國立交通大學

資訊科學與工程研究所

碩 士 論 文

交 通 大 學 校 園 資 訊 電 子 化 及 其 整 合 ：

分 散 式 事 件 驅 動 管 理 資 料 庫

NCTU Administrative e-Office System & Integration：

Distributed Event-based Middleware Database Management

研 究 生：蔡東泰

指導教授：謝筱齡、蔡文能　教授

中 華 民 國　九 十 五　年 七 月

交通大學校園資訊電子化及其整合：分散式事件驅動管理資料庫

NCTU Administrative e-Office System & Integration：Distributed Event-based Middleware Database Management

研 究 生：蔡東泰　　Student：Dung-Tai Tsai

指導教授：謝筱齡、蔡文能　Advisor：Sheau-Ling Hsieh、Wen-Nung Tsai

國 立 交 通 大 學

資 訊 科 學 與 工 程 研 究 所

碩 士 論 文

A Thesis

Submitted to Institute of Computer Science and Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

June 2006

Hsinchu, Taiwan, Republic of China

中華民國九十五年七月

交通大學校園資訊電子化及其整合：

分散式事件驅動技術應用於共同資料庫

研究生：蔡東泰　　指導教授：謝筱齡、蔡文能

國立交通大學資訊科學與工程研究所

摘要

在本研究中，針對問題我們提出了一個架構—DEMCD。DEMCD 兩大核心技術是 JMS 與 SSNS。透過這樣的架構，可以提供各部門之間資訊交換的管道，解決部門之間因爲異質平台所造成的問題。DEMCD 架構最初的構想就是以迅速更新資料爲主，因此利用 SQL-NS 所提供的功能，套用於共同資料庫上，來產生資料庫變更事件的觸發，最後再將這些更新的資訊，以主動式傳輸的 MOM 架構來傳遞資訊給所需要的部門。所以在 DEMCD 架構中，不同部門之間的系統資料，可以在最快速的時間之內，達成資料庫內容的一致性。

關鍵字：交通大學、校務資訊整合、中介軟體、事件驅動、JMS (Java 訊息服務)、SSNS (SQL Server 通知服務)

NCTU Administrator e-Office Systems and Integration：Distributed Event-based Middleware Technologies for Common Database System

Student：Dung-Tai Tsai　　**Advisor**：Dr. Sheau-Ling Hsieh、Dr. Wen-Nung Tsai

Institute of Computer Science and Engineering
College of Computer Science
National Chiao Tung University

Abstract

In this research, we proposed an architecture, DEMCD. The two core technologies of DEMCD are JMS and SQL-NS. Using DEMCD architecture, we can provide a platform on which systems can communicate to each other to solve the problems of heterogeneous systems. The original concept of DEMCD is primary based on changing data swiftly, so, with the functionalities provided in SQL-NS, we can trigger the database changed events in Common Database, then provide the renewed information to the divisions which need it by active MOM architecture in order to keep the consistence of database in different divisions as soon as possible in DEMCD Architecture.

Keyword：NCTU、Administrative e-Office、Middleware、Event-based、JMS (Java Service)、SSNS (SQL Server Notification Services)

誌謝

在這長不算長，短不算短的兩年之中，我增長了很多知識，不管是課業相關、或是待人處世的道理，我都獲益良多。感謝家人對我的支持，如果沒有他們的鼓勵，也許今天我沒辦法在這裡求學。

感謝我的指導教授謝筱齡老師，這兩年來對我們的栽培與照顧，在我們有疑惑的時候，不辭辛勞的替我們解答。除此之外，也包容我們草草處理事情的態度。在她身上我學到了許多做人處事的道理，也懂得以更謙卑的態度來面對未來的挑戰。

感謝這兩年陪伴的實驗室夥伴，Moder 游名樑、Virgogo 黃宏文、Datokuen 沈嘉崑以及後來一年才加入我們實驗室的楊葉薰，我們一起經歷過無數個作業與考試的難關，如果沒有他們的幫忙，也許我一個人無法完成這些挑戰。也感謝其他實驗室的同學們，bry 洪振洲、尤清華以及噗噗威 黃威力，因爲有了他們，在這兩年的時間，更顯的多彩多姿。

還要感謝很多在這兩年之間陪伴我的夥伴們，我相信，如果沒有你們，不會有今天的我。

東泰

目錄

表目錄

圖目錄

1. 緒論

1.1. 研究背景及動機

歷史悠久的國立交通大學[1]，自四十七年成立至已經有四十八年。學校規模已經相當龐大，行政組織也相當完整，校內業務以部門而分工，各司其職。例如，註冊組負責處理學生資料、人事室負責教職員資料…等。校內雖然有組織化的分工機制，但各處室業務依然相當繁多。

資訊科技的急速發展已對我們的日常生活產生重要影響，資訊科技也成爲一個提供學校服務的重要途徑，在此趨勢下，各部門不斷地發展行政業務電子化，例如，文書組的郵件管理系統、圖書館的櫃檯流通系統…等。業務電子化發展致力於提高學校業務服務品質、縮短行政時程，讓教職人員、學生能夠更有效率地處理與學校之間的事務

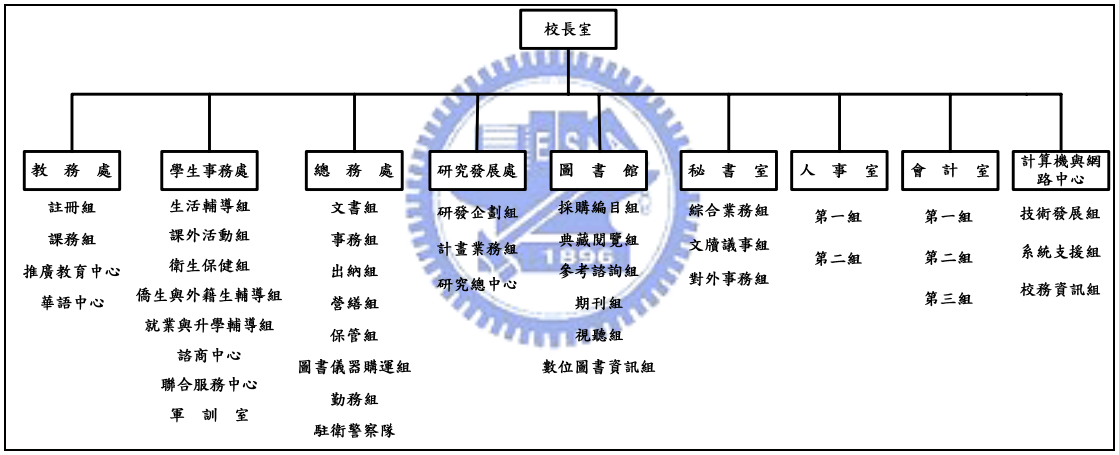


圖 1：交通大學行政組織圖

如今，現有系統在處理獨自業務上都已經相當成熟而且穩定，但是在處理各部門之間的業務時，常常會用到非本部門所維護的資料，資料來源來自其他部門，所以在處理業務之前，必須仰賴其他部門提供相關資料，業務才可以正常運作，因此在部門之間，必須有個良好的溝通管道，讓彼此之間的資料可以互通有無。但是在目前校內系統發展中，資料傳送的的機制尚未健全。

因校內缺乏系統之間傳送資料的管道，所以在資料的表現上無法達成一致性的目標，在處理業務的過程中，常常會出現資料不正確的問題。爲了改善上述這些問題，所以本研究中，希望透過分析這些問題的特性，提出整體的架構，期望解決資料互通上的問題，以提高學校在處理業務上的效率與正確性。

1.2. 研究目的

本研究的目的爲了解校內需求，發現整合上的癥結，並且提出解決問題的系統架構—DEMCD (Distributed Event-based Middleware for Common Database System)。

DEMCD 架構以 MOM (Message-Oriented Middleware)[2][3][4][5] 與 SSNS (SQL Server Notification Services)[6][7] 爲基礎，提供校內系統一個資料交換的平台。透過 DEMCD 架構，校內系統可迅速、有效率且可靠地彼此之間交換資料，甚至可以簡化目前系統的流程，利於快速開發新系統。

1.3. 論文架構

本論文一共分爲五章，第一章爲《緒論》，對研究動機、背景、目的以及論文整體架構做一簡單的說明。第二章爲《需求分析》，詳述訪查學校目前各部門系統運作狀況以及問題…等資料。第三章爲《方法》，介紹在研究中提出架構的基礎方法，包含了 MOM (Message-Oriented Middleware) 與 SSNS (SQL Server Notification Services) 兩種技術。第四章爲《架構》，詳述在本研究中提出有效解決在需求分析時所發現問題的架構，包含架構中各元件功能與設計方法。第五章《未來工作與結論》，總結本研究的成果，並且對未來可能的研究方向加以說明。

第2章 需求蒐集及分析

本研究的目的是整合校內開發已久的系統，先得知各部門所開發系統情況，並且分析整合上的需求。透過訪查的結果，可以歸類出幾個系統整合層面上的問題，本章節中將詳述這些問題。

需求分析[8]往往是系統開發的一個重要的環節，因此本研究透過需求分析了解校內各部門之間的需求，希望能夠實際的探討校內行政系統的問題，而不是天馬行空的討論。

以下爲各部門訪問時主要提出的幾個問題：

1. 貴部門所負責的業務
2. 目前正在使用的電子化系統
3. 這些電子化系統使用架構爲何？
4. 與其他部門有互相配合的業務？
5. 部門之間配合上是否有改善空間或者不方便之處？
6. 其他

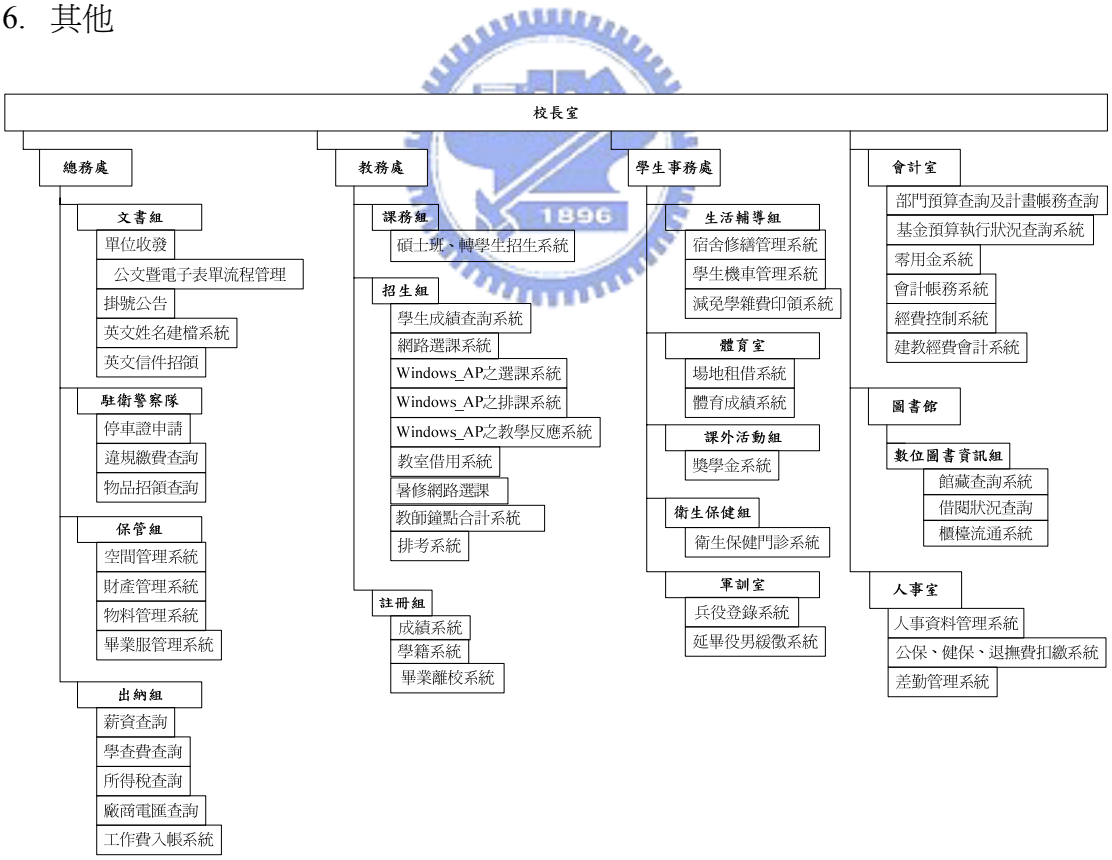


圖 2：校內目前運作系統概況[1]

各部門的訪談的詳細情形整理於後面章節。

2.1. 文書組

2.1.1. 負責業務

文書組負責處理學校中文件相關的業務[9]，如下面所示：

(1) 信件處理及學生掛號信件管理
(2) 收文及公文電子交換
(3) 發文
(4) 公文繕打及電子發文
(5) 檔案管理
(6) 檔案調閱
(7) 蓋用印信及管理

2.1.2. 郵件管理系統

掛號信件管理是文書組的業務之一，校內教職員生的掛號信件都將透過文書組負責發送，因此發展郵件管理系統[10][11]以紀錄掛號信件的資訊以及通知收件人領信…等功能。以下爲郵件管理系統處理流程：

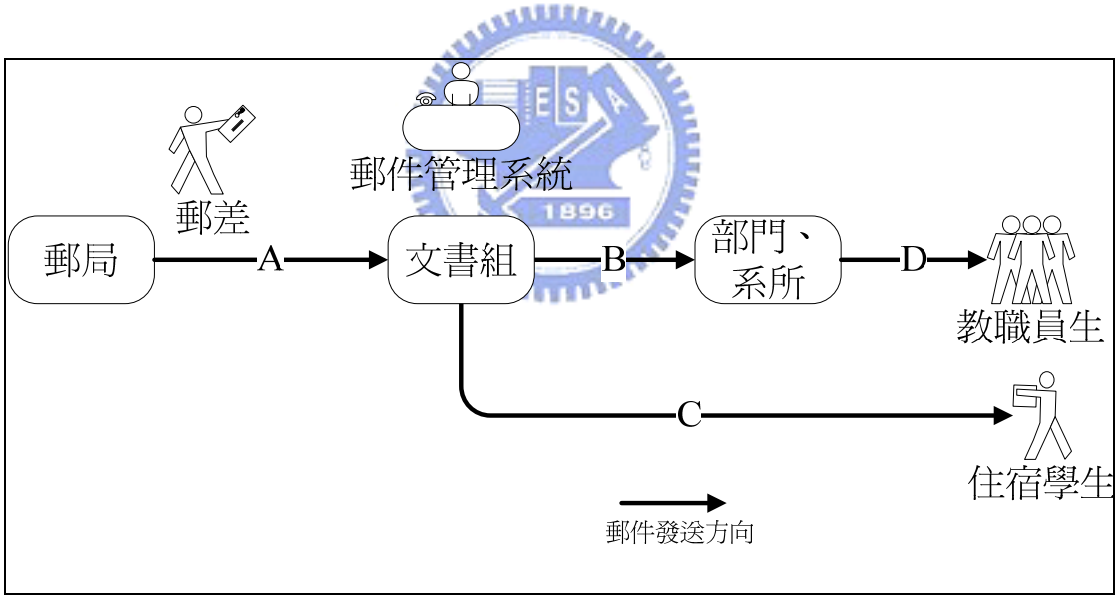


圖 3：郵件管理系統流程圖

A. 郵差將收件地址爲交通大學的郵件集中統一送至文書組處理
B. 郵件管理人員處理之後，各部門、系所派工友至文書組將單位的郵件領回
C. 住宿學生親自至文書組領取個人信件
D. 部門、系所的教職員生，至各部門、系所領取個人信件

2.1.2.1. 環境與架構

(1) 作業系統：Windows Server 2003
(2) 資料庫系統：Microsoft SQL Server 2000
(3) 網頁伺服器：Microsoft IIS
(4) 系統語言：ASP

2.1.2.2. 管理系統功能

郵件管理系統功能大致上可以分爲： 1) 紀錄郵件資訊 2) 領取通知 3) 郵件領取 4) 英文姓名建檔，以下針對這些功能分別介紹。

2.1.2.2.1. 紀錄郵件資訊

郵件所需要記錄的資訊有：

1. 收件者姓名
2. 收件者所屬單位、系所
3. 郵件條碼 (其中包含有：郵件號碼、投寄地、郵別，以刷條碼方式輸入)
4. 郵件建檔時間 (系統自動產生)
5. 備註 (可空白)

上列第 2 點，可以自動化處理：
當輸入收件者姓名(教職員、住宿學生)之後，可自動地帶出其所屬單位、系所。

例如：輸入戴淑欣 (文書組組長)，可自動帶出文書組、信箱號碼。

中 文 姓 名：	戴淑欣
信箱號碼：	072 (文書組)

2.1.2.2.2. 通知領件

依照領件者身分不同，通知的方式有兩種：

1. Email 通知 – 若收件者爲教職員、非住宿學生，以 Email 通知，至其所屬單位領取信件。
2. BBS 公告 – 若收件者爲住宿學生，則在 BBS 板上統一公告，至文書組領取郵件。

2.1.2.2.3. 郵件領取

1. 單位領件
   - 由單位、系所派人員至文書組領取其郵件。其中包含有教職員工、非住宿學生的郵件。

- 領件時所必須紀錄的資訊有：
  i. 單位名稱 (整批領取)
  ii. 領件人 (可輸入職員代碼、學號，透過資料庫查詢帶出姓名)
  iii. 領件時間 (系統自動產生)

2. 學生領件
   - 住宿學生攜帶學生證至文書組領取信件。
   - 領件時所必須紀錄的資訊有：
     i. 宿舍名稱 (列出此宿舍中所有尚未領取郵件)
     ii. 郵件編號 (領取兩件以上，可用”，”來隔開編號)
     iii. 領件人 (可輸入學號，透過資料庫查詢帶出姓名)
     iv. 領件時間 (系統自動產生)

2.1.2.2.4. 英文姓名建檔

緣由：

英文姓名書寫方式很多樣化，同一個人可能會有以下幾種英文名字：

- Dung-Tai, Tsai (中文姓名羅馬拼音，姓氏放後面)
- Tsai, Dung-Tai (中文姓名羅馬拼音，姓氏放前面)
- D.T. Tsai (羅馬拼音簡寫)
- Tony (非羅馬拼音之英文姓名)

當郵件上面所標示的姓名不足以清楚地找到收件者的時候，可能會造成此郵件無法及時的送達到收件者的手上。所以我們需要蒐集收件者的英文姓名，使得郵件管理系統有更多的資訊來判斷收件者的身分。

方法：

建置一個新的英文姓名建檔系統，可讓教職員、學生建立/管理自己的英文姓名。當在郵件管理系統中輸入收件者姓名的時候，會自動搜尋英文姓名建檔系統中的英文姓名，即可查詢收件者所屬的單位、系所。

2.1.2.3. 系統資料庫

以下介紹文書組郵件管理系統的資料庫，包括系統資料來源、系統資料表以及系統資料更新。

2.1.2.3.1. 系統資料來源

郵件管理系統之中搜尋收件者功能，需要有人事室教職員資料，用以查詢教職員的部門；生輔組的學生住宿資料以及註冊組的學生資料，用以查詢學生住宿情況或者系所，所以在資料方便需仰賴人事室、生輔組以及註冊組提供，才有辦法完成系統功能。以下爲郵件管理系統資料來源示意圖：

