Rennequinepolis (RQS)

Version du 19 novembre 2014 12:38

Structure de l'information et bases de données
1ère Master en Sciences de l'ingénieur industriel finalité informatique (groupe M18)

Ludovic Kuty <ludovic.kuty@hepl.be>

2014 – 2015

Table des matières

1 Préambule

Ce document présente le projet à réaliser dans le cadre des laboratoires du cours "Structure de l'information et bases de données" des M18. Ce projet est divisé en deux parties.

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 que nous vous demandons de bien vouloir lire attentivement et signer pour indiquer votre prise de connaissance de celles-ci.

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.

Le login et le mot de passe à utiliser sont respectivement "isil" et "lisi".

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'informations : 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.

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.
- **Les cinémas - CCx** - Les différents complexes cinématographiques que gère RQS dont CC1.

2.2 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, ses genres, le public concerné, les acteurs principaux, une affiche, une ou plusieurs langues parlées, …. Cette signalétique est à distinguer de la copie physique 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 être suffisamment complètes pour pouvoir distinguer les copies (chaque copie est donc identifiée de manière précise) 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 stocké dans une BD MongoDB (cfr. section 4, page 9). On retrouve 26 propriétés (ou champs) par film expliquées au tableau 1. De ces 26 propriétés, nous n'en utiliserons que 21, à savoir celle qui n'ont pas été barrées dans le tableau.

Attention, certaines données pourraient légèrement différer de ce qui est mentionné. Par exemple certaines valeurs pour la propriété `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. On vous demande donc d'examiner 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 de la méthode de shell `distinct`[1] et des informations statistiques fournies dans le fichier `stats.txt`.

Dans le même esprit, vérifiez que les propriétés existent et si elles existent avec une valeur spéciale indiquant l'absence de valeur comme `null`, 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 votre base de données Oracle CB est un point important de l'évaluation de la fonctionnalité concernée.

Tab. 1: Propriétés présentes dans un film

Nom	Explications
`actors`	Un tableau d'acteurs. Chaque acteur est un document contenant les propriétés suivantes: `id`, `name`, `character` (le nom de leur personnage), `order`, `cast_id` et `profile_path` qui contient un chemin permettant de construire l'URL vers une image de la personne. Nous sommes exclusivement intéressés par `id`, `name` et `character`.
~~`adult`~~	Indique si le film est réservé aux adultes ou non. Valeurs: `true` ou `false`. Nous ne l'utiliserons pas.
~~`backdrop_path`~~	Un chemin permettant de construire l'URL vers une image de type backdrop pour le film. La manière de procéder est indiquée à la section 5.1, page 18. Nous ne l'utiliserons pas.
~~`belongs_to_collection`~~	Des informations sur la collection dont le film fait potentiellement partie. Nous ne l'utiliserons pas.
`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[2].

Suite sur la page suivante

[1] Cfr. `http://docs.mongodb.org/manual/reference/method/db.collection.distinct/#db.collection.distinct`

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

Tab. 1 – continuation de la page précédente

Nom	Explications
`directors`	Un tableau de réalisateurs. Chaque réalisateur est un document contenant les propriétés suivantes: `id`, `name`, `department` qui vaut "Directing", `job` qui vaut "Director" et `profile_path` qui contient un chemin permettant de construire l'URL vers une image de la personne. Nous sommes exclusivement intéressés par `id` et `name`.
`genres`	Les genres dont fait partie le film. Un tableau de documents dont chacun possède les propriétés suivantes: `id` qui est l'identifiant du genre sous forme de *number* et `name` qui est le nom du genre comme "Action", "Comedy" ou "Thriller".
`homepage`	Page web du film.
`id`	L'identifiant TMDb du film.
~~`imdb_id`~~	L'identifiant IMDb. Nous ne l'utiliserons pas.
`original_title`	Le titre original du film. 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 en UTF-8. Ce résumé peut être écrit dans une langue autre que l'anglais.
~~`popularity`~~	La popularité du film. Nous ne l'utiliserons pas. A ne pas confondre avec la popularité dont nous faisons usage.
`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 5.1, page 18.
`production_companies`	Un tableau des entreprises ayant participé à la production du film. Pour chaque entreprise on a un objet avec les propriétés suivantes: `id` et `name`.
`production_countries`	Un tableau des pays ayant participé à la production du film. Pour chaque pays on a un objet avec les propriétés suivantes: `iso_3166_1` et `name`. Le premier pays indiqué est considéré comme étant le pays d'origine. Inutile de reprendre les autres.
`release_date`	La date de sortie du film au format `YYYY-MM-DD`.
`revenue`	Le revenu qu'a généré le film en dollars.
`runtime`	La durée du film en minutes.

Suite sur la page suivante

Tab. 1 – continuation de la page précédente

Nom	Explications
`spoken_languages`	Un tableau d'objets représentant les langues parlées dans le film. Chaque objet possède les propriétés suivantes: `iso_639_1` pour identifier la langue (comme "de" ou "en") et `name` qui est le nom de la langue (comme "German" ou "English").
`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 qui a servi à établir le `rating`.

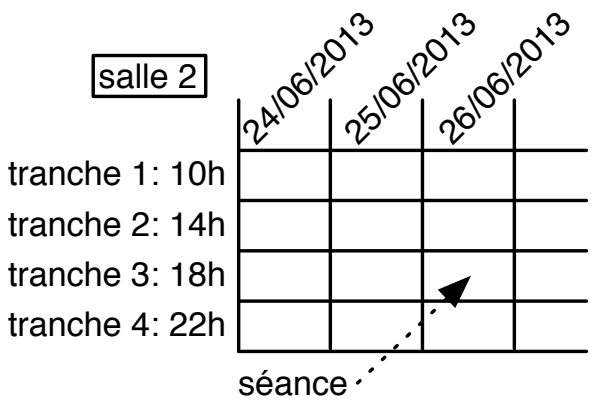
2.3 Les programmations

Dans les complexes, nous allons programmer des projections de films. Le vocabulaire utilisé est le suivant :

- Une **salle** (room) d'un cinéma est identifiée par un nombre au sein du cinéma. Par exemple la salle 1, 2 ou 3.
- Chaque **jour** (day) donné est identifié par une date précise de la forme JJ/MM/AAAA.
- Une **tranche horaire** (time slot) est identifiée par un nombre. Il s'agit de la période de projection d'un film dans une journée : 1 pour 10h, 2 pour 14h, 3 pour 18h et 4 pour 22h.
- Une **séance** (projection) désigne la projection d'un film donné (avec une copie précise), à un jour donné dans une salle donnée à une tranche horaire donnée.

Les différents objets utilisés pour désigner une séance sont illustrés à la figure 1.

Fig. 1: Les programmations



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 :

- Utiliser une VM pour Oracle et installer vous-même MongoDB sur votre machine hôte ou sur la VM.
- Installer tout manuellement : Oracle et MongoDB.

