Rennequinepolis (RQS)

Concernant le cours de :

- Systèmes de Gestion de Bases de Données;

3e Bachelier en Informatique et Systèmes
Finalités "Systèmes" et "Réseaux et Télécommunications"

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

2012 – 2013

Contents

1 Préambule

Ce document présente les applications à réaliser dans les laboratoires du cours de SGBD. Il comporte le cahier des charges des applications attendues dans le cadre de ce cours.

Vous y trouverez le contexte général du système informatique qui devra être réalisé. On y détaille ensuite les différentes applications qui le constituent, en les associant aux cours auxquels elles se rattachent. Ce document peut donc être considéré comme le "contrat de travail" du ou des cours concerné(s). Il permettra de mieux appréhender l'organisation du travail de l'année et sa planification au travers des différentes semaines.

Les modalités d'évaluation se trouvent à la fin de cet énoncé reprenant les explications quant à la construction de la cote finale. Il convient d'en prendre connaissance, de le signer et nous le transmettre; il vous engage pour toute la durée de l'année scolaire.

1.1 Forum de l'Ecole Virtuelle

Les questions d'intérêt général ou concernant l'analyse du projet ainsi que les informations relevant des différentes techniques mises en oeuvre seront débattues ou présentées par l'intermédiaire du forum "Système de gestion de bases de données" des 3èmes de l'Ecole Virtuelle. *Ce forum sera le moyen privilégié de partager des informations entre enseignants et étudiants.*

Chacun aura donc le souci de le consulter régulièrement et de soumettre ses questions, remarques ou conseils via cet outil.

1.2 Page Web du cours

Toutes les ressources (énoncé, code source, fichiers textes, images, …) sont disponibles sur la page Web du cours à `https://cours.khi.be/sgbd3/`. 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.

Le login et le mot de passe à utiliser sont respectivement "inpres" et "resinp".

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.

Une première analyse a permis de distinguer un premier lot d'information : les intervenants, les sites et les objets manipulés. Sur base de cela un circuit fonctionnel et informationnel a pu être établi.

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 des fonctionnalités demandée à la section 9.2, page 38, pour savoir ce que vous devez exactement réaliser et donc modéliser.

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 à l'entreprise.
- **La centrale belge - CB** - Il s'agit donc ici de notre entreprise RQS. Située à Bruxelles, l'entreprise y dispose d'un entrepôt où elle peut stocker les médias physiques avant qu'ils ne soient distribués dans les cinémas, de même que de larges serveurs pouvant accueillir les données numériques et a fortiori les copies numériques des films.
- **Les cinémas - CCx** - Les différents complexes cinématographiques que gère RQS. Dans le cadre de cet énoncé il n'y a .
- **La banque de données des personnes - BP** - Site reprenant les archives relatives à l'ensemble des acteurs et réalisateurs mentionnés à l'affiche des films proposés par la centrale internationale.

2.2 Intervenants

Liste des intervenants provisoirement relevés:

- Les **opérateurs de RQS** - Divers gestionnaires ont la charge de la mise à jour du catalogue de films, la gestion des copies numériques ou des supports physiques, ….
- Les **gestionnaires des cinémas** - Ils sont responsables des opérations courantes des complexes, de la programmation des salles et des films, de l'alimentation en nouveaux films, la gestion des copies locales, ….
- **Les fournisseurs** - soit la centrale internationale, soit la banque de données.
- Les **clients acheteurs** - Les clients ont la possibilité de réserver leurs tickets de cinéma en ligne, selon les différentes programmations proposées.
- Les **directeurs de RQS** consultent régulièrement les données collectées auprès des différents sites de l'entreprise afin d'adapter la stratégie de celle-ci pour obtenir de meilleurs profits.

2.3 Objets manipulés

Les principaux objets manipulés dans notre système d'information sont de plusieurs types. On trouve :

- Des films et des copies de films;
- Des signalétiques de membres du personnel, d'acteurs;
- Des commandes de tickets;
- Des commandes de copies de films;
- Des commandes et des livraisons de fournisseurs;
- Des informations comptables.

2.4 Les films et les copies des films

La signalétique d'un film comporte toutes les informations relatives à ce film. Par exemple, son titre, sa durée, son (ses) genre(s), le public concerné, les acteurs principaux, une ou plusieurs affiches ou images clés, une langue parlée, une langue de sous-titres, ….. Les acteurs n'y sont cependant présents que par leur identifiant, leur nom et leur prénom. Cette signalétique est à distinguer de la copie physique ou numérique de celui-ci qui est l'élément réellement transmis aux complexes cinématographiques. Ainsi donc, les films et les copies devront être identifiés de manière fiable et robuste. A tout moment, il doit être possible de déterminer où se trouve chaque copie existante d'un film. Les informations collectées auprès de la CI devront donc être suffisamment complètes pour pouvoir distinguer les copies et le type de support sur lequel elles sont faites, mais également informer correctement les clients qui consulteront les fiches des films en vue de réserver des places dans les complexes.

Chaque film est un document JSON stocké dans une BD CouchDB (cfr. section 4, page 13), c'est-à-dire dans notre cas un objet (*object*) JSON. Tous les documents JSON ont la même structure dans ce cas-ci même si ce n'est pas imposé par CouchDB. On retrouve 26 propriétés par film expliquées au tableau 1, chacune possédant une valeur qui est un des types reconnus par JSON: *object*, *array*, *string*, *number*, `true`, `false` ou `null`. Les noms en italique représentent une classe de valeurs possibles comme les nombres (*number*) ou les chaînes de caractères (*string*) et les noms en police "de type code" représentent des valeurs litérales. Je vous invite à consulter le site Web[1] relatif à JSON ou le RFC 4627[2].

Notez que JSON n'est qu'une notation permettant de représenter textuellement, de manière sérialisée, des données structurées.

Attention, certaines données pourraient légèrement différer de ce qui est mentionné. Par exemple certains identifiants IMDb sont incorrects car on a une chaîne de caractères ne contenant pas l'identifiant mais plutôt soit l'URL de la page Web de IMDb, soit le titre du film, soit autre chose. Donc si 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`. On vous demande donc de regarder les données de CouchDB de manière à établir une politique d'importation de celles-ci dans Oracle. Il est assez facile de découvrir les différents types de cas qui poseront souci lors de l'importation à l'aide des vues de type unique de CouchDB qui sont expliquées à la section 4.5, page 18.

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

[1] `http://www.json.org/`
[2] `http://www.ietf.org/rfc/rfc4627.txt`

Table 1: Propriétés présentes dans un film

Nom	Explications
`adult`	Indique si le film est réservé aux adultes ou non. Valeurs: `true` ou `false`.
`alternative_name`	Autre nom du film de type *string* s'il existe ou `null`.
`backdrops`	Ce sont des images liées au film mais qui ne sont pas le poster officiel du film. C'est un tableau d'objets représentant chacun une image. Le tableau peut être vide (`[]`). S'il n'est pas vide, on retrouve des objets qui possèdent les propriétés suivantes: `type` qui est "backdrop", `size` qui est un mot indiquant la taille comme `thumb`, `poster`, `w1280`, `original`, …, `height` qui est la hauteur en pixels, `width` qui est la largeur en pixels, `url` qui permet d'obtenir l'image et `id` qui identifie l'image en sachant que des images qui ne diffèrent que par la taille ont le même `id`.
`budget`	Le coût du film en dollars sous forme de *number* ou `null` ou `0` s'il est inconnu.
`cast`	Un tableau d'objets éventuellement vide représentant les personnes ayant participé au film dont les réalisateurs et les acteurs. Chaque objet contient les propriétés suivantes: `name`, `job` ("Director", "Actor", "Producer", …), `department`, `character` (le nom de leur personnage) ou une chaîne vide, `id` qui est l'identifiant TMDb, `order`, `cast_id` et `profile` qui contient une URL vers une image de la personne. Nous sommes exclusivement intéressés par `name`, `job` et `id`.
`certification`	Le rating qui indique le public cible du film. Certaines abbréviations sont tirées de l'association MPAA[3]. De type *string*.
`countries`	Un tableau des pays ayant participé au film. Pour chaque pays on a un objet avec les propriétés suivantes: `code`, `name` et `url`.
`genres`	Les genres dont fait partie le film. Un tableau d'objets éventuellement vide dont chacun possède les propriétés suivantes: `type` qui est "genre", `name` qui est le nom du genre comme "Action", "Comedy" ou "Trhiller", `id` qui est l'identifiant du genre sous forme de *number* et une `url`.
`homepage`	Page Web du film sous forme de *string*. Si la valeur est inconnue, alors nous avons une chaîne vide ou `null`.
`imdb_id`	L'identifiant IMDb sous forme de *string*. Si la valeur est inconnue, alors nous avons `null`.
`keywords`	Un tableau éventuellement vide de mots-clés associés associés au film. Chaque mot-clé est de type *string*.

