Міністерство освіти і науки України
Житомирський державний університет імені Івана Франка

Курсова робота
на тему:

«Побудова мережі Інтернет в рамках концепції Semantic Web»

Студента 36 групи
Абрамовича Ігоря

Житомир 2010

Зміст

Вступ

Однією з причин підвищеного інтересу до проекту Semantic Web є практична зацікавленість у поліпшенні якості пошуку у Веб. Дослідження з цієї проблеми ведуться в різних напрямках і дають різноманітні результати у вигляді нових пошукових систем. Такі системи, як Swoogle, дозволяють лише виконувати пошук онтологій за ключовими словами. Але такий сервіс є дуже корисним для розробників семантичних систем і онтологій, хоча він і не розрахований на простого користувача. Джерелами інформації в них служать набори RDF-даних, включаючи дані, пов'язані в рамках проекту Linked Open Data, і мікроформати.

Можна відзначити й інші пошукові системи Semantic Web, багато з яких знаходяться на стадії бета-тестування, тому оцінити їх можливості складно. Деякі системи йдуть по шляху «поглиблення у Веб», інші – більш прискіпливо розвивають алгоритми інтелектуального аналізу та використовують різноманітні джерела інформації про документи, які знаходяться «поза документом» у Веб. Розвиток технологій інформаційного пошуку призвів до інтенсивного використання мета-інформаційно-пошукових систем; багатоагентних інформаційно-пошукових систем; систем, побудованих на реалізації онтологічних, мовних та управлінських угод і т.п. Більшість пошукових систем йдуть по шляху розвитку персоналізації пошуку, тобто розпізнавання та задоволення потреб користувача. Традиційні пошукові системи стають все більш точними та об'ємними, однак вони не можуть перевершити інтелект людини. Вони можуть лише порівнювати слова, а не зміст ідеї, яка обговорюються ними. Нові технології пошукових систем 3-го покоління ще знаходяться в стадії формування, але вже зараз вони дають позитивні результати. Нові пошукові системи можуть допомогти зробити пошук більш значущим, суб'єктивним і прив'язаним до задач (task-based), що стоять перед користувачем. Таким чином, розвиток пошукових систем йде по шляху, метою якого є задоволення потреб індивідуального користувача, з його перевагами, характером, рівнем підготовки , знань тощо.

Мета роботи полягає у дослідженні концепції Semantic Web, побудови семантики в загальному, принцип роботи семантичної системи і її зв'язків.

Об'єктом дослідження є проект Semantic Web, його задача і проблеми. Предметом дослідження у цій роботі є вивчення і розгляд уже реалізованих моделей, побудованих на платформі семантичної мережі.

1.Поняття Semantic Web

Феномен World Wide Web став можливий тільки завдяки практичному використанню набору широко поширених стандартів на різних рівнях, що забезпечило інтероперабельність даних. Сучасна тенденція розвитку Інтернету полягає в переході від документів, "що читаються комп'ютером" (machine readable) до документів, які "комп'ютер розуміє" (machine understandable).

Web розроблявся, як інформаційний простір, корисний не тільки для комунікації людини з людиною, але і як простір, в якому зможуть ефективно співпрацювати і комп'ютери. Одне з головних перешкод на шляху до цього полягає в тому, що більша частина інформації в Web призначена для її розуміння людиною. Очевидно, що така структура даних не може бути зрозумілою для веб-робота, що її проглядає. Підхід Semantic Web базується на розробці мов, для вираження інформації у формі, придатній для машинної обробки.

Ідея Semantic Web була запропонована в 1998 році Тімом Бернерс-Лі (Tim Berners-Lee), який є винахідником WWW, URI, HTTP і HTML.

Semantic Web являє собою мережу інформаційних вузлів, які пов'язані один з одним таким чином, щоб наявна інформація могла легко оброблятися комп'ютером. Його можна розглядати як ефективний спосіб представлення даних у Всесвітній павутині, або як глобально пов'язану базу даних. Даний проект пропонує реалізацію повної системи з автоматизованого створення та зберігання семантичного ядра контенту, наданого у Всесвітній павутині.

Проект Semantic Web - це спроба зібрати всі сталі ідеї і зробити так, щоб вони змогли працювати разом всередині мережі Інтернет. Для досягнення цієї мети використовуються стандарти, які розроблені не тільки консорціумом W3C, а й іншими організаціями. Мета проекту - дозволити взаємодіяти цим стандартам між собою, всередині децентралізованої системи, без втручання людини.

Проект Semantic Web [1], започаткований у 2001 році, на даний момент знаходиться в стадії активної розробки, намагається інтегрувати в себе всі вже наявні на даний момент підходи, з метою створити дійсно універсальний засіб семантичного пошуку інформації [2, 3]. Велика увага приділяється архітектурі та моделі розподіленого середовища [4], архітектурі метаданих [5 - 8]. Як сказано у визначенні, яке надане на домашній сторінці проекту - «Semantic Web є абстрактним поданням даних у Всесвітній павутині, яке базується на стандартах RDF та інших стандартах, які мають поширення. Проект розробляється Консорціумом W3C у співдружності з великою кількістю дослідників, вчених і промислових партнерів »[9].

«Semantic Web - це розширення поточного Web, в якому інформація надається з добре певним значенням, яке краще дозволить комп'ютерам і людям працювати разом. … Його ідея в тому, щоб мати дані в Web, визначені і пов'язані між собою таким чином, щоб їх можна було використовувати для більш ефективного дослідження, автоматизації, інтеграції та повторного використання в різних додатках ... ці дані можуть бути загальнодоступними і обробленими, автоматичними засобами так само, як і людьми »[2].

У рамках даного проекту задіяні такі передові технології, як агентно-орієнтовний підхід у програмуванні [10] , онтології [15, 16], XML [17 - 19], RDF [20 - 22], та інші. В даний час поширюється використання Web-агентів (у спрощеному вигляді веб-сервісів), які розробляються як для окремих завдань, так і для створення ядра Semantic Web [23 - 28].

Як зазначив професор Джон Сова, - Semantic Web - багато-дисциплінарна тема, яка об'єднує теорії та методи трьох областей:

- логіка - формальні структури і правила логічного висновку;
- онтології - опис типів сутностей, які відносяться до предметної області;
- теорія моделей.

Інтернет - це мережа комп'ютерів, об'єднаних каналами, які використовують протоколи (TCP / IP) для зв'язку між собою. Web - це мережа сайтів, які використовують гіперпосилання для переходів між сторінками [29]. Традиційний Web базується на мові розмітки документів HTML. HTML-сторінка описує форму подання інформації в Web-браузері, а ця мова важко піддається автоматичному змістовному аналізу. Автоматизувати навіть такі тривіальні завдання, як пошук людей, проектів, програм в Інтернеті - неможливо. Наступний етап розвитку Інтернет - Semantic Web - представляє собою перехід на новий рівень представлення даних - рівень знань та автоматизованої обробки. Технологія Semantic Web дозволить комп'ютеру інтерпретувати інформацію, представлену в Web, нарівні з людьми, для чого й розроблена графова модель опису ресурсів RDF (Resource Description Framework).

У загальному вигляді Semantic Web (за Тіму Бернерс-Лі) - це:

- інтероперабельність даних між програмними додатками та організаціями;
- набір інтероперабельних стандартів для обміну знаннями;
- архітектура для взаємопов'язаних спільнот та словників [30].

Архітектура Semantic Web

З точки зору архітектури Semantic Web можна розглядати, як три яруси (мал. 1):
базис, який складається з унікальної глобальної ідентифікації ресурсу, метаданих для декларування фактів про ресурси, і спільної мови для вираження метаданих і знань, що реалізовані за допомогою онтологій, для загальнодоступного розуміння і загального словника метаданих, і правил для додавання нових метаданих та знань;
базовий сервіс, наприклад, логічний висновок і запити до метаданих, і онтологія, роз'яснення таких висновків, управління довірою, агенти, пошукові системи, онтології;
сервіси додатків, наприклад сервіс агентства подорожей.

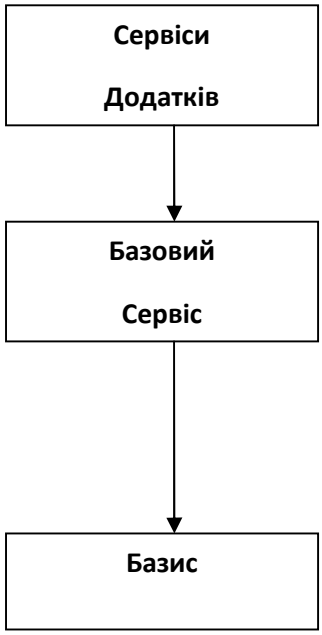


Рис. Мал.1. 1. Три яруси мережі Semantic Web

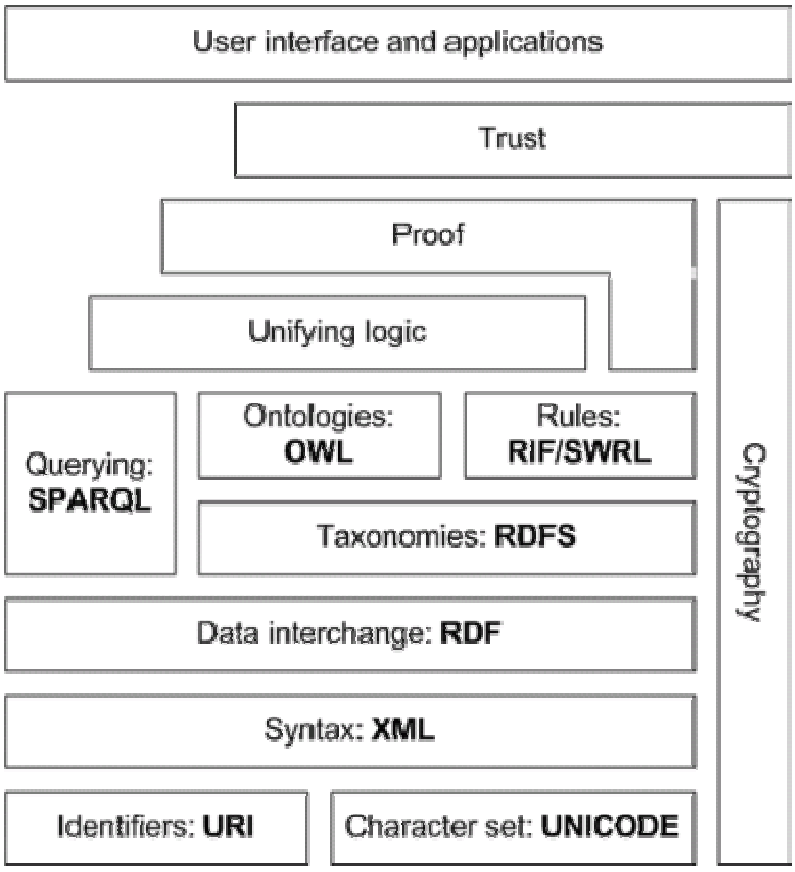
Технології, які задіяні у розробці Semantic Web:

- Семантичний пошук;
- Питально-відповідні системи;
- Агенти;
- Об'єднання знань (інтеграція баз даних);
- Проникливі обчислення [29].

У 1998 році Тім Бернерс-Лі запропонував наступний логічний план побудови Semantic Web [31]:

1. 1. Синтаксис для представлення знань, який використовує посилання на онтології (RDF);
2. 2. Мова опису онтологій (OWL);
3. 3. Мова опису веб-сервісів (WSDL, OWL-S);
4. 4. Інструменти читання / розробки документів Semantic Web (Jena, Haystack, Protege);
5. 5. Мова запитів до знань, які записані в RDF (SPARQL);
6. 6. Логічний висновок знань (знаходиться на етапі обговорення);
7. 7. семантична пошукова система (наприклад, SHOE).

Базова модель Semantic Web (пиріг Тіма) в редакції 2006 показана на мал.2 [32].



Мал 2. Базова модель Semantic Web в редакції 2006

Фундаментальними основами Semantic Web є:

- графова модель представлення на пів структурованих даних (OEM, Lore);
- формальна логіка (логіка першого порядку, бази знань, фрейми);
- архітектура WWW (URI / IRI, Unicode, XML, HTTP);
- криптографія з відкритим ключем.
- Розглянемо структуру базової моделі Semantic Web більш детально в наступних пунктах.

1.1 URI – універсальний ідентифікатор ресурсів

В Web для ідентифікації елементів використовуються "Уніфіковані ідентифікатори ресурсів", або скорочено URI (Uniform Resource Identifier). URI можна присвоїти до чого завгодно, і якщо ця сутність має URI, то про неї можна говорити, що вона знаходиться "в Web": це може бути людина, книга, абстрактна концепція, тобто все, що має назву.
URI є базисом Web. «URI - це компактний рядок символів, який використовується для ідентифікації абстрактних або фізичних ресурсів» [33].
Однією з форм URI є URL (Uniform Resource Locator), уніфікований покажчик ресурсу. URL - це адреса, за якою завантажується Web-сторінка.
Також необхідно вказати, що в початковій базової моделі в нижньому ярусі, було вказано ще й базове кодування – тобто, загальний для всіх принцип кодування всіх можливих символів багатьох мов - кодова таблиця UNICODE.
За синтаксисом URI стежить комітет IETF. Документ, який опублікований цим комітетом RFC 2396, є спільною специфікацією URI. Консорціум W3C підтримує список схем URI.
У 2005 році на зміну URI був запропонований інтернаціоналізувати ідентифікатор ресурсу - Internationalized Resource Identifiers (IRI), що ідентифікує абстрактний або фізичний ресурс будь-якою мовою світу. URI можуть містити тільки латинські символи та знаки пунктуації з набору символів US-ASCII (в цілому близько 60 символів).
Для забезпечення принципів інтернаціоналізму, збереження «читабельності» для людини, в IRI було запропоновано, що ці ідентифікатори можуть містити будь-які символи Юнікоду (Unicode/ISO10646) у чистому вигляді, без будь-якого кодування. IRI не обмежують права інших мов і ведуть до більш високого ступеня рівноправності користувачів Інтернету. У майбутньому ідентифікатори IRI покликані замінити URI.
Зазвичай посилання URI є відносною для будь-якого документа, в якому вона знайдена. Якщо, наприклад, проглядається документ з базовим URI http://exslt.org/math/min/math.min.template.xsl, і в ньому виявляється URI-посилання .. / .. / random / random.xml, то вона призведе до даного документу з адресою http://exslt.org/random/random.xml. У форматі HTML є можливість винести базовий елемент в заголовок документа, щоб перекрити базовий URI. Базова специфікація XML (XML Base) забезпечує еквівалентну форму в XML.

1.2. Розширювана мова розмітки (XML)

XML [34] (eXtensible Markup Language) являє собою дуже простий і при цьому потужний, і гнучкий текстовий формат, для опису документів довільної структури. XML був розроблений і затверджений в якості стандарту в ProductID в 1998 р Консорціумом W3C, для спрощення реалізації, а також для забезпечення інтероперабельності між SGML і HTML. Він є підкласом мови SGML, однак більш простий для розуміння і обробки.
Опції XML:
Подання синтаксису для інших мов розмітки;
Семантична розмітка Web-сторінок. XML-представлення може використовуватися на Web-сторінці разом з таблицею стилів XSL, що визначає коректний вивід на екран різних елементів;
Єдиний формат обміну даних. XML-представлення може передаватися між двома застосуваннями, як об'єкт даних.

Мова XML дозволяє кожному створювати свій власний формат документів і потім писати документи в цьому форматі. Ці формати документів можуть включати розмітку, яка уточнює зміст контенту документа. Документ з розміткою може "читатися" комп'ютером.
XML і RDF - сучасні Internet-стандарти, які служать для забезпечення семантичної інтероперабельності в Web. При цьому XML піднімає питання, пов'язані тільки зі структурою документів. RDF більше пристосований для забезпечення семантичної інтероперабельності, оскільки пропонує модель даних, яку можна розширити таким чином, щоб вона охоплювала більш досконалі методики подання онтології.

1.3. Загальна схема опису ресурсів RDF

Для опису предметної області ресурсів запропонований стандарт RDF (Resource Description Framework) [35 - 42], прийнятий у 1999 році консорціумом W3C і підтриманий багатьма провідними виробниками ПЗ, і постачальниками контенту. Початкове призначення RDF було в описі XML-ресурсів з різних точок зору. RDF представляє собою модель опису метаданих. Ця мова використовує XML-синтаксис.
У той час, як модель даних XML є графом з позначеними вершинами і не позначеними дугами (тобто без зв'язків), модель даних RDF є графом з позначеними, як вершинами, так і дугами, що дозволяє визначати зв'язки між сутностями.
Модель Resource Description Framework має мету: стандартизувати визначення та використання метаданих, які описують ресурси Web. Однак, RDF також добре підходить і для представлення даних [43].
Стандарт RDF (Resource Description Framework) включає дві основні частини - власне спосіб опису ресурсів, а також спосіб завдання схем, за якими ресурс описується.
Перша частина RDF [44] визначає просту модель для опису об'єкта, який розглядається, як ресурс, як зв'язок між ресурсами в термінах, найменованих властивостей і значень.
Друга (RDF Schema - RDFS) [45, 46] служить для завдання структури предметної області та аналогічно - діаграмі класів в UML.
На RDF можна описувати, як структуру ресурсу, так і пов'язану з ним предметну область. RDF описує ресурси у вигляді орієнтованого розміченого графа - кожен ресурс може мати властивості, які в свою чергу, також можуть бути ресурсами або їх колекціями.
Базовий будівельний блок у RDF - це трійка об’єктів «об'єкт - атрибут - значення», який часто записують у вигляді A (O, V), тобто «Об'єкт O має атрибут A зі значенням V». Такий зв'язок можна також представити, як ребро з міткою A, яке об'єднує два вузли, O і V: [O] - A -> [V]. Така нотація досить корисна, оскільки RDF дозволяє міняти місцями об'єкти та значення. Таким чином, кожен об'єкт може грати роль значення, яке в графічному представленні відповідає ланцюжку з двох ребер з мітками.
Крім усього вищезгаданого, RDF допускає форму подання, в якій будь-який вираз RDF в трійці може бути об'єктом або значенням, тобто графи можуть бути, як вкладеними, так і лінійними. В Web це дозволено, наприклад, висловлювати сумнів або згоду з виразами, створеними іншими людьми.
Головна мета RDF - запропонувати базову модель даних «об'єкт - атрибут - значення» для метаданих. Окрім цієї семантики, що описана в стандарті лише неформально, RDF не містить будь-яких чітких правил, орієнтованих на моделювання даних. Також, як XML Schema використовується для визначення словника, RDF Schema дозволяє розробникам

визначати конкретний словник для даних RDF (такий, як authorOf) і вказувати види об'єктів, до яких можуть застосовуватися ці атрибути. Іншими словами, механізм RDF Schema надає базову систему типів для моделей RDF.
Таким чином, RDF надає можливість формулювати твердження у вигляді, придатному для обробки комп'ютером і це є основою Semantic Web.
Метадані – це дані,призначені для ідентифікації, опису або локалізації інформаційних ресурсів, не залежно від фізичної природи ресурсу. А RDF – одна із стандартизованих форм представлення цих метаданих.