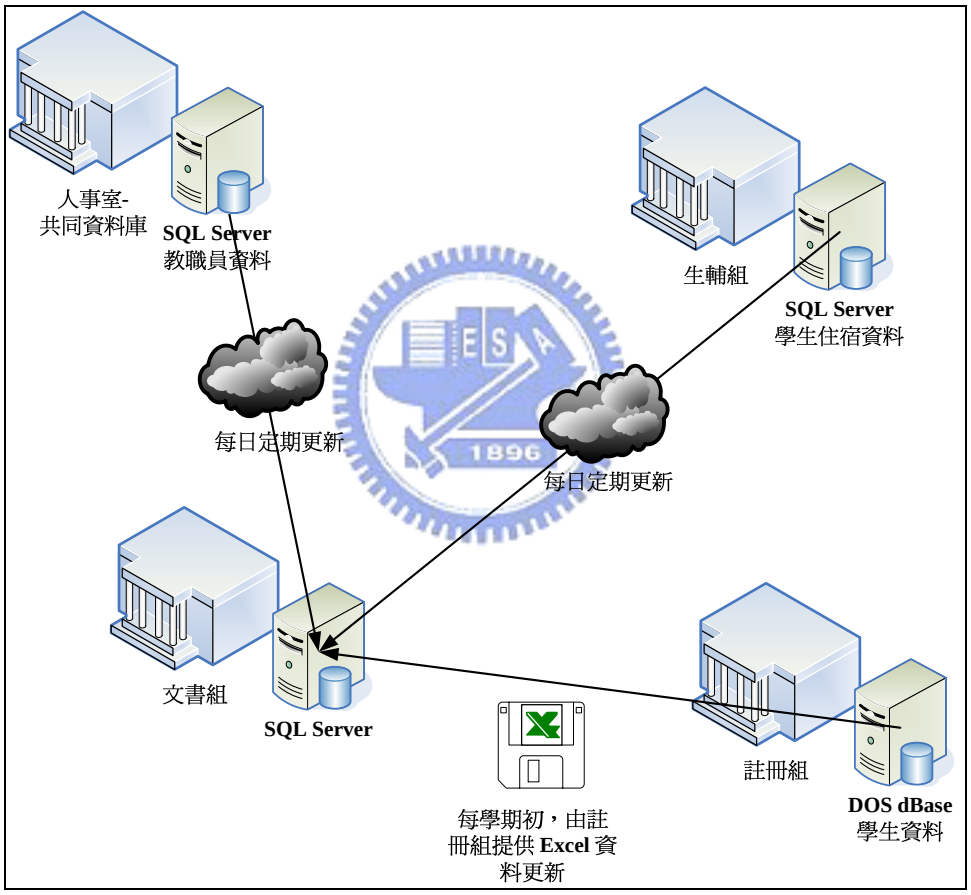


圖 4：郵件管理系統資料來源圖

2.1.2.3.2. 系統資料表

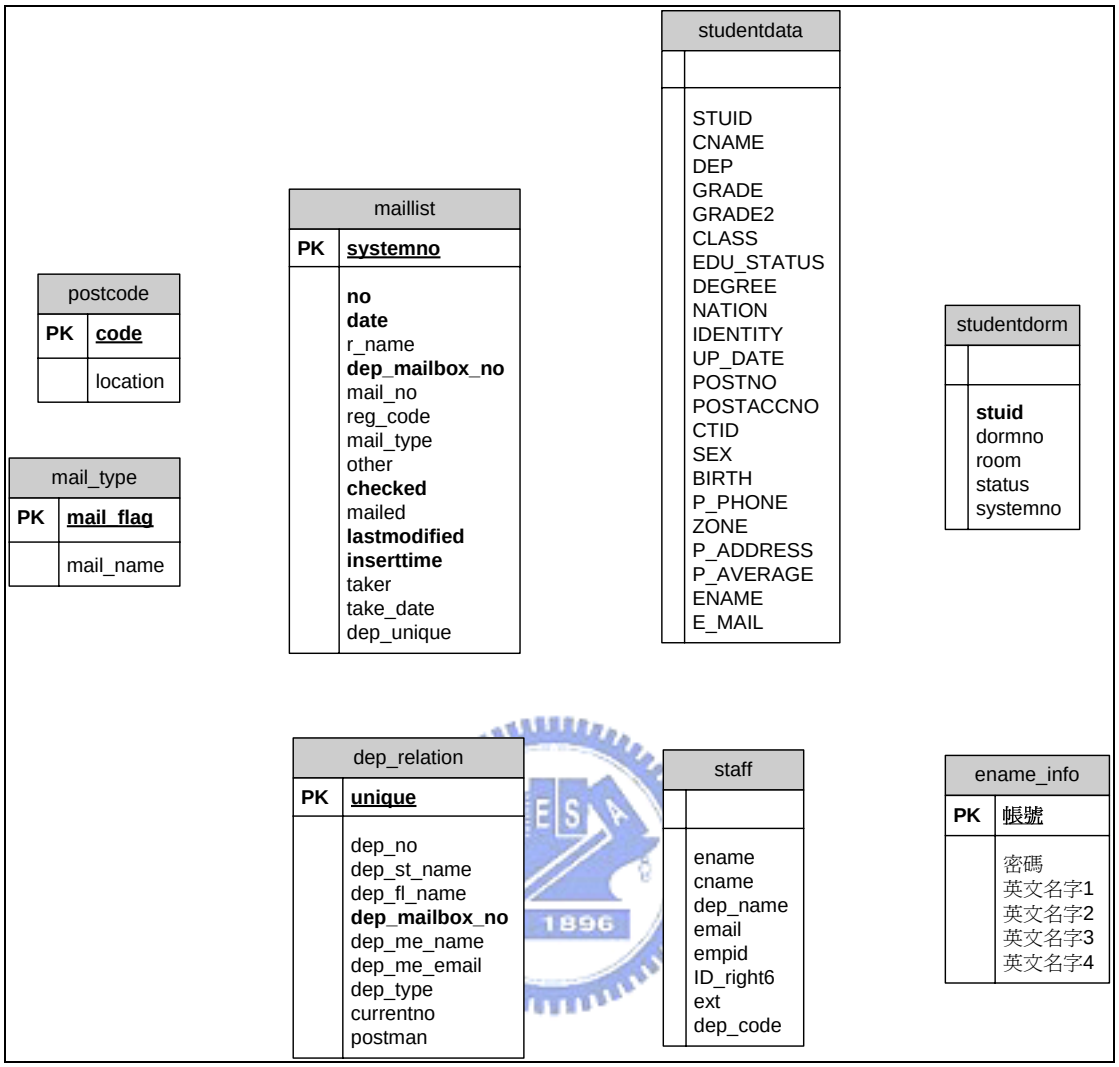


表 1：郵件管理系統資料表

郵件管理系統，共有六個主要的資料表：

1) maillist – 紀錄郵件資訊
2) staff – 教職員資料
3) studentdata – 學生資料
4) studentdorm – 學生住宿資料
5) dep_relation – 各部門對應關係
6) ename_info – 英文姓名資料

以下詳細介紹各資料表內容與其對應關係：

郵件的資料表—maillist

maillist	
PK	**systemno**
	no **date** r_name **dep_mailbox_no** mail_no reg_code mail_type other **checked** mailed **lastmodified** **inserttime** taker take_date dep_unique

部門相關的資訊都會記錄在 dep_relation 資料表中。

unique 欄位爲 primary key。(PK)

unique、dep_mailbox_no 欄位，不可以爲 NULL。(粗體字)
郵件相關的資訊都會記錄在 maillist 資

- systemno：系統編號，唯一且自動產生。爲此資料表的 primary key。
- no：編號，管理人員爲了方便區別信件所使用的編號。
- date：郵件到達日期
- r_name：收件者姓名
- dep_mailbox_no：所有的單位、系所、宿舍都會有一個相對應的信箱編號，管理人員爲了方便區別單位所使用的編號。
- mail_no：郵件編號，郵局對掛號郵件所產生的編號，可由刷條碼方式輸入。
- reg_code：郵遞區號，郵局將地方各區域編號，可由刷條碼方式輸入。

postcode 資料表有郵遞區號與區域名稱的對應資料，其架構如下。

postcode	
PK	**code**
	location

例如：”臺北市中正” (location) 的郵遞區號爲 “100” (code)

- mail_type：郵件類型，郵局將郵件的類型編號，可由刷條碼方式輸入。

Mail_type 資料表有編號與郵件類型的對應資料，其架構如下。

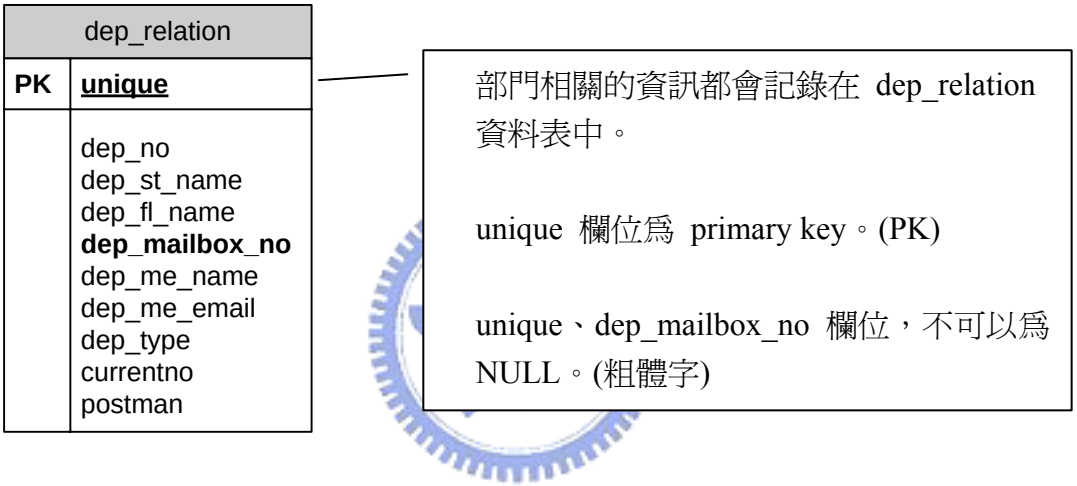
mail_type	
PK	**mail_flag**
	mail_name

例如：”限掛” (mail_name) 的編號爲 “2” (mail_flag)

- other：註解，可在紀錄郵件的同時，對此郵件加上註解。
- checked：記錄此信件是否已被領取。
- mailed：記錄此信件是否有寄信通知收信人。
- lastmodified：最後修改此信件資料的時間。
- inserttime：信件新增的時間。
- taker：領取人的資料。若被領取，此欄位必須有領件人資訊才行。
- take_date：信件被領取的時間。
- dep_unique：信件會對應至收件者的單位、系所或宿舍。此值爲其部門 ID。

  例如：文書組的 ID 爲 “{A8A26627-F489-4AC5-8EB7-86804D391BF3}”

部門的資料表—dep_relation

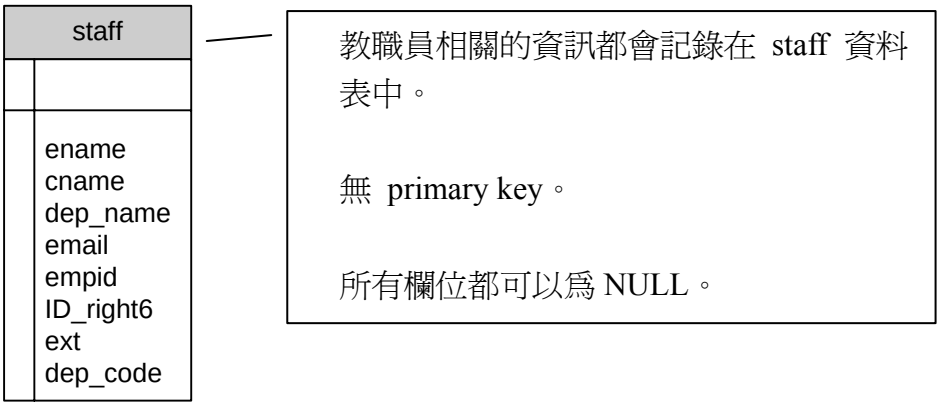


- unique：部門 ID，唯一且自動產生。爲此資料表的 primary key。
- dep_no：部門代碼。

  例如：工學院的部門代碼爲 “COE”
- dep_st_name：部門名稱-簡稱。
- dep_fl_name：部門名稱-全名。
- dep_mailbox_no：所有的單位、系所、宿舍都會有一個相對應的信箱編號，管理人員爲了方便區別單位所使用的編號。
- dep_me_name：單位聯絡人。
- dep_me_email：單位的電子郵件信箱。
- dep_type：”行政單位” 或者 “宿舍”。
- postman：領件人代號。

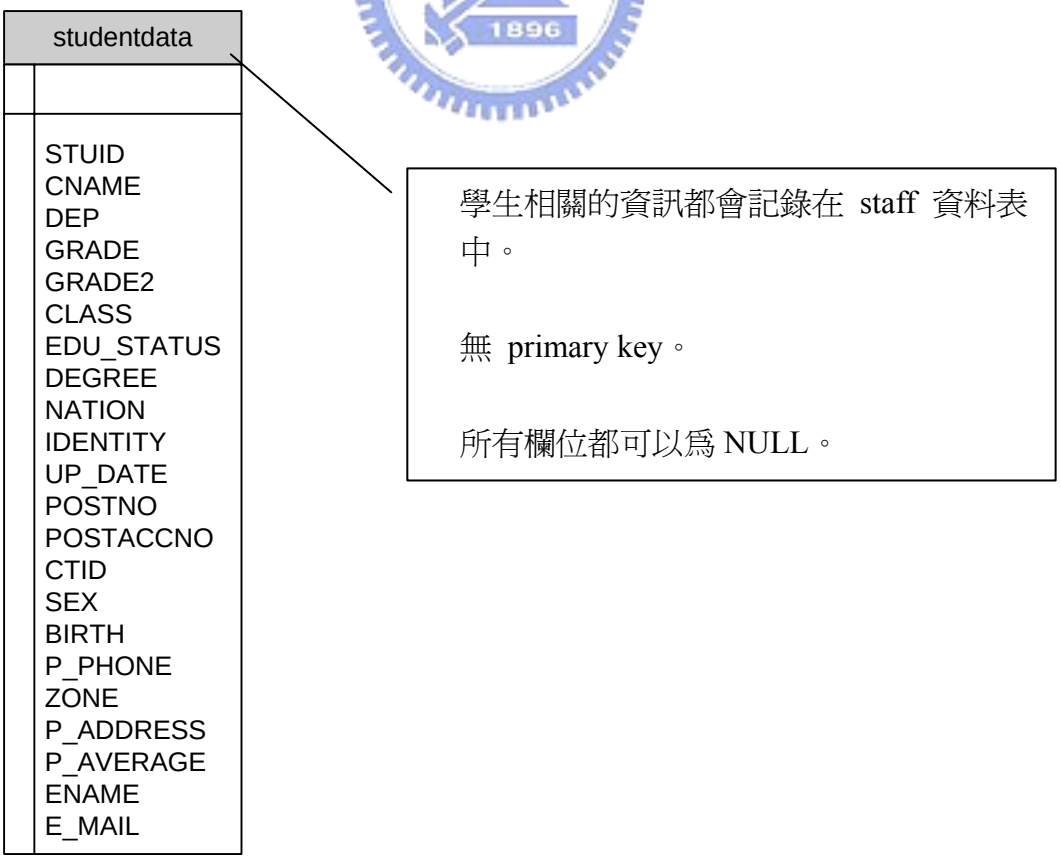
  例如：”工學院” 的領件人代號爲 ”G”

教職員的資料表—staff



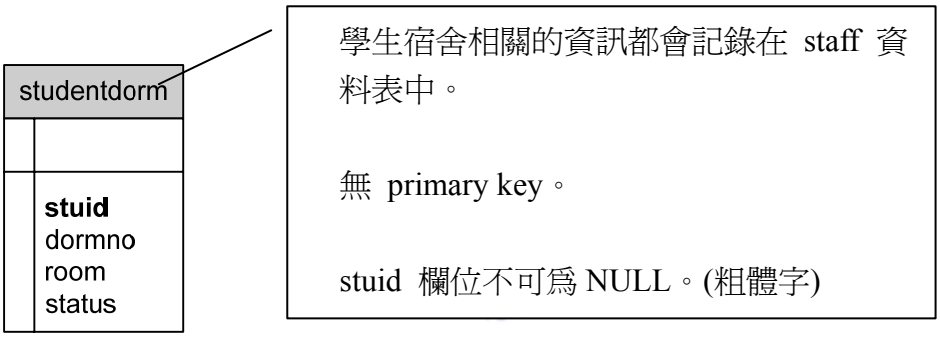
- ename：英文姓名
- cname：中文姓名
- dep_name：所屬部門名稱
- email：電子郵件
- empid：人事代號
- ID_right6：身份字號 (後六碼)
- ext：分機號碼
- dep_code：所屬部門代碼

學生的資料表—studentdata



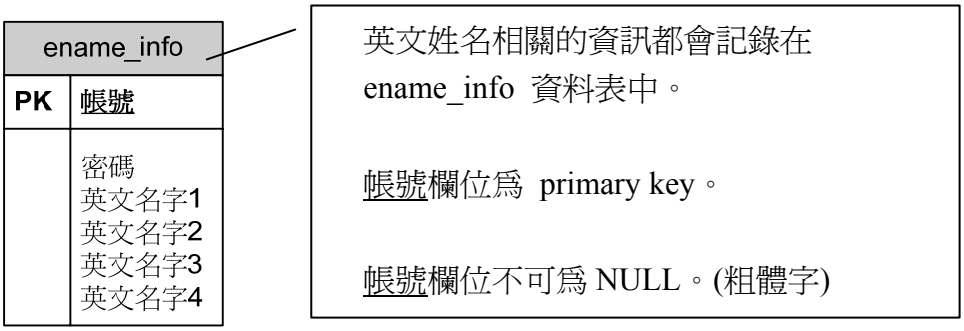
- STUID：學號
- CNAME：中文姓名
- DEP：所屬系所代碼
- CTID：身份字號
- SEX：性別，”M” 或者 “F”。
- BIRTH：生日
- P_PHONE：電話
- ZONE：郵遞區號
- P_ADDRESS：住址
- P_AVERAGE：畢業學校
- ENAME：英文姓名
- EMAIL：電子郵件

學生宿舍的資料表—studentdorm



- stuid：學號
- dormno：所住宿舍名稱
- room：房間號碼
- status：”未借用” 或 “借用中”

英文姓名的資料表—ename_info



- 帳號：登入 ”英文姓名建檔系統” 的帳號。
  - ◆ 學生的帳號—學號。
  - ◆ 教職員的帳號—人事代號。
- 密碼：登入 ”英文姓名建檔系統” 的密碼。
  - ◆ 第一次登入的密碼，預設爲身分證字號後六碼。
- 英文名字 1：主要的英文名字。
- 英文名字 2：其他的英文名字。
- 英文名字 3：其他的英文名字。
- 英文名字 4：其他的英文名字。

