東吳大學商學院
資訊管理學系碩士論文

指導教授：連志誠 博士

以實例與綱要為基礎的綱要整合技術之研究

Use Instance-Based And Schema-Based for Integration XML Schema

研究生：尹中文 撰

中華民國一百零三年一月

以實例與綱要為基礎的綱要整合技術之研究
Use Instance-Based And Schema-Based for Integration XML Schema

研究生：尹中文　　Student：Chung-Wen Yin
指導教授：連志誠　博士　　Advisor：Chih-Cheng Lien

東吳大學商學院
資訊管理學系
碩士論文

A Thesis

Submitted to Department of Computer Science and Information Management

School of Business

Soochow University

January 2014

Taipei, Taiwan, Republic of China

中華民國一零三年一月

誌謝

能順利的取得學位，要感謝指導教授連志誠博士悉心的指引研究方向。撰寫論文時，也給予了寶貴的智慧及經驗，使我在研究過程中獲益匪淺。在此致上誠心的感謝之意。

在研究的過程中，得到林聰武教授在研究計畫審查時，給予無私及寶貴的意見。更感謝口試委員呂芳懌教授與段裘慶教授提供論文研究方向的修正意見，使學生論文得以補足不完整之處。

此外，感謝各位同學們給予的幫助和鼓勵，因為你們的陪伴在這幾年學習的的研究生活變得更加充實，在此也敬上無比的感謝，並將此份友誼銘記於心。

最後，感謝家人無微不至的關懷與支持及默默支持更是我前進的動力，讓我能更加專注於學術上的研究。沒有家人的體諒、包容及激勵，相信也沒有今天的這一刻，滿滿的感謝之情無法言表，謹以此文獻給我最親愛的妻子。

中文摘要

近年來在實務上，利用 XML 技術進行異質性資料整合是一個相當重要的議題，目前大多數的綱要整合機制，著重於利用 XML 綱要資訊找出可整合的節點。而這樣的方式是無法有效解決因整合而產生結構異質性以及因參考資訊不完整而影響整合結果的問題。

其它的整合機制，則著重於 XML 文件中的實例資訊，雖然可以透過良好的資料配對方法，而得到較高的精確率，但會因資料的重置性過高，而無法得到較高水準的完整率。因此本研究以 XML 文件中的綱要資訊以及實例資訊進行綱要整合之研究。為了提昇整合時的完整率與精確率，本文分別利用 XML 綱要以及 XML 實例的特性，將各別求得的節點相似度予以合併。經由此合併機制，增加綱要整合的精確性率及完整率。本文並將此機制應用於實務層面，藉以實證評估與分析探討方法之可行性及實用性。

關鍵詞 :XML、XML 綱要、XML 實例、綱要整合

Abstract

In recent years, in practice, the use of XML technology for heterogeneous data integration is a very important issue. Most of the current framework integrating mechanism, focusing on the use of XML schema information to identify nodes that can be integrated. However, this is not an effective way to solve the heterogeneous problems arose from integration and the influence of the lack of reference information.

The other integration mechanisms, is focused on the XML document instance information. Although it is possible to obtain high precision through great data matching method, in case of the high duplication, we can not get a higher level of recall. Therefore, this research is focused on XML schema and XML instance for schema integration. In order to enhance the integration of the recall and precision, the research utilizing the characteristics of XML schema and XML instance to merge similarity of each individual nodes. Through this merger mechanisms to increase the recall and precision. This research using the mechanism in practice to prove the feasibility and the practicability.

Keywords: XML、XML schema、XML instance、Schema integration.

目錄

圖形目錄

1. 研究動機與目的

由於企業內有不同的部門及專業領域，會需要不同的資訊系統滿足其特殊的需求，也隨著經濟環境越趨複雜，企業內部的組成也愈來愈多樣化，資訊系統專業分工的細化程度也就隨之提高。美國資訊管理系統專家 Richard. L. Nolan 觀察 200 多家公司與部門發展資訊系統的過程和經驗 [29]，提出了資訊系統進化的 Nolan 模型 (圖 : 1.1)。依據學者 Nolan 的成長階段理論可分為六階段：初創期、擴張期、控制期、整合期、資料管理時期、成熟期。前 4 個階段較偏重資訊系統的規劃與控制，直至資料管理時期，開始重視資料的共享、管理並開始整合各部門的資訊系統；由此可知企業發展愈趨成熟，其內部的資料整合也就愈加重要。

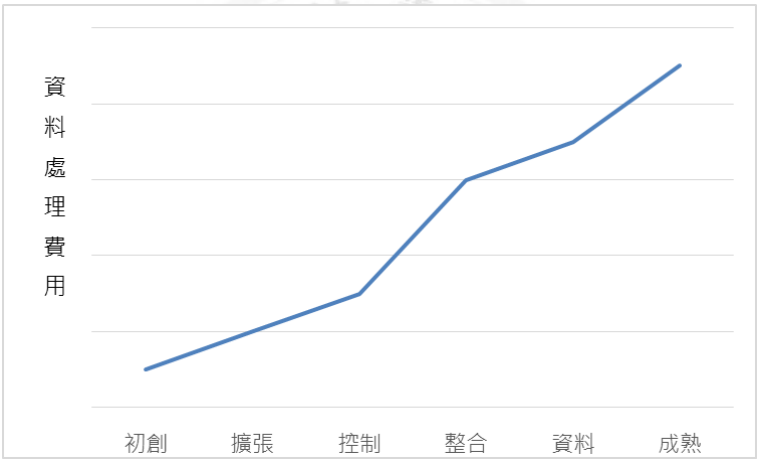


圖 1.1: Norlan 資訊系統成長理論

隨著企業持續的發展，資訊系統要能夠不斷給予充份的支援，而資訊系統在運作的過程中會產生出相當龐大的資料量，其資料必然有相當高的關聯性。然而企業內的資訊系統若沒有整合，則無法有效的提供跨部門、跨系統、跨組織的綜合性信息，例如：客服人員想透過顧客關係管理 (Customer relationship management, CRM) 系統得知某張訂單生產的狀況如何？市場行銷人員想知道最有價值的客戶是誰？專案經理如何即時的得知哪些專案已經延遲了？所以企業組織為了作出正確的決策，常需要整合組織內、外部的各類資料，經過全面的透視和控制後，進而強化企業價值鏈。

企業內各系統的資料進行整合時，會因為其應用的專業領域、系統建置時的

背景、所需遵循的法律規範等因素不同，資料結構上的設計亦有相當的迥異；當來源不同的資料進行整合時，常面臨了資料異質性 (Heterogeneity) 的問題，在整合過程中需要透過中介軟體 (Middleware) 進行資料交換。IDC (International Data Corporation) 於 2010 年的研究報告中指出 [1] 台灣的雲端運算將進入服務元年，各資訊廠商於 2010 年先後推出相關服務，包括軟體即服務（SaaS）、平台即服務（PaaS）及基礎架構即服務（IaaS）；其在運算各種不同來源與格式的資料時，常利用延伸標記語言 (eXtensible Markup Language, XML) 的技術，產出大量的 XML 文件，作為服務間資訊交換的媒介。隨著資訊整合的系統類型及範圍愈來愈複雜時，其所耗費的成本也逐漸增高，所以如何正確且有效率的完成資料交換，將會是整合異質資料中相當重要的一環。

1.1 研究動機

隨著電腦科技的演進，加上網際網路的盛行及雲端運算的技術發展，已經改變了資訊傳遞的方式和速度。各種資訊透過網際網路進行快速的傳遞及交換時，將產生大量且多樣性的電子文件，而這些電子文件目前仍在持續的成長中。這類的電子文件多數是以非結構化的資料格式在傳遞著，對於來源繁多的信息資料，專業人士可以根據信息的格式加以分類及辨識，但當面對的是數以萬計的電子文件時，透過電腦系統來處理異質資訊的整合，將是更好的選擇。

企業內部系統與外部收集而來的異質資料，常需為了滿足決策支援管理的需求，而將這些不同來源的異質資料整合至同一儲存體中。在過程中常面臨如何更有效的管理、整合、分析及擷取各種不同且大量資料的問題。目前電腦科技的進步、網際網路的盛行以及雲端技術的興起，進行資料傳遞時，大多使用 XML 標記語言作為溝通的基礎文件，導致現今企業組織除了處理部門間的資料整合，還必須面對整合各種不同來源的資料及大量不同格式的外部文件，其中又以外部來源資料的整合、交換或檢索更為複雜。

以資料整合而言，異質性最廣為討論的是語法 (syntax)、綱要 (schema) 及語義 (semantic) [6] 等議題；語法異質性是由資料模式與系統開發的語言所造成的歧異性，綱要異質性為資料結構面向的歧異性，而語意異質性則是不同資訊源用相同的詞彙表達不同的訊息或是用不同的詞彙表達相同的訊息。在解決這些異質性的問題或許可以要求各資料來源運用一致性的規範來建構系統。但這些一致性的規範，通常是由各別的系統開發人員依自身所了解的領域知識而整理的規則，不但花費大量人力而且更會因系統應用面向的不同，更加深了資料的

異質性，反而造成許多資料無法透過一致性的規範進行映對。並且在實務上，要求已上線的資訊系統進行調整，勢必需要重起專案並投入更多的人力與費用，若未來有任何法律增修或為因應市場變化而產生的需求變更，各系統恐怕將面臨全面調整的窘境。

資訊系統大多將各類資料儲存至關聯式資料庫中，用以輔助企業日常運作。過去綱要整合的研究，大多偏向為關聯式的資料模式。自 1998 年 W3C（全球資訊網聯盟）發佈 XML 1.0 後，因使用者可在嚴謹的使用規則下，簡單地描述出資料內容，並且在使用上具有跨平台、延伸性高的特性，近年來，XML 已成為應用最廣的資料交換格式標準。由於 XML 文件本身僅在描述資訊訊息，需另透過 XML 綱要驗證文件是否”符合”規範，雖然可利用 XML 綱要的定義來解決大部份語法異質性的問題，但 XML 綱要本身也屬於 XML 文件的一種，所以不同的 XML 綱要，也同時存在綱要及語意的異質性。XML 本身具有高彈性的設計概念，在設計同類 XML 文件時，可能因為應用的面向不同而創造不同的 XML 結構。這些大量異質性的 XML 文件，更加重了 XML 文件進行資料整合時的困難度。有鑑於此引發我們探究如何有效利用 XML 實例與綱要資訊，對不同來源的 XML 文件進行解析以達到資料整合的目的。

1.2 研究目的

企業內部經常利用網際網路進行電子文件的交換，但由於文件格式並無一定的規格與標準，大多透過領域專家進行文件的解析。但若由人力處理大量異質性的電子文件，則需考量其整理分析、歸納分類的成本、粹取的資料品質以及檢索的效率是否符合效益。所以如何利用電腦系統的整合技術，進行資料的分類、檢索與整合，是目前相當熱門且重要的研究課題。

XML 綱要整合就是依據來源 XML 文件與目的 XML 文件的資訊加以映對並決定其綱要整合規則。目前的研究大多以綱要階層 (Schema-level) 或實例階層 (Instance-level) 的資訊，作為綱要映對及整合的依據，但忽略了兩方法互補的可能性。若同時利用綱要階層資訊及實例階層資訊進行映射時，可提高其整合的準確率 [32] [33]，那把其方法應用在資訊整合上，應該可以比使用綱要階層或實例階層單一資料的整合方法，得到更好的成果。

本研究主要目的係將 XML 綱要整合問題，以 XML 文件解析出的綱要資訊與實例資訊分別進行節點整合，再予以整合二者的整合規則。XML 綱要中的節點概念名稱，引用領域知識本體增進語意的辨識，以提昇終端節點整合的效

果。XML 實例中的相同記錄配對方法，則是利用資訊檢索與資料探勘常使用的 TF-IDF (term frequency – inverse document frequency) 進行權重分析，找出高相似度的實例資訊，以此加強屬性內容比較時的效果，進而提高整合時的精確率。最後將各方法求得節點整合規則予以合併，期望透過合併兩種綱要整合方法的效果，能比使用單一整合的方法有更高的精確率及完整率。本研究後續包含下列章節：第二章概述相關之研究文獻，包括 XML 綱要映對和整合相關文獻的整理與探討。第三章說明本研究所提出的方法架構。第四章進行本研究的實驗結果分析，利用不同資料來源的顧客基本資料，依本研究所提出的方法架構進行實驗，並分析實驗結果，以驗證本研究的可行性。第五章說明本研究之結論。

2. 文獻探討

隨著資訊科技的進步，資訊架構、技術及工具不斷的推陳出新。為提升電子資訊交換的準確性，關於 XML 之技術與應用，在學術研究或產業上的應用，均有長足之發展。本章節將對於本研究相關文獻進行探討，我們將先說明 XML 之發展沿革與特性以及在資料交換上的應用，次闡述綱要映射的分類，而後探討異質性相關課題及說明知識本體在資料整合上的應用，並說明如何利用綱要整合技術處理企業內不同業務系統間資料異質性的問題。

2.1 XML 沿革與特點

國際標準組織 (International Standard Origination, ISO) 為了能有效的進行資料交換，於 1986 年制定了標準通用標籤語言 (Standard Generalized Markup Language, SGML)，標籤語言意指將文字及其相關訊息予以結合，用以展現網路上文件的結構及表達出文件各部分的意義，以利電腦閱讀處理。但 SGML 龐大且複雜的功能雖然使得 SGML 具有較高的彈性，但也增加了應用程式開發上的難度，並且 SGML 無法在網際網路上呈現動態資料以及主流瀏覽器廠商也不推廣與支持。所以 SGML 產生的一個分支，超文件標示語言 (HyperText Markup Language, HTML) 取代了 SGML 在網際網路上的應用，以其方便使用的特性，用最簡單的方式將文字或資料呈現於網際網路上，供不同瀏覽器展現給使用者。但由於 HTML 是一種專為 WWW 網頁顯示及瀏覽而設計的簡易標示語言，其標籤集是固定且大都屬於呈現導向的標籤，若利用 HTML 作為資料交換的格式，將很難清楚的描述要交換的資料內容。

基於上述緣由，W3C 之 XML Schema Working Group 遂將 SGML 大幅簡化制定出 XML，其繼承 SGML 自訂標籤的優點，同時刪除一些 SGML 複雜的部分。只需了解 XML 綱要語言的資料模型及元素 (Element、complexType、simpleType、Attribute) 之名稱、型態及限制條件的使用方式，就能處理 XML 綱要。而 XML 的目的是用來描述資料內容，並沒有預設標籤，而是由使用者自行定義描述資料所需的標籤，並且 XML 文件為純文字格式編輯，所以可讓應用系統作為資料儲存格式，也可作為資料交換的標準。依 XML 特點在下列說明 [2]：

- 語法簡單規範嚴謹：XML 文件用來標示節點稱為元素 (Element)，元素以起

始標籤 < TagName > 及結束標籤 < /TagName > 包含描述該節點的資訊；如屬性 (Attribute)、字元資料、實體參照或其他元素等。

- 高彈性及可延展性：XML 僅提供基本的語法，由使用者依據應用需求自訂標籤。因此 XML 具有相當大的彈性及可延展性。XML 的可延展性使其成為一種 Meta-language (詮釋語言)，是一種用來「描述 (或設計) 語言的語言」，可隨著不同的應用而無限延伸。

- 公開制訂且開放：任何人均可無償的使用，不用顧慮某些廠商制訂特殊規格，進而把持相關資源。

- 具物件導向觀念：XML 文件的每個元素均可視為是一個物件，物件導向的特性，如封裝 (Encapsulation)、繼承 (Inheritance)、多型 (Polymorphism) 或類別化等均可在 XML 中實現。

- 具樹狀結構的資料分析：XML 文件樹狀結構是物件導向觀念的展現，並同時有利於 XML 文件的存取及內容搜尋。

- 規範與內容分開處理：XML 文件的規範通常定義在 XML 綱要 (XML Schema) 中，透用解析器 (Parser) 依據 XML 綱要對文件內標籤或元素的定義，進行驗證 XML 文件是否符合規範。

- 多種方式呈現：可透過 CSS(Cascading Style Sheets) 及 XSL(eXtensible Stylesheet Language) 等方法展現 XML 文件的內容。

- 適用於各類應用標準：XML 已被廣泛地運用作為數據交互媒介。目前已有 Oracle、SQL Server、 MySQL 及 Informix 等資料庫軟體均提供 XML 格式的資料輸出。也可作通訊協定 (Protocol) 的資料格式，如網路服務 (Web Service) 中的 SOAP(Simple Object Access Protocol) 協定。

2.2 利用 XML 進行 EDI 之應用

過去若要進行跨企業的資料交換時，通常是導入電子資料交換系統 (EDI - Electronic Data Interchange)，讓資料的透通性增加，使得企業在營運上得到更多的優勢。但由於每一廠商均有其特殊的資料格式，所以為了與其進行資料交換，就必需建置一套專屬的 EDI 系統。而且又因環境與交易行為的變動，常需修改 EDI 系統中的交易規則。這些問題都增加了企業維運 EDI 系統的成本及人

力。為解決上述問題，從 1968 年開始就陸續推出應用於各領域的 EDI 標準，目前僅剩 ANSI X.12 及 EDIFACT(適用於歐、亞) 二大標準。由於其導入門檻與限制較高，僅有較大型的企業才有能力導入，並且修訂緩慢，跟不上市場實際的需求，所以無法有效的推廣。在 XML 推出後，因其可訂定以 XML 為格式的 EDI，利用網際網路（Internet）傳輸、文件的可延伸性、能與 ANSI X.12 和 EDIFACT 相容、提供安全機制、多語系等等優點，逐漸取代 EDI 而作為資料交換的標準。使用 XML 作為兩方企業交換資料格式的定義，即不用投資額外的軟硬體設備，也不用修改資料的格式。

許多企業利用 XML/EDI 作為資料交換的解決方案時，僅以 XML 及 XML Schema 定義交換資料的格式，再利用網際網路進行資料傳遞。但接收資訊的企業仍需利用轉換工具的協助，才能自動化的進行資料交換。因為 XML 具有自訂標籤的特性，所以各企業會依其慣用的字彙進行編寫 XML 標籤。也由於同一產業並無一致性的資料字典，所以常需經轉換後，才能進行溝通。

2.3 知識本體論

在資料進行交換與整合的過程中，常將資料來源的語意描述至知識本體中，其主要是定義在特定的應用領域中，什麼是可被接受的實體與事件以及實體間如何進行交互影響。知識本體是由許多知識術語所組成的集合，包含語彙、語意上的相互連結以及在推論和邏輯上的簡單規則 [17]，並且用來描述與定義各種知識的語言，以便達到知識分享共用的目的。

企業內不同的應用系統進行資訊整合及交換的工作時，首先會找到具有共通性的資料詞彙，將這些詞彙定義在知識本體後，可以告知使用者這些詞彙屬於哪個應用領域、定義以及代表的知識範圍與架構為何。在特定的應用領域定義其知識本體需具備以下條件 [16]：

- 將實體予以概念化，予以結構性的定義。
- 以清晰明瞭的語言進行描述，可為一般人所讀取。
- 明確定義概念的使用方式以及其限制，可供電腦系統解析。
- 其知識是共同認可的，並可分享至應用領域中。

企業內部的員工因其不同的學業背景、開發語言與技術應用，進行資訊整合時常發生溝通的障礙，這時就可以利用知識本體作為對事物的共同認知，以

降低或避免概念與術語上的衝突。雖然在各應用領域中的知識本體均是由各專家所定義，但仍然會受不同的文化教育、社會環境以及產業別所影響而制定出不同的內容。本研究為解決整合時所產生結構異質性與語義異質性的問題，將本體資訊的定義分為二類：

- 概念類型：包含了同義異詞以及慣用縮寫的字詞比對，例如：no 與 id 以及 birth 與 birthday。
- 關聯類型：代表了字詞的組合關係，例如：name 與 first_name、last_name。

本研究解析 XML 文件後，將其綱要的節點名稱取出，除利用節點名稱進行綱要元素間的映射外，也藉由知識本體在節點名稱及字詞組合關係上的導引，用以協助完成不同來源綱要元素間之映對，進而提升以綱要資訊進行綱要整合時的準確率。

2.4 綱要映射的分類

映射 (mapping) 是對二個不同來源的資料集，利用其中的綱要資訊及實例實訊進行語意或結構層面的對應工作，藉此得到統一的規範。一般而言，不論是搜尋引擎、資料倉儲或是資料整合都需使用映射的相關技術。進行異質資料映射時，最常面臨的問題是資料的不一致，過去的研究中將資料映射時產生的不一致的原因分為"因應用層面不同，其涵義以及結構的考量也會不同"與"無法控制資料品質的優劣"這二類 [21]。目前應用系統常使用 XML 作為系統間溝通的資料媒介，這些資料需要經轉換後，產生格式一致性的資料，才會成為有價值的資訊。Erhard Rahm, Philip A. Bernstein 在其研究中提出對於綱要映射方法的分類 [26]，該研究依使用映射方法的數量進行分類：單一映射方法 (individual matcher approaches) 或是結合多種映射方法 (Combining matchers，例如：Hybrid 與 Composite 兩種映射方法)。映射的參考資訊分為：只考慮綱要資訊 (Schema-only based) 或是將實例 / 內容的資訊一併考慮 (Instance/content-based)。再來是以元素屬性 (Element-level) 或是加入結構資料 (Structure-level) 作對應。最後以語義 (Linguistic) 或是限制條件 (Constraint-based) 作分類。依 XML 映射的方法在下列分別說明：