Suite sur la page suivante

[3] `http://www.mpaa.org/ratings/what-each-rating-means`

Table 1 – continuation de la page précédente

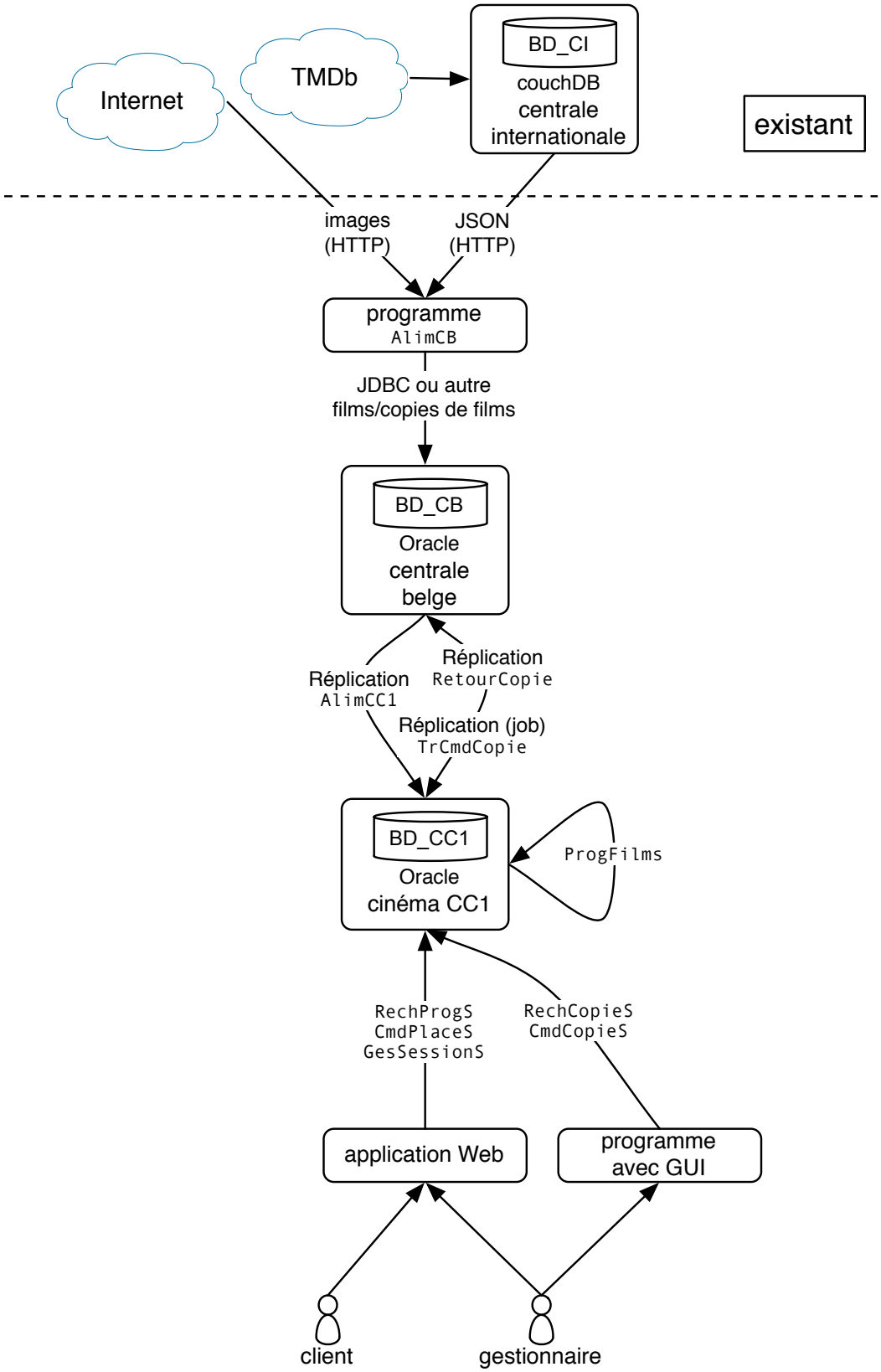
Nom	Explications
`languages_spoken`	Un tableau éventuellement vide d'objets représentant les langues parlées dans le film. Chaque objet possède les propriétés suivantes: `code` pour identifier la langue (comme "de"), `name` qui est le nom de la langue (comme "German") et `native_name` qui est le nom original de la langue (comme "Deutsch").
`name`	Le titre du film de type *string*. Pour rappel, les données sont encodées en UTF-8.
`original_name`	Le titre original du film de type *string*. Pour rappel, les données sont encodées en UTF-8. Typiquement il s'agit du titre dans la langue d'origine du film.
`overview`	Un résumé du film sous forme de *string* en UTF-8. Ce résumé peut être écrit dans une langue autre que l'anglais. Si le résumé n'existe pas, on a la valeur `null`, une série d'un ou plusieurs ".", une chaîne vide, ou juste des espaces blancs.
`posters`	Un tableau éventuellement vide des différents posters du film. On retrouve la même structure que pour la propriété `backdrops` décrite précédemment.
`rating`	La cote qu'a obtenu le film suite aux votes des membres TMDb sous forme d'un *number*.
`released`	La date de sortie du film de type *string* et au format `YYYY-MM-DD`.
`revenue`	Le revenu qu'a généré le film en dollars sous forme de *number* ou `null` ou `0` s'il est inconnu.
`runtime`	La durée du film en minutes sous forme de *number* ou `null` ou `0` si elle est inconnue.
`status`	L'état de production du film. Cela peut être: "Canceled", "In Production", "Planned", "Post Production", "Released", "Rumored". S'il est inconnu, on retrouve `null` ou la chaîne vide.
`studios`	Un tableau éventuellement vide contenant des objets représentant les studios qui ont participé au film. Chaque objet contient les propriétés suivantes: `name` de type *string* est le nom du studio, `id` de type *number* est l'identifiant TMDb du studio et une `url`.
`tagline`	Le texte qui accompagne le film et qui est souvent écrit sur la pochette du film sous forme de *string*. S'il est inconnu, on a la valeur `null` ou une chaîne vide.
`trailer`	L'URL sous forme de *string* du trailer du film, typiquement sur YouTube ou `null` si on ne possède pas l'information.

Suite sur la page suivante

Table 1 – continuation de la page précédente

Nom	Explications
`translated`	Indique si le film a été traduit dans une autre langue. `true` ou `false`.
`votes`	Le nombre de votes sous forme de *number* qui a servi à établir le `rating`. S'il est inconnu, la valeur `0` est indiquée.

Figure 1: Flux globaux



3 Bases de données

3.1 Infrastructure au sein de l'école

Le récapitulatif des bases de données est indiqué au tableau 2. Le mot `user` doit être remplacé par le nom de votre utilisateur. Ceux-ci sont indiqués sur l'Ecole Virtuelle dans le forum "Système de gestion de bases de données". Les bases de données Oracle au sein de l'école ont toutes le jeu de caractères UTF-8 par défaut, c'est-à-dire `AL32UTF8`.

Le domaine au sein de l'école est `tech.hepl.local` et les serveurs DNS sont `10.59.26.111`, `10.59.26.112` et `10.7.0.100`.

Les FQDNs[4] et les IPs des machines `diderot`, `indochine`, `eve` et `nemo` sont :

- `eve.tech.hepl.local`: 10.59.26.132
- `nemo.tech.hepl.local`: 10.59.26.135
- `doris.tech.hepl.local`: 10.59.26.138

Table 2: Bases de données utilisées

BD énoncé	Type	Machine	Schéma / BD	BD Oracle	
				SID	Nom global
`BD_CI`	CouchDB	eve	`movies_20122013`	N/A	
`BD_CB`	Oracle	nemo	`user_CB`	`nemo1`	`nemo1.sgbd`
`BD_CC1`	Oracle	doris	`user_CC1`	`doris1`	`doris1.sgbd`

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.

Nous vous invitons à reprendre la machine virtuelle intitulée "Database App Development VM" sur le site d'Oracle à l'URL `http://www.oracle.com/technetwork/community/developer-vm/index.html`. 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 télécharger une image nommée `Oracle Developer Day.ova` de 4.3GB après avoir créé un compte gratuit chez Oracle.

Les instructions pour démarrer la VM sont indiquées sur la page de téléchargement d'Oracle. Parmi celles-ci vous remarquerez que 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
...
```

[4] Fully Qualified Domain Names.

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

4 CouchDB

Les données de CI sont stockées dans une base de données de type Apache CouchDB[5]. Elle est nommée `movies_20122013`. C'est une base non relationnelle orientée documents qui fait partie de la catégorie des BDs dites NoSQL[6] comme MongoDB.

Nous avons rapatrié les informations de tous les films de TMDb à l'aide de requêtes HTPP GET en utilisant l'API 2.1 de type REST. De cet ensemble conséquent de 85195 films, nous avons sélectionné un sous-ensemble de 10000 films en ne prenant que ceux dont la cote est supérieure ou égale à six avec au moins cinq votes et ensuite en complétant avec les plus récents jusque mai 2012 inclus de manière à disposer du nombre voulu de films.

Il s'agit pour vous de récupérer notre BD `movies_20122013` de manière à en avoir une copie locale et ensuite grâce à un programme d'aller stocker les informations dans les BDs Oracle correspondantes selon ce qui est indiqué dans l'énoncé.

Nous vous invitons à lire la courte introduction[7] au système CouchDB qui se trouve sur le Wiki officiel de CouchDB. Les données manipulées sont au format JSON[8]. Il s'agit d'un format textuel structuré plus simple que XML, à l'instar de YAML. Toutes les données textuelles sont stockées en UTF-8.

4.1 Installation

CouchDB est disponible sous Linux, Windows et OS X. La version 1.2.0 est supportée dans le cadre du cours. Sous Windows, je vous invite à ne pas installer CouchDB comme un service mais à le lancer manuellement pour plus de facilité de débugagge en cas de souci. Actuellement, vous disposez des possibilités suivantes :

- Sous Windows, installez la version basée sur Erlang R15B que vous trouverez dans la section des téléchargements du site de Apache CouchDB.
- Sous Linux vous avez deux options: compiler à partir des sources (cela implique d'avoir installé les dépendances de CouchDB dont Erlang) ou trouver la version 1.2.0 comme un package de votre distribution. Par exemple pour Ubuntu 12.04 vous pouvez suivre les instructions du didacticiel "Installing CouchDB 1.2 in Ubuntu 12.04"[9].
- Sous OS X, installez le package `couchdb` avec Homebrew[10] ou avec MacPorts[11]. Notez que dans les deux cas de figure vous aurez besoin d'installer Xcode grâce à l'App Store.

4.2 Post-installation

1. Arrêtez le serveur CouchDB.
2. Téléchargez le fichier rar contenant la BD CouchDB et la génération des vues: `movies_20122013_couchdb.rar`[12]. La BD contient 10000 documents (ou films) et fait 112 MB. A cela on ajoute les vues générées pour 478 MB.
3. Décompressez l'archive dans le bon répertoire ("extract here"). Le répertoire diffère selon l'OS :
   - Windows : `C:\Program Files\Apache Software Foundation\CouchDB\var\lib\couchdb`
   - Linux : `/var/lib/couchdb` ou `/usr/local/var/lib/couchdb`

[5] `http://couchdb.apache.org/`
[6] `http://en.wikipedia.org/wiki/NoSQL`
[7] `http://wiki.apache.org/couchdb/Introduction`
[8] `http://www.json.org/`
[9] `http://onabai.wordpress.com/2012/05/10/installing-couchdb-1-2-in-ubuntu-12-04/`
[10] `http://mxcl.github.com/homebrew/`
[11] `http://www.macports.org/`
[12] `https://cours.khi.be/sgbd3/labo/movies_20122013_couchdb.rar`

- OS X : `/usr/local/var/lib/couchdb` (avec Homebrew)

4. Redémarrez le serveur CouchDB.

Vous aurez après décompression un fichier `movies_20122013.couch` dans le répertoire principal et un répertoire `.movies_20122013_design` contenant un fichier avec l'extension `view`. On a typiquement un fichier par "design document" lui-même contenant les données des vues mais d'anciens fichiers contenant la génération des anciennes vues peuvent subsister, d'où l'opération de nettoyage indiquée ci-dessous à la section 4.3.2.

Pour éviter d'avoir une BD trop conséquente, les images vous sont fournies dans une archive à part de 217 MB nommée `movies_20122013_images.rar`[13].

4.3 Administration et interrogation de la BD

4.3.1 Futon et curl

Deux manières d'interagir avec le serveur CouchDB et ses BDs pour les administrer sont communément utilisées :

- Soit avec l'interface Web Futon accessible à `http://<host>:5984/_utils/` où `<host>` est typiquement `localhost`. Nous avons donc l'adresse `http://localhost:5984/_utils/`.
- Soit avec un programme en ligne de commande qui permet d'interagir avec un serveur HTTP comme `curl` ou `wget` en utilisant tous les verbes HTTP et qui sont donc particulièrement adaptés pour "parler" REST avec le serveur. En plus des méthodes HTTP habituelles qui sont GET, POST et HEAD, on peut aussi utiliser PUT, DELETE et d'autres méthodes. Il est possible d'utiliser un navigateur Web mais cela ne permet que de faire des GETs en indiquant l'URL dans la barre d'adresse. Concrètement, dans le cadre du laboratoire, vous n'aurez besoin que de GET puisqu'il s'agit de lire des données, pas de les créer, les mettre à jour ou les effacer.

Si vous désirez que le serveur CouchDB écoute sur une interface publique plutôt que `localhost`, vous pouvez avec Futon modifier le paramètre `bind_address` dans la partie "Configuration"[14] et le changer en `0.0.0.0`.

4.3.2 Nettoyage de la BD

Pour éviter que la BD ne grandisse de trop, on a limité le nombre de révisions d'un document que la BD garde sur disque à une seule révision en exécutant :

```
curl -X PUT -d "1" http://localhost:5984/movies_20122013/_revs_limit
```

Vous pouvez vous assurer que ce paramètre a la bonne valeur en exécutant une requête GET avant de vouloir le modifier ou après l'avoir modifié. Par exemple, dans mon cas, j'exécute :

```
$ curl -X GET http://localhost:5984/movies_20122013/_revs_limit
1
```

On s'arrange aussi pour exécuter avec régularité les opérations ci-dessous que l'on peut exécuter aussi avec Futon. Avec Futon, on peut voir l'état d'avancement des opérations en cours avec "Status".

- Compactage de la BD.

