整合關聯式資料庫與 XML 資料之仲介者架構

張雅惠，邱豐傑
國立台灣海洋大學資訊科學系

摘要：

由於電子商務的蓬勃發展與需求，延伸式標記語言 XML 被提出來且發展為資料交換的標準，而成為公司間供應鏈管理的重要一環。但是，另一方面，企業內部的營運資料，還是儲存在擁有高成熟度、高穩定性的關聯式資料庫系統。所以，本篇論文計劃建構一個介於關聯式資料庫及 XML 文件之間的仲介者(mediator)，其中包含一個對應字典，能適當地表示不同型態資料的對應關係，以便將關聯式資料庫資料轉為一份符合供應鏈格式的 XML 文件(例如 RosettaNet XML 文件)，且能將一份複雜的 XML 文件轉為關聯式資料型態，從而達到企業內部資料與供應鍊資料整合與共享的目的。

一・緒論

近年來所謂的電子商務，也就是透過全球資訊網 (World-Wide-Web)從事商業行為，正蓬勃發展中。而在面臨全球化市場競爭的效應下，未來企業間之競爭模式，也已經由個別企業間之競爭擴大成為產業供應鏈體系間之競爭。所以，對組織而言，能適當利用 Web 的便利性來進行供應鏈管理 (Supply Chain Management; SCM)，被視為維持優勢競爭力的關鍵要素。

過去企業的供應鍊管理主要是經由封閉性的加值網路進行電子資料交換(Electronic Data Interchange; EDI)，但是該方法在應用上有許多限制與缺失，例如高固定成本、固定商業流程缺乏彈性、EDI 電子交換機制過於複雜繁瑣等。有鑑於此，在 1998 年美國最大之供應鏈軟體與服務供應商-Ingram Micro，號召 IBM、HP、Intel、Microsoft 等大廠，籌組了新的標準組織 RosettaNet，以因應全球資訊網的興起。

RosettaNet 替供應鍊規劃一套完整的組成架構，並提出一系列詳細的商業夥伴介面流程 (Partner Interface Processes；PIP) 來作為交易夥伴間之標準共通作業程序。由於它基本上是 Web-based，所以它可以透過 HTTP 或 SMTP 以 INTERNET 方式來與各個交易夥伴傳遞訊息，打破封閉性與需龐大費用的限制。另外其最大的優點，在於它所傳送的訊息是架構於 XML 語言上，在訊息的產生、交換、處理以及與後端應用系統整合上，有著更大的彈性及便利性。

XML（Extensible Markup Language）文件如同 HTML 文件，內含特殊的標註 (markup)，但是其特點為使用者可以自訂標註，來說明文件內容的意義，從而便利資料的傳遞與處理。所以，XML 正快速地成為公司間結構性文件（Structured documents）資料交換的標準，不管是與事業夥伴間內部資料的交換，還是在網際網路上透過公開的應用軟體的資料傳送。但是，另一方面，在後端企業內部的營運資料，還是儲存在擁有高成熟度、高穩定性的關聯式資料庫系統，以確保資料與交易(transaction)之安全性。

由於體認到關聯式資料庫與 XML 資料互通與整合的重要性，各資料庫大廠(如 Oracle、IBM、Microsoft 等)，不斷地研究如何新增其所支援的資料，包括 XML 資料格式，以便將一份 XML 文件轉換存入關聯式資料庫、或由關聯式資料庫匯出一份 XML 文件。譬如，目前 IBM 的 DB2 可以將 XML 檔案轉換為資料庫中的資料[12]，而 Microsoft 的 SQL Server 在支援 XML 的功能中有一類似檢視 (view) 的功能，名稱為 XDR (XML-Data Reduced) 的結構描述[17]。但是個個廠商所提供的功能仍然個自有所限制。

Order_Id	Order_ToRegion	Order_ToCountry	Order_ToCity	Order_ToAdd	Order_PostOfficeBox

