différées (deferrable) et à l'aide d'effacements en cascade (on delete cascade). Vous utiliserez des instructions `ALTER TABLE` pour mettre

en place la première technique avant de tester l'effacement et ensuite vous remettrez tout dans l'état initial comme si rien ne s'était passé (à part les données effacées). Ensuite vous testerez la seconde technique et vous ferez de même.

Ensuite vous implémenterez l'effacement logique, c'est-à-dire que vous marquerez les données comme effacées sans les effacer physiquement. On gardera cette technique d'implémentation comme définitive. On vous demande de garder comme information la date-heure d'effacement de l'utilisateur.

Dans les trois cas de figure, vous afficherez les clients avant et après le ou les effacements grâce à la fonctionnalité UserRead (section 5.4, page 11).

5.7 EvalFilmLight - Ajout de cotes et d'avis

Un utilisateur donné peut ajouter une cote à un film et cette cote peut être accompagné d'un avis textuel. Il peut également donner une nouvelle cote à un film qu'il a déjà évalué, ce qui écrase l'ancienne valeur. Cette nouvelle cote peut disposer d'un nouvel avis qui écrase également l'ancien avis.

5.8 BackupCBLight - Sauvegarde de CB 1

Vous allez réaliser deux réplications de manière à sauvegarder les informations de CB dans CBB :

- Une réplication synchrone (déclencheurs) pour copier les cotes des utilisateurs et leurs avis.
- Une réplication asynchrone chaque nuit pour copier les nouveaux utilisateurs créés durant la journée ainsi que leurs cotes et avis (qui n'ont pas été copiés de manière synchrone). Lorsque vous compléterez plus tard cette fonctionnalité pour réaliser BackupCB, vous ajouterez la sauvegarde des informations liées aux films. Nous testerons cela en invoquant directement votre procédure sans utiliser `DBMS_SCHEDULER`. Veillez à gérer correctement votre transaction.

Techniques attendues : SQL, PL/SQL, déclencheurs, jobs (`DBMS_SCHEDULER`), transactions, séquences, logs.

5.9 RestoreCBLight - Restauration de CB 1

Lorsque CB n'est plus accessible, le système informatique utilise CBB et donc toute une série d'informations peuvent être créées, modifiées ou effacées sur CBB dont de nouveaux utilisateurs, des cotes et des avis. Une fois que la connectivité vers CB est restaurée, il est nécessaire de recopier ces nouvelles informations sur CB. C'est l'objectif de cette fonctionnalité.

Il s'agit de transférer les ajouts, mises à jour et effacements sur CB de manière à ce que les informations contenues dans les deux bases de données soient équivalentes.

Nous testerons cela à l'aide d'un script PL/SQL qui est chargé de faire des ajouts/modifications/effacements ainsi que d'invoquer le sous-programme de restauration vers CB. Notez qu'on ne demande pas de tester la disponibilité de CB de manière automatique.

Techniques attendues : SQL, PL/SQL, déclencheurs, transactions, séquences, logs.

5.10 UserReadWeb - Liste des utilisateurs via le Web

On vous demande de configurer et d'utiliser le Embedded PL/SQL Gateway (EPG) de manière à fournir la liste des utilisateurs à l'instar de la sous-fonctionnalité UserReadList. La réponse sera fournie en HTML et testable directement dans un navigateur web. Veillez à bien distinguer le code de génération des pages web du code qui accède à la base de données. Utilisez la technique des `BULK COLLECT`.

5.11 EvalReadWeb - Liste des cotes et avis via le Web

On vous demande d'utiliser le Embedded PL/SQL Gateway (EPG) de manière à fournir la liste des cotes et avis des utilisateurs pour un film donné dont on connaît l'identifiant. Vous fournirez l'identifiant dans l'URL d'accès à la ressource. La réponse sera fournie en HTML et testable directement dans un navigateur web. Veillez à bien distinguer le code de génération des pages web du code qui accède à la base de données. Utilisez la technique des `BULK COLLECT`.

6 Modalités d'évaluation de l'AA 1

Cette section indique les modalités d'évaluation de l'AA "Bases de données".

