Université de Montpellier — Master Informatique

Traitement sémantique des données (HAI824I)

TP : Système OBDA avec Obi-wan

Dans ce TP, nous allons utiliser **Obi-wan** un système d'accès aux données basé sur une ontologie (OBDA), qui a été développé dans le cadre de la thèse de Maxime Buron. Obi-wan permet l'interrogation de sources de données hétérogènes (actuellement, PostgreSQL, SQLite, MongoDB, Jena TDB) via un niveau ontologique en RDFS. Les mappings permettent d'associer une requête sur une base de données source à un ensemble de triplets RDF utilisant le vocabulaire de l'ontologie. Obi-wan intègre un certain nombre d'outils existants, comme le médiateur Tatooine (`https://team.inria.fr/cedar/tatooine/`) capable de traiter des plans de requêtes portant sur plusieurs bases de données hétérogènes, et un algorithme de reformulation de requête avec des règles provenant du moteur de règles Graal (`https://graphik-team.github.io/graal/`). L'interrogation peut reposer soit sur la matérialisation totale (création d'une base de faits avec les mappings et saturation de cette base), soit sur la réécriture totale (reformulation de la requête avec l'ontologie et les règles d'implication de RDFS puis réécriture de la requête obtenue avec les mappings), soit sur des techniques mixtes. Dans ce TP, nous considérons la **réécriture totale** (appelée REW-CA dans Obi-wan).

Le but de ce TP est de développer un système OBDA simplifié utilisant une seule base de données extraite de IMDb (base de données très connue sur les films). Cette base de données est en SQLite, un système léger de gestion de BD relationnelles qui a l'avantage d'être intégré à Linux (et MacOS) et qui ne nécessite pas de service fonctionnant en arrière plan.

Vous aurez à construire une petite ontologie en RDFS, à mettre au point des mappings du schéma de IMDb vers cette ontologie, à concevoir quelques requêtes intéressantes, et à les exécuter avec Obi-wan en observant le processus de reformulation puis réécriture.

Pour en savoir plus sur les techniques implémentées par Obi-wan :
Démo à la conférence VLDB : `https://hal.inria.fr/hal-02921434/file/main.pdf`
Thèse : `https://hal.inria.fr/tel-03107689/`

Pour commencer : Téléchargez et décompressez le fichier TP-OBIWAN.zip depuis moodle. Vous obtenez un dossier dont nous allons examiner le contenu.

1 Structure d'un projet Obi-wan

Ouvrez le sous-dossier `obda-systems/example` qui donne un exemple de projet obi-wan.

De façon générale, un projet Obi-wan contient les 5 fichiers suivants (en italiques, des noms indicatifs que vous pouvez changer, seule l'extension du fichier est importante) :

- *`file-name`*`.properties` : le fichier qui pilote l'application (ici obi-wan.properties)
- *`ontology-name`*`.nt` : l'ontologie RDFS au format N-triples (ici ontology.nt)
- *`mappings`*`.json` : les mappings au format JSON (ici ris.json)
- *`queries`*`.txt` : des requêtes sous la forme de requêtes conjonctives avec le prédicat triple (ici queries.txt)
- un fichier `querysession.properties` qui est nécessaire pour des raisons techniques liées à l'approche de matérialisation mais dont vous n'aurez pas à vous préoccuper.

Le dossier `example` contient aussi la base de données SQLite `DBExample.db` (que vous ne pouvez pas éditer telle quelle, voir plus loin).