  ```
  curl -H "Content-Type: application/json" -X POST http://localhost:5984/
  movies_20122013/_compact
  ```

- Nettoyage des anciennes sorties des vues (les arbres générés) et qui sont inutilisées.

[13] `https://cours.khi.be/sgbd3/labo/movies_20122013_images.rar`
[14] `http://127.0.0.1:5984/_utils/config.html`

```
curl -H "Content-Type: application/json" -X POST http://localhost:5984/
movies_20122013/_view_cleanup
```

- Compactage des vues du design document "main".

```
curl -H "Content-Type: application/json" -X POST http://localhost:5984/
movies_20122013/_compact/main
```

4.4 L'interrogation des données et les vues

4.4.1 Introduction

L'interrogation (querying) d'une BD CouchDB se fait principalement à l'aide de vues[15]. On peut les voir comme l'équivalent des SELECT en SQL mais avec une approche différente. Un certain nombre de vues ont déjà été crées dans la BD `movies_20122013` mais rien ne vous empêche d'ajouter les vôtres. Le code alimentant les BDs Oracle utilisera ces vues.

Les vues se trouvent dans un type de documents appelé "design document" qui contient du code Javascript. Une vue est le résultat de l'exécution de code JS provenant d'un design document. Un design document peut contenir plusieurs vues et chaque vue est crée à l'aide de code JS comme les vues du monde SQL qui sont crées grâce à des instructions SQL.

Une BD CouchDB est consitutée d'un ensemble de documents, chacun possédant un identifiant qui peut être qualifié de clé primaire si on utilise la jargon des BDs relationnelles. Lorsque le code qui permet de générer une vue est exécuté, un arbre est construit et gardé en cache sur disque de manière à pouvoir afficher rapidemment la vue et faire des recherches efficaces. La construction de cet arbre *implique de parcourir l'entièreté des documents* de la BD. Cette opération lourde n'intervient pas souvent, surtout dans une BD où la majorité des opérations sont faites en lecture. Il est important de bien comprendre que le code JS génère une liste de paires (clé, valeur) (sous forme d'arbre), lors de la création de la vue ou de sa modification.

Les deux opérations qui sont utilisés pour construire des vues sont "map" et "reduce". Le nom de ces opérations provient des langages de programmation dit fonctionnels (Haskell, Common Lisp, Scheme, …) et sont à l'origine du succès de Google grâce à son architecture distribuée MapReduce[16]. Dans notre cas, nous allons utiliser uniquement l'opération "map". Elle consiste à appliquer une opération donnée à une collection d'éléments de manière à générer une nouvelle collection d'éléments de même taille. Comme par exemple l'opération "multiplier par 2" appliquée à la collection `{1,2,3,4}` qui donne la nouvelle collection `{2,4,6,8}`.

Dans le cas de CouchDB, l'opération map est appliquée aux document de la BDs de manière à générer une collection de paires (clé, valeur). Notez qu'il peut y avoir moins de paires que de documents, donc la taille de la collection générée n'est pas nécessairement la même que la taille de la collection initiale.

4.4.2 Comment créer des vues ?

Tout d'abord, devez-vous créer des vues ? Pas nécessairemment. Un grand nombre de vues ont déjà été crées de manière à vous faciliter la vie. Cependant rien ne vous empêche d'en ajouter des nouvelles pour répondre à vos besoins spécifiques. Notez que la génération du résultats des vues prend beaucoup de temps, jusqu'à plusieurs heures dans le pire des cas. Nous avons 48 vues avec 10 000 documents à parcourir par vue lors de la génération, cela veut dire 480 000 exécutions de petites fonctions JS.

Le plus simple est de réaliser les étapes suivantes dans l'ordre :

[15] `http://wiki.apache.org/couchdb/HTTP_view_API`
[16] `http://en.wikipedia.org/wiki/MapReduce`

1. Créer un design document contenant nos vues dans un fichier texte.
2. Placer ce design document dans la BD.
3. Plus tard, mettre à jour le design document dans le fichier texte.
4. Plus tard, mettre à jour le design document dans la BD en le remplaçant par celui du fichier texte.
5. On refait les étapes 3 et 4 à chaque modification.

Pour nous faciliter la vie, il est intéressant de disposer d'un programme qui va se charger de créer ou mettre à jour notre design document dans la BD en fonction du contenu de notre fichier texte. Ce n'est pas compliqué car notre design document est le contenu de notre fichier texte mais il faut veiller à retirer l'indentation superflue du code JS pour que le code soit un document JSON valide et soit bien interprété par CouchDB. Pour ce faire, j'ai utilisé un script Ruby 1.9 appelé `create_or_replace_design.rb`[17]. Celui-ci utilise les librairies (ou gems) `couchrest` pour accéder au serveur CouchDB, `json` pour créer et parser le format JSON et `optparse` pour lire les arguments en ligne de commande.

Dans l'exemple ci-dessous, on met à jour le design document appelé `main` sur la BD `movies_20122013` à partir du fichier texte `design_main_movies_20122013_generated.json` du répertoire courant. Le programme va rechercher le numéro de révision courant, met à jour le document à l'aide de ce numéro et reçoit en retour le nouveau numéro.

```
$ ./create_or_replace_design.rb --database movies_20122013 --host localhost ↩
design_main_movies_20122013_generated.json
Current Rev: 13-b60246c4b78b05f49f433fe3baa94d82
{"ok"=>true, "id"=>"_design/main", "rev"=>"14-6984d205940b39dc0a770a15e6b19312"}
```

Chaque design document avec ses vues va avoir le format canonique ci-dessous (pas de reduce ici). On remarque le nom du design document `nom_du_design_document` qui contient plusieurs vues dans l'objet nommé `views` dont la vue `nom_de_la_vue` qui est elle-même un objet avec une propriété `map` et une valeur de type string qui est la définition d'une fonction JS. Cette fonction JS prend en argument le document en cours `doc` pendant de l'examination de tous les documents de la BD lors de l'opération de mapping (génération de la vue). Si l'on désire que de l'information relative à ce document soit présente dans la vue, on utilise la fonction `emit` en lui passant la clé et la valeur de manière à constituer la paire (clé, valeur) dont nous avons parlé précédemment. Un exemple va clarifier tout cela.

```
{
    "_id": "_design/nom_du_design_document",
    "views": {
        "nom_de_la_vue": {
            "map": "function(doc) { ... }"
        },
        ...
    }
}
```

Nous allons utiliser le document `design_test.json`[18]. Celui-ci contient tous les exemples. Nous le plaçons dans la BD comme design document appelé `test`. Notez qu'il est déjà présent dans votre BD.

```
$ ./create_or_replace_design.rb --database movies_20122013 --host localhost --design ↩
test design_test.json
{"ok"=>true, "id"=>"_design/test", "rev"=>"1-2154b756066b24cd7c3b023417ca2a9c"}
```

[17] `https://cours.khi.be/sgbd3/labo/couch/create_or_replace_design.rb`. Notez que le chemin pour localiser l'exécutable `ruby` sera différent du mien si vous n'utilisez pas Homebrew sous OS X. Il faut donc l'actualiser dans le programme.

[18] `https://cours.khi.be/sgbd3/labo/couch/design_test.json`

Exemple 1 : tous les films (clé `_id`) Si nous voulons afficher tous les films sans afficher les autres documents comme les design documents alors il faut faire une vue qui n'affiche que les documents possédant une propriété comme "id" (ou "name"). Comme si en Java vous ne désiriez afficher que les objets possédant une variable membre nommée "id". Notez que "_id" est la propriété utilisée par CouchDB comme PK pour identifier un document et est de type string. De ce fait, chaque document quel qu'il soit la possède. La propriété "id" est une propriété qui n'existe que dans un document qui représente un film.

Dans ce cas, la fonction servant au mapping est indiquée ci-dessous. La clé est l'identifiant du document et la valeur est le document lui-même. Le test portant sur l'existence de la propriété `doc.id` permet de différencier les films des design documents.

```
"all": {
  "map": "function(doc) {
    if (doc.id) emit(doc._id, doc);
  }"
}
```

Ensuite on peut interroger la vue avec un navigateur ou avec `curl` en utilisant l'URL `http://localhost:5984/movies_20122013/_design/test/_view/all?limit=10`. La réponse elle-même est au format JSON. Notez qu'on limite le nombre de résultats pour rendre l'affichage plus rapide.

Comme nous avons choisi `_id` comme clé et que l'identifiant CouchDB choisi correspond à l'identifiant TMDb, nous pouvons rechercher un film dont nous connaissons l'identifiant TMDb comme Alien[19] dont le numéro est 348. Il suffit d'accéder à l'URL `http://localhost:5984/movies_20122013/_design/test/_view/all?key=%22348%22` où les `%22` représentent les guillemets URL-encoded. Les guillemets sont nécessaires car `_id` est de type string.

Les paramètres qu'on peut utiliser dans l'URL comme `key` ou `limit` sont indiqués dans la section "Querying Options"[20] de "HTTP View API"[21].

Exemple 2 : tous les films (clé `id`) Si nous utilisons la vue `all_id` dont la clé est `id` et plus `_id` alors nous disposons des mêmes fonctionnalités que précédemment si ce n'est que le type de donnée de `id` est number. Par conséquent l'URL à utiliser pour reprendre les informations du film Alien est `http://localhost:5984/movies_20122013/_design/test/_view/all?key=348` sans guillemets.

Exemple 3 : tous les films (clé `name`) Cette fois nous allons avoir la liste des paires (`name`, `doc`) où `name` est le nom du film et `doc` est le document JSON complet représentant le film. L'avantage d'avoir `name` comme clé est que nous pouvons faire une recherche sur un nom de film si on connaît le nom complet ou en partant du nom indiqué en utilisant l'ordre alphabatique.