6.1 Tableau synthétique

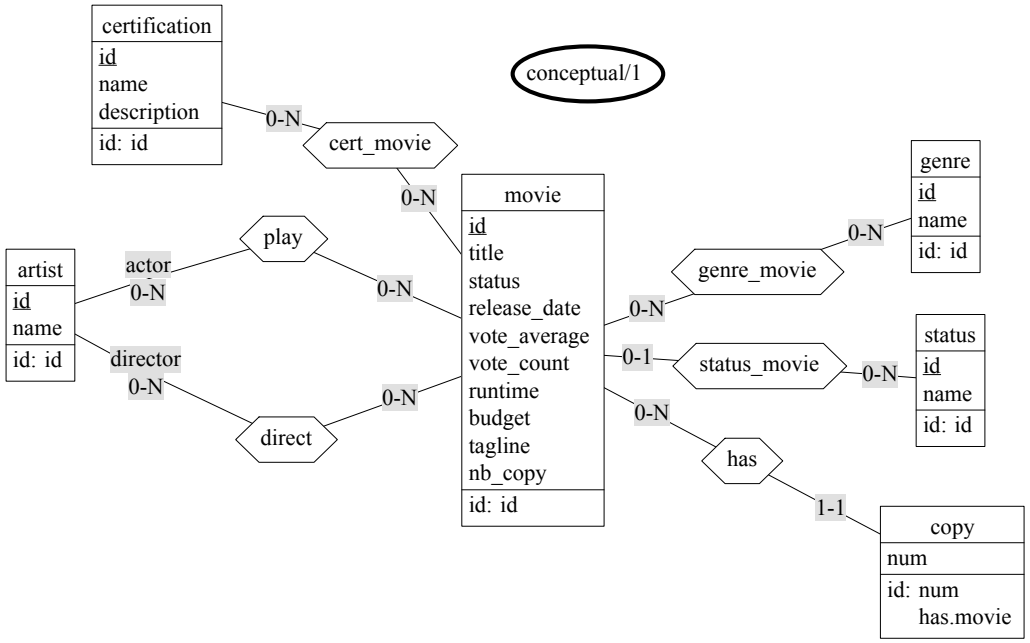
Tab. 2: Pondération des fonctionnalités

Fonctionnalité	Pond
CreaCBLight	15
UserAdd	15
UserRead	20
EvalFilmLight	10
UserUpdate	10
UserDel	20
BackupCBLight	15
RestoreCBLight	15
UserReadWeb	15
EvalReadWeb	15
Total	150

6.2 Modalités d'évaluation du laboratoire

- L'évaluation de laboratoire du premier quadrimestre relative est composée d'une évaluation continue et d'un examen oral.
- La cote de laboratoire constitue 60% de la cote finale de l'AA. La théorie compte pour 40%.
- Chaque étudiant travaille seul.
- Vous pouvez présenter les fonctionnalités quand vous le désirez durant les séances de laboratoires avec la contrainte que les fonctionnalités CreaCBLight, UserAdd, UserRead et UserUpdate doivent avoir été présentées avant le congé d'automne, c'est-à-dire le jeudi 29/10/2015 au plus tard.
- La cote de laboratoire de cette AA représente la somme des points des différentes fonctionnalités réalisées au cours du quadrimestre. Les points attribués à chaque fonctionnalité sont indiqués dans le tableau 2.
- En janvier, il est possible si vous le souhaitez de proposer une amélioration de fonctionnalité(s) nécessairement présentée(s) précédemment. Dans ce cas, la cote de la fonctionnalité sera la moyenne arithmétique des cotes obtenues aux deux présentations.
- Les étudiants sont libres d'organiser et de répartir leur travail comme ils le souhaitent en tenant compte des règles indiquées ici.

Fig. 2: Schéma conceptuel (modèle EA)



On peut vous demander de modifier certaines fonctionnalités lors de l'évaluation.

