

Міністерство освіти і науки України
Запорізький національний технічний університет

МЕТОДИЧНІ ВКАЗІВКИ
до лабораторних робіт
з дисципліни
“МЕТОДИ АНАЛІЗУ ДАНИХ ТА ЇХ ОРГАНІЗАЦІЯ”
для студентів
напряму підготовки 6.050101 «Комп’ютерні науки»
спеціальності 8.05010102 «Інформаційні технології проектування»
усіх форм навчання

2013

Методичні вказівки до лабораторних робіт з дисципліни «Методи аналізу даних та їх організація» для студентів напряму підготовки 6.050101 «Комп'ютерні науки» спеціальності 8.05010102 «Інформаційні технології проектування» усіх форм навчання / Т.В. Юр. – Запоріжжя: ЗНТУ, 2013. – 50 с.

Автори: Т. В. Юр, к.т.н.

Рецензент: В.І. Дубровін, к.т.н., проф.

Відповідальний
за випуск: А.В. Притула, к.т.н., проф.

Затверджено
на засіданні кафедри
програмних засобів

Протокол №2
від “16” жовтня 2013 р.

ЗМІСТ

Вступ.....	4
Лабораторна робота № 1 Представлення даних в форматі XML.....	5
Лабораторна робота № 2 Представлення даних в форматі RDF.....	15
Лабораторна робота № 3 Мова представлення онтологій OWL.....	25
Лабораторна робота № 4 Інженерія знань на основі логіки.....	38
Лабораторна робота № 5 Інтеграція RDF та HTML.....	45
Література.....	50

ВСТУП

Семантичне павутина (Semantic Web) – це частина глобального розвитку мережі Інтернет, метою якої є реалізація можливості машинної обробки інформації, що доступна в мережі WWW. Головний акцент цієї концепції робиться на роботі з метаданими, що однозначно характеризують властивості та вміст ресурсів WWW, замість текстового аналізу документів, що використовується наразі.

В семантичній павутині передбачається повсюдне використання універсальних ідентифікаторів ресурсів (URI) та онтологій і мов опису метаданих.

Для впровадження цієї концепції передбачається створення мережі документів, що містять метадані про ресурси WWW та існують паралельно з ними. В той час як самі ресурси призначені для сприйняття людиною, метадані використовуються машинами (пошуковими роботами та іншими інтелектуальними агентами) для отримання однозначної інформації про властивості цих ресурсів за допомогою механізмів логічного висновку.

Технічну частину семантичної павутини складає сімейство стандартів на мови опису даних, що включає XML, XML Schema, RDF, RDF Schema, OWL, тощо.

Практична частина курсу «Методи аналізу даних та їх організація» складається з п'яти лабораторних робіт і призначена для отримання практичних навичок зі структурування та розмітки довільної інформації, а також отримання знань на основі логічного висновку.

Матеріал до кожної лабораторної роботи містить мету, теоретичні довідки та рекомендації, загальне завдання, порядок виконання роботи, зміст звіту про виконання лабораторних робіт, а також список рекомендованих інформаційних джерел для підготовки і виконання лабораторних робіт.

ЛАБОРАТОРНА РОБОТА № 1

ПРЕДСТАВЛЕННЯ ДАНИХ В ФОРМАТІ XML

1.1 Мета роботи

Ознайомитися з форматом даних XML та його можливостями, отримати практичні навички роботи з ним в середовищі EditX. Змодельовати певну область знань за допомогою мови XML.

1.2 Короткі теоретичні відомості

XML - розширювана мова розмітки (Extensible Markup Language) - запропонований консорціумом World Wide Web (W3C) стандарт побудови мов розмітки ієрархічно структурованих даних для обміну між різними додатками, зокрема, через Інтернет. XML документ складається із текстових знаків, і придатний до читання людиною.

Стандарт XML визначає набір базових лексичних та синтаксичних правил для побудови мови описання інформації шляхом застосування простих тегів. Запропонований стандарт визначає метамову, на основі якої, шляхом запровадження обмежень на структуру та зміст документів визначаються специфічні, предметно-орієнтовані мови розмітки даних. Ці обмеження описуються мовами схем (Schema), таких як XML Schema (XSD), DTD або RELAX NG. Прикладами мов, оснований на XML є: XSLT, XAML, XUL, RSS, MathML, GraphML, XHTML, SVG, тощо.

XPath (XML Path Language) - це мова виразів для адресації частин XML документу, або для обчислення величин (рядкових, числових або булевих) на основі вмісту XML документа. Мова XPath основана на представленні XML документа у вигляді дерева, і надає можливість навігації всередині дерева, вибирати вузли за різними критеріями. XPath був створений щоб організувати доступ до елементів документа XML з файлу стилів XSLT.

Extensible Stylesheet Language Transformations, або XSLT - функціональна повна за Тюрінгом мова програмування, яка використовується для програмування переробки XML документів. При цьому вихідний документ не змінюється, натомість, на основі результатів переробки створюється новий. Новий документ може бути

серіалізовано (виведено) обробником в стандартний синтаксис XML, або інший формат, такий як HTML або простий текст. Найчастіше, XSLT використовується для перетворення структурованих XML документів із однієї XML схеми в іншу, або для перетворення у веб сторінки або PDF документи.

1.3 Завдання

1.3.1 Ознайомитися з синтаксисом мови XML.

1.3.2 Навчитися формувати валідну схему документа DTD та асоціювати її з документом XML.

1.3.3 Навчитися формувати валідну схему XML Schema та асоціювати її з документом XML.

1.3.4 Порівняти можливості XML Schema та DTD.

1.3.5 Розібратися з виконанням XPath виразів та їхніми результатами.

1.3.6 Розібратися з XSLT перетвореннями та інтеграцією XSLT та XML.

1.3.7 Навчитися зберігати вихідні дані в єдиному розміщенні та подавати їх в декількох представленнях на різних веб-сторінках.

Необхідні файли для роботи: amazon.xml, amazon.xsd, books.xml, amazon-extended.xml.

1.4 Порядок виконання лабораторної роботи

1.4.1 Знайомство з синтаксисом XML та DTD

1. Відкрийте файл «amazon.xml» за допомогою EditX.

Цей файл описує невелику підмножину книжок доступних в інтернет-магазині www.amazon.com.

Опис кожної книжки містить:

- категорію, до якою належить книжка,
- назву,
- список авторів,
- користувацький рейтинг,
- вартість.

Примітка. Короткий практичний посібник щодо XML знаходиться за адресою: <http://www.w3schools.com/xml/default.asp>

2. Перевірити, чи є відкритий XML документ правильно сформованим (тобто синтаксично вірним). Скорегуйте потенційні помилки.

Замість ручної перевірки документу використовуйте функцію “Check Document” програми EditX.

3. Побудуйте DTD документ, що правильно описує XML документ «amazon.xml», та зв'яжіть їх (оголошіть асоціацію в XML документі). Створений документ назвіть amazon.dtd.

Примітка. Короткий практичний посібник щодо DTD знаходиться за адресою: <http://www.w3schools.com/dtd/default.asp>.

Ці завдання повинні бути виконані в середовищі EditX. Зверніть особливу увагу на оголошення входження кожного з елементів. Перед тим як зв'язати файли переконайтесь, що DTD документ не містить помилок. Для зв'язування файлів скористайтесь функцією “Assign DTD to document” програми EditX.

4. Розширте DTD таким чином, щоб він включав додаткові типи товарів окрім книжок (наприклад, програмне забезпечення, відео ігри, одяг, продукти домашнього вжитку, тощо). Ви можете додати реальні типи, що містяться на сайти магазину, чи вигадати власні.

5. Додайте до XML документу принаймні по 2 елементи кожного з типів доданих на попередньому кроці.

1.4.2 Розширений синтаксис XML та DTD

1. Оберіть предметну область за власним бажанням (наприклад, інтернет-магазин, бібліотека, рецепти) та визначте її базові елементи (наприклад, бібліотека має книжки, персонал, тощо) та їхні атрибути (наприклад, книжка має назву, список авторів, тощо). Намалюйте невелику діаграму, що містить всі визначені елементи з їхніми атрибутами та зв'язки між елементами.

Включіть до моделі якнайменше 3 але не більше 7 елементів. На кількість атрибутів немає обмежень, але уникайте вичерпного опису.

2. За допомогою програми EditX побудуйте DTD документ, що описує змодельовану предметну область.

Чи може будь-який з атрибутів бути оголошений як елемент та навпаки? Який шлях моделювання ви обрали та чому?

Примітка. Скористайтесь наступним посиланням для ідей: http://www.w3schools.com/dtd/dtd_el_vs_attr.asp.

Зверніть особливу увагу на оголошення входження кожного з елементів.

Намагайтесь включити до DTD атрибути ID та IDREF(S) в додаток до REQUIRED атрибутів.

Переконайтесь, що фінальний створений DTD документ не містить помилок.

Примітка. Дивіться приклади документів DTD для наступних ідей щодо моделювання предметної області за наступним посиланням: http://www.w3schools.com/dtd/dtd_examples.asp

3. За допомогою програми EditX побудуйте XML документ, що відповідає побудованому DTD документу. Оголосіть всередині XML документу асоціацію з DTD документом.

Не створюйте більше двох екземплярів кожного з елементів DTD документу. Для зв'язування файлів скористайтесь функцією “Assign DTD to document” програми EditX.

4. Перевірте правильність створеного XML документу. Скорегуйте потенційні помилки.

Примітка. Ви можете скористатися програмою EditX для валідації документу або спробувати он-лайн валідатори, наприклад такий як: <http://www.xmlvalidation.com/>

1.4.3 Знайомство зі схемою XML Schema

1. Відкрийте файли «amazon.xml», «amazon.dtd» та «amazon.xsd» за допомогою EditX.

Файл «amazon.xsd» - це документ типу XML Schema, що описує частину структури «amazon.xml», тобто XML документу, використаного в п. 1.4.1. XML Schema – це більш потужна та багата альтернатива DTD. Документ XML може мати посилання на DTD чи на XML Schema.

