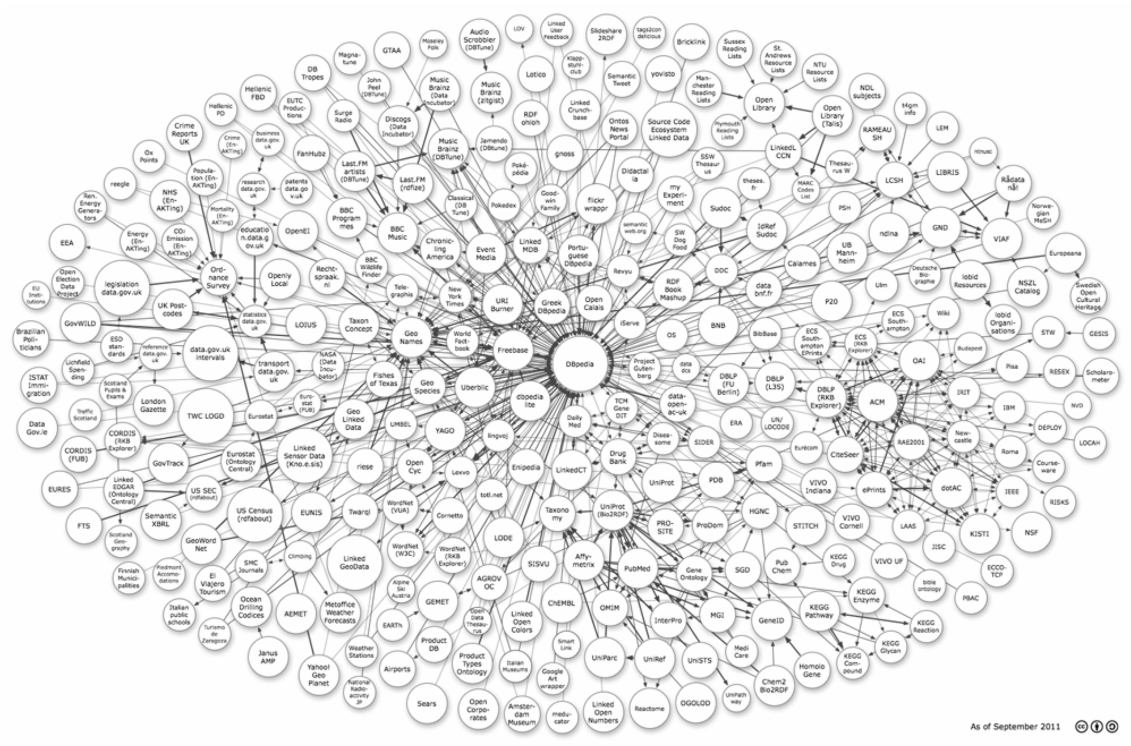
Magatzems de dades en el context de la web semàntica



José Sansano Martínez
Projecte de Fi de Carrera
Consultor: Felipe Geva Urbano
Curs 2011-2012 / 2on. semestre

Dedicat amb estima
a Emma, Beatriz i
Mª Angeles

Sinopsi

El present document és la memòria del projecte final de carrera d'Enginyeria Informàtica de la Universitat Oberta de Catalunya, corresponent a l'àrea d'XML i web semàntica.

L'objecte del projecte és estudiar amb profunditat els sistemes gestors de bases de dates en l'àmbit de la web semàntica. Aquests sistemes són gestors de bases de dades especialitzats en l'emmagatzemament i tractament de dades semàntiques, tot tenint en compte les peculiaritats d'aquestes. La qual cosa els fa diferents dels sistemes gestors relacionals.

Per tal d'aprofundir en la matèria referida el projecte es divideix en dues parts principals. En la primera es recerca l'estat de l'art, fent un estudi comparatiu entre els magatzems semàntics més coneguts en l'actualitat. En la segona part s'estudia en profunditat un d'aquests magatzems (Virtuoso Universal Server).

Índex

Índex de figures

Índex de taules

1. Introducció

A continuació es raona sobre la conveniència del projecte i es descriu l'entorn en què es troba la web semàntica avui en dia.

1.1. Justificació i context

La filosofia de la web semàntica és afegir coneixement a les dades. Actualment la web està dissenyada per ser llegida per les persones, amb aquesta nova tecnologia es pot fer que les dades de la web siguen llegibles per aplicacions informàtiques. Actualment s'estan fent avanços en aquest sentit i en un futur les màquines podran fer cerques i actuar segons la informació que hi haja en la web. D'aquesta forma una màquina podrà processar coneixement en comptes de text.

La base de la web semàntica són les ontologies. Aquestes representen informació d'un o més dominis que fan que la informació estiga estructurada semànticament. Si una web les usa per descriure la seua informació, les preguntes complexes fetes per humans poden ser enteses i resoltes per les màquines. Els elements informàtics capaços de fer-ho són anomenats agents intel·ligents.

L'HTML (Hypertext Markup Language) és insuficient per aquesta finalitat, ja que només és vàlid per presentar informació pels humans. L'HTML juntament amb un agent d'usuari, com ara un navegador, podem crear i presentar una pàgina amb una llista d'ítems junt a la seua descripció i preu, però no poden establir, de forma no ambigua, un lligam entre les tres dades, ni tan sols especificar que la pàgina és un catàleg, ni que cada dada és un nom de producte o un preu.

La web semàntica, aporta solucions que van més enllà, fent ús de llenguatges orientats a les dades, en comptes d'estar orientats als documents. Entre aquests components de la web semàntica podem destacar:

- XML (Extensible Markup Language): aporta una sintaxi.
- XML schema: serveix per restringir l'estructura dels documents XML.
- RDF (Resource Description Framework): model de dades per descriure els recursos i les seues relacions.
- RDF schema: vocabulari per descriure les propietats i classes dels recursos RDF.
- OWL (Web Ontology Language): llenguatge per definir ontologies mitjançant la descripció detallada de propietats i classes.
- SPARQL (SPARQL Protocol and RDF Query Language): llenguatge per consultar conjunts de dades RDF.

Tim Berners-Lee, director del W3C (World Wide Web Consortium), anomena *Global Giant Graf* a la xarxa resultant d'enllaçar dades amb aquestos llenguatges, en contraposició a l'actual "World Wide Web". Conseqüentment es passa de la compartició de documents a la compartició de dades. Tim Berners-Lee també indica que la web semàntica és un dels components de la futura Web 3.0.

La implantació de la web semàntica té dues dificultats que no ha pogut superar encara. Per una banda està la complexitat tecnològica que suposa convertir el contingut de les pàgines web en ontologies expressades, per exemple, en OWL. Afortunadament, cada vegada, més pàgines tenen la seua informació en bases de

dades, la qual cosa afavoreix la seua conversió automàtica. L'altra barrera a l'expansió de la web semàntica és el model de negoci de la web basat en la publicitat, el procés automàtic de la informació evitaria que les persones visitaren les pàgines i veiessin els anuncis. És lògic pensar que davant una implantació generalitzada de la web semàntica l'actual model de negoci s'hauria d'adaptar.

En altres PFC (Projecte Fi de Carrera) d'aquesta àrea s'estudien les ontologies, la web semàntica i els llenguatges esmentats, però en aquest PFC s'estudien els sistemes gestors de bases de dades que donen suport per l'emmagatzemament de dades semàntiques.

Els SGBD semàntics, també anomenats magatzems semàntics, són similars als SGBD relacionals ja que permeten emmagatzemar, consultar i gestionar dades estructurades. Però hi ha unes diferències principals:

- Fan ús d'ontologies com esquemes semàntics. Açò permet raonar automàticament sobre les dades.

- Treballen amb models de dades flexibles i genèrics (grafs,...). Açò permet interpretar i adoptar fàcilment noves ontologies o esquemes de metadades de forma dinàmica.

Durant els darrers anys, la web semàntica ha estat una àrea on els SGBD semàntics han esdevingut tant importants com els servidors HTTP. Aquesta tendència ha conduït a uns estàndards en metadades i ontologies, a través del W3C, especialment referits a RDF i OWL. Aquests estàndards estan tenint un paper similar al que va tenir l'SQL en la implantació dels SGBD relacionals. Tot i que aquests estàndards estan adreçats a la web semàntica estant tenint una amplia acceptació en la integració d'aplicacions d'empresa i biologia.

Entre els SGBD semàntics existents podem anomenar alguns com Virtuoso Universal Server, OWLIM, eXist, Sesame, Oracle Semantic Techonologies. En capítols posteriors aprofundirem en l'estudi d'alguns d'ells.

1.2. Objectius

L'àrea d'XML i web semàntica té com objectius generals assolir coneixements sobre:

- Conceptes bàsics de la web semàntica.
- Concepte d'ontologia.
- Llenguatges de representació d'ontologies (OWL, RDF,...).
- Eines d'edició i gestió d'ontologies.
- Usar ontologies per fer aplicacions riques semànticament.
- Realitzar casos pràctics.
- Llenguatges de consulta semàntica (SPARQL).

Els objectius concrets del PFC *Magatzems de dades en el context de la web semàntica* són:

- Fer un estudi rigorós dels SGBD més usats en el context de la web semàntica.
  - Identificar diferències amb els SGBD tradicionals.

  - Descriure la seua finalitat.
  - Explicar com s'utilitzen.
  - Llistar les limitacions i avantatges respecte els SGBD tradicionals.

- Analitzar a fons un SGBD en concret, explicant:
  - Les seues funcionalitats.
  - El sistema d'emmagatzemament de la informació.
  - Les operacions de consulta, alta i modificació d'informació semàntica.
  - El procediment d'importació d'ontologies.
  - Com es relacionen dades semàntiques.
  - Instal·lar-lo i fer proves de rendiment.

1.3. Enfocament i mètode seguit

Com que als estudis d'Enginyeria Informàtica a la UOC no es tracta la web semàntica, donada la seua novetat, al capítol 2 es farà una introducció a la matèria incloent-hi els conceptes bàsics, així com una introducció a les metodologies i llenguatges més utilitzats.

En un primer bloc del treball es farà una recerca dels SGBD més utilitzats en la web semàntica. D'aquests SGBD s'elaborarà un estudi, en el qual es compararan amb els SGDB relacionals per una banda. Per altra banda es farà un estudi comparatiu dels mateixos, per tal de tenir una base fonamentada amb la qual poder prendre decisions pel cas de tenir que triar-ne un.

En el segon bloc del treball es triarà un dels SGBD revisats anteriorment, per tal d'estudiar-lo amb profunditat. En aquest bloc es descriuran detingudament les seues funcionalitats, el seu sistema d'emmagatzemament de la informació, les operacions de consulta i modificació de la informació semàntica, com es fa la importació d'ontologies, com es relacionen les dades semàntiques i a més es faran proves de rendiment.

En cadascuna de les parts esmentades serà molt important establir els paràmetres a estudiar, tant en l'estudi comparatiu com en l'anàlisi en profunditat. Açò permetrà que la cerca d'informació i la redacció del projecte estiguen ben estructurats i conseqüentment que la memòria elaborada siga més fàcil d'entendre.

En el cas de les proves de rendiment s'haurà de preparar l'entorn de proves amb els corresponents sistema operatiu i base de dades triats. A més caldrà definir el joc de proves i el format dels resultats obtinguts per tal que siguen prou significatius a l'hora de valorar l'SGBD estudiat d'una forma global.

1.4. Planificació

En aquest apartat es defineixen totes les tasques que caldrà acomplir per realitzar el PFC. Així com la planificació temporal de les mateixes.

1.4.1. Tasques principals del PFC

Les tasques principals per portar a terme el PFC són:

- Tasca 1.- "Pla de treball del PFC". És la base per iniciar el PFC, estableix els objectius principals i fa una planificació del futur desenvolupament del projecte.

- Tasca 2.- "Redacció de la introducció del PFC". Es detallen els objectius, enfocament i mètode a seguir al PFC.

- Tasca 3.- "Conceptes bàsics sobre la web semàntica". S'elabora un compendi de conceptes que ens introdueixen en el mon de la web semàntica, com pas previ a l'estudi dels SGBD semàntics.

- Tasca 4.- "Estat actual dels SGBD a la web semàntica". Es farà un estudi detallat dels SGBD, d'aquest tipus, existents a l'actualitat. Descrivint els trets generals que els caracteritzen i diferencien respecte dels SGBD relacionals, així com els trets diferenciadors de cadascun respecte dels altres.

- Tasca 5.- "Anàlisi detallat d'un SGBD (Virtuoso/OWLIM)". Es procedirà a triar un dels SGBD estudiats per tal de fer un anàlisi en profunditat.

- Tasca 6.- "Finalització del PFC". Amb aquesta tasca es finalitzaran la memòria (conclusions, resum,...) i la presentació.

- Tasca 7.- "Debat virtual". S'haurà de respondre a les preguntes plantejades pel tribunal avaluador sorgides arran de la lectura del projecte.

1.4.2. Taula detallada de tasques

En aquest apartat es mostra una taula on es detallen les subtasques que composen cadascuna de les tasques principals, una descripció de cadascuna, les dependències entre elles, les dates previstes per la seua realització i la dedicació necessària a cadascuna.

Notes sobre la planificació temporal:

- S'ha definit un calendari laboral de 7 dies, ja que el més habitual és dedicar més temps els caps de setmana i festius.

- Per simplificar els càlculs de dedicació s'ha calculat una dedicació mitjana de dues hores per dia.

- S'han definit fites pels lliuraments oficials i també pels lliuraments previs per que el consultor els puga supervisar.

Tasca	Depèn de	Dates	Dedicació (dies)
1.- Pla de treball del PFC		**29/02 -> 12/03**	**13**
1.1.- Redacció del pla de treball (part general): inclou tot el pla excepte la planificació temporal.		29/02 -> 06/03	**7**
1.2.- Definició de tasques. Planificació temporal: llistat de tasques, diagrama Gantt.		07/02 -> 12/03	**6**
1.3.- Lliurament del pla de treball (PAC1).	1.1, 1.2	12/03	**Fita**
2.- Redacció de la introducció del PFC	**1**	**13/03 -> 16/03**	**4**
2.1.- Objectius generals del PFC: part introductòria del PFC.		13/03 -> 14/03	**2**
2.2.- Mètode de treball i enfocament del PFC: part introductòria del PFC.		15/03 -> 16/03	**2**
3.- Conceptes bàsics sobre la web semàntica	**2**	**17/03 -> 23/03**	**7**
3.1.- Recopilació d'informació sobre webs semàntiques: cercar informació sobre XML i web semàntica per tenir material introductori.		17/03 -> 19/03	**3**
3.2.- Estructuració de la informació i redacció: organitzar la informació obtinguda i redactar-la.	3.1	20/03 -> 22/03	**3**
3.3.- Presentació part I: elaborar la part de la presentació corresponent als conceptes bàsics documentats a la memòria.	3.2	23/03 -> 23/03	**1**
4.- Estat actual dels SGBD a la web semàntica	**3**	**24/03 -> 16/04**	**24**
4.1.- Recopilació d'informació sobre SGBD semàntics: fer una cerca intensiva sobre els SGBD semàntics més habituals.		24/03 -> 30/03	7
4.2.- Classificació de la informació: establir un sistema per classificar la informació obtinguda i facilitar les comparatives posteriors.	4.1	31/03 -> 01/04	2
4.3.- Comparativa amb SGBD relacionals: amb la informació anterior establir el trets comuns i diferenciadors amb els SGBD relacionals.	4.2	02/04 -> 09/04	8
4.4.- Comparativa entre els SGBD semàntics: amb la informació anterior realitzar una comparativa exhaustiva entres tots els SGBD estudiats.			
4.5.- Redacció dels capítols d'aquest apartat del PFC: finalitzar la redacció d'aquesta part.	4.3, 4.4	10/04 -> 14/04	5
4.6.- Lliurament provisional PAC2: lliurar el capítol al consultor per obtenir les seues indicacions i aplicar les correccions necessàries.		12/04	Fita

Tasca	Depèn de	Dates	Dedicació (dies)
4.7.- Presentació part II: elaborar la part de la presentació corresponent a l'estat actual dels SGBD a la web semàntica.		15/04 -> 16/04	2
4.8.- Lliurament PAC 2	4.5	16/04	Fita
5.- Anàlisi detallat d'un SGBD (Virtuoso/OWLIM)	**4**	**17/04 -> 28/05**	**42**
5.1.- Recopilació d'informació sobre l'SGBD triat: arran del capítol anterior triar un SGBD i cercar informació a fons sobre ell, referent a cadascun dels punts següents.		17/04 -> 27/04	11
5.2.- Redacció de funcionalitats	5.1	28/04 -> 30/04	3
5.3.- Redacció sistema d'emmagatzemament	5.1	01/05 -> 03/05	3
5.4.- Redacció operacions sobre informació semàntica	5.3	04/05 -> 06/05	3
5.5.- Redacció importació ontologies	5.1	07/05 -> 09/05	3
5.6.- Redacció relació de dades semàntiques	5.4	10/05 -> 12/05	3
5.7.- Realització de proves de rendiment: instal·lar el programari, definir el joc de proves i executar-les.	5.2, 5.5, 5.6	13/05 -> 20/05	8
5.8.- Documentació de les proves de rendiment: donar-li la forma adient a les dades obtingudes perquè siguen significatives i comprensibles.	5.7	21/05 -> 26/05	6
5.9.- Lliurament provisional PAC 3: lliurar el capítol al consultor per obtenir les seues indicacions i aplicar les correccions necessàries.		24/05	Fita
5.10.- Presentació part III: elaborar la part de la presentació corresponent a l'anàlisi detallat d'un SGBD.		27/05 -> 28/05	2
5.11.- Lliurament PAC 3	5.8	28/05	Fita
6.- Finalització del PFC	**5**	**29/05 -> 19/06**	**22**
6.1.- Finalitzar redacció d'apartats generals: acabar i revisar els capítols generals de la memòria.		29/05 -> 05/06	8
6.2.- Revisió general de la memòria: detectar errades, incoherències, repeticions, etc...	6.1	06/06 -> 08/06	3
6.3.- Finalitzar la presentació: revisar les parts elaborades i completar-les donant-li la forma final.		09/06 -> 19/06	11

Tasca	Depèn de	Dates	Dedicació (dies)
6.4.- Entrega final (provisional) : lliurar la memòria al consultor per obtenir les seues indicacions i aplicar les correccions necessàries.		12/06	Fita
6.5.- Entrega final	6.2, 6.3	19/06	Fita
7.- Debat virtual	**6**	**25/06 -> 28/06**	**4**
7.1.- Resoldre les qüestions del tribunal: el tribunal demanarà els aclariments que crega convenients i s'haurà de respondre abans de 24 hores.		25/06 -> 28/06	4
Duració total en dies			**116**
Duració total en hores (mitjana de 2 hores per dia)			**232**

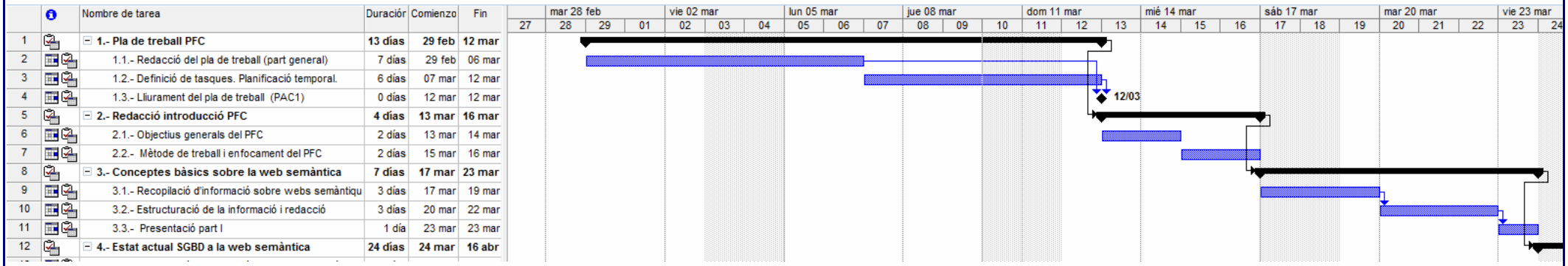
Taula 1. Fites del projecte fi de carrera.

Fita	Data
1.3.- Lliurament del pla de treball (PAC1)	12/03/2012
4.6.- Lliurament provisional PAC2	12/04/2012
4.8.- Lliurament PAC2	16/04/2012
5.9.- Lliurament provisional PAC3	24/05/2012
5.11.- Lliurament PAC3	28/05/2012
6.4.- Entrega final (provisional)	12/06/2012
6.5.- Entrega final	19/06/2012

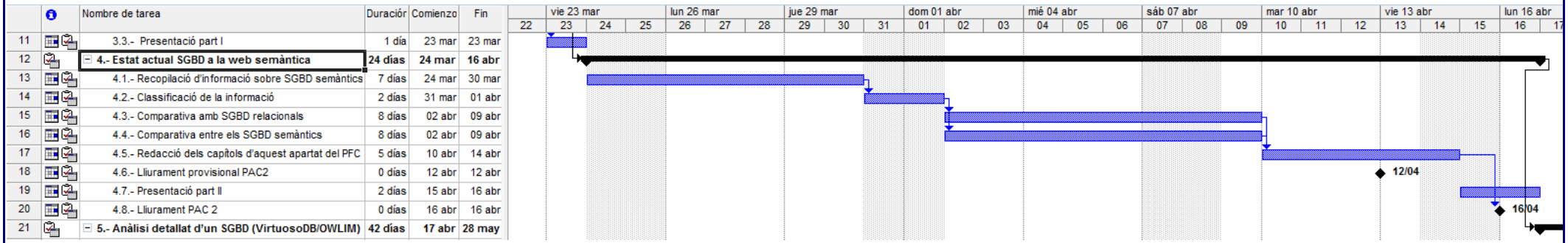
1.4.3. Diagrama de Gantt del PFC

A partir de l'anterior taula detallada de tasques obtenim el següent diagrama de Gantt.

Com es pot observar el diagrama ha estat imprès en quatre parts separades per que siga més llegible. Aquestes parts estan en ordre cronològic, com es veu a les dates de l'escala temporal.

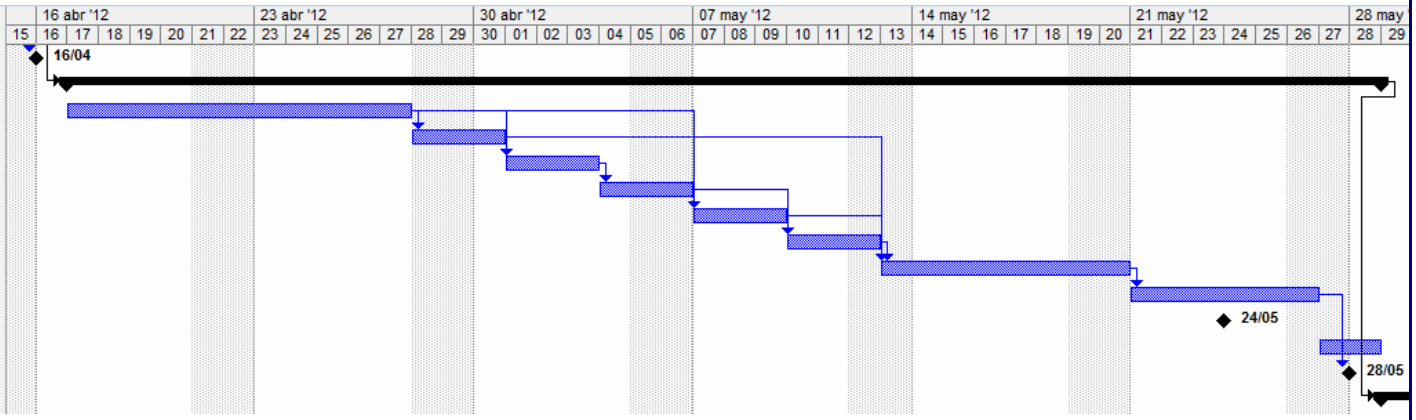


	Nombre de tarea	Duración	Comienzo	Fin
1	**1.- Pla de treball PFC**	**13 días**	**29 feb**	**12 mar**
2	1.1.- Redacció del pla de treball (part general)	7 días	29 feb	06 mar
3	1.2.- Definició de tasques. Planificació temporal.	6 días	07 mar	12 mar
4	1.3.- Lliurament del pla de treball (PAC1)	0 días	12 mar	12 mar
5	**2.- Redacció introducció PFC**	**4 días**	**13 mar**	**16 mar**
6	2.1.- Objectius generals del PFC	2 días	13 mar	14 mar
7	2.2.- Mètode de treball i enfocament del PFC	2 días	15 mar	16 mar
8	**3.- Conceptes bàsics sobre la web semàntica**	**7 días**	**17 mar**	**23 mar**
9	3.1.- Recopilació d'informació sobre webs semàntiqu	3 días	17 mar	19 mar
10	3.2.- Estructuració de la informació i redacció	3 días	20 mar	22 mar
11	3.3.- Presentació part I	1 día	23 mar	23 mar
12	**4.- Estat actual SGBD a la web semàntica**	**24 días**	**24 mar**	**16 abr**

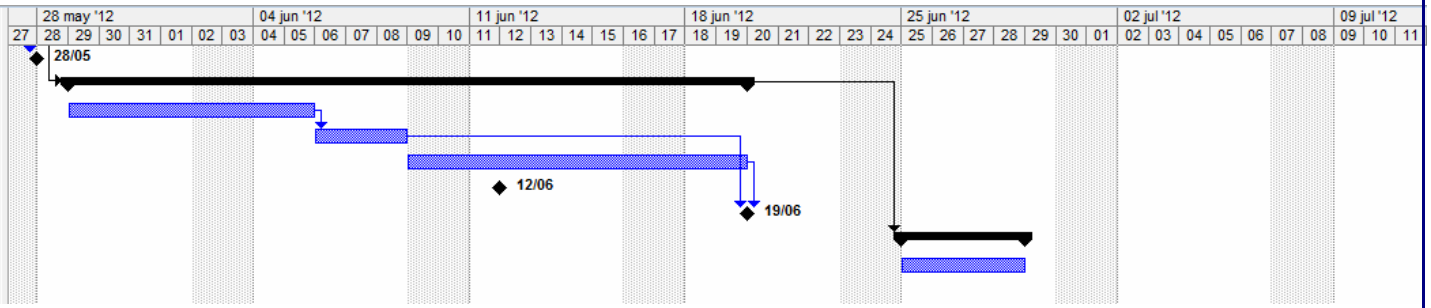


	Nombre de tarea	Duración	Comienzo	Fin
11	3.3.- Presentació part I	1 día	23 mar	23 mar
12	**4.- Estat actual SGBD a la web semàntica**	**24 días**	**24 mar**	**16 abr**
13	4.1.- Recopilació d'informació sobre SGBD semàntics	7 días	24 mar	30 mar
14	4.2.- Classificació de la informació	2 días	31 mar	01 abr
15	4.3.- Comparativa amb SGBD relacionals	8 días	02 abr	09 abr
16	4.4.- Comparativa entre els SGBD semàntics	8 días	02 abr	09 abr
17	4.5.- Redacció dels capítols d'aquest apartat del PFC	5 días	10 abr	14 abr
18	4.6.- Lliurament provisional PAC2	0 días	12 abr	12 abr
19	4.7.- Presentació part II	2 días	15 abr	16 abr
20	4.8.- Lliurament PAC 2	0 días	16 abr	16 abr
21	**5.- Anàlisi detallat d'un SGBD (VirtuosoDB/OWLIM)**	**42 días**	**17 abr**	**28 may**

	Nombre de tarea	Duración	Comienzo	Fin
20	4.8.- Lliurament PAC 2	0 días	16 abr	16 abr
21	**5.- Anàlisi detallat d'un SGBD (VirtuosoDB/OWLIM)**	**42 días**	**17 abr**	**28 may**
22	5.1.- Recopilació de informació sobre SGBD triat	11 días	17 abr	27 abr
23	5.2.- Redacció de funcionalitats	3 días	28 abr	30 abr
24	5.3.- Redacció sistema d'emmagatzemament	3 días	01 may	03 may
25	5.4.- Redacció operacions sobre informació semàntic	3 días	04 may	06 may
26	5.5.- Redacció importació ontologies	3 días	07 may	09 may
27	5.6.- Redacció relació de dades semàntiques	3 días	10 may	12 may
28	5.7.- Realització de proves de rendiment	8 días	13 may	20 may
29	5.8.- Documentació de les proves de rendiment	6 días	21 may	26 may
30	5.9.- Lliurament provisional PAC 3	0 días	24 may	24 may
31	5.10.- Presentació part III	2 días	27 may	28 may
32	5.11.- Lliurament PAC 3	0 días	28 may	28 may
33	**6.- Finalització PFC**	**22 días**	**29 may**	**19 jun**



	Nombre de tarea	Duración	Comienzo	Fin
32	5.11.- Lliurament PAC 3	0 días	28 may	28 may
33	**6.- Finalització PFC**	**22 días**	**29 may**	**19 jun**
34	6.1.- Finalitzar redacció de apartats generals	8 días	29 may	05 jun
35	6.2.- Revisió general de la memòria	3 días	06 jun	08 jun
36	6.3.- Finalitzar la presentació	11 días	09 jun	19 jun
37	6.4.- Entrega final (provisional)	0 días	12 jun	12 jun
38	6.5.- Entrega final	0 días	19 jun	19 jun
39	**7.- Debat virtual**	**4 días**	**25 jun**	**28 jun**
40	7.1.- Resoldre les qüestions del tribunal	4 días	25 jun	28 jun



1.4.4. Gestió de riscos

La finalitat d'aquest apartat és preveure aquelles circumstàncies, de qualsevol mena, que puguen afectar negativament al desenvolupament del PFC. Addicionalment s'esbossa un pla de contingència pel cas que realment es produeixen.

Cal tenir en compte que el riscos s'han de gestionar al llarg de tot el projecte per tal de preveure'n de nous que puguen sorgir, evitant que sorgeixen sense haver ideat cap pla de contingència.

Taula 2. Riscos del projecte.

Risc	Pla contingència
Endarreriment del treball respecte del pla establert comportant risc de no arribar a temps als lliuraments obligatoris.	El pla establert és prou realista i per tant caldrà comprovar periòdicament si l'estem complint correctament. Així es podran detectar petits endarreriments i corregir-los abans que es facen grans. Una causa de possible endarreriment són el dubtes sobre la matèria estudiada o sobre la pròpia elaboració del projecte. Aleshores cal fer ús de la consultoria per evitar arribar a un punt mort.
Avaria al lloc de treball.	Tot el treball que es vaja elaborant estarà replicat en dos ordinadors i en dues memòries USB externes (utilitzant la utilitat SyncToy de Microsoft.
Manca de temps per dedicació a altres assignatures.	Aquest risc no es pot donar ja que vaig planificar les matriculacions per tal de matricular-me en aquest semestre del projecte únicament.
Circumstàncies laborals o familiars adverses.	Cas d'ocórrer-ne qualsevol caldrà distribuir la carrega de treball dels dies afectats en els dies precedents si estava prevista o següents si és sobrevinguda.

1.5. Recursos

- Bibliografia i fonts d'informació recomanades al pla docent. Pot ser calga trobar altres fonts més específiques pels SGBD semàntics.

- Programari
  - Microsoft Word
  - Microsoft Project
  - Microsoft Powerpoint
  - Camtasia (creació presentació en vídeo)
  - Microsoft Visio
  - Calendari telèfon mòbil (seguiment planificació temporal)
  - SGBD semàntic triat per fer les proves (Virtuoso / OWLIM)
  - Sistema operatiu per l'SGBD (Virtuoso i OWLIM es poden instal·lar tant a Windows com a Linux)

- Maquinari
  - Punt de treball estàndard de la UOC.
  - Telèfon mòbil.

1.6. Productes

El present projecte no té com objectiu el desenvolupament d'un programari. Com a conseqüència la present memòria i la presentació virtual associada esdevenen parts clau del projecte i com a tal és important tenir cura que siguen especialment correctes en la seua forma i contingut.

