第四章 RDF 與 Topic Maps 之描述與解釋

本章將對 RDF 與 Topic Maps 在規範的發展歷程、概念、語法、以及相關研究與應用等面向加以探討和說明。第一節為 RDF 之描述與解釋，第二節則為 Topic Maps 之描述與解釋。

第一節 RDF 之描述與解釋

本節擬說明 RDF 規範發展的過程、RDF 之基本概念、RDF/XML 語法、以及 RDF 之相關研究與應用。

一、RDF 之發展與概要

（一）RDF 的發展過程

基本上可將 RDF 視為一組包含主詞、述語、與受詞的簡單敘述（Statement 或 Assertions）。將此種結構視覺化呈現的方式，是以節點表示主詞和受詞，以箭頭標示介於主詞和受詞間的述語。

W3C 在 1997 年首次公布的 RDF 工作草案揭示了 RDF 的目的，RDF 可作為處理詮釋資料的基礎。RDF 讓網路上的應用程式間能有互通性，來交換彼此的機讀資訊。(Lassila & Swick, 1997)RDF 的重點在促進網路資源的自動處理；換言之，RDF 是能促使結構化詮釋資料的編碼、交換、及複用的資料模型。RDF 提供了發佈可供人及機器閱讀之詞彙的方法，有助於不同社群所建立的詮釋資料之語意的複用與擴充。(Miller, 1998)下方表 4-1，羅列幾項與 RDF 發展有關的重要事件：

表 4-1　RDF 發展之重要事件

日期	事件
2004 年 2 月	公布 RDF 與 RDFS 1.0 等六份文件為 W3C 的建議（Recommendation）。
2001 年 2 月	啟動語意網活動（Semantic Web Activity）。
2000 年 3 月	公布 RDF Schema 1.0 版為 W3C 候選建議(Candidate Recommendation）。（1999 年至 2003 年 12 月亦持續不斷修訂 RDF 與 RDFS 候選建議）
1999 年 8 月	成立 RDF 興趣小組，此興趣小組在 2005 年時改為語意網興趣小組（Semantic Web Interest Group, SWIG）
1999 年 2 月	公布第一份 RDF 候選建議—RDF 模型與語法規範（RDF Model and Syntax Specification）。
1997 年 10 月	公布第一份 RDF 工作草案（Working Draft）。

資料來源：本研究

RDF 不僅要使資訊能被人閱讀，同樣要使資訊能為應用程式處理。RDF 提供一種表達資訊，並使資訊能在應用程式間交換，且不喪失語意的通用架構（Common Framework）。(Manola & Miller, 2004)以下進一步說明 RDF 的概念及語法。

（二）RDF 概要

根據 Manola and Miller(2004)的定義，RDF 係用於表徵網路資源的語言。RDF 不只可表徵網路資源的詮釋資料，例如網頁的標題、作者、與修改時間，或是網路文件的版權資訊等，事實上，RDF 可用來表徵任何可在網路上被辨識（identified）的事物，即使該事物或資源（例如抽象的概念）無法直接在網路上被獲。

RDF 的概念是透過網路識別碼（Web identifiers），即 URI 來標示資源，並以簡單的屬性（Properties）和屬性值（Values）來描述資源。如此一來，RDF 便能將有關資源的敘述，運用節點和弧（Arc）所組成的圖形，來表示資源、資源的屬性、和屬性值。此外，RDF 也提供

了 XML 的語法 RDF/XML 來標記及交換 RDF 圖形。(Manola & Miller, 2004)

RDF/XML 和 HTML 一樣，皆可被機器處理，並使用 URI 連結網路上的任何資源；但和傳統的超文件（Hypertext）不同的是，RDF URI 能用來指涉任何可被標示的事物，包括無法在網路上獲取的事物。這麼一來，RDF 將不只可用來描述網頁這類可在網路上直接獲取的事物，還可描述像汽車、商業活動、人、以及新聞事件等，無法在網路上被獲取的事物；此外，由於 RDF 的屬性可透過 URI 來標示，因此便能精確地指出被連結的事物間之關係。(Manola & Miller, 2004)

二、RDF 基本概念

（一）RDF 三元素—主詞、述語、受詞

若欲表示某位名叫 Inman 的人建立了某網頁，如果用英文的自然語言敘述的話，可以透過下方這種簡單的敘述方式表達：

```
http://www.example.org/inman.html has a creator whose value is Inman
```

上方的敘述顯示，為能描述事物，基本上需要標記的事物包括：該敘述所描述的事物（本例中的網頁）；該敘述所描述的事物之特定屬性（本例中的建立者）；以及該敘述所描述的作為該特定屬性之值的事物（本例中該網頁的建立者 Inman）。由此可知，RDF 的概念在於，被描述的事物會具有屬性，而這些屬性各有其值；RDF 對資源的描述即如上例，就是對該事物作出指定屬性及屬性值的敘述。不過 RDF 規範對需要標記的基本事物有特定術語，稱作「主詞」、「述語」、與「受詞」。同樣以上例說明如下：

- **主詞**：該敘述所描述的事物，本例中的網頁 `http://www.example.org/inman.html`；
- **述語**：該敘述所描述的事物之特定屬性，本例中的建立者 `creator`；
- **受詞**：該敘述所描述的作為該特定屬性之值的事物，本例中該網頁的建立者 `Inman`。

為使 RDF 作出的敘述是機器可處理的，一方面需要一套可用來標示敘述中的主詞、述語、與受詞，且能被機器處理的識別碼系統，同時該識別碼系統能清楚區別不同的識別碼（即使極為相似的識別碼）；另一方面，則需要以機器可處理的語言來表示敘述，並使敘述能在不同的機器間交換。而這二項需求在現有的網路環境中便可尋得解決方法，分別為 URI 及 XML，進一步說明如下：(Manola & Miller, 2004)

- RDF 利用 URI 可以標示敘述中的任何資源，包括可在網路上獲取的資源，例如電子文件、圖片、或一組其他資源；網路上無法獲取的實體資源，例如人、公司、或是書籍；實體上不存在的抽象概念，例如「作者」這個概念。換言之，RDF 用 URI 作為標示敘述中的主詞、述語、及受詞的機制，更精確來說，RDF 使用 URI References（URIs 或 URIref）來完成這件事。URIs 是一個在末端加了可選的片段識別碼（Fragment Identifier）的 URI，例如 `http://www.example.org/index.html#section2` 這個 URI，是由 `http://www.example.org/index.html` 這個 URI 及 `Section2` 這個片段識別碼所組成（由符號#分隔）。
- RDF 定義了一個特殊的 XML 標記語言，稱為 RDF/XML，利用 RDF/XML 可以標示 RDF 資訊，並在機器間交換這些資訊。

（二）RDF 模型

上一小節說明了 RDF 敘述的基本概念，也就是用 URIs 標示在 RDF 敘述中提及的事物，而 RDF/XML 則是表示 RDF 敘述的一種機器可處理的語言。本小節將進一步闡述 RDF 如何使用 URIs 來作出有

關資源的敘述。同樣以上一小節的例子為例，下方方框內的敘述：

```
http://www.example.org/inman.html has a creator whose
value is Inman
```

可由 RDF 敘述（有主詞、述語、及受詞）表示為：

- **主詞** http://www.example.org/inman.html
- **述語** http://purl.org/dc/elements/1.1/creator
- **受詞** http://www.example.org/studentid/0013

在上方的 RDF 敘述中，述語和受詞並未使用”creator”和”Inman”這二個字詞來表示，而是用 URIs 標示。進一步來說，RDF「三元素」分別為：(Klyne & Carroll, 2004)

- **主詞（資源）**：其為 RDF URIs 或空白節點（Blank Node）
- **述語（屬性）**：其為 RDF URIs
- **受詞（屬性值）**：其為 RDF URIs、文字、或空白節點

一組「三元素」又稱為 RDF 圖形，可表示為「節點–弧–節點」的連結，如下圖所示。RDF 圖形的節點為主詞和受詞，弧則永遠由主詞指向受詞。RDF 圖形的意義是其所包含的所有「三元素」之敘述的結合（邏輯的 AND 關係）。(Klyne & Carroll, 2004)

圖 4-1　RDF 圖形

因此上方的 RDF 敘述的例子，便可表示為圖 4-2：

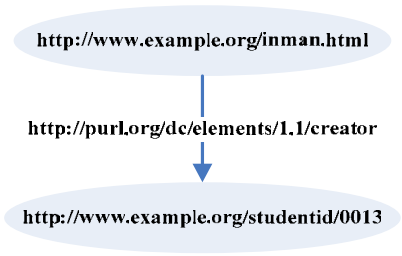

圖 4-2　簡單的 RDF 敘述

一組敘述是由相應的一組節點和弧表示，承接上方的例子，繼續對該網頁（http://www.example.org/inman.html）加上另二項敘述：

```
http://www.example.org/inman.html has a creation-date
whose value is May 20, 2005
http://www.example.org/inman.html has a language whose
value is Chinese
```

圖 4-3 即為以 RDF 圖形表示上方述敘述，本例以 Dubline Core 的 URI 標示 ”language” 屬性；以自訂詞彙的 URI 標示 ”creation-date” 屬性。

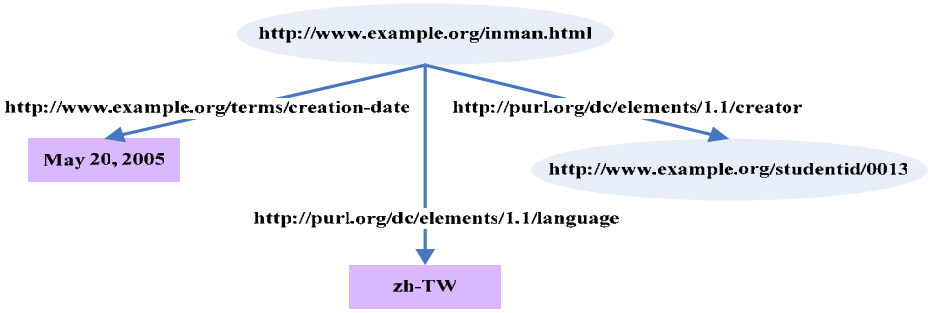

圖 4-3　同一資源的多個敘述

三、RDF 語法簡介

RDF 定義了一個特殊的 XML 標記語言，稱為 RDF/XML，利用 RDF/XML 可以標示及在機器間交換 RDF 圖形。以下進一步說明 RDF/XML 的基本原理，以及定義 RDF 詞彙的 RDF Schema。

（一）基本原理

1. 單一敘述

RDF/XML 語法的基本原理，可利用下方這段英文敘述為例來說明之：(Manola & Miller, 2004)

```
http://www.example.org/inman.html has a language whose value is Chinese
```

上方的敘述可以用 RDF 圖形表示（為 language 屬性指定 Dubline Core 的 URI），如圖 4-4 所示：

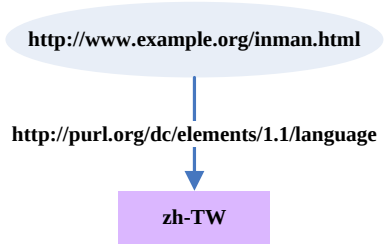

圖 4-4　描述一個網頁所使用的語文

下方的編碼實例 4-1 顯示了上方的圖 4-4 所相應的 RDF/XML 語法（加入行數僅是為了解釋之用）：

```
1. <?xml version="1.0"?>
2. <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
```

```
3.      xmlns:dc="http://purl.org/dc/elements/1.1/">

4.    <rdf:Description rdf:about="http://www.example.org/inman.html">
5.        <dc:language>zh-TW</dc:language>
6.    </rdf:Description>

7. </rdf:RDF>
```

編碼實例 4-1　　描述一個網頁所使用的語文的 RDF/XML

關於上方的編碼實例 4-1，以下逐行說明之：

表 4-2　編碼實例 4-1 之語法說明

行數	說明
1	<?xml version="1.0"?>是 XML 宣告，標示以下內容為 XML，以及 XML 的版本。
2	第二行以 rdf:RDF 元素開始，表示以下的 XML 內容（由此處開始至第七行的</rdf:RDF>為止）是在表示 RDF。本行緊跟著 rdf:RDF 之後的是 XML Namespace 宣告，即 rdf:RDF 為起始標籤的 xmlns 屬性。該宣告指出，在目前的文件中出現所有前置詞為 rdf:的標籤，都是屬於由 http://www.w3.org/1999/02/22-rdf-syntax-ns#這個 URIs 所標示的 Namespace。以 http://www.w3.org/1999/02/22-rdf-syntax-ns#這個開頭的 URIs，用來標示來自 RDF 詞彙（Vocabulary）中的術語。
3	另一個前置詞為 dc:的標籤之 XML Namespace 宣告。表示另一個 rdf:RDF 元素的 xmlns 屬性。它指出前置詞為 dc:的標籤屬於 http://purl.org/dc/elements/1.1/這個 URIs 所標示的 Namespace。以 http://purl.org/dc/elements/1.1/這個開頭的 URIs，用來標示本文件採用 Dubline Core 所定義的詞彙中的術語。第三行最後的”>”符號標示 rdf:RDF 起始標籤的結束。
1-3	是表明此為 RDF/XML 內容的必備部分，同時也標示在

	RDF/XML 內容中使用的 Namespace。
4-6	第四至六行是以 RDF/XML 標記上方圖 4-4 所示的敘述。當談論 RDF 敘述時，可將之稱為「描述」(Description)，並且是「有關」(About) 該敘述的主詞的描述，本例是有關 `http://www.example.org/inman.html` 的描述，此正為 RDF/XML 表示敘述的方式。
4	`rdf:Description` 的起始標籤表明某個資源的描述的開始，接著以 `rdf:about` 屬性指出主詞所代表的資源的 URIs，這麼一來便指出這段敘述是和什麼資源有關。
5	用 QName `dc:language` 作為標籤，提供屬性元素（Property Element）來表示敘述中的述語及受詞。採用 QName `dc:language`，那麼就可將附加的本地名稱 `language` 套至前置詞 `dc:` 所代表的 URIs `http://purl.org/dc/elements/1.1/` 之後，也就形成敘述中的述語之 URIs `http://purl.org/dc/elements/1.1/language`。屬性元素的內容即為敘述的受詞，也就是 `zh-TW`。在 `rdf:Description` 中的屬性元素會巢狀 (Nested) 出現，也就意味著該屬性應用於 `rdf:Description` 元素的 `rdf:about` 屬性所指出的資源。
6	表示這個 `rdf:Description` 到此結束。
7	表示由第二行開始的 `rdf:RDF` 元素到此結束。若能隨情境辨識 XML 內容是否為 RDF/XML 時，並不一定要用 `rdf:RDF` 元素來特別標示為 RDF/XML 內容，不過不管在任何情況，最好還是用 `rdf:RDF` 元素標示內容為 RDF/XML 較好。

註：本表的說明內容主要參考自 Manola and Miller(2004)。

2. 同一資源的多個敘述

RDF/XML 語法提供了若干種簡略表達形式以便標記，例如若欲表達之前圖 4-3 的 RDF 圖形，也就是對同一資源有多個敘述的話，一種方式是在標示該主詞的 `rdf:Description` 元素下，嵌入多個屬性元素來表達屬性，如編碼實例 4-2 所示；另一種方式則是將各個敘述分別標記，如編碼實例 4-3 所示。

```
1.  <?xml version="1.0"?>
2.  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3.           xmlns:dc="http://purl.org/dc/elements/1.1/"
4.           xmlns:exterms="http://www.example.org/terms/">

5.    <rdf:Description rdf:about="http://www.example.org/inman.html">
6.      <exterms:creation-date>May 20, 2005</exterms:creation-date>
7.      <dc:language>zh-TW</dc:language>
8.      <dc:creator rdf:resource="http://www.example.org/studentid/0013"/>
9.    </rdf:Description>

10. </rdf:RDF>
```

編碼實例 4-2　簡略表達多個屬性

編碼實例 4-2 和編碼實例 4-1 一樣，只是在第四行宣告了另一個 Namespace，另也多了 dc:creator 這個屬性元素（在第八行）。第八行中的 dc:creator 元素，是屬性值為另一資源（而不是純文字）的屬性，其屬性值為 URIs （http://www.example.org/staffid/85740）所引用的資源，而非字串 http://www.example.org/staffid/85740，所以 dc:creator 元素以「空元素」的標籤形式標記（即沒有結束標籤），同時用 rdf:resource 屬性（Attribute）來表達屬性元素的值是由該 URIs 所標示的資源。由於該 URIs 要作為屬性的值（Attribute Value），因此 RDF/XML 要求這種 URIs 的形式必須為「絕對 URIs」或「相對 URIs」，不能簡略為 QName 的形式。(Manola & Miller, 2004) 編碼實例 4-3 也是表示同一個 RDF 圖形（圖 4-3）的 RDF/XML，只是當中的各個敘述被分別標記：

```
<?xml version="1.0"?>
 <rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
           xmlns:dc="http://purl.org/dc/elements/1.1/"
           xmlns:exterms="http://www.example.org/terms/">
```

```
  <rdf:Description rdf:about="http://www.example.org/inman.html">
     <exterms:creation-date>May 20, 2005</exterms:creation-date>
  </rdf:Description>

  <rdf:Description rdf:about="http://www.example.org/inman.html">
     <dc:language>zh-TW</dc:language>
  </rdf:Description>

  <rdf:Description rdf:about="http://www.example.org/inman.html">
     <dc:creator rdf:resource="http://www.example.org/studentid/0013"/>
  </rdf:Description>

</rdf:RDF>
```

編碼實例 4-3　　同一資源的各個屬性分別標記

3.　含有空白節點的敘述

「空白節點」(Blank Node) 是表示某個沒有 URIs，但可用其他資訊來描述的事物。(Manola & Miller, 2004)圖 4-5 是顯示表達以下資訊的 RDF 圖形：http://www.glis.ntnu.edu.tw/stuid0013-thesis 這個文件，有名為 RDF and Topic Maps 的題名，而編者的姓名為 C-L Chung，編者的網頁是 http://www.glis.ntnu.edu.tw/stuid0013/homepage。

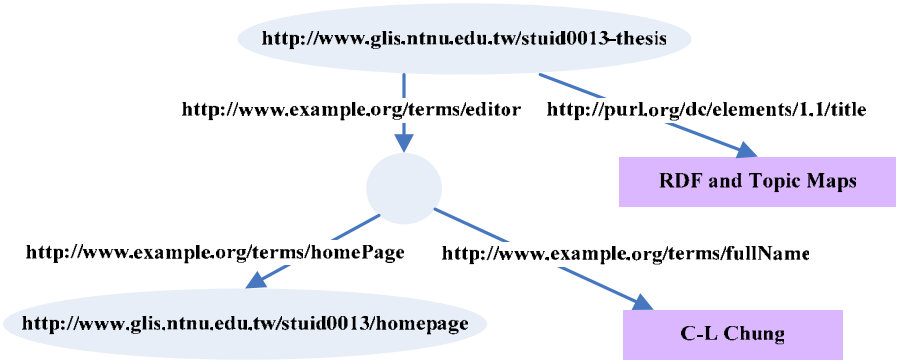

圖 4-5　一個包含空白節點的 RDF 圖形

RDF/XML 提供了多種方式來表示含有「空白節點」的 RDF 圖形，下方所使用的方式是最直接的一種，即為每個「空白節點」指定一個「空白節點識別碼」(Blank Node Identifier)，相較於其他表示方式，使用「空白節點識別碼」的好處在於，可以在同一 RDF/XML 文件中多次引用同一個「空白節點」。「空白節點識別碼」用於標示某特定 RDF/XML 文件內部的某一「空白節點」，與 URIs 不同的地方是，「空白節點識別碼」在所屬文件外部是無法識別的。(Manola & Miller, 2004)

在 RDF/XML 中，所有可以用 URIs 標示資源的地方都可以用 rdf:nodeID 屬性（Attributte）來引用「空白節點」(「空白節點識別碼」為該屬性〔Attributte〕的值)。若敘述中的主詞為一「空白節點」，用 RDF/XML 表示的話，可以用含有 rdf:nodeID 屬性（Attribute）的 rdf:Description 元素表示（而不是用 rdf:about 屬性)；若敘述中的受詞為一「空白節點」的話，可以用含有 rdf:nodeID 屬性（Attribute）的屬性元素表示（而不是用 rdf:resource 屬性)。(Manola & Miller, 2004)編碼實例 4-4 展示了 RDF/XML 如何用 rdf:nodeID 來描述上方的圖 4-5（本例將「空白節點識別碼」指定為 abc)：

```
1.<?xml version="1.0"?>
2.<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3.         xmlns:dc="http://purl.org/dc/elements/1.1/"
4.         xmlns:exterms="http://www.example.org/terms/">

5.  <rdf:Description
rdf:about="http://www.glis.ntnu.edu.tw/stuid0013-thesis">
6.    <dc:title>RDF and Topic Maps</dc:title>
7.    <exterms:editor rdf:nodeID="abc"/>
8.  </rdf:Description>

9.  <rdf:Description rdf:nodeID="abc">
10.   <exterms:fullName>C-L Chung</exterms:fullName>
```

```
11.    <exterms:homePage
rdf:resource="http://www.glis.ntnu.edu.tw/stuid0013/homepage"/>
12.  </rdf:Description>
13.</rdf:RDF>
```

編碼實例 4-4　描述空白節點的 RDF/XML

在編碼實例 4-4 中，第九行使用「空白節點識別碼」abc 來標示作為多個敘述之主詞的「空白節點」，而在第七行利用該「空白節點識別碼」來表示，該「空白節點」為資源之 exterms:editor 屬性的值。

4. 使用類型文字的 RDF/XML

「類形文字」(Typed Literals）在 RDF/XML 中的表示方式是：替包含該文字的屬性元素加入 rdf:datatype 屬性（Attribute)，並透過該屬性指定資料類型的 URIs。若將圖 4-3 中的敘述之 exterms:creation-date 屬性的值，改為「類型文字」而非純文字，那麼它的 RDF/XML 可改寫成：

```
1. <?xml version="1.0"?>
2. <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3.          xmlns:exterms="http://www.example.org/terms/">

4.   <rdf:Description
rdf:about="http://www.example.org/inman.html">
5.     <exterms:creation-date rdf:datatype=
         "http://www.w3.org/2001/XMLSchema#date">2005-05-20
       </exterms:creation-date>
6.   </rdf:Description>
<!--..............為求簡明剩餘部分予以省略..............-->
7. </rdf:RDF>
```

編碼實例 4-5　使用類型文字的 RDF/XML

在編碼實例 4-5 中，第五行 exterms:creation-date 這個屬性元素的值是一個「類型文字」，這是透過為 exterms:creation-date 元素的起始標籤增加一個 rdf:datatype 屬性（Attribute），並由此屬性指定資料類型來達成。rdf:datatype 屬性（Attribute）的值應為某資料類型的 URIs，在本例中它是 XML Schema 中的 date 資料類型的 URIs。作為屬性值（Attribute Value），URIs 的形式必須為「絕對 URIs」或「相對 URIs」，不能簡略為 QName 形式。(Manola & Miller, 2004) 元素內容必須是符合此資料類型的文字，本例是文字 2005-05-20，也就是用 XML Schema 的 date 資料類型，表示 2005 年 5 月 20 日的文字。