1.4. Метаданные Метадані

У базовій моделі Semantic Web, представленої вище, запропонованої Тімом Бернерс-Лі, явно не виділено наявність засобів опису метаданихТим не менш, у своїх роботах, наприклад, [30, 31], а також у роботах інших вчених вказується на важливість включення в концепцію Semantic Web поняття метаданих.
Метадані це дані про дані. Більш точно, це дані, призначені для ідентифікації, опису або локалізації (місця розташування) інформаційних ресурсів, не залежно від фізичної природи ресурсу.
Було розроблено безліч схем опису метаданих, серед яких слід згадати наступні:
Topic Maps (XMT) [47] - стандарт ISO (ISO / IEC 13250:2003) для представлення та обміну знаннями з точки зору пошуку інформації.
Text Encoding Initiative (TEI) [48] - міжнародний проект з розробки нормативів для розмітки (marking up) електронних текстів, таких як романи, пьєси, вірші; головним чином для підтримки досліджень у гуманітарній сфері.
Metadata Encoding and Transmission Standard (METS) [49] - стандарт кодування і передачі метаданих, був розроблений для задоволення потреби у стандартній структурі даних для опису складних цифрових бібліотечних об'єктів.
Metadata Object Description Schema (MODS) [50] - схема метаданих опису об'єктів, яка була виведена з MARC 21, і призначена для перенесення відібраних даних з існуючих записів метаданих MARC 21 або для створення оригінальної запису опису ресурсу.
Encoded Archival Description (EAD) [51] - закодований архівний опис, був розроблений, як спосіб розмітки даних, які містяться в пошукових коштах, для того, щоб вони знаходилися й показувалися в оперативному режимі.
Learning Object Metadata (LOM) [52] - стандарт IEEE 1484.12.1-2002 метаданих об'єктів навчального процесу для повторного використання ресурсів навчального характеру, таких, як: комп'ютерне та дистанційне навчання.
Online Information Exchange (ONIX) [53] - міжнародний стандарт схеми метаданих, який розроблений видавцями книжкової промисловості Сполучених Штатів і Європи.
Однак, базовими для Semantic Web в даний момент визнаються стандарти Dublin Core, FOAF, SIOC і DOAP [54].
FOAF (Friand-Of-A-Friend) [55 – 57] – це формат машинно-оброблюваних сторінок, що описують персональну інформацію про людей і їх діяльності (фотографії, календарі та інше) у форматі XML.
SIOC (Semantically-Interlinked Online Communities) [58] – документи, що описують онлайн-спільноти. SIOC забезпечує взаємозв'язок таких засобів обговорення інформації, як блоги, форуми і поштові розсилки, між собою.

Description of a Project Description of a Project (DOAP) [59] - документи, що описують в мережі проекти з відкритим вихідним кодом.
Серед цих стандартів виділяється Dublin Core [60], як один з базових стандартів для представлення даних про інформаційні ресурси в Semantic Web. Dublin Core [61, 62] - набір елементів (властивостей) для опису документів, який був розроблений в березні 1995 року. Мета Dublin Core - забезпечення мінімального набору елементів опису, які сприяють впровадженню опису та автоматичної індексації документоподібних мережевих об'єктів за принципом, подібного карткам бібліотечного каталогу. Набір метаданих Dublin Core призначався для використання засобами дослідження ресурсів Інтернету, такими, як веб-кроулери пошукових систем, а також передбачалося, щоб Dublin Core був досить простим набором для розуміння і використання широким колом авторів і випадкових публікаторів, які розміщують інформацію в Інтернеті. Елементи Dublin Core широко використовуються в документуванні Інтернет-ресурсів. На даний момент елементи Dublin Core визначені в Dublin Core Metadata Element Set, Version 1.1: Reference Description [63]. Розширювати сам набір елементів можна, як самостійно, так і з використанням вже наявних стандартів. Наприклад, для опису людей і організацій (які виступають як елементи метаданих Dublin Core: Creator, Publisher або Contributor) можна застосувати стандарт для електронних бізнес-карт (vCard [64]). Загальні міркування з цього приводу даються в [65], а конкретна пропозиція надається в [66 - 68].
Як наголошується в офіційному описі RDF, метадані можуть бути вбудованими (embedded) в сам ресурс, наприклад, в HTML сторінки [69] або документи, наприклад, MsWord (це найпростіший підхід для опису сторінок), а можуть зберігатися і оновлюватися незалежно від ресурсів. Багато хто з виробників програмного забезпечення вже випускають ряд продуктів, які автоматично формують деякий невеликий блок RDF-опису, всередині документа. Другий підхід є більш універсальним, так, як в цьому випадку метадані можуть бути створені для будь-якого ресурсу. В даний час вже розпочато проект на основі Open Directory [70] (пошукова система Google) з автоматичним створенням репозиторії RDF-описів ресурсів Інтернет.
У разі розміщення метаданих окремо від ресурсу, самі метадані переважно зберігаються (і передаються) у форматі XML. При цьому максимально використовуються можливості моделі RDF та забезпечується вільний обмін інформацією (interoperability). Обмін метаданими зводиться до пересилання RDF / XML-файлів (тобто текстових файлів у форматі XML або просто посилань на ці файли), тобто може бути повністю автоматизований.
RDF Schema слугує для метаданих тим, що вона може представити конкретні дані(метадані) в RDF форматі, уже згідно з RDF Schema .

1.5 RDF Schema

Першим "пластом" Semantic Web над тільки, що обговорених синтаксисом, є проста модель типізації даних. Схема і онтологія - це кошти для опису змісту і зв'язку між термами.
На основі RDF 23 січня 2003 був запропонований робочий проект RDF Vocabulary Description Language 1.0: RDF Schema [71]. Схема RDF була розроблена, як проста модель типізації даних для RDF. Як вказується в документі, RDF є мовою загального застосування для подання інформації в Інтернет. Дана специфікація описує як

використовувати RDF для опису RDF-словників. Вона визначає базовий словник, призначений для цих цілей і прийняті угоди, які можуть бути використані при створенні додатків Semantic Web для підтримки більш складних словників RDF-описів. Мова опису словника RDF визначає класи і властивості, які можуть бути використані для опису інших класів і властивостей, а також робити деякі більш складні речі, такі, як створення діапазонів і областей для властивостей.

Три найбільш важливих поняття, які дає нам RDF і схема RDF - це "Ресурс" (rdfs: Resource), "Клас" (rdfs: Class) і "Властивість" (rdfs: Property). Ці поняття є "класами" в тому розумінні, що цим класам можуть належати терміни.

Як вже було зазначено, RDF Schema визначається в термінах базової інформаційної моделі RDF - структури графа, який описує ресурси і властивості. Всі словники RDF використовують деяку базову структуру: вони описують класи ресурсів і типи зв'язків між ресурсами. Ця спільність дозволяє різнорідні словники, створені для машинної обробки, і відповідає вимогам, щодо створення метаданих, в яких твердження можуть бути отримані з безлічі різнорідних децентралізованих словників, створених різними спільнотами за різними принципами і різними методами.

Опис за допомогою RDF не обмежується тільки описом документів Інтернет. Цей стандарт досить універсальний і гнучкий для того, щоб описувати більшість типів структурованих даних. Наприклад, в RDF природно виражаються діаграмами сутній зв'язки, які широко застосовувані для проектування баз даних. Опис семантики ресурсу на RDF може бути як «зовнішнім», коли описується ресурс в цілому, так і «внутрішнім», коли описується внутрішня структура ресурсу - будь-то база даних, XML-документ, або цілий сайт.

Важливою особливістю стандарту RDF, який лежить в основі XML, є розширюваність. На RDF можна задати структуру опису джерела, використовуючи і розширюючи вбудовані поняття RDF-схем, такі як класи, властивості, типи, колекції. Модель схеми RDF включає спадкування; успадковуватися можуть як класи, так і властивості.

Крім опису структури, RDF дозволяє оперувати твердженнями. Вираз «ресурс R1, як властивість P має ресурс R2» можна проінтерпретувати і як предикат P (R1, R2), а потім використовувати це твердження як об'єкт інших тверджень. Така інтерпретація дозволяє описувати, з допомогою RDF, концептуальну інформацію.

Таким чином, RDF цілком підходить на роль універсальної мови опису семантики ресурсів і взаємозв'язків між ними.

Однак, як стверджують самі автори стандарту, RDF має й ряд відсутніх властивостей, які вказують як наступні:

- неможливість вказати потужність множини значень властивості, наприклад, що «Людина має тільки одного біологічного батька»;
- неможливість вказати того, що подана властивість (наприклад, hasAncestor - має предка, прототип) є транзитивна, наприклад, що «якщо A hasAncestor B, і B hasAncestor C, тоді A hasAncestor C»;
- неможливість вказівки того, що два різних класи, визначені у різних схемах, фактично представляють одне і те ж поняття;
- неможливість вказівки того, що два різних примірника (instances), визначені окремо, фактично представляють один і той самий суб'єкт;

- неможливість визначення нових класів у термінах операцій (наприклад, об'єднання і перетин) над іншими класами.

Найбільш розвиненою мовою представлення онтологій в даний час є OWL (Web Ontology Language), яка розширює можливості XML, RDF, і RDF Schema. Онтології грунтуються на математичному апараті формальної логіки (descriptive logic, DL)- мала підмножина, якого охоплена RDF-схемою

1.6. Онтології

Онтології, в загальному вигляді, визначаються, як спільно використовувані формальні концепції конкретних предметних областей, вони дають загальне уявлення про поняття, інформацією, з яких, можуть обмінюватися люди та програми. Вони дозволяють скласти в концепцію домен фіксуванням сутностей і зв'язків у домені. Вказівка, в яких зв'язках бере участь сутність, частково дозволяє зрозуміти і її значення, оскільки це надає можливість бачити, де дана сутність входить у відносини з іншим доменом.

Онтології грунтуються на математичному апараті формальної логіки (descriptive logic, DL), мале підмножина, якого охоплена RDF-схемою. DL є підмножиною логіки першого порядку, яке обчислюваних.

Додаткові можливості, вище зазначені, в додатку до наявних в RDF, є метою онтологічних мов, таких, як DAML + OIL [72, 73] і OWL [74, 75]. Дані дві мови засновані на RDF і RDF Schema. Мета даних мов - забезпечення ресурсів додаткової машинно-оброблюваної семантикою, тобто вони спрямовані на забезпечення машинного подання ресурсів у формі, який більш відповідає їх оригіналу з реального світу.

Розмітка документів Semantic Web, за допомогою онтологічних термінів, дозволить виробляти автоматичну обробку їх контенту. Таким чином, онтології визначаються, як ключова технологія для розвитку Semantic Web.

Онтології в змозі зіграти критично важливу роль в організації обробки знань на базі Web, їх загального використання та їх обміну між додатками.

Мова OWL. Найбільш розвиненою мовою представлення онтологій в даний час є OWL (Web Ontology Language), яка розширює можливості XML, RDF, і RDF Schema. Ця мова заснована на DAML + OIL. Проблеми, які виникли в DAML + OIL, були викликані постійною зміною ядра специфікацій RDF, на якому заснований DAML + OIL.

Як вказується в основному робочому проекті, OWL майже повністю схожий на DAML + OIL. Основні й істотні відмінності від DAML + OIL полягають у наступному:

- усунення деяких обмежень;
- здатність прямо вказувати, що властивість може бути симетричною;
- відміна деяких невикористовуваних конструкцій DAML + OIL, особливо обмеження з додатковими компонентами.

Існує також кілька незначних розбіжностей, які включають в себе деякі зміни імен деяких конструкцій, однак основна мета, яка ставилася при створенні OWL, полягала в тому, щоб максимально коректно зберегти імена DAML + OIL.

Онтологія OWL є послідовністю аксіом і фактів з додаванням посилань на інші онтології, які вважаються включеними в онтологію. Онтології OWL є Web-документами і на них можна посилатися. Онтології також мають не пов'язану з логікою компоненту (поки ще не визначену), що може бути використана для запису авторства, і інша не пов'язана з логікою

інформація, асоційована з онтологією. Фактично це словник, який розширює набір термінів, визначених у RDFS.
Онтології включають інформацію про класи, властивості і окремі випадки, кожен з яких може мати ідентифікатор ID, що є посиланням URI.
OWL має три модифікації:

- OWL Lite (простий);
- OWL DL (з повним доступом);
- OWL Full (з повною виразною потужністю).

Кожна з цих модифікацій (крім Lite) є розширенням попередньої. Як наслідок: будь-яка OWL Lite онтологія є OWL DL онтологією, а будь-яка OWL DL онтологія є OWL Full онтологією.
Головні характеристики мови веб-онтологій - OWL:

- OWL використовує синтаксис XML;
- OWL має інструкції для представлення дерева класів;
- OWL має інструкції для вказівки приналежності індивідів до класів;
- OWL має систему опису властивостей: область визначення, область значень;
- OWL може задавати характеристики властивостей: симетричність, транзитивність, функціональність;
- OWL має інструкції для вказівки еквівалентності (склеювання) класів.

Використання готової онтології дозволить розробникам, безпосередньо, приступити до заповнення даних та побудови шаблонів і дизайну. У разі відкритої публікації RDF-даних можлива реалізація програмних агентів для пошуку цих даних (наприклад, за допомогою спеціальних запитів системи Google), агрегація в єдиному сховищі та надання даних користувачеві (наприклад, абітурієнту) в єдиному інтерфейсі зі специфічними функціями. Можуть бути просто інтегровані дані підрозділів і представництв вузу, які просто редагуються редактором онтологій на місці, та імпортуються з основного веб-сайту цього вузу. У разі інтеграції досить великих і часто мінливих розподілених даних (наприклад, для агрегації інформації про конференції регіону з веб-представництв вузів і наукових організацій), можливе використання RDF-сховищ з відкритими інтерфейсами для вибірки тільки необхідних даних (наприклад, Joseki RDF Server[121])

1.7. Мови запитів до RDF сховищ

Говорячи про мови запитів, фактично мова йде про інтеграції різних мов (інформаційно-пошукових, баз даних, маніпулювання даними, обміну даними і т.п.) в єдину мову запитів Web. При цьому всі фахівці об’єднуються в думці, що це має бути декларативна мова, побудована на моделі не повністю структурованих даних (semistructured).
Документ "XML-QL: A Query Language for XML" [76] був підготовлений до семінару W3C по пошуковим мовам, який пройшов в кінці 1998 року і виявився далеко не єдиною спробою узагальнення такого роду.
В даний момент з'явилося декілька мов запитів до XML-джерел даних: XQL (1998) [77], XML QL (1998) [78 - 80]. Пошук в XML-документі полягає в знаходженні елементів, які задовольняють умови запиту, з подальшим перетворенням знайдених елементів у структуру, задану у запиті.
Мова запитів до RDF-джерел даних (RDF Query), запропонована в 1998 [81 - 85] і в даний час має вже практичну реалізацію в проекті Sesame [86].

У 2006 році консорціум W3C почав розробку мови запитів до RDF та OWL-сховищ - SPARQL Query Language for RDF, який зараз має статус рекомендованого кандидата (candidate recommendation) [87].
SPARQL - мова запитів, яка базується на патерну графів.
SPARQL одночасно є, як мовою запитів, так і протоколом доступу до даних, також SPARQL є одною з ключових компонент додатків Web 2.0: в якості стандарту, для підтримки гнучкої моделі даних, він дає загальний механізм запитів для всіх додатків Web 2.

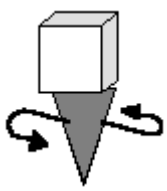
1.8. Принцип "логічного висновку"

Принцип "логічного висновку" дуже простий: це можливість виводити нові дані з даних, які вже є. В математичному сенсі, виконання запиту є однією з форм логічного висновку (наприклад, можливість вивести з маси даних, деякий результат пошуку). Логічний висновок є одним з провідних принципів Semantic Web, так як він дозволяє дуже легко створювати SW-програми [88].
Для того, щоб Semantic Web став досить виразним і зміг допомагати людям у різних ситуаціях, виникає необхідність побудови потужної логічної мови, яка підтримує логічний висновок. Дискусії, щодо методів, і навіть можливостей виконання цього завдання, до цих пір ведуться дуже активно; звертається увага на те, що в RDF недостатні можливості квантифікації, і що ця область визначена недостатньо добре. Проблеми логіки предикатів докладно розглянуті в базовій монографії Джона Сова (John Sowa's) «Математичні передумови (логіка предикатів)» - «Mathematical Background (Predicate Logic)» [89].
Rule Interchange Format (RIF) - формат обміну правилами. Мета якого розробляється консорціумом W3C стандарту [90] - визначення формату, який би дозволив транслювати правила між різними мовами і завдяки цьому забезпечити обмін правилами між системами, заснованими на правилах.
Системи, які грунтуються на правилах, одержали широке поширення в інформаційних технологіях. До їх числа відносяться, наприклад, експертні системи і системи дедуктивних баз даних. Розробки технологій Semantic Web забезпечують нове середовище використання таких систем. Тому консорціум W3C приділяє окрему увагу цій галузі. Специфікація RIF може розглядатися, як складова частина комплексу стандартів Semantic Web.
В даний час робочою групою, організованої за консорціумі для розробки цього стандарту, підготовлений, та обговорюється, робочий проект документа, який систематизує випадки використання RIF та вимоги до цієї мови. Найважливіша вимога до створюваного стандарту - забезпечення можливості його використання не тільки при поточному стані технологій, заснованих на правилах, але і його гнучкості, достатньої для забезпечення його використання в процесі їх еволюції.
Робочий проект документа, який описує випадки використання, дасть можливість визначити функціональні вимоги до RIF і на цій основі розробити адекватні специфікації мови.
Правила виведення нових фактів SWRL. Завдяки доповненню OWL мовою RuleML [91] (підмножина Datalog) у вигляді словника SWRL (A Semantic Web Rule Language) [92] з'явилася можливість використовувати діз'юнкти Хорна (Horn-like rules) для явної вказівки

способу виведення нових фактів з RDF-тверджень. Поки словник SWRL знаходиться в стадії стандартизації [93].
Хоча роботи над цим рівнем Semantic Web тривають, проте в нашому розпорядженні є вже достатній набір засобів для побудови Semantic Web: твердження, цитування (матеріалізація) у RDF, класи, властивості, області, документування у схемі RDF, непересічні класи, властивості однозначності та унікальності , типи даних, інверсії, еквівалентності, списки та інше.

