Intégration des systèmes d'information
Médiateur - Wrappers

Compléments

TP : Amal Htait

TABLE DES MATIÈRES

MÉDIATEUR - WRAPPERS

Ce document complète les informations contenues dans la feuille de TP distribuée lors de la première séance du cours ISI. Et il formalise aussi les échanges que nous avons eus jusqu'à présent pour la construction d'un système médiateur-wrappers.
Nous vous proposons de nouvelles sources de données ainsi qu'un cannevas de programme afin de faciliter la programmation.
Chaque groupe/élève devra choisir un scenario (Section 2), ou alors proposer le sien. Les scenarii servent de base et peuvent être modifiés ou même totalement inventés. Ce document regroupe les principales informations nécessaires au projet.

Example : les comparateurs en ligne : https ://www.lesfurets.com/qui-sommes-nous

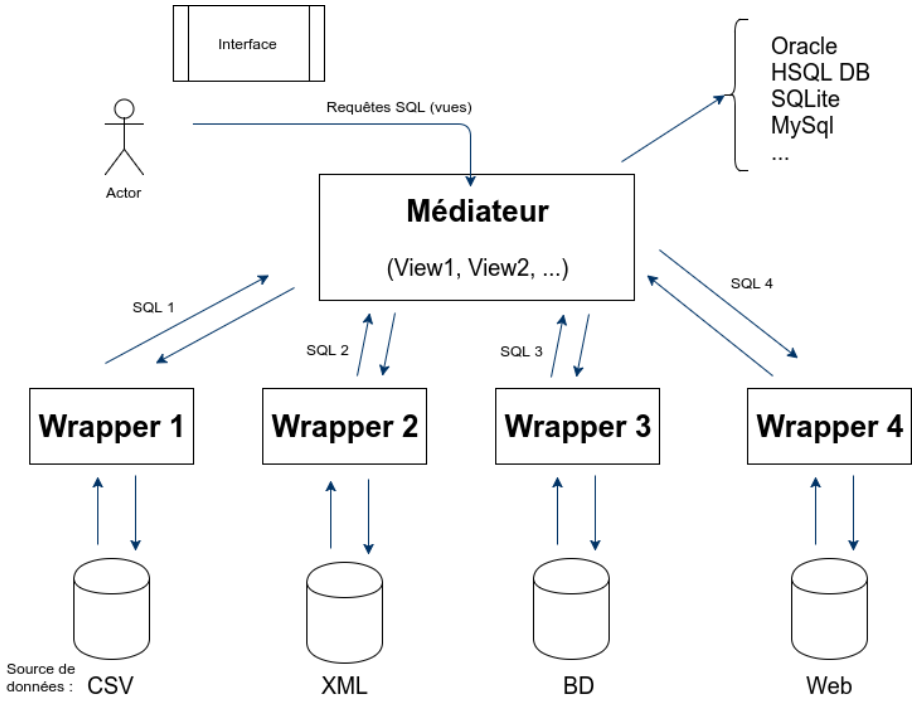


FIGURE 1: Illustration d'un système Médiateur-Wrappers

Vous aurez besoin de crée un compte sur https://www.kaggle.com pour accèder certains données.

La santé et la pollution

L'utilisateur cherche les villes les plus appropriées de point de vue santé et pollution aux États-unis. Ci-dessous plusieurs sources de données :

1. **Big Cities Health / La santé des grandes villes** :
   Cet ensemble de données illustre l'état de santé de 26 des villes les plus grandes et les plus urbaines aux États-unis.
   https://query.data.world/s/0xKZ6O52ZiSCQK3aZTwX1yWfY52_0T

2. **Behavioral Risk Factor Data : Health-Related Quality of Life (HRQOL) 1993-2010 / Données sur les facteurs de risque comportementaux : Qualité de vie liée à la santé (QVLS) 1993-2010** : Les données proviennent du Système de surveillance du facteur de risque comportemental (BRFSS, adultes non institutionnalisés, 18 ans ou plus). La surveillance de la QVLS est utilisée pour identifier les besoins de santé de la population non satisfaits, y compris la reconnaissance des tendances, des disparités et des déterminants de la santé dans la population. Les données de surveillance de la QVLS peuvent être utilisées pour éclairer la prise de décision et l'élaboration de programmes et de politiques. Pour s'assurer que la population bénéficie des programmes de santé publique, les données de surveillance de la QVLS peuvent être utilisées pour l'évaluation du programme.
   https://query.data.world/s/0xKZ6O52ZiSCQK3aZTwX1yWfY52_0T