各資料表欄位關係

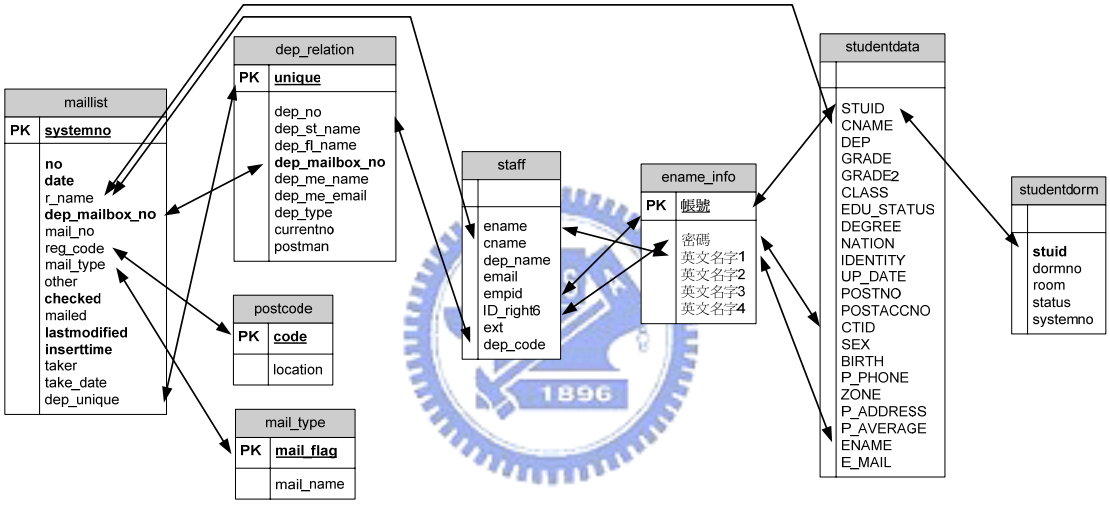


圖 5：郵件管理系統資料表欄位對應關係圖

2.1.2.3.3. 資料庫更新方式

教職員資料更新，使用 SQL Server Enterprise Manager 之管理功能，每天定時查詢共同資料庫之教職員資料，更新本地端的教職員資料；學生資料更新，於每學期初，以 Excel 匯入學生資料。

使用 SQL server 中的管理功能，設定自動更新的排程，以下爲郵件管理系統所設定的自動更新排程：

- studentdorm update

將學生宿舍管理系統的住宿生資料表更新至本地端的 studentdorm 資料表。於每天凌晨 0 時更新

- vi_DocumentMailsys update

將人事共同資料庫所提供的 view 資料表，更新至本地端的 vi_DocumentMailsys 資料表。於每天凌晨 1 時更新。

- staff update

將本地端已經更新的 vi_DocumentMailsys 資料表，整合更新至 staff 資料表。於每天凌晨 2 時更新。

- 資料庫維護計畫 (資料備份)：

於每天下午五時，定期備份 maillist 資料表。所存資料保存一個月，超過一個月的資料表自動刪除。

2.2. 浩然圖書館

2.2.1. 服務項目

(1) 圖書借閱服務[12]
(2) 新書展示
(3) 浩然展示廳
(4) 期刊館藏服務
(5) 視聽資料服務
(6) 讀者諮詢服務
(7) 國內外館際合作服務
(8) 複印服務
(9) 教師指定參考書服務
(10) 圖書館利用指導
(11) 書刊推薦服務
(12) 協尋下落不明書刊服務
(13) 浩然過季會議廳場地出界服務研究小間服務
(14) 報紙閱覽服務
(15) 楊英風研究中心
(16) 交大發展館
(17) 數位圖書館
(18) 藝文空間
(19) 科幻研究中心
(20) 陳慧坤藝術研究中心
(21) 漫畫研究中心
(22) 浩然藝文原稿特藏室暨浩然藝文數位博物館

2.2.2. 櫃檯流通系統

圖書館主要功能就是提供使用者借閱圖書，浩然圖書館有超過一百七十萬冊的書冊，因此需要完善的圖書管理系統才能處理如此眾多且繁雜的業務。

圖書管理系統，就是因應這樣的需求而委託校外廠商所開發的一套圖書管理系統，系統由許多模組組成，例如：流通、編目、館藏…等。在『國家圖書館』的網頁中有『各層級圖書館資訊系統軟硬體規範書』，裡面有詳細介紹圖書館資訊系統的規格，而其中所使用模組其之間的關係如下圖所示：

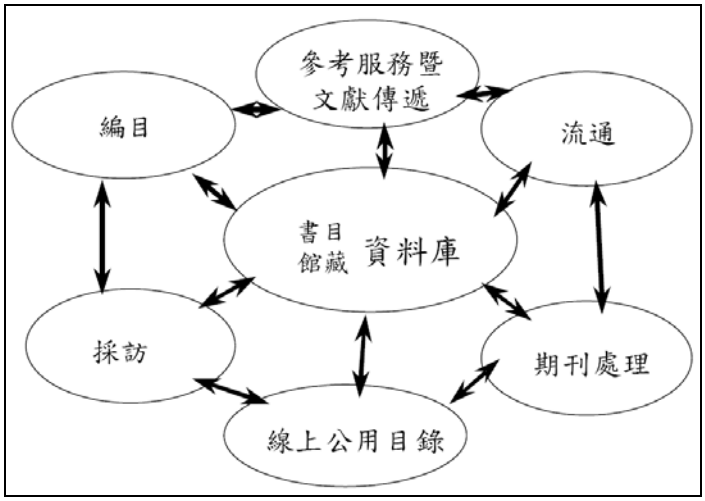


圖 6：圖書管理系統模組示意圖

其中的流通模組爲使用者最常使用到的系統，在櫃檯借閱圖書的系統即爲此模組所提供的功能。本研究中所要探討是以櫃檯流通系統[13]爲主。

系統功能八大功能如下：

(1) 借書
(2) 還書/還書箱還書
(3) 讀者
(4) 館藏
(5) 排序預約
(6) 時段預約
(7) 罰款
(8) 報失/聲明歸還

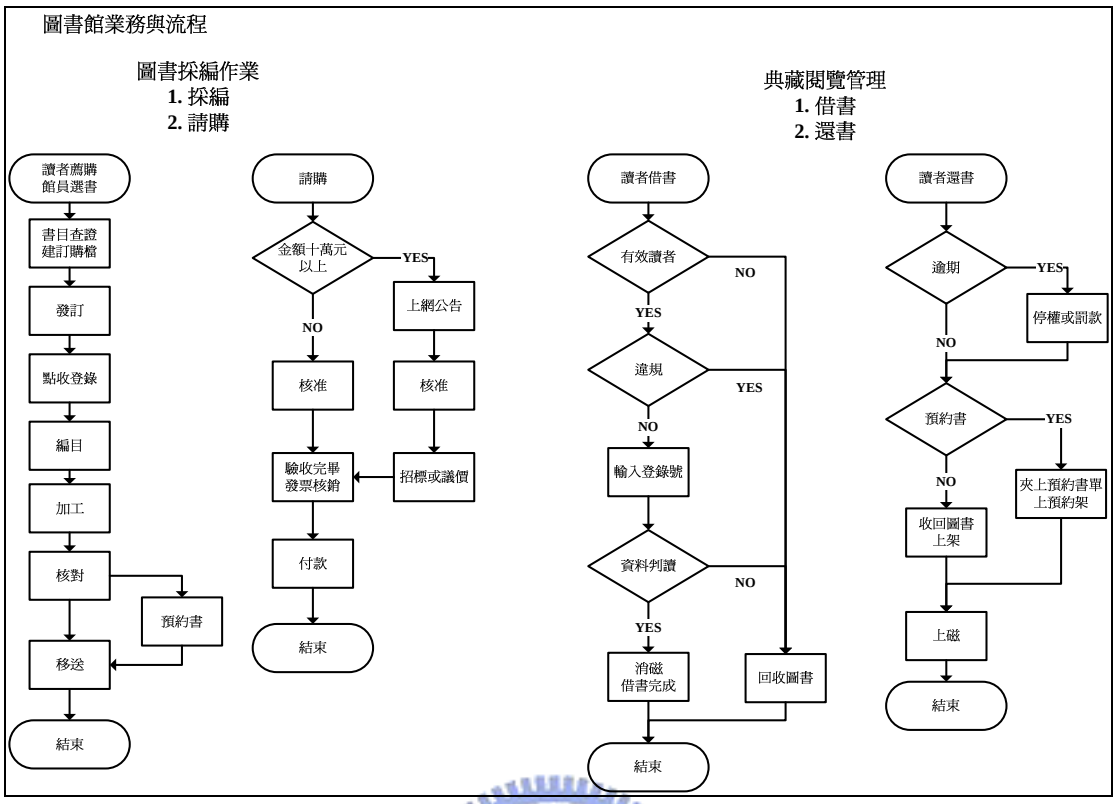


圖 7：圖書館業務流程圖

2.2.2.1. 系統環境與架構

- DB Server
  作業系統：SunOS　5.8 Generic_108528-27 sun4u sparc SUNW,Sun-Fire-880
  資料庫系統：Sybase Adaptive Server Enterprise/12.5.0.2/EBF 10746 ESD#1

- Web Server
  Windows 2003 Server (service pack1)

2.2.2.2. 系統功能

2.2.2.2.1. 借書管理流程

在借書選項中，可處理 1) 借書 2) 變更應還日期 3) 續借 4) 增/修註記 5) 收據列印 6) 報失 7) 聲明歸還，七個業務。以下爲借書流程圖：

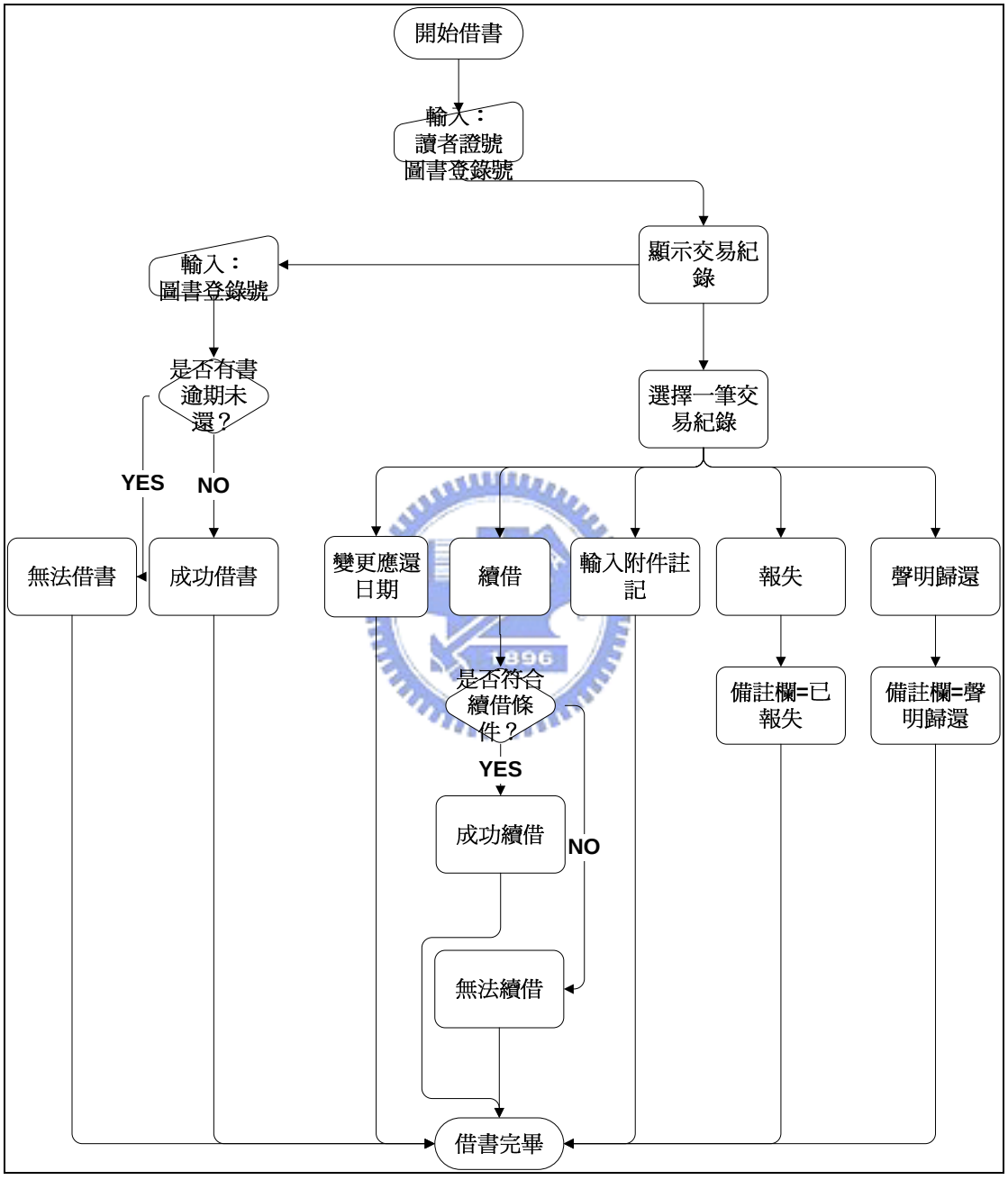


圖 8：借書功能流程圖

以下將詳述借書選項中各項功能：1) 借書 2) 變更應還日期 3) 續借 4) 增/修註記 5) 收據列印 6) 報失 7) 聲明歸還。

借書

要執行借書的功能時，先由下拉式選單方式選擇合作館，輸入讀者證號後按”Enter”鍵，再輸入圖書登錄號，之後再按”Enter”鍵即會出現一筆借書新記錄；無借閱權限或非本館藏書或讀者有書逾期未還等狀況，則出現無法借書的警告視窗。以下圖示爲借書功能畫面：

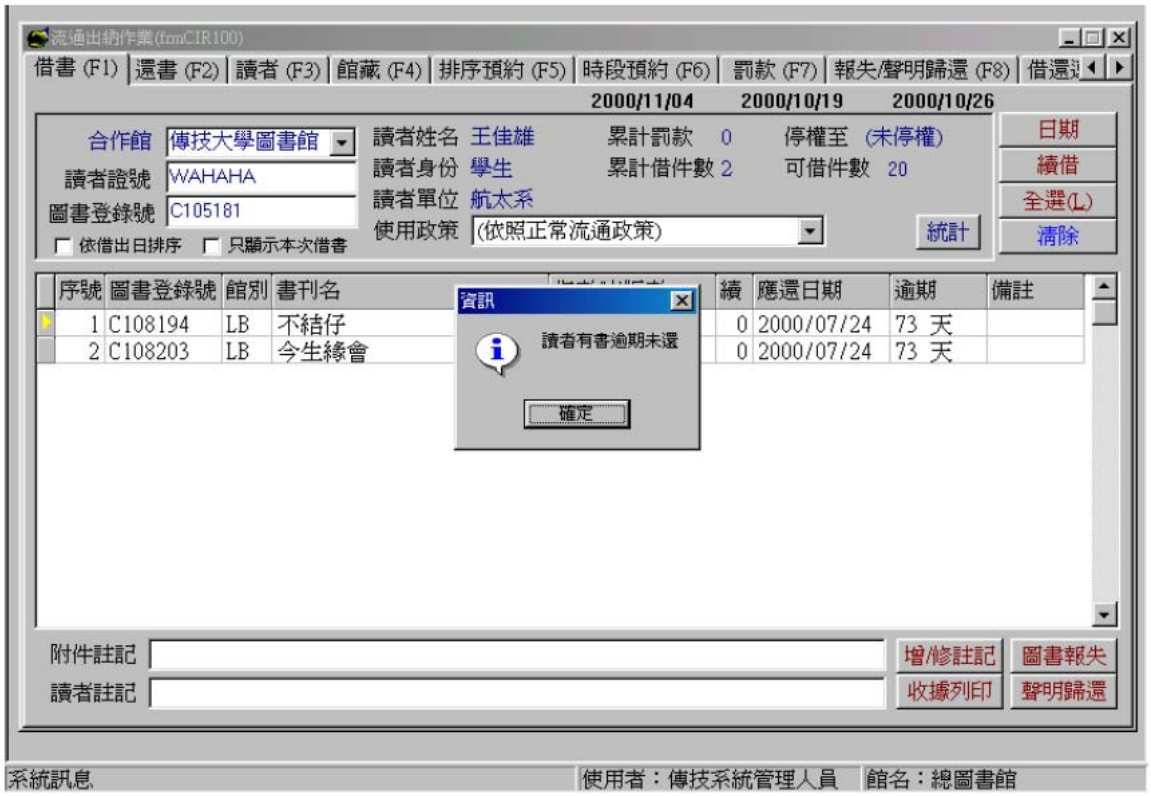


圖 9：無法借書警告視窗畫面

變更應還日期

若遇特殊狀況需更改應還書日期時可在此處更改。選定一筆交易記錄，執行日期功能出現更改借期視窗，顯示圖書登錄號及原應還日期，並有新應還日期欄，輸入後按下確定鍵則更改成功，如下圖所示。

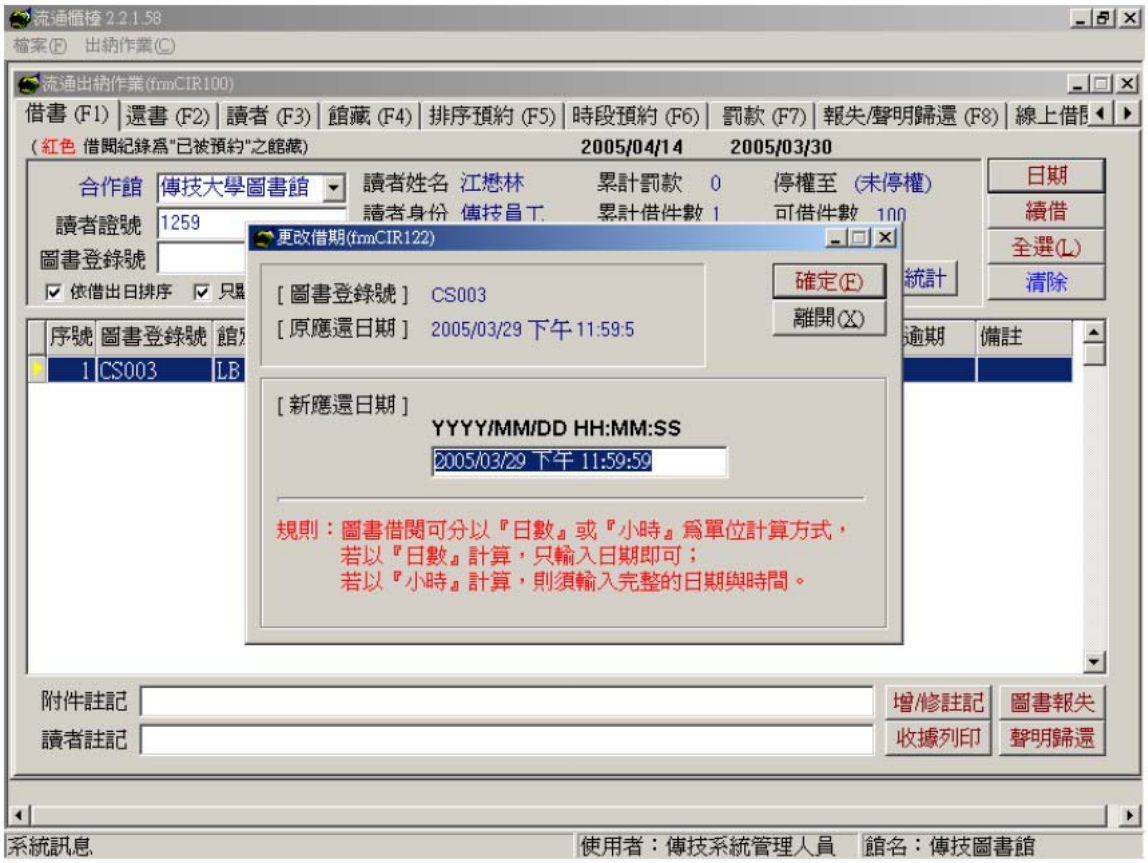


圖 10：更改借期視窗

續借

選定一筆交易記錄，按下【續借】的功能鍵，在交易記錄上即會出現續借的記錄。續借功能應注意是否符合在【流通管理模組】→【流通政策】→【流通借閱/預約政策】→【天數借閱規則】所設定之在應還日期前幾天可續借，否則出現無法續借警告視窗。