1.9. Агенти та сервіси

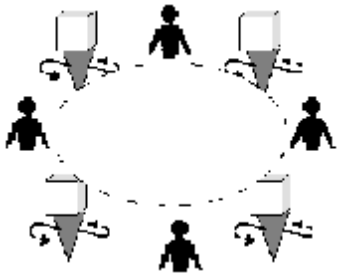
Провідну роль в Semantic Web повинні зіграти програмні агенти. При вище описаної архітектурі інформаційного простору, передбачається, що агенти, що володіють інтелектуальними здібностями, зможуть виконувати поставлені ним, користувачеві, цілі та завдання самостійно. Наприклад, з пошуку необхідної інформації, підбору та вибору оптимальних варіантів і т.п. Це в перспективі мобільні, інтелектуальні агенти, здатні до цілеспрямованості, планування, спільній взаємодії з іншими агентами для досягнення мети, що мають знання як про себе, так і про зовнішній світ. Для досягнення поставлених завдань вони повинні мати можливість користуватися деякими стандартними наборами послуг, представленими в Web в якості веб-сервісів. [123]
Відмінність між агентом та сервісом - один і той же сервіс може бути забезпечений різними агентами. [122]
Програмні агенти
Цифрова пам'ять не мала б ніякого значення, якби не існувало агентів, які забезпечують можливість швидкого пошуку та подання потрібної інформації. Кім Вельтман вважає використання,освітою, цифрових багатств однією з головних проблем найближчих десятиліть.
Хороші новини полягають у тому, що все більше і більше фондоутримувачів всього світу надають відкритий доступ хоча б до частини своїх колекцій через всесвітню павутину. .
Погані новини полягають у тому, що це поки що представляє дуже невеликий відсоток того, що є в самих музеях. Зробити багатства, що зберігаються в установах культури, доступними для освіти та досліджень, представляється одним з головних викликів найближчих десятиліть.[124]
Флуссер переконливо показує, що пам'ять не об'єкт, а процес і розглядати її слід не з точки зору того, що в пам'яті лежить, а з точки зору того, що з цим можна робити. .
Агенти це - помічники, яким ми довіряємо виконання певних завдань. Це - хтось, хто виконує інструкції. Слово агент походить від латинського слова agere - вести, діяти. Головне якість агентів - здатність виконувати якусь делегованих йому роботу в чиїхось інтересах. Надалі ми постійно будемо згадувати різних агентів, тому відразу ж позначимо їх зовнішній вигляд –



Політична коректність і ввічливість по відношенню до агентів вимагає, щоб ми відразу ж визначили та інших учасником комунікаційного процесу, яких в подальшому будемо

називати комп'ютерними користувачами, читачами, письменниками, а іноді й просто людьми - .

Цифрові помічники полегшили нам виконання безлічі розумових завдань і дозволили нам поглянути на процеси нашого мислення і нашого спілкування з іншими людьми з нової точки зору. Свідомість окремої людини все частіше розглядається як суспільство, в якому взаємодіє маса розумових агентів, кожний з яких вирішує певне завдання. Марвін Мінський написав про цю чудову книгу - Society of Mind. Крім того, людська культура все частіше (наприклад, у Хейлігена і Турчина) розглядається як величезна мережа, що складається з безлічі агентів - людських і електронних. Все частіше ми опиняємося в ситуації, коли наша розумова діяльність і наше спілкування з іншими людьми відбуваються за участю програмних агентів –

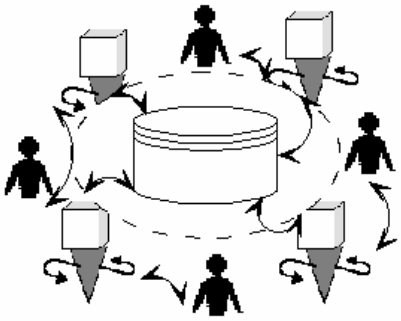


Круглий стіл, за яким сидять і люди, і програмні агенти, не просто «така метафора». Згідно роботі Рівса і НАСА люди схильні мислити програмні засоби та образи медіа в термінах простору або міжособистісних стосунків. Комп'ютер та комп'ютерні програми розглядаються і оцінюються нами так само, як і інші люди. Причиною цьому є навіть не стільки, зазначене Лаурелл [Laurel B. 1992, 1993] в запропонованої їй театральної метафори, бажання бути «обдуреним вимислом», скільки вироблена за тисячолітню еволюцію звичка сприймати кожен об'єкт, наділений активністю як жива істота і приписувати йому людські риси. Люди еволюціонували в світі, де найбільші проблеми і можливості їжі, покрівлі та різних небезпек були пов'язані з іншими людьми. Ціна помилки у відносинах людини з навколишнім світом була величезна. Приймеш змію за гілку - пиши пропало. Приймеш чоловіка за байдужого Колода - залишишся без чоловіка і т.д. У цих умовах серйозні переваги давала наступна стратегія - «Якщо існує хоча б низька ймовірність того, що інша тобі сутність є людиною, сприймай її як людини».

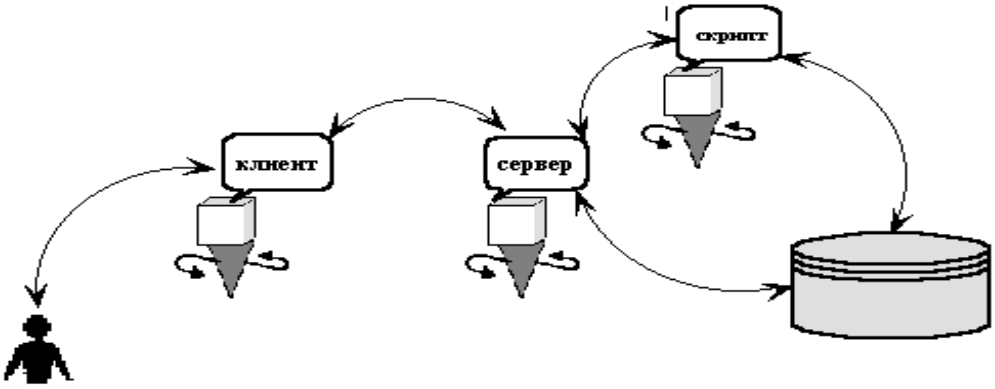
Цікаво, що розробники програмного забезпечення інколи переносять людські відносини не тільки на випадки взаємин «людина - програма», а й на відносини «програма - програма». Так, Ларрі Уолл дуже весело пише про терпимість програм:

«Люди легко розуміють, що найкраща політика для комп'ютерної програми, що взаємодіє з іншими програми, це бути як можна точніше і суворіше в тому, що ця програма передає іншим, і бути як можна вільніше і ліберальніші в тому, що ця програма приймає від інших. Дивність у тому, що люди не прагнуть бути суворіше до своїх власних висловлювань та ліберальніші до того, що вони чують ".

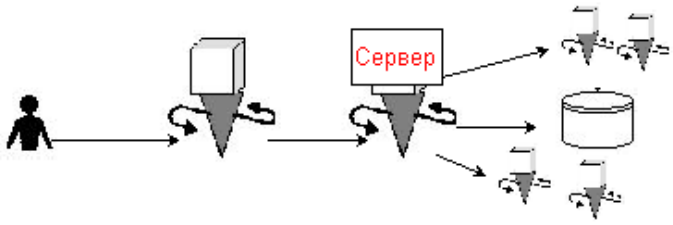
Продовжуючи користуватися метафорою круглого столу, за яким спілкуються люди і програмні агенти ми повинні додати на стіл смачного і цікавого змісту. Ми покладемо на стіл базу даних -



Існує стародавня притча «Вінчестер» з Silicon Life [125]. На день народження товариша Командо, якому подарували новий вінчестер. Зібралися всі, на столі лежить вінчестер, великий, листковий, змащення ще не висохло, і кожен свого шматка вінчестера чекає. Над ним командир Нортон в лівій руці тримає prefor, в правій, як учили в кадетському корпусі, - fdisk. - На скільки частин його різати товариш Командо? А той і відповідає: - На одну.
У мережевому співтоваристві для отримання інформації з бази даних людина звертається до програми клієнту, клієнт передає запит серверу, сервер додатковим програмами - скриптам, які спілкуються з базою даних. Схема цих непростих відносин представлена на наступному малюнку –



Клієнт - програма, якою потрібно поговорити з сервером. Читач використовує програму клієнт для того, щоб отримати доступ до інформації. Прикладами мережевих клієнтів можуть слугувати такі програми як Microsoft Explorer ? , Mozilla, Netscape, Opera. Оскільки вони допомагають людям перегортувати та проглядати, то їх називають - броузерами або браузерами. У чому полягають завдання такого клієнта, як браузер? Отримати запит від людини. Якщо запит про документ на локальній машині, то вважати вміст документа. Якщо запит про документ на віддаленій машині, то передати запит програмі-серверу на віддаленій машині. Отримати відповідь від програми-сервера. Передати на екран вміст запитуваної документа.
Як правило, браузер це - клієнт, разом з яким людина починає своє мережеве подорож. Важливо, що він не тільки допомагає переглядати інформацію, але і дозволяє людині запам'ятовувати пройдені стежки і створювати власні ланцюжка документів.
Сервер - комп'ютерна програма, яка підтримує розділений доступ до загального ресурсу або сервісу в мережі -



Часто сам сервер не займається нічим іншим, крім постійного очікування запитів до порту, які посилають клієнти. Веб-сервер постійно очікує вхідні запити з'єднання.
Веб-сервіс - це програмна система, що надає деяку послугу та забезпечує взаємодію по мережі. Зазвичай це веб-ресурс, що характеризується абстрактним набором функціональних можливостей, які в ньому реалізуються. Функціонально веб-сервіс може бути агентом, а може бути звичайною програмою.
Визначення веб-сервісу, подане в вікіпедії [126] - наступне: це «програмна система, що ідентифікується рядком URI, та публічний інтерфейс, визначений та описаний мовою XML. Опис цієї програмної системи можна бути знайдений іншими програмними системами, які можуть взаємодіяти з нею, відповідно до цього опису, за допомогою повідомлень, заснованих на XML, і переданих за допомогою інтернет-протоколів.
Архітектура веб-сервісів базується на компонентному підході, тобто, сервіс має бути досить автономним, а також, може складатися з декількох сервісів, що підбираються динамічно для виконання конкретного завдання у відповідності з різними критеріями.
Важливим аспектом при виборі сервісу є його доступність. Інтернет являє собою динамічне середовище, і питання доступності ресурсу або сервісу є дуже актуальним. При проектуванні композиції сервісів дуже важливо враховувати цей аспект.
Завдання побудови нових сервісів, з вже наявних, піднімає проблему синтезу сервісів. Для того, щоб скористатися послугами, повинна існувати можливість їх виявлення, механізм отримання інформації про те, які послуги вони надають, як до них звертатися, формат повідомлень. Рішенням цього завдання стало створення каталогів послуг за допомогою стандартних методів доступу. Сервіси повинні бути описані в стандартних термінах, а інформація про те, як до них звертатися і інша наявна інформація, повинна кодуватися стандартним способом.
Технологія веб-сервісів базується на наступних відкритих XML-стандартах:
SOAP (Simple Object Access Protocol) [94 - 100] - XML-протокол для віддаленого виклику методів, веб-сервісів;
UDDI (Universal Description, Discovery and Integration) [101] - описує модель даних, призначену для каталогізації та виявлення послуг, що надаються веб-сервісами;
WSDL (Web Service Definition Language (WSDL)) [102] - мова опису інтерфейсів веб-сервісів.
Додатки, що формуються до них, наприклад, WSCoordination / WS-Transaction (транзакції), WSSecurity (безпека), WS-Routing (маршрутизації повідомлень) і т.д., покликані розширити можливості цієї платформи, в задоволенні вимог завдань інтеграції додатків. У рамках ініціативи WS-I розробляються приклади прикладних рішень, пропозиції та додаткові вимоги, покликані гарантувати сумісність рішень різних

постачальників. Це обіцяє широкі можливості, по інтеграції різних інформаційних систем в рамках єдиного узгодженого набору специфікацій.
У багатьох випадках інтеграція інформаційних ресурсів вимагає комбінування звернень більше ніж до одного веб-сервісу, для реалізації користувальницького запиту. Таким чином, веб-сервіси повинні мати можливість підтримувати взаємодію з іншими додатками, на додаток до стандартних процедур обробки даних. Більш того, процес надання агрегованої розподіленої інформації, може включати в себе розбивка на набір взаємозв'язаних етапів обробки даних, взаємодія ряду веб-сервісів, втручання людей в процес обробки користувальницьких запитів і інші елементи прикладної логіки. Тому процес збору та інтеграції гетерогенних даних може являти собою логічно складну композицію звернень до сховищ інформаційних сутностей за допомогою інтерфейсів веб-сервісів - визначати автоматизований потік обробки даних.
Для опису композицій веб-сервісів, на даний момент, різними асоціаціями пропонується ряд стандартів. Серед них можна відзначити наступні мови опису автоматизованих потоків робіт, учасниками яких є веб-сервіси:
WSFL (Web Services Flow Language) - дозволяє визначати композиції веб-сервісів у вигляді графових моделей робочого процесу;
BPML (Business Process Modeling Language) - визначає блочну модель композиції веб-сервісів;
BPEL4WS (Business Process Execution Language For Web-Services) - являє собою гібрид блокової і графової моделі опису взаємодій веб-сервісів.
Ці мови дозволяють описувати композиції веб-сервісів, що дозволяє визначати складні, розподілені процеси по вилученню, обробці та інтеграції інформації.
Для вирішення таких складних розподілених завдань особливо добре підходить мульти-агентна технологія.
Як вже було вище сказано, для виконання конкретних завдань веб-сервіси повинні обмінюватися повідомленнями, повідомляти інформацію про себе і послуги, що надаються у вигляді, зручному, як для машинної обробки, так і доступному для розуміння людиною. Для вирішення цього завдання консорціумом були запропоновані мови мета-опису сервісів WSDL, а також онтологічна мова веб-сервісів OWL-S [103]. В даний час консорціумом запропонований проект мови моделювання сервісів - Service Modeling Language (SML) [104].
Зазвичай визначення агента полягає в тому, що програмний агент це програмна сутність, яка функціонує тривало і автономно в конкретному оточенні, часто - разом з іншими агентами. Агенти можуть бути спеціалізовані, вони повинні вміти спілкуватися з іншими агентами, з метою виявлення сервісів, продуктів, інформації або інших агентів. Сервіси, представлені в мережі, можуть бути реалізовані як агенти. Виникає проблема створення архітектури для взаємодії агентів, де б агенти могли описувати свої цілі з використанням заздалегідь визначених словників, де можливо було б виробляти пошук і підбір необхідних сервісів і інформаційних ресурсів, а також використовувати багато інших можливостей.

1.10. Практична реалізація Semantic Web

Технологія Semantic Web на даний час успішно вирішує наступні задачі:

- незалежність даних від програм;
- семантична інтеграція даних;
- створення основи для повсюдного використання комп'ютерних агентів (сервісів).

Формування Semantic Web стане можливим тільки за умови забезпечення більш високого рівня інтерперабельності. Проте вже зараз зроблено багато практичних кроків з реалізації даного проекту. Новий проект на базі пошукової системи Google недавно надав свої ресурси, для запитів, агентам, на виконання пошукових функцій та перевірки правопису [105]. Також представляє інтерес новий проект з автоматичного створення RDF-описів і сховища метаданих, що створюється на основі Open Directory [70] пошуковим механізмом Google [106]. Крім того, необхідно також відзначити і проект консорціуму W3C SWAD-Europe [107], який займається проблемою зв'язку сховищ семантичних даних з використовуваними реляційними системами баз даних, особливо ліцензованих, такі як Free Software / Open Source (FS / OS).

В даний час необхідно констатувати, що загальний обсяг мета-інформації досяг вже критичної маси і неухильно зростає. На вересень 2006, простору імен OWL, були використані в 113 000 документах Semantic Web (це 8% загального обсягу), простір імен RDFS оголошено в 677 тисяч (47%). Owl: Class є найбільш використовуваним термом з простору імен OWL, він використовується в 1 800 000 висловлюваннях з 68 000 документів. У серпні 2007 року в мережі налічувалося понад 2 більйонів RDF-трійок [32, 54, 108, 109].

Інтерес до використання даної інформації також постійно підвищується. На березень 2006 року [108] з аналізу запитів пошукової системи Google, видно, що звичайними рядовими користувачами, була Дисертація на здобуття 2120000 запитів до типу "RDF filetype: rdf" і 13 600 "ontology filetype: owl". Такі цифри говорять про популяризації ідей Semantic Web і дає можливість вже реально починати використовувати дану мета-інформацію в прикладній сфері.

Подальшому розвитку Semantic Web сприяє наявність вільно розповсюджуваних систем, для розробки додатків Semantic Web:

Jena Framework (Java);

Drive RDF Parser (C #).

В даний момент вже існують:

- бібліотеки для інтерпретації стека, таких мов, як RDF, для всіх популярних мов програмування (Jena, Redland, RDFLib);
- редактори онтологій (Protege);
- системи міркувань над онтологіями (Racer, KAON, FACT);
- семантичні сховища (Sesame, Kowari, YARS);
- семантичні браузери (Simile, Piggy Bank, Gnowsis, Haystack);
- «пошуковими» семантичних даних (Swoogle);
- конвертори з різних форматів представлення даних в / з RDF / XML (Aperture, RDFizers, D2R);
- прикладні програми (Bibster, FOAF Explorer).

