Llenguatges de consulta a l'àmbit de la Web Semàntica.

Cas d'estudi: SPARQL

Adrià Martínez Tort
Enginyeria Tècnica d'informàtica de Sistemes
Treball Fi de Carrera
Curs 2006-07/1

1.- Índex

Índex de continguts

2.- Introducció

El present document és la memòria del treball fi de carrera elaborat al voltant d'un projecte que porta per títol “Llenguatges de consulta a l'àmbit de la Web Semàntica: Cas d'estudi SPARQL”.

Aquest projecte, com índica el seu títol, s'emmarca dintre la idea de la Web Semàntica. Aquesta idea prové del fet que actualment totes les webs estan orientades per ser llegides per una persona i per tant disposen de tot un entorn gràfic i una organització per fer-les visualment atractives però poc útils per ser avaluades automàticament per una màquina. La Web Semàntica pretén, entre d'altres coses, introduir a les Webs un contingut estandarditzat que permeti el tractament automatitzat d'aquestes.

Aquest contingut pretén ser estructurat i amb alguna mena de significat que permeti, no només la llegibilitat per part d'una màquina, sinó que permeti fer alguna mena d'interpretació d'aquesta informació i establir relacions entre Webs o entre conceptes d'una mateixa web de manera automàtica però no aleatòria.

Per dur a terme aquesta tasca la Web semàntica està concebuda al voltant d'una sèrie de conceptes com són XML, XML-schema, RDF, RDF-schema, OWL i SPARQL els quals són l'objectiu de la primera part del nostre projecte. Mentre que la segona part busca una aplicació més pràctica de tots aquests conceptes com és una aplicació Web.

En concret la primera part del projecte intenta establir la idoneïtat d'utilitzar Sistemes Gestors de Bases de Dades (SGBDs) per emmagatzemar i explotar la informació de la Web Semàntica. Mentre que la segona intenta fer una aproximació més pràctica sobre el tema mitjançant l'elaboració d'una aplicació web utilitzant precisament alguns dels SGBDs

treballats.

Concretament els objectius fixats per al present projecte són els següents:

- Conèixer l'estructura i organització dels SGBDs que treballen amb informació basada en RDF.
- Avaluar l'adequació d'utilització dels SGBDs per desar i recuperar descripcions en RDF.
- Anàlisi comparatiu exhaustiu dels diferents (almenys dos) SGBDs estudiats a l'objectiu anterior.
- Estudi del llenguatge de consulta SPARQL.
- Realització de casos pràctics d'aquestes tecnologies, implementant una aplicació web senzilla que permeti fer cerques sobre documents RDF, amb dades relacionades amb la iniciativa FOAF.

La nostra memòria per tant està organitzada al voltant d'aquests dos objectius. A la primera part intentarem anar-nos introduint progressivament dintre el tema de la Web Semàntica fins arribar a establir la necessitat de tenir SGBDs per emmagatzemar la informació i intentarem avaluar dos d'aquests SGBDs.

La segona part intenta explotar algun dels SGBDs estudiats anteriorment per realitzar una aplicació web que permeti mostrar alguna aplicació de la Web Semàntica. Per tant aquesta part estarà organitzada més com un projecte de programari i no d'una manera tant teòrica com la primera part.

3.- La Web semàntica

3.1.-Introducció

La Web semàntica és un projecte conduit pel World Wide Web Consortium (W3C) que intenta definir un entorn comú que permeti que les webs puguin ser llegides i enteses (semàntica) per un ordinador. Actualment les webs presents a la Wold Wide Web (WWW) són uns documents orientats a ser visualitzats a través d'un explorador per un usuari humà i per tant es tracta de webs visualment atractives, però sense cap mena d'estructura estandarditzada que permeti a una màquina processar i entendre la informació continguda en aquestes Webs.

En concret el W3C defineix la Web semàntica de la següent manera:

- "La Web semàntica proveeix un entorn comú per compartir i utilitzar les dades en entorns d'aplicacions, empreses i/o comunitats. Es tracta d'un esforç cooperatiu conduit pel W3C amb la participació d'un gran nombre d'investigadors i empreses. Està basat en el Resource Description Framework (RDF)".

Per dur a terme aquest projecte la Web Semàntica es nodreix de tecnologies descriptives com són el Resource Description Framework (RDF) i la Web Ontology Language (OWL) així com de llenguatges altament flexibles per a la representació de dades com és l'eXtension Markup Language (XML).

La Web Semàntica no és rés més que una nova manera de representar dades i dintre d'aquest concepte l'XML és un llenguatge altament acceptat per l'intercanvi i

representació de la informació. Per tant no és estrany que l'estructura de les Webs de la web semàntica tinguin una estructura que es basi en l'XML.

L'XML però són una sèrie de normes bàsiques que defineixen un llenguatge de marques que permeten determinar que un fitxer és sintàcticament correcte pel XML però no determina si aquestes marques tenen un concepte al seu darrere o una estructura concreta.

Per fixar una estructura concreta i una sèrie de marques a un document XML existeix l'XML Schema. L'XML Schema és també un estàndard del W3C que permet definir una sèrie de normes que ha de seguir un document XML per ser considerat vàlid. Un XML Schema permet establir quines marques seran vàlides dintre d'un document XML i com és l'estructura interna d'aquest document: si una estructura pot estar repetida dintre d'aquest document, si només pot haver una estructura amb un nom determinat, etc.

L'RDF, Resource Description Framework, és un XML Schema pensat expressament per a la descripció de qualsevol tipus de recurs en el món real. L'RDF està basat en el concepte de tripletes subjecte-predicat-objecte i permet descriure tan les propietats d'aquests objectes del món real com la relació entre ells. L'RDF proporciona les eines per poder descriure els objectes però no proporciona un vocabulari específic per descriure tipus concrets d'objectes. Per dur a terme aquesta tasca existeix l'RDF Schema (RDFS).

L'RDFS ens permet treballar amb el llenguatge RDF com si fos un llenguatge orientat a objectes: podem definir classes i subclasses d'objectes, podem definir les propietats que tenen els objectes d'una classe i ens permet definir el tipus de dades d'aquestes propietats. Els objectes serien doncs instàncies d'aquestes classes. El projecte FOAF (Friend Of A Friend) és un RDFS concebut per descriure persones i les relacions entre

elles i és el que utilitzarem per la segona part d'aquest projecte.

Un cop definides classes i propietats d'aquestes classes podem determinar com són aquests predicats ontològicament segons la seva semàntica. Podem determinar si dos predicats són equivalents (email i adreça electrónica), si un predicat és simètric o no (si una persona és amiga d'una altra, aquesta última és amiga de la primera), o bé si aquest predicat és únic (una persona, una adreça electrònica). Totes aquestes propietats es defineixen mitjançant la Web Ontology Language (OWL) el qual és una extensió de vocabulari RDF que permet definir tota una sèrie de propietats relacionades amb el significat dels predicats. Aquesta darrera part no és un objectiu d'aquest projecte.

Un cop definits els objectius de la Web semàntica i presentats breument els seus components, en els propers apartats intentarem ampliar aquesta introducció encaminant les explicacions des del XML cap a la part principal d'aquest projecte com és bàsicament l'RDF i la seva estructura la qual és susceptible de ser desada dintre d'una base de dades per ser explotades per una aplicació.

3.2.-L'eXtensible Markup Language (XML)

Segons el propi W3C l'XML és un llenguatge de marques de propòsit general per la creació de llenguatges basats en marques de propòsit específic que permeti descriure qualsevol tipus d'informació. Per tant l'XML és un llenguatge bàsic que permet des de fer un document de qualsevol tipus amb informació estructurada, fins ser ell mateix el continent de la informació com una base de dades.

L'XML és una versió simplificada del Standard Generalized Markup Language (SGML). L'objectiu principal del XML és facilitar la compartició d'informació entre diferents sistemes, principalment aquells sistemes connectats via internet. Alguns dels llenguatges basats en l'XML són el Geography Markup Language (GML), RDF/SML, RSS, XHTML,...

L'XML és un llenguatge alfanumèric dissenyat expressament per poder aplicar una estructura d'arbre a la informació. L'XML està basat en una estructura molt simple que es pot repetir i niuar.

<nom atribut=”valor”>contingut</nom>

Aquesta estructura té tres parts: l'etiqueta (o tag)-inicial és el que conté el nom de l'element i els atributs d'aquest; el contingut; i finalment el tag-final que determina el final de l'element. S'anomena tag a aquella sèrie de caràcters englobats dintre els caràcters “< >”.

Per que un document es consideri ben format ha de seguir dos requeriments mol bàsics:

- Ha de contenir exactament un element **root**, on cap part d'aquest pot estar contingut en cap altre element. Per tant tot document XML té un element arrel.
- Pel que fa a la resta d'elements si el tag-inicial es troba en el contingut d'un element, el tag-final també ha d'estar en el mateix contingut.

Seguint aquestes normes el document XML es considera ben format però com es pot observar aquesta estructura té una sèrie de pros i contres. Entre els pros trobem els següents:

- És un format llegible simultàniament per humans i màquines.
- El format està basat en l'Unicode cosa que fa que gairebé qualsevol informació escrita per un humà pugui ser expressada en XML.
- Pot representar les estructures bàsiques de dades de la ciència computacional: vectors, llistes i arbres.
- Es tracta d'un llenguatge auto-documentat, és a dir, l'estructura i el nom dels elements permeten interpretar el document sense necessitat de conèixer amb detall el contingut del document.
- La estricta sintaxi permet l'existència d'algoritmes per l'anàlisi d'aquesta sintaxi simples, eficients i robustos.

Entre els contres podriem citar:

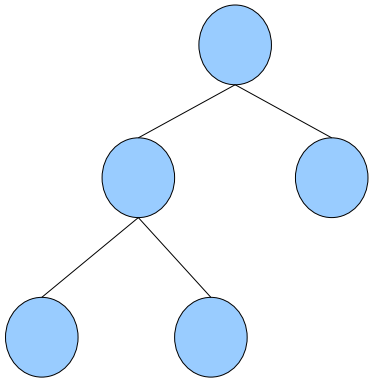
- Es tracta d'una sintaxi redundant que pot complicar la llegibilitat i la eficiència de les aplicacions i té un alt cost emmagatzematge.

- Els parsers han d'estar dissenyats per estructures arbitràriament niades i ha de suportar controls addicionals per detectar sintaxis o dades formatades incorrectament o ordenades malament. Això suposa un sobre cost per usos bàsics de l'XML sobretot per sistemes amb pocs recursos.
- L'XML no es una estructura dissenyada per validar tipus de dades per tant comprovar alguns tipus de dades suposa un esforç addicional.
- L'XML és un model jeràrquic que a diferència del model relacional dóna un visió fixe de la informació.
- Modelar estructures d'informació no jeràrquica requerix un esforç addicional.
- Transformar l'XML al model relacional és sovint molt costós.

D'aquests pros i contres els que més ens afecten a aquest projecte són els següents:

- És un format llegible simultàniament per humans i màquines.
- És tracta d'un llenguatge que pot representar les estructures bàsiques d'informació.
- L'XML és un model jeràrquic no orientat a la representació de dades del model relacional.