A noter : Le fichier obi-wan.properties spécifie le nom du fichier qui contient les mappings et la stratégie de réponse aux requêtes :

```
obiwan.ris.file = ris.json
obiwan.qastrategy.type = REW_CA
```

2 L'exemple jouet

La base de données exemple contient deux tables `EMPLOYEES` et `EMPLOYEES2` :

```
IDEMP|NAME|TEL|IDSUP
10|Paul|+330606060600|1
1|Paul|1840|
11|Chloé|0637234586|
12|Marie|0637234586|11

IDEMP|NAME|MAIL|DPT
10|Paul|paul@mybox.com|logistique
13|Bob|bob@mybox.com|logistique
```

L'ontologie décrit des classes (Telephone[1], Mail, Contact, Person, Employee) et leurs liens de sous-classe, des propriétés (hasTelephone, hasMail, hasContact, hasName) et leurs liens de sous-propriété, ainsi que les signatures de ces propriétés.

Remarque : Pour que la visualisation graphique de l'ontologie (voir section 4.2) fonctionne correctement, il faut sur chaque ligne du fichier .nt un espace avant le point final.

```
<ex:Telephone> <http://www.w3.org/2000/01/rdf-schema#subClassOf> <ex:Contact> .
<ex:Mail> <http://www.w3.org/2000/01/rdf-schema#subClassOf> <ex:Contact> .
<ex:Employee> <http://www.w3.org/2000/01/rdf-schema#subClassOf> <ex:Person> .

<ex:hasTelephone> <http://www.w3.org/2000/01/rdf-schema#subPropertyOf> <ex:hasContact> .
<ex:hasMail> <http://www.w3.org/2000/01/rdf-schema#subPropertyOf> <ex:hasContact> .
<ex:hasContact> <http://www.w3.org/2000/01/rdf-schema#range> <ex:Contact> .
<ex:hasContact> <http://www.w3.org/2000/01/rdf-schema#domain> <ex:Employee> .

<ex:hasTelephone> <http://www.w3.org/2000/01/rdf-schema#range> <ex:Telephone> .
<ex:hasMail> <http://www.w3.org/2000/01/rdf-schema#range> <ex:Mail> .

<ex:hasName> <http://www.w3.org/2000/01/rdf-schema#domain> <ex:Employee> .
```

Le fichier des mappings (`ris.json`) décrit 4 mappings (MhasName, MhasName2, MhasTel, MhasMail) et des informations sur l'environnement (connexion à des sources de données, nom de l'ontologie, etc.). Vous pouvez visualiser ce fichier avec un éditeur de texte, mais pour avoir une coloration syntaxique et voir les erreurs de syntaxe (ce sera très utile quand vous écrirez vos propres mappings !), nous vous conseillons d'utiliser un éditeur de code (par exemple Visual Code).

1. qui devrait s'appeler Phone mais il est trop tard pour changer son nom !

Voici une partie de ce fichier (1 mapping et les informations d'environnement) :

```
{
  "mappings": [
    {
      "name": "MhasName",
      "head": [
        [ "$VAR_PERSON", "<ex:hasName>", "$VAR_NAME" ]
      ],
      "body": {
        "datasource": "DBExample",
        "templates": {
          "VAR_PERSON": "<db:{IDEMP}>",
          "VAR_NAME": "NAME"
        },
        "query": "SELECT IDEMP, NAME FROM EMPLOYEES"
      }
    },
     ..... Autres mappings .....
   ],

  "datasources": [
    {
      "name": "DBExample",
      "type": "SQLITE",
      "parameters": {
      "databasePath":"./DBExample.db"
      }
    }
  ],
  "RDFSRuleSet": "RDFS",
  "name": "Base exemple",
  "description": "petite base exemple pour montrer comment on utilise obi-wan",
  "ontology": "./ontology.nt"
}
```

Notez ces quelques limitations des mappings dans Obi-Wan :