增/修註記

當讀者借的圖書有附件時，可在附件註記欄中輸入附光碟或錄音帶等註記，再按【增/修註記】，就會出現告知修改完畢的視窗，確認後即可。

收據列印

按下收據列印鈕，可列印讀者借書收據。

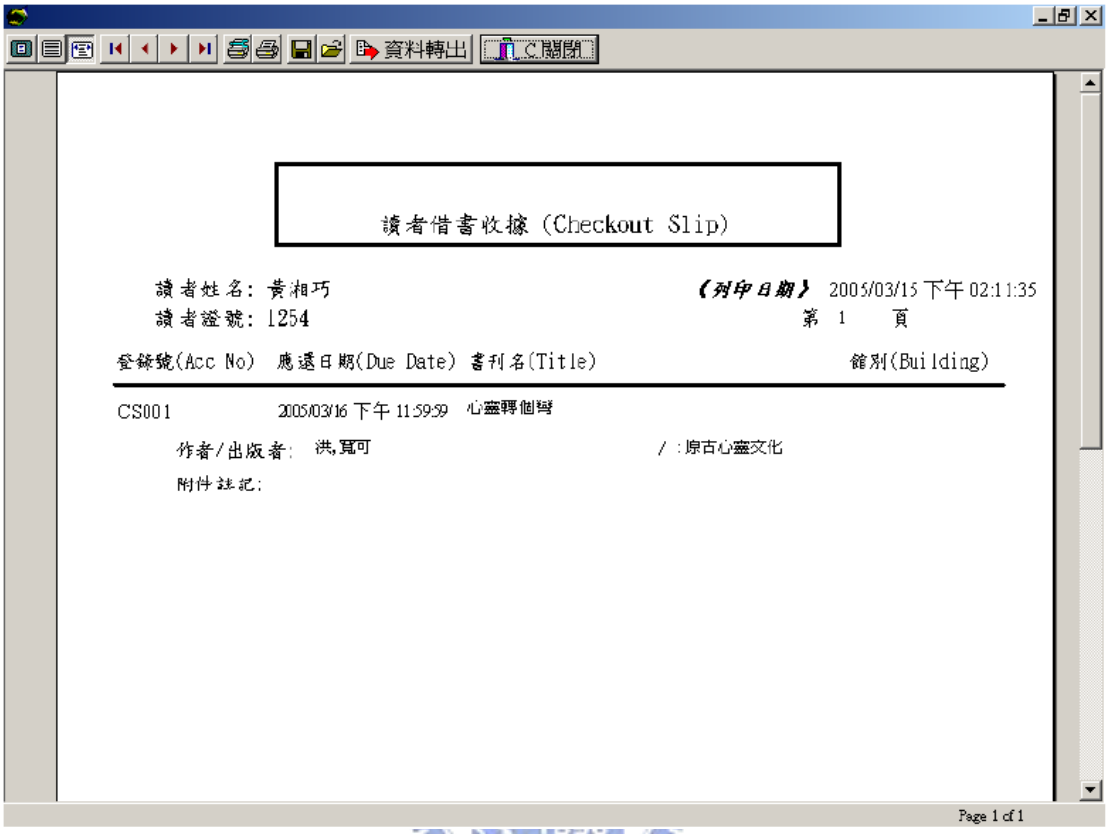


圖 11：讀者借書收據圖

圖書報失

當讀者將圖書遺失時，要有圖書報失的手續，此處可將讀者借書記錄作報失手續。選定一筆欲報失的交易記錄，執行圖書報失功能鈕，則該借書圖書記錄備註欄會註記”已報失”，如下圖所示。

聲明歸還

當讀者已還的圖書借書記錄中仍有此筆圖書記錄，若需聲明歸還可在此處作聲明歸還的手續。選定一筆欲聲明歸還的交易記錄，執行聲明歸還的功能，則在所借圖書記錄備註欄出現”聲明歸還”的註記，如下圖所示。

圖 12：備註欄中圖書報失、聲明歸還的記錄圖

2.2.2.2.2. 還書管理流程

在還書選項中，可處理 1) 還書 2) 還書箱還書 3) 查詢 4) 列印個人今日還書收據，四個業務。以下爲還書流程圖：

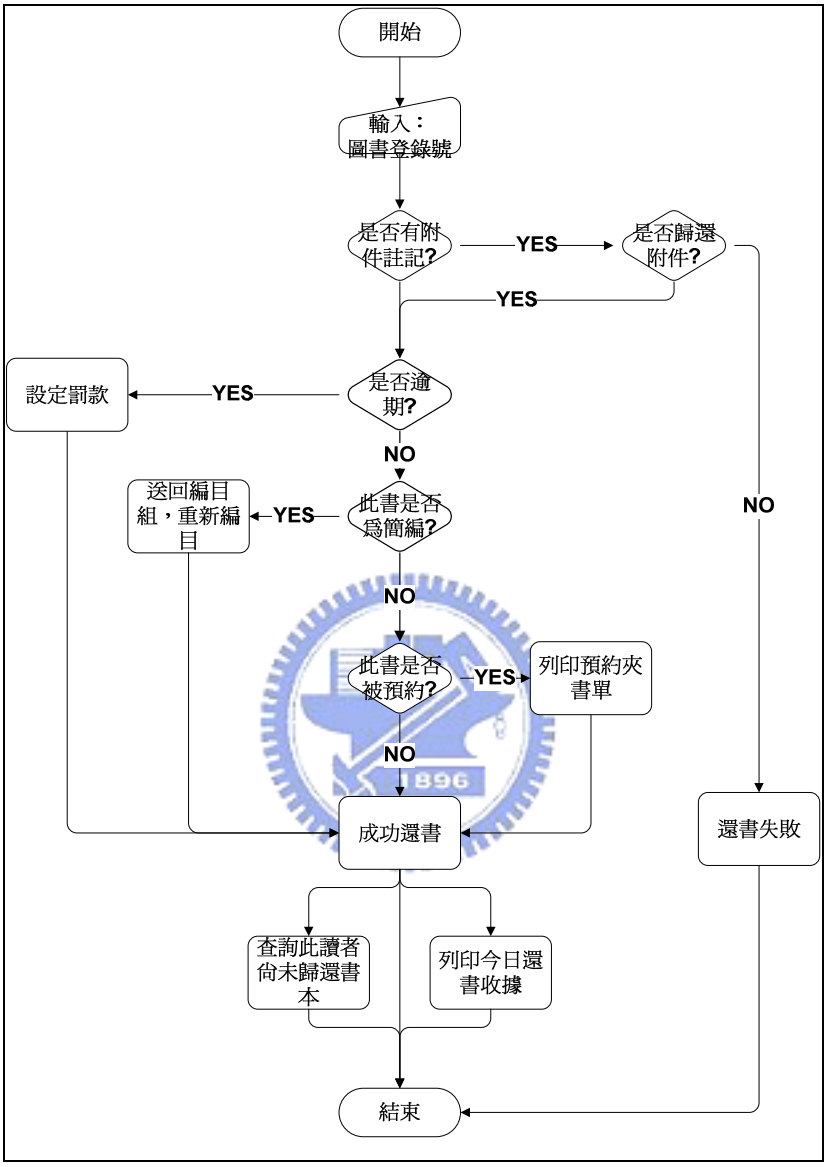


圖 13：還書功能流程圖

以下將詳述還書選項中各項功能：1) 還書 2) 還書箱還書 3) 查詢 4) 列印個人今日還書收據。

還書

選擇還書的作業方式，輸入圖書登錄號即可還書。若圖書有附件，會出現提示附件是否歸還的視窗，下圖所示。

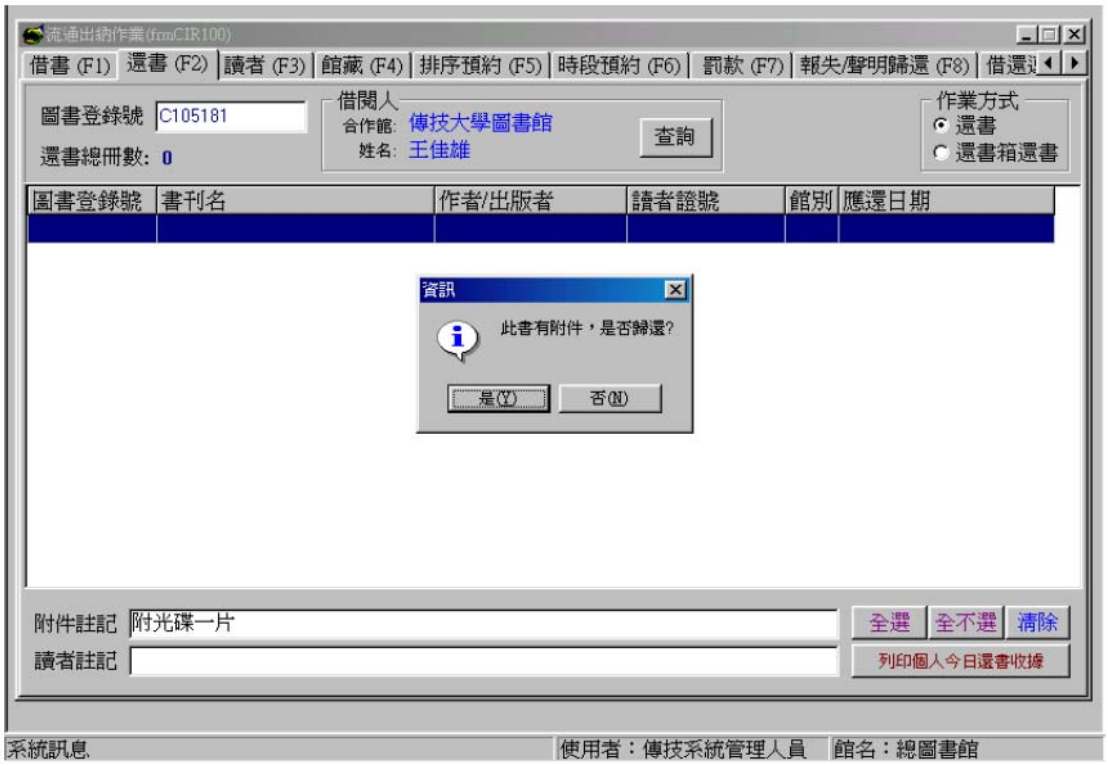


圖 14：還書提示附件畫面

若讀者逾期還書，則在還書時會出現罰款提示的資訊視窗，下圖所示。接下來可至罰款(F7)摺頁，執行後續的罰款作業。

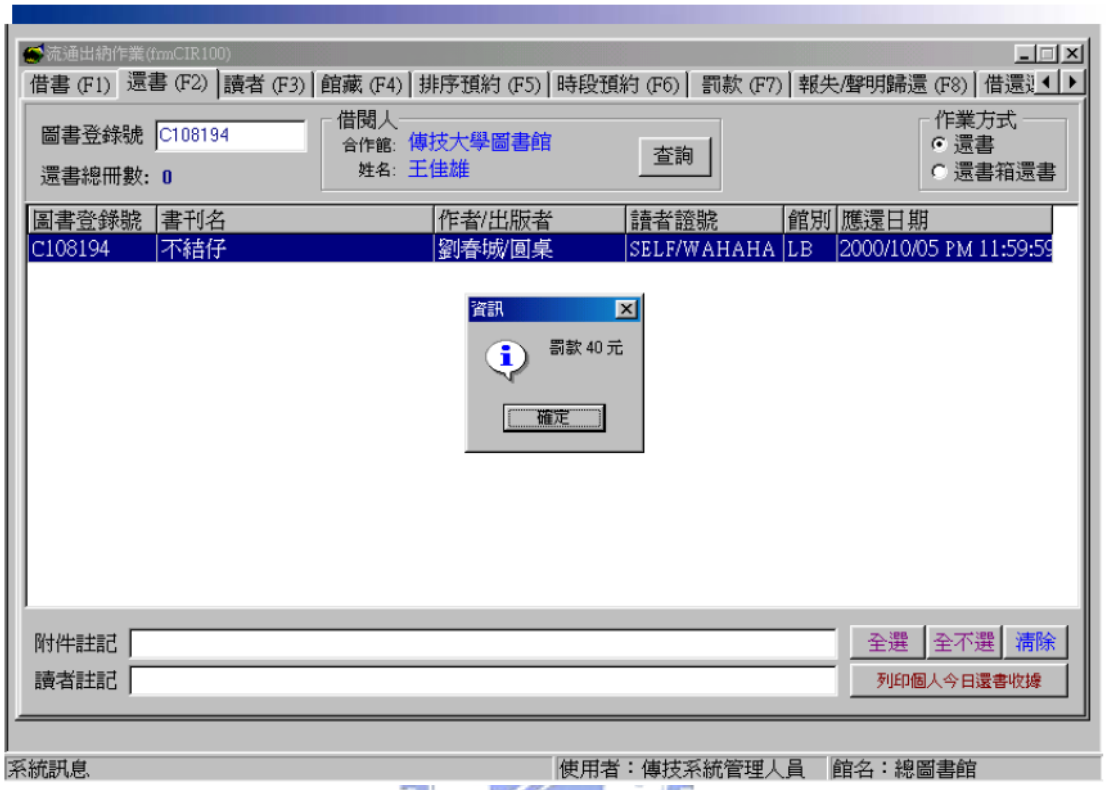


圖 15：逾期罰款提示視窗

若讀者所還的書已有他人預約，系統會自動在還書時出現預約夾書單的預覽畫面，方便館員辨識此書爲已被預約的書，如需列印此夾書單，按下左上角的印表機圖示即可印出此預約夾書單，列印完畢後按下關閉鈕，即可關閉此視窗，如下圖所示。

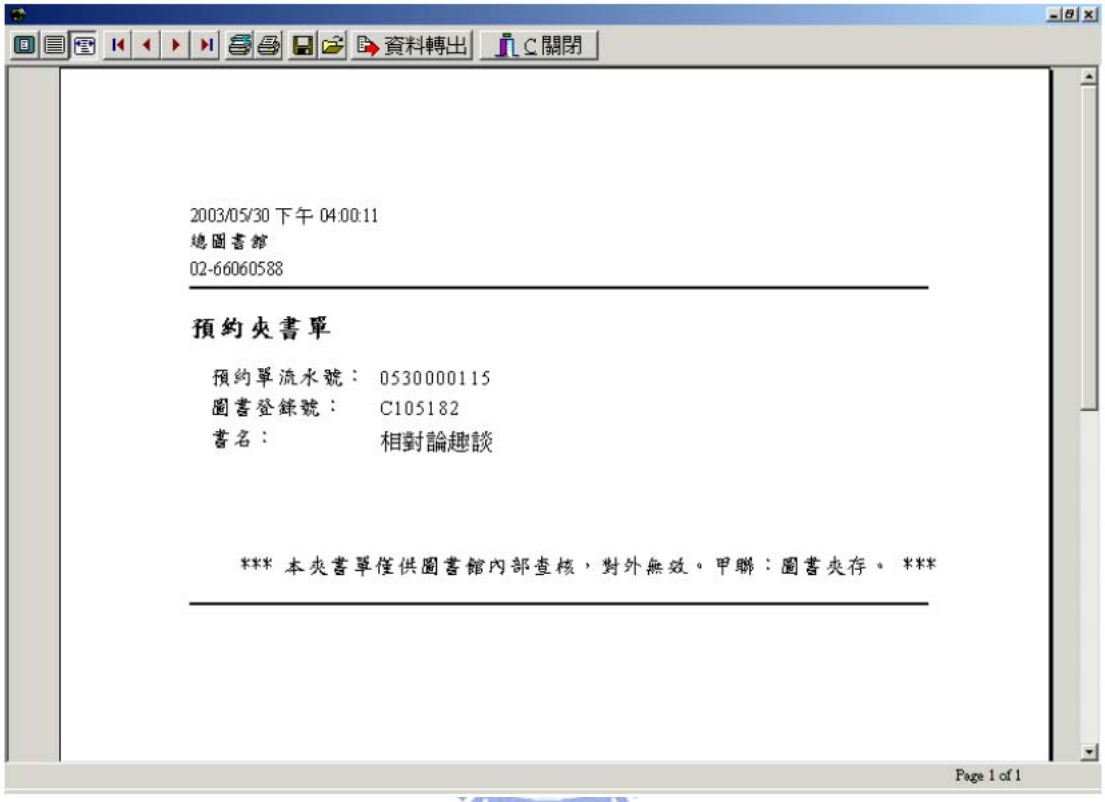


圖 16：預約夾書單

因爲預約的狀況隨時會變動，讀者可由 Webpac 自行取消預約，故預約夾書單上不會出現預約的讀者資料。

還書箱還書

系統依【流通管理模組】→【流通政策】→【流通借閱/預約政策】的天數借閱規則中所定之還書箱寬限期，判斷讀者是否逾期還書，若讀者逾期還書，還書時系統會出現逾期的相關訊息提示。