3. **U.S. Pollution Data / Pollution aux États**
   : Cet ensemble de données traite de la pollution aux États - Unis. La pollution aux États - Unis a été bien documentée par l 'EPA américaine mais il est difficile de télécharger toutes les données et de les organiser dans un format qui intéresse les scientifiques de données. J'ai donc rassemblé quatre polluants majeurs (dioxyde d'azote, dioxyde de soufre, monoxyde de carbone et ozone) pour chaque jour de 2000 à 2016 et les place proprement dans un fichier csv.
   https://www.kaggle.com/sogun3/uspollution/data

4. **Toxic Release Inventory, US EPA data on release of toxic chemicals for 1987-2016 / Données de l'US EPA sur la libération de produits chimiques toxiques pour 1987-2016** :
   https://www.kaggle.com/epa/toxic-release-inventory/data

5. **Hospital General Information, General information & quality ratings for almost all US hospitals / Informations générales et notes de qualité pour presque tous les hôpitaux américains** :
   https://www.kaggle.com/cms/hospital-general-information/data

Sécurité des compagnies aériennes vis à vis la météo

L'utilisateur cherche un vol sécurisé, avec minimum risque d'accident causé par le météo aux États-unis.

1. **U.S. weather history visualization / Visualisation de l'historique météorologique américain**
   https://github.com/fivethirtyeight/data/tree/master/us-weather-history
2. **Airline safety data / Données de sécurité des compagnies aériennes**
   https://github.com/fivethirtyeight/data/tree/master/airline-safety
3. **Airplane Crashes Since 1908 / Les accidents d'avion depuis 1908**
   https://www.kaggle.com/saurograndi/airplane-crashes-since-1908/data
4. **2015 Flight Delays and Cancellations / Retards et annulations de vols en 2015**
   https://www.kaggle.com/usdot/flight-delays
5. **Airline Database, A database of over 5000 airlines / Base de données aérienne**
   https://www.kaggle.com/open-flights/airline-database/data
6. **Airline Fleets, The top 100+ airlines and their fleet specifics / Les 100 meilleures compagnies aériennes et leurs spécificités de flotte**
   https://www.kaggle.com/traceyvanp/airlinefleet/data

Crimes et Terrorisme

L'utilisateur cherche les villes les plus sécurisées, avec minimum risque de crime et terrorisme.

1. **Base de données mondiale sur le terrorisme** :
   La base de données mondiale sur le terrorisme (GTD) est une base de données open source contenant des informations sur les attentats terroristes dans le monde de 1970 à 2016.
   https://www.kaggle.com/START-UMD/gtd/data
2. **Crimes de Haine aux États-unis** :
   De https://fivethirtyeight.com/features/higher-rates-of-hate-crimes-are-tied-to-income-inequality/, et les données à :
   https://github.com/fivethirtyeight/data/tree/master/hate-crimes.
3. **Victime par la police (police-killings) aux États-unis** :
   https://github.com/fivethirtyeight/data/tree/master/police-killings
4. **Victime de la police (police-deaths) aux États-unis** :
   https://github.com/fivethirtyeight/data/tree/master/police-deaths

5. **Crime en 2015-2016 aux États-unis** :
   https://github.com/fivethirtyeight/data/tree/master/murder_2016

6. **Plusieurs données à propos des crimes au monde peuvent être récupérés de** : Crimes

Sport

L'investisseur veut faire le rapprochement entre les installations sportives mises à disposition et la valeur sportive de chaque ville en France. Pour cela vous devrez croiser les classements sportifs du sport de votre choix : football, tennis, handball, etc. avec les installations des villes concernées.

1. **Classements** : http://www.lequipe.fr/Football/ligue-2-classement.html (ou autres ligues fran-çaises et autre sport)

2. **Données d'équipements** : http://www.data.gouv.fr/fr/datasets/recensement-des-equipements-sportifs-espaces-et-sites-de-pratiques/

3. **Sports olympiques et médailles, 1896-2014** : Dont les données des villes en France qui ont accueilli les jeux Olympics. https://www.kaggle.com/the-guardian/olympic-games/data

Tourisme de cinéphile

L'utilisateur souhaite visiter les lieux de tournages de films de Paris en toute tranquilité. Il n'a pas beaucoup de temps et souhaite donc choisir le lieu avec le plus de concentration de cinéma et lieux de tournage. Sans le sou, il souhaite également pouvoir y circuler en velib et utiliser les hotspots pour streamer son vlog.