7 Réalisations attendues AA 2

Pour les réalisations de la partie 2, on vous demande d'installer[9] la version Enterprise Edition 12.1.0.2.0 de Oracle Database. Une possibilité est d'utiliser Virtual Box avec la VM appelée "Oracle DB Developer VM"[10] utilisée pour le "Oracle Technology Network Developer Day".

7.1 CreaCBLight2 - Script SQL initial de création de CB et CBB

Dans un premier temps, vous allez réaliser les scripts SQL de création des tables de CB et CBB en vous basant sur CreaCBLight (section 5.1, page 10). On désire pouvoir stocker les informations des utilisateurs avec leurs cotes et leurs avis comme précédemment ainsi que les données concernant les films.

Le dimensionnement des attributs de type VARCHAR2 et NUMBER se fera en vous basant sur le maximum des tailles des données de manière à pouvoir tout stocker. On affinera ce dimensionnement par la suite en recréant un nouveau schéma plus précis et contenant davantage de contraintes (cfr. fonctionnalité CreaCB (section 7.4, page 16)).

On vous demande d'utiliser le schéma conceptuel de la figure 2 (page 14) exprimé à l'aide du modèle entité-association (modèle EA) comme point de départ pour la construction de votre schéma. Incorporez celui réalisé avec CreaCBLight précédemment. Vous pouvez traduire les noms en français si vous le souhaitez mais tâchez d'être cohérent.

7.2 ReplicCB - Sauvegarde et restauration de CB

Vous allez réaliser deux types de sauvegarde/restauration :

[9] http://www.oracle.com/technetwork/database/enterprise-edition/downloads/index.html
[10] http://www.oracle.com/technetwork/database/enterprise-edition/databaseappdev-vm-161299.html

- Une sauvegarde de CB vers CBB et une restauration de CBB vers CB à froid (cold) des films. On utilisera une fonctionnalité d'import/export à l'aide d'un job (scheduler). Les copies ne sont pas concernées.
- Une sauvegarde de CB vers CBB et une restauration de CBB vers CB à chaud (hot) des utilisateurs et de leurs cotes et avis. Cela a déjà été évalué avec les fonctionnalités BackupCBLight (section 5.8, page 12) et RestoreCBLight (section 5.9, page 12). On suppose que c'est fonctionnel.

Pour la sauvegarde à chaud, vous allez réaliser deux types de réplications :

- Une réplication synchrone (déclencheurs) pour copier les cotes des utilisateurs et leurs avis.
- Une réplication asynchrone chaque nuit pour copier les nouveaux utilisateurs créés durant la journée ainsi que leurs cotes et avis (qui n'ont pas été copiés de manière synchrone).

Lorsque CB n'est plus accessible, le système informatique utilise CBB ce qui implique que des utilisateurs peuvent être ajoutés et des cotes et/ou avis ajoutés ou modifiés (écrasement des anciennes valeurs) sur CBB. Une fois que la connectivité vers CB est restaurée, il est nécessaire de recopier ces nouvelles informations sur CB. Nous invoquerons cette restauration manuellement et dans une fonctionnalité ultérieure cela sera réalisé automatiquement en détectant la disponibilité de CB.

Il s'agit de transférer les ajouts (utilisateurs, cotes, avis) et mises à jour (cotes, avis) sur CB de manière à ce que les informations contenues dans les deux bases de données soient équivalentes.

Nous testerons cela à l'aide d'un script PL/SQL qui est chargé de faire des ajouts/modifications/effacements ainsi que d'invoquer le sous-programme de restauration vers CB.

Techniques attendues : SQL, PL/SQL, déclencheurs, transactions, séquences, logs, `DBMS_SCHEDULER`.

7.3 AnalyseCI - Analyse des informations de CI

Vous allez analyser avec attention et de manière exhaustive les données contenues dans la table externe (base de données CI) en utilisant une combinaison de SQL et PL/SQL. Ceci va vous permettre de disposer d'une information précise pour ensuite créer des schémas de qualité pour CB et CBB (cfr. CreaCB (section 7.4, page 16)) dont par exemple les attributs des tables sont correctement dimensionnés en fonction des contraintes énoncées.