Order 表格

DO_Id	DO_ShipFrom	DO_ProdId	DO_ProdQuantity	DO_OrderId

DetailOrder 表格

圖一：關聯式資料庫範例

所以本篇論文研究的方向，就在於提供關聯式型態的資料與 XML 資料的交換與整合。我們計劃建構一個介於關聯式資料庫及 XML 資料之間的仲介者(mediator)，此仲介者系統裡，包含了兩種不同型態資料的對應關係，以提供將資料庫資料匯出成為一份複雜的 XML 文件(例如 RosettaNet XML 文件)，及將一份複雜的 XML 文件匯入資料庫中等功能，使企業內部資料可以與供應鍊的資料整合與共享。與現有資料庫所提供的功能相比，我們的仲介者將具有更大的彈性，不受限於資料廠牌的限制。

接下來簡介本篇論文的架構如下：在第二節，我們會提出作為範例資料的 RosettaNet XML 文件與關聯式資料庫的定義，用以說明在關聯式資料庫與 XML 文件之間，存在於資料表示方面的差異性。在第三節則會敘述我們所提出的仲介者系統架構，以及如何適當地表示對應資料，以解決在第二節所列舉出來的問題。最後在第四節討論相關研究，以及在第五節提出本篇論文的結論與未來展望。

二、關聯式資料與 XML 資料之差異性

本節討論以關聯式資料庫儲存資料，與 XML 型態資料之差異性。本論文以供應鏈上下游間的訂單需求作為範例。

圖一的兩個表格是關聯式資料庫裡常見的定義方式。其中表格 Order 表示該訂單的運送資料，其中包含了為訂單編號(Order_Id)、收貨目的地區域 (Order_ToRegionName) 、收貨目的地國家 (Order_ToCountry) 、收貨目的地城市 (Order_ToCity)、收貨目的地地址(Order_ToAdd)、收貨目的郵政信箱(Order_PostOfficeBox)。其中以 Order_Id 為 Order 表格的 primary key。

```
L1：  <DeliveryHeader>
L2：    <messageTrackingID>
L3：      <InstanceIdentifier>200204040001
L4：      </InstanceIdentifier>
L5：    </messageTrackingID>
L6：  </DeliveryHeader>
L7：  <Pip3A4PurchaseOrderRequest>
L8：    <PurchaseOrder>
L9：      <deliverTo>
L10：       <PhysicalAddress>
L11：         <cityName>
L12：           <FreeFormText xml:lang="EN">
L13：             Taipei</FreeFormText>
L14：         </cityName>
L15：         <addressLine1>
L16：           <FreeFormText xml:lang="EN">
L17：             No17 JiLin St.
L18：           </FreeFormText>
L19：         </addressLine1>
L20：         <regionName>
L21：           <FreeFormText xml:lang="EN">
L22：             Asia</FreeFormText>
L23：         </regionName>
L24：         <postOfficeBoxIdentifier>
L25：           <FreeFormText xml:lang="EN">
L26：             20202</FreeFormText>
L27：         </postOfficeBoxIdentifier>
L28：         <GlobalCountryCode>Taiwan
L29：         </GlobalCountryCode>
L30：       </PhysicalAddress>
L31：     </deliverTo>
L32：     <ProductLineItem>
L33：       <shipFrom>
L34：         <GlobalLocationIdentifier>N00015
L35：         </GlobalLocationIdentifier>
L36：       </shipFrom>
L37：       <ProductQuantity>1</ProductQuantity>
L38：       <productUnit>
L39：         <ProductPackageDescription>
L40：           <ProductIdentification>
L41：             <GlobalProductIdentifier> P0001
L42：             </GlobalProductIdentifier>
L43：           </ProductIdentification>
L44：         </ProductPackageDescription>
L45：       </productUnit>
L46：     </ProductLineItem>
L47：   </PurchaseOrder>
L48：  </Pip3A4PurchaseOrderRequest>
```

圖二：RosettaNet 3A4 XML 文件範例

3A4 行數	3A4 資料內容	對應目標欄位
L3	200204040001	Order_Id
L13	Taipei	Order_City
L17	No17 JiLin St.	Order_ToAdd
L22	Asia	Order_ToRegion
L26	20202	Order_PostOfficeBox
L28	Taiwan	Order_ToCountry
L34	N00015	DO_ShipFrom
L37	1	DO_ProdQuantity
L41	P0001	DO_ProdId