1. Schema-only based or Instance/content based：綱要的資訊包含了名稱、描述、資料型別、關係類型 (例如：屬於、部分)、限制、綱要結構等元素

屬性。絕大部份的綱要映射均屬於此類，例如：Jin Pei, Jun Hong and David Bell [24]。若綱要異質性過高，而造成映射資訊不足。為提高映射的品質，則會利用文件內的實例資料進行映射 [4] [5] [33]。

2. Element-level or Structure-level：以綱要內個別元素的屬性，或是在同結構中一起出現的多個屬性作為映射的依據 (例如：產品代碼、單價、數量的訂單基本資料)。

3. Linguistic or Constraint-based：依照綱要元素的文字描述和名稱找出相似語意的資料，包含相同文字 (equality)、相同字源 (after stemming)、同義字 (synonyms)、同類字 (hypernyms)、同聲字、使用者提供、同型異義字 (homonyms) 等等辨識的方式 [11]; 而限制條件則是以已知的特性作為映射的參考 (例如：資料型別、資料值的區間、關係類型、對應程度 (Cardinality) 等等。

4. Hybrid or Composite：為了更好的適用性和準確性，合併多種映射方法是較為可行的方法。而合併的方法分為混合式 (Hybrid)，將多個映射方法整合至同一程序中。此種方法可以同時進行辨識，較早將不符合的資料過濾掉 [11] [12]。另一類是複合 (Composite)，多個映射方法單獨執行，再將其產出結果合併後進行辨識 [13]。

在映射的過程中，常需處理欄位異質性的問題，包括名稱異質 (Naming heterogeneity)、結構異質 (Structural heterogeneity)、語意異質 (Semantic heterogeneity) [14] [20] [19]。對於名稱異質性，常利用兩個 XML 文件中的節點名稱進行比較。利用 A 字串轉成 B 字串之間的編輯距離計算其相似度，如 :Edit Distance, Affine Gap Distance, Smith-Waterman distance and q-gram [19]。結構的異質性指的是定義同樣語義但不同結構的本體規範 [8]。語義異質性是指在不同的商業環境中相同或相關的資料其含義、解釋或用途上的差異；解決的方法分為二類：

1. 語義網 (Semantic Web) 把能被電腦解析的語義（Meta Data）加入文件中，並放置於網路上。文件內的資訊將利用網路的透通性，成為共用資訊的參考介面。

2. 本體論 (Ontology) 依照嚴謹的定義將使用到的概念抽象化後，使用其抽象化的相似值，作為 XML 結構、資料間語義的參照。

2.5 TF-IDF

本文利用 TF-IDF（term frequency – inverse document frequency）對分詞後的實例資料，進行記錄配對的權重分析。TF-IDF 常用於資訊檢索與資料探勘，用於評估字詞在文件中的重要程度。該方法有兩大核心概念：一、TF 詞頻 (Term Frequency)：表示一個詞語在文件中出現的頻率。二、IDF 逆向文件頻率 (Inverse Document Frequency)：度量一個詞語在所有文件中的重要性。字詞在度量時，若出現在文件中的次數越多，該字詞對該文件代表性愈強；包含字詞的文件愈少，則該字詞的區分力愈大。

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}}$$

在上述公式中 ,$n_{i,j}$ 表示字詞 i 在文件 j 中出現的頻率，$\sum_k n_{k,j}$ 表示所有字串的出現次數的總和；由此公式可看出 j 文件中共有 k 個字詞。

$$idf_i = log\frac{|D|}{|\{j : t_i \in d_j\}|}$$

在上述公式中，以 $|D|$ 表示所有的文件數，$|\{j : t_i \in d_j\}|$ 表示字串出現的文件數。

$$tfidf_{i,j} = tf(i,j) * idf(i)$$

經由上述算式 , 便可計算出字詞在文件中的權重值。

2.6 綱要整合

企業為了做出有效的決策以及達成其營運目標，需要同時執行各類不同的工作 [7]，而為了能得到最佳的競爭優勢，常需蒐集與萃取散佈在各處的資料，這樣才能提高經營的競爭力，所以使得綱要整合的議題至今仍被熱烈的討論。企業為了達到資源分配最佳化和企業經營價值最大化的目標，而引進企業資源規劃系統 (Enterprise Resource Planning)。雖然 ERP 系統的目的在於有效的掌握內部所有資源，以協助企業進行有效的整合規劃，但此類的大型系統，也常受限於商業邏輯的設計或是資料儲存方式等其他因素，導致無法與其它系統進行資料交換。經營環境的快速變化原本就考驗著經營者的應變能力，因此經營者若能取得企業內部整合後的各種資訊，才能更提高商業經營的競爭力。所以如何進行資訊的整合及交換，是目前進行整體業務發展及管理前，必須要先克服的問題。

以金融業控股公司為例，它可透過投資控股的方式進行跨業整合，其範圍可分為銀行業、保險業、信託業、證券業和租賃業等等金融機構。由於其不同的商業特性分別有不同的法令規範以及不同的主管機關進行監督。為因應各自的金融業務，其主要系統資料庫以及檔案格式都有些許的迥異，但其都存放著部份重複的資料，例如：顧客基本資料、KYC 資訊、信用額度、交易黑名單等等。而金融控股公司為了有效的運用這些資訊，必須由各子系統提供相關的資料，再由統籌資料整合的系統進行資料解析與辨識，並依照資料的相關性進行匯集運用。因此不論來源資料的正確性以及資料傳遞的時效性，在資料整合時確實有其重要性與迫切性。進行資訊整合及交換的工作時，首先會面對到的問題就是該如何去找到具有代表性資料的資料來源，而當我們找到適當的資料來源後，就需要考慮如何在不同的環境間進行資料交換，並作有效的溝通。Afsarmanesh, H. 在其研究中將綱要整合分為三個階段 [28]:

1. 整合前準備 (Pre-integration): 將欲整合的綱要概念化，使其具有同樣的表示方式，以方便進行綱要間的比較。

2. 映射 (Mapping): 透過映射程序依據綱要內元素的名稱、屬性、關聯或結構，計算元素彼此間的相似度。

3. 整合 (Integration): 綱要映射的結果作為整合的參考，先將整合的綱要置放於整體綱要中。

在三階段中的整合階段是透過兩個異質性資料來源中的綱要元素進行映射後，將其整合至整體綱要中。在整合的過程中將面臨多種的問題，例如：整合的資料來源可能依其應用領域不同，而分別具有不同的結構、命名方式及知識領域，這些不同點將使得綱要整合困難重重。綱要整合能利用綱要的屬性、綱要間的關聯來進行異質資料源的整合問題，或可加入實例整合以提高整合的正確性及完整性。而不相似的元素再透過其它技術 (如 :wordnet、本體論、語義網) 解決結構或語義異質性的衝突。

3. 研究方法

目前大多數的研究是針對 XML 綱要中的元素名稱、描述、型別、關係類型、限制、結構等資訊進行判斷 [22] [25] [18]。但若僅使用 XML 綱要的資訊進行資料整合，會因資料來源的異質性而造成整合上的錯誤。例如：使用不同的名稱來表示相同的概念（ Amount 和 Quantity）、使用相同的名稱代表不同的概念（簽核系統中的 Employee 可代表送簽者或是審核者）或者是用兩個不同的格式傳達同樣的訊息（Profit 和 Net）。正因如此，才需要利用 XML 的資料內容及資料型態進行比對 [23]。

實例資訊主要是表達元素的內容，透過這些資訊較可準確的評估資料間的相似性。但在利用實例資訊進行映射或整合時，也常會發生同樣的資料出現在不同的綱要上。例如：透過實例資訊進行綱要整合時，實例 A 內的 Phone 節點資料對應到實例 B 的 DayPhone 及 NightPhone 節點資料，在發生整合衝突時，此時可以加入綱要的屬性進行比較（例如：節點名稱、型態、節點的結構等等），以判斷出節點的差異性。本研究分別利用實例資訊以及綱要資訊找尋可進行整合的節點，再將比對出來的資訊進行結合後，找出可以整合的節點。本研究將分為四個步驟進行綱要整合：

1. 分析來源 XML 文件（圖：3.1) 及目的 XML 文件（圖：3.2)，分別解析出綱要資訊（圖：3.3、 3.4) 及實例資訊（圖：3.5)。

2. 綱要資訊依綱要節點整合的機制 [3]。先求得各 XML 葉節點間的相似度。

3. 將目的實例資訊清理及分詞後，把分詞結果記錄至索引庫中。以來源實例資訊分詞後的字詞作為索引條件，至索引庫中取出符合的字詞，找出配對的相似資料。並求得 XML 葉節點的相似度以及整合規則。

   (a) 將目的 XML 文件的實例資訊進行資料清理，並統一表示的方式（例如：小寫英文全部轉大寫）。清理時利用正規表示式識別特定規則的字詞（例如：電子郵件、網址），避免過度的資料清理，而造成比較時的錯誤。將清理後的實例資訊進行分詞，並把分詞結果記錄至索引庫中。

(b) 依序取出來源 XML 文件中的實例資訊，將各記錄 (Record) 進行分詞。以來源記錄分詞的結果作為索引條件，至索引庫中取出最高相似度的配對記錄。

(c) 利用相似度函式計算每個配對記錄分割後的字詞相似度。

(d) 將相似度超過門檻值的目的 XML 實例，依其規則表中的資訊求得可整合的節點後，將其記錄至節點整合表中。

4. 將依網要資訊及實例資訊求得的網要整合資訊彙整後，求得可進行整合的葉節點。

```
1  <?xml version="1.0" encoding="Big5"?>
2  <!--連絡資訊-->
3  <contactlist>
4    <contact>
5      <id>01</id>
6      <name>John Smith</name>
7      <telephone>0956112341</telephone>
8    </contact>
9    <contact>
10     <id>02</id>
11     <name>王會安</name>
12     <telephone>02-2512-2234</telephone>
13   </contact>
14   <contact>
15     <id>03</id>
16     <name>周禮仁</name>
17     <telephone>0945-32125</telephone>
18   </contact>
19 </contactlist>
```

圖 3.1: 原始的 XML 文件 A

```xml
<?xml version="1.0" encoding="Big5"?>
<!--連絡資訊-->
<contactlist>
  <contact>
    <id>01</id>
    <firstname>John</firstname>
    <lastname>Smith</lastname>
    <telephone>
      <areacode></areacode>
      <phonenumber>0956-1112341</phonenumber>
    </telephone>
  </contact>
  <contact>
    <id>02</id>
    <firstname>會安</firstname>
    <lastname>王</lastname>
    <telephone>
      <areacode>02</areacode>
      <phonenumber>2512-2234</phonenumber>
    </telephone>
  </contact>
</contactlist>
```

圖 3.2: 原始的 XML 文件 B

```xml
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" version="1.0">
  <xsd:element name="contactlist">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="contact">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="id"/>
              <xsd:element name="name"/>
              <xsd:element name="telephone"/>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

XML Document A 綱要資訊

圖 3.3: 解析後的綱要資訊 A

圖 3.4: 解析後的綱要資訊 B

Record	id	name	telephone
r1	01	John Smith	0956112341
r2	02	王會安	02-2515-2234
r3	03	周禮仁	0945-32125

XML Document A 實例資訊

Record	id	firstname	lastname	areacode	phonenumber
s1	01	John	Smith		0956-1112341
s2	02	會安	王	02	2512-2234

XML Document B 實例資訊

圖 3.5: 解析後的實例資訊

3.1 XML 綱要整合機制

先前研究探討不同來源之 XML 綱要進行映對的情況及問題 [30] [31]，在過去的研究中使用知識本體作為塑模及設計的概念及研究中將 XML 綱要映對問題提升至概念的層次進行綱要的映對 [9]，藉由門檻設定提升節點映對的正確及完整率，達到資料整合及交換的目的。不過，因 XML 文件語意的異質性，XML 綱要具有自訂標籤名稱的靈活性，即使是屬於相同的應用範圍資料，也會因使用者對於節點元素名稱定義的差異，造成映對不符的結果，例如：相同元素但命

名的名稱不同，相同型態的元素但其內容的意義卻不同。資料整合及資料交換常常會遇到具有相同的命名名稱但實質上卻是代表不同涵義的資料混雜在一起的窘境，所以如何定義良好的綱要映對規則，是資料整合、交換、檢索所面對的一個重要的課題。

XML 綱要資訊是由多個描述記錄的屬性內容所組成，本文將利用屬性名稱進行相似度的比較。除判斷字詞完全或部分字串相同，並利用知識本體定義之概念相似度，增進名稱語意及綱要結構的判斷 [3]。以整合 XML 文件 A(圖 : 3.6) 及 XML 文件 B(圖 : 3.7) 的綱要資訊為例，可表示為 Integrate(A,B)。以 A 為目的綱要，產生的整合綱要節點有 "id"、"name"、"telephone"，這些節點也代表接下來提供查詢的葉節點。若僅以節點名稱是否相符，作為整合的規則時，僅有 i:id 傳回綱要 A 與綱要 B 共同有的節點名稱 "id"。

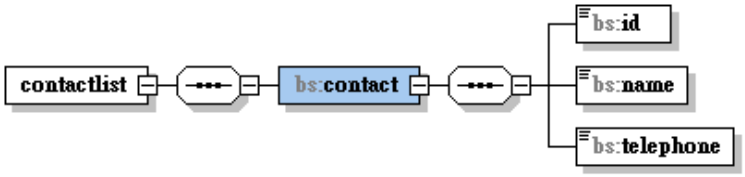


圖 3.6: XML 文件 A 綱要資訊

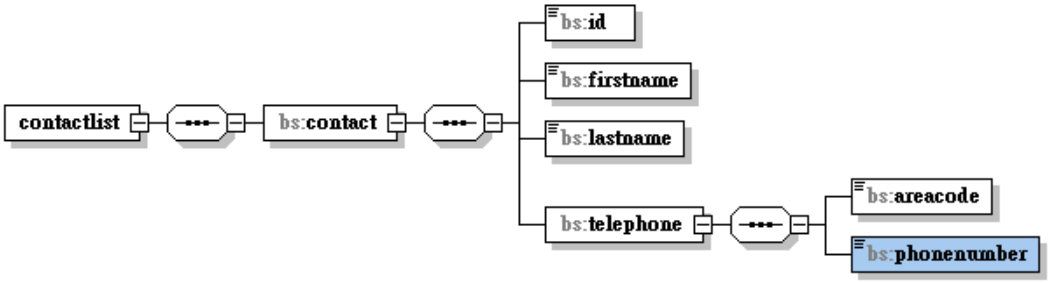


圖 3.7: XML 文件 B 綱要資訊

但經由 name Ontology 及 telephone Ontology 的概念關係 (圖 : 3.8)，可以對應到部份和整體的關係，可經由這層關係導出對應的規則。借助本體 O 可以決定 i:name 傳回綱要 A"contactlist/contact/name" 及綱要 B"contactlist/contact/firstname"、"contactlist/contact/lastname"。以及 i:telephone 傳回綱要 A"contactlist/contact/telephone" 及綱要 B"contactlist/contact/telephone/areacode"、"contactlist/contact/telephone/phonenumber"。

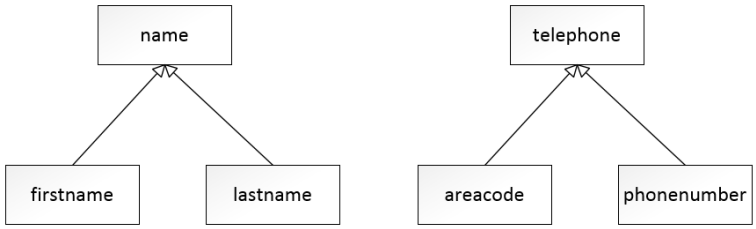


圖 3.8: 本體論 (O)

演算法 CNS(n_i,n_j) 為對 Count Node Similarity 模組之簡要流程。演算法第 1 行先利用本體資訊比對 *sourceNode* 與 *targetNode* 節點名稱是否相符，若相等則得到相似度 1；若不相等則需考量來源不同的資料可能結構或語義上有所不同。所以從第 4 行至第 15 行開始利用兩層迴圈計算各字詞的相似度，當字詞的名稱相符時，則相似度累加 1，否則利用本體資訊比對字詞間名稱的相似度。

Algorithm 3.1 CNS(n_i,n_j)

Input: n_i: source node; n_j: target node;
Output: node similarity $ns_{i,j}$