```
"name_doc": {
  "map": "function(doc) {
    if (doc.id) emit(doc.name, doc);
  }"
}
```

On peut reprendre les films en commançant par les films dont le nom commence par la lettre "k" avec `http://localhost:5984/movies_20122013/_design/test/_view/name_doc?startkey=%22k%22&limit=10`. Cela nous donne donc tous les films à partir de la lettre "k" jusque "z" inclus. De même on peut spécifier une `endkey` de manière à délimiter une étendue. Remarquez la différence entre `http://localhost:5984/movies_`

[19] `http://www.themoviedb.org/movie/348`
[20] `http://wiki.apache.org/couchdb/HTTP_view_API#Querying_Options`
[21] `http://wiki.apache.org/couchdb/HTTP_view_API`

`20122013/_design/test/_view/name_doc?startkey=%22ka%22&endkey=%22l%22` et `http://localhost:5984/movies_20122013/_design/test/_view/name_doc?startkey=%22ka%22&endkey=%22lz%22`. La première URL ne nous permet pas d'avoir les films commençant par la lettre "l" sauf si un film s'appelait "l" tandis que la seconde nous permet d'avoir tous les films commençant par "l" sauf les films commençant par "lz" et composé de plus de deux lettres.

Exemple 3: tous les noms de films (clé `name`) Cette fois nous retournons les paires (`name`, `null`). Le but est d'avoir la liste des titres des films mais pas les films complets. La requête est plus rapide à être exécutée et nous recevons les documents plus rapidemment. De plus la quantité de donnée à exploiter est nettement moindre. Il est intéressant d'exécuter les requêtes précédentes: `http://localhost:5984/movies_20122013/_design/test/_view/name?startkey=%22ka%22&endkey=%22l%22` et `http://localhost:5984/movies_20122013/_design/test/_view/name?startkey=%22ka%22&endkey=%22lz%22`. Maintenant on voit clairement la différence.

```
"name": {
  "map": "function(doc) {
    if (doc.id) emit(doc.name, null);
  }"
}
```

4.5 Les vues de la BD

On retrouve 48 vues dans la BD permettant de l'interroger pour déterminer un certain nombre de choses.[22]

Chaque vue produit une liste de paires (clé, valeur) comme indiqué précédemment. Donc lorsqu'on désire montrer le résultat renvoyé par une vue, il est nécessaire de préciser ce que sont la clé et la valeur associée. C'est ce que nous allons faire.

En règle générale, nous avons des vues dont la clé est la propriété JSON (d'un document de la BD) portant le même nom que la vue et dont la valeur est `doc.id`, c'est-à-dire l'identifiant TMDb de notre document. Par exemple la vue `certification` donne la liste de 10 000 paires (`doc.certification`, `doc.id`) dans laquelle pour chaque document on associe sa certification avec son identifiant. Les vues suivantes sont de ce type : `adult`, `alternative_name`, `backdrops`, `budget`, `cast`, `certification`, `countries`, `genres`, `homepage`, `imdb_id`, `keywords`, `languages_spoken`, `name`, `original_name`, `overview`, `posters`, `rating`, `released`, `revenue`, `runtime`, `status`, `studios`, `tagline`, `trailer`, `translated`, `votes`.

On vous demande de vous référer au tableau 1, page 7, pour connaître la signification de chaque propriété.

Les vues dont le nom se termine par `unique` permettent d'obtenir la liste des valeurs uniques pour une propriété donnée à l'instar de ce que fait un `SELECT DISTINCT` en SQL. Pour interroger une vue de type unique on ajoute la "query string" `group=true` dans l'URL. Par exemple, l'URL pour la vue `certification_unique` est la suivante :

`http://localhost:5984/movies_20122013/_design/main/_view/certification_unique?group=true`

Le résultat de la requête est :

```
{"rows":[
{"key":null,"value":3632},
{"key":"","value":2920},
{"key":" ","value":1},
{"key":"-","value":3},
{"key":"12","value":1},
```

[22] En réalité il y en a 49 mais la vue `popularity` n'est pas utilisée.

```
{"key":"G","value":288},
{"key":"MA","value":1},
{"key":"N/A","value":2},
{"key":"NC-17","value":236},
{"key":"None","value":1},
{"key":"Not Rated","value":7},
{"key":"Not Yet Rated","value":1},
{"key":"NR","value":132},
{"key":"PG","value":506},
{"key":"pg-13","value":1},
{"key":"PG-13","value":820},
{"key":"PG13","value":1},
{"key":"R","value":1367},
{"key":"TV-14","value":6},
{"key":"TV14","value":1},
{"key":"unrated","value":1},
{"key":"Unrated","value":6},
{"key":"UR","value":5},
{"key":"X","value":55},
{"key":"XXX","value":6}
]}
```


Nous disposons ainsi de la liste des valeurs prises par la propriété `certification` pour l'ensemble des 10 000 documents ainsi que pour chaque valeur de certification le nombre de documents ayant cette valeur. Cela nous permet également de remarquer que 3 632 documents n'ont pas de valeur pour cette propriété (`null`), 2 920 ont la chaîne vide, un document a un seul espace et trois documents ont un tiret. Cela vous permet de savoir ce que vous allez rencontrer et donc la manière dont votre traitement doit s'effectuer. On peut imaginer sans difficulté que les valeurs `null`, `""`, `" "` et `"-"` correspondent au `NULL` du SQL. Nous voyons aussi qu'il est nécessaire de traiter certaines valeurs comme `pg-13` ou `PG13` pour qu'elles correspondent à la valeur correcte `PG-13`. A vous de regarder les valeurs possibles des autres propriétés.

Les vues suivantes sont de ce type : `certification_unique`, `countries_code_unique`, `countries_name_unique`, `genres_id_unique`, `genres_name_unique`, `languages_spoken_code_unique`, `languages_spoken_name_unique`, `status_unique`, `studios_id_unique`, `studios_name_unique`.

La valeur de certaines propriétés est de type tableau d'objets. Dans ce cas, en plus de donner les différents tableaux, nous fournissons aussi des vues pour aller chercher les propriétés contenues dans ces objets. C'est par exemple le cas avec la vue `countries` qui renvoie des paires dont la clé est formée d'un tableau d'objets comme :


```
{"id":"41271","key":[{"code":"FR","name":"France","url":"http://www.themoviedb.org/country
/fr"},{"code":"BE","name":"Belgium","url":"http://www.themoviedb.org/country/be"}],"value
":41271}
```


Ce qui nous intéresse, c'est de pouvoir accéder aux propriétés `code` et `name` des différents pays, ce que nous pouvons faire avec les vues `countries_code` et `countries_name`. Il existe aussi une version "unique" pour chaque propriété : `countries_code_unique` et `countries_name_unique`.

Listing 1: Résultat de la vue `countries_code` pour le film 41271.


```
...
{"id":"41271","key":"BE","value":41271},
{"id":"41271","key":"FR","value":41271},
...
```

Listing 2: Résultat de la vue `countries_name` pour le film 41271.

```
...
{"id":"41271","key":"Belgium","value":41271},
{"id":"41271","key":"France","value":41271},
...
```

Les vues de ce type sont : `countries_code`, `countries_name`, `countries_code_unique`, `countries_name_unique`, `genres_id`, `genres_name`, `genres_id_unique`, `genres_name_unique`, `languages_spoken_code`, `languages_spoken_name`, `languages_spoken_code_unique`, `languages_spoken_name_unique`, `studios_id`, `studios_name`, `studios_id_unique`, `studios_name_unique`.

Enfin, certaines vues ne rentrent pas dans les catégories de vues expliquées ci-dessus. Elles sont expliquées dans le tableau 3.

Table 3: Vues présentes dans la BD des films

Nom	Explications
`id`	(`doc.id`, `null`)
`id_doc`	(`doc.id`, `doc`)
`movie_actors`	Si le film possède un "cast" et a au moins un acteur (job = "Actor") alors on renvoie un tableau d'objets contenant les acteurs du films uniquement : (`doc.id`, acteurs)
`movie_directors`	Si le film possède un "cast" et a au moins un réalisateur (job = "Director") alors on renvoie un tableau d'objets contenant les réalisateurs du films uniquement : (`doc.id`, réalisateurs)

4.6 CouchDB et Java

Un certain nombre de librairies sont nécessaires pour accéder à une base CouchDB dont Ektorp et Jackson. Ektorp compte sur d'autres librairies indiquées ci-dessous. Les JARs sont disponibles sur le site Web du cours dans une archive[23].

4.6.1 JARs

- Le processeur Jackson est utilisé pour lire et produire des données au format JSON. Le mapping avec les types Java est indiqué dans le tableau de la section Simple Data Binding Example du tutoriel. On demande d'utiliser la version 1.7.7.
- Vous allez utiliser la librairie ektorp pour accéder aux documents de la base de données. On vous demande d'utiliser la version 1.2.2.
- SLF4J pour le logging. Il faut prendre la version 1.7.0. Parmi les JARs décompressés, seuls `slf4j-api-(version).jar` et `slf4j-nop-(version).jar` nous intéressent.
- HttpComponents de Apache pour accéder aux ressources par HTTP. Nous avons besoin de la version 4.2.1 de HttpClient.

[23] `https://cours.khi.be/sgbd3/labo/ektorp-1.2.2-needed-jars.zip`

4.6.2 Exemples

Différents programmes Java vont vous servir d'exemples :

- `CouchTest1.java`[24] montre le rapatriement des informations de `rating` des films et l'utilisation lors de la recherche avec la fourchette $[7, 7.2[$ ainsi que l'utilisation d'un système de cache pour accélérer les recherches.
- `CouchTest2.java`[25] montre comment reprendre les informations d'un film sans passer par un mapping document JSON → CouchDbDocument ou un POJO. Cela simplifie assez fort le code.
- `ImageGetter.java`[26] montre comment reprendre une image par HTTP.

[24] `https://cours.khi.be/sgbd3/labo/CouchTest1.java`
[25] `https://cours.khi.be/sgbd3/labo/CouchTest2.java`
[26] `https://cours.khi.be/sgbd3/labo/ImageGetter.java`

5 Images

Les images des films et des personnes sont accessibles par HTTPS sur le site Web du cours et par HTTP sur `eve`. Vous pouvez aussi installer un serveur Web léger sur votre machine.

5.1 Accès en dehors de l'école

Pour un accès aux images se trouvant sur `cours.khi.be`, l'URL à utiliser est indiquée ci-dessous où `type` vaut `movies` ou `people` et `id` est le numéro identifiant le film. Le code Java de l'exemple `ImageGetter.java` permet d'accéder à une image par HTTPS en précisant une authentification basique HTTP.

```
https://cours.khi.be/sgbd3/labo/nobackup/<type>/<id>.jpg
```

Par exemple pour Alien : `https://cours.khi.be/sgbd3/labo/nobackup/movies/348.jpg`.

5.2 Accès au sein de l'école

Pour un accès aux images se trouvant sur `eve`, l'URL à utiliser est indiquée ci-dessous et la signification de `type` et de `id` est la même que pour un accès extérieur à l'école.