Також необхідно вказати і існуючі комерційні продукти: Adobe's XMP - інструментарій для створення мета опису про файли;
Oracle's 10.2 Database - вже має вбудовану підтримку моделі RDF; Tucana's Knowledge Discovery Suite - платформа для інтеграції інформації застосувань (Enterprise Information Integration, EII)
На останній VI міжнародної конференції з Semantic Web - Sixth International Semantic Web Conference, яка проходила 11-15ноября 2007 в Кореї [109], позначено таке положення справ у напрямку поширення Semantic Web:
відзначилося різке зростання і виникнення компаній, що використовують технологію Semantic Web (Joost, Radar Networks, MetaWeb, Siderean, SandPiper, SiberLogic, Ontology Works, Intellidimension, Intellisophic, TopQuadrant, Data Grid, etc.);
відбулося залучення великих постачальників ПЗ - Adobe, Cisco, HP, Microsoft, Nokia, Oracle, Sun, Vodaphone;
активно розвиваються урядові програми - у США, Об'єднаної Європи, Японії, Кореї, Китаї;
сильно зріс такий важливий ринок, як медико-фармацевтичний - створена спеціальна група
при консорціумі Health Care and Life Sciences Interest Group at W3C;
з'явилося багато інструментів з відкритим кодом - Kowari, RDFLib, Jena, Sesame, Protégé, SWOOP, Onto (xxx). Wilbur.

На цій конференції Semantic Web розглядався, як колекція всіх формальних, машинооброблюючих, доступних в Web, заснованих на онтологіях тверджень (семантичних метаданих) про веб-ресурсах та інших сутності світобудови, виражених мовою представлення знань, заснованому на синтаксисі XML (наприклад, OWL, DAML , DAML + OIL, RDF, etc.). Необхідно констатувати, що в Web вже представлено досить велику кількість такої інформації. Все більше постає проблема її обробки, об'єднання, вирівнювання, виявлення зв'язків.
З 2003 року щорічно проводиться всесвітній конкурс Semantic Web Challenge [110], покликаний зібрати самі останні напрацювання і показати світу стан справ, щодо практичної реалізації ідей Semantic Web. При цьому був сформульований наступний перелік мінімальних критеріїв, що визначають поняття «додаток Semantic Web».
По-перше, програма має використовувати інформаційні джерела, які:
географічно розподілені;
мають різних власників, що передбачає відсутність контролю за їх розвитком;
є гетерогенними (синтаксично, структурно, і семантично);
містять дані реального світу, тобто джерела повинні бути більше, ніж іграшкові приклади.
По-друге, програма має сприймати відкритий світ; це означає, що воно знає, що інформація ніколи не буває повною і постійно змінюється.
По-третє, програма має використовувати деякий формальний опис значення даних.
Крім цих мінімальних критеріїв, були визначені кілька бажаних якостей. Додаток повинен використовувати джерела даних в інших цілях або по-іншому, ніж спочатку було намічено. Воно також повинно використовувати контент мультимедійних документів. Користувачі повинні бути в змозі отримати доступ до додатка на безлічі мов або з інших,

відмінних від PC, пристроїв. Додаток повинен використовувати як статичні, так і динамічні знання, наприклад, комбінація статичних онтологій і динамічних технологічних процесів. Нарешті, програма має бути масштабованим (в термінах кількості використовуваних даних і спільно працюючих розподілених компонент).
Підсумки змагання між представленими проектами, щорічно підсумовуються на Всесвітній конференції з Semantic Web, де обговорюються наукові рішення і проблеми, що виникли на даному етапі розвитку Semantic Web. На останній VI конференції 2007 р. в Кореї було виділено 2 покоління додатків Semantic Web [111]. Перше покоління - Семантично прив'язані програми Semantic Web - Semantically Closed SW Applications. Ці програми використовують єдину онтологію, дуже прив'язані до семантичним ресурсів, обмежені в інтерактивності. Такі програми надають однорідне подання гетерогенних джерел даних і дуже обмежено використовують існуючі в Semantic Web дані. Існуючі на даний момент програми Semantic Web більше схожі на традиційні системи, орієнтовані на знання.
В даний час постає завдання створення додатків другого покоління. Друге покоління додатків Semantic Web повинні використовувати весь величезний запас вже накопиченої семантики. Програми Semantic Web 2-го покоління повинні мати здатність використовувати:

- безліч онтологій;
- бути відкритими для семантичних ресурсів;
- бути відкритими для роботи з користувачем (user interaction).

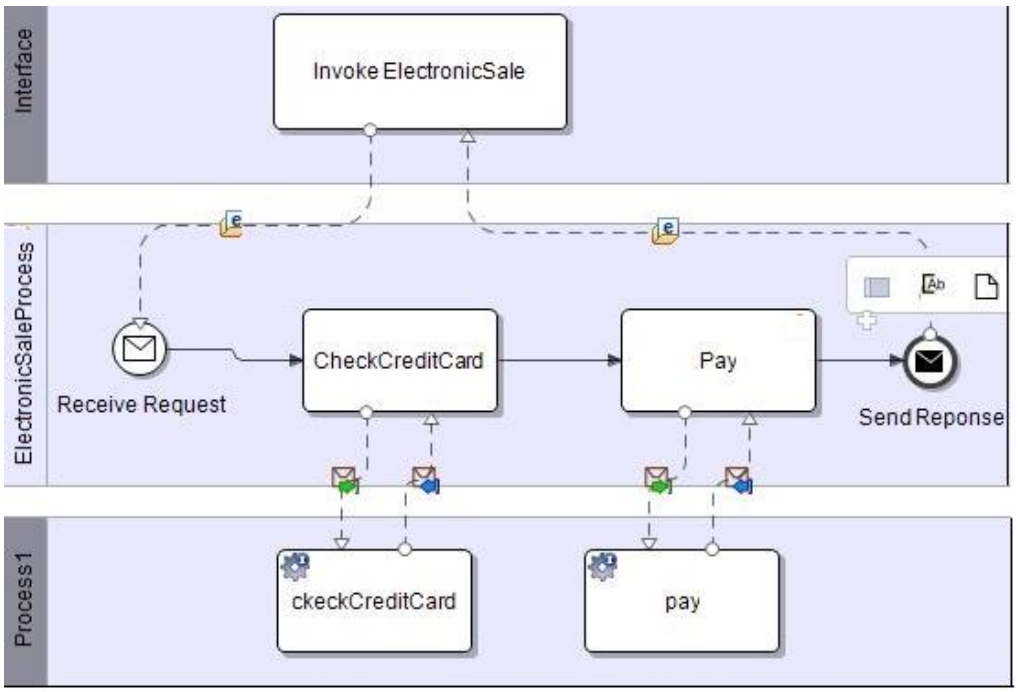
В ідеалі вони також повинні вміти використовувати не тільки дані Semantic Web, але й інші формати даних, наприклад, фолксономії тощо, отже повинні мати потужні механізми з автоматичного вилучення інформації.
Також на цій конференції було показано, як Semantic Web пропонує вирішення проблеми об'єднання даних, а також практичні результати цієї роботи.
Результати VI конференції з Semantic Web показали, що:
більшість з подій, які були припущені, здійснилися, або здійснюються в даний момент, темпи цього руху прискорюються;
деякі досягнення відбуваються швидше, ніж планувалося раніше (масове зростання RDF-сховищ, подання міркувань, наявність онтологій - але дуже погано зв'язаних);
деякі плани поки слабо реалізуються, але рух у цих напрямках продовжується (публічні джерела інформації RDF, OWL, зародження «всепроникних» обчислень);
слабкий розвиток технології агентів [108].

В якості прикладу можна розглянути семантичний пошук. (http://sindice.com/) та порівняємо метод, за якими цей самий пошук здійснюється.

Sindice є сучасної інфраструктури для процесу консолідації та запитів в Інтернеті даних. Sindice зіставляє ці мільярди штук метаданих в узгоджену «парасольку» функціональних можливостей і послуг.
Google, в свою чергу, забезпечує пошук по гіпертекстовим документам, які перебувають у будь-яких мовних зонах - англійській, російській, українській, німецькій та ін Пошукова

система Google має власні піддомени для більшості країн, наприклад, для України - http://www.google.com.ua.
Тобто, з цього вище написаного можна зробити висновки і сказати, що: Sindice, по суті, шукає логічно зв'язані дані і цілі блоги шуканої інформації, так, як Google здійснює пошук по ключовим словам.

А в якості композиції веб-сервісів розглянемо приклад нижче:
Моделювання поведінки композитних веб-служб
В перспективі, вибираємо для позначення BPMN (Business Process Modeling Notation (Моделювання бізнес-процесів)). Цей новий стандарт більш зручний для оркестровки бізнес процесів, ніж діаграм UML діяльності. BPMN, і це є шанс стати новим стандартом для моделювання бізнес- процесів і веб-послуг [5]. Бізнес-процеси, розроблені з використанням стандартних BPMN, можуть безпосередньо зіставляється з будь-яким бізнес-моделюванням виконуваної мови і піддатися негайному виконанню.
Обидва стандарти: BPEL4WS (Business Process Execution Language for Web Services) і BPML (Business Process Modeling Language) забезпечує специфікацію для потоків даних, повідомлення, подій, бізнес правил, винятків та угод. Оркестровки веб-служб можна визначити, як процес діяльності складових бізнес-процесу. Контроль таких конструкцій, як послідовність, паралельний розкол, синхронізація, ексклюзивний вибір і кілька варіантів, а також повідомлення та події, стають бути розглянуті. Зокрема, виконання композитних веб-служб може потребувати послідовного виконання певних компонентів. Інший компонент може, наприклад, виконуватися паралельно в залежності від терміну дії заданій умові. У випадку складеної веб-служби для подорожей, ця послуга може забронювати квиток на певну дату. Якщо замовлення підтвердилося, паралельне виконання послуг готелю - бронювання автомобіля, воно може виникнути автоматично.
На малюнку «а)» показано, яким чином працює модель поведінки композитних ElectronicSale служб за допомогою BPMN. Ця робота проводилася за допомогою Intalio дизайнер інструментів. Поведінка послуги ElectronicSale моделюється послідовно: оркестровками послуг CardValidate та електронних платежів.
Ця послідовність залежить від правильності даної карти.



Мал.. а) BPMN моделі композитних веб-служб ElectronicSale

Коли крок моделювання поведінки досягнутий, це означає, що можна створювати SAWSDL і BPEL файлів.
Витяги з SADWSL BPEL та вміст файлів, пов'язаний з композитними ElectronicSale веб-службами, приведеними нижче, мал. «б)» «в)». Файл BPEL дозволяє серверу виконати композитні веб-служби.
Де SAWSDL - це розширенням синтаксичного опису WSDL [128], а BPEL (Business Process Execution Language) - мова на базі XML для формального опису бізнес-процесів і протоколів їх взаємодії між собою. BPEL розширює модель взаємодії веб-служб і включає в цю модель підтримку транзакцій. [129]

```
<wsdl:import namespace="http://example.com/ElectronicSale/
      ElectronicSaleProcess"
      location="ElectronicSale-ElectronicSaleProcess.wsdl"/>
<wsdl:import namespace="http://ws.intalio.com/CardValidate/"
      location="Service/CardValidate.wsdl"/>
<wsdl:import namespace="http://ws.intalio.com/ePayement/"
      location="Service/ePayement.wsdl"/>
<pnlk:partnerLinkType
      name="ElectronicSaleProcessAndProcesslForPortTypeCardValidateSoap
      <pnlk:role name="Processl_for_ElectronicSaleProcess"
      portType="tns:CardValidateServiceSoap"/>
</pnlk:partnerLinkType>
<pnlk:partnerLinkType
      name="ElectronicSaleProcessAndProcesslForPortTypeePayementSoapPlk
      <pnlk:role name="Processl_for_ElectronicSaleProcess"
      portType="tns:ePayementServiceSoap"/>
</pnlk:partnerLinkType>
<pnlk:partnerLinkType name="ElectronicSaleProcessAndInterface">
      <pnlk:role name="ElectronicSaleProcess_for_Interface"
      portType="ElectronicSaleProcess:ForInterface"/>
</pnlk:partnerLinkType>
```

б) Витяг зі змісту файлу SAWSDL, пов'язаного з веб композитними послугами ElectronicSale.

```
<bpel:variables>
<bpel:variable name="thisReceive_RequestRequestMsg"
      messageType="this:Receive_RequestRequest"/>
<bpel:variable name="thisReceive_RequestResponseMsg"
      messageType="this:Receive_RequestResponse"/>
<bpel:variable name="tnsCkeckCreditCardRequestMsg"
      messageType="tns:checkCreditCardpIn"/>
<bpel:variable name="CardValidateServiceCkeckCreditCardResponseMsg"
      messageType="Processl:ckeckCreditCardResponse"/>
<bpel:variable name="tnsPayRequestMsg"
      messageType="tns:paySoapIn"/>
<bpel:variable name="tnsPayRequestMsg"
      messageType="tns:payeSoapIn"/>
<bpel:variable name="ePayementPayResponseMsg"
      messageType="Processl:payResponse"/>
<bpel:variable name="ePayementPayResponseMsg"
      messageType="Processl:payResponse"/>
</bpel:variables>

<bpel:sequence>
<bpel:receive
      partnerLink="electronicSaleProcessAndInterfacePlkVar"
      portType="this:ForInterface" operation="Receive_Request"
      variable="thisReceive_RequestRequestMsg" createInstance="yes"
bpmn:label="Receive Request" name="Receive_Request"
bpmn:id="_iS9B8NIXEd6NDptfMmnqFg"></bpel:receive>
```

в) *Витяг зі змісту файлу BPEL, пов'язаного з веб композитними послугами ElectronicSale..*

2.Представлення знань

Для того, щоб семантична мережа могла функціонувати, комп'ютери повинні мати доступ до структурованих сховищ інформації та множинам правил виведення, які вони могли б використовуватися для проведення автоматичних міркувань. Дослідники у галузі штучного інтелекту займалися вивченням подібних систем задовго до виникнення мережі. Представлення знань, як часто називають цю технологію, в даний час знаходиться в такому ж самому стані, в якому, було поняття гіпертексту до появи мережі: ідея, безсумнівно, дуже хороша, і вже існують досить хороші досвідчені зразки, але поки ще світ ця ідея не змінила. У неї вже є зачатки та основи важливих додатків, але щоб реалізувати весь її потенціал, необхідно пов'язати її в єдину глобальну систему.

Традиційні системи подання знань часто були централізованими, вимагаючи, щоб всі використовували в точності одні й ті ж визначення загальних понять, як то «батько» або «автомобіль». Але подібний контроль є занадто стримуючим, і у міру зростання розмірів і масштабів такої системи вона досить швидко стає неконтрольованою.

Більш того, в таких системах, зазвичай, завбачливо обмежують коло тих питань, які можна їй задати, для того, щоб комп'ютер був в змозі дати на них достовірну (або хоча б яку-небудь) відповідь. Ця проблема дуже нагадує теорему Геделя з математики: будь-яка система, що досить складна, щоб бути хоч якось корисною, обов'язково повинна містити питання, на які в принципі неможливо дати відповідь; останні дуже схожі на ускладнені версії найпростішого парадоксу: «Ця пропозиція помилкова ». Щоб уникнути подібних проблем, будь-яка традиційна система подання знань, як правило, намагається обмежитися досить вузьким і характерним для неї набором правил для побудови висновків з наявних у них даних. То приміром, генеалогічна система, що працює з базою даних родоводів, може включати в себе таке правило: «дружина дядька є тітка». При цьому навіть якщо дані і можна було б перенести з однієї системи в іншу, то правила, які самі по собі існують у зовсім іншому вигляді, на відміну від даних, вже зазвичай перенести не вдається.

Дослідники ж в області семантичної мережі, навпаки, допускають такі парадокси і питання, що не мають відповіді, як ціну за досягнення гнучкості. Мова, на якій передбачається формулювання правил виведення, спочатку створюється настільки виразна, щоб дозволяла мережі користуватися міркуваннями як можна ширше. Філософія тут схожа з тією, що застосовується у звичайній мережі: ще на зорі її розвитку скептики вказували на те, що вона ніколи не зможе стати чітко організованою бібліотекою, а дехто, не маючи централізованої бази даних і структури у вигляді дерева, не зможе бути впевненим , що в ній взагалі щось можна буде відшукати. І вони мали рацію. Однак виразна сила цієї системи зробила цілком доступним гігантську кількість інформації, і пошукові служби (які здавалися майже нездійсненними всього якийсь десяток років тому) зараз пропонують нам «дивно повні» каталоги величезної кількості матеріалу по всій мережі. Таким чином, ціль семантичної мережі - створити мову, на якій можна буде описувати як дані, так і правила міркувань про ці дані, так щоб він дозволяв правила виведення, що існують у якій-небудь одній системі подання знань, передавати по мережі інших подібних систем.

Принести в мережу логіку (як то: способи застосування правил виводу для проведення міркувань, методи вибору тактик виконання операцій з даними і засоби для відповідей на запитання) - ось те завдання, яке стоїть перед спільнотою семантичної мережі зараз.

Комбінування існуючих математичних та інженерних рішень ускладнює це завдання. Ця логіка має бути, з одного боку, досить сильною, щоб дозволяти описувати складні властивості об'єктів, а з іншого - не на стільки сильною, щоб агента можна було поставити в глухий кут, давши йому парадоксальний запит. На щастя, переважна більшість інформації, яку ми хочемо висловити, являє собою щось на кшталт «шестигранний болт є типом машинних болтів», що без праці вписується у вже існуючі мови, розширені деякими додатковими мовними конструкціями.

Зараз вже створено дві важливі технології для розвитку семантичної мережі: розширювана мова розмітки (eXtensible Markup Language, XML) і Система Опису Ресурсів (Resource Description Framework, RDF). [Прим. Також з'явився Мова Мережевих Онтологій (Web Ontology Language, OWL), якому 10 лютого 2004 WWW-Консорціум (W3C) присвоїв статус рекомендованої до реалізації технології. Дехто вже пропонує вважати цю дату офіційним днем народження Семантичної мережі. Мова XML дозволяє створювати свої власні теги - приховані мітки типу <zip code> [поштовий індекс], або <alma mater> [закінчений університет або коледж], якими можна постачати веб-сторінки або розділи тексту на сторінках. Скрипти та програми можуть використовувати ці теги самим хитрим чином, але при цьому програміст, що пише ці скрипти, повинен знати, для чого автором веб-сторінки використовується той чи інший тег. Коротше кажучи, мова XML дає можливість користувачам постачати свої документи довільної структури, проте дана мова нічого не говорить про те, що означає ця структура.