En concret ens podríem quedar amb la idea de que l'XML com una estructura arbòria amb un únic node arrel:



3.3.-XML Schema

Fins aquí hem vist quan es considera un XML ben format i també hem comprovat que l'XML és un llenguatge molt obert que no imposa cap restricció ni a la estructura ni al contingut només hi ha una certa restricció en la sintaxi.

L'XML Schema es tracta d'una recomanació del W3C i com el seu propi nom ja proposa s'utilitza per expressar un esquema sobre l'XML, és a dir, establir una sèrie de normes que ha complir un document XML per ser considerat vàlid per aquell esquema.

Les normes que es poden establir en un XML Schema per un document XML inclouen:

- El vocabulari (noms dels atributs i dels elements)
- El contingut (estructura i relacions entre elements)
- Els tipus de dades

Un Schema el pot definir qualsevol entitat segons les necessitats del seu aplicatiu mitjançant un fitxer XML Schema Definition (XSD). Aquest fitxer el pot distribuir a les entitats origen de la informació, si es considera necessari, per que aquestes puguin validar el document en origen i detectar si és vàlid sense necessitat de que l'entitat destí determini si la informació és vàlida o no.

Un exemple d'XML Schema Definition podria ser el següent fitxer *pais.xsd*:

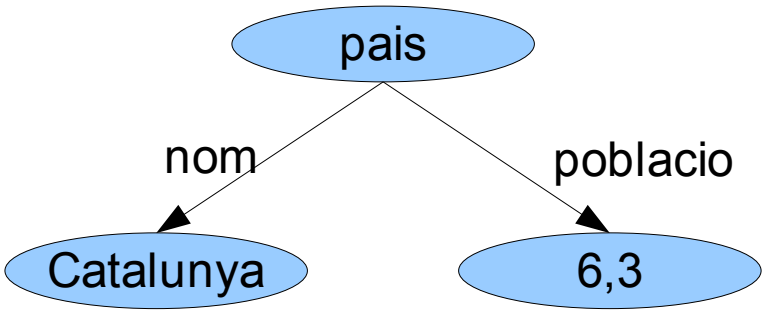
```
<xs:schema
 xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="pais" type="Pais"/>
 <xs:complexType name="Pais">
  <xs:sequence>
   <xs:element name="nom" type="xs:string"/>
```

```
   <xs:element name="poblacio" type="xs:decimal"/>
  </xs:sequence>
 </xs:complexType>
</xs:schema>
```

I un exemple de document vàlid per aquest esquema seria :

```
<pais
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="pais.xsd">
  <nom>Catalunya</nom>
  <poblacio>6.3</poblacio>
</pais>
```

Gràficament, i enllaçant amb la introducció realitzada del llenguatge XML, podríem interpretar l'anterior document com:

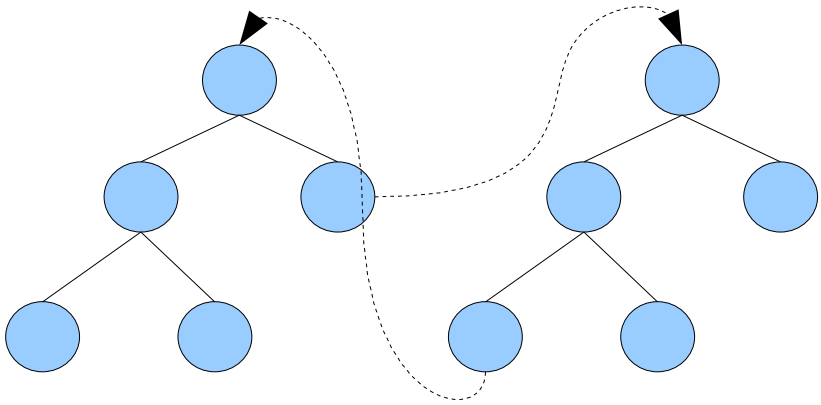


Ara ja podem dir que hem vist un llenguatge que pot representar gairebé qualsevol tipus d'informació i ja hem vist una manera de poder imposar una certa estructura a aquests documents XML. Per tant sembla que ja es van complint els requeriments que demanàvem a la Web Semàntica però ara el que ens queda per seguir avançant és definir una estructura semàntica que pugui ésser compartida dins la Web semàntica. És en aquest punt on entra en joc el Resource Description Framework. (RDF).

3.4.-Resource Description Framework (RDF)

L'RDF és un model de dades senzill que permet descriure tot tipus de recursos. Aquest model es basa en la descomposició de les descripcions en sentències formades per un subjecte, un predicat i un objecte (tripleta). El subjecte és qualsevol tipus de recurs disponible a la World Wide Web (WWWW) identificat per la seva adreça Uniform Resource Identificator (URI). El predicat és qualsevol mena de propietat que li volem descriure d'aquest recurs i l'objecte és el valor que pren aquest predicat per al recurs objecte. Per tant l'RDF es basa en tripletes per descriure els recursos disponibles a la xarxa.

La flexibilitat d'aquest model rau en el fet de que tant el subjecte com l'objecte poden ser recursos de la WWW per tant cada sentència RDF la podríem representar com un un link en un graf d'una xarxa infinitament extensa d'informació. Per tant gràcies al RDF estem passant d'una estructura arbòria a una estructura de graf:



Per tant gràcies al model RDF estem passant de l'estructura arbòria d'un document XML a una estructura de graf que ens podria arribar a relacionar totes les Webs de la WWW tot transformant-la en una base de dades extensa i dinàmica com és la WWW.

3.4.1.-Motivacions del RDF

El desenvolupament del RDF ha estat motivat per donar una sèrie de funcionalitats dintre d'internet:

- *Web metadata*: donar informació sobre recursos Web i els sistemes que l'utilitzen.
- Aplicacions que necessiten models dades de dades oberts més que limitats (per exemple: descriure activitats programades, descriure processos organitzats, fer anotacions a recursos web, etc.)
- Fer que la informació de la WWW, descrita mitjançant l'hypertext per un ús concret, fer-la processable per les màquines.
- Interoperabilitat entre aplicacions: permetre la combinació d'informació de múltiples aplicacions per construir nova informació.
- Automatitzar el processament de la informació Web mitjançant programes. La Web està evolucionant de només informació llegible per humans cap a una xarxa mundial de processos cooperants. Per tant l'RDF dota a la xarxa mundial d'un entorn independent del llenguatge per aquests processos.

L'RDF està dissenyat per descriure qualsevol objecte del món real. Per tant pot ser

utilitzat fins i tot en aplicacions aïllades on es requereix compartir informació. En aquest cas l'RDF proveeix un valor afegit ja que aquesta informació es podrà compartir de manera globalitzada, si algun dia es considera, sense necessitat de realitzar grans modificacions a l'aplicació. El valor de la informació per tant creix ja que pot ser accessible per moltes aplicacions a través d'Internet.

3.4.2.-Punts forts del RDF

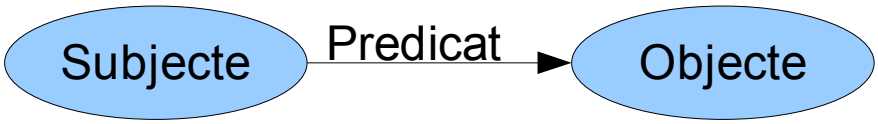
Resumint, els punts forts de l'RDF són els següents:

- És un model de dades senzill.
- Defineix una semàntica formal i fàcilment deduïble.
- Té un vocabulari extensible basat en les URI.
- Utilitza una sintaxi basada en l'XML.
- Suporta els mateixos tipus de dades que l'XML.
- Permet a qualsevol crear sentències sobre qualsevol recurs.

El punt més fort que ens interessa per seguir amb el nostre discurs és que utilitza una sintaxi basada en l'XML i per tant es pot definir mitjançant un XML Schema (o una ontologia, usant el llenguatge RDFS) que permeti a la informació ser processada per qualsevol aplicació que treballi en XML.

3.4.3.-Conceptes del RDF

La estructura bàsica de qualsevol document RDF és la d'una col·lecció de tripletes, cadascuna consistent en un subjecte, un predicat i un objecte. Cada tripleta la podem representar com un graf RDF. El graf RDF podria ser il·lustrat com un graf dirigit on cada tripleta estaria representada de la següent manera:



Cada node, tant el subjecte com l'objecte, de l'estructura poden ser identificats unívocament mitjançant una URI (Uniform Resource Identifier) que defineix la ubicació d'un recurs a Internet, un literal o un node blanc.

Una URI com a subjecte identifica el que el node representa. Una URI utilitzada com a predicat identifica un tipus de relació concreta entre les coses representades pels nodes que connecta. Una URI com a predicat pot ser considerat un node al graf igual que el subjecte.

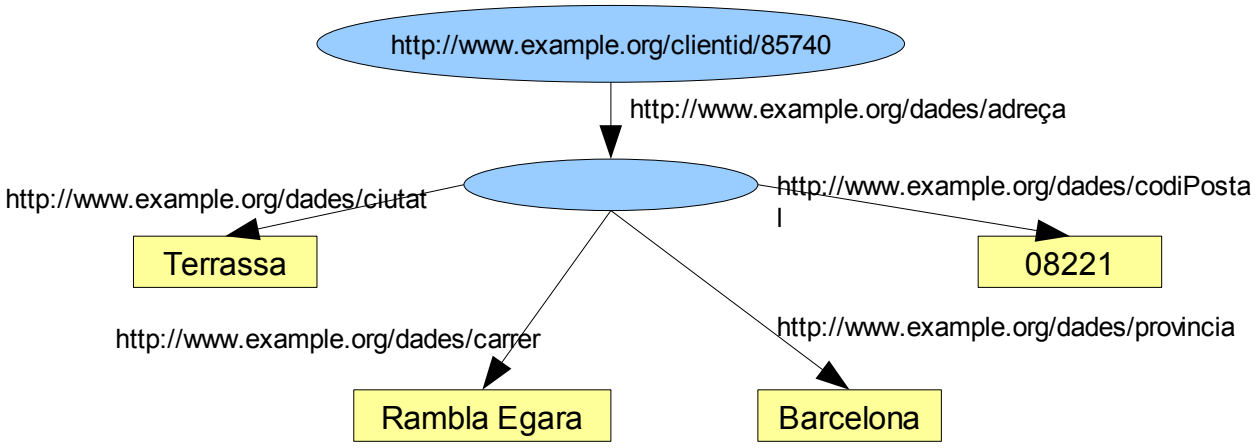
L'RDF, com a llenguatge, només predefineix un únic tipus de dades *rdf:XMLLiteral* utilitzat per incrustar codi XML dintre del RDF. En qualsevol cas al ser l'RDF un document XML es poden utilitzar els tipus de dades definits per aquests documents. Si és necessari utilitzar un nou tipus de dades el podem definir mitjançant XML Schemas.

Els literals són utilitzats per donar valors com números i dates. Qualsevol cosa identificada per un literal podria ser representada com una URI però sovint s'utilitzen literals per

resultar més convenient o intuïtiu.

