整合關聯式資料庫於 CORBA 架構之封裝實作

楊仙維、廖宜恩
國立中興大學應用數學系
台中市南區國光路 250 號
TEL:(04)2840424 EXT. 511,621
EMAIL:swyoung@amath.nchu.edu.tw

摘要

隨著分散式系統及物件導向技術的發展，以物件爲基礎的分散式架構也受到高度重視。因此，OMG(Object Management Group)在 1990 年就制定了一個分散式物件溝通的標準 CORBA(Common Object Request Broker Architecture)，這是目前最受業界支持的分散式物件的標準。在本研究中，我們將介紹整合 CORBA 及關聯式資料庫的三層式架構，並使用 CORBA、Java 及 JDBC 設計一個 CORBA/RDB Wrapper，透過這個 Wrapper，就可以很容易實作出一個整合 CORBA 及關聯式資料庫的應用程式，而客戶端(Client)透過支援 Java 的瀏覽器就可以存取關聯式資料庫中的資料，進而達到分散式儲存及計算的目的。

Keywords：CORBA, Java, JDBC

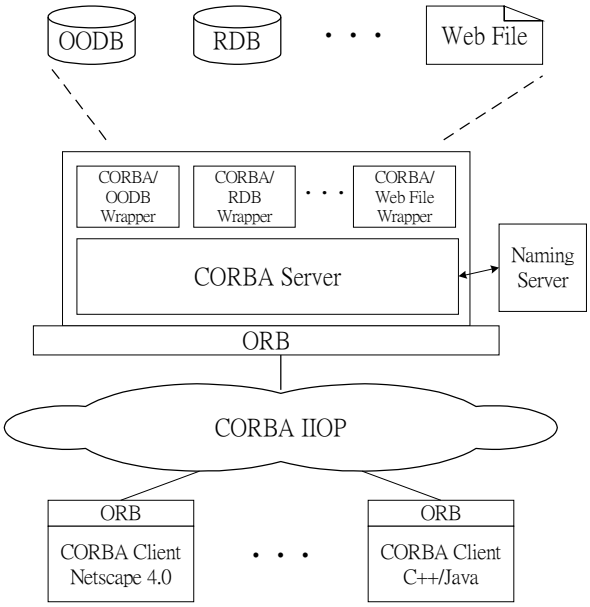
一、研究動機

近年來網際網路的蓬勃發展，使資料的分享及交流越來越普遍，但是資料的形式眾多，例如物件的資料、關聯式資料庫的資料、檔案形式的資料、網頁形式的資料、多媒體的資料…等，使資料的交流及溝通變的不容易。雖然有部分的資料形式已經有標準化，只要遵照標準就能容易的進行資料的交流溝通，但仍然有很多資料的形式是沒有標準的，是由個人或公司所自創的資料形式，所以如何將網際網路上的各種資料形式加以整合，讓使用者能夠輕易的存取這些資料，將是我們研究的目的。

在研究如何解決這個問題時，當然是要配合當前網際網路的環境以及未來的趨勢。當前網際網路的環境從以往的 Client/Server 兩層式的檔案存取，漸漸朝向多層式的服務方式，也就是在 Client 及 Server 之間加入一層或更多的 Application Server，Client 透過這個 Application Server 就能取得 Server 的服務，Application Server 也能將各種不同服務的 Server 加以整合管理，使 Client 能利用統一的介面，向 Application Server 要求各種不同的服務。另一個趨勢就是物件導向的分散式計算架構，目前以 Microsoft 的 DCOM [1,2]及 OMG [3,4]的 CORBA 爲主流，而 CORBA 爲開放的標準，最爲業界及學界所接受。所以我們研究的方向就是在物件導向的分散式架構上，整合網路上的異質性資料以及各種服務。

在不同的資料形式上，必須透過轉換才能溝通，所以我們針對不同的資料型態，設計不同的轉換器(wrapper) [5]，如圖一，將各個不同的 wrapper 集合在一起，使用者只要向 CORBA Server 要求，CORBA Server 就會自動使用適合的 wrapper 去向後端的資料來源 Server 存取資料。

