第4章 系統建置

本研究參考銀行公會的交易明細查詢回應訊息、外幣進帳通知訊息及相關網路銀行網站，建置銀行端的帳戶欄位 XML 文件。透過本章雛形系統將不同欄位標籤合併成單一標準標籤。下述分成四個小節將說明雛形系統的建置方式：UDDI API 介面、銀行端網路服務、共通平台整合服務以及帳戶代理人。

第1節 UDDI API 介面說明

UDDI 提供一個簡易的 Request/Response 機制，可供尋找商業實體(business Entity)、商業服務(service)，與技術服務連結(tModel)的相關資訊。UDDI API 介面主要是讓程式人員更方便簡易的方式使用 UDDI 服務，故使用簡易性是 UDDI API 的一大特色，UDDI API 主要可分爲 Publishers API 與 Inquiry API 兩大介面：

1. Publishers API 介面：提供使用者與 UDDI Registry 的互動，可以儲存或變更 UDDI Registry 中的資料。本研究中的 AUDDI 註冊中心，可透過 Internet Explorer 瀏覽器線上註冊網路服務，其使用步驟將第五章說明之。
2. Inquiry API 介面：提供程式設計師取得 UDDI Registry 中的資訊。Publisher API 有使用權限上的限制，每一個提供 UDDI 存取服務站可以自訂與使用者互動的協定，Inquiry API 是透過 HTTP 協定來傳輸 SOAP 訊息，使用上沒有權限的限制。

目前在 UDDI 的最新版本爲 UDDI 第三版，在第三版規範中提供一組 Inquiry API 介面，讓使用者透過 Inquiry API 存取 UDDI 註冊中心的資料，其中包含 find_binding、find_business、find_service、find_tModel 等搜尋操作，以下將詳細說明之。

(一) find_binding

屬性名稱	必要性	說明
authInfo	非必要	授權資訊。商業實體在 UDDI 操作入口中的名字
findQualifers	非必要	查詢條件
tModel	必要	指定 tModel Key，找出 binding Template 元素

(二) find_business

屬性名稱	必要性	說明
authInfo	非必要	授權資訊
businessKey	必要	商業實體編號之系統編碼
name	必要	商業實體的名稱
categoryBag	非必要	商業實體的分類法
discoveryURLs	必要	預設商業實體儲存位址

(三) find_service

屬性名稱	必要性	說明
serviceKey	必要	商業服務編號之系統編碼
name	必要	商業服務的名稱
businessKey	必要	商業實體編號
businessTemplate	必要	繫結元素，如：accessPoint 元素(說明如下)
categoryBag	非必要	商業服務的分類法

accessPoint 元素是一個由元素屬性指定網路服務的入口點，而連結入口點的方式可爲 mailto、http、https、ftp、fax、phone 等連結方式，其中以 http、https 連結網路服務居多。

(四) find_tModel

屬性名稱	必要性	說明
tModelKey	必要	tModel 編號之系統編碼，如：uuid:xxxx
name	非必要	tModel 的名稱
instanceDetails	非必要	設定特定 tModel 的描述資訊

(五) UDDI XML 文件範例

圖 4-1 爲在 UDDI 註冊中心搜尋 A 銀行的查詢結果，UDDI 註冊中心結果包裝成 XML 文件，經過剖析後，呈現在瀏覽器畫面上，其文件主要分成三個部分：

1. 商業實體(Business Entity)

第 1 行到第 14 行指的是商業實體的資訊，所謂的商業實體是代表一個提供服務的公司的相關資訊，包含公司代碼<businessKey>、公司名稱<name>、公司聯絡方式<contact>等標籤。<businessKey>是 UDDI 給定的系統唯一值，以 8-4-4-4-12 的 16 進位的格式編碼排列。<contact>是公司聯絡方式，可記載公司擁有人<personName>、電話<phone>、電子郵件<email>及位址<address>等資訊，所以說，註冊商業實體如同註冊一家實體公司。

2. 商業服務(Business Service)

第 15 行到第 20 行指的是商業服務的描述性資訊，包含服務代碼<servicekey>、服務名稱<name>、繫結樣版<bindingTemplates>等標籤。<servicekey>是商業服務唯一代碼，同樣採用<businessKey>的編碼方式編碼。<bindingTemplates>指的是服務繫結資訊，其中服務實際入口點<accesePoint>是服務實際的位置，如：http://localhost/abank/Service1.asmx/CreateXmlData。屬性<URLType="http">指的是採用 http 網路協定繫結服務。

3. 技術模型(tModel)

第 21 行到第 27 行指的是服務的技術模型，由於 UDDI 沒有定義一個標準方法來描述商業實體和商業服務的詳細資訊，而這些詳細資訊並非 UDDI 所規範，大多由標準化團體、工業組織、軟體商所定義，故 UDDI 提供另一機制來指到這些非 UDDI 的詳細資訊，而 tModel 就是解決非 UDDI 的解釋資料。

finding_tModel 便是查詢商業服務的技術模型。詳細資訊包含技術模型代碼<tModelKey>、技術模型說明檔<overviewURL>等標籤。<tModelKey>同樣是採用通用獨一識別碼(UUID[7])的方式編碼，確保代碼是獨一無二。<overviewURL>是 tModel 說明附件網址，用以詳述 tModel 內容。

```
1  <businessDetail generic="2.0" operator="http://localhost:7081" xmlns="urn:uddi-org:ap
2    <businessEntity businessKey="4701f86e-73ef-47ff-8f98-2c020426515f" authorizedName="
     operator="http://localhost:7081">
3      <discoveryURLs>
4        <discoveryURL
         useType="businessEntity">http://localhost:7081/uddi/uddilistener?businessKey=47
         -47ff-8f98-2c020426515f</discoveryURL>
5      </discoveryURLs>
6      <name xml:lang="en">Bank_ABank</name>
7      <contacts><contact useType="CEO">
8          <personName>Hung-Pin Chang</personName>
9          <phone useType="Taiwan">886-2-29393095</phone>
10         <email useType="nccu.edu.tw">redice@mis.nccu.edu.tw</email>
11         <address>
12           <addressLine keyName="school" keyValue="address">64, Sec. 2,Zhi-nan Rd., We
             Taipei 116, Taiwan, Republic of China</addressLine>
13         </address></contact>
14     </contacts>
15     <businessServices>
16       <businessService serviceKey="84622f0a-ffaa-4743-80e4-6c65d3be2603"
         businessKey="4701f86e-73ef-47ff-8f98-2c020426515f">
17         <name xml:lang="en">General Account</name>
18         <bindingTemplates>
19           <bindingTemplate bindingKey="6aeb1387-7b16-4620-a5a2-7f5543e7dae7"
             serviceKey="84622f0a-ffaa-4743-80e4-6c65d3be2603">
20             <accessPoint
               URLType="http">http://localhost/abank/Service1.asmx/CreateXmlData</access
21             <tModelInstanceDetails>
22               <tModelInstanceInfo tModelKey="uuid:00bdf895-7c04-4c51-b479-42fa2e4606a
23                 <instanceDetails>
24                   <overviewDoc>
25                     <description xml:lang="en">doc desc</description>
26                     <overviewURL>http://localhost/abank/banking_tModel_readme.txt</ov
27                   </overviewDoc>
```