1. **Tournages en exterieur à Paris** : https://www.data.gouv.fr/fr/datasets/lieux-de-tournage-de-films-long-metrage-prs/

2. **Wifi gratuits** : https://www.data.gouv.fr/fr/datasets/liste-des-sites-des-hotspots-paris-wifi-prs/

3. **Velib** : https://www.data.gouv.fr/fr/datasets/velib-paris-et-communes-limitrophes-idf/

4. **salles et fréquentation** : https://www.data.gouv.fr/fr/datasets/geographie-du-cinema-equipement-et-frequentation/

WRAPPER CSV-SQL

Le but est de prendre un fichier CSV source et de générer dynamiquement une table qui contiendra les informations du fichier CSV structurées en tuples (en Utilisant le langage de programmation Java, Python, etc). Cette table sera interrogée par la sous-requête SQL envoyée depuis le médiateur (Moteur SQLite, HSQLDB, MySql, ORACLE, HSQL, POSTGRES, etc.).
Un example d'un outil csv-sql en ligne :
http://www.convertcsv.com/csv-to-sql.htm

Vous pouvez utiliser le langage de programmation que vous maîtrisez, soit : Java, Python, C, C++, perl, etc. Et la base de données que vous préferez (suivant sa compatibilité avec le language de programmation et sa disponibilité), soit : SQLite, HSQLDB, MySql, ORACLE, HSQL, POSTGRES, etc.

Ci-dessous un example d'installation de Python3 :
$ sudo apt-get update
$ sudo apt-get install python3.6

Excecution du script python en ligne de command :
$ python3 fileName.py

Liens utiles :
http://docs.python-guide.org/en/latest/starting/install3/linux/
http://www.formation-django.fr/python/comment-installer-python.html

Wrapper rapide

1. Lire un fichier ligne par ligne :

   [–] En Python, la fonction *read_File* retourne la liste des lignes du fichier en paramètre sous forme d'un array :

```
1 def read_File(fileName):
2   lignes = ""
3   try:
4     f = open(fileName, 'r')
5     lignes  = f.readlines()
6     f.close()
7   except FileNotFoundError:
8     print("Wrong file or file path")
9   return lignes
```

   NB : le paramètre *fileName* est l'emplacement et le nom du document.

   [–] En Java :

```
1 public static String readFile( String path )
```

```
2                   throws IOException {
3     byte[] encoded = Files.readAllBytes(Paths.get(path));
4     return new String(encoded, StandardCharsets.UTF_8) ;
5 }
```

2. Découper une chaine de caractères suivant un séparateur.

   [–] En Python :

```
1    myString.split(",")
```

   [–] En Java :

```
1    myString.split(",");
```

3. Prendre les informations et construire la requête SQL

   [–] En Python :

```
1 sqlQuery = ""
2 sqlQuery += "CREATE␣TABLE␣table_name"
3 sqlQuery += "␣(mes␣informations␣issues␣du␣CSV)"
4 sqlQuery += "\n"
5 sqlQuery += "INSERT␣INTO␣table_name"
6 sqlQuery += "␣VALUES␣(mes␣valeurs␣issues␣du␣CSV)"
```

   [–] En Java :

```
1  StringBuilder sb = new StringBuilder();
2  sb.append("CREATE␣TABLE␣table_name");
3  sb.append("mes␣informations␣issues␣du␣CSV");
4  sb.append[("\n")
5  sb.append("INSERT␣INTO␣table_name");
6  sb.append("␣VALUES␣(mes␣valeurs␣issues␣du␣CSV)");
7  //retourner la chaine de caractere finale.
8  return sb.toString();
9  // Ensuite l'envoyer␣en␣requete␣a␣la␣base␣de␣donnees
10 ␣␣␣␣␣␣␣␣(voir␣section␣suivante)
```