Si vous souhaitez utiliser Oracle en VM, nous vous invitons à reprendre la machine virtuelle intitulée "Database App Development VM" sur le site d'Oracle[3]. 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
...
```

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

[3] `http://www.oracle.com/technetwork/community/developer-vm/index.html`

- 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 MongoDB

Les données de CI sont stockées dans une base de données MongoDB[4] nommée `movies` qui contient une collection. Cette collection est également nommée `movies` et elle contient des documents qui représentent les films. Une collection est l'équivalent d'une table dans le monde relationnel et un document est l'équivalent d'un tuple. MongoDB une base non relationnelle orientée documents qui fait partie de la catégorie des BDs dites NoSQL[5] comme CouchDB.

Le nom de collection complètement qualifié `movies.movies` qui est la concaténation du nom de la base de données avec la nom de la collection est appelé un espace de noms (namespace) ou NS pour faire court.

Nous avons rapatrié les informations de tous les films de TMDb[6] à l'aide de requêtes HTPP GET en utilisant leur API[7] version 3 de type REST. Nous disposons d'un ensemble de taille réaliste de plus de 180 000 films. Cet ensemble de films pourrait être mis à jour régulièrement sur le site web du cours.

Il s'agira pour vous de récupérer notre BD `movies` de manière à disposer d'une copie locale. Ensuite grâce à un programme vous irez stocker les informations dans les BDs Oracle correspondantes selon ce qui est indiqué à la section 7, page 24.

Un document est un ensemble ordonné de paires clé / valeur. La représentation interne des documents est réalisée dans le format BSON[8] qui est une sérialisation binaire des informations contenues dans le document. La représentation externe varie selon le langage de programmation utilisé mais on rencontre typiquement celle de Javascript où un document correspond à un objet JS contenant des propriétés ayant chacune une valeur. D'ailleurs, le shell `mongo` est un interpréteur JS ce qui fait que les résultat des requêtes sont des données JS. En Java par exemple, on pourra utiliser une classe spécifique du driver ou une structure de données classique comme une `Map`.

Par conséquent, nous sommes particulièrement intéressés par la correspondance (mapping) entre les types de données de BSON et les types de données du langage que nous avons choisi d'utiliser. Cet intérêt est récurrent en informatique puisqu'on doit souvent faire interagir des systèmes dont les types de données diffèrent. Pensez par exemple à SQL et Java par l'intermédiaire de JDBC.

Tous les documents ont la même structure dans ce cas-ci même si ce n'est pas imposé par MongoDB.

Nous vous invitons à lire la section "Getting Started" de la documentation[9] qui débute par une introduction à MongoDB. Nous utiliserons une API pour interagir avec la base de données ou du Javascript dans la console `mongo` ce qui implique que nous ne serons pas directement confrontés à du BSON et à son encodage/décodage binaire. Toutes les données textuelles (les chaînes de caractères) sont encodées en UTF-8.

[4] `http://www.mongodb.org/`
[5] `http://en.wikipedia.org/wiki/NoSQL`
[6] `http://www.themoviedb.org/`
[7] `http://docs.themoviedb.apiary.io/`
[8] `http://bsonspec.org/`
[9] `http://docs.mongodb.org/manual/`

4.1 Installation et démarrage

MongoDB est entre autre disponible sous Linux, Windows, OS X et Solaris. Les versions 2.6.x sont supportées dans le cadre du cours. Nous vous conseillons de lire la procédure relative à l'installation indiquée dans la documentation. Actuellement vous disposez des possibilités suivantes :

- Sous Windows, cfr. ci-dessous.
- Sous Linux, vous avez deux options: compiler à partir des sources ou trouver MongoDB comme un package de votre distribution.
- Sous OS X, installez le package `mongodb` avec Homebrew[10]. Notez que vous aurez besoin d'installer Xcode grâce à l'App Store.

4.1.1 Windows

L'installation a été testée sous Windows 7. Celle-ci demande quelques opérations manuelles.

1. Installez d'abord le hotfix[11] si vous avez un Windows 7.
2. Téléchargez et décompressez le fichier zip qui correspond à votre version de Windows. Ce sera probablement une variante de `mongodb-win32-x86_64-2008plus-2.6.3.zip`. Inutile d'utiliser le `.msi` étant donné que l'installation ne vous fournit rien de plus qu'une copie des fichiers dans un répertoire.
3. Pour simplifier les chemins, nous avons choisi d'installer MongoDB dans le répertoire `c:\mongodb`. Si la décompression a produit un autre nom de répertoire se trouvant à un autre endroit nous vous suggérons de le renommer et de le déplacer.
4. Cette étape est optionnelle mais vous facilitera la vie en rendant accessible les commandes MongoDB depuis n'importe quel répertoire dans le shell. Mettez à jour la variable d'environnement système `Path` en allant sur `Computer / clic-droit / Properties / Advanced system settings / Advanced / Environment Variables... / System variables / Path` et ajoutez `c:\mongodb\bin` comme dernier chemin en le séparant du chemin précédent par un point-virgule.
5. Créez un répertoire `data` dans le répertoire de MongoDB, c'est-à-dire dans notre cas `c:\mongodb\data`. Celui-ci contiendra nos bases de données et en particulier la base `movies`.
6. Démarrez MongoDB en lançant un shell de commande, en vous plaçant dans le répertoire `c:\mongodb\bin` et en écrivant `mongod --dbpath c:\mongodb\data`. Vous remarquerez que MongoDB installe un certain nombre de fichiers et répertoires dans le répertoire `data`. Lorsque vous avez fini d'utiliser le serveur, vous pouvez l'arrêter avec un CTRL+C.

```
c:\mongodb\bin>mongod --dbpath c:\mongodb\data
2014-06-21T11:01:39.983+0200 [initandlisten] MongoDB starting : pid=3532 port=27
017 dbpath=c:\mongodb\data 64-bit host=ludo
2014-06-21T11:01:39.984+0200 [initandlisten] targetMinOS: Windows 7/Windows Serv
er 2008 R2
2014-06-21T11:01:39.984+0200 [initandlisten] db version v2.6.3
...
2014-06-21T11:01:39.984+0200 [initandlisten] options: { storage: { dbPath: "c:\m
ongodb\data" } }
2014-06-21T11:01:40.006+0200 [initandlisten] journal dir=c:\mongodb\data\journal
2014-06-21T11:01:40.027+0200 [initandlisten] waiting for connections on port 270
17
...
```

[10] http://brew.sh/

[11] http://support.microsoft.com/kb/2731284

4.2 Post-installation

1. Arrêtez le serveur MongoDB s'il est lancé.
2. Téléchargez le fichier rar nommé `movies.rar` contenant la BD `movies` sur le site web du cours.
3. Décompressez l'archive dans le bon répertoire ("extract here"). Le répertoire diffère selon l'OS comme indiqué ci-dessous. Dans ce répertoire vous devez avoir des fichiers nommés `movies.xxx` où `xxx` est remplacé par des nombres 0, 1, … ou `ns`.
   - Windows : `c:\mongodb\data`
   - Linux : `/var/lib/mongodb`
   - OS X : `/usr/local/var/mongodb` (avec Homebrew)
4. Redémarrez le serveur MongoDB et testez si la collection `movies` est bien présente.