圖 4- 1 UDDI XML 文件範例

[7] UUID(Universally Unique Identifier) 是以目前的日期與時間的唯一性，並加上網路卡 (NIC) 48 位元 IEEE 802 的唯一位址而產生其獨特性質。

第2節　銀行網路服務的建立

各銀行依照銀行公會訊息標準訂定帳戶欄位名稱 XML 檔，但是，仍有部分銀行之即有系統無法產生符合銀行公會所訂定標籤，或是銀行特有標籤未定義在銀行公會規範中，而這些標籤正是某些銀行特有帳戶欄位，本研究爲了讓帳戶整合更加完整，更符合實際應用，採用銀行端網路服務和共通平台整合服務技術，解決 XML 標準不一致的問題，若能透過合併機制，將彼此帳戶欄位名稱彙整，便能互通銀行間的資訊，達到資訊交換的整合功效。

銀行端建立銀行帳戶網路服務，以提供共通平台之帳戶整合。本研究採用.NET 技術模擬銀行端網路服務，將銀行資料庫的帳戶資料藉由兩個網路服務傳遞出去，一爲帳戶欄位名稱網路服務，將帳戶欄位名稱 XML 傳遞給共通平台，另一爲帳戶欄位實際值網路服務，將帳戶欄位實際值 XML 傳遞給客戶端帳戶代理人。

圖 4-2 所示爲帳戶欄位名稱網路服務結果，圖 4-3 所示爲帳戶欄位名稱實際值網路服務結果，每一個欄位名稱正對應一個欄位名稱實際值，不過，欄位名稱實際值只有客戶本人才可以取得。

```
- <DepActTrnInqRs flag="金融帳戶明細">
    <!-- A銀行的欄位資料-->
  - <DepAcctTrnRec flag="資產帳戶明細">
    - <BankAcctTrnRec flag="帳戶交易明細">
        <TrnType flag="借貸別">string</TrnType>
        <PostedDt flag="交易日期">int</PostedDt>
        <PostedTrn flag="交易時間">int</PostedTrn>
        <TrnDesc flag="交易摘要">string</TrnDesc>
      - <CurAmt flag="交易金額">
          <Amt flag="金額">double</Amt>
        </CurAmt>
        <Memo flag="備註">string</Memo>
        <SPRefId flag="本次交易序號">string</SPRefId>
      - <SPRefIdCorrect flag="交易參考">
          <SPRefId flag="原交易序號">string</SPRefId>
          <CorrectAction flag="更正記號">string</CorrectAction>
        </SPRefIdCorrect>
      - <AcctBal flag="帳戶餘額">
          <BalType flag="餘額類別">string</BalType>
        - <CurAmt flag="可用餘額">
            <Amt flag="金額">double</Amt>
          </CurAmt>
        </AcctBal>
      </BankAcctTrnRec>
      <ChkNum flag="支票號碼">string</ChkNum>
    </DepAcctTrnRec>
  </DepActTrnInqRs>
```

圖 4- 2 銀行帳戶欄位名稱 XML 訊息

	BankAcctTrnRec				ChkNum
1	BankAcctTrnRec				9866
		TrnType	Debit		
		PostedDt	20040422		
		PostedTrn	112759		
		TrnDesc	提款		
		CurAmt			
			Amt	2000.00	
		Memo	ATM提款		
		SPRefId	1676220101		
		SPRefIdCorrect			
			SPRefId	73028288	
			CorrectAction	7302201	
		AcctBal			
			BalType	Current	
			CurAmt		
				Amt	13000.00
2	BankAcctTrnRec				9876
		TrnType	Debit		
		PostedDt	20040423		
		PostedTrn	142759		
		TrnDesc	轉帳		
		CurAmt			
			Amt	2500.00	
		Memo	劃撥轉帳		
		SPRefId	1676220111		
		SPRefIdCorrect			
			SPRefId	73028298	
			CorrectAction	7302301	
		AcctBal			
			BalType	Avail	

圖 4- 3 銀行帳戶欄位實際值 XML 訊息

上圖 4-3 的兩筆銀行帳戶明細資料，則來自於銀行端的資料庫中，其代表兩筆帳戶明細資料：第一筆資料爲 2004/04/22 使用 ATM 提款一筆 2,000 元現金，

帳戶餘額爲 13,000 元，第二筆資料爲 2004/04/23 劃撥轉出帳款 2500 元，帳戶餘額爲 10,500 元。

TrnId	memberId	TrnType	PostedDt	PostedTrn	TrnDesc	CurAmt	Memo	SPRefId	SCSPRefId	SCCorrectAction	ABBalType	ABCurAmt	ChkNum
2	Fly	Debit	20040422	112759	提款	2000	ATM 提款	1676220101	73028288	7302201	Current	13000	9866
3	Fly	Debit	20040423	142759	轉帳	2500	劃撥轉帳	1676220111	73028298	7302301	Avail	10500	9876