```
 1: if ns_{i,j} ← OntologyRelationSimilarity(n_i,n_j) > 0
 2:     return ns_{i,j} ;
 3: else
 4:     for iw = 1 to n_i.word.length
 5:         maxSimilary = 0;
 6:         for jw = 1 to n_j.word.length
 7:             if n_i.word(iw).equals(n_j.word(jw))
 8:                 s_{i,j} += 1;
 9:             else
10:                 s_{i,j} ← OntologyRelationSimilarity(n_i.word(iw),n_j.word(jw)) > 0
11:             if s_{i,j} > maxSimilary
12:                 maxSimilary = s_{i,j};
13:         endfor
14:         ns_{i,j} += maxSimilary;
15:     endfor
16:     return ns_{i,j} / maxWordLength(n_i,n_j);
```

假設以 n_i 節點名稱"Company Name" 與 n_j 節點名稱"Corp Addr" 為例，iw 為 n_i 包含的字詞數量 2，jw 為 n_j 包含的字詞數量 2。若本體資訊中設定"Company" 與"Corp" 的相似度為 1，在經過 CNS 演算法逐一比對各字詞間的相似度後，可取得名稱相似度 0.5。

3.2 XML 實例整合機制

實例整合模式 (Instance-based)，是利用資料內容進行比對，只包含元素層級的映對。以先前研究為例 : 語意映對及限制條件映對方法，這類映對方式較強調元素內容的意義 [10]，而不考慮綱要資訊，所以映對方法的處理對象是資料內容而非元素名稱，主要的優點是可以精確評估元素內容值的特性，最適合用在半結構化或非結構化的資料，因為這種資料的綱要資訊較少，如果綱要資訊有錯誤也可以被發現，遇到模稜兩可的映對情況時，例如：綱要 A 的 Customer 元素應該對應到綱要 B 的 CustomerID 元素還是 CustomerName 元素？這時比對元素名稱和資料型態可能無法直接判斷，若再比對元素內容則可清楚判斷。甚至有些沒有綱要的資料也可使用此種方式建立綱要並產生映對。

3.2.1 前置處理 (資料清理)

實例資料的內容會包含一些”無意義”、”語助詞”、”符號”等字元，為避免比對時的誤差，所以會先將這些資料進行前置處理。例如 :U.S.A 會轉換成 USA、02-25117233 轉換成 02、25117233。前置處理的項目還包含了，大寫字母的縮略語、電子郵件、網址、各式數字或是無意義符號，例如 :”_”、”-”、”/” 等等。如圖 : 3.5中的 r2(02, 王會安 ,02-2515-2234)，當中的 02-2515-2234 為例。為求得最細的分詞結果本研究會將 ”-” 符號以空白替代，分詞時預期將會產出 02、2515、2234 三個字詞。

3.2.2 分詞

較早期依 instance-based 進行整合的論文 [5] [10]，大多是以單一個葉節點的資料內容，作為比較的單位。但是整合規則出現單葉節點對映多個葉節點時，就會無法有效的進行整合。例如 :r1(TagName:username , Value:John Chen) 經過映射後，可對應至 s1(TagName:firstname , Value:John) 及 s1(TagName:lastname , Value:Chen) 兩個節點。若映射時僅以單一節點的資料為最小比較單位時，將無法有效的決定整合規則。

所以本研究進行分詞時分為兩個部份，第一部份 : 目前分詞方式大多是依據特定字元或空白符號進行分詞，常使用於英文、數字或英數混合的資料上。但中文的資料就無法有效的利用某些特定的符號進行分詞。以字串型別的地址資料為例 ”北市貴陽街一段 56 號”，資料當中並無包含任一有效的分割符號。所

以本研究對具有特殊規則的資料，會先準備好對應的方法進行分詞。若以地址格式的資料為例，將分為 2 階段處理：

1. 利用地址格式中特定的字詞進行分割，如縣 / 市、鄉 / 鎮 / 市 / 區、里 ... 等等。

2. 由於部份資料是由人工或其它系統匯入，而造成資料建立時間的差異或省略詞的差異。為避免比對時的錯誤，需將資料內容進行統一化，例如：”臺 ”=” 台 ”、” 北市 ”=” 台北市 ”、” 台北縣 ”=” 新北市 ”... 等等。

以 ” 北市貴陽街一段 56 號 ” 地址為例；將分割為 ” 台北市 ”，” 貴陽街一段 ”，”56 號 ”。第二部份：對於不在字典及規則的資料，會將葉節點內的資料依照分隔字元切割至單一字詞，如 ”John Chen” 分詞後會變成 ”John”、”Chen” 兩個字詞。這樣可以增加比對的細度，也能將整合的準確度提高。目的 XML 中的實例資訊，將分詞後的結果存入索引庫中 (圖 : 3.9) 分詞的資訊表示為；term = (r:t:s)，其中 r 代表出現的記錄、t 為資料所在的葉節點名稱、s 為單一節點內的資料分詞後出現的順序。

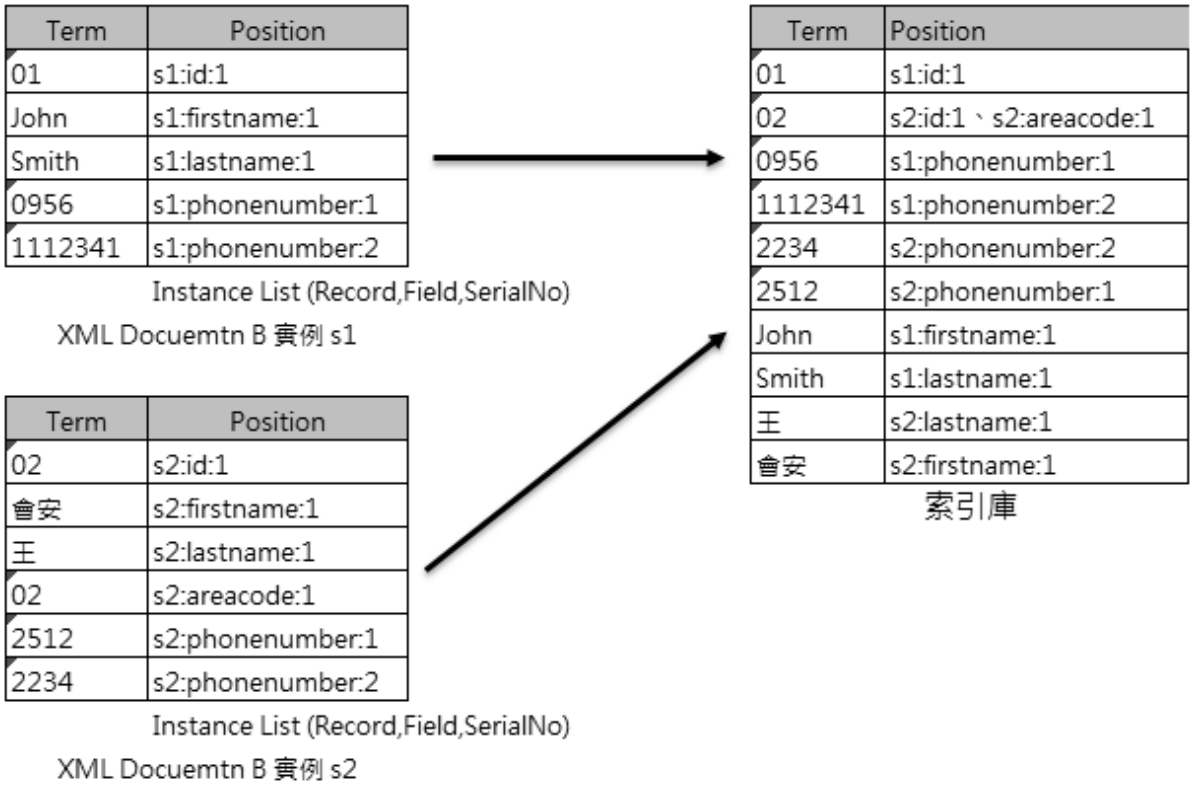


圖 3.9: 分詞的結果，存入索引庫中

3.2.3 相同記錄的配對

描述同樣事件的記錄其包涵的屬性內容，應當比其它記錄的相似度高，所以也更能協助找出配對的屬性。William B. Frakes and Ricardo Baeza-Yates 在研究中提出透過資料檢索的機制 [15]，先計算每個分詞的權重後，求得整筆資料的相似度。將相似度超過閥值的資料，依相似度作遞減排序，讓相同屬性個數較多的記錄先進行處理。

各資料在進行字詞權重計算時，會將字詞在該文件中的重要程度（區域權重 :local weight）以及在全部文件中的重要程度（全域權重 :global weight）列入計算的考量之中，本研究將引用的 TF-IDF（Term Frequency and Inverse 作為權重的計算方式。利用 TF-IDF 計算檢索詞條在目的 Instance 中的權重，透過上述的轉換，將記錄透過權重 w 來表示 Instance(w_1,w_2...,w_n)。利用餘弦函式的相似度函式來計算文章之間的相似度，以便於找出相似度最高的記錄。

向量空間模型運用餘弦函式計算相似度是由 Gerard Salton 在 1979 年所提出 [27]，主要的概念是將每份文件轉換為多維度空間中的一個向量，而文件其中包含的詞彙可視為向量中的維度若 A 向量是 [x1,y1]，而 B 向量是 [x2,y2]，如圖 : 3.10。此兩向量的夾角可由下面的公式求得。

$$\cos\theta = \frac{x_1 x_2 + y_1 y_2}{\sqrt{x_1^2 + y_1^2} \times \sqrt{x_2^2 + y_2^2}}$$

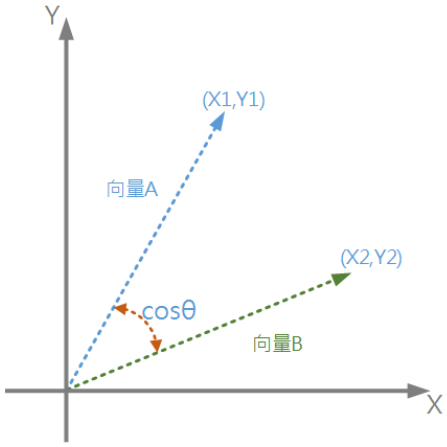


圖 3.10: 向量空間模型

利用前述的 TF-IDF 可求出字詞的權重（可視為各元素的量值）。此時每份文件 D 可視為 D=(w1,w2,⋯ wn)，其中 w 為該詞彙在文件 D 中之權重。利用餘

弦函式 (cosine) 計算兩向量間的相似度，可利用這兩向量的夾角大小斷其相似的程度；夾角成為 90 度時，餘弦值等於 0，代表兩篇文章完全沒有任何相似之處，反之如果兩向量的夾角為 0 度時，餘弦值等於 1，也就是代表著兩篇文章在此多維度空間中是高度相似的。假設 X 和 Y 是兩個 n 維的向量，A 表示為 $(a_1,a_2...,a_n)$，，B 表示為 $(b_1,b_2...,b_n)$，則 X 與 Y 的夾角 θ 的余弦可由下面的公式求得。

$$\cos\theta = \frac{\sum_{i=1}^{n}(a_i \times b_i)}{\sqrt{\sum_{i=1}^{n}(a_i)^2} \times \sqrt{\sum_{i=1}^{n}(b_i)^2}}$$

本研究將來源 XML 文件內的 Instance，各自透過分詞機制產生 term t 的集合表示為來源 Instance$(t_1,t_2...,t_n)$，將 $t_1,t_2...,t_n$ 視為檢索詞條。透過在索引庫檢索的結果，可以得到有哪些 Term 是相符的，也可得知哪些目的 Instance 包含了這些 Term。檢索詞條的判斷標準說明如下：

1. Equality: 字詞完全相同，表示為”E”。
2. Prefix: 前綴字相同，表示為”PF(s,e,tl)”。
3. Suffix: 後綴字相同，表示為”SF(s,e,tl)”。
4. SubString: 中間部分的字相同，表示為”SS(s,e,tl)”。
5. Acronym: 縮寫字，找出開頭字母相同於另一個字，表示為”A”。

說明：s 是相似的起始位置，e 是相似的結束位置，tl 是檢索字詞的總長度。

以配對記錄 (r1,s1) 為例 (圖：3.11)。把 r1(01,John Smith,0956112341) 透過分詞機制取出查詢索引庫的字詞 r1(01,John,Smith,0956112341)。將 r1 的字詞至索引庫中查詢 s1 是否有全部相符的字詞，若有則將其它記錄下來。但由於分詞時可能因為資料中有分詞符號，將資料分出 1 個以上的字詞 (例如：”0956-112341”，分詞為 ”0956”，”112341”)，所以若索引庫中沒有全部相符 (Equality) 的字詞時，則判斷是否有部份相符的字詞 (例如：”r1:telephone:1” or ”s1:phonenumber:1”,”s1:phonenumber:2”)，將這些資訊記錄至節點整合表中。

r1 Term	01	John	Smith	0956112341	
Position	r1:id:1	r1:name:1	r1:name:2	r1:telephone:1	
比對結果	E	E	E	PF(1,4,10)	SF(5,10,10)
s1 Term	01	John	Smith	0956	112341
Position	s1:id:1	s1:firstname:1	s1:lastname:1	s1:phonenumber:1	s1:phonenumber:2

圖 3.11: 配對記錄 (s1,r1)

演算法 IIM(n_i,n_j) 為對 Count Term Similarity 模組之簡要流程。演算法第 1、2 行利用兩層迴圈取出 $sourceTerm$ 與 $targetTerm$ 字詞內容。若字詞內容相符，則得到相似度 1。若不相等則需考量字詞內容結構上的問題，所以從第 7 行至第 14 行開始判斷 $st_x.term(i)$ 與 $tt_y.term(j)$ 是否有相似的部份，若符合任一判斷標準，則計算其字詞的相似度。

Algorithm 3.2 IIM(st_x,tt_y)

Input: st_x: source record; tt_y: target record;
Output: term similarity $ts_{i,j}$

```
1:  for i = 1 to st_x.term.length
2:      for j = 1 to tt_y.term.length
3:          if st_x.term(i) = tt_y.term(j)
4:              mappingType = 'E';
5:              termSimilary = 1;
6:          else
7:              if st_x.term(i).endsWith(tt_y.term(j))
8:                  mappingType = 'SF';
9:              elseif st_x.term(i).startsWith(tt_y.term(j)) > 0
10:                 mappingType = 'PF';
11:             elseif st_x.term(i).indexOf(tt_y.term(j)) > 0
12:                 mappingType = 'SS';
13:             elseif st_x.term(i).replaceAll([not(A-Z)], '').equals(tt_y.term(j))
14:                 mappingType = 'A';
15:
16:             if mappingType not null
17:                 termSimilary = st_x.term(i).length / tt_y.term(j).length;
18:
19:         Records the termSimilary to Database
```

假設以 $st_x.term(i)$ 字詞內容 "John" 與 $tt_y.term(j)$ 字詞內容 "JohnChen" 為例，因為 "John" 包含於 "JohnChen" 之中，所以符合前綴字相同 (SF) 的條件。$st_x.term(i)$ 字詞內容的字數 4，$tt_y.term(j)$ 字詞內容的字數 8，在經過 IIM 演算法

計算後，可取得字詞相似度 0.5。

3.2.4 可整合的節點

在利用演算法 IIM(n_i,n_j) 計算後，可以得到配對記錄中每個字詞間的關係及相似度。以配對記錄 (r1,s1) 為例 (圖 : 3.11)，可將其比對結果整理為 (E:id:1,id:1)、(E:name:1,firstname:1)、(E:name:2,lastname:1)、 (PF:telephone:1,phonenumber:1) 與 (SF:telephone:1,phonenumber:2)。再依照 r1 中相同字詞的順序進行交集比對，例如 :(E:name:1,firstname:1)、 (E:name:2,lastname:1) 可整合為 (name,firstname\lastname)，而 (PF:telephone:1,phonenumber:1)、(SF:telephone:1,phonenumber:2) 則可整合為 (telephone,phonenumber)。最後將配對記錄 (r1,s1) 中可整合的節點整理為 (id,id)、(name,firstname\lastname)、(telephone,phonenumber)，並將其記錄至整合池中。各配對記錄利用上列方式取出可整合的節點資料後，分別記錄至整合池中 (圖 : 3.12)，再依照各節點被參照的累積次數進行降冪排序後，取出被參照到最多次的整合結果。依照該判斷方式作整合節點的選取，可得到依實例資訊進行綱要整合的 3 個整合節點規則 (圖 3.12)。

實例A葉節點 / 實例A葉節點	id	name	telephone
id	10		
firstname		2	
lastname		5	
areacode	3		
phonenumber			6
firstname/lastname		9	
area/phonenumber			8

圖 3.12: 整合池

3.3 實例階層和綱要階層整合規則的結合

3.1 和 3.2 節分別利用實例資訊以及綱要資訊進行相似度的演算，因為其計算的基準不同，所以各自突顯出綱要樹中屬性的部分特性。例如：3.1 節考慮綱要名稱來計算節點的相似度，除判斷名稱的相似度外，另加入知識本體定義增進名稱語意的判斷；3.2 節根據實例資訊計算相似度度，也就是利用屬性中的文字片段求得對於屬性相似度的影響。我們認為 XML 文件中的綱要及實例資訊

會因為不同的狀況，而造成其資料的判讀性有所差異，例如實例階層的相似度權重為 w_1，綱要階層的相似度權重為 w_2，並且 w_1+w_2 應當為 1。所以可以依狀況給予各領域不同的相似度權重，應用時再予以調整。但本次實驗為避免因人為參與而影響實驗結果，所以 w_1 與 w_2 均設為 0.5。整合規則結合的步驟：首先，將實例階層與綱要階層都存在的整合規則，將其相似度相加總後存入整合池中。再來分別將僅存在於實例階層與綱要階層中的整合規則，存入整合池中。最後將小於門檻值的整合規則移出整合池。在利用多個測量方法除了可以增加計算的彈性，對於不同的應用領域也能有不錯的表現。僅使用單一的測量方法，求得屬性間的相似度是不夠的，若沒有將這些測量方法適當的整合，將無法有效的整合綱要屬性。

4. 實作與結果分析

實驗以某機構進行企業內部系統整合的客戶基本資料為例，以本研究提出的利用實例資訊與綱要資訊為基礎的綱要整合技術進行實驗。將欲整合的兩系統客戶資料作為實驗中的來源端與目的端的 XML 文件，透過解析將 XML 文件中的 XML 綱要及 XML 實例分離，分別使用 XML 綱要整合機制及 XML 實例整合機制取得各自的整合規則。將實例階層和綱要階層整合規則予以結合，以解決單一整合方法所產生的整合衝突，並加強整合規則的精確率及完整率。

本實驗考量企業內部系統整合時，其資料的相似度最大，但因為應用的專業領域、系統建置時的背景、所需遵循的法律規範等因素不同，在資料結構上的設計亦有相當的迥異。當客戶資料變動時，需人工至各系統進行修改，其中可能造成耗時、重工、系統邏輯差異以及人為疏失等等的問題。一旦資訊整合後，員工就可以利用整合過的資訊，達到資料一致性和降低疏失的目標 (圖 : 4.1)。

圖 4.1: 企業內部資訊整合

4.1 實作項目

本研究實驗項目為”依 XML 綱要資訊整合”、”依 XML 實例資訊整合”及”彙整綱要及實例資訊”共 3 種，分析整合規則以精確率與完整率評估本研究的各實作方法。比較僅利用單一 XML 資訊 (綱要或實例) 的整合規則以及利用多個 XML 資訊進行整合的差異性。如圖 : 4.2所示，為本研究依據實驗結果在計算精

專家 / 程式計算	可整合	不可整合
可整合	TP (true positive)	FP (false positive)
不可整合	FN (false negative)	TN (true negative)

圖 4.2: 精確度及完整性評估

確率及完整率的計算方式。

為了驗證綱要整合的效益如何，由負責該應用領域的專家，先行產生正確的綱要整合集合。其結果可以表示為 C_h；透過程式得到的綱要整合的集合表示為 Cp。以圖 : 4.2作說明：

- TP = $C_p \cap C_h$，專家判斷可整合，程式計算可整合。
- FP = C_p - C_h，專家判斷不可整合，程式計算可整合。
- FN = C_h - C_p，專家判斷可整合，程式計算不可整合。
- TN = $C_p \cup C_h$，專家判斷不可整合，程式計算不可整合。

評估項目 TP 是本研究要提升的部份。將以 Precision(精確度)，Recall(完整性) 作為評估的方式，其定義如下 :

$$precision = \frac{TP}{TP \cup FP}$$

$$recall = \frac{TP}{TP \cup FN}$$

Precision 代表被程式判定可整合的資料中真正被專家判斷可整合資料的比率，Recall 則是程式正確判定可整合的資料占專家判定可整合資料的比率。本文將評估分別使用實例階層資訊和綱要階層資訊對於節點整合的影響，以方便修正所提出的相似度度計算方法。

4.2 實作環境

實作環境之軟硬體包含：

1. 硬體平台：採用 AMD Athlon(tm) IIX4 640 Processor 3.00GHz CPU、10.0 GB RAM 規格的個人電腦。
2. 作業系統：採用 Microsoft Windows 7 Professional 版本。
3. 程式語言：採用 Java SE Development Kit 6。
4. 程式套件：Document Object Model (org.w3c.dom)、Apache Lucene。
5. 資料庫：MySQL Community Server 5.0。

4.2.1 實作資料項目

本研究以企業合併時的顧客資料作為參考來源，設計本研究所探討的整合機制所會面臨的情境，將 XML 文件的資料主體分為綱要資訊及實例資訊兩種。圖： 4.3為實驗資料之 XML 文件的綱要資訊，利用”同義異形”的命名方式定義節點名稱，並設計來源 XML 綱要及目的 XML 綱要適用的本體資訊。利用設計資料測試本研究所考慮的綱要結構在本體資訊不完整的狀況下，是否能有效地利用自身所含資訊進行整合。圖： 4.4、4.5為實驗資料之 XML 文件的實例資訊，本實驗預先篩選”關聯性”較高的實例作為測試資料，測試實例資訊在無”單一且唯一”的狀況下，其資訊整合的狀況。最後測試本研究所考慮的結合綱要資訊及實例資訊，是否能有效地彌補單一整合方式的盲點，利用交叉實驗進行本研究的精確度及完整度的測試。

4.2.2 程式建置

圖： 4.6為實作程式的循序圖，包含使用者介面、XML 文件解析（產生 XML 綱要及 XML 實例）、XML 綱要整合 (利用綱要節點名稱及本體資訊進行映射後，依求得的節點間的相似度進行節點整合)、XML 實例 Dual Record 偵測、XML 實例整合規則以及整合 XML 綱要及 XML 實例的整合結果 :

1. 使用者介面：如圖： 4.7所示，使用者輸入來源 XML 文件 (A)、目的 XML 文件 (B) 及綱要整合所需要使用的本體 O 後，會先將 XML 文件剖析出

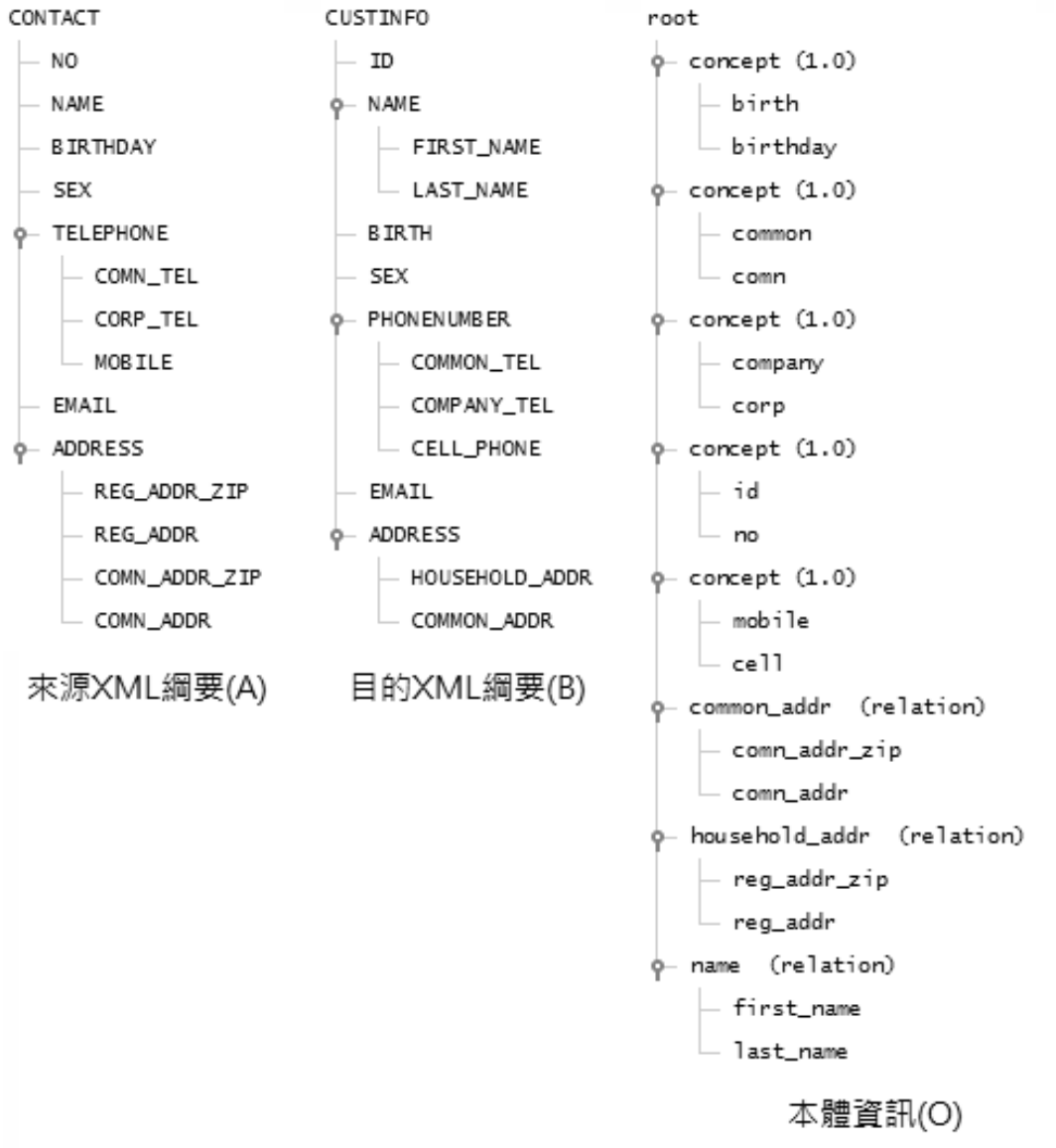

CONTACT
NO
NAME
BIRTHDAY
SEX
TELEPHONE
COMN_TEL
CORP_TEL
MOBILE
EMAIL
ADDRESS
REG_ADDR_ZIP
REG_ADDR
COMN_ADDR_ZIP
COMN_ADDR
來源XML綱要(A)
CUSTINFO
ID
NAME
FIRST_NAME
LAST_NAME
BIRTH
SEX
PHONENUMBER
COMMON_TEL
COMPANY_TEL
CELL_PHONE
EMAIL
ADDRESS
HOUSEHOLD_ADDR
COMMON_ADDR
目的XML綱要(B)
root
concept (1.0)
birth
birthday
concept (1.0)
common
comn
concept (1.0)
company
corp
concept (1.0)
id
no
concept (1.0)
mobile
cell
common_addr (relation)
comn_addr_zip
comn_addr
household_addr (relation)
reg_addr_zip
reg_addr
name (relation)
first_name
last_name
本體資訊(O)

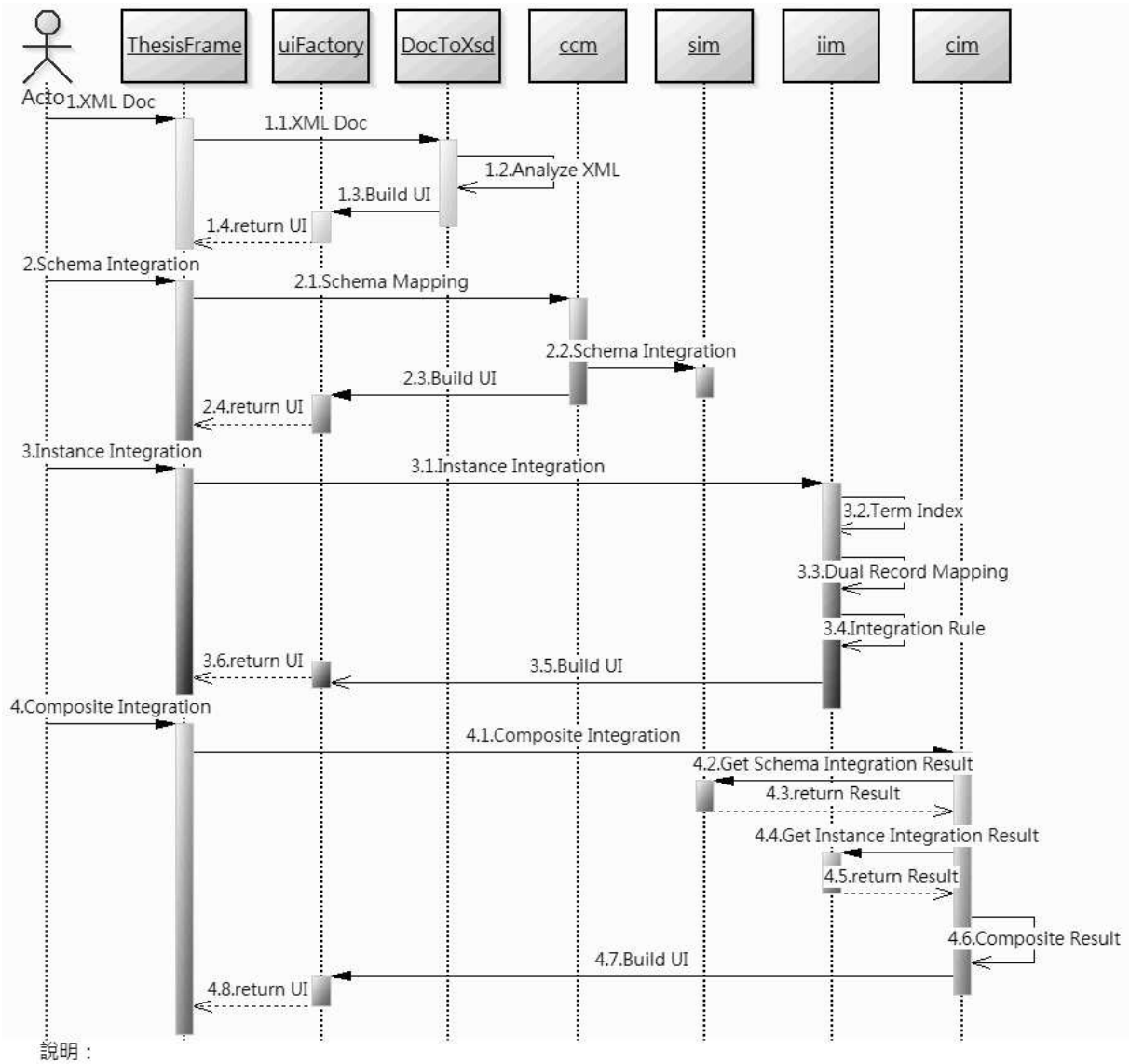
圖 4.3: 綱要實驗資訊