編碼實例 4-5 說明了需要為每個元素值為「類型文字」的元素，增加 rdf:datatype 屬性（Attribute），並用一個標示資料類型的 URIs 作為屬性值（Attribute Value）。不過由於 RDF/XML 規定作為屬性值（Attribute Value）的 URIs，必須為「絕對 URIs」或「相對 URIs」，不能簡略為 QName 的形式，因此 RDF/XML 允許使用 XML「實體」來提高可讀性，也就是為 URIs 提供另一種簡寫形式。XML 實體宣告是將實體名稱與字串相關聯。若實體名稱在 XML 文件中被參照（Referenced），XML 處理器將以相應的字串來替換該實體名稱。舉例來說，ENTITY 宣告（寫在 RDF/XML 文件的開頭的 DOCTYPE 宣告中）：(Manola & Miller, 2004)

```
<!DOCTYPE rdf:RDF [<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">]>
```

上方的宣告將實體 xsd 定義為一個代表 XML Schema 資料類型的 Namespace URIs 的字串。這麼一來，完整的 Namespace URIs 可在 XML 文件中被簡寫成實體參照（Entity Reference）&xsd; 。(Manola & Miller, 2004)若將編碼實例 4-5 以這種方式簡寫的話，即為編碼實例 4-6。

```
1. <?xml version="1.0"?>
2. <!DOCTYPE rdf:RDF [<!ENTITY xsd "http://www.w3.org/2001/XMLSchema#">]>
```

```
3. <rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
4.         xmlns:exterms="http://www.example.org/terms/">

5.   <rdf:Description
rdf:about="http://www.example.org/inman.html">
6.     <exterms:creation-date rdf:datatype="&xsd;date">2005-05-20
</exterms:creation-date>
7.   </rdf:Description>

8. </rdf:RDF>
```

編碼實例 4-6　使用類型文字及 XML 實體的 RDF/XML

第二行的 DOCTYPE 宣告，定義了實體 xsd，該實體被用在第六行。在 RDF/XML 中是否用 XML 實體作為簡寫方式不是強制性的，因此在 RDF/XML 中是否使用 XML DOCTYPE 宣告也是可選擇的，因為 RDF/XML 只需是 ”well-formed” 的 XML 即可。儘管還有其他可用來標記 RDF/XML 的簡寫方式，但以上所介紹的方法，提供了一種既簡單又一般的，表示 RDF 圖形的 RDF/XML 語法。簡言之，用 RDF/XML 表示 RDF 圖形可依循二項要點：(Manola & Miller, 2004)

- 為所有「空白節點」指定一個「空白節點識別碼」。
- 依次列出每個節點，將它作為非巢狀的 rdf:Description 元素的主詞。若該節點有 URIs，則 rdf:Description 元素用 rdf:about 屬性（Attribute）；若該節點為空白節點，則 rdf:Description 元素用 rdf:nodeID 屬性（Attribute）。對每個以該節點作為主詞的「三元素」，建立一個適當的屬性元素（Property Element），該屬性元素或元素內容可能是文字，也可能是一個指定該「三元素」的受詞的 rdf:resource 屬性（Attribute）（若受詞有 URIs）；或者是一個指定該「三元素」的受詞的 rdf:nodeID 屬性（Attribute）（若受詞為「空白節點」）。

5. 描述一組事物

(1) RDF 容器（Containers）

為能描述一組事物，例如標明某書是由那些作者撰寫，或列舉修某門課程的學生名單等，RDF 提供了三種預先定義的容器類型（Types）及其相關屬性。一個「容器」是一個包含一些事物的資源，被包含的事物稱為成員（Members）。「容器」的成員可能是資源（包括「空白節點」）或文字。以下，進一步說明 RDF 預先定義的三個容器詞彙（Container Vocabulary）：(Manola & Miller, 2004)

表 4-3　RDF 容器詞彙

容器詞彙	說明
rdf:Bag	「包裹容器」是類型為 rdf:Bag 的資源。表示一組可能包含重複成員的資源或文字，但成員間的順序不重要。
rdf:Seq	「序列容器」是類型為 rdf:Seq 的資源。表示一組可能包含重複成員的資源或文字，成員間的順序是重要的。例如用「序列容器」敘述一組須按字母順排列的事物。
rdf:Alt	「替換容器」是類型為 rdf:Alt 的資源。表示一組可以替換的資源或文字（通常是屬性的單一值）。若屬性的類型為「替換容器」，那麼應用程式可以選擇該組資源中的任一合適成員，作為該屬性之值。例如用「替換容器」描述某著作的不同語言譯作。

為表示某資源是某種類型的容器，則該資源必須有 rdf:type 屬性，且該屬性的值為 rdf:Bag、rdf:Seq、或 rdf:Alt。容器資源（「空白節點」或有標示 URIs 的資源）表示的是一個整體中的一組事物。容器的成員和容器資源本身，可藉由對每個成員，定義其「容器成員之屬性」（Container Membership Property）來描述，亦即屬性的主詞為容器資源本身，受詞則是容器的成員。「容器成員之屬性」以 rdf:_n 的形式命名，當中的 n 是大於 0 的十進位整數，例如

rdf:_1、rdf:_2、或 rdf:_3 等。若欲描述容器，除了「容器成員之屬性」和 rdf:type 屬性之外，還可利用其他屬性。(Manola & Miller, 2004)

容器通常是用來表示，有個屬性其值為一組事物。例如，要表示「修 IR 課程的學生有 Anadem、Kate、Poyu、與 Inman」，那麼我們可以為該課程加入 exterms:students 屬性，該屬性值為 rdf:Bag 包裹容器(表示一組學生)，並用「容器成員之屬性」將每個學生標示為該容器的成員，其 RDF 圖形如下：

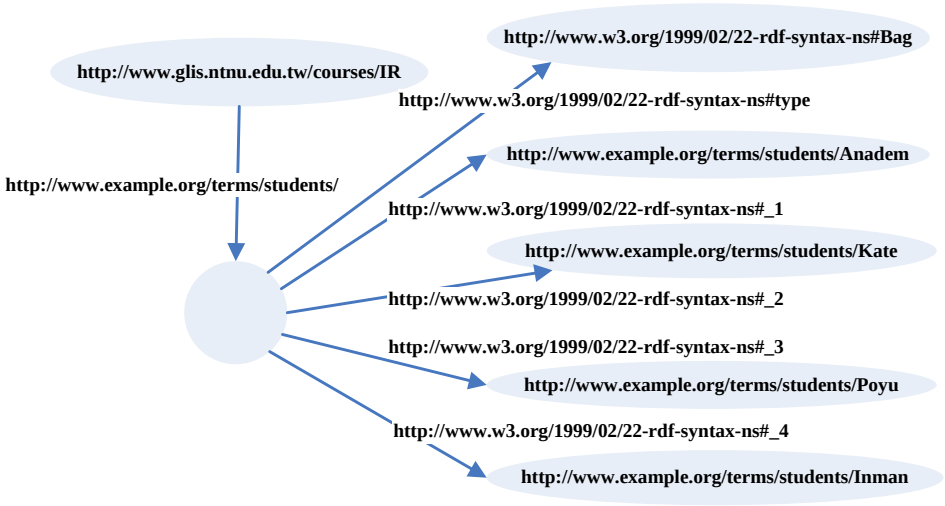

圖 4-6　簡單的「包裹容器」描述

由於本例 exterms:students 屬性的值為「包裹容器」，因此學生的 URIs 的順序並不重要；也就是說，應用程式在建立及處理「包裹容器」時，不用在乎學生的 URIs 結尾所標示的整數之順序。(Manola & Miller, 2004)在描述那樣的容器時，RDF/XML 提供一些可以簡化標記的特殊語法及縮寫，上方圖 4-6 的編碼實例如下：

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:exterms="http://www.example.org/terms/">
```

```
  <rdf:Description
rdf:about="http://www.glis.ntnu.edu.tw/courses/IR">
    <exterms:students>
      <rdf:Bag>
      <rdf:li rdf:resource="http://www.example.org/terms/students/Anadem"/>
       <rdf:li rdf:resource="http://www.example.org/terms/students/Kate"/>
       <rdf:li rdf:resource="http://www.example.org/terms/students/Poyu"/>
       <rdf:li rdf:resource="http://www.example.org/terms/students/Inman"/>
      </rdf:Bag>
     </exterms:students>
  </rdf:Description>
</rdf:RDF>
```

編碼實例 4-7　描述修課學生的「包裹容器」之 RDF/XML

編碼實例 4-7 顯示 RDF/XML 可以利用 rdf:li 元素，來避免詳盡列出每個「容器成員之屬性」的數字。標號的屬性，例如 rdf:_1 和 rdf:_2 等，在形成相對應的圖形時會由 rdf:li 元素產生。在<exterms:students>屬性元素內成巢狀出現的<rdf:Bag>，是另一種縮寫方式；也就是當描述一個類型的實例時（本例是 rdf:Bag 的實例），可以用單一元素取代 rdf:Description 元素和 rdf:type 元素。因為沒有指定 URIs，因此這個「包裹容器」是一個空白節點。<rdf:Bag>在<exterms:students>屬性元素內成巢狀出現的縮寫方式，即表示該屬性的值就是「空白節點」。(Manola & Miller, 2004)

「序列容器」（rdf:Seq）的 RDF 圖形結構及相應的 RDF/XML 和 rdf:Bag 相似，唯一不同之處在其類型為 rdf:Seq。應用程式在建立及處理「序列容器」的圖形時，要能適切地剖析屬性名稱中整數值的順序。(Manola & Miller, 2004)而下方的例子是說明如何用「替換容器」（rdf:Alt）描述「stuid0013-thesis 的文件可以在 ftp.glis.ntnu.edu.tw、ftp1.glis.ntnu.edu.tw、或 ftp2.glis.ntnu.edu.tw 中找到。」這句敘述的 RDF 圖形如下方圖 4-7 所示。

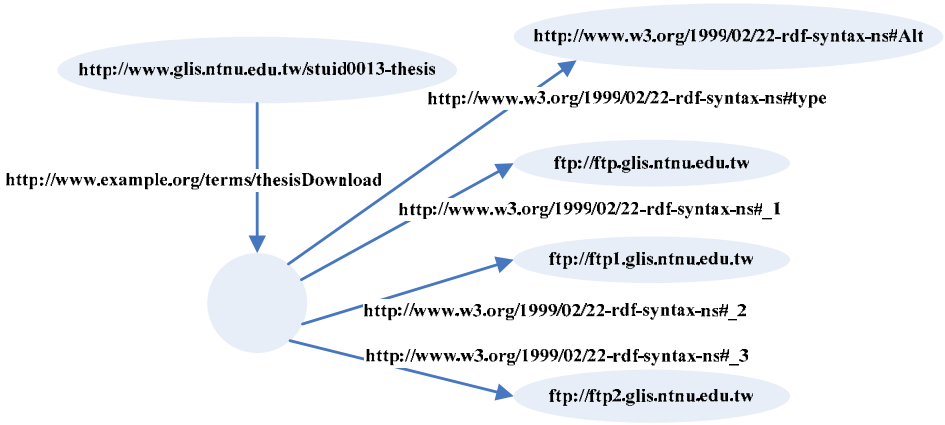

圖 4-7　簡單的「替換容器」描述

上方圖 4-7 可以 RDF/XML 表示如下：

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
       xmlns:exterms="http://www.example.org/terms/">

  <rdf:Description
rdf:about="http://www.glis.ntnu.edu.tw/stuid0013-thesis">
     <exterms:thesisDownload>
       <rdf:Alt>
         <rdf:li rdf:resource="ftp://ftp.glis.ntnu.edu.tw"/>
         <rdf:li rdf:resource="ftp://ftp1.glis.ntnu.edu.tw"/>
         <rdf:li rdf:resource="ftp://ftp2.glis.ntnu.edu.tw"/>
       </rdf:Alt>
     </exterms:thesisDownload>
  </rdf:Description>
</rdf:RDF>
```

編碼實例 4-8　「替換容器」的 RDF/XML

「替換容器」至少要有一個標示 rdf:_1 屬性的成員，其被視為預設或偏好值，除了該成員外，其餘成員的順序並不重要。上方圖 4-7

表示 exterms:thesisDownload 這個屬性的值，是一個類型為「替換容器」的資源。關於該 RDF 圖形的其他意義須由應用程式來剖析。

使用者可以不用 RDF 容器詞彙，而用自己的方式來描述一組資源。這些 RDF 容器只是提供一種通用方式，若廣為使用的話，則可使描述一組資源的資料提高互通性。當用 RDF 容器時，這些敘述是在描述已存在的容器（一組事物），而不是在建立（Constructing）容器。此外，「容器成員之屬性」只是描述一個容器資源有某些事物為其成員，而非說這些事物是容器僅有的成員。(Manola & Miller, 2004)

(2) RDF 集合（Collection）

RDF 容器無法限定所描述的成員，即為該容器的所有成員。若需描述一組特定成員的資源，可利用 RDF 集合來達成。RDF 集合是將一組事物以列表結構表示，這個列表結構由預先定義的集合詞彙表示，包括 rdf:List 類型，rdf:first 與 rdf:rest 屬性，及 rdf:nil 資源。(Manola & Miller, 2004)下方圖 4-8 是描述「修 KM 課程的學生有 Nicole、Tom、和 Renee」的 RDF 圖形。

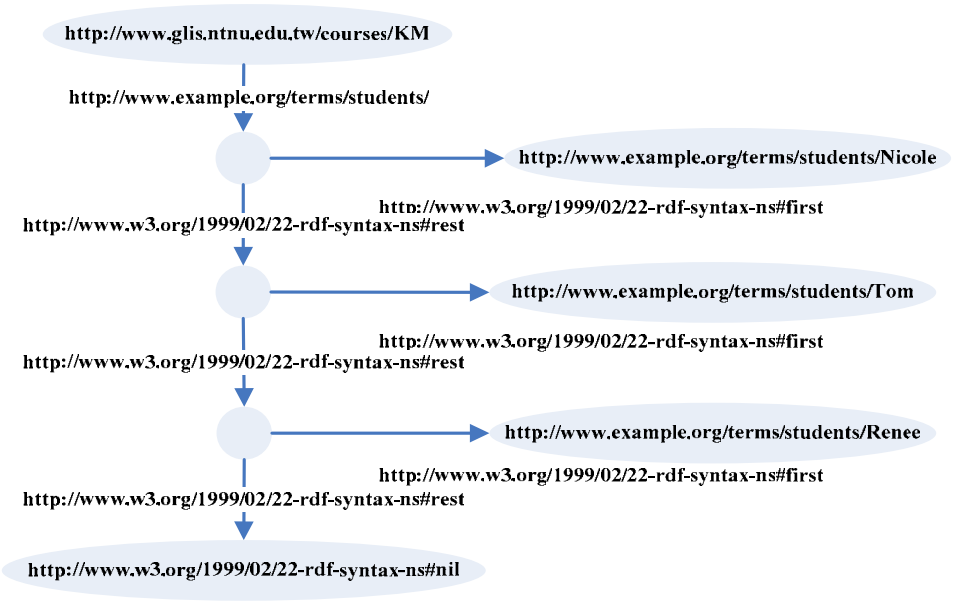

圖 4-8　RDF 集合（列表結構）

上方圖 4-8 顯示，集合中的每個成員，例如 Nicole 是 rdf:first 屬性的受詞，該屬性的主詞表示的是一個列表的資源（本例該主詞為空白節點）。這個列表的資源透過 rdf:rest 屬性連向其餘資源。列表的結束用 rdf:rest 屬性指向受詞為 rdf:nil 的資源（rdf:nil 資源表示的是空列表，它的類型是 rdf:List）。(Manola & Miller, 2004)

RDF/XML 提供了一種特殊的標記法，使描述集合較為簡易。在 RDF/XML 中，一個集合可藉由有 rdf:parseType="Collection" 屬性（Attribute）的屬性元素（Property Element），且其包含一組巢狀元素來表示集合中的成員的方式描述。下方的編碼實例 4-9 即為可以產生上方圖 4-8 的 RDF/XML：(Manola & Miller, 2004)

```
<?xml version="1.0"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:exterms="http://www.example.org/terms/">

 <rdf:Description rdf:about="http://www.glis.ntnu.edu.tw/courses/KM">
   <exterms:students rdf:parseType="Collection">
      <rdf:Description
rdf:about="http://www.example.org/terms/students/Nicole"/>
      <rdf:Description
rdf:about="http://www.example.org/terms/students/Tom"/>
      <rdf:Description
rdf:about="http://www.example.org/terms/students/Renee"/>
   </exterms:students>
  </rdf:Description>
</rdf:RDF>
```

編碼實例 4-9　描述學生集合的 RDF/XML

綜上所述，RDF 的語法詞彙（Syntax Terms）包括：rdf:RDF, rdf:Description, rdf:ID, rdf:about, rdf:parseType,

rdf:resource, rdf:li, rdf:nodeID 及 rdf:datatype 等九個。

（二）定義 RDF 詞彙：RDF Schema

RDF 使用已命名的屬性（Named Properties）及屬性值，來表達有關資源的簡單敘述，不過若要能根據自己的需求定義詞彙，特別是描述資源的特定類，或描述資源所使用的特定屬性的話，並無法以 RDF 來定義類和屬性。類和屬性需透過「RDF 詞彙描述語言」（RDF Vocabulary Description Language 1.0）：RDF Schema 來定義。(Manola & Miller, 2004)運用 RDF Schema 詞彙，可建立特定主題領域的綱要。而 RDF Schema 的建模原詞（Modelling Primitives），可定義的事項基本上包括：

- **類的階層**：rdfs:Class, rdfs:subClassOf
- **屬性的階層**：rdf:Property, rdfs:subPropertyOf
- **限定屬性與類的可能結合**：rdfs:domain, rdfs:range
- **提供資源的名稱或描述**：rdfs:label, rdfs:comment

以下方圖 4-9 為例，虛線下方是 RDF 主詞、述語、與受詞的資料模型，而虛線上方則是 Schema 的部分，它可以定義主詞 "…/hohome.html" 是 "VicePresident" 這個類的實例；而受詞 "…/enersearch.html" 是 "Company" 這個類的實例。"VicePresident" 是 "Employee" 的子類。"worksFor" 這個屬性的主題領域（Domain）必須為 "Employee"，而其值域（Range）則必須是 "Company "。

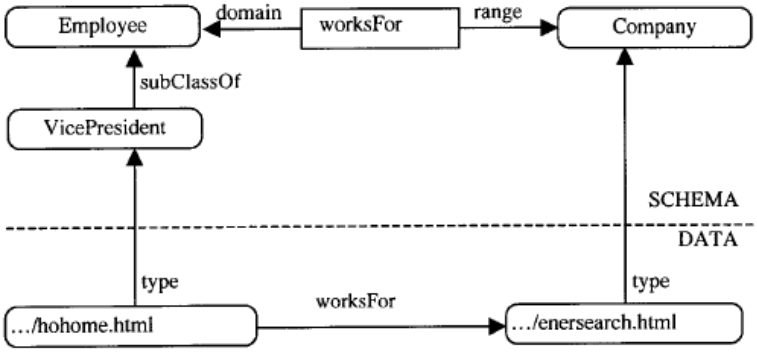

圖 4-9　RDF Schema 實例

資料來源：Broekstra, J., Kampman, A., & van Harmelen, F.(2003). Sesame: a generic architecture for storing and querying RDF and RDF Schema. In J. Davies, D. Fensel & F. van Harmelen(Eds.), Towards the Semantic Web: Ontology-driven knowledge management(pp.71-89). New York: John Wiley & Sons.