圖 4- 4 銀行端帳戶明細資料

下述文件說明爲.NET 程式語言開發網路服務，此片斷程式碼將圖 4-4 帳戶明細資料轉換成圖 4-3 欄位名稱實際值 XML，說明如圖 4-5 所示：

```
[WebMethod(Description="銀行存簿補摺(若帳號、密碼正確，才會回傳起迄時間之內的銀行XML資料，新版)")]
    public XmlDocument CreateXmlData(string id, string pwd,int dateStart,int dateEnd){
        //透過資料庫取得起始日期和最終日期之間的帳戶明細資料
        xmlBankTran = new XmlBankTran(conn,id,dateStart,dateEnd);
        xmlBankTran.createXML(); //抓取資料庫資料產生出XML訊息
        doc = xmlBankTran.getXmlDocument();
        conn.Close();
      }
      return doc;
    }
```

```
1  public void createXML(){
2    String sSQL = "SELECT
     TrnType,PostedDt,PostedTrn,TrnDesc,CurAmt,Memo,SPRefId,SCSPRefId,SCC
     urAmt,ChkNum FROM BankAcctTrnRec WHERE PostedDt BETWEEN "+ dateStart
     AND memberId ='"+ memberId +"'";
3    //DataAdapter設定
4    SqlDataAdapter objAdapter = new SqlDataAdapter(sSQL,conn);
5    //資料集&配接器&資料表
6    DataSet objDataSet = new DataSet("CommentsPage");
7    //將Adapter資料全部倒入DataSet中
8    objAdapter.Fill(objDataSet,"dtBankAcctTrnRec");
9    //將資料庫查出來的資料一一做成xml標籤，產生XML文件
10   XmlTextWriter wtr = new XmlTextWriter(sFile, Encoding.UTF8);
11   //寫入<DepActTrnInqRs>開始標籤
12   wtr.WriteStartElement("DepActTrnInqRs");
13   //寫入<DepActTrnInqRs>的屬性 message = "OK" 資料
14   wtr.WriteAttributeString("message","OK");
15   foreach(DataRow row in objDataSet.Tables["dtBankAcctTrnRec"].Rows){
16     //寫入<DepAcctTrnRec>開始標籤
17     wtr.WriteStartElement("DepAcctTrnRec");
18     //寫入<BankAcctTrnRec>開始標籤
19     wtr.WriteStartElement("BankAcctTrnRec");
20     wtr.WriteElementString("TrnType",row[0].ToString());
21     wtr.WriteElementString("PostedDt",row[1].ToString());
22     wtr.WriteElementString("PostedTrn",row[2].ToString());
23     wtr.WriteElementString("TrnDesc",row[3].ToString());
24     //寫入<CurAmt>開始標籤
25     wtr.WriteStartElement("CurAmt");
26     ............
27     ............
```

圖 4- 5 帳戶實際值網路服務之程式碼

第3節　共通平台帳戶欄位名稱整合服務的建置

共通平台目的在於將各銀行帳戶欄位名稱 XML 訊息整合一份完整的 XML 檔。顧客根據自身帳戶需求，透過共通平台動態整合不同帳戶，提供完整性的欄位名稱檔，再執行下一節提及的帳戶代理人，便能依照完整欄位名稱檔，將各銀行的帳戶明細資料一一對應，呈現在客戶端畫面上。

共通平台的角色被定位爲服務仲介者，功能不同於 UDDI。UDDI 主要掌管服務註冊、搜尋及目錄管理的工作，共通平台主要是整合網路服務的應用服務(Application Services)，負責服務之間的溝通及整合相關工作。

本研究之共通平台採用企業前三大的應用伺服器[8]的 BEA WebLogic Application Server，做爲平台的技術底層架構，由於 WebLogic 採用 J2EE1.3 技術可進行 UDDI 搜尋呼叫及網路服務整合和實作。

共通平台整合服務主要分成三大模組，一爲擷取服務(Retrieval Service)，目的在呼叫各服務端的網路服務，取得銀行帳戶欄位名稱訊息，透過第二模組的訊息合併規則的對應服務(Mapping Service)，產生出對應欄位元名稱表格，最後顧客端呼叫第三模組的表格取回服務(Acquire Service)，目的回傳客戶端要求的對應表格。其整合服務運作流程如圖 4-6 所示。

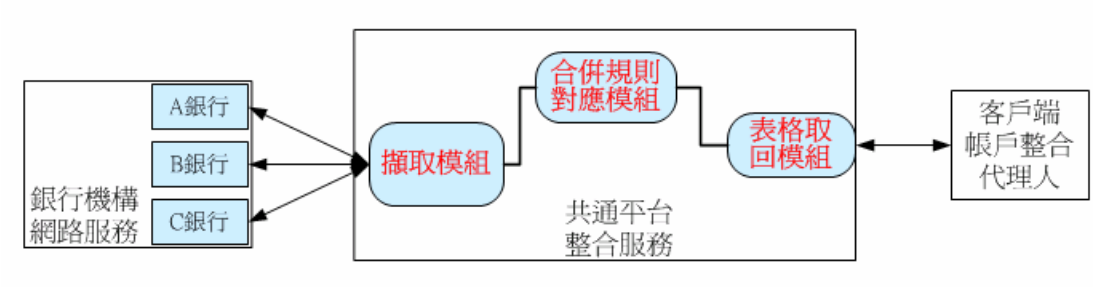

圖 4- 6 共通平台運作流程

下述將成三個部分詳細說明擷取模組、合併規則的對應模組及表格取回模組的建置步驟。

[8] 根據 Gartner Dataquest 報告指出，企業前三大應用伺服器爲 IBM WebSphare(37%)、BEA WebLogic(29%)、Sun Java System(4%) Application Server。