圖三：3A4 資料內容與資料庫表格欄位的對應

另一個資料庫表格為 DetailOrder，包含了訂單內詳細的訂購內容，每一筆訂購內容給予一個唯一的編號(DO_Id)，並記錄所購買產品的出貨處(DO_ShipFrom)、購買產品的編號(DO_ProdId)、購買產品的數量(DO_ProdQuantity)。屬性 DO_OrderId 則紀錄該筆訂購的來源訂單，為 DetailOrder 表格的 foregin key，可以用來與 Order 表格的 Order_Id 屬性建立關聯。

圖二則為表示訂單資料的 XML 文件，其格式取自於 RosettaNet3A4 的 XML 文件。RosettaNet 3A4 針對訂單要求定義了相當複雜且完整的 XML 標註，在此我們只摘錄部份以作為說明。在圖二，L2 行至 L5 行的<messageTrackingID>元素表示此筆訂單的流水編號，由 L9 行至 L31 行的<deliverTo>元素則表示貨運的目的地。詳細的訂購資料，包含由 L33 行至 L36 行的<shipFrom>元素，表示所訂購的產品出貨處，另外 L37 行的<ProductQuantity>元素表示所訂購的產品數量，而 L40 至 L43 的<ProductIdentification>元素則表示所訂購的產品編號。為了更清楚表示該份文件所表示的訂單資料，我們利用圖三表示該份 XML 文件的內容值，與圖一資料庫表格欄位的關係。

接下來列舉由 XML 文件對應至資料庫、或由資料庫輸出資料成為一份 XML 文件時，所面臨的資料表示差異性。其中第 I 點到第 IV 點是從任一型態轉到另一型態共同會面臨的問題，接下來則分述當特定一方為來源時所會面臨的問題。

I： XML 文件與資料表格間 1 對多或多對 1 的對應：例如以圖三的 XML 文件為例，其 L1 至 L6 行、L9 至 L31 行對應到 Order 表格，而 L32 至 L46 行則對應到 DetailOrder 表格。

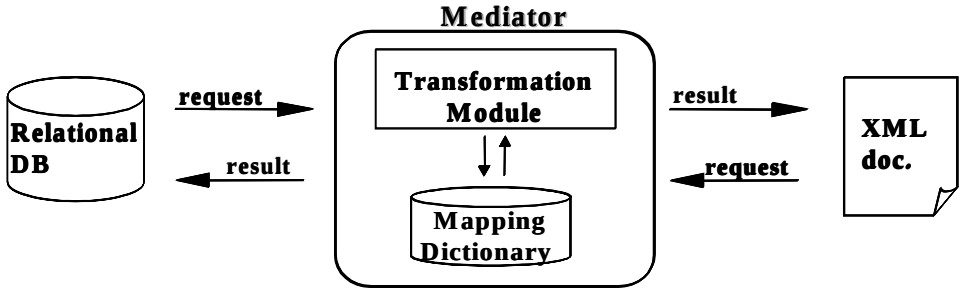
II： 對完整路徑的需求：因為 XML 相同的元素(element)可以出現在不同的路徑(path)下，其所對應的關聯式表格及表格下的欄位可能不同。例如圖二的<FreeFormText>元素分別在 L12、L16、L21、L25 行重覆使用，但其所對應的關聯式表格欄位皆不相同。

III： 意義 (semantics)相同，但所對應的元素名稱或表格欄位名稱不一定相同：例如圖二之 XML 文件以 L11 行至 L14 行的元素 cityName 表示送貨的城市，但是相同的資料在對應的 Order 表格裡所用的欄位名稱為 Order_City。

IV：整合性限制的不同：如在關聯式資料庫裡，每個表格有主鍵(primary key)，或外來鍵(foreign key)等限制，而 XML 文件亦可以對屬性型態定義 ID 或是 IDREF，關聯式資料表格也許和對應的 XML 文件沒有相同的整合性限制。

以下針對 XML 匯入關聯式資料庫所面臨的問題加以討論，其產生原因之一在於關聯式資料具有固定的結構，同一表格的資料列都具有相同的欄位，而 XML 資料是以文件方式呈現，為半結構化資料，資料的表示法有較多的彈性。

