Estudi dels llenguatges de consulta per a documents RDF

Rafael Cèsar Llasera Fantova
Enginyeria en Informàtica

Òscar Celma Herrada

9 de Gener de 2006

Als meus pares, gràcies per l'ajuda
que m'heu donat en moments difícils.
Us estaré eternament agraït.

Títol: *Llenguatges de consulta per a documents RDF.*

Resum: *Tim Berners-Lee tenia dues visions del futur de la Web: la primera era fer de la Web un món més col.laboratiu i la segona que fos un mitjà més intel.ligible, i per tant, més processable per les màquines.*

Aquest projecte es centra en la Web Semàntica (veure 3.1.1), concretament en els llenguatges de consulta per a documents RDF. Es farà un estudi dels llenguatges de consulta existents en l'actualitat. S'estudiaran també els SGBDs que poden desar i recuperar dades en format RDF, estudiant les seves característiques arquitectòniques i tecnològiques a fons. Finalment, s'implementarà un Web senzill, que sigui capaç de realitzar consultes sobre un d'aquestos SGBDs, que haurà estat instal.lat i configurat prèviament.

Título: *Lenguajes de consulta para documentos RDF.*

Resumen: *Tim Berners-Lee tenia dos visiones del futuro de la Web: la primera era hacer de la Web un mundo más colaborativo y la segunda, que fuese un medio más inteligible, y por tanto, más procesable para las máquinas.*

Este proyecto se centra en la Web Semántica (ver 3.1.1), concretamente en los lenguajes de consulta para documentos RDF. Se realizará un estudio de los lenguajes de consulta existentes en la actualidad. Se estudiarán también los SGBDs que pueden guardar y recuperar datos en formato RDF, haciendo incapié sus características arquitectónicas y tecnológicas a fondo. Finalmente, se implementará una Web sencilla, que sea capaz de realizar consultas sobre uno de éstos SGBDs, que habrá sido instalado y configurado previamente.

Title: *Query languages for RDF documents.*

Abstract: *Tim Berners-Lee had two visions of the future of the Semantic Web: the first was to make the Web a more cooperative world, and the second, that it were a more intelligible medium, and therefore easier for machines process.*

This project is centered on the Semantic Web (see 3.1.1) and more specifically on the consultation languages for RDF documents. A study will be carried out on the consultation languages that exist at present. There will also be a study of RDBMs that can save and recover data in RDF format, with an in depth look at their architectural and technological characteristics. Finally, a simple Web will be implemented that will be capable of making queries on one of the RDBMs, that will have to have been installed and configured previously.

Índex

Índex de figures

Índex de taules

Capítol 1

Introducció

1.1 Justificació del PFC

L'eficient accés a dades, compartició de dades, l'extracció informació de les mateixes i el seu l'ús s'ha tornat una necessitat urgent de les empreses actuals. Amb la quantitat de dades que hi ha avui dia a la Web, la gestió d'aquestes s'està tornant quelcom impossible ([1]).

Les noves eines i tècniques és necessari que proveeixin interoperatibilitat, *tal com en el datawarehousing fa entre una gran diversitat de fonts i sistemes*, i extraient l'informació de les bases de dades. Les bases de dades XML i la Web Semàntica es situen com les noves tecnologies Web que les empreses necessiten per a extreure informació de la Web i per usar-la eficientment.

Per extreure aquesta informació s'utilitza el concepte d'ontologia, el qual ens proveeix d'un vocabulari comú d'un àrea determinada i defineix, amb diferents nivells de formalitat, el significat dels termes i de les relacions entre ells. L'*Enginyeria Ontològica* fa referència a un conjunt d'activitats que formen part del procés de desenvolupament d'ontologies, el cicle de vida de les ontologies, i dels mètodes i metodologies per a la construcció d'ontologies i de les eines i llenguatges que les suporten.

Durant la darrera dècada s'ha potenciat l'estudi de les ontologies. Les ontologies, avui dia, són àmpliament utilitzades en l'Enginyeria del Coneixement, la Intel.ligència Artificial i la Informàtica; també en aplicacions de Gestió del Coneixement, en el Llenguatge Natural, el Comerç Electrònic, la Integració Intel.ligent d'Informació, l'Educació i en nous camps emergents de la Web Semàntica.

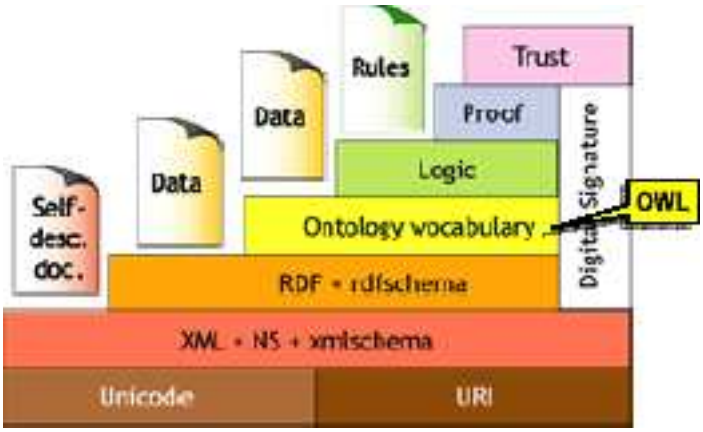

Figura 1.1: Nivells de la Web Semàntica.

Construir ontologies és un procés complexa i consumeix molt de temps, i ho es encara més si es fa directament amb el llenguatge de les ontologies, sense cap suport d'eines. El llenguatge RDF (Resource Description Framework) ens proveeix d'interoperabilitat per al intercanvi d'informació a través de la Web i per poder representar el coneixement ontològic, encara que existeixen d'altres llenguatges per la creació d'ontologies (com OWL, etc.), ens centrarem però amb aquest.

1.2 Objectius del PFC

Aquest projecte es centra en l'estudi i avaluació d'alguns sistemes gestors de bases de dades (SGBD), que hi ha en l'actualitat, per desar informació dins del context de la Web Semàntica (per exemple: RDFSuite, Sesame, RDF Gateway, Redland, 3store, etc.), avaluant-los en la seva capacitat per a recuperar i emmagatzemar descripcions en RDF (mitjançant emmagatzematge en tripletes), fent un anàlisi comparatiu entre varis d'ells i a diferents nivells. També s'estudiaran alguns dels llenguatges de consulta per a documents RDF que existeixen en l'actualitat, posteriorment aplicarem aquest coneixement en la construcció d'una web senzilla.

Podem dir, doncs, que els nostres objectius bàsics seran:

- Estudiar les diferents propostes de llenguatges de consulta per a documents RDF que hi ha en l'actualitat.
- Conèixer l'estructura i organització dels SGBDs que treballen amb informació basada en RDF.
- Avaluar l'adequació d'utilització dels SGBDs per a desar i recuperar descripcions en RDF.
- Anàlisi comparatiu exhaustiu dels diferents SGBDs estudiats.

- Realització de casos pràctics d'aquestes tecnologies, implementant una aplicació web senzilla que permeti fer cerques sobre documents RDF.

1.3 Enfocament i mètode seguit

En el desenvolupament d'aquesta memòria s'han tingut molt en compte els objectius a assolir en aquest projecte. Tot i que la part d'XML s'ha de suposar que en aquest nivell ja és coneguda, he considerat necessari incloure els conceptes fonamentals, per a que el lector pugui entendre posteriorment els capítols següents.

La metodologia d'aquest projecte es basa en el model tradicional de descomposició en cinc fases, que són:

- **Aprovació:** Dintre d'aquesta fase el consultor va proposar el projecte i va ser validat i acceptat, incloent-lo posteriorment en l'oferta de PFCs.
- **Definició i Planificació:** Segons la proposta assignada de PFC, s'elabora un Pla de Treball on es defineix el projecte, s'especifiquen els objectius i s'estableix una planificació temporal. Aquesta planificació serà presentada al consultor de l'assignatura perquè sigui validada.
- **Execució:** Una vegada aprovada la planificació del projecte, s'inicia la fase de recollida d'informació i posteriorment es comença amb el desenvolupament del projecte. Aquest desenvolupament ha anat controlat per part del consultor mitjançant diferents lliuraments parcials, que s'han realitzat segons les fites establertes. La fase de desenvolupament ha estat la més costosa de totes.
- **Tancament:** S'ha preparat la memòria final del projecte, indicant si s'han assolit els objectius indicats a l'inici. Finalment serà el Tribunal l'encarregat d'avaluar el grau d'assoliment del mateixos.

Finalment, es podria dir que tot i que la fase d'Execució és molt important, penso que la de definició i planificació encara ho és mes, ja que un mal plantejament em podria haver fet arrossegar els mateixos errors durant tot el projecte.

1.4 Planificació del projecte

La planificació del projecte es va fer tenint en compte les dates els diferents lliuraments parcials (PAC) i els diferents interfaces a generar. Es a dir, es va subdividir la feina a realitzar en vàries subtasques. Posteriorment, es va fer una valoració del volum de treball que significava cadascuna de les tasques. A continuació es pot veure una taula resum de les principals

tasques i un diagrama de GANTT que mostra la planificació temporal i les fites principals.

1.4.1 Divisió de les tasques

	Nombre de tarea	Duración	Comienzo	Fin	Predeceso
1	T01 - Inici del PFC	0 días	vie 16/09/05	vie 16/09/05	
2	T02 - Seguiment i tutorització	83 días	sáb 17/09/05	lun 09/01/06	1
3	**T03 - Elaboració del pla del Projecte**	**1 día**	**sáb 17/09/05**	**sáb 17/09/05**	**1**
8	**FITA DE CONTROL PAC1**	**0 días**	**lun 26/09/05**	**lun 26/09/05**	
9	**T04 - Introducció**	**5 días**	**dom 18/09/05**	**jue 22/09/05**	**3**
14	**T05 - Conceptes Previs**	**12 días**	**vie 23/09/05**	**lun 10/10/05**	**13**
19	**T06 - El Resource Description Framework**	**14 días**	**mar 11/10/05**	**vie 28/10/05**	**18**
24	**T07 - Estudi d'SGBDs per a documents RDF**	**23 días**	**lun 31/10/05**	**mié 30/11/05**	**23**
28	**FITA DE CONTROL PAC2**	**0 días**	**mié 02/11/05**	**mié 02/11/05**	
29	**T08 - Cas Pràctic**	**20 días**	**jue 01/12/05**	**mié 28/12/05**	**27**
36	**FITA DE CONTROL PAC3**	**0 días**	**lun 12/12/05**	**lun 12/12/05**	
37	T09 - Conclusions	2 días	jue 29/12/05	vie 30/12/05	35
38	T10 - Repasar i corregir la memória	6 días	lun 02/01/06	lun 09/01/06	37
39	T11 - Finalització i lliurament	0 días	lun 09/01/06	lun 09/01/06	38

Figura 1.2: Divisió de tasques del projecte

1.4.2 Diagrama de Gantt

A continuació es mostra el diagrama global de cadascuna de les activitats principals a desenvolupar.

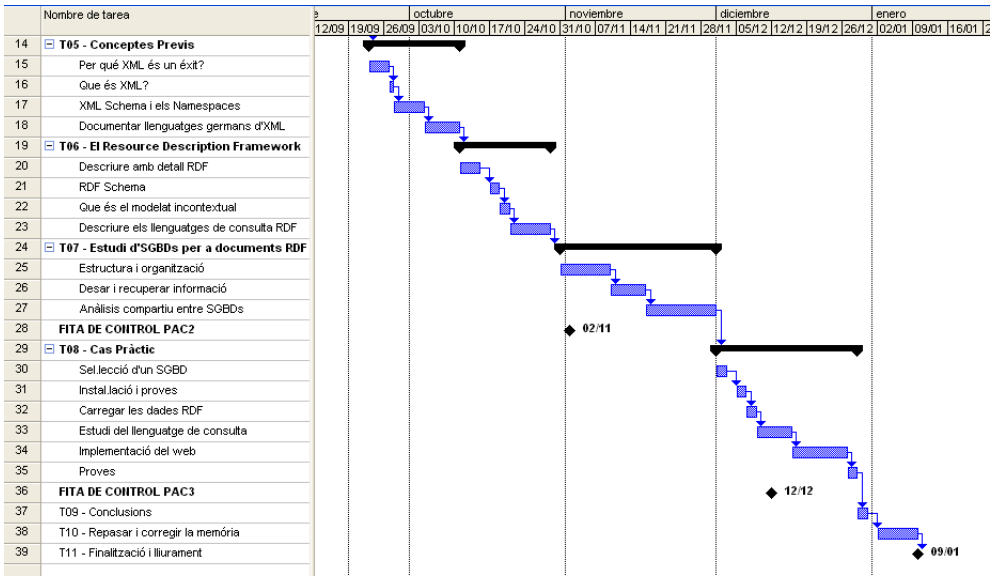

Figura 1.3: Diagrama de Gantt.

1.5 Productes obtinguts

Com a productes resultants de portar a terme aquest projecte, i que podrien ser utilitzats com a base per a posteriors ampliacions del mateix, podem anomenar la web de la botiga Musicland, creada amb la finalitat de poder realitzar consultes sobre el *Sesame,* que és el SGBD[1] de dades RDF que he escollit per la seva gran difusió, i utilitzant com a backend el *MySQL.*

Musicland és una empresa que es dedica a la venda de música per Internet, però també permet que l'usuari pugui vendre la seva pròpia música. L'usuari pot cercar la música de l'artista que li interessi, i un cop escollida la composició que vol, la pot adquirir.

Podem dir, doncs, que hem obtingut els següents productes:

- *Un SGBD de dades RDF que muntat sobre el contenidor de servlets Tomcat, i que utilitza com a backend la base de dades relacional MySQL.*
- *Un conjunt de dades RDF carregades sobre l'SGBD Sesame i que poden ser consultades.*
- *Un Web d'una botiga virtual que permet a l'usuari fer consultes, compres i vendes de música, entre d'altres coses. La botiga virtual ha estat codificada utilitzant el llenguatge PHP 5.0 (veure 6), i utilitzant una sèrie de classes i extensions del llenguatge (algunes classes de PEAR) que faciliten l'enllaç amb el Sesame.*
- *Un conjunt de sentències escrites amb el llenguatge SeRQL (veure 4.2.2.2) que serveixen per a fer les tasques requerides.*

L'instal.lació de la botiga virtual no és molt complicada, però cal senyalar que prèviament necessitarem tenir configurat adequadament el Sesame i també el PHP 5.0, tots dos funcionat sobre el contenidor de servlets Tomcat 4.x o una versió superior (això s'explicarà detalladament en els apèndixs final d'aquesta memòria).

1.6 Breu descripció dels altres capítols

- **Capítol 2 - Conceptes Previs:** S'introdueixen els conceptes bàsics del llenguatges XML (*Extensible Markup Language*), i també s'introduiran els seus llenguatges associats o derivats, com: *XSL*, *XSTL*, *XPath*, *XQuery*, etc.
- **Capítol 3 - El Resource Description Framework (RDF):** Introdueix, al igual que en el capítol anterior, una sèrie de conceptes previs i desprès, descriu detalladament el llenguatge RDF i també el RDF Schema, comparant-lo amb el XML Schema.

[1] Sistema Gestor de Bases de Dades.

- **Capítol 4 - Llenguatges de consulta RDF:** Seguidament, es detalla els llenguatges de consulta per a documents RDF que hi ha en l'actualitat, descrivint les principals característiques de cadascun d'ells i a més fa una comparativa dels llenguatges segons varis casos d'ús.

- **Capítol 5 - Estudi d'SGBDs per a documents RDF:** Es fa una descripció de les principals característiques dels SGBDs que són capaços de desar i recuperar dades RDF, per exemple: *RDFSuite, Sesame, RDF Gateway, RedLand, etc.* Es fa també una comparativa exhaustiva dels llenguatges *RDSuite*, *Sesame* i RDF *Gateway*.

Capítol 2

Conceptes Previs

En aquest capítol s'introduiran els conceptes bàsics del llenguatge XML (*Extensible Markup Language*), fent especial rellevància amb el *XML Schema* i els *Namespaces*, i també s'introduiran d'altres llenguatges associats o derivats com: XSL, XSLT, XPath, XQuery, XPointer, XLink, XInclude i XHTML (aquests conceptes han estat extrets del llibre de Michael C. Daconta [2], i dels articles [3, 4, 5, 6, 7, 8, 9]).

2.1 Per què XML és un èxit?

Actualment, l'ús principal d'XML és per a l'intercanvi de dades entre organitzacions externes. En aquest aspecte, XML juga un paper important en el mecanisme d'interoperatibilitat. Com *XQuery* i *XML Schema* estan aconseguint una gran maduresa i difusió, XML pot convertir-se la eina sintàctica més important per a l'intercanvi d'informació entre empreses. *Per què XML tant èxit?* XML té quatre propietats molt importants, aquestes són:

- L'XML crea documents i dades independents de l'aplicació.
- Té una sintaxis estàndard per a metadades.
- Té una estructura estàndard per a ambdós: documents i dades.
- L'XML no és una tecnologia nova.

Un altra de les raons del perquè la tecnologia XML és un èxit és perquè és text pla el qual fa que estigui en un format fàcilment llegible per als humans. La següent figura mostra un document MS Word que contrasta amb el format binari amb el que MS-Word emmagatzema aquest fitxer.

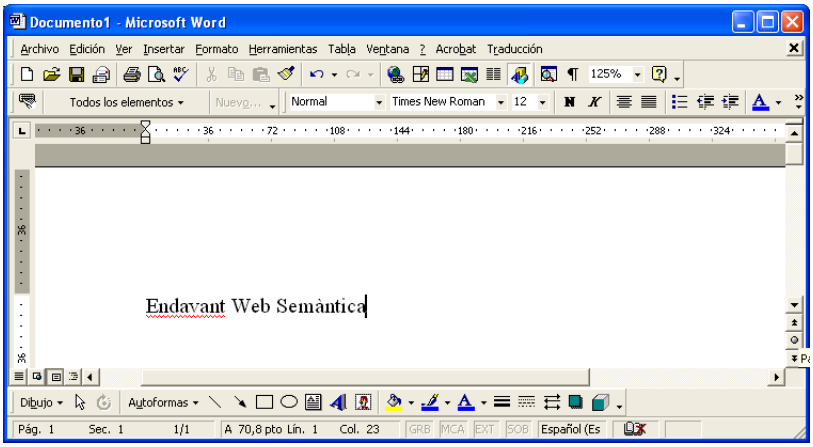

Figura 2.1: Document on-line en Microsoft Word.

Degut a l'ús d'una sintaxi i clara en les descripcions de les dades, XML és un llenguatge que pot ésser comprès per tothom, no solament per les aplicacions i persones que l'han generat. Aquest és un dels punts més importants per entendre la Web Semàntica (veure 3.1.1), ja que no poden preveure la varietat d'agents i sistemes que necessiten consumir dades en el *World Wide Web*. Un dels beneficis de emmagatzemar dades en XML, més que en format binari, és que pot fer cerques més fàcilment en les pàgines Web.

A continuació podem veure el document XML corresponent a l'anterior figura i que seria el que l'ordinador veuria:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE office:document-content PUBLIC
"-//OpenOffice.org/DTD Office-Document 1.0//en" "office.dtd">
<office:document-content xmlns:office
="http://openoffice.org/2000/office"
...

xmlns:script="http://openoffice.org/2000/script" office:class="text"
office:version="1.0">
<office:version="1.0">
<office:font-decls>
...
xmlns:script="http://openoffice.org/2000/script" office:class="text"
office:version="1.0">
<office:version="1.0">
<office:font-decls>
...
<style:font-decl style:name="Thorndale" fo:font-family="Thorndale"
style:font-family-generic="roman" style:font-pinch="variable"/>
</office:automatic-styles/>
<office:body>
<text:sequence-decls>
...
</text:sequence-decls>
<text:p text:style-name="Standard"> Endavant Web Semàntica!</text:p>
</office:body>
</office:document-content>
```

Figura 2.2: XML generat a partir del document de la figura anterior.

XML és una manera d'estandarditzar una estructura per manegar informació semàntica per als documents i els camps de dades. L'estructura d'XML utilitza una jerarquia d'arbre, que s'assembla a la estructura d'arbre de l'explorador de Windows. Aquesta estructura permet a l'usuari desglossar un concepte en els seus components de manera recursiva.

DATA	**META-DADES**
Albert Fernández	Nom
Arcadi Balaguer	Adreça
Castelldefels	Ciutat
Barcelona	Provincia

Taula 2.2: Comparació de les dades amb les metadades

Finalment cal remarcar que XML no és una tecnologia nova, es un subconjunt del llenguatge SGML[1] que va ser desenvolupant al 1969 pel Dr. Charles Goldfarb, Ed Mosher, i Ray Lorie. D'aquesta manera els conceptes del XML van ser plantejats fa més de 30 anys enrere i han estat contínuament perfeccionats, testejats i implementats.

2.2 Què és XML?

XML és un llenguatge extensible de marques que es defineix un format estàndard per a l'estructuració de dades i informaciò. L'XML va ser desenvolupat pel Grup de Treball XML l'any 1996, amb la supervisió de l'W3C[2] i va sorgir a partir de l'estàndard previ SGML (*Standard Generalized Markup Language*), molt més potent que XML però també més complex i difícil d'implementar.

Més que un llenguatge, XML és un *meta-llenguatge* que permet dissenyar llenguatges d'etiquetes per a diferents usos específics. Així doncs, XML no proporciona un conjunt fix d'etiquetes com en el cas d'HTML, sinó que permet crear les etiquetes que realment els desenvolupadors necessiten.

Tal com el món de la publicitat ha evolucionat cap al món de la electrònica, varis llenguatges s'han desenvolupat amb la filosofia de les marques, per exemple el TEX i el *Postscript* (veure la següent figura).

```
\documentsytle[doublespace, 12pt]{article}
\title{Exemple de Processament de Textes Electrònics}
\autor{R. Llasera}
\date{10 de Novembre de 2005}
\begin{document}
\maketitle
Això és un exemple d'un article.  Es pot escriure aquí dintre sense
haver delimitat el nombre de línies i l'espaiat.  LaTeX s'encarrega
de manegar això.
\end{document}
```

Figura 2.3: Exemple d'utilització de marques amb TEX

D'aquesta manera es pot dir que el principal principi de l'XML *és que les marques estan separades del contingut.*

Un element en XML és un contenidor XML que consisteix en un tag (excepte els elements buits que utilitzen un tag senzill), delimitant l'inici i

[1] Standardized Generalized Markup Language.

[2] World Wide Web Consortium, http://www.w3c.org. Va ser creat l'octubre de 1994 amb l'objectiu de potenciar el rendiment de la Web mitjançant el desenvolupament de protocols. Està format per més de 400 organitzacions de tot el món i està reconegut internacionalment per la seva contribució al creixement d'Internet.

el final de l'element. L'exemple següent mostra com es defineix un element:

```
<referencia>
<autor> F. J. Ceballos </autor>, <titol> Java </titol>
</referencia>
```

En aquest exemple es pot veure, un element anomenat "referencia", que conté 2 elements anomenats: "autor" i "titol".

Tipus de Tag	**Exemple**
Tag inicial	<autor>
Tag final	</autor>
Tag buït	<img />

Taula 2.4: Mostra de 3 tipus de tags XML

Un altre efecte és la divisió d'un document en parts semàntiques, per exemple podem dividir aquest capítol en <capitol>, <seccio> i <para> elements. La divisió en diverses parts d'una mateixa entitat ens permet classificar o agrupar en parts l'informació. En XML, aquesta classificació comença en la verificació de si un document és vàlid o no, segons la estructuració dels diferents elements que el composen. D'aquesta manera podem crear un jerarquia, o estructura d'arbre, per a cada document XML. Seguidament mostraré un exemple de la jerarquia d'un document XHTML:

```
<HTML>
  <HEAD>
    <TITLE> La meva pàgina </TITLE>
  </HEAD>
  <BODY>
    Endavant Web Semàntica
  </BODY>
</HTML>
```

Figura 2.4: Document senzill en HTML

Els elements buits els representem amb un tag *<img>*, que pot incloure un sèrie d'atributs, per exemple, *scr="poma.gif"*. Un element pot tenir més d'un atribut, seguidament es mostra un exemple d'element amb més d'un atribut.