On estime qu'il est nécessaire de déterminer les informations de statistique descriptive suivantes pour certains champs énumérés ci-dessous : la moyenne, l'écart-type, la médiane, le minimum, le maximum, le nombre total de valeurs, le nombre de valeurs non nulles, le nombre de valeurs nulles, le nombre de valeurs dite zéro (chaîne vide, zéro pour un nombre, …), et si cela a un sens le 99ème 100-quantile, le 9999ème 10000-quantile, le nombre de valeurs uniques et les valeurs en question (comme dans le cas des certifications par exemple).

Chacune de ces informations doit pouvoir être interprétée de manière à acquérir une connaissance plus fine du jeu de données utilisé. On vous demande de réaliser à l'aide du package UTL_FILE un rapport au format texte (un ou plusieurs fichiers texte) contenant les informations statistiques décrites ci-dessus pour chaque champ pertinent. Dans certains cas, ce n'est pas la valeur du champ qui nous intéresse mais plutôt la longueur de celui-ci comme dans le cas du titre par exemple.

Les champs à analyser sont : `_id`, `title`, `release_date`, `status`, `vote_average`, `vote_count`, `runtime`, `certification`, `budget`, `tagline`, `overview`, `genres.id`, `genres.name`, `directors.id`, `directors.name`, `actors.id`, `actors.name`.

La table externe `movies_ext` vous donne accès aux données sans les stocker dans la base de données. On peut donc les analyser avec des requêtes SQL complexes avant de les insérer dans les tables de la base de données. En particulier, ce mécanisme nous permet de découvrir à quoi ressemblent nos données et de pouvoir ainsi dimensionner correctement les champs de nos tables.

Attention, toute recherche sur la table externe implique de la parcourir entièrement étant donné qu'il n'existe

pas d'index sur cette table particulière. Vous veillerez donc à minimiser les accès à la table externe, c'est-à-dire de réaliser vos calculs à l'aide d'un nombre de requêtes minimal (une seule requête si possible).

Parmi les 262601 films présents dans le fichier `movies.txt`, seuls deux d'entre eux sont exclus de la table externe. On peut voir dans le fichier de log (nommé par ex. `MOVIES_EXT_1300_4660`) que les enregistrements (records) 23255 et 164269 ont été rejetés à cause de resp. un champ trop long pour être stocké dans `genres` (1000 caractères max.) et un champ trop long pour être stocké dans `tagline` (2000 caractères max.). Etant donné que nos enregistrements sont délimités par les retours à la ligne, cela signifie que ce sont les films aux lignes 23255 et 164269 de `movies.txt`.

On peut aussi voir dans le fichier contenant les mauvais enregistrements (nommé par ex. `MOVIES_EXT_1300_4660.bad`) les deux films en question. Les numéros des films sont 36574 et 265358.

```
$ wc movies.txt
  262601 13743777 134737903 movies.txt
$ rlwrap sqlplus kuty@//192.168.0.1/orcl
...
kuty@ORCL> select count(*) from movies_ext;

  COUNT(*)
----------
    262599

1 row selected.
$ sed -n 23255p movies.txt | ggrep -o -P "^\d+"
36574
$ sed -n 164269p movies.txt | ggrep -o -P "^\d+"
265358
```

Techniques attendues : SQL, PL/SQL, procédures stockées.

7.4 CreaCB - Script SQL de création de CB et CBB

Vous allez parfaire le schéma de CB (et CBB) en redimensionnant les attributs pour permettre le stockage des films et des informations qui leur sont associées en fonction des contraintes énoncées ci-dessous.

On vous demande de rédiger les scripts de création des tables *en veillant à placer un maximum de contraintes au niveau du schéma.*

Vous ne devez rédiger qu'un seul script SQL de création des tables puisque le contenu des deux bases de données CB et CBB est équivalent. Toutes les contraintes applicatives qui ne peuvent pas être définies au moyen du LDD et qui n'ont pas été définies à l'aide de déclencheurs devront apparaître sous la forme de commentaires dans les scripts.