```
http://eve/images/<type>/<id>.jpg
```

Par exemple pour Alien : `http://eve/images/movies/348.jpg`.

5.3 Sur votre machine

Vous pouvez installer un serveur Web simple comme webfs[27] sous Linux ou OS X et un serveur comme mongoose[28] sous Windows.

[27] `http://linux.bytesex.org/misc/webfs.html`
[28] `http://code.google.com/p/mongoose/`

6 Rappels mathématiques

Ce qui suit n'a pas pour objectif de remplacer un cours de mathématiques mais simplement de rappeler quelques notions qui pourront être utiles dans la réalisation des applications attendues dans le cadre de ce laboratoire.

6.1 Notions de base

6.1.1 Expérience, éventualité, univers et évènement

Une *expérience* comme un lancer de pièce de monnaie possède un ensemble de résultats possibles appelés *éventualités* et notés ω. Cet ensemble est nommé *univers* et est noté Ω. Dans notre cas l'ensemble des résultats possibles est $\{\text{pile}, \text{face}\}$ ou encore $\{\text{P}, \text{F}\}$. On écrit donc $\Omega = \{\text{P}, \text{F}\}$. P et F sont chacun une éventualité ω de notre expérience. Un *évènement* est un sous-ensemble de l'univers, c'est-à-dire dans notre cas un des quatre ensembles suivants : $\emptyset$, $\{\text{P}\}$, $\{\text{F}\}$ ou $\{\text{P}, \text{F}\}$. Enfin, à un évènement nous associons une probabilité toujours comprise entre 0 et 1 inclus qui est celle que l'évènement survienne à l'issue de notre expérience.

Dans notre cas, on sait que :

- La probabilité que ni pile ni face ne sorte ou que pile et face sortent en même temps est nulle. C'est l'évènement impossible noté $\emptyset$. On écrit $P(\emptyset) = 0$.
- La probabilité que pile sorte est 1/2. $P(\{\text{P}\}) = 1/2$.
- La probabilité que face sorte est 1/2. $P(\{\text{F}\}) = 1/2$.
- La probabilité que pile ou face sorte est 1 car cet évènement est certain. $P(\{\text{P}, \text{F}\}) = P(\Omega) = 1$.

6.2 Variable

Une variable aléatoire X prend une valeur numérique bien précise pour chaque éventualité ω de notre univers Ω. C'est une fonction. Dans le cas de notre lancer de dé, on peut imaginer que X représente le nombre de pile obtenu à l'issue d'une expérience : 0 ou 1. On a donc $X(\text{P}) = 1/2$ et $X(\text{F}) = 1/2$.

La probabilité que le nombre de pile vaille 0 est $P(X = 0) = 1/2$. La probabilité que le nombre de pile vaille 1 est $P(X = 1) = 1/2$.

6.2.1 Variable discrète

Une variable discrète est une variable numérique qui prend uniquement un nombre limité de valeurs habituellement entières. Par exemple, la variable X peut seulement être égale à 1, 3, 5 ou 1000.

6.2.2 Variable continue

Une variable continue est une variable numérique qui peut prendre un nombre infini[29]. L'âge, la distance et la température par exemple sont considérées comme des variables continues. Par exemple une personne peut parcourir 3 642 531 km à pied ou de l'eau peut avoir une température de 89.26 °C.

[29] Strictement parlant, un infini indénombrable.

6.2.3 Moyenne

En statistique, la moyenne est la valeur unique que devraient avoir tous les individus d'une population (ou d'un échantillon) pour que leur total soit inchangé. C'est un critère de position. Dans la plupart des cas, le total formé par les individus d'une population est la somme de leurs valeurs. La moyenne est alors la moyenne arithmétique. Mais si le total représenté par une population ou un échantillon n'est pas la somme de leurs valeurs, la moyenne pertinente ne sera plus la moyenne arithmétique.

Pour rappel, la moyenne arithmétique s'exprime comme :

$$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

6.2.4 Variance et écart-type

Si sur cette base de moyenne arithmétique, nous tentons maintenant de calculer la moyenne des séries de nombres suivantes :

- 666, 666, 666, 666, 666, 666, 666
- 507, 693, 750, 644, 576, 680, 812
- 302, 1194, 500, 107, 1542, 599, 418

On constate que ces 3 séries ont exactement la même moyenne : 666. Mais cela n'a pas la même signification à chaque fois.

En effet, si l'on tente de mesurer les écarts entre chaque nombre des séries ci-dessus et la moyenne de chaque série, ce qu'on appelle la **déviation**, on constate le résultat suivant :

- $0 + 0 + 0 + 0 + 0 + 0 = 0$
- $-159 + 27 + 84 + -22 + -90 + 14 + 146 = 0$
- $-364 + 528 + -166 + -559 + 876 + -67 + -248 = 0$

La somme des écarts par rapport à la moyenne est égale à 0. C'est une des propriétés de la moyenne. En outre, on peut constater que ces écarts sont plus ou moins importants.

La **variance** est très utile pour éviter d'avoir une déviation qui est nulle. Elle sert à mesurer le niveau de dispersion d'une série de nombres par rapport à la moyenne. Elle consiste en la moyenne de la somme des carrés des déviations d'un groupe de nombres déterminés. Sa formule est la suivante :

$$\mathrm{Var}(X) = \frac{\sum (X - \overline{X})^2}{N}$$

Dans le cas des écarts qui nous intéressent, on obtient :

- $(0 + 0 + 0 + 0 + 0 + 0)/6 = 0$
- $(25281 + 729 + 7056 + 484 + 8100 + 196 + 21316)/7 = 9023.14286$
- $(132496 + 278784 + 27556 + 312481 + 767376 + 4489 + 61504)/7 = 226383.714$

Tout cela est bien beau mais l'unité de mesure de la variance est différente de l'unité de mesure de notre série de nombres initiale. Si par exemple, nos conversions sont en euros, nous nous retrouverons avec une variance en euro au carré. Si nos valeurs sont en mètres, la variance s'exprimera en m^2.

L'**écart-type** σ (déviation standard) permet de contourner ce problème. Il consiste tout simplement en la racine carrée de la variance. Par conséquent la variance peut aussi s'écrire σ^2.

- $\sqrt{0} = 0$

- $\sqrt{9023.14286} = 94.9902251$
- $\sqrt{226383.714} = 475.797976$

Une moyenne c'est intéressant. Mais le niveau de dispersion des données par rapport à la moyenne (l'écart-type) peut être très déstabilisant, au point de devenir monstrueux, anormal. Plus l'écart est élevé, plus la dispersion des données est élevée, plus il est faible, plus les données sont resserrées autour de la moyenne. L'écart type nous permet donc d'évaluer la dispersion des données par rapport à la moyenne théorique de celles-ci et l'écart moyen entre les données et la moyenne des données prises en considération.

6.2.5 Propriétés de l'écart-type

En résumé, on dispose des propriétés suivantes quand on utilise l'écart-type.

- On n'utilise l'écart-type que pour mesurer la dispersion autour de la moyenne d'un ensemble de données.
- L'écart-type n'est jamais négatif.
- L'écart-type est sensible aux valeurs aberrantes. Une seule valeur aberrante peut accroître l'écart-type et, par le fait même, déformer le portrait de la dispersion.
- Dans le cas des données ayant approximativement la même moyenne, plus la dispersion est grande, plus l'écart-type est grand.
- L'écart-type est zéro si toutes les valeurs d'un ensemble de données sont les mêmes (parce que chaque valeur est égale à la moyenne).

Quand on analyse des données normalement distribuées (voir plus loin), on peut utiliser l'écart-type parallèlement à la moyenne pour calculer des intervalles de données.

Si $\overline{x}$ = moyenne, σ = écart-type et x = une valeur incluse dans l'ensemble de données, alors :

- Environ 68 % des données se situent à l'intérieur de l'intervalle : $\overline{x} - \sigma < x < \overline{x} + \sigma$
- Environ 95 % des données se situent à l'intérieur de l'intervalle : $\overline{x} - 2\sigma < x < \overline{x} + 2\sigma$
- Environ 99 % des données se situent à l'intérieur de l'intervalle : $\overline{x} - 3\sigma < x < \overline{x} + 3\sigma$

6.3 Distribution

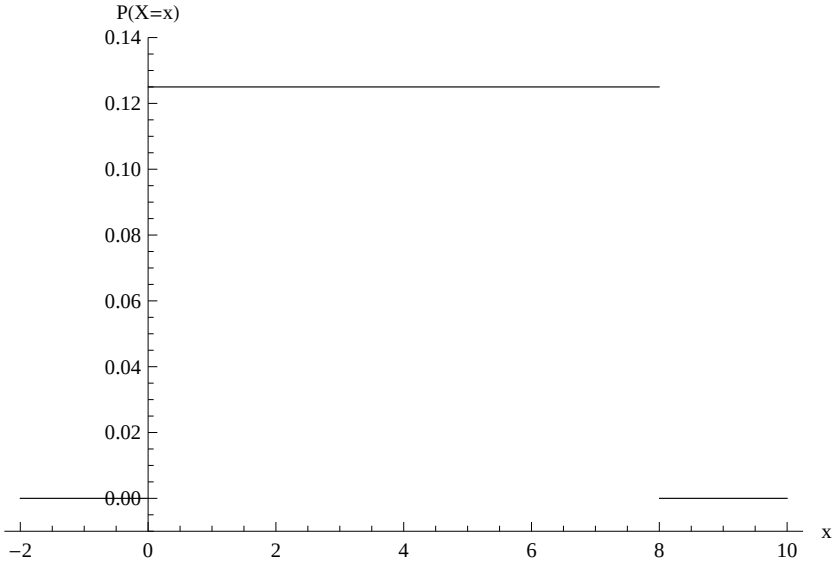
La distribution désigne l'action de répartir des choses ou des personnes selon différents critères. En statistique, ces critères sont souvent la moyenne et les paramètres qu'on y lie comme l'écart type ou la fréquence. Mais ce ne sont pas les seuls critères utilisés.

6.3.1 Distribution uniforme

Comme le nom de cette distribution l'indique, la répartition des valeurs respectant ce type de distribution est uniforme. Cela signifie qu'il y a une même probabilité d'avoir une valeur ou une autre parmi celles qui respectent cette distribution. Toute loi ou formule statistique qui respecte ce type de distribution aura donc cette propriété. Graphiquement, elle se présente sous la forme d'un rectangle.

Le graphique de la figure 2 représente une distribution uniforme de nombres réels entre 0 et 8. On remarque que pour les nombres $x < 0$ et $x > 8$, la probabilité est nulle, c'est-à-dire que $P(X = x) = 0$. Mais quelle est la probabilité d'obtenir un nombre entre 0 et 8 ? On sait que c'est la même quel que soit le nombre choisi puisque la distribution est uniforme et on sait aussi que la somme de toutes ces probabilités est 1. Donc la surface en-dessous de la droite entre 0 et 8 vaut 1. On parle de la surface d'une zone rectangulaire que l'on peut calculer par la formule bien connue *longueur* $\times$ *hauteur*. On a donc $1 = (8 - 0) * hauteur \Rightarrow hauteur = \frac{1}{8} = 0.125$. Il s'agit de la probabilité.

Figure 2: Distribution uniforme



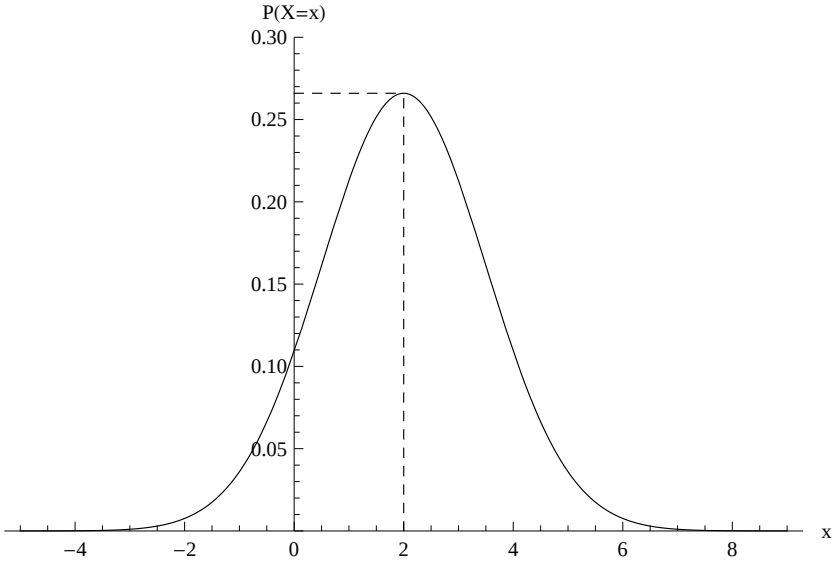
6.3.2 Distribution normale

Cette distribution est souvent appelée simplement ”courbe en cloche”, ”courbe en forme de cloche”, loi de Gauss ou loi normale. En effet, la plupart des résultats de ce diagramme sont groupés au centre. La moyenne, la médiane et le mode sont égaux et les résultats situés à chaque extrémité de la distribution sont moins fréquents. Par exemple, dans une courbe représentant les résultats d'un test de mesure du QI, la plupart des personnes se trouvent au centre ou autour de l'intervalle de QI ”moyen”, tandis que le nombre de personnes diminue des deux côtés à mesure que les résultats s'éloignent de la moyenne, d'où la forme et le nom de la courbe.

Le graphique de la figure 3 représente une distribution normale de moyenne 2 et d'écart-type 1.5. La probabilité d'obtenir la valeur x pour notre variable X notée $P(X = x)$ est donnée par la valeur de la courbe sur l'axe des ordonnées au point en question. Par exemple, $P(X = 2) = 0.265962$. Comme indiqué précédemment, environ 68 % des données se situent à l'intérieur de l'intervalle : $[\overline{x} - \sigma, \overline{x} + \sigma]$ c'est-à-dire à l'intérieur de l'intervalle $[0.5, 3.5]$. Cela veut dire que si on fait la somme de toutes les probabilités $P(X = x)$ pour $x \in [0.5, 3.5]$ on obtiendra approximativement la valeur 0.68[30].

[30]Lorsqu'une variable aléatoire est continue comme ici, la manière d'additionner un très grand nombre de probabilités sur de très petits intervalles pour réussir à avoir la probabilité sur l'intervalle complet $[0.5, 3.5]$ est d'utiliser une intégrale : $\int_{x=0.5}^{3.5} P(X = x)\, dx = 0.682689$.

Figure 3: Distribution normale



6.4 Génération de nombres aléatoires

Produire des nombres aléatoires pose une double difficulté : la production en elle-même bien sûr, mais surtout savoir caractériser le hasard. Et ce deuxième point est encore aujourd'hui un véritable problème ! En effet, même pour les mathématiciens, le caractère aléatoire est une notion difficile à appréhender.

Un algorithme est une suite d'opérations prédéfinies que l'on applique à des paramètres pour les modifier. Si l'on applique le même traitement aux mêmes paramètres, les résultats sont donc identiques. En ce sens, un algorithme est donc déterministe, à l'opposé de ce que l'on veut obtenir. Pourtant certaines opérations sont suffisamment imprévisibles pour donner des résultats qui semblent aléatoires. Les nombres obtenus sont donc appelés **pseudo-aléatoires**.

La principale raison pour laquelle on utilise de tels nombres est qu'il est plus facile d'en produire et que les méthodes sont plus efficaces. Il existe des domaines où l'utilisation de ces nombres à la place de "vrais" nombres aléatoires est possible. Ceci est possible à condition d'effectuer une étude numérique rigoureuse pour le prouver.

De ce fait, plusieurs algorithmes existent permettant de réaliser respectivement des générations de nombres respectant une distribution uniforme comme le "Mersenne Twister" ou une distribution normale comme le "Box Muller".

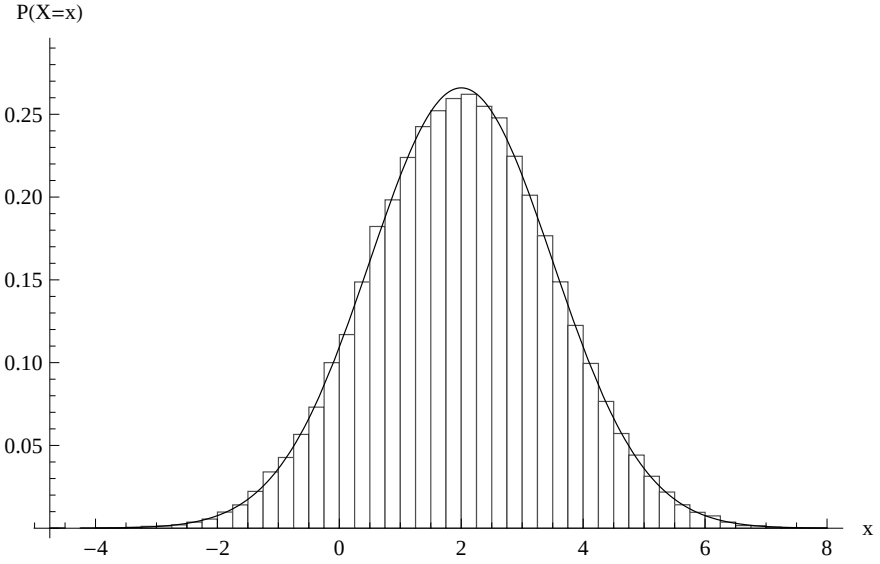
On peut partir de l'hypothèse que votre langage de programmation permet de générer une suite de nombres aléatoires selon une distribution uniforme comme en Java avec `Math.random()`. Le pseudo-code de la fonction `gaussian` ci-dessous permet de générer une paire de nombres aléatoires selon une distribution normale de moyenne et d'écart-type donnés. On peut se baser sur ce code et l'intégrer dans une classe de manière à pouvoir créer des générateurs de nombres aléatoires gaussiens. Notez que l'appel à cette fonction renvoie deux nombres, ce qui implique d'en utiliser un et de garder l'autre lorsqu'on aura besoin d'un nouveau nombre aléatoire. Ce fonctionnement peut être encapsulé dans une classe.