```
<cotxe color="vermell" marca="Renault" model="Megane">
El meu cotxe
</cotxe>
```

La combinació d'elements i atributs formen l'XML, i això dona la possibilitat de que sigui un llenguatge preparat per modelar relacions i orientat a objectes.

2.3 Per què els documents tenen que ser ben formats i vàlids?

L'especificació XML defineix dos nivells de conformitat per als documents XML, aquests nivells són que els documents tenen que ser *ben formats i vàlids (la primera propietat és obligatòria i la segona opcional).*

- Un document *ben format* compleix amb totes les regles sintàctiques de l'*W3C* definides per a l'*XML*, i les qual es referència en els codis XML, per exemple: noms, nidació i claus dels atributs. Aquests requeriments ens garanteixen que un processador XML pot llegir un document sencer sense cap error. En el cas de que un processador XML trobi una violació, es a dir, de que el document no estigui ben format, es pararà el processat i es donarà un error en la aplicació que ha cridat al parser.

- Un document *vàlid* referència i satisfà un esquema. Un esquema (*Schema*) és un document on s'especifiquen els elements, atributs i la estructura d'un document XML. En general, podem veure un *Schema* com la definició d'un vocabulari legal, del nombre i de la ubicació d'elements en un llenguatge de marques. És per això de que un *Schema* defineix una classe particular de documents.

- Els processadors que compleixen les especificacions de l'W3C comproven sempre si un document és *ben format*, però no la seva *validesa*, que en alguns parsers XML és desactivada. El procés de *validació* consumeix molt de temps i no sempre és necessari, lo millor és fer aquesta validació només d'una part del document, o bé, immediatament després de la seva creació.

2.4 Què és el XML Schema?

XML Schema[3] és un llenguatge de definició que permet la validació de restriccions en un document XML a partir d'un vocabulari específic i d'una estructura jeràrquica. Les coses que normalment voldrem definir en el nostre llenguatge seran: tipus d'elements, tipus d'atributs, i la seva unió en tipus compostos (que s'anomenen *tipus complexos*). L'XML Schema és similar a l'esquema d'una base de dades, el qual defineix noms de

[3] L'Schema es torna un estàndard de l'W3C el 5 de Maig de 2001.

columnes i tipus de dades per les taules de la base de dades. Però no és l'únic llenguatge de definició, n'existeixen d'altres com *Document Type Definitions* (DTDs), *RELAX NG*, i *Schematron*.

- **DTD (Document Type Definition):** Va ser el primer llenguatge de definició i deriva del SGML. La seva sintaxis es defineix com a part de l'especificació de l'XML 1.0, que es va publicar el 10 de Febrer de 1998. Alguns llenguatges de marques encara utilitzen els DTD, però avui dia la majoria d'organitzacions i empreses han decidit canviar-ho per l'*Schema*. Les principals deficiències dels DTDs són que no utilitzen una sintaxis no XML, els seus tipus de dades limitats i el no suport dels *namespaces*. Aquestos són els 3 punts que l'XML Schema va solucionar.

- **RELAX NG**: És el principal competidor del XML Schema de l'W3C i està considerat tècnicament superior a l'XML Schema per alguns membres de la comunitat XML. Per altra banda, la majoria de venedors de programari, com *Microsoft* i *IBM*, han apostat per l'estandardització de l'XML Schema i per la solució de les deficiències que pugui tenir. *RELAX NG* representa la combinació d'aquestos esforços: RELAX i TREX. Aquesta és la definició de RELAX NG segons la seva especificació: *"Un RELAX Schema especifica un patró de l'estructura i del contingut d'un document XML. Per tant, un RELAX Schema identifica una classe de documents XML que compleixen un patró determinat. Un document RELAX NG és un document XML, que per la seva interoperatibilitat utilitza els tipus de dades de l'XML Schema".*

- **SCHEMATRON:** És una eina de validació de codi obert que utilitza una combinació de plantilles, regles, i referències a partir d'expressions *XPath* (veure 2.7) per la validació d'instàncies XML. El més remarcable d'aquest llenguatge de validació és que la validació basada en regles és una tècnica que es diferència de la validació basada en la gramàtica del document, que és la que utilitzen l'XML Schema i RELAX NG.

Com podem veure en la següent figura, tenim dos tipus de documents, un document de tipus *Schema* (o document de definició) i una sèrie d'instàncies de documents que formen *l'Schema*.

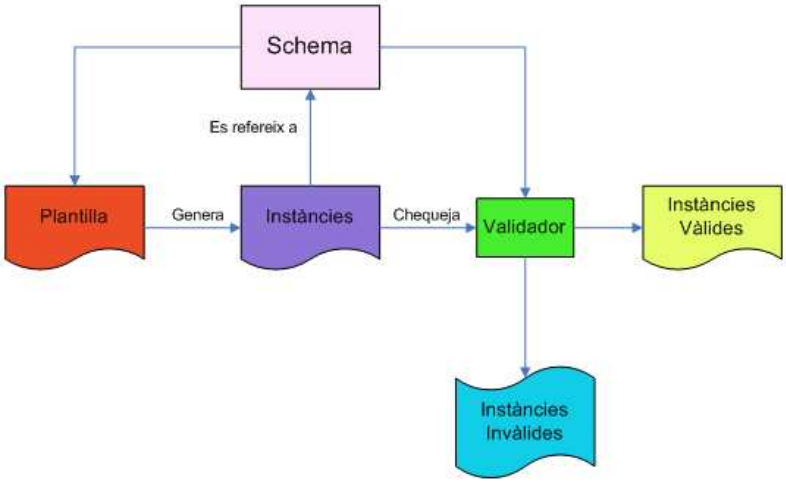

Figura 2.5: Schema i instàncies

2.4.1 A que s'assembla el XML Schema?

Un *XML Schema* utilitza una sintaxi XML per la declaració de *tipus* (types), ja siguin simples o compostos. Als tipus els anomenarem *template*s i poden contenir un o varis valors; en el cas del tipus *simples*, només poden contenir un valor. Els tipus *complexos* estan formats per múltiples valors simples i tenen dos claus característiques: un nom i un conjunt de valors.

Un tipus simple és una declaració d'un element que inclou les restriccions del seu nom i el seu valor. Seguidament mostraré un exemple d'un element anomenat "autor" que pot contenir qualsevol nombre de caràcters numèrics:

```
<xsd:element name="autor" type="xsd:string" />
```

L'anterior declaració ens permet que qualsevol document "instància" d'aquest pugui tenir elements com aquest:

```
<autor> Rafa Llasera </autor>
```

Tal com es pot apreciar en l'anterior exemple, el tipus de l'element declarat anteriorment és un *string*, el qual indica que és un seqüència de caràcters. Existeixen diversos tipus de tipus suportats per l'XML Schema, els quals es poden utilitzar en la definició d'un Schema per a un document determinat. En la següent taula podem veure els tipus existents:

Un *tipus complex* és un element que conté d'altres elements o que té atributs associats. Seguidament mostraré un exemple de la definició d'un element que té atributs associats:

Tipus de Dada	Descripció
string	Conjunt de caràcters Unicode d'una longitud específica.
boolean	Valor binari (true o false).
ID	Identificador únic d'un tipus d'atribut XML 1.0.
IDREF	Referència a un ID.
integer	Conjunt de nombres sencers.
long	Un long es deriva d'un integer, fixant els valors màxims i mínims..
int	Un int es deriva d'un long, fixant els valors màxims i mínims.
short	Un short es deriva d'un int, fixant els valors màxims i mínims.
decimal	Representa un nombre real.
float	Nombres expressats coma flotant, segons l'especificació simple de 32 bits.
double	Nombres expressats coma flotant, segons l'especificació doble de 64 bits.
date	Aquest tipus string es defineix en la norma ISO 8601.
time	Aquest tipus string es defineix en la norma ISO 8601.

Taula 2.6: Tipus primitius suportats per *l'XML Schema*

```
<xsd:element name="libre">
  <xsd:complexType>
    <xsd:attribute name="titol" type="xsd:string" />
    <xsd:attribute name="pagines" type="xsd:string" />
  </xsd:complexType>
</xsd:element name="libre">
```

D'aquesta manera podríem definir una instància de l'element *llibre* d'aquesta manera:

```
<book titol="Això és una prova" pagines="329">
```

Ara mostraré com definir un element anomenat "producte" amb tres atributs i dos elements fill. L'element producte tindrà tres atributs: identificador, títol i preu. També tindrà dos elements fill: descripció i categoria. Els elements categoria seran obligatoris i repetibles, i la descripció serà un element opcional:

```
<xsd:element name="producte">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="descripcio" type="xsd:string"
        minOccurs="0" maxOccurs="1" />
      <xsd:element name="categoria" type="xsd:string"
        minOccurs="1" maxOccurs="unbounded" />
    </xsd:sequence>
    <xsd:attribute name="id" type="xsd:ID" />
    <xsd:attribute name="titol" type="xsd:string" />
```

```
      <xsd:attribute name="preu" type="xsd:decimal" />
    </xsd:complexType>
  </xsd:element>
```

i un exemple d'una instància XML de l'element producte, seria el següent:

```
<producte id="P01" titol="nino Epi" preu="40.77">
  <descripcio>
  El nino Epi ha estat el més venut de l'any.
  </descripcio>
  <categoria> joguines </categoria>
  <categoria> nino de Barri Sèsam </categoria>
</producte>
```

Les definicions de l'XML Schema poden ser molt complexes i poden requerir ésser construïdes per experts, per això algunes empreses han escollit ignorar fer validacions dintre del seu programari.

2.4.2 El problema de la validació

Qualsevol que ha treballat amb eines de validació sap que fins que els llenguatges existents per definir esquemes (Schema) madurin, el procés de *validació* és bastant frustrant que requereix el testeig de moltes eines. Afortunadament, el suport de les eines de validació és, avui dia, bastant bo i som capaços de validar correctament esquemes bastant complexos.

El procés de validació és crític perquè XML, per la seva naturalesa, és utilitzat i compartit per un gran nombre d'aplicacions informàtiques. Qualsevol instància d'un document XML tindria que ser validada durant la seva creació per a garantir la concordança entre tipus de dades i per a garantir la funcionalitat.

2.5 Què són els XML Namespaces?

Els *namespaces* són un mecanisme simple per a la creació de noms globals i únics per als elements i atributs del nostre llenguatge. Són importants per dues raons:

- Per a simplificar el problema d'utilitzar noms idèntics en diferents llenguatges de marques.
- Per a permetre ajuntar diferents llenguatges de marques sense que hi hagi ambigüitat.

Desafortunadament, els *namespaces* no són completament compatibles amb els DTDs i per això la seva difusió pot esser lenta. Els llenguatges de marques actuals, com XML Schema, suporten completament els *namespaces*.

```
<xsd:integer>
```

Els *namespaces* s'implementen a partir de dos parts: una prefix i una part local. La part local és l'identificador per les metadades (en l'anterior exemple, la parta local era "*integer*"), i el prefix és una abreviació de l'actual *namespace* en la seva declaració. L'actual *namespace* és un únic *URI*[4].

A continuació mostraré un exemple de la declaració d'un *namespace:*

```
<xsd:schema xmlns:xsd="http://www.w3.org/XMLSchema">
```

És important entendre que el *prefix* no és mateix que el *namespace*. Per a especificar el namespace dels nous elements que s'està definint, utilitzarem el atribut *targetNamespace*:

```
<xsd:schema xmlns:xsd="http://www.w3.org/XMLSchema"
  targetNamespace="http://www.laempresa.com/markup">
```

Existeixen dues maneres d'aplicar un *namespace* a un document: associar el prefix a cada element i atribut en el document o declarar un *namespace* per defecte per al document. S'aconsegueix un *namespace* per defecte eliminant el prefix de la seva declaració:

```
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>
      Testeig del namespace per defecte
    </title>
  </head>
<body> Endavant Web Semàntica </body>
</html>
```

Seguidament es mostra com l'anterior document es tradueix internament per un processador XML (fixem-nos que d'ús de claus per a separar els *namespaces* és una manera bona de separar els namespaces de la part local).

```
<{http://www.w3.org/1999/xhtml}html>
<{http://www.w3.org/1999/xhtml}head>
<{http://www.w3.org/1999/xhtml}title> Test namespaces
</{http://www.w3.org/1999/xhtml}title> </head>
<{http://www.w3.org/1999/xhtml}body> Endavant Web Semantica!!
</{http://www.w3.org/1999/xhml}body>
</{http://www.w3.org/1999/xhtml}html>
```

[4] *Uniform Resource Identifier* (URI), és una manera estàndard d'identificar un recurs. L'URI és un terme genèric que fa referència a adreces i objectes del *World Wide Web*.

Aquest processat en produeix durant el "*parsejat*" que porta a terme una aplicació. El "*parsejat*" és la dissecció en blocs de texte en paraules interpretables (tokens). Existeixen 3 maneres diferents de "parsejar" un document XML: usar el API d'XML (SAX), construir un *Document Object Model* (DOM), i una tècnica nova anomenada *pull parsing*.

2.6 Què és el DOM (Document Object Model)

El *Document Object Model* (DOM) és un llenguatge neutral per a models de dades i per la programació de l'API que permet l'accés i la manipulació de XML i HTML. A diferència de les instàncies XML i dels *Schema d'XML*, els quals resideixen en el disc, el DOM permet una representació d'un document en memòria.

La programació orientada a objectes (POO) introdueix dos conceptes claus que utilitzarem en el llenguatge RDF: les classes i els objectes. Una *classe* és una definició o una plantilla que descriu les característiques i propietats que té una entitat o concepte. A partir d'aquesta definició, podem construir una instància en memòria d'una classe, la qual s'anomena objecte. Un *objecte* és una instància específica d'una classe.

La manera més simple de pensar en DOM és com un conjunt de classes que permet crear arbres d'objectes en memòria que representen una versió de documents XML o HTML.

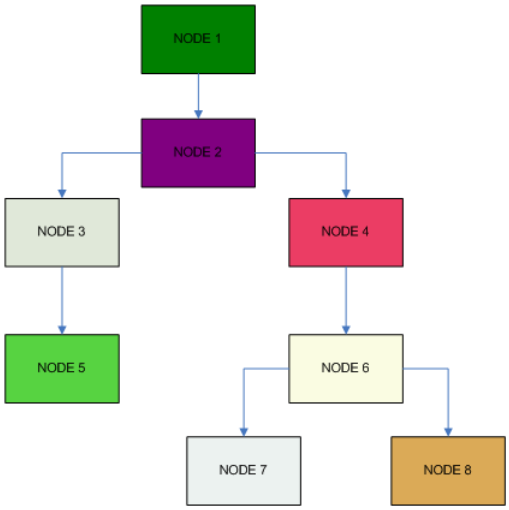

Figura 2.6: Arbre de nodes DOM

Un arbre DOM pot ser accedit mitjançant subclasses específiques dels nodes i pot ser explorat amb més detall afegint nous mètodes de manipulació a les classes.

2.7 XPath

XPath és un llenguatge molt important dintre de la família d'estàndards XML. És un llenguatge que permet accedir a les diferents parts d'un document XML. La importància de *XPath* radica en que aporta possibilitats semàntiques, de sintaxis, i de funcionalitat per una gran varietat d'estàndards com *XSLT*, *XPointer*, i *XQuery*. Utilitzant les expressions d'XPath amb alguns *frameworks* i APIs és possible fer referència i trobar components individuals dintre d'un document XML.

L'especificació W3C de 1999 d'*XPath 1.0* es va desenvolupar juntament amb el grup de treball *XSL* del W3C i amb el grup de treball *XML Linking Working Group*. A més de la funcionalitat de direccionar o accedir a àrees determinades d'un document XML, ens aporta facilitat de manipulació d'strings, nombres o booleans. XPath utilitza una sintaxis compacta per a facilitar el seu ús amb *URIs* i atributs XML.

Utilitzar XPath combinat amb altres estàndards XML és un sistema eficient i s'utilitza en moltes eines del mercat actual.

Expressió XPath	**Que significa?**
//TascaItem[@id]	Donam tots els 'TascaItem' que tenen atributs ID.
//[@id]	Donam tots els atributs 'ID'.
/Tasca/Trobada	Sel.lecciona elements 'Trobada' fills de 'Tasca'.

Taula 2.8: Exemples d'expressions XPath.

Per exemple, podem definir amb XSLT una plantilla i mitjançant expressions *Path* fer consultes sobre bases de dades XML i també sobre un gran nombre d'arxius XML.

Finalment, es pot dir que *XPath* ha tingut un gran èxit, la seva expansió ha estat molt gran, juntament amb altres estàndards XML en un gran nombre d'aplicacions tecnològiques.

2.8 La familia dels fulls d'estil: XSL, XSLT i XSLFO

Els fulls d'estil ens permeten especificar com un document XML pot ser transformat en nous documents, i de com aquest document XML pot ser presentat en diferents formats. Els llenguatges associats amb els full d'estil són:

- ***XML:*** *Extensible Stylesheet Language.*
- ***XSLT:*** *Extensible Stylesheet Language Transformations.*
- ***XSLFO:*** *Extensible Stylesheet Language Formating Objects.*

Un processador de fulls d'estil agafa un document XML i un full d'estil i dona un resultat. XSL consisteix en dos parts, d'una banda ens dona un mecanisme per a transformar documents XML en nous documents XML (XSLT), i ens aporta un vocabulari per a la transformació d'objectes (XSLFO).

XSLT és un llenguatge de marques que utilitza les regles definides en plantilles, i que especifiquen com un processador de fulls d'estil transformat un document, i és una recomanació del W3C[5]. XSLFO és un llenguatge de marques de paginació i de formatejat definit a la recomanació XSL del W3C[6].

És important entendre la rellevància de l'estil en les pàgines. Utilitzar fulls d'estil permet una representació de les dades XML. Separant el context (dades XML) de la seva presentació (el full d'estil), s'obté el paradigma del *MVC*[7]. El fet de separar les dades (el model), com les dades es visualitzen (la visió), i els recursos ens donen la possibilitat potenciar la reutilització dels mateixos. *Microsoft Internet Explorer* té una sèrie de processadors que permeten tractar les dades XML al mateix temps que són descarregades.

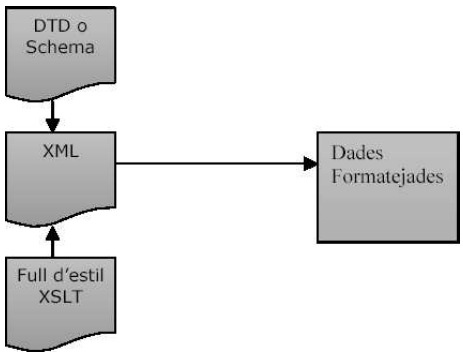

Figura 2.7: Estil d'un document

Seguidament mostraré un exemple de l'ús de full d'estil per afegir presentació al context.

[5]http://www.w3.org/TR/xslt/
[6]http://www.w3.org/TR/xsl/.
[7]Model-View-Controller (MVC).

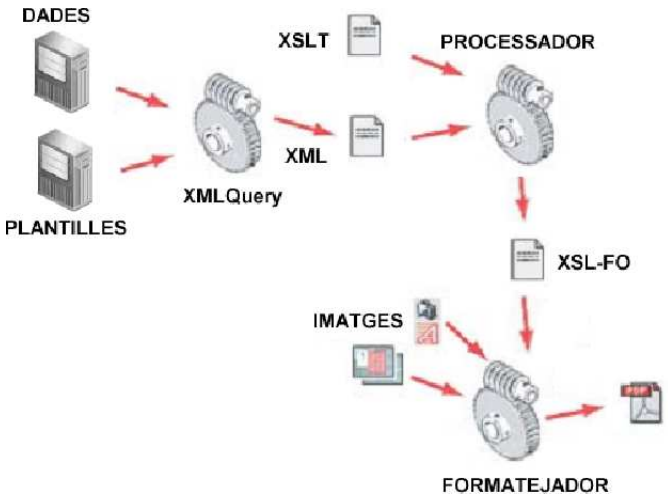

Figura 2.8: Exemple utilització d'un full d'estil en la generació de PDF's.

Per a crear un pàgina HTML amb l'informació del nostre arxiu XML, necessitem escriure un full d'estil. Totes les cerques de plantilles es fan mitjançant expressions *XPath*. Per exemple, l'element *<xls:value-of>* retorna el valor dels items escollits d'una expressió *XPath*.

2.9 XQuery

XQuery és un llenguatge dissenyat per al procés de dades XML, i es defineix com un llenguatge de consulta fàcil per a bases de dades XML, i es pot dir que qualsevol expressió XPath (2.7) és un subconjunt d'aquest llenguatge. El *Query Working Group* del W3C ha fet que sigui fàcil de llegir per als humans degut a la seva sintaxis.

Tal com usar les transformacions dels fulls d'estil, el *Query* utilitza expressions *XPath* i les seves pròpies funcions per a la manipulació de dades XML, la seva sintaxis és similar a la del Perl. Per a demostrar com funciona XQuery, seguidament mostraré un exemple.

```
let $project := document{"projecte.xml"}/project
let $day := $project/schelude/workday
return $day sortby {description}
```

L'anterior expressió retorna els dies de treball (workday) d'un arxiu, ordenats alfabèticament per descripció.

2.10 XLink

XLink (*XML Linking Language*) és el llenguatge que defineix la sintaxis que, basada en atributs XML, permet la creació d'enllaços entre dos o més

documents. Aquests documents no han de ser obligatòriament documents XML, ja que un enllaç situat en un document XML pot enllaçar amb un altre document XML, o bé, a d'altres tipus de documents no XML (JPG's, documents PDF, documents de text, etc.).

Un link es defineix en la *XLink W3C recommendation* com "una relació entre recursos o fragments dels mateixos". Amb XLink no sols es poden referenciar documents externs, també es poden relacionar elements amb d'altres elements, i també es poden enllaçar les relacions entre els mateixos. En la figura 2.9 podem veure un exemple senzill d'utilització d'XLink:

```
<?xml version="1.0" encoding="UTF-8">
<doc>
<name xlink:type=·'simple'
xlink:href=='instructors/conductordebus.xml'>Pere Cremades</name>
és un empleat a
  <companyia xlink:type="simple" xlink:href='#afina'>
    Afina S.A
  <companyia>, i ensenya
  <course xlink:type='simple' xlink:href='#CS593'>
    Informatica 593
  </course>
  <empreses>
    <companyia id='afina'>Afina S.A</companyia>
    <companyia id='tribugest'>Tribugest S.A</companyia>
    <companyia id='ids'>Informatica del Segre S.L</companyia>
  </empreses>
  <llistadecursos>
    <curs id='CS141'>CS 141 - Introduccio a SCI</companyia>
    <curs id='CS593'>Sistemes Operatius</companyia>
    <ccurs id='CS669'>Disseny Orientat a Objectes</companyia>
  </llistadecursos>
</doc>
```

Figura 2.9: Exemple d'utilització d'XLink

També d'altres llenguatges com SVC (*Scalable Vector Graphics*) i *MathML* utilitzen *XLink* per a crear referències *hypertext*.

2.11 XPointer

XPointer especifica una sintaxis no XML que permet identificar i fer referències a punts o fragments que es troben dins d'un document XML. La sintaxis de XPointer està basada en la sintaxis del XPath, que permet navegar per l'estructura lògica que forma un document XML.

Conceptualment, l'objectiu que persegueix XPointer és similar al de l'etiqueta <a> de l'HMLT, que ens permet identificar un cert punt dins d'un document HTML i posteriorment, fer referència a aquest punt, afegint el

fragment "#*nom_etiqueta*" a la URL del document.

Seguidament es mostra un exemple de l'ús de *XPointer* que complementaria l'exemple 2.10.

```
<extendedlink xlink:type='extended'>
  <loc xlink:type='locator' xlink:label='afina'
    xlink:href='#xpointer(//company[@id='afina'])'></loc>
  <loc xlink:type='locator' xlink:label='cs593'
    xlink:href='#xpointer(//company[@id='cs593'])'></loc>
  <loc xlink:type='locator' xlink:label='cs593descripcio'
    xlink:href='cursos/cs593.xml'></loc>
</extendedlink>
```

Figura 2.10: Exemple d'utilització d'XPointer

L'anterior exemple referència el nom de l'empresa <empresa> amb el ID 'afina'.

2.12 XInclude

Les inclusions XML (XInclude) serveix per incloure documents, elements i d'altra informació dintre d'un document XML. Mitjançant *XInclude* podem crear grans documents a partir de fragments més petits, documents *ben formats* o documents no XML. En el següent exemple podem veure com un document es pot construir a partir d'altres:

```
<?xml version="1.0">
<chapter xmlns:xi="http://www.w3.org/2001/XInclude">
  <title>Entenent l'alfabet Soup</title>
  <xi:include href="http://www.afina.net/SemWeb/ch6/xpath.xml"/>
  <xi:include href="http://www.afina.net/SemWeb/ch6/stylesheets.xml"/>
  <xi:include href="http://www.afina.net/SemWeb/ch6/xquery.xml"/>
  <xi:include href="http://www.afina.net/SemWeb/ch6/xlink.xml"/>
</chapter>
```

Figura 2.11: Exemple d'utilització d'XInclude

2.13 XHTML

XHTML (*Extensible Hypertext Markup Language*) és la reformulació de XML dintre de XML. Aquesta especificació va ser creada amb el propòsit de donar major potència de processat a la Web actual. Encara que HTML fàcil d'escriure, aquesta estructura és desfasada en els nostre camí cap a la *Web Semàntica*. HTML no és un bon llenguatge per a descriure dades i tampoc per a ser usat en la Web Semàntica.

XHTML és XML i ens dona una estructura i unes possibilitats que permeten la inclusió d'altres llenguatges basats en XML i de *namespaces*. XHTML 1.0 és una recomanació de l'W3C (gener del 2000) que és una reformulació del HTML 4.0 dintre de XML. La transició de l'HTML cap a XHTML és molt fàcil. Aquests són alguns dels punts més importants:

- Un document XHTML 1.0 té que ser declarat com un document XML que fa servir una declaració XML.
- Un document XHTML 1.0 és vàlid i ben format. Té que contenir un *DOCTYPE* que diu que és un document *HTML 1.0*, i també diu quin *DTD* es fa anar en el document.
- Cada tag té que ser tancat.
- L'element arrel d'un document XHTML 1.0 és <*html*> i té que contenir un *namespace* que l'identifiqui con un document XHTML.
- Com XML és sensible a les majúscules, els elements i atributs XHTML tenen que estar en minúscules.

Capítol 3

El Resource Description Framework (RDF)

En aquest capítol d'introduirà al lector en el món del *Resource Description Framework (RDF)*, es parlarà de l'*RDF Schema* i de com serveix per modelar classes, també es escriuran les principals diferències existents entre el XML Schema i el RDF Schema ([2]). He considerat necessari també l'inclusió d'una sèrie de conceptes previs, extrets del llibre [15], per a poder entendre el context de l'RDF, començaré el capítol introduint aquestos conceptes.

3.1 Conceptes Necessaris

3.1.1 Què és la Web Semàntica?

El fet de que l'ordinador no sàpiga que significa l'informació limita molt la seva capacitat, i provoca que sigui poc viable automatitzar mitjançant programari les cerques que habitualment efectuen les persones, ja que aquesta feina és molt complexa i molt difícil de reproduir i mantenir, tal com està actualment codificada la Web.

Així, per superar aquesta limitació, va sorgir un nou concepte anomenat Web Semàntica. La Web Semàntica parteix de la idea bàsica de que l'informació pugui ser utilitzada i compresa pels ordinadors sense la necessitat de supervisió humana, de manera que el programari web pugui ser dissenyat per tractar la informació de les planes web de manera semi-automàtica. Es tracta de convertir la informació que contenen les planes web en coneixement.

La Web Semàntica manté els principis que han portat a l'èxit a la Web actual, i recupera el concepte d'ontologia del camp de la Intel.ligència Artificial, com a vehicle per a dur a terme un objectiu clau: *aconseguir una entesa entre les parts que han d'intervenir en la construcció i explotació de la Web*. La Web Semàntica està formada per nodes, igual que la Web ac-

tual, però en aquest cas estan clarament tipificats, i cada node o recurs té un tipus que l'identifica inequívocament. Els nodes estan units per enllaços que representes les relacions que hi ha entre ells i també estan explícitament diferenciats.

3.1.2 Les ontologies

Les ontologies són àmpliament usades en l'Enginyeria del Coneixement, la Intel.ligencia Artificial i la Informàtica, en aplicacions que fan referència a la manipulació del coneixement, al processament del llenguatge natural, al comerç electrònic, al disseny de bases de dades i l'integració, l'educació, i en nous camps emergents, com la Web Semàntica.

Al 1991, la "DARPA Knowledge Sharing Effort" estudia la manera de dissenyar una nova estratègia de crear sistemes intel.ligents ([15]). Ens proposen el següent:

"Construir sistemes que es fonamenten en el coneixement avui dia és una tasca que té que tenir unes bases noves i pot ser entesa com la unió de nous components. Els desenvolupadors de sistemes sols necessiten preocupar-se de crear nous mètodes o coneixement i raonament d'una nova tasca específica del sistema. Aquest nou sistema ha d'interoperar amb els sistemes existents, fent-los servir per a fer part d'aquest raonament.

El coneixement declaratiu és modelat pels significats de les ontologies i que els mètodes de resolució de problemes especifiquen mecanismes genèrics de raonaments. Ambdós tipus de components poden ser vistos com entitats complementàries que poden ser usades per a configurar nous sistemes que es fonamenten en el coneixement de components existents.

3.1.3 Definició d'una ontologia

La paraula ontologia ha estat presa de la filosofia, en la darrera dècada s'ha tornat rellevant en la comunitat d'Enginyeria del Coneixement (*Knowledge Comunity*).

Una ontologia és una especificació formal i explícita d'una conceptualització compartida. Una conceptualització fa referència a un model abstracte d'alguns fenòmens en el món que identifiquen els conceptes d'aquest fenomen. Explícit significa que el tipus de conceptes usats i les seves restriccions s'especifiquen detalladament. Formal referència el fet de que un ontologia pot ser llegida per les màquines. Compartida reflexa la noció de que un ontologia captura el coneixement consensual, que no és quelcom aïllat sinó que és acceptat per un grup ([15]).

3.1.4 Representació del coneixement amb ontologies

Una ontologia de representació del coneixement (KR) agafa les primitives de modelat per a representar el paradigma del coneixement. La més representativa ontologia de representació del coneixement és la *Frame Ontology* construïda per a capturar el coneixement mitjançant la representació amb *frames*, i s'utilitza a *l'Ontolingua*.

D'altres llenguatges per a representar ontologies com CycL (Lenat i Guha, 1990) i OCML (Motta, 1999) tenen també el seu propi model de representació (KR). Més recentment, els llenguatges de marques per a la representació d'ontologies s'han creat en el context de la Web Semàntica: *RDF (Lassila i Swick, 1999)* i *RDF Schema (Brickley i Guha, 2003)*, *OIL (Horrocks i al., 2000) DAML+OIL (Horrocks i van Harmelen, 2001)* i *OWL (Dean i Schreiber, 2003)*. Tots aquestos llenguatges tenen el seu corresponent KR:

3.1.5 Eines per la creació d'ontologies

Construir ontologies és un procés complexa i que consumeix molt de temps, i més si les ontologies es construeixen directament amb un llenguatge de construcció d'ontologies, sense el suport de cap eina. Per a facilitar aquest procés en els anys 90 es crea la primera d'aquestes eines. En els darrers anys el nombre d'eines d'aquest tipus s'ha multiplicat, segons el llibre [15] es distingeixen els següents tipus d'eines:

- *Eines per a desenvolupament d'ontologies:* Aquest grup inclou les eines i suites que poden ser usades per a la construcció d'ontologies. Aquestes eines ens donen suport per a la documentació, importació/exportació en diferents formats, edició gràfica, tractament de llibreries, etc.

- *Eines d'avaluació:* S'utilitzen per a avaluar el contingut de les ontologies i les seves tecnologies.

- *Eines per la fusió i alineament:* S'utilitzen per a solucionar el problema de la fusió i alineament d'ontologies dintre d'un mateix domini.

- *Eines d'anotació d'ontologies:* Amb aquestes eines els usuaris poden inserir instàncies de conceptes i de relacions i mantenir semi-automàticament les marques d'ontologies en pàgines Web.

- *Eines per a la consulta d'ontologies i sistemes d'inferència:* Permeten la consulta d'ontologies fàcilment i fer inferència amb elles. Normalment, estan molt lligats al llenguatge que s'ha utilitzat per a implementar les ontologies.

- *Eines d'aprenentatge d'ontologies:* Poden construir ontologies de manera semi-automàtica a partir de textos escrits en llenguatge natural a partir de tècniques d'anàlisis.

Alguns exemples d'aquestes eines són: KAON (Maedche, 2003), OilEd (Bechhofer, 2001), Ontolingua Server (Faquhar, 1997), OntoSaurus (Sqartout, 1997), Protégé-2000 (Noy, 2000), WebODE (Arpírez, 2003), i WebOnto (Domingue, 1998).

3.2 Què és RDF?

Simplificant-ho al màxim, podem dir que RDF[1] és un llenguatge que es fonamenta amb XML i que serveix per a descriure recursos (resources, [2]). La descripció d'un recurs pot ser molt àmplia, però centrem-nos primer amb veure un recurs com un document electrònic disponible mitjançant la Web. Un recurs es localitza via una URL[2]. Mentre els documents XML afegeixen metadades a parts d'un document, l'ús d'RDF és crear metadades sobre un document com si aquest fos una entitat autònoma. En altres paraules, RDF captura les metadades de les característiques d'un document, com per exemple, l'autor, la data de creació, el tipus, etc.

Un ús particularment bo de l'RDF és el descriure recursos que són "opacs" com imatges de arxius d'àudio. L'aplicació *RDFPic* és una aplicació de mostra desenvolupada per l'W3C per a afegir metadades RDF dintre d'imatges JPEG. Aquesta aplicació pot treballar conjuntament amb el servidor *Jigsaw Web Server* de l'W3C per a extreure automàticament les metadades RDF emmagatzemades dintre de les imatges. Els dos esquemes (*Schemas*) disponibles per a descriure una imatge són els elements del *Dublin Core* (*www.dublincore.org*) i l'esquema tècnic amb propietats de les metadades.

Avanç d'afegir les metadades en, per exemple, una foto, es poden exportar en un arxiu extern, com es mostra en el següent llistat (3.1):

[1] Resource Description Framework (RDF).
[2] Uniform Resource Locator.