本研究針對關聯式資料庫的部分來作，利用 CORBA 及 JDBC 來實作關聯式資料庫的 wrapper，透過此架構，使用者將擁有一致的介面，可輕易的得到分散在網路上的關聯式資料庫中的資料。



圖一、CORBA 環境整合異質性資料的架構

二、相關研究

本研究結合了WWW、Java、JDBC及CORBA等技術，開發一個在分散式物件架構下存取關聯式資料庫中資料的環境，讓不同的關聯式資料庫中的資料都能在這個環境中來整合。

1.Java

Java [6]是以 C++的優點爲基礎，它擷取了C++的長處，同時也並除了容易引起問題或是錯誤的部分，在這個簡潔的核心部分，它增加了garbage collection 垃圾收集(自動記憶體管理)、多執行緒(一個可使程式在同一時間處理不同事情的能力)和安全性特性。

2.JDBC

JDBC [7,8]是一提供執行 SQL 指令之 Java API，它是由一組以 Java 語言撰寫之類別與介面組成，JDBC 爲資料庫工具開發者提供了一標準 API，使得以純 Java API 開發應用程式成爲可能的事。

透過 JDBC，將可輕而易舉地在任何關聯式資料庫系統執行 SQL 指令，換言之，不再需要爲 Sybase、Oracle、Informix 等各種資料庫撰寫不同的程式；以 Java 語言撰寫應用程式，亦不需擔心跨平台的問題，只要透過 Java 與 JDBC 的整合，便能使程式設計師「撰寫一次，隨處執行」。

3.CORBA

CORBA(Common Object Request Broker Architecture)所定義的分散式物件架構與規格，可讓不同廠商所發展出來的 ORB(Object Request Broker) [9]物件元件可以在任何網路及作業系統上相互運作。CORBA 之標準可以讓 CORBA 物件相互引用，而不需要知道物件內部存取方式及被 Request 之物件是由哪一種語言所實作出的。CORBA 物件相較於傳統程式語言物件有以下幾個不同的地方：(1)可以置於網路上的任何一個地方；(2)可以和其他平台上的物件相互運作；(3)透過 IDL(Interface Definition Language)作爲物件介面的映對，可以用任何程式語言來撰寫。

三、系統架構

本系統是一個以 CORBA 及 JDBC 爲基礎的存取關聯式資料庫資料的系統，主要的功能就是結合 CORBA 與 JDBC，讓使用者在 CORBA 架構的分散式物件環境下，能夠存取關聯式資料庫中的資料。本系統利用 CORBA IDL 定義存取關聯式資料庫資料的介面，提供給 Client 使用。系統的架構圖如圖二所示。以下簡述系統大略的架構：

1.Client：

本系統的 Client 採用 Java Applet 及 Web 介面，Client 需到 Web Server 下載所需的 Web Page 及 Java Applet，還有 Client 所需的 ORB 也是跟隨 Java Applet 一起從 Web Server 下載，所以只要可以執行 Java Applet 的瀏覽器都可以執行 Client 程式。

2.ORB：

ORB 是 CORBA 的核心，所謂的 ORB 並不是一個獨立的的程式，而是一些程式庫及其他資源的集合，用來處理 Server 物件及 Client 的通訊。

3.Web Server：

Client 所需的 Web Page、Java Applet 及 ORB

都在 Web Server 上。

4.Naming Server：

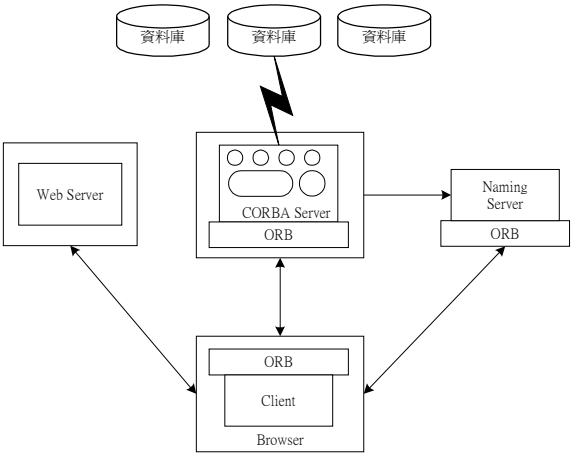
Server的物件必須在Naming Server [10]上註冊，然後 Client 就可以到 Naming Server 上得到 Server 物件的 reference。