Un literal pot ser l'objecte d'una sentència RDF, però mai el subjecte o el predicat.

Un node blanc és aquell que no és ni una referència URI ni un literal i s'utilitza per expressar fets simples. Per exemple en una base de dades relacional tenim que en una taula amb un nom determinat, cada fila representa una relació entre un recurs i un nombre il·limitat de columnes. Per poder-ho expressar en RDF hem de descompondre aquesta relació en tripletes. Una forma de descompondre aquesta fila és introduir un node blanc, corresponent a la fila, i es defineix una tripleta per cada camp de la fila entre el node blanc i el concepte del camp. Un exemple podria ser el següent:



Com veiem en aquest cas podria correspondre a una taula “Adreces_clients” amb diferents columnes com són *clientid, ciutat, carrer, província, codiPostal*. Aquest sistema ens permet representar en RDF qualsevol informació d'una base de dades.

3.5.-RDF Schema

L'RDF és un llenguatge de propòsit general per representar informació a la Web. El RDF ens permet expressar sentències simples sobre recursos, utilitzant propietats amb noms i valors. De totes maneres les comunitats que utilitzen l'RDF tenen la necessitat de definir els seus propis vocabularis (o ontologies) que utilitzaran a les sentències per descriure específicament un tipus de recurs amb unes propietats pròpies. Per exemple a una empresa *ex1* li interessaria descriure una classe com *ex1:Motxilla*, i utilitzar propietats com *ex1:model*, *ex1:pesEnKg* i *ex1:tamany* per descriure-la. Igualment una comunitat interessada en descriure recursos bibliogràfics *ex2* voldria descriure classes com *ex2:llibre* o *ex2:ArticleRevista* i utilitzar propietats com *ex2:Autor*, *ex2:Titol*, i *ex2:Tema* per descriure'ls. L'RDF per si mateix no té recursos propis per descriure amb tant de detall classes i propietats. En canvi aquestes classes i propietats les podem definir com un vocabulari RDF concret definit mitjançant l'RDF Schema (RDFS).

El RDFS no proporciona directament un vocabulari específic per cada classe com *ex1:Motxilla*, *ex2:Llibre,*etc. i per les propietats com *ex1:model*, *ex1:tamany*, *ex2:Autor,* etc. En canvi proporciona les eines per descriure aquestes classes i propietats, i també per indicar quines classes i propietats s'espera que vagin juntes (per exemple per dir que la propietat *ex2:Autor* serà utilitzada per descriure *ex2:Llibre*).

El RDFS és similar en molts aspectes amb els llenguatges de programació orientats a objectes, com pot ser el Java. En aquest sentit l'RDFS permet definir classes i subclasses i organitzar-les de forma jeràrquica i permet definir els recursos com a instàncies d'aquestes classes. Per exemple la classe *ex:Gos* podria ser definit com una subclasse de

la classe *ex:Mamífer* que alhora podria ser una subclasse de *ex:Animal*, volent significar que qualsevol recurs que es trobi a la classe *ex:Gos* forma part implícitament de les classe *ex:Mamífer* i *ex:Animal*.

Per altra banda, pel que fa a les propietats d'aquestes classes, el comportament és molt diferent respecte als llenguatges de programació. Una propietat no va lligada estretament a una classe, es pot definir una propietat independentment de la classe sobre la que serà utilitzada. Per exemple podríem definir una propietat *ex:Autor* i aquesta mateixa la podríem incloure dintre les classes *ex:Llibre*, *ex:Article*, etc. segons ho consideréssim pertinent.

Un RDF Schema és un graf RDF vàlid amb un vocabulari propi per tant qualsevol aplicació que treballi l'RDF no li trobarà cap significat en concret però no li trobarà cap error. Les aplicacions hauran de tenir per tant el mòdul adient per interpretar-los si volen treballar amb ells. Els recursos del vocabulari del RDF Schema tenen una referència URI amb el prefixe `http://www.w3.org/2000/01/rdf-schema#` (convencionalment representat mitjançant el prefix *rdfs:*).

Com a resum podríem dir que l'RDFS permet definir una ontologia sobre objectes del món real mitjançant classes i propietats mentre que l'RDF permet definir les instàncies d'aquestes classes.

Per acabar volem recalcar les principals diferències entre els RDFS i els llenguatges de programació orientats a objectes. Una diferència important és que en lloc de descriure les classes amb una sèrie de propietats, el RDF Schema descriu les propietats per sí mateixes i després les associa a les classes mitjançant les propietats domini (*domain)* i rant (*range)*. Per exemple un llenguatge orientat a objectes definiria una classe *llibre* amb

una propietat anomenada *autor* que tindria un valor de tipus *persona*. Un RDFS equivalent descriuria una classe *ex:Llibre* i en separadament una propietat *ex:Autor* que tindria un *domain* de *ex:Llibre* i un *range* de *ex:Persona*. Això que sembla només un canvi sintàctic és en realitat un canvi més profund. En Java una propietat *autor* d'una classe *Llibre* es considera un propietat diferent de la propietat *autor* de la classe *Article*. En RDF Schema la propietat expressa un concepte *Autor* que pot ser assignat a les classes adients *Llibre*, *Article* o altres.

Aquesta característica té l'avantatge de que es poden afegir propietats a les classes on inicialment no s'havia previst. Però per altra banda aquesta flexibilitat també pot provocar que una propietat pugui ser mal utilitzada en algun cas.

Una altra limitació és que no es pot definir un àmbit segons la utilització de la propietat. Per exemple la propietat *pare* aplicada a una classe *ex:Huma* és de tipus *Humà* en canvi la mateixa propietat aplicada a un *Tigre* és de tipus animal.

Una altra diferència és que en RDFS les descripcions no són necessàriament prescriptives com en el cas dels llenguatges de programació. És a dir, en un llenguatge de programació si es defineix una classe *llibre* amb una propietat *autor* de tipus *persona* tot això és considerat com un grup de limitacions. El llenguatge no permet la creació d'una instància de *llibre* sense l'atribut *autor*, i tampoc permet que la instància de *llibre* amb la propietat *autor* no sigui de tipus *persona*.

En el cas del RDFS això no es veu com una limitació. En RDFS podríem definir una propietat *autor* de tipus *persona, i* podríem tenir una instància d'un tipus desconegut. Aquesta instància podríem assignar-la a la propietat *autor* i per deducció podrem dir que aquesta instància és de tipus *persona* i en cap cas, de les deduccions que hem fet, es

donaria cap error. Això té avantatges i inconvenients ja que podríem arribar a tenir una empresa com a autora d'un llibre cosa que seria un error en el cas dels llenguatges de programació, però en RDF només estaríem ampliant el sentit d'autor a les persones jurídiques.

3.6.-Ontology Web Language (OWL)

Segons el W3C, el OWL proposa un vocabulari per descriure propietats i classes: entre d'altres, relacions entre classes (ex: disjuncions), cardinalitat (per exemple: exactament una), igualtat, característiques de les propietats (ex: simetria) i classes enumerades.

Sense entrar en més detalls, el OWL permet ampliar el Vocabulari del RDFS per assignar propietats a les propietats i a les classes i també ens permet determinar relacions entre classes. És a dir, ens permet, entre d'altres coses, definir si dues propietats o classes són equivalents: si la classe *cotxe* és equivalent a la classe *automòbil*. També ens permet definir si dues classes són una la inversa de l'altra: si *TePare* és la inversa de *TeFill* aleshores si la *Anna* és la instància de *TePare* de *Andreu* podem deduir que *Andreu* és la instància *TeFill* de *Anna*. També ens permet definir si una propietat és transitiva, (per exemple si definim *amic* com una propietat transitiva si A és amic de B i B és amic de C podem deduir que A és Amic de C). I així amb altres propietats.

Això, com podem veure, ens permet en llenguatge RDF generar tripletes noves a partir de les instàncies pròpies d'un document RDF ja que mitjançant l'OWL i aplicant les seves propietats podem anar deduint tripletes que enriqueixen les propietats disponibles. Això significa ampliar la interpretació dels documents i ens permet enriquir les possibilitats del RDF.

3.7.-Bases de dades

Com hem vist fins ara l'RDF és un llenguatge que permet representar internet com un graf de recursos web interrelacionats. Això suposa que la quantitat d'informació que ha de manegar qualsevol aplicació que treballi en RDF pugui arribar a ser considerablement gran.

Quan parlem de manegar informació, els sistemes més utilitzats actualment, els quals estan dissenyats expressament per ser eficients en el maneig de grans quantitats d'informació, són els sistemes gestors de bases de dades (SGBDs) relacionals.

Això ens permet dir que la millor manera per explotar la informació que puguem obtenir mitjançant documents RDF és mitjançant la utilització de SGBDs.

La prova d'això és que actualment ja existeixen SGBDs que permeten treballar directament amb informació RDF. En aquest apartat intentarem descriure alguns d'aquests sistemes que considerem interessants per algun motiu.

3.7.1.-Jena

Jena són un conjunt d'eines per tractar informació RDF introduïda com grafs RDF. Jena pot treballar amb RDF, RDFS i OWL veient totes aquestes com transformacions graf-a-graf produint grafs amb tripletes virtuals.

Jena és el sistema bàsic, programat en Java, que implementa totes les aplicacions que té el RDF, el RDFS i la OWL.

Les dues bases de l'arquitectura de Jena són:

- Múltiples i flexibles representacions dels grafs RDF al programador d'aplicacions. Això permet fàcil accés i manipulació de la informació dels grafs permetent al programador navegar fàcilment per l'estructura de tripletes. Particularment la “Model API” presenta el graf utilitzant termes i conceptes de les recomanacions RDF, i la “Ontology API” presenta els grafs utilitzant conceptes del OWL i RDFS.
- Una vista simple i minimalista dels grafs RDF al programador que els permet veure com tripletes. Això resulta útil a l'hora de realitzar deduccions o inferències sobre l'ontologia (RDFS o OWL) definida.

El primer objectiu és una capa superior a la segona, és a dir, Jena ha de poder treballar amb qualsevol font d'informació RDF, per exemple bases de dades, tripletes a memòria, o tripletes virtuals provinents de processos de deducció. Això es realitza mitjançant la definició de vistes en Java.

El llenguatges de consulta suportats per Jena és RDQL i SPARQL que poden ser utilitzats tant en grafs reals com en grafs virtuals provinents de processos de deducció.

Jena està estructurada per capes:

La “Graph Layer”: Tripletes com l'estructura universal de dades

La capa “Graph Layer” està basada en la “RDF Abstract Syntax”. L'objectiu d'aquesta capa és tenir els components mínims, qualsevol possible funcionalitat es farà en altres capes. Aquesta capa està pensada per implementar:

- L'emmagatzematge de tripletes a memòria o bé en entorns persistents.
- Vistes de només lectura com tripletes d'informació no estructurada en tripletes originalment, com la informació jeràrquica del sistema de fitxers, o obtinguda d'una pàgina web.
- Tripletes virtuals provinents del resultat de la deducció mitjançant tripletes definides com premises.