4.3.1 擷取模組

根據前一節銀行端的網路服務，提供不同方式呼叫網路服務，以因應不同的情況要求，主要分別爲 SOAP Message、HTTP GET、HTTP POST 等三種方式。本研究仍採用 HTTP GET 方式呼叫銀行網路服務，以取得帳戶欄位名稱 XML 檔。下述片斷文件爲共通平台呼叫 A 銀行的帳戶欄位名稱的程式碼說明，如圖 4-7 所示：

```
1  /*
2   * urlURL指的是A銀行網路服務的位置、urlURL2指的是共通平台目前擁有的標準整合欄位的檔案位置
3   * table是urlURL的資料庫表格，"standard"是urlURL2的資料庫表格
4   */
5  private void doAllThing(String table,URL urlURL,String url2) {
6    List parsingResult = null;
7    try {
8      parsing = new RecurionTagParsing(urlURL);
9      parsingResult = (List)parsing.doParsing(); //將A銀行的xml轉換到資料庫的Bank_A資料表中
10     creatorDao = new CreatorDao(table);
11     creatorDao.insert(parsingResult); //A銀行的xml剖析後的結果存入Bank_A資料表格中
12     ..........
32     //執行mapping模組將A銀行與標準欄位進行合併規則對應
33     mapping  new RecurionMapping(urlURL,urlURL2);
34     mapping.doMapping(); //將A銀行的xml與標準欄位的xml合併成一個完整的xml
35     ......
73     parsing = new RecurionTagParsing(new FileInputStream(url2));
74     parsingResult = (List)parsing.doParsing(); //將A銀行與標準欄位合併後的xml轉換成資料庫格式
75     standardDao = new StandardDao(table);
76     standardDao.insert(parsingResult); //儲存到Standard資料庫表格中
77   }
78   catch (Exception e) {
79     e.printStackTrace();
80   }}
```

圖 4- 7 呼叫銀行端網路服務的程式文件

第 7 行到第 12 行程式將 A 銀行網址透過網路服務呼叫後，剖析 XML 檔，並儲存於 Bank_A 資料表格中。

第 32 行到第 35 行程式執行下一小節的合併規則模組，將 A 銀行與標準欄位的 XML 文件進行合併規則之對應。

第 73 行到第 76 行程式剖析合併過後的標準欄位的 XML 檔，儲存到 Standard 資料表格中。

4.3.2 合併規則對應模組

根據 4.3.1 小節擷取服務取得銀行端的欄位訊息，並透過遞迴演算法剖析整份欄位名稱 XML 文件的各個子節點，在本系統中採用 JDOM.ORG 所發展的 DOM XML 技術，透過 JDOM API 介面將 XML 文件剖析成記憶體的 DOM 剖析樹，由應用程式呼叫 JDOM API 控制 XML 處理器產生 XML 檔，從另一方向來說，解析 XML 檔，呼叫 XML 處理器，剖析 XML 文件成 DOM 剖析樹，再給應用程式取用，其剖析流程如圖 4-8 所示。

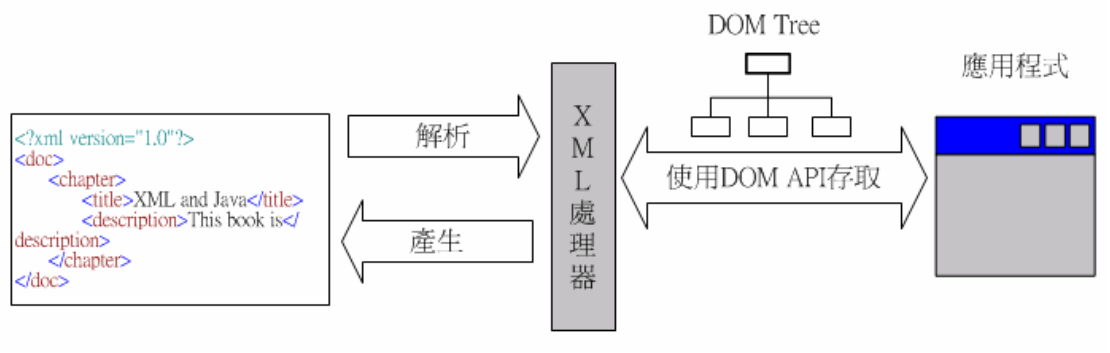

圖 4- 8 XML 剖析流程圖

剖析流程需用迴圈和遞迴演算法，原因在於 XML 檔中每個節點可能有多個兄弟節點(sibling node)，每個節點又有子樹節點(child node)，形成樹狀結構，所以要能夠走訪所有的節點時，必須迴圈走訪兄弟節點，遇到子樹節點則遞迴走訪，如此一來，即可針對每個節點妥善控制及處理。合併規則對應模組仍根據這樣概念，設計流程以處理任何可能發生的情況，其運作流程可分爲以下 10 項步驟，如圖 4-9 所示：

1. 利用 JDOM 的技術剖析銀行欄位訊息，將節點製作成 DOM 樹，一次取得一節點，進行後續判斷程式。
2. 判斷檔節點是否有子節點，若節點本身就是 Leaf 的話，執行步驟(4)，反之，若節點爲 Node(節點有子樹存在)，則執行步驟(3)。
3. 節點爲 Node，則表示子樹有節點存在，故要執行迴圈程式，走訪子樹內所

有兄弟節點。

4. 判斷檔節點名稱是否與標準表格資料名稱是否相同，若彼此名稱相同，則執行步驟(5)，反之，執行步驟(9)。標準欄位元表格所有資料儲存在共通平台的 SQL Server 資料庫，標準欄位元表格保有最豐富且完整的欄位名稱資訊，作爲所有欄位名稱對應的基石。
5. 判斷該節點的子節點是否還有子節點，意思是說，該子節點是否爲 Node 或是 Leaf，若子節點屬於 Node，執行步驟(6)，反之，執行步驟(7)。
6. 若子節點屬於 Node，進行深一層的遞迴剖析，反覆步驟(2)剖析子樹節點。
7. 繼續剖析同一層其他兄弟節點。
8. 判斷兄弟節點是否存在，若兄弟節點存在，執行步驟(4)，將指標指向下個兄弟節點，繼續遞迴程式，直到走訪所有節點爲止。
9. 若檔節點名稱與標準表格記錄的名稱不相同，搜尋辭彙庫是否有相同的辭彙，若辭彙存在，執行步驟(7)，反之，辭彙不存在，執行步驟(10)。
10. 若辭彙不存在，則將該節點新增到標準表格欄位元中。新增標準欄位後，執行步驟(9)，剖析下一兄弟節點。