5.CORBA Server：

本系統的 CORBA Server 是負責接受 Client 的要求，並依要求向後端的關聯式資料庫存取資料，連接關聯式資料庫的作法是使用 JDBC。

6.DBMS Server：

只要有適合的 JDBC Drivers 的關聯式資料庫都可使用。

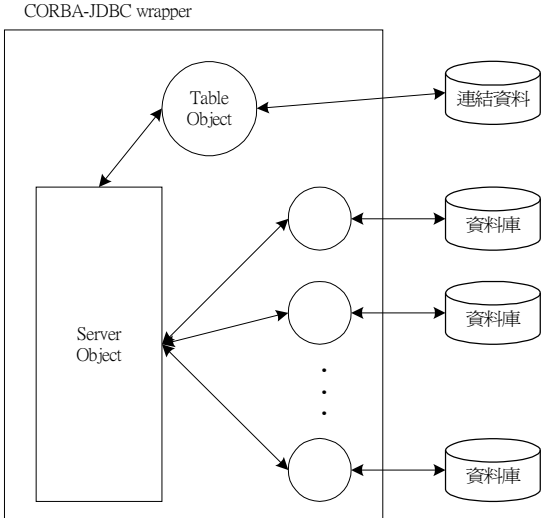


圖二、系統架構圖

四、系統實作

本研究主要實作爲 CORBA Server 及 Client 這兩部分，分述如下：

1.CORBA Server



在 CORBA Server 的部分我們要設計一個 CORBA-JDBC wrapper，作爲 CORBA 物件與關聯式資料庫的資料溝通整合之用。如圖三所示，CORBA-JDBC wrapper 主要由三個部分所組成，Server Object、Table Object 和 Query Object，下面我們將詳細敘述這三個部分：

圖三、CORBA-JDBC Wrapper

1.1 Server Object

這個部分主要是負責和 Client 溝通，以及分配查詢工作給 Query Object，最後再整合每個 Query Object 查詢的結果，將結果傳回給 Client。分述如下：

(1)和 Client 溝通：

Client 可以向 Server Object 提出兩種要求，一個是查詢，另一個是讀取查詢的結果。

(2)分配查詢工作：

因爲後端的資料庫分散在網路上，所以我們必須將工作分配給不同的 Query Object 去做，每個 Query Object 負責查詢一個資料庫。

在分配查詢的工作之前，必須知道有哪些資料庫可供查詢，以及如何連結這些資料庫，這個工作就交給 Table Object 去做。我們將可用的資料庫紀錄在一個資料庫檔案中，Table Object 會先查詢這些紀錄，然後告知 Server Object，Server Object 就可知道有哪些資料庫可供查詢，也就可以分配查詢的工作給 Query Object 了。

(3)整合查詢結果：

在每個 Query Server 查詢結束後，Server Object 會將這些結果整合在一起，並依照排序的欄位進行排序，然後將結果傳回給 Client。

1.2 Table Object

這個部分負責查詢有哪些資料庫可以讓 Client 查詢。

我們將這些資料庫的資訊紀錄在一個資料庫檔案中，包括連結該資料庫所使用的 JDBC Driver 名稱、該資料庫的 URL 位址、登入所需的

帳號及密碼、資料表的名稱和該資料庫所屬的公司名稱。如此我們就能管理這些資料庫，新增、修改或刪除這些資料庫時，只要更改資料庫的資訊即可，不必更改到程式的部分。

Table Object 連結到上述的資料庫檔案，查詢這些資料庫的資訊後，Server Object 可透過 Table Object 的介面來讀取連接這些資料庫所需的資訊，以及資料庫的數量，然後就可以決定使用多少的 Query Object 及分配查詢給這些 Query Object。