查詢

還書時，直接刷圖書登錄號，即可完成還書動作，若讀者需要查詢還有幾本書未還，可利用查詢鈕，系統會將該讀者證號帶回 F1 的借書畫面中，即可看到讀者尚未歸還的資料。

列印個人今日還書收據

可利用列印個人今日還書收據鈕，列印出該讀者今日的還書收據，如下圖所示。

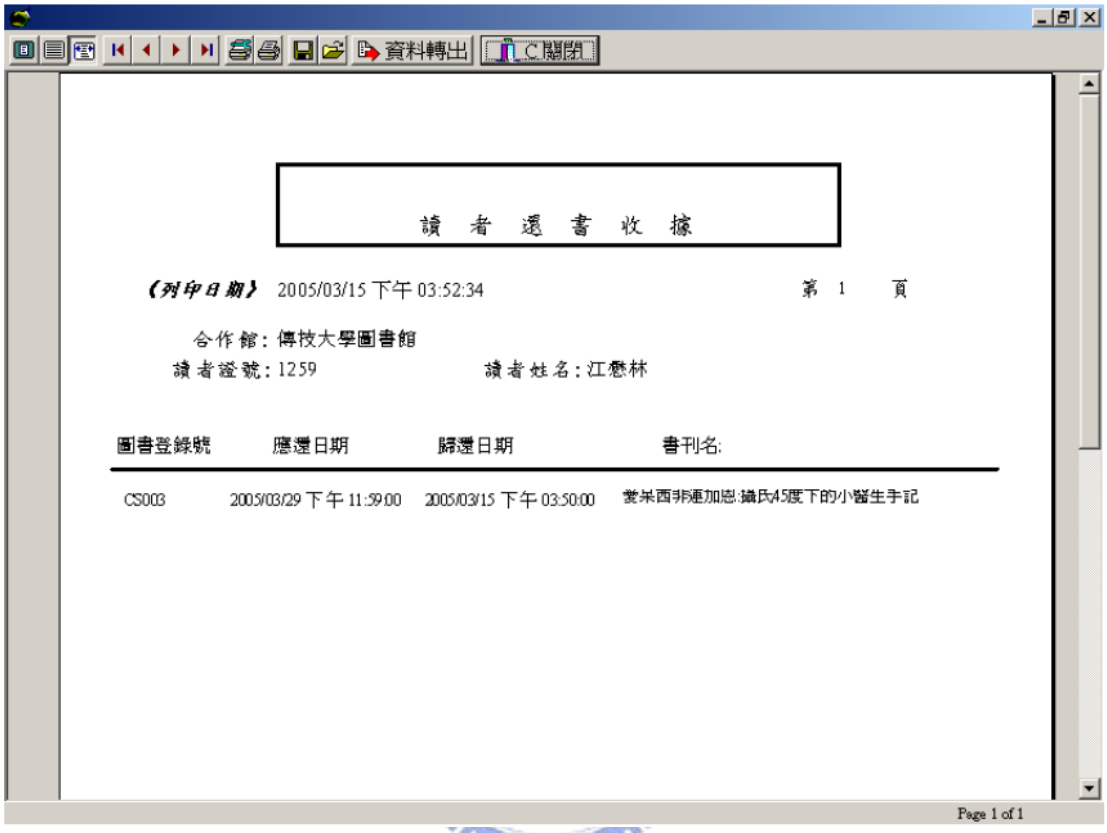


圖 17：還書收據圖

2.2.2.3. 系統資料表

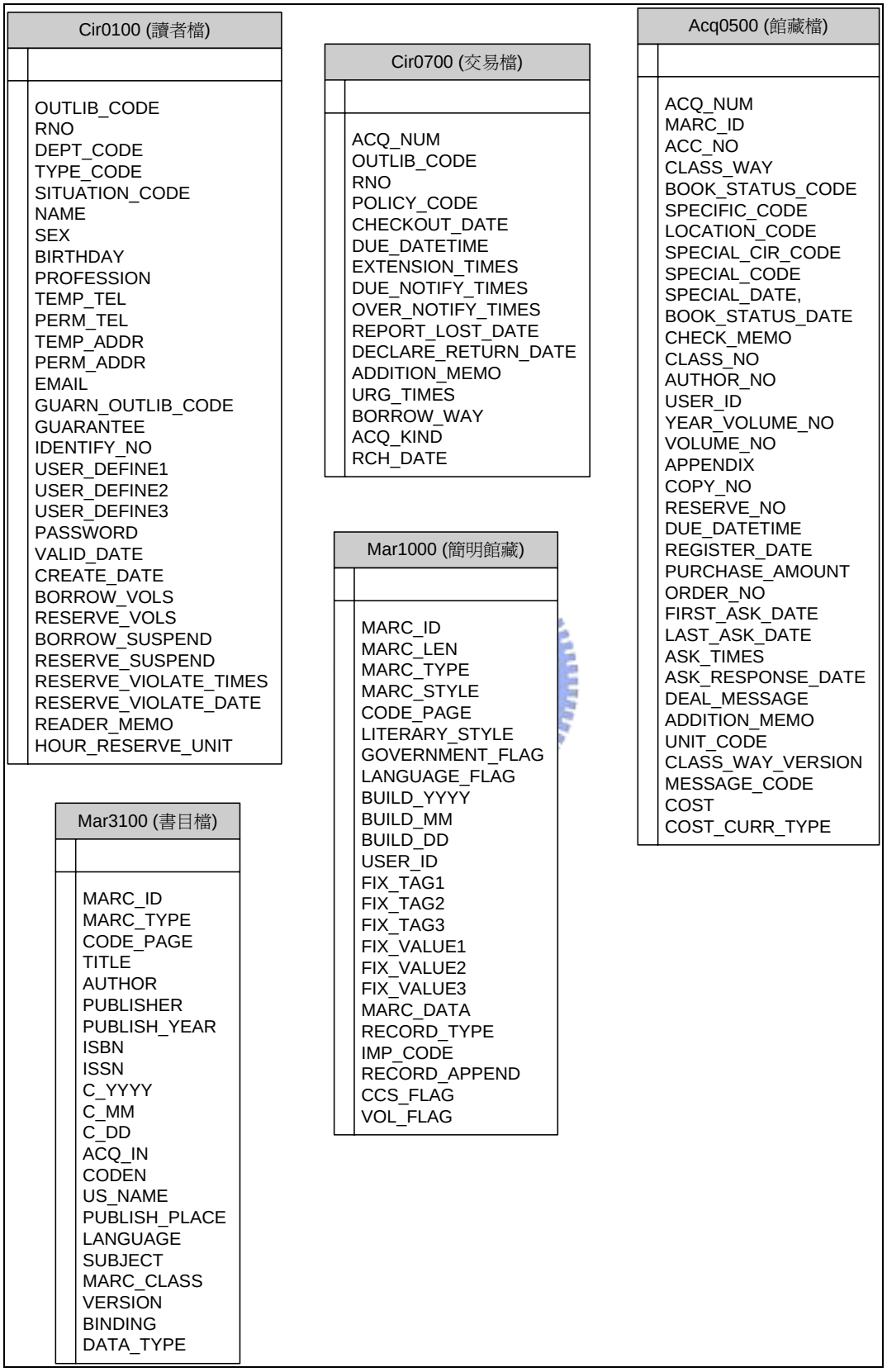


表 2：櫃檯流通系統之資料表

2.3. 課務組

課務組網頁中有提到，首要目標爲『全方位的課務革新，建構高品質的修課環境』，並且四大方向爲：提昇教學品質、提高排課品質、教學反應意見信箱與問卷調查、擴大暑期開課彈性。

2.3.1. 負責業務

- 課程編排[14]
- 選課教學反應問卷
- 大學部傑出教學獎
- 教師授課鐘點核計
- 暑期班
- 教室管理及借用
- 支援其他業務事項

2.3.2. 電子化系統

現有系統[15]

- 網路選課系統
- Windows_AP 之選課系統
- Windows_AP 之排課系統
- Windows_AP 之教學反應系統
- 教室借用系統
- 暑修網路選課
- 教師鐘點合計系統

系統環境

- 作業系統：Windows Server 2003
- 資料庫系統：Microsoft SQL Server 2000

教職員資料表結構 (欄位)

教職員資料
員工編號
姓名
單位
職稱
URL
密碼
分證號期
郵局帳號
郵局局號
Email
tmp 權限
權限
教學候選人
地址
離職
退休
死亡
職級

表 3：課務組人事資料表

2.3.3. 人事資料來源與更新

- 教職員資料

教職員的異動爲不定時且少量的，所以目前的處理方式爲由人事室以紙本告知課務組有異動的教職員資料，再由課務組以手動方法新增/修改此異動的教職員資料。

- 學生資料

於每學年都有大量的新進學生，由註冊組提供 dbf 型態的資料庫檔案給課務組，課務組系統維護人員再以手動的方式整批匯入系統中。其餘零星的學生異動，則個別以手動方式新增學生資料。

2.4. 駐警隊

2.4.1. 負責業務

1. 負責學校門禁管制勤務[16]
2. 校內各大門警衛勤務之排定及調配
3. 夜間各館舍外圍定時巡邏
4. 學校防護團訓練及消防教育訓練
5. 協助學校交通安全管理委員會執行校內交通相關管理工作
6. 全校汽、機車進出之管制及檢查
7. 日間校內交通秩序維護、取締違規停放之汽車及機車拖吊
8. 每年教職員工及廠商汽、機車識別證換發。

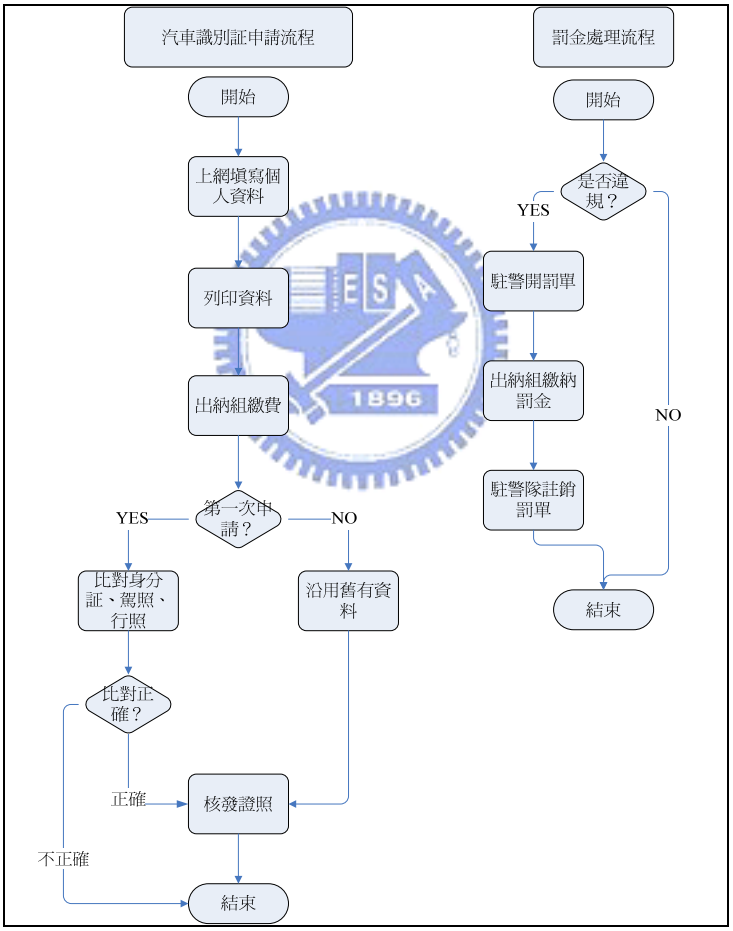


圖 18：汽車識別證系統業務流程圖[17]

系統環境

- 後端管理系統開發語言：VB 6.0
- 前端查詢系統開發語言：ASP
- 系統資料庫：SQL Server 2000

2.5. 共同資料庫

因爲校內部門有許多業務的處理上必須仰賴其他部門的資料，所以計算機中心系統開發人員，因此提出了共同資料庫的架構，將各部門所需的資料整合在共同資料庫中，以方便其他部門使用。

2.5.1. 資料庫概況

共同資料庫[18]建立的緣由是爲了將校內各部門資料庫資料做整合。透過這樣的一個集中式的架構，將資料彙集於一地，如此一來可以降低部門之間交互運作的複雜度，同時也可以透過計算機中心開發人員來集中管理。共同資料庫構想如下圖所示：

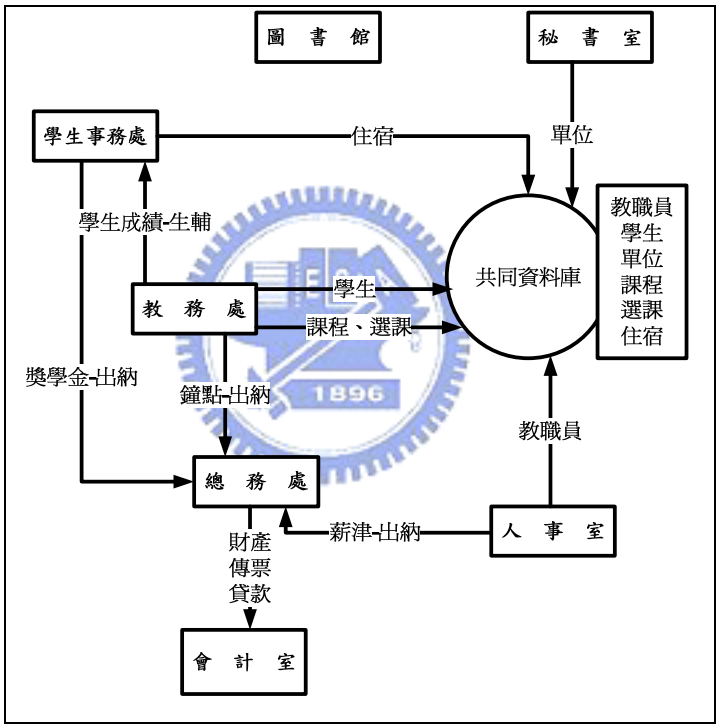


圖 19：共同資料庫架構圖

2.5.2. 資料庫建置

- **共同資料庫資料情形**

校內各部門將彼此之間會使用到的資料集中到共同資料庫，共同資料庫預計維護的資料有 1) 教職員 2) 學生 3) 單位 4) 課程 5) 選課 6) 住宿。

除此之外在上圖中亦可呈現部門之間彼此運作的情形，並非所有資料都匯集至共同資料庫，因爲這些資料只有特定少數部門會使用。不同部門之間的訊息交換是相當頻繁的，若無建立良好的溝通處理機制，在處理業務的作業上，會導致

校務效率上較差，且行政人員的負擔也相對沉重。

以學生事務處的學生獎學金業務處理爲例，業務處理過程中必須有教務處課務組所提供的學生成績，統計出獎學金結果之後，將獎學金名單傳送給出納組，由出納組負責發送獎學金。一個業務的流程須由多個部門共同合作才有辦法完成，若有一個環節延遲或發生錯誤，則會容易形成業務處理上的瓶頸。

- **共同資料庫問卷調查結果**

共同資料庫的核心即爲資料庫層面的整合，首先調查校內各部門之間的資料需求，在決定出共同資料庫所要維護的資料，因爲共同資料庫所提供的資料即爲其他部門常使用的資料。以下爲問卷調查狀況：

1. 需要使用人事共同資料庫單位

   圖書館、綜教組、課務組、出版組、推教中心、網教祖、衛保組、事務組、出納組、保管組、研究發展組、會計室、研發處計畫組

2. 各單位所使用之作業系統平台及資料庫平台

A. 作業平台：

- Windows 各類作業系統 (Win 95、Win98、NT、Win2000)：

  秘書室、綜教組、課務組、出版組、推教中心、衛保組、事務組、保管組、研究發展處、會計室、研發處計畫組
- Unix：

  圖書館
- Linux：

  網教組
- Dos：

  註冊組、生輔組、出納組

B. 資料庫平台：

- Windows 各式資料存放方式 (Excel、Word、Dbase、Foxpro、SQL Server、Access)：

  秘書室、綜教組、課務組、出版組、推教中心、衛保組、事務組、保管組、研究發展處、會計室、研發處計畫組、註冊組 (Dbase)、出納組 (Foxpro)
- Sybase：

  圖書館
- MySQL：

  網教組

➢ **共同資料庫資料來源**

共同資料庫的來源有 1) 人事室 2) 事務組。人事室提供的資料有公務人員資料、約用人員資料以及計畫人員資料。事務組則提供工友資料。以下圖示爲目前校內共同資料庫的架構：

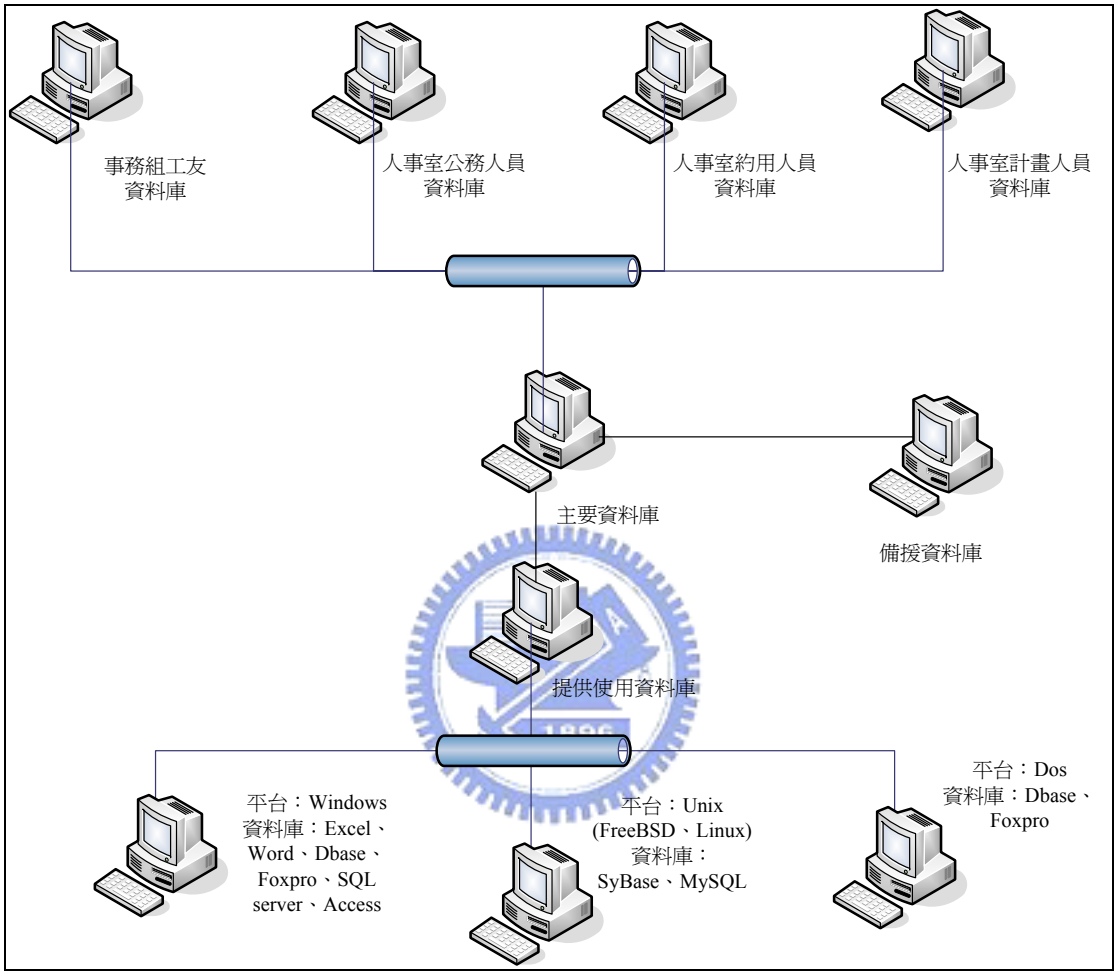


圖 20：共同資料庫架構圖

2.5.3. 資料庫使用方法

欲使用共同資料庫的部門，須先向資料所屬單位提出申請。例如，教職員的資料是隸屬於人事室，因此若使用教職員的資料，須先向人事室提出申請，再由計算機中心開發人員負責資料權限上的設定，之後提書申請的系統方可使用。以下爲人事共同資料庫申請表詳細內容，共四頁：

人事共同資料庫申請表 - Page 1	
申請單位	
申請人	
申請日期	
申請用途	
請在下頁人事共同資料庫結構中，直接選取您所需要的欄位	
請在後頁取用人事共同資料庫方法中，直接選取您所想要的方法	

表 4：人事共同資料庫申請表

人事共同資料庫資料結構　- Page2
(請直接在欄位前勾選您所需要的欄位)