1.7. Resta de capítols

En aquest apartat es fa un esbós dels capítols i apartats que tindrà la memòria del projecte. Tot tractant d'assolir tots els objectius descrits a l'anterior apartat.

- Capítol 2. Conceptes bàsics sobre la web semàntica

  Es fa un recorregut, per les principals tècniques i llenguatges que formen la base de la web semàntica i les ontologies, que servirà per entendre les característiques dels magatzems semàntics.

- Capítol 3. Estat de l'art dels SGBD a la web semàntica

  S'enumeren les principals funcions i prestacions dels magatzems semàntics i s'explica com les implementen una sèrie d'SGBD semàntics reals.

- Capítol 4. Anàlisi detallat d'un SGBD (Virtuoso/OWLIM)

Es fa un estudi profund d'un SGBD elegit d'entre els revisats en el capítol anterior. Estudiant les seues característiques en detall, també s'instal·larà i es faran proves de rendiment.

- Capítol 5. Conclusions

Amb tot el que s'ha exposat als capítols anteriors s'extreuen les conclusions que permeten saber l'estat actual dels magatzem semàntics i quina pot ser la seua evolució en el futur.

2. Conceptes bàsics sobre la web semàntica

En el present capítol s'exposen alguns dels conceptes, llenguatges i estàndards que són la base teòrica de la web semàntica. La web semàntica i el concepte *linked data (dades enllaçades)* són parts fonamentals de la Web 3.0, en aquest entorn es poden enllaçar dades dins la web, entre sistemes d'arreu el món, amb relacions autodescriptives.

La finalitat de la web semàntica és que la informació, que ara està disponible a través de documents HTML, passe a estar disponible com si fos una base de dades. Per aconseguir-ho s'haurà de descriure la informació amb models de dades que permeten que siga tractada i cercada de forma automàtica. Els beneficis que proporcionaria aquesta cerca automàtica, de les dades de la web, respecte a les ferramentes i programes actuals, és immensa.

La filosofia *linked data* permet enllaçar dades estructurades d'arreu la web de forma transparent. Les ferramentes tecnològiques que usa són els URI (Uniform Resource Identifier, per identificar entitats o conceptes a tota la web), HTTP (per recuperar recursos) i RDF (model de dades basat en grafs).

2.1. XML

L'XML és una de les formes de representació de dades semiestructurades més usades a la web i en general. S'usa, per exemple, pel marcat de documents (XHMTL), llibreries de música (iTunes), scripts per compilacions (Ant) i fulles d'estil (XSLT). XML també s'usa per la serialització de formats de definició de dades semàntiques com RDF i TopicMaps.

XML es defineix com un llenguatge de marques genèric que serveix per descriure estructures de dades. L'aportació principal d'XML és una sintaxi per representar dades de forma jeràrquica. Els ítems de dades s'anomenen *elements* i s'encerclen amb *tags* d'inici i fi, amb el mateix nom o *label* (per exemple <*author*>*...*</*author*>). En la zona de punts suspensius es poden escriure altres *elements* o dades, anomenades *children* (fills de l'element inicial). Veiem un exemple senzill:

```
<bib xmlns:dc="http://purl.org/dc/elements/1.1/">
<article journal="Computer Journal" id="12">
<dc:title>...Semantic Web...</dc:title>
<year>2005</year>
<authors>
<author>
<first>John</first> <last>Doe</last> </author>
<author>
<first>Mary</first> <last>Smith</last> </author>
</authors>
</article>
<article journal="Web Journal">
<dc:title>...Web...</dc:title>
<year>2003</year>
<authors>
<author>
<first>Peter</first> <last>Jones</last> </author>
```

```
<author>
<first>Sue</first> <last>Robinson</last> </author>
</authors>
</article>
</bib>
```

Altres components de l'XML són:

- *Attributes*: són noms i parells de valors associats amb *tags* d'inici. L'ordre dels *attributes* no és significatiu però sí ho és el dels *elements*.

  `<article journal="Computer Journal" id="12">`

- *Namespace*: són *elements* anomenats, formats per tres parts: *local name, namespace prefix i namespace URI*.

  `dc:title` (`title` és el *local name* i `dc` és el *namespace prefix)*

  El *namespace URI* és `http://purl.org/dc/elements/1.1` que es pot deduir del *namespace declaration* pel prefix `dc` a la primer línia `xmlns:dc ="http://....`

- Les *base URI*: s'usen per resoldre els URI relatius d'un document XML. Com en el cas anterior on `dc:title` hereta la *base URI* de `xmlns:dc="http://purl.org/dc/elements/1.1/"`, donant com a resultat de substituir `dc:` el següent:
  `http://purl.org/dc/elements/1.1/title`

- Els *comments, processing instructions* (parells nom i valor), *document level information, entities* i *notations* (són altres tipus de contenidors d'informació).

2.2. DTD i XML Schema

Donat que l'XML no té un vocabulari predeterminat, és correcte inventar-se un cert llenguatge XML segons el nostre gust o necessitat, serà correcte o bé format sempre que els *tags* estiguen escrits correctament.

Òbviament pot caldre un vocabulari XML específic per exemple per matemàtiques o per fórmules químiques. El que cal és un mecanisme per definir formalment un llenguatge XML concret que assegure que està ben format i que usa correctament el vocabulari del llenguatge definit.

Hi ha diverses formes d'especificar formalment un llenguatge de marques. Les dues més comuns són els DTD (Document Type Definitions) i els esquemes XML.

Els **DTD** es basen en XML 1.0. Normalment són documents separats als que un XML pot referir-se o bé pot formar part d'un XML. Els DTD són un conjunt de regles o declaracions que descriuen els elements i altres objectes de l'XML. Cada declaració afegeix un nou tipus d'element al vocabulari i dona el seu model de contingut

(elements que pot contenir i amb quin ordre). Un tipus d'element no declarat es considera il·legal, així com un element que continga quelcom no declarat.

Els DTD no restringeixen els tipus de dades que poden anar dins dels elements i aquest és un dels punt febles d'aquest model.

Exemple de DTD i XML que l'acompleix:

```
<!DOCTYPE etiqueta[
       <!ELEMENT etiqueta (nom, carrer, ciutat, pais, codi)>
       <!ELEMENT nom (#PCDATA)>
       <!ELEMENT carrer (#PCDATA)>
       <!ELEMENT ciutat (#PCDATA)>
       <!ELEMENT pais (#PCDATA)>
       <!ELEMENT codi (#PCDATA)>
       ]>

       <etiqueta>
       <nom>Pepe García</nom>
       <carrer>C/Ronda, 3</carrer>
       <ciutat>Armilla</ciutat>
       <pais>Espanya</pais>
       <codi>18465</codi>
       </etiqueta>
```

L'***XML schema*** és posterior als DTD, usa la sintaxi XML i com a conseqüència és més flexible que els DTD, pot especificar tipus de dades, utilitzar espais de noms i és extensible. Es podria declarar un element cridat `data` i exigir que el seu contingut fos una data amb un format concret (AAAA-MM-DD).

Els elements principals són:

- *attribute*, declara un atribut d'un element determinat.
- *element*, declara que un element pot estar contingut per un altre.

En els *XML schema* cal definir primer els elements més profunds del document XML, és a dir s'ha de treballar de dins a fora. Veiem un exemple d'esquema XML i XML que l'acompleix (observeu com l'XML referencia l'esquema *vehicles.xsd* :

Esquema XML (vehicles.xsd):

```
<?xml version = "1.0" encoding = "UTF-8"?>
<xsd:schema xmlns:xsd = "http://www.w3.org/2001/XMLSchema">
    <xsd:element name = "vehicles">
       <xsd:complexType>
         <xsd:sequence>
            <xsd:element name = "nom"
                   type = "xsd:string"
                   maxOccurs = "unbounded"/>
         </xsd:sequence>
       </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

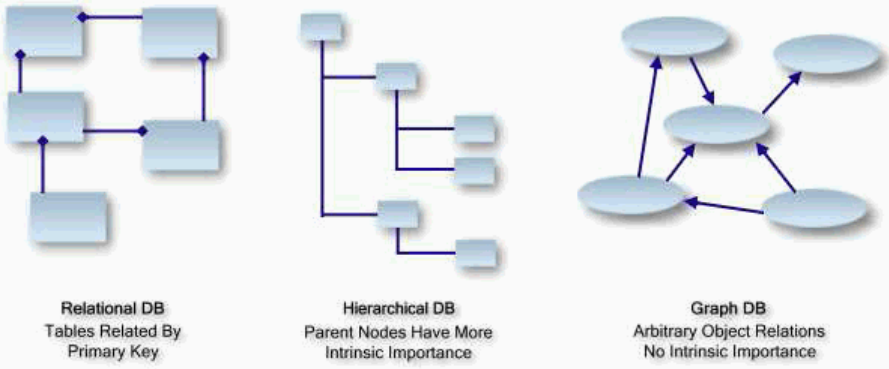
XML:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<vehicles
    xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation = "vehicles.xsd"   >
        <nombre>cotxe</nombre>
        <nombre>moto</nombre>
        <nombre>carro</nombre>
</vehicles>
```

2.3. Bases de dades orientades a grafs

En la majoria dels sistemes d'emmagatzemament hi ha conceptes com nodes o taules de dades que normalment tenen més importància que els altres elements del sistema. En el cas de l'XML hi ha una jerarquia que culmina en el node del nivell superior, l'arrel. En el cas de les dades en graf no hi ha cap arrel o jerarquia. Simplement són recursos relacionats amb altres recursos, on cap d'ells té rellevància respecte dels altres.

Figura 1. Esquema comparatiu SGBD relacional, jeràrquic i en graf.

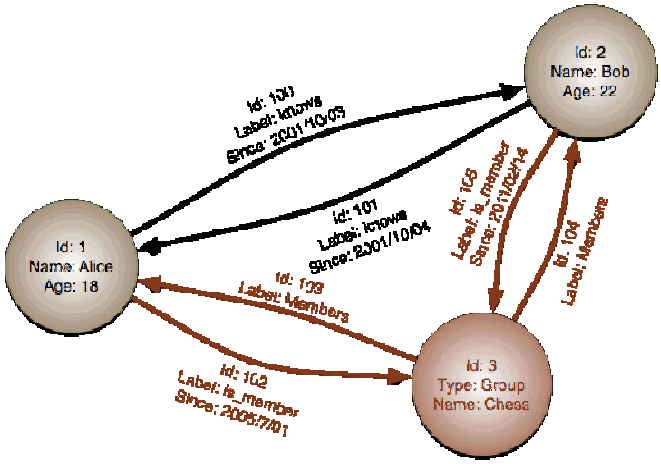


(Font: http://www.linkeddatatools.com/introducing-rdf, 2012)

Característiques i avantatges de les BDOG (Bases de dades orientades a grafs):

- Combinació d'atributs multivalor i complexes.
- Flexible als canvis d'estructura. Per exemple si la font de dades és la web.
- Unifica la representació de dades, esquemes i consultes.
- Poden rebre o retornar grafs complets segons el criteris de cerca.
- Consultes amplies no restringides a taules. Es poden demanar totes les taules amb un nom Carles.
- No cal definir un nombre determinat d'atributs. Un persona pot tenir 3 noms i altra 2.
- Es poden recórrer de forma jeràrquica.
- Isomorfisme, quan s'introdueix un graf en una BDOG s'ha de poder detectar si ja hi existeix o es pot obtenir permutant altre graf. Això permet estalviar espai d'emmagatzemament.

Figura 2. Esquema de diversos triples emmagatzemats en graf.



(Font: http://en.wikipedia.org/wiki/Graph_database, 2012)

2.4. RDF

RDF permet descriure, de forma bàsica, un model de dades orientat a grafs, usat per la web semàntica, amb grafs dirigits i etiquetats. Els arcs dels grafs (predicats) uneixen el subjecte amb l'objecte. RDF defineix declaracions usant allò anomenat *triple,* format per la relació dels tres conceptes anomenats (subjecte, predicat i objecte).

- Subjecte: és un recurs identificat per un URI RDF o un node buit i denota allò que es vol descriure.
- Predicat: està identificat per un URI RDF, i expressa la característica del subjecte, que es vol especificar.
- Objecte: s'identifica per un URI RDF, un node buit o un literal, serveix per donar un valor a una propietat (predicat) d'un subjecte.

Allò que fa que RDF siga el format de dades per a la web semàntica, és que les dades expressades en RDF contenen significat. Aquest significat permet inferir coneixement addicional més enllà del que es declara explícitament. Els llenguatges de consulta han de ser consistents amb la vinculació RDF (simple, completa, vinculació RDFS), és a dir, aquells grafs equivalents, dins la seua respectiva vinculació, han de subministrar la mateixa resposta.

La informació RDF es pot presentar en diferents formats. Un dels més habituals és la sintaxi RDF /XML. Un altre format molt més simple és Turtle, usat per SPARQL per manipular dades RDF.

Exemple de *triple*:

subjecte →	***predicat →***	***objecte***
(t-shirt)	(color)	(white)

El següent RDF/XML declara el *triple* anterior:

```
<?xml version="1.0" encoding="UTF8"?>
<rdf:RDF
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:feature="http://www.linkeddatatools.com/clothing-features#">
<rdf:Description rdf:about="http://www.linkeddatatools.com/clothes#t-shirt">
   <feature:color rdf:resource="http://www.linkeddatatools.com/colors#white"/>
</rdf:Description>
</rdf:RDF>
```

Veiem a continuació com es construeix un document RDF:

- Primerament cal afegir l'arrel de RDF de W3, definit pel *namespace* indicat en la línia 02, el qual indica que el document és una declaració RDF.
- A continuació s'afegeix una declaració (podria haver-ne més) començant per l'etiqueta `rdf:Description` , indicant que descriu un subjecte i donant-li l'identificador únic de la línia 05, definit per l'etiqueta `rdf:about`.
- Finalment s'afegeixen un parell de predicats o propietats, definides per les propietats `feature:size` i `feature:color`, a les línies 07 i 08. Aquestes propietats pertanyen al segon *namespace* (línia 03).

```
01 <rdf:RDF
02 xmlns:rdf="http://www.w3.org/1999/02/22rdfsyntaxns#"
03 xmlns:feature="http://www.linkeddatatools.com/clothingfeatures#">
04
05 <rdf:Description rdf:about="http://www.linkeddatatools.com/clothes#tshirt">
06
07 <feature:size>12</feature:size>
08 <feature:color rdf:resource="http://www.linkeddatatools.com/colors#white"/>
09
10 </rdf:Description>
11
12 </rdf:RDF>
```

2.5. RDF Schema

RDF Schema (RDFS) és un llenguatge per descriure el vocabulari RDF, de forma anàloga als DTD respecte a XML. Això és necessari per que RDF no té mecanismes per descriure les propietats ni les relacions entre aquestes i altres recursos. Aquest últim és el rol de RDFS mitjançant classes i propietats usades per descriure classes, propietats i altres recursos.

RDFS és una extensió semàntica de RDF, aquesta extensió s'escriu en RDF usant els termes descrits pel W3C, permetent descriure grups de recursos relacionats i les relacions entre ells. RDFS permet expressar taxonomies senzilles i jerarquies entre declaracions. També es poden descriure característiques dels recursos com són el seu domini o rang de valors permesos.

El W3C defineix un conjunt de classes, propietats i altre vocabulari per RDFS, que es resumeix a continuació:

- Classes.
  - `rdfs:Resource`: classe de recursos, és a dir tot.
  - `rdfs:Class`: classe de classes.
  - `rdfs:Literal`: valors literals de text o enters.
  - `rdfs:Datatype`: classe de tipus de dades RDF.
  - `rdf:XMLLiteral`: classe de valors XML literals.
  - `rdf:Property`: classe de propietats RDF.

- Propietats
  - `rdfs:range`: defineix un rang pel subject.
  - `rdfs:domain`: defineix un domini pel subjecte.
  - `rdf:type`: defineix el subjecte com una instància d'una classe.
  - `rdfs:subClassOf`: el subjecte és una subclasse d'altra classe.
  - `rdfs:subPropertyOf`: el subjecte és una subpropietat d'una altra.
  - `rdfs:label`: nom descriptiu d'un subjecte.
  - `rdfs:comment`: descripció d'un subjecte.
- Altre vocabulari.
  - Classes contenidor i propietats.
    - `rdfs:Container`, `rdf:Bag`, `rdf:Seq`, `rdf:Alt`, `rdfs:member`.
  - Coleccions RDF.
    - `rdf:List`, `rdf:first`, `rdf:rest`, `rdf:nil`.
  - Vocabulari de reificació.
    - `rdf:Statement`, `rdf:predicate`, `rdf:object`, `rdfs:seeAlso`, `rdfs:isDefinedBy`, `rdf:value`.

Com a exemple a continuació tenim un graf que representa una jerarquia de classes de vehícles.

Figura 3. Jerarquia de vehicles representada en un graf.

(Font: http://www.w3.org/TR/rdf-primer/#figure18, 2012)

A continuació es mostra el mateix esquema expressat en RDF/XML.

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">]>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xml:base="http://example.org/schemas/vehicles">

<rdf:Description rdf:ID="MotorVehicle">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
</rdf:Description>
```

```
<rdf:Description rdf:ID="PassengerVehicle">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
  <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdf:Description>

<rdf:Description rdf:ID="Truck">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
  <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdf:Description>

<rdf:Description rdf:ID="Van">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
  <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdf:Description>

<rdf:Description rdf:ID="MiniVan">
  <rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
  <rdfs:subClassOf rdf:resource="#Van"/>
  <rdfs:subClassOf rdf:resource="#PassengerVehicle"/>
</rdf:Description>

</rdf:RDF>
```

2.6. OWL

OWL (Web Ontology Language) s'ha dissenyat per ser usat per aplicacions que necessiten processar el contingut de la informació, en comptes de només mostrar-lo a les persones. OWL ofereix una millor interpretabilitat del contingut web per a les màquines, que l'ofert per XML, RDF i RDFS, ja que subministra vocabulari addicional així com una semàntica formal. OWL té tres versions amb una expressivitat creixent: OWL Lite, OWL DL i OWL Full.

OWL pot ser usat per expressar ontologies, que consisteixen en representar el significat de termes, en certs vocabularis, i les relacions entre aquests termes. OWL té més prestacions per expressar significats i semàntiques que XML, RDF i RDFS.

L'OWL sorgeix per la necessitat d'afegir, a sobre de RDF, un llenguatge d'ontologies per descriure formalment el significat de la terminologia usada als documents web. Amb RDF Schema únicament, no és possible que les màquines facen tasques de raonament útils sobre aquests documents, en canvi l'ús d'ontologies sí ho permet.

OWL afegeix més vocabulari per descriure propietats i classes, com ara, relacions entre classes (disjunció), cardinalitat, igualtat, tipus de propietats enriquits, característiques de les propietats (simetria) i classes enumerades.

A continuació veiem una breu descripció dels tres subllenguatges OWL.

- *OWL Lite* serveix per a usuaris que requereixen una classificació jeràrquica i restriccions senzilles. La seua complexitat formal és inferior que la de les versions superiors.

- *OWL DL* dona la màxima expressivitat alhora que manté la completesa computacional (es garanteix que totes les conclusions són computables) i la decidibilitat (totes les computacions acabaran en un temps finit). OWL DL permet totes les construccions OWL però amb restriccions.

- *OWL Full* està orientat a aquells que volen la màxima expressivitat i llibertat de sintaxi RDF, això sí, sense garanties computacionals. OWL permet que una ontologia augmente el vocabulari predefinit (RDF o OWL). No és probable que un programari de raonament done suport pel raonament complet d'OWL Full.

Cada subllentguatge és una extensió del seu predecessor més senzill. OWL Full es pot veure com una extensió de RDF, mentre que OWL Lite i OWL DL són l'extensió d'una vista restringida de RDF. Conseqüentment tot document RDF és un document OWL Full però pot no ser OWL Lite o OWL DL *(www.w3.org[1], 2012).*

A l'annex A es mostra un llistat dels termes dels llenguatges OWL.

Per tal d'il·lustrar l'ús dels termes exposats a l'annex, es presenta a continuació l'estructura esquemàtica d'una ontologia, els elements bàsics d'OWL, amb una breu explicació de cadascun, i uns breus exemples extrets de dues ontologies, sobre vi i menjar, explicades a la guia d'OWL del W3C.

- **Espais de noms (namespaces)**

  Com es veu a continuació, les declaracions d'espais de noms es posen dins una etiqueta `rdf:RDF`. Aquesta secció serveix per definir quins vocabularis es van a usar. Com és obvi s'usen els d'XML Schema, RDF, RDFS i OWL. Addicionalment es defineix a la primera línia, quin serà l'espai de noms predeterminat que s'aplicarà als noms sense prefix. La segona i tercera línies identifiquen respectivament el prefix de l'ontologia i la base URI per la resta del document. La quarta línia identifica una ontologia auxiliar.

```
<rdf:RDF
xmlns     ="http://www.w3.org/TR/2004/REC-owl-guide-20040210/wine#"
xmlns:vin ="http://www.w3.org/TR/2004/REC-owl-guide-20040210/wine#"
xml:base  ="http://www.w3.org/TR/2004/REC-owl-guide-20040210/wine#"
xmlns:food="http://www.w3.org/TR/2004/REC-owl-guide-20040210/food#"
xmlns:owl ="http://www.w3.org/2002/07/owl#"
xmlns:rdf ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:xsd ="http://www.w3.org/2001/XMLSchema#">
...
```

Com en OWL es fan referències freqüents als identificadors d'ontologies, pot ser-nos útil un mètode abreviat per definir una ENTITY:

```
<!DOCTYPE rdf:RDF [
<!ENTITY  vin  "http://www.w3.org/TR/2004/REC-owl-guide-20040210/wine#"
>]
```

Aleshores podrem usar `&vin;merlot` per referir-nos a `http://www.w3.org/TR/2004/REC-owl-guide-20040210/wine#merlot`.

- **Capçaleres de l'ontologia**

  Després l'usual és incloure un seguit d'assercions sobre l'ontologia, agrupades sota l'etiqueta `owl:Ontology`. Açò dona suport per tasques com comentaris, control de versions i inclusió d'altres ontologies. És on es recopilen moltes de les metadades OWL.

```
<owl:Ontology rdf:about="">
```

```
<rdfs:comment>An example OWL ontology</rdfs:comment>
<owl:priorVersion  rdf:resource="http://www.w3.org/TR/2003/PR-owl-guide-
20031215/wine"/>

<owl:imports      rdf:resource="http://www.w3.org/TR/2004/REC-owl-guide-
20040210/food"/>

<rdfs:label>Wine Ontology</rdfs:label>
...
</owl:Ontology>
```

`rdf:about` dona el nom o referència per l'ontologia.

`rdfs:comment` serveix per posar anotacions a l'ontologia.

`owl:priorversion` és útil pels sistemes de control de versions.

`owl:imports` és un mecanisme d'inclusió d'altres ontologies, té un sol paràmete, identificat per l'atribut `rdf:resource`.

- **Agregació de dades i privacitat**

  OWL pot expressar equivalència entre dues dades de diferents orígens amb `owl:sameAs`, indicant que són la mateixa cosa. `Owl:InverseFunctionalProperty` també pot servir per enllaçar elements individuals, per exemple si tenen una propietat d'aquest tipus amb el mateix valor, com pot ser "IdCardNumber". Aquest mena d'agregacions pot ser usada per inferir fets que no estan directament representats en cap lloc.

- Aquesta capacitat de la web semàntica per enllaçar informació de múltiples orígens és una característica desitjable i potent. Però açò combinat amb la capacitat d'inferir, d'OWL, pot donar lloc a un ús abusiu *(www.w3.org[2], 2012)*.

.

- **Classes senzilles i element individuals**

  Per tal de poder raonar sobre els elements individuals cal definir les classes a les que aquests pertanyeran. La majoria dels conceptes bàsics d'un domini correspon a les classes que són les arrels de diversos arbres taxonòmics. Tota classe definida serà una subclasse d'`owl:Thing`.

  Per l'exemple dels vins és creen tres classes arrel:

```
<owl:Class rdf:ID="Winery"/>
<owl:Class rdf:ID="Region"/>
<owl:Class rdf:ID="ConsumableThing"/>
```

  Ara ja poden referenciar aquestes classes amb la seua etiqueta XML `vin:Winery`, el valor de l'atribut `&vin;Winery` o el seu URI complet: `http://www.w3.org/TR/2004/REC-owl-guide-20040210/wine#Winery`.

  El constructor taxonòmic de classes fonamental és `rdfs:subClassOf`, que relaciona una classe més específica amb una més general, per exemple:

```
<owl:Class rdf:ID="PotableLiquid">
<rdfs:subClassOf rdf:resource="#ConsumableThing" />
...
</owl:Class>
```

Els elements individuals es declaren com a membres d'una classe:

```
<PotableLiquid rdf:ID="Wine" />
```

- **Propietats**

Les propietats permeten definir característiques generals sobre els membres de les classes i característiques específiques sobre els elements individuals. Hi ha propietats sobre objectes (*ObjectProperties*) i sobre tipus de dades (*DatatypeProperties*). Una propietat és una relació binaria que pot ser restringida pel domini i el rang.

La següent propietat `madeFromGrape` serveix per relacionar un vi amb el tipus de raïm amb que es fa:

```
<owl:ObjectProperty rdf:ID="madeFromGrape">
<rdfs:domain rdf:resource="#Wine"/>
<rdfs:range rdf:resource="#WineGrape"/>
</owl:ObjectProperty>
```

En aquest cas la propietat `yearValue` relaciona la classe `VintageYear` amb els enter positius:

```
<owl:Class rdf:ID="VintageYear" />

<owl:DatatypeProperty rdf:ID="yearValue">
<rdfs:domain rdf:resource="#VintageYear" />
<rdfs:range rdf:resource="&xsd;positiveInteger"/>
</owl:DatatypeProperty>
```

- **Característiques de les propietats**

Les propietats d'OWL poden ser etiquetades amb les següents característiques per millorar el raonament sobre elles.

Si una propietat *P* és,

- Etiquetada com transitiva aleshores *P(x,y) i P(x,z)* implica *P(x,z)*
- Etiquetada com simètrica aleshores *P(x,y)* implica *P(y,x)*
- Etiquetada com funcional aleshores *P(x,y) i P(x,z)* implica *y=z*
- Etiquetada com "*inverseOf*" de P' aleshores *P(x,y)* implica *P'(y,x)*
- Etiquetada com "*InverseFunctionalProperty*" aleshores *P(y,x)* i *P(z,x)* implica *y=z*

Exemple de propietat transitiva (`locatedIn`)

```
<owl:ObjectProperty rdf:ID="locatedIn">
      <rdf:type rdf:resource="&owl;TransitiveProperty" />
      <rdfs:domain rdf:resource="&owl;Thing" />
      <rdfs:range rdf:resource="#Region" />
</owl:ObjectProperty>

<Region rdf:ID="SantaCruzMountainsRegion">
      <locatedIn rdf:resource="#CaliforniaRegion" />
```

```
</Region>

<Region rdf:ID="CaliforniaRegion">
      <locatedIn rdf:resource="#USRegion" />
</Region>
```

Com `SantaCruzMountainsRegion` està `locatedIn` la `CaliforniaRegion` aleshores també ha d'estar `locatedIn` la `USRegion` degut a que `locatedIn` , com es veu, ha estat etiquetada com transitiva.

- **Restriccions de les propietats**

A més de amb les característiques, podem restringir més el rang d'una propietat de varies formes. Es fa en el context d'una `owl:Restriction`, on `owl:onProperty` indica la propietat objecte de la restricció.

- *allValuesFrom, someValuesFrom*

`owl:allValuesFrom` requereix que la propietat restringida tinga tots els seus valors dins la classe indicada. És una restricció local a la classe contenidora.

`Owl:someValuesFrom` requereix que la propietat restringida tinga al menys un valor dins la classe indicada. És una restricció local a la classe contenidora.

En el següent exemple la propietat `hasMaker`, de la classe `Wine`, ha de tenir tots els seus valors dins la classe `Winery`.

```
<owl:Class rdf:ID="Wine">
<rdfs:subClassOf rdf:resource="&food;PotableLiquid" />
...
<rdfs:subClassOf>
<owl:Restriction>
<owl:onProperty rdf:resource="#hasMaker" />
<owl:allValuesFrom rdf:resource="#Winery" />
</owl:Restriction>
</rdfs:subClassOf>
...
</owl:Class>
```

- Cardinalitat

`Owl:cardinality, owl:minCardinality i owl:maxCardinality` restringeixen la propietat a tenir un nombre exacte, mínim o màxim d'elements, respectivament, dins la classe amb que es relaciona.

En el següent exemple `hasVintageYear` ha de tenir un i només un valor dins els enters positius.

```
...
<owl:Restriction>
<owl:onProperty rdf:resource="#hasVintageYear"/>
<owl:cardinality
rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
</owl:Restriction>
...
```

- *hasValue*

`owl:hasValue` permet especificar que un element individual podrà ser membre d'una classe si, al menys un valor de la propietat restringida, és l'indicat per `owl:hasValue`. És una restricció local a la classe on s'aplica.

- **Mapejat d'ontologies**

Si es vol que les ontologies tinguen una bona implantació cal que siguen ampliament compartides. A més, donat l'esforç intel·lectual que suposa el desenvolupament d'una ontologia és necessari que siguen reusades. Per suposat també han de poder ser combinades. Molt de l'esforç que implica crear una ontologia es dedica a unir classes i propietats per tal de maximitzar les implicacions. Si trobem una ontologia d'ús extens i refinada, el més lògic és adoptar-la.

Veiem els mecanismes que OWL té per facilitar la combinació de classes i propietats.

o Equivalència entre classes i propietats

`equivalentClass` i `equivalentProperty` serveixen per indicar que una classe o propietat d'una ontologia és equivalent a una classe o propietat d'una altra ontologia. Cal tenir en compte que les ontologies no siguen contradictòries, perquè aleshores no hi hauria elements individuals ni relacions que compliren la combinació.

Per exemple a dins de l'ontologia *food* podem definir una classe equivalent a la classe *Wine* de l'ontologia *wine*, per tal d'usar les característiques del vins en la descripció dels plats.

```
   Document que descriu l'ontologia Food:
...
   <owl:Class rdf:ID="Wine">
   <owl:equivalentClass rdf:resource="&vin;Wine"/>
   </owl:Class>
...
```

o Identitat entre elements individuals

`sameAs` declara que dos elements són idèntics de forma similar a com ho fa amb les classes.

```
<Wine rdf:ID="MikesFavoriteWine">
   <owl:sameAs rdf:resource="#StGenevieveTexasWhite" />
</Wine>
```

Es pot comprovar a l'exemple que OWL no assumeix la unicitat de noms, en aquest cas `MikesFavoriteWine` i `StGenevieveTexasWhite` són el mateix element.

o Diferenciar entre elements individuals

`differentFrom` i `AllDifferent` són mecanismes contraris a `sameAs`, en definir que certs elements són mútuament distints. En el següent exemple es defineix que *Dry*, *Swee*t i *OffDry* són diferents.

```
<WineSugar rdf:ID="Dry" />
```

```
<WineSugar rdf:ID="Sweet">
   <owl:differentFrom rdf:resource="#Dry"/>
</WineSugar>

<WineSugar rdf:ID="OffDry">
   <owl:differentFrom rdf:resource="#Dry"/>
   <owl:differentFrom rdf:resource="#Sweet"/>
</WineSugar>
```

- **Classes complexes**
  - Operadors de conjunts

`intersectionOf` defineix una classe especificant els seus membres. En l'exemple següent es defineix la classe `WhiteWine` com la intersecció de la classe `Wine` i allò que té color `(hasColor) White`.