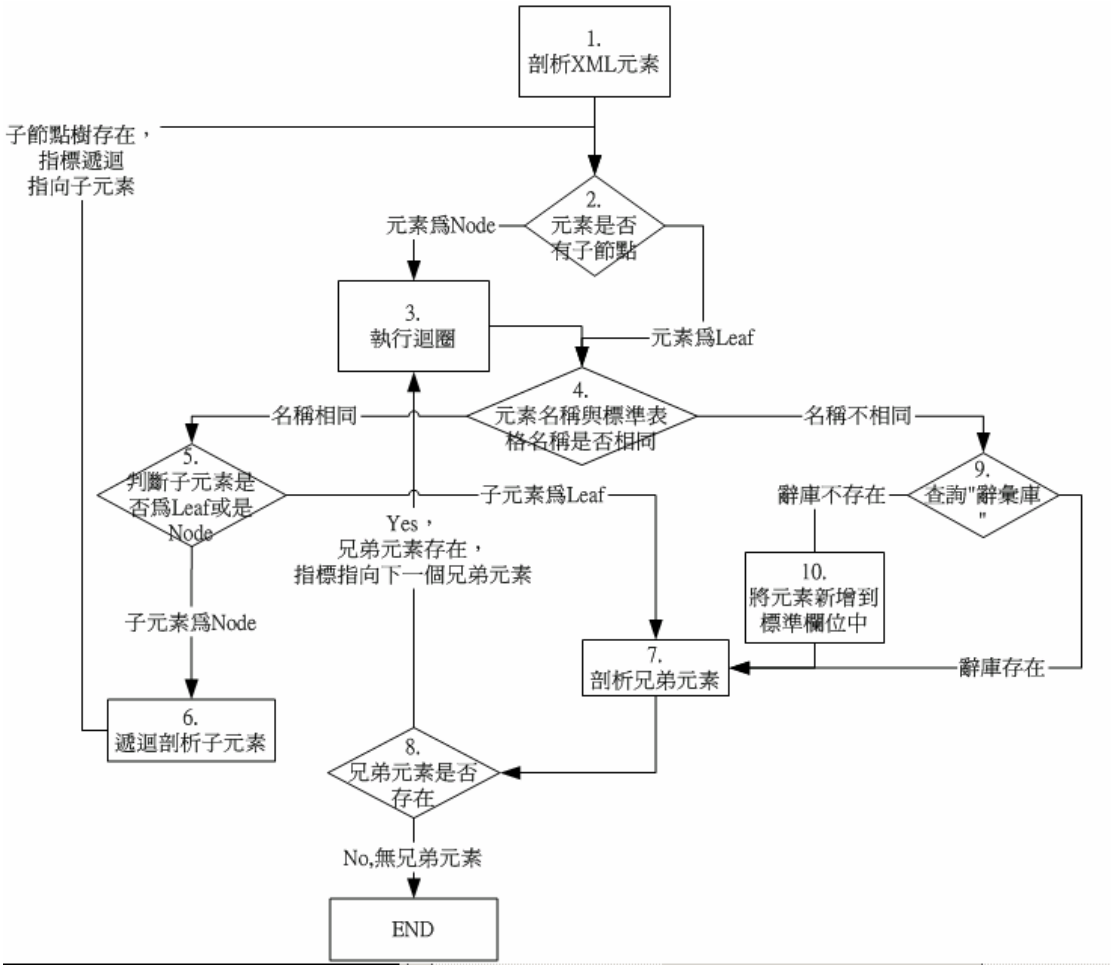

圖 4- 9 合併規則對應運作流程

經過上述 10 項步驟，將一個個節點與共通平台的標準欄位進行合併比對，若節點未出現共通平台標準欄位中，則將節點新增到標準欄位中。透過一連串的合併之後，再將合併訊息以第二章文獻探討的 Edge Approach 資料模式儲存於資料庫中，其結果如圖 4-10 所示。

	source	ordinal	name	flag	target	tokey	path	sourceA	ordinalA	sourceB	ordinalB
1											
2	-1	1	DepActTmInqRs	金融帳戶明細	0	0	/DepActTmInqRs	-1	1	-1	1
3	0	1	DepAcctTmRec	資產帳戶明細	0	1	/DepActTmInqRs/DepAcctTmRec	0	1	0	1
4	1	1	BankAcctTmRec	帳戶交易明細	0	2	/DepActTmInqRs/DepAcctTmRec/BankAcctTmRec	1	1	1	1
5	2	1	TmType	借貸別	string	-3	/DepActTmInqRs/DepAcctTmRec/BankAcctTmRec/TmType	2	1	2	1
6	2	2	PostedDt	交易日期	double	-3	/DepActTmInqRs/DepAcctTmRec/BankAcctTmRec/PostedDt	2	2	2	2
7	2	3	PostedTm	交易時間	double	-3	/DepActTmInqRs/DepAcctTmRec/BankAcctTmRec/PostedTm	2	3	2	3
8	2	5	CurAmt	交易金額	0	3	/DepActTmInqRs/DepAcctTmRec/BankAcctTmRec/CurAmt	2	5	2	6
9	2	7	SPRefId	本次交易序號	string	-3	/DepActTmInqRs/DepAcctTmRec/BankAcctTmRec/SPRefId	2	7	2	7
10	2	9	AcctBal	帳戶餘額	0	5	/DepActTmInqRs/DepAcctTmRec/BankAcctTmRec/AcctBal	2	9	2	5
11	3	1	Amt	金額	double	-3	/DepActTmInqRs/DepAcctTmRec/BankAcctTmRec/CurAmt/Amt	3	1	6	1
12	5	1	BalType	餘額類別	string	-3	/DepActTmInqRs/DepAcctTmRec/BankAcctTmRec/AcctBal/BalType	5	1	4	1
13	5	2	CurAmt	餘額金額	0	6	/DepActTmInqRs/DepAcctTmRec/BankAcctTmRec/AcctBal/CurAmt	5	2	4	2
14	6	1	Amt	金額	double	-3	/DepActTmInqRs/DepAcctTmRec/BankAcctTmRec/AcctBal/CurAmt/Amt	6	1	5	1

圖 4- 10 Edge Approach 資料模式結果

下述將介紹整合服務的合併規則對應的片斷程式，主要分成主程式和遞迴程式，見圖 4-11、圖 4-12 所示。

```
1  //執行mapping模組將A銀行與標準欄位進行合併規則對應
2  public void doMapping() throws JDOMException, Exception {
3    root = document.getRootElement(); // 取得A銀行欄位xml的根
4    st_root = st_document.getRootElement(); //取得標準欄位xml的根
5    if (root != null && st_root != null) {
6      if (root.getName().equals(st_root.getName())) { //比較欄位名稱是否相同
7        recursion(root,st_root); //相同的話，進行子樹節點遞迴判斷
8      }else{
9        throw new Exception("根節點不相同");
10     }}} // end
```