```
def gaussian(mean, stddev)
 theta = 2 * Math::PI * random()
 rho = Math.sqrt(-2 * Math.log(1 - random()))
 scale = stddev * rho
 x = mean + scale * Math.cos(theta)
 y = mean + scale * Math.sin(theta)
```

```
 return x, y
end
```

On peut se convaincre que le code est correct en générant par exemple 50 000 nombres avec Mathematica et en superposant ensuite la distribution de Gauss sur l'histogramme des probabilités des nombres générés. La résultat est indiqué sur le graphique de la figure 4. On voit clairement que ça concorde.

Figure 4: Validation du pseudo-code de génération selon une distribution normale



7 Indications préalables

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 8, page 31.

7.1 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 au près de l'un de des professeurs concernés par le projet.

7.2 Gestion des erreurs

Dans certains contextes de développement, les erreurs générées par le système ou par le programme ne sont pas gérées et affichées de manière correcte. Vos applications doivent utiliser des mécanismes de gestion des erreurs/exceptions mis à votre disposition dans les langages utilisés.

Il y aura donc presque systématiquement deux messages produits : un message généraliste mais clair destiné à l'utilisateur lambda et un message stocké précis destiné à l'administrateur de la base de données indiquant exactement quelle est l'erreur et à quel endroit elle a eu lieu. Toute erreur liée à la base de données doit donc obligatoirement être enregistrée dans une table en plus du message affiché à l'utilisateur.

Il est vivement conseillé d'utiliser un package ou une procédure indépendante, voire autonome, pour gérer ces insertions dans une table de log 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 / etc.), le numéro de l'erreur et le message d'erreur correspondant. Ces informations sont destinées à l'administrateur de la base.

7.3 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 comme Java Data Objects (JDO), Java Persistence API (JPA) dont Hibernate, ActiveRecord (Ruby), DataMapper (Ruby), Linq, etc. 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.

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.

7.4 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 comme JRuby[31].
- Un autre choix est d'utiliser ODP.NET[32] 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. L'année dernière des étudiants ont choisi d'utiliser divers langages comme Python, Perl ou PHP et ils ont rencontré des problèmes principalement 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 a résulté de ces choix nous a poussé à fortement déconseiller leur utilisation. Néanmoins, nous désirons laisser le choix du langage libre. Mais 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 si vous comptez les utiliser.

7.5 Choix du framework Web

Le choix du framework Web dépend du choix du langage. Nous insistons sur le fait qu'on ne veut pas que vous utilisiez une couche ORM que vous n'auriez pas écrite vous-même. Hormis cette restriction, le choix est libre.

Notez que nous avons intentionnellement demandé peu d'écriture de code côté Web. Notre idée est que dans le cadre d'un cours de SGBD, le Web doit intervenir un minimum. Par conséquent, nous privilégions l'utilisation de frameworks Web simples à mettre en oeuvre et l'utilisation de procédures stockées PL/SQL dans les bases Oracle de manière à réduire davantage la quantité de logique se trouvant du côté de l'application Web. Cela revient aussi à bien dissocier la partie modèle des parties contrôleurs et vues dans l'esprit du modèle MVC.

Nous attirons votre attention sur les points suivants :

- Le choix supporté par le cours est d'utiliser Java avec les servlets et les JSPs. Vous pouvez utiliser le moteur à servlets que vous préférez : Tomcat, Jetty ou un autre.
- Un choix non supporté par le cours est d'utiliser .NET pour réaliser l'application Web avec par exemple le langage C#. Cependant étant donné le bon interfaçage entre .NEt et Oracle, nous distinguons ce choix des autres choix non supportés.
- Les autres choix ne sont pas supportés par le cours et nous ne pourrons vous apporter qu'une aide limitée. Cet avertissement rejoint celui à propos du choix du langage de programmation.

[31] Il y a même des ressources à propos de JRuby chez Oracle : `http://www.oracle.com/technetwork/articles/dsl/jruby-oracle11g-330825.html`

[32] Oracle Data Provider for .NET: `http://www.oracle.com/technetwork/topics/dotnet/index-085163.html`.

8 Réalisations attendues

8.1 Pré-requis - Modélisation des données

8.1.1 Préambule quant à l'évaluation

Les trois sections qui suivent concernent des fonctionnalités qui forment un pré-requis à la présentation du reste des développements à réaliser. Il ne sera pas possible de présenter la moindre partie de l'application tant que les modélisations et scripts n'auront pas été soumis et validés.

8.1.2 Modélisations - MCD/MRD

On vous demande de modéliser à l'aide du modèle entité-association le système informatique présenté précédemment. Vous devrez également le convertir dans le modèle relationnel. Vous disposez de l'outil DB-MAIN[33] ou de tout autre outil de modélisation.[34]. Vous devrez rendre une version électronique au format pdf. Une mise en page soignée de vos schémas vous est demandée de manière à ce qu'on puisse les imprimer facilement si le besoin s'en fait sentir.

Notez que chaque BD du système doit faire l'objet d'une modélisation séparée. Donc, s'il y a n BDs, il y aura n paires de schémas conceptuels et logiques.

Toutes les contraintes applicatives qui ne peuvent pas être définies au moyen du LDD devront apparaître sous la forme de notes liées aux schémas.

8.1.3 Scripts SQL de création des schémas - CreaSql

Il s'agit de rédiger les scripts de création des tables *en veillant à placer un maximum de contraintes au niveau du schéma*. Les formats fournis par DB-Main ne tiennent pas compte des types propres à Oracle; il convient donc d'adapter le script LDD obtenu.

8.1.4 Scripts SQL d'insertion de données - InsertSql

Une fois rédigés les scripts de création des tables, il est demandé de prévoir les scripts d'insertion de données correspondant. Un minimum de 20 tuples est demandé par les tables. Il faut des scripts pour chacune des bases de données de l'application. Seules les tables relatives aux films et informations directement liées et qui seront alimentées par les procédures prévues ci-dessous ne font pas l'objet d'un script d'insertion.

8.2 Partie I - Première alimentation de CB et mise à jour de CB

Différentes sources d'alimentations sont possibles pour CB :

- CI pour les films et copies
- CC1 pour les retours de copies

Chacune de ces sources utilise des formats de données et surtout des structures de données différentes. Il faut donc les interroger mais aussi les transformer s'il y a lieu pour les intégrer dans CB.

[33] http://www.db-main.com/

[34] Parmi ces autres outils, Oracle propose le SQL Developer Data Modeler qui a pour avantage de permettre la rétro-ingénierie des structures

8.2.1 Alimentation de la centrale CB par CI - AlimCB

Un programme extérieur doit permettre d'alimenter régulièrement CB en films et copies (supports) sur base des informations se trouvant dans CI. C'est à dire que sur la base de critères prédéfinis sélectionnés ou non sur une petite interface, la liste de films correspondant doit apparaître. Un ou plusieurs de ces films ainsi que le nombre de copies que l'on en souhaite doivent alors pouvoir être transférés dans la CB.

Ce programme devra dès lors permettre au moins les deux types d'alimentation suivante :

- Une liste de film correspondant à un ou plusieurs critère(s) de recherche simultané(s) (au minimum : acteur(s), titre, réalisateur(s), date de sortie, genre (type), MPAA, popularité (rating), nombre de votes).
- Un film sur base de son identifiant.

Deux possibilités devront être offertes pour les quantités à commander.

- La première commandera une même quantité - fournie en paramètre - pour l'ensemble des films retenus.
- La seconde commandera une quantité aléatoire basée sur une distribution normale (cfr. section 6, page 23) pour chaque élément de la liste des films retenus avec une moyenne de 7.5 et un écart-type de 2. Bien entendu, le nombre de copies commandées est un nombre entier.

En ce qui concerne les quantités de copies à commander, il doit être possible de commander des copies physiques (bandes) ou des copies numériques.

Tous les films déjà présents dans CB ne devront bien sûr pas être dupliqués ni re-transférés mais la quantité de copies disponibles devra être mise à jour.

En ce qui concerne les images liées à un film, nous vous demandons de reprendre de la propriété `posters` une seule image dont la taille `size` vaut `cover`. Ces images seront téléchargées par HTTP et stockées sous forme de BLOBs dans CB.

Ces données seront reprises dans votre programme dans des structures de données de votre choix comme des tables hash, des ensembles ou autres[35] ainsi qu'une combinaison de ceux-ci de manière à pouvoir effectuer la recherche. Prenons un exemple : imaginons que vous deviez chercher les films dont les noms des réalisateurs sont R_1, R_2 et R_3 et dont le nombre de votes est compris entre 10 et 20. Vous allez d'abord filtrer votre structure de données contenant les informations ID/noms des réalisateurs de manière à obtenir l'ensemble des IDs qui conviennent pour ce premier filtre. Ensuite vous faites de même avec une structure de données triée contenant les informations ID/votes pour obtenir l'ensemble des IDs qui conviennent. Et enfin, il suffit de faire l'intersection des deux ensembles précédents puis d'interroger la vue CouchDB qui vous donnera toutes les informations sur un film grâce à son ID et ce pour chacun des IDs résultants.

Pensez également à utiliser les différents paramètres possibles lors de l'interrogation des vues CouchDB.[36]

Une attention particulière sera donnée à l'optimisation des connexions et du nombre d'interactions avec la base de données.

Techniques attendues : CouchDB/JSON, un langage au choix (cfr. 7.4, page 30), PL/SQL + séquences, transferts de BLOB.

8.2.2 Alimentation par défaut de CC1 - AlimCC1

Lors de la première réception de copies d'un film dans CB, un nombre aléatoire de celles-ci est livré automatiquement dans CC1. Une copie ne pouvant se trouver en même temps à la centrale CB et dans un complexe, il conviendra de décrémenter les stocks de copies. Ce nombre aléatoire devra se baser sur une

[35] En Java et dans les autres langages, il existe une série de structures de données efficaces et optimisées pour stocker les informations. Évitez donc d'utiliser des simples listes lorsque cela ne se justifie pas. Pour Java, vous pouvez aller voir dans le package `java.util` à `http://download.oracle.com/javase/`. En particulier on trouve des structures comme des `Set`.

[36] `http://wiki.apache.org/couchdb/HTTP_view_API#Querying_Options`

distribution uniforme allant de 0 à $\frac{n}{2}$ où n est le nombre de copies reçues lors de la première réception (cfr. section 6, page 23).