```
<owl:Class rdf:ID="WhiteWine">
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#Wine" />
      <owl:Restriction>
         <owl:onProperty rdf:resource="#hasColor" />
         <owl:hasValue rdf:resource="#White" />
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```

`unionOf` defineix una classe con la unió de dues extensions. En el cas de l'exemple, la classe `Fruit` està composta per les extensions `SweetFruit` i `NonSweetFruit`.

```
<owl:Class rdf:ID="Fruit">
  <owl:unionOf rdf:parseType="Collection">
         <owl:Class rdf:about="#SweetFruit" />
         <owl:Class rdf:about="#NonSweetFruit" />
  </owl:unionOf>
</owl:Class>
```

`complementOf` selecciona tots els elements d'un domini que no pertanyen a una classe. Per exemple `NonComsumableThing` és allò complementari de `ConsumableThing`.

```
<owl:Class rdf:ID="ConsumableThing" />
...
<owl:Class rdf:ID="NonConsumableThing">
      <owl:complementOf rdf:resource="#ConsumableThing" />
</owl:Class>
```

  - Classes enumerades

`oneOf` permet especificar una classe enumerant tots els seus membres.

  - Classes disjuntes

`disjointWith` serveix per especificar que els elements que pertanyen a una classe no pertanyen, a cap, d'una sèrie d'altres classes.

- **Versionat d'ontologies**

  Les ontologies, com qualsevol programari, canvien al llarg del temps, com a conseqüència cal versionar-les. Es poden enllaçar versions prèvies amb `owl:priorVersion` i l'URI de l'anterior ontologia.

  Com que les versions anteriors poden contenir declaracions contradictòries amb l'ontologia actual es pot etiquetar la seua compatibilitat amb `owl:backwardCompatibleWith` o `owl:incompatibleWith`. L'etiqueta `owl:versionInfo` permet fer anotacions sobre classes i propietats a més d'ontologies.

  Cal fer notar que el versionat pot ser necessari a nivells inferiors a l'ontologia com ara classes, propietats, elements individuals i fins i tot a les restriccions.

2.7. SPARQL

SPARQL és un llenguatge que permet consultar i manipular contingut RDF a la web o en magatzems RDF, de forma similar a com SQL permet consultar dades en bases de dades relacionals. SPARQL són les sigles d'*SPARQL Protocol and RDF Query Language*. SPARQL és un dels estàndards del W3C i la seua última versió és la SPARQL 1.1.

L'estructura general d'una sentencia SPARQL té les següents seccions:

PREFIX (Espai de noms de prefixes)

SELECT | CONSTRUCT | ASK | DESCRIBE (Llista de resultats desitjats / Forma de consulta triada)

FROM (Conjunt de dades RDF consultat)

WHERE (Patrons de triples demanats)

ORDER BY, DISTINCT, etc... (Modificadors)

Les consultes SPARQL poden tenir quatre formes que s'expliquen a continuació juntament amb uns senzills exemples.

- El format *SELECT* dona com a resultat el conjunt de variables especificat i que coincideixen amb el patró demanat. Per exemple:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?nameX ?nameY ?nickY
WHERE
  { ?x foaf:knows ?y ;
      foaf:name ?nameX
    ?y foaf:name ?nameY
    OPTIONAL { ?y foaf:nick ?nickY }
  }
```

pot donar com resultat:

```
nameX   nameY   nickY
-----   -----   -----
"Alice" "Bob"
```

```
                                    "Alice" "Clare" "CT"
```

- El format *CONSTRUCT* retorna un graf RDF format per cada solució de la consulta, amb les variables substituïdes pels seus valors i unint els *triples* obtinguts en el graf RDF esmentat. Exemple de consulta de construcció:

```
PREFIX foaf:      <http://xmlns.com/foaf/0.1/>
PREFIX vcard:     <http://www.w3.org/2001/vcard-rdf/3.0#>
CONSTRUCT   { <http://example.org/person#Alice> vcard:FN ?name }
WHERE       { ?x foaf:name ?name }
```

Crea les *vcard* a partir del contingut RDF consultat:

```
@prefix vcard: http://www.w3.org/2001/vcard-rdf/3.0#

<http://example.org/person#Alice> vcard:FN "Alice"
<http://example.org/person#Alice> vcard:FN "Bob"
```

- El format *ASK* serveix per comprovar si un patró de consulta té solució o no. No es retorna cap informació. Per exemple:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
ASK { ?x foaf:name "Alice" }
```

Pot donar com resultat:

```
yes
```

- El format *DESCRIBE* retorna un únic graf RDF que conté dades RDF sobre els recursos de la solució. Per exemple:

```
PREFIX ent: <http://org.example.com/employees#>
DESCRIBE ?x WHERE { ?x ent:employeeId "1234" }
```

Pot retornar una descripció del treballador i algunes dades més:

```
@prefix foaf: <http://xmlns.com/foaf/0.1/>
@prefix vcard: <http://www.w3.org/2001/vcard-rdf/3.0>
@prefix exOrg: <http://org.example.com/employees#>
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
@prefix owl: <http://www.w3.org/2002/07/owl#>
_:a         exOrg:employeeId "1234" ;
            foaf:mbox_sha1sum "ABCD1234" ;
            vcard:N
                  [ vcard:Family "Smith" ;
                  vcard:Given "John" ]

foaf:mbox_sha1sum rdf:type owl:InverseFunctionalProperty
```

SPARQL té diversos formats de sortida per tal que els resultats siguen llegibles de forma automàtica. Aquestos són XML, JSON *JavaScript Object Notation*, CSV *Comma Separated Values* i TSV *Tab Separated Values*.

SPARQL en la seua última versió, la 1.1, ja disposa d'operacions d'actualització, anomenades genèricament SPARUL. Les operacions disponibles són *update, create* i *remove*, les quals s'apliquen sobre grafs RDF ubicats en magatzems RDF.

A continuació es mostren un parell d'exemples:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

INSERT DATA { <http://www.example.org/alice#me> foaf:knows [ foaf:name
"Dorothy" ] } ;