4. **NB : Pour lire le contenu d'une page web**, la fonction *read_page* retourne le contenu d'une page web, dont le URL est en paramètre, sous forme d'une chaine de charactères (string) de format HTML. Mais cela nécessite ensuite l'utilisation du librairie BeautifulSoup pour extraire les données du HTML.

```
1 import urllib.request
2 def read_page(link):
3   try:
4     with urllib.request.urlopen(link) as response:
5         html = response.read()
```

```
6     except urllib.error.URLError as e:
7       print(e.reason)
8     return html
```

En plus, il existe des outils pour l'extraction des données des pages web (web extraction) que vous pouvez utiliser :
https://www.capterra.com/data-extraction-software/

Wrapper modulaire

Pour aller plus loin

Bien entendu le mieux serait d'avoir un wrapper adaptatif modulaire. Pour cela :

- Créer une classe Wrapper
- Rendre son instanciation adaptative aux formats et nombre de colonnes

Enfin, étape ultime, ne pas oublier le caractère dynamique du traitement de requêtes : le wrapper n'est sollicité que si une source de données est concernée par une requête, c'est à dire seulement si la vue qui utilise cette source a elle même été sollicitée par une requête utilisateur.

Le code et les instructions ci-dessous correspondent à SQLite, BD HSQLDB, mais le TP pourra être développé sur n'importe quelle base de données relationnelle MySql, ORACLE, POSTGRES, etc.).

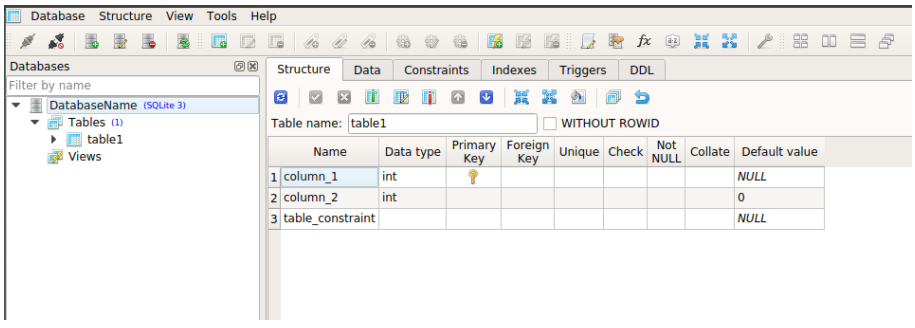
HSQLDB

- Télécharger HSQL (c'est juste un jar à mettre en dépendance à votre projet) :
  http://hsqldb.org/
- En utilisant le wrapper csv2sql, créer vos différentes bases de données à l'aide du code en Java présent ici par Gael Guibon :
  https://github.com/gguibon/teaching-isi-projet.
- Quelques sources d'aide :
  https://www.tutorialspoint.com/jdbc/index.htm http://baptiste-wicht.developpez.com/tutoriels/java/db
  A

SQLite (Linux)

- Télécharger :
  $ sudo apt-get update
  $ sudo apt-get install sqlite3 libsqlite3-dev
  OU
  $ wget http ://www.sqlite.org/2016/sqlite-amalgamation-3130000.zip
  $ tar xvfz sqlite-autoconf-3070603.tar.gz
  $ cd sqlite-autoconf-3070603
  $ ./configure
  $ make
  $ make install

  http://www.codebind.com/sqlite/how-to-install-sqlite-on/
- Pour une interface graphique vous pouvez utiliser :
  sqlitestudio https://sqlitestudio.pl/index.rvt



Autre Base de données

- **Mysql** :
  $ sudo apt-get install mysql-server

  Ensuite pour le gérer :
  $ sudo service mysql status
  $ sudo service mysql start
  $ sudo service mysql stop

  Et quelques informations à propos de l'utilisation MySQL avec Python :
  http://apprendre-python.com/page-database-data-base-donnees-query-sql-mysql-postgre-sqlite

- **Oracle** :
  Quelques informations à propos de l'utilisation Oracle avec Python :
  http://www.chicoree.fr/w/Acc%C3%A9der_%C3%A0_une_base_de_donn%C3%A9es_Oracle_%C3%
  Oracl DB link :
  https://docs.oracle.com/cd/B19306_01/server.102/b14200/statements_5005.htm

Quelques sources d'aide : Python et SQLite

- **Introduction Python** : https://eric.univ-lyon2.fr/ ricco/cours/slides/PA%20-%20intro%20python%20-%20bases%20algorithmiques.pdf
- **Python 3 Basics Tutorial** : https://www.gitbook.com/book/krother/python-3-basics-tutorial/details
- **Python 3 Tutorial** : https://www.tutorialspoint.com/python3/
- **Apprendre à programmer avec Python 3** : https://python.developpez.com/cours/apprendre-python3/?page=page_18
- **Base de données et Python** : http://apprendre-python.com/page-database-data-base-donnees-query-sql-mysql-postgre-sqlite
- **SQLite Tutorial** : http://www.sqlitetutorial.net/

Examples en HSQLDB

https://github.com/gguibon/teaching-isi-projet.