```
<?xml version='1.0' enconding='ISO-8859-1' ?>
  <rdf:  RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#"
    xmlns:s0="http://www.w3.org/2000/PhotoRDF/dc-1-0#"
    xmlns:s1="http://sophia.inria.fr/~enerbornn/rdfpiclang#"
    xmlns:s2="http://www.w3.org/2000/PhotoRDF/technical-1-0#"
    <rdf:Description rdf:about="http://www.c2i2.com/~budstv/images/c1.jpg">
      <s0:relation>part-of Store Front</s0:relation>
      <s0:type>image</s0:type>
      <s0:format>image/jpeg</s0:format>
      <s1:xmllang>en</s1:xmllang>
      <s0:description>Foto de l'Àlex a la seva casa</s0:description>
      <s2:camera>Kodak EasyShare</s2:camera>
      <s0:title>Foto de l'Àlex a la seva casa</s0:title>
    </rdf:Description>
  </rdf:RDF>
```

Figura 3.1: RDF generat amb l'RDFPic

La primera cosa que cal remarcar del llistat 3.1 és l'ús consistent de *namespaces* que es fa. En l'element arrel *<rdf:RDF>* es declaren quatre *namespaces.* L'element arrel especifica que aquest és un document RDF. Un document RDF conté una o més "descripcions" de recursos. Una *descripció* és un conjunt de característiques sobre un dispositiu. L'element *<rdf:Description>* conté un atribut *<rdf:about>* que referència al recurs que s'està descrivint. En el llistat 3.1, el atribut *<rdf:about>* apunta cap a la URL[3] d'una imatge JPEG anomenada *c1.jpg*. L'atribut rdf:about és molt important de que sigui entès per a comprendre com funciona RDF, ja que tots els recursos descrits amb RDF tenen que ser localitzats amb un URI[4]. Els elements fills de l'element *Description* són totes les propietats de l'element que s'està descrivint. Així doncs, acabem de veure la manera de descriure un recurs, com un conjunt de propietats i de valors de les mateixes. Aquest model de tres parts es separa de la sintaxis RDF.

El model RDF s'anomena freqüentment model de tripletes (*triples*) ja que té tres parts, com he descrit anteriorment. Amb els termes de les propietats dels recursos descrites anteriorment, i segons el model de representació del coneixement, podem dir que existeixen tres parts ben diferenciades: *subjecte, predicat i objecte*. En la figura podem veure aquests elements que composen el model en tres parts i la seva simbologia associada:

[3]Universal Request Locator.

[4]Com es va mencionar en el capítol *2, l'Uniform Resource Identifier* (URI), és una manera estàndard d'identificar un recurs. L'URI és un terme genèric que fa referència a adreces i objectes del *World Wide Web*.

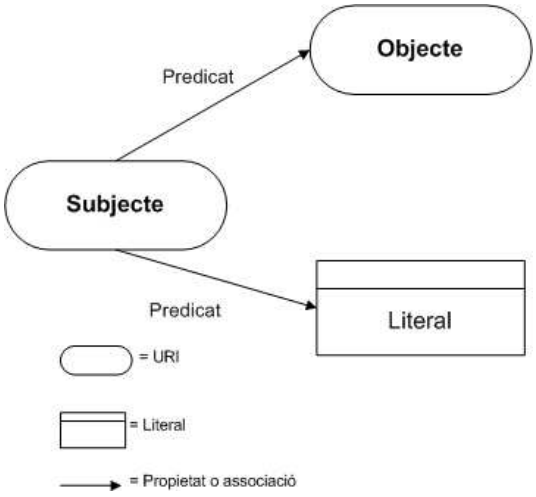

Figura 3.2: Tripleta RDF

Els elements clau de les tripletes RDF són:

Subjecte: Gramaticalment, el subjecte és el que fa l'acció. En RDF, aquest recurs[5] el descriuen l'objecte i el predicat de la tripleta. Per tant, en RDF, per exemple, necessitem un URI per a descriure el concepte únic *"empresa"* tal com *"http://www.empresa.org/ontology/#empresa"* per a descriure una relació.

Predicat: Gramaticalment, és una part d'una sentència que modifica el subjecte i que inclou un verb a l'oració. En RDF, un predicat és una relació entre un subjecte i un objecte. Per tant, en RDF, podríem definir un únic URI per al concepte *"ven"* com: *"http://www.empresa.org/ontology/#ven"*.

Objecte: Gramaticalment, és a qui es refereix l'acció del verb. En RDF, un objecte és el recurs referenciat, ja sigui per un predicat o per un valor literal. En el nostre exemple, podríem definir un únic objecte *"bateria"* de la següent manera: *"http://www.business.org/ontology/#bateria"*.

Sentència: En RDF, és la combinació dels tres elements anteriors: subjecte, predicat i objecte com una única unitat. La figura mostra la representació d'un graf amb dos sentencies RDF. Aquestes dos sentencies il.lustren els conceptes de la figura .

[5]Un recurs RDF pot ser entès com un recurs electrònic, tal com arxius, o conceptes. Una manera de pensar en un recurs RDF és : *"qualsevol cosa que té una identitat"*.

3.3 Capturant el coneixement amb RDF

Existeix un ampli consens en que la representació en tripletes és més simple que el format RDF/XML., el qual s'anomena "*format de serialització*". A conseqüència d'això han aparegut una gran varietat de formats més simples que són capaços de capturar el coneixement de manera ràpida mitjançant la representació en tripletes.

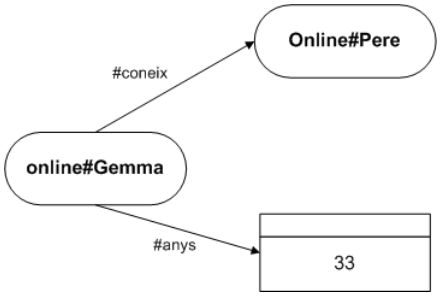

Figura 3.3: Graf amb dues sentències RDF

A la notació en tripletes se l'anomena *N3*. Seguidament mostrarem quatre maneres diferents de representar els conceptes: en sentències del llenguatge natural, en notació N3, en format de serialització RDF/XML, i, finalment, com un graf de tripletes.

Seguidament es mostra el model lingüístic de subjecte, predicat i objecte. Per exemple:

```
La Gemma té una empresa
La empresa té un Web accessible via: http://www.c2i2.com/~gemma
La Gemma és la mare d'en Pere
```

A continuació es mostra l'anterior sentència representada amb format N3:

```
<#Gemma> <#te> <#negoci>
<#negoci> <#te-website> <http://www.c2i2.com/~gemma>
<#Gemma> <#mare-de> <#Pere>
```

Com es pot apreciar, per a cada sentència hem extret el subjecte, el predicat i l'objecte. El signe # significa que el URI dels conceptes pot ser l'actual document. Aquest símbol és un *shortcut* posat per abreviar, és millor substituir el símbol # per un URI absolut, com: *"http://www.c2i2.com/gemma/ontology"*. En N3 podem fer això amb un tag *prefix*, com aquest:

```
@prefix bt: <http://www.c2i2.com/gemma/ontology>
```

D'aquesta manera podem utilitzar prefixes en els nostres recursos d'aquesta manera:

```
<bt:gemma> <bt:te> <bt:empresa>
```

Les eines d'avui dia poden convertir automàticament la notació N3 en format RDF/XML. Una de les eines més populars és el *"Jena Semantic Web toolkit" (veure 5.2.4.2)* desenvolupat als laboratoris de *Hewlett-Packard*. A continuació mostro el resultat de convertir l'exemple anterior:

```
<rdf:RDF>
  xmlns:RDFNsId1='#'
  xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
  <rdf:Description rdf:about='#Gemma'>
    <RDFNsId1:te>
      <rdf:Description rdf:about='#empresa'>
        <RDFNsId1:te-website
          rdf:resource='http://www.c2i2.com/~gemma'>
      </rdf:Description>
    <RDFNsId1:  mare-de rdf:resource='#Pere" />
  </rdf:Description>
</rdf:RDF>
```

Figura 3.4: RDF/XML generat a partir de N3

La primera cosa que hem de tenir en compte és la sintaxis RDF/XML, una sentència RDF es posa dins d'un altra. La segona és com els predicats són representats, per exemple: *RDFNsId1:te* o *RDFNsId1:mare-de*. Els objectes es representen pel *rdf:resource* o pel valor del literal.

3.4 Altres característiques de l'RDF

La resta de les característiques de l'RDF el que fan és incrementar la potència en la creació de sentències. Les dues principals categories que fan això són: el model de contenidor simple i la *"reification"* (fer sentències de sentències). El model de contenidor permet grups de recursos o valors. La *"reification"* permet construir sentències d'alt nivell per a capturar el coneixement d'altres sentències.

L'RDF, és doncs, un model formal per a la representació de les propietats i dels valors d'aquestes propietats[6]. L'RDF utilitza els namespaces XML per a qualificar els recursos. Això soluciona el problema de les col.lisions de noms que es podrien donar quan es combinen múltiples vocabularis. Els *namespaces* també s'utilitzen per a identificar l'esquema RDF.

El model de dades bàsic consisteix en tres tipus d'objectes:

[6]W3C World Wide Web Consortium. Resource Description Framework (RDF): Model and Syntax Specification. W3C Recommendation, 22 February 1999. [Ora Lassila i Ralph R. Swich, eds]. Disponible a: http://www.w3.org/TR/1999/REC-rdf-syntax-19990222.

- **Recursos:** Tot allò que pot ser descrit per una expressió RDF és un "recurs". Pot ser una pàgina web, o una part d'una pàgina web, una col.lecció sencera de pàgines o un site web.
- **Propietats:** Una propietat és una característica, aspecte, atribut o relació usada per a descriure un recurs. Cada propietat té un significat específic, defineix els seus valors possibles, els tipus de recursos que pot descriure i la seva relació amb altres propietats.
- **Sentències:** Un recurs juntament amb una propietat amb nom i amb el valor de la propietat per a aquest recurs s'anomena *sentència*. Les *sentències* representen relacions binàries específiques entre els objectes. A les tres parts d'una sentència se les anomena: subjecte (recurs), predicat (propietat) i objecte (recurs o literal). El objecte pot ser un altre recurs, o bé, el valor d'una propietat.

3.5 Per què RDF no és més conegut?

El *Resource Description Framework* va ser una recomanació (sinònim d'estàndard) el 22 de Febrer de 1999, solament un any desprès de que surtis la versió 1.0 d'XML. Molta gent no ha sentit parlar mai de l'RDF, existeixen diverses raons:

- *RDF no s'ajusta bé amb els documents XML:* De moment, no és possible validar el RDF que hi ha en altres documents XML o XHTML degut a que RDF té una gramàtica oberta.
- *Parts de l'RDF són complexes:* Els documents RDF són més complexos que els documents XML. Els factors més destacats són: la fusió de metàfores, la sintaxis de serialització i la "reificació".
- *No es ressalta l'RDF:* Les aplicacions com *Dublin Core (DC)* i *RDF Site Summary (RSS)* no ressalten les característiques de l'RDF. Els elements Dublic Core són útils definicions de recursos, però no són exclusives de l'RDF i poden ser utilitzades amb documents HTML o XML.

La majoria dels autors escriuen les sentències RDF en format N3 i desprès les converteixen de N3 a la sintaxis RDF/XML mitjançant un eina de conversió (per exemple: el Jena N3).

3.6 Què és el RDF Schema?

L'RDF Schema (RDFS) és un llenguatge situat per sobre de RDF. Aquesta aproximació en capes a la Web Semàntica va ser presentada per l'W3C i Tim Berners-Lee com "la pila de la Web Semàntica", com es mostra en la figura .

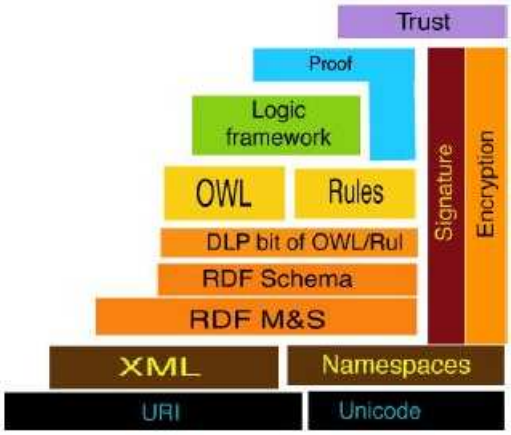

Figura 3.5: Pila de la Web Semàntica

En la base de la pila hi han els conceptes de l'URI (Universal Identification) i el Unicode (Universal Character Set). Per sobre d'aquestos conceptes, hi ha una capa amb la sintaxis XML (elements, atributs, etc.) i els *namespaces*. Per sobre de XML hi han les tripletes del model RDF. Si s'utilitza les tripletes per a denotar les classes, propietats de les classes, i valors, podem crear jerarquies de classes per la classificació i descripció d'objectes. Aquest és l'objectiu de l'RDF Schema.

Per sobre de l'RDF Schema trobem les ontologies (3.1.3) i per sobre les regles lògiques que s'usen per a inferir en el coneixement i per a agafar decisions. En aquest apartat ens centrarem en la cap de l'RDF Schema. *L'RDF Schema és un conjunt simple de recursos RDF i propietats que ens permeten crear els nostres propis vocabularis RDF.*

El model de dades representat per l'RDF Schema és el mateix usat per la POO[7], tal com en Java. El model de dades de l'RDF Schema ens permet crear classes de dades. Un *classe* es defineix com un grup de coses amb característiques comuns. Els llenguatges orientats a objectes permeten que les classes heredin les característiques i propietats de la classes pare, anomenada *"superclasse"*. L'industria del software ha estandarditzat una notació senzilla anomenada UML (*Unified Modeling Language*) que permet modelar jerarquies de classes. La figura 3.6 mostra un diagrama UML que modela dos tipus d'empleats i les seves associacions amb els documents que produeixen i les coses que saben.

[7]Programació orientada a objectes.

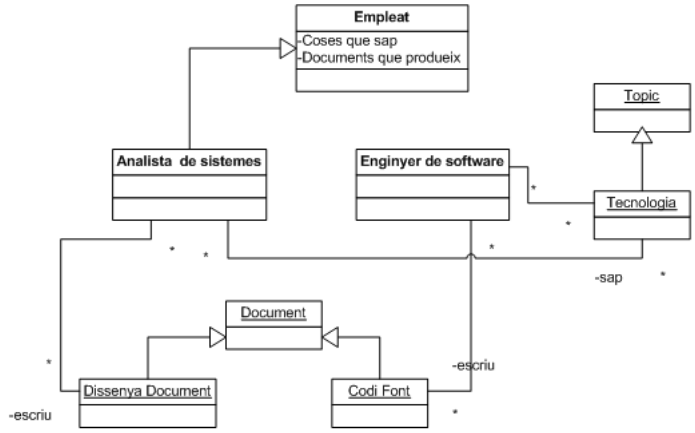

Figura 3.6: Exemple de diagrama de classes UML

Com podem veure en l'anterior exemple, s'utilitzen varis símbols UML per a descriure conceptes d'una classe, herència i associació. En aquest exemple es modelen dos tipus d'empleats: enginyers de software i analistes de sistemes. La clau diferenciadora que ens interessa són les diferents tipus de documents que ells creen.

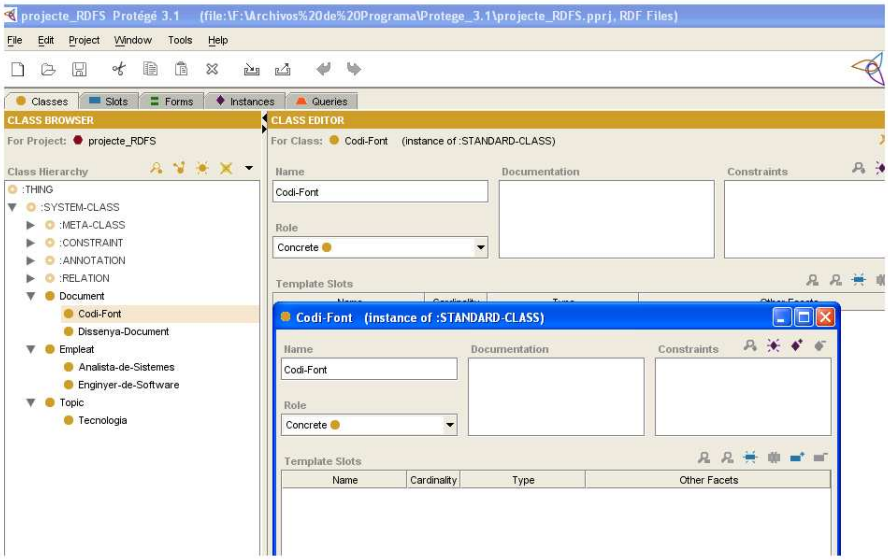

Figura 3.7: Exemple de RDFS amb el Protégé 3.1

La figura 3.7 mostra l'anterior exemple amb el *Protégé* que és una eina

de codi obert desenvolupada a la Universitat d'Stanford, i que es pot trobar a l'adreça: *http://protege.stanford.edu/*. El *Protégé* ens permet descriure classes i jerarquies. Com podem veure en la figura 3.7 en la part dreta es pot veure el contingut de la pestanya escollida i en la part esquerra es pot escollir la classe de la llista de classes.

Després de modelar les classes del RDFS, *Protégé* ens permet generar instàncies del esquema que hem creat (pestanya *Instances*), d'aquesta manera si una classe descriu els propietats d'una adreça, com: carrer, ciutat, país. Podem crear instàncies de l'adreça, com: "Arcadi Balaguer", Castelldefels, Catalunya.

A partir del model creat en la figura 3.6, obtenim el següent RDFS utilitzant el *Protégé*:

```
<rdf:RDF>
<!DOCTYPE rdf:RDF [
  <!ENTITY rdf 'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
  <!ENTITY Projecte_RDFS 'http://protege.standford.edu/Projecte_RDFS#'>
  <!ENTITY rdfs 'http://www.w3.org/TR/1999/PR-rdf-schema-19990303#'>]>
<rdf:RDF xmlns:rdf="&rdf;" xmlns:Projecte_RDFS="&Projecte_RDFS"
                          xmlns:rdfs="&rdfs;">
<rdfs:Class rdf:about="&Projecte_RDFS;Document" rdfs:label="Document">
  <rdfs:subclassOf rdf:resource="&rdfs;Recurs" />
<rdfs:Class>
<rdfs:Class rdf:about="&Projecte_RDFS,Dissenya-Document"
  rdfs:label="Disseya-Document">
  <rdfs:subClassOf rdf:resoruce="&Projecte_RDFS:Document" />
</rdfs:Class>
<rdfs:Class rdf:about="&Projecte_RDFS;Empleat" rdfs:label="Empleat">
  rdfs:label="Engiyer-de-Software">
  <rdfs:subClassOf rdf:recurs="&Projecte_RDFS;Empleat" />
</rdfs:Class>
<!- Classes Codi-Font, Analista-de-Sistemes; Tecnologia, i s'ha omitit
Topic
    per abreviar.  Són similars a les classes anteriors.  ->
<rdfs:Property rdf:about="&Projecte_RDFS;sap" rdfs:label="sap">
  <rdfs:domain rdf:resource="&Projecte_RDFS;Empleat" />
  <rdfs:range rdf:resource="&Projecte_RDFS;Topic" />
</rdf:Property>
<rdf:Property rdf:about="&Projecte_RDFS;escriu" rdfs:label="escriu">
  <rdfs:range rdf:resource="&Projecte_RDFS;Document" />
  <rdfs:domini rdf:resource="&Projecte_RDFS;Empleat" />
</rdf:Property>
</rdf:RDF>
```

Figura 3.8: RDF Schema obtingut

L'anterior llistat utilitza els següents components clau de RDF Schema:

rdfs:Class. És un element que defineix un grup de coses que comparteixen una sèrie de propietats. Treballa sempre conjuntament amb el *rdf:Property*, *rdf:range*, *rdfs:domain* per assignar propietats.

rdfs:label. És un atribut que defineix una etiqueta dins d'una classe llegible per als humans.

rdfs:subclassOf. És un element que especifica que una classe és una especialització d'una classe ja existent. L'idea d'una especialització és que una subclasse afegeix algunes característiques específiques a un concepte general.

rdf:Property. És un element que defineix una propietat d'una classe i el rang de valors que pot representar. S'utilitza juntament amb el *rdfs:domain* i el *rdfs:range*.

rdfs:domain. Aquesta propietat defineix a quina classe pertany una determinada propietat. El valor d'aquesta propietat té que ser una classe definida ja prèviament.

rdfs:range. Aquest propietat defineix el conjunt de valors d'una propietat. El valor d'aquest atribut té que ser una classe definida ja prèviament.

Existeixen d'altres propietats de l'RDFS, no utilitzades en el llistat 3.8 com:

- *rdf:type*
- *rdfs:subPropertyof*
- *rdfs:seeAlso*
- *rdfs:inDefinedBy*
- *rdfs:comment*
- *rdfs:Literal*
- *rdfs:XMLLiteral*

per brevetat no les detallaré.

El següent exemple ha estat agafat de l'especificació RDFS 1.0:

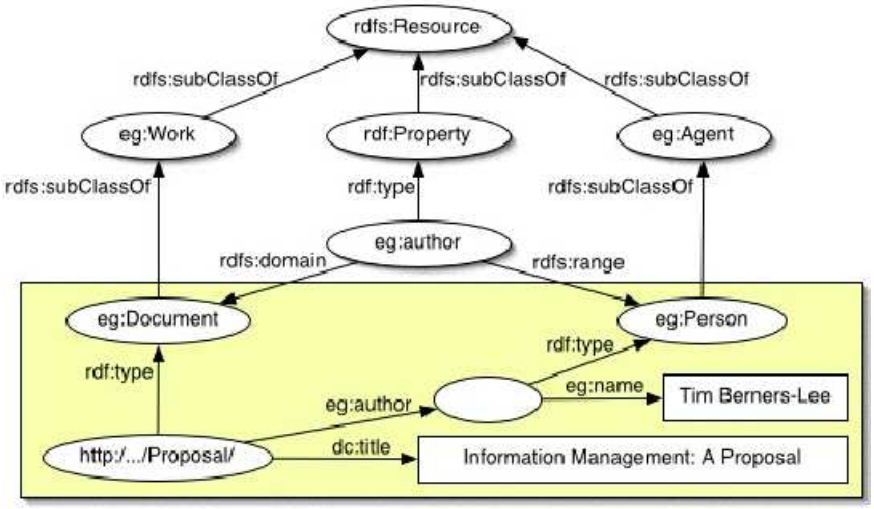

Figura 3.9: Exemple de graf RDFS

3.7 XML Schema i RDF Schema

Confusament, el terme "esquema" s'associa amb les dues especificacions W3C rivals, l'esquema XML i l'esquema RDF. En un sentit ampli, un esquema XML es dissenya per analitzar i validar la estructura d'etiquetes d'arxius de metadades (en aquest sentit, un esquema XML és un "esquema de document"). En contrast, per a representar les relacions de termes particulars amb altres termes del esquema o amb termes definits amb altres esquemes del Web, és preferible un esquema RDF (amb aquest sentit, un esquema RDF és un esquema semàntic).

A pesar de la similitud del seus noms, RDFS fa un paper diferent al de XMLS. XML, com els DTDs, fixa l'ordre i la combinació de les etiquetes d'un document XML. Pel contrari, RDFS proporciona informació de l'interpretació de les sentències contingudes en un model de dades RDF, però no restringeix l'apariència sintàctica d'una descripció RDF.

Desenvolupats independentment per dos grups de treball paral.lels a finals dels 90, la relació entre aquestes dues especificacions és la causa de moltes confusions, i des de gener de 2002, el W3C està liderant els esforços per a combinar les funcionalitats d'ambdós amb un llenguatge d'esquema integrat.

Patel-Scheneider i Simeon[8] senyalen que les diferències entre RDFS i XML: XML és ordenat, RDF no; XML fa anar un model d'arbre, RDF de grafs; RDF distingeix entre classes i propietats, i en XML, tot són ele-

[8]Patel-Scheneider, P. Simeon, J. The Ying/Yang Web: XML syntax i RDF semantics. WWW2002, Honolulu, maig de 2002. Disponible a: http://www2002.org/CDROM/refereed/231/index.html

ments. No obstant, es creu que és possible desenvolupar un model unificat que serveixi com a base per les aplicacions que treballin tan amb documents (XML) com amb semàntica (RDF). El model *Ying/Yang* es desenvolupa sobre el model *XQuery 1.0* i *XPath 2.0*. La seva aproximació es centra en l'incorporació de extensions de tipus de dades XMLS en documents XML/RDF.

El seu model no resol aspectes com la "reificació" i els contenidors de RDF, ni té clar com incorporar les cardinalitats de XMLS en un entorn semàntic. Tampoc està clar si seria més apropiat consultar el model Ying/Yang com un llenguatge basat en extensions de XQuery (XML), entenent els llenguatges de consulta RDF, com RQL, o dissenyant un llenguatge unificat que proporcioni les característiques d'ambdós llenguatges.

Capítol 4

Llenguatges de consulta RDF

L'RDF és considerat l'estàndard més rellevant per a l'intercanvi de dades i per la representació de models ([13]). Han sorgit varis llenguatges de consulta per a documents RDF, alguns similars als llenguatges tradicionals per a la consulta de bases de dades, com SQL i OQL, d'altres més tancats i inspirats en llenguatges de regles. No s'ha definit encara clarament un llenguatge per a la consulta de documents RDF que sigui un estàndard i ara la discussió està entre els amants de la Web Semàntica i el W3C.

En aquest capítol veurem una introducció als diferents llenguatges de consulta per a documents RDF existents en l'actualitat. Aquestos llenguatge es centren en dos paradigmes:

- *Navegacional:* Centrat en la cerca de camins com es fa amb XPath (2.7).
- *Posicional:* Centrat en la lògica (programació lògica).

4.1 Dimensions dels llenguatges

Existeixen diverses característiques que són importants per a dissenyar i comparar els llenguatges de consulta. Anem a descriure-les seguidament.

4.1.1 Suport per a models de dades RDF

El model de dades escollit influència directament el conjunt d'operacions que un llenguatge de consulta ens proporciona. A continuació especificaré els conceptes bàsics de l'RDF i les seves implicacions dintre dels llenguatges de consulta.

- **Model de dades abstracte:** L'estructura d'un document RDF és desglossa en tripletes, i se l'anomena usualment graf RDF. Cada estat

d'una tripleta representa una relació entre dos nodes dintre d'aquest graf. Els llenguatges de consulta, sovint, ens proporcionen eines per a fer consultes de serialització, per exemple, l'ordre de serialització.

- **Semàntica formal i inferència:** El llenguatge RDF té una sèrie de primitives formals que ens permeten consultar sobre la semàntica d'un graf. A aquestes primitives se les anomena "*entailment rules*", i ens permeten diferenciar les dades implícites de les explícites.

- **Suport per a l'XML Schema:** El tipus de dades XML poden ser usats dintre d'un document RDF. L'XML Schema ens permet definir també nous tipus de dades per a ser utilitzats dintre d'un document RDF, i per tant, poden ser usats dintre dels llenguatges de consulta RDF.

- **Lliure suport per a definir característiques d'un recurs:** En general, no és possible obtenir l'informació complerta de qualsevol recurs definit en un document RDF. Un llenguatge de consulta té que tenir compte amb això i poder manegar l'informació incomplerta i contradictòria.

4.1.2 Propietats d'un llenguatge de consulta

Les característiques principals d'un llenguatge de consulta són les següents:

- **Expressivitat:** L'expressivitat determina com són i com poden ser de potents les consultes en un determinat llenguatge de consulta. Normalment un llenguatge té que donar-nos almenys la disponibilitat dels operadors de l'àlgebra relacional, o sigui, ser relacionalment complert. Usualment, aquesta expressivitat es limita al manteniment d'altres propietats com la seguretat i permet l'execució de les consultes eficientment, i també ens dona la possibilitat de que aquestes puguin ser optimitzades.

- **Clausura:** Les propietats de clausura requereixen que els resultats d'una operació siguin també elements del model de dades. Això significa que si un llenguatge de consulta opera dintre del model d'un graf, els resultats d'una consulta tenen que ser també grafs.

- **Adequació:** A un llenguatge de consulta se l'anomena adequat si utilitza els conceptes del seu model de dades. Aquesta propietat és complementària de la propietat de clausura.

- **Ortogonalitat:** L'ortogonalitat d'una consulta requereix que totes les operacions d'un llenguatge de consulta poder ser usades independentment del context on es facin servir.

- **Seguretat:** Un llenguatge de consulta es considera segur si qualsevol consulta sintàcticament correcta retorna un conjunt finit de resultats.

4.2 Descripció dels llenguatges de consulta RDF

4.2.1 La familia SPARQL

La família *SPARQL* consisteix en quatre llenguatges de consulta: *SquishQL*, *RDQL*, *SPARQL* i *TriQL*. La característica comú d'aquestos quatre llenguatges és que veuen les dades de les tripletes RDF sense cap informació addicional (informació d'un esquema o d'una ontologia) més que la continguda dintre del codi RDF.

4.2.1.1 Accés bàsic RDF: SquishQL i RDQL

El Squish té un model de consulta propi per RDF i és bastant similar a SQL. Ofereix les anomenades plantilles de tripletes (*triple patterns*) i les conjuncions que permeten especificar les parts d'un graf RDF que volen ser recuperades. Un altra propietat d'aquest llenguatge és que no permet expressar recursió.

Les consultes en *SquishQL* tenen la següent forma:

```
SELECT ?composicio,?autor,?nomAutor
>FROM http://exemple.org/musica
WHERE (?composicio, <rdf:type>, <musica:Composicio>),
        (?composicio, <rdf:autor>, <musica:autor>),
        (?autor, <rdf:nom>, <musica:nomAutor>)
USING musica FOR http://exemple.org/musica#,
       rdf FOR http://www.w3.org/1999/02/22-rdf-syntax-ns#
```

Figura 4.1: Consulta en SquishQL

L'anterior consulta sel.lecciona totes les composicions amb els seus autors i els noms del mateixos. Segons l'anterior patró podríem expressar la consulta: "*sel.leccionar totes les composicions que es diuen 'la primavera'*" de la següent forma:

```
SELECT ?property, ?propertyValue
>FROM http://exemple.org/musica
WHERE  (?composicio, <musica>:composicio-titol:
        'la primavera')
       (?composicio, ?property, ?propertyValue),
USING musica FOR http://exemple.org/musica#
```

Dintre d'una consulta SquishQL, la clàusula AND serveix per expressar restriccions de valors. La següent consulta retorna els URIs de les persones que han composat un peça anomenada: 'la primavera':

```
SELECT ?persona
>FROM http://exemple.org/musica
```

```
WHERE (?composicio, <musica:autor>, ?persona)
      (?composicio, <musica:titol>, ?titol)
AND   ?titol = 'la primavera'
```

El *SquishQL* no suporta RDFS i el resultat d'un consulta és un enllaç a les variables que la composen.
El llenguatge RDQL (*RDF Data Query Language*) és una evolució del SquishQL i ha esta estandarditzat recentment per l'W3C. Les consultes RDQL tenen el mateix format que les consultes SquishQL. Tal com en l'SquishQL, el resultat d'una consulta RDQL és un conjunt d'enllaços a les variables que hi ha a la consulta. Igualment, només suporta consultes de sel.lecció i d'extracció. L'RDQLPlus (*http://rdqlplus.sourceforge.net/*) és una implementació de l'RDQL que ens aporta una extensió d'aquest llenguatge anomenada RIDIQL.

4.2.1.2 SPARQL

El SPARQL és una extensió del RDQL dissenyada d'acord amb els seus requeriments i que està encara en evolució. Així doncs, podem dir que SPARQL estén RDQL i facilita el:

- **Extreure** subgrafs RDF.
- **Construir**, mitjançant la clàusula CONSTRUCT un nou graf RDF les dades del graf RDF en qüestió. Tal com les consultes RDQL, el nou graf pot ser especificat amb **tripletes** o **les plantilles de grafs**.
- **Retorn**, usant clàusules DESCRIBE, que són "*descripcions*" dels recursos que coincideixen amb la consulta escollida. El significat exacte de "*descripció*" escara no ha estat definit.
- **Especificar** la part OPCIONAL de la la tripleta o plantilla del graf de la consulta.

Les consultes SPARQL tenen la següent forma:

PREFIX Especificació del nom per un URI (tal com el USING del RDQL).

SELECT Retorna totes o algunes variables que són enllaçades mitjançant la clàusula WHERE.

CONSTRUCT Retorna un graf RDF amb tots els enllaços a variables.

DESCRIBE Retorna una descripció dels recursos enllaçats.

ASK Retorna si una patró de consulta coincideix o no.

WHERE Llista de patrons coincideix o no.

OPTIONAL Llista de patrons opcionals.

AND Expressió booleana (el filtre que té que ser aplicat al resultat).

Per exemple, podríem fer una extensió de la consulta 4.1, fent que ens retorni A més de totes les composicions i els seus autors també els instruments que hi intervenen, això seria:

```
PREFIX     musica:  http://exemple..org/musica#
PREFIX     rdf:  http://www.w3.org/1999/02/22-rdf-syntax-ns#
SELECT     ?composicio, ?autor, ?nomAutor, ?instrument
>FROM     http://exemple.org/musica
WHERE     (?composicio musica:autor ?autor),
            (?autor musica:autorNom ?autorNom)
OPTIONAL (?composicio musica:instrument ?instrument)
```

Figura 4.2: Consulta en SPARQL

Mitjançant la clàusula CONSTRUCT, es poden fer consultes d'inferència no recursives i de reestructuració. *Per exemple, podríem invertir la relació d'autor (de una composició a un autor) a la relació "composa" (d'un autor a una composició) de la següent manera*:

```
PREFIX    musica: http://exemple.org/musica#
CONSTRUCT (?y musica:composa ?x)
>FROM     http://exemple.org/musica
WHERE     (?x musica:autor ?y)
```

4.2.1.3 TriQL

TriQL estén RDQL mitjançant el suport a la construcció de consultes amb grafs amb nom (*named graphs*). Els *named graphs* permeten filtrar sentències RDF desprès dels seus recursos o autors. Per exemple, podríem fer una consulta que ens retornés totes les composicions on la informació ha estat inserida per '*Marcus Tullius Cicero*'. Aquesta expressió en TriQL quedaria com segueix:

```
SELECT ?composicio
WHERE ?graph ( ?composicio musica:rating ?rating)
        (?graph sqp:assertedBy ?warrant)
        (?warrant swp:authority <http://persónes.net/cicero>)
USING musica FOR http://exemple.org/musica#,
       swp FOR <http://www.w3.org/2004/03/trix/swp-1/>
```

Figura 4.3: Consulta en TriQL

4.2.2 La familia RQL

La familia de llenguatges de consulta RQL està formada pels llenguatges : **RQL**, **SeRQL** i **eRQL**. La característica comú d'aquests llenguatges és que suporten la combinació de dades i consultes amb esquema (*schema querying*). Encara més, podem dir que el seu model de dades RDF es desvia del model de dades RDF estàndard.

- Els cicles en les relacions entre classe i subclasse estan prohibits.
- Per a cada propietat és obligatori definir el seu domini i el seu rang.

Això assegura una separació clara entre les 3 capes d'abstracció de RDF i de RDFS:

1. **Dades**, per exemple, la descripció de recursos com si fossin persones.
2. **Esquemes**, per exemple, les classificacions d'aquestos recursos.
3. **Meta-esquemes**, especificant meta-classes tal com *rdfs:Class* i *rdfs:Property*.

4.2.2.1 RQL

RQL (*RDF Query Language*) és la base dels llenguatges SeRQL (*Sesame RDF Query Language*) i del eRQL. La característica principal d'aquest llenguatge és l'ús d'esquemes RDFS. Es tracta d'un llenguatge funcional que suporta expressions de camins generalitzades amb variables o nodes d'un graf RDF. RQL té un model de graf formal que captura les primitives de modelat RDF i que permet l'interpretació de descripcions de recursos mitjançant un o més esquemes.

Aquest llenguatge permet la combinació d'esquemes i de dades en una consulta. Anem a veure un exemple d'una consulta RQL:

```
SELECT $C1, $C2 FROM {$C1}musica:autor{$C2}
USING NAMESPACE musica = &http://exemple.org/musica#
```

Figura 4.4: Consulta en RQL

L'exemple 4.4 retorna el domini (*$C1*) i el rang (*$C2*) de la propietat autor definida en l'URI anomenada *música*[1]. Aquesta consulta pot ser expressada de dues maneres més, però per simplificar no les descriurem.

RQL té a més variables de propietat (*property variables*), que es defineixen amb el signe @ i utilitzant les propietats RDF poden ser consultades (tal com les classes utilitzant les variables de les classes). *La següent consulta retorna les propietats conjuntament amb els seus rangs que poden ser assignades als recursos classificats com "musica: Composar"*:

[1] El prefix $ indica una classe de variable, per exemple, un rang de variables o de esquemes de classes.

```
SELECT @P; $V FROM [;musica:Composar]@P{$V}
USING NAMESPACE musica = &http://exemple.org/musica#
```

Amb RQL, les dades poden ser recuperades pels seus tipus, mitjançant la navegació apropiada pel graf, i les restriccions poden ser expressades utilitzant filtres. El llenguatge RQL és molt més expressiu que la majoria dels altres llenguatges RDF.

Pel contrari, RQL és criticat per:

- El seu gran nombre de característiques.
- El gran nombre possible de construccions sintàctiques (tal com els prefixos ^ per a les crides i la @ per a les propietats de les variables).

La majoria de les consultes, excepte les consultes que fan referència a clausures transitives de relacions arbitràries, poden ser expressades amb RQL: RDF suporta solament les clausures transitives de *rdfs:subClassOf* i *rdfs:subPropertyOf*.

És convenient utilitzar RQL per expressar consultes d'agregació tal com ara veurem. *Per exemple, podem fer una consulta que ens retorni cadascuna de les subclasses de 'Composar', juntament amb la mitja d'autors per a cada composició d'aquesta subclasse.*

```
max (SELECT Y
 FROM (B;musica:Composar}musica:autor.musica:nomAuthor{A},
      {B}musica:composicioAny{Y}
 WHERE A = "La Traviata")
```

Finalment, diré que es podrien dir moltes altres característiques del llenguatge RQL, però si que es pot dir que ha influenciat molt en d'altres llenguatges de consulta com BrQL i SPARQL.

4.2.2.2 SeRQL

El SeRQL (*Sesame RDF Query Language*) és derivat del llenguatge RQL, és un nou llenguatge de consulta RDF/RDFS que està encara sent desenvolupat per Aduna com a part del Sesame[2]. Combina les millors característiques d'altres llenguatges de consulta, com RQL, RDQL, N-Triples i N3, i afegeix noves funcionalitats. Es diferència del llenguatge RQL en el següent:

- No suporta els tipus RDF i RDFS, excepte els tipus literal.
- Modifica i amplia les expressions de camins (path) RQL. SeRQL forma els camins amb un node buit per la concatenació de camins.

[2]És un gestor de bases de dades per documents RDF que descriuré en el següent capítol.

- SeRQL ens proveeix d'una notació abreviada per a la recuperació de varis valors d'una propietat en una expressió senzilla.
- SeRQL permet consultar una declaració "reificada" (*reified*) tancant la resta entre parèntesis.

No tots els exemples de consulta que hem vist fins ara poden ser expressats amb SeRQL, les consultes de sel.lecció i extracció si que poden ser expressades amb SeRQL. D'altra banda, en SeRQL no té cap conjunt d'-operadors existencials o quantificadors universals. Gràcies a la clàusula *CONSTRUCT*, en SeRQL, al igual que en RQL, es pot fer consultes de reestructuració o d'inferència simple. En la consulta 4.5 es pot veure com s'inverteix la relació autor ("*des de composició a un autor*") en la relació composa ("*des de un autor a una composició*"):

```
CONSTRUCT    {Author} <lamevamusica:composa> {Composicio}
>FROM          {Composicio} <musica:autor> {Autor}
USING NAMESPACE musica = <!http://exemple.org/musica#>
        lamevamusica = <!http://exemple.org/musica-rdfs-extension#>
```

Figura 4.5: Consulta en SeRQL

Algunes característiques d'aquest llenguatge són:

- Les variables s'identifiquen per noms. Aquestos noms tenen que començar amb una lletra o amb un "*underscore*" ('_') i poden ser seguides per zero o més lletres, nombres, *underscores*, guions o punts. Alguns exemples de variables són:

  ```
  var1
  _var2
  unwise.var_nom_isnt-it
  ```

- En SeRQL no hi ha forma possible d'abreviar el nom dels URIs, tenen que tenir el nom complert i estar tancats entre els símbols "<" i ">".

  ```
  <http://www.openrdf.org/index>
  <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
  ```

- Una de les parts més importants de SeRQL són les expressions de camins (*Path Expressions*) que especifiquen camins dintre d'un graf RDF. La majoria del llenguatges de consulta permeten definir expressions de camins de longitud 1, les poden ser utilitzades per a trobar tripletes dintre d'un graf RDF. SeRQL, al igual que RQL, ens permet definir expressions de camins de longitud arbitrària. Un exemple d'aquesta notació seria el següent, i també es mostra el seu graf associat:

```
[Persona] ex:TreballaPer {Companyia} rdf:tipus {ex:ITCompanyia}
```

Figura 4.6: Exemple de graf en SeRQL

- En una consulta SeRQL, es poden especificar múltiples expressions de camins, que tenen que ser separades per comes. Per exemple, l'anterior expressió pot ser escrita com dues expressions més senzilles:

```
{Persona} ex:TreballaPer {Companyia},
{Companyia rdf:tipus {ex:ITCompanyia}
```

- Els nodes de les expressions de camins poder ser variables, URIs i literals. També un node pot ser buit en el cas de que no ens interessi el valor d'aquest node.

- En els casos de que es vulgui consultar dos o més tripletes amb idèntic subjecte i predicat, i no es vulgui repetir tota la estona el mateix, es pot utilitzar una expressió amb molts valors (multivalor). Aquestes expressions tenen la següent sintaxis:

```
{subj1} pred1 {obj1, obj2, obj3}
```

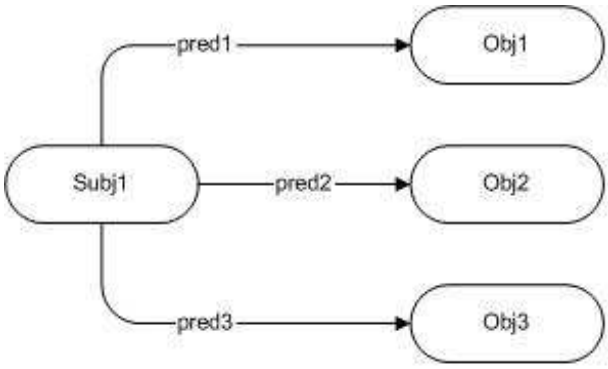

Figura 4.7: Exemple d'expressió multivalor en SeRQL

- Les sentències de "reificació" (reified) poden ser escrites de la següent manera:

```
{ {reifSubj} reifPred {reifObj} } pred {obj}
```

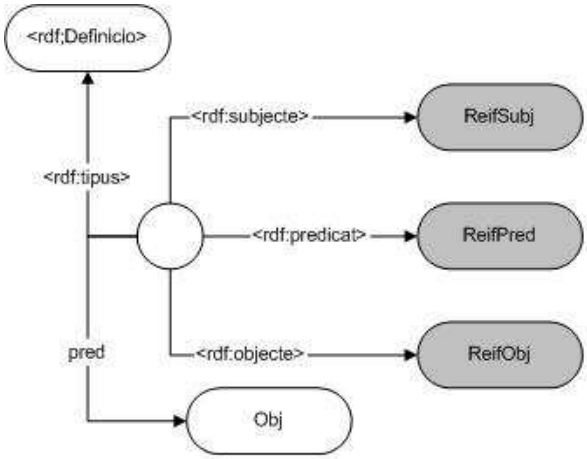

Figura 4.8: Exemple d'una expressió amb "reificació"

- Les expressions de camins opcionals es distingeixen de les expressions de camins 'normals' en que no hi tenen que haver coincidències per a retornar resultats. El motor de consultes de SeRQL intenta trobar camins dintre del graf RDF fent coincidir l'expressió del camí, però si no troba cap resultat es salta l'expressió.

Per finalitzar, mostraré un exemple de consulta SeRQL i el seu graf associat. *L'exemple consisteix en trobar tots els documents RDF sobre consultes i els seus autors.*

```
SELECT Autor, Paper
FROM {Paper} rdf:tipus {foo:Paper};
             ex:clau {"RDF","Consultes"};
             dc:autor {Autor}
USING NAMESPACE
  dc = <http://purl.org/dc/elements/1.0/>,
  ex = <http://example.org/thing#>
```

El graf associat a aquest consulta seria el següent:

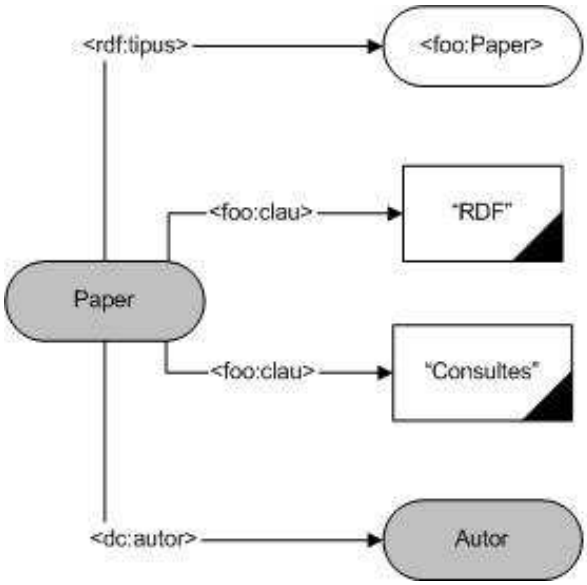

Figura 4.9: Exemple graf de consulta SeRQL

4.2.3 Llenguatges de consulta inspirats en XPath, XSLT o XQuery

4.2.3.1 XQuery per RDF

La primera aproximació sintàctica a aquest llenguatge és la següent:

- Requereix una normalització preliminar de les dades RDF consultades principalment per a serialitzar les dades RDF en XML i per agrupar les sentències pel seu subjecte.
- També fa falta definir funcions en *XQuery* per a convertir la semàntica de RDFS utilitzant la primitiva *rdf:instance-of-class.*

Un exemple de consulta en *XQuery* per RDF seria, per exemple, que volguéssim sel.leccionar totes les composicions, els seus autors i els seus noms corresponents. Això ho faríem de la següent manera:

```
let $t := document("http://exemple.org/musica") //descripcio
for $composicio in rdf:instance-of-class($t, "musica:composicio"),
   $autor in $t[rdf:about = $composicio/musica:autor]
return <result> {$composicio, $autor} </result>
```

Figura 4.10: Consulta en XQuery per RDF

L'aproximació sintàctica d'aquest llenguatge fa possible el següent:

- Retornar els resultats en qualsevol format possible XML.

- Consultar dades en la Web Estàndard i dades de la Web Semàntica amb el mateix llenguatge de consulta.

4.2.3.2 XSLT per RDF: TreeHugger i RDF Twig

Similarment al llenguatge 4.10 l'aproximació sintàctica proposa centrar-se en el XSLT per a consultar i transformar dades (extensió de XLST 1.0 anomenada TreeHugger).

L'RDF Twig és un altra extensió de l'XLST 1.0 amb funcions per a fer consultes RDF. Està basat en recorreguts redundants o no redundants d'un graf RDF. A més, el XSLT per RDF ens proporciona dos mecanismes de consulta:

- Un conjunt lògic d'operacions per als grafs RDF.
- Un interface RDQL.

4.2.3.3 Versa

Va ser desenvolupat com una part de la Suite XML i RDF Python 2.3 i és un llenguatge de consulta RDF inspirat amb *XPath* però significativament diferent. Tal com les aproximacions sintàctiques descrites a 4.2.3.2 (TreeHugger i RDF Twig), el Versa està alineat amb XML, i tal com l'XPath per XML, pot ser estès amb funcions externes.

El Versa ofereix suport per regles que permeten creuar predicats transitivament. Permet també la creació de llistes de llistes, les quals permeten sel.leccionar varies propietats d'una determinada llista de recursos.

4.2.4 Metalog: consultes en anglès controlat

El *Metalog* es diferència dels altres llenguatges de consulta RDF per dues raons:

- Combina capacitat de consulta i de raonament.
- La sintaxis del Metalog és un *llenguatge natural controlat* (anglès).

Seguidament es mostra un exemple en aquest llenguatge, que ja ha estat traduït a la sintaxis d'altres llenguatges anteriors (*sel.leccionar totes les composicions, els seus autors i els seus corresponents títols*) :

```
COMPOSICIO represents the term "Composicio"
COMPOSADA-PER represents the verb "autor"
IS represents the verb "rdf:type"
LA_TRAVIATA represents la composicio "La_Traviata" from the collection of
                                        music http://exemple.org/musica#"
```

Figura 4.11: Consulta en Metalog

4.2.5 Algae: un llenguatge amb regles "reactives"

El llenguatge "reactiu" Algae, o Algae2, està basat en dos conceptes:

- Les accions són les directives *ask*, *assert* i *fwrule* que determinen si una expressió s'utilitza per a consultar dades RDF, inserir dades dintre d'un graf, o especificar regles.
- Els resultats de les consultes en *Algae* són enllaços a les variables de la consulta tal com les tripletes dels grafs RDF.

Sintàcticament el *Algae* està basat en la sintaxis RDF de N-triples estesa amb algunes accions i restriccions (que s'escriuen entre parèntesis i especifiquen comparacions). Anem a traduir l'exemple de l'apartat 4.11 a aquest llenguatge:

```
ns rdf = <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
ns musica = <http://exemple.org/musica#>
read <http://exemple.org/musica> ( )
ask ( ?composicio rdf:type <http://exemple.org/musica#composicio>
     ?composicio musica:autor ?autor
     ?autor musica:nomAutor ?autorNom
    )
collect (?title, ?autor, ?nomAutor)
```

Figura 4.12: Consulta en Algae

4.2.6 Llenguatges de consulta deductius

4.2.6.1 N3QL

El llenguatge N3QL és derivat de la *Notation 3*, una extensió sintàctica de l'RDF amb variables, regles i notació per a construir expressions d'expressions. Una consulta N3QL és una expressió N3. Totes les paraules reservades són propietats RDF d'un node representat a la consulta.

Seguidament mostraré el mateix exemple de l'apartat anterior (4.12), però traduït a la sintaxis de N3QL:

```
@prefix musica:  <http://exemple.org/musica#>.
@prefix n3ql:  <http://www.w3.org/2004/ql#>.
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
[ ] n3ql:select {n3ql:result n3ql:is (?composicio ?autor ?nomAutor) };
   n3ql:where { ?composicio rdf.type musica:composicio;
                ?composicio musica:autor ?autor;
                ?autor musica:nomAutor ?nomAutor }.
```

Figura 4.13: Consulta en N3QL

El resultat d'aquesta consulta és el graf RDF especificat en la clàusula *n3ql:select*, un conjunt de col.leccions RDF i d'enllaços a variables. La següent regla especifica la clausura transitiva d'una propietat RDF:

```
@prefix rdfs:<http://www.w3.org/2000/01/rdf-schema#>.
{?x rdfs:subClassOf ?z; ?z rdfs:subClassOf ?y}
                    => {?x rdfs:subClassOf ?y}
```

4.2.6.2 TRIPLE

El *TRIPLE* és un llenguatge basat amb consultes amb *regles*, *inferència* i *transformació* per a documents RDF. Ha estat dissenyat per a solucionar dues debilitats dels llenguatges de consulta RDF:

- Les construccions predefinides restringeixen la semàntica RDFS i no permet l'extensibilitat dels llenguatges.
- La falta de semàntica formal.

Al contrari de les altres construccions predefinides d'RDFS en altres llenguatges, el TRIPLE ofereix regles de Horn amb les quals es poden implementar característiques de l'RDFS. On la lògica del *Horn* no és suficient, com és el cas de OWL, el TRIPLE permet ser estès amb mòduls externs que implementen un raonador OWL.

El llenguatge TRIPLE es diferència de la lògica de Horn i de la lògica de programació amb el següent:

- Suporta recursos identificats per URIs.
- Les declaracions RDF es presenten en TRIPLE per *slots*, que permeten l'agrupació i la recurrència, les expressions poden ser usades per a creuar varies propietats.
- Proveeix un suport ampli per declaracions "reificades".
- Té una noció de mòdul que permet l'especificació del model que és cert en cada sentència o àtom.
- Requereix una determinació explícita de variables.

Seguidament, mostraré l'exemple utilitzat en d'altres apartats, com 4.11, i que *sel.lecciona totes les composicions i els autors amb el seu nom.* La consulta seria la següent:

```
rdf := 'http://www.w3.org/1999/02/22-rdf-syntax-ns#'.
musica := 'http://exemple.org/musica#'.
musicaModel := 'http://exemple.org/musica'.
FORALL B, A, AN result (B, A, AN) <-
  B[rdf:type -> musica:Composicio; musica:autor ->
    A[musica:nomAutor -> AN]]@musicaModel.
```

Figura 4.14: Consulta en TRIPLE

En la següent consulta es mostra un altre exemple amb llenguatge TRIPLE, que sel.lecciona tots els recursos classificats com *'musica:Composicio'*, cal senyalar com les regles del TRIPLE aporten una potenciació de la semàntica RDFS. Veiem-lo:

```
rdf := 'http://www.w3.org/1999/02/22-rdf-syntax-ns#'.
rdfs:= 'http://www.w3.org/2000/01/rdf-schema#'.
type:= rdf:type.
subPropertyOf := rdfs:subPropertyOf.
subClassOf := rdfs:subClassOf.
FORALL Mdl @rdfschema(Mdl) {
  transitive(subPropertyOf).
  transitive(subClassOf).
  FORALL O,P,V O[P->V] <- O[P->V]@Mdl.
  FORALL O,P,V O[P->V] <-
     EXISTS S S[subPropertyOf->P] AND O[S->V].
  FORALL O,P,V O[P->V] <-
     transitive(P) AND EXISTS W (O[P->W] AND W[P->V]).
  FORALL O,T O[type->T] <-
     EXISTS S (S[subClassOf->T] AND O[type->S]).
```

Finalment, cal senyalar que les consultes d'agregació no poden ser expressades en llenguatge TRIPLE.

4.2.6.3 Xcerpt

El *Xcerpt* és un llenguatge per a la consulta de dades en la "*Web Estàndard*" i per a la consulta de dades en la Web Semàntica. *Xcerpt* té tres propietats que són convenients per a la consulta de dades RDF:

- Els patrons de l'Xcerpt completen les consultes incomplertes, mitjançant la connexió de recursos veïns i la diagonalització de grafs amb longitud indefinida.
- L'encadenament (possibles regles recursives) és convenien per a expressar semàntiques RDFS (per exemple: la clausura transitiva de la relació *subClassOf*) i totes les classes de grafs transversals.

- La construcció *opcional* és convenient per a recollir les propietats de recursos.

Totes les consultes d'exemple vistes en les anteriors seccions poden ser expressades en *Xcerpt* en serialització XML i en serialització RDF. Anem ara a veure l'exemple ja desenvolupat en els altres seccions segons la vista Xcerpt del graf RDF:

```
DECLARE ns-prefix rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
DECLARE ns-prefix musica="http://exemple.org/musica#"
GOAL
  result [
    all composicio [
      id [ var Composicio ],
      all autor [
        id [ var Autor ],
        all name [ var nomAutor ],
      ]
    ]
   ]
FROM
  RDFS-GRAPH {{
    var Composicio:uri {{
      rdf:type {{ "musica:Composicio":uri {{ }} }},
      musica:autor {{
        var Autor:uri {{
          musica:nom {{ var nomAutor }}
        }}
      }}
    }}
  }}
END
```

Figura 4.15: Consulta en Xcerpt

La vista del graf RDFS s'utilitza per estendre el graf RDF tal com passa amb el *RDF-TRIPLE* amb el *RDFS-TRIPLE*. El model de dades Xcerpt és un graf amb arrel i cap arbre d'aquest graf pot ser representat sense la duplicació de recursos.

Anem a veure ara un exemple d'una regla que especifica la vista de graf a partir de la vista de tripletes introduïda anteriorment:

```
CONSTRUCT
  RDF-GRAPH {
    all var Subject @ var Subjet:var SubjectType {
      all optional var Predicate {
         ^var Object
      },
```

```
        all optional var Predicate {
          var Literal
        }
        }
      } }
FROM
  or{
   RDF-TRIPLE[
     var Subject:var SubjectType{},
     var Predicate:uri{},
     optional var Literal -> literal{{}},
     optional var Object :/uri|blank/{{}}
   ]
   RDF-TRIPLE[ /.*/:/.*/{{}}, /.*/:/.*/{{}},
    var Subject:var SubjectType
  }
END
```

La sentència *optional* indica que una part del patró no té que complir-se i que les variables estan enllaçades correctament. L'*optional* permet que les consultes amb varies alternatives puguin ser expressades concísament, això és molt important per RDF ja que totes les propietats en RDF són opcionals per defecte. Els operadors '^' i '?@' s'utilitzen per a construir enllaços.

Finalment, podem dir que les regles de l'Xcerpt fan de la serialització quelcom transparent, fen que expressin una transformació en la serialització desitjada.

4.2.7 Comparativa d'alguns llenguages segons casos d'ús

En aquesta secció presentaré una breu comparativa d'alguns dels llenguatges estudiats, concretament: *RDQL, TRIPLE, SeRQL, Versa, N3 i RQL*, per a alguns casos d'ús ([13]).

4.2.7.1 Expressions de camins

La principal característica per a la cerca dintre d'un graf s'anomena expressió de camí, que s'utilitza típicament per a fer un recorregut dintre d'un graf. Una expressió de camí pot ser descomposta amb varis *joins* i s'implementa freqüentment amb *joins*. Per tant, no és de sorprendre que les expressions de camins siguin ofertes, en diverses formes sintàctiques, per tots els llenguatges de consulta RDF.

Consulta	RDQL	TRIPLE	SeRQL	Versa	N3	RQL
Camí	Sí	Sí	Sí	Sí	Sí	Parcial

Taula 4.2: Expressions de camins

4.2.7.2 Expressions opcionals de camins

Els grafs RDF representen un model de dades semi-estructurat. Aquesta estructura permet representar informació irregular i incomplerta, és per això que alguns llenguatges de consulta RDF permeten treballar amb aquesta informació irregular i incomplerta. Una irregularitat particular, que té que ser tinguda molt en compte, és que el valor cercat pot no existir.

Consulta	RDQL	TRIPLE	SeRQL	Versa	N3	RQL
Camí opcional	No	No	Sí	Sí	No	No

Taula 4.4: Expressions opcional de camins

Desafortunadament, solament dos llenguatges, el Versa i el SeRQL, ens proveeixen de mitjans per a tractar informació incomplerta. Per exemple, el llenguatge SeRQL ens dona una sèrie d'expressions opcionals per a trobar coincidències de camins que tenen en compte aquesta irregularitat.

4.2.7.3 Relacions

El llenguatge RDF s'utilitza freqüentment per a modelar estructures relacionals. En efecte, les taules n-aries, com poden ser trobades en el model de dades relacional, poden ser codificades fàcilment com una tripleta RDF.

En el model de dades relacional, varies operacions algebraiques són considerades: projecció, producte cartesià, diferència i la sel.lecció. Aquestes operacions poden ser combinades per expressar altres operacions com la intersecció, varis tipus de joins, etc. L'importància d'aquestes operacions és molt gran, i podem dir, que dintre del món relacional és pot dir que un llenguatge és relacionalment complert si suporta un conjunt bàsic d'operacions relacionals.

Les tres bàsiques operacions algebraiques: sel.lecció, projecció i producte són suportades per tots els llenguatges de consulta RDF. Ens centrarem en les altres dues operacions bàsiques mencionades avanç: la unió i la diferència.

Consulta	RDQL	TRIPLE	SeRQL	Versa	N3	RQL
Unió	No	Sí	No	Sí	Sí	Sí
Diferència	No	No	No	No	No	Sí

Taula 4.6: Expressions d'unió i diferència

La unió és introduïda en el llenguatge RDF, el llenguatge Versa conté un operador explícit d'unió també. Els llenguatges Versa i TRIPLE poden simular la unió amb regles. De la mateixa manera, l'únic llenguatge que té un operador algebraic de diferència és RQL.

4.2.7.4 Agregació i agrupament

Les funcions d'agregat retornen un valor escalar a partir d'un conjunt de valors múltiples. Aquestes funcions són necessitades regularment per a contar el nombre de valors. Per exemple, és necessiten per a identificar el valor màxim o el mínim d'un conjunt de valors. Un cas especial d'agregació és el nombre d'elements d'un conjunt (*count*):

Consulta	RDQL	TRIPLE	SeRQL	Versa	N3	RQL
Count	No	No	No	Sí	Sí	Sí

Taula 4.8: Expressions d'agregació (count)

Els llenguatges que suporten el *count* són: N3, Versa i RQL

4.2.7.5 Recursió

Les consultes recursives apareixen sovint en els sistemes d'informació, típicament representant relacions transitives. Podem assumir que les definicions ontològiques, com la transitivitat, no són suportades nativament per RDF.

Consulta	RDQL	TRIPLE	SeRQL	Versa	N3	RQL
Recursió	No	Sí	No	Sí	Sí	No

Taula 4.10: Expressions de recursió

Els llenguatges TRIPLE i N3, que són basant amb regles, suporten de manera natural la recursió mitjançant la definició de regles auxiliars. El Versa no suporta la recursió general, però té una sentència ("*traverse*") amb efectes transitius sobre una propietat especificada.

4.2.7.6 "Reificació"

La reificació és una propietat única de l'RDF, afegeix una capa al graf RDF i permet tractar les declaracions com si fossin recursos, tal com es poden fer declaracions de declaracions.

Consulta	RDQL	TRIPLE	SeRQL	Versa	N3	RQL
"Reificació"	Parcial	Parcial	Si	Parcial	No	Parcial

Taula 4.12: Expressions de reificació

Només el llenguatge TRIPLE suporta de manera nativa la reificació amb una sintaxis especial, on les definicions poden ser "reificades" (*reified*) mitjançant claus usant la sentència *FORALL*. Podem expressar les consultes amb Versa i RDQL utilitzant la propietat *<rdf:subject>* per a les declaracions "reificades".

4.2.7.7 Col.leccions i contenidors

El llenguatge RDF permet definir grups d'entitats utilitzant col.leccions i contenidors, anomenats, *Bag*, *Sequence* i *Container*, els quals ens donen un significat detallat dels elements que contenen. Un llenguatge de consulta té que ser capaç de recuperar individualment i en conjunt tots els elements d'aquestes col.leccions i contenidors, conjuntament amb d'altra informació.

Consulta	RDQL	TRIPLE	SeRQL	Versa	N3	RQL
Recursió	Parcial	Parcial	Parcial	Parcial	Parcial	Parcial

Taula 4.14: Expressions de col.leccions i de contenidors

Encara que cap dels llenguatges de consulta ens proporcionen suport explícit per al processament de contenidors, en tots els llenguatges és possible la consulta d'un element particular d'un contenidor amb l'ajuda del predicat especial: *<rdf:_n>*, que permet adreçar l'element n d'un contenidor.

4.2.7.8 Namespaces

Els *namespaces* són una part integral de qualsevol llenguatge de consulta de dades basat amb la Web. Donat un conjunt de recursos, pot ser interessant consultar tots els valors de les propietats de cert *namespace* o d'un *namespace* amb un determinat patró.

Consulta	RDQL	TRIPLE	SeRQL	Versa	N3	RQL
Namespace	Parcial	No	Sí	No	Sí	Sí

Taula 4.16: Expressions per Namespaces

Els llenguatges *SeRQL*, *RQL* i *N3* ens permeten fer coincidència de patrons fent coincidir predicats en URIs de la mateixa manera que els literals, els qual ens permeten realitzar una consulta RDQL.

4.2.7.9 Llenguatge

El RDF permet usar tags al l'estil XML. Els tags XML poden contenir literals RDF i poden portar opcionalment uns atributs <*xml:lang*>.

Consulta	RDQL	TRIPLE	SeRQL	Versa	N3	RQL
Llenguatge	No	No	Sí	No	No	No

Taula 4.18: Expressions de Llenguatge

El llenguatge *SeRQL* és l'únic que té un suport específic per a la consulta d'informació del llenguatge. El *SeRQL* té una funció especial per a la recuperació informació específica d'un literal. Això ho podem veure en el següent exemple:

```
select deLabel
from {} <rdfs:label> {deLabel, enLabel}
where lang(deLabel) = "de" and lang(enLabel) = "en" and
      label(enLabel) = "Administració de Bases de Dades"
```

Figura 4.16: Recuperació d'informació del llenguatge d'un literal

4.2.7.10 Literals i tipus de dades

Els literals d'utilitzen per a identificar valors, per exemple, nombres i dates pels significats d'una representació lèxica. Un llenguatge de consulta RDF sol suportar els tipus de dades del XML Schema. Un tipus de dades consisteix en un espai lèxic, un valor i un "mapeig" a aquest valor.

Consulta	RDQL	TRIPLE	SeRQL	Versa	N3	RQL
Espai lèxic	Sí	Sí	Sí	Sí	Sí	Sí
Valor Espai	Parcial	No	Sí	No	No	No

Taula 4.20: Expressions de literals i tipus de dades

Tots els llenguatges de consulta són capaços de consultar l'espai lèxic, però la majoria de llenguatges no suporten els tipus de dades ni tampoc la distinció entre l'espai lèxic i el valor d'aquest espai. *RDQL* i *SeRQL* ens proveeixen suport per a tipus de dades utilitzant una sintaxis especial per a indicar el tipus de dades.

4.2.7.11 "Entailment"

El vocabulari de l'RDF Schema suporta el "entailment" si l'informació és implícita. El "entailment" es centra en el següent:

- Dintre de l'RDF Schema les relacions entre les classes i les seves propietats no estan especificades.
- Classificació de recursos: per un recurs que té una propietat, podem fer que el recurs sigui una instància d'una classe.

Consulta	**RDQL**	**TRIPLE**	**SeRQL**	**Versa**	**N3**	**RQL**
"Entailment"	No	Parcial	Sí	No	Parcial	No

Taula 4.22: Expressions de "entailment"

Els llenguatges RDQL i Versa no donen suport al "entailment", SeRQL i RQL donen suport directe, i N3 i TRIPLE requereixen una "axiomatització" de la semàntica RDFS. Encara que el SeRQL suporta les semàntiques RDFS a través de la seva especificació, en la implementació de Sesame[3] la inferència la realitza més seu repositori que pel seu motor de consulta.

Un exemple de consulta per a avaluar la "reificació" de l'RDF Schema seria el següent: *Retornar totes les instàncies del membres de la classe Publicació.*

```
select publicacions
from ns3:Publicacio{publicacions}
using namespace ns3 =
  <!http://www.aifb.uni-karlsruhe.de/WBS/pha/rdf-query/sample.rdf#>
```

Figura 4.17: Exemple d'avaluació de "entailment"

[3]SGBD (sistema gestor de bases de dades) per a documents RDF que veurem en el capítol següent.

Capítol 5

Estudi d'SGBDs per a documents RDF

En aquest capítol es descriuran l'estructura i arquitectura d'alguns dels *Sistemes Gestors de Bases de Dades* (SGBD) existents en l'actualitat que treballen amb dades RDF. També s'avaluarà la manera en que recuperen i emmagatzemen les dades, fent menció de les seves característiques principals, i finalment es farà un estudi comparatiu d'alguns SGBDs estudiats, concretament es farà de *RDFSuite*, *Sesame* i de *RDF Gateway*.

5.1 Emmagatzemar dades RDF

El bloc bàsic de construcció de la Web Semàntica són les tripletes RDF, que estan formades pel subjecte, predicat i l'objecte. El subjecte conté un URI que identifica un recurs i el predicat és un URI a una taxonomia[1]. Una base de dades persistent RDF emmagatzema les dades de les descripcions RDF dels recursos i les ontologies com un esquema RDF (RDF Schema), llavors l'informació és sol.licitada pels clients mitjançant consultes.

La manera més fàcil de emmagatzemar les dades RDF és escriure totes declaracions en arxius XML (l'estàndard RDF/XML descriu com introduir l'RDF dintre d'XML). Un altra sintaxis és la *Notacio3* (N3, 3.3) que consisteix en escriure l'RDF en arxius plans de text. Les bases de dades relacionals existents poden ser utilitzades per emmagatzemar les tripletes dintre de taules, i també els índexs optimitzats.

El disseny de com representar les tripletes RDF dintre d'aquestes bases de dades relacionals i el construir índexs útils per incrementar la rapidesa de la bases de dades és la finalitat de les bases de dades RDF. La finalitat

[1] Es defineix com l'estudi dels principis de la classificació científica, classificació sistemàtica; especialitzada: *per exemple la classificació ordenada de plantes i animals en relació a les seves propietats naturals.*

d'alguns sistemes de la Web Semàntica és la publicació de les dades existents en format RDF, aquestos sistemes no necessiten un repositori RDF per a emmagatzemar les dades ja que el que fan és extreure l'informació de les bases de dades existents. A mesura que la Web Semàntica evoluciona es fa necessària la creació de sistemes d'emmagatzematge natiu en RDF, ja que les aplicacions tenen que ser estar basades íntegrament en el format RDF.

5.2 Descripció d'alguns SBGDs

5.2.1 FORTH-RDFSuite

RDFSuite és un framework[2] d'eines d'alt nivell creada per l'Institut d'Informàtica d'ICS-FORTH[3] ubicat a Heraklion, Grècia, i és suportat pels projecte de la UE (Unió Europea). El treball més rellevant en l'RDFSuite es va fer entre el 1999 i el 2000, i va servir de model per d'altres projectes RDF.

5.2.1.1 Arquitectura d'RDFSuite

Tres aplicacions formen la suite. El VRP (*Validating RDF Parser*) llegeix les dades RDF i les valida. El RSSDB (*RDF Schema Specific Data Base*) llegeix la sortida de l'VRP i ho emmagatzema en tripletes. El llenguatge de consulta RQL interpreta les consultes i recupera i les retorna fora de l'RSSDB. Aquestes tres aplicacions poden ser usades independentment.

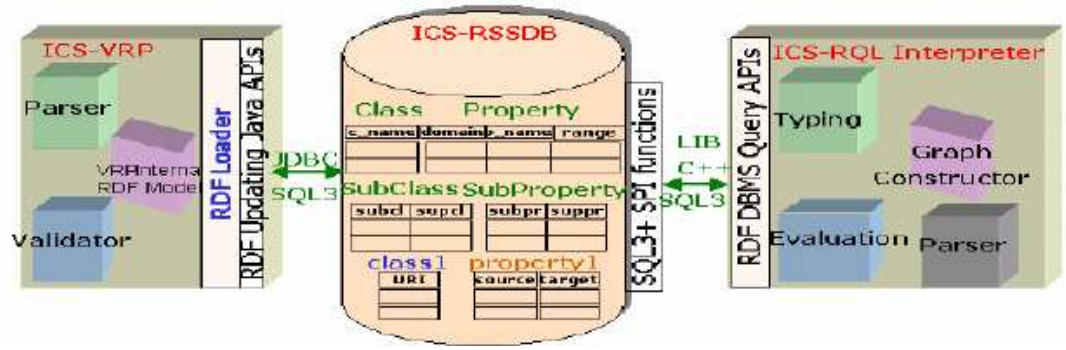

Figura 5.1: Arquitectura d l'ICS-FORTH RDFSuite

[2]S'entén el framework com una capa per sobre de la bases de dades.
[3]Institute of Computer Science of the Foundation for Research and Technology.

VRP - Validating RDF Parser

Està dissenyat per analitzar, validar i processar les descripcions RDF. Està basat en el generador de compiladors CUP/JFlex, el qual el distingeix d'altres eines RDF que utilitzen els parsers RDF per a llegir les dades. L'VRP entén l'RDF incrustat dintre l'XML i dona suport complert al mapa de caràcters *Unicode*. Les dades de l'Schema RDF són separades de les tripletes, les dades de l'esquema (Schema) es verifiquen contra la seva especificació i les restriccions addicionals es verifiquen contra el FORTH. Les dades RDF són verificades contra l'informació ontològica continguda en l'RDF. La sortida de l'VRP són tripletes que poden ser utilitzades per d'altres aplicacions, especialment per la entrada de l'RSSDB.

RSSDB - RDF Schema Specific DataBase

El nucli de l'RSSDB són dos mòduls Java per a carregar i modificar les dades, els mòduls implementen funcions per inserir, esborrar i per modificar tripletes. Les tripletes s'emmagatzemen en una base de dades relacional, i els mòduls de Java utilitzen el JDBC[4] per a connectar-se a la base de dades. L'RSSDB crea les taules en la base de dades relacional i els *Namespaces*, esquemes i tripletes s'emmagatzemen dintre. Les taules amb les tripletes només contenen referències a els *Namespaces* i a als esquemes continguts en les altres taules, també cal senyalar que és creen índexs per fer més ràpid l'accés a les taules.

RQL - RDF Query Language

L'RQL és un llenguatge de consulta que he descrit ja a la secció 4.2.2. Aquesta aplicació està formada per tres mòduls, el parser, s'encarrega d'analitzar sintàcticament les consultes; el segon mòdul és el *Graph Constructor*, que captura la semàntica de les consultes, i el tercer és l'*Evaluation Engine* que accedeix a les descripcions RDF a través de consultes SQL3.

5.2.1.2 Característiques

Entrada / Sortida del parser

El parser suporta diversos formats (RDF incrustat en HTML o XML, i XSD[5]). És capaç de processar arxius molt grans i no necessita cap software XML addicional. És possible la creació d'estadístiques dels esquemes validats i de les descripcions dels recursos. Un altra característica curiosa, és que pot donar la sortida en format SVC[6] per a poder visualitzar el graf.

[4]Java Database Connector.
[5]Schema Data Types.
[6]Scalable Vector Graphics.

Validació

Consisteix en una validació sintàctica de la correcció de l'XML i del RDF, o sigui, es valida que les descripcions RDF es corresponguin amb els arxius que contenen la definició de l'ontologia (veure 3.1.2).

Enumeracions

L'RSSDB estén la funcionalitat de l'RDF Schema afegint el tipus de dades d'enumeració que s'utilitza per a restringir una propietat fora dels valors establerts.

Funcionalitats per a consultes

- Tipus de dades XML Schema (per a filtrar valors literals).
- Primitives d'agrupació (per a la construcció de resultats XML recurrents).
- Operacions aritmètiques (per a la conversió de valors de literals).
- Funcions d'agregat (per a treure estadístiques).
- Facilitats dels namespaces (per a manegar diferents esquemes).
- Consulta de meta-esquemes.
- Recursivitat de classes i propietats jeràrquiques.

Facilitat de creixement

El disseny i l'implementació del sistema està pensat per a poder créixer. L'espai ocupat per de la base de dades és en funció de les tripletes carregades. El sistema és capaç de manegar i provar un directori amb uns 6 milions de tripletes.

5.2.2 Sesame

El Sesame és un repositori basat amb l'RDF Schema i a més una eina de consulta. Va ser presentat en el Projecte Europeu de l'IST (*Information Society Technologies*) On-To-Knowledge i va ser desenvolupat en un projecte de *Aiministrator Nederland*. Serveix com un eina bàsica d'emmagatzematge i consulta RDF, i s'ha continuat utilitzant amb posteriors entregues d'aquest projecte. Les eines OnToShare, RDF Ferret, Spectacle (també d'Aiministrator), IsaViz, OntoEdit i Jena poden treballar amb Sesame ([20]). La primera versió de Sesame va sortir a l'Octubre de 2001, i és de codi obert (*open source*). El Sesame està parcialment connectat a RDFSuite (5.2.1.1), ja que ambdós són part d'un Projecte Europeu de l'IST. El llenguatge de consulta RQL d'RDFSuite va ser adaptat i ara és part de Sesame i s'anomena *SeRQL*.

5.2.2.1 Arquitectura de Sesame

El Sesame està format per varies parts clarament separades cadascuna de l'altra i juntes donen les funcionalitats necessàries al seu servidor (figura 5.2).

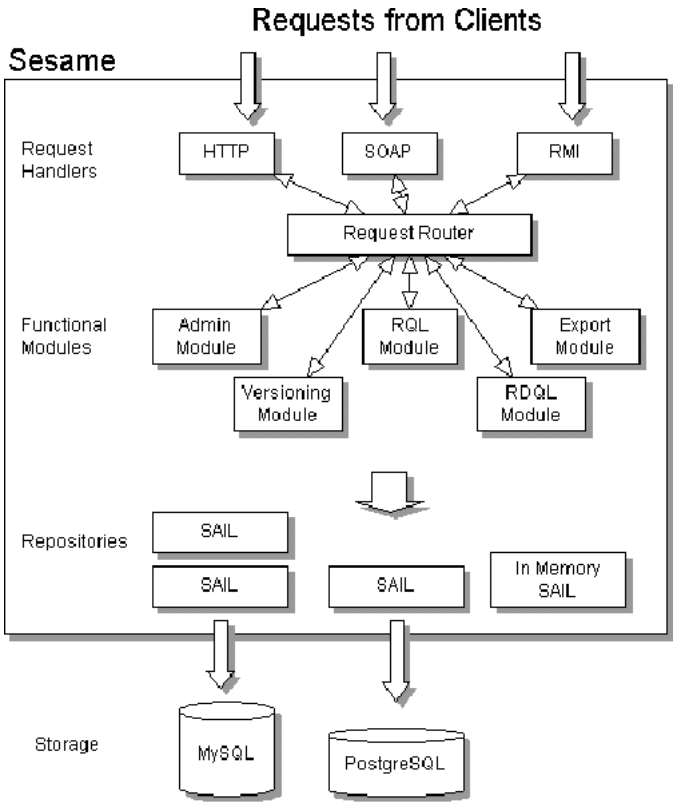

Figura 5.2: Arquitectura de Sesame

La capa d'inferència i d'emmagatzematge - SAIL

El Sesame pot emmagatzemar dades RDF en diferents bases de dades. Els mòduls funcionals no accedeixen a la base de dades directament, el SAIL els dona una interfície abstracta per connectar-se a la base de dades. El SAIL està format per mòduls, que són responsables de l'accés als sistemes d'emmagatzematge de relacions, com bases de dades relacionals i arxius, i també conté l'inferència funcional. Es poden apilar els mòduls SAIL, un a sobre de l'altre i d'aquesta manera afegir funcionalitat al servidor. Les crides a un SAIL apilat són propagades cap baix de la pila, fins l'últim mòdul SAIL. Finalment es processa la petició i el resultat es propaga cap dalt un altre cop. Es pot utilitzar un mòdul de la pila per restriccions d'accés. Existeixen diferents versions de SAIL per les bases de dades suportades: PostgreSQL, MySQL, Oracle; i actualment s'està treballant per donar suport a DB2 i Poet.

Mòduls Funcionals

Totes les funcions que són disponibles per les aplicacions que utilitzen Sesame estan separades per mòduls funcionals que treballen independentment. Actualment hi ha cinc mòduls:

- Un mòdul d'administració per afegir i esborrar dades.
- Un mòdul RDF d'exportació que pot ser utilitzat per exportar dades (parcials) d'un repositori com un document RDF.
- Un motor de consulta RQL que pot ser utilitzat per avaluar consultes SeRQL (l'RQL va ser crear per FORTH).
- Un motor de consulta RDQL que pot ser utilitzat per avaluar consultes RDQL (RDQL és part de JENA[21]).
- Un mòdul de control de versions per ontologies.

Manegadors de protocols: El servidor publica les funcionalitats en diferents protocols: HTTP, JAVA RMI[7] i SOAP.

5.2.2.2 Característiques

RDF Schema

El Sesame té suport bàsic per RDF Schema. Les classes, propietats i herències són emmagatzemades en taules específiques de la bases de dades escollida. El llenguatge de consulta SeRQL té una sintaxis que el fa fàcil d'utilitzar, les consultes es fan segons les recomanacions del RDF Schema.

Càrrega de dades

Els registres RDF poden ser carregats de dues maneres, mitjançant una URL passada al servidor que apunta a un document o en format de text. El Sesame accepta dades RDF codificades en dintre d'XML o en format de tripletes.

Consultes SeRQL (derivat del RQL)

L'RQL va ser creat per l'institut ICS-FORTH per la eina RDFSuite. És un llenguatge de procediments declaratiu per RDF i RDFS. Aquest llenguatge va ser adaptat posteriorment per el Sesame i l'implementació del motor de consulta és diferent que en el RDFSuite. Les consultes són interpretades i traduïdes al model de consulta. En aquest model la consulta és optimitzada i cada node de l'arbre resultant s'avalua. Una consulta conté tripletes que contenen variables, el motor de consulta cerca les tripletes i posa els

[7] Remote Method Invocation

valors corresponents dintre de les variables. El resultat de les consultes són una llista de valors associats a les variables de la consulta i el format pot ser: HTML, RDF/XML, o notació XML especial. *El problema més gran que té el Sesame és que el resultat no conté el nom de les variables consultades, per tant, el programador té que fer coincidir aquestos valors per índex amb les variables associades.* Això canviarà en un futur.

Consultes RDQL

Hewlett Packard va crear l'RDQL per al Jena, i està basat amb *SquishQL* i *rdfDB*. La sintaxis d'RDQL és diferent de la de SeRQL però l'avaluació de processos és similar.

Extracció de dades

Les declaracions senzilles poden ser extretes mitjançant consultes. Per extreure el repositori sencer existeix una funció, i retorna totes les dades i l'informació del esquema, els format pot ser: RDF/XML, N-Triples o N3.

Esborrar dades

Les declaracions RDF poden ser esborrades del repositori, i la seva sel.lecció es fa per objecte, subjecte i predicat. Per a esborrar totes les declaracions d'un determinat subjecte es crida a la comanda d'esborrar sense objecte i predicat.

Interfície Web

El Sesame té un inteficie HTML que permet a l'usuari connectar-se al servidor, sel.leccionar el repositori amb el que vol treballar i accedir a la majoria de les seves funcions. També l'ajuda en el procés d'administració i testeig.

Explorador RDF

Una part de l'interfície web és l'explorador que permet a l'usuari navegar pel repositori. L'usuari entra l'URI d'un recurs i obté totes les sentències que té el recurs escollit.

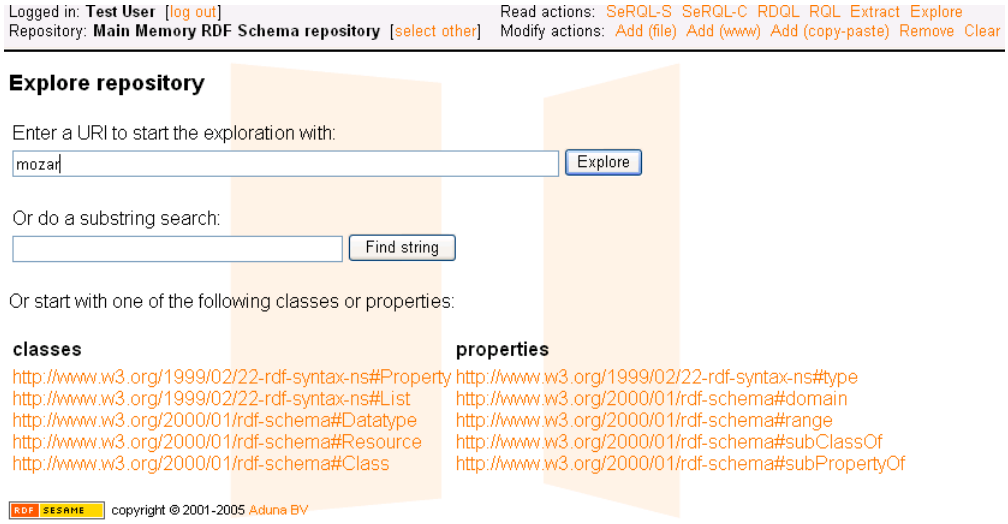

Figura 5.3: Mostra de l'explorador de Sesame

Múltiples repositoris i peer to peer

Un servidor pot manegar varis repositoris amb diferents opcions de seguretat. Cada repositori pot utilitzar diferents versions de SAIL. A més Sesame permet accedir a repositoris ubicats en d'altres servidors.

Escalabilitat

Existeixen diverses maneres d'utilitzar i estendre el Sesame. El mode bàsic és instal.lar-lo sobre el servidor Tomcat i amb un base de dades MySQL, d'aquesta manera una aplicació pot accedir-lo amb els seus protocols per desar i recuperar dades. Els desenvolupadors d'aplicacions poden afegir a Sesame els seus propis mòduls i integrar-los amb el servidor Sesame, també el poden afegir nous protocols, per tant, podem dir que és molt escalable.

5.2.3 RDF Gateway

Va ser creat per la companyia *Intellidimension* localitzada a la ciutat de Windsor. Només funciona sobre *Microsoft Windows* i encara no s'ha fet cap versió per Linux, es tracta d'un producte comercial i és necessària una llicència per a poder-lo utilitzar.

5.2.3.1 Arquitectura d'RDF Gateway

RDF Gateway és un servidor "*lleuger*" i ràpid que combina la gestió de bases de dades amb un servidor web, està dissenyat per la recollida, consulta i transformació de dades RDF. La seva peça central és un processador d'scripts que dona accés a totes les funcions de l'RDF Gateway,

la sintaxis del seu llenguatge de consulta (RDFQL) és similar a la del Javascript. Inclou algunes extensions per a la execució de consultes en el motor de bases de dades i donar accés a d'altres funcions del servidor.

Les aplicacions que utilitzen aquest servidor tenen que estar programades íntegrament en *RDFQL Script* i tenen que ser instal.lades com a part del servidor RDF Gateway, un altra manera d'interactuar amb el servidor és utilitzar les consultes RDFQL sobre ADO o JDBC, i d'aquesta manera editar i consultar les tripletes. L'RDF Gateway s'utilitza per a treballar amb diversos projectes de codi obert, per exemple, *l'RDF parser Raptor de David Becket* utilitzat per a entrar descripcions RDF.

Processador d'Scripts RDFQL

El processador d'scripts RDFQL és una màquina virtual que compila i executa els scripts RDFQL. L'RDFQL és un llenguatge de scripting orientat a servidors que integra extensions similars a l'SQL per a donar fàcil accés al motor de bases de dades deductiu de l'RDF Gateway. L'RDFQL permet que les pàgines continguin una mescla entre script i contingut estàtic, similar al de les ASPs[8].

Motor de bases de dades

L'RDF Gateway és un motor de bases de dades deductiu que està dissenyat per donar suport a l'RDF. Avalua les consultes de baix a dalt sobre totes les fonts de dades suportades. L'inferència lògica possibilita que es doni suport per a sintaxis de regles declaratives en RDFQL, i el motor de bases de dades deductiu implementa la persistència del arxiu i la possibilitat de cerca.

[8] Microsoft Active Server Pages (ASP).

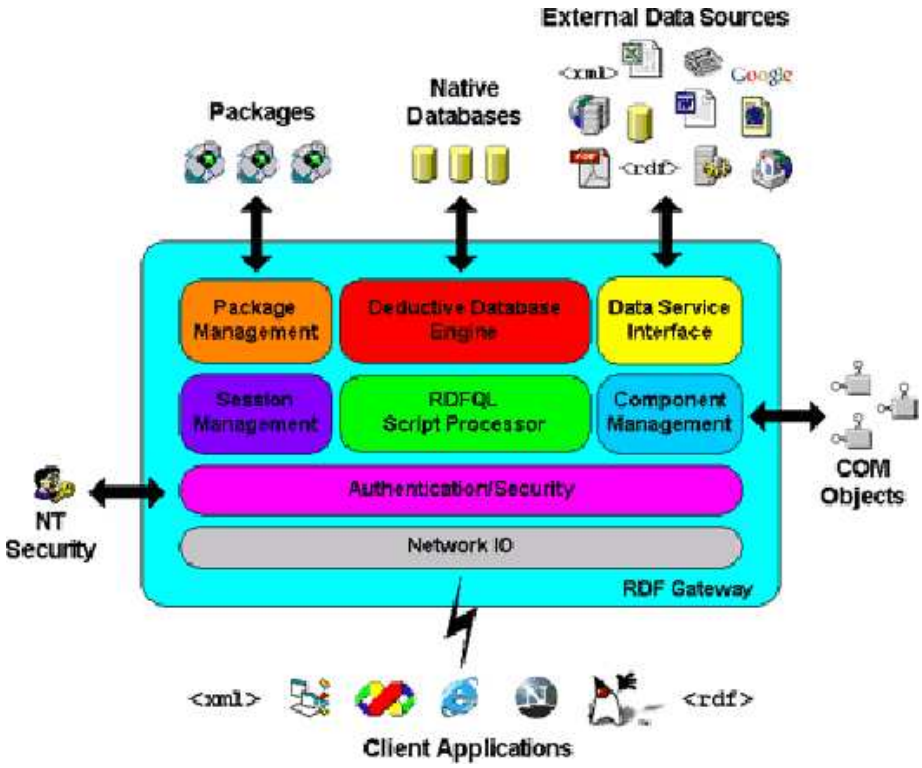

Figura 5.4: Arquitectura d'RDF Gateway

Data Service Interface

Permet l'accés a fonts de dades externes i que puguin ser integrades dintre de l'RDF Gateway. Un *data service interface* és un mòdul que implementa una interfície que mostra els continguts d'una font de dades com dades RDF.

Autentificació/Seguretat

L'RDF Gateway és un model de seguretat basat en els drets i permisos per a controlar l'accés als recursos del servidor i de la base de dades.

5.2.3.2 Característiques

Representació de tripletes RDF com dades

El paradigma del SGBDs relacionals de desar l'informació en taules ha estat adaptat per l'emmagatzematge de tripletes RDF. Les tripletes s'emmagatzemen en taules que no tenen nom de camps, i les tuples primer contenen el predicat. Existeixen a més quatre columnes per emmagatzemar metadades de les tripletes (aquesta informació s'anomena *context de la tripleta*). El camp "contexte" pot emmagatzemar l'identificador del re-

curs que pot ser utilitzat per a la gestió dels temes de seguretat, entre d'altres.

Bases de dades

L'emmagatzematge de les taules és dividit en bases de dades. Un servidor pot contenir diferents bases de dades i el seu format depèn de la base de dades amb la que es treballi.

El llenguatge d'scripts RDFQL

Aquest llenguatge es coneix com Javascript i implementa els següents conceptes: funcions, variables i arrays, bucles, control d'excepcions, importació d'altres scripts, comentaris i declaracions específiques d'RDF Gateway.

Recuperar i desar dades

Les comandes de manipulació de dades són similars a les de l'SQL, i es pot estendre la seva funcionalitat segons es requereixi. *Per exemple, podríem utilitzar la següent comanda per afegir una tripleta en RDFQL:*