```xml
<?xml version="1.0" encoding="BIG5"?>
<CONTACTLIST>
 <CONTACT>
  <NO>G229624928</NO>
  <NAME>林O羽 </NAME>
  <BIRTHDAY>1972/02/24</BIRTHDAY>
  <SEX>F</SEX>
  <TELEPHONE>
   <COMN_TEL>0879207969</COMN_TEL>
   <CORP_TEL>0929178149</CORP_TEL>
   <MOBILE>0939266836</MOBILE>
  </TELEPHONE>
<EMAIL  > </EMAIL>
  <ADDRESS>
   <REG_ADDR_ZIP>900</REG_ADDR_ZIP>
   <REG_ADDR> 屏東縣屏東市大連里53鄰德豐街176號 </REG_ADDR>
   <COMN_ADDR_ZIP>900</COMN_ADDR_ZIP>
   <COMN_ADDR> 屏東縣屏東市大連里5鄰德豐街120號 </COMN_ADDR>
  </ADDRESS>
 </CONTACT>
 <CONTACT>
  <NO>G226386802</NO>
  <NAME>曾O婷 </NAME>
  <BIRTHDAY>1976/10/28</BIRTHDAY>
```

圖 4.4: 實例實驗資訊 - 來源 XML 實例 (A)

```xml
<?xml version="1.0" encoding="BIG5"?>
<CUSTINFOLIST>
 <CUSTINFO>
  <ID>G226386802</ID>
  <NAME>
   <FIRST_NAME>O婷 </FIRST_NAME>
   <LAST_NAME>曾</LAST_NAME>
  </NAME>
  <BIRTH>1976/10/28</BIRTH>
  <SEX>F</SEX>
  <PHONENUMBER>
   <COMMON_TEL>0879591618</COMMON_TEL>
   <COMPANY_TEL>0559363599</COMPANY_TEL>
   <CELL_PHONE>092243138</CELL_PHONE>
  </PHONENUMBER>
  <EMAIL>XXXXXXXXXX@YAHOO.COM.TW</EMAIL>
  <ADDRESS>
   <HOUSEHOLD_ADDR>912 雲林縣古坑鄉東和村78鄰文化路120號之5 </HOUSEHOLD_ADDR>
   <COMMON_ADDR>912 雲林縣古坑鄉東和村43鄰文化路14號之2 </COMMON_ADDR>
  </ADDRESS>
 </CUSTINFO>
 <CUSTINFO>
  <ID>G234724915</ID>
  <NAME>
   <FIRST_NAME>O玉  </FIRST_NAME>
   <LAST_NAME>潘</LAST_NAME>
  </NAME>
```

圖 4.5: 實例實驗資訊 - 目的 XML 實例 (B)



說明：

ThesisFrame：使用者介面
DocToXsd：XML文件剖析
ccm：XML綱要映射
sim：XML綱要整合
uiFactory：顯示節點資訊
iim：XML實例整合
cim：合併綱要及實例整合的結果

圖 4.6: 實作程式循序圖

XML 綱要和 XML 實例，以備後續整合程序時使用。接著可選擇整合的類型分為 "Schema-Based"、"Instance-Based"、"Composite-Based"，各整合類型於下列說明：

a Schema-Based：將來源 XML 綱要及目的 XML 綱要利用概念名稱及參考知識本體定義計算出每個節點的相似度。將相似度高於門檻值的節點記錄至整合表中。

b Instance-Based：利用 Dual-Record Detect 程序，找出相似度較高的來源 XML 實例與目的 XML 實例配對，並將各配對的實例資料拆解出最小字詞後進行相似度比對。利用各實例對應群之間的交集關係，將相似的節點整合起來。

c Composite-Based：透過合併 XML 綱要及 XML 實例的整合規則，增加節點整合的效能。

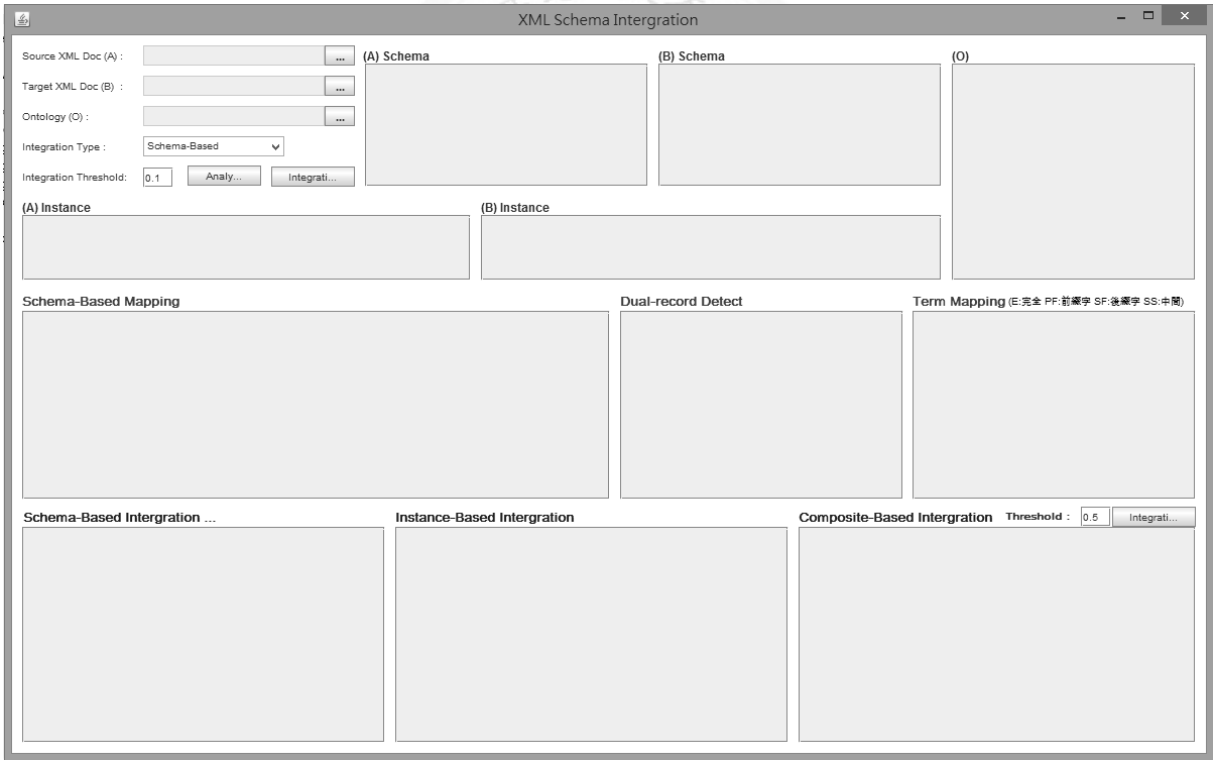


圖 4.7: 使用者介面

2. XML 文件剖析：利用 org.w3c.dom.Document 載入 XML 文件，從 root 節點開始記錄各 XML 節點的結構及名稱。然後以 org.w3c.dom.Node 取出 XML 文件中的資料，並依序記錄每筆實例資料（圖：如圖： 4.8）。

3. XML 綱要映射：將來源 XML 綱要和目的 XML 綱要中的節點，放至 Matrix 中進行比較 (圖： 4.9)。舉例說明：來源 XML 綱要 ”name” 與目的 XML 綱要 ”first_name” 進行映射比對時，利用分詞程序產生新的節點映射字詞 ”name”：”first”、”name”。所以 ”name” 需分別比對 ”first” 以及 ”name” 兩個字詞，最後取出最高得分的相似度再除以字元數以取得平均相似度。

4. XML 綱要整合：將映射結果高於門檻值的 XML 綱要節點記錄至整合池中。以圖： 4.9為例，實例整合規則表示 ”id”:”id”，”name”:”first_name”、”last_name”，”teleephone”:”areacode”、”phonenumber”。

5. XML 實例整合：主要是利用來源實例與目的實例進行比對，本研究中此方法只包含元素層級的映對 [10]，而不考慮綱要資訊，所以映對方法的處理對象是資料內容而非元素名稱。先將目的 XML 實例分詞後透過 Apache Lucene 進行字詞索引；再將來源 XML 實例中的資料，利用同樣的機制進行分詞後將其組合成查詢條件，利用 TF-IDF 及餘弦函式 (cosine) 至索引庫中取出相似度最高的配對實例。再比較配對實例內的各節點資料得到節點間的相似度，最後整合彼此相似的節點資料。

6. 合併綱要及實例整合的結果：合併 XML 實例以及 XML 綱要整合規則中的相似度後，再依照各整合規則合併後的相似度降冪排序。依序記錄至 Composite 整合規則中，進行記錄前，若整合規則中已重覆存在時，則該筆記錄應予以放棄 (圖： 4.11)。

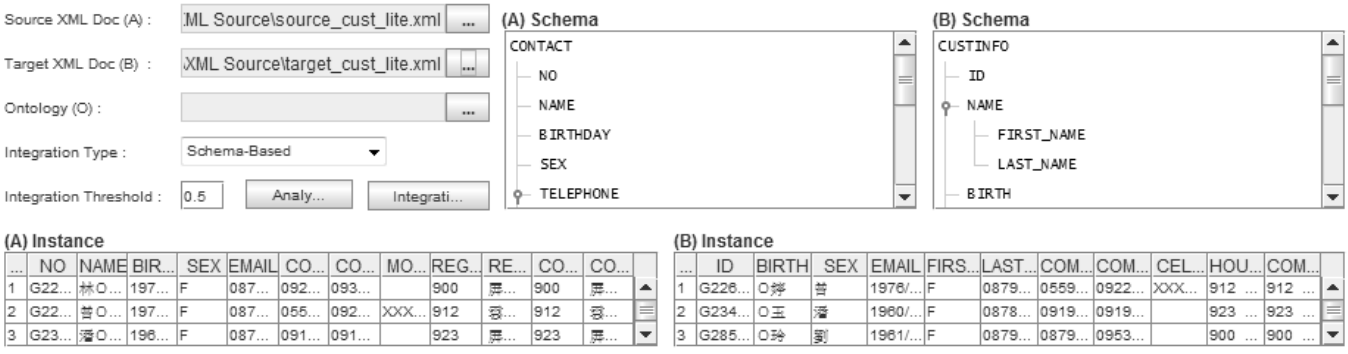


圖 4.8: XML 文件剖析

Schema-Based Mapping

(B) / (A)	no	name	birthday	sex	comn...	corp_...	mobile	email	reg_...	reg_a...	co...	co...
id	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
first_name	0.0	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
last_name	0.0	0.5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
birth	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
sex	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
common_tel	0.0	0.0	0.0	0.0	1.0	0.5	0.0	0.0	0.0	0.0	0....	0.5
company_tel	0.0	0.0	0.0	0.0	0.5	1.0	0.0	0.0	0.0	0.0	0.0	0.0
cell_phone	0.0	0.0	0.0	0.0	0.0	0.0	0.5	0.0	0.0	0.0	0.0	0.0
email	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
household_addr	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.3...	0.5	0....	0.5
common_addr	0.0	0.0	0.0	0.0	0.5	0.0	0.0	0.0	0.3	0.5	0	1.0

Schema-Based Intergration ...

(I)Element	(S)Path	(T)Path
id	/CONTACT/NO	/CUSTINFO/ID
common_addr	/CONTACT/TELEPHONE/C...	/CUSTINFO/ADDRESS/CO...
common_addr	/CONTACT/ADDRESS/COM...	/CUSTINFO/ADDRESS/CO...
common_addr	/CONTACT/ADDRESS/COM...	/CUSTINFO/ADDRESS/CO...
household_addr	/CONTACT/ADDRESS/COM...	/CUSTINFO/ADDRESS/HO...
household_addr	/CONTACT/ADDRESS/REG...	/CUSTINFO/ADDRESS/HO...
email	/CONTACT/EMAIL	/CUSTINFO/EMAIL
cell_phone	/CONTACT/TELEPHONE/M...	/CUSTINFO/PHONENUMBE...
company_tel	/CONTACT/TELEPHONE/C...	/CUSTINFO/PHONENUMBE...
company_tel	/CONTACT/TELEPHONE/C...	/CUSTINFO/PHONENUMBE...
common_tel	/CONTACT/ADDRESS/COM...	/CUSTINFO/PHONENUMBE...
common_tel	/CONTACT/TELEPHONE/C...	/CUSTINFO/PHONENUMBE...

圖 4.9: XML 綱要映射及 XML 綱要整合

4.3 實作結果

對於同一組實驗資料分別進行綱要資訊和實例資訊進行綱要整合測試；將綱要資訊整合測驗中的實驗門檻值設定為 0.1 至 1.0，以檢驗門檻值變動時的差異性。除了以單一綱要整合進行測試(包含名稱與關聯相似度)，另會僅依名稱資訊(未參考本體定義)進行整合，探討本體資訊的覆蓋度對於綱要整合的影響。再將實例資訊進行整合測試，其門檻值設定為 0.1 至 1.0。檢驗門檻值變動時，整合的精確率及完整率的關係。最後再測試本研究建議的混合整合方法，與單一整合方法的差異性。圖： 4.12是依綱要資訊進行整合測試時，實驗資料在參考及未參考本體資訊進行測驗的精確率及完整率。當門檻值是 0.4、0.5、0.6、0.7 時，未使用本體資訊整合的精確率分別為 75%、75%、100% 與 100%，完

Dual-record Detect

RowNum(A)	RowNum(B)	threshold
1	9	0.014675053
10	9	5.7347593
11	10	9.113893
12	11	7.401042
13	12	6.3083057
14	13	6.699916
15	14	6.0283566
16	15	8.800984
17	16	8.016961
18	17	5.845556

Term Mapping (E:完全 PF:前綴字 SF:後綴字 SS:中間)

Row:Node(A)	Row:Node(...	Type	threshold
1:COMN_AD...	9:COMMON...	PF	0.1875
1:COMN_AD...	9:HOUSEHO...	PF	0.1875
1:COMN_AD...	9:COMMON...	PF	0.1875
1:COMN_AD...	9:HOUSEHO...	PF	0.1875
1:REG_ADD...	9:COMMON...	PF	0.1875
1:REG_ADD...	9:HOUSEHO...	PF	0.1875
1:REG_ADD...	9:COMMON...	PF	0.1875
1:REG_ADD...	9:HOUSEHO...	PF	0.1875
1:SEX:0	9:SEX:0	E	1
1:SEX:0	9:SEX:0	E	1

Instance-Based Intergration

(I)Element	(S)Path	(T)Path
*CELL_PHONE	/CONTACT/TELEPHONE/MOBILE	/CUSTINFO/PHONENUMBER/...
*ID	/CONTACT/NO	/CUSTINFO/ID
*COMMON_TEL	/CONTACT/TELEPHONE/COM...	/CUSTINFO/PHONENUMBER/...
*SEX	/CONTACT/SEX	/CUSTINFO/SEX
*COMPANY_TEL	/CONTACT/TELEPHONE/COR...	/CUSTINFO/PHONENUMBER/...
*BIRTH	/CONTACT/BIRTHDAY	/CUSTINFO/BIRTH
FIRST_NAME	/CONTACT/NAME	/CUSTINFO/NAME/FIRST_NAME
*COMMON_ADDR	/CONTACT/ADDRESS/COMN_...	/CUSTINFO/ADDRESS/COMM...
*HOUSEHOLD_A...	/CONTACT/ADDRESS/REG_A...	/CUSTINFO/ADDRESS/HOUSE...

圖 4.10: XML 實例整合

整率分別為 46%、46%、15% 與 15%。使用本體資訊整合的精確率分別為 65%、65%、100% 與 100%，完整率分別為 85%、85%、62% 與 54%。從圖： 4.12得到不論是否使用本體資訊進行綱要整合，其精確率與門檻值成正比，完整率與門檻值成反比。進行綱要整合時，若參考本體資訊則可有效的提升其完整率。

當門檻值設定為 0.7 時，其依綱要資訊整合的結果分析如下：

1. 相同屬性名稱的整合，對於來源及目的 XML 綱要的葉節點名稱，採用屬性名稱完全相同的比較方法予以處理。所以來源綱要及目的綱要可整合的節點有 ”email” 與 ”sex”。

2. 概念 (Concept) 的整合，透過本體資訊 (O) 的定義，可將相同概念但不同命名的節點名稱進行整合，例如：”id” 與 ”no” 透過映射的過程中，可以映對

Composite-Based Intergration Threshold : 0.5 Integrati...

(I)Element	(S)Path	(T)Path
BIRTH	/CONTACT/BIRTHDAY	/CUSTINFO/BIRTH
CELL_PHONE	/CONTACT/TELEPHONE/MOBI...	/CUSTINFO/PHONENUMBER/...
COMMON_ADDR	/CONTACT/ADDRESS/COMN_...	/CUSTINFO/ADDRESS/COMM...
common_addr	/CONTACT/ADDRESS/COMN_...	/CUSTINFO/ADDRESS/COMM...
COMMON_TEL	/CONTACT/TELEPHONE/COM...	/CUSTINFO/PHONENUMBER/...
COMPANY_TEL	/CONTACT/TELEPHONE/COR...	/CUSTINFO/PHONENUMBER/...
email	/CONTACT/EMAIL	/CUSTINFO/EMAIL
FIRST_NAME	/CONTACT/NAME	/CUSTINFO/NAME/FIRST_NA...
HOUSEHOLD_A...	/CONTACT/ADDRESS/REG_A...	/CUSTINFO/ADDRESS/HOUS...
household_addr	/CONTACT/ADDRESS/REG_A...	/CUSTINFO/ADDRESS/HOUS...
ID	/CONTACT/NO	/CUSTINFO/ID
last_name	/CONTACT/NAME	/CUSTINFO/NAME/LAST_NAME

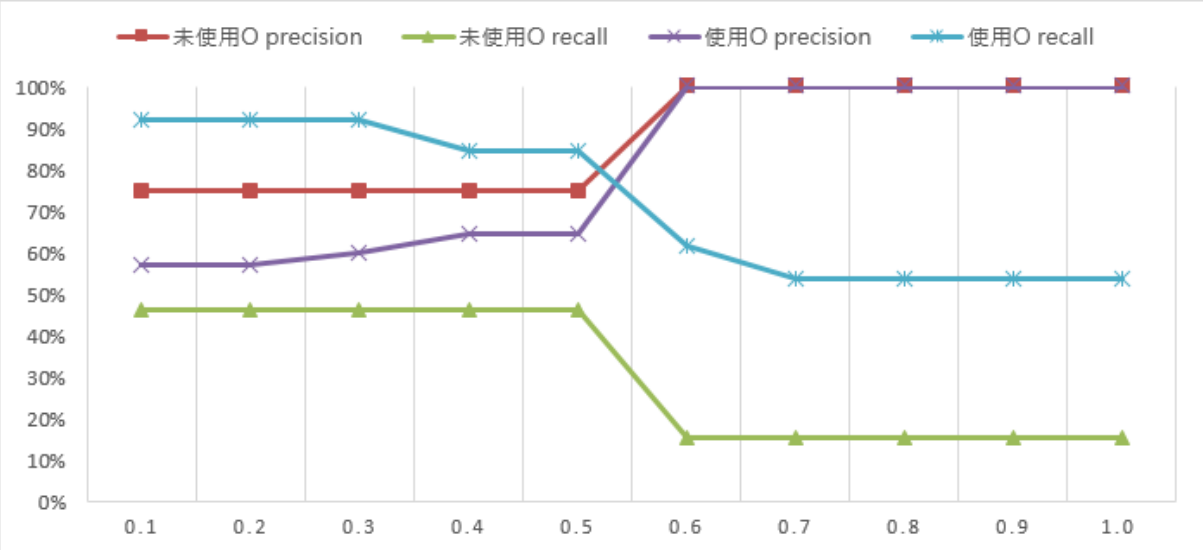
圖 4.11: 合併綱要及實例整合的結果

到來源綱要的"CUSTINFO/ID" 以及目的綱要的"CONTACT/NO"。並且由於在本體論中設定其相似度為"1"，所以可以納入整合規則中。對於這些同義異詞的節點名稱進行整合時，就不會被視為不相同而不予處理。

3. 關聯 (Relation) 的整合，統一將中的聚合關係 (Aggregation) 以及組合 (Composition) 關係記錄至本體資訊 (O) 中，可提供綱要映射時使用。例如：本體資訊 (O) 定義了"common_addr" 與"comn_addr_zip"、"comn_addr"，所以會得到"COMMON_ADDR" 與"COMN_ADDR"、"COMN_ADDR_ZIP" 的整合規則。

綱要資訊整合未參考本體資訊進行整合時，由於僅依靠節點名稱進行完全相同的比較。在門檻值低於 0.5 時，可得到較高的精確率，但完整率仍然是偏低的狀況，隨著門檻值的提高，精確率提高至 100%，但完整率降低至 15%，可整合的節點有"email" 與"sex"。第二次依綱要資訊整合加入了本體資訊的參考，因為多了概念以及關聯的整合方式，可以比較出較多的整合規則。所以在門檻值低於 0.5 時，其精確率較低，但完整率甚至可達 100%。接著持續調高門檻值，精確率提升至 100%，而完整率降低至 54%。由實驗結果可得知，利用綱要資訊進行整合時，綱要節點的資訊不足以進行映射與整合。所以需要參考較多的外部資訊，才能提高整合的精確率及完整率。

如圖：4.13進行依實例資訊整合測試，當門檻值設定為 0.1 時，得到精確率 87% 完整率 100%，隨著門檻值的提高完整度則持續降低。由於本研究在進行整合前，先進行了相同記錄配對的程序，所以進行整合時, 能得到較高的精確率。3 由於實例資訊的數量與質量並不一定，所以無法有效地利用實例資訊

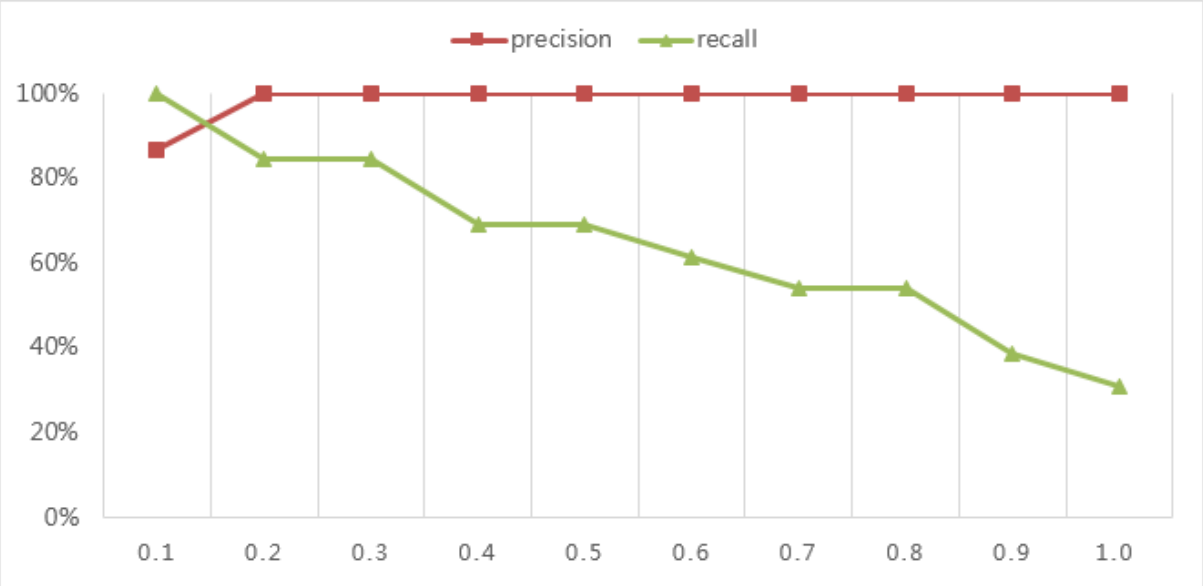


本體資訊	門檻值	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
未使用O	precision	75%	75%	75%	75%	75%	100%	100%	100%	100%	100%
	recall	46%	46%	46%	46%	46%	15%	15%	15%	15%	15%
使用O	precision	57%	57%	60%	65%	65%	100%	100%	100%	100%	100%
	recall	92%	92%	92%	85%	85%	62%	54%	54%	54%	54%

圖 4.12: 利用綱要資訊進行整合時，門檻值不同而造成精確率與完整率的變化

進行整合。例如：依實例資訊比較完後，目的綱要的”HOUSEHOLD_ADDR”(戶籍地址)應當與來源綱要的”REG_ADDR_ZIP”、”REG_ADDR” 進行整合。但因為許多實例的戶籍地址與通訊地址是一樣的，所以相同本實驗在門檻值為0.1 時，會另外得到與來源綱要的”COMN_ADDR_ZIP”、”COMN_ADDR”(通訊地址)整合規則。而”REG_ADDR_ZIP” 與”COMN_ADDR_ZIP” 在實例中常被省略，造成在整合時其相似度較低，所以當門檻值提高至 0.2 時，”REG_ADDR_ZIP” 與”COMN_ADDR_ZIP” 被過濾掉了。

由實驗結果可得知，利用實例資訊進行整合時，其精確率及完整率取決於資料內容的完整度以及單一性。但由於實例資料的取得本來就是不完整的，而且具有重覆的特性，所以整合前需要運用較多的時間進行資料清理及記錄配對。本研究方法將”B” 方法(依綱要資訊)與”C” 方法(依實例資訊)的結果進行相似度整合，從圖： 4.14可得知本研究方法在各門檻值設定之下，其完整率均高於”B” 方法與”C” 方法。雖然”C” 方法的精確率可達到 100%，但門檻值在 0.5 以下時，因為”B” 方法使用了本體資訊進行映射，產生了較多不可整合的資訊，進而影響了本研究方法的精確率。 由圖： 4.15可看出在門檻值 0.5 的條件下”B” 方法與”C” 方法的精確率與完整率是 65%、85% 與 100%、69%。因門檻值過於低下，整合結果內有較多不可整合的規則，如：”B” 方法內因本體資訊中設定”com-



門檻值	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
precision	87%	100%	100%	100%	100%	100%	100%	100%	100%	100%
recall	100%	85%	85%	69%	69%	62%	54%	54%	38%	31%

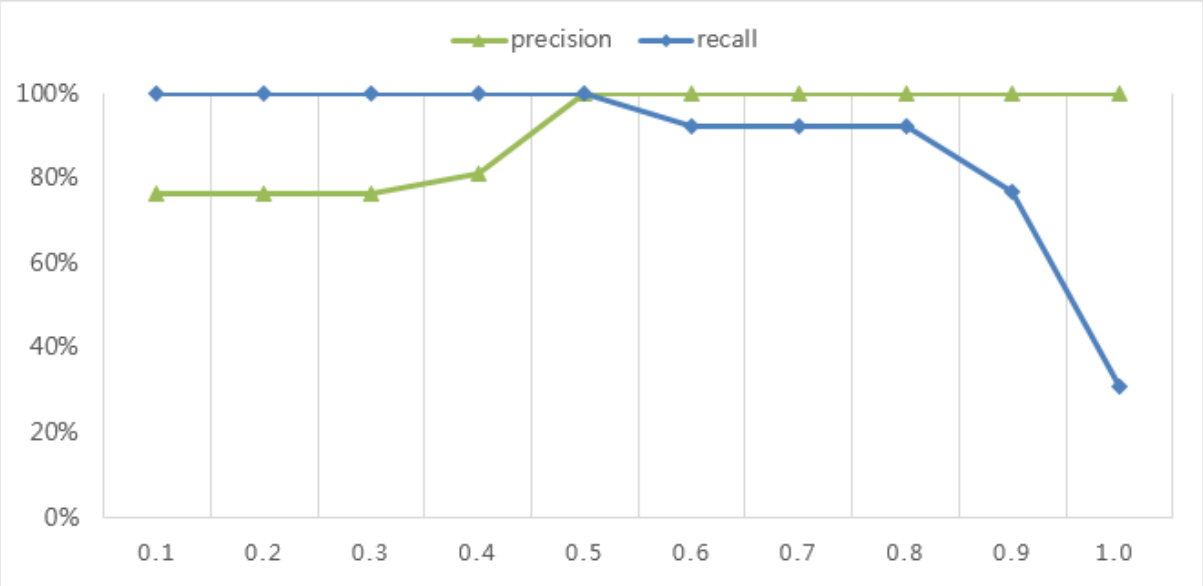
圖 4.13: 利用實例資訊進行整合時，門檻值不同而造成精確率與完整率的變化

mon” 與 ”comn” 的概念 (Concept)，而判定 ”comn_addr” 與 ”common_tel” 可整合。而 ”C” 方法則因為實例資訊內的 ”mobile” 與 ”common_tel” 有重覆出現的資料，而產生可進行整合的規則。本研究方法可在不同門檻值的設定下，有效的利用 ”B” 方法與 ”C” 方法的資訊，例如：用 ”B” 方法中 ”REG_ADDR_ZIP”、”HOUSEHOLD_ADDR” 與 ”COMN_ADDR_ZIP”、”COMMON_ADDR” 的整合規則，判別出 ”C” 方法中 ”COMN_ADDR_ZIP” 與 ”HOUSEHOLD_ADDR”、”REG_ADDR_ZIP” 與 ”COMMON_ADDR” 錯誤的整合資訊。

4.4 結果分析

門檻值為 0.5 條件下圖： 4.16分別是 ”B” 方法、”C” 方法、本研究方法整合結果的精確率與完整率。