Огляньте документи «amazon.dtd» та «amazon.xsd» та коротко порівняйте їх. Зверніть особливу увагу на data facets (restrictions on data) та data patterns (data formats).

Примітка. Скористайтесь наступним посиланням, щоб побачити доречні приклади зв'язування: http://www.w3schools.com/Schema/schema_howto.asp.

2. Завершіть документ XML Schema додавши недостатню інформацію.

Примітка. У документі відсутній змішаний складний тип, що описує елемент «книга». Не забудьте додати атрибут «категорія».

Елементи змішаного складного типу можуть мати атрибути, елементи та текст. Вивчить їх скориставшись посиланням: http://www.w3schools.com/schema/schema_complex_mixed.asp

3. Модифікуйте XML документ «amazon.xml» оголосивши асоціацію з документом XML Schema (змінивши посилання на DTD документ).

4. Перевірте правильність створеного XML Schema документу.

Примітка. Ви можете скористатися програмою EditX для валідації документу або спробувати он-лайн валідатори, наприклад такий як: <http://www.w3.org/2001/03/webdata/xsv>.

1.4.4 Розширені можливості XML Schema

1. Оберіть XML документ побудований в п. 1.4.2 .

2. Скористайтесь програмою EditX для ручної побудови документу XML Schema, що описує обрану предметну область.

Щоб побудувати схему ви можете просто прослідкувати структуру XML документу та визначити кожний елемент, що зустрівся в ній. Цей метод є дуже простим та наївним але є важливим для розуміння XML Schema.

Примітка. Розгляньте наступний приклад XML Schema: http://www.w3schools.com/schema/schema_example.asp.

Візьміть до уваги простий та складний типи елементів та оголосіть кількість можливих входжень елементу за допомогою атрибутів maxOccurs та minOccurs.

Не забудьте почати документ зі стандартного XML оголошення за яким розташовано елемент, що визначає схему документу:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
...
</xs:schema>
```

3. Модифікуйте свій XML документ оголосивши асоціацію зі створеним документом XML Schema.

4. Перевірте правильність створеного XML Schema документу.

5. За допомогою програми EditX створіть документ XML Schema, що описує обрану вами предметну область.

Попередній метод побудови є дуже простим, але побудований за його допомогою схема складного документу може бути складною для читання та підтримки. Альтернативний метод побудови заснований на початковому визначенні всіх елементів та їхніх атрибутів та наступному посиленні на них за допомогою атрибута `ref`.

Примітка. Дивіться приклад XML Schema sample (Divide the Schema): http://www.w3schools.com/schema/schema_example.asp.

6. Модифікуйте свій XML документ оголосивши асоціацію зі створеним документом XML Schema.

7. Перевірте правильність створеного XML Schema документу.

1.4.5 Знайомство з XPath

1. Відкрийте XML файл «books.xml» в програмі EditX та виконайте наступні вирази XPath. Який результат кожного з виразів?

- `catalog/book`
- `//author`
- `//book/title`
- `//book/@id`

2. Виконайте наступні вирази XPath. Який результат кожного з них?

- `catalog/book[2]`
- `catalog/book[last()]`
- `catalog/book[last()-2]`
- `catalog/book[position()<3]`
- `catalog/book[price>10]`
- `catalog/book[price>10]/title`
- `catalog/book[@id]`
- `catalog/book[@id="bk102"]`

Такі вирази, що містять предикати, використовуються для отримання певних вузлів (nodes) чи вузлів, що містять певні значення.

Предикати завжди включаються в квадратні скобки.

3. Виконайте наступні вирази XPath та дослідіть їхні результати..

- `catalog/*`
- `//*`
- `//book[@*]`

- //book/node()
- //book[2]/node()
- //book/title | //book/price

Груповий символ * використовується для вибору невідомих XML елементів.

Оператор "|" у виразі XPath отримує декілька шляхів.

1.4.6 Розширені запити XPath

1. Відкрийте XML файл «books.xml» в програмі EditX та виконайте наступні вирази XPath. Який результат кожного з виразів?

- child::book
- attribute:: id
- • child::*
- • attribute::*
- • child::text()
- • child::node()
- • descendant::book
- • ancestor::book
- • ancestor-or-self::book
- • child::* / child::price

Всі приведені виразу вимагають визначення поточного вузла.

Виконуйте вирази кожен раз з поточного вузла (“current node”, визначається відповідною кнопкою вибору в EditX).

Зверніть особливу увагу на вибір поточного вузла.

2. Виконайте наступні вирази XPath. Який результат кожного з них?

- //book | //author
- catalog/book[price!=5.95]
- catalog/book[price>=4.95]
- catalog/book[price=4.95 or price=5.95]
- catalog/book[price>=4.95 and price<10]
- count(//book[price>10])
- //*[name()='title']
- //*[starts-with(name(),'p')]
- //*[contains(name(),'p')]
- //*[string-length(name()) >6]

1.4.7 Знайомство з XSLT

1. Відкрийте файл «amazon-extended.xml» за допомогою EditX . Цей файл містить 2 типи продуктів: книги та програмне забезпечення. Для цього завдання ви також можете використати власний створений XML документ.

Створіть XSL transformation document для представлення всіх продуктів XML файлу. Перетворіть XML документ за допомогою створеного XSLT. Збережіть створене XSL перетворення під назвою «all-products.xsl». Огляньте кінцевий HTML документ в браузері.

Новий XSLT документ повинний бути створений з програми EditX шляхом використання команди «File > New > XSL Transformation for HTML output». Ви повинні використати команду «Transform using XSLT» (з меню «XSLT/XQuery») визначаючи вихідний HTML.

2. Зв'яжіть початковий XML файл зі створеним файлом «all-products.xsl».

Відкрийте XML файл у браузері. Якщо браузер, що ви використовуєте є XML- та XSLT-сумісним, він застосує XSL перетворення до файл та відповідно відобразить його вміст. Це відображення повинно бути ідентичним до створеного на попередньому кроці HTML файлу.

Спробуйте внести будь-які зміни до XML документу та поновіть вікно браузера: результат повинен відразу оновитися .

1.4.8 Розширені можливості XSLT

1. Модифікуйте файл «all-products.xsl» додавши до нього можливості сортування.

Відсортуйте продукти за назвою та ціною. Збережіть оновлення у вигляді двох нових XSLT файлів «all-products-title.xsl» та «all-products-price.xsl».

Зверніть увагу, що відбувається при застосуванні елементу «<xsl:sort>» в числових значеннях, як наприклад ціна.

Примітка. Якщо нам потрібно, щоб XML елемент оброблявся XSLT як реальне числове значення, необхідно, щоб інструкція «<xsl:sort>» мала додаткову інформацію:

```
<xsl:sort select="price" data-type="number"/>
```

2. Відкрийте файл «amazon-extended.xml» чи створений

власноручно.

Створіть XSL transformation document для:

- сортування всіх продуктів за ціною в порядку зменшення;
- сортування всіх продуктів спочатку за ціною, а потім за назвою (два ключа сортування в цьому порядку);
- сортування всіх продуктів за категорією;
- відображення всіх продуктів з ціною > 15;
- відображення всіх продуктів з ціною > 15 червоним шрифтом, а всіх інших зеленим;
- відображення всіх продуктів з ціною > 25 червоним шрифтом, всіх продуктів з ціною > 15 оранжевим, а решти - зеленим.

Для виконання цього завдання необхідно скористатися командою «Transform using XSLT» в меню «XSLT/XQuery» для формування вихідного HTML.

1.4.9 Представлення XML

1. Створіть 3 нових XML файли, що задовольняють наступному::

- all-products.xml застосовує all-products.xsl.
- all-products-title.xml застосовує all-products-title.xsl.
- all-products-price.xml застосовує all-products-price.xsl.

Всередині кожного з XML файлів включіть початковий XML («amazon-extended.xml») в якості ENTITY та створіть кореневий елемент «amazon».

До цих чотирьох XML файлів необхідно внести синтаксичні зміни.

В файлі «amazon-extended.xml» необхідно видалити посилання на DTD. Чому?

Які ще зміни необхідно внести до 3 інших XML файлів?

Примітка. Якщо ви зв'яжете 3 XML файли з файлом «amazon.dtd», невелика модифікація буде необхідна в початковому XML. Яка саме це буде модифікація?

Коли всі необхідні зміни будуть зроблені, ви будете мати централізований репозиторій даних (amazon-extended.xml) та різні його представлення (3 XML файли).

2. Створіть простий веб-сайт, заснований на фреймовому представленні, що відобразить створений раніше контент. Веб-сайт

матиме зліва навігаційних фрейм зі списком посилань та справа другий фрейм для відображення вмісту зв'язаних документів.

Примітка. Приклад роботи з фреймами можна знайти за посиланням: http://www.w3schools.com/html/html_frames.asp

3. Яка серія дій тепер необхідна для простого оновлення каталогу продуктів? (наприклад, додавання нових продуктів в список).

1.5 Зміст звіту

1.5.1 Тема та мета роботи.

1.5.2 Короткі теоретичні відомості.

1.5.3 Результати роботи.

1.5.4 Висновки, що відображують результати виконання роботи та їх критичний аналіз.

ЛАБОРАТОРНА РОБОТА № 2

ПРЕДСТАВЛЕННЯ ДАНИХ В ФОРМАТІ RDF

2.1 Мета роботи

Ознайомитися з моделлю даних RDF та її можливостями, отримати практичні навички роботи з нею. Змодельовати певну область знань за допомогою мови RDF. Навчитися користуватися словниками метаданих при моделюванні.

2.2 Короткі теоретичні відомості

RDF – це розроблена консорціумом W3C технологія семантичної павутини, яка включає в себе середовище опису ресурсів (Resource Description Framework, RDF), визначає загальну архітектуру метаданих і призначена для забезпечення сумісності метаданих за допомогою спільної семантики, структури та синтаксису. Технологія семантичної мережі передбачає розширення можливостей інтернету завдяки механізмам надання інформації чітко визначеного значення, яке дозволяє ефективно використовувати її у спільній роботі як комп'ютерів, так і людей.

RDF – це модель для представлення даних та особливо метаданих. RDF представляє твердження про ресурси у вигляді , придатному для машинної обробки .

Ресурсом в RDF може бути будь-яка сутність - як інформаційна (наприклад , веб-сайт або зображення) , так і неінформаційних (наприклад, людина , місто чи якесь абстрактне поняття) . Твердження, що висловлюється про ресурс , має вигляд « суб'єкт - предикат - об'єкт » і називається триплетом. Твердження « небо блакитного кольору » в RDF - термінології можна представити таким чином : суб'єкт - «небо» , предикат - « має колір » , об'єкт - « блакитний ». Для позначення суб'єктів , предикатів і об'єктів в RDF використовуються URI .

Множина RDF- тверджень утворює орієнтований граф , в якому вершинами є суб'єкти та об'єкти , а ребра помічені предикатами .

RDF сам по собі є не форматом файлу , а тільки лише абстрактною моделлю . Для запису і передачі RDF використовується кілька форматів, у тому числі:

RDF / XML - запис у вигляді XML - документа;

RDF / JSON - запис у вигляді JSON - даних;

RDFa (RDF in attributes) - запис всередині атрибутів довільного HTML - або XHTML - документа;

N- Triples , Turtle , N3 - компактні форми запису тверджень.

Для обробки даних, представлених в RDF , можуть використовуватися мови запитів : SPARQL (стандарт W3C) , RQL , RDQL .

SPARQL (рекурсивний акронім від SPARQL Protocol and RDF Query Language) - мова запитів до даних , що представлені за моделлю RDF , а також протокол для передачі цих запитів і відповідей на них. SPARQL є рекомендацією консорціуму W3C і однією з технологій семантичної павутини. Надання SPARQL - точок доступу (SPARQL - endpoint) є рекомендованою практикою при публікації даних у всесвітній павутині.

RDF надає засоби для побудови інформаційних моделей , але не стосується семантики того, що описується . Граф RDF взятий окремо можна розуміти тільки як граф . Тлумачення значення ґрунтується на здатності користувачів RDF інтерпретувати URI , рядкові літерали і структуру графа.

Для вираження семантики потрібні словники (vocabularies) , таксономії (taxonomies) і онтології (ontologies) .

Словник представляє з себе зібрання певних термінів, що використовуються в інформації та мають однаковий зміст у всіх контекстах .

Таксономія - це словник ієрархічно організованих термінів.

Онтологія використовує зумовлений зарезервований словник термінів для визначення концепцій і відносин між ними для конкретної предметної області. Онтології можна використовувати для вираження семантики термінів словника , їх взаємин і контекстів використання.

RDF Schema надає спеціальний словник для RDF , який може бути використаний для визначення таксономій класів , властивостей , контейнерів , реіфікації (Reification - втілення) . Для запису семантики предметних областей в онтологіях служить мова OWL .

Дублінське ядро (Dublin Core) - стандарт метаданих (формат метаданих) , простий і ефективний набір для опису найширшого діапазону мережевих ресурсів .

Стандарт розділений на два рівні:

- простий (некваліфікований , simple), що складається з 15 елементів ;
- компетентний (кваліфікований , qualified), що складається з 18 елементів і групи т. зв. тонкощів (або кваліфікаторів) , які уточнюють семантику елементів для підвищення корисності пошуку ресурсів .

Семантика Дублінського ядра була створена міжнародною міждисциплінарною групою професіоналів бібліотечної справи , комп'ютерних наук , кодування текстів , музейної справи та інших суміжних груп .

2.3 Завдання

- 2.3.1 Ознайомитися з синтаксисом RDF та його особливостями.
- 2.3.2 Навчитися конструювати RDF документи.
- 2.3.3 Навчитися формувати валідну схему RDF Schema.
- 2.3.4 Ознайомитися з середовищем MR3 та навчитися моделювати в ньому предметні області.
- 2.3.5 Ознайомитися зі словником метаданих Дублінське ядро.
- 2.3.6 Включити терми (terms) Дублінського ядра в додаток за допомогою середовища MR3.
- 2.3.7 Ознайомитися з генераторами метаданих Дублінського ядра.
- 2.3.8 Ознайомитися з директорією словників SchemaWeb.
- 2.3.9 Побудувати додаток на основі словників третьої сторони.
- 2.3.10 Навчитися використовувати переваги інструментів метаданих при побудові веб-додатків.
- 2.3.11 Ознайомитися з мовою запитів SPARQL.

Необхідні файли для роботи: (optional) amazon-extended.xml, amazon.mr3, amazon-books.mr3

2.4 Порядок виконання лабораторної роботи

2.4.1 Знайомство з синтаксисом RDF

1. Для прикладу з товарами інтернет-магазину «Amazon» визначте ресурси, властивості та їхні значення.

RDF – це фреймворк для опису веб-ресурсів, таких як назва, автор, дата модифікації, вміст та відомості про авторське право веб-сторінки.

- Ресурс – це все, що може мати URI.
- Властивість – це Ресурс, що має ім'я, наприклад, «автор».
- Значення Властивості – це наприклад, «Жан Поль».

Примітка. Дивись наступне посилання для визначень: http://www.w3schools.com/rdf/rdf_rules.asp

2. Побудуйте RDF документ, що коректно опише приклад з інтернет-магазину.

Для виконання цього завдання ви можете використати будь-який текстовий редактор, такий як TextPad, чи будь-який XML-редактор, такий як Editix.

Примітка. Використовуйте default example prefix (xmlns:ex=«[#»](http://www.example.org)).

Комбінація ресурсу, властивості та значення властивості формує твердження (statement), що має subject, predicate та object. Прикладом твердження може бути «Автором <http://www.sample.gr> є Жан Поль».

- Суб'єктом твердження є <http://www.sample.gr>
- Предикат – автор;
- Об'єкт – Жан Поль.

Спрощений приклад RDF для приведеного прикладу буде:

```
<rdf:RDF>
<rdf:Description rdf:about="http://www.sample.gr">
<ex:author>Jan Paul</ex:author>
</rdf:Description>
</rdf:RDF>
```

Примітка. Невеликий приклад RDF дивіться за посиланням: http://www.w3schools.com/rdf/rdf_example.asp.

3. Перевірте правильність створеного RDF документу.

Замість ручної перевірки документу ви можете скористатися онлайн валідатором за посиланням :<http://www.w3.org/RDF/Validator/>. Цей RDF Валідатор розбирає ваш RDF документ, перевіряє його синтаксис та генерує його табличне та графічне представлення..

Примітка. Результати перевірки будуть виглядати наступним образом: <http://www.w3schools.com/rdf/RDFvalidationresult.htm>

2.4.2 Розширений синтаксис RDF

1. Розширте створений RDF документ, включивши до нього інші типи продуктів окрім книжок та програмного забезпечення.

Розширте свій RDF документ включивши до моделі елементи програмного забезпечення типу «software1» та «software2». Включіть до моделі якнайменше 2 та не більше ніж 5 елементів.

2. Перевірте правильність створеного RDF документу.

3. Використайте RDF контейнери для поліпшення створеної RDF моделі. RDF контейнери використовуються для опису груп елементів.

Скористайтеся наступними RDF елементами: <Bag>, <Seq> та <Alt>:

- елемент <rdf:Bag> використовується для опису списку значень, що не повинні бути в спеціальному порядку.

- елемент <rdf:Seq> використовується для опису впорядкованого списку значень (наприклад, в алфавітному порядку).

- елемент <rdf:Alt> використовується для опису списку альтернативних значень (користувач може обрати лише одне з значень).

Примітка. Короткі приклади можна знайти за посиланням: http://www.w3schools.com/rdf/rdf_containers.asp.

4. Перевірте правильність створеного RDF документу.

5. Використайте RDF колекції (collections) для поліпшення створеної RDF моделі.

Треба зазначити, що контейнер вказує, що ресурси, що в ньому містяться, є членами, але він не вказує, що інші члени заборонені. Таким чином, RDF колекції використовуються для опису груп, що містять тільки вказаних членів.

Примітка. Колекція описується атрибутом `rdf:parseType=«Collection»`.

Примітка. Короткий приклад знаходиться за посиланням: http://www.w3schools.com/rdf/rdf_collections.asp.

6. Перевірте правильність створеного RDF документу.

2.4.3 Знайомство з RDF Schema

1. Вивчіть приклад з інтернет-магазину та побудуйте для нього діаграму RDF Schema.

RDF Schema надає фреймворк для опису класів та властивостей, специфічних для конкретного додатку.

Класи в RDF Schema є значно схожими на класи в об'єктно-орієнтованому програмуванні. Це дозволяє визначати ресурси як екземпляри класів та підкласів.

Примітка. Дивіться наступне посилання, за яким можна знайти інформацію про RDF Schema: http://www.w3schools.com/rdf/rdf_schema.asp.

2.3а допомогою середовища MR3 побудуйте створену діаграму RDF Schema.

Створіть новий проект в MR3 (amazon.mr3). Використайте вікно "Class Editor" для визначення класів і вікно «Property Editor» для визначення властивостей.

Примітка. Для редагування атрибутів класів та властивостей скористайтеся вікном «Attribute Dialog» (меню можна визвати за допомогою правої кнопки миші).

Примітка. Зверніть увагу на відношення суперклас/підклас та супервластивість/підвластивість.

3. Додайте екземпляри RDF до створеного MR3 проекту.

Для цього скористайтеся вікном "RDF Editor". Додайте екземпляр "book1".

2.4.4 Розширені можливості RDF Schema

1. Відкрийте файл "amazon.mr3" за допомогою програми MR3.

Файл описує приклад інтернет-магазину, включаючи діаграми класів та властивостей. Крім того, він описує екземпляр RDF (book1).

2. Розширте цей проект, включивши до нього решту RDF екземплярів книжок.

Виконайте це завдання за допомогою вікна "RDF Editor" програми MR3.

3. Оновіть властивості ціна та рейтинг, змінивши їхній тип з Literals на Numeric.

Не забудьте оновити всі екземпляри книжок.

Примітка. Використайте вікно "RDF Editor" для створення нового екземпляру (наприклад, ціни) та включіть новий тип даних до щойно розробленого Literal.

4. Розширте проект, включивши елементи програмного

забезпечення в якості підкласів класу “AmazonItem”.

Розширте схему RDF Schema включивши до моделі елементи типу “software1” та “software2”, використані в п.2.4.2 . Включіть до моделі якнайменше 2 та не більше ніж 5 елементів.

Примітка. Використайте вікно "Class Editor" для визначення нових класів та їх відношень.

5. Визначте властивості елементів програмного забезпечення.

Розширте схему RDF Schema включивши до моделі властивості елементів програмного забезпечення.

Примітка. Використайте вікно "Property Editor" для їхнього визначення та зв'язку.

Примітка. Зверніть увагу, що спільні властивості книжок та програмного забезпечення повинні бути визначені лише один раз.

Таким чином, вам необхідно модифікувати їхній домен та оновити його до “AmazonItem”.

6. Розширте проект, щоб він включав нові RDF екземпляри типу програмного забезпечення. Виконайте завдання за допомогою вікна "RDF Editor" середовища MR3.

7. Модифікуйте властивість “author” як новий клас, підклас класу “person”.

Примітка. Для цього використайте вікна "Class Editor" та "Property Editor".

8. Оновіть створений проект коректно перетворивши його властивості.

Примітка. Наприклад, повинна бути визначена нова властивість об'єкту “hasAuthor” .).

2.4.5 Використання словника метаданих Дублінське ядро

1. Відкрийте файл "amazon-books.mr3" за допомогою MR3.

2. В мережі Інтернет існує велика кількість словників метаданих для використання в створюваних додатках. Наприклад, ознайомтеся зі словником Дублінське ядро (DC) за посиланням: <http://dublincore.org/documents/dcmi-terms/#H2>. Розгляньте включення термів Дублінського ядра до файлу "amazon-books.mr3".

Властивості, що знаходяться в ньому, можуть бути використані з вашого додатку. Набір метаданих Дублінське ядро надає фундаментальну групу текстових елементів через які може бути

описана більшість ресурсів (тобто фізичних ресурсів, цифрових матеріалів та складного медіа). Записи метаданих засновані на DC задумані бути крос-доменними описами та стали стандартом в бібліотечній справі та комп'ютерних науках.

3. Оновіть файл "amazon-books.mr3" за допомогою DC термів. Ви можете додати інформацію до книжок (за допомогою додаткових елементів) чи замінити існуючі елементи DC термами.

Не забудьте оголосити простір імен DC всередині файлу за допомогою таблиці NameSpace програми MR3!

Застосуйте якнайменше 3 DC терми у вашому документі.

Примітка. Приклад дивіться за наступним посиланням: http://www.w3schools.com/rdf/rdf_dublin.asp.

4. В Інтернеті можна знайти певну кількість генераторів метаданих DC. Протестуйте наступні два інструменти:

<http://www.ukoln.ac.uk/metadata/dcdot/> and

http://www.webarchiv.cz/generator/dc_generator.php?lang=en

Введіть URL (наприклад, <http://www.zntu.edu.ua/>) та вивчіть отримані вихідні дані в форматі RDF.

5. Увійдіть на сторінку Stanford's DC Metadata Generator за посиланням <http://soap.stanford.edu/plugins/dublincore/> та створіть простий елемент.

Stanford's DC Metadata Generator створює RDF файли, що містять DC терми, що описують ресурс. Створений фрагмент може бути включений до інших додатків.

2.4.6 Використання інших словників метаданих

1. Ознайомтеся з SchemaWeb за посиланням: <http://www.schemaweb.info/>. SchemaWeb – це директорія RDF схем, виражених в RDFS, OWL та DAML+OIL.

Розгляньте схеми за посиланням: <http://www.schemaweb.info/schema/BrowseSchema.aspx>.

Деякі з популярних словників це:

- FOAF (Friend-Of-A-Friend), що описує особи, їх діяльність та відношення з іншими людьми та об'єктами ;

- SKOS (Simple Knowledge Organization System), - це RDF словник, що виражає концептуальні схеми, такі як тезауруси, схеми

класифікацій, таксономії та інші типи контрольованих словників, а також концептуальні схеми, включені до глосаріїв та термінологій.

- VCard, що описує профіль електронних візитівок.

2. Знайдіть "Pet Profile Ontology", що описує домашніх улюбленців та їхні характеристики. Включіть її простір імен до MR3 та створіть просту онтологію, що описує вашого улюбленця.

Якщо у вас немає улюбленця, вигадайте одного!

Зверніть увагу, що терми онтології конкретного улюбленця використовує FOAF словник. Таким чином, якщо ви застосовуєте специфічні терми, ви повинні також включити простір імен FOAF.

3. Сайт-компаньон для "Pet Profile Ontology" розташований за посиланням: <http://hyperdata.org/xmlns/pets/>. Скористайтесь Pet-a-matic metadata generator та заново створіть свого улюбленця через онлайн форму. Збережіть створений код у файлі "pet.rdf".

Існує схожий додаток (FOAF-a-Matic) для автоматичної генерації профілів FOAF, що доступний за посиланням: <http://www.ldodds.com/foaf/foaf-a-matic.html>.

Вивчіть як ви можете опублікувати FOAF профіль та профіль улюбленця у веб. Це є перший крок до "practicing what you preach".

2.4.7 Виконання запитів до RDF даних за допомогою SPARQL

1. Відкрийте файл RDF "amazon-books+software.rdf" із SPARQL GUI, що є частиною програми SW Workbench, та виконайте наступні SPARQL запити. Що є результатом кожного з запитів?

- Отримати ID продукту для всіх ігор (наприклад, "software2").
- Отримати ID та назви всіх продуктів.
- Отримати назви та ціни всіх книжок.
- Отримати назви та ціни всіх наукових книжок.

Не забудьте визначити необхідні префікси в SPARQL запитах.

Примітка. Специфікація мови запитів SPARQL знаходиться за посиланням: <http://www.w3.org/TR/rdf-sparql-query/>

2. Виконайте наступні SPARQL запити. Що є результатом кожного з запитів?

- Отримати ID та назви всіх продуктів, відсортувати результат за назвою.

- Отримати назви та ціни всіх книжок, відсортувати результат за категорією.
- Отримати назви та ціни всіх книжок, відсортувати результат за ціною.
- Отримати назви та ціни всіх книжок, які дешевші за \$30.
- Отримати назви та (опціонально) рейтинг всіх книжок.
- Отримати назви та ціни всіх книжок та програмного забезпечення.

Примітка. В останньому завданні доречним буде використати ключове слово UNION.

3. Виконайте наступні SPARQL запити. Що є результатом кожного з запитів?

- Отримати назви та (опціонально) рейтинг всіх книжок , рейтинг є більшим 4.
- Отримати ID та назви всіх продуктів, відсортувати результат за ціною в порядку зменшення.
- Отримати назви та ціни двох товарів, що є найдешевшими.