```
C:\Users\ludo>mongo
MongoDB shell version: 2.6.3
connecting to: test
Welcome to the MongoDB shell.
...
> show dbs
admin  (empty)
local  0.078GB
movies   0.953GB
> use movies
switched to db movies
> db.movies.count()
174966
> exit
bye
```

4.3 Administration et interrogation de la BD

On interagit avec la base de données de deux manières que ce soit pour l'administrer ou pour l'utiliser :

- Soit avec l'outil en ligne de commande appelé `mongo`[12].
- Soit par programmation à l'aide de ce qu'on appelle un driver. Les drivers officiellement supportés sont indiqués sur le site de MongoDB[13]. Nous effectuerons des tests avec le driver de Java[14].

Le serveur de MongoDB appelé `mongod` est en écoute sur le port 27017.

Le serveur `mongod` est configuré à l'aide d'un fichier dont on peut personnaliser l'emplacement à l'aide de l'option `--config` lors du lancement de celui-ci. Il existe deux formats de fichier : l'ancien de la version 2.4[15] qui est toujours utilisable et celui de la version 2.6[16] au format YAML.

Si vous désirez que le serveur MongoDB écoute sur une interface publique plutôt que `localhost`, vous pouvez ajouter ou modifier le paramètre `bind_ip` (version 2.4) dans le fichier de configuration. Celui-ci est localisé à différents endroits selon l'OS. Par exemple, sous OS X on lance le serveur en écrivant dans un terminal après une installation par défaut avec Homebrew :

[12] `http://docs.mongodb.org/manual/reference/program/mongo/`
[13] `http://docs.mongodb.org/ecosystem/drivers/`
[14] `http://docs.mongodb.org/ecosystem/drivers/java/`
[15] `http://docs.mongodb.org/v2.4/reference/configuration-options/`
[16] `http://docs.mongodb.org/manual/reference/configuration-options/`

```
mongod --config /usr/local/etc/mongod.conf
```

Le fichier est localisé à :

- Windows : `c:\mongodb\mongod.conf` par exemple
- Linux : `/etc/mongod.conf`
- OS X : `/usr/local/etc/mongod.conf` avec Homebrew par défaut

4.4 L'interrogation des données

Une BD MongoDB est constituée d'un ensemble de documents, chacun possédant un identifiant `_id` qui peut être qualifié de clé primaire si on utilise le jargon des BDs relationnelles. Par défaut, un index est construit sur ce champ (ou propriété) `_id` de manière à pouvoir accéder rapidement aux données comme on peut le voir ci-dessous. Il s'agit d'ailleurs du seul index défini par défaut sur toute collection.

```
mac:~ ludo$ mongo
MongoDB shell version: 2.6.2
connecting to: test
> use movies
switched to db movies
> db.movies.getIndexes()
[
  {
    "v" : 1,
    "key" : {
      "_id" : 1
    },
    "name" : "_id_",
    "ns" : "movies.movies"
  }
]
> exit
bye
```

L'interrogation (querying) d'une BD MongoDB se fait à l'aide de "query documents" qui sont des sortes de templates de document dans lesquels on précise les valeurs désirées ou les expressions de recherche pour certains champs.

On peut aussi faire des projections comme en SQL de manière à n'avoir qu'un sous-ensemble des données.

Nous allons utiliser la documentation de référence pour déterminer la syntaxe des différents opérateurs et commandes. Celle-ci est disponible à `http://docs.mongodb.org/manual/reference/` et en particulier les méthodes du shell `mongo` à `http://docs.mongodb.org/manual/reference/method/`. Il est également intéressant de disposer du glossaire des termes MongoDB à `http://docs.mongodb.org/manual/reference/glossary/`.

Avant l'exécution des exemples ci-dessous, il est nécessaire de sélectionner la base de données `movies` à l'aide de la commande `use`.

```
> use movies
switched to db movies
```

Lorsqu'on désire utiliser un langage de programmation pour accéder à une base de donnée MongoDB, on utilise un driver pour ce langage. Pour Java, la documentation et le driver lui-même sont disponibles

à `http://docs.mongodb.org/ecosystem/drivers/java/`. En particulier la Javadoc de l'API est accessible à `http://api.mongodb.org/java/current/`. Dans la section "Getting Started with Java Driver" vous avez un exemple de connexion à MongoDB de manière à obtenir un objet de type `com.mongodb.DB` représentant votre base de données.

En java, nous utiliserons toujours le code ci-dessous pour nous connecter à notre base de données.

```
import com.mongodb.DB;
import com.mongodb.DBCollection;
import com.mongodb.MongoClient;
import java.net.UnknownHostException;
import java.util.logging.Level;
import java.util.logging.Logger;

public class MongoDBConnection {
    public static void main(String[] args) {
        try {
            MongoClient mongoClient = new MongoClient("localhost");
            DB db = mongoClient.getDB("movies");
            DBCollection collection = db.getCollection("movies");
            System.out.printf("Size of movies collection is %d\n", collection.count());
        } catch (UnknownHostException ex) {
            Logger.getLogger(MongoDBConnection.class.getName()).log(Level.SEVERE, null, ↩
            ex);
        }
    }
}
```

Nous allons également utiliser notre méthode `iterateOverCursor` pour afficher en détail tous les documents renvoyés par un curseur. Cette méthode utilise les streams de Java 8 et l'opération `forEach`.

```
public static void displayDocument(DBObject dbObject) {
    System.out.printf("*** Object %d ***\n", dbObject.hashCode());
    dbObject.keySet().stream().forEach(key -> {
        Object value = dbObject.get(key);
        System.out.printf("KEY: %s, TYPE: %s, VALUE: %s\n",
                key,
                value != null ? value.getClass().getName() : "N/A",
                value != null ? value.toString() : "null");
    });
}

public static void iterateOverCursor(DBCursor dbCursor) {
    dbCursor.forEach(dbObject -> {
        displayDocument(dbObject);
    });
}
```

Les exemples sont repris dans le code source disponibles à `https://cours.khi.be/sgbd3/labo/MongoDBExamples.java`.

4.4.1 Exemple : le titre du film 100

Nous allons utiliser la méthode `find` du shell `mongo` pour trouver le titre du film 100. Le critère de recherche est un document qualifié de "query document". Pour disposer de tous les films, on peut ne pas spécifier de critère ou alors indiquer un document vide. Remarquez l'utilisation de la méthode de curseur (cursor method) `limit` pour limiter le nombre de documents renvoyés par le curseur.