Techniques attendues : PL/SQL + réplication + DBlink.

8.3 Partie II - Gestion des copies

8.3.1 Recherche de copies - RechCopie

CC1 a la possibilité de recommander des copies de films fonctionnant bien ou ayant bien fonctionnés dans sa programmation. En effet, la centrale envoie régulièrement un nombre limité de ses nouveautés dans les complexes. Or ce nombre n'est pas forcément suffisant.

Il est demandé de générer une petite interface graphique (cfr. 7.4, page 30) listant les films pour lesquels le nombre de copies restante est sous un seuil fixé pour une popularité (nombre de tickets commandés) supérieure à un autre seuil. On l'aura compris, deux seuils sont donc à fournir à l'appel de cette fonction et elle devra retourner une liste triée de ces films sur base de la popularité.

Il n'est pas demandé de mise en forme particulière des films obtenus ou de pagination de la liste. Le plus important étant de pouvoir les identifier sans équivoque, les sélectionner et déterminer une quantité à commander ultérieurement. Attention cependant, seul un gestionnaire de complexe dûment identifié peut effectuer ces recherches.

Techniques attendues : Application cliente dans un langage au choix, PL/SQL.
Contrainte obligatoire : Aucun SQL direct, passage par des appels et retours de fonctions PL/SQL.

8.3.2 Commande de copies - CmdCopie

Le résultat de RechCopie (section 8.3.1, page 33) doit permettre de commander des copies. C'est à dire qu'il doit permettre de sélectionner certains des films sur base de cases à cocher et de préciser des quantités de copies à commander sur la même interface si l'on souhaite une quantité personnalisée autre que celle fixée par le système qui est de deux. Une fois les films sélectionnés et les quantités déterminées, la validation de ce choix génère automatiquement une commande qui arrive à centrale CB pour un traitement ultérieur.

Seul un gestionnaire identifié peut générer ces commandes.

Techniques attendues : Application cliente dans un langage au choix, PL/SQL, réplication, DBLink.
Contrainte obligatoire : Aucun SQL direct, passage par des appels et retours de fonctions PL/SQL.

8.3.3 Traitement des commandes de copies - TrCmdCopie

Quotidiennement, une ou plusieurs commandes de copies provenant des complexes arrivent à la centrale grâce à CmdCopie (section 8.3.2, page 33). Tous les jours également, un système automatisé sous la forme d'un job, collectera ces commandes et initiera les livraisons des nombres de copies demandées en tenant compte des stocks de copies disponibles. A l'arrivée de nouveaux stocks provenant de CI, certaines commandes n'ayant pas pu être totalement honorées devront donc être vérifiées pour les compléter.

Techniques attendues : PL/SQL + réplication + job + DBLink.

8.3.4 Retour de copies - RetourCopie

A chaque fois qu'une copie n'est plus utilisée dans une programmation, c'est à dire sitôt que se termine sa programmation, elle retourne à la centrale CB de manière à pouvoir être redistribuée. Un contrôle doit donc être effectué régulièrement pour déterminer si une copie est encore programmée ou non. Attention, il conviendra de pouvoir distinguer les copies non encore programmées, des copies en cours de programmation

mais aussi des copies n'étant plus programmées. Une copie répond à ce critère quand elle a été liée à une séance et une salle pendant une période mais qu'elle ne l'est plus. Les copies entrant dans ce cadre doivent repartir à la CB.

Cependant, pour optimiser les trajets des divers livreurs employés par l'entreprise, un retour à la CB ne pourra s'effectuer que lorsqu'une livraison de copies provenant justement de la CB arrive au complexe de manière à repartir dans le même camion.

Techniques attendues : PL/SQL + réplication + DBLink.

8.4 Partie III - Programmation des complexes

8.4.1 Programmation - ProgFilms

CC1 dispose de trois salles pour quatre séances par jour; une séance le matin, une l'après midi, deux en soirée. Un programme automatise la programmation du complexe en attribuant à chaque séance non pourvue une copie de film parmi les disponibles pour une durée de n jours, n étant choisi aléatoirement de manière uniforme avec $n \leq 21$ et entier. Dès lors, sitôt qu'on détecte qu'une copie n'est plus programmée, ce programme s'exécute pour attribuer une nouvelle copie aléatoirement à la séance ainsi libérée.

Techniques attendues : PL/SQL + nombres aléatoires.

8.5 Partie IV - Commandes de tickets

8.5.1 Recherche de Programmation - RechProg

Le complexe propose un site Web permettant la commande en ligne de places de cinéma. Un module de recherche reprenant la programmation hebdomadaire du complexe est mis à la disposition des visiteurs du site Web.

En effet, il est demandé de pouvoir faire des recherches sur :

- Un film
- Le nombre de places disponibles, sous une limite, au-dessus d'une limite
- Une séance
- Une période

Ou toute combinaison d'un ou plusieurs des critères ci-dessus. Ainsi, par exemple, on doit pouvoir retrouver toutes les séances disposant de plus de x places disponibles.

Différentes solutions de codage sont possibles pour construire la requête correspondante comme par exemple une combinaison de `AND` et de `OR`, ou l'utilisation de `LIKE %` ou encore l'utilisation de `EXECUTE IMMEDIATE`. A vous de voir la technique que vous choisirez. Il vous est demandé de la construire de la manière la plus générique possible et que tout ajout ultérieur d'un ou plusieurs critères n'engendre pas une multitude de code supplémentaire. De même qu'une recherche lancée sans aucun critère doit retourner l'ensemble de la programmation.

Outre les détails strictement utiles à l'identification formelle d'une séance et du nombre de places encore disponibles, les informations que reprend le résultat de la recherche se base sur les films et leur popularité (nombre de tickets vendus depuis que le film est projeté dans le complexe). Un clic sur une séance doit permettre l'affichage de la fiche signalétique du film projeté à cette séance ainsi que son affiche.

Cet affichage est également conçu afin d'être complété plus tard par une sélection directe des places que l'on souhaite commander. Il faut donc le prévoir suffisamment modulaire que pour pouvoir facilement adapter le code à cette modification.

Les séances sont affichées par bloc d'une journée par page. Il doit être aisé de naviguer de page en page. La navigation se fait au moyen de boutons ou de liens. La requête permettant de récupérer ces listes est également optimisée. La base de données étant assez conséquente, il est hors de question qu'apparaisse une requête générale reprenant tous les films, suivie d'une boucle ne conservant que la journée à afficher. Différentes techniques PL/SQL existent à ce sujet, il est demandé de les exploiter.

Une des possibilités consiste à utiliser les ROWNUMs. La pseudo-colonne `ROWNUM` est utilisée de manière à connaître l'ordre des résultats. Mais attention, comme il est indiqué dans la documentation Oracle, "conditions testing for ROWNUM values greater than a positive integer are always false. For example, this query returns no rows: `SELECT * FROM employees WHERE ROWNUM > 1`. The first row fetched is assigned a `ROWNUM` of 1 and makes the condition false. The second row to be fetched is now the first row and is also assigned a `ROWNUM` of 1 and makes the condition false. All rows subsequently fail to satisfy the condition, so no rows are returned."

Une manière de procéder est indiquée à la section "Pagination with ROWNUM" de l'article "On ROWNUM and Limiting Results"[37] par Tom Kyte. On vous demande d'implémenter cette technique. Vous n'êtes pas obligé d'utiliser le commentaire destiné à l'optimiseur Oracle dont la forme est `/*+ FIRST_ROWS(n)*/`.

Une autre technique passe par la commande SQL OVER qui permet de définir une fenêtre ou zone de données parmi les résultats d'une requête et ne retourner que les tuples se trouvant délimité par les bornes supérieures et inférieures de la fenêtre.

Techniques attendues : Web dans un langage au choix, ressources statiques, PL/SQL, manipulation de BLOBs, pagination.
Contrainte obligatoire : Aucun SQL direct, passage par des appels et retours de fonctions PL/SQL.

8.5.2 Commande Places - CmdPlaces

A partir de RechProg (section 8.5.1, page 34), il doit être possible de commander des places de cinéma. Le visiteur indique donc une quantité souhaitée face à la séance qui l'intéresse, et insère la demande dans un "panier". En outre, si l'on sélectionne une séance se trouvant déjà dans le panier, sa quantité sera simplement augmentée de la nouvelle quantité demandée. Il doit être possible de commander des places pour des séances différentes en même temps.

Le contenu du panier doit être consultable à tout moment. De même que l'on doit pouvoir modifier des quantités, voir supprimer tout ou partie du panier. A nouveau, le panier est paginé, navigable et intégré dans le site. Un maximum de cinq séances sont visibles par page.

Finalement, le gestionnaire peut valider son panier ce qui a pour effet de générer une commande sur CC1.

Une attention particulière doit être portée à l'ergonomie et la convivialité de votre site. L'utilisateur doit savoir si son opération s'est bien déroulée ou pas. De même, il doit pouvoir être sollicité pour confirmer une opération si nécessaire. Toute erreur ou problème doit être signifié à l'utilisateur dans un langage clair et compréhensible, comme pour l'ensemble des interfaces à réaliser.

Techniques attendues : Web, PL/SQL + réplication + DBLink.
Contrainte obligatoire : mise en oeuvre de GesSession (section 8.5.3, page 35).

8.5.3 Gestion de session - GesSession

Un panier est lié à une session et non pas à un utilisateur. Cependant, on ne peut emmagasiner des paniers inutilement. C'est la raison pour laquelle il est demandé de mettre en oeuvre deux mécanismes de gestion de fin de session :

[37] `http://cours.khi.be/sgbd3/labo/o56asktom.html` ou `http://www.oracle.com/technetwork/issue-archive/2006/06-sep/o56asktom-086197.html`

- Par demande volontaire (logoff)
- Par timer (au terme de 30 min de navigation)

A la fin de la session, le panier correspondant doit être vidé.

Techniques attendues : Web, PL/SQL.

9 Organisation pratique et évaluations

9.1 Téléversement du code avant évaluation

Tout le code utilisé dans votre projet (scripts SQL, code PL/SQL, code source de votre langage, …) concerné par l'évaluation que vous allez présenter doit être téléversé avant l'évaluation. On vous demande de respecter le format indiqué ci-dessous où le `groupe` est une séquence de quatre chiffres comme 2321, les noms `nom1` et `nom2` sont indiqués en minuscules par ordre alphabétique en ayant supprimé les espaces et transformé les lettres accentuées en leur équivalent non accentué (par exemple `é` → `e`) et les fonctionnalités présentées avec ce code sont indiquées en minuscules dans n'importe quel ordre sans espace séparées par des tirets comme avec `fct1` et `fct2`. L'archive est de type rar ou zip. Notez que si l'évaluation est réalisée seul alors il n'y aura pas de `nom2`.

Le format est le suivant :