換言之，RDF Schema 並未提供針對特定應用需求所定義的類及屬性，而是提供描述類與屬性，及指出那些類及屬性應共同使用的能力（例如描述 ex3:Person 這個類時，應用 ex3:jobTitle 屬性）。RDF Schema 為 RDF 提供一個型別系統（Type System），例如 RDF Schema 允許資源被定義為一或多個類的實例（Instances）；此外，RDF Schema 也可以將類組織成階層結構。RDF 的類和屬性描述，可以提供有關其所描述之資源一些額外資訊。(Manola & Miller, 2004)

RDF Schema 所具備的能力本身也是以 RDF 詞彙的形式提供；也就是說，RDF 詞彙是一組具有特殊涵義，且預先定義的 RDF 資源。RDF Schema 詞彙所代表的資源，是具有前置詞為 http://www.w3.org/2000/01/rdf-schema#的 URIs（QName 前置詞為 rdfs:）。以 RDF Schema 語言所下的詞彙描述也是合法的（Legal）RDF 圖形。(Manola & Miller, 2004)

1. 描述類（Class）

進行描述的過程，最基本的動作便是將所描述事物分類。RDF

Schema 將事物的種類稱為「類」。RDF Schema 所稱的「類」，和人們一般所稱的「類型」（Type）或「分類」（Category）在概念上是相同的。「類」可用 RDF Schema 中的資源（rdfs:Class 和 rdfs:Resource）及屬性（rdf:type 和 rdfs:subClassOf）來表示（在撰寫 RDF/XML 時，慣例是將類名的首字母大寫，而屬性名和實例名的首字母則小寫）。屬於某類的資源，稱為「實例」。在 RDF Schema 中，「類」是具有 rdf:type 屬性，且屬性值為 rdf:Class 的任何資源。rdf:type 屬性是用來指出，該資源為某類的實例。一個資源可以是一或多個類的實例。若欲表明某類為另一類的子類，則可用屬性 rdfs:subClassOf 來表示。rdfs:subClassOf 屬性具有遞移性。(Manola & Miller, 2004)

例如描述國家圖書館（NationalLibrary）、學術圖書館（AcademicLibrary）與公共圖書館（PublicLibrary）這三種類型的圖書館（Library），皆為圖書館的一種，且鄉鎮圖書館（CountyLibrary）為公共圖書館的一種。以上敘述的 RDF 圖形如下方圖 4-10 所示（為求簡明，將關聯每個類和 rdfs:Class 的屬性 rdf:type 都予以省略）：

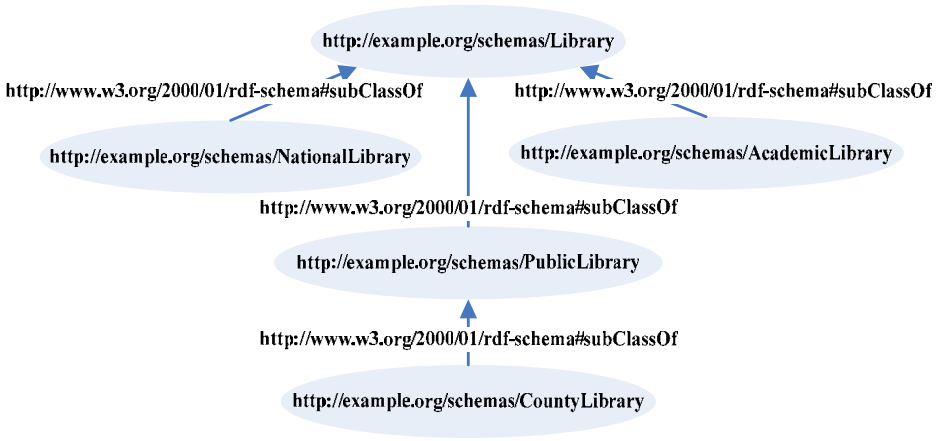

圖 4-10　圖書館的類型階層

上方圖 4-10 的 RDF 圖形，若以 RDF/XML 來表示，有二種寫法，分別如編碼實例 4-10 及編碼實例 4-11 所示：

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

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

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

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

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

<rdf:Description rdf:ID="CountyLibrary">
```

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

</rdf:RDF>
```

編碼實例 4-10　圖書館的類型階層的 RDF/XML

RDF/XML 提供一種描述具有 rdf:type 屬性（類型節點〔Typed Nodes〕）的資源的簡寫方式。因為 RDF Schema 的類也是 RDF 資源，因此這種簡寫方式可應用於對「類」的描述。(Manola & Miller, 2004) 簡寫方式如下方編碼實例 4-11 所示。

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

<rdfs:Class rdf:ID="Library"/>

<rdfs:Class rdf:ID="PublicLibrary">
  <rdfs:subClassOf rdf:resource="#Library"/>
</rdfs:Class>

<rdfs:Class rdf:ID="NationalLibrary">
  <rdfs:subClassOf rdf:resource="#Library"/>
</rdfs:Class>

<rdfs:Class rdf:ID="AcademicLibrary">
  <rdfs:subClassOf rdf:resource="#Library"/>
```

```
</rdfs:Class>

<rdfs:Class rdf:ID="CountyLibrary">
  <rdfs:subClassOf rdf:resource="#PublicLibrary"/>
</rdfs:Class>

</rdf:RDF>
```

編碼實例 4-11　簡寫圖書館的類型階層之 RDF/XML

編碼實例 4-10 及編碼實例 4-11 中的 RDF/XML，用 rdf:ID 來為所描述的資源（類）指定名稱（例如 Library），這相當於為資源（類）指派了一個相對於 Schema 文件的 URIs。rdf:ID 不僅是 URIs 的簡寫型式，同時也確保 rdf:ID 屬性（Attribute）的值，對目前的基準（Base）URI 的唯一性。在 RDF Schema 中定義類及屬性的名稱時，以這種方式編碼的話將有助於檢查是否出現重複的 rdf:ID 值。(Manola & Miller, 2004)

2. 描述屬性（property）

使用者除了描述事物的特定種類之外，可能也需要描述構成事物種類的特定屬性（例如用 paperback 描述書本）。在 RDF Schema 中，屬性是用 RDF 類 rdf:Property，及 RDF Schema 屬性 rdfs:domain、rdfs:range、與 rdfs:subPropertyOf 來描述。

在 RDF 中的所有屬性都可利用 RDF Schema 屬性 rdfs:range 和 rdfs:domain 來進一步描述有特殊應用的屬性。rdfs:range 屬性用來表示，屬性值為指定的類的實例。一個屬性（例如 ex:hasAuthor）可以有零、一、或多個值域屬性。若 ex:hasAuthor 沒有值域屬性，那麼就表示對 ex:hasAuthor 屬性的值沒有任何限制；若 ex:hasAuthor 有多個值域屬性，則表示其值是所有被值域指定的類的實例。rdfs:range 屬性也可用於表示屬性值為一個「類型文字」。rdfs:domain 表示某屬性應用於被指定的類，也就是指定該屬性的主詞為何。一個屬性可以有零、一、或多個主題領域屬性。

若某屬性沒有主題領域屬性，那麼就表示任何資源都可作為該屬性的主詞；若某屬性有多個主題領域屬性，則表示該屬性的所有資源，是所有被指定為主題領域的類的實例。(Manola & Miller, 2004)

此外，可用 rdfs:subPropertyOf 屬性表示一個屬性是另一個屬性的子屬性（Subproperty）。一個屬性可以是零個、一個、或多個屬性的子屬性。當 rdfs:range 和 rdfs:domain 屬性應用於某個 RDF 屬性時，它們也會應用於該屬性的子屬性。(Manola & Miller, 2004)下方以編碼實例 4-12 為例，表示如何用 RDF/XML，並透過上述的 RDF 類及屬性詞彙來描述圖書館的類型階層。

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

<rdfs:Class rdf:ID="Library"/>

<rdfs:Class rdf:ID="PublicLibrary">
  <rdfs:subClassOf rdf:resource="#Library"/>
</rdfs:Class>

<rdfs:Class rdf:ID="NationalLibrary">
  <rdfs:subClassOf rdf:resource="#Library"/>
</rdfs:Class>

<rdfs:Class rdf:ID="AcademicLibrary">
  <rdfs:subClassOf rdf:resource="#Library"/>
</rdfs:Class>
```

```
<rdfs:Class rdf:ID="CountyLibrary">
  <rdfs:subClassOf rdf:resource="#PublicLibrary"/>
</rdfs:Class>

<rdfs:Class rdf:ID="Member"/>

<rdfs:Datatype rdf:about="&xsd;integer"/>

<rdf:Property rdf:ID="registeredTo">
  <rdfs:domain rdf:resource="#Library"/>
  <rdfs:range rdf:resource="#Member"/>
</rdf:Property>

<rdf:Property rdf:ID="nubmer">
  <rdfs:domain rdf:resource="#PublicLibrary"/>
  <rdfs:range rdf:resource="&xsd;integer"/>
</rdf:Property>

<rdf:Property rdf:ID="administrator">
  <rdfs:domain rdf:resource="#Library"/>
</rdf:Property>

<rdf:Property rdf:ID="primaryAdministrator">
  <rdfs:subPropertyOf rdf:resource="#administrator"/>
</rdf:Property>

</rdf:RDF>
```

編碼實例 4-12　完整的車型的種類階層的 RDF/XML

3. RDF 的類及屬性詞彙

RDF 是能促使結構化詮釋資料的編碼、交換、及複用的資料模型，而 RDF Schema 則擴充了 RDF 在指定主題領域詞彙（Domain

Vocabulary）及物件結構（Object Structure）上的不足。(Broekstra, Kampman, & van Harmelen, 2003)以下，進一步分別表列 RDF 的類及屬性詞彙：(Brickley & Guha, 2004)

表 4-4　RDF 類

類名（Class name）	說明
rdfs:Resource	RDF 描述的所有事物皆稱爲「資源」，所有資源都是 rdfs:Resource 這個類的實例。rdfs:Resource 是所有事物的類，其他的類都是 rdfs:Resource 這個類的子類。rdfs:Resource 爲 rdfs:Class 的實例。
rdfs:Literal	rdfs:Literal 爲文字值（例如字串和整數）的類。文字可爲純（Plain）文字或類型文字。其爲 rdfs:Class 的實例，而爲 rdfs:Resource 的子類。
rdf:XMLLiteral	XML 文字值（Literals Values）的類。rdf:XMLLiteral 爲 rdfs:Datatype 的實例，而爲 rdfs:Literal 的子類。
rdfs:Class	rdfs:Class 爲 rdfs:Class 的實例。
rdf:Property	RDF 屬性的類。rdf:Property 爲 rdfs:Class 的實例。
rdfs:Datatype	rdfs:Datatype 爲資料類型的類。rdfs:Datatype 爲 rdfs:Class 的實例及子類。每個 rdfs:Datatype 的實例，都是 rdfs:Literal 的子類。
rdf:Statement	rdf:Statement 爲 rdfs:Class 的實例。它是爲了表示 RDF 敘述當中的類。rdf:Statement 屬 rdf:predicate、rdf:subject、和 rdf:object 這三個屬性的主題領域。每個不同的

	rdf:Statement 實例，其 rdf:predicate、rdf:subject、和 rdf:object 這三個屬性，可以有相同的值。
rdf:Bag	rdf:Bag 爲 RDF「包裹容器」的類。其爲 rdfs:Container 的子類。rdf:Bag 這個類，是指容器內之資源的順序並不重要。
rdf:Seq	rdf:Seq 爲 RDF「序列容器」的類。其爲 rdfs:Container 的子類。rdf:Seq 這個類，是指容器內之資源的「容器成員之屬性」之數字的順序是很重要的。
rdf:Alt	rdf:Alt 爲 RDF「替換容器」的類。其爲 rdfs:Container 的子類。在處理過程中，容器內的第一個成員（即 rdf:_1 屬性的值）將作爲預設值。
rdfs:Container	rdfs:Container 爲 RDF 容器類（即 rdf:Bag、rdf:Seq、與 rdf:Alt）的最上層類。
rdfs:ContainerMembershipProperty	「容器成員之屬性」（rdf:_1, rdf:_2, rdf:_3…）的類，而 rdf:_1, rdf:_2, rdf:_3…是用來宣告該資源爲容器的成員。rdfs:ContainerMembershipProperty 爲 rdf:Property 的子類。
rdf:List	rdf:List 爲 rdfs:Class 的實例。它可用來作清單或類似清單結構的描述。

資料來源：Brickley, D. & Guha, R.V.(Eds.).(2004).RDF vocabulary description language 1.0: RDF Schema. W3C Recommendation 10 February 2004. Retrieved Sept. 20, 2004, from http://www.w3.org/TR/2004/REC-rdf-schema-20040210/

表 4-5　RDF 屬性

屬性名（Property Name）	說明	屬性的主題領域（Domain）	屬性的值域（Range）
rdf:type	rdf:type 爲 rdf:Property 的實例，用來表示資源是類的實例。	rdfs:Resource	rdfs:Class
rdfs:subClassOf	rdfs:subClassOf 屬性爲 rdf:Property 的實例，用來表示某類的所有實例，皆爲另一類的實例。	rdfs:Class	rdfs:Class
rdfs:subPropertyOf	rdfs:subPropertyOf 屬性爲 rdf:Property 的實例，用來表示和某屬性有關的所有資源，也因另一屬而有關性。	rdf:Property	rdf:Property
rdfs:domain	rdfs:domain 爲 rdf:Property 的實例，用來表示具有某屬性的任何資源，爲一或多個類的實例。	rdf:Property	rdfs:Class
rdfs:range	rdfs:range 爲 rdf:Property 的實例，用來表示某屬性的值，爲一或多個類的實例。	rdf:Property	rdfs:Class
rdfs:label	rdfs:label 爲 rdf:Property 的實例，可用來提供可供人閱讀的資源名稱。	rdfs:Resource	rdfs:Literal

rdfs:comment	rdfs:comment 爲 rdf:Property 的實例，可用來對資源提供可供人閱讀的描述。	rdfs:Resource	rdfs:Literal
rdfs:member	rdfs:member 爲 rdf:Property 的實例，其爲所有「容器成員之屬性」的最上層屬性；也就是說，每個「容器成員之屬性」和 rdfs:member 屬性間都有 rdfs:subPropertyOf 的關係。	rdfs:Resource	rdfs:Resource
rdf:first	rdf:first 爲 rdf:Property 的實例，其可用來對清單及其他類似清單結構的資源進行描述。	rdf:List	rdfs:Resource
rdf:rest	rdf:rest 爲 rdf:Property 的實例，其可用來對清單及其他類似清單結構的資源進行描述。	rdf:List	rdf:List
rdfs:seeAlso	rdfs:seeAlso 爲 rdf:Property 的實例，其用來指出可能提供有關主詞資源額外資訊的資源。	rdfs:Resource	rdfs:Resource
rdfs:isDefinedBy	rdfs:isDefinedBy 爲 rdf:Property 的	rdfs:Resource	rdfs:Resource

	實例，其用來指出定義主詞資源的資源。		
rdf:value	rdf:value 爲 rdf:Property 的實例，其可以用來描述結構化的值（Structured Values）。	rdfs:Resource	rdfs:Resource
rdf:subject	rdf:subject 爲 rdf:Property 的實例，其用來指定敘述的主詞。	rdf:Statement	rdfs:Resource
rdf:predicate	rdf:predicate 爲 rdf:Property 的實例，其用來指定敘述的述語。	rdf:Statement	rdfs:Resource
rdf:object	rdf:object 爲 rdf:Property 的實例，其用來指定敘述的受詞。	rdf:Statement	rdfs:Resource

資料來源：Brickley, D. & Guha, R.V.(Eds.).(2004).RDF vocabulary description language 1.0: RDF Schema. W3C Recommendation 10 February 2004. Retrieved Sept. 20, 2004, from http://www.w3.org/TR/2004/REC-rdf-schema-20040210/

除了以上表列的 RDF 的類及屬性詞彙之外，還有一個 rdf:nil 詞彙，它表示的是作為 rdf:List 之實例的資源，rdf:nil 可代表一個空清單（Empty List）或其他類似清單結構的資源。（Brickley & Guha, 2004）雖然 RDF 與 RDF Schema 已定義了不少詞彙，但其實它們仍只是基本的網路資源知識表徵方法，因其尚無法對下列事項加以定義：(Manola & Miller, 2004)

- 對屬性的基數限制。例如，一個人有且只有一位生父。
- 指定某屬性具有遞移性。
- 指定某屬性是某一個類的實例的唯一識別碼。
- 指定兩個不同的類（有不同的 URIs）實際上代表同一個類。
- 指定兩個不同的實例（有不同的 URIs）實際上代表同一個實例。
- 指定屬性的值域或基數限制，取決於屬性應用的類，例如，說一個足球隊的屬性 `ex:hasPlayers` 值的個數為 11，而這個屬性應用於籃球隊時，值的個數應是 5。
- 能夠透過對類的組合（例如交集、聯集）得到新的類，或定義某兩個類是相離的（即兩個類沒有共同的實例）。

四、RDF 之相關研究與應用

Miller(2004)曾表示，RDF 是可作為提供資產（Asset）管理、企業整合、及分享與複用網路資料的架構（Framework）。Lassila and Swick(1997)則進一步指出，以 RDF 標記詮釋資料可應用於不同領域，例如，在資源發掘（Resource Discovery）方面提供較好的搜尋能力；在編目方面，能描述在特定網站、網頁、或數位圖書館中的內容以及內容間的關聯；在智慧型軟體代理人（Intelligent Software Agents）方面，促進知識分享及交換；含有數位簽章（Digital Signatures）的 RDF 也會是建立電子商務，及協同合作等應用所需之「信賴網」（Web of Trust）的關鍵。本小節將分別介紹 RDF 於工具、發佈資料、資源組織、以及搜尋引擎之相關研究與應用。

（一）工具

1. RDF 編輯器

目前已發展的 RDF 編輯器包括 FRODO RDFSViz[9]、InferEd[10]、IsaViz[11]、Protégé Editor[12]、RDFAuthor[13]、RDFe[14]、RDFedt[15]、RDFViz[16]、

[9] http://www.dfki.uni-kl.de/frodo/RDFSViz/
[10] http://www.intellidimension.com/pages/site/products/infered/default.rsp
[11] http://www.w3.org/2001/11/IsaViz/

以及 Simplistic RDF Editor[17]等。RDF 編輯器一般來說，分為為文字型（如下方圖 4-11 的 InferEd）與視覺型（如下方圖 4-12 的 IsaViz）。

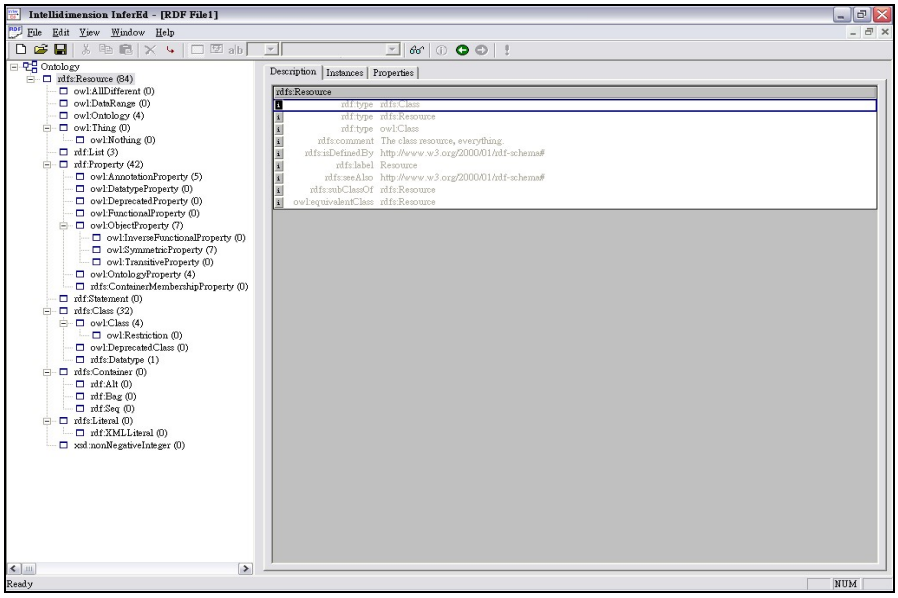

圖 4-11　InferEd 編輯畫面

[12] http://protege.stanford.edu/
[13] http://rdfweb.org/people/damian/RDFAuthor/
[14] http://infomesh.net/pyrple/rdfe/
[15] http://www.jan-winkler.de/dev/e_rdfe.htm
[16] http://www.ilrt.bris.ac.uk/discovery/rdf-dev/rudolf/rdfviz/
[17] http://jibbering.com/rdf/editor.html

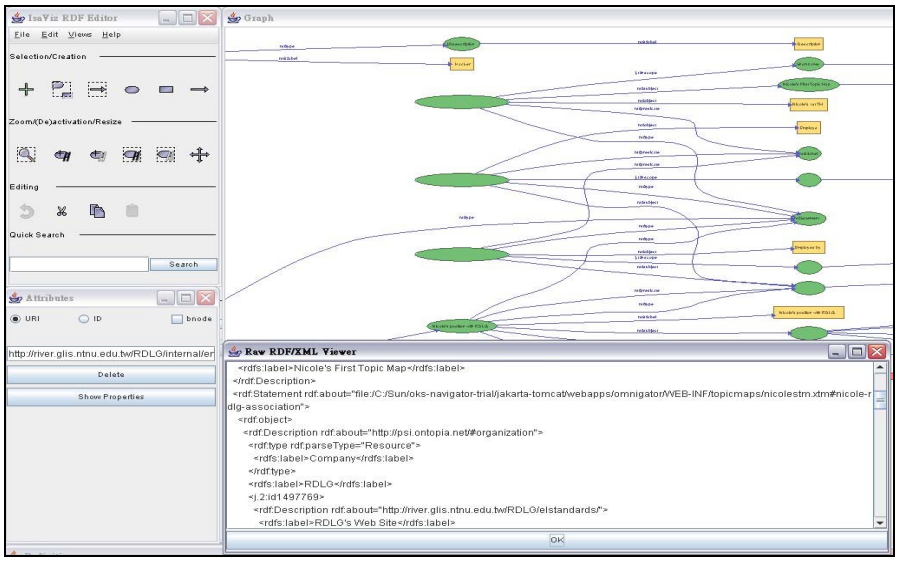

圖 4-12　IsaViz 編輯畫面

2. RDF 的呈現

(1) Longwell

Longwell RDF 瀏覽器[18]是由 W3C、惠普（HP）、MIT 圖書館、及 MIT 電腦科學與人工智慧實驗室[19]共同執行的 SIMILE 計畫[20]所開發，如圖 4-13 所示。能夠搜尋與瀏覽 RDF 資料集（Datasets），SIMILE 計畫更希望能將 Lonwell 與 DSpace[21]進一步整合，提昇 DSpace 的資源搜尋效率。

Longwell 在介面上的特殊之處在其「面向瀏覽」（Faceted Browsing）的功能，例如文件類型、出版年、與定義等。使用者可下載 Lonwell，匯入自己撰寫的 RDF 文件，並自行定義所欲顯示的特定詮釋資料欄位作為面向，供使用者搜尋與瀏覽。Longwell 目前並不支援中文。

---

[18] http://simile.mit.edu/longwell/
[19] http://www.csail.mit.edu/index.php
[20] SIMILE 全稱 “Semantic Interoperability of Metadata and Information in unLike Environments” 。SIMILE 計畫網站：http://simile.mit.edu/
[21] http://www.dspace.org/

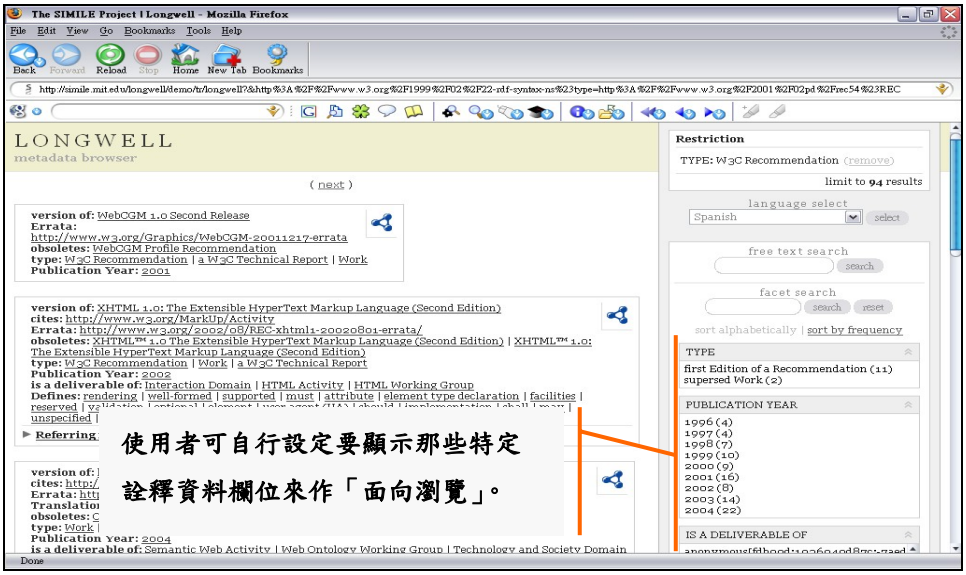

圖 4-13　Longwell RDF 瀏覽器[22]

(2) BrownSauce

BrownSauce RDF 瀏覽器[23]是由 Damian Steer 開發。作者有感於網路上已有許多 RDF 資料，因此嘗試開發能瀏覽分散各處的 RDF 資訊之瀏覽器。BrownSauce 的運作原理是，先將資料區分為可用的資料集，例如有關某人的資訊，接著將區分後的資料集合串聯（Aggregation）在一起。第一部分功能已經完成，使用者可藉由<rdfs:seeAlso>瀏覽不同資源，第二部分目前仍在開發建置中。圖 4-14 為其瀏覽畫面。

[22] 資料來源：Longwell RDF 瀏覽器—W3C Technical Reports。上網日期：民 94 年 5 月 13 日。網址：http://simile.mit.edu/longwell/demo/tr/longwell