DELETE { ?person foaf:name ?mbox }
      WHERE { <http://www.example.org/alice#me> foaf:knows ?person
            ?person foaf:name ?name FILTER ( lang(?name) = "EN" ) }
```

3. Estat de l'art dels SGBD a la web semàntica

En els darrers anys ha sorgit el concepte de Web Semàntica, dins d'aquesta àrea els magatzems semàntics són tan necessaris com els servidors HTTP ho són en l'àrea de la web tradicional. Durant aquest període el W3C ha desenvolupat un ventall d'estàndards relacionats amb la Web Semàntica, alguns dels quals s'han esmentat a l'apartat anterior. Aquests estàndards juguen un paper similar, en la web semàntica, al que SQL té en els SGBD relacionals.

El terme magatzem semàntic té diversos sinònims, potser menys usats, com magatzem RDF/*triple*, dipòsit (*repositori)*, servidor d'ontologies, raonador, base de coneixements, etc... Cadascun dels termes fa referència a l'enfocament de les distintes implementacions o al camp d'aplicació al que està orientat. En el present treball s'usa normalment el terme *SGBD semàntic* o *magatzem semàntic*.

"Magatzem semàntic" fa referència a sistemes que emmagatzemen dades estructurades, normalment RDF, referides a ontologies. Aquests sistemes també permeten inferir o raonar sobre les dades. Comparat amb l'enfocament dels SGBD relacionals, aquests permeten canviar i combinar esquemes de dades més fàcilment i interpretar dades automàticament.

Els magatzems semàntics objecte d'estudi en aquest apartat són AllegroGraph, Oracle Semantic Technologies, OWLIM i Virtuoso Universal Server.

Un dels magatzems proposats per aquest projecte és *eXist*. Una vegada consultada la informació disponible a la seua pàgina web, es veu que és un magatzem XML pur. Es comprova que efectivament disposa d'eines específiques per XML com XUpdate, XQuery Update i interfícies per l'entorn HTTP, de fet indica que és recomanable pel disseny de pàgines web amb tecnologia 2.0 però no fa cap referència a tècniques semàntiques en la seua documentació *(http://exist-db.org, 2012)*. Podria ser útil com a capa d'emmagtzemament per un SGBD semàntic però no he trobat cap referència al respecte. Per aquestes raons he descartat el seu estudi.

3.1. Conceptes bàsics dels magatzems semàntics

En aquest apartat s'introdueixen els conceptes en què es basen les implementacions existents de magatzems semàntics i els enfocaments d'aquests.

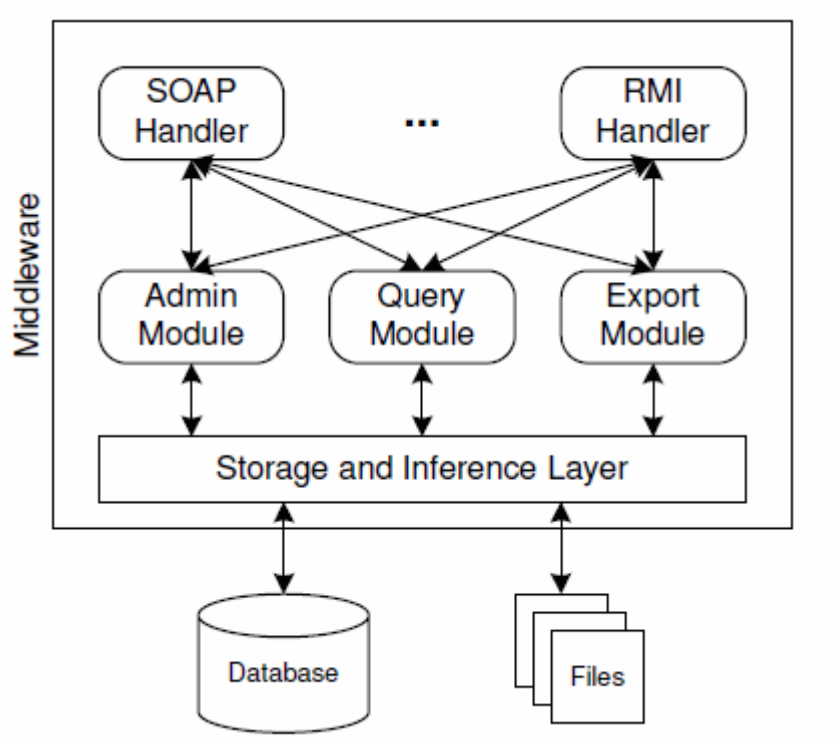
3.1.1. Arquitectura genèrica

Els principals elements que composen un magatzem semàntic són el dipòsit de dades RDF i una capa superior formada pel *middleware* (programari intermediari).

Els possibles dipòsits poden ser bases de dades, memòria o fitxers en disc, això no deu afectar als mètodes d'accés.

La capa del programari intermediari inclou, en ordre creixent, la capa d'inferència i emmagatzemament (anomenada SAIL en Sesame, "Storage And Inference Layer), els diferents mètodes d'accés i els protocols de crida local i remota.

Figura 4. Arquitectura genèrica dels magatzems semàntics (basada en Sesame).



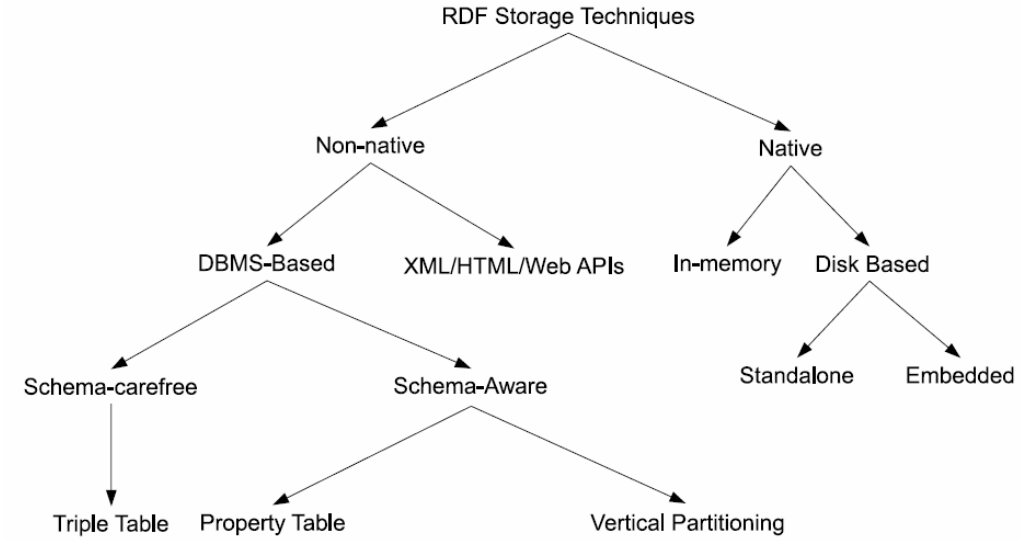
(A. Hertel, J. Broekstra., and H. Stuckenschmidt, 2008)

3.1.2. Taxonomia dels magatzems semàntics

Els enfocaments per implementar magatzems semàntics segueixen diferents estratègies. La divisió principal s'estableix entre els que compleixen directament el model RDF (natius) i els que adapten altres SGDB o altres sistemes d'emmagatzemament per acomplir el model (no natius).

En la següent figura es mostra la taxonomia completa. En els següents apartats s'expliquen les característiques d'ambdues estratègies:

Figura 5. Categorització dels magatzems semàntics.



(D. Faye, O. Cure, O. Blin, 2012)

3.1.3. Magatzems semàntics natius

Els magatzems natius proveeixen una forma d'emmagatzemar dades RDF pròxima al propi model RDF, evitant alhora el mapejat sobre un SGBD d'altre tipus. Es divideixen en *basats en disc (persistents)* i *basats en memòria*.

Els magatzems *basats en memòria* usen una porció important de la memòria per posar-hi un graf RDF complet. Perquè l'accés siga ràpid cal utilitzar tècniques d'indexat i de processament eficient. Les tasques més costoses són l'anàlisi i carrega de fitxers RDF, i la creació dels índexs. Tasques que òbviament cal repetir en cas de reiniciar el sistema, en no ser persistents. Una vegada en funcionament, el que cal és que hi haja memòria suficient pels algoritmes de cerca i que aquests siguen prou eficients ja que aquesta és l'operació més habitual en contrast amb les actualitzacions.

Els SGBD RDF *basats en disc* usen fitxers per guardar les dades RDF en el sistema de fitxers. Usen tècniques d'indexat, com *B-Tree*, per millorar l'accés a les dades. Cal fer notar que les cerques RDF en disc són tan lentes que resulten inacceptables *(D. Faye, O. Cure, O. Blin, 2012)*, per tant aquests sistemes resulten útils per donar persistència a les dades però les operacions s'han de fer igualment en memòria perquè siguen eficients. En aquest cas les carregues en memòria són menys costoses, que en els sistemes *basats en memòria*, perquè els índexs i dades RDF ja estan creats en disc i no cal refer-los en cada carrega a memòria. Les actualitzacions fetes en memòria són portades a disc de forma periòdica (*checkpoint*, punt de control).

Finalment els SGBD *basats en disc* es poden dividir en aquells que són *independents* o *encastats*. Els *independents* guarden, transmeten i processen les dades RDF pels seus propis medis. En canvi els *encastats* formen part d'una aplicació específica o marc de treball RDF, per tant només són accessibles a través del marc de treball i la representació de dades RDF també es fa indirectament.

3.1.4. Magatzems semàntics no natius

Entre els magatzems no natius hi ha dos dipòsits de dades RDF habituals: SGBD (normalment relacionals) i altres sistemes capaços d'obtenir dades RDF de documents XML com *GRDDL* (Gleaning Resource Descriptions from Dialects of Languages) per pàgines XHTML i *microformats* per reusar etiquetes HTML/XHTML aconseguint que dades que eren per l'ús de persones siguen processables automàticament.

Els SGBD utilitzats per emmagatzemar dades RDF usen dues tècniques d'organització distintes, anomenades *schema-carefree* i *schema-aware*.

En els *schema-carefree* s'usa una sola taula per guardar l'esquema RDF/S i les descripcions dels recursos en forma de *triples* (subjecte, predicat, objecte). Aquesta tècnica també s'anomena *Triple Table*.

Els SGBD *schema-aware* aprofiten les propietats de l'esquema RDF/S i les classes per definir un conjunt taules per adaptar-se millor a l'esquema esmentat. Finalment direm que els SGBD *schema-aware* es subdivideixen en dos enfocaments: *property-table* i *vertical partitioning*, ambdós tracten de millorar l'adaptació a l'esquema RDF.

3.1.5. Programari intermediari

Com ja s'ha vist, la capa de programari intermedi implementa els mètodes d'accés al magatzem físic de dades RDF. Aquest a més del mecanisme d'inferència proveeix funcions per crear, consultar i esborrar dades del magatzem.

Afegir dades requereix l'anàlisi i potser la validació de les sentències RDF. Açò es pot fer per mig de crides a una API, llegint dades RDF des d'un fitxer o d'un origen en línia. Per l'anàlisi hi ha diversos *parsers* (analitzadors) com ARP de Jena i Raptor RDF *(A. Hertel, J. Broekstra., and H. Stuckenschmidt, 2008)*. En l'esborrat s'ha de tenir molta cura perquè un esborrat individual pot comportar l'esborrat d'altres sentències requerint el recàlcul del tancament deductiu, per tant pot ser molt costós.

Per altra banda les consultes necessiten un llenguatge o una API i la corresponent interpretació i traducció en crides al magatzem RDF físic. Els llenguatges de consulta més comuns són SPARQL, RQL i RDQL. En cas de tenir un SGBD relacional per davall, es farà una traducció a SQL. L'optimització la sol fer la base de dades, però hi ha casos en què pot ser convenient fer-la en la capa de programari intermediari si el motor de consulta és independent de la base de dades *(A. Hertel, J. Broekstra., and H. Stuckenschmidt, 2008)*.

També es sol implementar la possibilitat d'exportar dades per portar-les a altres sistemes. Per fer-ho les ontologies i les dades són serialitzades a fitxer, els formats habituals són N-Triple, notació N3 i RDF/XML.

3.1.6. Escalabilitat

L'escalabilitat dels magatzems de dades depèn del fet que el seu rendiment no decaiga conforme la quantitat de dades emmagatzemades creix. Hi ha conjunts de proves que es realitzen per poder saber l'escalabilitat dels diferents enfocaments que hi ha per implementar magatzems semàntics. Bàsicament apliquen les proves especificades a SGBD concrets i les fan públiques. Un banc de proves ben elaborat i amb un bon conjunt de ferramentes és LUBM (Lehigh University Benchmark), està orientat a aplicacions OWL però es pot aplicar a la majoria de magatzems RDF.

La possibilitat de crear *clústers* de magatzems RDF, a més de proporcionar alta disponibilitat també proveeix escalabilitat.

3.2. Introducció als SGBD semàntics estudiats

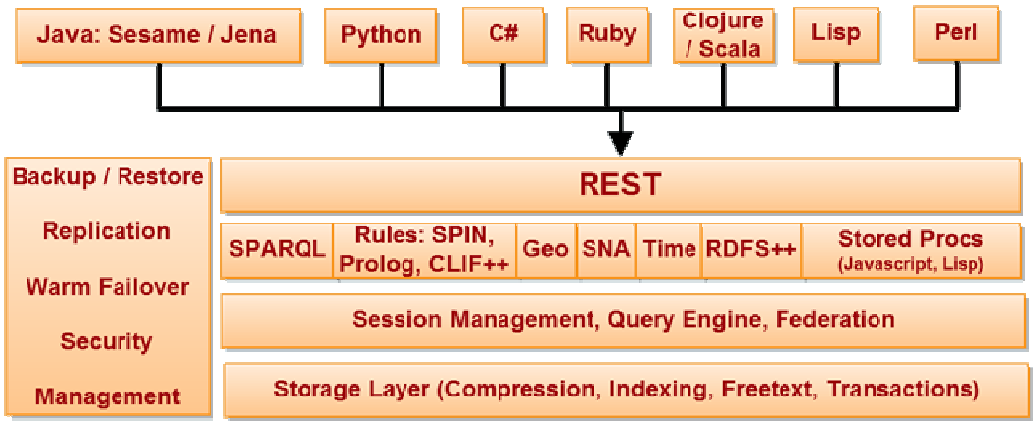
3.2.1. AllegroGraph

AllegroGraph és una base de dades i marc de treball, per desenvolupar aplicacions per la Web Semàntica, desenvolupada per Franz Inc. Pot allotjar dades com *triples* de forma nativa i consultar-les amb SPARQL i Prolog, a més pot aplicar raonament RDFS++. AllegroGraph s'instal·la sobre Linux 64-bit (rpm per RHEL/Fedora i comprimit per la resta de distribucions). Suporta Federació de bases de dades, anàlisi de xarxes socials, dades geoespacials i raonament temporal. Les edicions ofertades són *Free* (fins 5 milions de *triples*), *Developer* (fins 500 milions de *triples*) i *Enterprise* (ilimitada). Els clients que permet són Java, Python, Lisp, Clojure, Ruby, Perl, RDF::Trine, C# i Scala. En els bancs de proves ha demostrat poder emmagatzemar des de 1.106 milions de *triples* en 37 minuts fins a 1 bilió de *triples* en 338 hores amb diferents plataformes de maquinari *(franz.com[1], 2012)*. La seua última versió és la 4.5 i algunes de les implantacions reals d'AllegroGraph són:

- **NASA** - Modeling knowledge of assets in an Enterprise
- **Pfizer** - Knowledge Sharing Using Semantic Technologies
- **Department of Defense** - Collaborative Workspace for Analysts

(franz.com[2], 2012)

Figura 6. Arquitectura d'AllegroGraph.



(Font: franz.com[1], 2012)

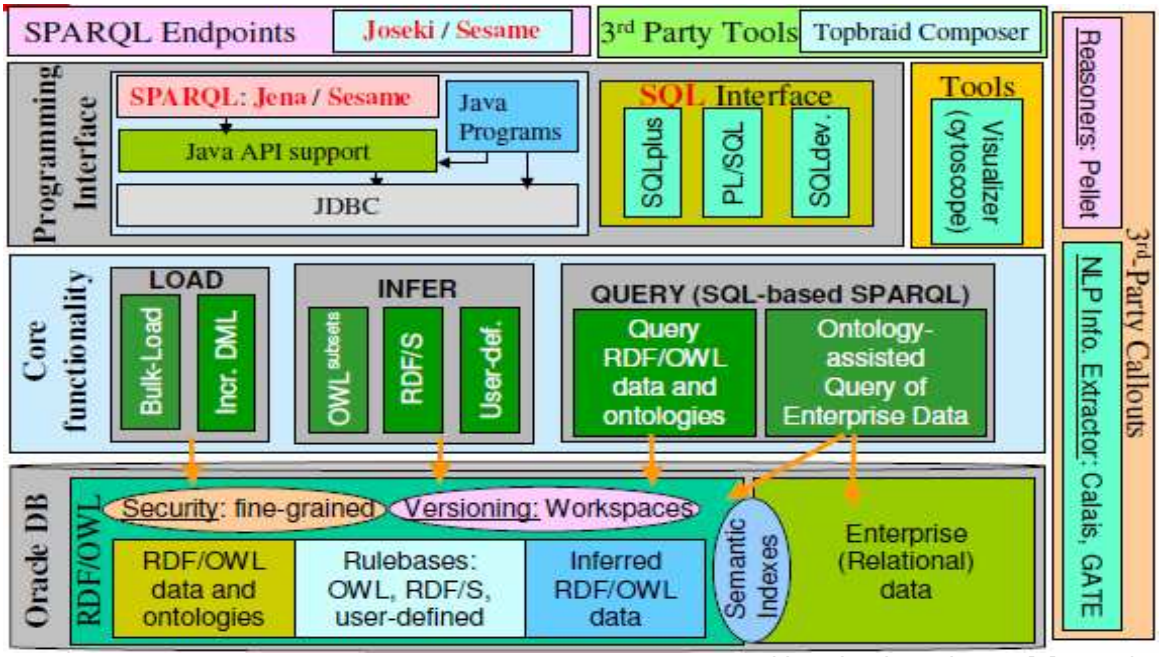
3.2.2. Oracle Semantic Technologies

Oracle Semantic Technologies forma part d'Oracle Spatial 11g, una opció d'Oracle Database 11g Enterprise i no és distribueix en edicions inferiors. És un SGBD relacional, però amb un magatzem RDF en paral·lel al relacional, per tant és un magatzem híbrid (relacional/RDF). Dona suport natiu pels estàndards d'inferència RDF, RDFS, OWL i SKOS, a més permet la definició de regles per part de l'usuari. Permet l'ús de patrons de consulta, similars als d'SPARQL, encastats a SQL. Oracle és un servidor independent que es pot instal·lar sobre Windows, Linux, Mac OS, HP-UX, Solaris i altres. S'han fet proves d'escalabilitat que han arribat als 10.000 milions de triples inserits *(www.oracle.com[1], 2012)*. La darrera versió és la 11GR2. Entre el clients que han desenvolupat projectes amb Oracle Semantic Technologies es troben:

- Swiss Institute of Bioinformatics.
- National Geospatial Intelligence Agence.
- Thomson Reuters (Westlaw).

(download.oracle.com[1], 2012)

Figura 7. Arquitectura d'Oracle.



(download.oracle.com[1], 2012)

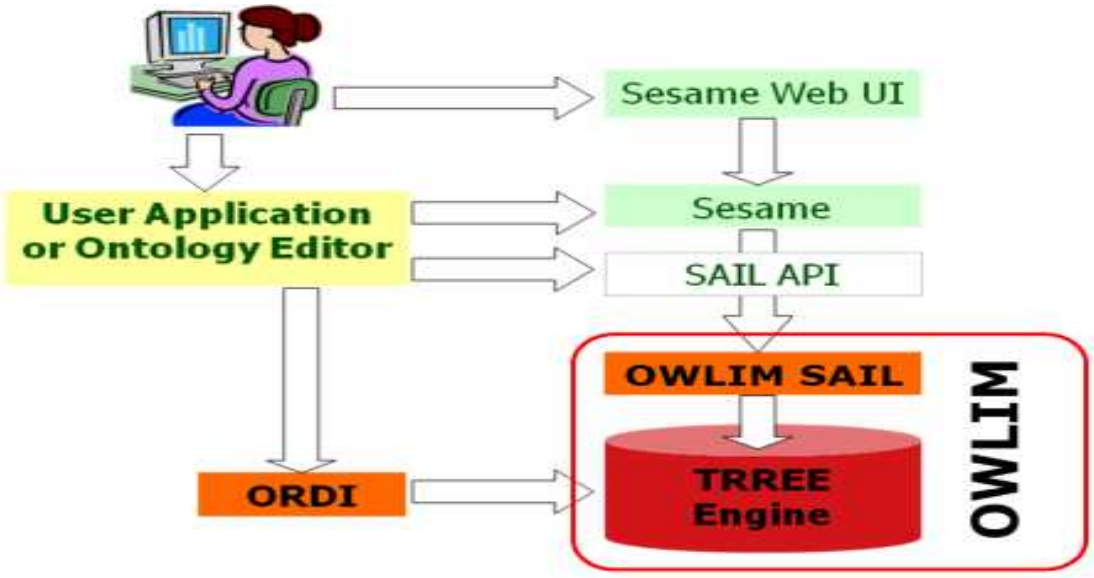
3.2.3. OWLIM

OWLIM és la família de magatzems semàntics de la casa Ontotext. Està dissenyat com a SAIL (Storage and Inference Layer) pel programari intermediari RDF Sesame, a través del qual és accedit. Entre les seues característiques destaca que és un motor de base de dades natiu implementat en Java i dona suport semàntic per RDFS, OWL 2 RL i OWL 2 QL. Es pot instal·lar com a llibreria de crides per una aplicació o dins un servidor d'aplicacions com Tomcat, per tant es pot utilitzar amb qualsevol sistema operatiu que soporte Tomcat. Les edicions disponibles d'OWLIM són, en ordre creixent, SwiftOWLIM, OWLIM-Lite, OWLIM-SE (cerca de text complet, dades geoespacials, optimitza owl:sameAs) i OWLIM-Enterprise (el mateix que OWLIM-SE i replicat en clúster). La seua última versió és la 4.3 i fou lliurada el 4 de Novembre de 2011. Alguns dels projectes reals on s'usa OWLIM són:

- **BBC:** 2010 Football World Cup website
- **LinkedLifeData:** servei públic que consolida més de 25 bases de dades biomèdiques.

(www.ontotext.com[1], 2012)

Figura 8. Arquitectura d'OWLIM.

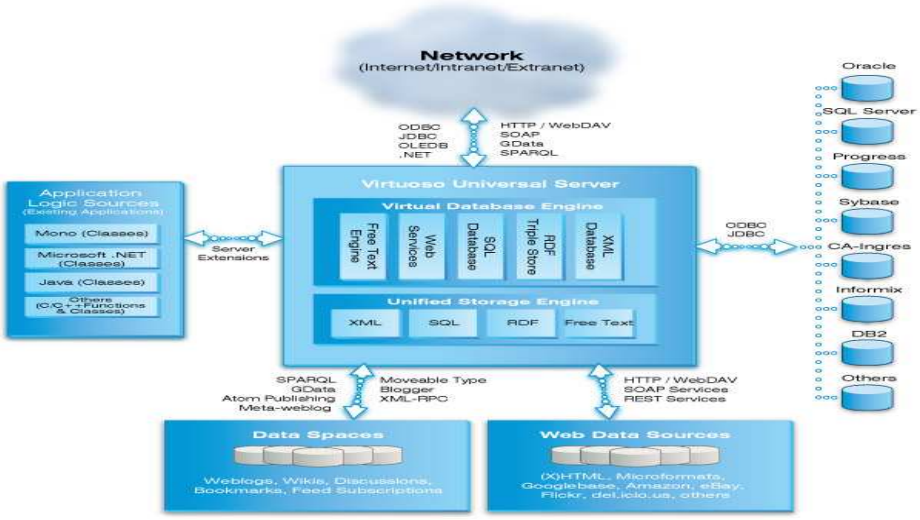


(www.ontotext.com[2], 2012)

3.2.4. Virtuoso Universal Server

Virtuoso Universal Server és un producte de la casa OpenLink Software. Es pot considerar que és una plataforma d'emmagatzemament universal en permetre l'allotjament de Web i fitxers, i disposar d'un magatzem objecte/relacional i XML/RDF natiu. A més implementa un programari intermediari per accés universal a dades donant accés transparent a bases de dades d'altres venedors (Oracle, MS SQL Server, DB/2, Informix i altres). Disposa d'una edició personal i una empresarial, ambdues comercials, que es poden instal·lar als sistemes operatius Windows, Linux, Solaris, AIX, HP-UX, MacOS i altres. També hi ha disponible una versió de codi obert. En un ordinador amb 16GB de memòria es poden gestionar adequadament 1000 milions de triples. La darrera versió comercial és la 6.3 i la de codi obert és la 6.1.5. Aquest SGBD ha estat usat pel projecte *dbpedia.org*, que estructura la informació de la *Wikipedia*.

Figura 9. Arquitectura de Virtuoso Universal Server.



(Font: http://virtuoso.openlinksw.com, 2012)

3.3. Funcionalitats generals dels magatzems semàntics

De l'estudi fet dels conceptes bàsics de la web semàntica, es dedueix un seguit de requeriments que els magatzems semàntics han d'acomplir per fer possible la persistència de les dades semàntiques, el compliment dels estàndards del W3C, el raonament, la filosofia *linked data* i altres.

Per a preparar aquest apartat s'ha recopilat informació de diversos SGBD semàntics nascuts en els darrers anys i que són objecte d'implantació real o d'estudi i investigació. Com a resultat, de la informació obtinguda, a continuació es mostra el ventall de funcionalitats que ofereixen per acomplir els requeriments de la web semàntica.

3.3.1. Llenguatges de consulta

Donat que els *triples* RDF es representen en XML es pot pensar que amb llenguatges de consulta XML es pot accedir a la informació semàntica, però no són adequats per representar grafs RDF. Aleshores els SGBD semàntics, com els d'altres tipus, proporcionen llenguatges de consulta específics per interrogar directament als SGBD i per ser incorporats dins de llenguatges de programació. La majoria incorporen llenguatges de consulta RDF, el més habitual és SPARQL però hi ha altres, com RDQL o SeRQL.

Taula 3. Llenguatges de consulta als SGBD semàntics estudiats.

SGBD	Llenguatges de consulta
AllegroGraph	• SPARQL La implementació d'AllegroGraph s'anomena *twinql* i suporta totes les consultes especificades per SPARQL 1.1 (cal triar el motor *sparql-1.1* en la instal·lació*)*. AllegroGraph dona suport parcial per SPIN, això permet: o Codificar consultes SPARQL en RDF, com *triples*. o Definir regles i restriccions en SPARQL. o Definir noves funcions, en SPARQL, i usar-les com filtres. o Guardar consultes SPARQL com plantilles per a reusar. o Implementar l'extensió *magic properties*, útil per exemple per la cerca de text lliure en consultes SPARQL. Permet operacions d'actualització que compleixen amb l'especificació del W3C (SPARQL 1.1).
Oracle Semantic Technologies	• SPARQL No hi ha suport natiu, està previst en properes versions. Es suporta a través de *Jena Adaptor Oracle* (conté una API SPARQL completa) *(www.oracle.com[1], 2012).* • SQL Permet fer consultes a l'estil d'SPARQL encastades, permetent combinar dades relacionals amb consultes semàntiques.
OWLIM	• SPARQL

SGBD	Llenguatges de consulta
	Acompleix amb la versió 1.1 del W3C, incloent l'extensió per les operacions d'actualització. • SeRQL (Sesame RDF Query Language) Forma part de Sesame. Segons els seus creadors combina les millors característiques d'altres llenguatges (RQL, RDQL, N-Triples, N3) amb alguns afegits propis. Inclou funcionalitats no existents a SPARQL, entre altres: o Transformació de grafs. o Suport per RDF Schema. o Suport per tipus de dades XML Schema.
Virtuoso	• SPARQL L'SGBD l'incorpora amb algunes extensions, com *business intelligence*. Afegeix seguretat basada en la del servidor que treballa a nivell de fila, autenticació i suport per grafs anomenats. No incorpora les característiques SPARQL: caràcters unicode i no permet comentaris quan està encastat en SQL. Incorpora algunes extensions a SPARQL: • Sentències SPARUL d'actualització (insert, modify, load,...). • Clàusules *define* pel compilador. • És possible passar arguments des de fora la sentència. • Funcions agregades. • S'han afegit operadors per mapejar dades relacionals a RDF. • Analitzador XML Integra un potent analitzador XML que dona suport per XPath, XQuery i validació XSLT XML Schema.

3.3.2. Raonament i regles d'inferència

En els SGBD semàntics el motor d'inferència sol estar en la capa de programari intermediari, que hi ha a sobre del magatzem RDF físic, però segons la implementació també es poden definir algoritmes d'inferència com a procediments emmagatzemats, deixant la tasca d'inferència al propi SGBD.

La inferència, en RDF, s'especifica per les regles de vincle RDF(S). Es divideixen en dos tipus:

- Inferència de tancament transitiu de les propietats `rdfs:subClassOf` i `rdfs:subPropertyOf.`
- Inferència de membres de classes analitzant l'ús de propietats i dominis o rangs.

Un possible enfocament és calcular el tancament transitiu, amb un algorisme recursiu, i deixar-lo en un vista. Es construeix una vista per la classe i s'afegeixen vistes per les seues subclasses examinant les declaracions `rdfs:subClassOf`. Es pot fer el mateix per cada propietat seguint les relacions `rdfs:subPropertyOf`.

Un altre enfocament és un sistema de regles de producció per generar nous fets a partir dels existents aplicant *forward chaining* o *backward chaining* a una consulta.

- **Inferència prèvia (forward chaining/eager eval*uation)***

En aquest cas es fa l'avaluació deductiva primer. Es parteix dels fets existents i aplicant les regles s'obtenen nous fets que recursivament es combinen amb els primers fins que no se'n generen més. El temps per avaluar la consulta serà reduït però pot generar-se una gran quantitat de dades emmagatzemades *(A. Hertel, J. Broekstra., and H. Stuckenschmidt, 2008)*.

- **Inferència en temps de consulta (backward chaining/*lazy evaluation)***

Només s'avaluen els vincles resultat d'una consulta o un fet existent i per tant no es generen vincles innecessaris, estalviant espai o memòria *(A. Hertel, J. Broekstra., and H. Stuckenschmidt, 2008)*.

El compromís entre rendiment i espai s'aconsegueix avaluant per avançat aquelles regles que generen pocs vincles i en temps de consulta les que requereixen més espai.

Taula 4. Tractament del raonament en cada SGBD estudiat.

SGBD	Raonament i regles d'inferència
AllegroGraph	El motor de raonament s'anomena RDFS++, suporta tots els predicats RDFS i part dels d'OWL (`owl:inverseOf`, `owl:sameAs`, `owl:TransitiveProperty`) i amb un mòdul opcional suporta a més (`owl:hasValue`, `owl:someValuesFrom` i `owl:allValuesFrom`). També suporta raonament Prolog. L'*RDFS++ Dynamic Materialization* manté de forma dinàmica els vincles ontològics, que en comparació amb la materialització estàtica, permet que les actualitzacions siguen més ràpides i les consultes es puguen reprendre abans *(franz.com[1], 2012)*.
Oracle Semantic Technologies	Disposa d'un motor d'inferència nativa eficient i escalable que implementa els termes més usats d'OWL. Permet definició de regles de raonament per l'usuari, i aplica el *forward chaining* per la inferència. En la versió 11.2 optimitza el rendiment d'owl:sameAs, realitza inferència incremental per actualitzar els vincles després les insercions i paral·lelitza la inferència en arquitectures multiprocessador o multinucli.
OWLIM	Fa raonament aplicant *forward chaining*, en concret aplica l'estratègia de materialització total. Aquesta consisteix en aplicar *forward chaining* de forma completa, després de cada actualització es torna a calcular el tancament transitiu perquè estiga disponible per consultar. La materialització té com desavantatges que les actualitzacions i els esborrats són lents i ocupa molt espai en disc o memòria. El gran avantatge és que les consultes són ràpides ja que el raonament previ és complet. OWLIM permet raonament basat en regles, bé definides per l'usuari o elegides d'unes col·leccions predefinides (RDFS, OWL-Horst, OWL2-ql, OWL2-rl,...).

SGBD	Raonament i regles d'inferència
	OWLIM aplica diverses optimitzacions a certes especificacions RDFS i OWL perquè poden tenir un impacte negatiu en el rendiment del motor de raonament. Les sentències afectades són: `owl:sameAs`, `rdfs:Resource`, `rdfs:domain`, `rdfs:range`, `rdf:type`, `owl:SymmetricProperty` i `owl:TransitiveProperty`. A les quals s'han eliminat certes formes de sentències molt ineficients *(www.ontotext.com[3], 2012)*.
Virtuoso	Aplica raonament *backward chaining*, amb la qual cosa no materialitza els vincles implicats en cada operació. Suporta un subconjunt de la semàntica RDFS i OWL, incloent classes jeràrquiques i propietats. Per inferir, primer importa les restriccions, representades pels grafs, en bases de regles. Aquestes bases són entitats persistents que poden ser referenciades des d'SPARQL. El suport que dona per `owl:sameAs` és limitat *(docs.openlinksw.com[1], 2012)*.

3.3.3. Transaccions, recuperació i còpies de seguretat

Com en altres tipus de bases de dades, les transaccions permeten garantir que les dades emmagatzemades són consistents en tot moment. Per fer-ho s'habiliten mecanismes de transaccions que permeten definir blocs de sentències, anomenats transaccions, que o s'executen completament o es fan enrere si no tenen èxit.

Les transaccions han de complir les quatre propietats clàssiques d'atomicitat, consistència, aïllament i definitivitat, anomenades ACID per les seues sigles en angles. Les operacions de les transaccions s'emmagatzemen a un dietari, per assegurar les seues propietats, fer recuperacions i còpies de seguretat.

Els procediments de recuperació permeten recuperar una base de dades a un estat concret aplicant, després de recuperar una còpia, les transaccions posteriors emmagatzemades al dietari de transaccions.

Les còpies de seguretat eviten la perduda de dades. Normalment en l'entorn de les bases de dades cal fer les còpies mentre la base de dades està operativa. El requisit principal és que la còpia efectuada siga coherent. Es poden fer còpies a nivell de la base de dades o del sistema de fitxers.

Taula 5. Transaccions, recuperació i còpies de seguretat als SGBD estudiats.

SGBD	Transaccions, recuperació i còpies de seguretat
AllegroGraph	Suporta completament les transaccions ACID amb les operacions *commit, rollback i checkpointing*. La consistència no contempla regles a nivell d'aplicació, és responsabilitat de l'usuari, i en cas de dues transaccions contradictòries és capaç d'evitar que la segona faça *commit*. L'aïllament s'implementa amb *snapshots* i no existeix la possibilitat de bloquejar un *triple*. Es disposa d'un dietari de transaccions amb una optimització que

SGBD	Transaccions, recuperació i còpies de seguretat
	consisteix en què les transaccions es guarden al dietari i s'apliquen a les dades en memòria únicament, amb la finalitat de millorar el rendiment, de forma periòdica es fa un *checkpoint* que porta els canvis al disc. Es possible fer recuperacions a un punt en el temps. Les còpies de seguretat són en línia i contenen els fitxers de dades i informació addicional per poder restaurar l'estat a l'hora de la còpia.
Oracle Semantic Technologies	Compleix completament amb les transaccions ACID. Usa un nombre SCN (System Change Number) com a marca de temps per cada transacció i per controlar el sistema de recuperació. Usa el mecanisme de bloqueig en dues fases per transaccions distribuïdes i detecta i resol les transaccions dubtoses. Les transaccions s'arxiven inicialment als *redo logs* i quan tenen una certa antiguitat als *archived logs*. Permet recuperació del medi per fitxers de dades danyats i de *Flashback* que recupera, a un punt en el temps, una taula, un espai de taules o tota una base de dades. Disposa de dos tipus de còpia: còpia física de la base de dades (fitxers de dades) i còpia lògica (*export/import*) dels distints objectes de l'esquema. Ambdues en línia i respectant la coherència.
OWLIM	Implementa el nivell d'aïllament *read commited* que permet consultar dades mentre són actualitzades. Disposa de dos modes de transacció: *safe* que porta totes les actualitzacions a disc com a part de la transacció i *bulk-loading* que fa els canvis només a memòria i els descarrega a disc quan es plena l'espai de memòria implicat. El segon mode té bon rendiment però les recuperacions són molt lentes perquè cal refer moltes operacions des del dietari de *logs*. No subministra un sistema de còpia en línia. Per fer una còpia cal detenir l'SGBD i copiar el sistema de fitxers que el conté.
Virtuoso	El suport que dona per transaccions compleix amb les condicions ACID per dades relacionals i RDF. Aplica el *commit* en dues fases per transaccions distribuïdes i és capaç de detectar interbloqueigs i evitar-los detenint una de les transaccions implicades. Disposa del mode *autocommit* que s'aplica a cada fila actualitzada, sense usar dietari, és ideal per grans carregues de dades RDF. Disposa d'un dietari de transaccions, on aquestes es registren fins que es fa un *checkpoint* i s'apliquen a la base de dades. En cas de ser necessari es poden usar per refer operacions posteriors a la restauració d'una còpia. Es poden fer còpies en línia amb una funció pròpia de Virtuoso, que copia l'estat fins l'últim *checkpoint*. També es poden copiar en línia els fitxers de base de dades si es desactiven els *checkpoint*, durant la còpia. Finalment és possible fer còpies fora de línia amb Virtuoso detingut. Per tenir una còpia més enllà de l'últim *checkpoint* (fins el

SGBD	Transaccions, recuperació i còpies de seguretat
	moment actual) cal copiar els fitxer de *logs* associats.

3.3.4. Representació de dades semàntiques, indexat i indexat de text complet

La unitat bàsica d'informació, que un magatzem semàntic ha d'allotjar, és òbviament el *triple*. Per fer-ho, els SGBD semàntics defineixen els tipus de dades i les estructures d'emmagatzemament necessàries.

També és necessari, per raons de rendiment, la creació d'índexs sobre les dades per que siguen accessibles en pocs temps. El problema més important a les consultes en grans magatzems RDF són les *joins* i *unions*, la majoria de les consultes les usen perquè les dades RDF són una col·lecció de tuples de tres columnes. Els índexs que es dissenyen deuen millorar el rendiment d'aquestes operacions per aconseguir que el magatzem siga escalable.

L'indexat de text complet no és un requeriment dels SGBD semàntics, ni els llenguatges RDF el suporten més enllà de les expressions regulars. Així i tot els magatzems semàntics l'implementen donada l'actual demanda.

Taula 6. Representació de dades i indexat als SGBD estudiats.

SGBD	Representació de dades semàntiques, indexat i indexat de text complet
AllegroGraph	Representa els *triples* amb una quintupla (subjecte (s), objecte (o), predicat (p), graf (g) i *triple-id* (i)). Les columnes s, p, o i g són cadenes de caràcters de longitud arbitraria, però per evitar duplicitats no s'emmagatzemen directament si no que se li associa un identificador *UPI Unique Part Identifier*. Un diccionari de cadenes les gestiona i evita duplicitats. Usa un conjunt d'índexs per identificar amb rapidesa conjunts de triples que puguen satisfer un cert patró de consulta. Els conjunts d'índexs s'anomenen: • **spogi**: útil per cerques de patrons coincidents amb les sigles del nom de l'índex (s ,sp, spo, spog). • **posgi**: útil quan es desconeix el subjecte. • **ospgi**: útil quan només es coneix l'objecte. Després s'obtenen amb rapidesa els corresponents subjecte i objecte. • **gspoi, gposi, gospi**: anomenats índex de grafs són útils quan el magatzem de *triple* està dividit en subgrafs. • **i**: és una llista de *triples* organitzats per l'identificador. És útil quan s'usa reificació i especialment per esborrar *triples*. Cadascuna de les lletres representa un component de la quintupla (subjecte, objecte, predicat, graf i *triple-id*). L'ordre de les lletres indica com està organitzat l'índex. Es suporta l'indexat de text complet de diverses formes, basades en l'algoritme *Patricia trie*. Es poden aplicar a camps específics de predicats concrets. Permeten cerques amb comodins i difuses. El procés d'indexat és transaccional i té capacitat per gestionar milers de

SGBD	Representació de dades semàntiques, indexat i indexat de text complet
	milions de documents. És molt flexible en disposar d'un ventall de configuracions modificables *(franz.com[3], 2012)*.
Oracle Semantic Technologies	Emmagatzema els *triples* com grafs dirigits, aquest és un model de dades suportat directament per Oracle Spatial Network Data Model. Els subjectes i objectes són nodes i els predicats són enllaços. Per la resta d'informació dels distints models RDF es proveeix un catàleg, on s'inclouen taules relacionals per emmagatzemar dades com els espais de noms, sentències RDF, nodes buits, predicats, etc. A més dels índexs comuns per dades relacionals (B-Tree, Bitmap,...) hi ha diversos tipus d'índexs específics per dades RDF i ontologies: • **Índexs semàntics**: crea un índex sobre la columna que conté el terme de l'ontologia desitjat, amb la clàusula *INDEXTYPE IS MDSYS.INDEXTYPE*. • **Índexs semàntic de documents**: indexa documents en columnes de tipus *CLOB* i *VARCHAR*. Cal indicar el tipus d'índex *MDSYS.SEMCONTEXT*. • **Índexs de regles**: conté *triples* precalculats inferits a l'aplicar un conjunt de regles base. Es crea amb el procediment *SDO_RDF_INFERENCE.CREATE_RULES_INDEX* . Per indexar text disposa de diversos tipus d'índex: • ***CONTEXT***: útil per documents llargs i coherents (text, MS Word, HTML). • ***CTXCAT***: indicat per cerques mixtes de texts curts, fragments i altres columnes. Pot indexar també les altres columnes involucrades. • ***CTXRULE***: crea una classificació de documents, es basa en una consulta prèvia que defineix el criteri de classificació. • ***CPATH***: indicat per columnes de tipus XML. *(www.oracle.com[2], 2007)*
OWLIM	L'estructura de dades usada és el *quad* (*triple* més el context): <S, P, O, C> (Subjecte de tipus URI, Predicat de tipus URI, Objecte de tipus URI o literal i Context de tipus URI) Inclou el marc de treball ORDI (Ontology Representation and Data Integration), el qual estén l'estructura anterior a una quintupla: <S, P, O, C, {TS1, ..., TSn}> El que s'afegeix és una col·lecció d'URI que identifiquen conjunts de *triples* (*tripleset)*, això serveix per associar una sentència amb un conjunt de *triples* i poder limitar l'abast d'una consulta. Per l'indexat hi ha dos índexs principals POS (predicat-objecte-subjecte) i PSO (predicat-subjecte-objecte). A més hi ha índexs opcionals que l'usuari pot activar segons les necessitats: • **Llista de predicats**: mapeja subjectes i objectes als seus predicats. És útil quan hi ha un nombre gran de predicats (més de 1000). • **Índex de context**: anomenats PCSO i PCOS, accelera les consultes amb l'identificador de context (C). • **Índex de literals**: orientat a consultes amb filtres amb

SGBD	Representació de dades semàntiques, indexat i indexat de text complet
	restriccions literals que usen comparacions i igualtats Els índexs poden ser comprimits, suposa una certa sobrecarrega, però estalvia espai. És útil per grans bases de dades amb discs cars (SSD, Solid State Disk). Per l'indexat de text complet, s'aporten dos enfocaments optatius: • **Cerca de nodes**: és una implementació propietària. Similar a les implementacions en SGBD relacionals. • **Cerca RDF**: basada en Lucene. És un concepte nou que millora l'extracció de dades de recursos RDF en grans magatzems on l'ordre dels resultats és important. *(www.ontotext.com[4], 2012)*
Virtuoso	Utilitza el tipus **IRI_ID** per representar els IRI (Internationalized Resource Identifiers) o URI, en l'estructura de les taules usades per la persistència dels *triples*. De fet el subjecte i el predicat sempre són URI i l'objecte és un URI o un tipus escalar XML Schema. Per bases de dades petites és un enter sense signe de 32 bits i en les grans es poden usar 64 bits. Els literals RDF no es corresponen amb cap tipus de dada usada en SQL. Per donar-li suport existeix el tipus **RDF_BOX** que consisteix en un tipus de dada, un llenguatge i el contingut (si aquest és molt gran es pot referenciar i ubicar-lo en una taula separada). Un *triple* es representa en una taula de quatre columnes (RDF_QUADS): grafs (G de tipus IRI_ID), subjecte (S de tipus IRI_IDI), predicat (P de tipus IRI_ID) i objecte (O de tipus any). L'esquema d'índexs consisteix en 2 índexs complets i 3 parcials sobre la taula RDF_QUADS: • **PSOG:** clau primària. • **POGS:** índex de mapa de bits per cercar pel valor de l'objecte. • **SP:** índex parcial per quan només s'especifica el subjecte (S). • **OP:** índex parcial per quan només s'especifica l'objecte (O) • **GS:** índex parcial per quan només s'especifica el graf (G). *(docs.openlinksw.com[2], 2012)* Existeix un índex opcional per text complet sobre els literals RDF i un predicat **bif:contains** per fer consultes de text complet. A més es pot especificar quines propietats s'indexen i a quins grafs s'aplica.

Com hem vist, el concepte de triple es veu ampliat normalment amb una quarta dada, anomenada graf, context, etc. Açò ens porta al concepte ***graf anomenat*** que ens permet conèixer el context o graf al que un *triple* pertany. Si el valor que conté és nul es diu que el *triple* pertany al graf per defecte. SPARQL suporta la consulta de grafs per defecte (FROM) i anomenats (FROM NAMED).

3.3.5. Seguretat

Els magatzems semàntics proporcionen mecanismes de seguretat, a diferents nivells, per tal de garantir un accés controlat a les dades. El fet que la filosofia *linked data* promoga la compartició de dades no vol dir que totes les dades compartides siguen públiques. Els gestors defineixen tipus d'accés que van des de l'accés públic fins a diferents graus de confidencialitat.

La seguretat s'implementa amb la creació d'usuaris, la definició de rols i les operacions que cada rol té permís d'executar. Addicionalment tenim el concepte de granularitat que fa referència a l'objecte de l'SGBD al que s'apliquen els rols i operacions. La granularitat pot ser fina o gruixuda segons la grandària de l'objecte de base de dades al que faça referència. Gruixuda pot ser a nivell de base de dades o graf, fina pot ser a nivell de *triple*.

Taula 7. Consideracions de seguretat als SGBD estudiats.

SGBD	Seguretat
AllegroGraph	Disposa d'una interfície web (AGWebView) que facilita la gestió de la seguretat. Permet la creació d'usuaris i rols amb una granularitat fina. Hi ha predefinides diverses classes d'usuari (*Superuser, Start Session, Eval i Replication*). L'accés es pot donar a nivell de base de dades, una part del catàleg o a tot el servidor de base de dades. Per la seguretat a nivell de *triple* es disposa de filtres de seguretat, amb els que es pot donar accés a tot un magatzem RDF, o limitar-lo a una vista determinada. Els filtres s'apliquen a una combinació d'usuaris o rols amb operacions (afegir, esborrar, consultar). També es pot especificar quins valors de subjecte, objecte o predicat estan permesos, per tant les consultes es filtren conseqüentment i les operacions d'afegir o modificar els *triples* filtrats fallaran quan intenten modificar *triples* no permesos.
Oracle Semantic Technologies	La seguretat d'accés per defecte a les dades semàntiques es fa a nivell de model, mitjançant vistes. L'administrador pot restringir l'accés d'un usuari a *triples* que siguen instàncies d'una certa classe o propietat RDF, fent ús de *Virtual Private Database* d'Oracle. També pot associar etiquetes de seguretat a nivell de *triples* individuals amb *Label Security Option*. Les etiquetes determinen la sensitivitat de les dades o els drets que un usuari ha de posseir per llegir-les o escriure-les. L'administrador assigna etiquetes a usuaris per autoritzar-los.
OWLIM	Ha d'estar instal·lat junt al servidor http Sesame, per separat no és possible controlar l'accés. Es poden definir restriccions d'accés per cada operació (consulta,escriptura,modificació, etc.) i pels rols que agrupen als usuaris. En l'ús del servidor http es pot associar el patró URL i el mètode HTTP (GET,POST,PUT,DELETE) amb un conjunt de rols resultant una granularitat fina. La creació d'usuaris es fa al servidor d'aplicacions Tomcat on s'executa OWLIM. Aquest permet diverses gestions d'usuaris, la usada per defecte és la menys segura *UserDatabaseRealm*.
Virtuoso	Suporta seguretat a nivell de magatzem RDF diferenciada de la

SGBD	Seguretat
	seguretat SQL degut a que les dades RDF resideixen en una sola taula. Els permisos per defecte permeten l'accés als recursos públics i s'apliquen a l'usuari *nobody*. Els permisos s'especifiquen amb 4 bits (Bit 1: lectura, bit 2: escriptura, bit 4: escriptura via *RDF Sponger*, bit 8: permet llistar els membres d'un grup de grafs). Una consulta SPARQL comprova si un usuari té accés a un graf donat comprovant: • 1er: permisos de l'usuari en el grafs específic. • 2on: permisos per defecte de l'usuari en tots els grafs. • 3er: permisos públics en el graf específic. • 4rt: permisos públics en tots els grafs. La configuració inicial de permisos convé fer-la en ordre invers al de la comprovació.

3.3.6. Mètodes d'accés i API disponibles

Els mètodes d'accés són distintes capes dins l'arquitectura dels magatzems, anomenades normalment adaptadors, que incorporen API que donen accés programàtic als magatzems RDF. Aquestes API permeten desenvolupar aplicacions, per emmagatzemar i consultar dades RDF, i aplicar raonament.

Taula 8. Mètodes d'accés

SGBD	Mètodes d'accés i API disponibles
AllegroGraph	Proveeix una interfície basada en el protocol REST (Representational state transfer). Sobre aquesta pila REST, AllegroGraph disposa d'API per diversos llenguatges: Java de Sesame i Jena, Python i Lisp. Addicionalment hi ha adaptadors de codi obert pertanyents a projectes de la comunitat que proveeixen API per C#, Ruby, Clojure, Scala i Perl. AllegroGraph integra un SPARQL Endpoint accessible per HTTP.
Oracle Semantic Technologies	Disposa d'un adaptador Jena (API Java) que treballa amb *triples*. L'adaptador Sesame (API Java) que treballa amb quads (s, o, p i context). El llenguatge SQL permet l'ús de patrons a l'estil d'SPARQL. L'adaptador Jena incorpora un *SPARQL Endpoint* integrat.
OWLIM	Incorpora un adaptador Jena (API de Java) amb suport per SPARQL, un adaptador Sesame (API JAVA) amb suport per SPARQL i SeRQL, un adaptador ORDI (Ontology Representation and Data Integration) per a la integració de dades de l'organització, SPARQL Endpoint de Sesame i el protocol HTTP a l'estil REST.
Virtuoso	Proveeix els adaptadors Jena i Sesame per donar suport API per Java. A més disposa d'un adaptador Redland que consisteix en llibreries 'C' basades en objectes. Redland proveeix suport per consultes SPARQL i RDQL i API pels llenguatges Perl, PHP, Python i Ruby.

3.3.7. Altres funcionalitats dels magatzems semàntics

En aquest apartat es fa un revisió breu a altres funcionalitats que també trobem als magatzems semàntics.

- **Control de canvis.**

És la funcionalitat que permet la creació d'un sistema de versions pels diferents elements del magatzem com poden ser la pròpia base de dades, grafs, jocs de dades, i altres objectes de les bases de dades.

- **Replicació.**

La replicació és el mecanisme que permet definir un servidor semàntic mestre i un o més esclaus, amb diverses topologies possibles, per tal de duplicar les dades del servidor mestre als esclaus, amb diferents estratègies. La finalitat es proveir un repartiment de la carrega de treball i un sistema d'alta disponibilitat. Normalment aquesta funcionalitat es proveu a les versions empresarials.

- **Carrega massiva de dades.**

La seua utilitat més habitual es facilitar la carrega inicial de dades en un sistema de base de dades nou. De vegades la funcionalitat està incorporada en el propi llenguatge de gestió de dades i d'altres en forma d'utilitats separades que permeten les operacions d'importació i exportació de dades.

- **Accés a dades d'altres orígens.**

Normalment els magatzems semàntics disposen de sistemes per connectar amb orígens de dades d'altres tipus, relacionals normalment. La connexió s'estableix per mig d'ODBC i JDBC i es mapeigen les dades relacionals a ontologies RDF amb l'aplicació d'un metaesquema. Com a benefici s'obté que es pot accedir a les dades relacionals amb SPARQL i explotar-les com si foren dades RDF.

3.4. Linked Data

El concepte Linked Data fa referència a l'ús de la web per connectar dades relacionades però que no estaven enllaçades prèviament. La Wikipedia el defineix com "un terme usat per descriure una bona pràctica recomanada per mostrar, compartir i connectar peces de dades, informació i coneixement en la Web Semàntica usant URI i RDF". És ben conegut el projecte coordinat pel W3C, *Linked Open Data,* que promou que les dades estiguen disponibles per tothom. Al setembre de 2011 havia 295 fonts de dades adherides al projecte, amb 31.000 milions de *triples* units per uns 504 milions d'enllaços RDF. Un graf representatiu del projecte es pot consultar a http://richard.cyganiak.de/2007/10/lod/lod-datasets_2011-09-19_1000px.png. Es pot dir que aquest projecte és el paradigma de la utilitat dels magatzems semàntics.

RDF és una excel·lent forma d'integrar dades, en especial si provenen d'orígens diversos. Al projecte esmentat s'hi troben diverses i grans bases de dades interconnectades, per exemple DBpedia, FOAF, Geonames, MusicBrainz, PubMed, US Census Data, BBC Programs i altres fins arribar a 295. Hi ha una expansió en el nombre de bases de dades que es van adherint al projecte, vista l'evolució es pot afirmar que s'ha passat del punt en què es publicaven certs continguts RDF, a un nivell incipient de maduresa doncs hi ha estàndards de qualitat, bones pràctiques i una bona

quantitat de ferramentes d'ajuda per publicar i recuperar dades RDF. Moltes de les bases de dades del projecte tracten un determinat domini i contenen una descripció de classes (ontologia), un conjunt d'instàncies i algunes relacions *owl:sameAs* cap a instàncies d'altres bases de dades del projecte. Algunes altres, com DBPedia, tenen diversos dominis.

Hi ha la visió, de *Linked Open Data*, que la representació única de cada tipus de coneixement farà que combinar jocs de dades siga fàcil. La realitat és que explorar aquest immens graf, no és trivial perquè hi ha magatzems de *triples,* que no ofereixen funcions de descobriment per connectar dues instàncies que es vol relacionar, mentre que altres manquen de domini i rang. Altre problema és que cal conèixer els espais de noms i de vegades els tipus dels objectes. Per millorar aquesta situació calen eines gràfiques que ajuden a la interacció entre els humans i els ordinadors respecte la web semàntica. Les facilitats que poden proporcionar aquestes eines són: construcció gràfica de consultes, capacitat d'anàlisi de grafs, disseny de grafs automàtic, cerca d'objectes per *rdfs:label*, URI, *rdf:type*, un text arbitrari en les propietats i selecció de nodes assistida gràficament.

Com a exemple d'aquestes eines hi ha *Gruff* d'AllegroGraph *(franz.com[4], 2012)*, amb unes capacitats gràfiques molt destacables, igual que *Sentient Knowledge Explorer* d'IO Informatics *(franz.com[5], 2012)* però que apareix al catàleg de productes d'AllegroGraph, *Sesame Windows Client Web (http://sourceforge.net/projects/sesamewinclient)* amb menys capacitats visuals però que permet connectar-se a AllegroGraph, OWLIM i *Virtuoso Faceted Browser* que es veu a l'apartat 4.3.6.

3.5. Comparativa amb els SGBD relacionals

Al llarg del present document hem vist que hi ha dos enfocaments principals per donar suport a l'emmagatzemament de dades RDF: els magatzems natius i els no natius. Els magatzems natius s'han dissenyat seguint els conceptes informadors de la web semàntica i els seus requeriments. Per altra banda, els no natius han aprofitat l'existència dels SGBD relacionals, per la seua ampla implantació, per plasmar-hi els mateixos conceptes semàntics. A continuació es tracten les diferències entre ambdós i els possibles avantatges i inconvenients de cadascun.

Trets diferenciadors dels magatzems RDF respecte els SGBD relacionals:

- **Major flexibilitat davant els canvis en l'estructura de les dades.**

RDF pot representar instàncies de dades i l'esquema que les descriu, això abasta des del registre de dades bàsiques fins a dominis complets. En tots dos casos, RDF també incorpora el coneixement del domini. El propi esquema serveix per comprovar la validesa i consistència de les dades. Quan es fa una carrega de dades no és obligatori que siguen completes, ni que casen amb un esquema preconcebut

Els SGBD relacionals, per la seua banda, quan són usats per dades RDF tenen dificultats per adaptar-se als canvis d'esquema, especialment quan apareixen relacions u a molts que requereixen taules addicionals. El programari intermediari que fa l'adaptació RDF/relacional sol fer transparentment aquest canvis en l'esquema relacional però realment açò fa que el rendiment i l'escalabilitat siguen pitjors. Els SGBD relacionals tenen un gran rendiment quan els esquemes que gestionen són coneguts i poc canviants, aleshores poden optimitzar les consultes

ja que coneixen l'estructura de les dades, com estan emmagatzemades, la longitud de les files, etc.

- **Interoperabilitat entre fonts de dades heterogènies.**

RDF destaca realment en aquest aspecte perquè pot captar metadades o informació de fonts no estructurades (p.ex. text), semiestructurades (p.ex. HTML) o estructurades (p.ex. bases de dades). Donada la senzillesa del model de dades RDF és fàcil crear convertidors per traduir notacions no RDF.

Una vegada que tenim la informació representada en RDF és fàcil incorporar nous atributs, agregar fonts de dades dispars o expressar que dos conceptes són el mateix.

- **RDF és evolutiu i extensible.**

Encara que un model RDF tinga informació poc detallada es pot inferir de les dades existents i d'aquesta forma estendre les dades i l'esquema. Aquestes representacions parcials es poden incorporar igual que les completes, i l'esquema creixerà i evolucionarà conforme es descobreix la nova estructura. Aquesta adaptabilitat és la que permet considerar RDF com un disseny dirigit per les dades.

El model relacional no va ser concebut amb el requeriment d'una integració de fonts dispars, no controlades i a gran escala.

- **Representació en graf.**

La combinació de *triples* RDF dona com a resultat un graf. De fet el model RDF es pot descriure formalment com un graf dirigit. Açò té diversos avantatges:

  - Són modulars i es poden combinar o separar, podent paral·lelitzar el processament i millorar l'escalabilitat. A més un graf extret segueix sent un model RDF vàlid.
  - Els algoritmes de graf han estat objecte d'ampli estudi per part de matemàtics i informàtics. Són conegudes la cerca del camí entre dos nodes, les recerques en profunditat i amplitud, i del camí més curt.
  - Les interconnexions dels grafs permeten explorar les dades de forma contextual aplicant filtres *faceted search* (cerca per facetes).
  - Les relacions u a molts s'expressen directament en tenir l'estructura d'un graf.

- **Accés a dades en la web semàntica (Linked Open Data).**

Les dades RDF identificades universalment pel seu URI poden ser publicades per afavorir el descobriment i enllaçat, aquesta és la base del concepte *Linked Open Data*. Encara que les dades de la nostra organització no siguen públiques es poden incorporar dades externes al nostre context. Avui en dia hi ha milers de milions de sentències RDF disponibles *(M. Bergman, 2009)*.

Una vegada revisades les característiques que mostren els principals avantatges dels magatzem semàntics i les tecnologies relacionades es procedeix a revisar els seus inconvenients:

- L'estructura impredictible dels documents RDF dificulta l'optimització de les consultes als magatzems semàntics, donat que poden contenir un gran

nombres de *triples*. El que es fa és establir unes magnituds d'emmagatzemament inferiors als dels SGBD semàntics per donar un temps de resposta adequat.

- Els SGBD semàntics són menys madurs que els relacionals i tenen menys prestacions. Per exemple el tractament de les transaccions és menys sofisticat.

- Dels dos punts anteriors es dedueix que per aconseguir un major rendiment i escalabilitat, la sortida més fàcil és crear un clúster de servidors, a l'espera que els magatzems semàntics milloren l'optimització de les consultes.

- No és fàcil trobar programadors i administradors.

- No s'integren bé amb les actuals eines de generació d'informes.

De tot allò exposat en aquest punt veiem que els magatzems semàntics tenen les seues avantatges centrades en el camp de la teoria (de grafs), en el model obert de dades en el sentit conceptual (Linked Open Data) i en el tècnic (fonts de dades heterogènies) . Per altra banda les seues limitacions es centren en temes pràctics, com ara optimització i rendiment. El que sembla clar és que cada model d'SGBD té un clar camp d'aplicació, els relacionals el de les dades més estructurades i d'entorns tancats i els semàntics el d'afegir coneixement a les dades i entorns més oberts.

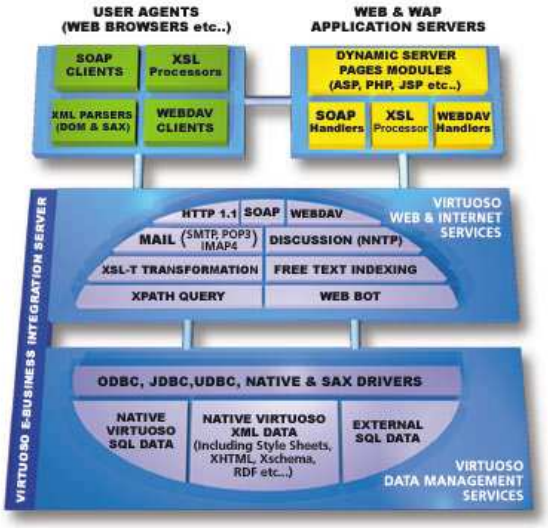
Veient que no hi ha un conflicte d'interessos i que la tecnologia RDF, és un tant immadura però amb una sòlida base, és d'esperar una implantació creixent de magatzems semàntics en el futur. Pot ser queda obert el debat de si usar magatzems RDF natius o no natius però conforme les limitacions dels magatzems natius, eminentment pràctiques, es solucionen aquest debat pot ser es resolga sol.

4. Anàlisi detallat del magatzem semàntic Virtuoso Universal Server

Per aquesta part del projecte ha estat triat Virtuoso Universal Server. Les principals raons són: l'existència d'una versió de codi obert, que facilita les proves de rendiment en no tenir límit de temps, i la gran quantitat de documentació disponible.

Virtuoso Universal Server d'Openlink és un dels primers servidors multiplataforma disponibles. Implementa un servidor de base de dades natiu objecte/relacional i XML/RDF (híbrid), servidor web i de fitxers. Incorpora un programari intermediari per accedir a múltiples fonts de dades externes com Oracle, MS SQL Server, DB/2 i altres bases de dades accessibles per ODBC, JDBC i altres protocols. Es pot instal·lar en un ample ventall de sistemes operatius.

Figura 10 Esquema bàsic de funcionalitats de Virtuoso Universal Server



(docs.openlinksw.com[3], 2009)

Com es pot observar Virtuoso reuneix funcionalitats que van molt més enllà de l'àrea que ens interessa, els magatzems RDF. En els següents punts s'estudiarà en profunditat el model d'emmagatzemament de dades RDF, es farà un petit llistat de les funcionalitats generals i es veuran les funcionalitats específiques RDF en profunditat. A continuació es revisarà com es poden fer amb Virtuoso les operacions de dades RDF més habituals. Finalment es presentaran els resultats obtinguts a les proves de rendiment efectuades.

El contingut d'aquest capítol i dels seus apartats ha estat extret, en la major part, de la referència bibliogràfica *(docs.openlinksw.com[3], 2009)*, un detallat i extens manual editat per Openlink Software Inc.

4.1. Sistema d'emmagatzemament d'informació

Per estudiar el sistema d'emmagatzemament de Virtuoso s'usarà l'enfocament per capes habitual en l'estudi dels SGBD, primerament es descriu la capa del model físic i en segon terme la capa del model lògic. Respecte a la tercera capa, l'esquema conceptual, s'ha de dir que ja ha estat tractat al capítol 2 del present document, és a dir, està compost pels conceptes de la web semàntica.

Virtuoso fou desenvolupat inicialment com un SGBD relacional amb emmagatzemament per files (*rowwise*), després es va redissenyar per ser també un SGBD de grafs RDF amb capacitat d'inferència, incloent el llenguatge SPARQL. Darrerament ha estat revisat per beneficiar-se de l'emmagatzemament comprimit per columnes (*columnwise*) i de l'execució vectorial *(O. Erling, 2012)*. Açò s'ha de tenir en compte en el present punt perquè de vegades pot semblar que es parla d'un

magatzem semàntic basat en un relacional, res més lluny de la realitat, *Openlink Software* ha desenvolupat un SGBD híbrid, relacional i semàntic alhora, on es prenen decisions de disseny orientades a obtenir un bon rendiment en ambdues facetes del producte.

4.1.1 Model físic de Virtuoso

En aquest punt es relacionen les característiques de més baix nivell de Virtuoso, que fan referència a com s'emmagatzemen les dades, com són indexades i algunes consideracions pel seu ús.

- **Emmagatzemament per files (*rowwise*)**

Ubica les files de forma seqüencial a l'estil d'un fitxer seqüencial. D'aquesta forma els valors de les distintes columnes, d'una mateixa fila, apareixen un darrere l'altre. Així s'obté un seguit de tipus de dades dispars, amb rangs de valors diferents. Aquesta ha estat l'única forma d'emmagatzemament fins la versió 6 de Virtuoso.

Aquest emmagatzemament és adequat per l'accés aleatori a una fila, quan les columnes són actualitzades freqüentment i especialment si s'actualitza una fila per sentència.

- **Emmagatzemament per columnes (*columnwise*)**

En aquesta forma d'emmagatzemament, cada columna d'una taula o índex, es guarda de forma contigua. És a dir, els valors d'una certa columna de files consecutives estan físicament adjacents. Així els valors adjacents són del mateix tipus i a menut formen una seqüència ascendent. Açò permet l'ús de tècniques de compressió més potents. Aquest sistema s'implantarà en la futura versió 7 de Virtuoso *(docs.openlinksw.com[3], 2009, apartat 5.1.3)*, on l'emmagatzemament per files seguirà sent l'opció predeterminada.

En l'emmagatzemament per columnes, quan hi ha consultes d'una part de les columnes d'una taula, només cal accedir a aquestes al disc, estalviant operacions d'entrada i eixida, i espai en memòria. Les columnes no consultades no són accedides en cap moment.

Dona un bon rendiment en l'accés seqüencial i també en l'accés aleatori a moltes files resultat d'una unió. L'alta capacitat de compressió obtinguda fa que amb taules grans es puga tenir una gran quantitat de dades en memòria estalviant accessos a disc. Les insercions i esborrats massius també són més eficients.

- **Índexs**

Virtuoso usa índexs clúster per taules, tant si estan organitzades per files com per columnes. És a dir, la taula és el propi índex pel que fa a la clau primària. Els índexs secundaris, tenen les seus pròpies columnes i apunten a les claus de l'índex primari (clúster).

Tots els índexs es poden representar per files (rowwise) o per columnes (columnwise). En ambdós casos es representen amb arbres B. En el cas

per files el que hi ha a les fulles són valors de les columnes. En canvi, en els organitzats per columnes, a les fulles hi ha un vector de nombres de pàgines que conté els valors comprimits de les columnes d'uns milers de files (es comprimeixen per aprofitar que poden estar en seqüència ascendent i que són valors homogenis).

- **Execució vectorial**

L'execució vectorial apareixerà en la versió 7 de Virtuoso *(docs.openlinksw.com[3], 2009, apartat 5.1.4)*. Els SGBD que permeten emmagatzemament per columnes solen tenir un motor d'execució vectorial perquè la latència d'accés a tuples és major ja que les diferents columnes, d'una fila, no estan contigües.

L'execució vectorial consisteix en executar consultes o procediments emmagatzemats sobre múltiples conjunts de dades de forma simultània. Per exemple a l'executar una *Join*, amb execució vectorial, cada pas consecutiu del *Join* tracta múltiples entrades de dades. Si cada pas s'executa amb desenes de milers de valors alhora s'obtenen dos beneficis, d'una banda s'elimina quasi per complet la sobrecarrega d'interpretació del codi SQL i per altra banda es treu profit en accedir files properes (*locality*).

- **Paral·lelització automàtica de consultes**

Quan una consulta no modifica dades, s'executa amb aïllament *Read Commited* i conté agregació o *ORDER BY*, pot ser paral·lelitzada de forma automàtica i transparent a l'usuari. La paral·lelització trosseja el bucle més extern de la consulta en parts aproximadament iguals, les avalua amb distints fils d'execució i finalment aplega els resultats.

Taula 9. Límits físics de Virtuoso

Límit	Valor
Grandària base de dades	32 TB / 32 TB per temporal
Grandària fitxer de base de dades	Depèn del límit de fitxer del sistema operatiu
Nombre de fitxers per base de dades	Il·limitat
Grandària de pàgina	8 KB

4.1.2 Model lògic de Virtuoso

El model lògic d'una base de dades està format pels elements que l'usuari pot usar per representar un esquema conceptual, extret de la realitat, en forma de dades a dins d'aquesta. Virtuoso ofereix la flexibilitat de l'accés relacional combinat amb l'herència, creació de tipus de dades en temps d'execució, *late binding* (enllaçat tardà) i accés basat en identitats *(docs.openlinksw.com[3], 2009)*.

- **Taula**

És una entitat amb nom únic composta per zero o més columnes, una clau primària, zero o més índexs, una supertaula de la que pot heretar, un ID d'objecte opcional i restriccions de taula (per exemple *CHECK*).

- **Columna**

Sempre forma part d'una taula i té un nom únic a dins d'aquesta. Pot aparèixer a diverses taules gràcies a l'herència però només es defineix a un lloc.

- **Clau / Índex**

Les claus o índexs serveixen per obtenir les direccions físiques, dins el fitxers de la base de dades, d'aquelles files que tenen certs valors en determinades columnes. Són d'aplicació a una taula, o varies per herència, dins la qual tenen un nom únic. Entre les seues característiques hi ha: un identificador dins la base de dades, les columnes de la taula referenciada, si és primària i/o única, i si és la clau d'un objecte.

- **Subtaula**

Donat que Virtuoso també està orientat a objectes, es poden definir subtaules que hereten totes les característiques d'una taula. Opcionalment pot definir columnes addicionals però no redefinir la clau primària.

- **ID d'objecte**

La definició de la clau primària no és obligatòria. Si no se'n defineix cap es crea automàticament l'ID d'objecte que és una columna autoincremental , invisible però que pot ser referenciat.

4.2. Funcionalitats generals

Taula 10. Funcionalitats generals de Virtuoso

Llistat de funcionalitats de Virtuoso
Creació i emmagatzemament de documents XML. Permet usar XML per accés transparent a dades estructurades o no estructurades, ubicades en Virtuoso, en la web o en altres SGBD. Es poden crear documents XML dinàmicament a partir de consultes SQL. Suporta els llenguatges de consulta XML XPATH 2.0 i XQuery 1.0 (de forma bàsica.)
Integra un servidor web per pàgines estàtiques o dinàmiques amb VSP (Virtuoso Server Pages).
Allotjament i creació de serveis web (tipus SOAP, Simple Object Access Protocol) a partir de procediments emmagatzemats. Crea automàticament els fitxers WSDL (Web Service Definition Language) i com a servidor UDDI (Universal Description Discovery and Integration) els ofereix a la xarxa.
Servidor WebDAV(Web Distributed Authoring and Versioning). Li permet fer de magatzem de contingut web per dades eBussines, text, gràfics i multimèdia.
Replicació i sincronització de contingut SQL i web a servidors Virtuoso distribuïts.
Accés transparent a dades heterogènies (VDB, Virtual Database Engine) ubicades en els SGBD més coneguts del mercat.

Llistat de funcionalitats de Virtuoso
Serveis de lliurament de correu (SMTP, POP3 i IMAP) que permeten el desenvolupament de solucions de correu allotjades en base de dades.
NNTP per suportar fors de notícies.
Runtime Hosting Virtuoso es pot estendre per allotjar diferents entorns d'execució (Microsoft .NET, Mono ECMA-CLI i Java Virtual Machine JVM, amb açò es poden crear mòduls persistents, funcions SQL i tipus de dades definits pels usuaris amb els corresponents llenguatges i amb altres llenguatges tradicionals com C/C++. Els objectes creats poden ser accedits des d'SQL i la lògica d'aplicació pot ser exposada en forma de serveis web.
Cerca per text lliure permet indexar columnes amb tipus caràcter que continguen text o XML, de forma eficient i compacta.
Ús de Tuxedo amb Virtuoso que permet enllaçar amb Tuxedo per integrar Virtuoso en entorns client/servidor distribuïts i heterogenis (independència de maquinari, sistema operatiu, xarxa i entorn de base de dades).

4.3. Funcionalitats detallades

Virtuoso, en ser un SGDB híbrid i amb vocació integradora d'altres SGBD, subministra un ventall de possibilitats prou extens. Donat que la finalitat del present treball és l'estudi dels magatzems semàntics, el present apartat es centra en aquelles característiques relacionades amb el magatzem RDF de Virtuoso. Algunes d'aquestes característiques s'han tractat en l'apartat 3.3 de forma breu i seran ampliades, d'altres s'estudien per primera vegada.

4.3.1. Representació de dades

En aquest apartat s'estudia la forma en què Virtuoso representa els triples RDF. En el punt 3.3.4 ja s'han explicat els tipus IRI_ID i RDF_BOX, i la taula RDF_QUAD (que ara s'amplia i en estudiarem de noves).

- **Taula RDF_QUAD**