- Les requêtes SQL utilisées dans les mappings doivent avoir la forme `SELECT ... FROM ... WHERE ...` où WHERE contient des égalités (pas de requêtes imbriquées).
- Dans les requêtes SQL, qui *joignent* plusieurs tables, les attributs doivent avoir la forme `T.C` où `T` est un nom de table et `C` est un attribut de la table `T`. **Pour des raisons techniques, les labels des tables doivent être précisés dans la clause FROM avec le mot-clé AS (et pas un simple espace)**. En effet, Obi-wan n'utilise quasiment pas les informations de schéma et ne sait donc pas où chercher un attribut lorsqu'il y a plusieurs tables. Par exemple, la requête suivante, n'est pas correctement gérée par Obi-Wan parce que les attributs `MAIL` et `IDSUP` n'ont pas de nom de table :

  ```
  SELECT E.IDEMP, MAIL FROM EMPLOYEES AS E, EMPLOYEES2 AS E2 WHERE E.IDEMP = E2.IDEMP
  AND IDSUP=1
  ```

  Ceci peut être corrigé par la requête équivalente suivante :

  ```
  SELECT E.IDEMP, E2.MAIL FROM EMPLOYEES AS E, EMPLOYEES2 AS E2 WHERE E.IDEMP =
  E2.IDEMP AND E.IDSUP=1
  ```

Le fichier de requêtes contient ces deux requêtes :

```
Q0<$x,$y> :-
        triple($x,<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>,<ex:Employee>),
        triple($x,<ex:hasName>,$y);
```

```
Q1<$x,$y> :-
        triple($x,<ex:hasName>,"Paul"),
        triple($x,<ex:hasContact>,$z),
        triple($z,<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>,$y),
        triple($y, <http://www.w3.org/2000/01/rdf-schema#subClassOf>,<ex:Contact>);
```

Q1 demande les identifiants et noms de tous les employés ; Q2 demande les identifiants et les types de contact des personnes nommées "Paul" (remarquez qu'on ne veut pas le type Contact mais seulement des sous-classes).

Pour évaluer ces requêtes, tapez la commande ci-dessous dans un terminal (ici, on se place dans le dossier obda-systems/example qui contient les fichiers obi-wan.properties et queries.txt ; si vous procédez différemment, indiquez les chemins d'accès adéquats aux fichiers).

`java -jar ../../obiwan.jar obi-wan.properties queries.txt`
Vous voyez s'afficher les ensembles de réponses aux deux requêtes.

De plus, Obi-wan crée un dossier "sessions" contenant diverses informations sur le déroulement de l'interrogation. Regardons en particulier le dossier queries/Q0 : il contient un fichier avec les réponses (answers.csv), la requête (query.txt), l'ensemble de ses reformulations avec l'ontologie (ref.txt) qu'on peut voir comme une UCQ et l'ensemble des réécritures des requêtes reformulées (rew.txt) qu'on peut voir comme une union de requêtes sur la base de données source.

3 Explorer la base de données SQLite

SQLite est déjà installé sous Linux et MacOS en général : taper la commande sqlite3 dans un terminal pour le vérifier.

Tutoriel sur SQL Lite : `https://www.tutorialspoint.com/sqlite/index.htm`

En particulier, essayez ceci (en vous plaçant dans le répertoire qui contient la BD) :

`sqlite3` (ceci ouvre l'invite de commande de SQLite : vous pouvez ensuite taper des commandes commençant par un point, ou des commandes SQL qui doivent impérativement se terminer par un point virgule)
`.open obda-systems/example/DBExample.db` (ceci ouvre la base de données ou la crée si elle n'existe pas)
`.tables` (ceci affiche les noms des tables de la base)
`.schema` (ceci affiche les requêtes SQL de création du schéma de la base et permet de voir quelles sont les tables et leurs attributs)
`.header on` (ceci met les noms des attributs quand les réponses sont affichées)
`SELECT * FROM table_name;` (en mettant le nom de la table voulue : ceci affiche le contenu de la table ; ne pas oublier le point virgule à la fin)
`.quit` (ceci quitte SQLite)

Avant d'aller plus loin dans le TP : ajoutez un ou deux mappings et une requête au système OBDA exemple.