[23] http://brownsauce.sourceforge.net/

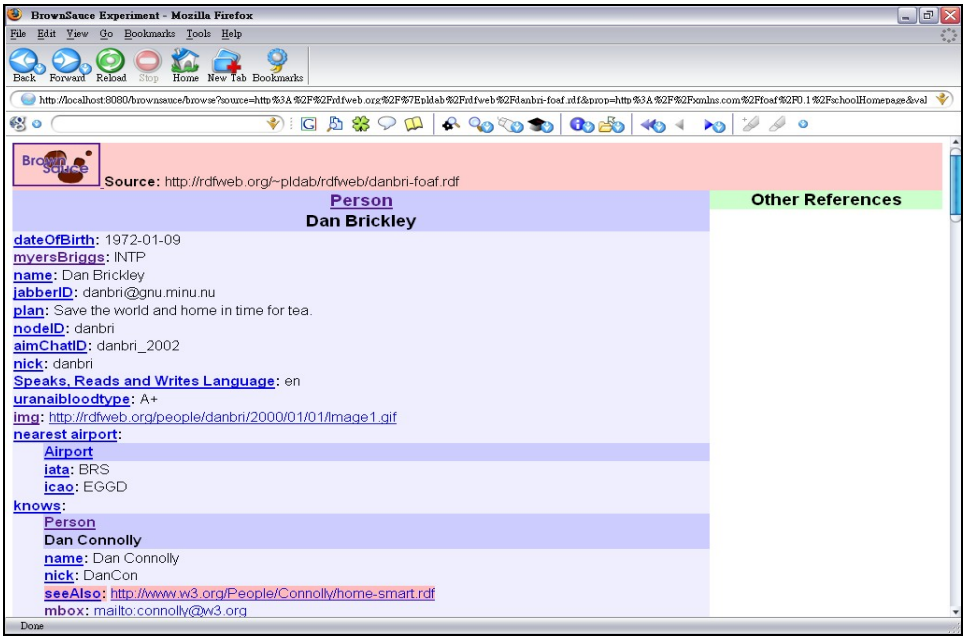

圖 4-14　BrownSauce RDF 瀏覽器

3. RDF 剖析（Parsing）與驗證（Validation）

RDF 剖析器能解析不同的 RDF 序列，透過程式介面或檢索來獲取 RDF「三元素」，並可對「三元素」進行一些基本處理。目前已發展的 RDF 剖析器與驗證器有 CARA[24]、Generic Interoperability Framework[25]、ICS-FORTH Validating RDF Parser[26]、Raptor RDF Parser Toolkit[27]、RDF Parser Based on Expat and、Redland[28]、SiRPAC[29]、SWI-Prolog RDF parser[30]、Transforming RDF with XSLT[31]、W3C RDF Validation Service[32]、Wilbur[33]、以及 XOTcl xoRDF Parser Module[34]等。

[24] http://zoe.mathematik.uni-osnabrueck.de/RDF/parser.html
[25] http://www-diglib.stanford.edu/diglib/ginf/
[26] http://139.91.183.30:9090/RDF/VRP/index.html
[27] http://librdf.org/raptor/
[28] http://librdf.org/demo
[29] http://www.w3.org/Library/src/HTRDF
[30] http://hcs.science.uva.nl/projects/SWI-Prolog/packages/sgml/online.html
[31] http://www.w3.org/XML/2000/04rdf-parse/
[32] http://www.w3.org/RDF/Validator/
[33] http://wilbur-rdf.sourceforge.net/docs/rdf-parser.html
[34] http://media.wu-wien.ac.at/RDF-Demo.html

下方以 W3C RDF Validation Service 為例，呈現 RDF 剖析器的基本功能。

本例是將本研究之某編碼實例送入 W3C RDF Validation Service 剖析器，使用者可以選擇顯示結果只顯示 RDF「三元素」、RDF 圖形、或兩者都顯示。按下「剖析」後，即可見剖析結果。下方圖 4-15 為剖析出來的 RDF「三元素」，而圖 4-16 則為剖析後的 RDF 圖形。

Triples of the Data Model

Number	Subject	Predicate	Object
1	genid:ARP91263	http://purl.org/dc/elements/1.1/title	"Healthy Meat"
2	genid:ARP91264	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://www.w3.org/1999/02/22-rdf-syntax-ns#Bag
3	genid:ARP91264	http://www.w3.org/1999/02/22-rdf-syntax-ns# 1	"Jon Doe"
4	genid:ARP91264	http://www.w3.org/1999/02/22-rdf-syntax-ns# 2	"Karin Mustermann"
5	genid:ARP91263	http://purl.org/dc/elements/1.1/creator	genid:ARP91264

圖 4-15　剖析編碼實例後顯示 RDF 三元素

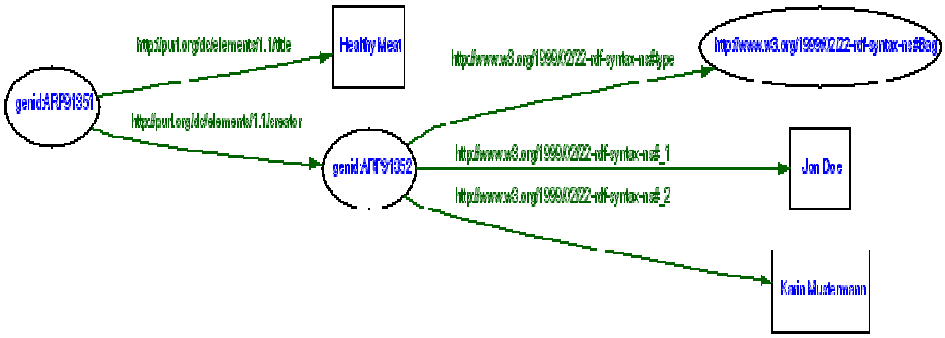

圖 4-16　剖析編碼實例後顯示 RDF 圖形

4.　RDF 儲存與查詢

RDF 的查詢就語法層次來看，雖然 RDF/XML 是以符合 XML 標準的方式來寫，但它並不適合用 XML 的查詢語言（Query Language）（例如 XQuery）來進行查詢。XQuery 是將資料結構轉成有被標示的節點的樹，而 RDF 不只是 XML 的標記語法，它的資料模型有別於 XML 的樹狀結構，RDF 的資料模型是圖，不是樹；此外，RDF 圖形的箭頭（述語）及結點（主詞與受詞）都是可以被標示的；再者，撰

寫 RDF 的語法也不一定要是 XML，且同一個 RDF 圖形，可以被表示成不同的 XML 序列，因此也就表示，所下的一個查詢其實並無法保證能從 RDF 模型中檢索出所有答案。(Broekstra, Kampman, & van Harmelen, 2003)

儘管有上述困難有代克服，但目前已發展出許多能用來儲存與查詢 RDF 資料的系統或 RDF 查詢語言，包括 Algae2[35]、IBM's RDF Query Specification[36]、Inkling[37]、Jena[38]、KAON[39]、Persistent RDF Databases[40]、rdfDB[41]、RDF Gateway: RDF Query Language（RDFQL）[42]、RDF Inference Language（RIL）[43]、RDF Query in Javascript[44]、RDF Query Language（RQL）[45]、RDF Squish query language[46]、RDFRDFStore[47]、Redland RDF Application Framework[48]、Sesame[49]、Storing RDF in a relational database[50]、TRIPLE[51]、以及 Versa[52]等。下圖為 ICS-FORTH RDFSuite 的 RQL 的檢索畫面。

[35] http://www.w3.org/1999/02/26-modules/User/Algae-HOWTO.html
[36] http://www.w3.org/TandS/QL/QL98/pp/rdfquery.html
[37] http://swordfish.rdfweb.org/rdfquery/
[38] http://jena.sourceforge.net/
[39] http://kaon.semanticweb.org/
[40] http://www.w3.org/1999/07/13-persistant-RDF-DB.html
[41] http://guha.com/rdfdb/
[42] http://www.intellidimension.com/default.rsp?topic=/pages/rdfgateway/reference/db/default.rsp
[43] http://xml.coverpages.org/RIL-20010510.html
[44] http://www.w3.org/1999/11/11-WWWProposal/rdfqdemo.html
[45] http://139.91.183.30:9090/RDF/RQL/index.html
[46] http://ilrt.org/discovery/2001/02/squish/
[47] http://rdfstore.sourceforge.net/
[48] http://librdf.org/
[49] http://www.openrdf.org/
[50] http://www-db.stanford.edu/~melnik/rdf/db.html
[51] http://triple.semanticweb.org/
[52] http://uche.ogbuji.net/tech/rdf/versa/versa.doc?xslt=/ftss/data/docbook_html1.xslt

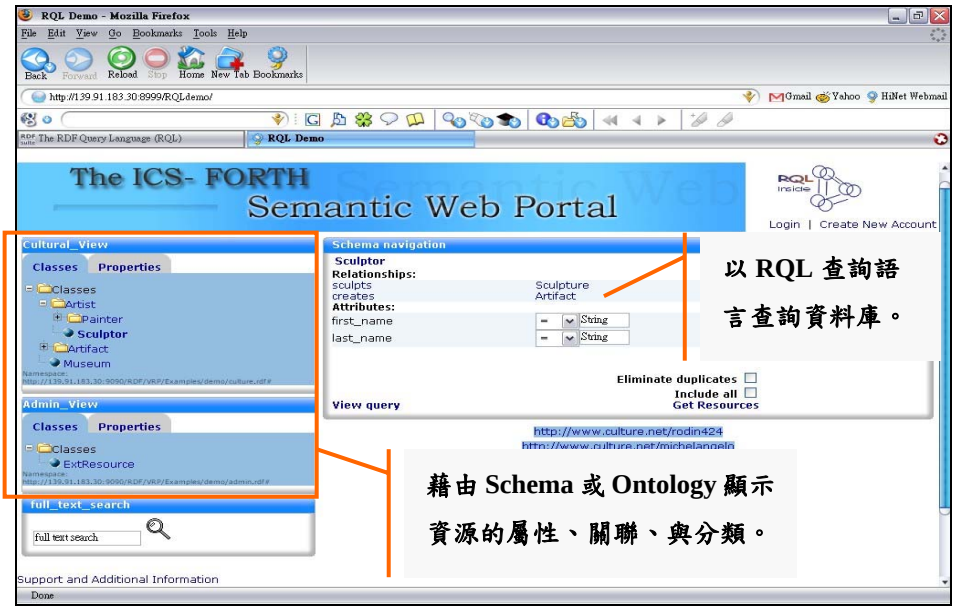

圖 4-17　ICS-FORTH RDFSuite 的 RQL 線上試用版[53]

W3C 的調查報告（Clark, 2005）也顯示，雖然目前已發展出許多查詢語言，但卻沒有任何一個語言成為能夠查詢與存取（Access）RDF 資料的標準。查詢語言缺乏標準，而在與本地或遠端的 RDF 伺服器互動方面，也欠缺資料存取協定（Data Access Protocol）標準，即使已有基本的標準化存取協定，如 HTTP、SOAP、或 XML-RPC，但仍缺乏共通且能存取多樣伺服器資料的標準。這些語言缺乏通的語法與語意，現有的查詢語言在語意上涵概了宣告式（Declarative）、類結構化查詢語言（SQL-like Languages）、以及路徑語言（Path Languages）等；再者，現有的語言在延伸性與內建能力（例如推論與分散式查詢）方面亦不盡相同。目前應用 RDF 查詢語言或協定或兩者，來嘗試解決的問題主要有十三類，如表 4-6 所示。

表 4-6　RDF 查詢語言或協定之應用類型

類型	實例
個人資訊管理 （Personal Information Management）	• 查找 Email Address、照片分享、個人資料識別

[53] 資料來源：ICS-FORTH RQL。上網日期：94 年 3 月 22 日。網址：http://139.91.183.30:8999/RQLdemo/

出版（Publishing）	• 找尋未知的媒體物件，如書籍、電影、與音樂等；查找新聞評論；建立目次
市場研究（Market Research）	找尋矛盾情況
交通業（Transportation）	避免交通擁塞
多媒體（Multimedia）	監控新聞報導
社會網絡分析（Social Network Analysis）	查找有關人的新資訊
供應鍊管理（Supply Chain Management）	找尋摩托車零件之資訊
資料整合（Data Aggregation）	搜尋電影原聲帶
語意網（Semantic Web）	Ontology 建置工具
軟體開發（Software Development）	找尋測試案例的輸入與輸出文件
網路服務（Web Services）	企業網路服務
教學科技（Instructional Technology）	發掘學習資源
設備獨立（Device Independence）	客製化內容傳遞
醫療（Health Care）	瀏覽病歷資料
觀光（Tourism）	探索景點資訊

資料來源：整理自 Clark, K.G.(Ed.).(2005). RDF Data Access Use Cases and Requirements. Retrieved May 12, 2005, from http://www.w3.org/TR/2005/WD-rdf-dawg-uc-20050325/

（二）應用 RDF／XML 發佈資料

目前已有許多特定主題領域的社群，應用 RDF／XML 來將它們的資料發佈到網路上，例如：(Miller, 2004)

- 提供描述數位版權及保存智財權的創意公用（Creative Commons[54]）
- 發展互通的詮釋資料標準的 Dublin Core[55]
- 有關人際網絡的 FOAF[56]
- 與數位學習有關的 IMS 全球學習聯盟（IMS Global Learning

[54] http://www.creativecommons.org/
[55] http://dublincore.org/documents/dcmes-xml/; http://dublincore.org/documents/dcq-rdf-xml/
[56] http://www.foaf-project.org/

Consortium[57]）所制定的相關規範，以及學習物件詮釋資料（LOM[58]）

- 有關音樂的編目與交互參照的 MusicBrainz[59]
- 處理同步新聞的 XMLNews[60]、PRISM[61]、與 RSS 1.0[62]

以下，以 Dublin Core 為例，說明其如何應用 RDF／XML 來表示。Dublin Core 是於 1995 年 3 月由「國際圖書館電腦中心」（Online Computer Library Center，簡稱 OCLC）和「美國國家高速運算中心」（National Center for Supercomputing Applications，簡稱 NCSA）所聯合贊助的研討會中，由五十二位學者專家所制定，目的是希望建立一套描述網路資源的格式及方法，來協助資訊檢索。

目前 Dublin Core 定義了十五個詮釋資料元素，分別為：貢獻者（Contributor）、時空涵蓋範圍（Coverage）、創作者（Creator）、日期時間（Date）、描述（Description）、資料格式（Format）、識別碼（Identifier）、語文（Language）、出版者（Publisher）、關聯（Relation）、權限（Rights）、來源（Source）、主題（Subject）、題名（Title）、及資源類型（Type）。(DCMI Usage Board, 2005)下圖即是應用 Dublin Core 詞彙的 RDF 圖形，敘述題名為 ”Healthy Meat” 的資源，其創作者為 ”Jon Doe” 和 ”Karin Mustermann”。

[57] http://www.imsglobal.org/rdf/
[58] http://ltsc.ieee.org/wg12/par1484-12-4.html
[59] http://www.musicbrainz.org/
[60] http://www.xmlnews.org/
[61] http://www.prismstandard.org/
[62] http://web.resource.org/rss/1.0/

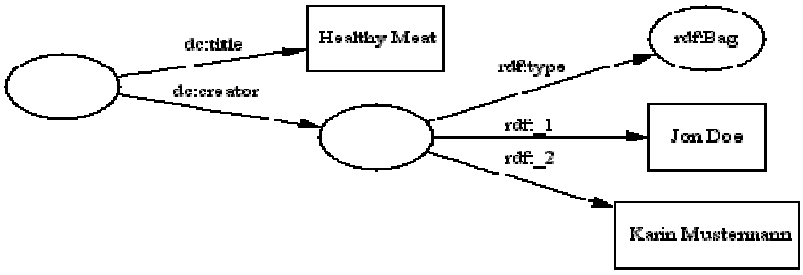

圖 4-18　應用 Dublin Core 的 RDF 圖形

資料來源：Kokkelink, S. & Schwänzl, R.(2002). Expressing qualified Dublin Core in RDF / XML. Retrieved March 4, 2005, from http://dublincore.org/documents/2002/05/15/dcq-rdf-xml/

圖 4-18 可表示為如下方編碼實例 4-13 的 RDF/XML：(Kokkelink & Schwänzl, 2002)

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
          xmlns:dc="http://purl.org/dc/elements/1.1/"
          xmlns:dcterms="http://purl.org/dc/terms/">
<rdf:Description dc:title="Healthy Meat">
 <dc:creator>
   <rdf:Bag>
     <rdf:li>Jon Doe</rdf:li>
     <rdf:li>Karin Mustermann</rdf:li>
   </rdf:Bag>
 </dc:creator>
</rdf:Description>
</rdf:RDF>
```

編碼實例 4-13　應用 Dublin Core 的 RDF/XML

（三）對資源加入以 RDF/XML 標記的詮釋資料

1. 網頁

若欲對網頁產生以 Dublin Core 描述的 RDF/XML 檔，可以利用英國巴斯大學（University of Bath）的 UKOLN 組織所開發的 DC-dot[63] 工具。下方圖 4-19 即是以 DC-dot 來描述師大圖資所網頁的編輯畫面，編輯好後按下 “re-submit” 即可產生 RDF/XML，然後我們可以儲存這份 RDF/XML（例如檔名為 `index.html.rdf`），並在所描述的 HTML 網頁的`<head>`部分加入`<link>`標籤（如下），讓網頁與描述該網頁的 RDF/XML 產生關聯。