```
create table DB.DBA.RDF_QUAD (
G IRI_ID,
S IRI_ID,
P IRI_ID,
O any,
primary key (G,S,P,O) );
create bitmap index RDF_QUAD_OGPS on DB.DBA.RDF_QUAD (O, G, P, S);
```

Cada triple (quad) es representa com una fila d'aquesta taula (Graf, Subjecte, Predicat i Objecte). Les columnes de tipus IRI_ID referencien la taula RDF_IRI. La columna ***O*** és de tipus *ANY*, si ***O*** és un nombre, una data o una cadena de menys de 20 caràcters es guarda en binari, si és una cadena llarga o un literal RDF es guarden com *RDF_BOX*

- **Taula RDF_PREFIX i RDF_IRI**

```
create table DB.DBA.RDF_PREFIX (
RP_NAME varchar primary key,
RP_ID int not null unique );
```

```
create table DB.DBA.RDF_IRI (
RI_NAME varchar primary key,
RI_ID IRI_ID not null unique );
```

Aquestes dues taules mapeigen els ID interns dels IRI (URI) amb la cadena de caràcters IRI (URI) habitual, RDF_PREFIX conté els *namespace prefix* i RDF_IRI els *local name*. Hi ha una memòria intermèdia que conté els IRI usats recentment per reduir l'accés a aquest taula.

- **Taula RDF_OBJ**

```
create table DB.DBA.RDF_OBJ (
RO_ID integer primary key,
RO_VAL varchar,
RO_LONG long varchar,
RO_DIGEST any
)
create index RO_VAL on DB.DBA.RDF_OBJ (RO_VAL)
create index RO_DIGEST on DB.DBA.RDF_OBJ (RO_DIGEST);
```

Quan un valor *O* de la taula RDF_QUAD excedeix un límit de longitud o és un text lliure indexat es guarda en aquesta taula. Depenent de la seva longitud es guarda a la columna *varchar* (RO_VAL) o *long varchar* (RO_LONG). La clau primària *RO_ID* es referenciada des de la columna *O* de la taula RDF_QUAD. Quan *RO_LONG* conté un valor molt gran *RO_VAL* conté el seu *checksum* per accelerar la cerca de valors idèntics..

- **Taula RDF_DATATYPE**

```
create table DB.DBA.RDF_DATATYPE (
RDT_IID IRI_ID not null primary key,
RDT_TWOBYTE integer not null unique,
RDT_QNAME varchar );
```

Quan la columna *O* de RDF_QUAD conté dades cadena d'un cert tipus d'XML Schema, aquest tipus es representa amb 2 octets dins *O*, que referencien a aquesta taula que els mapeja a un espai d'IRI més ample.

- **Taula RDF_LANGUAGE**

Quan la columna *O* de RDF_QUAD és una cadena en un determinat idioma, aquest es representa amb 2 octets dins O, que referencien a aquesta taula que mapeja aquest valor en l'idioma real com "es-Us", "es-ES", etc.

```
create table DB.DBA.RDF_LANGUAGE (
RL_ID varchar not null primary key,
RL_TWOBYTE integer not null unique );
```

4.3.2. SPARQL

Virtuoso integra el llenguatge SPARQL, que és el més usat en l'accés a magatzems semàntics i que en la versió 1.1 afegeix les sentències per actualitzacions (SPARUL). Al punt 2.7 d'aquest document hi ha una breu introducció a aquest llenguatge, al punt 3.3.1 hi ha una breu ressenya de les sues característiques en la seua implementació dins Virtuoso, que ara s'amplia, finalment al punt 4.4 hi ha alguns exemples del seu ús.

- ***SPARQL Endpoint***

Virtuoso proveeix un *SPARQL Endpoint*, que dona accés als serveis web SPARQL a través del protocol REST. Aquest servei és accessible a través de l'URL *http://servidor.com:8890/sparql* i permet peticions GET i POST. Si la petició és curta s'usa GET i si és llarga (major de 1900 bytes) s'usa POST. Si s'usa l'anterior URL sense paràmetres apareix una pàgina interactiva per introduir codi SPARQL directament. Un paràmetre obligatori de les peticions és *query* que conté el propi codi SPARQL i altre és *named_graf* per indicar el graf que es consulta. Estan suportats tots els formats de resposta MIME definits pel protocol SPARQL del W3C i s'afegeixen alguns com *application/rdf+xml, text/rdf+n3*,...

Exemple de consulta SPARQL usant *curl*:

```
curl -F "query=SELECT DISTINCT ?p FROM
<http://demo.openlinksw.com/DAV/home/demo/rdf_sink/> WHERE {?s ?p ?o}"
http://demo.openlinksw.com/sparql
```

- **Ús conjunt d'SPARQL i SQL**

Virtuoso estén SQL 92 amb consultes i subconsultes SPARQL, açò permet usar SPARQL per qualsevol medi que permeta l'ús d'SQL (qualsevol API suportada, ODBC, JDBC,...), per fer-ho la consulta s'ha de precedir de la paraula *SPARQL* , per exemple:

```
SQL>SPARQL SELECT DISTINCT ?p WHERE { graph ?g { ?s ?p ?o } };
...
SQL>SELECT distinct subseq ("p", strchr ("p", '#')) as fragment
FROM (SPARQL SELECT DISTINCT ?p WHERE { graph ?g { ?s ?p ?o } } ) as
all_predicates
      WHERE "p" like '%#%';
```

De forma complementaria SPARQL pot cridar funcions integrades a SQL de Virtuoso amb el prefix `bif:` i a procediments desenvolupats amb `sql:`, per exemple:.

```
SQL>SPARQL
SELECT DISTINCT ?cityUri ?cityName (sql:BEST_LANGMATCH (?cityName, 'en, en-
gb;q=0.8, fr;q=0.7, *;q=0.1', '')) as ?bestCityName
WHERE
  {
    ?cityUri ?predicate ?value.
    ?cityUri a <http://dbpedia.org/ontology/City>.
    ?value bif:contains "London".
    OPTIONAL
      {
        ?cityUri rdfs:label ?cityName
      }
  };
```

4.3.3. Extensions

Les extensions són funcionalitats que aporten un valor afegit a un producte, que complementen les funcionalitats bàsiques o implementen nous estàndards, les que inclou Virtuoso són:

- Cerca de text complet (vist al punt 3.3.4)
- SPARUL

Es suporta des de la versió 5.0.

- Business Intelligence (SPARQL_BI)

Virtuoso estén SPARQL amb expressions pels resultats, subconsultes, agregacions i agrupacions. Açò facilita la traducció d'SQL a SPARQL, amb l'objectiu que SPARQL complisca les necessitats de BI.

4.3.4. Vistes Linked Data sobre fonts de dades relacionals

Virtuoso permet definir vistes semàntiques de dades ubicades en SGBD relacionals, açò permet que SPARQL les accedeixa de forma transparent com si foren dades semàntiques. Donat que la majoria de les dades susceptibles de ser usades en la web semàntica estan en SGBD relacionals aquesta funcionalitat representa un gran avantatge.

Virtuoso proveu el llenguatge declaratiu *Meta Schema Mapping Language* que mapeja dades SQL a Ontologies RDF. Aquest llenguatge està integrat al traductor d'SPARQL a SQL. Hi ha altres llenguatges per fer açò mateix, dels que Virtuoso es diferencia en *(docs.openlinksw.com[3], 2009, apartat 16.5.1)*:

- Integració amb el magatzem semàntic en permetre que una mateixa consulta accedisca a dades relacionals remotes i triples RDF locals.
- L'optimitzador d'SPARQL i SQL és únic i a més té en compte la ubicació de les dades. Açò és clau per consultes mixtes i en entorns distribuïts.
- No s'imposa un límit a SPARQL, quan fa consultes de grafs o predicats no especificats, per evitar la baixa eficiència.
- Suport complet del model relacional.

Virtuoso disposa també d'un assistent, a la seua interfície web Conductor (http://servidorVirtuoso:8890/conductor), que permet generar i publicar vistes RDF a partir de dades relacionals. Es pot fer de forma automàtica o amb certes opcions que l'usuari pot personalitzar.

- **Suport R2RML a Virtuoso**

R2MRL és esborrany de treball del W3C per definir un llenguatge estàndard per expressar el mapejat de dades relacions a dades RDF. Els mapejats R2RML són grafs RDF escrits en Turtle. Virtuoso va desenvolupar *Meta Schema Mapping Language* abans que el W3C definís aquest esborrany. Aleshores el suport que Virtuoso dona a aquest estàndard consisteix en un simple traductor de R2RML a *Meta Schema Mapping Language*.

4.3.5. Programari Sponger

Virtuoso Sponger és un programari intermediari que genera dades RDF des de diverses fonts de dades Web, no estructurades o semiestructurades. Sponger sorgeix per facilitar l'avanç de la implantació de la web semàntica. Admet un gran ventall de formats de representació de dades i formats de serialització. Sponger és un servei proxy HTTP i és accessible via SOAP o REST.

Els productes per extraure i generar dades RDF són anomenats en anglès *RDFizers*.

Sponger es compon de cartutxos (Cartridges) de dos tipus:

- *Extractor*, orientat a extraure i transformar dades.
- *Meta Cartridge*, proveeix cerques i *joins* d'altres espais RDF i API Web 2.0.

Els cartutxos són molt configurables i es poden desenvolupar en qualsevol llenguatge admès per Virtuoso, així es poden generar dades RDF des de fonts no incloses en la col·lecció de cartutxos que Virtuoso subministra. Hi ha cartutxos per desenes de formats i orígens de dades com CSV, iCalendar, Amazon, Delicious, Facebook, etc. Veure *(docs.openlinksw.com[3], 2009 apartat 16.9.9)*

El procés pel qual un client RDF obté dades via Sponger és el següent:

- Un client RDF fa una petició de dades en format RDF, si es retorna RDF acaba el procés.
- Si no, Sponger obté les dades amb un procés *Pipeline Extraction Metadata* (usant un *Extractor*).
- Les dades obtingudes són transformades a RDF amb un procés *Mapping Pipeline* (mitjançant un casament i mapejat amb l'ontologia) obtenint les instàncies de dades RDF.
- Finalment les dades RDF són lliurades al client.

Les dades obtingudes segueixen les regles d'expiració dels navegadors web, les primeres dades carregades deuen tenir una capçalera d'expiració, en cas contrari Sponger aplicarà el seu propi sistema d'invalidació.

4.3.6. Navegació per facetes

Amb aquesta funcionalitat de Virtuoso un usuari pot combinar cerques per text amb criteris RDF estructurats (classes, valors de propietats,...). Es pot fer amb SPARQL, cridant serveis web o interactuant amb una pàgina web. Hi ha una implementació d'aquesta funcionalitat a http://lod.openlinksw.com/, és una combinació de DBpedia, MusicBrainz, Freebase, UniProt, Bio2RDF i altres.

La navegació per facetes requereix consultes que impliquen l'agregació de milions de subjectes, sobretot en els primers passos de la cerca. Per fer possible que aquest treball es faça en un temps acceptable per interactuar amb persones cal d'una banda un optimitzador capaç de produir un ordre perfecte per a les unions i d'altra un motor d'execució de la consulta que puga retornar resultats parcials quan s'excedeix un temps màxim d'execució.

El servei web Virtuoso Facets rep una descripció de la consulta a realitzar en format XML i retorna el resultats en el mateix format. El client pot fer ús d'una pàgina d'estil XSLT per presentar les dades a l'usuari final, es fa així perquè és més fàcil processar el format XML que el codi SPARQL.

A l'annex B es mostren els passos per fer una cerca de les ubicacions relacionades amb les guerres de Napoleó. En cada pas es mostra la petició XML i l'acció equivalent amb la interfície web.

4.3.7. Linked Data

El concepte Linked Data ha estat explicat a l'apartat 3.4 de forma genèrica. El present apartat es centra en afegir alguns conceptes i característiques específics de la seua implementació a Virtuoso.

- ***IRI* Dereferencing (Eliminació de referències a IRI)**

Hi ha casos en què és interessant descarregar recursos RDF d'un IRI concret. Aquest procés s'anomena *IRI Dereferencing* perquè l'IRI del graf a emmagatzemar és normalment igual a l'IRI des d'on es descarrega. Virtuoso estén SPARQL per tal de poder-ho fer, durant una consulta el recurs és descarregat, analitzat i els triples resultants són emmagatzemats. L'extensió consisteix en clàusules *define* : `define get:parameter` i els paràmetres indiquen com ha de ser l'eliminació de referències `(soft, replacing, uri, method, refresh , proxy, ...)`

Hi ha dos casos d'ús d'aquesta funcionalitat, en el més senzill el *from* de la consulta enumera els grafs a processar, com que alguns no els trobarà a la taula local de triples, la consulta fa primer l'eliminació de referències i la resta continua com és normal. L'altre cas d'ús, més complex, és quan la consulta es processa a dins d'un bucle, cada execució produeix un resultat parcial amb IRIs que pot ser no existeixen en la taula local de triples i fa la corresponent eliminació de referències, el processador SPARQL compara cada resultat parcial amb el de l'anterior iteració fins que coincideixen. Com últim pas el processador SPARQL construeix el resultat final.

- **Reescriptura d'URL**

La reescriptura d'URL és la modificació d'una URL abans de processar-la en un servidor web. Les raons són diverses: canviar l'URL d'un recurs sense afectar als favorits dels agents d'usuari (p.ex. navegadors web), compactació d'URL, creació d'URL adequades pel motors de cerca perquè les URL amb paràmetres no són indexables fàcilment.

Per l'entorn Linked Data la reescriptura resol problemes com el de l'identificador de fragment "#" que no es processat pels servidors webs produint que diferents URL semblen la mateixa. La solució consisteix en preprocessar els URI amb expressions regulars o substitucions del tipus *sprintf*.

Virtuoso disposa de diverses eines per solucionar aquests problemes con són: un reescriptor d'URL basat en regles que es pot habilitar per URL que casen amb certs patrons, definició de directoris i dominis virtuals i l'ús d'URL curtes que codifiquen URL llargues amb paràmetres.

- **Vistes per facetes sobre Linked Data a gran escala**

La transició de la web com a repositori de documents a una base de dades universal i ubiqua requereix replantejar l'escalabilitat per suportar la interacció dels usuaris. Per exemple l'optimitzador ha de conèixer la cardinalitat per estimar el cost d'una consulta, en comptes de calcular la cardinalitat real el que fa és estimar-la fent un mostreig dels índexs. L'optimitzador està basat en costos però adaptat a jerarquies grans i combinat amb patrons de text. Un optimitzador SQL seria incapaç de funcionar amb grans jerarquies com les que hi ha a l'entorn Linked Data *(docs.openlinksw.com[3], 2009, apartat 16.12.5)*.

- **Consultes que no retornen res**

En alguns casos, com al navegar per dades socials, s'obtenen resultats buits degut a que hi ha molts nodes buits. Existeix l'opció `DEFINE input:inference "b3sifp"` que declara que el *sha1sum* de `foaf:name` i `foaf:mbox` s'ha de tractar com

a IFP (Inverse functional property), és a dir si diversos individus tenen el mateix valor es poden enllaçar, evitant resultats buits.

- **Classificació d'entitats**

És habitual que l'usuari final vulga que els resultats de les cerques de text estiguen ordenades per la seua rellevància. La classificació d'entitats consisteix en expressar numèricament la rellevància d'un URI dins un graf. L'expressió `( <LONG::IRI_RANK> (?s) )` referida a un subjecte dona aquesta rellevància amb un valor numèric, pot ser usada en `SELECT` per ser mostrada i en `ORDER BY` per ordenar els resultats.

- **Límit de temps en l'avaluació de consultes**

En el model Linked Data és conegut que la carrega de treball és impredictible i que normalment hi ha molt creadors de consultes que encara són inexperts. El que Virtuoso tracta de fer en aquests casos és donar alguna resposta fins i tot quan s'ha excedit el límit de temps.

S'ha de tenir en compte que les consultes normalment impliquen agregació, ordenació i potser transitivitat. Per tant arribar a tenir un resultat parcial tampoc és trivial. Virtuoso implementa mecanismes per controlar el límit de temps en clústers on cada node deu arribar al límit al mateix temps aproximadament i enviar el resultats per formar el resultat conjunt.

- **voiD (Vocabulary of Interliked Data)**

Aquest vocabulari es desitjable en l'entorn Linked Data. Consisteix en una subclasse usada per emmagatzemar dades que expressen la relació entre distints conjunts de dades dins Linked Data. Virtuoso genera de forma automàtica descripcions *voiD*, pels conjunts de dades que allotja.

4.3.8. Raonament i regles d'inferència

A l'apartat 3.3.2 es feu una introducció teòrica al raonament i es donà una breu referència respecte al raonament en Virtuoso. En el present apartat es desenvolupen algunes característiques del raonament implementades per Virtuoso.

Virtuoso pot inferir triples que no estan físicament emmagatzemats a partir d'un o més grafs que contiguen triples RDF. Les restriccions OWL i RDF Schema són importades, d'aquests grafs, i agrupades en bases de regles, que poden ser referenciades per consultes SPARQL. Aquestes consultes executades, amb aquestes bases de regles, treballen com si els triples inferits existiren realment al graf accedit.

Virtuoso suporta raonament per `rdfs:subClassOf`, `rdfs:subPropertyOf`, `owl:sameAs` (parcialment), `owl:equivalentClass`, `owl:equivalentProperty`, `owl:InverseFunctinalProperty`, `owl:inverseOf`, `owl:SymmetricalProperty` i `owl:TransitivePropery`.

El raonament i la inferència no són aplicables als triples virtuals accedits per mig de vistes Linked Data sobre dades relacionals.

Els esquemes RDF i OWL poden ser introduïts en un magatzem de triples, Virtuoso permet afegir-hi assercions per modificar el context d'inferència en forma de

conjunts de regles. S'ha de tenir en compte que si es modifica un conjunt de regles aquesta modificació no afecta a les consultes compilades prèviament.

La implementació que Virtuoso fa del raonament, no emmagatzema físicament els triples inferits, el que fa és afegir-los al resultat en temps d'execució de les consultes. Aquesta mena de raonament s'anomena *backward chaining* (veure apartat 3.3.2).

En Virtuoso la inferència s'habilita dins una consulta amb la clàusula `input:inference "conjunt_regles"` que indica al compilador que ha d'usar les regles del conjunt indicat.

4.3.9. Dades RDF i les geometries

En Virtuoso les geometries es poden expressar com el valor d'un objecte en un triple RDF (a partir de la versió 6). En aquest cas no es fa ús d'un objecte geomètric nu si no que s'usa un literal RDF del tipus `virtrdf:Geometry`. Aquest literal s'indexa automàticament amb un índex d'arbre R, que conté totes les geometries de tots els quads en qualsevol graf i predicat indistintament.

La inserció de geometries es fa normalment amb la funció `ttlp`, mentre que l'esborrat es fa amb operacions SPARUL. Molts conjunts de dades usen les propietats `geo:lat i geo:long` per indicar posicions. Virtuoso incorpora una funció per convertir aquestes propietats en geometries (`DB.DBA.RDF_GEO_FILL()`), aquesta funció recorre el graf i per cada subjecte amb un parell `geo:lat i geo:long` crea un punt geomètric per cada parell (lat/long) distint. Aquestes geometries s'afegeixen als subjectes com el valor de la propietat `geo:geometry`, en el mateix graf d'origen.

4.3.10. Proveïdors d'accés a dades RDF

Els proveïdors d'accés llistats a continuació són implementacions de Virtuoso que permeten que aplicacions desenvolupades amb distints marcs de treball externs puguen accedir a dades emmagatzemades per Virtuoso directament.

- **Virtuoso Jena Provider**

Jena és un marc de treball de codi obert Java per la web semàntica, amb una API per extreure i escriure dades en magatzems RDF.

Virtuoso Jena Provider és un *Native Graph Model Storage Provider for Jena Framework* completament funcional que habilita les aplicacions Jena per consultar els triples emmagatzemats per Virtuoso directament. Aquest proveïdor ha estat provat amb Jena 2.5.5.

- **Virtuoso Sesame Provider**

Sesame és un marc de treball de codi obert Java per emmagatzemar, consultar i raonar amb RDF i RDF Schema. Es pot usar com base de dades autònoma o com llibreria Java.

Virtuoso Sesame Provider és un *Native Graph Model Storage Provider for Sesame Framework* plenament funcional que permet usar Java per consultar, modificar i raonar amb triples allotjats per Virtuoso. L'API *Sesame Repository* ofereix

un punt d'accés per connectar amb l'SGBD Virtuoso a l'estil Java. Aquest proveïdor ha estat provat amb Sesame 2.1.2.

- **Virtuoso Redland Provider**

Redland és un conjunt de llibreries lliures en C, modulars i basades en objectes, que permeten manipular grafs RDF, triples, URI i literals.

Virtuoso Redland Provider és una implementació de la interfície de consulta, i de l'API d'emmagatzemament de Redland. Aquest proveïdor permet les consultes des del llenguatge Redland Rasqal. Aquest proveïdor dona accés a Virtuoso usant el controlador ODBC de Virtuoso.

4.4. Operacions de gestió de dades

En aquest apartat es revisen les diferents opcions que Virtuoso ofereix per afegir, consultar i modificar dades semàntiques.

4.4.1. Mètodes d'inserció de dades RDF

Tot i que l'operació més habitual serà la consulta de dades, és obvi que per promoure la implantació dels magatzems semàntics cal afavorir la carrega inicial de dades. Per aquesta raó Virtuoso proveeix un ampli ventall de mètodes d'inserció que s'estudien en aquest apartat.

- **SPARQL insert via *isql* i SPARQL Endpoint**

Isql, acrònim d'*Interactive SQL*, és un comandament que ens introdueix en l'entorn interactiu d'SQL de Virtuoso, en aquest entorn de línia de comandament es poden introduir sentències SPARQL precedint-les de la paraula *SPARQL*.

L'*SPARQL Endpoint* accessible a *http://servidor.com:8890/sparql* permet també la introducció de sentències SPARQL de forma interactiva, en aquest cas al ser un editor SPARQL únicament no cal usar la paraula *SPARQL* prèvia a les sentències.

- **Funcions de l'API de Virtuoso**

`DB.DBA.TTLP(origen, base_IRI, IRI_graf_desti, flags)`:

importa triples des d'un origen en format Turtle al graf indicat.