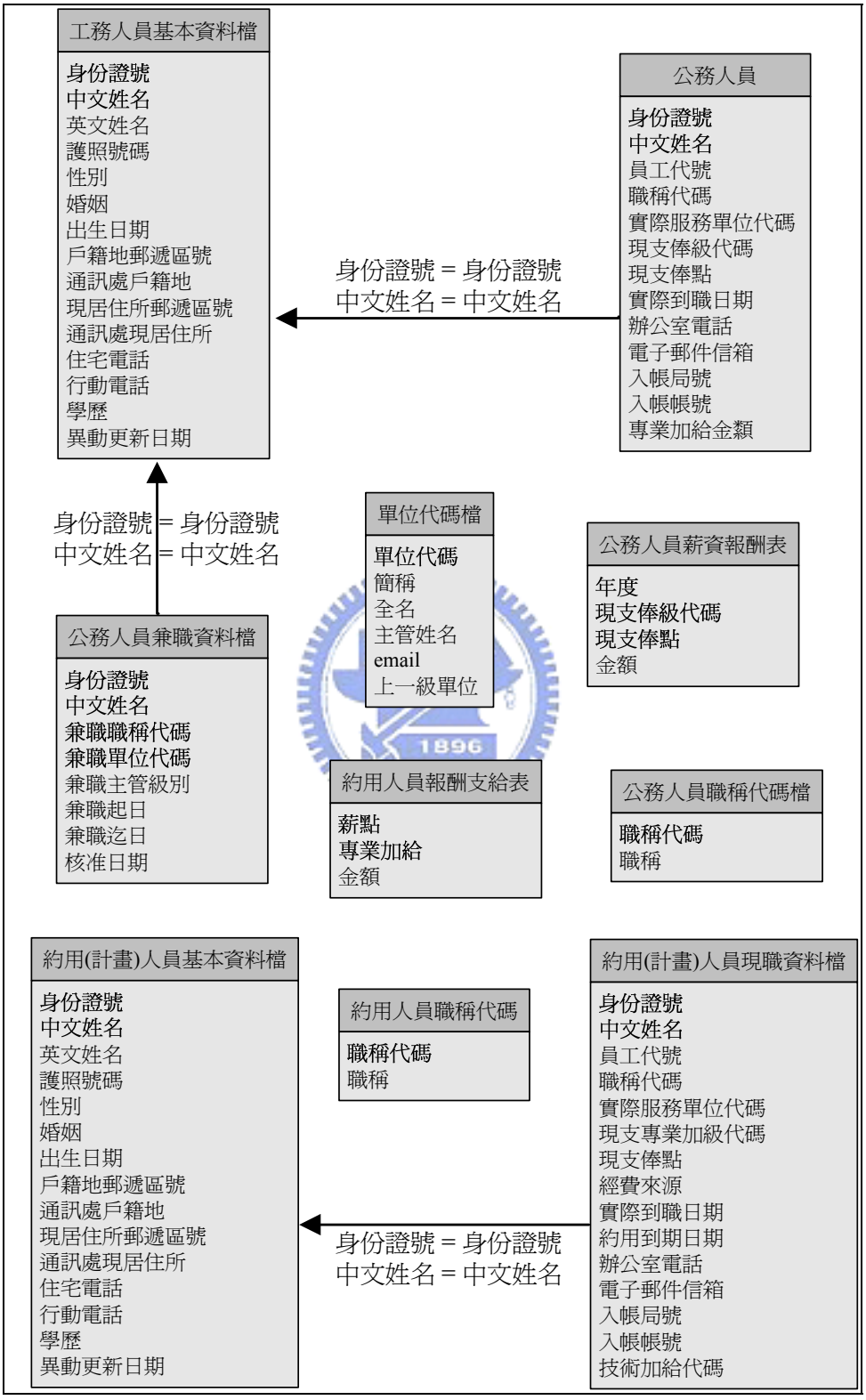


表 5：人事共同資料庫資料結構表

取用人事共同資料庫方法 – Page 3

方法一、由人事共同資料庫管理員，將使用者所需之資料存放在使用者所指定的電腦 (或資料庫) 內

- 說明：由人事共同資料庫管理員，將使用者所需之資料存放在使用者所指定的電腦 (或資料庫) 內
- 步驟：
  3. 請給人事共同資料庫管理員一組進去使用者端該電腦 (或資料庫) 的資料 (由使用者端提供)。
     - ✓ IP 位址
     - ✓ 檔案格式及存放位置
     - ✓ 帳號
     - ✓ 密碼

  4. 人事共同資料庫管理員將定期把資料放到使用者端該電腦 (或資料庫) 中指定位置內。

方法二、由使用者自行進去人事共同資料庫，以取得所需之資料。

A. 方法 A：取用格式爲 SQL Server 資料庫

- 說明：
  1. 人事共同資料庫管理人員將依資料所屬單位 (人事室、事務組) 所開放權限之資料，匯集在一 View 中。
  2. 共同資料庫只開放該 View 中 Select 的權限。
- 步驟：
  1. 申請一組進入人事共同資料庫的資料 (由共同資料庫端提供)。
     - ✓ IP 位址
     - ✓ 檔案格式及存放位置
     - ✓ 帳號
     - ✓ 密碼

取用人事共同資料庫方法 (續) – Page 4

方法 B：取用格式爲 非 SQL Server 資料庫

- 說明：
  3. 人事共同資料庫管理人員將依資料所屬單位 (人事室、事務組) 所開放權限之資料，匯集在一目錄下。
  2. 共同資料庫只開放該目錄讀取的權限。
- 步驟：
  1. 申請一組進入人事共同資料庫的資料 (由共同資料庫端提供)。
     - ✓ IP 位址
     - ✓ 該資料所存放位置
     - ✓ 帳號
     - ✓ 密碼

2.6. 蒐尋結果與問題

在訪問學校各部門相關人員之後，可以發現校內系統在整合過程中所遭遇的困難，最後依部門歸納出這些問題，內容爲部門人員在訪問過程中所提及的問題或是在分析資料之後所發現的問題。以下爲發現問題的內容：

2.6.1. 文書組

1) 郵件開發系統功能無法正常運作，搜尋收件人所屬單位系所功能結果不正確。
2) 需獨自開發英文姓名系統，此功能應仰賴其他部門可提供的英文姓名資料。

2.6.2. 圖書館

1) 平台與資料庫系統較爲特殊。圖書館的平台爲 SunOS, Sybase。
2) 在資料更新方面會有欄位對應的問題。圖書館系統維護人員反應，他們在資料的更新上尚未建立一個良好的管道，雖然已經使用共同資料庫的資料，但是仍有資料庫差異性的問題存在。

2.6.3. 課務組

1) 人事異動時，若人事室未及時通知課務組，會導致薪資計算錯誤問題。課務組表示，他們在處理教師鐘點的業務上，需仰賴人事室提供的資料才有辦法正常運作，但是系統之間溝通的管道尚未建立，所以資料常常發生尚未更新而發生錯誤的情形。
2) 通知方式是以書面通知，仍未電子化。因爲課務組與人事室之間的溝通管道尚未建立，所以彼此之間仍以傳統的紙本方式傳遞訊息，對於課務組與人事室都是相當不人性化的處理方式。
3) 對於共同資料庫的資料正確性持保留態度。若共同資料庫的資料錯誤，則可能造成許多作業上的問題，如此一來處理的事務上會變得更爲不方便，所以課務組希望共同資料庫的資料是正確而且爲最新的資料。

2.6.4. 警衛隊

1) 無法判斷教職員工是否已經離職。警衛室系統是對於資料的維護方式上採用獨自作業的方式，原因是警衛室系統所服務的對象包含了校外人士。但是在身份的判斷上，仍需要區別校內與校外人士的身分，所以在申請資料時，必須先將基本資料給與警衛室審核，才可以判斷個人的身分，在給予相對的權利。但是第二次申請時，因爲資料庫裡面已經有個人資料，所以就省略了審核的步驟，若申請者身分有變動時也就無法察覺。

2.6.5. 共同資料庫

1) 使用共同資料庫的單位並不是很普及。共同資料庫建立的目的，爲提供不同系統之間建立資料交換管道，但是校內系統使用的並不普遍，所以無法達成當初建立共同資料庫的目標。
2) 共同資料庫資料不完整。當初建立共同資料庫時，所規劃的資料包括了教職員、學生、宿舍、課程、選課…等資料，但是目前共同資料庫所維護的系統只有教職員資料
3) 共同資料庫彈性不足。因爲共同資料庫中的資料爲問卷需求所制定出來的，但是若有新系統啓用時，可能會使用到共同資料庫無法提供的資料，此時新系統與其他系統之間的資料交換，可能又會出現問題。
4) 在處理資料的更新速度仍稍嫌不足。因爲共同資料庫的資料開放給校內各部門所使用，所以在更新速度上無法很頻繁，否則可能會造成系統負荷過大。

第3章 事件驅動之整合方法

3.1. MOM (Message-Oriented Middleware)

透過網路，人們可以使用 Email 彼此互相通訊，這也是最爲普及的人與人之間的訊息系統，而這裡我們要介紹的通訊系統是可以在不同的應用程式之間可以互相溝通，也就是應用軟體與應用軟體之間的通訊機制，此機制被統稱爲 MOM (Message-Oriented Message)。

企業訊息系統，可供兩個或更多的應用軟體以訊息的形式彼此之間交換資訊。訊息中包含了商業資料和網路路由所需要的標頭。商業資料可以爲任意資訊，由使用者自行定義，通常是有關交易的資訊。在企業訊息系統中，訊息提供其他系統有關於應用軟體的事件觸發或狀態改變的資訊。

使用 Message-Oriented Middleware，訊息可以透過網路從某應用軟體傳送至其他的應用軟體。MOM 的產品可使訊息在分散的應用軟體之間正確的傳送。除此之外，MOM 通常會提供容錯、附載平衡、有彈性和提供可靠地大量交易處理的機制。

MOM 業者們使用不同的訊息格式或協定來交換訊息，但是基本的概念都是相同的。使用 API 產生訊息，使訊息能夠附加資料，加上路由的相關資訊，最後送出此訊息；使用同樣的 API 可以在其他的電腦接受訊息。

在企業訊息系統之中，應用軟體之間會使用一個虛擬的頻道來交換訊息，稱之 destinations。當某應用軟體發送訊息時，會送至某個 destination，而不是另一個特定的應用軟體。其他任何的應用軟體若之前對此 destination 有訂閱 (subscribe) 或者註冊 (register)的動作，即可收到此訊息。因此在送出訊息與接收訊息的應用軟體之間是非耦合的 (decoupled)，發送者與接收者之間並無絕對的關係。

所有的 MOM 業者們提供了應用軟體開發人員收送訊息的 API，MOM 業者們也實作了他們自行定義的網路協定、路由和管理方式。對於程式開發者來說，這些不同 MOM 業者所提供的 API 是類似的，使用相同的 API 是大眾所趨，JMS (Java Message Service) 也因而誕生。

JMS 是與業者無關的 Java API，不同的 MOM 業者都可以使用 JMS。JMS 與 JDBC 是類似的，應用程式的開發人員可以使用相同的 API 來存取不同的系統。如果業者順從 JMS API 提供完整的服務，則程式開發人員可以透過 JMS

API 使用業者提供的系統來發送接收訊息。例如，在 Progess' SonicMQ 上使用 JMS API 發送訊息，相同地也可以在 IBM's MQSeries 上使用 JMS 完成發送訊息的動作。

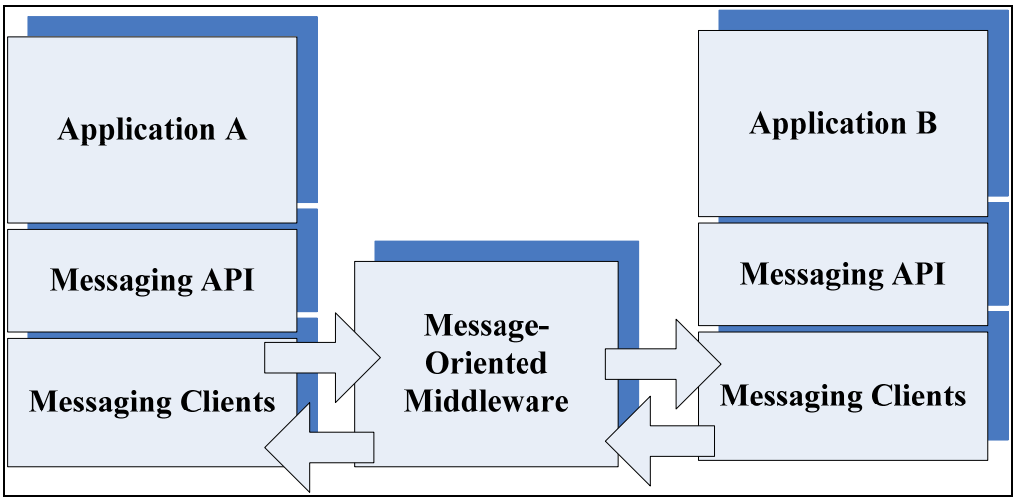


圖 21：MOM 架構圖

3.2. JMS (Java Message Service)

JMS[19][20] API 是 Sun Microsystems 所訂定訊息傳遞的標準，提供 J2EE 應用程式元件產生、傳送、接收與讀取訊息的功能。在分散式通訊的環境中，具有鬆散式耦合、可靠與非同步的特性。

JMS 本身並非是訊息系統，是 messaging clients 與 messaging systems 溝通所需要的介面與類別的抽象概念。如同 JDBC 使 relational database 的存取方式抽象化；JNDI 使 naming & directory services 的存取方式抽象化。因此只要使用 JMS 的應用軟體，可以很輕易地在不同的 MOM 產品之間運作。

JMS 的產生是由於工業界努力的成果。由 JavaSoft 領導並且與多個訊息系統業者所制定的規格。最初目的是要提供一個 Java API 來連結各個 MOM 的系統，後來發展爲更爲寬廣的目標，使 JMS 成爲 Java 分散式環境中的範本，如同 RPC 在 CORBA 與 Enterprise JavaBeans 中的地位。

JMS 提供兩種訊息傳遞模式：publish-and –subscribe 與 point-to- point queuing。在 JMS 中稱之 messaging domains。在 JMS 的專有名詞之中，常以 pub/sub 與 PTP 來簡寫。

簡單的來說，pub/sub 就是以 1-N 方式廣播訊息；而 PTP 就是以 1-1 的方式傳遞訊息。如下圖所示：

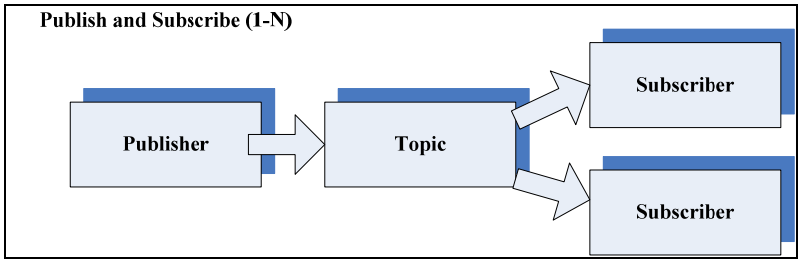


圖 22：Publish and Subscribe 示意圖

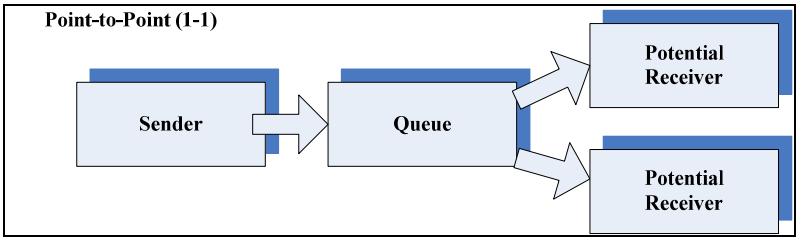


圖 23：Point-to-Point 示意圖

訊息用戶端程式，在 JMS 中稱之 JMS client；訊息系統 (也就是 MOM)，在 JMS 中稱之 JMS provider，而在商業系統中的 JMS application 指的就是包括許多的 JMS client 與一個 JMS provider。

產生訊息的 JMS client 稱之 producer；接受訊息的 JMS 稱之 consumer。JMS client 可以同時爲 producer & consumer，兩者身分並不衝突。

- **Publish/Subscribe Messaging**

首先 JMS provider 必須建立起通訊的管道『Topics』，producer 與 consumer 可以透過相同的 Topic 來互相溝通。

Consumer 對於有興趣的 Topic 做訂閱 (Subscribe) 的動作，若有 producer 對於已經訂閱的 Topic 做散佈 (Publish) 的動作時，則 JMS provider 複製此訊息並且分別送給訂閱此 Topic 的使用者。所以單一 Topic 允許有眾多 Consumer；也允許有眾多的 producer，是多對多的通訊。

基本上 pub/sub 是一個 push-based 的模式，訊息會自動的廣播給 consumer，而這些 consumer 不必發出請求或者不定時地查看是否有新訊息。

在 pub/sub 的傳輸模式中，producer 送出訊息時，不需依賴 consumer 是否接受到此訊息。使用 pub/sub 的 JMS client 可以建立持續性的訂閱 (durable subscription)，允許 consumer 重新連線時仍可接收 producer 在斷線時所發送的訊息。

- **Point-to-Point Messaging**

PTP 傳輸模式基本上與 pub/sub 是類似的，在 pub/sub 中的 producer 與 consumer 在此以 sender 與 receiver 稱之。以下則列出這兩種模式的相異之處：

- JMS client 彼此交換訊息的管道爲 queue，爲一個 destination。Sender 送訊息至 queue；receiver 從 queue 接收訊息。
- 每封訊息只會有一個接收者。一個 queue 可以有許多的 receiver，但是只有其中一個 receiver 會接收到訊息。
- 訊息是有順序之分的。Queue 依照 sender 所傳遞的順序儲存訊息，而每當 receiver 消耗訊息之後，則把訊息從 queue 中移除。
- Sender 與 receiver 之間並非耦合，兩者可以動態的在系統中加入/離開。

- **JMS Guaranteed Messaging Property**

保證傳遞是訊息系統中一個重要的部分。JMS provider 必須具有 store-and-forward 的機制，才有辦法提供保證傳遞的功能。Storage 機制是爲了儲存具有 persistent 性質的訊息，若 JMS provider 或者 consuming client 連結發生了故障，則可重新取爲此訊息。Forwarding 機制負責萃取出此訊息然後傳遞給 consumer。

除此之外，仍需要有 Message Acknowledgments 的機制，確定訊息已經成功送達。有了 Acknowledgment 協定則可以確定 JMS provider 已經收到 producer 產生的訊息；consumer 已經收到 JMS provider 傳遞的訊息。

以下爲 Nonpersistent messages 與 durable subscriber 特性的示意圖：

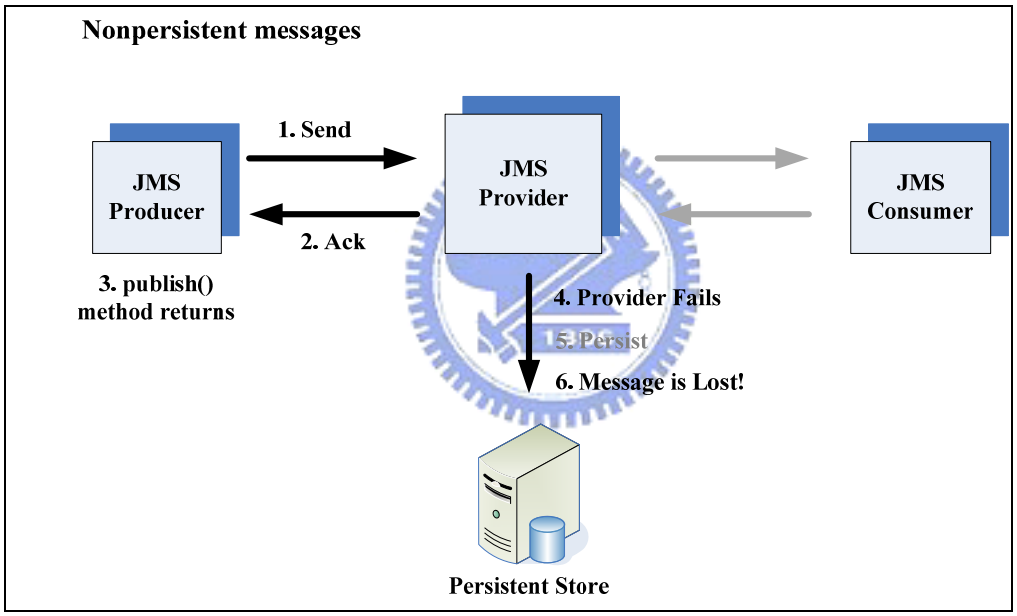


圖 24：Nonpersistent messages 示意圖

JMS Provider 在接收到 Nonpersistent 訊息時，會在第一時間直接回傳 Ack 給 JMS Producer。若此時 JMS Provider 若發生障礙，此訊息可能就會遺失。

JMS Producer 在接收到 JMS Provider 的 Ack 回應，即 publish() method return。但是此時訊息已經遺失，而 JMS Producer 無法察覺，亦無法補救。