Examples en SQLite

– Par ligne de commande, la création d'une base de données en SQLite :
$ sqlite3 DatabaseName.db

NB : Une base de données SQLite est un fichier régulier. Il est créé dans votre répertoire actuel.

– En Python, connexion à la base de données, création et ajout de données dans une table :

```
1   import sqlite3
2
3   #connect to database:
4   connexion = sqlite3.connect('path/DatabaseName.db')
5
6   #create a cursor on the database:
7   curseur = connexion.cursor()
8
9   #create a table in the database:
10  curseur.execute("
11  CREATE TABLE IF NOT EXISTS celebrites
12  (nom TEXT, prenom TEXT, annee INTEGER)")
13
14  #Insert data into the table:
15
16  curseur.execute("INSERT INTO celebrites
17  (nom, prenom, annee) VALUES('Turing','Alan',1912)")
18
19  curseur.execute("INSERT INTO celebrites
20  (nom, prenom, annee) VALUES('Love','Ada',1815)")
21
22  #Commit the changer into the database:
23  connexion.commit()
24
25  #Close the database:
26  connexion.close()
```

NB : pour la connection, il faut préciser l'emplacement et le nom du document de base de données SQLite.

– En Python, la lecture de la base de données

```
1   connexion = sqlite3.connect("path/DatabaseName.db")
2   curseur   = connexion.cursor()
3
4   curseur.execute("SELECT * FROM celebrites")
5   resultat  = curseur.fetchall()
```

La liste résultat contient alors tous les enregistrements.

– En Python, modifier un enregistrement

```
1    connexion = sqlite3.connect("path/DatabaseName.db")
2    curseur   = connexion.cursor()
3
4    curseur.execute("UPDATE celebrites
5  ␣␣SET prenom='Alan Mathison'
6  ␣␣WHERE nom='Turing'")
7
8    connexion.commit()
```

– En Python, une requête de recherche ciblée :

```
1    curseur.execute("SELECT * FROM celebrites
2  ␣␣WHERE nom = 'Turing'")
3    resultat = list(curseur)
4    print(resultat)
```

La requête recherche et extrait seulement les lignes de la table dont l'entrée [nom] est 'Turing'. On transforme (transtype) le curseur en liste avant de l'afficher en tant que résultat.

– En Python, utiliser une variable dans une requête :

```
1    qui = "Shannon"
2    curseur.execute("SELECT * FROM celebrites
3  ␣␣WHERE nom = '" + qui + "'")
4    quand = 1515
5    curseur.execute("SELECT * FROM celebrites
6  ␣␣WHERE annee >= " + str(quand))
```

*** Les exemples en Python sont sur Ametice ***.

CRÉATION DE VUES INTÉRESSANTES

La création de vues est ce qui donnera une plus-value à votre projet.

En SQL, une vue est une table virtuelle basée sur le jeu de résultats d'une instruction SQL, et à travers laquelle une partie sélective des données d'une ou plusieurs tables peut être vu. Une vue contient des lignes et des colonnes, comme une vraie table, mais ne contiennent pas de données personnelles. Les champs d'une vue sont des champs d'une ou de plusieurs tables réelles de la base de données. Les vues sont utilisés pour restreindre l'accès à la base de données ou pour masquer la complexité des données.

En fonction des besoins de l'utilisateur, une vue différente devra être créée. Cette vue permet d'aggréger des informations spécifiques de plusieurs bases de données en une, et ainsi d'avoir un accès dédié et personnalisé aux données.

Voici le format d'une création de vue :

```
1    CREATE [TEMP] VIEW [IF NOT EXISTS] view_name
2    AS
3        select-statement;
```

View en SQLite : http://www.sqlitetutorial.net/sqlite-create-view/
View en HSQLDB : http://www.hsqldb.org/doc/guide/ch09.html#create_view-section