```
groupe-nom1-nom2-fct1-fct2.ext
```

Par exemple, si Ludovic Kuty et Laurence Herbiet du groupe 2327 présentent les fonctionnalités `RechCopie` et `CmdCopie`, on aura une de ces possibilités :

```
2327-herbiet-kuty-cmdcopie-rechcopie.rar
2327-herbiet-kuty-rechcopie-cmdcopie.rar
2327-herbiet-kuty-cmdcopie-rechcopie.zip
2327-herbiet-kuty-rechcopie-cmdcopie.zip
```

L'archive devra être téléversé par FTP (port 21) sur `nemo` dont l'IP est 10.59.26.135 avec l'utilisateur `anonymous` sans mot de passe et dans le répertoire `/upload`. Notez qu'il n'est pas possible de voir la liste des fichiers déjà présents, ni d'en télécharger ou d'en effacer. Vous pouvez juste téléverser (upload) votre archive. Normalement votre client FTP devrait ne pas rencontrer de problème même si les commandes de listage (`LIST` et `NLST`) sont interdites. C'est le cas pour Filezilla par exemple.

Voici un exemple de session en ligne de commande avec `ncftp` :

```
lk@eve:~$ dd if=/dev/zero of=2327-herbiet-kuty-cmdcopie-rechcopie.rar bs=1024k count=1
1+0 records in
1+0 records out
1048576 bytes (1.0 MB) copied, 0.0138724 s, 75.6 MB/s
lk@eve:~$ ls -l 2327-herbiet-kuty-cmdcopie-rechcopie.rar
-rw-r--r-- 1 lk prof 1048576 Sep  8 17:29 2327-herbiet-kuty-cmdcopie-rechcopie.rar
lk@eve:~$ ncftp 10.59.26.135
Connecting to 10.59.26.135...
(vsFTPd 2.2.2)
Logging in...
Login successful.
Permission denied.
Logged in to 10.59.26.135.
ncftp / > cd upload
Directory successfully changed.
ncftp /upload > put 2327-herbiet-kuty-cmdcopie-rechcopie.rar
2327-herbiet-kuty-cmdcopie-rechcopie.rar:                1.00 MB    4.59 MB/s
Could not preserve times for 2327-herbiet-kuty-cmdcopie-rechcopie.rar: UTIME failed.
ncftp /upload > ls
List failed.
```

9.2 Tableau synthétique

Table 4: Organigramme d'évaluation

Échéance maximale	Fonctionnalité	Poids	Pondération
Pre-requis	MCD/MRD		
	CreaSQL		
	InsertSQL		
16/11/2012	AlimCB	2	30
	AlimCC1	1	10
	RechCopie	1	15
	CmdCopie	1	15
	TrCmdCopie	1	10
	RetourCopie	1	10
	ProgFilms	2	20
21/12/2012	RechProg	2	30
	CmdPlaces	2	20
	GesSession	1	10

9.3 Modalités d'évaluation du laboratoire

9.3.1 Premier semestre et Evaluation continue

- L'évaluation de laboratoire du cours de Systèmes de Gestion de Bases de Données (SGBD) est une évaluation continue.
- Le laboratoire constitue 100% de la cote finale de ce cours.
- Le travail est prévu pour deux mais chaque étudiant doit être capable de répondre à toute question du titulaire portant sur l'application présentée ou la matière théorique et pratique liée à l'application.
- A chaque fonctionnalité est associée une date butoir. Au delà de cette date, toute fonctionnalité non présentée aura la cote de zéro et ne pourra faire l'objet d'une présentation d'amélioration en janvier.
- Les étudiants sont libres d'organiser et répartir leur travail comme ils le souhaitent.
- A chaque fonctionnalité est associé un poids, le cumul des poids des fonctionnalités présentées lors d'une séance de laboratoire ne peut excéder 2.
- La cote finale représente le cumul des points des différentes fonctionnalités réalisées au cours du semestre. Les points attribués à chaque fonctionnalités sont indiqués dans le tableau 4 de la section 9.2.
- En janvier, il est possible, à toute équipe le souhaitant, de proposer une amélioration de fonctionnalité(s) présentée(s) précédemment à condition que la cote obtenue pour chaque fonctionnalité soit $\geq$ à 25 % du total de la fonctionnalité considérée. Dans ce cas, la cote finale de la fonctionnalité sera la moyenne arithmétique des cotes obtenues aux deux présentations.
- Le code des fonctionnalités présentées au cours d'une séance doit être envoyé au préalable sur le serveur FTP mis à disposition des cours concernés (cfr. section 9.1). C'est sur ce code que sera effectuée l'évaluation. Sans code déposé sur le serveur, il ne peut y avoir de présentation. De même, si le format des noms de fichier n'est pas respecté. La date de dépôt sur le serveur FTP est celle qui constitue la date de remise du code de vos fonctionnalités.
- En janvier, seules des améliorations peuvent être présentées.

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

9.3.2 Deuxième session

L'examen de laboratoire de seconde session sera pondéré à 100%. Il s'agit de compléter les fonctionnalités du premier semestre. L'évaluation sera reprise de zéro et compte tenu du délai supplémentaire, des commentaires et corrections donnés lors de l'évaluation continue de première session, il est possible de ne pas obtenir autant de points pour certaines réalisations qu'en janvier. La cote de première session n'est pas conservée si celle de 2ème session est moins bonne.

Vous pouvez être dispensé d'une fonctionnalité et garder la cote de première session selon les modalités indiquées dans le tableau à la section 9.4.

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

9.3.3 Prolongation de session

Pour les étudiants qui se retrouveraient dans la situation d'une prolongation de session pour l'année 2012-2013, l'épreuve portera sur l'énoncé de l'année académique prolongée. L'examen peut être présenté dès qu'un rendez-vous est pris avec le titulaire de laboratoire. Les modalités sont les mêmes qu'en 2ème session.

L'évaluation de laboratoire sera reprise de zéro et compte tenu du délai supplémentaire, il est possible de ne pas obtenir autant de points pour certaines réalisations qu'aux sessions précédentes. La cote de première session ni celle de 2e session ne sont conservées si celle de prolongation session est moins bonne.

Vous pouvez être dispensé d'une fonctionnalité et garder la cote de seconde session selon les modalités indiquées dans le tableau à la section 9.4.

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

9.4 Synthèse des modalités d'évaluation

Session	Résultat	Dispenses partielles pour la session suivante
1ère	Somme des fonctionnalités (moyenne si amélioration)	$\geq 50\%$ pour la fonctionnalité
2ème	Somme des fonctionnalités (repart de 0)	$\geq 50\%$ pour la fonctionnalité
Prolongation	Somme des fonctionnalités (repart de 0)	