V： 資料的重複性：XML 准予定義元素出現的次數，所以必須處理相同元素型態重複出現的問題。譬如，RosettaNet3A4 在其 DTD 的定義中，指定<ProductLineItem>元素 (圖二 L32)至少要出現一次，所以有可能出現多次，例如一筆訂單訂了 5 個不同的商品，則<ProductLineItem>便會出現 5 次。



圖四：系統架構圖

Ⅵ：資料的順序性：在 XML 文件裡元素出現的先後順序有時是有意義的，譬如在多個 <ProductLineItem>元素中，如果對應到 A 商品的<ProductLineItem>元素出現在對應到 B 商品的<ProductLineItem>元素前，則表示在買 B 商品前，須先訂購 A 商品。但在關聯式資料庫裡，資料列先後順序並無意義，所以將 XML 資料匯入關聯式資料庫，資料的順序性必須保持。

Ⅶ： 意義相同但型態不同：由於檢驗 XML 文件格式的 DTD 只支援 PCDATA 資料型態，相對於關聯式資料庫的字串、數字等類別，可能會有元素資料型態與欄位資料型態不同的問題。另一方面，XML Schema 則提供相當複雜的定義功能，也可能會造成與關聯式資料型態的差異性。

Ⅷ： 資料選擇性：XML 文件內有些元素或屬性是用來提供處理內容資料的方式，所以不一定 XML 文件中所有的資料都必須存入資料庫中。如圖二 L12 行的<FreeFormText>，其屬性 lang 是用來表示內容格式的語系，所以其值 “EN”並不需要存到對應的資料表格裡。

相對於上述最後一點，從關聯式資料庫匯出 XML 資料時，可能會面臨下列問題：

Ⅸ：由於關聯式資料庫表格中可能沒有包含 XML 文件所需要的所有資料，導致無法產生一份完整 XML 文件的問題。

三． 仲介者

本篇論文提出一個仲介者的架構 (mediator architecture)，以達到關聯式資料庫與 XML 文件之間的資料轉換與整合。仲介者架構如圖四所示，其中包含了對應字典(mapping dictionary)及轉換模組(transformation module)。對應字典裡記錄了關聯式資料與 XML 資料間彼此的對應關係，而轉換模組裡的轉換規則(transformation rules)則可以利用對應字典裡的資訊做適當的資料轉換。舉例來說，假設企業內部的關聯式資料庫要將透過 web 傳過來的 RosettaNet XML 訂單文件匯進系統內處理，則必須向仲介者下達 request 來觸發 transformation module，transformation module 將會根據事先儲存在 mapping dictionary 的對應資料來執行轉換機制，將 XML 文件資料轉成符合關聯式資料庫中的一筆筆資料紀錄，以便適當的儲存在該系統中。

接下來我們針對對應字典的格式詳加說明。如同 SQL Server 裡 XDR schema 的做法，所有的對應資料是以 XML 的格式表示之。由於 XML 資料與關聯式資料庫表示能力並不相同 (equivalent)，所以我們訂定了兩套語彙 (vocabulary)，分別支援兩種對應方向。

在圖五的範例中，表示了將關聯式資料庫資料對應到一份 XML 文件的對應字典格式，換句話說，它提供了建立圖二的 XML 文件時，如何利用圖一的表格來產生適當的資料。在此，我們只列舉部分對應以作說明。

在圖五中，每一個 element 元素表示一個欲輸出至 3A4 文件裡的元素，每一個 element 元素可以有數個屬性，其中屬性 name 表示 3A4 裡元素的名稱，屬性 FromTable 表示提供對應資料的表格，而屬性 FromField 表示所對應的欄位。譬如 L3 行至 L5 行的 element 元素，是為了建立 3A4 裡的 GlobalLocationIdentifier 元素，而其資料來源是來自於 Order 表格裡的 Order_ShipFrom 欄位。注意的是，在對應字典裡，此 element 是表示在對應到 ProductLineItem 和 shipFrom 元素的 element 裡層，也就是，在對應字典裡仍保持 3A4 文件裡元素的階層關係 (nesting relationship)。