Choisissez judicieusement vos types de données et la taille de ceux-ci de manière à ce que 99% des informations puissent être stockées telle quelles dans le cas des champs `title`, `tagline`, `directors.name` et `actors.name`. Donc il s'agit des tailles inférieures ou égales au quantile 99% (99ème 100-quantile). Les autres informations seront tronquées ou arrondies de manière à être présentes dans la base de données[11].

On doit pouvoir stocker toutes les informations concernant les champs `_id`, `release_date`, `status`, `vote_average`, `vote_count`, `runtime`, `certification`, `budget`, `genres.id`, `genres.name`, `actors.id` et `directors.id` **pour autant que celles-ci soient correctes**. Cela veut dire par exemple que des certifications ou des genres dont le nom est incorrect ne doivent pas se retrouver dans la base de données. Ou encore un budget négatif ou nul (zéro). A vous de vérifier la qualité de vos données.

[11] Sauf celles qui sont trop grandes. Cfr. AlimCB (section 7.5, page 17)

Attention, on stockera les posters des films sous la forme de BLOBs dans la base de données. Le champ `poster_path` ne sera donc pas stocké tel quel mais servira à aller rechercher l'image du poster du film et à le stocker dans la base de données.

Techniques attendues : SQL, PL/SQL, procédures stockées.

7.5 AlimCB - Alimentation de CB

On alimente CB de deux manières :

- En indiquant un identifiant précis d'un film de manière à ajouter ce film.
- Avec un paramètre n qui indique le nombre de films qu'on désire ajouter dans la base de données CB.

On vous demande de réaliser l'ajout à l'aide de SQL et PL/SQL en utilisant la table externe et en veillant à choisir les films de manière aléatoire. Vous utiliserez pour ce faire un `ORDER BY DBMS_RANDOM.VALUE` et la pseudo-colonne `ROWNUM`.

Nous avons trois cas de figure selon la taille de la donnée considérée pour les champs `title`, `tagline`, `directors.name` et `actors.name`. Notez que les quantiles dépendent du champ considéré et sont tous différents.

1. On doit pouvoir stocker, sans aucune modification liée à la taille, les données dont la taille est inférieure ou égale au 99ème 100-quantile, que l'on appelle aussi le 99ème centile ($\leq 99\%$).
2. On doit pouvoir stocker après troncage ou arrondi, les données dont la taille est inférieure ou égale au 9999ème 10000-quantile ($\leq 99.99\%$).
3. On ne stocke pas les données qui sont au-delà de ce quantile. Elles sont tellement grandes qu'on estime qu'elles contiennent une erreur. On utilisera la valeur NULL dans certains cas et l'absence de tuple dans d'autres cas. Il s'agit des informations dont la valeur est strictement supérieure au 9999ème 10000-quantile ($> 99.99\%$).

Pour chaque film ajouté, on détermine de manière aléatoire le nombre de copies à ajouter au film en se basant sur une distribution normale. On utilise pour cela une moyenne de 5 et un écart-type de 2. Bien entendu, le nombre de copies commandées pour chaque film doit être un nombre entier strictement supérieur à 0. Le stock des copies devra être mis à jour.

On vous demande d'ajouter les films et leurs informations associées (acteurs, réalisateurs, …). Bien entendu, on n'ajoute pas et on ne modifie pas des données déjà présentes. Chaque fois qu'un film est ajouté dans la base de données et qu'il y a eu une altération des informations, indiquez le dans une table de log en précisant quelles valeurs ont été altérées avant de les stocker en indiquant clairement le nom du champ, la modification effectuée (troncage, arrondi, mise à NULL, mise à zéro, …), les valeurs avant et après (sauf pour les champs de type texte) et les longueurs avant et après (dans le cas des champs texte) dans le cas d'un troncage ou d'un arrondi. Si des données ne sont pas insérées (cas de l'absence de tuple), vous devez aussi le logger.

Les images des posters des films seront stockées sous la forme de BLOBs dans la base de données.