```
> db.movies.find().count()
174966
> db.movies.find().limit(1)
{ "_id" : 32, ..., "title" : "Cannon Fodder", ... }
```

Pour trouver le film dont le champ `_id` vaut 100, on utilise un objet JS contenant uniquement le champ `_id` avec la valeur 100.

```
> db.movies.find({"_id": 100})
{ "_id" : 100, ..., "title" : "Lock, Stock and Two Smoking Barrels", ...}
```

Pour projeter le résultat sur les champs `_id` et `titre`, on utilise un second objet JS avec la valeur 1 pour le champ `title`. Le champ `_id` est présent par défaut mais on peut l'exclure du résultat avec la valeur 0.

```
> db.movies.find({"_id": 100}, {"title": 1})
{ "_id" : 100, "title" : "Lock, Stock and Two Smoking Barrels" }
> db.movies.find({"_id": 100}, {"_id": 0, "title": 1})
{ "title" : "Lock, Stock and Two Smoking Barrels" }
```

Bien que cela ne soit pas apparent dans l'exemple ci-dessus, la méthode `find` renvoie un curseur. Par défaut, un curseur non assigné à une variable locale est itéré un maximum de 20 fois de manière à afficher les résultats à l'écran. Si le curseur est assigné à une variable alors la taille du batch est plus importante mais néanmoins tous les enregistrements ne sont pas retournés en une fois pour peu que ceux-ci excèdent une certaine taille.

```
> var c = db.movies.find({"_id": 100}, {"_id": 0, "title": 1})
> c.hasNext()
true
> c.next()
{ "title" : "Lock, Stock and Two Smoking Barrels" }
```

La méthode `findOne` quant à elle ne renvoie qu'un seul document et non un curseur. Si plusieurs documents satisfont le critère de recherche alors le premier selon un certain ordre est renvoyé.

```
> var d = db.movies.findOne({"_id": 100}, {"_id": 0, "title": 1})
> d
{ "title" : "Lock, Stock and Two Smoking Barrels" }
```

En Java, un document de la base de données est représenté par l'interface `com.mongodb.DBObject` qui peut être utilisée comme une map (`java.util.Map`). La classe concrète est `com.mongodb.BasicDBObject`.

```
DBObject document;
document = movies.findOne(queryDoc, projectionDoc);
displayDocument(document);
```

```
DBCollection movies = db.getCollection("movies");
DBObject queryDoc = new BasicDBObject();
```

```
queryDoc.put("_id", 100);
DBObject projectionDoc = new BasicDBObject();
projectionDoc.put("_id", 0);
projectionDoc.put("title", 1);
DBCursor dbCursor = movies.find(queryDoc, projectionDoc);
iterateOverCursor(dbCursor);
```

Ce qui donne :

```
*** Object 1937011546 ***
KEY: title, TYPE: java.lang.String, VALUE: Lock, Stock and Two Smoking Barrels
```

On peut aussi construire un `BasicDBObject` à l'aide de la classe `BasicDBObjectBuilder` qui implémente le pattern builder qui est expliqué dans le livre "Effective Java" ou sur Java Code Geeks[17] et qui permet de palier aux manques de Java en matière d'arguments nommés et d'arguments par défaut lors de la construction d'objets.

```
projectionDoc = BasicDBObjectBuilder.start()
        .add("_id", 0)
        .add("title", 1)
        .get();
```

Documentation :

- Méthode de collection `find`
  `http://docs.mongodb.org/manual/reference/method/db.collection.find/`
- Méthode de collection `findOne`
  `http://docs.mongodb.org/manual/reference/method/db.collection.findOne/`
- Méthode de curseur `limit`
  `http://docs.mongodb.org/manual/reference/method/cursor.limit/#cursor.limit`
- Projection
  `http://docs.mongodb.org/manual/tutorial/project-fields-from-query-results/#projection`

4.4.2 Exemple : les films et leur cote

La cote d'un film est exprimée à l'aide du champ `vote_average` et du champ `vote_count` qui permet de connaître le nombre de votes.

On reçoit un curseur pour accéder aux films dont le `vote_average` vaut 9.0. Notez que l'accès à la BD n'a lieu que lorsqu'on désire connaître le nombre de documents accessible avec ce curseur. La première exécution prend un certain temps étant donné qu'on doit accéder à chaque document pour voir s'il convient. En effet nous ne disposons pas d'index sur le champ `vote_average`.

```
> var c = db.movies.find({"vote_average": 9.0}, {"title": 1})
> c.count()
1877
> c
{ "_id" : 21, "title" : "The Endless Summer" }
...
Type "it" for more
> it
```

[17] `http://www.javacodegeeks.com/2013/01/the-builder-pattern-in-practice.html`

```
{ "_id" : 2574, "title" : "The Lovers" }
...
```

Si on désire trouver tous les films ayant une cote supérieure à 9.0 ou dans une autre requête une cote comprise entre 9.0 et 9.2 exclu, on utilisera des sélecteurs de comparaison (comparison query selectors) dans un document jouant le rôle de valeur de champ (inner document), c'est-à-dire de la forme `"name": document` où `document` est qualifié de document interne (inner document) car il est dans le champ `name`.

L'évaluation du curseur dans le shell avec `limit` (ou sans) provoque la "consommation" suivie de la fermeture de celui-ci. C'est pourquoi il est nécessaire de faire une nouvelle requête `find` pour pouvoir réaliser le second test.

```
> var c = db.movies.find({"vote_average": {"$gte": 9.0}}, {"title": 1})
> c.count()
6648
> var c = db.movies.find({"vote_average": {"$gte": 9.0, "$lt": 9.2}}, {"title": 1})
> c.count()
1900
> c.limit(1)
{ "_id" : 21, "title" : "The Endless Summer" }
> c.forEach(function(d) { print(d.title) })
...
> var c = db.movies.find({"vote_average": {"$gte": 9.0, "$lt": 9.2}}, {"title": 1})
> c.limit(1).forEach(function(d) { print(d.title) })
The Endless Summer
> c.limit(1)
2014-06-24T05:35:06.747+0200 query already executed at src/mongo/shell/query.js:75
```

```
queryDoc = QueryBuilder.start("vote_average")
           .greaterThanEquals(9.0).lessThan(9.2).get();
cursor = movies.find(queryDoc, new BasicDBObject("title", 1).append("vote_average", 1));
cursor.limit(3);
iterateOverCursor(cursor);
cursor.close();
```

Documentation :

- Query and Projection Operators
  `http://docs.mongodb.org/manual/reference/operator/query/`

4.4.3 Exemple : tous les films avec un titre donné

Nous pouvons rechercher tous les films ayant un titre donné ou tous les films dont le titre "matche" une expression régulière[18] à la Perl (PCRE). Ce mécanisme est puissant mais assez lent. Il est également possible de faire des recherches sur des mots dans du texte à l'aide d'un index mais ce n'est pas aussi puissant que les expressions régulières. Cfr. le code en commentaire dans le programme Java.