```
SQL> DB.DBA.TTLP (file_to_string_output ('.\tmp\data.ttl'), '', 'http://my_graph', 0);
```

`DB.DBA.TTLP_MT(origen, base_IRI, IRI_graf_desti, flags)`:

igual que l'anterior funció però usa múltiples fils d'execució i entrada/eixida paral·lela si estan disponibles. No usa transaccions, cal fer *checkpoint* per fer persistents els canvis.

`DB.DBA.RDF_LOAD_RDFXML(origen, base_IRI, IRI_graf_desti)`:

importa triples des d'un origen en format RDF/XML.

```
SQL>DB.DBA.RDF_LOAD_RDFXML (http_get ('http://www.w3.org/People/Berners-Lee/card#i'),
'no', 'http://www.w3.org/people#');
```

`DB.DBA.RDF_LOAD_RDFXML_MT(origen, base_IRI, IRI_graf_desti)`: igual que la funció anterior usant múltiples fils d'execució, per importar recursos grans si no és important la integritat transaccional.
`DB.DBA.RDF_TRIPLES_TO_TTL(vector_triples, stream_eixida_turtle)` i `DB.DBA.RDF_TRIPLES_TO_RDF_XML_TEXT(vector_triples, flags, stream_eixida_rdf-xml)`: exporten un conjunt de triples i els serialitzen a TURTLE i RDF/XML respectivament.
`DB.DBA.SPARQL_EVAL(consulta_SPARQL, IRI_graf, max_files)`: executa una consulta SPARQL , retorna un conjunt de valors. Si `IRI_graf` no és nul s'usa en comptes de l'especificat en la consulta
`DB.DBA.SPARQL_EVAL_TO_ARRAY(consulta_SPARQL, IRI_graf, max_files_retornades)`: igual que l'anterior però retorna un vector de conjunts de valors.
`DB.DBA.SPARQL_REXEC(URI_de_servici, consulta_SPARQL, IRI_graf, graf_anomenat, peticio, max_files, diccionari_bnode)`: igual que DB.DBA.SPARQL_EVAL però s'executa en un URI remot.

- **API REST de l'SPARQL Endpoint**

Amb el mètode POST es poden llançar sentències SPARQL que continguen INSERT, UPDATE i DELETE, igual que amb PUT, però aquest darrer és més útil i expressiu en identificar en la seua sintaxi, l'URI de l'entitat objecte de la petició (veure exemple) . Amb el mètode GET es poden consultar els triples emmagatzemats.

En els exemples següents es fa ús de *curl* que permet llançar peticions HTTP des de la línia de comandament.

Exemple POST

```
curl -i -d "INSERT {<http://demo.openlinksw.com/DAV/home/demo_about.rdf>
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://rdfs.org/sioc/ns#User>}" -u "demo:demo"
-H "Content-Type: application/sparql-query" http://localhost:8890/DAV/xx/yy
```

Exemple PUT (La diferència amb POST resideix en què amb PUT s'especifica un fitxer *myfoaf.rdf* que conté la sentència a enviar al servidor)

```
curl -T myfoaf.rdf
http://demo.openlinksw.com/DAV/home/demo/rdf_sink/myfoaf.rdf -u
demo:demo
```

Exemple GET

```
curl -F "query=SELECT DISTINCT ?p FROM
<http://demo.openlinksw.com/DAV/home/demo/rdf_sink/> WHERE {?s ?p ?o}"
http://demo.openlinksw.com/sparql
```

- **Usant la sentència LOAD d'SPARQL**

Es pot usar una de les tres sintaxis següents:

```
SPARQL INSERT INTO <..> { .... } [[FROM ...] { ... }]
```

```
SPARQL LOAD <x> [INTO <y>]
```

```
-- <ResourceURL> és l'IRI del graf que conté les dades:
SPARQL LOAD <ResourceURL>
```

Exemple de cadascuna:

```
SPARQL insert in graph <http://mygraph.com>
{
<http://myopenlink.net/dataspace/Kingsley#this>
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://rdfs.org/sioc/ns#User> .
<http://myopenlink.net/dataspace/Kingsley#this>
<http://rdfs.org/sioc/ns#id>
<Kingsley> .
<http://myopenlink.net/dataspace/Matt#this>
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://rdfs.org/sioc/ns#User> .
<http://myopenlink.net/dataspace/Matt#this>
<http://rdfs.org/sioc/ns#id>
<Matt> .
.};
```

```
SQL>SPARQL
load "http://www.example.com/DAV/tmp/listall.rq") into graph
<http://myNewGraph.com>;

El fitxer listall.rq pot contenir una sentència SPARQL del tipus:

SPARQL
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX sioc: <http://rdfs.org/sioc/ns#>
SELECT ?x ?p ?o
FROM <http://mygraph.com>
WHERE
{
?x rdf:type sioc:User .
?x ?p ?o.
?x sioc:id ?id .
FILTER REGEX(str(?id), "^King")
}
ORDER BY ?x
.....................................
```

```
SQL> SPARQL LOAD <http://www.w3.org/People/Berners-Lee/card#i>;
```

- **Usant WebDAV**

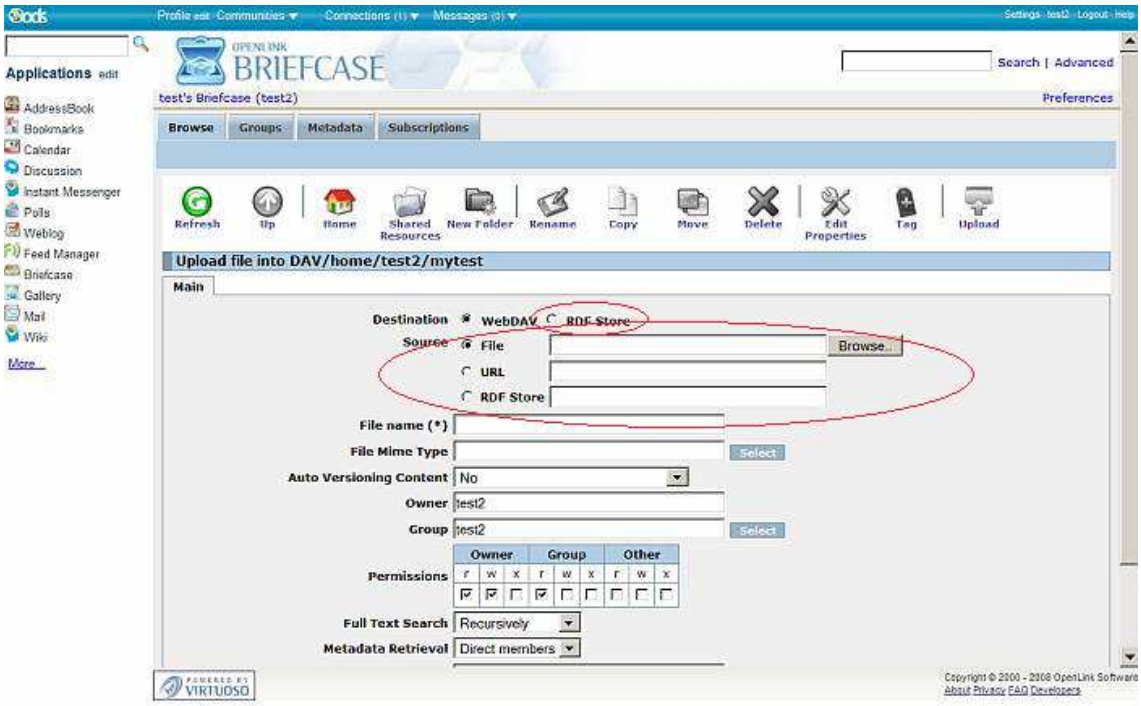
Amb WebDAV és possible crear una carpeta i bolcar-hi dades RDF al magatzem semàntic de Virtuoso. Es pot fer de dues formes, ambdues gràfiques. La primera és amb l'*ODS Briefcase*., L'altra via és usant *Conductor UI*.

A continuació es presenten amb els respectius exemples:

Exemple amb *ODS Briefcase*.

S'ha d'anar a (http://servidorVirtuoso:8890/ods), crear una carpeta WebDAV amb l'opció *New folder* i aleshores triar *Upload* (apareixerà la següent pàgina). On es pot triar la destinació de les dades (RDF Store), i l'origen pot ser un fitxer rdf, una URL o un altre *RDF Store*.

Figura 11 Inserció de dades via WebDAV amb ODS

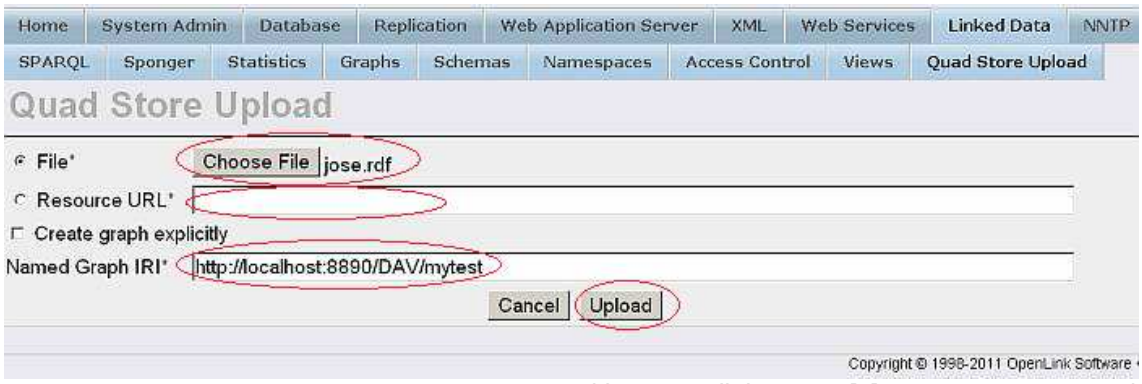


(docs.openlinksw.com[3], 2009, apartat 16.8.7)

Exemple amb *Conductor UI*.

S'ha d'anar a la direcció (http://servidorVirtuoso:8890/conductor), i s'ha de triar *Linked Data* i *Quad Store Upload*. En la següent imatge tenim la pàgina de carrega de dades via carpeta WebDAV. Es pot triar com origen un fitxer rdf o l'URL d'algun recurs, i com a destí l'IRI del graf desitjat. Finalment, amb el botó *Upload*, es fa la carrega de dades.

Figura 12 Inserció de dades via WebDAV amb *Conductor UI*



(docs.openlinksw.com[3], 2009, apartat 16.8.7)

- **Usant Virtuoso Crawler**

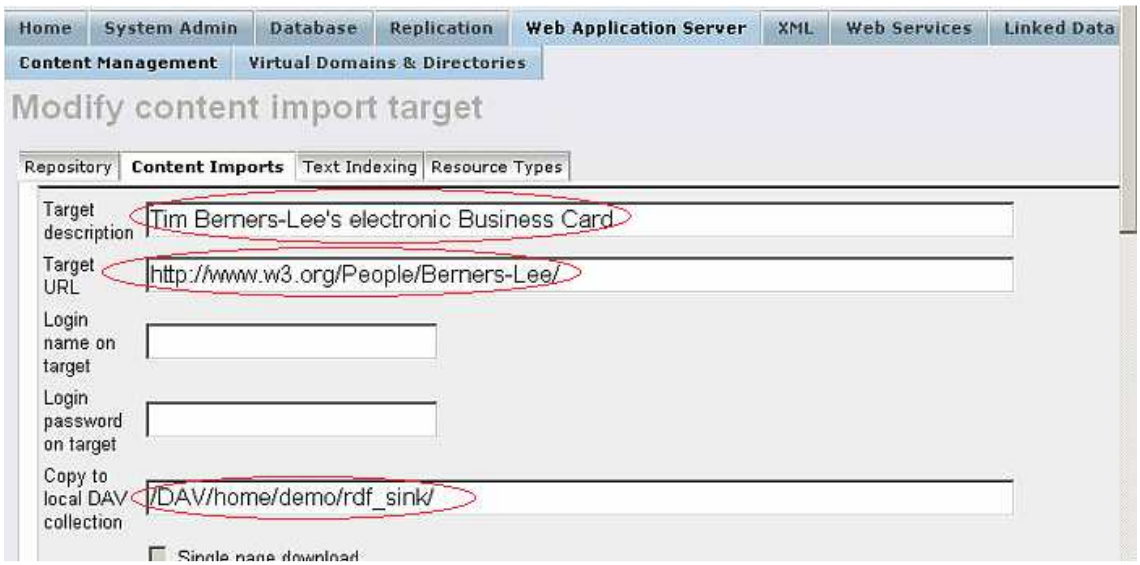
Virtuoso Crawler permet fer ús de les opcions Sponger, per importar orígens de dades no RDF, des de *Conductor UI*. En l'exemple següent es rastregen els elements

que hi ha per davall de l'URL indicada i amb l'ús dels cartutxos adients es generen dades RDF i s'emmagatzemen al magatzem RDF local.

Exemple Virtuoso Crawler (Conductor UI https://servidorVirtuoso:8890/conductor):

S'ha d'anar a *Web Application Server, Content Import* i triar *New Target* i emplenar el formulari com es mostra en la següent pàgina.

Figura 13 Inserció de dades amb Virtuoso Crawler



(docs.openlinksw.com[3], 2009, apartat 16.8.8)

Addicionalment seleccionar *Store metadata* i *Cartridges*. Per llançar l'operació es prem el botó *Import Queues* i es fa *Run* al *Target* recentment creat.

- **Consulta SPARQL i Sponger**

En aquest cas es mostra l'ús combinat d'una sentència SPARQL amb Sponger. Amb açò es poden importar recursos des de la xarxa usant el processador d'SPARQL. A continuació es mostra un exemple.

```
sparql
define get:uri "http://www.ivan-herman.net/foaf.html"
define get:soft "soft"
select * from <http://mygraph> where {?s ?p ?o}
```

Els pragmes Sponger `get:uri` i `get:soft` defineixen el recurs que s'ha d'importar i la forma en què es farà respectivament. En definir aquest pragmes, el que fa el processador d'SPARQL és invocar els cartutxos Sponger per convertir les dades no RDF del recurs accedit en dades RDF.

- **Usant l'API SIMILE RDF Bank**

Virtuoso implementa l'API *Semantic Bank*. Aquesta API permet que certes aplicacions client puguen bolcar dades al magatzem de Virtuoso. Un exemple de client és *PiggyBank* de *SIMILE* (http://simile.mit.edu/piggy-bank), és un plug-in per Firefox que permet a un usuari final encontrar dades RDF embegudes en pàgines web, les dades trobades poden dipositar-se, usant l'API *Semantic Bank*, a magatzem locals o remots anomenats Bancs Semàntics.

- **Usant el servei Proxy RDF d'Sponger**

En certs casos, com les aplicacions Ajax, no es poden fer peticions HTTP a servidors distints de l'original i en altres casos cal transformar el contingut accedit a un format RDF. El servei *Proxy Sponger* fa aquestes dues funcions alhora, de fet la segona funció és l'habitual d'Sponger. Aquest servei pren com argument una URL i pot retornar el contingut obtingut tal qual o Sponger el pot transformar prèviament. Es mostren a continuació uns exemples.

Sintaxi general:

```
http://host:port/[about|proxy]/[format]/URL_dest

about: per quan està instal·lat el paquet rdf_mappers (Sponger)
proxy: per quan no està instal·lat

format: [ id |html |data | rdf | http ]

Accés a un URL per veure el resultat en format HTML:
http://host:port/about/html/http/www.w3.org/People/Berners-Lee/card

Accés a un URL per veure el resultat en format RDF:
http://host:port/about/rdf/http://www.w3.org/People/Berners-Lee/card

Accés usant l'estil d'invocació proxy i format RDF:
http://host:port/proxy/rdf/http://www.w3.org/People/Berners-Lee/card
```

4.4.2. Replicació de grafs RDF

En aquest apartat es mostra com Virtuoso permet replicar grafs d'una instància a una altra, la funcionalitat que ho permet s'anomena *RDF Replication Feature*. Per aquesta funció s'estableix una instància principal, anomenada publicadora, i una o diverses instàncies de destí, anomenades subscriptores. La finalitat és que tots els canvis que es facen a determinats grafs en la instància publicadora es repliquen en cadascuna de les instàncies subscriptores.

Aquesta funcionalitat té com a finalitat proveure escalabilitat a l'hora de respondre a consultes massives, per exemple en exposar grafs a Internet, creant granges de servidors amb la finalitat de respondre a aquestes consultes.

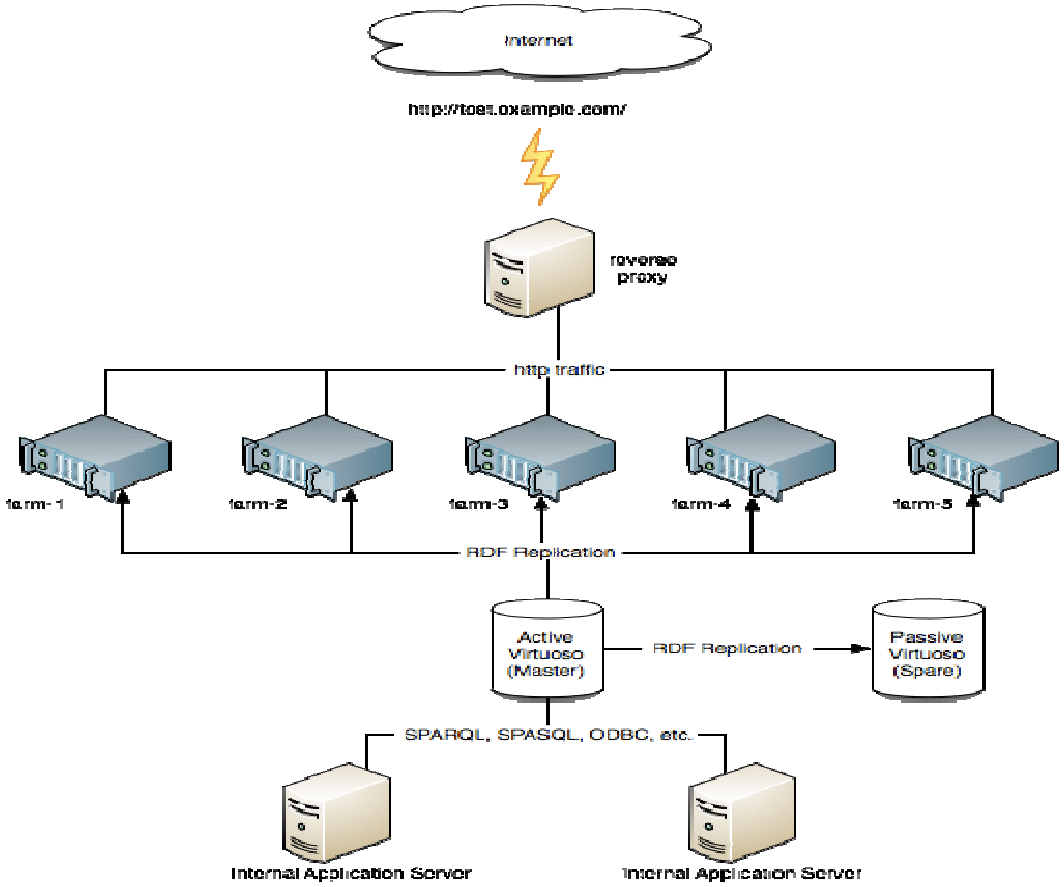
Cal diferenciar la replicació de grafs d'un clúster de servidors. En el primer cas les actualitzacions de dades, de certs grafs, es porten a terme en un sol servidor mentre que els altres només atenen consultes. En el cas d'un clúster, tots els servidors que el formen atenen, de forma indistinta, consultes i actualitzacions per a tots els grafs de la instància.

Una vegada definides la instància publicadora i les subscriptores, la dinàmica de funcionament consisteix en publicar un graf anomenat a la instància publicadora, després cal subscriure's a aquest graf des de les instàncies subscriptores per finalment comprovar que les modificacions al graf publicat apareixen al mateix graf en les instàncies subscriptores.

En la figura següent es mostra una configuració típica que usa la replicació de Virtuoso. La instància que té activada la publicació és l'*Active Virtuoso (Master)*, mentre que les restants instàncies són les subscriptores (*farm-1 a 5* i *Passive Virtuoso*

(Spare)). Els servidors *farm-1* a *5* formen un front-end per atendre peticions massives, procedents d'Internet, de forma transparent a través del proxy. El servidor *Passive Virtuoso (Spare)* és una copia del servidor actiu per substituir-lo en cas de caiguda.

Figura 14 Configuració típica de replicació de grafs amb Virtuoso



(docs.openlinksw.com[3], 2009, apartat 16.17.1)

- **Configuració de la instància de publicació**

```
$ isql IP-DEL-MASTER:1111

-- executar la següent sentència
-- per habilitar la publicació

rdf_repl_start();

-- afegir graf a la llista de replicació

rdf_repl_graph_ins('http://test.example.com');
```

- **Configuració de les instàncies de subscripció**

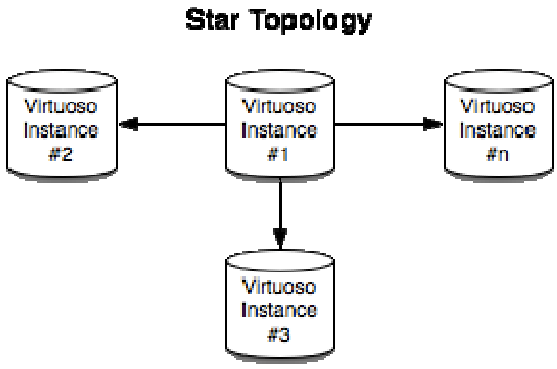
```
$ isql IP-REPLICADOR:1111

-- connectar al màster

repl_server ('MASTER', 'MASTER_DSN');

-- iniciar la subscripció
```

```
repl_subscribe ('MASTER', '__rdf_repl', 'dav', 'dav', 'dba',
'dba');

-- fer la replicació inicial

repl_sync_all ();

-- afegir la subscripció al planificador de tasques

DB.DBA.SUB_SCHEDULE ('MASTER', '__rdf_repl', 1);
```

- **Topologia de replicació en estrella**

En aquesta topologia hi ha una sola publicadora (#1) i diverses subscriptores (#2, #3,...,#n).

Figura 15 Topologia de replicació en estrella

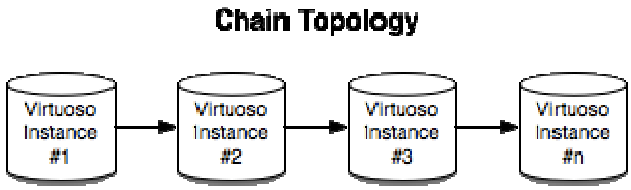


(docs.openlinksw.com[3], 2009, apartat 16.17.2)

- **Topologia de replicació en cadena**

En la topologia en cadena hi ha una publicadora original (#1), que té una sola subscriptora (#2). La subscriptora (#2) és alhora publicadora i també té una única subscriptora (#3) i així successivament fins a la subscriptora (#n) que no és publicadora, finalitzant així la cadena.

Figura 16 Topologia de replicació en cadena

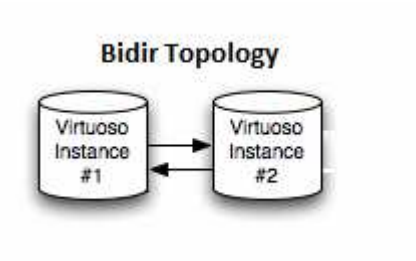


(docs.openlinksw.com[3], 2009, apartat 16.17.2)

- **Topologia de replicació bidireccional**

En aquesta topologia hi ha dues instàncies que són recíprocament publicadora i subscriptora l'una respecte l'altra.

Figura 17 Topologia de replicació bidireccional



(docs.openlinksw.com[3], 2009, apartat 16.17.2)

4.4.3. Importació d'ontologies

La importació d'una ontologia permet reusar vocabulari ja existent a la web semàntica i per tant s'evita la creació de distintes classes per la mateixa finalitat. D'aquesta forma es facilita la consecució dels objectius de la filosofia Linked Data. Així i tot s'ha de tenir en compte que quan s'importa una ontologia s'estan important totes les seues definicions i pot resultar que la nostra ontologia esdevinga més complexa del necessari. Una possible solució és usar qualificadors amb l'espai de noms adient per referenciar només aquelles classes, propietats i/o individus de l'ontologia de referència i no importar-la de forma completa. Altra solució seria establir algun mecanisme per fer importacions parcials.

En el cas de Virtuoso la importació d'ontologies es fa de forma completa. A continuació es mostra el mètode per portar-la a terme.

Com a requisit previ a la importació s'ha de dir que Sponger ha d'estar instal·lat. Per comprovar-ho i si escau instal·lar-lo cal connectar al *Conductor UI* (http://servidorVirtuoso:8890/conductor) com *dba* i anar a *System Admin, Packages* i veure si el paquet *rdf_mappers* està instal·lat.

Figura 18 Instal·lació d'Sponger (rdf_mappers)

Abans d'instal·lar

Name	Target	Installed Version	New Version	Last Update	Action
iSPARQL	dav	Not Installed	1.27.82/ 2012-05-07		Install
rdf_mappers	dav	Not Installed	1.34.37/ 2012-05-07		Install
tutorial	dav	Not Installed	1.00.6878/ 2012-05-07		Install

Després d'instal·lar

Name	Target	Installed Version	New Version	Last Update	Action
iSPARQL	dav	Not Installed	1.27.82/ 2012-05-07		Install
rdf_mappers	dav	1.34.37/ 2012-05-07	Not available	2012-05-20 18:53	Uninstall
tutorial	dav	Not Installed	1.00.6878/ 2012-05-07		Install

- **Procediment d'importació d'una ontologia**

L'ontologia triada és *http://purl.org/ontology/mo/*. Per importar-la es pot usar una de les dues sentències SPARQL següents.