另外我們可以觀察到，在 ProductLineItem 的數個子元素中，L2 行至 L6 行的資料來源是 Order 表格，而 L7 行至 L18 行的資料來源則是 DetailOrder 表格，若是我們只有個自指定來源表格，而沒有做適當的連結，則其產生的資料會如同 cartesian product 的結果，所以在 L19 至 L22 行中，我們再利用 TableRelationship 元素，來表示表格間的關聯，同時也表示了來源表格的資料限制。在此範例中，此兩表格是以訂單編號作為連結，由於它對應到 Order 表格的 Order_Id 欄位，且該欄位為該

```
L1：  <element name=" ProductLineItem">
L2：     <element name="shipFrom">
L3：        <element name="GlobalLocationIdentifier"
L4：          FromTable="Order"
L5：          FromField="Order_ShipFrom"/>
L6：     </element >
L7：     <element name="ProductQuantity"
L8：       FromTable="DetailOrder"
L9：       FromField="DO_ProdQuantity"/>
L10：    <element name="productUnit">
L11：       <element name="ProductPackageDescription">
L12：          <element name="ProductIdentification">
L13：             <element name="GlobalProductIdentifier"
L14：               FromTable="DetailOrder"
L15：               FromField="DO_ProdId"/>
L16：          </element >
L17：       </element >
L18：    </element >
L19：    <TableRelationship primary-Table="Order"
L20：       primary-key="Order_Id"
L21：       foreign-Table="DetailOrder"
L22：       foreign-key="DO_OrderId"/>
L23：  </element>
```

圖五：對應字典格式範例—關聯式資料至 Xml

```
L1：  <Table Name="Order">
L2：     <field name="Order_Id" type="integer"
L3：       FromDoc ="D01" FromPath =
L4：       "//DeliveryHeader/messageTrackingID/
L5：        InstanceIdentifier" occurrence ="required"/>
L6：     <field name="Order_City" type="char"
L7：       FromDoc ="D01" FromPath =
L8：       "//Pip3A4PurchaseOrderRequest/
L9：        PurchaseOrder/deliverTo/PhysicalAddress/
L10：       cityName/FreeFormText"
L11：       occurrence = "implied"/>
L12：  </Table>
```

圖六：對應字典格式範例—Xml 至關聯式資料

表格的主鍵，所以用 TableRelationship 元素的屬性 primary-Tableh 和 primary-key 來表示。另一方面，訂單編號在 DetailOrder 表格可以重複出現，所以用 TableRelationship 元素的另兩個屬性 foreign-Table 和 foreign-key 來表示。其意味著在取出資料庫資料時須符合其關聯性。

圖六的對應字典範例，則表示了將 XML 文件轉換為資料庫資料的對應資料，也就是說，將圖二的 3A4 XML 文件的資料，轉入圖一的關聯式表格。在此，每一個 Table 元素對應到資料庫裡的一個表格，以其屬性 Name 來表示所對應的表格名稱。一個 Table 元素內含數個 field 子元素，用以表示該表格的每一個欄位。在此我們只舉 Order 表格的兩個欄位為例。

每一個 field 子元素定義了數個屬性，其中 name 屬性是表格裡的欄位名稱，type 描述該欄位的型態，FromDoc 表示提供資料來源的 XML 文件編號，FromPath 則表示在該文件的完整路徑。最後利用 occurrence 來表示是否會有多筆資料符合該路徑。若其值為 required，則表示該元素在 XML 文件中為必要元素，至少會有一筆資料產生，若為 0 次或 n 次，則以 implied 表示。

以下我們針對第二節所提出的問題，說明圖五及圖六的對應字典格式，可以提供何種解決途徑：

問題Ⅰ與問題Ⅲ：為了解決多個表格與 XML 文件之間資料交錯對應，且名稱不一致的問題，在圖五中我們利用 FromTable 和 FromField 明確地指出，從何種表格的特定欄位可以得到該 XML 元素的資