1. 使用綱要資訊的狀況下，來源綱要與目的綱要可整合的節點有 13 個。在未使用本體資訊的狀況下，對映出 8 個可整合的節點，未對映的數量有 7 個，而可整合的節點中有 2 個是錯的。在使用本體資訊後，對映出 17 個可整合的節點，未對映的數量有 2 個，而可整合的節點中有 6 個是錯的。”B” 方法在使用本體資訊時，應會比未使用本體資訊的實驗有更高的精確率，但由於本體資訊中設計的概念 (Concept) 資訊，與綱要中有太多的關



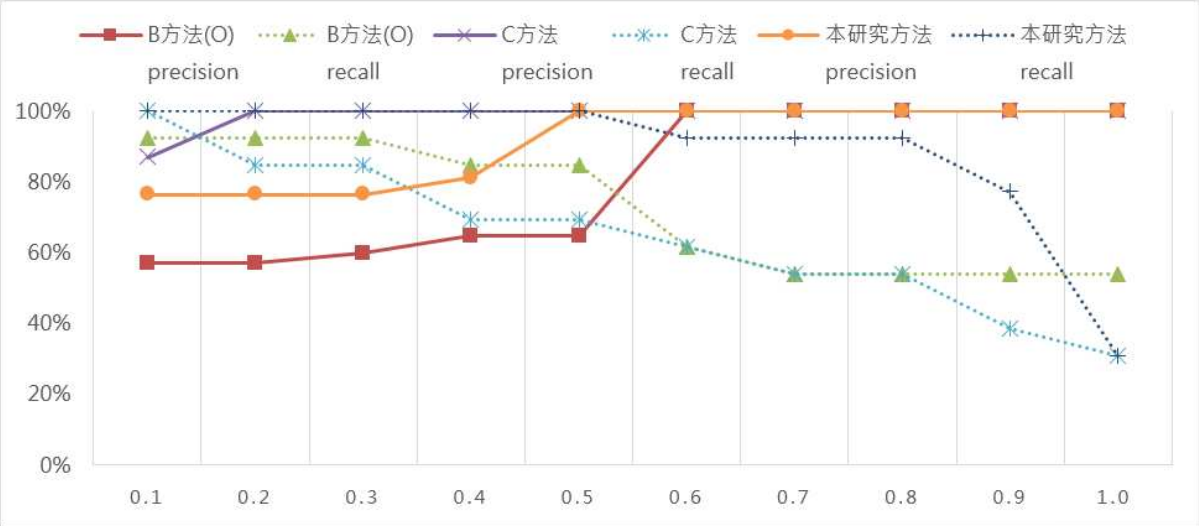
門檻值	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
precision	76%	76%	76%	81%	100%	100%	100%	100%	100%	100%
recall	100%	100%	100%	100%	100%	92%	92%	92%	77%	31%

圖 4.14: 彙整綱要資訊與實例資訊進行整合時，門檻值不同而造成精確率與完整率的變化

聯（例如:"common" 與 "comn" 的概念，而判定 "comn_addr" 與 "common_tel" 的整合），而造成了誤判的狀況。若想得到較高的完整率，雖然可透過調整本體資訊來達成，但會造成過度依賴使用者經驗，而失去了利用系統自動化的原意。

2. 使用實例資訊的狀況下，對映出 9 個可整合的節點，未對映的數量有 0 個，可整合的節點中有 4 個錯的。由於進行整合測試前先行進行資料清理與篩選，在門檻值 0.2 條件下即可達到 100% 精確率。但實例資訊具有高重覆性的特性（例如：" 行動電話號碼 " 與 " 常用電話號碼 " 的資料常是相似的），所以隨著門檻值的提升，而造成完整率降低。實驗結果經檢測後，發現實例資料並非是單一且唯一的存在，會產生小範圍的對映錯誤，而降低了節點的相似度，並影響了整合過程中的判斷。所以需參考其它的相關資訊，以提高其整合的完整率。

3. 本研究方式合併了綱要資訊及實例資訊的整合結果，對映出 13 個可整合的節點，均為正確可整合的節點。由此可證明利用多個相似度測量的方法



門檻值	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
B方法(O) precision	57%	57%	60%	65%	65%	100%	100%	100%	100%	100%
B方法(O) recall	92%	92%	92%	85%	85%	62%	54%	54%	54%	54%
C方法 precision	87%	100%	100%	100%	100%	100%	100%	100%	100%	100%
C方法 recall	100%	85%	85%	69%	69%	62%	54%	54%	38%	31%
本研究方法 precision	76%	76%	76%	81%	100%	100%	100%	100%	100%	100%
本研究方法 recall	100%	100%	100%	100%	100%	92%	92%	92%	77%	31%

圖 4.15: "B" 方法、"C" 方法與本研究方法精確率與完整率的比較

除了可增加彈性外，對於提高精確率與完整率也能有不錯的表現。若僅使用單一的相似度測量方法，將不足以有效的求得屬性間的相似度，需將這些測量方法適當的整合，才能有效的整合綱要屬性。

B方法	TP	FP	FN	TP+FP	TP+FN	精確率TP/(TP+FP)	完整率TP/(TP+FN)
未使用O	6	2	7	8	13	75%	46%
使用O	11	6	2	17	13	65%	85%

C方法	TP	FP	FN	TP+FP	TP+FN	精確率TP/(TP+FP)	完整率TP/(TP+FN)
	9	0	4	9	13	100%	69%

本研究方法	TP	FP	FN	TP+FP	TP+FN	精確率TP/(TP+FP)	完整率TP/(TP+FN)
	13	0	0	13	13	100%	100%

圖 4.16: 門檻值為 0.5- 各方法的精確率與完整率

5. 結論及未來研究方向

自 W3C 在 1998 年發佈 XML 1.0 之後，因其可利用網際網路傳輸、文件的高延伸性、緊密的安全機制及支援多語系等等的優點。逐漸成為網頁資料的呈現和資料轉換的基準，目前已是網際網路發展的重要趨勢之一。在辨識和發掘相異綱要間對應的需求日漸增加，所以在綱要整合的應用方向等研究議題，值得大量投入研究，能更增進在網際網路上發展的價值。本節將針對本文所提出的方法及其效益整理成結論，並提出本文後續可再加以衍生之研究議題與方向。

5.1 結論

本研究延續先前的綱要整合研究方法 [3]，導入參考實例資訊推導不同來源的 XML 綱要元素的整合關係，對照先前研究方法的整合結果，本文提議同時使用綱要資訊及實例資訊進行綱要整合的可行性。利用 XML 文件本身具有的綱要資訊與實例資訊，使用餘弦函式配對相同的記錄 [27]，並使用綱要間的概念關係來解決可能遭遇到的節點名稱對應的問題。藉由取得實例資訊的相似度，擷取屬性名稱的特徵，再使用綱要資訊相似度來取得屬性間的關係，故不需事先的訓練。在系統自動化對應屬性上，也能有令人滿意的效能，整體來說運用實例資訊與綱要資訊進行節點整合是確實有幫助的。

實例整合模式 (Instance-based)，是利用資料內容進行比對，只包含元素層級的映對。以先前研究為例：語意映對及限制條件映對方法，這類映對方式較強調元素內容的意義 [10]，而不考慮綱要資訊，所以映對方法的處理對象是資料內容而非元素名稱，主要的優點是可以精確評估元素內容值的特性，最適合用在半結構化或非結構化的資料，因為這種資料的綱要資訊較少，如果綱要資訊有錯誤也可以被發現，遇到模稜兩可的映對情況時，例如：綱要 A 的 NAME 元素應該對應到綱要 B 的 FIRST_NAME 元素還是 LAST_NAME 元素？這時比對元素名稱可能無法直接判斷，若再比對元素內容則可清楚判斷。甚至有些沒有綱要的資料也可使用此種方式建立綱要並產生映對。但只利用實例整合模式也有缺陷之處，例如：實例資訊常因為資料的共用性，造成資料重置的狀況出現，例如：XML 文件 A 的 COMN_TEL 資料會同時出現在 XML 文件 B 的 COMMON_TEL 及 CELL_PHONE 資料中。本研究同時使用了綱要資訊以及實例資訊避免了使

用單一整合方法而造成比對時的誤差，由圖： 4.12與圖： 4.13可看出，使用單一綱要整合的方法時，雖然精確率會隨著門檻值提高而呈現正向的成長，但是完整率則成呈反向的下降。利用本研究方法同時參考綱要資訊與實例資訊進行比較時，設定門檻值為 0.5 時，即可有效的取得 100% 的精確率與 100% 的完整率。對於實驗資料中的 1:N 與 N:1 的綱要資料，綱要整合模式 (Schema-based) 可以運用本體資訊中各概念之間的關係，實例整合模式 (Instance-based) 則可以運用記錄配對的檢索詞條功能找出可整合的規則，最後本研究彙整兩種整合方式，降低了比對時的誤差及取得較高的精確率與完整率。

5.2 未來研究方向

綜合以上各章節之討論，在未來研究方面，可以朝以下幾個方向持續作進一步的提升及深入探討，以期提升 XML 綱要整合之實用價值。

1. 本研究在實例資料分析著重的部份是節點資料中的全部或部份內容 (Content) 的相等，但在實務應用上，依附於此實例之資料本身，也是有相當的異質性，甚至是充滿歧義的。例如：國籍"ROC" 代表的是" Republic of China" 或" Republic of Chad" 的縮寫，即便都是同一種資料，那麼仁愛路「一段」與仁愛路「1 段」是相同的嗎？所以依實例資料整合前，先對原始資料的「內容」進行分析，即可避免上述的窘境。

2. 進行配對記錄的比對時，本研究採用類似搜尋引擎的資料搜尋方式，進行資料的辨識。未來可以嘗試利用其它領域的的配對方法，例如：常使用於 DNA 序列比對的生物資訊演算法 Longest Common Subsequence，應該能更有效率的找出可以配對的記錄。

3. 增強對於綱要定義的分析，可提升 XML 文件所描述的語意與實務結合。例如：在定義基金的成交價金時，除了投資金額外，再加上手續費部份，會定義為成交價金＝投資金額＋銷售機構手續費拆帳＋基金公司手續費拆帳，或每個月的信管費提撥，會定義為信管費＝投資金額 * 0.1%。實務上常有類似這方面的定義，若能強化這部份，必能提升和實務的結合。

參考文獻

[1] iThome. 2010 台灣雲端元年 ,it 支出成長 4.3%, 01 2010.

[2] 連志誠教授 . 延伸標記語言 *XML* 新引 . 儒林 , 2002.

[3] 葉美惠 . XML 綱要中複雜型態的整合之研究 . 東吳大學資訊管理學系碩士論文 , 2011.

[4] Jacob Berlin and Amihai Motro. Autoplex: Automated discovery of content for virtual databases. In *Proceedings of the 9th International Conference on Cooperative Information Systems*, CooplS '01, pages 108–122, London, UK, UK, 2001. Springer-Verlag.

[5] Alexander Bilke and Felix Naumann. Schema matching using duplicates. In Karl Aberer, Michael J. Franklin, and Shojiro Nishio, editors, *ICDE*, pages 69–80. IEEE Computer Society, 2005.

[6] Y. Bishr. Overcoming the semantic and other barriers to gis interoperability. *International Journal of Geographical Information Science*, 12:229 – 314, 1998.

[7] Serafini L. Bouquet, P. and S. Zanobini. Semantic coordination: A new approach and an application. *Sanibel Island, Florida, USA,*, page 2003., 2003.

[8] Silvana Castano, Alfio Ferrara, Stefano Montanelli, and Gaia Varese. Knowledge-driven multimedia information extraction and ontology evolution. In Georgios Paliouras, Constantine D. Spyropoulos, and George Tsatsaronis, editors, *Knowledge-Driven Multimedia Information Extraction and Ontology Evolution*, chapter Ontology and instance matching, pages 167–195. Springer-Verlag, Berlin, Heidelberg, 2011.

[9] Su-Mei Huang Chih-Cheng Lien. An improved ontology-guided approach for xml schema mapping. In *Submitted to Department of Computer and Information Science*, 2006.

[10] Cecil Eng H. Chua, Roger H. L. Chiang, and Ee-Peng Lim. Instance-based attribute identification in database integration. *The VLDB Journal*, 12(3):228–243, October 2003.

[11] Robin Dhamankar, Yoonkyong Lee, AnHai Doan, Alon Halevy, and Pedro Domingos. imap: discovering complex semantic matches between database schemas. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, SIGMOD '04, pages 383–394, New York, NY, USA, 2004. ACM.

[12] Hong-Hai Do and Erhard Rahm. Coma: a system for flexible combination of schema matching approaches. In *Proceedings of the 28th international conference on Very Large Data Bases*, VLDB '02, pages 610–621. VLDB Endowment, 2002.

[13] AnHai Doan, Pedro Domingos, and Alon Y. Halevy. Reconciling schemas of disparate data sources: a machine-learning approach. In *Proceedings of the 2001 ACM SIGMOD international conference on Management of data*, SIGMOD '01, pages 509–520, New York, NY, USA, 2001. ACM.

[14] Hazem T. El-Khatib, M. Howard Williams, Lachlan M. MacKinnon, and David H. Marwick. A framework and test-suite for assessing approaches to resolving heterogeneity in distributed databases. *Information & Software Technology*, 42(7):505–515, 2000.

[15] William B. Frakes and Ricardo Baeza-Yates. *Information Retrieval: Data Structures and Algorithms*. Prentice Hall, facsimile edition, 1992.

[16] T. R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquistion, 5(2)*, 10:199–220, 1993.

[17] J. Hendler. Agents and the semantic web. *IEEE Intelligent Systems16(2)*, 20:30–37, 2001.

[18] K. Seluk Candan Maria Luisa Sapino Huiping Cao, Yan Qi. Xml data integration: Schema extraction and mapping. 2010.

[19] Henning Kohler, Xiaofang Zhou, Shazia Wasim Sadiq, Yanfeng Shu, and Kerry L. Taylor. Sampling dirty data for matching attributes. In Ahmed K. Elmagarmid and Divyakant Agrawal, editors, *SIGMOD Conference*, pages 63–74. ACM, 2010.

[20] Guofeng Liu, Shaobin Huang, and Yuan Cheng. Research on semantic integration across heterogeneous data sources in grid. In Sabo Sambath and Egui Zhu, editors, *ICFCE*, volume 133 of *Advances in Soft Computing*, pages 397–404. Springer, 2011.

[21] W. Fan C.H. Goh S.E. Madnick Lu, H. and D.W. Cheung. Discovering and reconciling value conflicts for data integration. *Information Systems*, 26:635–656, 2001.

[22] R. J. Miller, Y. E. Ioannidis, and R. Ramakrishnan. The use of information capacity in schema integration and translation. In *In VLDB*, pages 120–133, 1993.

[23] Fabian Panse, Maurice van Keulen, Ander de Keijzer, and Norbert Ritter. Duplicate detection in probabilistic data. In *ICDE Workshops*, pages 179–182. IEEE, 2010.

[24] Jin Pei, Jun Hong, and David Bell. A novel clustering-based approach to schema matching. In *Proceedings of the 4th international conference on Advances in Information Systems*, ADVIS'06, pages 60–69, Berlin, Heidelberg, 2006. Springer-Verlag.

[25] Eric Peukert, Julian Eberius, and Erhard Rahm. Rule-based construction of matching processes. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, CIKM '11, pages 2421–2424, New York, NY, USA, 2011. ACM.

[26] Erhard Rahm and Philip A. Bernstein. A survey of approaches to automatic schema matching. *VLDB JOURNAL*, 10:2001, 2001.

[27] Gerard Salton and Michael J. McGill. *Introduction to modern information retrieval*. McGraw-Hill, New York, 1983.

[28] Ozgul Unal and Hamideh Afsarmanesh. Schema matching and integration for data sharing among collaborating organizations. *JSW*, 4(3):248–261, 2009.

[29] Wikipedia. Stages of growth model, 01 1979.

[30] H. Xiao and I.F. Cruz. Integrating and exchanging xml data using ontologies. *Springer Journal on Data Semantics VI*, 4090:67–89, 2006.

[31] Huang B. Yi, S. and W.T. Chan. Xml application schema matching using similarity measure and relaxation labeling. *Information Sciences*, 169:27–46, 2005.

[32] Huimin Zhao and Sudha Ram. Combining schema and instance information for integrating heterogeneous data sources. *Data Knowl. Eng.*, 61(2):281–303, May 2007.

[33] Huimin Zhou and Sudha Ram. Clustering schema elements for semantic integration of heterogeneous data sources. *J. Database Manag.*, 15(4):88–106, 2004.

附錄

附錄一：XML 文件解析程式

說明：將使用者輸入 XML 文件進行解晰，程式利用 org.w3c.dom.Document 取出 XML 綱要資訊及實例資訊，作為後續整合時的依據。程式中第 50-82 行，由 root 節點開始往下搜尋為 ELEMENT_NODE 的節點型態，將該節點的資料存入資料表格中。第 94-124 行，將 XML Schema 剖析出的葉節點資訊，包括綱要名稱、節點名稱、父節點等資訊存入資料表格。