```
> db.movies.findOne({"title": "The Fellowship of the Ring"}, {"title": 1})
null
> db.movies.findOne({"title": /^.*Fellowship.*Ring$/}, {"title": 1})
{
  "_id" : 120,
```

[18] `http://www.regular-expressions.info/`

```
  "title" : "The Lord of the Rings: The Fellowship of the Ring"
}
> db.movies.find({"title": /^.*Fellowship.*Ring$/}, {"title": 1})
{ "_id" : 120, "title" : "The Lord of the Rings: The Fellowship of the Ring" }
```

```
queryDoc = new BasicDBObject("title", Pattern.compile("^.*Fellowship.*Ring$"));
cursor = movies.find(queryDoc, projectionDoc);
iterateOverCursor(cursor);
cursor.close();
```

Documentation :

- Opérateur `$regex`
  `http://docs.mongodb.org/manual/reference/operator/query/regex/`

4.4.4 Exemple : avec ou sans index

Si l'on désire accélérer les recherches, on peut utiliser des index. L'exemple que nous avons choisi est basé sur une recherche du `vote_average` compris entre 9.0 et 9.2 exclu. Lors de notre test, sans index le temps est de l'ordre de 500 ms et avec l'index il est de 5 ms.

Pour créer un index, on peut utiliser le shell ou Java.

```
> db.movies.ensureIndex( { "vote_average": 1 } )
{
  "createdCollectionAutomatically" : false,
  "numIndexesBefore" : 1,
  "numIndexesAfter" : 2,
  "ok" : 1
}
> db.movies.getIndexes()
[
  {
    "v" : 1,
    "key" : {
      "_id" : 1
    },
    "name" : "_id_",
    "ns" : "movies.movies"
  },
  {
    "v" : 1,
    "key" : {
      "vote_average" : 1
    },
    "name" : "vote_average_1",
    "ns" : "movies.movies"
  }
]
```

```
movies.createIndex(new BasicDBObject("vote_average", 1));
```

Pour supprimer l'index, on indique son nom ou ses clés à l'aide d'un document.

```
> db.movies.dropIndex( { "vote_average": 1 } )
{ "nIndexesWas" : 2, "ok" : 1 }
> db.movies.getIndexes()
[
  {
    "v" : 1,
    "key" : {
      "_id" : 1
    },
    "name" : "_id_",
    "ns" : "movies.movies"
  }
]
```

```
movies.dropIndex(new BasicDBObject("vote_average", 1));
```

Documentation :

- Introduction aux index
  `http://docs.mongodb.org/manual/core/indexes-introduction/`
- Création d'un index
  `http://docs.mongodb.org/manual/tutorial/create-an-index/`

5 Informations techniques

5.1 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` 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 1", on concatènera `http://cf2.imgobject.com/t/p/`, `w185` et `/uU9R1byS3USozpzWJ5o` `.jpg` pour avoir
`http://cf2.imgobject.com/t/p/w185/uU9R1byS3USozpzWJ5oz7YAkXyk.jpg`

Listing 1: Document