- Перечислити всіх категорії продуктів, дублікати не допускаються.
- Отримати назви всіх продуктів програмного забезпечення, що сумісні з будь-якою платформою Windows.

Примітка. Скоріш за все вам буде необхідно використати регулярні вирази в деяких з цих завдань.

2.5 Зміст звіту

2.5.1 Тема та мета роботи.

2.5.2 Короткі теоретичні відомості.

2.5.3 Результати роботи.

2.5.4 Висновки, що містять відповіді на контрольні запитання, а також відображують результати виконання роботи та їх критичний аналіз.

ЛАБОРАТОРНА РОБОТА № 3 **МОВА ПРЕДСТАВЛЕННЯ ОНТОЛОГІЙ OWL**

3.1 Мета роботи

Ознайомитися з онтологіями та логічними висновками. Ознайомитися з мовою представлення онтологій OWL та отримати практичні навички роботи з нею.

3.2 Короткі теоретичні відомості

Онтологія – це формальний явний опис понять (класів) аналізованої предметної області, властивостей кожного поняття, що описують різні властивості і атрибути поняття (слотів, іноді їх називають ролями), і обмежень, накладених на слоти (фацетів, іноді їх називають обмеженнями ролей). Онтологія разом з набором індивідуальних примірників класів утворює базу знань.

Зазвичай така концептуальна схема предметної області складається з структури даних, що містить всі релевантні класи об'єктів, їхні зв'язки і правила (обмеження), прийняті в цій області.

OWL (Web Ontology Language) – мова опису онтологій для семантичної павутини. Мова OWL дозволяє описувати класи і відносини між ними, властиві для веб-документів і застосунків. OWL є рекомендованим консорціумом Всесвітньої павутини.

В основі мови – уявлення дійсності в моделі даних «об'єкт - властивість». OWL придатна для опису не тільки веб-сторінок, але і будь-яких об'єктів дійсності. Кожному елементу опису в цій мові (в тому числі властивостями, що зв'язує об'єкти) ставиться у відповідність URI.

Уніфікований ідентифікатор ресурсів (Uniform Resource Identifier, URI) - короткий рядок літер, який однозначно ідентифікує окремий ресурс. Основне призначення таких ідентифікаторів — зробити можливою взаємодію з представленнями ресурсів через мережу, використовуючи спеціальні протоколи. URI визначається схемами, які визначають синтаксис та відповідні протоколи.

OWL має три діалекти (у порядку зростання виразності).

OWL Lite призначена для користувачів, які потребують передусім класифікаційної ієрархії і простих обмежень.

OWL DL призначена для користувачів, яким потрібна максимальна виразність при збереженні повноти обчислень (всі логічні висновки, що припускаються тією чи іншою онтологією, будуть гарантовано обчислюваними) і розв'язуваності (всі обчислення завершаться за певний час). OWL DL включає всі мовні конструкції OWL, але вони можуть використовуватися лише згідно з певним обмеженням (наприклад, клас може бути підкласом багатьох класів, але не може сам бути представником іншого класу). OWL DL так названий з-за його відповідності дескрипційній логіці – дисципліні, в якій розроблені логіки, що складають формальну основу OWL.

OWL Full призначена для користувачів, яким потрібна максимальна виразність і синтаксична свобода RDF без гарантій обчислення.

Описові логіки (Description logics, іноді ще їх називають дескрипційними логіками) – сімейство мов представлення знань, що дозволяють описувати поняття предметної області в недвозначному, формалізованому вигляді. Вони поєднують в собі, з одного боку, багаті виражальні можливості, а з іншого - хороші обчислювальні властивості, такі як розв'язність і відносно невисока обчислювальна складність основних логічних проблем, що робить можливим їх застосування на практиці. Таким чином, описові логіки представляють собою компроміс між виражальністю і розв'язністю. Описову логіку можна розглядати як розв'язні фрагменти логіки предикатів, синтаксично ж вони близькі до модальних логік.

3.3 Завдання

3.3.1 Ознайомитися з властивостями типів даних (Datatype) та об'єктів мови OWL.

3.3.2 Ознайомитися з популярною інженерною методологією “Ontology Development 101”.

3.3.3 Навчитися будувати та повторно використовувати онтології вручну.

3.3.4 Ознайомитися з системами логічного висновку.

3.3.5 Ознайомитися з "Manchester Syntax".

3.3.6 Ознайомитися з середовищем Protégé 4.0 та його можливостями для створення ієрархій і онтологій та логічним висновком.

Необхідні файли для роботи: orders.owl, people.owl pets.owl

3.4 Порядок виконання лабораторної роботи

3.4.1 Знайомство з Protégé

1. Запустіть програму Protégé:

- Оберіть пункт меню "Create New OWL Ontology".
- Кожна онтологія називається за допомогою URI. Замініть стандартний URI на свій власний (наприклад, <http://www.example.com/ontologies/orders.owl>).

- Збережіть онтологію під назвою "orders.owl".

Онтологія "orders" буде описувати заклади, доставки, замовників та продукти.

Вкладка "Active Ontology Tab" дозволяє уточнити інформацію про онтологію.

Коли та якщо програма задасть питання про синтаксис онтології, оберіть "Manchester Syntax" (який часто називають "Abstract OWL Syntax", який є більш компактним синтаксисом мови OWL).

Примітка. Програма Protégé доступна за посиланням: <http://protege.stanford.edu/overview/>

2. Додайте нотатки до онтології:

- Натисніть на іконку додавання (+), що знаходиться поруч з кнопкою (?) «Annotations» на вкладці "Active Ontology Tab" та оберіть "comment" зі списку вбудованих URI анотацій

- Введіть свій коментар в правій панелі для опису онтології.

- Всі вставлені анотації будуть відображені у вкладці "Active Ontology Tab".

Додайте декілька анотацій властивостей для онтології, наприклад "creator", "description", тощо.

3. Створіть необхідні класи всередині вкладки "Classes".

Використайте засоби середовища Protégé: "Add Subclass" та "Add Sibling Class".

Пропонуються класи: "Person", "Customer", "Product", "Supplier", "PurchaseOrder".

Створіть якнайменше 7 класів, що погоджуються з ієрархією класів, що лежить в основі.

Примітка. Зазвичай для іменування класів використовуються

верблюда нотація (в OWL, Java, тощо).

4. Визначте диз'юнктивність (неперекриття) між створеними класами.

Для цього отримайте доступ до представлення опису класів.

Зверніть увагу, що коли визначається диз'юнктивність в одному класі, ця властивість каскадно включається також до інших класів.

5. Створіть 2 (чи більше) підкласів Продукту Product: "Книжка" та "Програмне забезпечення" за допомогою засобу Protégé "Create Class Hierarchy".

Нажміть на кореневий клас ("Product") та оберіть опцію "Create Class Hierarchy" з меню засобів на панелі меню програми Protégé.

Включіть одразу два нових класи. Оберіть опцію "Make all new classes disjoint".

3.4.2 Створення простої онтології за допомогою Protégé

1. Запустіть програму та завантажте "orders.owl":

- Переключіться до вкладки 'Datatype Properties'.

- Створіть для продуктів нову властивість типу даних з назвою "hasPrice".

- Визначте домен (domain) та діапазон властивості (range).

Домен визначає множину індивідів, до яких застосовується властивість (тобто область визначення), а діапазон визначає множину індивідів, що можуть бути зв'язані цією властивістю з об'єктами з домену (тобто область значень).

Додайте ще 5 властивостей типів даних для кожного об'єкту в онтології.

Використайте засоби програми Protégé: "Add Sub-property" та "Add Sibling Property".

Пропонуються властивості: "hasProductName", "hasAddress", "discount", "firstName", "lastName", тощо.

2. Створіть якнайменше 5 прикладів екземплярів продуктів (книжки та програмне забезпечення) з вкладки 'Individuals' та задайте значення їх властивостей.

Переконайтеся, що кожного разу обираються вірні типи даних властивостей.

3. Створіть новий клас "ExpensiveBook", що є класом-нащадком класу "Book". Коштовні книжки мають ціну ≥ 40 .

- У ‘Class Description view’ нажміть кнопку ‘+’ в ‘Superclasses section’ для додання нового обмеження.
 - У ‘Class expression editor’, напишіть ‘Book that hasPrice some integer[>= 40]’ та натисніть ‘OK’.
 - Сконвертуйте клас на визначений клас (‘Ctrl-D’).
- Таким же чином створіть Дешеву Книгу (CheapBook). Дешеві книжки вартують ≤ 15 .

3.4.3 Розширені властивості онтологій

1. Запустіть програму Protégé та завантажте "orders.owl".
 - Переключитися на вкладку ‘Object Properties’.
 - Створіть нову властивість об’єкту під назвою "shipTo", що визначає асоціацію між доставкою продукту та покупцем.
 - Визначте домен та діапазон властивостей.

Додайте якнайменше 5 властивостей для об’єктів в онтології. Пропонуються властивості: "hasItem", "shippedBy", "billTo", тощо.
2. Створіть декілька (3) властивостей, зворотних створеним на попередньому кроці.
3. Визначте, які зі створених властивостей є функціональними.

Якщо властивість є функціональною, для заданого індивіду може бути максимум один індивід, зв’язаний з ним цією властивістю. Функціональні властивості також відомі як ‘single valued properties’ (однозначні властивості) або ‘features’ (ознаки).

Наприклад, властивість типу даних "hasPrice" є функціональною адже предмет може мати лише одну ціну.

Зробіть те саме для визначення зворотніх функціональних властивостей (зворотня властивість є функціональною) та симетричних властивостей (якщо властивість пов’язує індивіда а з b, то індивід b також пов’язаний з індивідом а цією ж властивістю) де це можливо.
4. Закази визначаються постачальником. Створіть обмеження, що відображує це.

Додайте суперклас до класу "Order".

Оберіть властивість "shippedBy" та переконайтеся, що обмеження існує.

Створіть більше властивостей на основі інших властивостей.

3.4.4 Створення онтологій вручну

1. Визначте домен та діапазон онтології. Рекомендується обрати приклад з інтернет-магазином Amazon.

Для того, щоб створити онтологію, необхідно визначити її домен та діапазон. Для цього необхідно дати відповіді на декілька базових питань:

- Яку предметну область буде покривати онтологія?
- Для чого буде використовуватися онтологія?
- Для відповіді на які типи питань онтологія буде давати інформацію?
- Кто буде використовувати та обслуговувати онтологію?

Більш за те, один з шляхів визначення діапазону онтології – зробити набросок переліку питань, на які повинна мати можливість дати відповідь онтологія. Ці питання згодом стануть засобом тестування онтології на відповідність:

- Чи містить онтологія достатньо інформації для відповіді на такі типи питань?
- Чи потребують відповіді особливого рівня деталізації чи представлення особливої сфери?