Сенс виражається за допомогою мови RDF, яка кодує його за допомогою безлічі триплетів, де кожен триплет складається з суб'єкта, дієслова і об'єкта елементарної пропозиції. Такі триплети можна записати за допомогою тегів мови XML. У мові RDF документ складається з тверджень про те, що щось (людина, веб-сторінка або що-небудь ще) має певне відношення (як то «бути сестрою», «бути автором») з деяким певним значенням (інша людина, інша веб -сторінка). Подібна структура виявляється досить природною для опису переважної більшості машинно-оброблюваних даних. Суб'єкт і об'єкт задаються за допомогою однакового Ідентифікатора Ресурсу (Uniform Resource

Identifier, URI), подібно посилань на веб-сторінках. URL - Універсальний Локатор Ресурсу (Universal Resource Locator) - являє собою найбільш поширений тип URI. Дієслова теж задаються за допомогою URI, що дозволяє визначати нове поняття або новий дієслово, просто вказавши його URI-адресу в мережі.
Людська мова процвітає завдяки тому, що одне і теж слово може мати кілька значень; але це зовсім не так для мови машинного світу. Уявіть собі, наприклад, що я наймаю клоунів-кур'єрів для доставки повітряних кульок моїм клієнтам на їх дні народження. Зовсім не до речі, ця розважальна служба перекачає мою базу даних з адресами клієнтів собі, не знаючи, що «адреса» у моїй базі даних - це те місце, куди доставляються рахунки, і що більшість з них - абонентські скриньки в поштових відділеннях. У підсумку мої клоуни повеселять поштових працівників - що саме по собі, можливо, не так вже й погано, але, очевидно, це не те, чого хотілося спочатку. Подібна проблема вирішується використанням різних URI для кожного конкретного поняття. Поштова адреса: тоді можна буде відрізнити від адреси проживання, і обидва ці поняття, у свою чергу, можна буде відрізнити від поняття «адресувати мову кому-небудь».
З триплетів мови RDF формуються мережі інформації про взаємопов'язаних речах. Оскільки RDF використовує URI-ідентифікатори для кодування даної інформації в документі, ці самі URI-ідентифікатори гарантують те, що кожне поняття, що використовується в документі - це не просто слово, а щось, прив'язане до єдиного визначення, яке кожен бажаючий може знайти в мережі. Наприклад, уявімо собі, що у нас є доступ до декількох баз даних про людей, що містить їх адреси. Якщо тепер ми хочемо знайти тих людей, які живуть у районі з якимось заданим поштовим індексом, то нам потрібно буде знати, яке саме поле в кожної з баз даних являє собою ім'я, а який - поштовий індекс. Це можна висловити на мові RDF у вигляді: «(поле 5 в базі даних A) (є полем типу) (поштовий індекс)», використовуючи URI-ідентифікатори замість слів для кожного терміна.

3.Linked Data в середовищі Semantic Web

Linked Data Project створює загальновживані ієрархії класів, словники прозивним і власних імен, а також допомагає власникам масивів даних об'єднувати їх бази знань в одну зв'язкову систему знань. У багатьох випадках учасники проекту об'єднують вже наявні великі бази, допомагаючи один одному встановити відповідність між ідентифікаторами однієї і тієї ж речі в різних базах. Якщо якась база знань заповнюється деякої автоматичної процедурою, то ця процедура може почати використовувати імена, вже використовувані іншими учасниками, якщо вона пишеться вручну, то автори можуть заглядати в DBpedia, GeoNames, WordNet або Yago як до словника, одночасно з цим перетворюючи свої дані в "замітки на полях" великої енциклопедії. У виграші всі --- і автори невеликих баз, складачі великих словників. Наявність перехресних зв'язків між різними базами знань не тільки робить ці бази більш корисними --- часто самі знання очищаються від помилок.

Навесні 2009 року сумарний обсяг баз проекту склав 4.5 гігаквада, і зараз проект знаходиться в майже некерованою, але дуже захоплюючої фазі експоненціального зростання. (Десять днів пролежала стаття без руху - і вже 4.7). Росте не тільки обсяг бази - одночасно зростає і кількість запитів до бази. Це створює цікаві проблеми для OpenLink, тому як саме ми надаємо SPARQL-доступ до основних ресурсів проекту. На ранніх етапах проекту - брали участь у створенні, скажімо, YAGO. На даний момент – знання у проект не додається, тільки лише безперервно удосконалюється OpenLink Virtuoso Universal Server, покращуючи його масштабованість, і плавно нарощується обчислювальна потужність. Якщо буде якась заминка в розвитку , то веб-сервіс здохне під навантаженням. Якщо зробити щось "таке собі", то отримаєм принципово кращу масштабованість - постачальники великих баз даних будуть раді відкрити доступ до масивів, на порядки більшим, ніж весь нинішній LOD, і гонка продовжиться. Одна тільки Ordinance Survey, геодезична служба Її Величності, оцінює доступний об'єм знань про одні тільки Англії та Ірландії в один петаквад. В планах є ціль надати їм всю необхідну для цього інфраструктуру, паралельно розширюючи можливості мови запитів.

Перш, ніж продовжувати хвалитися, хотілося б пояснити мету цього хвастощів стосовно до російських умов. Очевидно, що Росія в найближчі роки не буде великим учасником цього "горизонтального" "загальновживаного" проекту і йому подібних. В минулому році сталася знаменна подія - з'явився президент, який вміє користуватися браузером, всього лише на два президентські терміни пізніше, ніж варто було б. При такому Лаге серйозних грошей на вітчизняний сем-веб не буде, як мінімум, ще три президентські терміни. Не буде навіть, якщо це прорубує здоровенну дірку в обороноздатності країни, бо потреби суми занадто малі порівняно, скажімо, з виробництвом авіатехніки. За частку в цих дрібних гроші ні один лобіст НЕ свербіло. Також очевидно, що Академія Наук у її нинішньому вигляді не зможе виступити ініціатором вертикальних галузевих проектів --- немає ні вільних кадрів в Академії, ні замовників у промисловості. Будуть невеликі локальні проекти, на голому або майже голому ентузіазмі. Що спільного між майбутньою інфраструктурою цих проектів і великими проектованими кластерами LOD? Та найголовніше - загальні проблеми з бюджетом, загальні закони фізики.

Спочатку про «наболіле». Ні повісті сумнішої на світі, ніж повість про запити і бюджеті. OpenLink - невелика приватна компанія, всього 50 чоловік. При цьому бази знань - зовсім не основний наший напрямок. Ми давно є міцним лідером у виробництві кросс-

платформенних драйверів баз даних, брокерів запитів та іншого RDBMS middleware. Компанію вже двадцять років годує саме це, а зовсім не семвеб. LOD --- це робота в надії на світле майбутнє, але при цьому кожний ящик в серверну --- це гроші, з кров'ю видер із бюджету якогось іншого проекту вже зараз. Крім того, всі питання з цієї тематики минуть звичайну тех. підтримку і потрапляють безпосередньо до провідних розробників, у доважок до основної роботи. Так що ми кровно зацікавлені в тому, щоб ПЗ не вимагало кваліфікований адміністрування і працювало на "комодах", тобто на серверах, зібраних в домашніх умовах з недорогих залізяк. Зокрема, ми використовуємо дешеві диски, відносно дешеві планки пам'яті, не самі швидкі процесори, і ми до останнього будемо уникати між-машинних з'єднань дорожче Gigabit Ethernet. Мені здається, що такі технічні обмеження підтримають багато вітчизняних бригади, яким абсолютно необхідно, щоб система коштувала мало, обробляла багато, обслуговувалася раз на тиждень ледачим студентом і при цьому дзижчала одна на весь інститут, тому як на другу грошей вже точно не буде. Таку річ ми і пишемо. Поки виходить.

Загальні закони фізики гарантують однакові для всіх проектів пропорції між продуктивністю процесорів з одного боку і швидкістю і латентністю міжмашинна обміну. Скільки б не коштувала мережева карта, сигнал по провіднику йде приблизно зі швидкістю одна ширина кристала процесора за такт цього процесора. Дюйм за такт, если наглядно. Дюйм за такт, якщо наочно. Параметри жорстких дисків теж досить близькі, в силу схожості габаритів і матеріалів. Схожі і основні тимчасові характеристики використовуваних операційних систем, у тому числі затримка втрата часу на перемиканні нитки, якщо нитка змушена чекати мьютекс, час входу в мьютекс "без перешкод", час обміну данимі з драйвером мережевої карти і т.п. Будь-яка "важка" операція, будь то некваплівий системний виклик або посилка байти кудись на периферію буде коштувати в сотні а то й сотні тисяч разів більше, ніж крок інтерпретації запиту. Якщо паралелізм хороший, а обмін між процесами спланований вірно, то важких операцій буде мало, і процесори не будуть простоювати в очікуванні --- готових до роботи ниток вистачить усім. Якщо паралелізм поганий --- ляже кластер будь-якого розміру і ціни. При цьому добре розшифрувати можна тільки первісно акуратний код, відомий розробнику зверху до низу. Це як раз наш випадок. Віртуоза краще за всіх не тому що ми самі розумні, а тому що вона у продажу з 1995-го року, у нас фора в десять з гаком років.

"Що вірно для бактерії, то вірно для слона" --- стара жарт генетиків. У нас імплікація в інший бік. Що працює для LOD --- запрацює і на парі ящиків під столом ентузіаста-одиначки.

Розповідь про бочці меду доречно почати з ложки дьогтю. Наш хостинг LOD не позбавлений обмежень. Розглянемо найважливіші з них: обмеження на час виконання запиту, заборона на використання реляційних джерел даних, заборона на матеріалізацію видів.

Обмеження на час виконання запиту.

SPARQL-запити очевидно більше виразні, ніж, скажімо, повнотекстовий пошук. Можна легко написати скільки завгодно трудомісткий запит, особливо помилково. Негайно виявилася проблема --- сервіс зможе виконати безліч нескладних пошуків або відповісти на значно менша кількість більш "розумних" питань. Що корисніше для суспільства --- незрозуміло. У результаті ми дозволяємо будь-які запити, але обмежуємо їх за часом виконання. Більше того, ми їх обмежуємо двічі. Спочатку ми відкидаємо без спроби

запуску ті запити, для яких компілятор видав дуже велику оцінку вартості. Потім ми обриває виконання "занадто замислених" запитів за реальним часу. Це не виключає деякої "нечесності" --- оцінка компілятора може виявитися несправедливо завищеною, наскільки одночасно запущених складних запитів одного користувача можуть забити всю пам'ять, привести до безперервної підкачування і тим самим вбити всі запити --- і хороші і погані, але по крайньою мірою система має хороший виховний ефект --- погані запити вбиваються завжди, відбиваючи полювання їх задавати.

Ті, кому дійсно треба виконувати складні запити можуть, зрозуміло, створити свою базу, завантажити потрібне підмножина LOD і своїх даних, і робити що завгодно. Або заощадити час і сили, орендувавши точну копію нашої бази на хмарі Amazon.

Заборона на використання реляційних джерел даних.

Одне з базових обмежень інфраструктури LOD --- використовується тільки "чистий" RDF, навіть якщо вихідні дані доступні в іншому поданні, скажімо, у вигляді реляційних даних. При цьому ми зумисне позбавляємо себе одного з найважливіших своїх інструментів. Справа в тому, що Virtuoso дозволяє описати відображення реляційних даних в RDF, і потім транслювати SPARQL-запити до цього придуманого RDF в ефективні SQL-запити до реальних таблиць. Такі відображення називаються RDF Views, і їх використання звичайно дозволяє виграти у продуктивності в порівнянні з SPARQL-запитами над дійсно експортованими даними. Виграш досягається за рахунок правильного використання індексів, специфічних для конкретних даних і конкретного застосування. Крім того, зникає весь набір проблем, пов'язаних з підтриманням актуальності RDF-копії реляційних даних. Больших, доложу вам, проблем. Великих, доповім вам, проблем. І тим не менше, ми в разі LOD миримося з цими проблемами і при цьому жертвуємо потенційним виграшем в продуктивності. Це пов'язано з тим, що час компіляції деяких запитів зростає як поліном від числа RDF-видів. Це не є непереборною проблемою для десятків або навіть сотень відображень, чого достатньо для корпоративних додатків, але могло б стати блокуючою перешкодою для LOD з його безперервно зростаючими різноманітними даними.

Знову-таки, охочі можуть використовувати будь-які схеми зберігання, це наше приватне рішення для даного конкретного проекту, а не якийсь фундаментальний заборону.

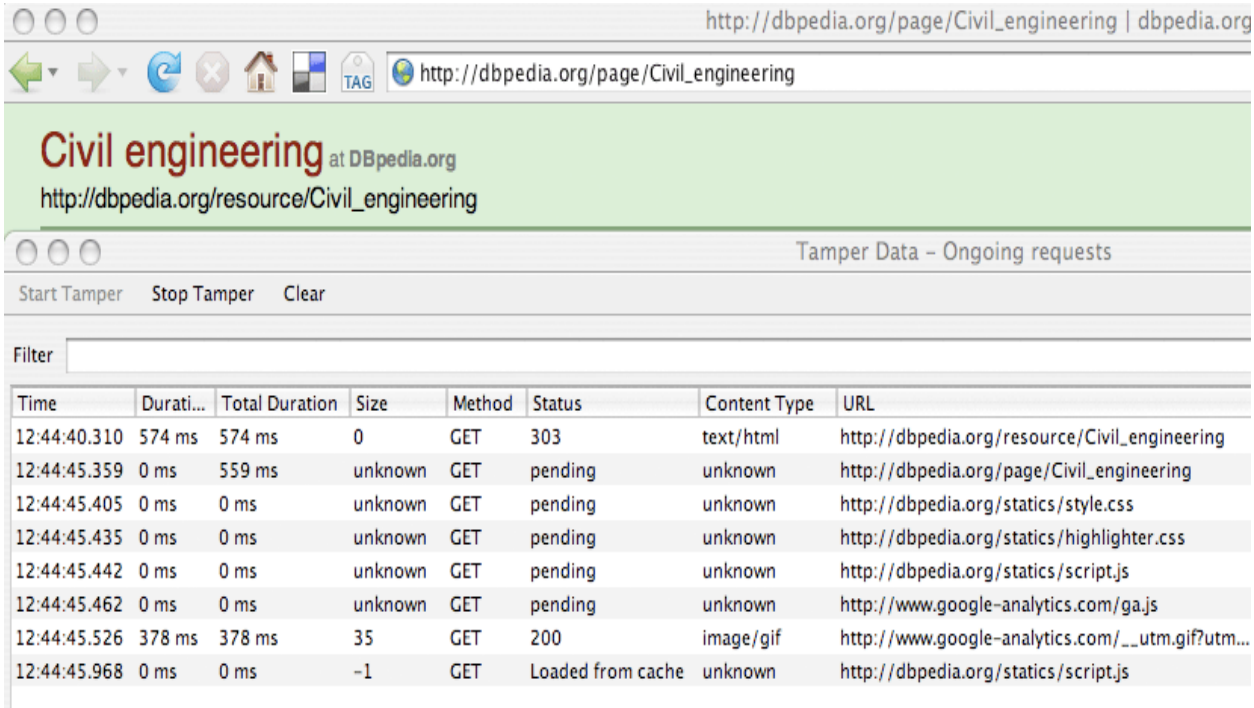
Заборона на матеріалізацію видів.

Справа в тому, що вартість поновлення нетривіальних видів зростає поліноміальної від обсягу бази, а зі зростанням різноманітності даних в базі зростає і кількість видів, які могли б комусь у нагоді. Якщо б ми вирішили почати використання SPMJV або інших схожих трюків, то зараз всі готівкові потужності були б зайняті оновленням видів, а не корисною роботою. Оракл використовує SPMJV, але для корпоративних додатків з постійною і відомої адміністратору тематикою запитів. Якщо корпоративний користувач Virtuoso хоче робити запити з відомою тематикою, то йому краще використовувати RDF Views, ніж комбінацію експорту та SPMJV. Тому MJV немає і в найближчому майбутньому не буде.

Тепер про більш приємне --- про те, що наша загальнодоступна точка доступу робити вміє. Перше, і найголовніше --- вона працює. Працює собі і працює, як і належить, що поважає себе системі індустріального якості. Як приклад, навесні 2009 року два скромних скриньки, кожен з одним quad-core Xeon і 8 гігабайтами, виконували всі "розумні" запити до lod.openlinksw.com, 4.5 гігаквада.

По-друге, система працює швидко. Ми стабільно показуємо найкращі часи в BSBM --- Berlin SPARQL Benchmark.
По-третє, Virtuoso може з тією ж швидкістю виконувати й більш складні запити. Наша мова запитів істотно потужніша за стандартного SPARQL, він навіть потужніше того, що буде передбачати специфікація SPARQL 2.0, яку зараз готує Data Access Working Group W3C. Ми додали висновок "додаткових" фактів відповідно до наявних онтологіями і предикатами same-as. Опитуючи властивості одного суб'єкта, можна заодно отримати і властивості всіх його синонімів, синонімів його синонімів і т.п. Схожим чином обробляються підкласів і часті випадки властивостей. Ми додали BI-(Business Intelligence) розширення до SPARQL, і в результаті можемо виконувати SPARQL-BI версії запитів TPC-H всього лише на 30 відсотків повільніше, ніж оригінальні версії SQL, а їх ми виконуємо з тією ж швидкістю, що й інші "серйозні" постачальники СУБД. Ми додали транзитивні підзапити, отримавши можливість шукати шляхи довільної довжини. Ми додали "Sponge" --- механізм завантаження з Мережі відсутніх даних у міру необхідності --- запит може сам додати в базу відсутні дані або оновити застарілі. Ми дуже акуратно вбудували SPARQL в SQL, так що SPARQL-запит може викликати вбудовані функції і процедури, що зберігаються SQL, і з іншого боку він може бути використаний, як підзапит в SQL-запиті, в тілі збереженої процедури . Більш того, запит може бути виконаний не тільки через SPARQL web service endpoint, але і через ODBC / UDBC / IODBC / JDBC. Ми підтримуємо потужний повнотекстовий пошук в SPARQL.
Що дійсно цікавить користувачів? Дуже популярними виявилися запити типу DESCRIBE. Це повністю розійшлося з прогнозами, згідно з якими DESCRIBE повинен був бути мертвою функціональністю, нікому не потрібною як мінімум до тих пір, поки в специфікації не додано докладний опис очікуваного результату. Довелося позапланово зайнятися спеціальної оптимізацією таких запитів. Очевидно, великої кількості додатків треба "розповісти хоч чого-небудь на задану тему". Крім того, дуже затребуваним виявився ще один тип запитів, який вимагав розширення не тільки інтерпретатора, але і протоколу SPARQL - "поверніть хоча б початок відповіді на питання, але за обмеженістю час". Такі запити дозволяють інтерактивним програмам оперативно отримувати підказки для користувача, ескізи звітів, а також необхідну для деяких інтерфейсів оціночну статистику "багато-мало" (наприклад, щоб вибрати тип подання за замовчуванням або вчасно попросити користувача розширити або звузити пошук). Кваліфіковані користувачі часто використовують "низькорівневий" веб-інтерфейс, вводячи запити простий статистики (наприклад, суми чого-небудь по країнах, відсортовані за значенням або по імені країни або за іншою статистикою) і вказуючи, що результат повинен імпортуватися в електронну таблицю.
Необхідне уточнення. З тих пір поведінка агентів істотно змінилося, як показує аналіз активності користувачів в роботі Knud Möller, Michael Hausenblas, Richard Cyganiak, Siegfried Handschuh and Gunnar Grimnes. Learning from Linked Open Data Usage: Patterns & Metrics. Web Science Conference 2010.
Раніше цей текст знаходився на http://webofdata.ru/LOD_infrastructure_experience , перенесено звідти на прохання редакторів webofdata.ru.