```
<link rel="meta" href="index.html.rdf">
```

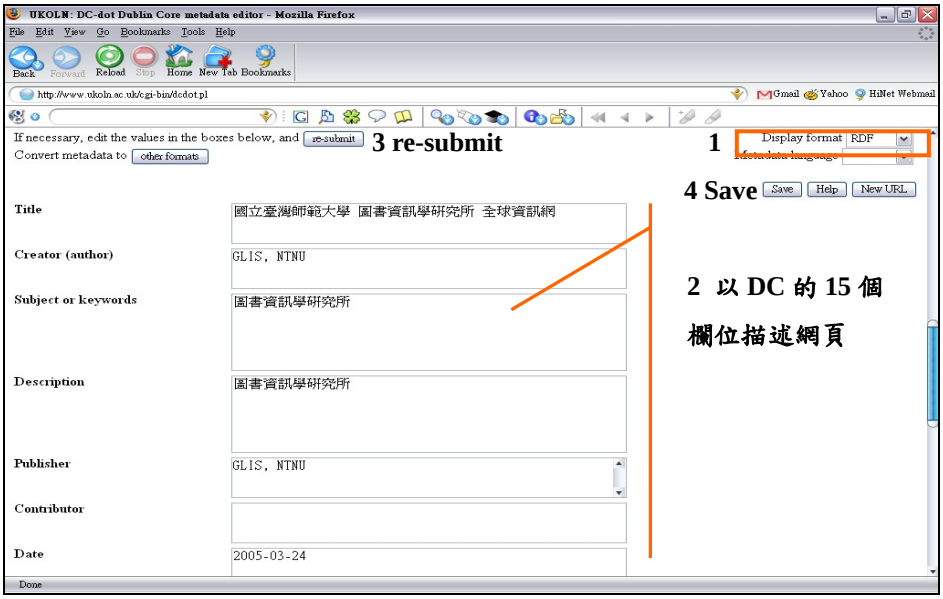

圖 4-19　用 DC-dot 產生以 Dublin Core 描述網頁的 RDF/XML

63 http://www.ukoln.ac.uk/metadata/dcdot/

2. 圖片

用 RDFPic[64]對圖片嵌入以 Dublin Core 描述的 RDF/XML，當使用者填完所欲描述的欄位後，選擇「匯出 RDF」，即可產生具有詮釋資料的圖片。

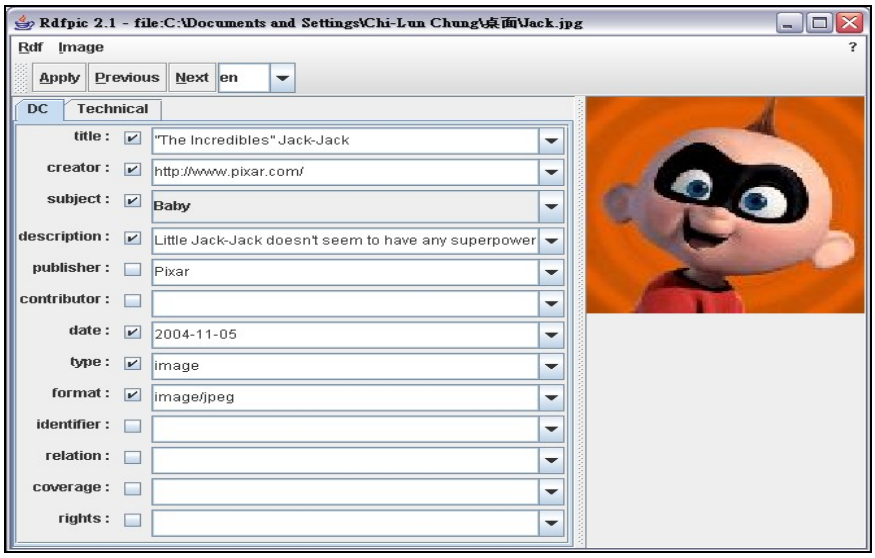

圖 4-20　對圖片嵌入以 Dublin Core 描述的 RDF/XML

（四）入口網站

1. 語意網環境目錄（SWED）

「語意網環境目錄」(The Semantic Web Environment Directory，簡稱 SWED[65]) 是由歐盟贊助的 SWAD-Europe 計畫[66]發展建置的目錄式入口網站。歐盟贊助的 SWED 計畫已於 2004 年十月結束，目前由 HP 在英國布裏斯托 (Bristol) 的實驗室接掌。SWED 目錄與現有一般目錄式入口網站的最大不同在，其並非以特定中央系統來儲存與管理資料，而是由提供資料的組織與計畫本身保存、維護、與發佈資料 (資

[64] http://jigsaw.w3.org/rdfpic/
[65] http://www.swed.org.uk/swed/index.html
[66] http://www.w3.org/2001/sw/Europe/

料以 RDF 格式儲存），SWED 只是擷取（Harvests）相關資料，然後顯示於目錄上。如圖 4-21 所示。此種運行方式的優點主要是可以減輕資料維護的負擔，並提高資料的複用性，因為任何人皆可擷取資料並以不同的方式呈現，同時亦可為資料添入其他資訊來豐富原資料的意義。

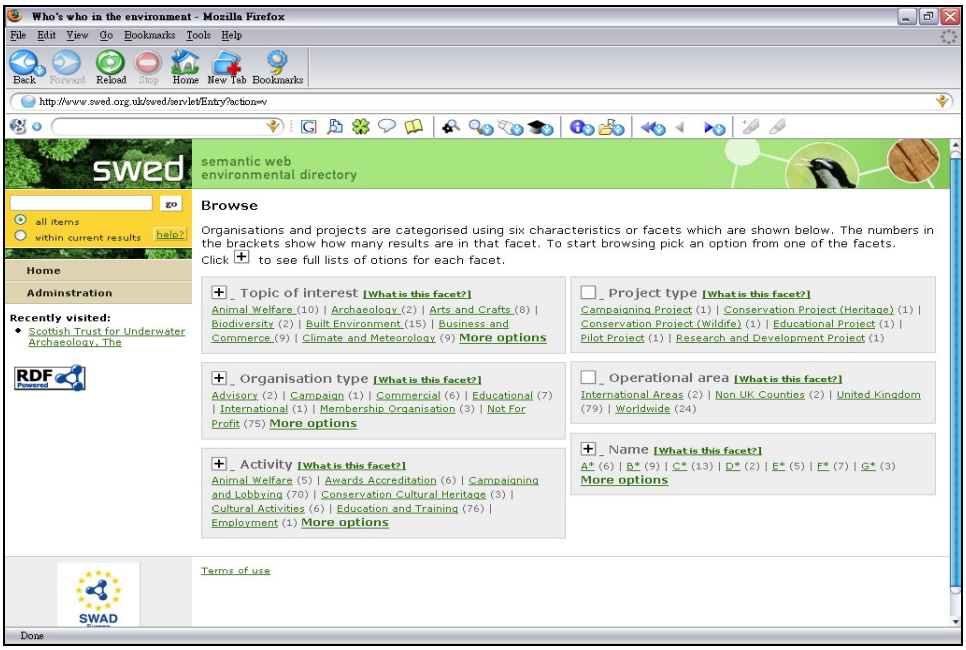

圖 4-21　SWED 目錄式入口網站[67]

2. Piggy-Bank—個人化資源整合入口

Piggy-Bank[68]是由 W3C、HP、MIT 圖書館、及 MIT 電腦科學與人工智慧實驗室共同執行的 SIMILE 計畫[69]所開發，是可外掛於 Mozilla Firefox 瀏覽器的軟體。只要網頁有用 RDF 來提供詮釋資料，Piggy-Bank 便能自動擷取網頁的 RDF 詮釋資料，使用者可自行決定儲存那些資料，然後由 Piggy-Bank 內建的面向瀏覽功能來瀏覽或查詢

[67] 資料來源：SWED Directory。上網日期：民 94 年 5 月 13 日。網址：http://www.swed.org.uk/swed/servlet/Entry?action=v

[68] http://simile.mit.edu/piggy-bank/

[69] SIMILE 全稱 “Semantic Interoperability of Metadata and Information in unLike Environments” 。SIMILE 計畫網站：http://simile.mit.edu/

原網頁資料；此外，使用者亦可對資源寫下評論，而所寫的評論在查詢時亦可被搜尋。Piggy-Bank 目前並不支援中文。如下圖所示。

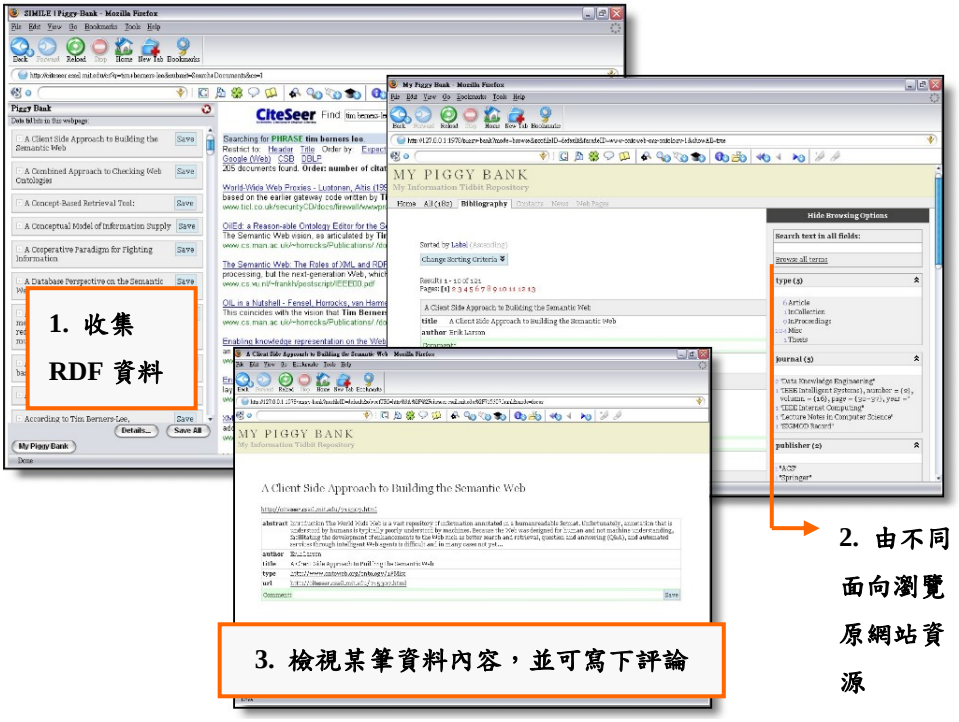

圖 4-22　Piggy-Bank 呈現畫面

（五）搜尋引擎

1. Intellidimension

Intellidimension[70]「語意網搜尋引擎」是由 Intellidimension 公司開發，採用 RDFQL 作為檢索語言。一般的網路搜尋引擎無法精確指定搜尋的資訊範圍，多半只能以關鍵詞搜尋，進行全文比對，搜尋結果可能會有不夠精確的問題，但對語意網搜尋引擎來說，使用者可以藉由語意網上定義良好的詞彙（如 `dc:creator`, `foaf:Person` 等）來進行搜尋。例如指定搜尋人（`foaf:Person`），其姓（`foaf:surname`）為 “Kidman” ，名（`foaf:firstName`）為

[70] http://www.semanticwebsearch.com/

“Nicole”，並進一步指定文件作者（dc:creator）需為 “Nicole Kidman”。下方圖 4-23 是用其 “Search Agent” 功能，設定搜尋任何資源類型其屬性「出版者」包含 “W3C” 的任何資源。

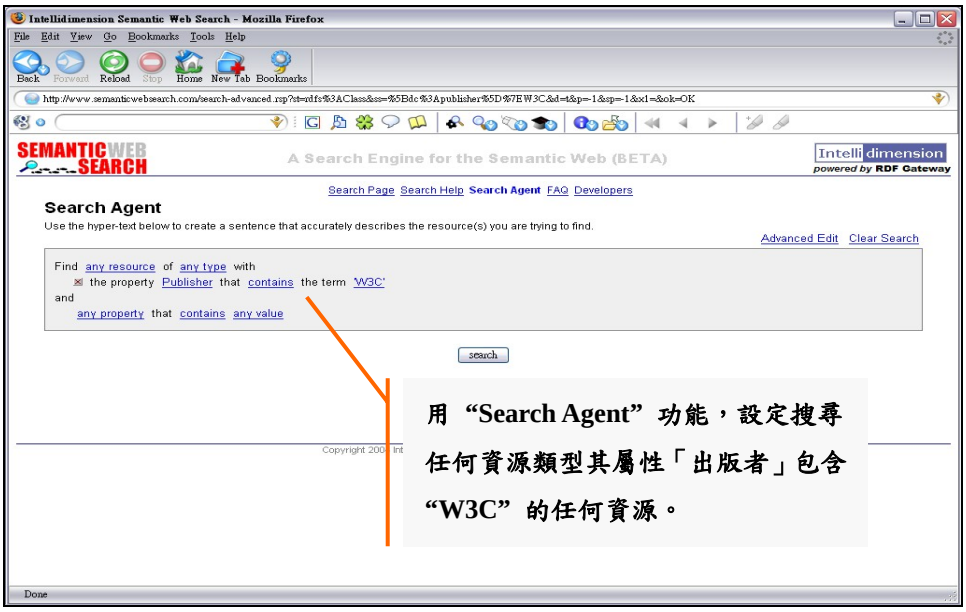

圖 4-23　Intellidimension「語意網搜尋引擎」

2. TAP

TAP[71]是由史丹佛大學知識系統實驗室（Knowledge Systems Laboratory[72]）、IBM Almaden 的知識管理研究團隊（Knowledge Management Group[73]）、以及 W3C 之語意網先進發展計畫（Semantic Web Advanced Development Initiatives[74]）共同開發。TAP 研發團隊認為「語意網」是要能處理有關人物、地點、與事件等物件（Objects），而非文字字串。為能提昇檢索效率，TAP 需能區分檢索詞彙所指涉的概念，因此 TAP 研發團隊進一步著手建置了 TAP KB 知識庫。

[71] http://panic.stanford.edu/sail
[72] http://ksl.stanford.edu/
[73] http://www.almaden.ibm.com/software/km/index.shtml
[74] http://www.w3.org/2000/01/sw/

TAP KB[75]知識庫以通俗物件（Popular Objects）為收錄對象，主題涵蓋音樂、電影、人物、公司、玩具、地點、與運動等主題之語彙（Lexical）與分類（Taxonomic）資訊。當使用者建入檢索詞彙後，TAP 會搜尋 Google 及 TAP KB，若知識庫中有該詞彙存在，則畫面右方會自動由 TAB KB 中擷取出有關該詞彙的資訊。下圖是以搜尋“tim burners lee“ 的檢索結果。

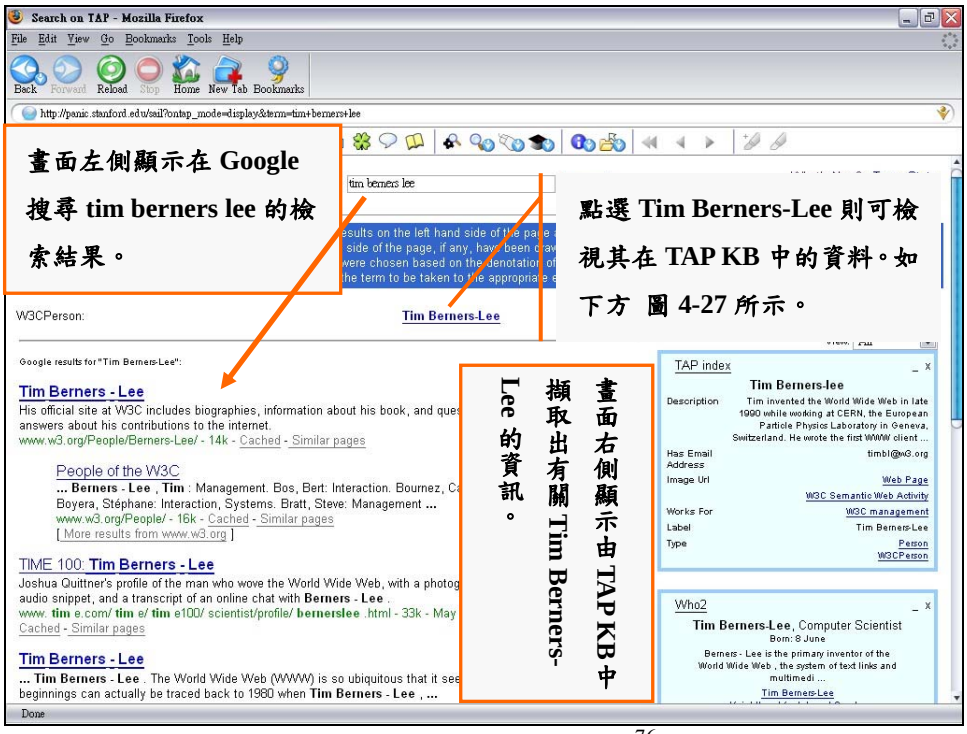

圖 4-24　TAP 檢索結果[76]

下圖是由上方圖 4-24 中的 TAP 檢索結果進一步點選檢索詞彙“Tim Berners-Lee“ ，來瀏覽該詞彙在 TAP KB 中的資訊。

[75] http://tap.stanford.edu/tap/tapkb.html

[76] 資料來源：Search on TAP。上網日期：民 94 年 5 月 19 日。網址：http://panic.stanford.edu/sail?ontap_mode=display&term=tim+berners+lee

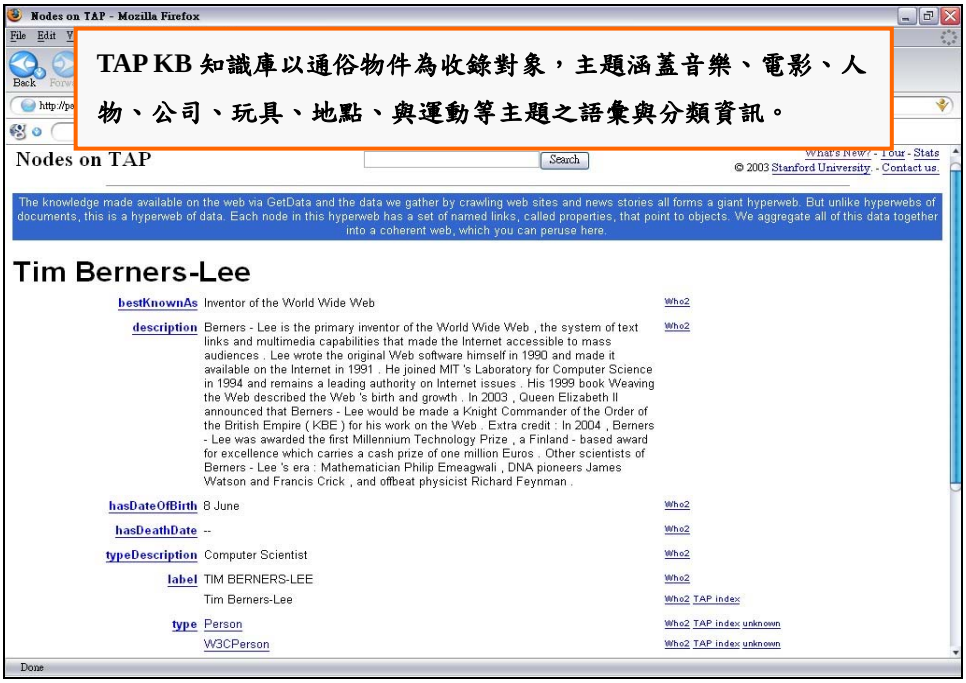

圖 4-25　檢視 TAP KB 中有關 Tim Berners-Lee 的資訊[77]

五、小結

RDF 文件及 RDFS 基本上可由語法、結構、及語意三種層面觀之，在語法層次上，它們都是 XML 文件[78]；在結構層次上，它們都是由一組「三元素」所組成；而在語意層次上，它們可以形成一或多個含有某些預先限定語意的圖形。(Broekstra, Kampman, & van Harmelen, 2003)RDF 除了可作為支援網路資源描述的資料模型和語法外，同樣也提供一種共通的結構，能有助 XML 資料的交換與互通。

RDF 是 W3C 所訂定的規範，是基於「三元素」的資料模型，該模型既簡單又複雜，簡單是因為每個敘述都是由「三元素」構成，複雜則是因為每個敘述之間又可能有關聯而彼此相連。總而言之，RDF

[77] 資料來源：Nodes on TAP。上網日期：民 94 年 5 月 19 日。網址：http://panic.stanford.edu/babble?ontap_mode=display&oid=tap%3AW3CPersonBerners%2DLee%2C%5FTim

[78] 表達 RDF 雖然並非一定要用 XML 語法，但 XML 是最廣爲使用的語法。

提供了一種表達資訊，並使資訊能在應用程式間交換，且不喪失語意的通用架構；此外，它也是建立 Ontology、進行邏輯推理、以及達成「語意網」的技術、工具、或方法。

第二節 Topic Maps 之描述與解釋

本節擬說明 Topic Maps 規範發展的過程、Topic Maps 之基本概念、XTM 語法、以及 Topic Maps 之相關研究與應用。

一、Topic Maps 規範之發展與概要

（一）標準與規範的發展過程

主題地圖典範（Topic Maps Paradigm）的起源可追溯至 1993 年，其首度在 Davenport Group 研究團體的工作文件中提出。爾後，在圖形傳播學會（Graphic Communication Association，簡稱 GCA。今日的 IDEAlliance）贊助的 HyTime 應用大會（Conventions for the Application of HyTime[79]）進一步發展，並自該活動之後開始獨立發展、執行、與公布。

到了 2000 年初期，主題地圖典範成為 ISO 國際標準（ISO/IEC 13250:2000），TopicMaps.Org 組織亦隨即成立，致力於將 ISO/IEC 13250:2000 應用於全球資訊網，並在 2001 年公布了 XML Topic Maps（XTM）v1.0。XTM 和 ISO 標準最大的不同在於，XTM 是以 XML 作為標記語法，XLink 作為連結語法（Linking Syntax）；而 ISO 13250 是以 SGML 為標記語法，HyTime 為連結語法。XTM 發展過程如下圖所示。

[79] HyTime 應用大會是由 S. R. Newcomb 與 M. Biezunski 主持，目的在發展一個比 HyTime 更易於瞭解與應用的子集。爲了能自動整合與處理索引，Newcomb 提出一可以記錄內含於書後索引的知識架構，並與 Biezunski 共同設計解決方案，稱爲「主題導航地圖」（Topic Navigation Maps），此爲「主題地圖」的前身。（林信成、歐陽慧、歐陽崇榮，民 92）

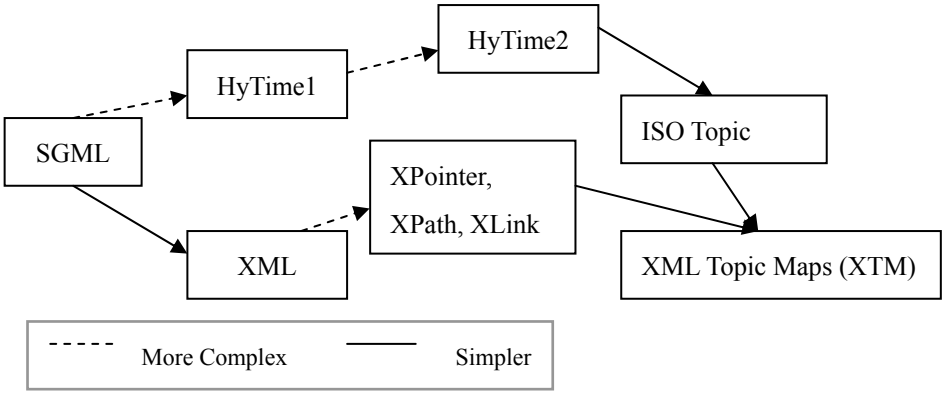

圖 4-26　XTM 發展過程

資料來源：Biezunski, M. (2003). Introduction to the Topic Maps Paradigm. In J. Park & S. Hunting (Ed.), XML Topic Maps: Creating and Using Topic Maps for the Web (pp. 17-30). Boston: Addison-Wesley.

Topic Maps 原本是為了改善索引問題而發展，不過，它的應用範圍則是超越了它原本的預期。Topic Maps 提供建立與交換知識架構之標準化資料模型。

（二）Topic Maps 規範概要

XTM 規範提供呈現資訊資源結構的模型及文法，藉此可定義主題及主題間的關聯。主題的特徵包括：主題的「名稱」、主題的「資源指引」、以及主題在關聯中所扮演的「角色」。主題在「範圍」內有其特徵；也就是說，在範圍內之特定情境下的名稱與資源，才會被視為是該主題的名稱、與資源等特徵。Topic Maps 亦可加以合併。「合併」是由 Topic Maps 作者在建立 Topic Maps 時有予以指示，或在執行中經過使用者或應用程式的判斷而發生。一或多個使用並遵守此文法且相互關聯的文件，即稱為 Topic Maps。(Pepper & Moore, 2001)

總而言之，Topic Maps 可說是將主題、關聯、及資源三者，運用主題索引的概念及網路的特性加以結合，提供新的知識組織模式，讓使用者可以定義複雜的知識架構，創造虛擬的知識地圖；同時，也幫

助使用者能有效的瀏覽資訊資源的一套標準方法。

二、Topic Maps 基本概念

（一）主題、關聯、資源指引—T. A. O.

Topic Maps 關注的焦點在資源的意義為何；換句話說，其所建立的標記是主題導向，不是資源導向。(Ahmed, 2002)Topic Maps 的基本單元是「主題[80]」、「關聯」與「資源指引」。「關聯」的作用是將不同的主題關聯在一起；其二則為「資源指引」，負責連結和主題有關之資源。綜上所述，「主題」、「關聯」、與「資源指引」即為 Topic Maps 的基本核心概念。其關係如圖 4-27 所示。

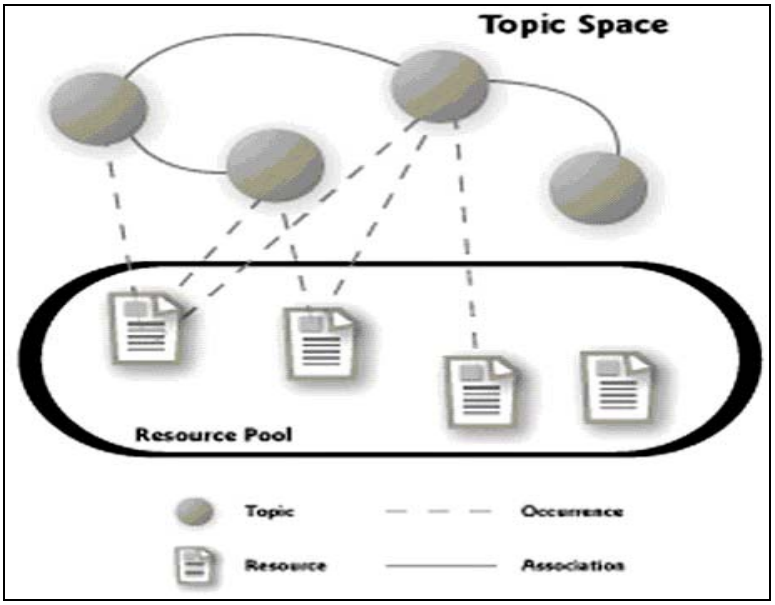

圖 4-27　Topic Maps 之基本概念

資料來源：Ahmed, K. (2002). Introducing Topic Maps: A powerful, subject-oriented approach to structuring sets of information. XML Journal, 3(10). Retrieved Aug. 6, 2004, from www.sys-con.com/xml/articleprint.cfm?id=507

[80] 一個「主題」(Topic) 通常會有一個名稱(Name) ，不過它也可能沒有名稱或有許多名稱 (例如一個主題在不同的語言或不同的學科之下，會有不同的名稱) 。所以，主題地圖不是將名稱連結在一起，而是將可能有多個名稱的主題連結起來。

Topic Maps 文件是由主題、建立主題間之關係的關聯、以及指出和主題有關之資源等方式共同組成。每一個在關聯中的主題都扮演著某種角色。此外，主題、資源指引、以及關聯，皆可透過其他主題加以定訂其類型。總而言之，Topic Maps 不是讓使用者邊走邊找所需的資訊，而像是替分散的資源鋪上地圖，提供使用者該往何處去之指引或參考；而在瀏覽的同時，也能幫助使用者建立該主題的知識。

（二）範圍、合併、已公布的主題指標

Topic Maps 除了主題、關聯、與資源指引之外，還有幾個重要概念。第一個要說明的就是「範圍」。在 Topic Maps 內，對每一個表徵主題的事物（例如主題的名稱、主題的資源指引、以及主題在關聯中所扮演的角色）皆可限定其範圍。換句話說，主題的名稱、資源指引、以及其在關聯中扮演的角色，都是在特定的「範圍」內才有效。

不同的主題有相同的名稱時，並不一定表示它們和同一個主題有關；但若在同一範圍內之主題有同樣的名稱時，就可以合理推斷它們指涉的是同一個主題 （Subject），此時，就可以將這些主題加以「合併」。此外，如果有兩個 Topic Maps，分別都有<topic>元素標示同一個主題（Subject），那麼合併這兩個 Topic Maps 文件的結果，必然是只產生一個主題，而該主題結合（Union）了那兩個<topic>元素的特徵（包括主題的名稱、主題的資源指引、以及主題在關聯中所扮演的角色）。綜上所述，「範圍」可以作為觸發或防止合併的機制。「合併」可能發生的情況則包括，在同一個主題地圖內的很多主題，或是在不同主題地圖間之主題。

「主題指標」（Subject Indicator）是指由 Topic Maps 作者提供主題（Subject ）之識別（Identity）一個既實際又明確的標誌（Indication），此標誌被視為是資源。「已公布的主題」（Published Subject），是指任何由主題指標提供給大眾使用，且可透過 URI 獲取的主題（Subject） 。而「已公布的主題指標」（Published Subject Indicators，

以下簡稱 PSI[81]），是公布主題指標，來幫助 Topic Maps 之交換和管理。因此，穩定、專業、與可信，對 PSI 的品質來說就顯得相當重要[82]。(Vatant, 2003)PSI 可以讓 Topic Maps 作者確保該 Topic Maps 在知識社群的一致性，也能讓電腦快速的比較及合併 Topic Maps。

進一步而言，Topic Maps 的名稱和 PSIs 皆可作為識別主題（Subject）的工具。因此，合併 Topic Maps 的規則也就可以區分為兩種：(Biezunski, 2003)

- 受主題名稱限制的合併規則（Topic Naming-constraint-based）：若在同一範圍內之主題有相同的基本名稱（Base Name），則這些主題要加以合併。
- 以主題為本的合併規則（Subject-based）：將有相同「主題識別」的主題加以合併。例如它們的<subjectIdentity>元素都指向同一資源時。

雖然第二種合併規則較第一種合併規則可靠，但主題所指的資源必須要一致。此外，以主題為本的合併規則，只有在已公布的主題廣為不同的 Topic Maps 作者知道的情況才較可行；不過，正因如此，因此以主題為本的合併規則相對則鼓勵了使用社群在網路上公布與分享這些提及同樣主題的 Ontology。

[81] 「已公布的主題指標」（Published Subject Indicators）之發行、管理、與使用，目前由 OASIS 主題地圖主題公布技術委員會（OASIS Topic Maps Published Subject Technical Committee）主導。它的目標是對已公布的主題（Subject）加以應用，來實現主題地圖的互通性之目標。更進一步的目的還包括，達成主題地圖和其他明顯使用主題之抽象表示技術（例如 RDF 以及 OWL）間之互通性。OASIS 的全稱是“Organization for the Advancement of Structured Information Standards”。OASIS 主題地圖主題公布技術委員會的官方網站：
http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=tm-pubsubj

[82] 穩定（Stability）是指資源的位址（Address）必需要能持久。專業是指主題（Subject）的定義，必須由權威的機構或組織來驗證。而可信（Trust）是表示唯有在穩定與專業的情況下，才能使 PSI 可信。

三、XTM 語法簡介

本節簡單介紹 XTM 的語法。首先，說明 XTM 的 19 個元素；其次，以實例展示如何撰寫 XTM；最後，說明建立 Topic Maps 的基本過程與 XTM 語法。

（一）XTM 元素

XTM 的 19 個元素，表列說明如下：

表 4-7　XTM 的 19 個元素

主題地圖（The Topic Map）	
<topicMap>	主題地圖文件的根元素。可運用的子元素包含<topic>、<association>、與<mergeMap>。
參照主題與主題指標（References to Topics and Subject Indicators）	
<topicRef>	參照主題元素
<subjectIndicatorRef>	參照主題指標
範圍與情境（Scope and Context）	
<scope>	參照構成範圍的主題。所以，被參照的主題有共同存在的情境。可運用的子元素包括<topicRef>、<resourceRef>、與<subjectIndicatorRef>。
類別與實例（Classes and Instances）	
<instanceOf>	指向可以表示主題類型的主題。可運用的子元素包括<topicRef>和<subjectIndicatorRef>。
主題（Topics and Subjects）	
<topic>	標示出主題。可運用的子元素包括<instanceOf>、<subjectIdentity>、<baseName>、和<occurrence>。
<subjectIdentity>	主題識別。可透過<resourceRef>、

	<subjectIndicatorRef>或<topicRef>子元素，來指明主題所具體化的主題 (subject) 。
主題名稱 (Topic Names)	
<baseName>	主題的基本名稱。可運用的子元素包括 <scope>、<baseNameString>、與 <variant>。
<baseNameString>	主題基本名稱之字串內容。
<variant>	主題的基本名稱以外的名稱；變異名稱。可運用的子元素包括<parameters>、<variantName>、及<variant>。
<variantName>	變異名稱之容器。可運用的子元素包括 <resourceRef>與<resourceData>。
<parameters>	變異名稱（<variant>）的處理情境。可運用的子元素包括<topicRef>和 <subjectIndicatorRef>。
關聯與成員 (Associations and Members)	
<association>	主題間之關聯。可運用的子元素包括 <instanceOf>、<scope>、及<member>。
<member>	關聯成員。指出在關聯中，扮演某種角色的主題。可運用的子元素包括 <roleSpec>、<topicRef>、<resourceRef>、與<subjectIndicatorRef>。
<roleSpec>	指明在關聯中，成員扮演的角色。可運用的子元素包括<topicRef>和 <subjectIndicatorRef>。
資源指引和資源 (Occurrences and Resources)	
<occurrence>	資源指引。指出和主題相關的資源。可運用的子元素包括<instanceOf>、<scope>、<resourceRef>、與<resourceData>。
<resourceRef>	提供所要參照之資源的 URI。
<resourceData>	資源資料的容器。只能容納字元資料。

合併 (Merging)	
<mergeMap>	與其他主題地圖合併之用。可運用的子元素包括<topicRef>、<resourceRef>、與<subjectIndicatorRef>。

下方進一步以圖形呈現 XTM 的元素關係。

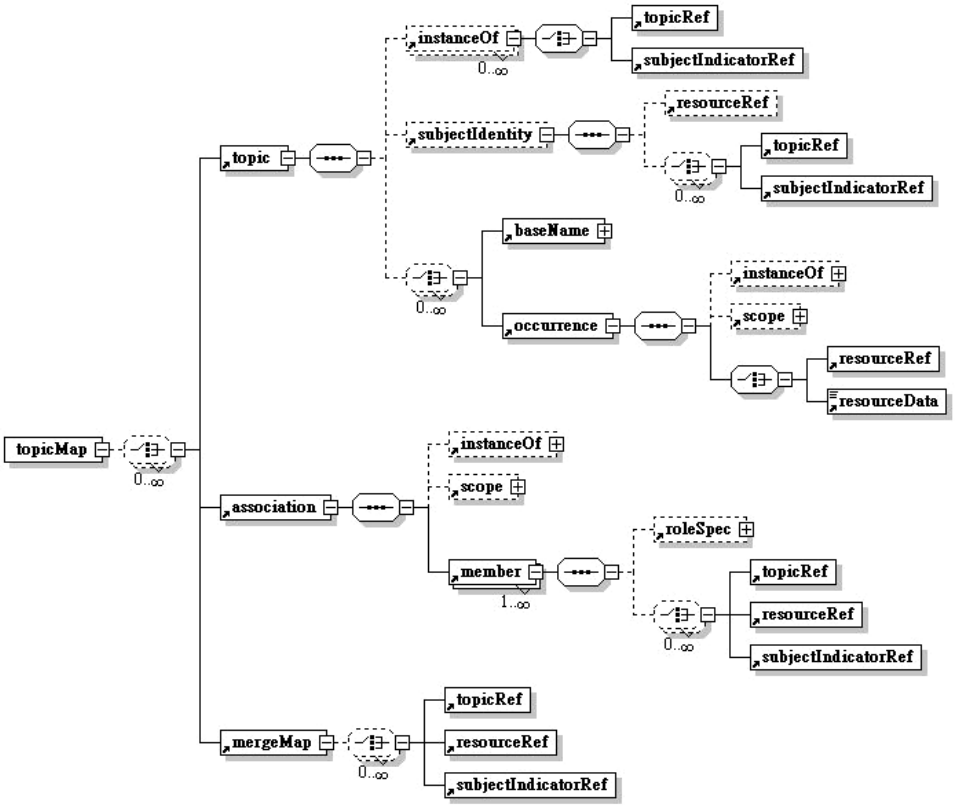

圖 4-28　XTM 元素

（二）語法實例

接下來，以實例說明如何撰寫 XTM。(Hunting, 2003)另有關建立 Topic Maps 文件之基本步驟，請參考下一小節：建立 Topic Maps 之基本過程與 XTM 語法。

1. 認識<topic>, <baseName>, <baseNameString>, 和<occurrence>

首先，建立一個有 ID 的主題元素。如此一來，才能方便之後識別這個主題。(ID 必須為字串，並在 Topic Maps 文件內具有唯一性)