1.3 Query Object

這個部分負責查詢資料庫，並提供介面給 Server Object，讓 Server Object 可以讀取查詢後的結果。

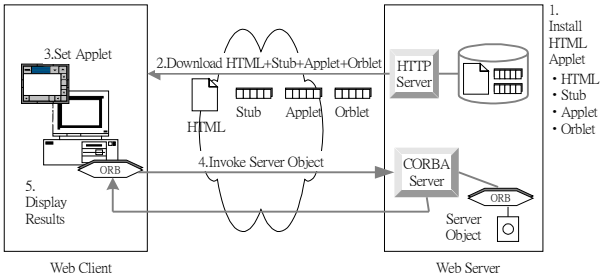
Query Object 的數量隨著資料庫的數量而變，是由 Server Object 根據 Table Object 所得到的資料而產生，而每個 Query Object 所做的工作就是利用 JDBC Driver 連結資料庫、送出查詢和得到結果。JDBC 連結資料庫的過程如下：

(1)載入驅動程式
(2)建立連結
(3)建立 JDBC 指令
(4)執行指令
(5)擷取結果資料
(6)結束查詢指令和資料庫連結

2.Client

本系統的 Client 是使用 Java Applet 顯示成網頁的形式，所需要的執行環境只要是可以執行 Java Applet 的瀏覽器就可以了，並不受作業系統及平台的影響，而執行 Client 所需要的網頁、Applet 和 ORB，都是從 Web Server 下載回來。Client 與 Server 溝通的詳細步驟如圖四所示，下面將敘述這五個步驟：

- 步驟 1：將執行 Client 需要的 HTML、Stub、Applet 和 Orblet 放到 Web Server 上特定的位置。這裡的 Web Server 只是單純用來放網頁及被下載的檔案，並沒有執行任何其他服務，所以任何一種 Web Server 及任何平台都可適用於本系統。
- 步驟 2：用可執行 Java Applet 的瀏覽器，連線到系統的 Web Server，此時瀏覽器會從 Web Server 下載網頁，包括 HTML 檔案以及 HTML 中所使用到的 Applet 和 ORB。
- 步驟 3：下載完所需的檔案後，瀏覽器會啓動 Java Applet，包括啓動 ORB 以及向 Naming Server 詢問 Server Object 的位置，這些將在下一節 Naming Service 中有更詳細的敘述。
- 步驟 4：向 Server Object 提出要求，只要使用 IDL 定義出來的介面，就可以呼叫 Server Object 執行服務。
- 步驟 5：將 Server Object 送回的執行結果顯示出來。

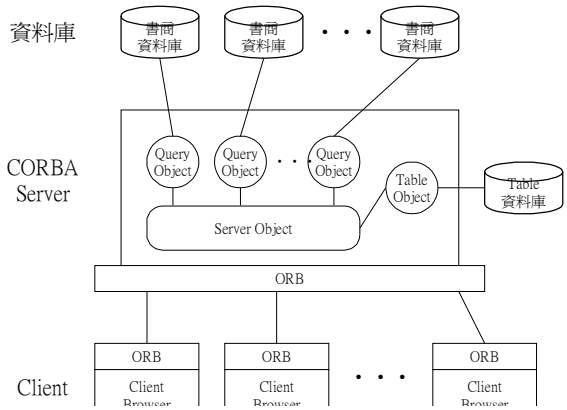


圖四、Client 與 Server 溝通的步驟

五、應用實例

我們利用本系統的架構，設計一個網路上書局的書籍資料的查詢服務，就是利用本系統來連接網路上的線上書局或書商的書籍關聯式資料庫，然後使用者就可以透過本系統來查詢這些書局或書商紀錄在資料庫內的書籍。