Attention, certaines données pourraient légèrement différer de ce qui est mentionné. Par exemple certaines valeurs pour le champ `certification` sont incorrectes car on a une chaîne de caractères ne contenant pas une certification valide mais plutôt une chaîne vide ou null ou un tiret ou encore autre chose. Donc lorsque vous devez insérer ces informations dans votre BD, veillez à les vérifier et le cas échéant soit ne pas insérer le film soit remplacer la valeur incorrecte par NULL selon ce qui semble le plus approprié.

Dans le même esprit, vérifiez que les champs existent et s'ils existent avec une valeur spéciale indiquant l'absence de valeur comme une chaîne vide (par ex. pour la `certification`) ou 0 (par ex. pour le `runtime`) alors il est assez logique des les représenter par le NULL du SQL dans CB. De même si elles n'existent pas bien entendu. De cette manière votre code d'importation sera robuste.

En ce qui concerne l'importation des données textuelles comme le titre, la tagline, etc. on vous demande de supprimer le ou les espaces blancs (\t \n \r et l'espace) en début et en fin de chaîne.

La qualité de votre code d'importation et donc des données importées présentes dans CB est un point important de l'évaluation de cette fonctionnalité.

L'instruction SQL `MERGE` peut s'avérer intéressante dans ce genre de cas de figure. Veillez à utiliser les transactions de manière correcte et à loguer vos erreurs. Il est également imaginable de remplir à l'avance certaines tables dont le contenu peut être considéré comme statique à l'aide par exemple d'un autre programme (ou package) qui n'est invoqué qu'une seule fois avant les importations répétées des films. On peut aussi imaginer exécuter manuellement des scripts SQL qui aurait été générés par le programme en question.

Bien évidemment comme vous l'avez fait lors de l'implémentation des fonctionnalités précédentes, vous gérez les exceptions pouvant survenir lors des INSERTs en évitant d'écrire des tests explicites étant donné que les contraintes spécifiées dans le schéma (par `CREATE TABLE` ou déclencheurs) gèrent déjà cela. Vous pourriez néanmoins utiliser du code générique se basant sur le dictionnaire de données si vous souhaitez malgré tout réaliser des vérifications dans le code PL/SQL ou tout du moins avoir généré ce code au préalable. Nous laissons libre cours à votre imagination.

Techniques attendues : SQL, PL/SQL, procédures stockées, transactions, log.

7.6 EvalFilm - Recherche et évaluation d'un film

Ecrivez un programme dans le langage de votre choix pour rechercher un film donné de manière à pouvoir ensuite lui donner une cote accompagnée éventuellement d'un avis. Etant donné les critères utilisés pour la recherche, il est possible que celle-ci renvoie plusieurs films parmi lesquels il faudra en choisir un.

On désire disposer des critères de recherche suivants :

- Soit un identifiant précis correspondant ou non à un film.
- Soit un ou plusieurs acteur(s), le titre, un ou plusieurs réalisateur(s), les films sortis avant, après ou pendant une année donnée. Tous ces critères peuvent être combinés.

La combinaison des critères est telle que chaque critère ajouté aux précédents filtre l'ensemble des résultats précédemment obtenus en réduisant ou laissant inchangé le nombre d'éléments (critère de type "et logique"). Cela signifie qu'ajouter un deuxième acteur dans la recherche ne renvoie pas les films dans lequels l'acteur 1 et/ou l'acteur 2 sont présents. On dispose des films dans lesquels les deux acteurs sont présents.

Vous n'utiliserez pas de SQL dynamique pour la recherche, c'est-à-dire que vous ne concaténerez pas des chaînes de caractères pour construire la requête. On vous demande d'utiliser une combinaison de conditions avec `IS NULL`, `IS NOT NULL`, `AND`, `OR`, `NOT`, etc.

Utilisez des requêtes récursives (`WITH` récursif ou `CONNECT BY`) pour faire les recherches sur les acteurs et réalisateurs.

On vous demande d'utiliser les expressions régulières (regex, regexp) à l'aide des fonctions SQL prévues à cet effet comme par exemple `REGEXP_LIKE` pour effectuer les recherches sur les chaînes de caractères. On suppose que les recherches se font sur des sous-chaînes et de manière insensible à la casse (case insensitive).