```
<topic id="myTomato">
```

接著，替這個主題訂定的一個名稱。(以<baseNameString>元素將名稱包起來)

```
<topic id="myTomato">
  <baseName>
    <baseNameString>tomato</baseNameString>
  </baseName>
</topic>
```

幫這個主題加入相關資源，讓人們可以清楚知道所談論的主題。本例指定了一張蕃茄的圖片，好讓網站可以顯示它。

```
<topic id="myTomato">
  <baseName>
    <baseNameString>tomato</baseNameString>
  </baseName>
  <occurrence>
    <resourceRef xlink:href="tomato.gif"/>
  </occurrence>
</topic>
```

2. 認識<subjectIdentity>

這個部份要說明<resourceRef>、<subjectIdentity>、<subjectIndicatorRef>、與<topicRef>的使用情況。

- <subjectIndicatorRef>指明象徵（Indicates）某主題（Subject）的資源；
- 以<resourceRef>指明的資源，必須是組成該主題的資源（必須是可定址的）；
- <topicRef>必須指向<topic>元素。

提供〝tomato〞主題一個可為機器與人所「瞭解」的識別。美國農業部（USDA）曾對蕃茄的種類訂定規範，因此，本例將識別標示如下：

```
<topic id="myTomato">
  <subjectIdentity>
    <subjectIndicatorRef
xlink:href=http://www.fed.gov/usda/doc/tomatogr.htm#gradeA/>
  </subjectIdentity>
  <baseName>
    <baseNameString>tomato</baseNameString>
  </baseName>
  <occurrence>
    <resourceRef xlink:href="tomato.gif"/>
  </occurrence>
</topic>
```

3. 認識<scope>

下方的例子想表示<baseName>〝tomato〞可能只適用於英文的情境；對法文來說，<baseName>則為〝tomate〞。（以 FR 表示法文；EN 表示英文）

```
<topic id="myTomato">
  <baseName>
    <scope>
      <topicRef xlink:href="#EN"/>
```

```
    </scope>
    <baseNameString>tomato</baseNameString>
  </baseName>
  <baseName>
    <scope>
      <topicRef xlink:href="#FR"/>
    </scope>
    <baseNameString>tomate</baseNameString>
  </baseName>
  <baseName>
    <baseNameString>tomato</baseNameString>
  </baseName>
   ...
</topic>
```

上例中，我們未對第三個<baseName>限定其<scope>，因此表示其值〝tomato〞為<baseName>的預設值，在任何範圍都有效。

此時可以發現，由於在上例中使用<topicRef>參考的主題（EN與 FR），尚未存在於本 Topic Maps 文件中，因此我們需為其建立〝EN〞與〝FR〞的主題，並且使用 PSIs 表示其主題識別。（表示主題識別可用<resourceRef>、<subjectIndicatorRef>、與<topicRef>。本例用<subjectIndicatorRef>參照 PSI）

```
<topic id="EN">
  <subjectIdentity>
    <subjectIndicatorRef
xlink:href=http://www.topicmaps.org/xtm/1.0/language.xtm#en/>
  </subjectIdentity>
</topic>
<topic id="FR">
  <subjectIdentity>
```

```
   <subjectIndicatorRef
xlink:href=http://www.topicmaps.org/xtm/1.0/language.xtm#fr/>
  </subjectIdentity>
</topic>
```

4. 認識<association>, <member>, 和<roleSpec>

要建立蕃茄（Tomato）這個食材，與蕃茄炒蛋（Tomato Eggs）這道菜的關聯。首先，需先建立蕃茄炒蛋這個主題。

```
<topic id="tomatoEggs">
  <baseName>
    <baseNameString>tomato eggs</baseNameString>
  </baseName>
</topic>
```

接著，用<association>將蕃茄這個食材，與蕃茄炒蛋這道菜關聯起來，並利用<topicRef>定義其為關聯的成員（<member>）。

```
<association id="tomate_eggs_association">
  <member>
    <topicRef xlink:href="#myTomato"/>
  </member>
  <member>
    <topicRef xlink:href="#tomatoEggs"/>
  </member>
</association>
```

再下來，進一步指出這兩個主題在關聯中所扮演的角色。（角色需先用主題定義出來）

```
<topic id="anIngredient">
  <baseName>
```

```
    <baseNameString>an ingredient</baseNameString>
  </baseName>
</topic>
<topic id="aDish">
  <baseName>
    <baseNameString>a dish</baseNameString>
  </baseName>
</topic>
```

最後，用<roleSpec>將角色加入關聯中。

```
<association id="tomate_eggs_association">
  <member>
    <roleSpec>
      <topicRef xlink:href="#anIngredient"/>
    </roleSpec>
    <topicRef xlink:href="#myTomato"/>
  </member>
  <member>
    <roleSpec>
      <topicRef xlink:href="#aDish"/>
    </roleSpec>
    <topicRef xlink:href="#tomatoEggs"/>
  </member>
</association>
```

5. 認識<instanceOf>

上面的例子，我們區分了一道菜（Dish）及其原料（Ingredient），不過關聯本身的類型又是如何？因此，我們再建立一個主題〝ingredient_of〞。

```
<topic id="ingredient_of">
  <baseName>
    <baseNameString>ingredient of</baseNameString>
  </baseName>
</topic>
```

然後，修訂剛才建的<association>，指明其關聯類型為〝ingredient_of〞。

```
<association id="tomate_eggs_association">
  <instanceOf>
    <topicRef xlink:href="#ingredient_of"/>
  </instanceOf>
  <member>
    <roleSpec>
      <topicRef xlink:href="#anIngredient"/>
    </roleSpec>
    <topicRef xlink:href="#myTomato"/>
  </member>
  <member>
    <roleSpec>
      <topicRef xlink:href="#aDish"/>
    </roleSpec>
    <topicRef xlink:href="#tomatoEggs"/>
  </member>
</association>
```

此外，我們亦可回頭修訂蕃茄炒蛋該主題，定義其主題類型。（假定其為主菜 [Entree] ，並且假設我們之前已建立過〝entree〞這個主題）

```
<topic id="tomatoEggs">
  <instanceOf>
```

```
    <topicRef xlink:href="entree"/>
  </instanceOf>
  <baseName>
    <baseNameString>tomato eggs</baseNameString>
  </baseName>
</topic>
```

6. 認識<variant>, <variantName>, <parameters>與<resourceData>

假設主題地圖文件，有可能需要顯示在個人數位助理（PDA）上的話，或許我們會希望在 PDA 顯示的字母較短。(假設我們之前以建立了 ID 為〝pda〞的主題)

```
<topic id="myTomato">
  <baseName>
    <baseNameString>tomato</baseNameString>
  <variant>
    <variantName>
      <resourceData>TMT</resourceData>
    </variantName>
      <parameters>
        <topicRef xlink:href="#pda"/>
      </parameters>
    </variant>
  </baseName>
</topic>
```

（三）建立 Topic Maps 之基本過程與 XTM 語法

參考 Ontopia(2004)建議的建立 Topic Maps 基本步驟，本研究以下擬以員工（假設姓名為 Nicole Kidman）和公司（假設名稱為 RDLG）的情境為例（Nicole Kidman 為 RDLG 公司的開發人員），簡要說明建立 Topic Maps 之基本過程與 XTM 語法。

1. 決定範圍與 Ontology

建立 Topic Maps 的首要步驟是，界定 Topic Maps 所要涵蓋的範圍，也就是呈現的主題領域為何；其次，設計 Ontology，亦即明確描述主題領域內所包含之事物的種類與本質。若以員工和公司為例，範圍可能包括同事、員工負責的專案、公司的產品或服務等。Ontology 則是由主題類型（Topic Types）“Developer” 和 ”Company”；關聯類型（Association Type）“Employed by” ；以及角色 “Employee” 與 “Employer” 所組成。以下說明皆以上方敘述為例。

2. 建立 XTM 文件

下方為最簡單的 XTM 文件。若未設定編碼屬性的話，會以 UTF-8 為預設值。目前這份 XTM 文件尚未添入任何主題。它有參考檔名為 `xtm1.dtd` 的 XTM v1.0 DTD。XTM 文件存檔時副檔名需設為 `.xtm`。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE topicMap SYSTEM “xtm1.dtd”>
<topicMap xmlns="http://www.topicmaps.org/xtm/1.0/"
        xmlns:xlink="http://www.w3.org/1999/xlink">
</topicMap>
```

編碼實例 4-13　宣告 XTM 文件

3. 建立第一個主題

建立該名員工主題（`<topic id="nicole">`）目前該主題只有一個特徵：它的名稱（`<baseName>`)。編碼如下。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE topicMap SYSTEM “xtm1.dtd”>
<topicMap xmlns="http://www.topicmaps.org/xtm/1.0/"
        xmlns:xlink="http://www.w3.org/1999/xlink">
```

```
<topic id="nicole">
  <baseName>
    <baseNameString>Nicole Kidman</baseNameString>
  </baseName>
</topic>

</topicMap>
```

編碼實例 4-14　一個主題的主題地圖

4.　加入主題類型

以<instanceOf>元素，來為主題定義類型。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE topicMap SYSTEM “xtm1.dtd”>
<topicMap xmlns="http://www.topicmaps.org/xtm/1.0/"
        xmlns:xlink="http://www.w3.org/1999/xlink">

<topic id="nicole">
  <instanceOf>
    <topicRef xlink:href="#developer"/>
  </instanceOf>
  <baseName>
    <baseNameString>Nicole Kidman</baseNameString>
  </baseName>
</topic>

</topicMap>
```

編碼實例 4-15　主題類型定義「不完整的」主題地圖

上方的主題類型定義尚未完成，因為它所參照的主題（“developer”）不在這份 Topic Maps 文件中。因此，接下來需要建立一個 “Developer” 主題。另外，再建立公司 “RDLG” 主題，且定義

其主題類型為 ”Company” 。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE topicMap SYSTEM “xtm1.dtd”>
<topicMap xmlns="http://www.topicmaps.org/xtm/1.0/"
        xmlns:xlink="http://www.w3.org/1999/xlink">

<topic id="nicole">
  <instanceOf>
    <topicRef xlink:href="#developer"/>
  </instanceOf>
  <baseName>
    <baseNameString>Nicole Kidman</baseNameString>
  </baseName>
</topic>

<topic id="developer">
  <baseName>
    <baseNameString>Developer</baseNameString>
  </baseName>
</topic>

<topic id="rdlg">
  <instanceOf>
    <topicRef xlink:href="#company"/>
  </instanceOf>
  <baseName>
    <baseNameString>RDLG</baseNameString>
  </baseName>
</topic>

<topic id="company">
  <baseName>
```

```
    <baseNameString>Company</baseNameString>
  </baseName>
</topic>

</topicMap>
```

編碼實例 4-16　主題類型定義「正確的」主題地圖

5. 加入資源指引

透過<occurrence>元素，來為主題加入資源指引。

```
<topic id="rdlg">
  <instanceOf>
    <topicRef xlink:href="#company"/>
  </instanceOf>
  <baseName>
    <baseNameString>RDLG</baseNameString>
  </baseName>
  <occurrence>
    <instanceOf>
      <topicRef xlink:href="#website"/>
    </instanceOf>
    <resourceRef
xlink:href="http://river.glis.ntnu.edu.tw/RDLG/elstandards/"/>
  </occurrence>
</topic>

<topic id="nicole">
  <instanceOf>
    <topicRef xlink:href="#developer"/>
  </instanceOf>
  <baseName>
    <baseNameString>Nicole Kidman</baseNameString>
```

```
  </baseName>
  <occurrence>
    <instanceOf>
      <topicRef xlink:href="#hacker"/>
    </instanceOf>
    <resourceData>Nicole is a cool girl and a great
hacker</resourceData>
  </occurrence>
</topic>

<topic id="website">
  <baseName>
    <baseNameString>Web site</baseNameString>
  </baseName>
</topic>

<topic id="hacker">
  <baseName>
    <baseNameString>Hacker</baseNameString>
  </baseName>
</topic>
```

編碼實例 4-17　有資源指引的主題

RDLG 的資源指引，是指向外部資源（藉由<resourceRef>元素，指出 URL）；而 Nicole 則是指向內部資源（用<resourceData>元素）。此外，也需為資源指引類型，新增 “Website” 與 “Hacker” 這兩個主題。

6. 主題識別

Topic Maps 中有一個相當重要的功能—合併，它可以讓任何與同一主題有關的 Topic Maps 能有機會整合在一起，不過前提是，要能提供資訊讓系統判斷是否和同一主題有關。透過定義 URI（稱為「主題

識別碼」）藉由 URI 解析資源（該資源稱為「主題指標」），讓人們可以指定主題識別之指標。以 “RDLG” 為例，其編碼如下：

```
<topic id="rdlg">
  <instanceOf>
    <topicRef xlink:href="#company"/>
  </instanceOf>
  <subjectIdentity>
    <subjectIndicatorRef
     xlink:href="http://psi.ontopia.net/#organization"/>
  </subjectIdentity>
  <baseName>
    <baseNameString>RDLG</baseNameString>
  </baseName>
</topic>
```

編碼實例 4-18　有「主題識別碼」的主題

「主題識別碼」可以用來表示任何主題（Subject），若該主題表示的主題為資訊資源（例如網頁），那麼該資源的位址亦可作為識別碼，這種情況稱為「主題定位」（Subject Locator)。

```
<topic id="rdlg-website">
  <instanceOf>
    <topicRef xlink:href="#website"/>
  </instanceOf>
  <subjectIdentity>
    <resourceRef
     xlink:href="http://river.glis.ntnu.edu.tw/RDLG/elstanda
rds/"/>
  </subjectIdentity>
  <baseName>
    <baseNameString>RDLG's Web Site</baseNameString>
  </baseName>
```

```
</topic>
```

編碼實例 4-19　有「主題定位」的主題

7. 加入關聯

編輯好主題、資源指引、以及識別碼後，接下來就可以為主題加入關聯了。以下，新增一個關聯，表示員工與雇主的關係。