Vous pouvez notamment ajouter deux mappings qui permettent d'une part de lier un employé à son département ("si un employé d'identifiant ID est dans un département de nom DPT alors on a un triplet de sujet cet employé, de propriété `<ex:hasDptName>` et d'objet ce nom de département") et d'autre part d'inférer le département d'un supérieur hiérarchique à partir de celui de l'un de ses subordonnés ("si un employé d'identifiant ID1 a pour supérieur hiérarchique un employé d'identifiant ID2 et que ID1 est dans un

département de nom DPT alors on a un triplet de sujet le deuxième employé, de propriété `<ex:hasDptName>` et d'objet ce nom de département"). Ensuite vous pouvez poser la requête "Donnez les employés qui ont un département connu, ainsi que le nom de ce département".

Assurez-vous qu'Obi-wan s'exécute sans erreur et que les requêtes retournent bien les réponses attendues.

4 Visualisation et SPARQL endpoint des systèmes OBDA

Vous allez maintenant démarrer deux outils liés à Obi-wan qui faciliteront la conception d'un système OBDA :

1. Le premier outil est un SPARQL endpoint local à votre machine qui va vous permettre d'interroger le système en utilisant la syntaxe SPARQL. Notez cependant qu'Obi-wan n'accepte que des requêtes simples qui correspondent à rechercher un ensemble de triplets ("Basic Graph Pattern Queries").
2. Le second outil permet de visualiser les différents composants du système OBDA.

Le script `run-endpoint.sh` permet de lancer les deux processus. Exécutez-le avec la commande ci-dessous qui correspond au système OBDA exemple jouet :

```
./run-endpoint.sh obda-systems/example/obi-wan.properties
```

Remarques :

- Il faut laisser le script lancé dans un terminal durant l'utilisation du SPARQL endpoint et de la visualisation.
- Après chaque modification des fichiers des mappings ou de l'ontologie (ici, `ris.json` ou `ontology.nt`), il faut relancer cette commande pour que les modifications soient prises en compte.

4.1 SPARQL endpoint

Vous pouvez accéder au SPARQL endpoint dans votre navigateur à l'URL `http://localhost:8000/`. L'interface est séparée en deux ; en haut, un éditeur de requêtes SPARQL et en bas, un tableau des réponses. Remarquez que la requête qui vous est proposée au départ permet de visualiser tous les triplets RDFS obtenus en saturant l'ontologie et la base de faits issue des mappings avec les règles d'implication RDFS.

Essayez plusieurs requêtes de votre choix.

4.2 Visualisation des systèmes

Vous pouvez accéder à la visualisation des systèmes OBDA contenus dans le dossier `obda-systems/` à partir de votre navigateur avec l'adresse : `http://localhost:7000/index.html`. En visitant le lien du système exemple jouet, vous pouvez visualiser à gauche son ontologie RDFS sous forme d'un graphe et à gauche la liste de ses mappings. Le détail de chacun des mappings est accessible en cliquant dessus. Si vous modifiez l'ontologie ou les mappings, vous pouvez mettre à jour la visualisation en rechargeant la page.

À chaque fois que des requêtes sont évaluées depuis un fichier ou que le SPARQL endpoint est lancé, une session de requête est créée et est associée à un sous-dossier de "sessions" comme décrit plus haut. Ces sessions sont visualisables en suivant le lien "sessions", qui montre le détail de l'éxecution de chacune des requêtes. Vous pouvez voir les reformulations, les réécritures avec les mappings utilisés et les plans d'exécution de chaque requête ainsi que leur version optimisée où les opérateurs relationnels ont été dans la mesure du possible "poussés" dans les requêtes SQL sur la source SQLite.

5 A vous de jouer avec IMDb

Le dossier `obda-systems/imdb` contient le squelette d'un système OBDA exploitant la base SQLite `imdb.db`.