La implementació d'aquesta capa subministrada per Jena permet una gran varietat de sistemes concrets d'emmagatzematge de tripletes.

La “Model Layer”: Vistes per programadors d'aplicacions

Aquesta capa manté la “Model API” per raons històriques que és la primera abstracció del graf RDF utilitzat pel programador d'aplicacions. Aquesta capa proporciona un major nombre de mètodes per treballar tant amb el graf mateix com amb els nodes que el componen. Aquesta capa proporciona també una altra API com és la “Ontology API” per treballar amb conceptes de la RDFS i OWL.

La “EnhGraph Layer”: Múltiples vistes simultànies

La “Model Layer” i la “Ontology Layer” es troben per sobre de la “Graph Layer” mitjançant una capa intermèdia com és la “EnhGraph Layer”.

Aquesta capa és una mena d'extensió que proporciona vistes dels grafs, i vistes dels nodes del graf necessàries per les capes superiors. L'objectiu d'aquesta capa és reflectir

les possibilitats del RDFS mitjançant les possibilitats de Java i proporcionar-les totes alhora per que les capes superiors en puguin treure profit.

Aquesta estructura es reflecteix en el següent esquema:

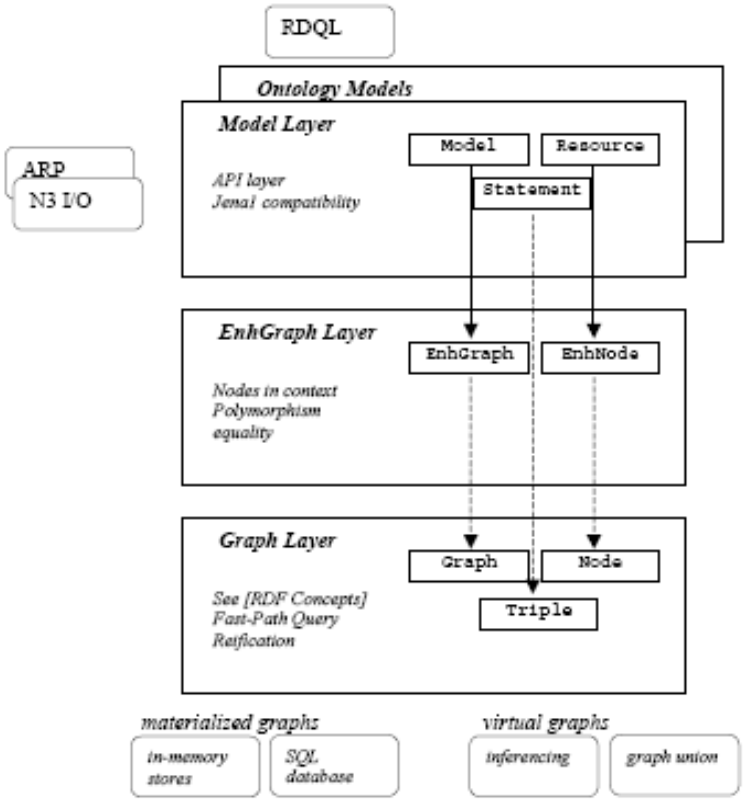


Figure. 1. The Jena2 Architecture

Entre les funcionalitats proporcionades per Jena es poden destacar:

- Una “Graph SPI (System programmer Interface)” que és la implementació que fa Jena de la “RDF abstract syntax”. Aquesta interfície té l'objectiu de fer eficients tan la consulta com la modificació de tripletes.
- Permet definir noves APIs per tal d'ajustar la presentació dels grafs a una aplicació en concret.

- Parsejar i deparsejar RDF/XML.
- Capacitat de realitzar deduccions mitjançant RDFS i OWL. Per dur a terme aquesta tasca disposa d'una sèrie de raonadors com són: “Transitive reasoner”, “RDFS Reasoner”, “Rubrik Reasoner”; i té la possibilitat d'incorporar raonadors externs.
- Realitzar consultes mitjançant RDQL i SPARQL.
- L'emmagatzematge persistent es realitza sobre bases de dades convencionals, com ara MySQL, Postgre, Oracle, etc.

3.7.2.-Sesame

Sesame és una base de dades RDF de codi obert amb suport per realitzar consultes i deduccions sobre RDFS. Sesame ha estat dissenyada des de la flexibilitat i pot ser implementada sobre una gran varietat de suports com són: bases de dades relacionals, sistemes de fitxers, indexadors de paraules claus, etc. i dóna als desenvolupadors una gran quantitat d'eines per tota mena d'aplicacions i llenguatges de consulta.

Una de les bases del projecte és que es vol mantenir Sesame independent del Sistema Gestor de Base de Dades (SGBD). Per aconseguir aquest objectiu Sesame disposa de SAIL (Storage and Inference Layer) una API que ofereix una sèrie de mètodes específics d'RDF als clients. Després, internament Sesame realitza la traducció d'aquests mètodes al SGBD específic. Això fa possible implementar Sesame en una gran varietat de sistemes sense canviar els components bàsics.

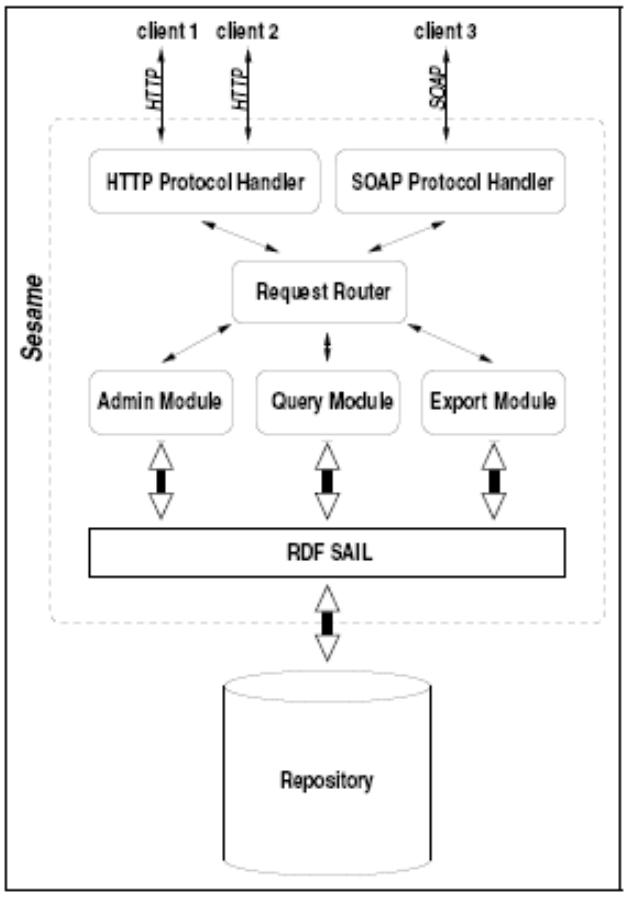


Fig. 3: Sesame's architecture.

Tots els mòduls funcionals de Sesame són clients de la API SAIL. Existeixen tres mòduls funcionals: El motor de consultes RQL i SPARQL, el mòdul d'administració RDF i el mòdul d'exportació RDF (veure Fig. 3).

El que es pot destacar del mòdul de consultes és que les consultes són optimitzades dintre d'aquest mòdul cosa que les fa independent del mètode d'emmagatzematge. Una altra aproximació seria transformar la consulta en una sentència executable al SGBD subjacent, però aleshores Sesame seria dependent del SGBD que és una de les premisses que no es volia complir.

El mòdul d'administració RDF permet a Sesame inserir informació RDF i RDFS a un

magatzem de dades. Les dues funcionalitats bàsiques d'aquest mòdul són :

- Afegir informació RDF i RDF Schema de forma incremental.
- Neteja del magatzem.

El mòdul d'administració agafa la informació d'una font RDF (normalment un document RDF en format XML) i l'analitza mitjançant un parser (usualment el parser de Jena), aquest envia la informació al mòdul d'administració sentència per sentència i el mòdul al seu temps, envia aquestes sentències de forma adient al SAIL i retorna els possibles errors que puguin haver sorgit en el magatzem pertinent.

El mòdul d'exportació RDF permet exportar informació del magatzem en format XML. La idea és que altres aplicacions RDF han de ser capaces de llegir, com a mínim, aquest format de document per tant aquest es considera la millor manera de compartir informació. Per altra banda, poden existir aplicacions que permetin l'ontologia, per tant la informació de l'esquema és útil, però per altra banda poden haver-hi aplicacions que no, per tant el mòdul d'exportació és capaç de generar ambdós tipus d'informació.

La principal eina de Sesame és la seva API SAIL. La API SAIL és una col·lecció de interfícies Java dissenyades específicament per l'emmagatzemament i recuperació de informació basada en el RDFS. Els principis de disseny de la API SAIL són:

- Definir una interfície bàsica per emmagatzemar informació RDF i RDFS, i recuperar i esborrar informació sobre repositoris persistents.
- Abstreure del sistema d'emmagatzematge: de si és un SGBD, sistema de fitxers , o a memòria,...

- Ha de ser utilitzable en dipositius simples com PDAs però ha de ser capaç, si és necessari, de manipular grans quantitats d'informació com una aplicació d'empresa.
- S'ha de poder extendre a altres llenguatges de consulta RDF.

Una de les principals característiques de la API SAIL és que és capaç de realitzar consultes tenint en compte l'RDFS, i ofereix mètodes per consultar limitacions de classe i propietats i restriccions de *domain* i *range*. Altres motors com Jena i Redland s'enfoquen exclusivament a les tripletes del RDF portant la interpretació de les tripletes a l'usuari.

Una altra de les avantatges de SAIL respecte altres APIs és que la API SAIL és molt lleugera: només es predefineixen quatre interfícies bàsiques, oferint funcionalitats bàsiques de consulta i recuperació i suport a les transaccions i poques coses més. Segons els desenvolupadors es considera aquest enfocament preferible a tenir APIs més complexes d'entendre i implementar.

3.7.3.-RAP

RAP és l'acrònim de “RDF API for PHP”. Es tracta d'una API per parsejar, cercar, manipular, seriar i servir models RDF dintre de la tecnologia LAMP (Linux Apache Mysql Php) i s'emmarca dintre la llincència GNU.

Les característiques principals de RAP són:

- Mètodes centrats en sentències per manipular un RDF com una col·lecció de tripletes.
- Mètodes centrats en recursos per manipular un RDF com una col·lecció de recursos.

- Mètodes centrats en l'ontologia per manipular un RDF a través de mètodes específics del vocabulari.
- Mètodes centrats en grafs per manipular conjunts de dades.
- Parser integrat per RDF/XML, N3, N-triple, TriX, GRDDL, RSS.
- Seriador integrat per RDF/XML, N3, N-triple, TriX.
- Emmagatzematge dels models a la memòria o a bases de dades.
- Motor de consultes per SPARQL i llibreria client per SPARQL.
- Motor de consultes RDQL.
- Motor de deduccions per raonaments RDF i per algunes deduccions OWL.
- Interfície gràfica per administrar els models RDF emmagatzemats en bases de dades.
- Suport pels vocabularis comuns.
- Dibuixar visualitzacions en graf.