```
INSERT {
  [http://www.arxiu.com/]
  [http://www.icom.com/schema.rdf#tecnica]
  [http://www.arxiu.com/mozart/requiem.jpg]
  'El Requiem'
} INTO composicio;
```

Figura 5.5: Exemple en RDFQL a RDF Gateway

Analitzador de consultes

Les declaracions RDFQL i les seves consultes poden ser creades utilitzant aquesta aplicació visual. L'analitzador de consultes és similar al avaluador de consultes del servidor SQL. És poden fer complexos scripts que poden ser utilitzats en pàgines web d'altres aplicacions. Les consultes poden ser avaluades contra un servidor Gateway local o remot.

El motor d'inferència

L'RDF Gateway conté un motor d'inferència que permet la generació de noves declaracions RDF basades en regles d'inferència i amb les tripletes existents.

Escalabilitat

L'RDF Gateway es distribueix en arxius binaris, i no està dissenyat per ser reprogramat. Es subministra un framework per aplicacions que permet la construcció d'aplicacions per sobre d'RDF Gateway. És possible estendre la funcionalitat de l'RDF Gateway afegint components al servidor, per exemple: components Active X, llibreries DLL.

5.2.4 D'altres SGBDs

5.2.4.1 3store

Va ser desenvolupat sota llicència GNU per Unix i està implementat amb C com una capa d'abstracció sobre una base de dades relacional. Utilitza un esquema que ha estat dissenyat per al emmagatzematge eficient i per a la recuperació de tripletes, i utilitza el *Raptor tookit* com a parser de RDF/XML.

Es distribueix amb la base de dades MySQL com a back-end. Encara que l'optimitzador de MySQL no és gaire sofisticat és adequat per a que es puguin optimitzar les consultes sobre els grafs RDF, que són transformades posteriorment en consultes SQL.

Característiques

- Les capacitats d'inferència de *3store* són implementades com regles de producció, que intenten buscar un equilibri entre l'optimització de les consultes i la reducció de les dades emmagatzemades.
- Utilitza una interfície lleugera mitjançant la qual els clients RDF es poden connectar, utilitzant *OKBC*[9], a la base de coneixements mitjançant una sèrie de Web Services creats per manegar aquesta base de dades de coneixement (*Open Knowledge Base*).
- Las interfície OKBC-HTTP s'utilitza en un gran nombre d'aplicacions el que potència la compatibilitat de 3store. El llenguatge de consulta utilitzat per 3store és similar al RDQL.
- Les aplicacions existents utilitzen molt la capacitat d'emmagatzematge de procediments del OKBC API, el qual indica que les crides simples a l'API no són suficients per desenvolupar aplicacions complexes.
- La interfície RDQL proporciona una interfície HTTP que retorna el resultat en format XML, i a més una *API C* que permet la consulta directa de la base de coneixement.

[9]Open Knowledge Base Conectivity.

5.2.4.2 Jena

El Jena és un eina Java per manipular models RDF i ha estat desenvolupada als laboratoris de Hewlett-Packard pel Jena Team. Suporta molt bé les consultes RDQL, però no és la eina adequada per treballar en amb grans volums de dades.

Inclou:

- Mètodes per a manipular models RDF com si fossin un conjunt de tripletes.
- Mètodes de recursos per manipular models RDF com un conjunt de recursos amb propietats.
- Crides a mètodes en cascada per a facilitar la programació.
- Un API basada en models i grafs.
- Emmagatzematge persistent i en memòria.
- Un parser RDF.
- Un llenguatge de consulta
- Suport per ontologies en DAML+OIL.

El Jena té un mètode d'abstracció d'emmagatzematge que permet integrar subsistemes d'emmagatzematge. El seu mètode d'emmagatzematge està basat amb el SQL.

Arquitectura

- **Parser ARP:** ARQ és un nou parser RDF dissenyat per a complir les darreres especificacions del "working group". L'ARQ és el parser per defecte de Jena.
- **Llenguatge de consulta RDQL:** L'RDQL és un motor de consulta per Jena, que suporta el llenguatge l'SquishQL (veure 4.2.1.1).
- **Un API DAML+OIL:** El DAML+OIL és un estàndard emergent per a codificar ontologies per a l'ús web, i d'altres serveis online. El Jena ens proporciona un API per accedir i manipular ontologies en DAML.
- **emmagatzematge persistent Berkeley DB:** Les darreres versions de Jena inclouen un mecanisme d'emmagatzematge basat en el SleepCat BerkeleyDB que permet accedir a les implementacions en format BerkeleyDB.
- **Un mòdul RDB:** Aquest mòdul ens permet l'emmagatzematge persistent en bases de dades relacionals. Suporta: *PostgresQL*, *MySQL*, *Oracle* i *Interbase*.

Característiques

- **Implementació:** L'implementació SQL d'aquest sistema d'emmagatzematge suporta la carrega de drivers codificats en Java que donen accés a diferents bases de dades. Les darreres versions suporten dues variants de visualització de tripletes.
- **Consultes:** El llenguatge de consulta de Jena és RDQL, que té una sintaxis i un API de consultes que extreu l'informació d'un model. El RDQL ens proporciona unes plantilles de subgrafs i de expressions booleanes en sintaxis SQL (veure SquishQL).
- **Inferència:** La versió darrera no té mecanismes d'inferència. L'RDQL no ens proporciona mecanismes d'inferència.
- **Escalabilitat:** Desconeguda. Les limitacions es deuen a la tecnologia d'emmagatzematge.
- **Potència:** Es capaç d'emmagatzemar, utilitzant mètodes SQL, en 10ms, i de realitzar consultes en 7ms.
- **API Java:** Suporta transaccions a través del backend (base de dades escollit), el seu subsistema BDB no suporta transaccions actualment.
- **Plataforma:** Utilitza Java 1.2 i superiors. Funciona sota Microsoft Windows i també amb Linux.

5.2.4.3 Parka

Parka és una sistema d'inferència escrit sobre un back-end d'una base de dades relacional; la versió ParkaSQ ha estat modificada per que suporti RDF. L'execució de les consultes és manegada per un motor relacional, tal com en *3store* (5.2.4.1), emmagatzema les dades en taules amb diferents formats.

El Parka és limitat a treballar amb bases de dades de al voltant de 2.5 milions de tripletes, el qual és degut a l'estructura dels seus índexs, que el fan inadequat per treballar amb volums de dades més grans.

5.2.4.4 Redland

Va ser creat per Dave Beckett a l'ILRT (Institute for Learning and Research Technology) de la Universitat de Bristol. El RedLand va ser dissenyat i implementat a l'any 2000 amb una aproximació modular. La seva arquitectura es basa en l'emmagatzematge en tripletes (*librdf_storage* en Redland). La Suite Redland apareix com un repositori capaç d'emmagatzemar grans volums de tripletes RDF, però no facilita eines per fer coincidència de grafs.

Identitat en RedLand

En RedLand una tripleta RDF i els termes de la tripleta: subjecte, predicat i objecte són nodes (librdf_node) de tres tipus; les referències a els URIs RDF són coneguts com recursos, nodes buits o literals (inclouen tipificació de dades). Aquesta estructura permet que els nodes de les tripletes es puguin copiar simplement per referència, i d'aquesta manera es facilita molt l'emmagatzematge.

Els grafs RDF no proporcionen cap altra estructura dintre del graf entre el nivell en tripletes i el graf sencer, tal com subgrafs. El Redland inclou una nova funcionalitat anomenada "Redland Contexts" que permet afegir grafs a les dades (tripletes/grafs) originals.

API per la recuperació de tripletetes en Redland

L'API del Redland permet que les tripletes siguin afegides, esborrades i cercades. El Redland no té mètodes per emmagatzemar nodes sols i arcs, però si permet navegar dintre del graf usant mètodes de model que retornen l'objecte d'una tripleta amb un determinat node al final de l'arc.

L'informació recuperada d'un graf RDF pot ser consultada preguntant per certes parts de les tripletes, això retorna conceptes RDF, per exemple, tripletes i nodes.

Desar tripletes en Redland

Existeixen dues classes per desar tripletes en Redland, una que s'anomena "memory" i un altra que s'anomena "hashes" (librdf_hash). El Redland utilitza librdf_hash per als valor duplicats d'una clau, això permet que les tripletes duplicades puguin ser emmagatzemades.

Contexts

Una propietat important de Redland és que suporta múltiples grafs RDF i també permet afegir grafs. Permet que les tripletes siguin inserides des de múltiples fonts (documents) i llavors amb el l'ús de la unió de grafs, l'informació retornada pel model de Redland proporciona un enllaç a la font original.

Característiques

- *Consultes:* Coincidència de tripletes.
- *Inferència:* Cap.
- *Escalabilitat:* Desconeguda. Redland ha estat testejat amb més de 1.5 millions de tripletes.
- *API:* C (nativa), Perl, Python, Tcl. Compila C++ i el suport Java no ha estat testejat.

- *Plataforma:* Està basat en POSIX, i ha estat testejat sota: Linux, Solaris, OSF/1 Alpha, FreeBSD, MacOS X.

5.2.4.5 TAP

El TAP KB és un SGBD per emmagatzemar tripletes RDF que apareix a partir de la versió 2 de 3store. Ens proporciona un mòdul Apache, el TAPapache, que permet l'accés remot a les dades RDF emmagatzemades, tal com fa el 3store. No obstant, el TAPache no ens proporciona cap interfície per fer consultes de coincidències de grafs com fa l'RDQL, i només proporciona un mètode de coincidència de tripletes anomenat GetData.

5.2.4.6 KAON

El projecte KAON és una implementació d'un servidor RDF i està descrit a la guia de desenvolupament KAONDEV[10] i és part del projecte IST - 2001. És una implementació d'un repositori RD com un EJB[11] i pot ser usat amb servidors d'aplicacions J2EE.

La persistència per defecte la proporciona el mecanisme contenidor EJB i el mecanisme relacional s'utilitza per a la construcció d'ontologies. El servidor KAON és una interfície ontològica d'alt nivell que inclou mòduls per accedir als repositoris basats en emmagatzematge de tripletes o als basats amb RQL, i permet l'ampliació afegint mòduls d'inferència.

La versió del 2002 permet treballar amb bases de dades: compatibles *SQL2, SQL Server, PostgreSQL, DB2 i Oracle 9i.*

5.2.4.7 TUCANA

El *Tucana Knowledge Server* és un SGBD amb un sistema d'emmagatzematge escrit en Java. Implementa sistemes de seguretat en cas de caiguda del sistema per a poder salvar l'informació.

El seu llenguatge de consulta és el iTQL (Tucana Query Language), aquest llenguatge aporta la funcionalitat de que tots els canvis s'actualitzen directament sobre la base de dades després de cada comanda. Permet també treballar amb RDQL, l'iTQL és un llenguatge derivat de RDQL i ampliat amb extensions per:

- Autentificació per propòsits de seguretat.
- Suport a consultes distribuïdes.
- Anomenat de models.
- Tipificació de dades.
- Facilitats de fer cerques complertes amb el seu motor de consultes.

[10] Karlsruhe Ontology and Semantic Web Infrastucture Developer Guide.
[11] Enterprise Java Beans.

Característiques

- Suporta els següents tipus de dades: URI, String, Number, Date, DateTime.
- Permet fer consultes sobre altres servidors i utilitza un sistema d'anomenat similar al DNS.
- El Tucana està empaquetat amb un JAR i amb un WAR, això permet al servidor funcionar en mode individual o format part d'un servidor d'aplicacions, com el *WebSphere*.
- Suport inferència mitjançant RDFS i OWL.
- Les aplicacions client, poden accedir al Tucana utilitzant les següents APIs i mètodes: *Jena API*[12], *JRD API*[13], *Web Services (SOAP/WSDL), JavaBean, COM, JSP tab library, descriptors basats en XSLT, Tucana Interface*.
- Té una consola que permet l'administració remota distribuïda o la local.
- Pot manegar aproximadament 500 milions de tripletes en un sistema operatiu de 64 bits.

5.2.4.8 Oracle 10g - release 2

El producte *Oracle Spatial Network* en la versió d'*Oracle 10g - release 2*, suporta RDF, aquí ens podem adonar del la progressió d'aquest llenguatge en eines tant comercials.

L'Oracle Spatial Network permet el parsejat i l'emmagatzematge de les tripletes RDF sota el model MDSYS Schema. Una tripleta RDF (subjecte, objecte i predicat) és tractada com un objecte de la bases de dades. Com a resultat d'això, un document RDF contenint múltiples tripletes és vist con un conjunt de múltiples objectes de la base de dades.

L'Oracle en dona un tipus anomenat URIType per manegar instàncies de qualsevol URI (HttpUri, DBUri i XDBUri). Aquest tipus s'utilitza per emmagatzemar els noms dels nodes i dels enllaços en la xarxa RDF.

La taula del sistema *MDSY.RDF_MODELS$* és la que conté els models RDF i Oracle la manté automàticament, també existeixen d'altres taules del sistema per emmagatzemar d'altra informació relacionada amb l'RDF:

- Namespaces: *MDSYS.RDF_NAMESPACE$*.
- Sentències: *MDSYS.RDF_VALUES$*.

L'Oracle també tracta d'altres aspectes importants per RDF com són:

[12] *http://www.hpl.hp.com/semweb/jena.htm*
[13] *http://jrdf.sourceforge.net/*

- Reificació.
- Contenidors.
- Col.leccions.
- Regles i regles base.

Finalment, podem dir que també contempla diversos aspectes de seguretat, i una sèrie de funcions i primitives addicionals que és potenciaran en un futur, a mesura de que l'ús de l'RDF es vagi ampliant.

5.3 Comparativa entre SGBDs

En aquest apartat compararé alguns SGBDs estudiats anteriorment (5.2.1, 5.2.2, 5.2.3), aquestos són: RDF Suite, Sesame i RDF Gateway. Les característiques bàsiques d'aquests tres servidors són les mateixes, desar dades de manera permanent i recuperar-les en forma de tripletes RDF és la finalitat principal dels SGBDs RDF. La possibilitat de crear nova informació basada en les dades existents en combinació amb les ontologies, regles també s'inclou en aquestos servidors.

Cada SGBDs (repositori) va ser creat amb un objectiu concret i ens dona unes possibilitats determinades, les comparatives entre SBGDs són útils, però és el desenvolupador el que ha de determinar quin és el més útil en un determinat projecte.

	RDF Suite	**Sesame**	**RDF Gateway**
Interfícies Client			
Java RMI	Sí	Sí	No
HTTP	No	Sí	Sí
SOAP	No	Sí	Sí
ADO	No	No	Sí
JDBC	No	No	Sí
Formats d'Entrada			
RDF/XML	Sí	Sí	Sí
HTML incrustat	Sí	No	No
N-Triple	No	Sí	Sí
Característiques			
RDF-Schema	Sí	Sí	No
DAML+OIL	Sí	Sí	No
Validació d'Ontologies	Sí	No	No
Consultes RQL	Sí	Sí	No
D'altres Consultes	No	RDQL (1)	RDFQL (2)
Motor d'Inferència	Sí	Sí	Sí (3)
Servidor Web	No	Sí	Sí (4)
Suport de Bases de Dades	(5)	(6)	(7)
Escalabilitat	Complicada	Mòduls Java	ActiveX, DLL
Codi Obert	Sí	Sí	No
Fàcil Instal.lació	No	Sí	Sí
Seguretat	(8)	(9)	(10)
Sistema Operatiu	Unix	Basat en Java	Windows 32
Model de Llicència	Compatible GPL	LGPL	Comercial

Taula 5.2: Taula de comparació entre SGBDs

1. *El llenguatge RDQL va ser creat pel projecte Jena.*
2. *És un llenguatge propietari de RDF Gateway.*
3. *Té que ser configurat.*
4. *Només és bàsic.*
5. *PostgreSQL, JDBC compliant.*
6. *PostgreSQL, MySQL, Oracle i pròximament d'altres.*
7. *Està basat en la seva pròpia BD.*
8. *Usuaris, nivell de servidor.*
9. *Usuaris, nivell de repositori.*
10. *Usuaris, fàcil administració i possibilitat de filar prim.*

5.3.1 RDF Suite

L'RDF Suite va ser creat dintre d'una comunitat científica i va ser testejat amb uns dels primers conceptes RDF. RDF Suite es centra en la potència, cadascun dels seus tres mòduls pot manegar grans volums de dades.

- El parser VRP utilitza el seu propi analitzador lèxic i pot manegar grans volums de dades d'una manera eficient.
- L'RSSDB ha estat testejat per diferents bases de dades i optimitzat per la millor rapidesa.
- El seu motor de bases de dades tradueix les consultes al format de la base de dades, aprofitant al màxim la seva potència d'optimització.

El servidor va ser creat en un entorn Unix i funciona millor sota aquest Sistema Operatiu. Els serveis de validació proveeixen les millors i més àmplies possibilitats de manera que són els més òptims avui per avui. La seva documentació té que ser estudiada detalladament per poder ser entesa.

L'RDFSuite s'utilitza en projectes en els que es treballa amb grans volums de dades, per exemple, en l'Open Directory s'inclou 6 milions de tripletes. Un altra característica molt important és que és molt escalable.

5.3.2 Sesame

Tal com RDFSuite, el Sesame va ser desenvolupat en un projecte científic de recerca. El seu repositori va ser dissenyat per ser la base de diferents aplicacions que són construïdes sobre ell. El Sesame pot ser utilitzat en diferents plataformes, està escrit íntegrament en Java i és capaç de connectar productes diferents com Tomcat i MySQL.

El Sesame ens aporta el següent:

- Fàcil instal.lació.
- Independència de plataforma.
- Facilitat d'extensió de les seves funcionalitats.
- Suport als protocols comuns d'accés.
- S'utilitza molt.

L'inconvenient més gran de Sesame és el treball amb grans volums de tripletes RDF. L'arquitectura modular de Sesame no suporta l'optimització de consultes en la base de dades amb la que es treballi, però això es solucionarà en un futur. El seu sistema de seguretat és bàsic, alguns dels seus mòduls és necessari programar-los. El Sesame pot ser utilitzat fàcilment en petits projectes i pot ser modificat per qualsevol usuari, es a dir, es pot utilitzar com un repositori de petites eines.

5.3.3 RDF Gateway

Està centrat en la idea de que es pugui construir sistemes amb ell. La companyia *Intellidimension* dona idees i exemples de com fer-ho. Aquest servidor és fàcil d'instal.lar i d'utilitzar.

RDF Gateway està restringit a ser utilitzat sota Windows, pot incloure fonts de dades ADO i es pot connectar a altres plataformes mitjançant mòduls. Els seus clients poden accedir a RDF Gateway utilitzant ADO, JDBC i HTTP, d'aquesta manera el servidor es capaç de suportar sistemes de clientes heterogenis. És fàcil de crear aplicacions semàntiques que poder integrar diferents fonts de dades i informació distribuïda gràcies a les característiques d'RDF Gateway, aquestes són:

- Fàcil instal.lació.
- No dependència en d'altres llibreries.
- Servidor Web.
- Llenguatge d'scripting.
- Integració amb ActiveX.
- Els clients es poden connectar a través de varies interfícies amb emmagatzematge natiu RDF i processat ràpid de consultes.

El seu motor d'inferència no suporta RDF Schema i utilitza un format propietari per crear regles d'inferència, i permet la inclusió d'una comanda especial que avalua la consulta enviada al servidor. La falta de suport a ontologies natives i la falta de funcionalitats de validació el fan difícil de ser integrat amb noves ontologies i amb arxius de fonts desconegudes.

RDF Gateway és perfecte per a crear petites aplicacions per la plataforma Windows, per d'altres tipus de projectes és necessari fer algunes parametritzacions.

Capítol 6

Cas Pràctic: La Botiga de Música - MusicLand

En aquest capítol es descriu com s'ha portat a terme el desenvolupament d'una Web senzilla que fa consultes sobre una base de dades RDF creada amb el Sesame (veure 5.2.2), i obtinguda a partir dels fitxers RDF subministrats pel consultor de l'assignatura. Cal senyalar que s'ha treballat amb la base de dades MySQL com a back-end, i que per la creació del Web s'ha utilitzat el llenguatge PHP[1] (*Hypertext Preprocessor*), juntament amb la llibreria de classes *Phesame* i *PhesameExt* (veure D), per poder fer consultes directament sobre el Sesame.

6.1 L'ontologia de la Botiga

L'ontologia (veure 3.1.2) subministrada per fer la part pràctica és la del projecte SIMAC[2], que representa coneixements del món de la música. D'aquesta manera proporciona als amants de la música una nova manera de descriure, visualitzar, explotar, reproduir i organitzar les col.leccions de música.

[1]Acrònim de *Hypertext Preprocessor*, és un llenguatge de programació interpretat. Va ser crear per Rasmus Lerdorf el tercer trimestre de 1994, però no va ser fins al 8 de Juny de 1995 que va ser llançada la versió 1.0. S'utilitza entre altres coses per la programació de pàgines web actives, i destaca per la seva capacitat d'interactuar amb HTML.

[2]*Semantic Interaction with Music Audio Contents*, és un projecte europeu del IST (*Information Society Technologies*) amb la participació de la Universitat Pompeu Fabra entre d'altres institucions europees, que permet la representació semàntica de coneixements sobre el món de la música. La principal finalitat del projecte SIMAC és el desenvolupament de prototipus que permetin la generació automàtica de dades semàntiques d'arxius d'àudio, la seva explotació visual, la recuperació i la organització de col.leccions de música.

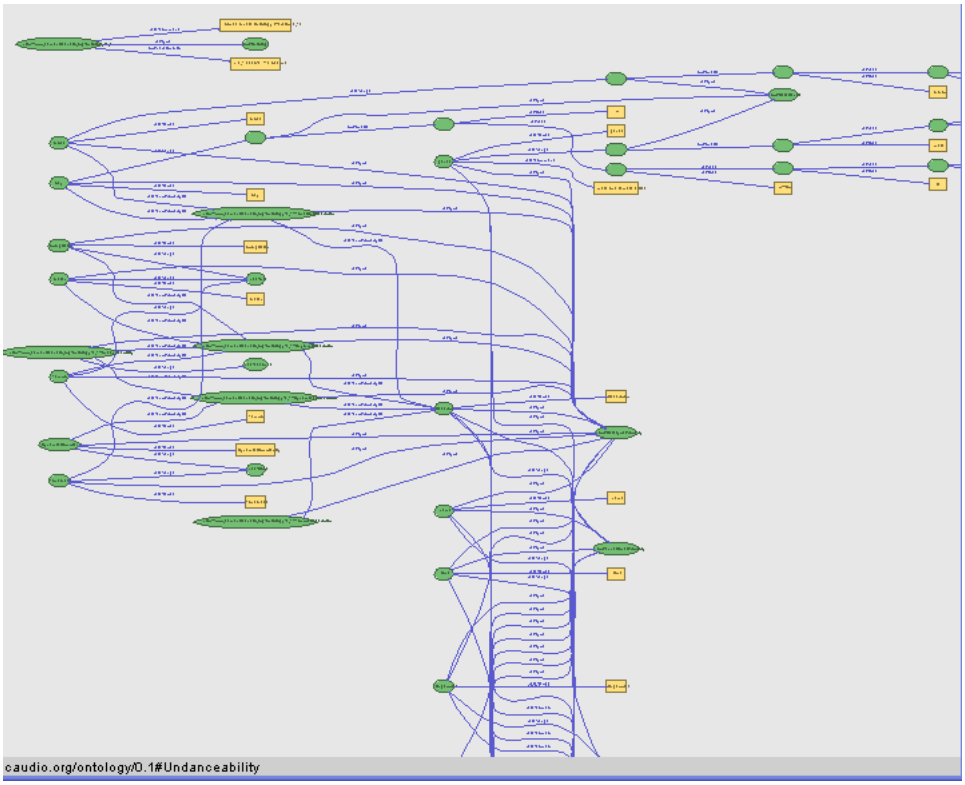

Figura 6.1: Ontologia de la botiga vista amb IsaViz 2.1

Com es pot apreciar, es tracta d'una ontologia molt complexa. En ella s'utilitzen totes les classes, subclasses, propietats i restriccions necessàries per poder emmagatzemar els coneixements semàntics del món de la música.

Evidentment, per a desenvolupar el cas pràctic d'aquest projecte no ens hagués fet falta una ontologia tant complexa i amb una de més senzilla n'haguéssim tingut prou, no obstant, fent servir aquesta ontologia s'amplia les possibilitats continuació d'aquest treball en futurs projectes.

En aquesta ontologia es pot veure com s'utilitza tota la potència semàntica del llenguatge RDF, d'aquesta manera es pot descriure còmodament els recursos del món de la música d'una manera natural i senzilla.

En la següent figura podem veure una mostra més detallada del disseny visual d'aquesta ontologia:

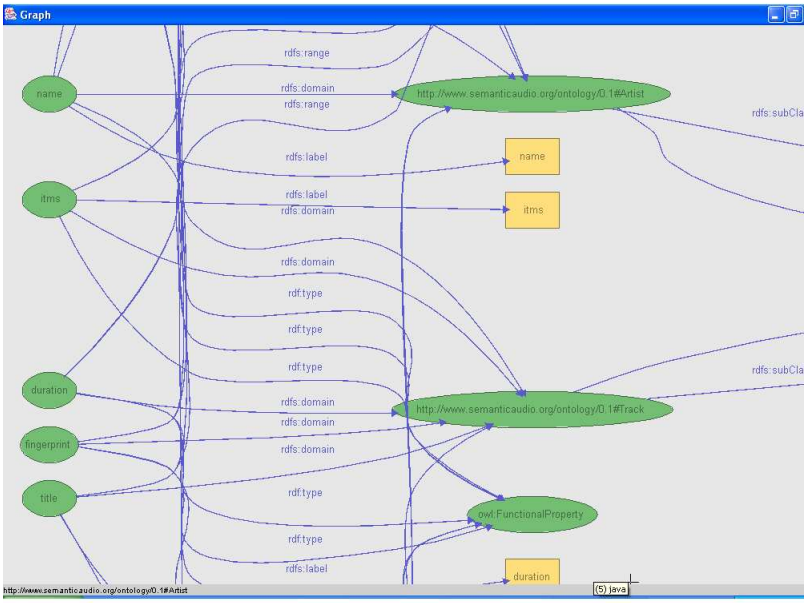

Figura 6.2: Restriccions i propietats de la ontologia

L'editor *IsaViz*[3] permet la exportació en format de tripletes (veure 3.3), anem a veure un fragment obtingut de la exportació de la ontologia donada:

[3] *IsaViz* és un entorn visual que permet la creació i visualització de models RDF en forma de grafs. La versió que he utilitzat en aquesta memòria és la 2.1, d'octubre de 2004.