```
1  package com.yin.model;
2
3  import java.io.*;
4  import java.util.*;
5  import java.sql.*;
6  import javax.xml.parsers.DocumentBuilder;
7  import javax.xml.parsers.DocumentBuilderFactory;
8  import javax.xml.parsers.ParserConfigurationException;
9  import org.w3c.dom.Node;
10 import org.w3c.dom.NodeList;
11 import com.yin.utility.ConnFactory;
12
13 public class DocToXsd {
14        private Connection con;
15        private static Statement state;
16        private org.w3c.dom.Document document;
17        private static String sql = "";
18        private static String tname = "";
19        private static int eloc = 0;
20        private static StringBuffer xmlValue = new StringBuffer();
21        private static LinkedList<StringBuffer> xmlInstanceList = new
              LinkedList<StringBuffer>();
22
23        public void trans(org.w3c.dom.Document document, String fileName)
              throws IOException, Exception {
24            tname = fileName;
25            ConnFactory cf = new ConnFactory();
26            con = cf.getConnection();
27            try {
28                   state = con.createStatement();
29                   sql = "DELETE from tdom where tname='" + tname + "'";
30                   state.executeUpdate(sql);
31                   sql = "DELETE from tInstanceterm where instanceSn in " +
32                                "( " +
33                                "    select sn from tinstance where tname =
                                         '" + tname + "' " +
34                                ")";
```

```
35                      state.executeUpdate(sql);
36                      sql = "DELETE from tInstance where tname='" + tname + "'"
                            ;
37                      state.executeUpdate(sql);
38              } catch (SQLException e1) {
39                      e1.printStackTrace();
40              }
41
42              Node root = document.getDocumentElement();
43              getXmlSchema(root, 1, "");
44              xmlInstanceList = new LinkedList<StringBuffer>();
45              getXmlInstance(root, 1);
46      }
47      private static void getXmlInstance(Node node, int level) throws
            Exception {
48              if(node.hasChildNodes()){
49                      NodeList nodes = node.getChildNodes();
50                      for (int i = 0; i < nodes.getLength(); i++) {
51                              int type = nodes.item(i).getNodeType();
52                              if(type== Node.ELEMENT_NODE){
53                                      String name = nodes.item(i).getNodeName();
54                                      String val = "";
55                                      if(nodes.item(i).getFirstChild() != null){
56                                              val = nodes.item(i).getFirstChild()
                                                    .getNodeValue();
57                                      }
58                                      val = val.replaceAll("\\r\\n|\\r|\\n", "");
59                                      if((val!=null && val.trim().length() > 0)
                                            || (val.equals(" ")) ){
60                                              if(xmlValue.length() > 0 ){
61                                                      xmlValue.append(",");
62                                              }
63                                              xmlValue.append( name + ":" + val);
64                                              //xmlValue.append( val);
65                                      }
66                                      getXmlInstance(nodes.item(i), level + 1);
67                              }
68                              if( level == 2 && i == nodes.getLength()-1) {
69                                      if(xmlValue != null){
70                                              xmlInstanceList.addFirst(xmlValue);
71                                              try {
72                                                      sql = "INSERT INTO tInstance
                                                            (tname, record_no,
                                                            Instance_data)" +
73                                                      "VALUES('"+tname+"', "+
                                                            xmlInstanceList.size()+"
                                                            , '"+xmlValue+"')";
74                                                      state.executeUpdate(sql);
75                                                      eloc ++;
76                                              } catch (SQLException e1) {
77                                                      e1.printStackTrace();
```

```
78                                          }
79                                          xmlValue = new StringBuffer();
80                                  }
81                          }
82                  }
83          }
84      }
85      static String xmlPath = "";
86      private static void getXmlSchema(Node node, int level, String
            parentXmlPath) throws Exception {
87          String parentNodeName;
88          String xmlPath = "" ;
89          if(level==1){
90                  parentNodeName = "root";
91          }else{
92                  parentNodeName = node.getNodeName();
93          }
94          if(node.hasChildNodes()){
95                  NodeList nodes = node.getChildNodes();
96                  for (int i = 0; i < nodes.getLength(); i++) {
97                          int type = nodes.item(i).getNodeType();
98                          if(type== Node.ELEMENT_NODE){
99                                  String nodeName = nodes.item(i).getNodeName
                                        ();
100                                 String leaf = "";
101                                 if (nodes.item(i).getChildNodes().getLength
                                        () == 1){
102                                         leaf = "Y";
103                                 }else{
104                                         leaf = "N";
105                                 }
106                                 try {
107                                         xmlPath = parentXmlPath +"/"+
                                                nodeName;
108                                         sql = "INSERT INTO tdom(tname,
                                                ename, eparent, elev, eloc,
                                                isleaf, path)" +
109                                         "VALUES('"+tname+"', '"+nodeName+"'
                                                , '"+parentNodeName+"', "+
                                                String.valueOf(level-1)+", "+
                                                eloc+", '"+leaf+"', '"+xmlPath+"
                                                ')";
110                                         state.executeUpdate(sql);
111                                         eloc ++;
112                                 } catch (SQLException e1) {
113                                         e1.printStackTrace();
114                                 }
115
116                                 getXmlSchema(nodes.item(i), level + 1,
                                        xmlPath);
117                         }
```

```
118
119                             if(level == 1 && i == 1){
120                                     break;
121                             }
122
123                     }
124             }
125     }
126     public static String padRight(String s, int n) {
127          return String.format("%1$-" + n + "s", s);
128     }
129
130 }
```

附錄二：本體資訊比對程式

說明：將解析得到來源 XML 綱要及目的 XML 綱要的節點資訊，依照本體資訊求得節點間的相似度。程式中第 34-47 行及第 50-63 行，分別是取出來源及目的 XML 綱要資訊。第 74-89 行，分別將來源 XML 綱要中的節點依序與目的 XML 綱要中的節點進行比較，並求得其在本體資訊中的相似度。第 99-129 行，先利用本體資訊進行節點間關聯性的比較，若節點間無關聯性；則將來源節點名稱拆解為單一字元，分別與目的節點拆解後的單一字元進行比較。比較後單一字元若相同，則給予相似度 1，否則利用本體資訊進行單一字元間概念性的比較，依序將單一字元求得的相似度加總。最後將加總後的相似度除以最大字數，回傳兩節點的最大相似度。第 131-145 行，依照本體資訊中的關聯性計算兩節點的相似度。第 146-159 行，依照本體資訊中的概念性計算兩節點的相似度。第 161-175 行，將節點名稱拆解為單一字元。

```
1  package com.yin.model;
2
3  import java.io.*;
4  import java.sql.Connection;
5  import java.sql.ResultSet;
6  import java.sql.SQLException;
7  import java.sql.Statement;
8  import java.util.LinkedList;
9  import com.yin.utility.ConnFactory;
10
11 public class ConceptComparisonModules {
12         Connection con;
13         Statement state;
14         {
15                 try {
16                         ConnFactory cf = new ConnFactory();
17                         con = cf.getConnection();
18                         state = con.createStatement();
19                 } catch (SQLException e) {
20                         e.printStackTrace();
21                 };
22         }
23         private ResultSet resultSet = null;
24         private static LinkedList<String[]> sourceXmlSchemaList = new
               LinkedList<String[]>();
25         private static LinkedList<String[]> targetXmlSchemaList = new
               LinkedList<String[]>();
26         private String[] xmlSchema = new String[3];
```

```
27      public void matcher(String sourceFileName, String targetFileName,
            String choiceontology) {
28          resultSet = null;
29          sourceXmlSchemaList = new LinkedList<String[]>();
30          targetXmlSchemaList = new LinkedList<String[]>();
31          xmlSchema = new String[3];
32
33          //Source Schema Data Init
34          try {
35                  String sql = "SELECT lower(ename) as ename, path FROM
                        tdom WHERE tname='" + sourceFileName
36                              + "' AND isleaf = 'Y' ";
37                  resultSet = state.executeQuery(sql);
38                  while (resultSet.next()) {
39                          xmlSchema = new String[3];
40                          xmlSchema[0] = resultSet.getString("ename");
41                          xmlSchema[1] = resultSet.getString("path");
42                          xmlSchema[2] = splitData(xmlSchema[0]);
43                          sourceXmlSchemaList.addLast(xmlSchema);
44                  }
45          } catch (SQLException e) {
46                  e.printStackTrace();
47          }
48
49          //Target Schema Data Init
50          try {
51                  String sql = "SELECT lower(ename) as ename, path FROM
                        tdom WHERE tname='" + targetFileName
52                              + "' AND isleaf = 'Y' ";
53                  resultSet = state.executeQuery(sql);
54                  while (resultSet.next()) {
55                          xmlSchema = new String[3];
56                          xmlSchema[0] = resultSet.getString("ename");
57                          xmlSchema[1] = resultSet.getString("path");
58                          xmlSchema[2] = splitData(xmlSchema[0]);
59                          targetXmlSchemaList.addLast(xmlSchema);
60                  }
61          } catch (SQLException e) {
62                  e.printStackTrace();
63          }
64          //node match
65          String[] sourceNode, targetNode;
66          String sql ="";
67          try {
68                  sql = "delete from mtable where type = 'schema' and x1 =
                        '"+sourceFileName+"' and x2 = '"+targetFileName+"'";
69                  state.executeUpdate(sql);
70          } catch (SQLException e1) {
71                  e1.printStackTrace();
72          }
73
```

```
74              for(int s = 0 ; s < sourceXmlSchemaList.size() ; s++ ){
75                      //get source node to match target node
76                      sourceNode = sourceXmlSchemaList.get(s);
77                      for(int t = 0 ; t < targetXmlSchemaList.size() ; t++ ){
78                              targetNode = targetXmlSchemaList.get(t);
79                              //count Similarity
80                              double nodeSimilarity = countNodeSimilarity(
                                    sourceNode,targetNode);
81                              try {
82                                      sql = "insert into mtable (type, x1, x1name
                                            , x1eparent, x2, x2name, x2eparent,
                                            threshold)" +
83                                      "VALUES('schema', '"+sourceFileName+"' , '"
                                            +sourceNode[0]+"', '"+sourceNode[1]+"',
                                             '"+targetFileName+"', '"+targetNode
                                            [0]+"', '"+targetNode[1]+"', "+
                                            nodeSimilarity+")";
84                                      state.executeUpdate(sql);
85                              } catch (SQLException e1) {
86                                      e1.printStackTrace();
87                              }
88                      }
89              }
90
91
92      }
93      //count source & target similarity
94      public double countNodeSimilarity(String[] sourceNode, String[]
            targetNode) {
95          double wordSimilarity = 0, NS = 0, maxSimilary = 0;
96          double relationWordSimilarity = 0;
97          String sourceWords[] = sourceNode[2].split(",");
98          String targetWords[] = targetNode[2].split(",");
99          relationWordSimilarity = getOntologyRelationSimilarity(
                sourceNode[0],targetNode[0]);
100         if(relationWordSimilarity > 0){
101                 NS = relationWordSimilarity;
102         }else{
103                 for(int i = 0 ; i < sourceWords.length ; i++ ){
104                         maxSimilary = 0;
105                         int o_match = 0;
106                         for(int j = 0 ; j < targetWords.length ; j++ ){
107                                 if(sourceWords[i].equals(targetWords[j])){
108                                         //source word = target word then
                                                Similarity = 1
109                                         wordSimilarity = 1;
110                                 }else{
111                                         //source word = target word in
                                                Ontology then Similarity =
                                                Ontology Similarity
112                                         wordSimilarity =
```