Розглянемо приклад використання Linked Data Project і SPARQL запитів:
Використовуючи матеріали з Вікіпедії, найбільшої енциклопедії в Інтернеті, користувачі можуть переглядати і виконувати повнотекстовий пошук, але програмний доступ до бази знань, є обмеженим.
Боротьба за статус проекту (DBpedia project) структурованої інформації з Вікіпедії [130], відкриття його для програмного доступу з використанням семантичного веб-технологій, таких як Linked Data і SPARQL. Це означає, що зв'язування та обдумування можливості RDF та OWL можуть бути використані і в запитах конкретної інформації, що можна зробити, використовуючи SPARQL[130].
Спрощено відображення з Вікіпедії HTML веб сторінок, для того щоб «боротьба» за статус RDF ресурсів можна було розглядати як заміну "http://en.wikipedia.org/wiki/" на "http://dbpedia.org/resource/", але насправді, є деякі додаткові тонкощі, які описані в статті з Вікіпедії про період існування URI, та боротьби за статус URI.
Вхід для Вікіпедії - "Цивільне будівництво" (http://en.wikipedia.org/wiki/Civil_Engineering) використовується як приклад, щоб показати, яким чином конкретні дані можуть бути вилучені з його боротьби за статус еквівалента (http://dbpedia.org/resource / Civil_engineering).
Коли обидва: і вхід в Вікіпедію (http://en.wikipedia.org/wiki/Civil_Engineering) і його боротьба за статус еквівалента (http://dbpedia.org/resource/Civil_engineering) відкриваються в стандартний веб-браузер, вони виявляють подібну інформацію, але боротьба за еквівалентний статус була перенаправлена на http://dbpedia.org/page/Civil_engineering.
Це перенаправлення можна побачити в Firefox, використанням Tamper Data для браузера Firefox, як показано на малюнку нижче.



Time	Durati...	Total Duration	Size	Method	Status	Content Type	URL
12:44:40.310	574 ms	574 ms	0	GET	303	text/html	http://dbpedia.org/resource/Civil_engineering
12:44:45.359	0 ms	559 ms	unknown	GET	pending	unknown	http://dbpedia.org/page/Civil_engineering
12:44:45.405	0 ms	0 ms	unknown	GET	pending	unknown	http://dbpedia.org/statics/style.css
12:44:45.435	0 ms	0 ms	unknown	GET	pending	unknown	http://dbpedia.org/statics/highlighter.css
12:44:45.442	0 ms	0 ms	unknown	GET	pending	unknown	http://dbpedia.org/statics/script.js
12:44:45.462	0 ms	0 ms	unknown	GET	pending	unknown	http://www.google-analytics.com/ga.js
12:44:45.526	378 ms	378 ms	35	GET	200	image/gif	http://www.google-analytics.com/__utm.gif?utm...
12:44:45.968	0 ms	0 ms	-1	GET	Loaded from cache	unknown	http://dbpedia.org/statics/script.js

Початковий стан «303» це код HTTP відповіді. Сервер відповів з кодом HTTP 303 тим, щоб направити браузер URI http://dbpedia.org/page/Civil_engineering , який HTML сторінки

браузера можуть переглянути. Оригінальні http://dbpedia.org/resource/Civil_engineering URI ресурсів є RDF, що не показали б, в браузері HTML.
Боротьба за статус реалізує механізм http, так званого, як зміст переговорів, з тим щоб забезпечити клієнтів, таких як веб-браузери, інформацією, вони звертаються з проханням у вигляді того, як вони можуть відобразитись. Як опублікувати Linked Data на веб-сайті, опис цього та інших пов'язаних даних методів, які використовуються в таких програмах, як «боротьба за статус(DBpedia)».
Для того, щоб отримати доступ до ресурсу RDF, безпосередньо веб-клієнт повинен повідомити серверу, щоб відправити його RDF дані. Клієнт може зробити це, надіславши запит HTTP заголовок, запит: застосування / + RDF XML в якості частини початкового запиту. (HTML браузер послав запит: текст / HTML заголовок HTTP про те, що вона просить сторінки HTML.)
Аддон Firefox RESTTest може бути використаний для встановлення прийнявши: застосування / + RDF XML в HTTP Request Header і прямо просити http://dbpedia.org/resource/Civil_engineering, як показано на малюнку нижче.

У цьому випадку запит http://dbpedia.org/resource/Civil_engineering вдався, як показав " Статус відповіді 200 (Response Status 200)" і документ RDF був отриманий, як показано в " Тексті відповіді (Response Text)".
SPARQL
DBpedia забезпечує громадський SPARQL [130] кінцевої точки на http://dbpedia.org/sparql, який дозволить користувачам запитувати джерело даних RDF з SPARQL запитом, таким як в наступному.

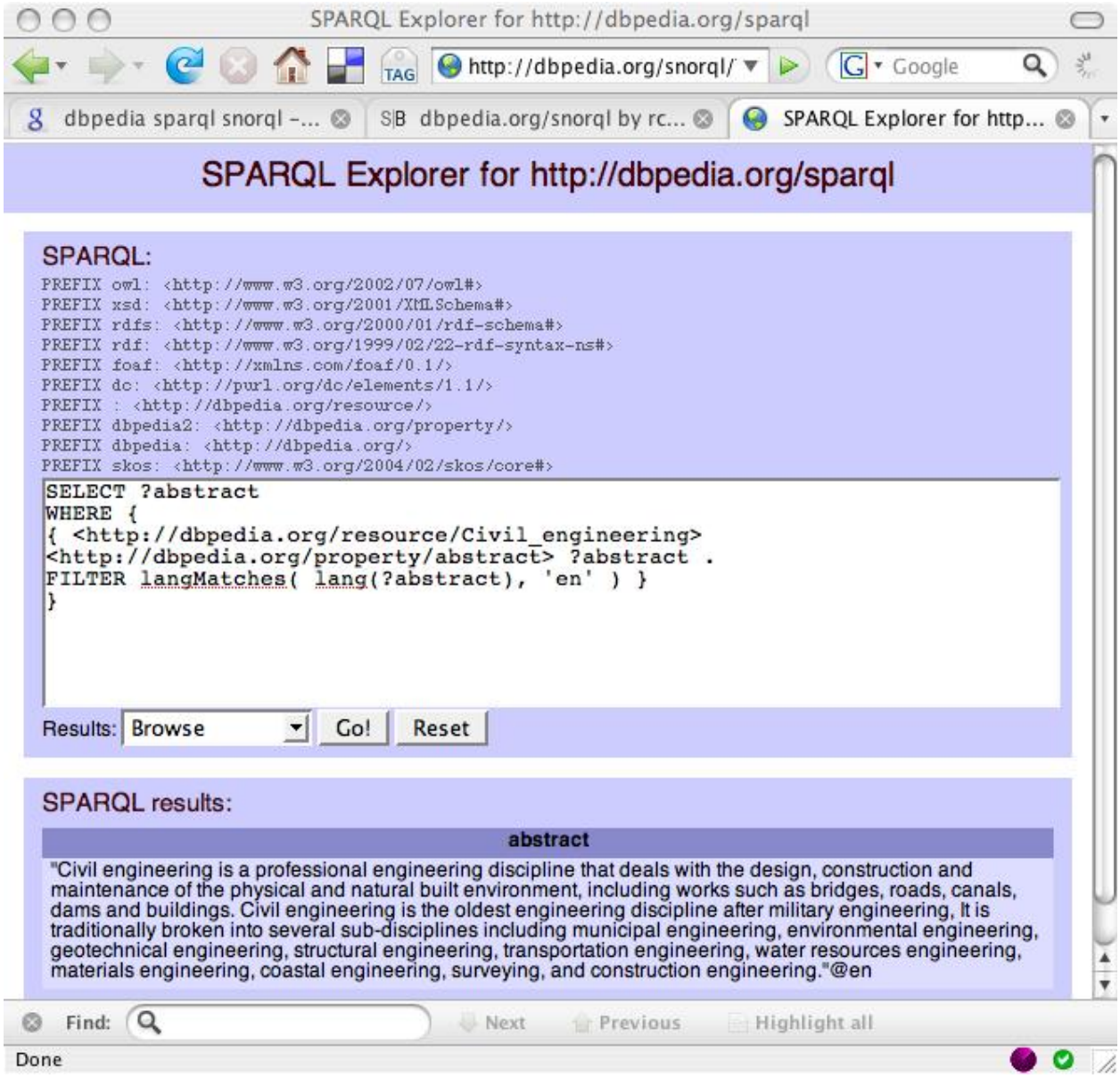
```
SELECT ?abstract
WHERE {
{ <http://dbpedia.org/resource/Civil_engineering> <http://dbpedia.org/property/abstract>
?abstract }
}
```

Запит повертає всі тези для цивільного будівництва, в кожну з доступних мов.

Наступний запит уточнює тези, спираючись тільки на мові, зазначений в цьому випадку 'en ' (англійською мовою).

```
SELECT ?abstract
WHERE {
{ <http://dbpedia.org/resource/Civil_engineering> <http://dbpedia.org/property/abstract>
?abstract .
FILTER langMatches( lang(?abstract), 'en') }
}
```

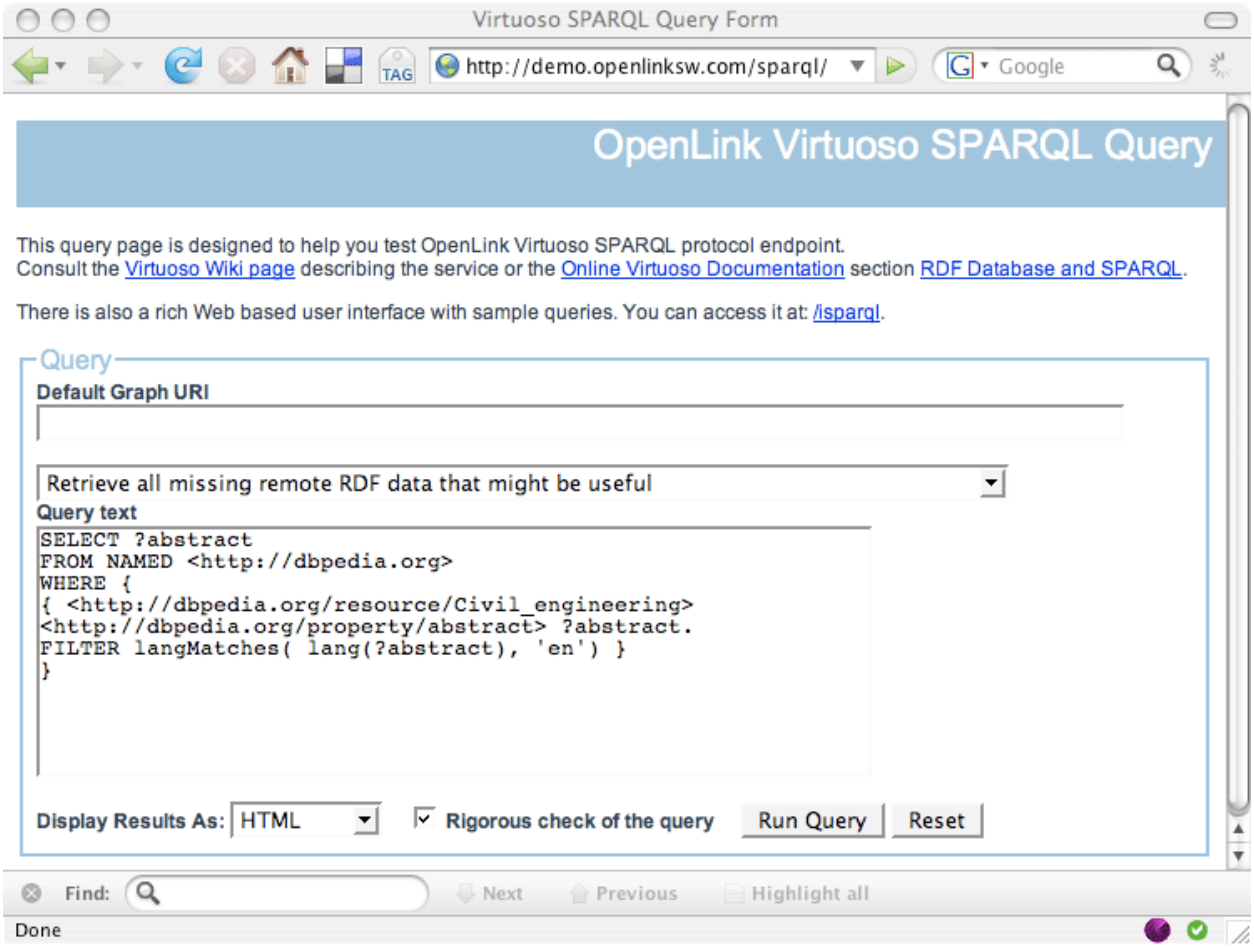
SNORQL запит, показаний на малюнку нижче, надає простий інтерфейс до кінцевої точки DBpedia SPARQL. На малюнку нижче показані обидва запити, і їх результат повернення.



Інші кінцеві точки SPARQL, такі як http://demo.openlinksw.com/sparql/ (див. нижче) може запросити DBpedia, вказавши ВІД ІМЕНІ положення для опису даних RDF.

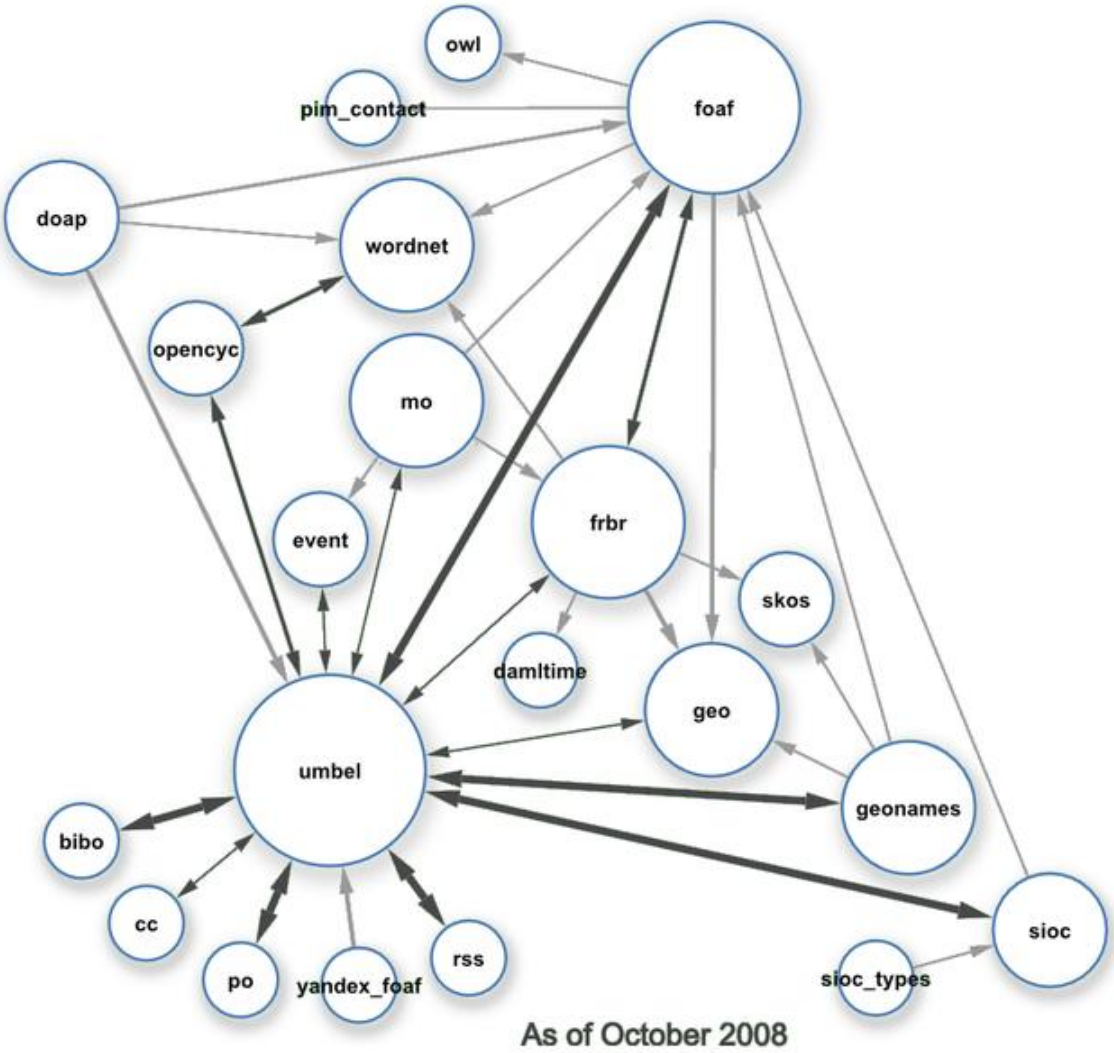
```
SELECT ?abstract
FROM NAMED <http://dbpedia.org>
WHERE {
```

{ <http://dbpedia.org/resource/Civil_engineering> <http://dbpedia.org/property/abstract>
?abstract.
FILTER langMatches(lang(?abstract), ‘en’) }
}



4.Linked Open Data and Web Of Data

Семантична Павутина є не тільки приміщенням даних в Інтернеті. Це є створенням посилань, таким чином, що персона або машина могла дослідити павутину даних. Із зв'язаними даними, коли ви маєте частину з цього, ви можете знайти інші спільні дані. Подібно до павутини гіпертексту, павутина даних конструюється з документами на павутині. Проте, на відміну від павутини гіпертексту, де посилання - анкери взаємин в гіпертекст-документах, написаних в HTML, для даних вони зв'язується між довільними речами, описаними RDF. URIs ідентифікують будь-який вид об'єкту або поняття. Але для HTML або RDF, ті ж очікування звертаються, щоб змусити павутину рости:



Мал. 4.1 Клас зв'язків в рамках Linking Open Data datasets

Використовуйте URIs як імена для речей

Використовуйте HTTP URIs таким чином, що люди можуть знайти ті імена.