```
<http://www.semanticaudio.org/ontology/0.1#name>
<http://www.w3.org/2000/0.1/rdf-schema#domain>
<http://www.semanticaudio.org/ontology/0.1#Artist> .
<http://www.semanticaudio.org/ontology/0.1#genre>
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://www.w3.org/2002/07/owl#DatatypeProperty> .
<http://www.semanticaudio.org/ontology/0.1#Undanceability>
<http://www.w3.org/2000/01/rdf-schema#subPropertyOf>
...
...
...
```

Figura 6.3: Mostra de la ontologia subministrada en format de tripletes.

6.2 Aplicació dels llenguatges de consulta RDF: La Botiga MusicLand

El cas pràctic desenvolupat en aquest projecte es centra en el món de la música, es tracta d'un Web d'una botiga virtual que permet fer consultes sobre les dades RDF subministrades pel consultor, entre d'altres coses.

6.2.1 Què és MusicLand

MusicLand és una botiga virtual que permet la consulta, compra i venda de música a través de la xarxa. Ofereix diferents possibilitats al internauta, la més rellevant és la de poder vendre la seva pròpia música permetent afegir noves composicions que queden enregistrades en la base de dades del sistema.

També, evidentment, podem realitzar compres de cançons cantades per diferents autors. Permet realitzar cerques per diferents patrons, per exemple, podem visualitzar els autors d'una determinada ciutat i un cop escollit un, podem veure les cançons que ha composat i comprar la que desitgem. Per a poder comprar un cançó tindrem que estar donats d'alta en la base de dades de la empresa o bé ho podrem fer online.

6.2.2 Interfície d'usuari

Aquest Web està estructurat amb un conjunt de frames (també podem anomenar-les parts) distribuïts de tal manera que permetin que l'usuari tingui en tot moment la informació necessària per la còmoda navegació pel mateix, a dir, que sigui molt usable.

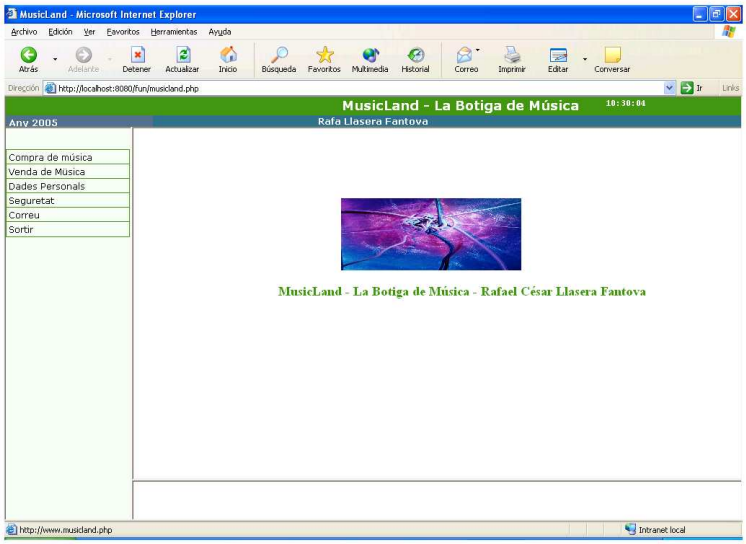

Figura 6.4: La botiga de música

Si ens fixem en la anterior figura, podem veure la distribució, en el frame superior se'ns mostra informació referent al nom del Web i l'hora actual, això últim s'ha aconseguit utilitzant un *Applet* de Java; a sota d'aquest frame n'hi ha un altre amb d'altra informació rellevant.

A la part esquerra ens apareix el menú de la botiga, es tracta d'un menú extensible fet amb *Javascript*[4], d'aquesta manera l'usuari podrà navegar còmodament i no perdre de vista en tot moment el lloc on es (el menú que apareix és el d'un usuari que ha entrat a la botiga i que té perfil d'administrador). En el frame de la part superior dreta es carreguen els manteniments i les consultes, així com les pantalles d'alta. En el frame de la part inferior dreta es visualitzaran els detalls de les pantalles a les quals s'accedirà, normalment, mitjançant enllaços.

[4]Llenguatge d'scripting, similar al Java i orientat a objectes, que es posa dintre de les pàgines HTML per donar dinamisme.

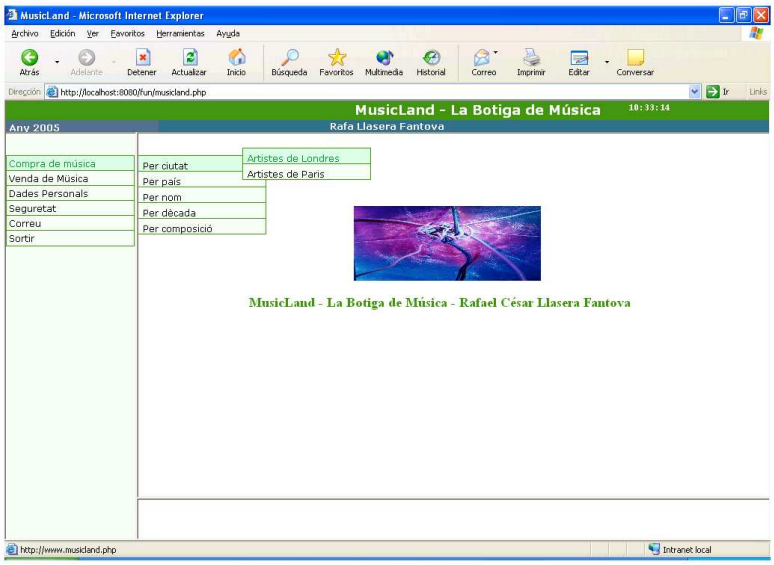

Figura 6.5: Menú de la botiga

En la següent figura es pot apreciar com és l'aspecte la consulta d'artistes, ens aquest cas, artistes de Londres. Es mostren els camps: nom, dècades i composicions. La pantalla permet filtrar per varis camps, un cop s'ha posat el filtre, s'ha de sel.leccionar el botó de recuperar, també se'ns ofereix la possibilitat de paginar els registres visualitzats utilitzant els botons de endavant i enrere. Cada pantalla mostrarà 20 registres, i tindrem que utilitzar aquestos botons de paginació per a poder visualitzar els següents.

```
$serql = "SELECT Composicions, Nom, Decades
        FROM {Composicions} rdf:type {ns2:Artist};
               ns2:name {Nom};
               ns2:decades {Decades};
               ns2:city {Ciutat}
               where label(Ciutat) = "London"
        USING NAMESPACE
        ns2 = <http://www.semanticaudio.org/ontology/0.1#>
```

Figura 6.6: Consulta d'autors de Londres.

En cas de que es vulgui filtrar per algun altre camp dels que permet la pantalla, es modifica la consulta afegint el filtre escollit i es torna a executar.

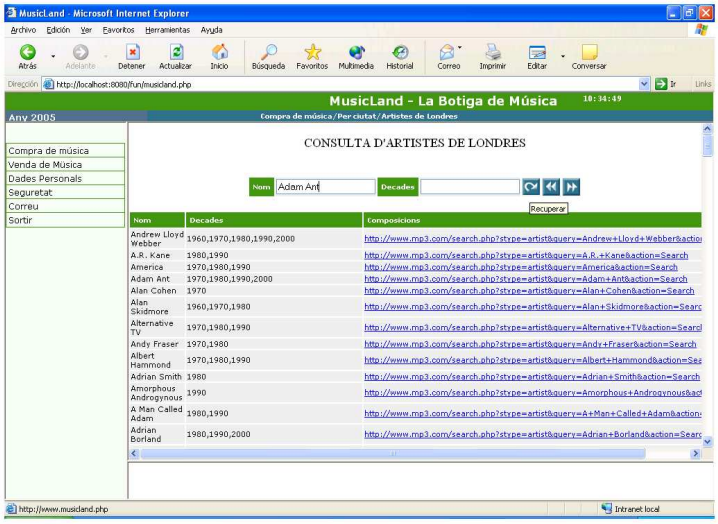

Figura 6.7: Detall de la consulta d'artistes

Aquest darrer camp té un enllaç[5] que ens permet accedir a la pantalla de detall del registre sel.leccionat, des d'aquí podem veure d'altres camps del registre com la nacionalitat i també ens permet des d'aquí fer la compra de la cançó (o composició), directament, sempre i quan estem enregistrats com usuaris.

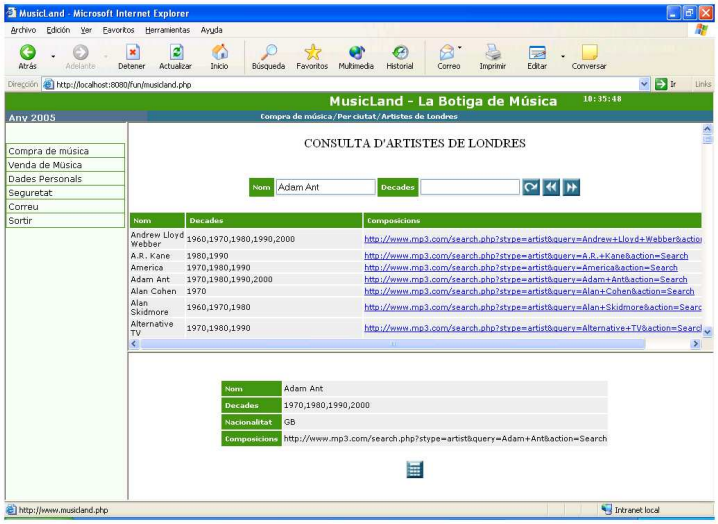

Figura 6.8: El detall d'artistes

L'anterior figura mostra la pantalla de *detall d'artistes*, que es visualitza

[5]Enllaç d'una pàgina que es coneix com *hyperlink* en anglès.

en el frame de la part inferior dreta. Podem observar en que apareixen en ella els camps corresponents al registre sel.leccionat.

El codi PHP que permet l'execució de la consulta utilitzada en aquesta pantalla de detall és el següent:

```
<html>
<body link="#f8fbb3" vlink="#f8fbb3">
<script language="JavaScript1.2" scr="cframe.js" type="text/javascript">
</script">
<?php
   require_once "phesame/class.PhesameExt.php";
   try
   {
      $ph = new PhesameExt("http://localhost:8080/sesame/servelts");
      $ph->login("testuser","testuser");
      $ph->setSelectedRepository("mem-rdf-db");
      $query = ' select Artistes, Nom, Decades, Nacionalitat
                from {Artistes} rdf:type {ns2:Artist};
                  ns2:name {Nom};
                  ns2:decades {Decades};
                  ns2:nationality {Nacionalitat};
                  ns2:city {Ciutat}
                  where label(Ciutat) = "London" and label(Nom)="Adam Ant"
                    using namespace
                    ns2 = <http://www.semanticaudio.org/ontology/0.1#>';
      $result = $ph->executeSeRQLQueryAndReturnTuple($query)
   }
   catch (Exception $e)
   {
      die($e);
   }
?>
</body>
</html>
```

Figura 6.9: Execució de la consulta autor: *Adam Ant* i ciutat: *Londres*

Com podem veure es filtra per ciutat i pel nom escollit, en el nostre cas, *Adam Ant.* També podem apreciar en aquest exemple com s'enllaça el PHP amb el Sesame, mitjançant les primitives de la classe *PhesameExt* (veure D). Un cop les variables han estat associades correctament, amb la consulta inclosa, s'executa la consulta i el resultat s'obté en format d'array.

Finalment, només cal tractar aquest array obtingut i muntar la pàgina tal com es desitgi. Com es pot apreciar, el mètode seguit és bastant senzill i a l'hora flexible i potent.

En el cas de clickar sobre el botó que surt en la pantalla, se'ns visualitzarà un altra pantalla amb la discografia de l'autor escollit. Des d'aquí podrem fer la compra de la cançó o cançons que més ens agradin.

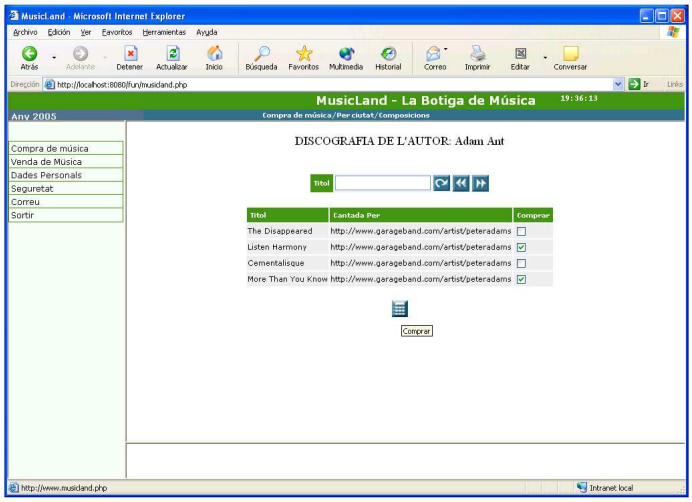

Figura 6.10: Discografia del autor escollit

El codi que s'executa en aquest cas és el següent:

```
<html>
<body link="#f8fbb3" vlink="#f8fbb3">
<script language="JavaScript1.2" scr="cframe.js" type="text/javascript">
</script">
<?php
   require_once "phesame/class.PhesameExt.php";
   try
   {
      $ph = new PhesameExt("http://localhost:8080/sesame/servelts");
      $ph->login("testuser","testuser");
      $ph->setSelectedRepository("mem-rdf-db");
      $query = ' select Discografia, Titol, CantadaPer
                from {Discografia} rdf:type {ns2:Track};
                  ns2:title {Titol};
                  ns2:played_by {CantadaPer}
                  where label(CantadaPer) = "Adam Ant"
                    using namespace
                    ns2 = <http://www.semanticaudio.org/ontology/0.1#>';
      $result = $ph->executeSeRQLQueryAndReturnTuple($query)
   }
   catch (Exception $e)
   {
      die($e);
   }
?>
</body>
</html>
```

Figura 6.11: Execució de la consulta: discografia de l'autor

6.2.3 Arquitectura utilitzada

Pel desenvolupament d'aquest Web s'ha utilitzat *Macromedia Dreamweaver Mx* com eina de disseny, així com funcions Javascript i Applets per la inclusió de dinamisme. Com a Sistema Gestor de Bases de Dades (SGBD) RDF s'ha utilitzat el que és sense cap dubte, el més popular avui per avui, el Sesame (veure 5.2.2), i s'ha utilitzat com a backend MySQL 4.0.13.

Per a poder connectar el Web amb Sesame es va estudiar la possibilitat de desenvolupar un Servlet de Java, però al final, per comoditat, s'ha optat per l'opció d'utilitzar el PHP 5.0 (*Hypertext Preprocessor*) amb algunes classes i extensions de llenguatge adaptades, que faciliten molt la tasca. Finalment, cal remarcar que les consultes s'han construït utilitzant el llenguatge *SeRQL* (veure 4.2.2.2).

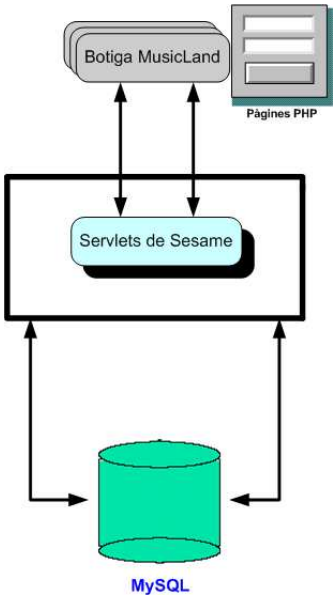

Figura 6.12: Arquitectura de MusicLand

Així doncs, podem l'arquitectura utilitzada per al desenvolupament i desplegament de la botiga MusicLand de la següent manera:

- *Macromedia Dreamweaver Mx* com a eina de disseny de les pàgines.
- *Jakarta Tomcat 4.1* com a contenidor de Servlets i Servidor Web.
- *Sesame 1.2.2* com a SBGD de dades RDF.
- *MySQL 4.0.13* com a backend de Sesame.
- *PHP 5.0* com llenguatge per enllaçar la botiga amb el Sesame (utilitzant les classes *Phesame* i *PhesameExt*).
- Pàgines PHP (.php) que formen la botiga virtual, la qual, utilitza com a base una estructura de frames.

Capítol 7

Conclusions

Les metadades són una part clau en l'infraestructura de l'informació necessària per ajudar a crear ordre dintre del caos de la Web, donant capacitat de descripció, classificació i organització que ajudi a crear magatzems d'informació més útils. Els orígens de les metadades, com els orígens dels recursos mateixos, seran de qualitat diferent i s'organitzaran al voltant de propòsits diferents per a senyalar els objectius dels propòsits de l'informació. Per a que aquestes oportunitats arribin a ser una realitat, serà necessària un certa convergència de formats de codificació i una semàntica consensuada.

L'RDF promet una arquitectura de metadades Web i s'ha desenvolupant com la "infrastuctura habilitadora" de l'activitat sobre la Web Semàntica dintre de l'W3C. Dissenyat per a promoure la reutilització i intercanvi de vocabularis, RDF és una capacitat addicional (semàntica) sobre XML (sintàctica) que simplifica la reutilització de termes de vocabulari entre namespaces.

L'RDF proporciona solucions que habilitaran un major grau de fiabilitat, rellevància, i exactitud de les aplicacions i serveis orientats al descobriment de recursos i a la gestió de sites Web i d'altres recursos d'Internet: fer semàntica que la màquina pugui entendre possibilitarà el descobriment automatitzat, gestió, i intercanvi de recursos; un major grau de detall en la descripció i de percepció per al descobriment de recursos; i l'interoperatibilitat semàntica entre diferents vocabularis de metadades que utilitzen una semàntica comú.

Es considera a RDFS (també a l'UML) com un model conceptual apropiat per a la descripció de recursos. No obstant, es considerarà un error el èmfasis en la seva sintaxis RDF/XML. RDFS com model de dades és més important que la seva sintaxis i no té que ser confós amb ella.

És necessari continuar amb els esforços de determinació de les primitives bàsiques de modelat proporcionades per *RDFS, OWL, UML* i els sistemes de marcs, així com determinar la millor manera d'implementació de les mateixes com a bases per a l'establiment de mapes de correspondèn-

cies entre els diferents models conceptuals de la Web Semàntica. Aquesta determinació possibilitarà omplir del buit entre el nivell sintàctic i el semàntic que està limitant el desenvolupament i implementació de les noves eines i aplicacions que suporten RDF i RDF/XML.

També és molt important la potenciació de les noves eines d'emmagatzematge en format RDF, com *Sesame*, i l'aplicació d'aquestes eines com eines de codi obert que puguin ser modificades pels usuaris lliurement, i d'aquesta manera possibilitar l'afegiment de nous mòduls que possibilitin l'ampliació de funcionalitats.

En aquest Projecte Fi de Carrera s'ha intentat aprofundir en el estudi dels llenguatges de consulta per documents RDF existents en l'actualitat, i s'ha intentat aplicar aquestos coneixement en un cas pràctic, fent consultes sobre un SGBDs amb emmagatzematge en format RDF, com és el Sesame.

Encara hi ha un llarg camí per recórrer, però cada cop es fa més necessari l'afegiment de semàntica la Web actual, per això, és de preveure un gran creixement d'aquest sector en els pròxims temps. És per això de que penso que en futurs projectes es pot aprofundir encara més en l'estudi dels SGBDs per a documents RDF, i fer més exemples pràctics amb els mateixos per a poder avaluar millor les seves característiques.

Glossari

Algae: Llenguatge de consulta per a documents RDF, que està basat en la sintaxis RDF de N-triples estesa amb algunes acciones i restriccions.

API: Interficie de programacio per al desenvolupament d'aplicacions.

Atribut: Component de l'element que actua com a modificador d'aquest, afegint informació addicional a l'element al que està associat.

Document XML: Document format únicament per text pla i que conté informació estructura i delimitada mitjançant marques.

Document XML ben format: Document XML que segueix les regles de marcatge correctament.

Document XML invàlid: Document XML que inclou o indica un DTD i no el satisfà.

Document XML vàlid: Document XML que inclou o indica un DTD i el satisfà.

DOM: *Document Object Model*, interfície de programació orientada a objectes que representa els documents XML mitjançant un conjunt d'objectes en forma d'arbre.

DTD: *Document Type Definitions*, document que defineix la gramàtica dels documents XML.

Element: Unitat bàsica amb capacitat per representar la lògica i la semàntica d'un document XML.

Element arrel: Element del document XML que penja directament del node arrel.

Esquema XML: *XML Schema*, document en sintaxis XML que defineix l'estructura, la gramàtica, el contingut i la semàntica dels documents XML de forma més robusta i estricta que els DTD.

HTML: *Hypertext Markup Language*, llenguatge de marques per a la creació de pàgines web.

Interficie: Plataforma de programació que proporciona un conjunt de classes i mètodes per facilitar les tasques de desenvolupament.

Namespace: És un mecanisme simple per a la creació de noms globals i únics per als elements i atributs del nostre llenguatge.

N3: Notació en tripletes.

N3QL: Llenguatge de consulta per a documents RDF, que és derivat de la *Notation 3*, una extensió sintàctica de l'RDF amb variables, regles i notació per a construir expressions d'expressions. Una consulta N3QL és una expressió N3. Totes les paraules reservades són propietats RDF d'un node representat a la consulta.

Ontologia: És una especificació formal i explícita d'una conceptualització compartida. Una conceptualització fa referència a un model abstracte d'alguns fenòmens en el món que identifiquen els conceptes d'aquest fenomen. Explícit significa que el tipus de conceptes usats i les seves restriccions s'especifiquen detalladament. Formal referència el fet de que un ontologia pot ser llegida per les màquines. Compartida reflexa la noció de que un ontologia captura el coneixement consensual, que no és quelcom aïllat sinó que és acceptat per un grup.

PHP: Acrònim de *Hypertext Preprocessor*, és un llenguatge de programació interpretat. Va ser crear per Rasmus Lerdorf el tercer trimestre de 1994, però no va ser fins al 8 de Juny de 1995 que va ser llançada la versió 1.0. S'utilitza entre altres coses per la programació de pàgines web actives, i destaca per la seva capacitat d'interactuar amb HTML.

RDF: *Resource Description Framework*, llenguatge que es fonamenta amb XML i que serveix per a descriure recursos (veure recurs).

RDF Schema: *Resource Description Framework Schema*, és un llenguatge situat per sobre d'RDF i presenta un conjunt simple de recursos RDF i propietats que ens permeten crear els nostres propis vocabularis RDF.

Recurs: La descripció d'un recurs pot ser molt àmplia, però ens centrarem amb veure un recurs com un document electrònic disponible mitjançant la Web i que pot ser accedit via URL.

RDQL: *RDF Data Query Language*, és una evolució del SquishQL i ha esta estandarditzat recentment per l'W3C. Les consultes RDQL tenen el mateix format que les consultes SquishQL.

RQL: *RDF Query Language*, és la base dels llenguatges SeRQL i del eRQL. La característica principal d'aquest llenguatge és l'ús d'esquemes RDFS.

SAX: Simple API for XML, interfície de programació simple basada en events i fàcil d'utilitzar per analitzar els documents XML.

SeRQL: *Sesame RDF Query Language*, és derivat del llenguatge RQL i l'utilitza l'SGBD Sesame.

SGBD: Sistema Gestor de Bases de Dades.

SGML: *Standard Generalized Markup Language*, llenguatge de marques que constitueix un estàndard extensible més potent que XML però més difícil d'implementar.

SQL: Llenguatge de consulta de les bases de dades relacionals.

Squish: És un llenguatge de consulta per a documents RDF que té un model de consulta propi per RDF i és bastant similar a SQL. Ofereix les anomenades plantilles de tripletes (*triple patterns*) i les conjuncions que permeten especificar les parts d'un graf RDF que volen ser recuperades.

TRIPLE: Llenguatge de consulta per a documents RDF, està basat amb consultes amb *regles*, *inferència* i *transformació* per a documents RDF. Ha estat dissenyat per a solucionar dues debilitats dels llenguatges de consulta RDF.

Tripleta: Element bàsic de la Web Semàntica que s'utilitza per a denotar les classes, propietats de les classes, i valors, podem crear jerarquies de classes per la classificació i descripció d'objectes, i que està format per un subject, un objecte i un predicat.

TriQL: Llenguatge de consulta per a documents RDF que estén la funcionalitat del llenguatge RDQL mitjançant el suport a la construcció de consultes amb grafs amb nom (*named graphs*).

URI: *Uniform Resource Identifier*, és una manera estàndard d'indentificar un recurs. L'URI és un terme genèric que fa referència a adreces i objectes del *World Wide Web*.

URL: *Universal Resource Locator*, direcció virtual d'un espai web.

Web Semàntica: Tracta de solucionar els problemes de la Web actual i intenta convertir la informació que contenen les planes web en coneixement que manera que pugin ser interpretades per les màquines.

W3C: *World Wide Web Consortium*, organisme creat amb l'objectiu de treure el màxim rendiment de la Web mitjançant el desenvolupament de protocols (especificacions, guies d'estil, software, etc.).

Xcerpt: Llenguatge de consulta per a documents RDF, permet la consulta de dades en la "*Web Estàndard*" i la consulta de dades en la Web Semàntica.

XLink: *XML Linking Language*, llenguatge que permet introduir enllaços en els documents XML, per tal de relacionar documents entre ells.

XML: *Extensible Markup Language*, llenguatge extensible de marques que defineix el format estàndard per a l'estructuració de dades i d'informació.

XMLS: Esquema *XML*.

XPath: Llenguatge que permet adreçar parts d'un document XML, seguint l'estructura lògica (arbre de nodes) que descriu el contingut del document.

XPointer: Llenguatge que permet identificar i fer referències a parts que es troben dins d'un document XML, mitjançant apuntadors o especificacions de camins.

XQuery: Llenguatge de consulta que basant-se en l'estructura dels documents XML, proporciona els mecanismes necessaris per poder interrogar, recuperar i interpretar l'informació.

XQuery per RDF: Llenguatge de consulta per a documents RDF que requereix una normalització preliminar de les dades RDF consultades, principalment per a serialitzar les dades RDF en XML i per agrupar les sentències pel seu subjecte.

XSL: *Extensible Stylesheet Language*, llenguatge que aplica format als documents XML, indicant com s'han de visualitzar i donant certa capacitat de transformació del contingut dels documents.

XSLT: *Extensible Stylesheet Language Transformation*s, llenguatge que permet transformar un document XML en un altre document XML.

XSLT-FO: *Extensible Stylesheet Language Formatting Objects*, llenguatge de formateig que descriu de forma precisa com s'ha de visualitzar el contingut dels documents XML.

XSLT per RDF: Llenguatge de consulta per a documents RDF, similar al llenguatge 4.10.

Apèndix A

Instal.lació de Sesame

Els següents passos descriuen la manera més fàcil d'instal.lar el *Sesame* sobre *Tomcat 4.x* o *5.x.* Aquest procediment no requereix la reconfiguració de Tomcat. Els passos són:

1. Instal.lar Tomcat. Normalment això consisteix en descarregar el Tomcat de *http://jakarta.apache.org/tomcat* i instal.lar-lo en una ubicació apropiada del disc dur. D'ara endavant em referiré al directori on està instal.lat el *Tomcat* com : *[TOMCAT_DIR]*. Veure la documentació de Tomcat per més informació i de com es posa en marxa.

2. Anar al directori d'aplicacions (*[TOMCAT_DIR]/webapps/* per defecte) i crear un directori '*sesame*' dintre.

3. Descomprimir l'arxiu *sesame.war* (que es troba en el directori *lib* de la distribució de *Sesame*) dintre del nou directori creat '*sesame*'. Això es pot fer a la línia de comandes amb la comanda: **jar -xf [PATH/TO/]sesame.war.** També podem utilitzar el WinZip o el upzip per descomprimir l'arxiu. Ens referirem al directori on hem descomprimir el *sesame.war* com *[SESAME_DIR]*.

4. En cas de que ens estem planejant d'utilitzar una base de dades amb *Sesame*, hem de copiar el driver JDBC apropiat en el directori *[SESAME_DIR]/WEB-INF/lib/*.

5. Canviar el nom de l'arxiu *[SESAME_DIR]/WEB-INF/system.conf.example* a *[SESAME_DIR]/WEB-INF/system.conf*. Només s'ha de fer això si és una nova instal.lació de *Sesame*, en el cas de que no fos una instal.lació nova destruiríem l'actual configuració. L'arxiu d'exemple subministrat conté algunes entrades de repositori per diferents bases de dades, i unes comptes d'usuari per defecte. L'arxiu pot requerir d'algunes modificacions per a poder treballar sota la nostra màquina. Veure l'apartat '*Server administration*' de la documentació de *Sesame* per més ajuda.

6. Engegar el servidor *Tomcat* i el Sesame es posarà en marxa. Per a poder accedir a la interfície web de Sesame ho hem de fer amb la URL: *http://[NOM_DE_MAQUINA]:8080/sesame* o *http://localhost:8080/sesame*.

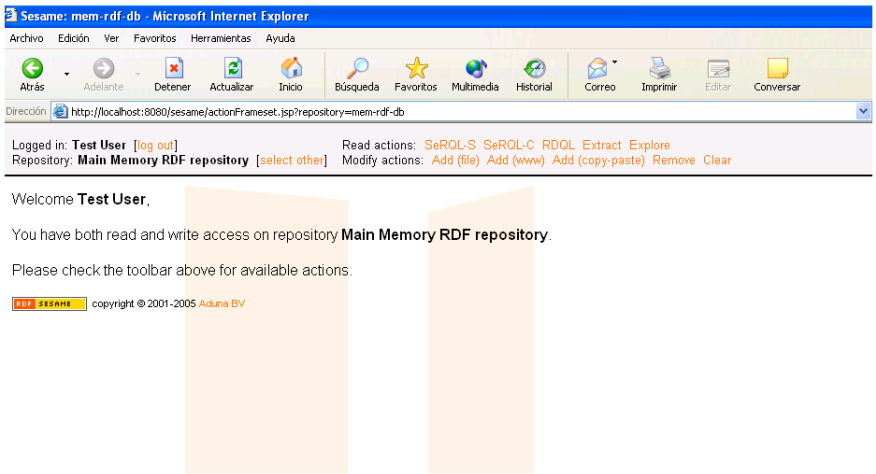

Figura A.1: Interfície web de Sesame

Apèndix B

Com instal.lar PHP 5.X sobre Tomcat

Els passos a seguir per a instal.lar el PHP 5.x sobre Tomcat són els següents:

- Descarregar l'ultima versió de PHP 5.x. Jo he utilitzat l'adreça: http://www.php.net/get/php-5.0.2-Win32.zip d'un mirror escollit (ubicat a Londres).
- Descarregar la última versió dels mòduls PECL. Jo he usat l'adreça: http://www.php.net/get/pecl-5.0.2-Win32.zip d'un mirror escollit (ubicat a Londres).
- Descomprimir l'arxiu PHP 5.x on es vulgui, en el meu cas **c:\php**.
- Canviar el nom de l'arxiu *php.ini-dist* ubicat a c:\php com *php.ini*.
- Descomentar la línia (esborrar el ; del començament) de l'arxiu *php.ini*: **;extension=php_java.dll**
- Descomprimir l'arxiu **php5servlet.dll** de l'arxiu zip de pecl a *c:\php*. Assegurem-nos de que hi és al directori *c:\php*.
- Instal.lar el Tomcat i crear un directori, anomenem-lo *fun*, dintre de la carpeta *WEBAPPS*.
- Crear un directori anomenat *WEB-INF* dintre de *fun*.
- Crear un directori *lib* dintre de *WEB-INF*.
- Crear un arxiu anomenat **web.xml** dintre de *WEB-INF* amb el següent contingut:

```
<?xml version="1.0" encondign='ISO-8859-1">
<!DOCTYPE web-app PUBLIC
"-//Sun Mircorsystems, Inc.//DTD Web Application 2.3//EN"
"http://java.sun.com/dtd/web-app_2_3.dtd">
```

```
<web-app>
    <servlet-name>php</servlet-name>
       <servlet-class>net-php-servlet</servlet-class>
    </servlet>
   <servlet>
       <servlet-name>php-formatter</servlet-name>
       <servlet-class>net-php-formatter</servlet-class>
    </servlet>
    <servlet-mapping>
       <servlet-name>php</servlet-name>
       <url-pattern>*.php</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
<servlet-name>php-formatter</servlet-name>
<url-pattern>*.phps</url-pattern>
</servlet-mapping>
</web-app>
```

- Descomprimir l'arxiu *php5srvlt.jar* i descomprimir-lo (jar xvf.....) sota la c:\.
- Modificar els arxius: *reflect.properties* i *servlet.properties*, canviant la línia **library=phpsrvlt** a **library=php5servlet** i desar-lo. Això indica el nom de la dll que carrega l'aplicació Java per servir les peticions. En la meva versió el nom de la .dll és: *php5servlet.dll*, però això pot canviar. No té res a veure amb el nom de l'arxiu .jar.
- Tornar a crear l'arxiu *.jar*.
- Copiar l'arxiu *.jar* al directori *WEB-INF\lib*.
- Afegir el camí **c:\php** al path del sistema o al path d'usuaris de l'entorn Windows (propietats de Mi-PC).
- Crear un arxiu **test.php** al directori *fun* amb el següent codi:
  <?php phpinfo(); ?>
- Inicialitzar el Tomcat (anar al directori d'instal.lació de Tomcat des de MS-DOS) i escriure *tomcat* o *run*).
- Obrir el navegador i anar a la URL: **http://localhost:8080/fun/test.php**.
- Assegurar-se de que no hi han errors. Ens ha d'aparèixer una pantalla amb l'informació de la versió de *php* i d'altres detalls.

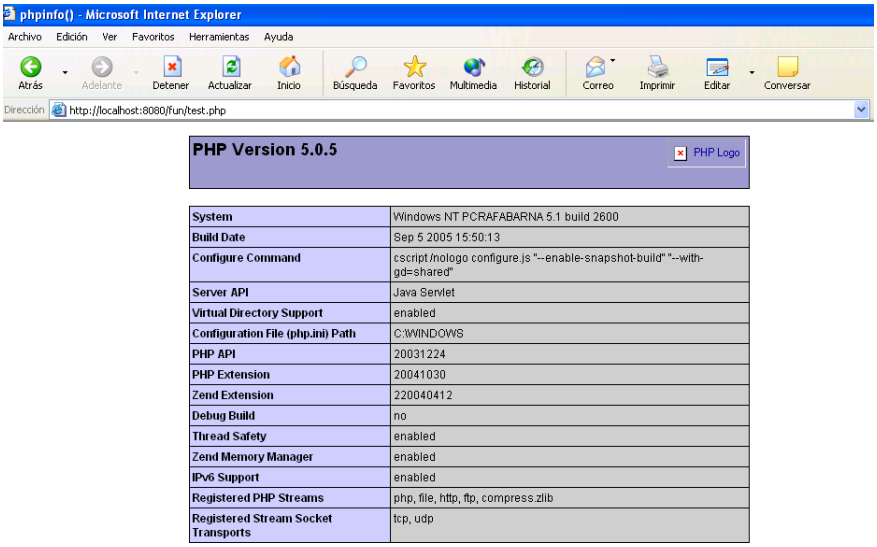

Figura B.1: Instal.lació de PHP sobre Tomcat 4.x

Apèndix C

Càrrega de dades sobre Sesame

En el següent apèndix es descriu el procés de càrrega de dades sobre el SGBD Sesame, d'aquesta manera s'intenta que el lector conegui el procediment i que no tingui que dedicar temps a l'apretentatge en futurs desenvolupaments.

Els passos que he seguit, són els següents:

- Primer ens hem d'identificar amb un usuari que tingui els privilegis suficients per a poder administrar la base de dades, en el meu cas, m'he creat un usuari *testuser* amb el password *testuser*.

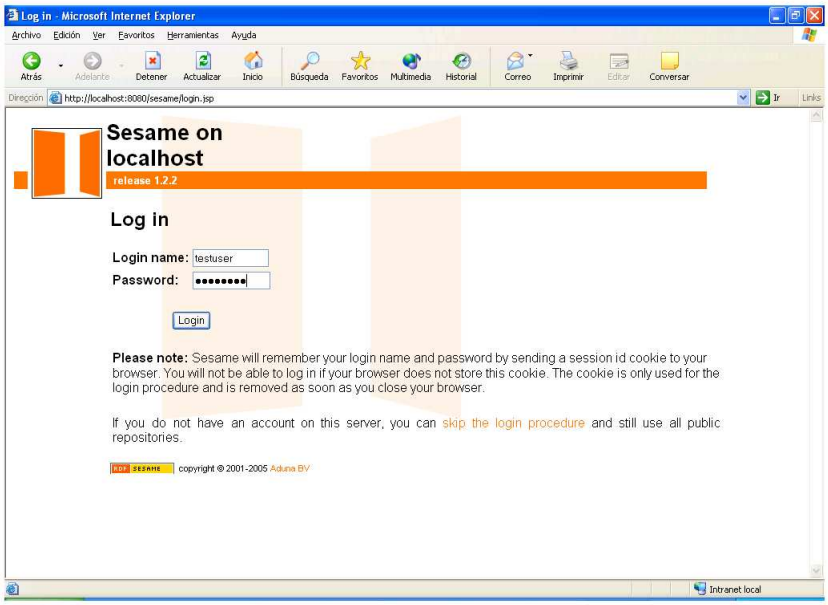

Figura C.1: Validació d'usuari i password amb Sesame.

- El següent pas seria la sel.lecció del repositori sobre el que anem a carregar les dades, en el meu cas escolliré la primera opció: "*Main Memory RDF Repository*".

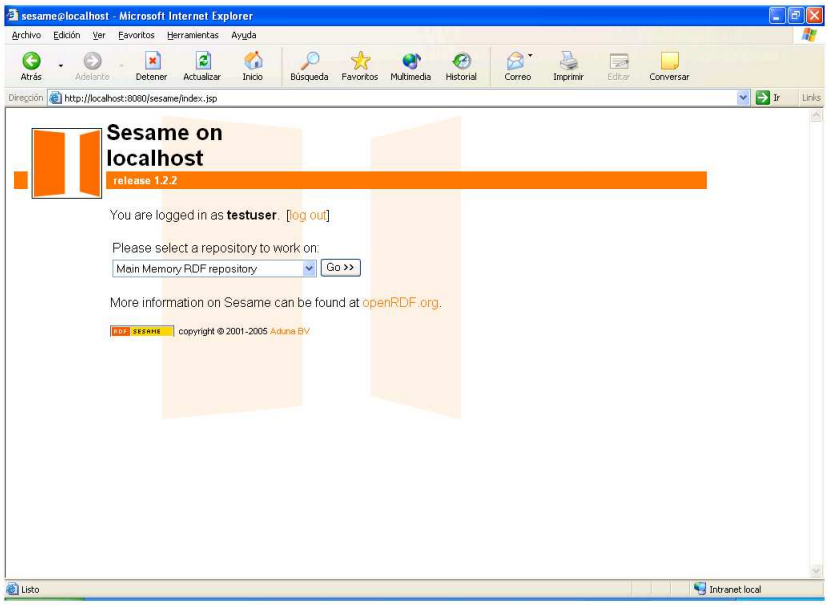

Figura C.2: Sel.lecció del repositori amb el que treballarem.

- Seguidament, se'ns mostra la pantalla principal amb les opcions que l'usuari pot fer segons el seu perfil. En el nostre cas, com es tractava d'un usuari administrador, les tenim totes disponibles.

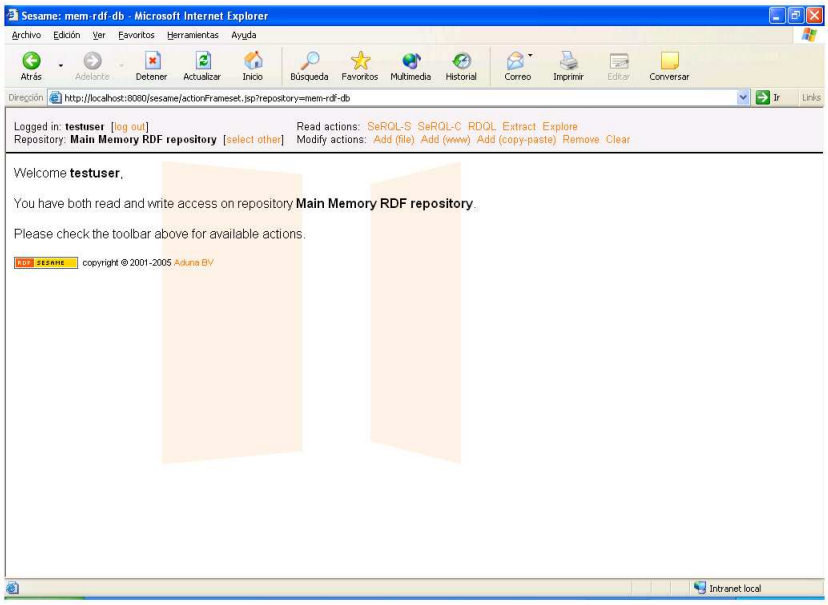

Figura C.3: Pantalla principal d'administració de Sesame.

- Finalment, escollim el fitxer que desitgem carregar i el Sesame s'encarrega de carregar-lo sobre el *MySQL* sense que ens calgui preocupar-nos per res més.

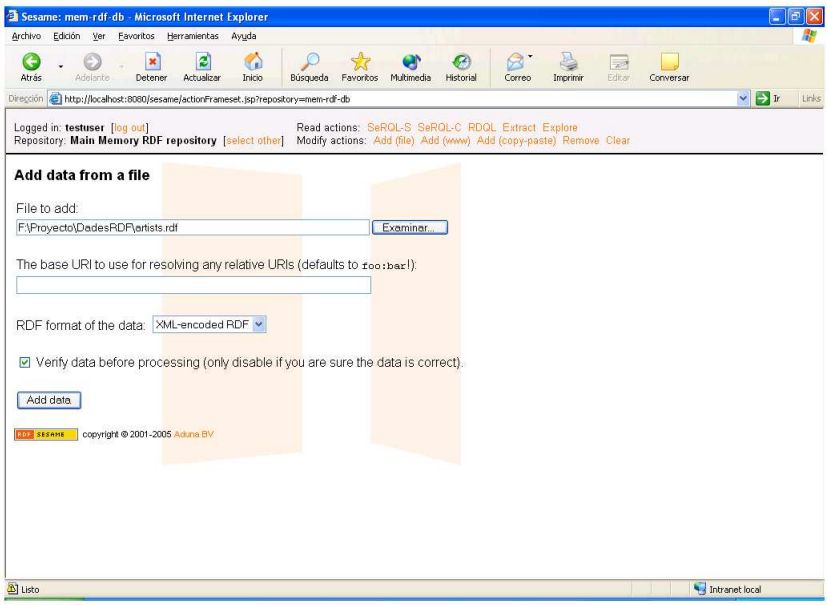

Figura C.4: Sel.lecció i càrrega d'un fitxer.

Apèndix D

La classe Phesame

Al estudiar la manera de poder implementar les consultes sobre el SGBD escollit, en el meu cas Sesame (veure 5.2.2), des de l'aplicació Web vaig plantejar-me utilitzar el Java o PHP. Finalment em vaig decantar per utilitzar el PHP 5, ja que al igual que Java té una gran difusió i a més és més senzill d'utilitzar.

Al igual que en PHP 4, el PHP 5 no té una integració directa amb Java, cosa que necessiten els mòduls de Java que conté el Sesame, anomenats *Sesame* i *RDFGrowth*. Per poder enllaçar el Sesame amb el PHP 5 he utilitzat les classes *Phesame*[1] i *PhesameExt*, que implementen una interfície PHP per poder accedir al API HTTP de Sesame.

El Phesame permet les següents operacions sobre l'API de Sesame:

- Login/Logout.
- Preguntar per la llista de repositoris disponibles.
- Avaluar una select en SeRQL, RQL o RDQL.
- Avaluar una consulta SeRQL.
- Extreure dades RDF d'un repositori.
- Desar dades en un repositori.
- Afegir dades des de la Web a un repositori.
- Esborrar un repositori.
- Esborrar sentències.

[1] La classe Phesame va ser escrita originàriament pel projecte *HyperJournal* per Michele Barbera i Riccardo Giomi. Quan van tenir noticia de que les consultes en XML sobre el Sesame eren molt costoses, van implementar un petit parser de consultes capaç de retornar el resultat en format d'array, i la van anomenar PhesameExt.

La classe Phesame utilitza la llibreria HTTP_Request de PEAR[2], la llibreria *PhesameExt* incorpora les funcions que ens són necessàries per a l'execució de consultes SeRQL (veure 4.2.2.2), retornant-les en format d'arrays PHP.

Inicialment ens fa falta connectar-nos al repositori de Sesame utilitzant un usuari i un password, això ho farem de la següent manera:

```
$phesame->login('usuari','password');
```

Podem també consultar els repositoris dels que disposem i també els seus mètodes d'aquest manera:

```
print_r($phesame->listWriteable());
```

Per a sel.leccionar un repositori específic ho farem de la següent manera:

```
$phesame->setSelectRepository('testRDFRepository');
```

Per afegir dades dintre d'un repositori, suposem que ja estem connectats al adequat, ho farem amb el mètode *uploadData*. Seguidament podem veure un exemple:

```
$rdf = "<rdf:RDF
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
        xmlns:foaf="http://xmlns.com/foaf/0.1#"
        <foaf:Person rdf:nodeID="me">
                <foaf:name>Michela Barbera</foaf:name>
                <foaf:title>Mr</foaf:title>
                <foaf:givenname>Michele</foaf:givenname>
                <foaf:family_name>Michele</foaf:family_name>
                <foaf:nick>barbz</foaf:nick>
        <foaf:mbox_shalsum>
                432432423423426564574765 7856c43
        </foaf:mbox_shalsum>
        </foaf:homepage rdf:resource="barbera.netseven.it"/>
        </foaf:Person>
</rdf:RDF>";
$phesame->uploadData($rdf, true, false, false);
```

Figura D.1: Com afegir dades a Sesame desde PHP 5.x

El mètode '*simpleQuery*' accepta una consulta en el llenguatge especificat mitjançant el mètode '*setQueryLanguage*' i retorna el resultat en el format especificat amb el métode '*setResultFormat*'.

Això ho podem veure en el següent exemple:

[2]És un conjunt de classes amb PHP reutilitzables.

```
$serql = "SELECT Autor, Paper
         FROM {Paper} rdf:type {foo:Paper};
                foo:keyword {"RDF","Querying"};
                dc:autor {Autor}
         USING NAMESPACE
         dc = <http://purl.org/dc/elements/1.0/>,
         foo = <http://www.foo.org/bar#>";
$result->$phesame->simpleQuery($serql);
```

Figura D.2: Com fer una consulta al Sesame des de PHP 5.x

El resultat de les consultes es pot tractar segons es desitgi.

Bibliografia

[1] Bhavani Thuraisingham. XML Databases and the Semantic Web. CRC Press, First edition, 2002.

[2] Michael C. Daconta, Leo J. Obrst, Kevin T. Smith. The Semantic Web, A Guide to the Future of XML, Web Services, and Knowledge Management. Wiley. First edition, 2003.

[3] XML Schema Part 0: Primer, W3C Recommendation, 2 May 2001.

[4] XML Schema Part 1: Structures, W3C Recommendation, 2 May 2001.

[5] XML Schema Part 2: Datatypes, W3C Recommendation, 2 May 2001.

[6] XML Path Language (XPath) Version 1.0, W3C Recommendation, 16 November 1999.

[7] XML Linking Language (XLink) Version 1.0, W3C Recommendation, 27 June 2001.

[8] XSL Transformations (XSLT) Version 1.0, W3C Recommendation, 16 November 1999.

[9] XML Path Language (XPath) Version 1.0, W3C Recommendation 16 November 1999. http://www.w3.org/TR/2001/REC-xlink-200010627?

[10] XML and RDF Databases, Research report 2003, VTT Information Technology.

[11] RDF Databases, Leo Sauermann, May 2003, Vienn.

[12] Ontology Query Languages for the Semantic Web, Department of Computer Science, University of Georgia.

[13] RDF Query Languages for the Semantic Web, Institute AIFB, University of Karlsruhe.

[14] XML/RDF Query by Example, Eric van der Vlist, Extreme Markup Languages 2005, Montreal.

[15] Asunción Gómez-Pérez, Mariano Fernández-López, Oscar Corcho. Ontological Engineering, with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web. Springer, Third edition, 2004.

[16] Web and Semantic Web Query Languajes: A Survey, James Bailey, François Bry, etc. 2005.

[17] Problemática y tendencias en la arquitectura de metadatos web. Pedro Manuel Díaz Ortuñó. Facultad de Ciencias de la Documentación. Universidad de Murcia.

[18] http://www.openrdf.org/

[19] Oracle Spatial (Resource Description Framework), 10g release 2 (10.2), July 2005.

[20] http://www.sesame.aiministrator.nl/

[21] http://www.hpl.hp.com/websem/rdql.htm/

[22] http://www.w3c.org/