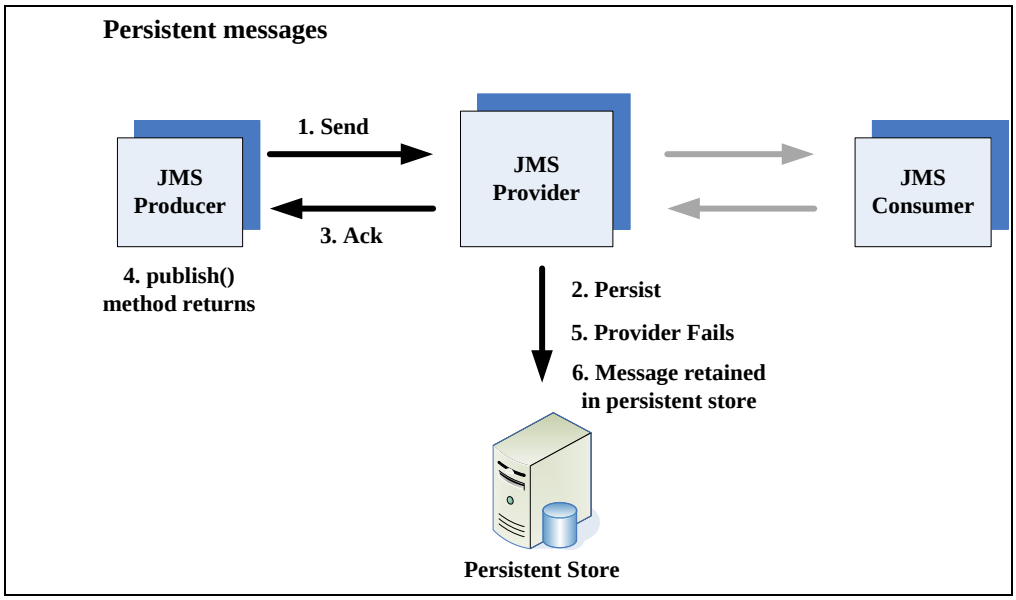


圖 25：Persistent messages 示意圖

Persistent 類型的訊息，會在訊息標頭上加上 Persistent 的記號，JMS Provider 在接收到此類型的訊息之後，會立即將訊息儲存在 Persistent Store 之中，下一個動作才是回 Ack 給 JMS Producer。以這樣的 Persistent 機制，可以確保訊息傳達至 JMS Provider。若 JMS Provider 在儲存之前發生障礙，JMS Producer 則不會收到 Ack，可以再次傳送訊息；若 JMS Provider 在儲存之後發生障礙，JMS Producer 已經接收到 Ack，所以不會再次傳送訊息，但是 JMS 仍可在 Persistent Store 取得之前 JMS Producer 傳送的訊息

因爲有了 Persistent messages 機制，當 JMS Producer 接收到 Ack 時，可以確認訊息已經成功傳送，此時 publish() method 即可以成功 return。

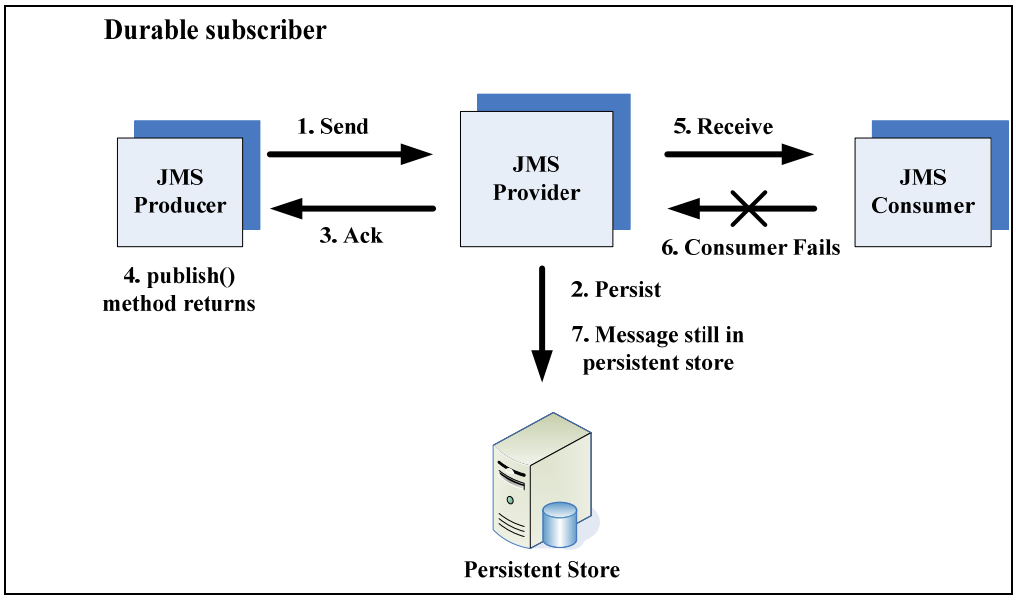


圖 26：Durable subscriber 示意圖

Durable subscriber 是 JMS Provider 所提供的功能。這類型的訂閱者，若以 durable subscriber 的方式向 JMS Provider 提出接收訊息的要求，JMS Provider 必須負起訊息傳送的任務，即使 subscriber 離線，也要保存此訊息，直到 subscriber 上線爲止 (或訊息過期)。

除此之外，若 JMS Consumer 發生障礙，亦可防止訊息遺失。當 JMS Provider 傳遞訊息給 JMS Consumer 時，JMS Consumer 發生障礙，JMS Provider 不會收到 JMS Consumer 回傳的 Ack，此時 JMS Provider 可從 Persistent Store 中取回之前訊息，然後再次傳送。

直到所有的 Durable subscriber 接收到訊息或者訊息過期，JMS Provider 才會將訊息從 Persistent Store 中刪除。

3.3. SQL Server Notification Services

SQL Server Notification Services (SSNS)[6][7] 提供了應用程式發展和部署的平台，透過 SSNS 發展的應用程式可以產生個人化的訊息且即時地通知訂閱者。以下爲 SSNS 的架構全覽圖：

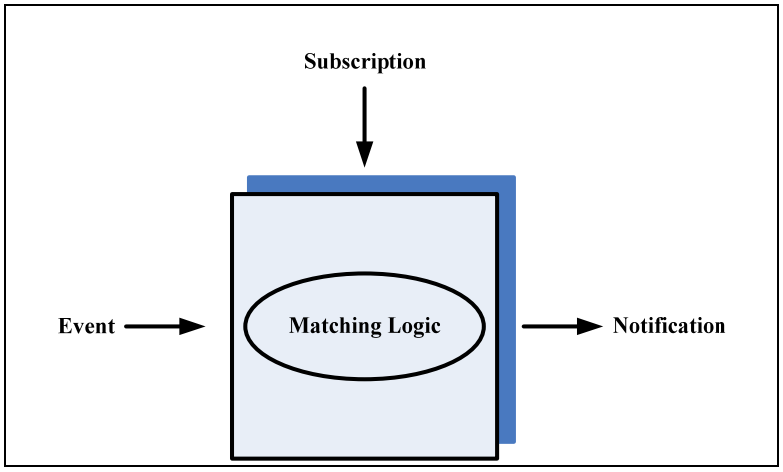


圖 27：SSNS Architecture Overview

Events 可以視爲資料，用來描述在真實世界中的事情。例如，股價的改變，以股票代碼與股價來描述這樣的事件。

不管是怎樣的型態的事件，都可以結構化的資料來表示。設計一資料表的欄位名稱與資料型態，最後以此資料表來呈現資料。以下爲 Stock Event 的例子：

Stock Symbol	Stock Price
XYZ	55.55
PQS	95.30
JKL	15.00

表 6：Stock Event Table

相同地，對於 Subscriptions 的描述，亦可以同樣的表格方式來呈現。Events 與 Subscriptions 以同樣的資料表結構來呈現資料，有助於提高比對效率。

例如，當股票高於目標價位時，則發出通知。以訂閱者、股票代碼以及價位的資料表欄位來呈現訂閱資料，以下爲 Stock Subscription 的例子：

Subscriber	Stock Symbol	Price Target
Bob	XYZ	50.00
Alex	PQS	90.00
Mary	JKL	12.00
Jane	PQS	100.00

表 7：Stock Subscription Table

因爲 Events 與 Subscriptions 都採用資料表的方式儲存資料，這樣的設計有助於 Events 與 Subscriptions 進行 SQL JOIN 的比對。根據使用者的需求，設計 SQL 語法的比對規則，執行規則可以呈現通知的內容。以下爲比對的規則：

```
SELECT    S.Subscriber, E.StockSymbol, E.StockPrice
FROM      Events E JOIN Subscriptions S
ON        E.StockSymbol = S.StockSymbol
WHERE     E.StockPrice >= S.PriceTarget
```

在比對規則中，將 Event table 與 Subscription table 以股票代號做 join 的比對，條件是股價 > 目標價位，並且顯示訂閱者、股票代號以及股價，以下爲比對結果，即爲通知的內容：

Subscriber	Stock Symbol	Stock Price
Bob	XYZ	55.55
Alex	PQS	95.30
Mary	JKL	15.00

表 8：Stock Notification Table

訂閱者 (subscriber) 可以使用通知服務應用軟體來自訂訂閱內容 (subscription)，訂閱內容可以呈現訂閱者有興趣的內容，例如，訂閱者的訂閱內容是『當股票價錢達到 70 元時通知我』。一旦事件發生，通知訊息可以在最快的時間被產生且送出給訂閱者。通知訊息也可以依照事先排定的行程來發送給訂閱者。

通知訊息傳送的目的地可以是各式各樣的裝置。例如，通知訊息可以傳送至手機、Personal Digit Assistant (PDA)、MSN (Microsoft Windows Messenger) 或是某個特定電子郵件帳號。因爲這些裝置常常伴隨在訂閱者身邊，所以重要通知訊息可以很迅速地被告知。

透過通知服務平台你可以快速、有彈性地建立且部署應用程式，來支援眾多訂閱者的需求。通知服務平台包含了有 1) 簡而有力的通知服務平台，幫助您快速的建立且部署通知應用程式，可以透過 XML 或 Notification Services Management Object (NMO) 發展應用程式。 2) 一個具有可靠、高效能、有彈性的引擎來執行通知應用軟體，通知服務引擎是以 Microsoft .NET 架構和 SQL Server 2005 爲基礎所建立的。

SSNS 架構分成四部份：

- Subscription Management Architecture
  提供訂閱者管理訂閱內容的介面
- Event Collection Architecture
  提供收集資料變動資訊的功能
- Subscription Processing Architecture
  提供比對事件與訂閱內容產生通知的功能
- Notification Formatting and Delivery Architecture。
  提供傳送通知的功能

以下章節分別詳述此四部份的內容。

3.3.1. Subscription management Architecture

訂閱內容管理應用程式是以自訂訂閱內容管理介面所建立的，介面可以是網頁應用程式、Microsoft Windows 應用程式、主控台應用程式或預存函式。在執行個體和應用程式資料庫下，介面提供了管理訂閱者、訂閱裝置 (subscriber device) 和訂閱內容的功能。

通知服務平台提供了管理與閱覽訂閱內容的功能，大大地簡化了發展介面的過程。透過以下圖例可顯示出，以訂閱內容管理介面來操作訂閱內容管理物件與通知服務資料做溝通。

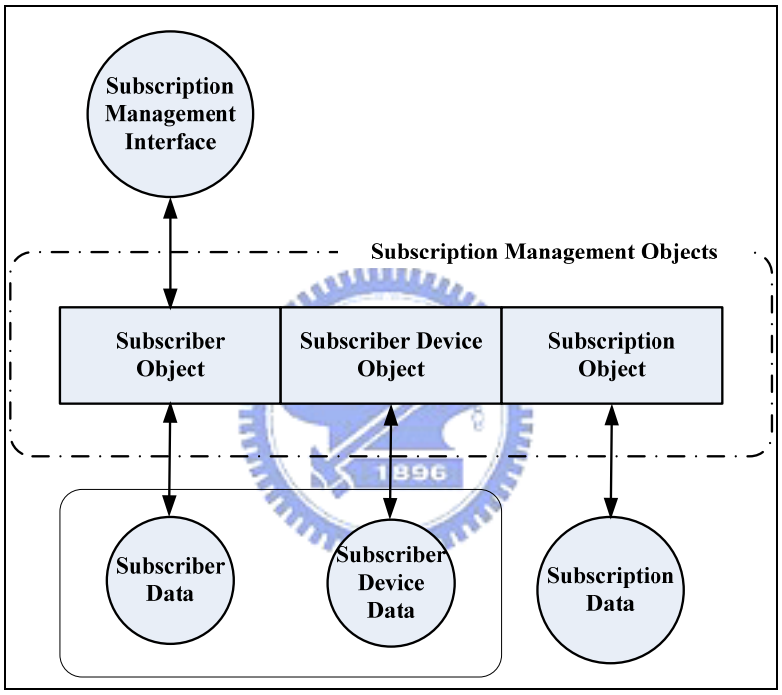


圖 28：Subscription Management Architecture

通知服務平台以執行個體資料來儲存訂閱者和訂閱裝置的相關資料，而訂閱內容則以應用軟體資料的方式來儲存。這樣的儲存方式，即使在不同的應用軟體儲存訂閱內容時，應用軟體之間還是可以分享訂閱者的資訊。

當通知服務應用程式執行時，應用程式透過訂閱內容的資料來產生通知訊息，並且以訂閱者與訂閱裝置資訊來格式化與散佈這些通知訊息。

當應用程式產生通知訊息時，每個通知訊息都必須包含訂閱裝置的資訊。通知訊息中的訂閱裝置必須與訂閱者設定的訂閱裝置產生對應，否則這個通知是不會被傳送的。

3.3.2. Event Collection Architecture

事件收集是從一個或多個來源 (XML 檔案、應用程式或資料庫) 來獲得事件資訊的過程，並且送出這些資訊給通知應用軟體。這就是 Event Provider 處理的事項。

每一個應用軟體可使用一個以上的 Event Provider 來收集事件。Event Provider 可以透過三種方式將資料送至應用程式，這三種方式分別爲：Event Object API、XML API 以及 SQL Server API。以下圖例顯示這三種 API 如何運作：

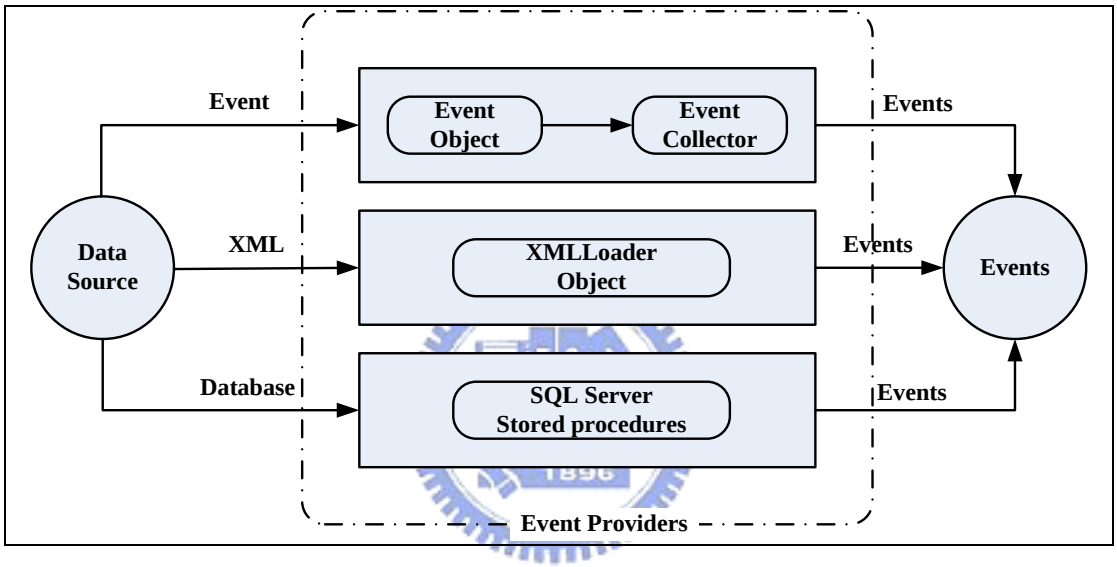


圖 29：Event Provider Architecture

Event Object API 使用 Event object 和 EventCollector 物件來收集並且送出事件。透過 Event table 中的一個欄位名稱，應用軟體可以送出一個 Event object 給 Event collector，最後 Event collector 再寫資料到 Event table。

XML API 提供存取 XML 資料的功能。XML Event Provider 從一個 XML 文件或者是串流收集事件資料，並且送出資料給 XML EventLoader，最後 XML EventLoader 再寫資料到 Event table。

SQL Server API 透過 stored procedure 方式來收集事件。通常有兩種使用 SQL Server Event Provider 的方法，第一是透過 stored procedure 呼叫 Event Provider，第二種是定期地執行一個查詢。Event Provider 接收到一些結果的集合之後，再使用 stored procedure 把這些事件寫到 Event table。

- **標準和自訂的 Event Provider**

通知服務應用程式開發者可以使用上述的任何一種 API 來撰寫自訂的 Event Provider，或者可以使用通知服務提供的 Standard Event Provider。Standard Event Provider 可以從一個監視的資料夾中擷取 XML 資料、查詢 SQL Server 資料庫或者 Analysis Services cubes 來獲取事件資料。

自訂的 Event Provider 可支援 Standard Event Provider 無法提供的功能。例如，你可以從一個檔案中取得以逗點相隔的股價資訊，透過使用通知服務 API，開發者可以建立具有這樣的一個功能的 Event Provider。

- **本機與非本機的 Event Provider**

本機提供者在通知服務中執行。本機的 Event Provider 可以以持續地或依照已排定的行程來執行任務。這些 Event Provider 可以被『Event Provider Host』的通知服務元件所執行。Event Provider Host 可以與 Generator 元件使用同一個行程。

非本機的 Event Provider 以外部應用軟體的身分執行，且以自己所訂定的行程送出事件。例如，某個 Event Provider 架設在 IIS 上，以網頁的方式送出事件，這樣就是一個非主機的 Event Provider。Event Provider 架設在自行撰寫的程式之中，也是屬於非主機的 Event Provider。

- **事件群組**

Event Provider 以群組方式供應事件。以群組方式供應事件可以讓 Generator 一次將所有的事件與訂閱內容做 join 動作。這樣群組導向的處理可以改善應用程式的效能。

3.3.3. Subscription Processing Architecture

收集事件之後，通知服務便可以處理訂閱內容來產生通知。比對訂閱內容來過濾事件是 Generator 元件主要的任務。

爲了要產生通知，應用軟體開發者會產生一個或多個規則。這些規則是以 Transact-SQL 來描述事件與訂閱內容之間的關係。

在一個簡單的應用軟體之中，當 Generator 觸發了某一個規則，應用軟體比對所有的訂閱內容與事件群組的關係。若有事件符合訂閱內容時，Generator 會立即產生一個通知。這通知包含了有關這事件、訂閱者和訂閱裝置的資訊，偶爾也包括訂閱者的所在地，和其他在散佈時所需要的資訊。