圖 4- 11 合併規　則對應主程式文件

第 3 行和第 4 行程式是分別取得 A 銀行欄位和標準欄位 XML 檔的根節點，取得之後，判斷兩者欄位名稱是否相同，相同的話，進行子樹節點遞迴判斷，詳細程式文件說明如圖 4-12 所示。

```
3  //elm爲銀行欄位的子節點；elm2爲標準欄位的父節點
4  private void recursion(Element elm, Element elm2){
5    List st_childlist = null; //標準欄位的子節點容器
6    Iterator st_child_it = null; //標準欄位的子節點迴圈
7    Element st_child = null; //標準欄位的子節點
8    List child_list = elm.getChildren(); //銀行欄位的子節點容器
9    Iterator child_it = child_list.iterator(); //銀行欄位的子節點迴圈
0    Element child = null; //銀行欄位的子節點
第~38行
2    while (child_it.hasNext()) { //比較銀行欄位子節點與標準欄位子節點：
3      boolean justify = true; //欄位名稱不相同的標記符號
4      child = (Element)child_it.next();
5      String name1 = child.getName();
6      st_childlist = elm2.getChildren();
7      st_child_it = st_childlist.iterator();
8      while (st_child_it.hasNext()) { //跳下一個兄弟節點，繼續與標準欄位子節點的兄弟節點比較
第~46行 st_child = (Element) st_child_it.next();
0        if (child.getName().equals(st_child.getName())) {
1          //若找到相同欄位名稱，且銀行欄位子節點有子樹，則遞迴到下一層子節點比較
2          justify = false;
3          try { recursion(child,st_child);
4          }catch (IllegalDataException ex) {
5            System.err.println("各銀行節點沒有children，故爲leaf");
6            break; }}}
7  //若標準欄位子節點的兄弟節點沒有一個欄位名稱與銀行子節點相同(都不相同)時，則將銀行欄位子節點新增到標準欄位子
   增之後，銀行欄位子節點下的所有子節點也一併加入到節點之下
8    if (justify) {
第~55行 Element new_child = new Element(child.getName());
0      new_child.setAttribute("flag",child.getAttributeValue("flag"));
1      new_child.addContent(child.getText().trim());
2      elm2.addContent(new_child);
3      //新增之後，銀行欄位子節點下的所有子節點也一併加入到節點之下
4      Element st_new = elm2.getChild(new_child.getName().trim());
5      recursion(child,st_new); }}}
```

圖 4- 12 合併規則對應遞迴程式文件

第 25 行到第 38 行程式爲設定 A 銀行與標準欄位的相關變數，並程式迴圈走訪 A 銀行子節點所有兄弟節點，比較與標準欄位子節點的兄弟節點。

第 39 行到第 46 行程式爲欄位名稱相同，且銀行欄位子節點有子樹時，執行遞迴程式，比較其子樹節點的欄位名稱。

第 48 行到第 55 行程式爲欄位名稱皆不相同時，需將 A 銀行子節點新增到標準欄位中，同時，A 銀行子節點也一併加入到標準欄位。

4.3.3 表格取回模組

表格取回模組目的是將銀行欄位訊息合併後的結果，透過網路服務回傳給下一節的客戶端代理人系統存取。主要是運用 BEA Weblogic Workshop 實作整合服務成爲一個網路服務，以供客戶端可以動態且快速取回共通平台合併後的欄位元表格。

表格取回模組目的是將儲存在共通平台資料庫的 standard 資料表格欄位名稱資料(如圖 4-10 所示)，並製作成 XML 文件，透過網路服務的方式傳遞給客戶

端，其片斷程式如圖 4-13 所示。

```
/**
 * @jws:operation
 * @jws:return-xml xml-map::
 *   <standard xmlns="http://www.openuri.org/">
 *   <record xm:multiple="s in return">
 *   <source>{s.source}</source>
 *   <ordinal>{s.ordinal}</ordinal>
 *   <name>{s.name}</name>
 *   <flag>{s.flag}</flag>
 *   <target>{s.target}</target>
 *   <tokey>{s.tokey}</tokey>
 *   <path>{s.path}</path>
 *   <level>{s.level}</level>
 *   <sourceOther>{s.sourceOther}</sourceOther>
 *   <ordinalOther>{s.ordinalOther}</ordinalOther>
 *   </record>
 *   </standard>
 * ::*/
public Standard[] getting(String table){
  String url = "http://localhost/"+ table +"/Service1.asmx/CreateTagData";
  String url2 = "./applications/commonPlatform/xmlTag_standard.xml";
  doTask(table, url, url2);
  List result = standardDao.getAllStandard();
  Standard[] stdArray = new Standard[result.size()];
  for(int i=0;i<result.size();i++){
    Standard std = (Standard)result.get(i);
    stdArray[i] = std;}
  return stdArray;}
```

圖 4- 13 對應服務程式說明

第4節　帳戶代理人

客戶端使用帳戶代理人之前，各銀行需先提供帳戶明細查詢的網路服務，並註冊網路服務於註冊中心，當客戶透過代理人搜尋註冊中心，便能夠找到有提供網路服務的銀行位置，尋找到適合的銀行，輸入客戶在該銀行的帳號和密碼，即可透過代理人系統取得帳戶明細。倘若進行多行帳戶整合，需要先向共通平台要求一份帳戶整合欄位元表格，根據欄位元表格定義，呼叫各銀行帳戶明細的網路服務，形成帳戶彙總報表。

根據上述說明，建構客戶端代理人系統主要可分三個模組：與註冊中心溝通的註冊搜尋模組(UDDI Finding)、呼叫共通平台整合服務模組(Data Mapping)以及繫結各銀行網路服務的繫結服務模組(Data Binding)。此外，以多行帳戶整合爲基礎，建置加值服務，提供客戶直接進行帳戶之間轉帳匯款等應用服務。系統雛型架構如下圖 4-12 所示：