```
<association>
  <instanceOf>
    <topicRef xlink:href="#employment"/>
  </instanceOf>
  <member>
    <roleSpec><topicRef xlink:href="#employee"/></roleSpec>
    <topicRef xlink:href="#nicole"/>
  </member>
  <member>
    <roleSpec><topicRef xlink:href="#employer"/></roleSpec>
    <topicRef xlink:href="#rdlg"/>
  </member>
</association>
```

編碼實例 4-20　關聯元素

關聯元素參考了三個尚未存在於此 XTM 文件中的主題，分別是：“Employee” 與 “Employer”（關聯角色類型 [Association Role Types]）與 “Employment”（關聯類型）。因此，進一步將這三個主題加入。

```
<topic id="employment">
  <baseName>
    <baseNameString>Employment</baseNameString>
  </baseName>
  <baseName>
    <scope><topicRef xlink:href="#employer"/></scope>
```

```
    <baseNameString>Employs</baseNameString>
  </baseName>
  <baseName>
    <scope><topicRef xlink:href="#employee"/></scope>
    <baseNameString>Employed by</baseNameString>
  </baseName>
</topic>

<topic id="employer">
  <baseName>
    <baseNameString>Employer</baseNameString>
  </baseName>
</topic>

<topic id="employee">
  <baseName>
    <baseNameString>Employee</baseNameString>
  </baseName>
</topic>
```

編碼實例 4-21　關聯類型與關聯角色類型主題

上方編碼實例 4-21，我們指定了三個基本名稱給 “Employment” 主題所定義的關聯類型：名詞 “Employment” 是此關聯類型的預設名稱（因為沒有指定「範圍」的話，就表示其在任何「範圍」都有效）；動詞 “Employs” 與 “Employed by” 則是分別在 “employer” 和 “employee” 的「範圍」內才有效。該步驟並非必要，但這麼做的好處是可以幫助系統根據情境，適切地標示出關聯類型。因此，善用「範圍」<scope>將能有助於處理 Topic Maps 中的關聯欠缺方向的問題。

8. 具體化（Reification）

「具體化」，是指將某種抽象或具體之事物，以某種具體事物將其指涉出來。在 Topic Maps 中，「具體化」是將「關聯」、「資源指引」、

或甚至是「主題地圖本身」，轉化為「主題」的過程。處理此過程，基本上有二個步驟：

(1) 將欲具體化的 Topic Maps 物件，為其元素指定一個 ID。
(2) 建立一個主題，使其「主題識別」具有參考該元素的主題指標。

下方的例子，是替<topicMap>元素加入 ID，建立一個主題，並以參考 Topic Maps 元素為其「主題指標」。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE topicMap SYSTEM “xtm1.dtd”>
<topicMap id="nicolestm"
        xmlns="http://www.topicmaps.org/xtm/1.0/"
        xmlns:xlink="http://www.w3.org/1999/xlink">

<!-- other topics and associations omitted for brevity -->

<topic id="nicolestm-topic">
  <subjectIdentity>
    <subjectIndicatorRef xlink:href="#nicolestm"/>
  </subjectIdentity>
  <baseName>
    <baseNameString>Nicole's First Topic Map</baseNameString>
  </baseName>
</topic>
</topicMap>
```

編碼實例 4-22 具體化主題地圖

若覺得 “Nicole's First Topic Map” 這段文字太長的話，還可以給同一個主題第二個名稱，此時，可以再為第二個名稱指定範圍。

```
<topic id="nicolestm-topic">
  <subjectIdentity>
```

```
    <subjectIndicatorRef xlink:href="#nicolestm"/>
  </subjectIdentity>
  <baseName>
    <baseNameString>Nicole's First Topic Map</baseNameString>
  </baseName>
  <baseName>
    <scope>
      <subjectIndicatorRef
       xlink:href="http://psi.ontopia.net/basename/#short-na
       me"/>
    </scope>
    <baseNameString>Nicole's 1st TM</baseNameString>
  </baseName>
</topic>
```

編碼實例 4-23　給同一個主題第二個名稱

接著，繼續將關聯具體化。產生這種情況的理由可能是，主題地圖作者想要對關聯（其類型與角色）有更多的說明[83]。例如，Nicole 和 RDLG 的關聯中，可能還有一些相關資訊，譬如職稱、契約等。藉由具體化關聯，就可以像「資源指引」一般，獲取和該主題有關的資源。

```
<topic id="nicole-rdlg-topic">
  <instanceOf>
    <topicRef xlink:href="#employment"/>
  </instanceOf>
  <subjectIdentity>
    <subjectIndicatorRef
xlink:href="#nicole-rdlg-association"/>
  </subjectIdentity>
  <baseName>
```

[83] 在 Topic Maps 中，要對任何事物加以說明的唯一途徑便是將它指定爲「主題」。

```
    <baseNameString>Nicole's position in RDLG</baseNameString>
  </baseName>
  <occurrence>
    <instanceOf>
      <topicRef xlink:href="#contract"/>
    </instanceOf>
    <resourceRef

xlink:href="http://river.glis.ntnu.edu.tw/RDLG/internal/emp
loyees/contracts/nicole.htm"/>
  </occurrence>
</topic>
```

編碼實例 4-24　將關聯具體化

從編碼實例 4-24 中可以發現，其實我們尚未指定 ID 為 ”nicole-rdlg-association” 的關聯；此外，也還沒建立 “contract” 這個主題。上述例子的完整 XTM 編碼，請參見附錄一。下圖為本研究將完成之 XTM 文件匯入 Ontopia 公司開發的 Omnigator 軟體後的呈現畫面。由圖可知，本研究建立的 XTM 文件定義了六個主題類型，一個關聯類型，二個角色類型，以及三個資源指引類型。

圖 4-29　Nicole's First Topic Maps 呈現畫面

四、Topic Maps 之相關研究與應用

目前 Topic Maps 的應用，多半是將 Topic Maps 導向的資訊搜尋方式運用在入口網站上，一方面協助建立網站架構，幫助組織網路資源；另一方面則可提供明確、多元的瀏覽模式。另外，Topic Maps 還可應用在內容管理系統中，用來組織內容，但不是像現今常見的詮釋資料或簡單的階層架構，並能藉由 Topic Maps 合併的功能，整合分散於各處的資訊。(Garshol, 2002)本小節將分別介紹 Topic Maps 於工具、資源整理、詮釋資料互通、知識管理系統、以及搜尋引擎之相關研究與應用實例。

（一）工具與查詢語言

1. Topic Maps 編輯器

網路上可下載的 Topic Maps 編輯器包括 Atop[84]、mapalizer[85]、

[84] http://sourceforge.net/projects/atop
[85] http://www.topicmapping.com/mapalizer.html

TMTab[86]、以及 Topic Map Designer[87]等。下圖為 Topic Map Designer 的編輯畫面。

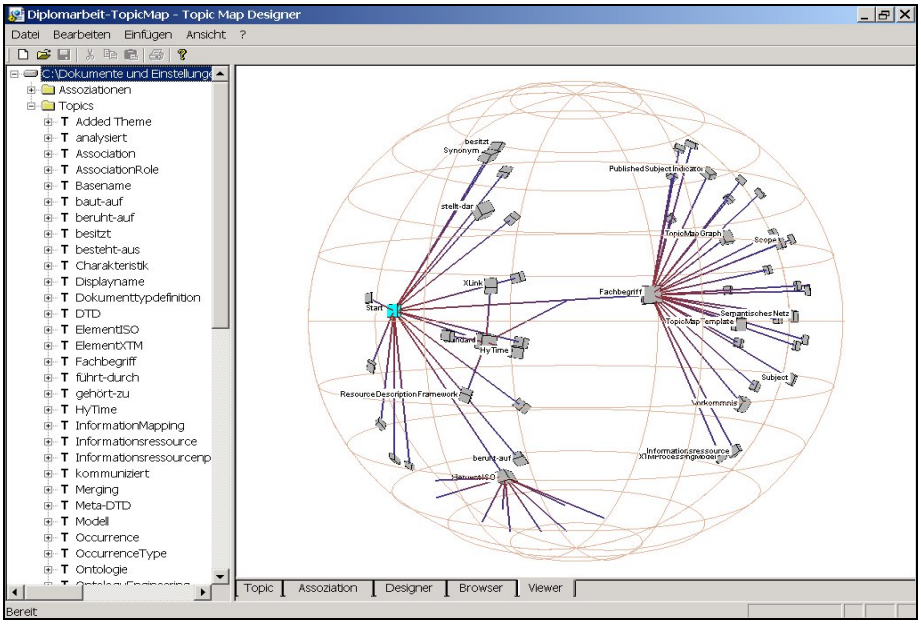

圖 4-30　Topic Maps 編輯器—Topic Map Designer[88]

2. Topic Maps 的呈現

（1）Ontopia Omnigator

Ontopia Omnigator 是 Ontopia 公司[89]根據 ISO 13250 所開發的 Omnigator 軟體，是一個目前廣為應用的 Topic Maps 產品。目前最新版的 Omnigator 甚至採用主題地圖查詢語言（Topic Maps Query Language，簡稱 TMQL[90]）設計其查詢功能。Omnigator 具有上載、輸出、瀏覽、查詢、與管理主題地圖等功能。

---

[86] http://www.techquila.com/tmtab/index.html
[87] http://www.topicmap-design.com/en/topicmap-designer.htm
[88] 資料來源：Topic Map Designer。上網日期：民 94 年 5 月 15 日。上網：http://www.topicmap-design.com/en/topicmap-designer.htm
[89] Ontopia 網址：http://www.ontopia.net/。Omnigator 試用版下載：http://www.ontopia.net/download/freedownload.html
[90] Garshol, L.M. & Barta, R.(Eds.).(2005). ISO 18048: Topic Maps Query Language. Retrieved May 15, 2005, from http://www.isotopicmaps.org/tmql/spec.html

下圖是以 Steve Pepper 所建立的義大利歌劇 XTM 為例，載入 Omnigator 後，以 “Puccini, Giacomo” 為主題的顯示畫面，此時頁面會顯示該主題之名稱、關聯、內部資源、與外部資源。

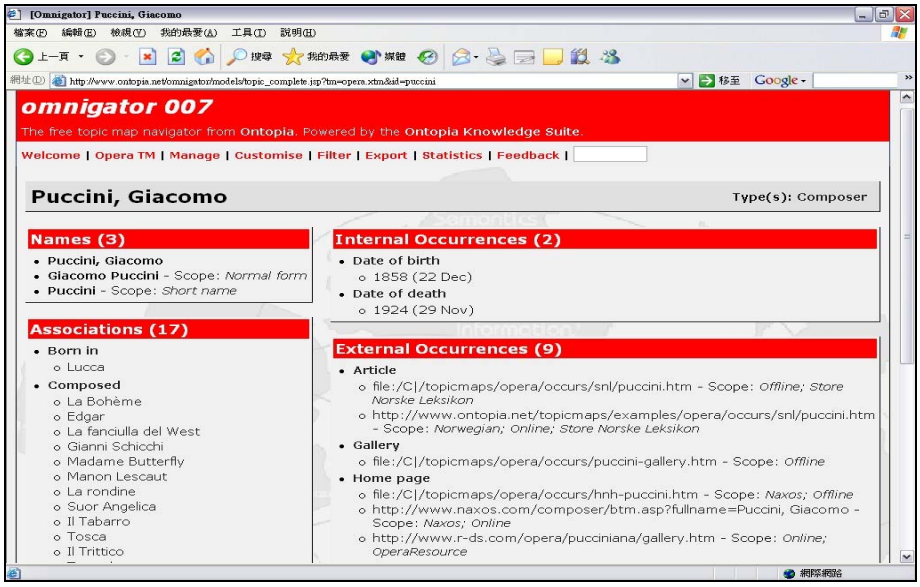

圖 4-31　以 “Puccini, Giacomo” 為主題的 Topic Maps[91]

另外還有 TMNav[92]，以及可將 Topic Maps 以 3D 方式呈現 HyperGrpah[93]、TM3D[94]、與 TM View[95]。下圖為 TM3D 的呈現畫面。

[91] 資料來源：The Italian Opera Topic Map。上網日期：民 93 年 8 月 6 日。網址：http://www.ontopia.net/omnigator/models/topic_complete.jsp?tm=opera.xtm&id=puccini
[92] http://tm4j.org/tmnav.html
[93] http://hypergraph.sourceforge.net/
[94] http://silpion.dyndns.org/rubrique.php3?id_rubrique=1
[95] http://www.spatialknowledge.com/projects/topicmaps/mirror/browser.html

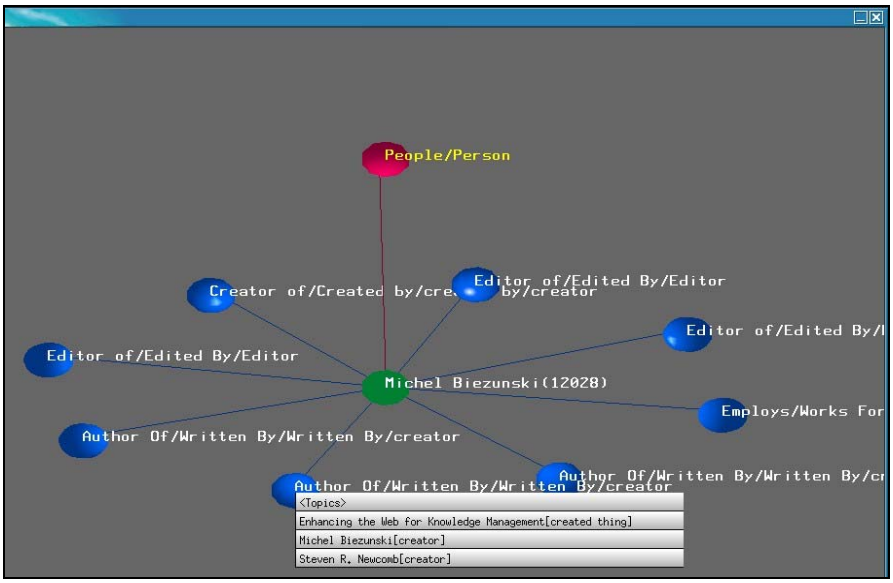

圖 4-32　TM3D—Topic Maps 3D 呈現軟體[96]

3.　Topic Maps 查詢語言（Query Language）

目前已有不少公司或個人提出可查詢 Topic Maps 資料的查詢語言，其中最著名的即為 2005 年 2 月剛公布的 TMQL，另外還包括 AsTMa[97]、TMPath[98]、TMRQL[99]、tolog 1.0[100]、以及 Toma[101]等。下圖為 AsTMa 的線上展示版。

[96] 資料來源：TM3D。上網日期：民 93 年 8 月 6 日。網址：http://silpion.dyndns.org/article.php3?id_article=5

[97] Barta, R.(2003). AsTMa. Retrieved May 15, 2005, from http://astma.it.bond.edu.au/astma%3F-spec.dbk

[98] Bogachev, D.(2003). TMPath. Retrieved May 15, 2005, from http://homepage.mac.com/dmitryv/TopicMaps/TMPath/TMPathIntroduction.html

[99] Moore, G. & Ahmed, K.(2005). Topic Map Relational Query Language(TMRQL). Retrieved May 15, 2005, from http://www.networkedplanet.com/download/TMRQL.pdf

[100] Garshol, L.M.(2003). tolog 1.0. Retrieved May 15, 2005, from http://www.ontopia.net/topicmaps/materials/tolog-spec.html

[101] Pinchuk, R.(2004). Toma. Retrieved May 15, 2005, from http://www.spaceapplications.com/toma/Toma.html

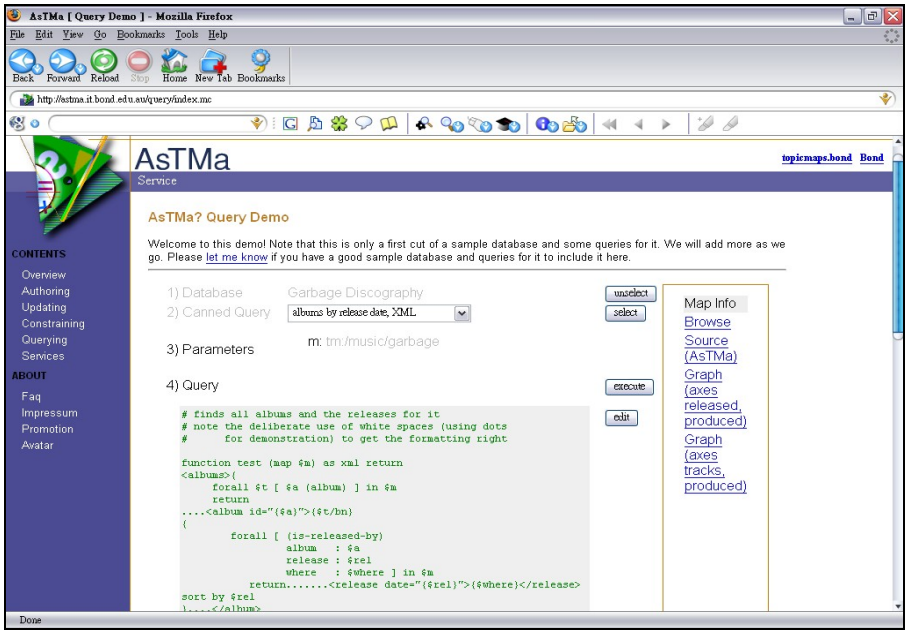

圖 4-33　AsTMa 的線上展示版[102]

Garshol and Barta(2003)曾指出幾個 TMQL 的可能應用情境包括知識內容為主的網站（Knowledge Web Sites）、入口網站、內容管理、企業資訊整合、高整合度系統（High-integrity Systems，Topic Maps 可以協助建立文件索引）、聯合詮釋資料處理（Metadata Syndication）、知識管理（Topic Maps 可以協助檢索問題的推演）、以及無形資產管理（例如查詢商業夥伴間的關聯）等。

（二）資源整理

利用 Topic Maps 作為網站資訊架構的實例有許多，包括 Cogito Ergo XML[103]、LmTM[104]、techquila[105]、XMLAD Topic Map[106]、以及

[102] 資料來源：AsTMa Query Demo。上網日期：民 94 年 5 月 15 日。網址：http://astma.it.bond.edu.au/query/
[103] http://www.cogx.com/
[104] http://www.lmtm.de/
[105] http://www.techquila.com/topicmaps/tmworld/
[106] http://www.augmentingminds.com/xmlad-tm/index.html

XML and Web Service glossary[107]等，以下另外再介紹 The Semantopic Map、PhotoGraph、與紐西蘭電子文件中心等，分別以 Topic Maps 整理網路資源、照片、以及書籍資料為主的網站實例。

1. The Semantopic Map—語意主題地圖計畫

語意主題地圖計畫（The Semantopic Map[108]）是由 Bernard Vatant 主持，計畫執行於 2001 年至 2002 年，使用 Mondeca 公司[109]開發的智慧型主題管理（Intelligent Topic Manager）軟體來建置以 Topic Maps 為資訊架構的網站[110]。網站將資源區分為工具、標準、組織、人物、事件、與計畫等六大類；裡頭每個主題皆標示出其實例、相關主題、狹義詞、相關詞，主題彼此間亦運用 Topic Maps 的概念關聯起來。如圖 4-34 所示。

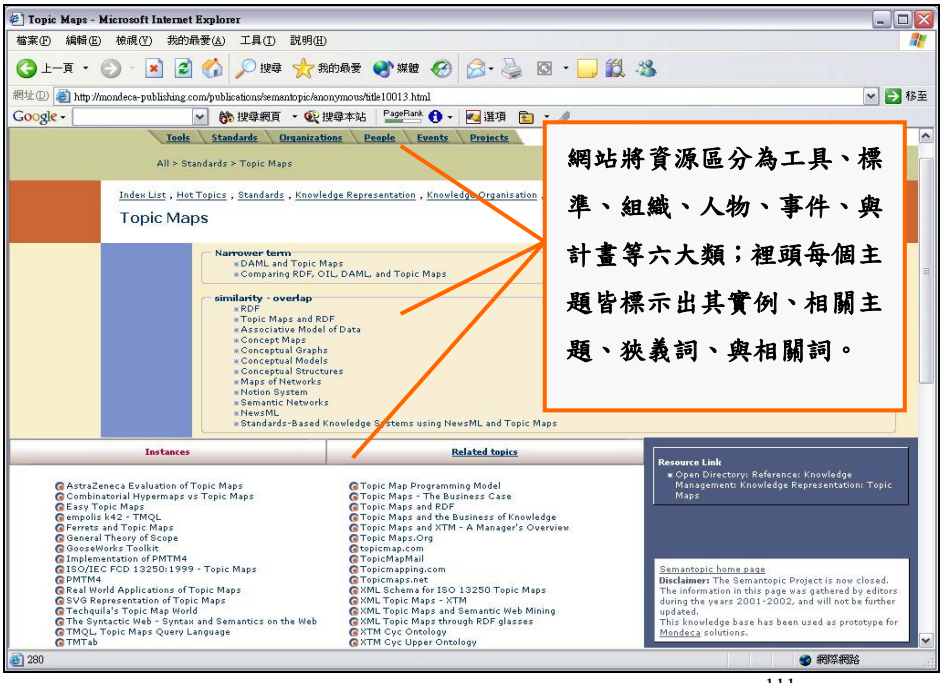

圖 4-34　主題為 Topic Maps 的主題瀏覽頁面[111]

107 http://dret.net/glossary/
108 http://perso.wanadoo.fr/universimmedia/semantopic.htm
109 http://www.mondeca.com/english/
110 http://mondeca-publishing.com/publications/semantopic/anonymous/title49.html
111 資料來源：The Semantopic Map。上網日期：民 93 年 8 月 6 日。網址：

圖 4-35 是點選了圖 4-34 中的實例—TopicMaps.Org 後所顯示的頁面。除了提供有關 TopicMaps.Org 簡要說明外，下方還顯示了事件、主題、協同者、贊助者、主持人等關連（每個主題所顯示的關聯會有所不同），每個關聯的主題都可彼此連結。畫面右方則是 TopicMaps.Org 的資源指引，讓使用者可以直接連到 TopicMaps.Org 的官方網站。