```
{"images":{"base_url":"http://cf2.imgobject.com/t/p/","poster_sizes":["w92","w154","w185
","w342","w500","original"],"backdrop_sizes":["w300","w780","w1280","original"],"
profile_sizes":["w45","w185","h632","original"],"logo_sizes":["w45","w92","w154","w185","
w300","w500","original"]},"change_keys":["adult","also_known_as","alternative_titles","
biography","birthday","budget","cast","character_names","crew","deathday","general","
genres","homepage","images","imdb_id","name","original_title","overview","plot_keywords","
production_companies","production_countries","releases","revenue","runtime","
spoken_languages","status","tagline","title","trailers","translations"]}
```


5.2 La pagination

La navigation se fera au moyen de boutons ou de liens. La requête permettant de récupérer ces listes sera 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 les dix à afficher. Différentes techniques PL/SQL existent à ce sujet, il est demandé de les exploiter pour certaines fonctionnalités.

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"[19] par Tom Kyte. Vous n'êtes pas obligé d'utiliser le commentaire destiné à l'optimizer Oracle dont la forme est `/*+ FIRST_ROWS(n)*/`.

Une autre technique passe par la commande SQL OVER[20] qui permet de définir une fenêtre ou zone de données parmi les résultats d'une requête et ne retourner que les tuples se trouvant délimités par les bornes supérieures et inférieures de la fenêtre.

5.3 Distributions

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.

5.3.1 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.

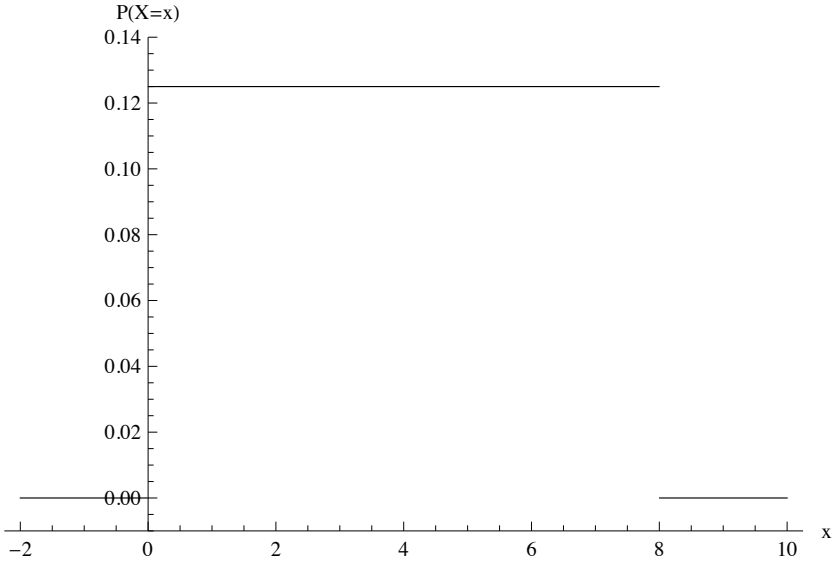
Le graphique de la figure 2 représente une distribution uniforme de nombres réels entre 0 et 8. On remarque que pour les nombres $x < 0$ et $x > 8$, la probabilité est nulle, c'est-à-dire que $P(X = x) = 0$. Mais quelle est la probabilité d'obtenir un nombre entre 0 et 8 ? On sait que c'est la même quel que soit le nombre choisi puisque

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

[20] Une présentation en français de cette commande peut être trouvée sur la page `http://lalystar.developpez.com/fonctionsAnalytiques/`

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.

FIG. 2: Distribution uniforme



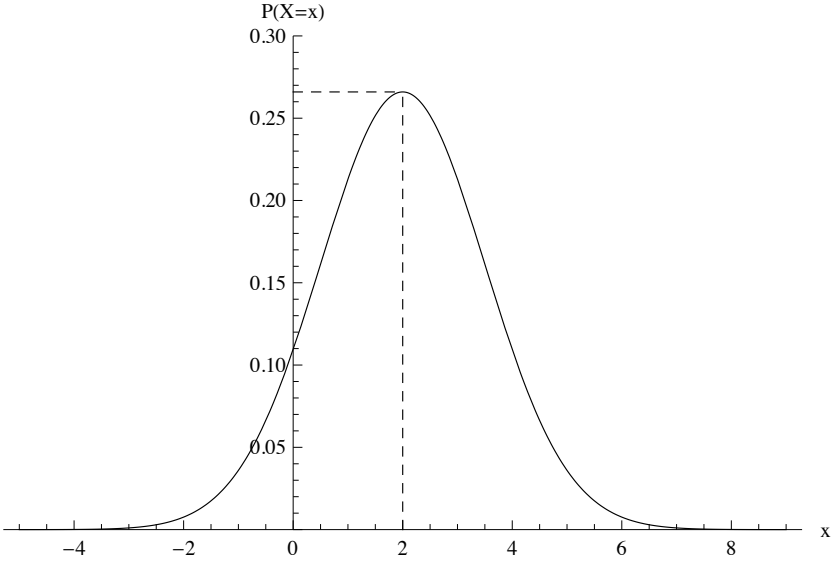
5.3.2 Distribution normale

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[21].

[21] 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 : $\int_{x=0.5}^{3.5} P(X = x)\, dx = 0.682689$.

FIG. 3: Distribution normale



5.4 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".

La fonction Java `java.util.Random.nextGaussian()` utilise une variante de la méthode précédente pour générer des `double` selon une distribution normale standard, c'est-à-dire dont la moyenne μ vaut 0 et l'écart-type σ vaut 1. On peut passer d'une variable X suivant une distribution normale générale à une variable Z suivant une distribution normale standard en utilisant le changement de variable $Z = (X - \mu)/\sigma$. On peut donc facilement faire l'opération inverse pour obtenir X à partir de Z.

6 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 7, page 24.

6.1 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 vis à vis des techniques attendues, il vous est vivement recommandé de prendre conseil au près de l'un de des professeurs concernés par le projet.

6.2 Gestion des erreurs

Chaque langage dispose de ses mécanismes internes de gestion d'erreurs. Cependant quand plusieurs systèmes interagissent les uns avec les autres, certaines précautions doivent être prise. Le mode console n'étant pas forcément disponibles des dispositifs de sauvegarde des messages d'erreurs doivent être mis en place.

Vous devez mettre en place un système de log de toutes les erreurs et exceptions générée dans et par le SGBD dans les différentes bases de données que vous manipulez. Une table spécifique doit être créée sur chaque site. Elle contient au minimum le timestamp précis de l'erreur/exception, l'objet où elle a été émise (package / procédure / déclencheur / job / etc.), le numéro de l'erreur (prédéfini oracle ou personnel) et le message d'erreur correspondant. Pour chaque tuple inséré, un ticket (numéro unique) sera créé et retourné, s'il y a lieu, à l'utilisateur final.

Dans cette optique, la table de log contiendra des messages d'erreurs précis permettant à un administrateur de la BD d'identifier le problème avec exactitude. Tandis que l'utilisateur face à l'écran applicatif recevra un message adapté ainsi qu'un numéro de ticket à transmettre à un administrateur.

Deux versions de log devront être prévue : l'enregistrement simple (mode débogage) et l'enregistrement avec relance d'exception et renvoi de ticket (mode erreur). On peut par exemple considérer qu'un tuple de log dont le code d'erreur vaut 0 est en réalité une simple message informationnel et non un message d'erreur.