Коли хто-небудь знаходить URI, забезпечте корисну інформацію, використовуючи стандарти (RDF, SPARQL)

Включайте зв'язки з іншим URIs. таким чином, що вони можуть виявити більше речей.

Простий. Фактично, хоча, дивовижна кількість даних не зв'язується в 2006, із-за проблем з один або більше з кроків. Ця стаття обговорює рішення до цих проблем, деталей виконання, і чинників, що впливають на альтернативи про те, як ви видаєте свої дані.

4 правила

Зробимо посилання на кроки вище за ті, як правила, але вони - очікування поведінки. Ломка їх не знищує, але припускає можливість зробити дані зв'язаними. Це у свою чергу обмежує шляхи, пізніше може використовуватися багато разів в несподіваних шляхах. Це - те, що несподівано повторно використовує інформація, значення якого додає павутина.

Перше правило, щоб солідаризуватися речі з URIs, хороше багато розуміється більшістю людей, що роблять семантичну мережеву технологію. Якщо це не використовує універсальну URI безліч символів, ми не називаємо це Семантичною Павутиною.

Друге правило, щоб використовувати HTTP URIs, також широко розуміється. Тільки відхилення було, починаючи з павутини, запущеної, постійна тенденція, щоб люди винайшли нові схеми (і під-схеми в урну: схема проїзду) URI як наприклад LSIDs і управляє і XRIs і DOIs, і так далі, для різних причин. Зазвичай, вони залучають не бажання зробити до встановленого Система (ДОМЕННА СИСТЕМА ІМЕН) Імені Domain для делегації повноважень але, щоб сконструювати що-небудь під окремим контролем. Іноді це має відношення до не розуміння, що HTTP URIs - імена (не адреси) і що пошук імені HTTP є складним, могутнім і розвиваючи набір стандартів. Цей результат обговорював в довжині де-небудь у іншому місці, і час не дозволяє нам ритися в цьому тут. [@@ref пошук ОЗНАКИ, etc])

Третє правило, що один повинен обслуговувати інформацію щодо павутини проти URI, є, в 2006, добре слідував більш всього для онтологій, але, з деякої причини, не для деяких головних наборів даних. Один може, взагалі, знаходять властивості і класифікує, один знаходить в даних, і отримують інформацію від RDF, RDFS, і COBA онтології, зокрема взаємини між термінами в онтології.

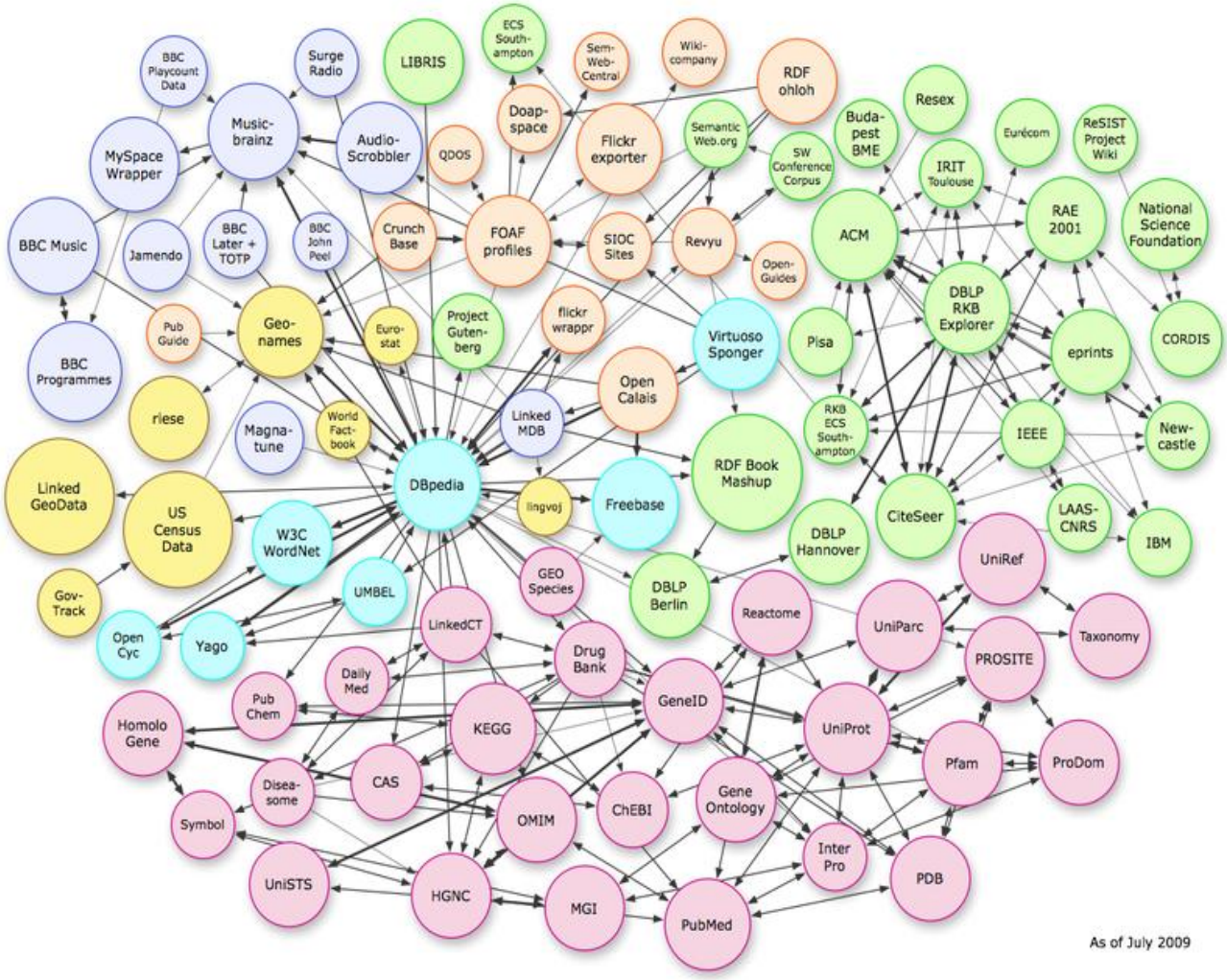
Багато хто досліджує і проекти оцінки за трохи років Семантичних Мережевих технологій проводили онтології, і істотний data запам'ятовує, але data, якщо доступно взагалі, ховається в архіві тріску де-небудь, замість доступного на павутині як зв'язані дані. Проект Biopax, дані CSAktive на дослідницьких людях інформатики і проектах були двома прикладами. [CSAktive data зараз (2007) доступний як зв'язані дані]

Є також великий і збільшуючи кількість URIs даних неонтології, які можуть бути знайдені. Семантичний wikis - один приклад. "Один один" (FOAF) і Опис Проектного (DOAP) онтологіх використовуються, щоб побудувати соціальні мережі через павутину. Типові соціальні мережеві портали не забезпечують зв'язки з іншими сайтами, ні виставляють їх дані в нормальній формі.

LiveJournal і Суспільство Opera - двох портальних веб-вузлів, які фактично видають їх дані в RDF на павутині. (Plaxo має схему сліду, і я не упевнений чи вони підтримують знає посилання). Це означає, що я можу написати в моєму файлі FOAF, що я знаю H kon не Діють використання його URI в Community даних Opera, і персоні або машинному розгляді, за яким дані можуть потім слідувати, це зв'язує і знаходить всіх його друзів. Це всі його друзі? Не дійсно: тільки його друзі, хто знаходиться в Суспільстві Opera. Система не робить, запам'ятовуючи URIs людей на різних системах. Так, поки соціальна мережа відкрита для посилань, що поступають, і поки це – в середині перегляду (browseable), він не робить посилання, що виконуються.

Четверте правило, щоб зробити посилання де-небудь у іншому місці, необхідне час сеансу даних, які ми маємо в павутині, серйозний, розв'язав павутину, в якій один може знайти al види речей, тільки так же на hypertext павутині ми зуміли вишикуватися.

У hypertext веб-вузлах це розглядається загалом швидше поганий етикет, щоб не пов'язати із зв'язаним зовнішнім матеріалом. Значення вашої власної інформації - дуже функція того, з чим це пов'язує, також як і властиве значення інформації в межах веб-сторінки. Так це знаходиться також в Семантичній Павутині.
Так давайте дивитися на шляхи з'єднання даних, починаючись з найпростішого шляху створення посилання.
Основний мережевий перегляд:



Мал.4.2 Інстанції зв'язків в рамках Linking Open Data datasets

Найпростіший шлях зробити зв'язані дані - використовувати, в одному файлі, URI, який указує в іншому.
Коли ви пишете файл RDF, скажімо <Http://example.org/smith>, потім ви можете використовувати локальні імена в межах файлу, говорять #albert, #brian і #carol. У N3 ви, можливо, говорили б

```
<#albert>  fam:child <#brian>, <#carol>.
```

або в RDF/XML

```
<rdf:Description about="#albert"
 <fam:child rdf:Resource="#brian">
  <fam:child rdf:Resource="#carol">
</rdf:Description>
```

Наприклад, в документ <http://example.org/jones> хто-небудь, можливо, написав би:

<#denise> fam:child <#edwin>, <smith#carol>.
або в RDF/XML
<rdf:Description about="#denise"
 <fam:child rdf:Resource="#edwin">
  <fam:child rdf:Resource="http://example.org/smith#carol">
</rdf:Description>
Ясно, що це виглядає зарозумілим для того, хто зустрічає ідентифікатор 'http://example.org/smith#carol":
Сформуйте URI документа усікаючи перед сміттям
Зверніться до документа, щоб отримати інформацію близьку до «#carol».
Ця річ має назву – «розіменований» URI. Це основа Semantic Web.
Є деякі обставини, в яких розділові ідентифікатори в документах, не працює дуже добре. Можливо, логічно є один глобальний символ у документі, і є небажаним для включення # у URI, як наприклад:
http://wordnet.example.net/antidisesablishmentarianism#word
Раніше Дублінське Ядро і словники FOAF не мали «#» у їх URIs. У всякому разі, коли HTTP URIs ,без сміття, використовуються для абстрактних понять, і є документом, який несе інформацію про них, тоді:
HTTP відсилає запит на URI, поняття повертає 303. Дивлячись також і на розташування: магістралі, URI документа.

Веб перегляд

Цей метод має перевагу, яку URIs може зробити зі всіх форм. Воно має недолік, що HTTP запит mBrowse-ableust зроблений без винятку. У випадку Дублінського Ядра, наприклад, dc:title і dc:creator etc фактично обслуговує той же документ онтології.
Варіація: FOAF і rdfs:seeAlso
Договір Friend-Of-a-friend використовує форму data link, але не використовуючи, будь-яку з двох форм згаданих вище. Щоб послатися на іншу персону у файлі FOAF, угода мала надати дві властивості, одна вказівка документу, в якому вони описані, і інший для ідентифікації їх в межах цього документа.
<#i> foaf:knows [
      foaf:mbox <mailto:joe@example.com>;
       rdfs:seeAlso <http://example.com/foaf/joe>].
Читання, "я знаю, який маю email joe@example.com і про який більше інформації знаходиться в <http://example.com/foafjoe>".
Фактично, для конфіденційності, часто люди не поміщають свої адреси електронної пошти на павутині безпосередньо, але фактично поміщають одностороннє сміття (SHA-1) їх адреси електронної пошти і надають це. Цей розумний прийом дозволяє людей, які знають їх адресу електронної пошти вже, щоб вирішити це, це - та ж персона, без віддання електронної пошти до інших.
<#i> foaf:knows [
      foaf:mbox_sha1sum "273816784612376482364 7""; # @@ макет
  rdfs:seeAslo <http://example.com/foaf/joe>].
Ця система, що зв'язується, була дуже успішна, формуючи соціальну мережу зростання, і домінування, в 2006, зв'язані дані, доступні на павутині.

Проте, система має сучок, що він не надає URIs людям, і такі основні зв'язки з ними не можуть бути зроблені.
В веб-блогах ,Links on the Semantic Web, надають самі собі URI, як прямих так і зворотніх зв'язків в RDF, які також є важливими), щоб ті, вимушуючи файл FOAF надати собі URI також як і використовуючи угоду FOAF.Так само, коли ви посилаєтеся на файл FOAF, який надає URI персоні, використовують це у вашому посиланні на цю персону, таким чином, що клієнти, які тільки використовують URIs і не знають про угоду FOAF, може слідувати за посиланням.

Перегляд графів

Зараз ми подивимось на шляхи створення посилань, які дозволяють нам дивитися на альтернативи того, коли зробити посилання.
Один важливий зразок - набір даних, які ви можете дослідити, оскільки ви їдете посилання на посилання привабливими даними. Кожного разу один знаходить URI для вузла в графові RDF, сервер повертає інформацію про дуги поза цим вузлом, і дугами в. Іншими словами, це повертає будь-які твердження RDF, в яких термін з'являється або як тема, або об'єкт.
Формально, назвіть G - Перегляд графів, якщо, для URI будь-якого вузла в G, якщо я знаходжу це URI, я буду поверненою інформацією, яка описує вузол, де, описуючи засоби вузла:
Повернення всіх заяв, у яких вузол суб'єкта або об'єкта.
Описуючи всі порожні вузли, прикладені до вузла однією дугою.
(Повернений підграф був направлений до того, як "мінімальний Граф (ПОВІДОМЛЕННЯ [@@ref]), що Охоплює, або молекула [@@ref]RDF, залежно від того, чи вузли розглядаються виділив, якщо вони можуть бути виражені як шлях функції, або зворотних зворотних функціональних властивостей. Стислий зв'язаний опис, який тільки витікає посилання з теми до об'єкту, не працює.)
На практиці, коли data запам'ятовується в двох документах, це означає, що будь-які твердження RDF, які мають відношення до речей в двох файлах, повинні бути повторними в кожному. Так, наприклад, в моїй сторінці FOAF я згадую, що я входжу в склад групи DIG, і що інформація повторна на DIG даних групи. Тому, хто-небудь, починаючись від поняття групи може також з'ясувати це, я входжу в склад. Фактично, хто-небудь, хто починає з моїм URI може знайти всіх людей, які знаходяться в тій же групі.
Обмеження щодо перегляду, даних
Так твердження, які мають відношення до речей в двох документах, повинні бути повторними в кожному. Це ясно проти першого правила зберігання даних: не запам'ятовуйте ті ж дані в двох різних місцях: ви матимете проблеми, що тримають це послідовним. Це дійсно результат з доступними даними. Набір цілком видимих даних з посиланнями в як напрями доводиться бути цілком послідовним, так і це бере координацію, особливо, якщо різні автори або різні програми залучені.
Ми можемо мати цілком доступні дані, проте, де це автоматично проводиться. Dbview сервер, наприклад, забезпечує доступні віртуальні документи, що містять дані з будь-якої довільної реляційної бази даних.
Коли ми маємо дані з багаторазових джерел, потім ми маємо компроміси. Їх часто врегулював здоровий глузд, ставлячи питання

"Якщо хто-небудь має URI цієї речі, що взаємини до того, що інші об'єкти - це корисний, щоб знати?"
Іноді, соціальні питання визначають відповідь. Я маю посилання в моєму файлі FOAF, що я знаю різних людей. Вони загалом не повторюють, що інформація в їх файлах FOAF. Хто-небудь, можливо, говорить, що вони знають мене, який є твердженням, яке, в угоді FOAF, - вони для заяви, і читач, щоб довірити або ні.
Інші часи, число дуг робить це непрактичним. Трек GPS надає тисячі разів, в яких моя широта, довгота відомі. Кожна персона, що завантажує мій файл FOAF, може чекати, отримати мою інформацію, візитки, але не всі ті точки . Розумно мати покажчик від трека (або навіть кожен пункт) до персони, чия позиція представлена, але не інакше.
Один зразок - мати посилання певної властивості в окремому документі. Початкова сторінка персони не складає список все їх публікацій, але замість поміщає зв'язок з цим окремий документ, лістинг їх. Є розуміння що foaf:made надає роботу деякого сортування, але foaf:pubs указує документу, що надає список робіт. Тому, хто-небудь, шукаючи що-небудь, foaf:made посилання зробило б well, щоб слідувати за foaf:pubs посиланням. Вона, можливо, була б корисна, щоб формалізувати поняття з твердженням подібно
foaf:made link:listDocumentProperty foaf:pubs.
в одній з онтологій.

Запит послуг

Іноді явний об'єм даних робить службу цього силою-силенною файлів, можливих, але незграбний для ефективних видалених запитів над набором даних. В даному випадку, це здається розумним, щоб надати SPARQL послугу запиту. Щоб змусити фактично зв'язати дані, хто-небудь, хто тільки має URI чого-небудь повинен бути здатний знайти їх шлях кінцевою крапкою SPARQL.
Тут знову відповідь HTTP 303 може використовуватися, щоб направити the01-15 enquirer до документа з метаданими, які ставлять під сумнів service, кінцеві крапки можуть забезпечити, що інформація, про яку говорилось в класі URIs.
Словники для виконання цього ще не були стандартизовані.

Висновок

Semantic Web – це динамічна концепція, яка знаходиться у постійному розвитку та не є набором комплексних, працюючих систем.

З точки зору машинної обробки даних – "Semantic Web – це ідея зберігання даних в Web таким чином, щоб вони були визначені та зв'язані подальшою змогою автоматизованої обробки, інтеграції і повторного використання їх в різноманітних додатках." [9] [9]

З точки зору інтелектуальних агентів «ціллю Semantic Web являється зробити існуючий Web більш машинозчитуючим тим, щоб мати змогу використати агентів для пошуку та обробки відповідної інформації." [112]

З точки зору розподілених баз даних «концепція Semantic Web заключається в «… забезпеченні достатньої гнучкості для змоги представлення всіх баз даних та правил логіки таким образом, щоб зв'язати їх всі разом...» [9] "Простий опис Semantic Web полягає в тому, що він представляє собою спробу реалізувати машинну обробку даних… Інколи, трансформувати обробку інформації забезпеченням спільного принципу, по якому дані можуть бути отримані, зв'язані разом та зрозумілі. Переклад Web від типу «великої книги з гіперпосиланнями» до великої зв'язаної бази даних "[112].