這個實作的架構如圖五所示，主要分成資料庫、CORBA Server 和 Client 三層。分述如下：



圖五、查詢線上書商之書籍資料庫的實作架構

1.資料庫

在資料庫這一層中，我們模擬書局或書商的書籍資料庫，將這些書籍資料庫分散在不同的機器上，在此假設每個書商的書籍資料庫中只有一個表格，表格中的欄位爲：分類、書名、頁數、書碼、作者、年份、定價、出版公司、ISBN。如果資料庫中有兩個以上關聯的表格時，我們可用 SQL 的 join 方式，仍然可以進行查詢。另外我們建立一個 Table 資料庫，用來紀錄連結這些書局或書商的書籍資料庫的一些資訊，然後利用這些資訊，系統就可以用 JDBC 連結到這些資料庫。

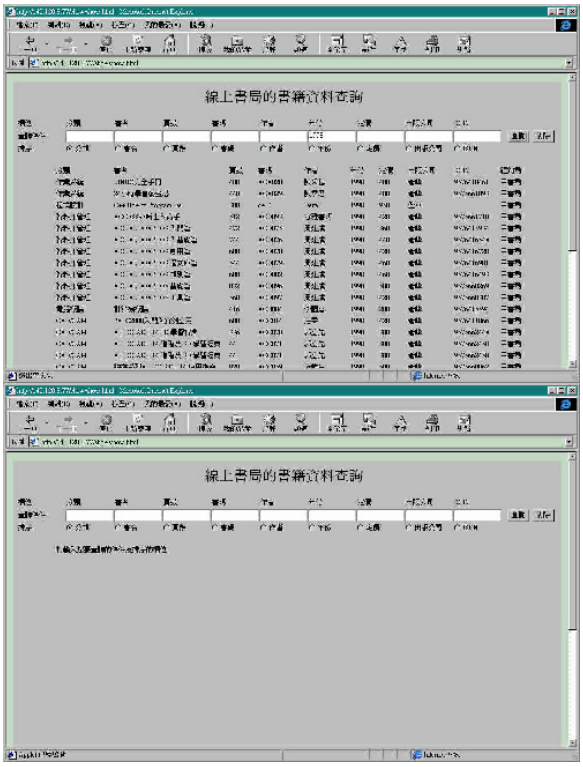
2.CORBA Server

在 CORBA Server 這一層中，主要分成三個部分，下面介紹這三個部分以及它們的作用：

- Server Object 負責接收 Client 的查詢命令，並分配查詢工作給 Query Object，還有整合 Query Object 傳回的查詢結果。
- Table Object 負責查看 Table 資料庫中記載的連接後端資料庫的資訊，然後告訴 Server Object，Server Object 就可用這些資訊來建立 Query Object。
- Query Object 負責連接資料庫、查詢資料庫的資料以及傳回查詢結果。

3.Client

在 Client 這一層中，我們用 Java Applet 來實作，使用者使用可執行 Java 的瀏覽器，連接到系統的網頁上，就可以進行查詢。在使用者操作的介面上，我們設計了表單(Form)的形式，也就是有文字欄位(Text Field)及確認盒(Checkbox)，讓使用者填上查詢的條件和選擇排序的欄位，按下查詢的按鈕後，就能送出查詢命令給 CORBA Server，查詢得到的結果也會顯示在網頁上。如圖六和圖七，分別顯示表單的畫面和查詢結果的畫面。