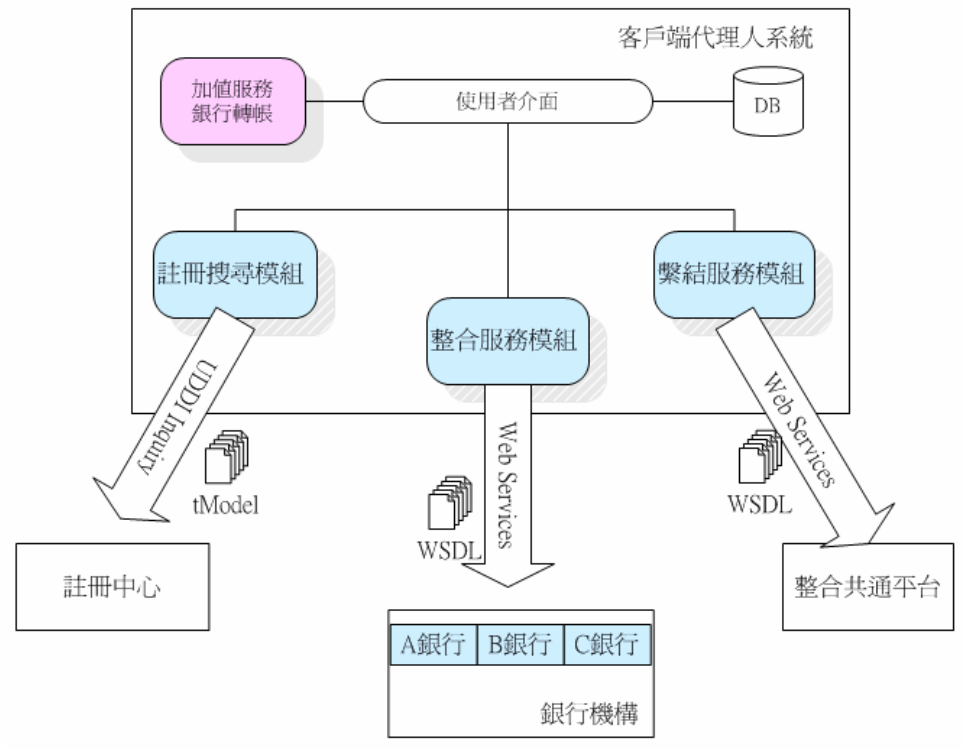

圖 4- 14　系統雛型架構圖

【資料來源：本研究】

下述將說明三個系統模組的設計理念及步驟：

1. 註冊搜尋模組：

客戶輸入關鍵字「bank」，查詢有在 UDDI 註冊的銀行商業服務，系統將透過本章第一節所提及 UDDI Inquiry API 介面。呼叫 Finding_business 介面將取得商業實體名稱，如：ABank、BBank 以及 CBank。再依各商業實體呼叫 finding_service 取得銀行所提供的網路服務，如：General Account、Check Account 以及 Currency Account。同時，呼叫 finding_tModel 取得 accessPoint 上的網址，如：http://140.119.75.215/abank/Service1.asmx/CreateXmlData 及 http://140.119.75,217/bbank/Service1.asmx/CreateXmlData。

經由 finding_business、finding_service 及 finding_tModel 三個程式介面取得資料後，儲存於資料庫 UddiAsset 表格中，其儲存結果如表 4-1 所示。

表 4- 1 UDDI 搜尋結果

類別	機構代號	機構名稱	科目別	服務名稱	網路服務位置
Bank	ABank	Bank_ABank	一般帳戶	General Account	http://localhost/abank/Service1.asmx/CreateXmlData
Bank	BBank	Bank_BBank	支票帳戶	Check Account	http://localhost/bbank/Service1.asmx/CreateXmlData
Bank	CBank	Bank_CBank	外幣帳戶	Currency Account	http://localhost/cbank/Service1.asmx/CreateXmlData

2. 整合服務模組：

使用者選擇使用者介面整合 A 銀行、B 銀行的帳戶明細，系統將執行 Data Mapping 模組呼叫共通平台整合服務，取得整合欄位元表格，並剖析檔儲存於資料庫「Standard」表格如表 4-2 所示：

表 4- 2 整合欄位表格

標準來源	標準順序	欄位代號	欄位名稱	子節點	A 銀行來源	A 銀行順序	B 銀行來源	B 銀行順序
0	1	DepAcctTrnRec	資產帳戶明細	1	0	1	0	1
1	1	BankAcctTrnRec	帳戶交易明細	2	1	1	1	1
1	2	ChkNum	支票號碼	-3	1	2	0	0
1	3	DeliveryMethod	通知方式	-3	0	0	1	2
2	1	TrnType	借貸別	-3	2	1	2	1
2	2	PostedDt	交易日期	-3	2	2	2	2
2	3	PostedTrn	交易時間	-3	2	3	2	3
2	4	TrnDesc	交易摘要	-3	2	4	0	0
2	5	CurAmt	交易金額	3	2	5	2	6
2	6	Memo	備註	-3	2	6	0	0
2	7	SPRefId	本次交易序號	-3	2	7	2	7
2	8	SPRefIdCorrect	交易參考	4	2	8	0	0
2	9	AcctBal	帳戶餘額	5	2	9	2	5
2	10	MultiBank	往來銀行	7	0	0	2	4
3	1	Amt	金額	-3	3	1	6	1
4	1	SPRefId	原交易序號	-3	4	1	0	0
4	2	CorrectAction	更正記號	-3	4	2	0	0
5	1	BalType	餘額類別	-3	5	1	4	1
5	2	CurAmt	餘額金額	6	5	2	4	2
6	1	Amt	金額	-3	6	1	5	1
6	2	ExpDt	逾期日期	-3	0	0	5	2
7	1	MultiBankId	銀行編號	-3	0	0	3	1
7	2	MultiStatus	往來狀況	-3	0	0	3	2

依據表 4-2 表格作爲 DTD 檔，採用第二章文獻探討的 DTD 檔關聯法(Relational DTD approach)，將欄位元表格轉換成七個資料表格，分別爲 table1、table2、table3、table4、table5、table6、table7，其表格間關係如下圖 4-15 所示。