RAP ofereix dues interfícies als programadors: Una API centrada en sentències, la *Model API*, que permet manipular els grafs RDF com un col·lecció de sentències; i un API centrada en els recursos, la *ResModel API* que permet manipular els grafs com una col·lecció de recursos.

La Model API suporta l'addició, esborrat o substitució de sentències dintre d'un model així com afegir models sencers. Hi ha quatre implementacions, o models, diferents de la Model API:

- MemModel: Model que emmagatzema els grafs a memòria. És el més ràpid però no

suporta deduccions.

- DbModel: Model que emmagatzema els grafs en bases de dades relacionals. Tampoc suporta deduccions.
- InfModelF: Model de deduccions encadenades cap endavant emmagatzema el graf base i les tripletes deduïdes a memòria.
- InfModelB: Model de deduccions encadenades cap enrera, emmagatzema el graf base a memòria i genera les tripletes deduïdes al moment.

La ResModel API representa els grafs RDF com recursos que tenen propietats. Aquesta API és la més semblant a la de Jena. La ResModel API sempre treballa sobre la Model API i té dues implementacions, o models, diferents:

- ResModel: Implementació bàsica de la ResModel API.
- OntModel: Es tracta de la ResModel API enriquida amb mètodes específics per RDFS.

3.7.4.-D2RQ

El projecte D2RQ permet tractar bases de dades relacionals clàssiques, no-RDF, com si es tractés de Grafs RDF virtuals. La base del projecte és que per adaptar bases de dades relacionals existents, actualment a l'entorn de la Web Semàntica i al RDF, no és necessari convertir totes les bases de dades a tripletes sinó que es pot definir un llenguatge declaratiu mapejat per tractar bases de dades relacionals no-RDF com grafs RDF virtuals mitjançant les eines de desenvolupament de Jena i Sesame.

Utilitzant D2RQ podem:

- Consultar bases de dades no-RDF utilitzant RDQL.
- Publicar contingut de bases de dades no-RDF a la Web Semàntica utilitzant el Joseki RDF server.
- Accedir a informació de bases de dades no-RDF utilitzant la API de JENA o la API de SESAME.
- Fer deduccions RDFS i OWL mitjançant del contingut d'una base de dades no RDF mitjançant l'ontologia de l'API JENA.

El D2RQ està implementat com un graf de JENA, la forma bàsica de representació de la informació del framework Jena. Reescriu les crides de l'API Jena, find() i consultes RDQL a crides SQL específiques del model de dades de l'aplicació. El resultat de les consultes SQL són transformades en tripletes que són passades a les capes superiors del framework de Jena.

L'estructura del D2RQ consisteix en:

- El “D2RQ Mapping Language”, un llenguatge per descriure entre la ontologia i el model de dades relacional.
- El GraphD2RQ, un plug-in per l'eina Jena Semanthic Web, el qual utilitza els mapejats per reescriure les crides de la API Jena a consultes SQL i passa el resultat de la consulta com tripletes RDF que són passades als nivells superiors del Jena Framework o altrament al nivells superiors del Sesame Framework.

3.7.5.-Taula resum

	Jena	**Sesame**	**RAP**	**D2RQ**
Llenguatge programació	Java	Java	PHP	Java
Format emmagatzematge	Qualsevol	Qualsevol	Qualsevol	Bases de dades relacionals
Model dades	RDF	RDF	RDF	Relacional
Llenguatge consulta	RDQL (SPARQL amb mòdul)	SeRQL, RQL, RDQL	SPARQL i RDQL	Jena/Sesame
Format sortida	RDF/XML, N3, N-tripes, Turtle	XML, HTML, RDF(RDF/XML, N-triples, Turtle)	RDF/XML, HTML	Jena/Sesame
Parser XML/RDF	Sí	No	Sí	No
Optimitzador	Sí	Sí	No	No
Ontologia	Sí	Sí	Si	Jena/Sesame
Control de restriccions domain i range	No	Sí	No	No
Operacions amb grafs	(UNION, INTERSECTION, DIFFERENCE)	No	No	No
Afegir/Modificar/Elminar tripletes	Si	Si	Si	No

4.- Cas pràctic

4.1.-Introducció

En aquesta part del projecte realitzarem una aplicació web senzilla que permeti fer cerques sobre sobre documents RDF amb dades relacionades amb la iniciativa "*Friend of a friend (FOAF)*" mitjançant el llenguatge de consultes SPARQL.

El desenvolupament d'aquesta segona part del projecte la iniciarem amb una breu introducció sobre els conceptes que intervenen en aquesta segona part de la pràctica: la iniciativa *FOAF* i el llenguatge de consultes SPARQL; i posteriorment detallarem els passos realitzats per a la realització de l'aplicació.

Per realitzar l'aplicació necessitarem una base de dades que ens permeti emmagatzemar el documents de la iniciativa FOAF i realitzar les cerques amb el llenguatge SPARQL. En el nostre cas, degut a que pretenem utilitzar al màxim els recursos del software lliure, ens hem decidit pel SGBD RAP. RAP que està programat en llenguatge PHP, està pensat per ser utilitzat en entorns de software lliure LAMP (Linux Apache Mysql Php) i a més compleix el requeriment de que es poden realitzar cerques mitjançant SPARQL.

4.2.-El projecte "Friend of a friend" FOAF

L'objectiu del projecte Friend of a Friend (FOAF) és la de crear una web amb pàgines llegibles per màquines que descriguin les persones, els enllaços entre elles i les coses que la gent crea i fa.

La filosofia del projecte FOAF es resumeix en la frase "FOAF és sobre el nostre lloc a la Web, i el lloc de la Web en el nostre món". FOAF es tracta d'una ontologia simple que fa fàcil compartir i utilitzar informació sobre gent i les seves activitats, per transferir informació entre llocs web, i automàticament ampliar, fusionar i reutilitzar-la on line.

El projecte FOAF bàsicament no és més que un RDFS on es defineix el vocabulari adient per poder descriure persones, l'activitat que desenvolupa i la relació amb altres persones. El projecte FOAF neix al considerar la relació entre persones la relació bàsica, fins i tot abans que la relació amb altres elements de la web (la gent té reunions amb altres persones, té cites amb altres persones, està present en fotografies, etc.).

L'evolució del projecte FOAF actualment està desenvolupant maneres de descriure qualsevol cosa de la Web o fins i tot agrupacions de coses, essent les persones un cas particular.

El vocabulari del projecte FOAF es pot agrupar en vàries categories:

- Vocabulari FOAF bàsic: és on es defineixen les classes principals, Person i Agent, i els conceptes més bàsics com són: name (nom), nick (sobrenom), homepage (pàgina web), etc.
- Informació personal: és on s'agrupen les propietats per donar informació de caràcter

personal: weblog (per indicar el weblog d'una persona), knows (per fer referència a les coneixences), interests (per fer referència als interessos de la persona), etc..

- Online Accounts / IM: és una categoria on s'agrupen les classes i propietats per fer referència a mitjans de comunicació web: ChatAccount, GameAccount, accountName, etc.
- Projectes i grups: En aquesta categoria s'agrupen les classes i propietats referents a grups i projectes. En aquesta categoria estarien les classes Project (pels projectes), Group (pels grups),... i les propietats member (per indicar els membres d'un projecte o grup), fundedBy (per indicar el fundador del grup o projecte), etc..
- Documents i imatges: En aquesta categoria és defineixen les classes i propietats necessàries per fer referència a qualsevol tipus de document, essent les imatges un tipus concret de document. Entre les propietats estarien made (per indicar l'autor), topic (per indicar el tema del document), etc..

La manera més habitual d'utilitzar el vocabulari FOAF és mitjançant la definició d'un espai de noms xml amb la següent referència:

```
xmlns:foaf="http://xmlns.com/foaf/0.1/"
```

Un document foaf bàsic tindria el següent aspecte:

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:foaf="http://xmlns.com/foaf/0.1/">
 <foaf:Person>
   <foaf:name>Adria Martinez</foaf:name>
   <foaf:mbox rdf:resource="mailto:amartineztor@uoc.edu"/>
   <foaf:workplaceHomepage rdf:resource="www.caixaterrassa.es"/>
   <foaf:schoolHomepage rdf:resource="www.uoc.edu"/></foaf:Person>
   <foaf:knows>
      <foaf:Person>
         <foaf:name>Oscar Celma</foaf:name>
         <foaf:mbox_sha1sum>1bc88e4e07879e806f7c1585e503cb225151f07d</foaf:mbox_
sha1sum>
```

```
      </foaf:Person>
   </foaf:knows>
 </foaf:Person>
</rdf:RDF>
```

En aquest document bàsic s'està descrivint una persona de nom “Adria Martinez”, que té una adreça de correu que es tracta d'un recurs disponible a la web “mailto:amartineztor@uoc.edu”, que la web del seu lloc de treball és també un recurs disponible a la web “www.caixaterrassa.es” que la web del seu lloc d'estudis és un recurs de nom “www.uoc.edu”, i que coneix una persona de nom “Oscar Celma” que té un compte de correu que en format sha1sum és la que s'indica al document.

4.3.-El llenguatge de consultes SPARQL

SPARQL és un llenguatge semblant al SQL que permet consultar, extreure i gestionar informació dels grafs RDF.

SPARQL és un llenguatge de consulta i un protocol d'accés per la Web Semàntica. SPARQL permet obtenir informació de grafs RDF i té les següents capacitats:

- Extreure informació en forma de URIs, nodes blancs i literals.
- Extreure subgrafs RDF.
- Construir nous grafs basats en informació de grafs consultats.

Com a protocol d'accés SPARQL permet tant l'ús local i l'ús remot.

Com a exemple podríem realitzar una consulta senzilla sobre el document FOAF de l'apartat anterior:

```
SELECT ?p1name
WHERE {?p1 <http://xmlns.com/foaf/0.1/name> ?p1name}
```

El resultat d'aquesta consulta seria el nom de totes les persones que surten en el document:

?p1name
Adria Martinez
Oscar Celma

Per simplificar la consulta la podríem reescriure-la de la següent manera:

```
PREFIX vcard:      <http://xmlns.com/foaf/0.1/>
SELECT ?p1name
WHERE {?p1 foaf:name ?p1name}
```

D'aquesta manera es facilita bàsicament la lectura de la sentència a un format més semblant a la dels documents RDF sense afectar per res al resultat. Mitjançant SPARQL també es poden fer consultes més elaborades com per exemple:

```
PREFIX vcard:      <http://xmlns.com/foaf/0.1/>
SELECT ?p1name ?p2name
WHERE {?p1 foaf:name ?p1name.
       ?p2 foaf:knows ?p2.
       ?p2 foaf:name ?p2name
}
```

Mitjançant aquesta consulta podem obtenir les coneixences de les persones presents en un document FOAF. En concret en el nostre cas el resultat de la consulta seria, indicant que ?p1name coneix a ?p2name:

?p1name	**?p2name**
Adria Martinez	Oscar Celma

4.4.-Aplicació web

4.4.1.-Instal·lació i configuració

La primera premissa per la realització de l'aplicació ha sigut la utilització de software lliure. Per aquest motiu partirem d'un equip PC estàndard amb una distribució linux instal·lada, en el nostre cas la distribució Ubuntu 6.06 TLS.

Degut a aquesta premissa la llibreria triada per manegar informació RDF ha estat el RAP. Aquest SGBD actualment està funcionant per entorns LAMP (Linux Apache Mysql Php). Per tant, mitjançant el gestor de paquets synaptic que incorpora la pròpia distribució de Linux hem procedit a instal·lar l'Apache 2, el PHP 5 i Mysql sense cap més indicació. L'únic retoc a realitzar a aquesta instal·lació és l'activació de l'extensió de Mysql a Apache 2 dintre del fitxer de configuració d'Apache. Aquesta extensió no es troba per defecte activada a la darrera versió per tant, tal com s'indica a tots els manuals, només cal descomentar la línia adient dintre el fitxer de configuració i tot funciona correctament.

Per tal de comprovar la configuració només cal crear un fitxer dintre de directori per defecte d'Apache amb la següent línea:

```
<? php phpinfo() ?>
```

Aquesta instrucció té com a resultat una pàgina on es mostra tota la configuració actual de nostre servidor web. En aquesta pàgina podem comprovar si s'han activat convenientment totes les extensions necessàries: php i mysql.

4.4.2.-Instal·lació de RAP

Per instal·lar RAP només cal descomprimir el fitxer de la seva pàgina web en algun punt de l'estructura de fitxers.

Per tal d'utilitzar aquesta API pels fitxers només cal afegir a la capçalera de les pàgines corresponents les següents línies:

```
define("RDFAPI_INCLUDE_DIR", "C:/Apache/htdocs/rdf_api/api/");

include(RDFAPI_INCLUDE_DIR . "RDFAPI.php");
```

La constant RDFAPI_INCLUDE_DIR és justament l'adreça on hem descomprimit el nostre paquet.

Finalment hem de modificar la següent línia de l'aplicació per accedir a la base de dades corresponent amb l'usuari i password adient. En el nostre cas:

```
$rdf_database = ModelFactory::getDbStore("MySQL", "localhost", "rap", "adria", "adria");
```

4.4.3.-Manual d'ús

L'objectiu de l'aplicació és oferir un entorn on un usuari pugui guardar models FOAF en la seva estructura de fitxer XML i pugui fer consultes sobre aquests models de la manera més fàcil possible.

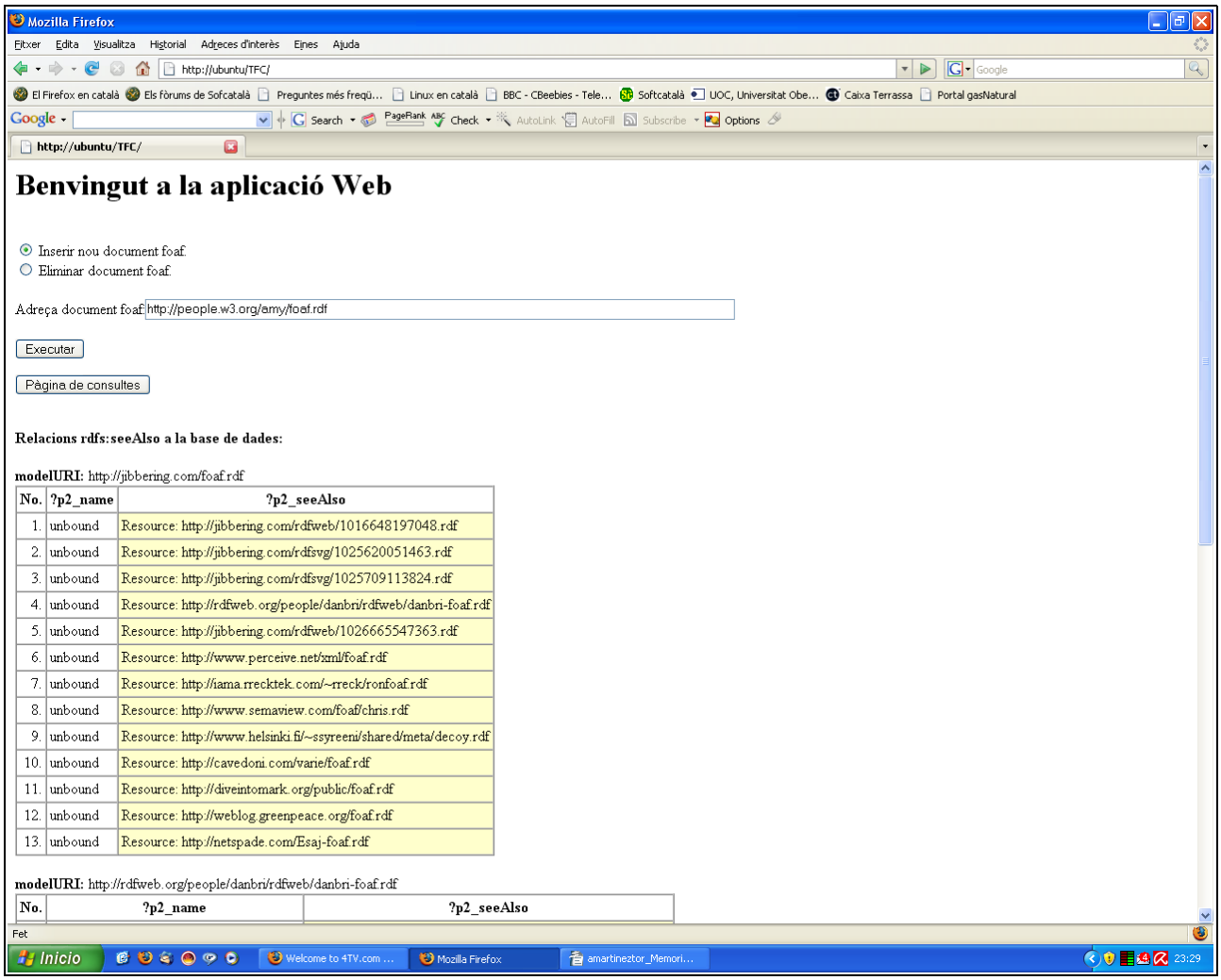
L'aplicació aprofita les capacitats gràfiques de representació de la informació RDF que té RAP i en concret la seva presentació de la informació en forma de taula HTML, per tal de mostrar el resultats de les cerques realitzades per l'aplicació.

L'aplicació s'inicia amb una pàgina des de la qual podem gestionar els models carregats en la nostra aplicació: podem carregar nous models o eliminar models existents.

Degut a que el projecte FOAF es basa en la relació entre persones, en aquesta pàgina inicial, a més dels models carregats a l'aplicació, també es mostren les relacions rdfs:seeAlso que hi ha en els models vinculats a relacions foaf:knows. L'objectiu d'aquest pas es mostrar noves pàgines FOAF relacionades en els models inicials, d'aquesta manera l'usuari pot triar carregar-les al seu temps dintre de l'aplicació per ampliar així els models sobre els quals realitzar les cerques.

La segona pàgina de la nostra aplicació és una pàgina on realitzar cerques mitjançant el llenguatge SPARQL sobre els models carregats a l'aplicació. En aquesta pàgina oferim una àrea de text on l'usuari pot introduir la seva consulta en format SPARQL i un botó per executar la sentència. El resultat de la cerca es mostra en format taula HTML per cadascun dels models carregats a l'aplicació.

4.4.4.-Captures de pantalla

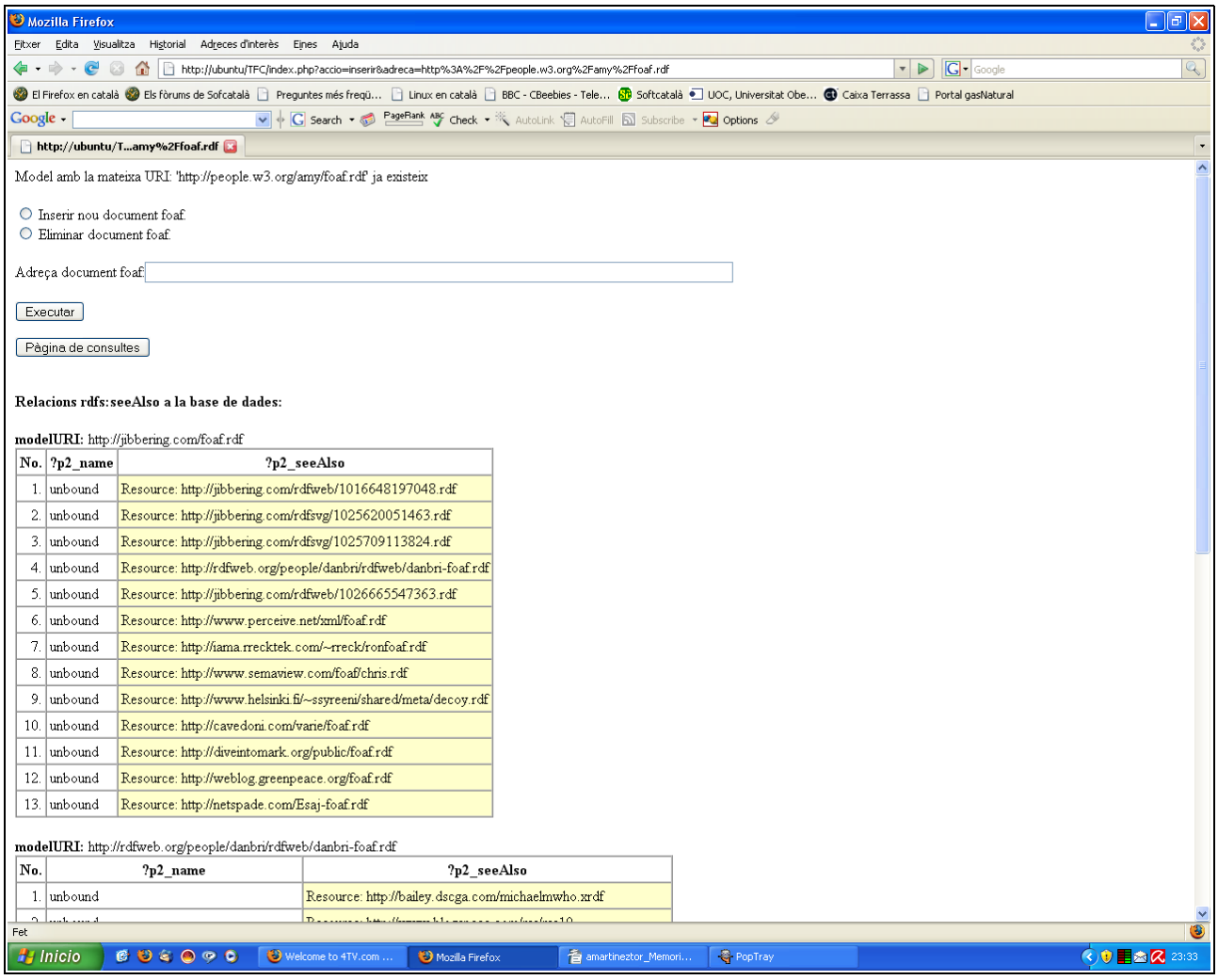


La pantalla inicial de l'aplicació dóna la benvinguda a l'usuari, ofereix les operacions disponibles i, si ja hi ha informació introduïda a la base de dades, ensenya la informació disponible. En la informació disponible es detalla el modelURI origen de la informació, i de cada model s'ofereixen les relacions *rdfs:seeAlso,* i el nom de la persona que ofereix aquesta relació. L'objectiu d'oferir aquestes relacions és que es tracta d'altres pàgines rdf que el propietari del model considera interessants de ser consultades i per tant són ofertes per l'aplicació per si l'usuari considera oportú incorporar-les a la base de dades.

Les opcions ofertes actualment són:

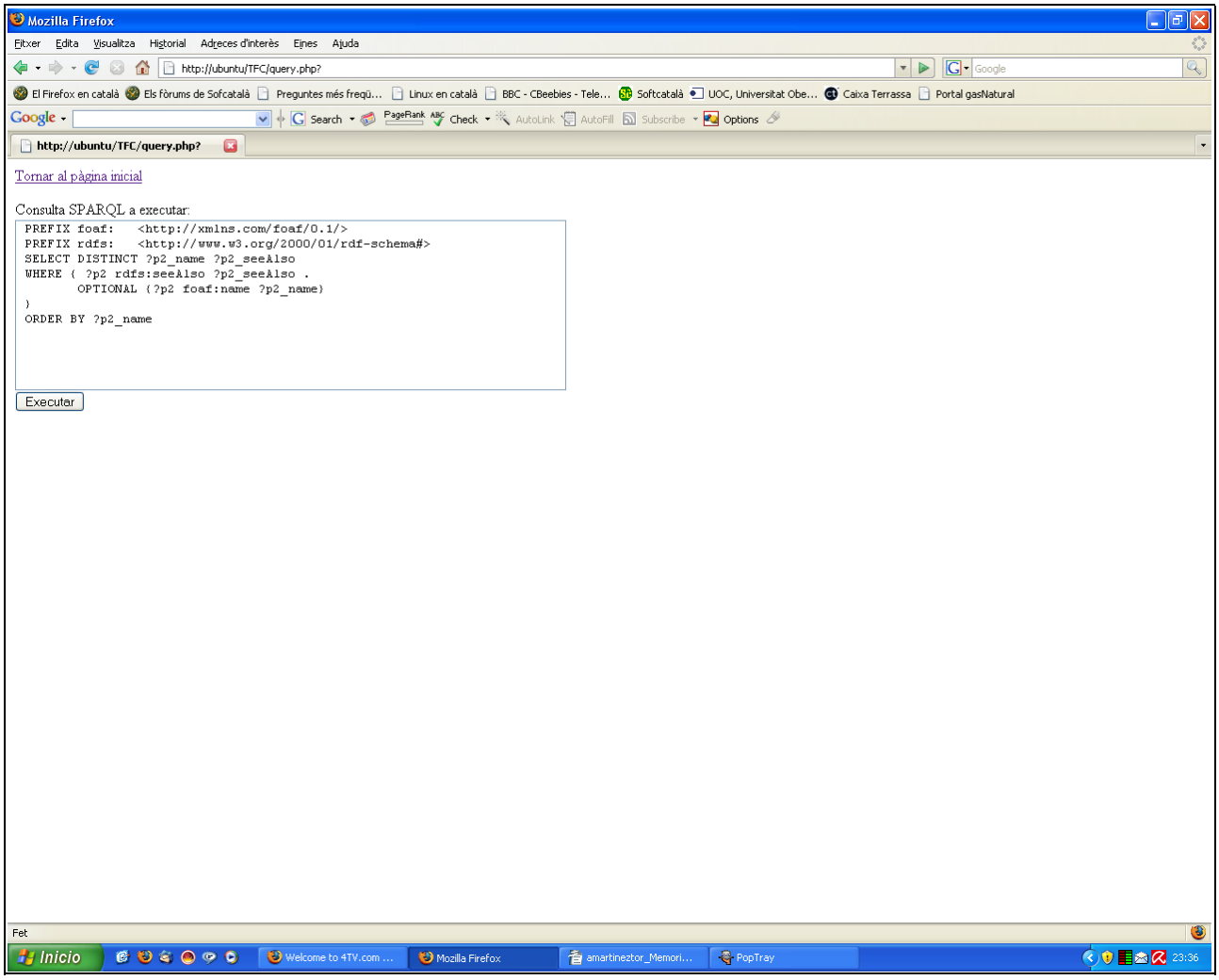
- Inserir nou document foaf: Insereix a la base de dades el document indicat al camp "Adreça document foaf".
- Eliminar document foaf: Elimina de la base de dades el document indicat al camp "Adreça document foaf".

L'opció escollida s'executa prement el botó "Executar".



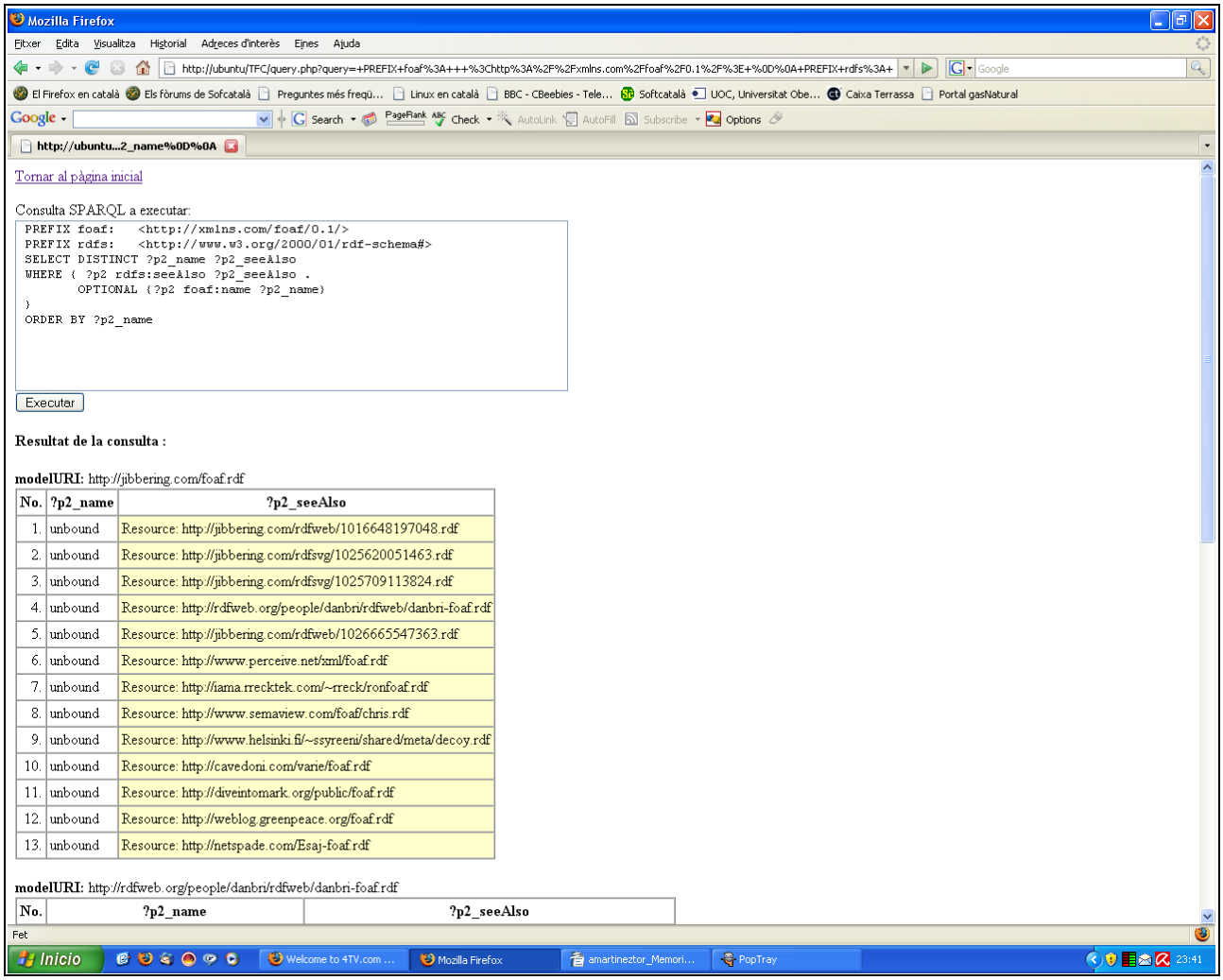
Un cop realitzada l'acció, el resultat és mostrat a la primera línia on anteriorment es mostrava el missatge de benvinguda.

Mitjançant el botó “Pàgina de consultes” enllacem amb la pantalla d'execució de sentències SPARQL.



La pàgina de consultes ofereix un enllaç per tornar a la pàgina inicial i un espai per introduir la consulta SPARQL que es consideri convenient realitzar sobre els models carregats a la base de dades. La sentència oferta inicialment és la mateixa que s'executa a la pàgina inicial.

Un cop introduïda la sentència, prement el botó “Executar”, executa la sentència per cadascun dels models carregats a la base de dades.



Un cop executada la consulta es mostra la sentència executada i el resultat de la consulta. El format del resultat, com a la pàgina inicial, és detallat per cadascun dels models identificats per la seva modelURI.

4.4.5.-Codi de l'aplicació

El codi de l'aplicació complert l'hem adjuntat a l'annex 2. En aquest apartat intentarem destacar els punts més importants de l'aplicació vinculats amb la Web Semàntica.

Com ja s'ha dit anteriorment, la nostra aplicació està basada en la API RAP. El primer punt d'interès de l'aplicació és la manera d'inserir un document dintre de la base de dades. Aquest procés es realitza, amb RAP, en quatre passos: creació de la instància, càrrega i parsejat del document, assignació del nom i inserció del model a la base de dades. El tros de codi corresponent és el següent:

```
// Creem un model
$memModel = ModelFactory::getDefaultModel();
// Load i parse del document
$memModel->load($base);
// Li assignem el nom al document que inserim
$modelURI = $_GET["adreca"];

// Abans d'inserir un document validem si ja existeix algún amb la mateixa modelURI
// si això fos així putModel() retornarà FALSE
if ($rdf_database->modelExists($modelURI))
  echo "Model amb la mateixa URI: '$modelURI' ja existeix";
else {
  $rdf_database->putModel($memModel, $modelURI);
  echo "Document inserit : ", $_GET["adreca"], "<br>";
}
```

Com es pot veure RAP utilitza dues classes diferents per realitzar aquesta acció. La classe *MemModel* és la encarregada de tot el treball amb els models, en el nostre cas la càrrega, parsejat i canvi de nom del document. Per altra banda hi ha la classe *DbStore* encarregada de la comunicació amb les bases de dades, en el nostre cas ens serveix per verificar si ja existeix i per inserir el model a la base de dades. Aquestes dues classes són

les bàsiques per al emmagatzematge permanent de models a la base de dades.

Un altre punt d'interès és la manera d'eliminar documents de l'aplicació. Aquesta acció es realitza a través de la classe *DbModel*. Aquesta classe és l'encarregada de la gestió dels models guardats de forma permanent en una base de dades. El codi corresponent a l'acció és el següent:

```
$dbModel = $rdf_database->getModel($_GET["adreca"]);
if($dbModel->isEmpty()){
        echo "Document inexistent.<br>";
} else {
        $dbModel->delete();
        echo "Document esborrat : ".$_GET["adreca"]."<br>";
}
break;
```

Finalment només volem destacar la manera en la que RAP realitza les consultes. Per realitzar aquesta acció RAP disposa de la classe *Model* la qual és l'encarregada de veure els grafs RDF i per tant l'encarregada de fer les consultes.