```
                                        getOntologyConceptSimilarity(
                                        sourceWords[i],targetWords[j]);
113                                 }
114                                 //source 內的nodeword 依序"" 至target 內的
                                        進行比較，僅記錄相似值最大
                                        的nodewordsimilary
115                                 if(wordSimilarity > maxSimilary){
116                                         maxSimilary = wordSimilarity;
117                                 }
118                         }
119                         NS = NS + maxSimilary;
120                 }
121                 int maxWordLength = 0;
122                 if(sourceWords.length >= targetWords.length){
123                         maxWordLength = sourceWords.length;
124                 }else{
125                         maxWordLength = targetWords.length;
126                 }
127                 NS = NS / maxWordLength;
128         }
129         return NS;
130     }
131     public double getOntologyRelationSimilarity(String sourceWords, String
            targetWords) {
132         double relationWordSimilarity = 0;
133         try {
134                 String sql = "select 1.0 as similar "+
135                                 "from otable where type = 'relation
                                        ' and ((lower(name1) = '"+
                                        sourceWords+"' and lower(name2)
                                        ='"+targetWords+"')" +
136                                 " or (lower(name2) = '"+sourceWords
                                        +"' and lower(name1) ='"+
                                        targetWords+"'))";
137                 resultSet = state.executeQuery(sql);
138                 if(resultSet.next()){
139                         relationWordSimilarity = resultSet.getDouble("
                                    similar");
140                 }
141         } catch (SQLException e) {
142                 e.printStackTrace();
143         }
144         return relationWordSimilarity;
145     }
146     public double getOntologyConceptSimilarity(String sourceWord, String
            targetWord) {
147         double conceptWordSimilarity = 0;
148         try {
149                 String sql = "select max(similar) as similar from otable
                            where type = 'concept' and "+
150                                 "((lower(name1) = '"+sourceWord+"' and
```

```
                                        lower(name2) ='"+targetWord+"') or (
                                        lower(name1) = '"+targetWord+"' and
                                        lower(name2) = '"+sourceWord+"'))";
151                     resultSet = state.executeQuery(sql);
152                     resultSet.next();
153                     conceptWordSimilarity = resultSet.getDouble("similar");
154
155             } catch (SQLException e) {
156                     e.printStackTrace();
157             }
158             return conceptWordSimilarity;
159     }
160     //split node name
161     public String splitData(String nodeName) {
162             char[] name_array = nodeName.toCharArray();
163             StringBuffer newSplitData = new StringBuffer();
164     for(int index=0; index < name_array.length; index++){
165       if(Character.isLetterOrDigit(name_array[index])) {
166               newSplitData.append(name_array[index]);
167       }
168       else{
169               if(newSplitData != null && newSplitData.length() > 0){
170                       newSplitData.append(',');
171               }
172       }
173     }
174             return newSplitData.toString();
175     }
176 }
```

附錄三：依 XML 綱要進行綱要整合程式

說明：將映射相似度高於門檻值的節點，記錄至資料庫中

```
1  package com.yin.model;
2
3  import java.sql.Connection;
4  import java.sql.ResultSet;
5  import java.sql.SQLException;
6  import java.sql.Statement;
7  import com.yin.utility.ConnFactory;
8
9  public class SchemaIntegrationModules {
10         Connection con;
11         Statement state;
12         Statement opState;
13         {
14                 try {
15                         ConnFactory cf = new ConnFactory();
16                         con = cf.getConnection();
17                         state = con.createStatement();
18                         opState = con.createStatement();
19                 } catch (SQLException e) {
20                         e.printStackTrace();
21                 };
22         }
23         private ResultSet resultSet = null;
24
25         public void integration(String sourceFileName, String targetFileName,
               float threshold) {
26                 resultSet = null;
27                 String ename;
28
29                 //get target node name
30                 try {
31                         String sql = "delete from itable " +
32                                         "where x1 = '"+sourceFileName+"' and x2 = '
                                            "+ targetFileName +"' "+
33                                         "and type = 'schema' " ;
34                         opState.executeUpdate(sql);
35
36                         sql = "select tname, ename, etype1, etype2, etype3,
                               eparent, elev, eloc, isleaf, path " +
37                                                 "from tdom " +
38                                                 "where tname = '"+
                                                    targetFileName +"' " +
39                                                 "and isleaf = 'Y' " ;
40                         resultSet = state.executeQuery(sql);
```

```
41                      while (resultSet.next()) {
42                              ename = resultSet.getString("ename");
43                              sql = "INSERT INTO itable(type, x1, x1name,
                                    x1eparent, x2, x2name, x2eparent, threshold)"
                                    +
44                              "select type, x1, x1name, x1eparent, x2, x2name,
                                    x2eparent, threshold " +
45                              "from mtable " +
46                              "where x1 = '"+sourceFileName+"' and x2 = '"+
                                    targetFileName +"' "+
47                              "and type = 'schema' " +
48                              "and threshold >= " + threshold + " " +
49                              "and x2name = '" + ename + "' " +
50                              "order by threshold desc " ;
51                              opState.executeUpdate(sql);
52                      }
53              } catch (SQLException e) {
54                      e.printStackTrace();
55              }
56      }
57 }
```

附錄四：依 XML 實例進行綱要整合程式

說明：取出可配對的實例資料，將節點中的內容切割為單一字元進行比較，求得各節點中字詞的相似度。取出配對相似度最高的節點，並記錄至整合規則中。程式中第 33-125 行，比較配對的實例資料，記錄其節點內容的相似度。第 126-256 行，將所有可整合的節點資料記錄至資料表格中。第 257-384 行，依照目的 XML 綱要節點，分別取出最高相似度的來源綱要節點，並記錄至整合表格中。

```
1  package com.yin.model;
2
3  import java.sql.Connection;
4  import java.sql.ResultSet;
5  import java.sql.SQLException;
6  import java.sql.Statement;
7  import com.yin.utility.ConnFactory;
8
9  public class InstanceIntegrationModules {
10         Connection con;
11         Statement state;
12         Statement sourceState;
13         Statement targetState;
14         Statement extState;
15         Statement opState;
16         {
17                 try {
18                         ConnFactory cf = new ConnFactory();
19                         con = cf.getConnection();
20                         state = con.createStatement();
21                         sourceState = con.createStatement();
22                         targetState = con.createStatement();
23                         extState = con.createStatement();
24                         opState = con.createStatement();
25                 } catch (SQLException e) {
26                         e.printStackTrace();
27                 };
28         }
29         private ResultSet resultSet = null;
30         private ResultSet sourceResultSet = null;
31         private ResultSet targetResultSet = null;
32         private ResultSet extResultSet = null;
33         public void integration(String sourceFileName, String targetFileName) {
34                 resultSet = null;
35                 int source_sn, target_sn, sourceTermSn, targetTermSn;
36                 String sourceNodeName, sourceTerm;
```

```
37              String targetNodeName, targetTerm;
38              String mappingType="";
39              float threshold = 0;
40              int sourceSeq, targetSeq;
41
42              //get target node name
43              try {
44                      String sql = "delete from itable " +
45                                      "where x1 = '"+sourceFileName+"' and x2 = '
                                            "+ targetFileName +"' "+
46                                      "and type = 'instance' " ;
47                      opState.executeUpdate(sql);
48
49                      sql = "delete from mtermmapping " ;
50                      opState.executeUpdate(sql);
51
52                      sql ="select a.x1, a.x1name, b.sn as source_sn, b.
                            Instance_data as source_instance, "+
53                              "    a.x2, a.x2name, c.sn as target_sn, c.
                                    Instance_data as target_instance "+
54                                      "from mtable a join tinstance b on a.x1 = b
                                            .tname and a.x1name = b.record_no
                                            "+
55                                      "                              join tinstance c
                                            on a.x2 = c.tname and a.x2name = c.
                                            record_no "+
56                                      "where a.type = 'instance'

                                            "+
57                                      "and a.x1 = '"+sourceFileName+"'

                                            "+
58                                      "and a.x2 = '"+ targetFileName +"'

                                            ";
59
60                      resultSet = state.executeQuery(sql);
61                      //source & target instancce pre-integration
62                      while (resultSet.next()) {
63                              source_sn = resultSet.getInt("source_sn");
64                              target_sn = resultSet.getInt("target_sn");
65                              //get source instance term
66                              sql ="select sn, cname, seq, term "+
67                                              "from tInstanceTerm    "+
68                                              "where instanceSn = "+source_sn+" "
                                                    +
69                                              "and refType = 's' and term != ''
                                                    ";
70                              sourceResultSet = sourceState.executeQuery(sql);
71                              while (sourceResultSet.next()) {
72                                      sourceTermSn = sourceResultSet.getInt("sn")
```

```
    ;
73  sourceNodeName = sourceResultSet.getString(
        "cname");
74  sourceTerm = sourceResultSet.getString("
        term");
75  sourceSeq = sourceResultSet.getInt("seq");
76  //search dual term
77  sql ="select sn, cname, seq, term "+
78                  "from tInstanceTerm    "+
79                  "where instanceSn = "+
                        target_sn+" "+
80                  "and refType = 't' and term
                        !='' ";
81  targetResultSet = targetState.executeQuery(
        sql);
82  while (targetResultSet.next()) {
83          targetTermSn = targetResultSet.
                getInt("sn");
84          targetNodeName = targetResultSet.
                getString("cname");
85          targetTerm = targetResultSet.
                getString("term");
86          targetSeq = targetResultSet.getInt(
                "seq");
87
88          mappingType="";
89          threshold = 0;
90          //判斷符合的條件
91          //E完全相同: PF前綴字相同: SF後綴字
                相同: SS中間部份的字相同: A縮寫
                字，開頭字母相同於另一個字:
92          if(sourceTerm.equalsIgnoreCase(
                targetTerm)){
93                  mappingType="E";
94                  threshold = 1;
95          }else{
96                  if(sourceTerm.endsWith(
                        targetTerm) || targetTerm
                        .endsWith(sourceTerm)){
97                          mappingType="SF";
98                  }else if(sourceTerm.
                        startsWith(targetTerm) ||
                         targetTerm.startsWith(
                        sourceTerm)){
99                          mappingType="PF";
100                 }else if(sourceTerm.indexOf(
                        targetTerm)> 0 ||
                        targetTerm.indexOf(
                        sourceTerm)> 0){
101                         mappingType="SS";
102                 }else if(sourceTerm.
```

```
                    replaceAll("[^(A-Z)]", ""
                    ).equals(targetTerm) ||
                    targetTerm.replaceAll("
                    [^(A-Z)]", "").equals(
                    sourceTerm)){
103                     mappingType="A";
104                 }
105                 if( !mappingType.equals(""))
                    {
106                     if(targetTerm.length
                        () < sourceTerm.
                        length()){
107                         threshold = (
                                float)
                                targetTerm
                                .length()
                                / (float)
                                sourceTerm
                                .length();
108                     }else{
109                         threshold = (
                                float)
                                sourceTerm
                                .length()
                                / (float)
                                targetTerm
                                .length();
110                     }
111                 }
112             }
113
114             if(!mappingType.equals("")){
115                 sql = "INSERT INTO
                    mtermmapping(sourceTermSn
                    , targetTermSn,
                    mappingType, threshold) "
                     +
116                         "VALUES("+
                            sourceTermSn
                            +", "+
                            targetTermSn
                            +", '"+
                            mappingType
                            +"', "+
                            threshold+
                            ") ";
117                 opState.executeUpdate(sql);
118             }
119         }
120     }
121 }
```

```
122             } catch (SQLException e) {
123                     e.printStackTrace();
124             }
125     }
126     public void preIntegration(String sourceFileName, String targetFileName
            ) {
127             resultSet = null;
128             int source_sn, target_sn, sourceTermSn, targetTermSn;
129             String sourceNodeName, sourceNodePath, sourceTerm;
130             String targetNodeName, targetNodePath, targetTerm;
131             String mappingType="";
132             float threshold = 0;
133             int sourceSeq, targetSeq;
134
135             //get target node name
136             try {
137                     String sql = "delete from pitable " +
138                                     "where x1 = '"+sourceFileName+"' and x2 = '
                                        "+ targetFileName +"' "+
139                                     "and type = 'pre-instance' " ;
140                     opState.executeUpdate(sql);
141                     //get dual record sn
142                     sql ="select x1name as source_sn, x2name as target_sn
                            from mtable "+
143                             "where type = 'instance' "+
144                             "order by x1name, x2name ";
145
146                     resultSet = state.executeQuery(sql);
147                     int pairs = 1;
148                     while (resultSet.next()) {
149                             source_sn = resultSet.getInt("source_sn");
150                             target_sn = resultSet.getInt("target_sn");
151                             //get source mapping node
152                             sql ="select s.cname as cname, td.path as path "+
153                                     "from mTermMapping a "+
154                                     "join tinstanceterm s on a.sourceTermSn = s
                                        .sn "+
155                                     "join tinstanceterm t on a.targetTermSn = t
                                        .sn "+
156                                     "join tinstance ss on s.instanceSn = ss.sn
                                        "+
157                                     "join tinstance ts on t.instanceSn = ts.sn
                                        "+
158                                     "join tdom td on ss.tname = td.tname and s.
                                        cname = td.ename "+
159                                     "where ss.record_no = "+ source_sn+ " "+
160                                     "and ts.record_no = "+ target_sn+ " "+
161                                     "group by s.cname "+
162                                     "order by s.cname ";
163                             sourceResultSet = sourceState.executeQuery(sql);
164                             while (sourceResultSet.next()) {
```

```
165                     sourceNodeName = sourceResultSet.getString(
                            "cname");
166                     sourceNodePath = sourceResultSet.getString(
                            "path");
167                     //將 完全相同的term 進中insertitable
168
169                     sql = "INSERT INTO pitable(type, x1, x1name
                            , x1eparent, pairs, pairs_sn, x2,
                            x2name, x2eparent, threshold) "+
170                                 "select 'pre-instance', '"+
                                        sourceFileName+"', '"+
                                        sourceNodeName+"', '"+
                                        sourceNodePath+"', "+
171                                 "0, 0, '"+targetFileName+"',
                                         t.cname, td.path, a.
                                        threshold "+
172                                 "from mTermMapping a "+
173                                 "join tinstanceterm s on a.
                                        sourceTermSn = s.sn "+
174                                 "join tinstanceterm t on a.
                                        targetTermSn = t.sn "+
175                                 "join tinstance ss on s.
                                        instanceSn = ss.sn "+
176                                 "join tinstance ts on t.
                                        instanceSn = ts.sn "+
177                                 "join tdom td on ts.tname =
                                        td.tname and t.cname = td
                                        .ename "+
178                                 "where ss.record_no = "+
                                        source_sn+ " "+
179                                 "and ts.record_no = "+
                                        target_sn+ " "+
180                                 "and s.cname = '"+
                                        sourceNodeName+"' "+
181                                 "and a.mappingType = 'E' " ;
182                     opState.executeUpdate(sql);
183                     //依照Source NodeName 取
                            出Target NodeName 的PF前綴資料()，間接
                            將只有SF or 的資料捨去ss
184                     sql ="select t.cname as tn,a.mappingType as
                             mappingType, a.threshold as threshold,
                             td.path as path "+
185                                "from mTermMapping a "+
186                                "join tinstanceterm s on a.
                                       sourceTermSn = s.sn "+
187                                "join tinstanceterm t on a.
                                       targetTermSn = t.sn "+
188                                "join tinstance ss on s.instanceSn
                                       = ss.sn "+
189                                "join tinstance ts on t.instanceSn
                                       = ts.sn "+
```

```
190                     "join tdom td on ts.tname = td.
                            tname and t.cname = td.ename "+
191                     "where ss.record_no = "+ source_sn+
                            " "+
192                     "and ts.record_no = "+ target_sn+ "
                            "+
193                     "and s.cname = '"+sourceNodeName+"'
                            "+
194                     "and a.mappingType = 'PF' ";
195
196                 targetResultSet = targetState.executeQuery(
                        sql);
197                 while (targetResultSet.next()) {
198                     targetNodeName = targetResultSet.
                            getString("tn");
199                     targetNodePath = targetResultSet.
                            getString("path");
200                     mappingType = targetResultSet.
                            getString("mappingType");
201                     threshold = targetResultSet.
                            getFloat("threshold");
202                     //將 pf 的資料存入pitable
203                     sql = "INSERT INTO pitable(type, x1
                            , x1name, x1eparent, pairs,
                            pairs_sn, x2, x2name, x2eparent,
                            threshold) VALUES"+
204                                 "('pre-instance', '"+
                                        sourceFileName+"',
                                         '"+sourceNodeName
                                        +"', '"+
                                        sourceNodePath+"',
                                         "+
205                                 pairs +", 0, '"+
                                        targetFileName+"',
                                         '"+targetNodeName
                                        +"', '"+
                                        targetNodePath+"',
                                         "+threshold+" )"
                                        ;
206                     opState.executeUpdate(sql);
207                     //依照
                            將sourceNodeNametargetNodeName
                            非( PF & E)的資料寫
                            入
208                     sql = "INSERT INTO pitable(type, x1
                            , x1name, x1eparent, pairs,
                            pairs_sn, x2, x2name, x2eparent,
                            threshold) "+
209                                 "select 'pre-instance
                                        ', '"+
                                        sourceFileName+"',
```

```
                                        '"+sourceNodeName
                                    +"', '"+
                                    sourceNodePath+"',
                                     "+
210                                 pairs + ", @rownum :=
                                     @rownum+1 as
                                    rownum , '"+
                                    targetFileName+"',
                                     pair.cname, pair.
                                    path, pair.
                                    threshold FROM "+
211                                 "(select distinct t.
                                    cname, td.path, a.
                                    threshold "+
212                                 "from mTermMapping a
                                    join tinstanceterm
                                     s on a.
                                    sourceTermSn = s.
                                    sn join
                                    tinstanceterm t on
                                     a.targetTermSn =
                                    t.sn "+
213                                 "join tinstance ss on
                                     s.instanceSn = ss
                                    .sn join tinstance
                                     ts on t.
                                    instanceSn = ts.sn
                                     "+
214                                 "join tdom td on ts.
                                    tname = td.tname
                                    and t.cname = td.
                                    ename "+
215                                 "where a.mappingType
                                    != 'PF' AND a.
                                    mappingType != 'E'
                                     "+
216                                 "AND s.cname = '"+
                                    sourceNodeName+"'
                                    "+
217                                 "and t.cname != '"+
                                    targetNodeName+"'
                                    "+
218                                 "and ss.record_no = "
                                    + source_sn+ " "+
219                                 "and ts.record_no = "
                                    + target_sn+ " "+
220                                 "order by a.
                                    mappingType desc)
                                    as pair , (SELECT
                                    @rownum := 0) r ";
221                         opState.executeUpdate(sql);
```

```
222                                     pairs = pairs + 1;
223                                 }
224                             }
225                         }
226                         //取出僅有SF後置資料()沒有PF前置資料()的nodename
227                         sql ="select s.cname as sn, sd.path as spath, t.cname as
                                tn,a.mappingType as mappingType, a.threshold as
                                threshold, td.path as path "+
228                             "from mTermMapping a "+
229                             "join tinstanceterm s on a.sourceTermSn = s.sn "+
230                             "join tinstanceterm t on a.targetTermSn = t.sn "+
231                             "join tinstance ss on s.instanceSn = ss.sn "+
232                             "join tinstance ts on t.instanceSn = ts.sn "+
233                             "join tdom td on ts.tname = td.tname and t.cname =
                                    td.ename "+
234                             "join tdom sd on ss.tname = sd.tname and s.cname =
                                    sd.ename "+
235                             "where (s.cname, t.cname ) not in "+
236                             "(select x.x1name, x.x2name from pitable x) ";
237
238                         extResultSet = extState.executeQuery(sql);
239                         while (extResultSet.next()) {
240                             sourceNodeName = extResultSet.getString("sn");
241                             sourceNodePath = extResultSet.getString("spath");
242                             targetNodeName = extResultSet.getString("tn");
243                             targetNodePath = extResultSet.getString("path");
244                             mappingType = extResultSet.getString("mappingType"
                                    );
245                             threshold = extResultSet.getFloat("threshold");
246                             //將 pf 的資料存入pitable
247                             sql = "INSERT INTO pitable(type, x1, x1name,
                                    x1eparent, pairs, pairs_sn, x2, x2name,
                                    x2eparent, threshold) VALUES"+
248                                             "('pre-instance', '"+sourceFileName
                                                +"', '"+sourceNodeName+"', '"+
                                                sourceNodePath+"', "+
249                                             pairs +", 0, '"+targetFileName+"',
                                                '"+targetNodeName+"', '"+
                                                targetNodePath+"', "+threshold+"
                                                 )" ;
250                             opState.executeUpdate(sql);
251                             pairs = pairs + 1;
252                         }
253                 } catch (SQLException e) {
254                         e.printStackTrace();
255                 }
256         }
257         public void integrationRule(String sourceFileName, String
            targetFileName, float thresholdbase) {
258             resultSet = null;
259             int source_sn, target_sn, sourceTermSn, targetTermSn;
```