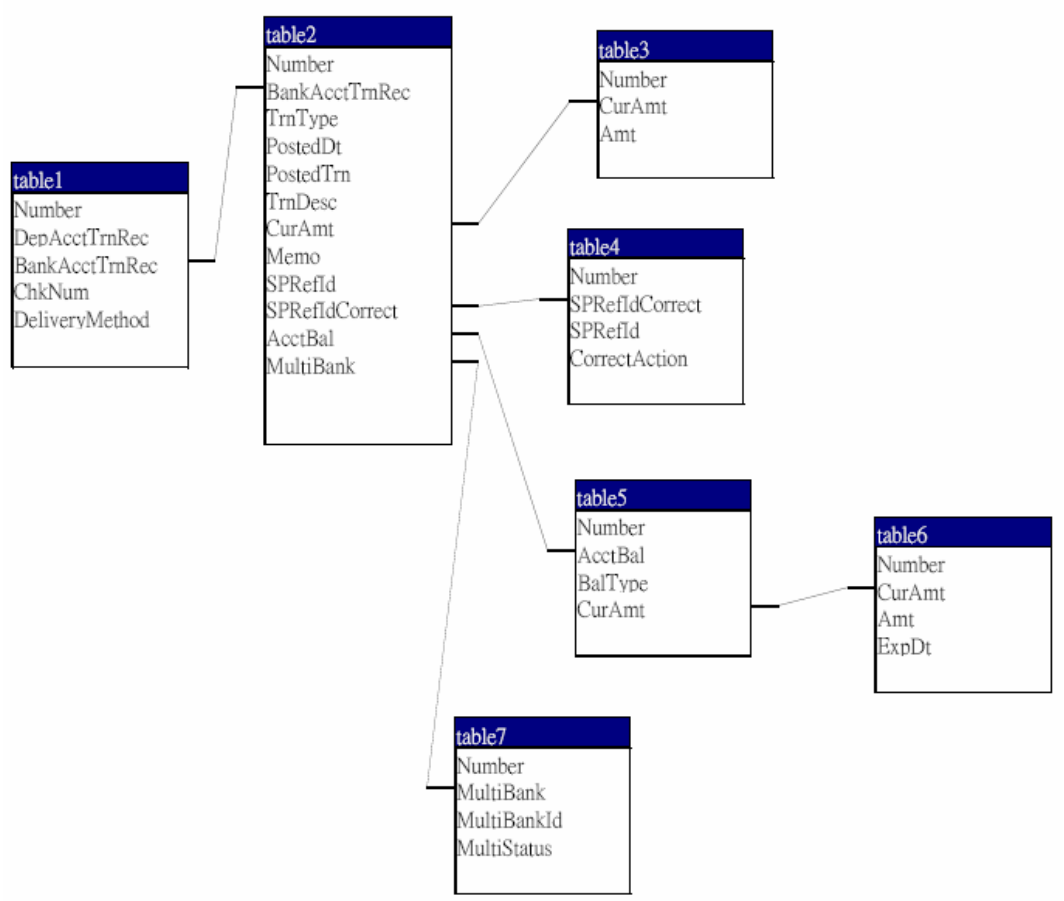

圖 4- 15 關聯表格關係圖

3. 繫結服務模組：

執行共通平台整合服務，得到上述關聯表格後，使用者輸入所在銀行開戶的帳號和密碼，系統將記錄帳號和密碼於本地資料庫中，方省去使用者重覆輸入的困惱，系統透過 Data Binding 模組，將帳戶欄位實際值取回，經由 JDOM API 介面剖析 XML 檔，放入本地資料庫。由於在剖析過程中，關聯表格做爲資料儲放的規範，根據標準來源、標準順序及子節點的定義，將實際值一對一對映到適當位置。設計概念如圖 4-16 所示。

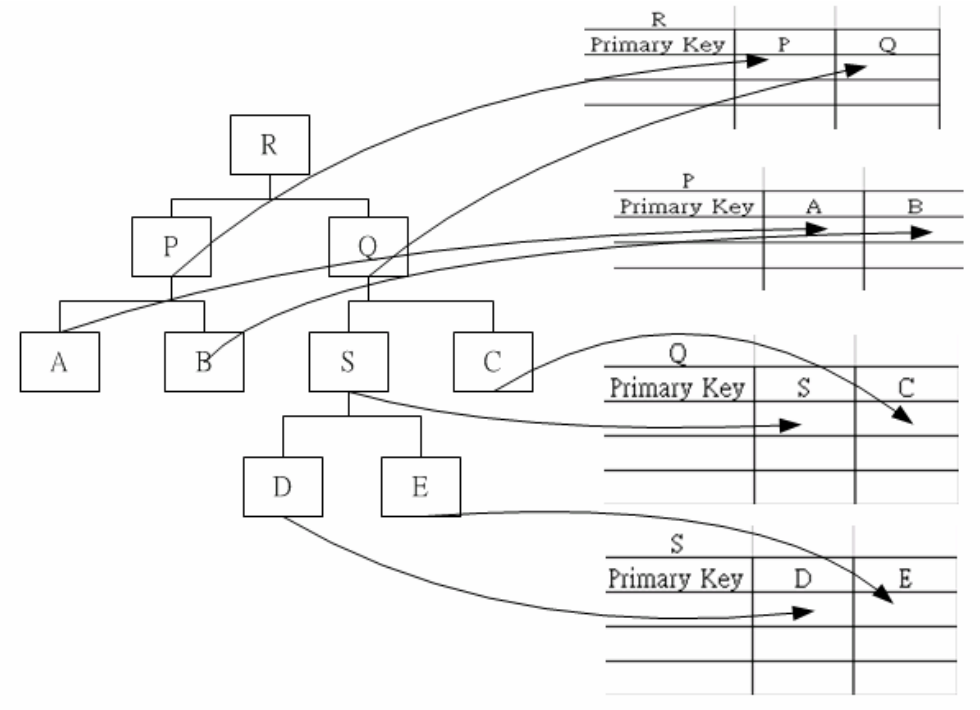
R
Primary Key
P
Q
R
P
Q
A
B
S
C
D
E
P
Primary Key
A
B
Q
Primary Key
S
C
S
Primary Key
D
E

圖 4- 16 實際值關係圖