料。相對地，在圖六中則使用 FromDoc 和 FromPath 來描述特定 XML 文件的指定路徑。

問題Ⅱ：為了解決相同元素名稱會在一分 XML 文件出現在不同路徑，在圖五中 element 呈現的方式是以輸出 XML 文件的巢狀結構排列，而圖六中的 FromPath 屬性值則是該元素的絕對路徑。

問題Ⅳ：表格間整合性的限制，是利用資料字典中的 TableRelationship 元素，以及其屬性 primary-Table 、 primary-key 、 foreign-Table 、 foreign-key 來描述，如圖五所示。

問題Ⅴ：當我們在圖六中描述 XML 資料來源時，我們會進一步利用 occurrence 屬性，來說明該資料會以何種方式重複，以便 transformation rule 做適當地轉換。

問題Ⅶ：為了避免對應資料格式不同，我們利用 type 屬性來描述對應欄位之屬性型態，以便在轉換時有所根據，如圖六所示。

問題Ⅷ：由於我們是針對所需要的輸出去描述資料來源，所以 XML 文件裡屬於資料處理指令的部分，不會表示在資料字典中。

所以，當我們欲達成企業內關聯式資料庫與供應鍊所採用之 RosettaNet XML 文件的資料互通與共享時，系統管理者 (DBA) 必須先分析兩邊綱要(schema)的對應關係，然後利用我們所提出的對應字典的格式，建立適當的對應關係，之後每當有轉換的需求時，轉換法則 (transformation rules)就會執行適當的步驟產生適當的資料。

四． 相關研究

目前市面上重要的資料庫產品都紛紛提供支援 XML 文件的功能。IBM 的 DB2 有兩種方式可以將 XML 檔案轉換為資料庫中的資料[12]。第一個方式，是將 XML 檔案原封不動地直接存入一 BLOB 格式的文字欄位。雖然因為不必事先知道 XML 檔案的結構，所以處理速度較快，但是受限於文字欄位，故只能執行文字搜尋 (text search)，不能執行結構型及數值上的搜尋 (如訂購 a 產品大於五個的購買者)。第二個方式，是利用對應的方式，將 XML 檔案中的元素及屬性，對應到資料表 (table) 及資料行 (column) 中。如此 XML 檔案的內容可以用 SQL 來處理，因此可運用於 SQL-based 的應用程式。但是此種對應雖然保留了部分結構，但是有些仍會遺失，如原本在 XML 檔案時，資料的順序性。

Microsoft 的 SQL Server 在支援 XML 的功能中則有一類似檢視 (view) 的功能，名稱為 XDR (XML-Data Reduced) 的結構描述[17]。其方法是將資料庫裡的表格以 XDR schema 來建立 XML 資料檢視 (data view)，之後就可以對其來執行一般 XML 文件的處理，如執行 Xpath 查詢。一個標準的 XDR schema 檔案是以<Schema>標籤開始，以</Schema>標籤作為結尾，在 XDR 文件內可定義許多元素 (ElementType) 及其屬性 (AttributeType)。如果是使用預設的對應 (Default Mapping)，SQL Server 會假設每個元素皆對應至相同名稱的資料表，而每個屬性或子元素則對應到資料表中相同名稱的資料行。第二個方式是使用註解 (Annotation) 做資料的對應，也就是以自訂的對應方法，在結構描述中加入註解，以指定元素或屬性是對應到資料庫中的某個表格或屬性。

另外許多研究者也針對 XML 與關聯式資料之間的對應技術與轉換機制，相繼提出不同的做法。例如有的方法使用了兩個表格來儲存由 XML 文件的內容，其一專司儲存元素路徑、元素值、父元素路徑等相關資訊，另一專司儲存屬性所屬的元素路徑、屬性名稱、屬性值等相關資訊，此對應方法相當直覺，但對於表示表格之間複雜的關聯性；便相對地有限[6]。

而有的研究者則不直接將 XML 資料轉成表格，而是將 DTD 轉換為 entity-relation model，利用分析 DTD 的特性，歸納演變出其演算法，透過演算法的規則便可一步步地將 DTD 轉換為

entity-relation model，但根據其轉換規則，轉換出的表格關係可能會過於複雜[9]。

五． 結論與未來展望

利用 XML 格式來進行供應鍊管理已經是時勢所趨，特別是在科技掛帥的台灣，眾多關於 RosettaNet 的計畫正不斷地在執行中或被提出，例如康柏電腦公司的「康柏 e 商網台威計畫」，又如台積電、聯電等公司共同提出的「晶采計畫」(半導體工單管理作業標準)，所以 RosettaNet 已逐漸成為全球 SCM 標準，往後不再是“No EDI, No Order”，而是“No RosettaNet, No Order”。