On doit pouvoir visualiser les détails du film lorsqu'on sélectionne ce film dans la liste des films correspondant aux critères de recherche fournis, c'est-à-dire les informations précédentes (qui servent de critères de recherche) mais également la moyenne des votes TMDb, le nombre de votes TMDb, la moyenne des votes des utilisateurs de RQS, le nombre de votes des utilisateurs de RQS, l'image du film, le runtime et le statut.

On a également la possibilité de donner une cote au film sélectionné et éventuellement de lui adjoindre un avis textuel. Si une cote ou un avis a déjà été donné par cet utilisateur pour le film en question, on met à jour la cote et l'avis, c'est-à-dire qu'on remplace ce qui a été précédemment indiqué.

Au niveau de votre interface, on doit pouvoir indiquer une cote entre 0 et 10 (bornes incluses).

Votre code applicatif doit invoquer des procédures stockées PL/SQL. Veillez à utiliser les transactions de manière correcte et à loguer vos erreurs.

Techniques attendues : pas de SQL dynamique, requêtes récursives, expressions régulières, SQL MERGE, PL/SQL, procédures stockées, transactions, log.

7.7 CrashCB - Simulation d'un plantage de CB

On peut simuler la perte de connectivité avec CB, comme s'il y avait un problème réseau ou que l'instance venait de crasher, à l'aide de trois étapes dans l'ordre :

1. On localise la session de l'utilisateur à l'aide de l'instruction SQL ci-dessous exécutée en tant que SYS. On parle de l'utilisateur CB qui a accès à son schéma nommé CB et qui correspond à la base de données de même nom dans le cadre de l'énoncé.

   ```
   SELECT s.sid,
          s.serial#,
          p.spid,
          s.username,
          s.program
   FROM   v$session s JOIN v$process p ON p.addr = s.paddr
   WHERE  s.type != 'BACKGROUND';
   ```

   L'exécution de l'instruction renverra une résultat qui ressemble au listing ci-dessous dans lequel on peut voir deux colonnes qui nous intéressent : SID et SERIAL#.

   ```
   SID SERIAL# SPID USERNAME PROGRAM
   --- ------- ---- -------- -------------
   161     316 5596      SYS SQL Developer
   ```

2. On tue la session précédemment identifiée par deux nombres dans l'ordre : SID et le SERIAL#. Pour ce faire on utilise l'instruction SQL ALTER SYSTEM DISCONNECT qui a comme effet de déconnecter la session en cours en détruisant le processus serveur dédié. L'option IMMEDIATE indique que l'effet est immédiat sans attendre la fin des transactions en cours et donc qu'un ROLLBACK est initié par la base de données. Il faut donc veiller à disposer d'une gestion robuste des transactions et de ne pas laisser le "hasard" se charger des COMMITs et ROLLBACKs.

   ```
   ALTER SYSTEM DISCONNECT SESSION '161,316' IMMEDIATE;
   ```

3. On locke l'utilisateur de manière à ce qu'il ne puisse plus se connecter.

   ```
   ALTER USER cb ACCOUNT LOCK;
   ```

Le programme qui accède à CB par l'intermédiaire de procédures stockées doit pouvoir recevoir une information (exception, timeout, …) indiquant que la session et la connexion ont été perdues. Dès que cela est reçu, le programme va se connecter automatiquement à la base CBB et continuer ses opérations de manière transparente.

Attention, si on était au milieu de la réalisation d'une opération, celle-ci doit être annulée de manière propre et on doit le cas échéant prévenir l'utilisateur qu'il doit recommencer son opération. Cela vous demande de gérer vos transactions de manière robuste.

Dans le cadre de l'évaluation de cette fonctionnalité, nous re-testerons la fonctionnalité EvalFilm (section 7.6, page 18).

Techniques attendues : SQL, PL/SQL.

7.8 CreaCC - Script de création de CC

La base de données CC contenant le cinéma est une base relationnelle contenant des données de type XML structuré (`XMLType`) que ce soit de type colonne ou de type table.

On simplifie grandement les informations conservées à propos d'un film : on ne souhaite disposer que de son identifiant, son titre, sa certification, son ou ses réalisateurs, son image.