```
$resultat = $model->sparqlQuery($q);
SPARQLEngine::writeQueryResultAsHtmlTable($resultat);
```

La cosa més interessant d'aquest punt és la funcionalitat de RAP per poder visualitzar les consultes. RAP disposa del paquet SPARQL que permet diferents maneres de visualitzar el resultat de consultes. En el nostre cas hem pensat que la manera més interessant és visualitzar-la en forma de taula però RAP també disposa per exemple de la visualització en forma de graf.

5.- Conclusions

Un cop realitzat el projecte un s'endú la sensació de que hi ha un consens, per part de la comunitat, de que la Web ha de poder ser processada per màquines i, per tant, la idea de la Web Semàntica és àmpliament acceptada. Això ha dut a un desenvolupament teòric molt ampli i que sembla cobrir tota mena d'aspectes que es puguin donar en el món real.

Actualment també sembla que la teoria va molt per davant de les aplicacions i que no hi ha un camí clar d'aplicació d'aquesta teoria. Això provoca que actualment no hi hagi un interès massiu per l'aplicació d'aquesta nova tecnologia.

Un producte que demostra que la teoria s'allunya del que realment necessiten les webs és el sistema gestor D2RQ el qual tracta d'acostar el món de les bases de dades relacionals, àmpliament acceptat, al món dels grafs RDF per tal de que la distància entre els món de la Web tradicional i de la Web semàntica sigui més curta.

Una altra mostra és l'SPARQL. Aquest llenguatge té una definició prou extensa i detallada però hom comprova que l'implementació per part dels diferents productes és força bàsica, ja que consultes aparentment senzilles actualment encara no es realitzen correctament.

Sota el meu punt de vista és molt interessant seguir evolucionant en els llenguatges de consulta ja que són aquests els que ens permetran treure tot el partit a la Web Semàntica i el que dóna sentit a aquest concepte. Per altra banda la meva opinió és que també caldria estudiar seriosament la manera d'acostar el món de la Web tradicional al món de la Web semàntica, com el D2RQ, per tal de que hi hagi un esforç global vers l'aplicació d'aquesta idea.

La meva valoració del projecte és molt positiva ja que l'interès personal residia en veure més aplicacions del XML, a part del simple intercanvi d'informació, i per introduir-me en el món de la Web Semàntica, concepte que trobava força interessant. Crec satisfetes ambdues expectatives i valoro molt positivament el fet d'haver-me introduït en el món de la Web Semàntica ja que efectivament és un concepte molt interessant i digne de mantenir-hi un ull permanentment alerta ja que és una idea que tard o d'hora ha d'acabar reflectit en les webs tradicionals.

Finalment agrair el suport i les instruccions del tutor Òscar Celma sobretot a l'hora de posar en ordre tots els conceptes vinculats a la Web Semàntica que són molts i inicialment molt difícils d'ubicar. Gràcies.

6.- Annex 1 - Webgrafia

http://es.wikipedia.org

Wikipedia. L'enciclopèdia lliure

http://www.w3.org/

World Wide Web Consortium

http://www.xml.com

xml from the inside out

http://ramonantonio.net/contents/web_semantica

Web de Ramón Antonio Parada amb una introducció a la Web Semàntica.

http://www.wordreference.com

Word Reference, Online French, Italian and Spanish dictionary

http://www.openrdf.org/

Home de Sesame

http://sites.wiwiss.fu-berlin.de/suhl/bizer/rdfapi/

Home de RAP

http://sites.wiwiss.fu-berlin.de/suhl/bizer/d2rq/index.htm

Home del projecte D2RQ

http://jena.sourceforge.net

Home de Jena, semanthic web framework

http://www.hpl.hp.com/techreports/2003/HPL-2003-146.html

Synopsi de Jena feta per HP

http://www.foaf-project.org/

Pàgina del projecte foaf

http://www.rdfweb.org/

Pàgina web dels desenvolupadors del projecte FOAF

7.- Annex 2 - Codi de l'aplicació

7.1.-Codi del fitxer index.php

```
<?php
 // Include RAP
define("RDFAPI_INCLUDE_DIR", "./rdfapi-php/api/");
include(RDFAPI_INCLUDE_DIR . "RdfAPI.php");

?>

<html>
<body>

<?php
// Validem que la estructura estigui creada
$rdf_database = ModelFactory::getDbStore("MySQL", "localhost", "rap", "adria", "adria");
if (!$rdf_database->isSetup("MySQL") {
        $rdf_database->createTables("MySQL");
        echo 'Estructura MySQL iniciada<br>';
}
 // Tancament de la connexió
$rdf_database->close();
echo "<br><br>";
?>


<?php
// Segons l'entrada del formulari realitzem l'acció corresponent
if(empty($_GET["accio"]))
        echo '<h1>Benvingut a la aplicació Web</h1><br>';
else {
         // Connexió a MySQL
        $rdf_database = ModelFactory::getDbStore("MySQL", "localhost", "rap", "adria", "adria");
        switch ($_GET["accio"]) {
        case "inserir":
                if(empty($_GET["adreca"])) {
                        echo "Falta introduir l'adreça";
                        break;
                }
                 // Adreça del document FOAF
                $base = $_GET["adreca"];
                 // Creem un model
```

```
                $memModel = ModelFactory::getDefaultModel();
                // Load i parse del document
                $memModel->load($base);
                // Li assignem el nom al document que inserim
                $modelURI = $_GET["adreca"];

                // Abans d'inserir un document validem si ja existeix algún amb la mateixa modelURI
                // si això fos així putModel() retornarà FALSE
                if ($rdf_database->modelExists($modelURI))
                  echo "Model amb la mateixa URI: '$modelURI' ja existeix";
                else {
                  $rdf_database->putModel($memModel, $modelURI);
                  echo "Document inserit : ", $_GET["adreca"], "<br>";
                }
                break;
        case "eliminar":
                if(empty($_GET["adreca"])) {
                        echo "Falta introduir l'adreça";
                        break;
                }
                $dbModel = $rdf_database->getModel($_GET["adreca"]);
                if($dbModel->isEmpty()){
                        echo "Document inexistent.<br>";
                } else {
                        $dbModel->delete();
                        echo "Document esborrat : ".$_GET["adreca"]."<br>";
                }
                break;
        }
         // Tancament de la connexió
        $rdf_database->close();
        echo "<br><br>";
}
?>

<form name="index" action="index.php" method="get">
<input type="radio" name="accio" value="inserir"> Inserir nou document foaf.<br>
<input type="radio" name="accio" value="eliminar"> Eliminar document foaf.<br>
<br> Adreça document foaf:<input type="text"  name="adreca" size=100><br>
<br>
<input type="submit" value="Executar">
</form>
```

```
<form name="index2" action="query.php" method="get">
<input type="submit" value="Pàgina de consultes">
</form>

<br><b>Relacions rdfs:seeAlso a la base de dades:</b><br>

<?php
$rdf_database = ModelFactory::getDbStore("MySQL", "localhost", "rap", "adria", "adria");
$q='
 PREFIX foaf:   <http://xmlns.com/foaf/0.1/>
 PREFIX rdfs:   <http://www.w3.org/2000/01/rdf-schema#>
 SELECT DISTINCT ?p2_name ?p2_seeAlso
 WHERE { ?p2 rdfs:seeAlso ?p2_seeAlso .
            OPTIONAL {?p2 foaf:name ?p2_name}
 }
 ORDER BY ?p2_name
';

 // Obtenim un llistat de tots els models guardats a la base de dades
$list = $rdf_database->listModels();

// Fem la consulta anterior sobre tots els models de la base de dades
foreach ($list as $model) {
   echo "<BR>";
   echo "<b>modelURI: </b>" .$model['modelURI'] ."<BR>";
             // Es crea un nou MemModel i carreguem el document
            $model = ModelFactory::getDbModel($rdf_database, $model['modelURI']);

            if (!$model->isEmpty()) {
                        $resultat = $model->sparqlQuery($q);
                        SPARQLEngine::writeQueryResultAsHtmlTable($resultat);
            }
}

 // Tancament de la connexió
$rdf_database->close();
?>

</body>
</html>
```

7.2.-Codi del fitxer query.php

```
<?php
 // Include RAP
define("RDFAPI_INCLUDE_DIR", "./rdfapi-php/api/");
include(RDFAPI_INCLUDE_DIR . "RdfAPI.php");

?>

<html>
<body>

<a href="index.php">Tornar al pàgina inicial</a><br>

<form name="menu" action="query.php" method="get">
<p> Consulta SPARQL a executar:<br>
<textarea NAME="query" ROWS="10" COLS="70" wrap="virtual">
<?php if(empty($_GET["query"])) { ?>
 PREFIX foaf:   <http://xmlns.com/foaf/0.1/>
 PREFIX rdfs:   <http://www.w3.org/2000/01/rdf-schema#>
 SELECT DISTINCT ?p2_name ?p2_seeAlso
 WHERE { ?p2 rdfs:seeAlso ?p2_seeAlso .
          OPTIONAL {?p2 foaf:name ?p2_name}
 }
 ORDER BY ?p2_name
<?php } else echo $_GET["query"]; ?>
</textarea><br>
<input type="submit" value="Executar">
</form>



<?php
if(!empty($_GET["query"])) {
          echo "<br><br><b>Resultat de la consulta : </b><br>";
           // Connexió a MySQL
          $rdf_database = ModelFactory::getDbStore("MySQL", "localhost", "rap", "adria", "adria");

          $q=$_GET["query"];

           // Get an array with modelURI and baseURI of all models stored in rdf database
          $list = $rdf_database->listModels();

          // Realitzar la consulta a tots els models de la base de dades
```

```
        foreach ($list as $model) {
          echo "<BR>";
          echo "<B>modelURI: </B>" .$model['modelURI'] ."<BR>";
                  // Creació d'un nou MemModel i càrrega del document
                  $modelcarregat= ModelFactory::getDbModel($rdf_database, $model['modelURI']);

                  if (!$modelcarregat->isEmpty()) {
                          $resultat = $modelcarregat->sparqlQuery($q);
                          SPARQLEngine::writeQueryResultAsHtmlTable($resultat);
                  }
        }

 // Tancament de la connexió
$rdf_database->close();
}
?>

</body>
</html>
```