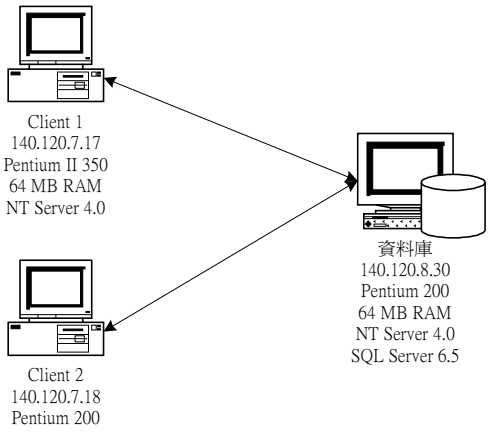
圖六、Client 的使用者介面

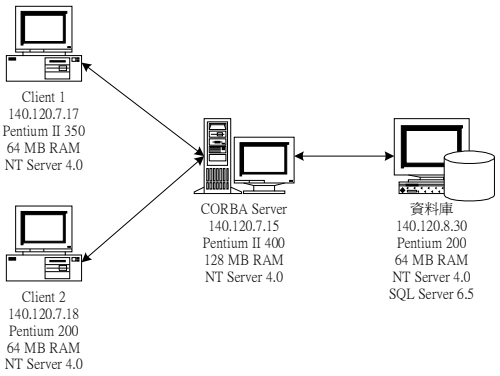
圖七、Client 顯示查詢結果的畫面

六、兩層式與三層式架構之效能評估

我們利用上一節中敘述的應用實例，修改成如圖八和圖九的兩層式架構及三層式架構，然後針對這兩種架構進行效能的評估。

圖八、兩層式架構效能評估的環境





圖九、三層式架構效能評估的環境

我們所作的效能評估，是測量 Client 查詢資料庫中的資料所需花費的時間，我們將 Client 從開始查詢到顯示結果的這段時間分成以下四個階段：

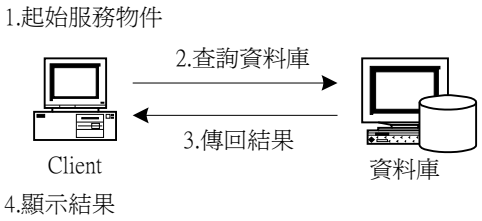
第一階段：啓始服務的物件

第二階段：查詢資料庫

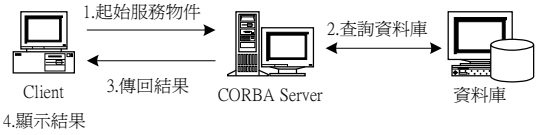
第三階段：傳回結果

第四階段：顯示結果

如圖十和圖十一所示，爲兩層式架構及三層式架構的運作的四個階段，測量的結果如表一和表二所示。



圖十、兩層式架構運作的四個階段



圖十一、三層式架構運作的四個階段

Client 1 (140.120.7.17、Pentium II 350) 時間單位：秒						
架構	資料數	第一階段	第二階段	第三階段	第四階段	全部
兩層式	200筆	0.583	0.318	0.001	5.329	6.240
三層式	200筆	0.942	0.199	0.092	5.303	6.537
兩層式	400筆	0.605	0.512	0	10.674	11.791
三層式	400筆	0.935	0.371	0.116	10.650	12.071

表一、Client 1 查詢資料庫所花費的時間

Client 2 (140.120.7.18、Pentium 200) 時間單位：秒						
架構	資料數	第一階段	第二階段	第三階段	第四階段	全部
兩層式	200筆	1.285	0.482	0.002	1.838	3.607
三層式	200筆	2.281	0.360	0.121	1.920	4.252
兩層式	400筆	1.219	0.685	0.001	3.680	5.584
三層式	400筆	2.193	0.475	0.148	3.757	6.574

表二、Client 2 查詢資料庫所花費的時間

從表一及表二中，我們可以得到下列推論：

1.第一階段中，兩層式比三層式花費較少時間，是因爲兩層式由 Client 自己啓動自己的服務物件，而三層式需由 Client 向 CORBA Server 要求服務物件。

2.第二階段中，三層式比兩層式花費較少時間，是因爲三層式由處理能力較強的 CORBA Server 負責查詢資料庫，而兩層式由 Client 自己負責查詢的工作。

3.第三階段中，兩層式比三層式花費較少時間，是因爲兩層式自己查詢後就可得到結果，而三階層的查詢結果需由 CORBA Server 傳回給 Client。

4.第四階段中，因爲顯示結果所花費的時間只和資料筆數有關，所以兩種架構並沒有差別。

5.從全部花費的時間看來，三層式比兩層式還多一些，主要就是三層式需花時間在 Client 與 CORBA Server 間的溝通，但從三層式的種種優點，如方便管理、較安全、較有彈性…等，確實可補其效能上與兩層式的些許差異，甚至在整體上來說，三層式架構比兩層式架構出色。