```
SPARQL LOAD <http://purl.org/ontology/mo/>;
```

```
SPARQL
DEFINE get:soft "replace"
SELECT DISTINCT *
FROM <http://purl.org/ontology/mo/>
WHERE
{
```

```
?s ?p ?o
}
```

La primera sentència per definició importa el graf de l'ontologia indicada. En el segon cas el pragma Sponger (`get:soft`) indica que la consulta ha d'accedir a grafs remots quan no existisquen en local.

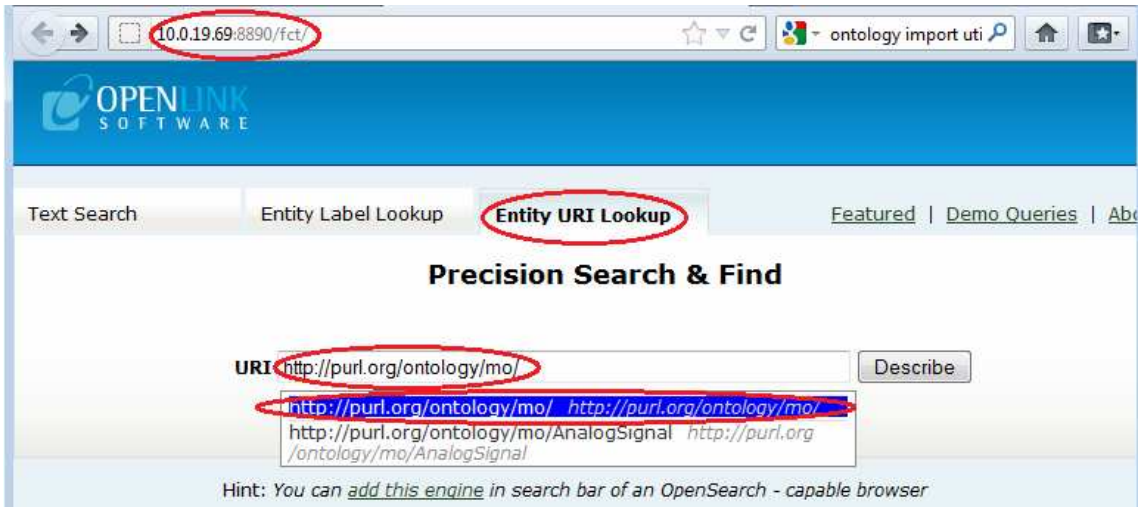
- **Comprovació de la importació**

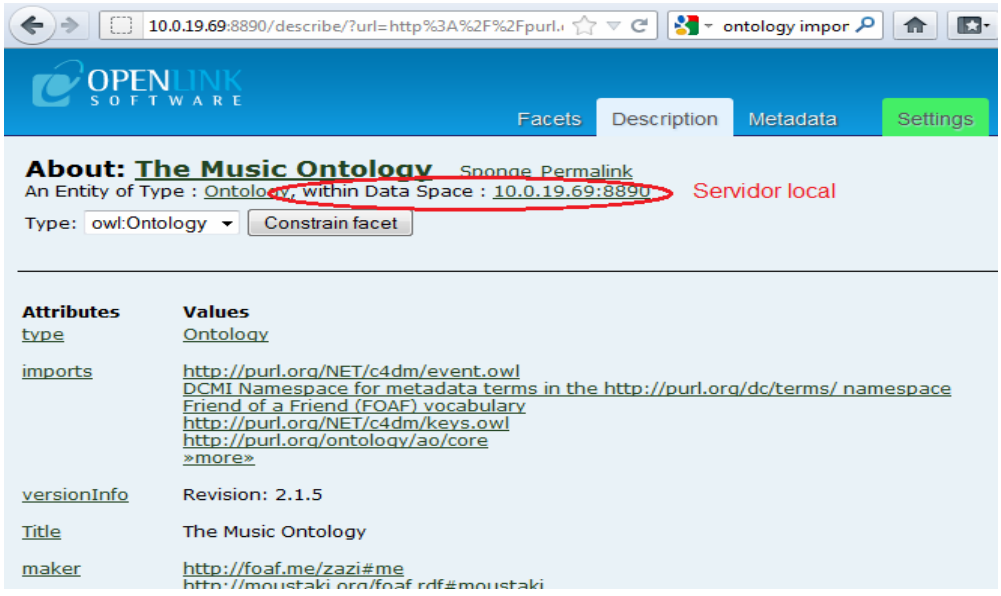
Es pot llançar la consulta anterior sense el pragma Sponger, en aquest cas només tindrà èxit si el graf existeix en el servidor Virtuoso local.

```
SPARQL
SELECT DISTINCT *
FROM <http://purl.org/ontology/mo/>
WHERE
{
?s ?p ?o
}
```

També es pot usar el navegador per facetes per consultar les ontologies locals.

Figura 19 Consulta d'ontologies amb Virtuoso Facet Browser





4.5. Avaluació del rendiment de Virtuoso

En aquest apartat es presenta el banc de proves LUBM, així com l'adaptació, d'aquest banc de proves, que incorpora Virtuoso. Finalment es mostra el procediment utilitzat per fer les proves i els resultats obtinguts.

4.5.1. Introducció

Per l'avaluació del rendiment de Virtuoso s'ha triat el banc de proves LUBM *(lehigh.edu, 2012)*. Aquest banc de proves ha estat dissenyat per avaluar SGBD semàntics de forma sistemàtica i estàndard, amb una sèrie de consultes sobre un conjunt de dades gran , generat automàticament, pertanyent a una sola ontologia.

A continuació es llisten els components del banc de proves LUBM:

- **Ontologia:** s'anomena *Univ-Bech* i el seu URI és *http://www.lehigh.edu/~zhp2/2004/0401/univ-bench.owl*.

- **Generador de dades (UBA1.7):** és una ferramenta que genera dades OWL sintètiques per l'ontologia *Univ-Bench*. Les dades són configurables en poder especificar: una llavor per a la generació de nombres aleatoris, un nombre d'universitats a crear i l'índex per la primera universitat. Les dades generades es refereixen a departaments, professors, estudiants, cursos i les seus relacions.

- **Consultes de prova:** el banc de proves proveeix 14 consultes, escrites en un llenguatge tipus KIF (Knowledge Interchange Format) , veure *(http://swat.cse.lehigh.edu/projects/lubm/query.htm)*.

- **Provador (UBT1.1):** mòdul de proves. Permet executar la carrega de dades i el joc de consultes.

4.5.2. Suport de Virtuoso per LUBM

Virtuoso usa les dades sintètiques generades amb UBA1.7 de LUBM sense cap modificació. Per altra part ha adaptat les consultes de prova a SPARQL i ha afegit la verificació que les consultes produeixen el resultat desitjat *(docs.openlinksw.com[4], 2012)*. En aquest treball s'ha fet ús del joc de proves adaptat per Virtuoso.

Virtuoso fa una part de la inferència a l'hora de la carrega perquè no és possible fer-la en temps de consulta. Aquestes són les inferències aplicades:

- Sempre es materialitza la relació de suborganització transitiva (*subOrganizationOf*). Per tant els triples de suborganització sempre estan presents.

- La inferència de relació inversa no es prova. Les consultes han estat reescrites per evitar la seua dependència.

- La inferència de subclasses i subpropietats es pot fer en temps de carrega o de consulta, es proven ambdues.

- El banc de proves no implica *owl:sameAs*. Conseqüentment el suport de Virtuoso per *owl:sameAs* no s'usa.

Virtuoso ha definit tres variants de les 14 consultes LUBM *(docs.openlinksw.com[4], 2012, apèndixs A i C)*:

- Inferència *Open Coded*. Les combinacions de subclasses i subpropietats s'expressen com unions.

- Inferència en temps de consulta de Virtuoso per subclasses i subpropietats, usant la clàusula `define input:inference 'inft'.`

- Amb dades prèviament materialitzades, on tots els triples, que impliquen les relacions de subclasses i subpropietats, estan físicament presents. La materialització no és el mode d'inferència per defecte de Virtuoso però es pot forçar la seua aplicació.

4.5.3. Procediment per a la realització de les proves

A continuació es descriuen els passos i ferramentes usats per preparar els jocs de dades de les proves i l'execució d'aquestes.

- **Entorn de proves**

Sistema utilitzat: màquina virtual sobre infraestructura virtual Vmware, amb 8GB de RAM, 50 GB de disc i sistema operatiu RHEL 5.

Procés d'instal·lació:

- Instal·lació dels requeriments del sistema operatiu per Virtuoso.

```
sudo yum install gcc gmake autoconf automake libtool flex \
    bison gperf gawk m4 make openssl-devel readline-devel wget
```

- Passos per la descarrega i instal·lació.

```
http://sourceforge.net/projects/virtuoso/files/virtuoso/6.1.5/virtuoso
-opensource-6.1.5.tar.gz

tar xvpfz virtuoso-opensource-6.1.5.tar.gz

cd virtuoso-opensource-6.1.5

./configure --prefix=/usr/local/ --with-readline

make install
```

El directori d'instal·lació és `/usr/local/virtuoso-opensource`

- Les úniques modificacions fetes al fitxer de configuració de la instància (`virtuoso.ini`) han estat les següents línies, recomanades per 8 GB de RAM.

```
NumberOfBuffers         = 680000
MaxDirtyBuffers         = 500000
```

- **Preparació dels jocs de dades**

S'ha de descarregar el paquet Java **UBA1.7**, de la pàgina web del projecte LUBM (http://swat.cse.lehigh.edu/projects/lubm/), i descomprimir-lo. Des del directori *classes* s'ha de llançar el comandament:

```
java edu.lehigh.swat.bench.uba.Generator -univ NNNN -onto
http://www.lehigh.edu/~zhp2/2004/0401/univ-bench.owl
```

On *NNNN* és el nombre d'universitats que s'han de generar. Els fitxers generats s'han d'ubicar en el directori *binsrc/tests/lubm/lubm_8000*, d'on seran llegits pels procediments de proves de Virtuoso.

He preparat jocs de dades progressivament per 1, 2, 4 , 8, 16, 32, 64, 128, 256, 512, 1024 i 2048 universitats. Executant per cadascun la carrega de dades i consultes descrites a continuació.

- **Carrega de les dades**

Virtuoso proveeix un paquet d'*scripts* de Linux i fitxers de procediments SPARQL per facilitar les proves LUBM. Aquests fitxers es troben al directori *binsrc/tests/lubm* dins l'arrel d'instal·lació de Virtuoso *(docs.openlinksw.com[5], 2012)*.

- Al directori de proves hi ha l'*script* Linux **`test_server.sh`** que s'ha d'executar una sola vegada per preparar el directori de proves i comprovar la instància de Virtuoso.

```
./test_server.sh virtuoso-t
```

- Per carregar les dades, de les universitats generades, s'ha d'executar el procediment **`lubm-load.sql`**, del qual obtindrem els triples inserits i el temps d'execució.

```
time isql 1111 dba dba lubm-load.sql
```

- **Llançament de les 14 consultes LUBM**

- Per executar el joc de 14 consultes LUBM, amb inferència *Open Coded*, s'ha d'executar el procediment **`lubm.sql`**, del qual s'obté un llistat de dades i el temps d'execució.

```
time isql 1111 dba dba lubm.sql
```

- Per executar el joc de 14 consultes LUBM, amb inferència en temps de consulta s'ha d'executar el procediment **`lubm-inf.sql`**, del qual s'obté un llistat de dades i el temps d'execució.

```
time isql 1111 dba dba lubm-inf.sql
```

- Per executar el joc de 14 consultes LUBM, amb dades prèviament materialitzades s'ha d'executar primer el procediment **`lubm-cp.sql`** (per materialitzar-les) i a continuació el procediment **`lubm-phys.sql`**, d'aquests s'obté un llistat de dades i el temps d'execució.

```
isql 1111 dba dba lubm-cp.sql
time isql 1111 dba dba lubm-phys.sql
```

4.5.4. Resultats de les proves

A continuació es mostren els resultats obtinguts amb les proves realitzades. Les taules i gràfiques s'han agrupat segons el nombre d'universitats, un grup d'1 a 32 universitats i l'altre grup de 64 a 2048 universitats. Aquesta agrupació s'ha fet per facilitar la visualització de les gràfiques a l'existir valors molt dispars.

Taula 11 Resultat de les proves LUBM a Virtuoso (1 a 32 universitats)

Nombre d'universitats	Carrega (Temps)	Carrega (Triples)	Consulta Inf Open Coded	Consulta Inf temps consulta	Consulta materialització prèvia
1	5,2	100.545	1,0	0,8	0,4
2	12,0	230.063	1,5	1,0	0,5
4	24,7	477.786	2,3	1,8	1,1
8	49,9	1.001.420	4,4	3,4	2,5
16	101,0	2.094.863	7,7	5,8	2,8
32	208,0	4.257.051	13,5	13,2	7,2

Figura 20 Temps de carrega de dades LUBM (1 a 32 universitats)

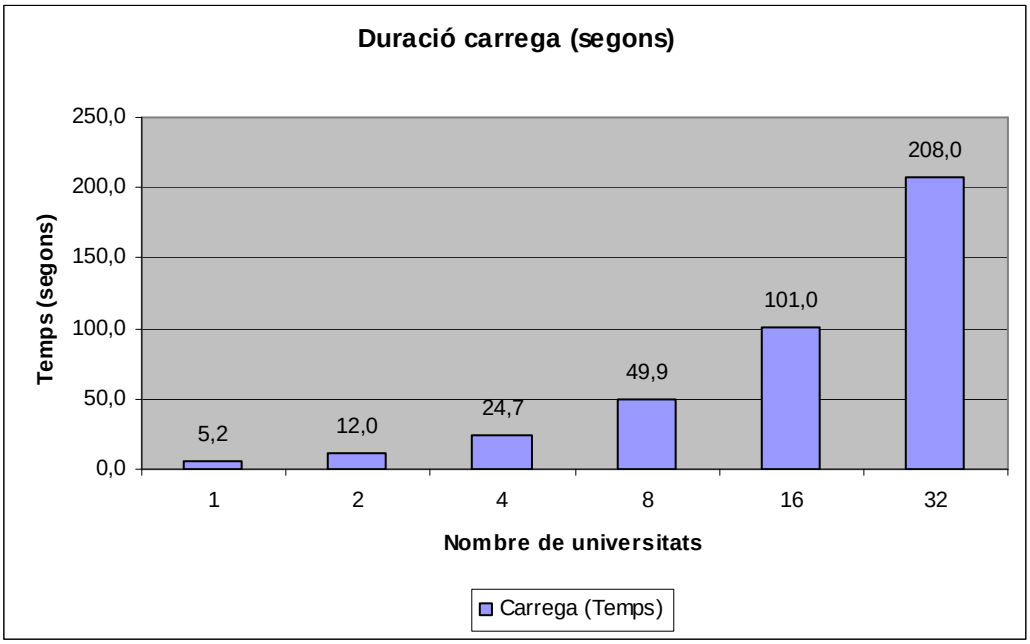


Figura 21 Nombre triples carregats (1 a 32 universitats)

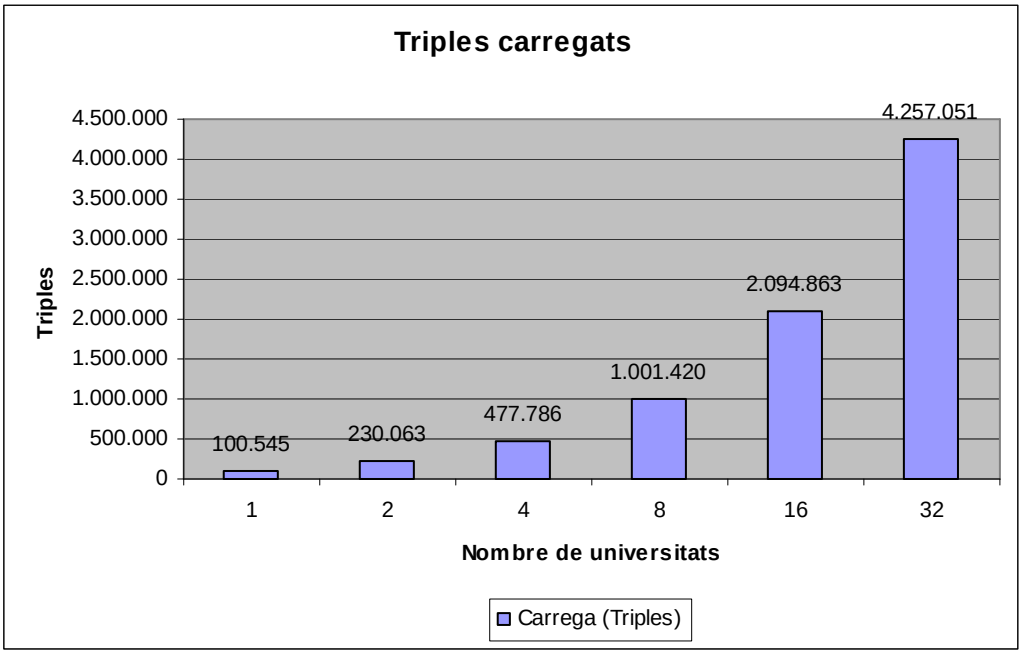
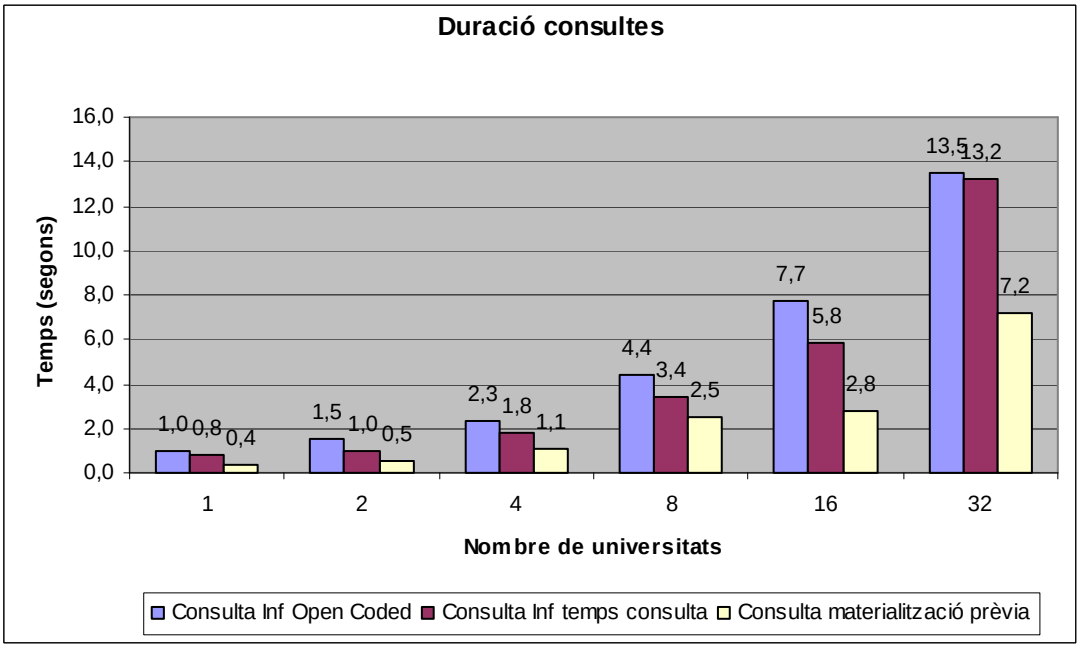


Figura 22 Duració dels 3 tipus de 14 consultes LUBM (1 a 32 universitats)



Taula 12 Resultat de les proves LUBM a Virtuoso (64 a 2048 universitats)

Nombre d'universitats	Carrega (Temps)	Carrega (Triples)	Consulta Inf Open Coded	Consulta Inf temps consulta	Consulta materialització prèvia
64	403	8.582.790	20	26	5
128	1.107	17.103.860	44	51	27
256	2.941	34.095.887	117	98	60
512	8.067	68.339.766	398	249	103

Nombre d'universitats	Carrega (Temps)	Carrega (Triples)	Consulta Inf Open Coded	Consulta Inf temps consulta	Consulta materialització prèvia
1024	10.324	136.695.264	435	353	238
2048	23.148	273.382.814	1.497	957	718

Figura 23 Temps de carrega de dades LUBM (64 a 2048 universitats)

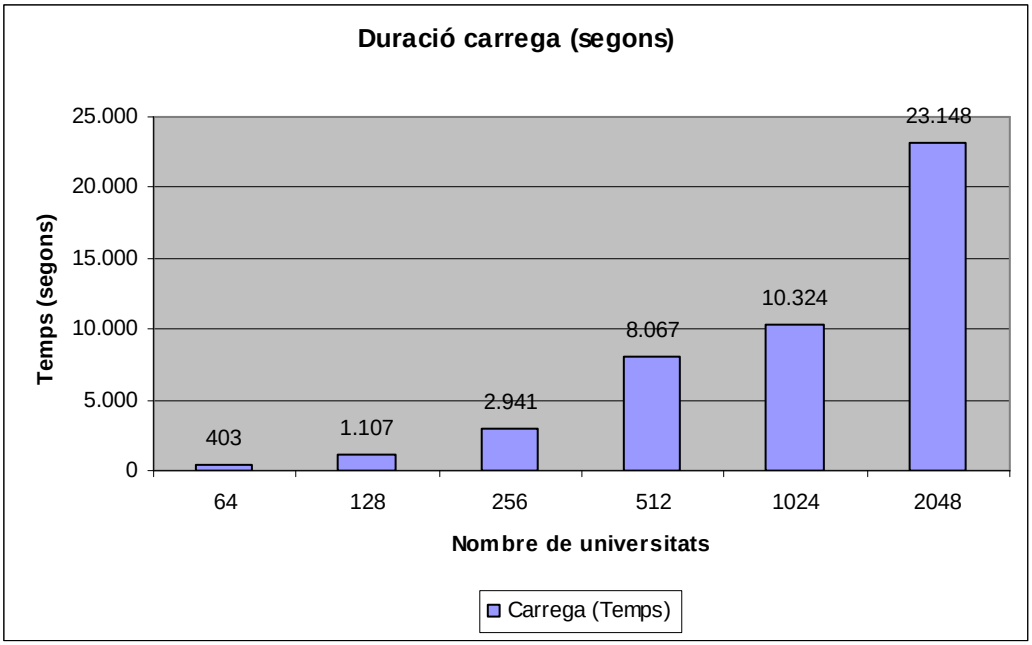


Figura 24 Nombre triples carregats (64 a 2048 universitats)

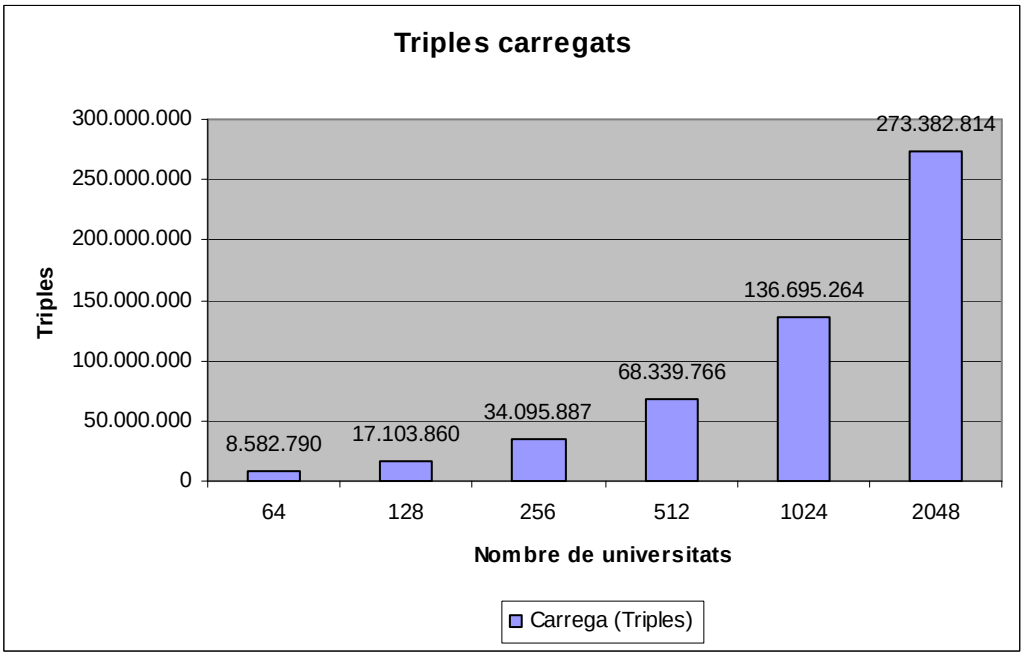
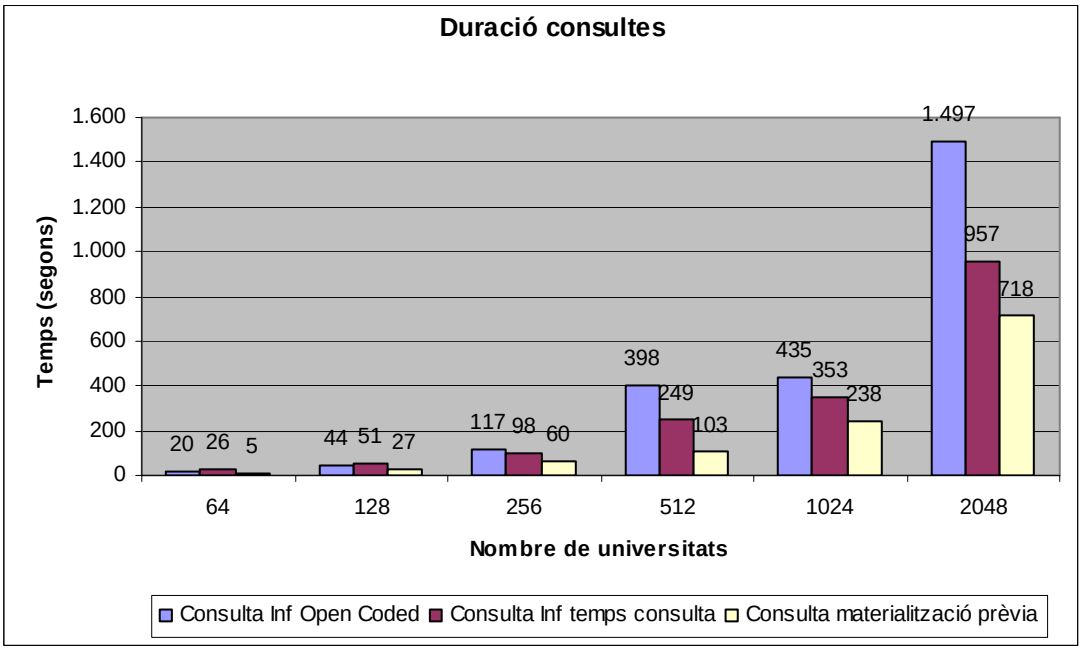


Figura 25 Duració dels 3 tipus de 14 consultes LUBM (64 a 2048 universitats)



Amb els resultats obtinguts s'observa que hi ha una evolució prou lineal dels temps d'execució, tant de les carregues com de les consultes. En la majoria del casos, al duplicar el nombre d'universitats els temps passen a ser una mica més del doble.

Comparant els resultats dels tres tipus de jocs de consultes es veu que les que apliquen materialització prèvia són les que s'executen en menys temps, seguides per les que apliquen la inferència de l'ontologia en temps de consulta.

Hauria estat interesant arribar a un nombre d'universitats on el temps d'execució hagués augmentat amb un factor de multiplicació molt superior a 2. Açò indicaria la quantitat de dades per la qual la màquina provada ja no seria vàlida i caldria escalar Virtuoso, usant una màquina més potent o creant un clúster amb varies màquines per repartir la carrega de treball. Aquest punt no l'he pogut assolir perquè en arribar a les proves amb 2048 universitats, els fitxers amb les dades de proves ocupaven 22 GB i el fitxer de base de dades, el mateix, i s'esgotà l'espai en disc disponible.

4.6. Integració d'aplicacions corporatives amb LOD

Al llarg del present document s'han estudiat els conceptes bàsics de la web semàntica, s'han presentat distintes solucions d'emmagatzemament específiques per aquesta finalitat i en la darrera part s'ha profunditzat en l'anàlisi d'un SGBD semàntic, Virtuoso Universal Server. Arran de l'apartat d'importació d'ontologies, m'ha semblat raonable fer una cerca sobre algun cas real que aprofités ontologies existents per integrar-les en una aplicació. Com a resultat de la cerca he trobat el projecte que s'exposa en els següents paràgrafs i que és representatiu de la filosofia de la web semàntica i que aporta una idea de l'evolució d'aquesta en el futur.

El projecte LOD2 està cofundat per la Unió Europea amb 15 socis tecnològics, entre els quals es troba OpenLink Software Inc com a proveïdor de l'SGBD que suporta el projecte, el projecte està coordinat pel grup d'investigació AKSW de la Universitat de Leipzig *(lod2.eu[1], 2012)*. L'objectiu principal és portar l'èxit inicial de

LOD a una realitat a escala mundial integrant dades *Linked Open* amb dades corporatives. Per aconseguir-ho cal millorar la coherència i qualitat de les dades publicades a la web semàntica, igualar el rendiment dels SGBD semàntics amb els relacionals, estendre la confiança amb la web Linked Data i afavorir l'accés als publicadors de dades i als usuaris. Per portar a terme aquests objectius, LOD2 ha aplegat una sèrie d'eines tecnològiques (veure Annex C).

Un dels programes de treball de LOD2, *WP8 – Use Case 2: LOD2 for enterprise data web (lod2.eu[2], 2012)*, té com a finalitat aplicar la plataforma LOD2 en un entorn corporatiu real amb forts requisits d'integració d'informació semàntica. Aquest programa de treball implementarà un conjunt de procediments per l'entrada de dades que integra anotacions i característiques semàntiques. També és la seua finalitat suggerir un llistat de bones pràctiques per emmagatzemar, gestionar, accedir i intercanviar dades semàntiques, així com donar una mesura dels beneficis que LOD2 pot aportar a una corporació (tecnològics i econòmics).

El WP8 està previst que finalitze cap a Juny de l'any 2014. El primer document que s'ha lliurat *(lod2.eu[3], 2012)* consisteix en l'anàlisi de requeriments d'un flux de treball per ajudar en els processos de selecció i contractació de recursos humans. Els lliurables més destacats que es generaran són el de bones pràctiques en gestió semàntica d'informació corporativa i unes versions de les eines tecnològiques de LOD2 adaptades per a grans empreses.

5. Conclusions

El present treball ha consistit en l'estudi d'alguns dels magatzems semàntics existents en l'actualitat. En concret s'ha triat AllegroGraph, Oracle Semantic Technologies, OWLIM i Virtuoso Universal Server com a representants principals de les distintes tipologies d'SGBD semàntics existents. En l'estudi comparatiu dels SGBD esmentats s'han revisat una sèrie de conceptes, característiques i funcionalitats que s'han de tenir en compte sempre que es valoren nous SGBD semàntics.

Abans de començar l'estudi dels magatzems semàntics es van exposar els conceptes bàsics sobre la web semàntica, que cal conèixer per tal de poder comprendre altres conceptes més complexos. Per tant, al començament del treball s'explicaren RDF i OWL com a eines bàsiques per a la representació d'ontologies.

En la segona part del treball, es trià Virtuoso Universal Server per un estudi més detallat de les funcionalitats d'un SGBD semàntic i per fer una instal·lació real sobre la que realitzar una sèrie de proves de carrega.

Com a conclusió principal de la informació recollida al llarg del treball es pot dir que actualment ens trobem davant d'un nou concepte, la web semàntica, que és una de les bases de la Web 3.0, la web del coneixement en contraposició a la web clàssica de la informació. Dins la web semàntica els magatzems RDF són el motor tecnològic que la fa possible.

Actualment ja són realitat projectes de caire públic i corporatiu que usen aquests conceptes i tecnologies en la pràctica. Com a paradigma de la web semàntica tenim el projecte *Linked Open Data (LOD)* format per uns 300 conjunts de dades interconnectats amb més de 31.000 milions de triples emmagatzemats, al cor de la qual es troba la DBpedia.

Si bé es pot dir que els magatzems semàntics es troben en un estadi inicial de maduresa, també és veritat que encara no són tant eficients com els relacionals, en els seus camps d'aplicació respectius. Açò es tracta de pal·liar amb la formació de *clústers* de servidors i limitant la quantitat de dades emmagatzemades a cada SGBD.

Tot açò no és impediment per deduir que els magatzems semàntics tenen un camp propi d'aplicació dins la web semàntica, no cobert correctament pels relacionals, i que han vingut per quedar-se.

Una de les principals línies de futur dels magatzems semàntics ha de ser la millora del seu rendiment fins posar-se al nivell dels relacionals. Tècniques d'aplicació en futures versions de Virtuoso, com l'emmagatzemament per columnes (columnwise) o l'execució vectorial es dirigeixen cap aquesta fita. A una investigació recent, explicada a la tesi *(A. Owens, 2011)*, es fa un recull de conceptes importants respecte al rendiment dels magatzems RDF i es centra en l'estudi dels índex AHRI com a medi per obtenir una latència acceptable per la interacció humana, deixant clar alhora que els SGBD relacionals no són la solució.

Per altra banda, una mica més pràctica, hi ha projectes com LOD2 *(lod2.eu[1], 2012)* de la Unió Europea que tracten d'estimular i ampliar l'efecte *Linked Open Data*, per tal d'estendre l'aplicació real de la web semàntica en diversos camps. Aquesta mena d'iniciatives són també importants per tal de promoure la implantació de la web

semàntica a les empreses i donar-la a conèixer cada vegada més a la comunitat informàtica i als responsables estratègics que pugen decidir usar-la.

Així mateix, es desitjable que els magatzems semàntics passen a formar part dels temaris de les assignatures de l'àrea de base de dades en un futur.

Glossari

Checkpoint (Punt de control) Moment en què una base de dades escriu en memòria externa tots els canvis produïts per les transaccions finalitzades, emmagatzemades al dietari.

Clúster Arquitectura d'ordinadors basada en unir dos o més ordinadors, mitjançant una xarxa, per processar una càrrega de treball. S'obté protecció davant fallades i millora de rendiment.

Diagrama de Gantt Eina de planificació del treball en el temps. Permet controlar l'avanç d'un projecte i reprogramar les tasques si cal.

Faceted search (cerca per facetes) Tècnica d'accés a la informació classificada per diferents facetes, permet a l'usuari aplicar múltiples filtres. Permet accedir i ordenar les classificacions de diverses formes.

Federació Conjunt de xarxes o ordinadors heterogenis que, amb l'ús d'un protocol comú, interoperen de forma col·lectiva.

Front-end (frontal) En el disseny d'un sistema informàtic és la part que es relaciona amb l'exterior (p.ex. els usuaris).

GRDDL (Gleaning Resource Descriptions from Dialects of Languages) Especificació d'estàndards per declarar que un document XML és compatible amb RDF.

IRI (Internationalized Resource Identifiers) Identificador de recursos (URI) que permet l'ús de caràcters internacionals (Unicode/ISO10646).

Jena Marc de treball Java per fer aplicacions per la web semàntica.

KIF (Knowledge Interchange Format) Llenguatge informàtic orientat a l'intercanvi de coneixement entre sistemes dispars.

Late binding (enllaçat tardà) Mecanisme de programació en què un mètode es cercat pel seu nom en temps d'execució en comptes de fer-ho en la compilació (early binding).

Latencia d'acces És la suma dels retards d'una sèrie d'operacions informàtiques.

Locality Es refereix a l'ús de dades emmagatzemades en ubicacions contigües o properes.

Lucene És una API de codi obert per a la recuperació de dades usant índexs i cerca de text complet.

LUBM (Lehigh University Benchmark) Banc de proves per avaluar els magatzems de la web semàntica de forma estàndard i sistemàtica.

Microformats Enfocament de marcat semàntic que tracta de reusar les etiquetes HTML i XHTML com a metadades per tal de processar aquesta informació de forma automàtica.

Middleware (Programari intermediari) Programari que permet a una aplicació interactuar o comunicar-se amb altres aplicacions, programaris, xarxes, ...

ORDI (Ontology Representation and Data Integration) Marc de treball en Java. És un programari intermediari de codi obert per ontologies que permet integrar dades empresarials via un model de dades RDF.

Parser (Analitzador) Analitzador sintàctic que processa un text per determinar si és gramaticalment correcte segons una gramàtica donada.

Read commited És el nivell d'aïllament més baix en base de dades. Permet llegir dades quan potser altra transacció l'està modificant.

Reificació És el procés pel qual una idea abstracta es convertida en un model de dades.

REST (Representational state transfer) Arquitectura de programari per sistemes distribuïts com la WWW. Els clients inicien peticions i els servidors les processen i responen. Tot basant-se en la transferència de representacions de recursos. Els mètodes disponibles són GET, POST, DELETE i PUT.

SCN (System Change Number) Marca de temps usada en la base de dades Oracle per determinar si està en un estat consistent.

Sesame Marc de treball, estàndard de facto, per processar dades RDF. Proveu anàlisi, emmagatzemament, inferència i consulta d'aquestes dades. Ofereix una senzilla API que es pot connectar als magatzems semàntics més coneguts.

Sha1sum Sentència de Linux que permet comprovar la integritat d'un fitxer per mig d'una suma de comprovació.

SKOS (Simple Knowledge Organization System) Desenvolupa especificacions i estàndards per sistemes d'organització del coneixement com thesaurus, esquemes de classificació, taxonomies, etc., per la web semàntica.

SOAP (Simple Object Access Protocol) protocol estàndard perquè dos objectes puguen comunicar-se intercanviant dades XML.

SPARQL Endpoint Protocol conforme a SPARQL que permet als usuaris (normalment automàtics) consultar dades amb aquest llenguatge. Els resultats es proveeixen en un format processable automàticament.

SPIN Estàndard de facto per representar regles SPARQL i restriccions sobre models de la web semàntica. Permet que els usuaris defineixen les seus funcions SPARQL i plantilles per consultes. De vegades SPIN s'anomena *SPARQL rules*.

Tancament deductiu El tancament deductiu d'una ontologia *O* conté tots els axiomes, vinculats amb ella, i que estan ben formats segons una lògica *L* donada.

Thesaurus És un llistat de paraules agrupades per tenir un significat paregut (sinònims).

UDDI (Universal Description Discovery and Integration) Estàndard bàsic pels serveis web, és accedit per mig de missatges SOAP per trobar documents WSDL.

URI (Uniform Resource Identifier) Cadena de caràcters per identificar un recurs. Aquesta identificació permet interactuar amb aquest recurs en una xarxa (per exemple WWW).

VSP (Virtuoso Server Pages) Forma part de Virtuoso Universal Server. És un sistema de fitxers que allotja codi PL de Virtuoso i codi HTML.

W3C (World Wide Web Consortium) Comunitat internacional que desenvolupa estàndards per assegurar el creixement de la web a llarg termini.

WebDAV (Web Distributed Authoring and Versioning) Conjunt d'extensions d'Http per possibilitar l'edició de fitxers d'un lloc web.

WSDL (Web Service Definition Language) Format XML usat per descriure la interfície pública dels serveis web.

Bibliografia i fonts d'informació

Documents

- F. Bry, J. Maluszynski. Semantic Techniques for the Web. Springer, 2009.
- C.R. RayMaden. Learning XML.O'REILLY, 2001.
- A. Hertel, J. Broekstra., and H. Stuckenschmidt. "RDF Storage and Retrieval Systems". On-line, 2008.
- D. Faye, O. Cure, O. Blin. "A survey of RDF storage approaches". ARIMA - Volume 15. On-line, 2012.
- Apunts "Bases de dades II". UOC.
- A. Owens. "An Investigation Into Improving RDF Store Performance" / "Using Low Latency Storage to Improve RDF Store Performance". University of Southampton. On-line, 2011.
- M. Bergman. "Advantages and Myths of RDF". AI3. On-line 2009
- O. Erling. "Virtuoso, a Hybrid RDBMS/Graph Column Store". Bulletin of the IEEE Computer Society Technical Committee on Data Engineering. On-line, 2012.

Webs

- http://en.wikipedia.org/wiki/Semantic_Web
- http://es.wikipedia.org/wiki/Web_semantica
- http://en.wikipedia.org/wiki/Graph_database
- http://linkeddata.org
- http://www.w3.org/TR/rdf-concepts/
- http://www.w3.org/TR/rdf-schema/
- http://www.w3.org/TR/rdf-primer/#rdfmodel
- (www.w3.org[1], 2012):http://www.w3.org/TR/2004/REC-owl-features-20040210
- (www.w3.org[2], 2012):http://www.w3.org/TR/owl-guide/
- http://www.w3.org/TR/sparql11-overview/
- http://www.w3.org/TR/rdf-sparql-query/
- http://www.linkeddatatools.com
- http://virtuoso.openlinksw.com
- http://docs.openlinksw.com

  - (docs.openlinksw.com[1], 2012): http://docs.openlinksw.com/virtuoso/rdfsparqlrule.html
  - (docs.openlinksw.com[2], 2012): http://docs.openlinksw.com/virtuoso/rdfperformancetuning.html
  - (docs.openlinksw.com[3], 2009) http://docs.openlinksw.com/pdf/virtdocs.pdf
  - (docs.openlinksw.com[4], 2012) http://virtuoso.openlinksw.com/dataspace/dav/wiki/Main/VOSArticleLUBMBenchmark
  - (docs.openlinksw.com[5], 2012) http://www.openlinksw.com/weblog/oerling/?id=1504

- http://www.ontotext.com/owlim
  - (www.ontotext.com[1], 2012): http://www.ontotext.com/owlim/usage
  - (www.ontotext.com[2], 2012): http://owlim.ontotext.com/display/OWLIMv42/Primer+Introduction+to+OWLIM#PrimerIntroductiontoOWLIM-OWLIMInteroperabilityandArchitecture
  - (www.ontotext.com[3], 2012): http://owlim.ontotext.com/display/OWLIMv50/OWLIM-SE+Reasoner#OWLIM-SEReasoner-PerformanceOptimizationsinRDFSandOWLSupport
  - (www.ontotext.com[4], 2012): http://owlim.ontotext.com/display/OWLIMv43/OWLIM-SE+Full-text+Search

- https://docs.oracle.com
- http://www.oracle.com
  - (www.oracle.com[1], 2012): http://www.oracle.com/technetwork/database/options/semantic-tech/semtech11gr2-featover-131765.pdf
  - (www.oracle.com[2], 2007): http://www.oracle.com/technetwork/database/enterprise-edition/11goracletexttwp-133192.pdf
- http://download.oracle.com
  - (download.oracle.com[1], 2012):
  - http://download.oracle.com/otndocs/tech/semantic_web/pdf/2010_ora_semtech_capintper.pdf

- http://www.franz.com/agraph
  - (franz.com[1], 2012): http://franz.com/agraph/allegrograph/
  - (franz.com[2], 2012): http://franz.com/agraph/success/index.lhtml
  - (franz.com[3], 2012): http://www.franz.com/agraph/support/documentation/4.5/text-index.html
  - (franz.com[4], 2012): http://www.franz.com/agraph/gruff/
  - (franz.com[5], 2012): http://www.franz.com/agraph/sentient/

- http://lod2.eu/
  - (lod2.eu[1], 2012): http://lod2.eu/Welcome.html
  - (lod2.eu[2], 2012): http://lod2.eu/WorkPackage/wp8.html
  - (lod2.eu[3], 2012): http://static.lod2.eu/Deliverables/LOD2_D8.1.1_Enterprise_Requirements.pdf

- (lehigh.edu, 2012): http://swat.cse.lehigh.edu/projects/lubm/

Annex A. Termes dels llenguatges OWL

Es fa notar que quan el terme expressat ja existeix a RDF o RDFS anirà precedit del prefix corresponent `rdf:` o `rdfs:`, mentre que els nous termes introduïts per OWL s'expressen amb cap prefix.

Termes d'OWL Lite:

RDF Schema Features:

Class (Thing, Nothing)
rdfs:subClassOf
rdf:Property
rdfs:subPropertyOf
rdfs:domain
rdfs:range
Individual

(In)Equality:

equivalentClass
equivalentProperty
sameAs
differentFrom
AllDifferent
distinctMembers

Property Characteristics:

ObjectProperty
DatatypeProperty
inverseOf
TransitiveProperty
SymmetricProperty
FunctionalProperty
InverseFunctionalProperty

Property Restrictions:

Restriction
onProperty
allValuesFrom
someValuesFrom

Restricted Cardinality:

minCardinality (only 0 or 1)
maxCardinality (only 0 or 1)
cardinality (only 0 or 1)

Header Information:

Ontology
imports

Class Intersection:

intersectionOf

Datatypes:

xsd datatypes

Versioning:

versionInfo
priorVersion
backwardCompatibleWith
incompatibleWith
DeprecatedClass
DeprecatedProperty

Annotation Properties:

rdfs:label
rdfs:comment
rdfs:seeAlso
rdfs:isDefinedBy
AnnotationProperty
OntologyProperty

Termes d'OWL DL i Full:

Class Axioms:

oneOf, dataRange
disjointWith
equivalentClass (applied to class expressions)
rdfs:subClassOf (applied to class expressions)

Boolean Combinations of Class Expressions:

unionOf
complementOf
intersectionOf

Arbitrary Cardinality:

minCardinality
maxCardinality
cardinality

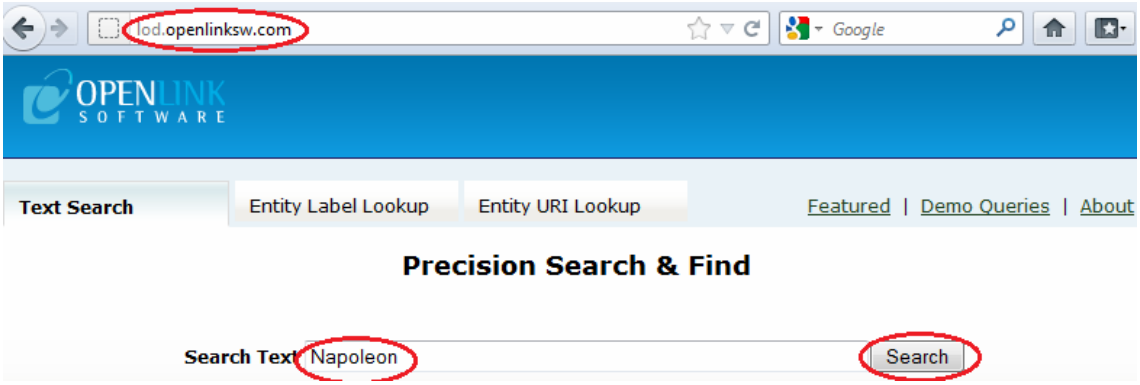
Filler Information:

hasValue

Annex B. Exemple de consulta per facetes (XML i Web)

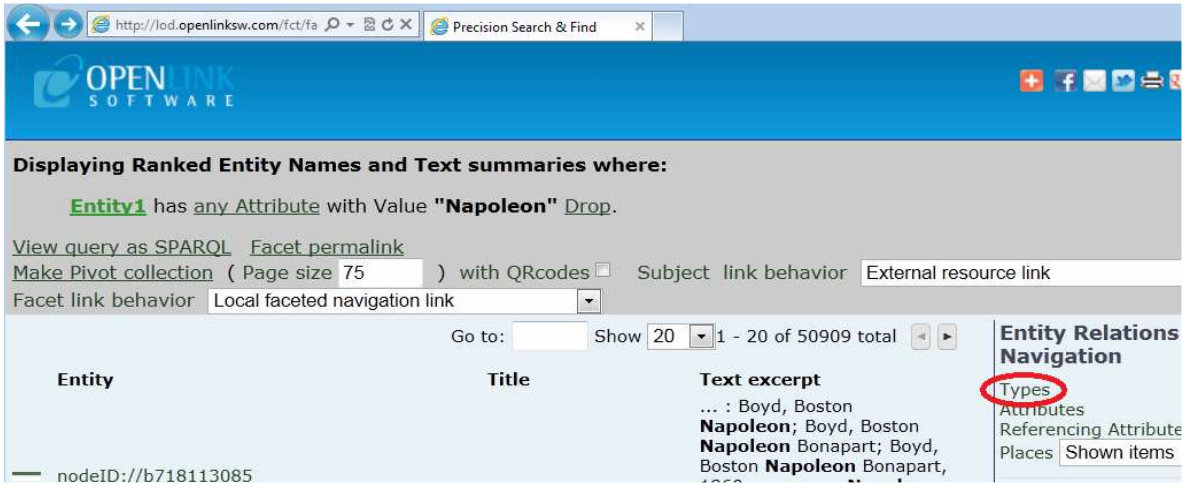
- Introduir el terme de cerca “Napoleon”