On disposera bien évidemment des copies présentes sur CC.

Vous devez étendre les modèles existants et rédiger le script de création des tables XMLType en plaçant un maximum de contraintes au niveau du schéma. On vous demande également les schémas XSD liés à chacune de ces tables. A défaut de script de création[12], il faut disposer des procédures permettant la création de ces tables.

Choisissez judicieusement vos types de données et la taille de ceux-ci de manière à ce que toutes les informations puissent être stockées telle quelles. Cfr. AlimCB (section 7.5, page 17).

Techniques attendues : SQL, XML, XDB, PL/SQL, procédures stockées.

7.9 AlimCC - Alimentation de CC

CC représente un complexe cinématographique dont la vocation est de proposer des programmations de films en utilisant des copies fournies par CB.

Toutes les semaines, une routine envoie un nombre aléatoire de copies de chaque film présent dans CB. Ce transfert se fera au moyen d'un job et toujours par le biais de documents XML.

Le nombre aléatoire se base sur une distribution uniforme allant de 0 (inclus) à $\lfloor \frac{n}{2} \rfloor$ où n est le nombre total de copies réellement disponibles sur CB et $\lfloor x \rfloor$ est l'entier le plus grand inférieur ou égal à x. On ne prend donc pas en compte uniquement le nouveau lot de copies d'un film mais l'ensemble des copies se trouvant sur CB.

Il est imposé qu'une copie ne puisse se trouver en même temps à la centrale CB et dans le complexe : il faut donc la supprimer de CB quand elle est transférée vers le complexe.

Si le nombre de copies à transférer est 0, les informations du film concerné ne seront pas transférées.

Dans le cas présent, les documents XML sont intégrés dans une table des films et des copies de type XMLType et automatiquement restructurés. Toutes les données transférées sont au format XML.

Techniques attendues : PL/SQL, réplication, DB link, XMLType, distribution uniforme, transferts de BLOB.

7.10 ProgFilms - Programmation des films

Chaque jour, des documents XML contenant des programmations sont déposés dans un DIRECTORY. Une routine en prend connaissance et effectue les programmations sollicitées. On désire disposer d'un autre document XML créé dans le même répertoire contenant le feedback de l'ajout de la programmation. Si une séance n'a pas pu être programmée, on indique la raison (slot déjà pris, copie indisponible, copie inexistante, ...). Le nom du fichier sera formé du nom original suffixé par `_feedback` avec l'extension `xml` dans les deux cas.

Si certaines d'entre-elles n'ont pas pu être réalisées, un rapport reprenant toutes les programmations conflictuelles est déposé dans le même répertoire en reprenant les motifs du conflit (plage non disponible, copie inexistante, ...).

Les programmations ne concernent que des dates dans le futur. On doit y retrouver au minimum un film, la salle et la plage horaire. La copie sera prise aléatoirement parmi celles existant pour le film s'il y en a

[12] Dans le cas d'une création automatique des tables par Oracle.

une de disponible. Le nombre de jour de programmation sera déterminé aléatoirement sur la base d'une distribution normale. On utilise pour cela une moyenne de 8 et un écart-type de 3. Bien entendu, il ne peut y avoir 0 jour de programmation. Une programmation existante doit également pouvoir être étendue un certain nombre de jour selon la même base de calcul du nombre de jours.

Pour faciliter le suivi de cette programmation, les différentes étapes de recherche et d'attribution seront enregistrées dans la table de log générale.

Techniques attendues : SQL, PL/SQL, DIRECTORY, UTL_FILE, XML.

8 Modalités d'évaluation de l'AA 2

Cette section indique les modalités d'évaluation de l'AA "Bases de données avancées".

8.1 Tableau synthétique

Tab. 3: Pondération des fonctionnalités

Fonctionnalité	Pond
CreaCBLight2	15
ReplicCB	15
AnalyseCI	20
CreaCB	10
AlimCB	20
EvalFilm	20
CrashCB	15
CreaCC	15
AlimCC	10
ProgFilms	20
Total	160