Relancer le SPARQL endpoint sur ce système OBDA (imdb) avec la commande :
`./run-endpoint.sh obda-systems/imdb/obi-wan.properties`
puis faire les petits exercices suivants pour se familiariser avec la base de données :

1. Visualiser le seul mapping du système, qui permet de renseigner les titres des oeuvres, puis écrire une requête SPARQL pour obtenir la liste des oeuvres avec leur titre.
2. Ajouter un triplet à l'ontologie (`imdb-ontology.nt`), pour pouvoir répondre à la requête SPARQL suivante :

   ```
   SELECT * WHERE {
     ?work <ex:hasTitle> ?title.
     ?work <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <ex:Work>.
   }
   ```

3. En utilisant une autre colonne de la table `title`, modifier le mapping pour répondre à la requête suivante :

   ```
   SELECT * WHERE {
     ?work <ex:hasTitle> ?title.
     ?work <ex:hasYear> "1931".
   }
   ```

4. En regardant les tables `movie_keyword` et `keyword`, ajouter un nouveau mapping pour répondre à la requête suivante :

   ```
   SELECT * WHERE {
     ?work <ex:hasTitle> ?title.
     ?work <ex:hasKeyword> "wizard".
   }
   ```

5. En utilisant les tables `name`, `cast_info` et `role_type`, exposer avec des mappings le rôle (acteur, réalisateur, scénariste, etc) des personnes dans chaque oeuvre. Trouver les réalisatrices qui ont joué dans un film qu'elles ont réalisé.

Voici quelques autres tables intéressantes de la base imdb :

- `cast_info` précise le personnage défini dans `char_name`, qui est joué par une actrice ou un acteur dans un film.
- `movie_info` et `movie_info_idx` précisent des informations sur les oeuvres, les différents types d'information sont dans dans `info_type`. On y trouve notamment les genres cinématographiques.
- `movie_link` précise des liens entre les oeuvres dont les différents types sont précisés dans `link_type`.
- `kind_type` contient les types d'oeuvre.
- `person_info` contient des informations sur les personnes, les différents types d'information sont à nouveau dans `info_type`.

Votre ontologie peut par exemple décrire les films avec leur titre, leur année, les rôles des personnes dans un film (Actor, Producer, Writer, ...), les keywords associés à un film, le genre des films, ...

Voici quelques idées de requêtes (à adapter selon votre ontologie) :

- Rechercher le genre d'un titre de film à partir de son titre.
- Rechercher des titres de films d'un certain genre à une certaine année.

- Rechercher dans quels films apparaît un personnage donné (par ex. Luke Skywalker).
- Rechercher si des personnes ont participé ensemble à des films.
- Rechercher les oeuvres télévisuelles (série, épisode, téléfilm).
- Rechercher les paires d'oeuvres réalisées par la même personne, dont la première est la suite de la deuxième.

6 Travail à rendre

Vous pouvez effectuer le TP par groupe de deux. On vous demande de choisir une thématique puis de construire une petite ontologie, quelques mappings pertinents pour 'peupler" l'ontologie et quelques requêtes. Choisissez des mappings et des requêtes qui montrent l'intérêt d'avoir une ontologie, que ce soit au niveau de la facilité de formulation des requêtes ou des capacités d'inférence (même si en RDFS celles-ci sont assez limitées). Préférez la qualité à la quantité !

Forme du travail rendu (sous moodle) : un fichier zip contenant le système OBDA construit (le dossier imdb, mais sans le dossier de sessions) et un document pdf qui décrit ce système (typiquement de 3 à 5 pages). Ce document devra comprendre :

- quelques explications sur la thématique choisie et une description de l'ontologie (éventuellement accompagnée d'une copie d'écran de sa visualisation) ;
- une description des mappings (en une phrase le rôle de chaque mapping, la requête SQL associée et le(s) triplet(s) produits) ;
- une liste de requêtes "intéressantes" avec leur nombre de réponses (et éventuellement quelques réponses).

Date de rendu : au plus tard le 8 mai