```
<query inference="" same-as="" view3="" s-term="e" c-term="type">
  <text>napoleon</text>
  <view type="text" limit="20" offset="" />
</query>
```



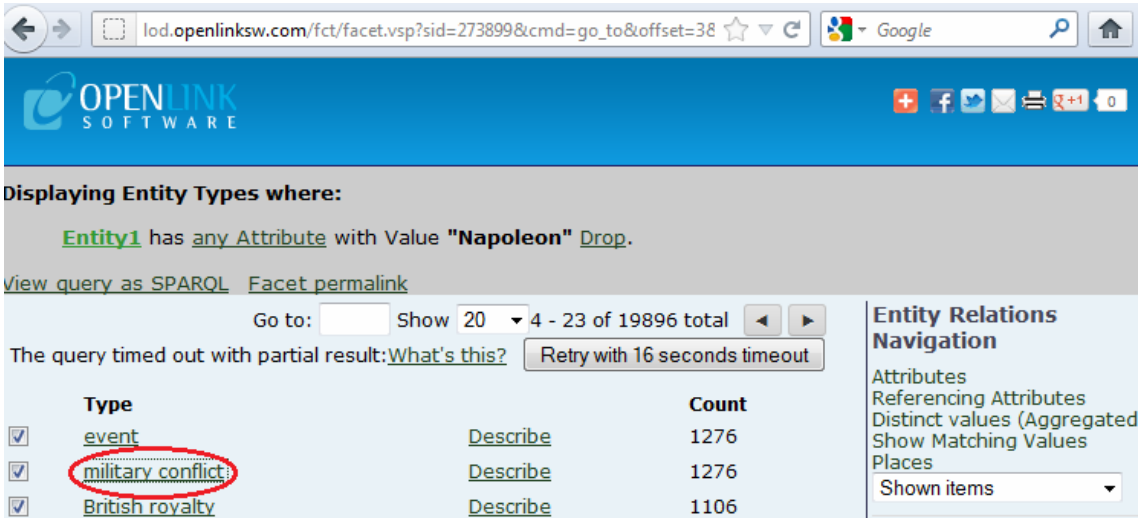
- Seleccionar la vista “Types”

```
<query inference="" same-as="" view3="" s-term="e" c-term="type">
  <text>napoleon</text>
  <view type="classes" limit="20" offset="0" location-prop="0" />
</query>
```



- Elegir “military conflict”

```
<query inference="" same-as="" view3="" s-term="e" c-term="type">
  <text>napoleon</text>
  <view type="classes" limit="20" offset="0" location-prop="0" />
  <class iri="yago:ontology/MilitaryConflict" />
</query>
```

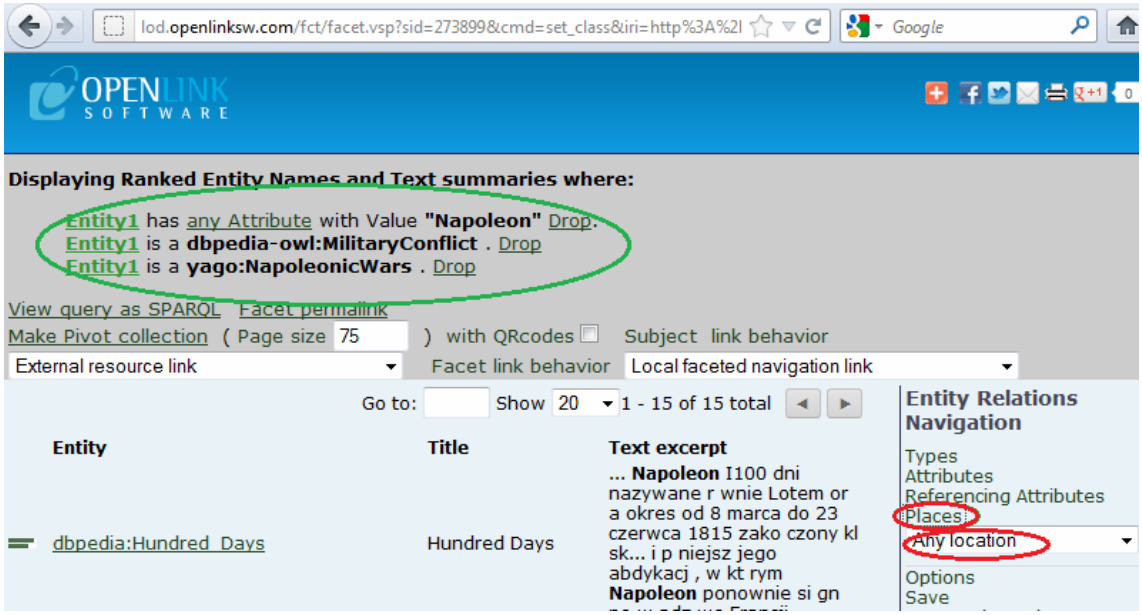


- Elegir “dbpedia:Napoleonic Wars”

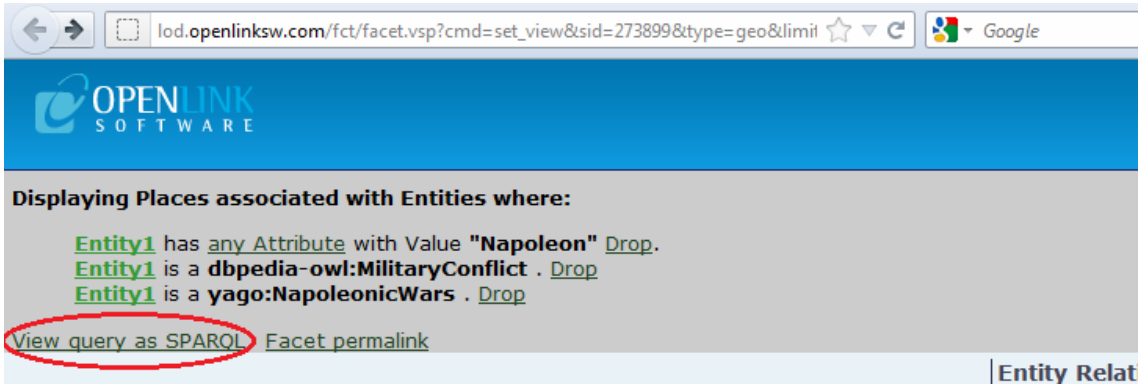
```
<query inference="" same-as="" view3="" s-term="e" c-term="type">
  <text>napoleon</text>
  <view type="classes" limit="20" offset="0" location-prop="0" />
  <class iri="yago:ontology/MilitaryConflict" />
  <class iri="yago:class/yago/NapoleonicWars" />
</query>
```

- Seleccionar “Any location” i elegir “Places”

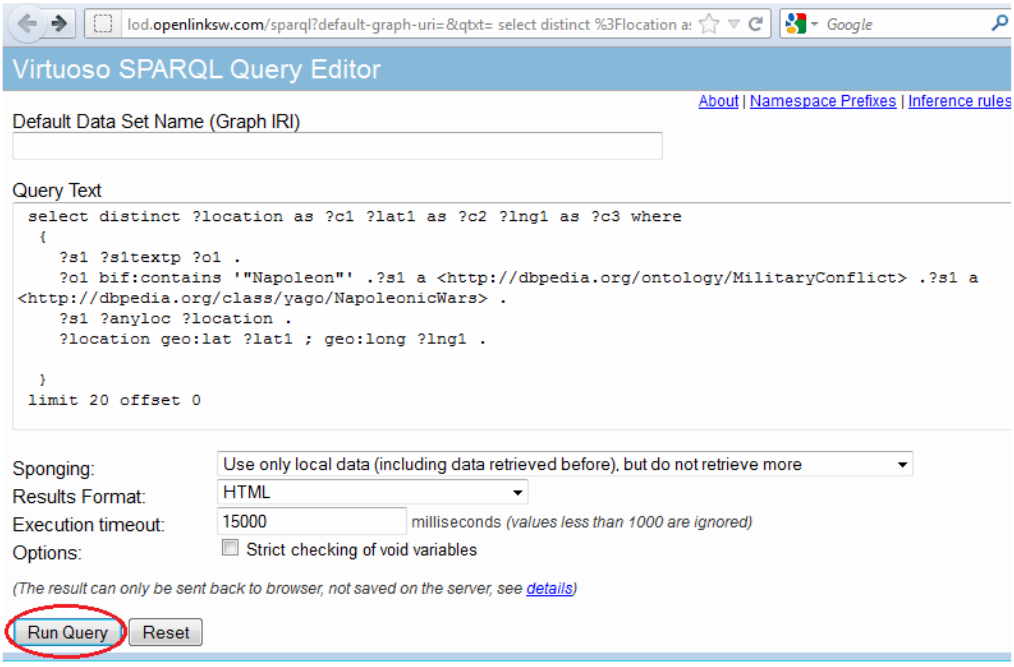
```
<query inference="" same-as="" view3="" s-term="e" c-term="type">
  <text>napoleon</text>
  <class iri="yago:ontology/MilitaryConflict" />
  <class iri="yago:class/yago/NapoleonicWars" />
  <view type="geo" limit="20" offset="0" location-prop="any" />
</query>
```



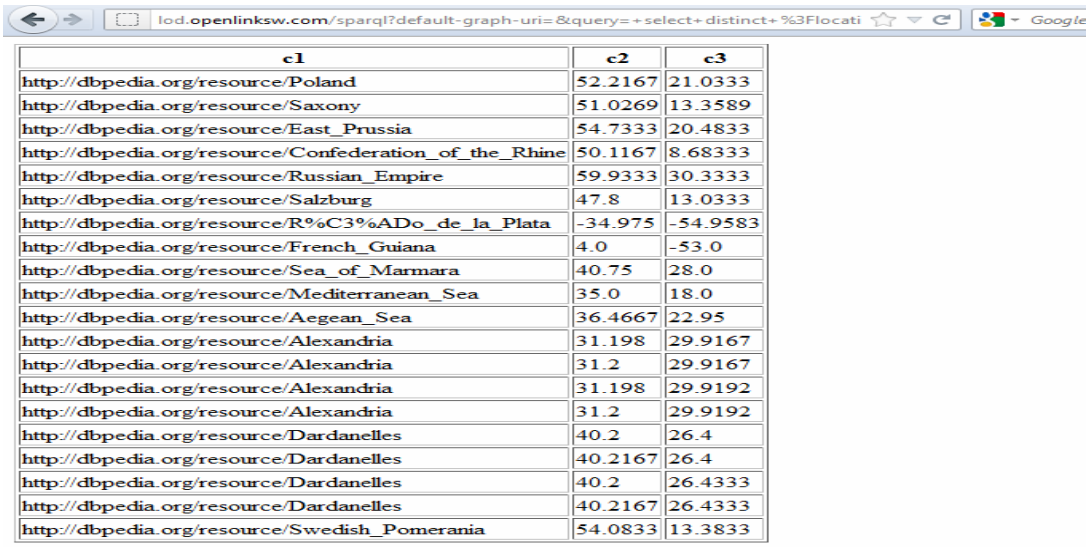
- Veure la consulta SPARQL generada



- Executar la consulta generada



- Veure els vint primers resultats obtinguts



c1	c2	c3
http://dbpedia.org/resource/Poland	52.2167	21.0333
http://dbpedia.org/resource/Saxony	51.0269	13.3589
http://dbpedia.org/resource/East_Prussia	54.7333	20.4833
http://dbpedia.org/resource/Confederation_of_the_Rhine	50.1167	8.68333
http://dbpedia.org/resource/Russian_Empire	59.9333	30.3333
http://dbpedia.org/resource/Salzburg	47.8	13.0333
http://dbpedia.org/resource/R%C3%ADo_de_la_Plata	-34.975	-54.9583
http://dbpedia.org/resource/French_Guiana	4.0	-53.0
http://dbpedia.org/resource/Sea_of_Marmara	40.75	28.0
http://dbpedia.org/resource/Mediterranean_Sea	35.0	18.0
http://dbpedia.org/resource/Aegean_Sea	36.4667	22.95
http://dbpedia.org/resource/Alexandria	31.198	29.9167
http://dbpedia.org/resource/Alexandria	31.2	29.9167
http://dbpedia.org/resource/Alexandria	31.198	29.9192
http://dbpedia.org/resource/Alexandria	31.2	29.9192
http://dbpedia.org/resource/Dardanelles	40.2	26.4
http://dbpedia.org/resource/Dardanelles	40.2167	26.4
http://dbpedia.org/resource/Dardanelles	40.2	26.4333
http://dbpedia.org/resource/Dardanelles	40.2167	26.4333
http://dbpedia.org/resource/Swedish_Pomerania	54.0833	13.3833

Annex C. Eines tecnològiques del projecte LOD2

- **OntoWiki**

És una ferramenta per crear contingut RDF, facilita la presentació visual d'una base de coneixement com un mapa d'informació.

- **PoolParty**

És un sistema de gestió de thesaurus i editor SKOS (Simple Knowledge Organization System) per a la web semàntica amb capacitats de mineria de text i *linked data*.

- **Sig.ma**

Ferramenta per explorar i aprofitar les dades de la web, de múltiples llocs que pugen tenir dades RDF encastades.

- **Comprehensive Knowledge Archive Network (CKAN)**

Catàleg de conjunts de dades RDF i altres fonts de coneixement.

- **D2R Server**

És una ferramenta per publicar bases de dades relacionals en la web semàntica.

- **DBpedia Extraction**

Projecte de la comunitat per extraure informació estructurada de la Wikipedia. La seua ontologia multidomini s'usarà com a base per diverses aplicacions del projecte LOD2.

- **DL-Learner**

És una eina d'aprenentatge automàtica per OWL i lògica descriptiva.

- **MonetDB**

Base de dades de codi obert i d'alt rendiment que permet emmagatzemar dades relacionals, XML i RDF.

- **SemMF**

És un marc de treball per calcular la similitud entre objectes representats com grafs RDF.

- **Silk Framework**

Marc de treball que ajuda als publicadors a establir enllaços RDF entre ítems de dades de diferents orígens.

- **Sindice**

Estableix una infraestructura coherent per processar, consolidar i consultar la web semàntica.

- **Sparallax**

Sparallax és un navegador per facetes per SPARQL Endpoints.

- **Triplify**

Facilita la "semantificació" d'aplicacions web, buscant les estructures semàntiques a les seues taules relacionals.

- **OpenLink Virtuoso**

Magatzem semàntic analitzat en aquest treball fi de carrera.

- **WIQA**

Marc de treball *Web Information Quality Assessment* per avaluar la qualitat de la informació abans de ser usada per qualsevol tasca determinada.

- **Spatial Semantic Browser**

És una aplicació per navegar i explorar dades RDF geogràfiques en un mapa.

- **LIMES**

És un marc de treball que serveix per descobrir enllaços a la web semàntica a gran escala amb bona eficiència temporal.