Attention, l'annulation d'une transaction ne doit pas annuler l'enregistrement de l'erreur.

6.3 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.

Vous pouvez lire le court article[22] intitulé "What ORMs have taught me: just learn SQL" si vous désirez un complément d'information.

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.

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

6.4 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[23].
- Un autre choix est d'utiliser ODP.NET[24] 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. Il est arrivé que des étudiants aient choisi d'utiliser divers langages comme Python, Perl ou PHP. Cependant 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.

6.5 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.

6.6 Pré-requis aux évaluations

Tout test de fonctionnalité nécessite des tuples en suffisance pour permettre l'évaluation du bon comportement de la fonction. Nous attendons de votre part que ces données soient présentes avant les évaluations. Voire mieux : que des jeux de test et des scénarios de test aient été préparés pour faciliter et accélérer l'évaluation.

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

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

7 Réalisations attendues

7.1 Partie I - Modélisation des données

7.1.1 Schémas conceptuels - MCD

Les schémas conceptuels ne sont pas demandés et ne sont pas fournis.

7.1.2 Schémas relationnels - MRD

Les schémas relationnels ne sont pas demandés mais sont fournis. Ils se trouvent respectivement aux figures 4 pour CB (page 25) et 5 pour CC1 (page 26).

Quelques remarques relatives aux schémas sont nécessaires :

- L'attribut `nb_copies` dans la table `movies` de CB correspond au nombre de copies présente dans tout le système informatique (CB + CCx) pour un film donné. Cela permet de ne pas devoir le recalculer à chaque fois à l'aide de DB links ce qui est plus efficace et cela simplifie également l'écriture du code.
- La clé primaire des tables `copies` est composée de deux attributs : `num_copy` et `movie`. Il s'agit donc d'une entité faible en terme du modèle entité-association et l'attribut `num_copy` représente le numéro de copie relatif au film, pas un numéro de séquence absolu et unique pour toutes les copies.
- Les identifiants des tables `movies` et `artists` sont ceux de TMDb.
- L'attribut `where` de la table `logs` correspond au nom de la procédure ou fonction où a eu lieu l'erreur ou s'il n'y a pas d'erreur d'où provient le message informatif.
- La table `projection_archives` est mise à jour lors de la programmation des séances et est utilisée pour calculer la popularité et la pérennité des films.

7.1.3 Scripts SQL de création des schémas - CreaSql

Les scripts SQL de création des tables ne sont pas demandés et sont fournis. Néanmoins, toutes les contraintes applicatives qui ne peuvent pas être définies au moyen du LDD et que vous souhaitez utiliser doivent être vérifiées à l'aide de déclencheurs. Vous êtes responsables de la qualité de votre schéma.

Fig. 4: Schéma relationnel pour CB

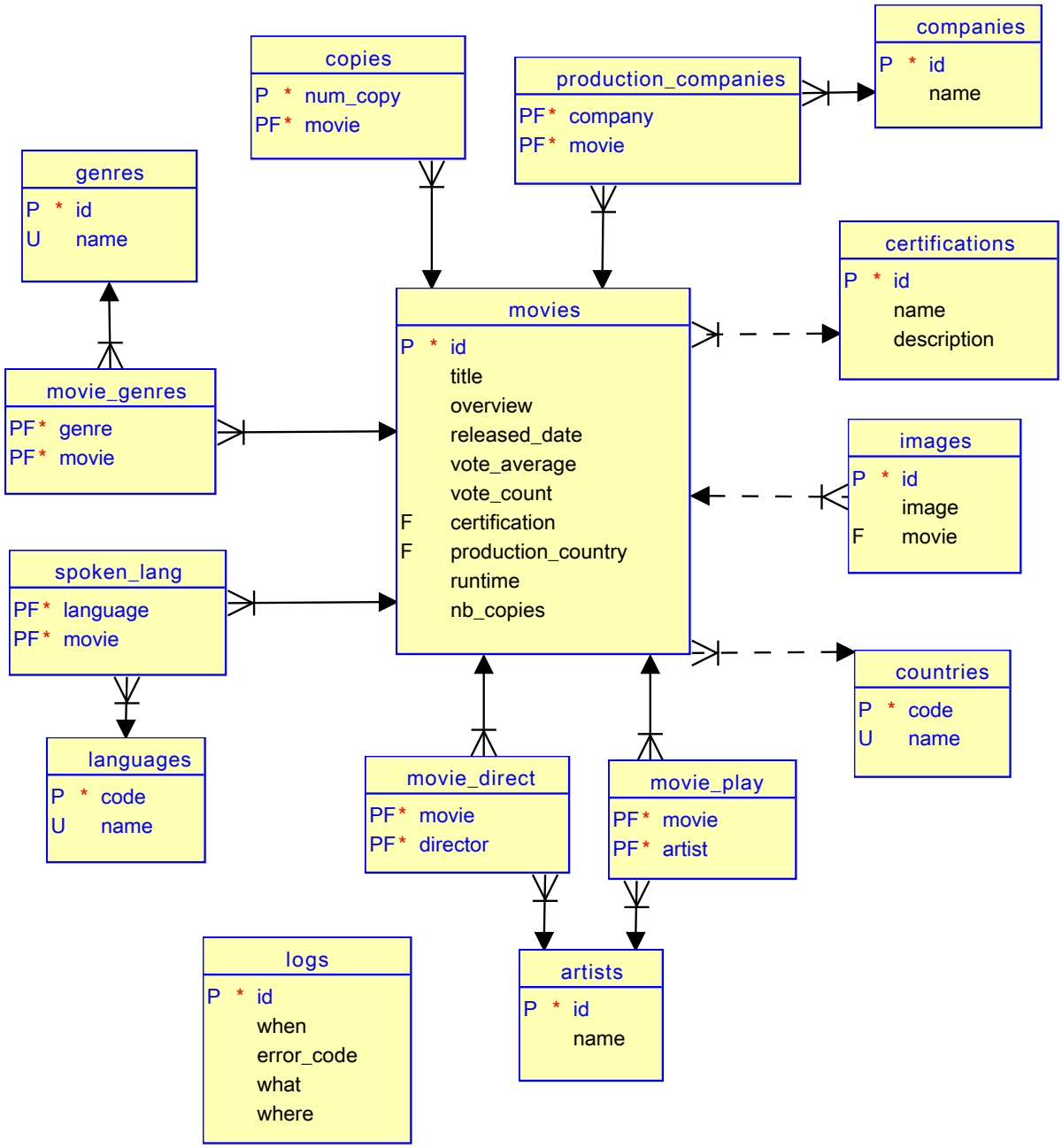
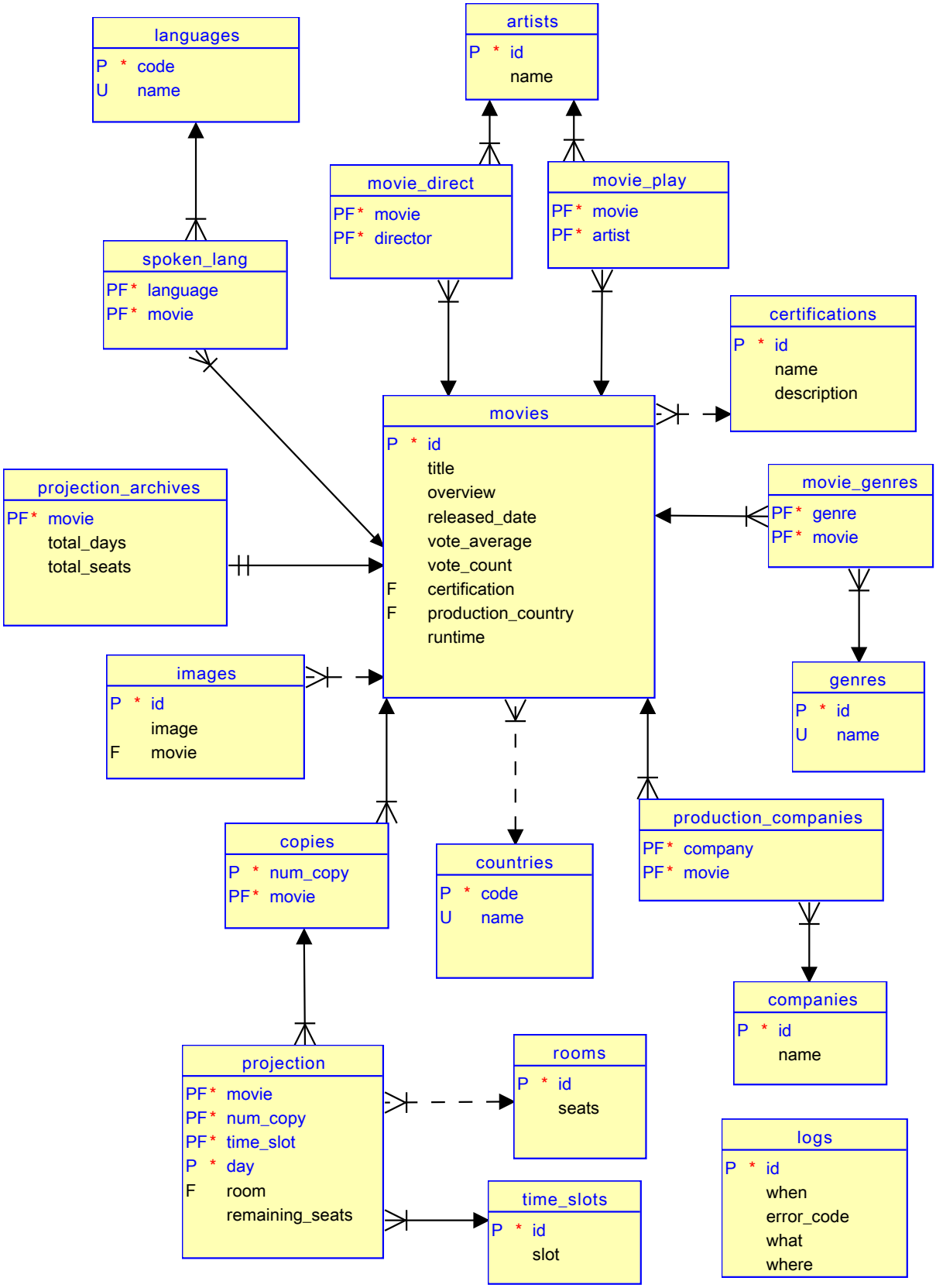


Fig. 5: Schéma relationnel pour CC1



7.2 Partie II - Alimentations

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 différente. Il faut donc les interroger mais aussi les transformer s'il y a lieu pour les intégrer dans CB. L'alimentation par l'entremise de CI étant assez conséquente, celle-ci a été découpée en deux étapes. Ces deux étapes ont pour objectif de permettre l'alimentation régulière de CB en films et copies (que l'on appelle aussi supports) à partir d'informations se trouvant dans CI selon des critères de recherche fournis par l'utilisateur.

1. RechCI effectue les recherches multi-critères sur les documents de CI et fournit la liste des films correspondants.
2. AlimCB transfère les films obtenus vers CB.

7.2.1 Recherche des films sur CI - RechCI

A partir de critères prédéfinis sélectionnés ou non à l'aide de votre programme (GUI ou en ligne de commande avec options à l'aide de `argparse4j`[25]), la liste de films correspondant doit apparaître. *Un ou plusieurs* de ces films doivent alors pouvoir être sélectionnés.

On désire disposer des critères d'interrogation 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 à une date au format `DD/MM/YYYY` ou dans une fourchette donnée, un ou plusieurs genres, la certification MPAA, les films dont le rating (`vote_average`) est inférieur, supérieur ou égal à un rating donné ou compris dans une fourchette donnée, les films dont le nombre de votes (`vote_count`) est inférieur, supérieur ou égal à un nombre de votes donné ou compris dans une fourchette donnée. Tous ces critères peuvent être combinés.