有鑒於 XML 型態的資料與企業內部關聯式資料彼此互通的殷切需求，本篇論文提出了一個仲介者的架構。在此架構中的一個重要部分是提供資料轉換根據的對應字典，我們首先討論關聯式資料與一般 XML 資料可能出現的不同表示法，然後做適當的設計。對應字典是以 XML 格式來呈現，我們分別提出兩組字彙，包含元素與屬性的定義，來提供將 XML 文件中的資料轉換為資料庫中的資料，或將資料庫中的資料轉換為 XML 文件中，所需的對應資料。由於我們提出的方法是針對關聯式資料庫與 XML 資料本身的限制，所以可適用於各個廠牌的關聯式資料庫，具有更大的彈性。

未來的研究方向如下所述;

1. 我們將比較文獻上所提出的方法，進一步改善對應字典的定義，使其提供更完善且明確地對應資料。
2. 我們將提出一系列的轉換法則，與對應字典互相配合，實作出整個仲介者的系統功能。
3. 我們希望能以實際的 RosettaNet 文件和關聯式資料，來驗證我們仲介者系統的可行性，同時也希望能與其他研究者所建立的系統，在功能上與效能上互相比較。

■ 誌謝

此計畫由國科會贊助，編號為：NSC 90-2213-E-019-011

六、參考文獻

[1] N. Alon, T. Milo, F. Neven, D. Suciu, V. Vianu, *“Typechecking XML views of relational databases”*, Proceedings of 16th Annual IEEE Symposium on Logic in Computer Science, page(s): 421 – 430, June 2001.

[2] Alin Deutsch, Mary Fernandez, Dan Suciu, *“Storing semistructured data with STORED”*, Proceedings of the 1999 international conference on Management of data, June 1999.

[3] D. Florescu, D. Kossmann, *“A Performance Evaluation of Alternative Mapping Schemes for Storing XML Data in a Relational Database”*, Rapport de Recherche No. 3680, INRIA, Rocquencourt, France, May 1999.

[4] C. Kanne, G. Moerkotte, *“Efficient Storage of XML Data”*, Proceedings of 16th International Conference on Data Engineering, page(s): 198 – 198, 29 Feb.-3 March 2000.

[5] Latifur Khan, Yan Rao, *“A performance evaluation of storing XML data in relational database management systems”*, Proceeding of the Third International Workshop on Web Information and Data Management, November 2001.

[6] Joo Kyung-Soo, *“A design of middleware components for the connection between XML and RDB”*, Proceedings of IEEE International Symposium on Industrial Electronics, vol.3 no.12-16, page(s): 1753 – 1756, June 2001.

[7] Wang-Chien Lee, Gail Mitchell, Xin Zhang, *“Integrating XML Data with Relational Databases”*, Proceedings of 2000 ICDCS WORKSHOP, page(s): 47-F53, April 2001.

[8] H. Schoning, *”Tamino - a DBMS designed for XML”*, Proceedings of 17th International Conference on Data Engineering, page(s): 149 – 154, April 2001.

[9] Iraklis Varlamis, Michalis Vazirgiannis, *“Bridging XML-schema and relational databases: a system for generating and manipulating relational databases using valid XML documents”*, Proceeding of the ACM Symposium on Document Engineering, November 2001.

[10] Masatoshi Yoshikawa, Toshiyuki Amagasa, Takeyuki Shimura, Shunsuke Uemura , *”XRel: a path-based approach to storage and retrieval of XML documents using relational databases”* ACM Transactions on Internet Technology (TOIT), Volume 1, Issue 1, August 2001.

[11] 梁秀麗 編譯, ”*Microsoft SQL Server 2000 程式開發-使用 XML*”, 華彩軟體股份有限公司.