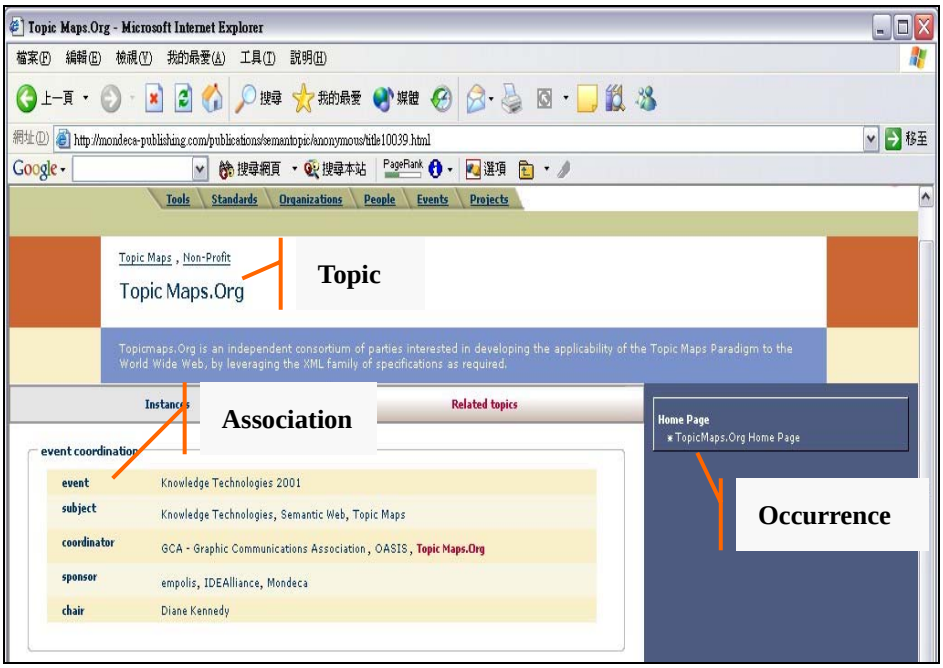

圖 4-35　主題為 TopicMaps.Org 的主題關聯頁面[112]

2. PhotoGraph—照片整理網站

PhotoGraph[113]照片整理網站是由 Stefan Lischke 應用主題地圖內容管理系統技術（TopicMap Content Management System，簡稱 XTM-CMS[114]）開發。網站利用 Topic Maps 的概念來整理所蒐集的照片，並提供使用者以主題、事件、人物、動作、物件、與位置等六大面向進行瀏覽，如下圖所示。點選某一主題後，畫面上方會顯示此主

http://mondeca-publishing.com/publications/semantopic/anonymous/title10013.html
[112] 資料來源：The Semantopic Map。上網日期：民 93 年 8 月 6 日。網址：http://mondeca-publishing.com/publications/semantopic/anonymous/title10039.html
[113] http://crewman-3.ivs.tu-berlin.de/PhotoGraph/index.html
[114] http://www.ivs.tu-berlin.de/XTMCMS/index_en.html

題與其他主題的語意關聯圖，下方則會顯示該主題的照片，及照片提供者對該照片的描述，例如照片的內容與事件等資訊。

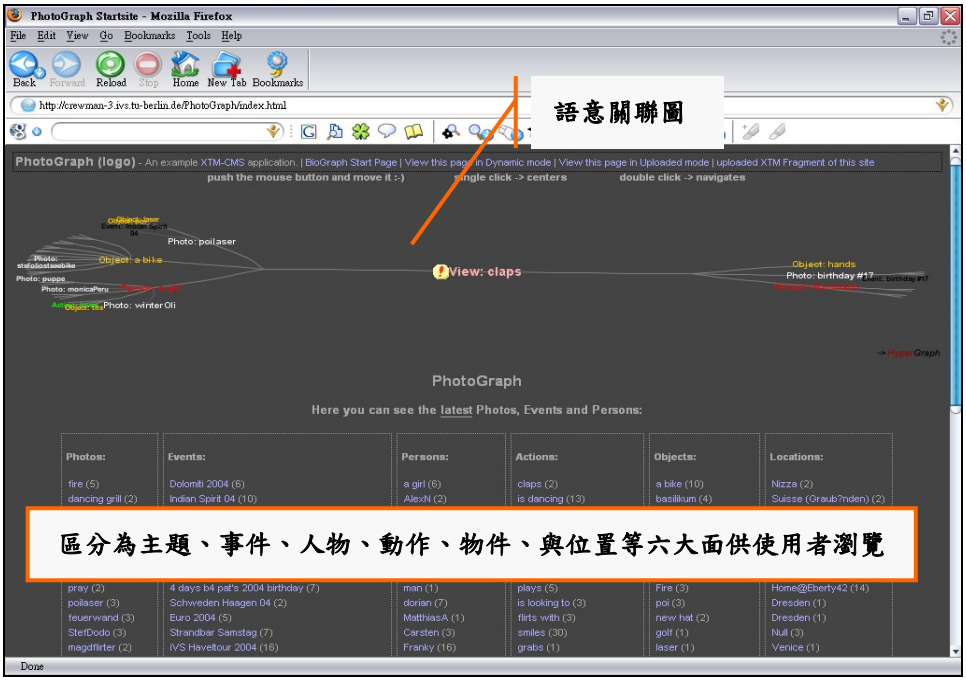

圖 4-36　PhotoGraph 照片整理網站[115]

3.　紐西蘭電子文件中心網站

紐西蘭電子文件中心（New Zealand Electronic Text Centre，以下簡稱 NZETC[116]）於 2001 年 12 月由紐西蘭威靈頓維多利亞大學（Victoria University of Wellington）設立，網站徵集的內容包含以紐西蘭及太平洋導為主的書籍全文、圖像、手稿、與期刊，文件依文件編碼格式（Text Encoding Initiative，簡稱 TEI[117]）標記。如圖 4-37。

NZETC 於 2005 年 5 月導入 Topic Maps 作為網站資訊架構，利用

[115] 資料來源：PhotoGraph。上網日期：民 94 年 5 月 13 日。網址：http://crewman-3.ivs.tu-berlin.de/PhotoGraph/index.html
[116] http://www.nzetc.org/
[117] http://www.tei-c.org/

XSLT stylesheets 將 XML 詮釋資料檔轉成 XTM，產生上百個 Topic Maps 文件，同時由 MADS[118]權威檔（Authority File）擷取所徵集之資源中提及的相關人物、地點、與組織等資訊，最後將收集來的 Topic Maps 文件利用開放原始碼程式 TM4J[119]進行合併，產生一個可以導覽整個網站的 Topic Map。NZETC Topic Maps 的「主題」為書、章、圖解、以及在書中提及的人物與地點，「關聯」則包括提及、描述等，每一網頁都會顯示主題及其關聯，使用者可依感興趣的面向來瀏覽資源，或是直接對資源進行全文檢索。

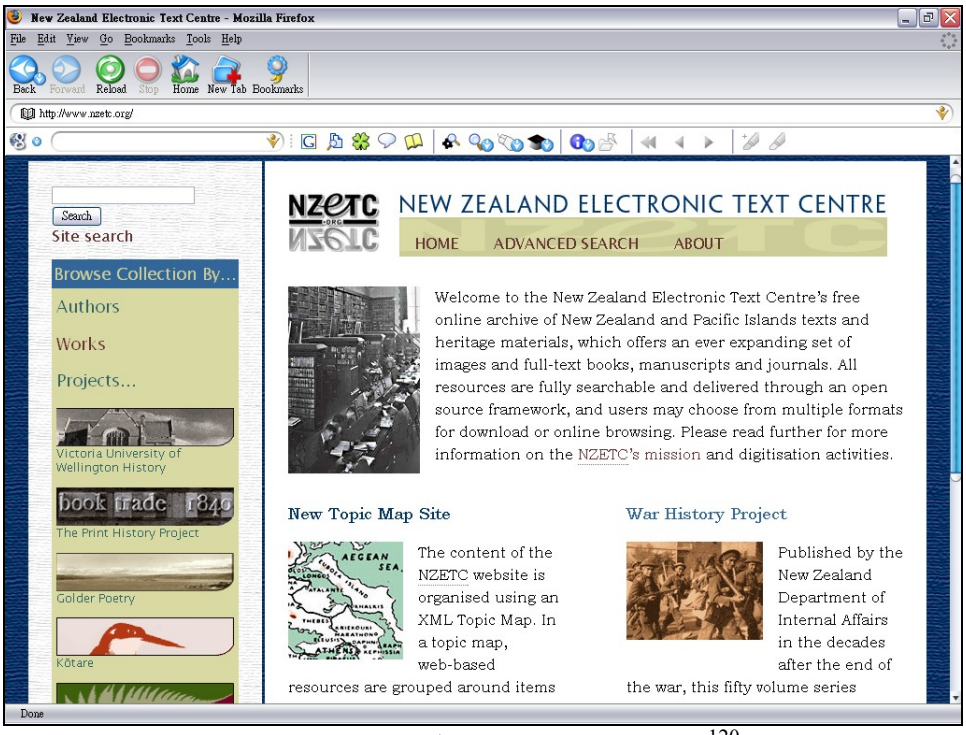

圖 4-37　紐西蘭電子文件中心網站[120]

透過上述實例不難發現，瀏覽 Topic Maps，一方面除了可以查找資料外，還可以幫助使用者建立該主題的知識。此外，由於 Topic Maps

[118] MADS 全稱 Metadata Authority Description Schema，是由美國國會圖書館網路發展與機讀編目格式標準辦公室所發展的權威格式的 XML Schema。網址：http://www.loc.gov/standards/mads/

[119] http://tm4j.org/tm4j-engine.html

[120] 資料來源：New Zealand Electronic Text Centre。上網日期：民 94 年 5 月 14 日。網址：http://www.nzetc.org/

通常包含許多交疊，且具有豐富的語意關係之連結（例如 A 是 B 的父親；C 和 D 是員工與雇主關係），因此，可以幫助資訊查詢，因為使用者不再需要按設計人員的規劃來查找資訊，而是可依循豐富的瀏覽路徑來找所需之資訊。當然，使用者亦可透過搜尋跳到某處，然後再由該處開始瀏覽。

（三）詮釋資料互通

de Graauw(2002)有感於 B2B 的資料交換存在許多資料交換標準，例如 ANSI X.12、ebXML、EDIFACT、FinXML、FpML、以及 xCBL 等，不同商業夥伴可能使用不同的詞彙，因此這些資料標準所記錄的資料之互通是亟需解決的問題。不過目前的解決之道多半是建立能涵括資料的新詞彙、對概念採用廣為人知的識別碼（例如身分證字號、ISO 國碼、與郵遞區號等）、或是詮釋資料對應（Mapping）等，但以上作法基本上仍都面臨資訊遺失等問題。

作者認為需要發展的是可攜式（Portable）、具複用性、且標準化的對應規則，於是作者應用 Topic Maps 提出 “Business Maps” ，如表 4-8。只要採用合適的 PSI，那麼便能藉由 Topic Maps 的合併機制整合不同的 Topic Maps 資源，而使用者也可再利用「範圍」來過濾出所需之資訊。

表 4-8 “Business Maps” 對應模型

對應模型（Mapping Model）	**實例**	**主題地圖元素（Topic Map Construct）**
文件（Document）	發票	Topic
項目（Item）	顧客姓名	Topic
文件對文件對應（Document-to-document Mapping）	發票對應到帳單	Association
項目對項目對應（Item-to-item Mapping）	顧客姓名對應到公司名	Association

情境（Context）	詞彙、公司、行業、地區	Scope
外部文件描述（External Document Description）	www.bizwords.org\invoice	Occurrence (role: business document description)
外部項目定義（External Item Definition）	www.bizwords.org\amount	Occurrence (role: definition)
外部項目資料類型（External Item Datatype）	www.bizwords.org\date	Occurrence (role: datatype)
外部項目實例（External Item Example）	www.bizwords.org\amount\example	Occurrence (role: example)
詞彙識別碼、文件識別碼、或項目識別碼（Vocabulary Identifier + Document Identifier or Item Identifier）	www.bizwords.org\amount	Subject Identity

資料來源：de Graauw, M.(2002). Business Maps: The interoperability Topic Map. Retrieved May 15, 2005, from http://www.marcdegraauw.com/itm/

（四）個人資訊管理與知識管理系統

1. Topic Map Explorer—個人資訊管理工具

Topic Map Explorer[121]是由 Semantic Web Technologies[122]開發的個人資訊管理軟體，如下圖所示。使用者可以自行定義分類架構，將個人電腦內的文件檔案、電子郵件、與網頁連結等依主題進行分類，主題間亦可建立關聯，以協助使用者由交互關聯的主題來瀏覽資源；此外，在使用者進行點對點（Peer-to-peer）設定配置（Configure）後，還可與他人分享自己建的主題與資源，或搜尋與獲取遠端正在線上的使用的所分享的主題與文件。

[121] Topic Map Explorer 試用版下載：http://www.tmexplorer.com/
[122] http://82.112.113.35/

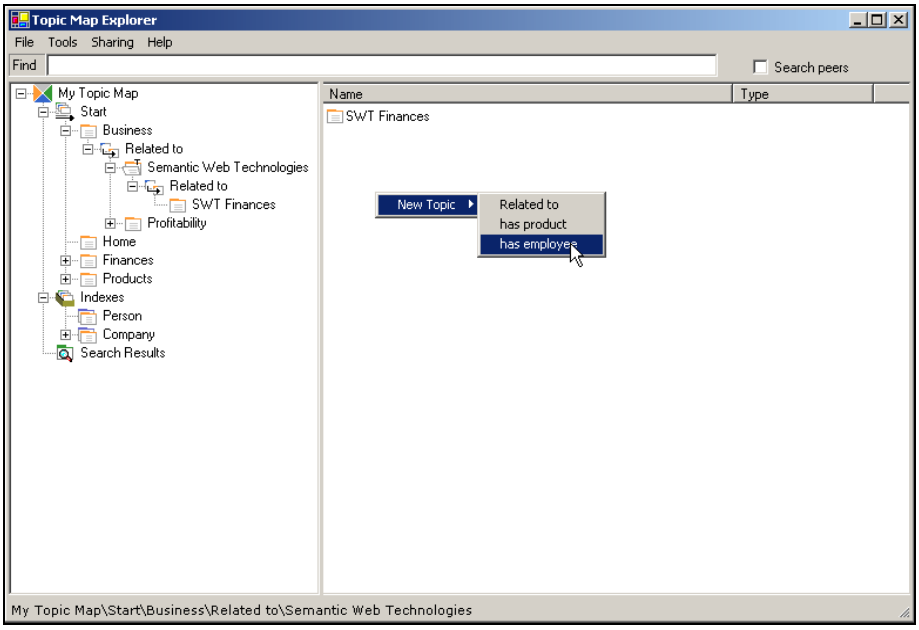

圖 4-38　Topic Map Explorer 個人資訊管理工具[123]

2.　佛教知識庫—知識管理系統之應用

資訊的變動性及資源維護是 Topic Maps 必須面對的問題，為能提高 Topic Maps 維護的效率，許多系統在開發時，為能以較有彈性的方式來方便 Topic Maps 的管理，多會結合資料庫並配合 Topic Maps 的觀念來應用。國內學者林光龍、歐陽彥正（民 91）即曾將 Topic Maps 標準和規範應用於佛教知識庫的建立。透過把 XTM 1.0 DTD 轉成實體關係圖，再以實體關係圖設計關聯式資料庫綱要，儲存 Topic Maps 之資訊，並滿足使用者易於新增與異動主題等維護作業之需求。

研究者將資料庫所記載的主題分成人、事、時、地、物五個子系統，在這五個子系統中又分別提供各主題「知識本體管理子系統」與「實體管理子系統」。前者負責管理類別名稱、類別階層、類別關係、屬性名稱、以及類別屬性；後者則負責有關實體之人物、資訊資源、與人際關係之維護。下圖是以「人物實體管理子系統」為例，顯示其如何維護文獻所記載之人物及其所屬類別與屬性。例如，玄奘法師其

[123] 資料來源：Topic Map Explorer–Quick Tour。上網日期：民 94 年 5 月 13 日。網址：http://www.tmexplorer.com/images/ug/create_topic_with_new_assoc_type.gif

類別有僧侶、徒弟、師長。

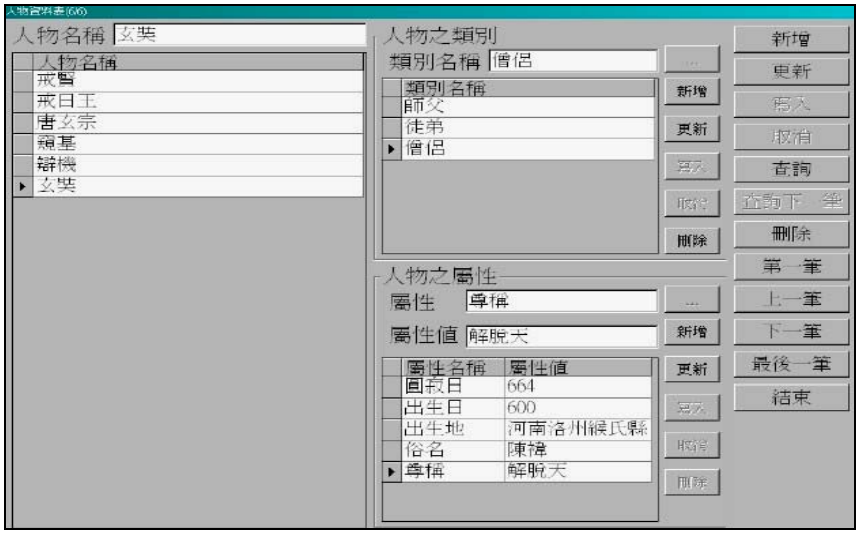

圖 4-39　人物實體管理主畫面

資料來源：林光龍、歐陽彥正（民 91）。佛教知識庫的建立─以 Topic Map 建置玄奘西域行為例。佛教圖書館館訊，32 期，頁 41-54。

（五）搜尋引擎

KartOO[124]是由法國的 KartOO 公司所開發的整合搜尋引擎（Metacrawler），畫面如圖 4-40 所示。雖然該系統並未利用 XTM 來儲存資料，但在介面呈現上則展現了 Topic Maps 的精神。網頁文件與文件間皆建立了語意關聯，系統會分析使用者鍵入的詞彙並以每頁提供 12 至 15 個網頁文件圖示的方式顯示，使用者點選文件圖示就會開啟所代表的網頁，或是點選文件間的關聯進一步搜尋，此外，畫面左下方還會依統計數據，提供其他熱門檢索主題供使用者參考。

[124] http://www.kartoo.com/

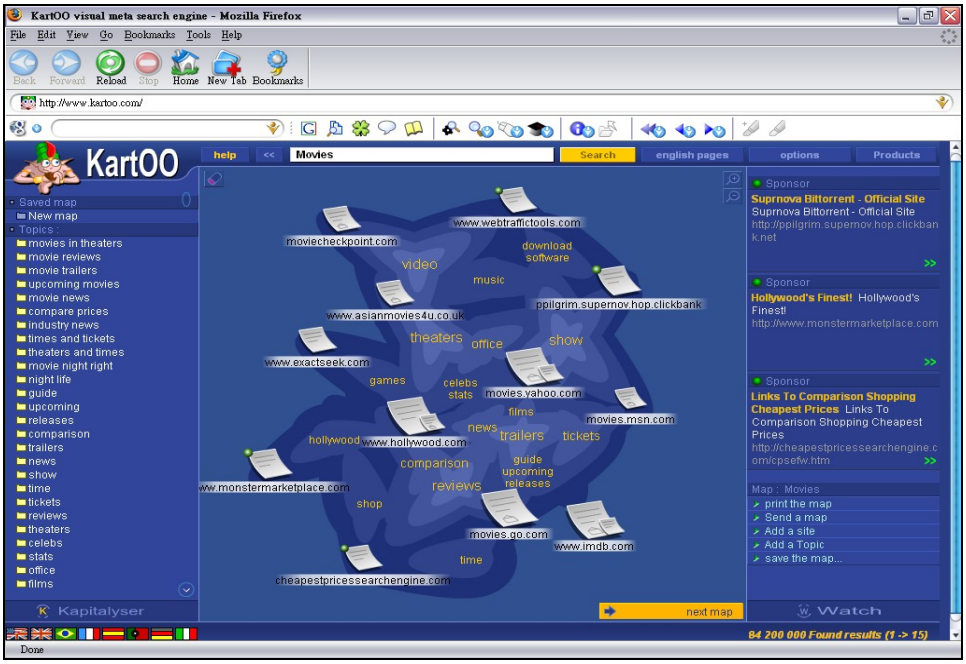

圖 4-40　KartOO 搜尋引擎[125]

五、小結

綜上所述，Topic Maps 除了可以作為知識表徵的方法外，還具有主題瀏覽與交換資訊之功能。不過，要能將 Topic Maps 的功能發揮極致，最根本的仍是要瞭解知識領域，並投入主題分析工作；換句話說，Topic Maps 最重要的是要能夠定義出 Ontology，找出資源的主題及主題間之關聯，並且進一步連結資源，提供有關資源的不同觀點，建立一個妥善的 Ontology，進而協助知識的分享與交換。

[125] 資料來源：KartOO。上網日期：民 94 年 5 月 13 日。網址：http://www.kartoo.com/

第三節 小結

RDF 與 Topic Maps 的發展背景並不相同，它們有不同的根源與觀點，RDF 則是根源於形式邏輯（Formal Logic）與數學圖形理論（Mathematical Graph Theory）；Topic Maps 源自於書後索引、索引典、及詞彙表等檢索輔助工具。由於出發點的不同，因此有學者認為 RDF 與 Topic Maps 分別是由機器及人的觀點，將知識表徵應用於資訊管理，RDF 是「以資源為中心」（Resource-centric），Topic Maps 則是「以主題為中心」（Subject-centric）。（Pepper, 2002）

RDF 與 Topic Maps 皆定義了抽象資料模型，並以 XML 作為交換語法，雖然表達 RDF 與 Topic Maps 並非一定要用 XML 語法，但 XML 是最廣為使用的語法。在 RDF 中，所有以 RDF 描述的事物皆稱為「資源」，資源的「屬性」本身可以也是資源，因此屬性可能也可以有屬性；在 Topic Maps 中，多數的事物都是「主題」，因為所定義的主題類型、關聯類型、以及資源指引類型亦皆可視為主題。總而言之，在 Topic Maps 中，要對任何事物加以描述的唯一途徑便是將它指定為主題。

在 RDF 中，敘述是有方向的，必須由主詞指向受詞，因此「我騎腳踏車」的敘述，不同於「腳踏車被我騎」，所以在 RDF 中，這種意義相同的不同敘述方式必須另外建立。不過在 Topic Maps 中，藉由主題在關聯中扮演的「角色」，宣告關聯時同時定義在關聯中的成員所扮演的角色，便可清楚表達「我騎腳踏車」及「腳踏車被我騎」這兩個敘述。此外，在 Topic Maps 中，關聯可以是 N 元的，關聯的「角色」可以使它超越二元關係，因為關聯可以包含任意數目的角色，因此也可以表示較複雜的關係。

再者，Topic Maps 透過「範圍」的概念，解決補捉情境正當性（Contextual Validity）的問題。在主題的名稱、資源指引、或關聯的角色被視為正當的情境下，人們便可藉由將相關的敘述納入某一「範圍」，來表示它們的情境。基於上述原因，Pepper(2002)認為和 Topic

Maps 相比，RDF 所能表達的是較低階的語意。RDF 中的資源有其屬性，屬性有屬性值（屬性值也可能資源），僅此而已。在 Topic Maps 中，主題有其特徵，包括名稱、資源指引、以及主題在關聯中所扮演的角色，而這些特徵在 RDF 中都未加以規範。

下一章本研究將進一步由規範及以實際的應用情境進行敘述，來比較 RDF 與 Topic Maps 在元素、語法、及語意表達之異同，並依上網蒐集之實作網站與系統，觀察比較兩者之應用情境。