Il doit être possible de voir le détail d'un film si on le sélectionne.

7.2.2 Alimentation de la centrale CB - AlimCB

Au moment de commander, une quantité aléatoire de copies, basée sur une distribution normale (cfr. section 5.3, page 19) sera générée séparément pour chaque film. 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.

Tous les films déjà présents dans CB ne devront pas être dupliqués ni re-transférés mais le stock des copies devra être mis à jour.

En ce qui concerne les images liées à un film, nous vous demandons de reprendre l'image de type `poster` de taille `w185`. Ces images seront téléchargées par HTTP et stockées sous forme de BLOBs dans CB.

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

Pensez également à utiliser des séquences Oracle lorsque cela se justifie.

Techniques attendues : MongoDB, un langage au choix (cfr. 6.4, page 23), SQL, séquences, transferts de BLOB.

[25] `http://argparse4j.sourceforge.net/`

7.2.3 Alimentation de CC1 - AlimCC1

CC1 reçoit les nouveaux stocks de copies de CB de deux manières différentes.

1. Chaque fois que des copies arrivent sur CB à partir de CI, un nombre aléatoire de celles-ci est livré automatiquement dans CC1. Ce transfert se fera au moyen d'un déclencheur.
2. 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.

Dans les deux cas, le nombre aléatoire se base sur une distribution uniforme allant de 0 à $\lfloor \frac{n}{2} \rfloor$ où n est le nombre de copies disponibles sur CB (cfr. section 5.3, page 19) et $\lfloor x \rfloor$ est l'entier le plus grand inférieur ou égal à x.

Il est imposé qu'une copie ne puisse se trouver en même temps à la centrale CB et dans un complexe : il faut donc la supprimer de CB quand elle est transférée vers le complexe et vice et versa. De même, si le nombre de copies à transférer est 0, la fiche signalétique du film concerné ne sera pas transférée.

Techniques attendues : PL/SQL, réplication, DBlink, distribution uniforme, transferts de BLOB.

8 Règles d'évaluation

Cette section contient les règles d'évaluation du cours de "Structure de l'information et bases de données" de la section M18 (HEPL - ISIL) pour l'année académique 2014-2015. Les professeurs concernés sont Ludovic Kuty et Laurence Herbiet.

Tout le code (scripts SQL, code source, …) doit être fourni au professeur pour le jour précédent l'évaluation sous format électronique soit à l'adresse ludovic.kuty@hepl.be soit par tout autre moyen dans l'hypothèse où la taille de l'archive est trop grande que pour être attachée à un email. On vous demande de fournir une archive de type rar, zip ou tgz.

8.1 Première session

8.1.1 Première partie

- L'évaluation de laboratoire de la première partie de SGBD est une évaluation continue. Elle est évaluée pendant le deuxième quadrimestre.
- Il n'y a pas d'évaluation théorique pour la première partie.
- La cote de laboratoire représente 40% de la cote finale de laboratoire de SGBD.
- Le travail de la première partie s'effectue seul ou en équipe de deux étudiants.
- Chaque étudiant d'une équipe peut être interrogé sur l'entièreté du travail effectué par l'équipe.

8.1.2 Deuxième partie

- L'évaluation de laboratoire de la deuxième partie de SGBD est une évaluation continue. Ce qui n'aura pas été évalué pendant le second quadrimestre sera évalué pendant la session de juin.
- L'évaluation théorique pour la deuxième partie est une évaluation continue.
- La cote de laboratoire représente 60% de la cote finale de laboratoire de SGBD.
- La cote de théorie représente 30% de la cote finale de SGBD. La cote de laboratoire représente donc 70% de la cote finale de SGBD.
- Le travail de la deuxième partie s'effectue seul ou en équipe de deux étudiants.

- Chaque étudiant d'une équipe peut être interrogé sur l'entièreté du travail effectué par l'équipe.

8.2 Deuxième session

L'examen de seconde session est pondéré à 70% (partie 1 : 30%, partie 2 : 40%) pour le travail de laboratoire complet et 30% pour la théorie. Il s'agit de compléter les fonctionnalités qui n'ont pu être terminées pour les évaluations des parties 1 et 2. Si la cote de la première partie est supérieure ou égale à 10, vous pouvez la garder. Dans le cas contraire, vous êtes obligés de représenter le travail et la cote est remise à zéro. La cote de première session n'est pas conservée si celle de 2ème session est moins bonne.

8.3 Prolongation de session

Pour les étudiants qui se retrouveraient dans la situation d'une prolongation de session pour l'année 2014-2015, l'épreuve portera sur le même énoncé. L'examen peut être présenté dès qu'un rendez vous est pris avec le professeur. L'évaluation sera reprise de zéro et les deux parties doivent être représentées. Ni la cote de première session ni celle de 2ème session ne sont conservées si celle de prolongation session est moins bonne.