2. Розгляньте можливість повторного використання існуючих онтологій.

Майже завжди вартує уваги розгляд того, що вже хтось зробив, та перевірка чи можливо уточнити та розширити існуючі ресурси для вашої особливої задачі. Повторне використання існуючої онтології може бути вимогою якщо ваша система повинна взаємодіяти з іншими додатками, що вже використовують якісь онтології чи контрольовані словники. Багато онтологій доступні в електронному вигляді і можуть бути імпортовані в середовище побудови онтологій, які ви використовуєте.

Існують бібліотеки онтологій в мережі Інтернет та в літературі. Наприклад, ви можете використати бібліотеку онтологій Ontolingua (<http://www.ksl.stanford.edu/software/ontolingua/>) чи бібліотеку DAML (<http://www.daml.org/ontologies/>). Також існує декілька публічно доступних комерційних онтологій, наприклад, UNSPSC (www.unspsc.org), RosettaNet (www.rosettanet.org), DMOZ (www.dmoz.org).

Для прикладу, розгляньте робочу групу ESSI WSMO та її

онтологію Amazon (<http://www.wsmo.org/TR/d3/d3.4/v0.2/#ontology>).

Примітка. Однак, для цієї роботи ми припускаємо, що досі не існує жодної доречної (відповідної) онтології та розпочнемо побудову онтології з самого початку.

3. Перечисліть важливі терми в онтології.

Доречно спочатку записати перелік термів, про які ви хотіли би зробити твердження чи які ви хотіли би пояснити користувачеві.

- Про які терміни ви хотіли би поговорити?
- Які властивості мають ці терміни?
- Що ви хотіли би сказати про ці терміни?

Примітка. Спочатку важливо отримати вичерпний список термів не турбуючись про перетин концепцій, які вони представляють, про відношення між термами чи про властивості, які вони можуть мати, чи про те чи є ці концепції класами чи слотами (slots).

4. Визначте класи та ієрархію класів, тобто визначте таксономію.

Існує декілька можливих підходів до створення ієрархії класів:

- Процес розробки зверху-донизу починається з визначення найбільш загальних концепцій в предметній області та наступної конкретизації концепцій.
- Процес розробки знизу-доверху починається з визначення найбільш конкретних класів, тобто листів ієрархії, з наступним групуванням цих класів в більш загальні концепції.
- Процес розробки як комбінація двох підходів: спочатку ми визначаємо найбільш помітні (визначні) концепції та потім узагальнюємо або конкретизуємо їх.

Жоден з цих трьох методів не є по своїй суті краще за інших. Вибір підходу залежить від особистого погляду розробнику на предметну область.

5. Визначення властивостей класів, тобто слотів.

Класи самі по собі не надають повної інформації, необхідної для відповіді на необхідні питання, визначені на кроці 1. Як тільки ви визначили деякі класи, ви повинні описати внутрішню структуру концепцій.

Примітка. Ви вже обрали класи зі списку визначених термів. Більшість термів, що залишилися, скоріш за все будуть властивостями цих класів.

Примітка. Для кожної властивості зі списку, ви повинні визначити, який клас вона описує. Ці властивості стають слотами класів.

Взагалі, існує декілька типів властивостей об'єктів, що можуть стати слотами онтології:

- внутрішні (притаманні) властивості, такі як присмак вина;
- зовнішні (сторонні) такі як назва вина, область її походження;
- частини, якщо об'єкт є структурованим, ці частини можуть бути фізичними чи абстрактними «частинами», тобто страви вечері;
- відношення до інших індивідів, це відношення між індивідуальними членами класу та іншими елементами (наприклад, виробник вина представляє відношення між вином та винним заводом).

Примітка. Всі підкласи класу наслідують слоти цього класу. Таким чином, слот повинен бути прикріплений до найбільш загального класу, що може мати цю властивість.

6. Визначення фасетів (facets) слотів (slots).

Слоти можуть мати різні фасети, що описують тип даних, дозволені значення, число значень (кардиналитет) та інші властивості значень, які може мати слот.

Примітка. Декілька загальних фасетів:

- Кардиналитет слоту визначає скільки значень може мати слот;
- Тип величини слоту визначає які типи значень можуть заповнювати слот: строка, число (Float или Integer), логічний тип, перелічуваний тип, екземпляр класу;
- Домен та діапазон слоту: дозволені класи для слоту типу екземпляр класу – діапазон слоту, а класи до яких слот прикріплений чи клас, чия властивість описує слот, - домен слоту.

7. Створіть екземпляри.

Створіть індивідуальні екземпляри класів в ієрархії.

Примітка. Визначення індивідуальних екземплярів класу вимагає (1) вибрати клас, (2) створити екземпляр цього класу та (3) заповнити значення слотів.

8. Перевірити аномалії.

Важливою перевагою використання OWL перед RDF Schema є

можливість визначити суперечності в онтології чи в її екземплярах.

Примітка. Загальні аномалії:

- Несумісні домен та діапазон визначення для транзитивних, симетричних чи зворотних властивостей;
- властивості кардинальності;
- вимоги до значень властивостей можуть конфліктувати з обмеженнями домена та діапазону.

3.4.5 Створення повної онтології за допомогою Protégé

1. Запустіть програму Protégé та оберіть пункт "Create New OWL Ontology" щоб створити онтологію Amazon.

Примітка. Кожна онтологія іменується з використанням URI. Змініть стандартний URI власним (наприклад, <http://www.example.com/ontologies/amazon.owl>).

2. Додайте до онтології анотації та визначте класи.

Примітка. Використайте опцію "Active Ontology Tab" для анотації онтології.

Примітка. Використайте засоби програми Protégé ("Add Subclass" та "Add Sibling Class") для визначення класів за допомогою "Class Tab" (див. крок 4 – п. 3.4.4).

3. Визначте властивості та фасети.

Примітка. Використайте засоби програми Protégé ("Add SubProperty" та "Add Sibling Property") для визначення класів за допомогою "Object Properties Tab". (див. кроки 5,6 – п. 3.4.4)

4. Визначте екземпляри онтології Amazon та призначте значення їх властивостей.

Примітка. Використайте "Individuals Tab" для визначення екземплярів (див. крок 7 – п. 3.4.4).

5. Перевірте аномалії.

Використайте середовище Protégé 4.0 для перевірки вашої онтології на аномалії (див. крок 8 – п. 3.4.4).

3.4.6 Базовий логічний висновок на класах та екземплярах

1. Завантажте онтологію "people.owl" та викличте одну з систем логічного висновку (СЛВ, Reasoner), що вбудовано в програму Protégé (Pellet чи Hermit, тощо). Завантажте онтологію до СЛВ через опцію 'Classify' для того, щоб автоматично розрахувати класифікаційну

ієрархію та перевірте логічну погодженість.

Ця ієрархія класів розраховується автоматично та називається ієрархією висновку «inferred hierarchy» (виводима ієрархія).

Всі зміни в онтології відображаються шляхом різних кольорів шрифту та іншими засобами, що залежать від версії програми Protégé.

Якщо клас було віднесено до несумісних, його іконка буде підсвічена червоним кольором.

2. Обґрунтуйте наступні результати логічного висновку:

– Чому зроблено висновок, що "driver" (водій) є супер-класом "bus_driver" (водій автобуса)?

Примітка. Підклас виведено завдяки тому, що підкласи використовуються в квантор існування.

– Чому виведено, що "cat_liker" (любитель котів) є супер-класом "cat_owner" (власник кота)?

Примітка. Підклас виведено завдяки твердженню підвластивості.

– Чому виведено, що "adult" (дорослий) є супер-класом "driver" (водія)?

Примітка. Для твердження про додаткову необхідну інформацію про клас використовується приклад аксіоми.

Нам не є необхідним знати, що водій є дорослим для того, щоб його розпізнати, але як тільки ми розпізнали водія, ми знаємо, що він повинний будти дорослим.

– Чому виведено, що "vegetarian" (вегетаріанець) є супер-класом "sheep" (вівці)?

Примітка. Зверніть увагу на повне визначення, що значить, що ми можемо розпізнати, коли речі є вегетаріанцями.

– Чому "vegetarian" (вегетаріанець) є супер-класом "giraffe" (жирафи)?

Примітка. Як і в попередньому прикладі з додатковим висновком, що забезпечується екзистенційним обмеженням у визначенні листа.

– Чому виведено, що "cat_owner" (власник кота) є супер-класом "old_lady" (старої пані)?

Примітка. Приклад взаємодії між квантором існування (твердження про існування кота) та універсальним квантором (обмеження можливих типів улюбленців).

- Чому виведено, що клас "mad_cow" є несумісним?

Примітка. Дика корова їсть частини тварин, що є несумісним з визначенням вегетаріанця.

3. Обґрунтуйте наступні висновки щодо екземплярів:

- Чому зроблено висновок, що "Daily_Mirror" є таблоїдом ("Tabloid")?

Примітка. Тут ми бачимо взаємодію між повним та частковим визначеннями плюс універсальний квантифікатор, що дозволяє зробити висновок про заповнювача ролі.

- Чому виведено, що "Spike" є улюбленцем ("pet")?

- Чому виведено, що "Pete" є персоною ("person")?

Примітка. Тут ми бачимо взаємодію між зворотнім відношенням та domain and range constraints на властивість. Домен властивості є діапазоном його зворотної властивості, а діапазон властивості – є доменом зворотної властивості.

- Чому виведено, що "Walt" є любителем тварин ("animal_lover")?

Примітка. Зверніть увагу, що в цьому випадку нам не є необхідним включати персону у визначення любителя тварин (тому що обмеження на домен дозволять нам побудувати цей висновок)

- Чому виведено, що "Tom" є котом ("cat")?

Примітка. Тут обмеження на домен дають нам додаткову інформацію, яка дозволяє нам зробити висновок про більш специфічний тип. Універсальний квантифікатор тоді дозволяє нам вивести інформацію про заповнювача ролі.

3.4.7 Подальше вивчення визначень класів

1. Завантажте онтологію "pets.owl" та викличте одну з СЛІВ в програмі Protégé. Завантажте онтологію до СЛІВ через опцію 'Classify' для того, щоб автоматично розрахувати класифікаційну ієрархію та перевірте логічну погодженість.

2. Додайте наступне визначення класу до онтології:

Class: PetOwner

SubClassOf: Person

that hasPet some Cat

Дайте відповіді на наступні запитання:

- Чи може PetOwner мати улюбленців, що не є котом (Cat)?

– Чи є володіння котом достатньою умовою, щоб розпізнати людину як PetOwner?

Примітка. Призначте властивість "has_pet" до класу "Person" та повторно класифікуйте онтологію. Чи відноситься тепер клас "Person" до підкласу "PetOwner"?

3. Повторно загрузіть початкову онтологію (чи відмініть зміни, зроблені на попередньому кроці) та додайте наступні визначення класів до онтології:

```
Class: PetOwner
    EquivalentTo: Person
        that hasPet some Cat
```

Чи є тепер володіння котом (Cat) достатнім, щоб розпізнати особу як PetOwner?

Примітка. Класифікуйте онтологію та вивчіть результати логічного висновку.

4. Повторно загрузіть початкову онтологію (чи відмініть зміни, зроблені на попередньому кроці) та додайте наступні визначення класів до онтології:

```
Class: PetOwner
    SubClassOf: Person
    EquivalentTo:
        hasPet some Cat
```

```
Individual: Sean
    Facts: hasPet Fluffy
Individual: Fluffy
    Types: Cat
```

– Що (чи є щось) ми вивели про Шона (Sean)?

– Що би ми могли вивести, якщо б ми замість мали наступні визначення?

```
Class: PetOwner
    EquivalentTo: Person
        that hasPet some Cat
```

```
Individual: Sean
    Facts: hasPet Fluffy
Individual: Fluffy
    Types: Cat
```

5. Повторно загрузіть початкову онтологію (чи відмініть зміни, зроблені на попередньому кроці) та додайте наступні визначення

класів до онтології:

DisjointClasses: Dog, Cat

Class: PetOwner1

EquivalentTo: Person

that (hasPet some Cat

and hasPet some Dog)

Class: PetOwner2

EquivalentTo: Person

that (hasPet some Cat

or hasPet some Dog)

Class: PetOwner3

EquivalentTo: Person

that hasPet some (Cat or Dog)

Class: PetOwner4

EquivalentTo: Person

that hasPet some (Cat and Dog)

– Які з цих класів (якщо такі є) еквівалентні?

– Які класи включають інші класи?

– Чи всі класи виконувані (satisfiable)?

6. Повторно загрузіть початкову онтологію (чи відмініть зміни, зроблені на попередньому кроці) та додайте наступні визначення класів до онтології:

ObjectProperty: likes

Characteristics

Reflexive

Class: Misanthrope

EquivalentTo:

Person

and (likes only (not (Person)))

Що ми знаємо про клас Мізантроп (Misanthrope) та чому?

3.5 Зміст звіту

3.5.1 Тема та мета роботи.

3.5.2 Короткі теоретичні відомості.

3.5.3 Результати роботи.

3.5.4 Висновки, що відображують результати виконання роботи та їх критичний аналіз.

ЛАБОРАТОРНА РОБОТА № 4

ІНЖЕНЕРІЯ ЗНАНЬ НА ОСНОВІ ЛОГІКИ

4.1 Мета роботи

Ознайомитися з логікою та логічними висновками, ознайомитися з мовою SWRL та отримати практичні навички роботи з ними.

4.2 Короткі теоретичні відомості

Semantic Web Rule Language (SWRL) – це мова для семантичної павутини, яка може бути використана для вираження правил, а також логіки. SWRL – це технологія, яка заснована на об'єднанні OWL з підмножиною RuleML.

SWRL має повну міць OWL DL, але за ціною розв'язуваності і практичних реалізацій.

Правила мають форми імплікації між передумовою (тіла) і наслідком (голови). Тобто: всякий раз, коли умови, зазначені у передумові, мають місце, то обставини, зазначені в наслідку також мають виконуватися.

RuleML (Rule Markup Language – мова розмітки для опису правил, підмножина Datalog) – це мова розмітки, розроблена, щоб виразити як прямі (знизу-вверх) та і зворотні (зверху-вниз) правила в XML для задач дедукції (висновку), представлення в іншому вигляді та подальшої трансформації на основі логічного висновку.

4.3 Завдання

4.3.1 Ознайомитися з логікою (правила та синтаксис) та з логічним висновком (через правила).

4.3.2 Зробити знання доступними для машин (у вигляді правил).

4.3.3 Ознайомитися з правилами кодування в XML.

4.3.4 Змодельовати предметну область за допомогою XML-подібної мови.

4.3.5 Ознайомитися зі зрозумілим людині синтаксисом SWRL та правилами SWRL.

4.3.6 Ознайомитися з синтаксисом SWRL, заснованим на XML.
Необхідні файли для роботи: (optional) Family_Relationships.txt

4.4 Порядок виконання лабораторної роботи

4.4.1 Базові правила та синтаксис

1. Уявіть базу даних фактів про деякі сімейні відносини. Припустимо, що ця база даних містить факти про наступні предикати:

- X це мати Y
- X це батько Y
- X є жінкою
- X є чоловіком

Використайте звичайний текстовий редактор для визначення цього знання.

2. Використайте відповідні правила для виводу подальших відносин:

- Визначте предикат *родитель*.

Примітка. Родитель – це чи батько, чи мати.

- Визначте предикати *брат* та *сестра*.

Примітка. Брат – це особа муж скоро роду, що розділяє з кимось предка.

Примітка. Використайте предикат “notSame” для позначення нерівності (inequality).

- Визначте предикати *дядько* та *тітка*.

Примітка. Дядько – це брат родителя.

- Визначте предикати *дідусь* та *бабуся*.

Примітка. Бабуся – це мати родителя.

- Визначте предикат *предок*.

Примітка. Предок – це родитель чи предок родителя.

4.4.2 Розмітка правил в XML

1. В програмі EditX створіть новий “Standard XML Document” для побудови XML документу, що описує сімейні відносини.

2. Закодуйте в XML чотири базових предикати: *мати*, *батько*, *жінка* та *чоловік*.

Примітка. Базовий предикат – це насправді факт. Факт – це проста атомарна формула, що оточується тегами <fact>.

Наприклад, факт $p(a)$ представляється наступним чином:

```
<fact>
<atom>
<predicate>p</predicate>
<term>
<const>a</const>
</term>
</atom>
</fact>
```

Примітка. Наприклад, факт $q(X)$ представляється наступним чином:

```
<fact>
<atom>
<predicate>q</predicate>
<term>
<var>X</var>
</term>
</atom>
</fact>
```

3. Закодуйте попередньо визначені правила в XML.

Примітка. Правило складається з голови та тіла. Голова – це атомарна формула. Тіло – це (можливо пуста) послідовність атомарних формул.

Використовуйте нові теги `<rule>`, `<head>` та `<body>`.

Наприклад, правило $p(X,a),q(Y,b) \rightarrow r(X,Y)$ представляється наступним чином:

```
<rule>
<head>
<atom>
<predicate>r</predicate>
<term>
<var>X</var>
</term>
<term>
<var>Y</var>
</term>
</atom>
</head>
```

```

<body>
<atom>
<predicate>p</predicate>
<term>
<var>X</var>
</term>
<term>
<const>a</const>
</term>
</atom>
<atom>
<predicate>q</predicate>
<term>
<var>Y</var>
</term>
<term>
<const>b</const>
</term>
</atom>
</body>
</rule>

```

4.4.3 Представлення повної предметної області за допомогою логіки

1. Оберіть власну предметну область та визначте її предикати та правила. Краще за все використати приклад інтернет-магазин Amazon.

Примітка. Використайте текстовий редактор для визначення предметної області.

Візьміть до уваги залучені сутності та відносини між ними.

2. Закодуйте знання визначені на попередньому кроці на будь-якій XML-подібній мові. Ви можете використати синтаксис з попереднього завдання, чи замість використати RuleML або навіть свою власну визначену XML-подібну мову.

Примітка. Для побудови XML документу використовуйте програму EditX.

3. За допомогою програми EditX побудуйте валідний документ XML Schema, що описує обрану предметну область.

4.4.4 Вступ до SWRL

1. Сформулюйте з допомогою синтаксису SWRL, зрозумілого для людини, базу даних сімейних відносин. Використайте текстовий редактор для визначення наступних знань:

- X це мати Y
- X це батько Y
- X є жінкою
- X є чоловіком

Примітка. Офіційна документація W3C стосовно SWRL доступна за посиланням: <http://www.w3.org/Submission/SWRL/>

2. Використайте відповідні правила SWRL, записані за допомогою синтаксису, зрозумілого для людини, для подальшого виводу відносин:

- Визначте предикат родитель.
- Визначте предикати брат та сестра.
- Визначте предикати дядько та тітка.
- Визначте предикати дідусь та бабуся.
- Визначте предикат предок.

3. Визначте наступні предикати:

- Молодий родитель (youngParent) – це родитель, чий вік ≤ 20 .
- Сумний родитель (sadParent) – це родитель без дітей.
- Щасливий родитель (happyParent) – це родитель у якого більш ніж 4 дитини.
- Старий родитель (oldParent) – це родитель, чий вік ≥ 60 та чий діти мають вік ≤ 10 .
- Втоmlений родитель (tiredParent) –це родитель чий діти всі жінки.

Примітка. Ви можете моделювати наведені предикати будь-яким зручним для вас способом. ЦЕ може включати визначення додаткових допоміжних предикатів.

4.4.5 Синтаксис SWRL, заснований на XML

1. Використайте редактор XML (наприклад, EditX) щоб сформулювати наступні знання за допомогою синтаксису SWRL, заснованому на XML:

- X це мати Y
- X це батько Y

- X є жінкою
- X є чоловіком

Примітка. Попереднє повинно бути сформульовано в якості SWRL class atoms, однак зверніть також увагу на datarange atoms.

Примітка. Короткий практичний посібник щодо XML синтаксису SWRL знаходиться за адресою: <http://www.w3.org/Submission/SWRL/#5>

Примітка. Інформація щодо XML Schema для XML синтаксису SWRL знаходиться за посиланням: <http://www.w3.org/Submission/SWRL/swrlx.xsd>

2. Сформулюйте наступні правила за допомогою синтаксису SWRL, заснованому на XML:

- Родитель – це батько чи мати.
- Два брати – це дві особи чоловічого роду, що розділяють родителя.
- Дві сестри - це дві особи жіночого роду, що розділяють родителя.
- Дядько – це брат родителя.
- Тітка – це сестра родителя.
- Дідусь – це батько родителя.
- Бабуся – це мати родителя.
- Предок – це чи родитель чи предок родителя.

3. Сформулюйте наступні правила за допомогою синтаксису SWRL, заснованому на XML:

- Молодий родитель (youngParent) – це родитель, чий вік ≤ 20 .
- Сумний родитель (sadParent) – це родитель без дітей.
- Щасливий родитель (happyParent) – це родитель у якого більш ніж 4 дитини.
- Старий родитель (oldParent) – це родитель, чий вік ≥ 60 та чії діти мають вік ≤ 10 .
- Втоmlений родитель (tiredParent) –це родитель чії діти всі жінки.

Примітка. Вам будуть необхідні SWRL builtins, інформацію щодо яких можна знайти за посиланням: <http://www.w3.org/Submission/SWRL/#8>

4.5 Зміст звіту

4.5.1 Тема та мета роботи.

4.5.2 Короткі теоретичні відомості.

4.5.3 Результати роботи.

4.5.4 Висновки, що відображують результати виконання роботи та їх критичний аналіз.

ЛАБОРАТОРНА РОБОТА № 5 ІНТЕГРАЦІЯ RDF ТА HTML

5.1 Мета роботи

Ознайомитися з технологією RDFa. Навчитися інтегрувати HTML та RDF.

5.2 Короткі теоретичні відомості

RDFa (Resource Description Framework in Attributes) – це рекомендація W3C, яка додає набір розширень рівня атрибутів до HTML, XHTML та різноманітних типів документів, заснованих на XML, для вбудовування метаданих у документи.

Відображення моделі даних RDF дозволяє використовувати його для вбудовування в XHTML документи RDF суб'єкт-предикат-об'єкт виразів. Він також дозволяє витяг моделей трійок RDF сумісними користувачькими агентами.

Ідея RDFa – надати набір атрибутів, які можуть бути використані переносу метаданих в XML мові.

Такими атрибутами є:

- `about=URI`, що визначає ресурс, який описують метадані;
- `rel` та `rev` – визначають відношення та зворотні відношення з іншими ресурсами;
- `src`, `href` та `resource` – визначають ресурс-партнер;
- `property` – визначають властивість змісту елемента чи ресурсу-партнеру;
- `content` – необов'язковий атрибут, що перевизначає вміст елемента при використанні атрибуту властивості;
- `datatype` – необов'язковий атрибут, що визначає тип даних тексту, визначеного для використання з атрибутом властивості;
- `typeof` – необов'язковий атрибут, що визначає RDF тип суб'єкту чи ресурсу-партнеру (ресурс, який описується метаданими).

RDFa є способом розмітки змісту для опису спеціального типу даних, наприклад відгуку про ресторани, заходи, людей або даних про товар. Ці типи даних називаються сутностями або елементами. Кожна сутність має ряд властивостей. Наприклад, сутність "людина" включає такі властивості, як ім'я, адреса, посада, місце роботи та адресу

електронної пошти.

У загальному випадку RDFa використовує прості атрибути в тегах XHTML (зазвичай `<span>` або `<div>`) для призначення коротких описових назв сутностей і їх властивостей. Нижче наведено приклад невеликого HTML-блоку, що показує основну контактну інформацію для Боба Сміта.