七、結論與未來發展方向

在本篇論文中，我們使用 CORBA 與 JDBC 設計一個三層式架構，使我們能在分散式物件環境中，整合關聯式資料庫的資料，本系統有以下幾個特點：

(1)可與網際網路上的服務結合。網際網路上有許多的服務是 Web Server 結合關聯式資料庫，例如電子商務、線上購物等，本系統提供了一個這方面的解決方案。

(2)本系統將連結後端資料庫的資訊，記錄在一資料表中，而透過此資料表，就可以管理連結後端資料庫的事宜，例如新增或刪除一個資料庫，如此在後端資料庫有所變動時，更改資料表即可，不必修改程式。

(3)本系統結合 CORBA 與 JDBC 的優點，提供使用者一個透通性(transparence)的服務，也就是使用者不用知道後端資料庫的所在位置，以及如何去連接那些資料庫，只要向系統提出查詢的條件即可，這些查詢的工作都由系統負責。

(4)本系統目前只使用一種 JDBC Driver，未來只要再增加其他的 JDBC Driver，就可以再連結其他的關聯式資料庫系統，而且同時使用多個不同的 JDBC Driver，就可以同時連接不同的關聯式資料庫了。

本系統還可有下列幾個改進的地方：

(1)目前在管理連結後端資料的事宜方面，都是人工去設定如何連結，以及必要時的資料轉換，未來希望發展一個智慧型的 wrapper，只要提供後端資料的位置，就能自動分析如何去連結及整合其資料。

(2)本系統目前針對關聯式資料庫來作 wrapper，未來希望能再發展其他方面資料的 wrapper，例如物件導向資料庫、Web 檔案、文字檔案及多媒體的資料等，然後將這些 wrapper 整合成一個 Service，提供一個跨資料形式的服務，例如搜尋相關而不限定檔案格式的資料，使在網路上存取資料時，不必考慮資料的格式。

從網際網路的發展看來，由以資料爲導向漸漸便成爲以服務爲導向的趨勢，與本系統未來研究的方向非常契合，因爲以服務爲導向，是讓使用者不必知道資料的形式及來源的位置，只要提出要求，就能得到適合的服務，如何搜尋資料及整理資料都交給 Server 來作即可，所以本系統在未來的網際網路的應用及服務上，能提供不少的幫助。

而在連結及整合不同的資料形式時，會面臨如何將不同形式的資料，整合成一致的資料，以及如何能將所有種類的資料包含到我們的系統中，這些困難也有待我們繼續努力研究，以得到最佳的解決方案。

參考文獻

[1] M. Horstmann and M. Kirtland, “DCOM Architecture,” Microsoft Corporation, July 23, 1997. http://msdn.microsoft.com/isapi/msdnlib.idc?theURL=/library/backgrnd/html/msdn_dcomarch.htm

[2] “DCOM Technical Overview,” Microsoft Corporation, November 1996. http://msdn.microsoft.com/isapi/msdnlib.idc?theURL=/library/backgrnd/html/msdn_dcomtec.htm

[3] “CORBA Overview,” Object Management Group. http://www.infosys.tuwien.ac.at/Research/Corba/OMG/arch2.htm#446864

[4] Doug Schmidt, “Overview of CORBA,” Object Management Group. http://www.cs.wustl.edu/~schmidt/corba-overview.html

[5] David Wells, “Wrappers,” Object Services and Consulting, 1996. http://www.objs.com/survey/wrap.htm

[6] “Java White Papers,” SunSoft, Inc. http://java.sun.com/docs/white/index.html

[7] JDBC API. http://java.sun.com/products/jdk/1.2/docs/guide/jdbc/

[8] JDBC Drivers. http://java.sun.com/products/jdbc/jdbc.drivers.html

[9] “CORBA: Common Object Request Broker Architecture and Specification Revision 2.0,” Object Management Group, July 1995.

[10] Visigenic VisiBroker for Java Naming Service. http://www.visigenic.com.