```
260             String sourceNodeName, sourceNodePath, sourceTerm;
261             String targetNodeName, targetNodePath, targetTerm;
262             String mappingType="";
263             float threshold = 0;
264             int sourceSeq, targetSeq, rowCount;
265
266             try {
267                     String sql = "delete from itable " +
268                                  "where x1 = '"+sourceFileName+"' and x2 = '
                                        "+ targetFileName +"' "+
269                                  "and type = 'instance' " ;
270                     opState.executeUpdate(sql);
271
272                     //get dual record rowcount
273                     sql ="select count(sn) as rowCount from mtable where type
                          = 'instance' ";
274                     resultSet = state.executeQuery(sql);
275                     resultSet.next();
276                     rowCount = resultSet.getInt("rowCount");
277
278                     //將 完全相同 (E)的資料，新增至中itable
279                     //得出 加總的相似度，需除以 DUAL RECORD 的筆數
280                     sql = "INSERT INTO itable( type, x1, x1name, x1eparent,
                        pairs, pairs_sn, x2, x2name, x2eparent, threshold) "+
281                           "select 'instance', x1, x1name, x1eparent, pairs,
                              pairs_sn, x2, x2name, x2eparent, sum(threshold
                              ) / " + rowCount + " " +
282                           "from pitable "+
283                           "where type = 'pre-instance' "+
284                           "and pairs = 0 "+
285                           "group by x1, x1name, x1eparent, pairs, pairs_sn,
                              x2, x2name, x2eparent "+
286                           "order by pairs_sn ";
287                     opState.executeUpdate(sql);
288
289                     //取出需合併的source node ，由最多節點的開始整合namenode
290                     sql = "select x1name, count(*) "+
291                            "from pitable "+
292                            "where type='pre-instance' "+
293                            "and pairs > 0 "+
294                            "group by x1name "+
295                            "order by count(*) desc, x1name ";
296                     sourceResultSet = sourceState.executeQuery(sql);
297
298                     while (sourceResultSet.next()) {
299                            int mainRow = 0;
300                            //取出擁有最多的放入整合中targetnodeparispool
301                            sourceNodeName = sourceResultSet.getString("x1name
                                ");
302                            sql = "select tot.x1, tot.x1name, tot.x1eparent,
                               tot.pairs, tot.pairs_sn, tot.x2, tot.x2name,
```

```
                    tot.x2eparent, tot.threshold "+
303                     "from pitable tot join ( "+
304                     "  select a.pairs as pairs "+
305                     "    from pitable a "+
306                     "    where a.x1name = '"+sourceNodeName+"' "+
307                     "    group by a.pairs "+
308                     "    order by count(*) desc LIMIT 1 "+
309                     ") st on tot.pairs = st.pairs "+
310                     "where tot.type = 'pre-instance' "+
311                     "order by tot.pairs_sn ";
312
313                 resultSet = state.executeQuery(sql);
314                 resultSet.last();
315                 String mainNode[][] = new String[resultSet.getRow
                        ()][resultSet.getMetaData().getColumnCount()];
316             // Move to beginning
317                 resultSet.beforeFirst();
318                 threshold = 0;
319                 while (resultSet.next()) {
320                     mainNode[mainRow][0] = resultSet.getString(
                            "x1");
321                     mainNode[mainRow][1] = resultSet.getString(
                            "x1name");
322                     mainNode[mainRow][2] = resultSet.getString(
                            "x1eparent");
323                     mainNode[mainRow][3] = resultSet.getString(
                            "pairs");
324                     mainNode[mainRow][4] = resultSet.getString(
                            "pairs_sn");
325                     mainNode[mainRow][5] = resultSet.getString(
                            "x2");
326                     mainNode[mainRow][6] = resultSet.getString(
                            "x2name");
327                     mainNode[mainRow][7] = resultSet.getString(
                            "x2eparent");
328                     mainNode[mainRow][8] = resultSet.getString(
                            "threshold");
329                     threshold = resultSet.getFloat("threshold")
                            ;
330                     //取出包含在 1st 的nodetarget 資料paris
331                     int poolRow = 0;
332                     sql = "select tot.x1, tot.x1name, tot.
                            x1eparent, tot.pairs, tot.pairs_sn, tot
                            .x2, tot.x2name, tot.x2eparent, tot.
                            threshold "+
333                         "from pitable tot "+
334                         "where tot.x2name = '"+mainNode[
                                mainRow][6]+"' "+
335                         "and tot.x1name = '"+mainNode[
                                mainRow][1]+"' "+
```

```
336                 "and tot.pairs in ( "+ //所有包含
                        在source 裡的nodestarget paris
337                 "select distinct tot.pairs "+
338                 "from pitable tot join "+
339                 "( select a.pairs from pitable a
                        where a.pairs != "+mainNode[
                        mainRow][3]+" "+
340                 "and a.x2name in (select x2name
                        from pitable where pairs = "+
                        mainNode[mainRow][3]+" group by
                        x2name) "+
341                 "group by a.pairs having count(*) =
                         (select count(*) from pitable
                        where pairs = a.pairs)) grp on
                        tot.pairs = grp.pairs ) ";
342
343             targetResultSet = targetState.executeQuery(
                    sql);
344             targetResultSet.last();
345             String poolNode[][] = new String[
                    targetResultSet.getRow()][
                    targetResultSet.getMetaData().
                    getColumnCount()];
346             targetResultSet.beforeFirst();
347             while (targetResultSet.next()) {
348                 poolNode[poolRow][0] =
                        targetResultSet.getString("x1");
349                 poolNode[poolRow][1] =
                        targetResultSet.getString("
                        x1name");
350                 poolNode[poolRow][2] =
                        targetResultSet.getString("
                        x1eparent");
351                 poolNode[poolRow][3] =
                        targetResultSet.getString("pairs
                        ");
352                 poolNode[poolRow][4] =
                        targetResultSet.getString("
                        pairs_sn");
353                 poolNode[poolRow][5] =
                        targetResultSet.getString("x2");
354                 poolNode[poolRow][6] =
                        targetResultSet.getString("
                        x2name");
355                 poolNode[poolRow][7] =
                        targetResultSet.getString("
                        x2eparent");
356                 poolNode[poolRow][8] =
                        targetResultSet.getString("
                        threshold");
357                 threshold = threshold +
```

```
                                        targetResultSet.getFloat("
                                        threshold");
358                                     sql = "update pitable set markto =
                                        "+ mainNode[mainRow][3] +" " +
359                                         "where pairs = "+poolNode[
                                            poolRow][3]+" ";
360                                     opState.executeUpdate(sql);
361                                     poolRow++;
362                                 }
363                                 sql = "INSERT INTO itable( type, x1, x1name
                                    , x1eparent, pairs, pairs_sn, x2,
                                    x2name, x2eparent, threshold) "+
364                                     "VALUES('instance','"+mainNode[
                                        mainRow][0]+"','"+mainNode[
                                        mainRow][1]+"','"+mainNode[
                                        mainRow][2]+"',"+mainNode[
                                        mainRow][3]+", "+
365                                     " "+mainNode[mainRow][4]+", '"+
                                        mainNode[mainRow][5]+"', '"+
                                        mainNode[mainRow][6]+"', '"+
                                        mainNode[mainRow][7]+"', "+
                                        threshold+" / "+rowCount+" ) ";
366                                 opState.executeUpdate(sql);
367                                 mainRow++;
368                             }
369                         }
370                         sql = "INSERT INTO itable( type, x1, x1name, x1eparent,
                            pairs, pairs_sn, x2, x2name, x2eparent, threshold) "+
371                                 "select 'instance', x1, x1name, x1eparent,
                                    0, 0, x2, x2name, x2eparent, sum(
                                    threshold) / "+ rowCount +" "+
372                                 "from pitable "+
373                                 "where markto is null "+
374                                 "and pairs not in "+
375                                 "(select distinct pairs from itable where
                                    type = 'instance') "+
376                                 "group by x1, x1name, x1eparent, x2, x2name
                                    , x2eparent ";
377                         opState.executeUpdate(sql);
378
379                         sql = "delete from itable where type = 'instance' and
                            threshold < "+ thresholdbase ;
380                         opState.executeUpdate(sql);
381                 } catch (SQLException e) {
382                         e.printStackTrace();
383                 }
384         }
385
386         public void splitTerm(String sourceFileName, String targetFileName) {
387                 resultSet = null;
388                 //get target node name
```

```
389             try {
390                     String sql = "delete from tInstanceTerm " ;
391                     opState.executeUpdate(sql);
392
393                     sql = "select a.x1, b.sn as source_sn, b.record_no as
                            source_no, b.Instance_data as source_data, "+
394                           "      a.x2, c.sn as target_sn, c.record_no as
                                  target_no, c.Instance_data as target_data " +
395                           "from mtable a left join tinstance b on a.x1 = b
                              .tname and a.x1name = b.record_no " +
396                           "            left join tinstance c on a.x2 = c.
                              tname and a.x2name = c.record_no " +
397                           "where a.type= 'instance' " +
398                           "    and a.x1 = '"+sourceFileName+"' " +
399                           "    and a.x2 = '"+targetFileName+"' " ;
400                     resultSet = state.executeQuery(sql);
401                     while (resultSet.next()) {
402                             String x1, source_data, x2, target_data;
403                             int source_sn, target_sn, source_no, target_no;
404
405                             x1 = resultSet.getString("x1");
406                             source_sn = resultSet.getInt("source_sn");
407                             source_no = resultSet.getInt("source_no");
408                             source_data = resultSet.getString("source_data");
409                             target_sn = resultSet.getInt("target_sn");
410                             target_no = resultSet.getInt("target_no");
411                             target_data = resultSet.getString("target_data");
412
413                             //split source data
414                             saveInstanceTerm(source_data, source_sn, "s" );
415                             //split target data
416                             saveInstanceTerm(target_data, target_sn, "t" );
417                     }
418             } catch (SQLException e) {
419                     e.printStackTrace();
420             }
421     }
422     //split node value
423     public String[] splitNodeValue(String nodeValue) {
424             char[] value_array = nodeValue.replaceAll("-", "").replaceAll("
                    ", "").toCharArray();
425             StringBuffer newSplitData = new StringBuffer();
426     for(int index=0; index < value_array.length; index++){
427       if(!Character.isWhitespace(value_array[index])) {
428               newSplitData.append(value_array[index]);
429       }
430       else{
431               if(newSplitData != null && newSplitData.length() > 0){
432                       newSplitData.append(',');
433               }
434       }
```

```
435         }
436     String[] newSplitValue = newSplitData.toString().split(",");
437                 return newSplitValue;
438         }
439         //save Instance Term
440         public void saveInstanceTerm(String source_data, int sn, String type )
                {
441                 String sql;
442                 if(source_data == null){
443                         return ;
444                 }
445
446                 String[] xmlNode = source_data.split(",");
447                 if(xmlNode.length > 0){
448                         for( int i=0 ; i< xmlNode.length ; i++){
449                                 //[0]:NodeName [1]:NodeValue
450                                 String[] xmlNodeInfo = xmlNode[i].split(":");
451                                 String xmlNodeInfoName = xmlNodeInfo[0];
452                                 String xmlNodeInfoValue = xmlNodeInfo[1];
453                                 String[] splitValue = splitNodeValue(
                                        xmlNodeInfoValue);
454                                 for( int j=0 ; j< splitValue.length ; j++){
455                                         sql = "INSERT INTO tinstanceterm(instanceSn
                                                , refType, cname, seq, term) "+
456                                                 "VALUES("+sn+", '"+type+"', '"+
                                                        xmlNodeInfoName+"', "+j+", '"+
                                                        splitValue[j]+"') ";
457                                         try {
458                                                 opState.executeUpdate(sql);
459                                         } catch (SQLException e) {
460                                                 e.printStackTrace();
461                                         }
462                                 }
463                         }
464                 }
465         }
466 }
```

附錄五：混合 XML 綱要及 XML 實例資訊進行綱要整合程式

說明：依目的 XML 綱要節點，取出彙總依綱要資訊及實例資訊的最高相似度後，記錄至整合表格中。

```
1  package com.yin.model;
2
3  import java.sql.Connection;
4  import java.sql.ResultSet;
5  import java.sql.SQLException;
6  import java.sql.Statement;
7  import com.yin.utility.ConnFactory;
8
9  public class CompositeIntegrationModules {
10         Connection con;
11         Statement state;
12         Statement lowState;
13         Statement opState;
14         {
15                 try {
16                         ConnFactory cf = new ConnFactory();
17                         con = cf.getConnection();
18                         state = con.createStatement();
19                         lowState = con.createStatement();
20                         opState = con.createStatement();
21                 } catch (SQLException e) {
22                         e.printStackTrace();
23                 };
24         }
25         private ResultSet resultSet = null;
26         private ResultSet sourceResultSet = null;
27         private ResultSet lowResultSet = null;
28
29         public void integrationRule(String sourceFileName, String
               targetFileName, float thresholdbase) {
30                 resultSet = null;
31                 int source_sn, target_sn, sourceTermSn, targetTermSn;
32                 String sourceNodeName, sourceNodePath, sourceTerm;
33                 String targetNodeName, targetNodePath, targetTerm;
34                 String x1Name, x2Name;
35                 String mappingType="";
36                 String low_threshold_sn = "";
37                 float threshold = 0;
```

```
38              int sourceSeq, targetSeq, rowCount;
39
40              try {
41                      String sql = "delete from itable " +
42                                      "where x1 = '"+sourceFileName+"' and x2 = '
                                           "+ targetFileName +"' "+
43                                      "and type = 'composite' " ;
44                      opState.executeUpdate(sql);
45                      sql     ="select 'composite',a.x1,a.x1name,a.x1eparent
                            ,0,0,a.x2,a.x2name,a.x2eparent, sum(a.threshold +
                            coalesce(b.threshold,0)) "+
46                          "from itable a left join itable b on b.type = '
                               schema' and a.x1name = b.x1name and a.x2name =
                                b.x2name and b.x1= '"+sourceFileName+"' and b
                               .x2='"+ targetFileName +"' "+
47                          "where a.type = 'instance' "+
48                          "and a.x1='"+sourceFileName+"' "+
49                          "and a.x2='"+ targetFileName +"' "+
50                          "group by a.x1,a.x1name,a.x1eparent,a.x2,a.x2name,
                               a.x2eparent "+
51                          "order by (a.threshold + coalesce(b.threshold,0))
                               desc ";
52
53                      resultSet = state.executeQuery(sql);
54
55                      //將取出的資料存入中，若itablex1name & 已經存在，則跳
                            過x2name
56                      while (resultSet.next()) {
57                          x1Name = resultSet.getString("x1name");
58                          x2Name = resultSet.getString("x2name");
59                          sql ="INSERT INTO itable( type, x1, x1name,
                               x1eparent, pairs, pairs_sn, x2, x2name,
                               x2eparent, threshold) "+
60                              "select 'composite',a.x1,a.x1name,a.
                                   x1eparent,0,0,a.x2,a.x2name,a.x2eparent
                                   , sum(a.threshold + coalesce(b.
                                   threshold,0)) "+
61                              "from itable a left join itable b on b.type
                                    = 'schema' and a.x1name = b.x1name and
                                    a.x2name = b.x2name and b.x1= '"+
                                   sourceFileName+"' and b.x2='"+
                                   targetFileName +"' "+
62                              "where a.type = 'instance' "+
63                              "and a.x1='"+sourceFileName+"' "+
64                              "and a.x2='"+ targetFileName +"' "+
65                              "and a.x1name = '"+x1Name+"' and a.x2name =
                                    '"+x2Name+"' "+
66                              "and (a.x1name, a.x2name) not in ( select
                                   x1name, x2name from itable where type =
                                    'composite') "+
67                              "group by a.x1,a.x1name,a.x1eparent,a.x2,a.
```

```
                        x2name,a.x2eparent "+
68                    "order by (a.threshold + coalesce(b.
                        threshold,0)) desc ";
69            opState.executeUpdate(sql);
70        }
71        //整合規則為基礎，新增無整合規則的綱要規
              則schemainstance
72        sql     ="select 'composite',a.x1,a.x1name,a.x1eparent
              ,0,0,a.x2,a.x2name,a.x2eparent, a.threshold "+
73            "from itable a "+
74            "where a.type = 'schema' "+
75            "and a.x1='"+sourceFileName+"' "+
76            "and a.x2='"+ targetFileName +"' "+
77            "and (a.x1name, a.x2name) not in ( select x1name,
                  x2name from itable where type = 'instance') "+
78            "group by a.x1,a.x1name,a.x1eparent,a.x2,a.x2name,
                  a.x2eparent "+
79            "order by a.threshold desc ";
80
81        resultSet = state.executeQuery(sql);
82
83        //將取出的資料存入中，若itablex1name & 已經存在，則跳
              過x2name
84        while (resultSet.next()) {
85            x1Name = resultSet.getString("x1name");
86            x2Name = resultSet.getString("x2name");
87            sql ="INSERT INTO itable( type, x1, x1name,
                  x1eparent, pairs, pairs_sn, x2, x2name,
                  x2eparent, threshold) "+
88                "select 'composite',a.x1,a.x1name,a.
                      x1eparent,1,0,a.x2,a.x2name,a.x2eparent
                      , a.threshold "+
89                "from itable a "+
90                "where a.type = 'schema' "+
91                "and a.x1='"+sourceFileName+"' "+
92                "and a.x2='"+ targetFileName +"' "+
93                "and a.x1name = '"+x1Name+"' and a.x2name =
                       '"+x2Name+"' "+
94                "and (a.x1name, a.x2name) not in ( select
                      x1name, x2name from itable where type =
                       'composite') "+
95                "group by a.x1,a.x1name,a.x1eparent,a.x2,a.
                      x2name,a.x2eparent "+
96                "order by a.threshold desc ";
97            opState.executeUpdate(sql);
98        }
99
100       //將同，且有一個以上同的資料進行比較，只留下最大相似值
              的整合規則x2namemappingtype
101       sql     ="select x2name , mappingType,count(*) " +
102           "from ( " +
```

```
103                     "    select b.cname as x1name, c.cname as x2name,
                            a.mappingType " +
104                     "     from mtermmapping a join tinstanceterm b on
                            a.sourceTermSn = b.sn join tinstanceterm c on
                            a.targetTermSn = c.sn " +
105                     "    where a.mappingType in ('PF','SF') " +
106                     "    group by b.cname, c.cname, a.mappingType " +
107                     "    order by c.cname, a.mappingType) as a " +
108                     "group by x2name,mappingType " +
109                     "having count(*) > 1 ";
110
111             resultSet = state.executeQuery(sql);
112             while (resultSet.next()) {
113                     x2Name = resultSet.getString("x2name");
114                     mappingType = resultSet.getString("mappingType");
115
116                     sql =  "select a.sn, a.x1name, a.x2name, a.
                            threshold from "+
117                                     "(select distinct a.sn, b.cname as
                                        x1name , c.cname as x2name , a.
                                        threshold "+
118                                     "from itable a join tinstanceterm b
                                         on a.x1name = b.cname "+
119                                     "                       join
                                        tinstanceterm c on a.x2name = c.
                                        cname "+
120                                     "                       join
                                        mtermmapping x on x.sourceTermSn
                                         = b.sn and x.targetTermSn = c.
                                        sn "+
121                                     "where type = 'composite' and
                                        x2name = '" + x2Name + "' and x.
                                        mappingType = '"+mappingType+"'
                                        and a.pairs = 0 "+
122                                     "union "+
123                                     "select distinct a.sn, a.x1name, a.
                                        x2name, a.threshold "+
124                                     "from itable a "+
125                                     "where type = 'composite' and
                                        x2name = '" + x2Name + "' and a.
                                        pairs = 1) as a "+
126                                     "order by a.threshold desc limit
                                        100 offset 1 ";
127                     lowResultSet = lowState.executeQuery(sql);
128                     while (lowResultSet.next()) {
129                             low_threshold_sn = lowResultSet.getString("
                                sn");
130                             sql ="DELETE FROM itable WHERE sn = " +
                                low_threshold_sn;
131                             opState.executeUpdate(sql);
132                     }
```

```
133                         }
134                         sql ="DELETE FROM itable WHERE threshold <= " +
                                thresholdbase;
135                         opState.executeUpdate(sql);
136
137                 } catch (SQLException e) {
138                         e.printStackTrace();
139                 }
140         }
141 }
```