```
<div>
```

Мене звати Боб Сміт, але друзі кличуть мене Смітті. Ось моя домашня сторінка: `<a href="http://www.abc.com">www.abc.com</a>`.

Я живу в Альбукерці, штат Нью-Мексико, та працюю інженером в ACME Corp.

```
</div>
```

Далі наведено це й же HTML-код, але розмічений за допомогою RDFa.

```
<div xmlns:v="http://rdf.data-vocabulary.org/#" typeof="v:Person">
```

Мене звати `<span property="v:name">Боб Сміт</span>`, але друзі кличуть мене `<span property="v:nickname">Смітті</span>`.

Ось моя домашня сторінка: `<a href="http://www.abc.com" rel="v:url">www.abc.com</a>`.

Я живу в Альбукерці, штат Нью-Мексико, та працюю `<span property="v:title">інженером</span>` в `<span property="v:affiliation">ACME Corp</span>`.

```
</div>
```

Приклад починається з оголошення простору імен за допомогою `xmlns`. Цей атрибут вказує на простір імен, що містить словник (список сутностей та їх компонентів). Оголошення простору імен `xmlns:v="http://rdf.data-vocabulary.org/#"` можна використовувати кожен раз при розмітці на сторінці особистих даних, відгуків, товарів або місць.

Крім того, `typeof="v:Person"` в першому рядку вказує, що розмічене зміст являє сутність "людина". Перед назвою властивості `typeof` ставиться префікс `v:` (`typeof="v:Person"`).

Кожна властивість людини (наприклад, ім'я або псевдонім) позначається за допомогою `property`. Перед ім'ям властивості ставиться префікс `v:` (`<span property="v:nickname">`).

Щоб вказати URL, використовуйте `rel` замість `property`. Наприклад: `<a href="http://www.abc.com" rel="v:url">www.abc.com</a>`. `Rel` служить для вказівки зв'язку між сутностями,

в даному випадку між сутністю "людина" і сутністю "веб- сторінка" .

У прикладі показана контактна інформація Боба Сміта. У ній зазначено його адресу , але ніщо не вказує на зв'язок між Бобом і цією адресою.

Досить часто один тип даних включається в інший. У даному прикладі потрібно включити інформацію про адресу Боба (typeof="v:Address") в сутність typeof="v:Person". Тут rel використовується для вказівки зв'язку між Бобом (сутність v:Person) і його адресою (сутність v:Address) . Потім додаємо </ span > , щоб вказати фактичну сутність " адреса".

```
<div xmlns:v="http://rdf.data-vocabulary.org/#" typeof="v:Person">
```

Мене звати Боб Смит, але друзі кличуть мене Смитти.

Ось моя домашня сторінка:

```
<a href="http://www.abc.com" rel="v:url">www.abc.com</a>.
```

Я живу в

```
<span rel="v:address">
```

```
<span typeof="v:Address">
```

```
<span property="v:locality"> Альбукерці </span> ,
```

```
штат <span property="v:region">Нью-Мексико</span>
```

```
</span>
```

```
</span>
```

```
та працюю <span property="v:title">інженером</span>
```

```
в <span property="v:affiliation">ACME Corp</span>.
```

```
</div>
```

Таким чином, зміст сторінки може бути представлено для користувача одним способом , а для пошукових систем та веб-додатків - за допомогою прихованого тексту. Розмічати слід той текст, який видно користувачеві при відвідуванні веб-сторінки.

Проте у ряді випадків може виявитися корисним надати пошуковим системам більш детальну інформацію , навіть якщо її не потрібно показувати відвідувачам сторінки. Наприклад , інформація про широту та довготу місця проведення заходу допоможе пошуковій системі правильно показати його на карті , а дата проведення в форматі ISO - своєчасно відобразити його в результатах пошуку.

У даному випадку за допомогою атрибута контент можна вказати, що синтаксичний аналізатор розширеного опису веб -

сторінок повинен використовувати значення атрибуту для пошуку дати початку проведення заходу.

```
<span property="v:dtstart" content="2009-10-15T19:00-08:00">15  
октября 2009 г., 19:00</span>
```

5.3 Завдання

5.3.1 Ознайомитися з RDFa.

5.3.2 Використати індивідуальні RDF/S словники для доповнення веб-сторінок метаданими.

Необхідні файли для роботи: example-blog.html

5.4 Порядок виконання лабораторної роботи

5.4.1 RDFa: Integrating HTML & RDF

1. Відкрийте файл "example-blog.html" за допомогою текстового редактору чи редактору HTML файлів. Заповніть відповідні метадані, що відображують назву документу та його автора.

Примітка. Короткий посібник щодо RDFa можна знайти за посиланням: <http://www.w3.org/TR/xhtml1-rdfa-primer/>

Примітка. Найкраще, використовуйте dc:creator та dc:title (словник Dublin Core), замість просто creator та title, адже XHTML не має зарезервованих ключових слів для цих двох концепцій.

Примітка. Необхідно імпортувати простір імен DC: `xmlns:dc=http://purl.org/dc/elements/1.1/`

2. Додайте FOAF метадані до деталей контакту Alice.

Примітка. Словник DC не надає імен властивостей для опису контактної інформації, а словник Friend-of-a-Friend (FOAF) надає. Таким чином, важливо імпортувати словник FOAF та оголосити foaf:Person.

Примітка. Радимо використати @typeof – атрибут RDFa, який спеціально призначений для оголошення нового елемента даних певного типу.

3. Додайте власні метадані до файлу. Створіть відповідну RDF Schema, що описує декілька класів та властивостей, зв'язує класи.

Примітка. Процес створення нового словника RDFa включає наступні кроки:

(а) Вибір URL, за яким буде знаходитися словник, наприклад,

<http://example.com/photos/vocab#>.

(b) Поширення RDF документу за тим URL, що визначає класи та властивості що образують словник.

(c) Використання словника в XHTML+RDFa за допомогою звичайного механізму prefix оголошення, тобто `xmlns:photo="http://example.com/photos/vocab#"`, та `typeof="photo:Camera"`.

4. Використайте Sindice Inspector для перегляду структурованих даних, що містяться в HTML документі.

Примітка. Sindice доступний за посиланням <http://inspector.sindice.com/>.

5.5 Зміст звіту

5.5.1 Тема та мета роботи.

5.5.2 Короткі теоретичні відомості.

5.5.3 Результати роботи.

5.5.4 Висновки, що відображують результати виконання роботи та їх критичний аналіз.

ЛИТЕРАТУРА

1. XML. Базовый курс / Дэвид Хантер, Джефф Рафтер, Джо Фаусетт и др. – М.: Вильямс, 2009. – 1344 с.
2. Hebel John. Semantic Web Programming / John Hebel, Matthew Fisher, Ryan Blace, Andrew Perez-Lopez. – Wiley Publishing, 2009. – 648 p.
3. Segaran Toby. Programming the Semantic Web / Toby Segaran, Colin Evans, Jamie Taylor.. - O'Reilly Media, 2009. - 302 p.
4. Pollock Jeffrey T. Semantic Web For Dummies / Jeffrey T. Pollock. – Wiley, 2009. – 432 p.
5. Allemang Dean. Semantic Web for the working ontologist : effective modeling in RDFS and OWL / Dean Allemang, Jim Hendler. – 2nd ed. - Morgan Kaufmann Publishers, 2011. – 354 p.
6. P. Hitzler, M. Krötzsch, S. Rudolph, Foundations of Semantic Web Technologies. – Chapman&Hall/CRC, 2009. – 455 p.
7. Онтологии и тезаурусы: модели, инструменты, приложения : учебное пособие / Б.В. Добров, В.В. Иванов, Н.В. Лукашевич, В.Д. Соловьев. – М.: БИНОМ. Лаборатория знаний, 2009. – 173 с.
8. Муромцев Д.И. Онтологический инжиниринг знаний в системе Protégé: Методическое пособие. — СПб: СПбГУ ИТМО, 2007. — 62 с. - [<http://window.edu.ru/resource/429/54429>]
9. Язык запросов SPARQL для RDF. – Режим доступа: http://shcherbak.net/translations/ru_sparql_shcherbak_net.html
10. OWL, язык веб-онтологий. Краткий обзор и руководство. Рекомендация W3C от 10 февраля 2004 г. http://sherdim.ru/pts/semantic_web.
11. Разработка онтологий 101: руководство по созданию Вашей первой онтологии. – Режим доступа: http://ifets.ieee.org/russian/depositary/ontology101_rus.doc.