З точки зору автоматизованої інфраструктури – «Semantic Web являється інфраструктурою, а не додатком» [113].

З точки зору надання послуг для людських потреб – ідея Semantic Web полягає в звільненні людини від жахливих рутинних задач по добуванню, пошуку, обліку та індексації інформації, яка знаходиться в Web. «Semantic Web – це бачення наступного покоління Інтернету, який дасть змогу веб-додаткам автоматично збирати веб-документи з різних джерел, враховувати та оброблювати інформацію, а також взаємодіяти з іншими додатками для виконання важких задач." [114]

З точки зору покращення анотування – «ідея Semantic Web полягає в забезпеченні існуючого Web анотаціями, вираженими в машинооброблюючій формі и зв'язаними між собою» [115].

З точки зору покращення пошуку – реалізація пошуку не тільки по ключовим словам, але і по контенту.

З точки зору веб-сервісів – «Semantic Web повинен забезпечити доступ не тільки до статичних документів, які містять корисну інформацію, але і до сервісів, які надають корисні послуги» [116].

Таким образом, задача Semantic Web, а також і його проблеми заключаються в наступному:

Індексація та пошук інформації;

Розробка та підтримка метаданих;

Розробка та підтримка методів анотування;

Представлення Web в вигляді великої, інтероперабельної бази даних;

Організація машинного добування даних;

Знаходження та пропонування веб-орієнтовних сервісів;

Пошук в області інтелектуальних програмних агентів.

Додаткова бібліографія по даній тематиці наведена в [117].

Список використаної літератури:

1. W3C Semantic Web
Activity. – http://www.w3.org/2001/sw/Activity
2. SemanticWeb organization. – http://www.semanticWeb.org/
3. Getting into RDF “Semantic
Web using N3”, Tim Berners-Lee – http://www.w3.org/2000/10/swap/Primer.html
4. Web Architecture: Describing and Exchanging Data”, Berners-Lee, Connolly, Swick, W3C Note 7 June 1999. – http://www.w3.org/1999/04/WebData
5. Metadata Architecture, W3C Design Issues. – http://www.w3.org/DesignIssues/Metadata
6. RDF and Metadata, Tim Bray, June 09, 1998. – http://www.xml.com/xml/pub/98/06/rdf.html
7. The Power of Metadata, book chapter by Rael Dornfest, Dan Brickley. – http://www.openp2p.com/pub/a/p2p/2001/01/18/metadata.html
8. Web Metadata: A Matter of Semantics by Ora Lassila, IEEE Internet Computing, July-August 1998. – http://computer.org/internet/ic1998/w4030abs.htm
9. W3C, The Semantic Web Home Page. – http://w3.org/sw/
10. AgentWeb, resource guide and newsfeed covering Agent-related technologies. – http://agents.umbc.edu/
11. A Model-Theoretic Semantics for DAML+OIL, W3C Note 18 December 2001. – http://www.w3.org/TR/daml+oil-model
12. An Axiomatic Semantics for RDF, RDF-S, and DAML+OIL, W3C Note 18 December 2001. – http://www.w3.org/TR/daml+oil-axioms
13. DAML+OIL (March 2001) Reference Description, W3C Note 18 December 2001. – http://www.w3.org/TR/2001/NOTE-daml+oil-reference-20011218
14. XML Schema, RDF Schema & DAML Comparison. – http://www.isi.edu/expect/Web/semanticWeb/comparison.html
15. W3C Web Ontology. – http://www.w3.org/2001/sw/WebOnt/
16. Requirements for a Web Ontology Language, W3C Working Draft. – http://www.w3.org/TR/Webont-req/
17. SemanticWeb: роль XML и RDF/ С. Декер, С. Мельник, Ф. ван Хермелен, Д. Фенсел, М. Клейн, Д. Брукстра, М. Эрдманн, Я. Хоррокс // Открытые системы. 2001 - № 9. – http://www.osp.ru/os/2001/09/041.htm.
18. Distributed XML: the role played by XML in the next-generation Web, Edd Dumbill. – http://www.xml.com/pub/2000/09/06/distributed.html
19. XML and the Web, by Tim Berners-Lee, XML World 2000, Boston 2000/09/06. – http://www.w3.org/2000/Talks/0906-xmlWeb-tbl/
20. An Introduction to the Resource
Description Framework by Eric Miller, D-Lib Magazine, May 1998. – http://www.dlib.org/dlib/may98/miller/05miller.html
21. Putting RDF to Work, Edd Dumbill. – http://www.xml.com/pub/2000/08/09/rdfdb/index.html
22. RDF tutorial, Pierre-Antoine Champin (for developers). – http://www710.univ-lyon1.fr/~champin/rdf-tutorial/
23. W3C Web Service`s Home /Page. – http://www.w3.org/2002/ws/
24.Web Services Architecture, W3C Working Draft 14 November 2002. – http://www.w3.org/TR/ws-arch/

25.Web Services Architecture Requirements, W3C Working Draft 14 November 2002. – http://www.w3.org/TR/wsa-reqs
26.Web Services Architecture Usage Scenarios, W3C Working Draft 30 July 2002. – http://www.w3.org/TR/ws-arch-scenarios/
27. Web Services Description Requirements, W3C Working Draft 28 October 2002. – http://www.w3.org/TR/ws-desc-reqs/
28. Web Services Glossary, W3C Working Draft 14 November 2002. – http://www.w3.org/TR/ws-gloss/
29. Лифшиц Ю., Семантический Веб, лекція, 2006. – http://logic.pdmi.ras.ru/~yura/internet.html
30. The Semantic Web. By Tim Berners-Lee, James Hendler and Ora Lassila. Scientific American, May 17, 2001. – http://www.scientificamerican.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21
31.The Semantic Web Roadmap, Tim Berners-Lee, 1998. – http://www.w3.org/DesignIssues/Semantic.html
State of The SemanticWeb, Ivan Herman, Stavanger, Norway, 2007.
33. Semantic Web for Developers. – http://logicerror.com/semanticWeb-Webdev
34. Extensible Markup Language (XML) 1.0, W3C Recommendation 10.02.1998. – http://www.w3.org/TR/1998/REC-xml-19980210
35. RDF/XML Syntax Specification (Revised), W3C Working Draft 25 March 2002. – http://www.w3.org/TR/rdf-syntax-grammar/
36. RDF Model Theory, W3C
Working Draft 29 April 2002. – http://www.w3.org/TR/rdf-mt/
37. RDF Semantics, W3C Working Draft 23 January 2003. – http://www.w3.org/TR/2003/WD-rdf-mt-20030123/
38.RDF Primer, W3C Working Draft 11 November 2002. – http://www.w3.org/TR/rdf-primer/
39.RDF Test Cases, W3C Working Draft 12 November 2002. – http://www.w3.org/TR/rdf-testcases
40.RDF Tutorial, W3C. – http://www.w3.org/TR/rdf-tuturial
41.Resource Description Framework (RDF): Concepts and Abstract Data Model, W3C Working Draft 29 August 2002. – http://www.w3.org/TR/rdf-concepts/
42.Resource Description
Framework (RDF) Model and Syntax Specification, W3C Recommendation 22 February 1999. – http://www.w3.org/TR/REC-rdf-syntax/
43.Using RDF to model multimedia content – slide "Relation with MPEG-7". – http://www.w3.org/Architecture/1998/06/Workshop/paper29/slides/slide13-0.html
44. RDF syntax, W3C Recommendation. – http://www.w3.org/TR/PR-rdf-syntax
45. RDF Schema, W3C Working Draft. – http://www.w3.org/TR/PR-rdf-schema
46.RDF Vocabulary Description Language 1.0: RDF Schema, W3C Working Draft 23 January 2003. – http://www.w3.org/TR/2003/WD-rdf-schema-20030123/
47.Topic Maps (XMT). – http://www.topicmaps.org/
48.Text Encoding Initiative. – http://www.tei-c.org/
49.Metadata Encoding and Transmission Standard. – http://www.loc.gov/standards/mets/
50. Metadata Object Description Schema (MODS). – http://www.loc.gov/standards/mods
51.Encoded Archival Description (EAD). – http://www.loc.gov/ead

52.Learning Object Metadata (LOM). – http://www.ltsc.ieee.org/wg12/
53. Online Information Exchange (ONIX). – http:// www.editeur.org/onix.html
54.Introduction to the Semantic Web, Ivan Herman, W3C, International Conference on Dublin Core and Metadata Applications, Singapore, 2007-08-31. – http://www.w3.org/2007/Talks/0831-Singapore-IH/
55.The Friend of a Friend (FOAF) project. – http:// www.foaf-project.org/
56.FOAF Vocabulary Specification. – http://www.xmlns.com/foaf/0.1/
57.FOAF Vocabulary Specification. – http://www.xmlns.com/foaf/spec/
58.Semantically-Interlinked Online Communities. – http://www.sioc-project.org/
59.Description of a Project Description of a Project (DOAP) vocabulary. – http://www.usefulinc.com/doap/
60.RFC2413, Dublin Core Metadata for Resource Discovery. – http://www.faqs.org/rfcs/rfc2413.html
61. "DublinCore Qualifiers/Substructure". – http://www.loc.gov/marc/dcqualif.html
62. "DublinCore qualifiers". – http://www.roads.lut.ac.uk/Metadata/DC-Qualifiers.html
63.Dublin Core Element Set, Version 1.1 – Reference Description. – http://www.dublincore.org/documents /1999/07/02/dces/
64.vCard. – http://www.imc.org/pdi/
65. Names in Dublin Core, Diane I. Hillmann. – http://purl.org/dc/documents/notes/notes-hillmann-19981027.htm
66."Guidance on expressing the Dublin Core within the Resource Description Framework (RDF)". – http://www.ukoln.ac.uk/metadata/resources/dc/datamodel/WD-dc-rdf/
67.Representing vCard v3.0 in RDF, Renato Iannella. – http://www.dstc.edu.au/RDU/RDF/draft-iannella-vcard-rdf-00.txt
68.ROADS. – http://ukoln.bath.ac.uk/roads/
69.Resource Description Framework (RDF) Model and Syntax Specification, W3C Recommendation 22 February 1999. – http://www.w3.org/TR/REC-rdf-syntax/
70.Open Directory Project. – http://dmoz.org/
71.RDF Vocabulary Description Language 1.0: RDF Schema, W3C Working Draft 23 January 2003. – http://www.w3.org/TR/2003/WD-rdf-schema-20030123/
72.DAML+OIL Project Homepage. – http://www.w3.org/TR/daml+oil-reference
73.DAML+OIL Primer. – http://www.w3.org/TR/rdf-primer/#ref-damloil
74. Мова OWL. – http://www.w3.org/TR/owl-ref/
75. OWL, Primer. – http://www.w3.org/TR/rdf-primer/#ref-owl
76.XML-QL: A Query Language for XML. Submission to the World Wide Web Consortium 19.08.1998. – www.w3.org/TR/NOTE-xml-ql/
77.XQL Tutorial (XML Query Language), Jonathan Robie. – http://www.metalab.unc.edu/xql/xql-tutorial.html
78.XML-QL : A Query Language for XML User's Guide Version 0.9. – http://www.research.att.com/~mff/xmlql/doc/
79.Home of the W3C's XML Query working group. – http://www.w3.org/XML/Query
80.A Query Language for XML. Alin Deutsch, Mary Fernandez, Daniela Florescu.University of Pennsylvania, Philadelpha. – http://www8.org/w8-papers/1c-xml/query/query.html
81.RDF Query Language (RQL). – http://139.91.183.30:9090/RDF/VRP/index.html/RQL/index.html

82.The RDF Query Rules, W3C. – http://www.w3.org/2001/11/13-RDF-Query-Rules/
83.The RDF Query Language (RQL), W3C. – http://139.91.183.30:9090/RDF/RQL/
84.RDF Query Specification, December 3, 1998. – http://www.w3.org/TandS/QL/QL98/pp/rdfquery.html
85.TRIPLE HomePage. – http://triple.semanticWeb.org/
86.Sesame, storage and querying middleware system for RDF and RDF Schema. – http://sesame.aidministrator.nl/
87.SPARQL Query Language for RDF W3C Candidate Recommendation 14 June 2007. – http://www.w3.org/TR/rdf-sparql-query/
88.The Semantic Web In Breadth, Aaron Swartz. – http://logicerror.com/semanticWeb-long
89.Математические предпосылки (логіка предикатів) – Mathematical Background (Predicate Logic), Джон Сова (John Sowa's). – http://www.jfsowa.com/logic/math.htm
90. RIF: Use Cases and Requirements, W3C Working Draft 10 July 2006. – http://www.w3.org/TR/2006/WD-rif-ucr-20060710/
91.RuleML. – http://www.ruleml.org/
92.SWRL: A Semantic Web Rule Language Combining OWL and RuleML, W3C Member Submission 21 May 2004. – http://www.w3.org/Submission/SWRL/
93. Презентація докладу «Семантический Веб: текущее состояние исследований и перспективные направления», Уланов Д., ИСП РАН, 03.02.2006.- http://dulanov.wordpress.com/2006/02/02prezentatsiya_o_proekte_semanticheskii_veb/
94. SOAP Version 1.2 Part 0: Primer, W3C Candidate Recommendation 19 December 2002. – http://www.w3.org/TR/2002/CR-soap12-part0-20021219/
95.SOAP Version 1.2 Part 1: Messaging Framework, W3C Candidate Recommendation 19 December 2002. – http://www.w3.org/TR/2002/CR-soap12-part1-20021219/
96.SOAP Version 1.2 Part 2: Adjuncts, W3C Candidate Recommendation 19 December 2002. – http://www.w3.org/TR/2002/CR-soap12-part2-20021219/
97.SOAP Version 1.2 Specification Assertions and Test Collection, W3C Working Draft 26 June 2002. – http://www.w3.org/TR/soap12-testcollection
98.SOAP Version 1.2 Usage Scenarios, W3C Working Draft 26 June 2002. – http://www.w3.org/TR/xmlp-scenarios/
99.SOAP 1.2 Attachment Feature, W3C Working Draft 24 September 2002. – http://www.w3.org/TR/soap12-af/
100.SOAP Version 1.2 Email Binding, W3C Note 3 July 2002. – http://www.w3.org/TR/soap12-email
101.Universal Description, Discovery, and Integration (UDDI) OASIS Standard. – http://www.uddi.org
102.Web Services Description Language (WSDL) Version 1.2, W3C Working Draft 24 January 2003. – http://www.w3.org/TR/2003/WD-wsdl12-20030124/
103.Semantic Markup for Web Services, W3C Member Submission 22 November 2004 http://www.w3.org/Submission/OWL-S/
104.Service Modeling Language, Version 1.1 W3C Working Draft 3 March 2008, http://www.w3.org/TR/sml/
105.Open Directory Project, RDF dumps. – http://dmoz.org/rdf.html
106.Google Search Engine. – http://google.com

107.SWAD-Europe: Mapping Semantic Web Data with RDBMSes, W3C Semantic Web Advanced Development for Europe (SWAD-Europe), 2003-01-23. – http://www.w3.org/2001/sw/Europe/reports/ scalable_rdbms_mapping_report/
108.Introduction and Overview to the Semantic Web, James A. Hendler , Rensselaer Polytechnic Institute, The 6th
International Semantic Web Conference and the 2nd Asian Semantic Web Conference, 11-15 ноября 2007г. – http://videolectures.net/iswc07_hendler_ios/
109.The 6th International Semantic Web Conference and the 2nd Asian Semantic Web Conference, 11-15 November 2007, Busan, Korea. – http://iswc2007.semanticweb.org/main/default.asp
110.Semantic Web Challenge Homepage. – http://challenge.semanticWeb.org/
111.Enrico Motta, The Open University, Semantic Web Applications, The 6th International Semantic Web Conference and the 2nd Asian Semantic Web Conference, 11-15 ноября 2007 г. –http://videolectures.net/iswc07_motta_swa/
112.Sean B. Palmer, The Semantic Web: An Introduction, 2001-09. – http://infomesh.net/2001/swintro
113.Semantic Web As “Perfection Seeking:” A View from Drug Terminology, Tuttle M., Brown S., Campbell K., Carter J., Keck K., Lincoln M., Nelson S., Stonebraker M., 2001.
114.Semantic Web Modeling and Programming with XDD, Anutariya, Wuwongse, Akama, Wattanapailin, In Proceedings of SWWS'2001.
115.Towards a principled approach to semantic interoperability, Euzenat, IJCAI 2001, Workshop on ontology and and information sharing, 2001, Seattle (WA US)
116.“Explorer's Guide to the Semantic Web”, Thomas B. Passin, June, 2004, 304 p.
117.Бібліографія по тематиці Semantic Web, Type of content, Class blog. – http://typecontent.net/blog/wp-content/uploads/2007/02/semanticbibliography.pdf
118. Towards The Semantic Web, Ontology-driven Knowledge Management, Dr John Davies, Dieter Fensel and Frank van Harmelen
119. A Semantic Web Primer, 2nd edition, Grigoris Antoniou, Frank van Harmelen
120. Semantic Annotations In Web Services, Meenakshi Nagarajan
http://www.springer.com/978-0-387-30239-3
121.HP Labs Semantic Web Research. // http://www.hpl.hp.com/semweb/
122. http://xmlhack.ru/texts/04/SOAvsWebservices/SOAvsWebservices.html
123. http://shcherbak.net/semantic-web-kak-novaya-model-informacionnogo-prostranstva-internet/
124. http://www.soobshestva.ru/wiki/ProgrammnyeAgenty
125. http://www.daviddarling.info/encyclopedia/S/siliconlife.html
126. ru.wikipedia.org/wiki/Веб-служба
127. D. Martin, M. Paolucci, and M. Wagner, "Towards Semantic Annotations of Web Services OWL-S from the SAWSDL Perspective", in OWL-S Experiences and Future Developments Workshop of 4th European ESWC’07 Conference, 2007.
128. Fatima-Zahra Belouadha, Hajar Omrana and Ounsa Roudies «A model-driven approach for composing SAWSDL semantic Web services»
129. http://ru.wikipedia.org/wiki/BPEL
130. http://blog.3kbo.com/2008/08/11/dbpedia-examples-using-linked-data-and-sparql/