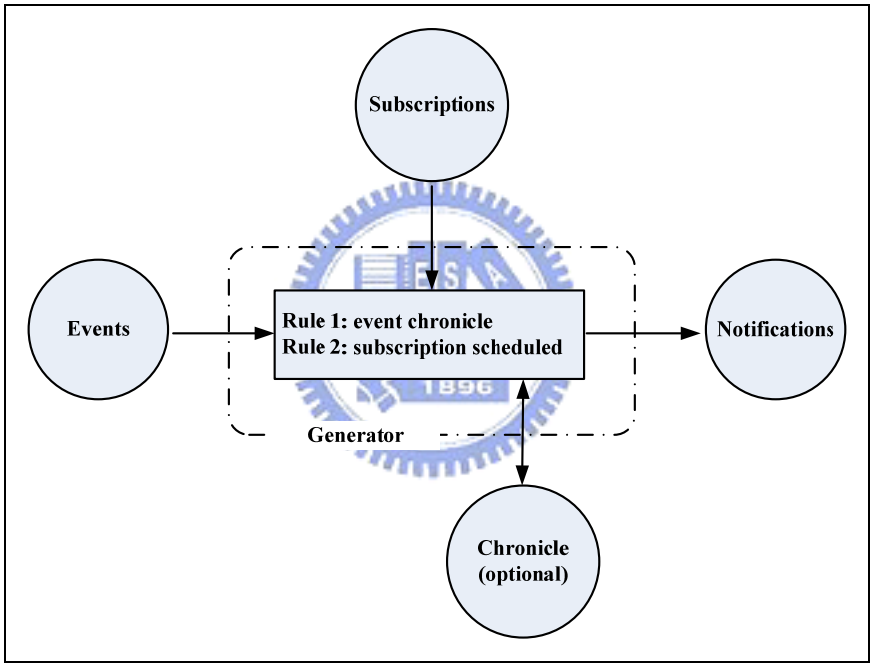


圖 30：Generator

這通知產生之後並不會馬上被送出。相對的，這 Generator 把這通知寫入 Notification table。當整批的通知已經準備好時，這些通知會被格式化，並且由 Distributor 散佈出去。

如果應用程式支援已排程的訂閱內容，當 Generator 處理這些已排程的訂閱內容，僅會查看應評估的訂閱。例如，若 Generator 每 15 分鐘執行一次，在 8:00 時，Generator 評估在 7:45 至 8:00 之間的行程。

如果應用程式可使用歷程記錄資料，應用程式可以在額外的資料表中，儲存資料和事件及訂閱資訊，並且使用這些資料來產生通知，這額外的資料表稱之

Chronicle。以下爲使用 Chronicle 的例子：

時間	股票代號	股價 ($USD)
09:00 GMT	AWKS	69.98
09:20 GMT	AWKS	70.35
09:40 GMT	AWKS	70.87
10:00 GMT	AWKS	71.55
10:20 GMT	AWKS	72.00

表 9：Chronicle Example

在範例中會顯示如何利用 Chronicle，根據指定期間，顯示最高股價的事件資料並且防止重複的通知。當所選股票超出預先定義的臨界值時，股票通知應用程式會向您發出通知。

應用程式會每隔 20 分鐘從 Web 服務取得新的股票事件資料。上表顯示早上的資料。事件驅動訂閱是針對 AWKS 股票的值到達 $71.00 或以上的通知。因此，您會收到以 10:00 GMT 資料的通知。在處理事件批次且產生通知之後，根據事先定義應用程式的 Event Chronicle Rules，會在 Chronicle Table 中輸入和更新當天的高值 (當前是 $71.55)。

10:20 GMT 事件資料到達，定義給訂閱規則的通知產生動作會利用 Chronicle Table 來防止重複的通知。它的方式是在過了觸發價格之後，除非到達新高，否則，便排除重複的通知。而 10:20 GMT 事件 $72.00 已經超出 $71.00 的臨界值且已經超越 Chronicle Table 中的 $71.55 的紀錄，所以會更新 Chronicle Table，並且發出通知。

3.3.4. Notification Formatting and Delivery Architecture

在通知服務 Distributor 元件的處理過程中，通知可以被格式化和散佈。在 Generator 產生一批通知之後，Distributor 會依照這些通知所使用的傳送通道來做分類，然後這些分類的項目會被 formatter 做格式化的動作。剛格式化結束之後，Distributor 將這些完成格式化的通知，依照他們所需的傳送通道來散佈通知。

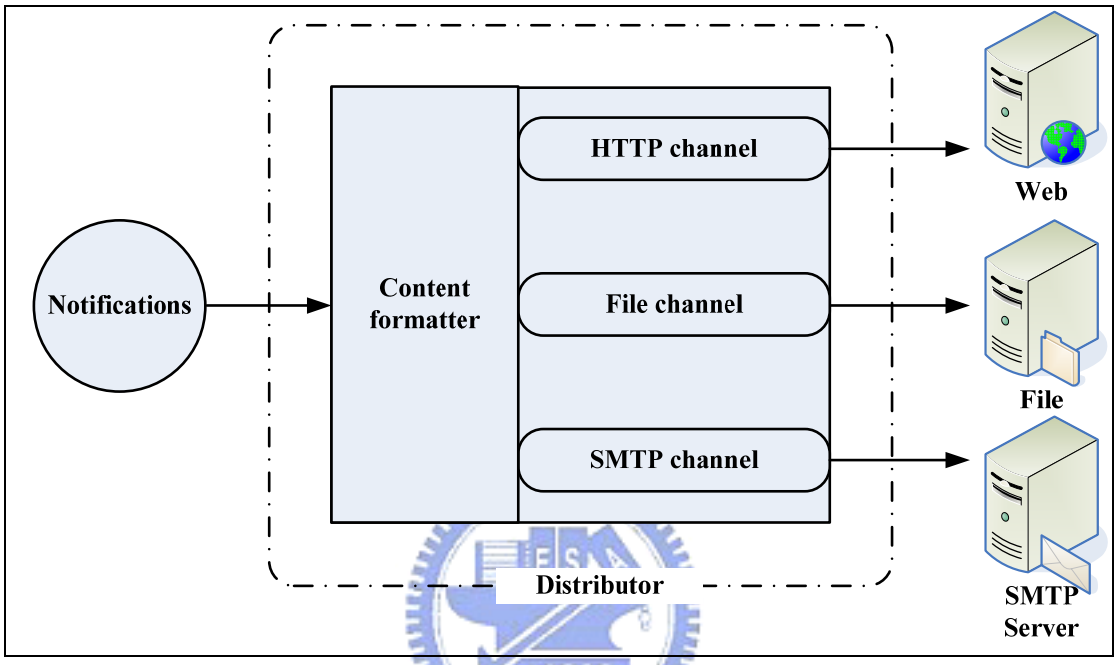


圖 31：Distributor

- **Message Formatting**

在開發應用程式的階段就已經定義了要如何將原始的通知資訊轉換成可方便讀取的訊息。通知服務提供了 XSLT (Extensible Stylesheet Language Transformation) 標準的格式轉換器，除此之外也可以發展自行定義的內容轉換器來格式化自己的通知訊息。

- **Delivery Channels**

通知服務本身並不處理最後傳送動作。相反地，通知服務使用像水管一樣的傳送通道來散佈通知，例如，可以透過 Simple Mail Transfer Protocol (SMTP) 的服務來傳送資料。通知服務送出通知給一個或者多個傳送通道。每一個傳送通道依次地將這些通知以傳送的協定打包起來，然後送給傳送服務。最後傳送服務負責傳送這些通知給訂閱者。

你可以設定通知執行個體有哪些傳送通道。所有架設在這執行個體上的應用軟體可以分享這些傳送通道。

每一個傳送通道使用一個特定的協定來傳送通知給訂閱者。通知服務包含了以下這些共用的協定：

1) SMTP 協定可送出通知給 Microsoft Exchange Server 或者是其他的 SMTP server。

2) 可延伸的 HTTP 協定可以使用 Simple Object Access Protocol (SOAP)、Short Message Service (SMS)、.NET Alerts 和其他與 HTTP 相關的協定。

3) 檔案協定可送出通知到作業系統中的檔案，通常是用來除錯。

如果需要的話，可以建立多個具有相同協定的傳送通道。你也可以自行定義傳送協定，如果你想要支援其他種類的裝置或者想要使用特定的 SMTP 或檔案特性。

- **Tracking and Retrying Notification Distribution**

爲了支援通道重傳、效能監控和除錯的功能，通知服務保留了通知傳送的資訊。傳送通道呼叫一個傳送通道回饋來報告每次通知的傳送是否成功或失敗。這個回饋可以透過更新通知表格，可反映出每一個通知的成功或失敗。你可以使用 NSNotificatinClassNameNotificationDistributin 檢視來查看傳送的狀態。

對於每一種你所傳送的通知型態，你可以設定不同的重傳行程。

- **Delivery Options**

除了標準的訊息格式化與傳送之外，提供了兩種傳送的選項：1) Digest delivery 2) Multicast delivery。要注意的是，以上兩種選項，只能選擇其一來啓用，不可以同時選擇這兩種方法。以下以例子來說明此兩種選項的特性：

此圖表爲即將要發出的通知資料：

NotificationId	SubscriberId	DeviceName	SubscriberLocale	City	State
1	Bob	e-mail	en-US	Seattle	WA
2	Alan	e-mail	en-US	Seattle	WA
3	Bob	e-mail	en-US	Spokane	WA
4	Bob	e-mail	en-US	San Francisco	CA
5	Bob	file	en-US	Tacoma	WA

表 10：Delivery Options Example

Digest delivery

如果有多個通知是要傳送給同一訂閱者，像是關於股價的資訊，你可以先整理這些資訊，使他們成爲單一的訊息，再以一個訊息的方式發出通知。Digest delivery 的傳送方式可以減輕系統格式化與傳送的負擔。

以上面圖表爲例，根據 [SubscriberID]、[DeviceName]、[SubscriberLocale] 以及 [State] 這四個欄位將通知分類，若此四個欄位相同，則代表此通知將送給同一個訂閱者。例子中，第 1 個與第 3 個的通知將會合併成一個訊息來發送，第 2 個通知的 [SubscriberID] 不同，第 4 個通知的 [State] 不同，而第 5 個通知的 [DeviceName] 不同，因此根據以上圖表的資料，這些通知都不會合併起來。

Multicast delivery

如果一個應用程式產生一個通知要送給許多的訂閱者，像是運動分數或天氣報告的資訊，你可以透過廣播傳送來傳送。將通知格式化一次之後，然後送給多個使用者，這樣的處理方式可以改善效能且減輕系統格式化的負擔。

以上面圖表爲例子，第 1 個和第 2 個通知的 [City] 和 [State] 資料相同，但是訂閱者不一樣。如果有啓用 Multicast delivery，Notification Services 會將第 1 個通知格式化，然後將格式化的訊息連同收件人清單 (Bob 和 Alan) 傳給傳遞通訊協定。不過，因爲第 3 個、第 4 個和第 5 個通知的 [City] 和 [State] 資料都不同，因此這些通知都不是多重傳遞。

第4章 DEMCD 系統設計與架構

4.1. 設計考量

在第二章《需求分析》之中，我們可以發現校內現有系統在整合的層面上，遇到了一些問題，例如：資料過時、異質平台，無法有效的交換資料、相同資料在不同資料庫中定義的欄位名稱不同、部分資訊的交換尚未完全電子化、資料有效認證問題、外包系統擴充開發不易…等，因此我們針對這些問題，提出 DEMCD (Distributed Event-based Middleware for Common Database System) 架構，來解決校內系統在整合上所遭遇到的問題。底下列出幾點來說明我們主要的設計考量：

(1) 相容性

校內現有系統在處理單獨的業務上已經表現地相當穩定，而且校內使用者也都已經相當熟悉這些系統的操作，若爲了整合不同系統之間的運作而變更現有系統，可能導致系統在處理單獨作業上會出現原本所沒有的問題，而把問題的層面給擴大，這樣的設計方式並不聰明的。所以在不改變現有系統的狀態下，我們希望提出的架構不會影響到正在運作的系統，而且在不改變使用者慣有的使用模式。

(2) 異質平台

校內系統並非一致，包括作業系統、資料庫系統、應用軟體開發語言…等等。這些因素都會增加我們在整合上的困難，也是校內系統之所以無法有效的相互運作的原因，因此我們必須提出一個可以跨平台、跨資料庫的架構，來達成我們整合上的需求。

(3) 即時更新資料

校內系統在處理單獨的業務上已經很成熟，但是對於需要部門之間合作處理的業務也許就會狀況百出。這些跨部門或跨系統的業務，常常會需要用到非自己部門或系統所能夠維護的資料，所以這處理這樣的一個業務上，必須仰賴其他部門或系統所提供的資料才有辦法正常處理業務。

例如：文書組的郵件管理系統，文書組本身的業務不包括維護教職員生的資料，但是在郵件管理系統上必須搜尋搜尋收件者的單位，以便於發送信件，所以文書組此時就需要人事室與註冊組提供相關的教職員生資料，才有辦法順利完成業務。

對於不同部門之間傳遞資料的機制上，校內尚未建立一個管道，所以傳遞資料就顯得相當不方便，也因如此，資料可能就因此沒有更新，而導致

業務上處理發生錯誤。所以這此我們希望提出的架構能夠提供即時的資料更新機制，以期能夠解決以上問題。

4.2. 系統架構

爲了整合校內各部門系統運作，在這我們提出一個以 MOM 爲主體的傳輸架構，再配合 SSNS 平台所提供的事件通知功能的 DEMCD (Distributed Event-based Middleware for Common Database System) 架構。

DEMCD 架構主要是提供校內部門系統之間一個溝通的管道，透過 DEMCD 不同系統之間可以迅速、可靠且有效率地傳遞資料。除此之外，亦可以利用 DEMCD 架構來簡化目前現有系統的處理程序，甚至可以快速地發展新的系統。

DEMCD 架構由以下三大元件所組成：

1) 資料庫事件接收模組
2) 訊息傳送/接收模組
3) 資料庫更新模組

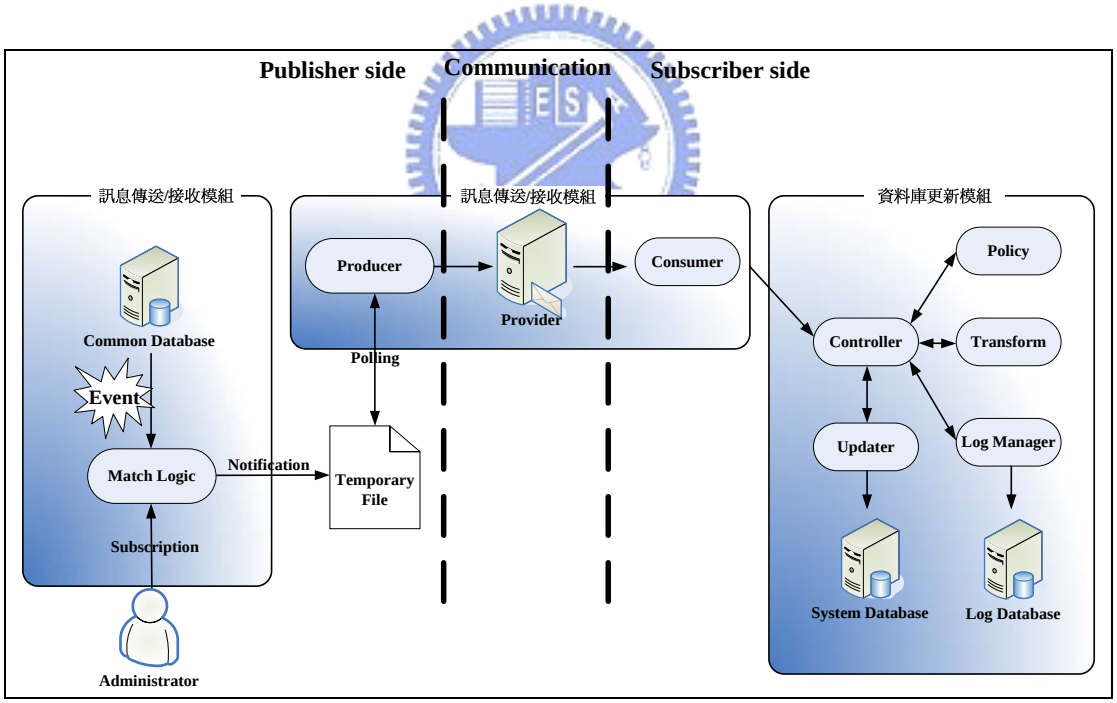


圖 32：架構全覽圖

4.2.1. DEMCD—資料庫事件接收模組

4.2.1.1. 模組功能說明

資料庫事件接收模組，是架構在 SQL Server Notification Services 與 .NET 之上的應用程式，主要功能爲接收資料庫異動時所產生的事件和提供訂閱者自行管理訂閱內容的介面。

在資料庫事件接收模組中，以 SQL Server Notification Services 爲平台，設計事件接收元件，用以接收 SQL Server 產生變更的事件。事件接收元件在收到資料庫變更的內容時，則將此資訊寫入 Temporary File。

此 Temporary File 爲事件接收元件與訊息傳送/接收模組的溝通橋樑[21]，因爲事件接收元件爲 .NET 應用程式而訊息傳送/接收模組爲 Java 應用程式。

資料庫事件接收模組在 DEMCD 架構中所扮演的腳色如下圖所示：

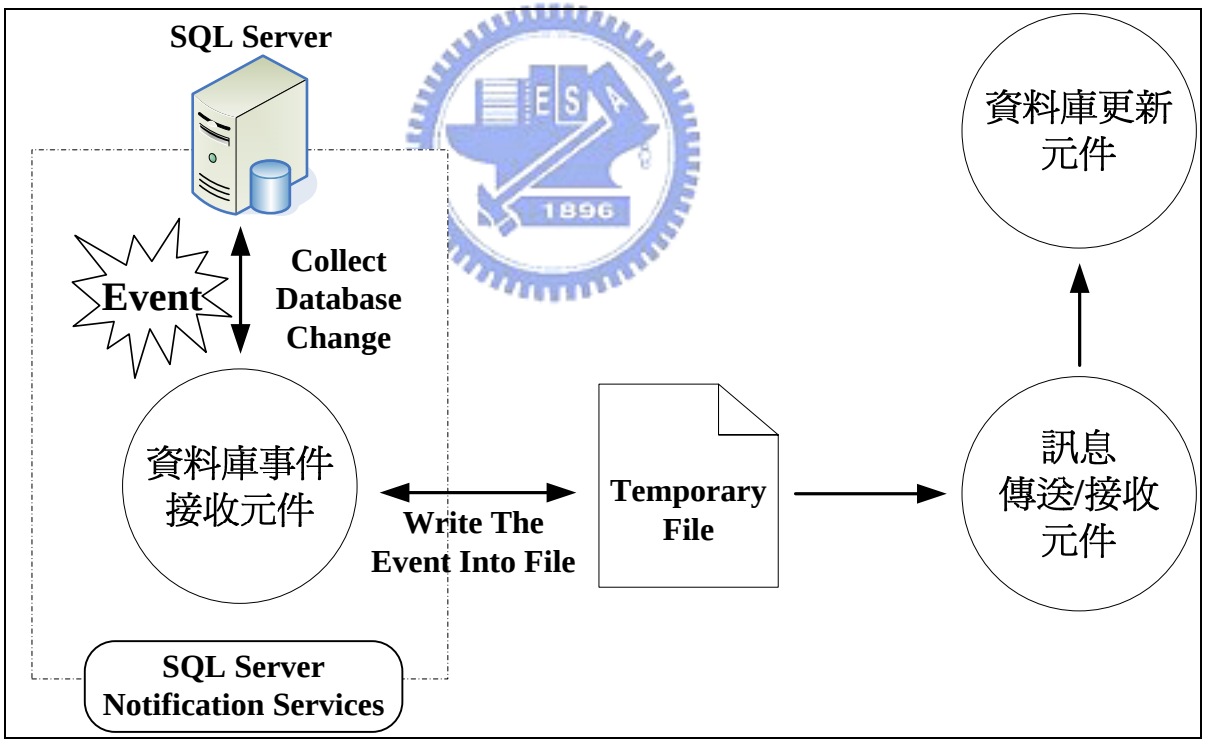


圖 33：資料庫事件接收模組

在資料庫事件接收模組之中，包含有 Subscription Management、Event Provider、Generator 以及 Distributor 四個元件，而且在 SQL Server 之中也會建立 Events table、Subscriptions table 以及 Notifications table，用以儲存事件、訂閱內容以及通知的資料，彼此之間的關係如下圖所示：

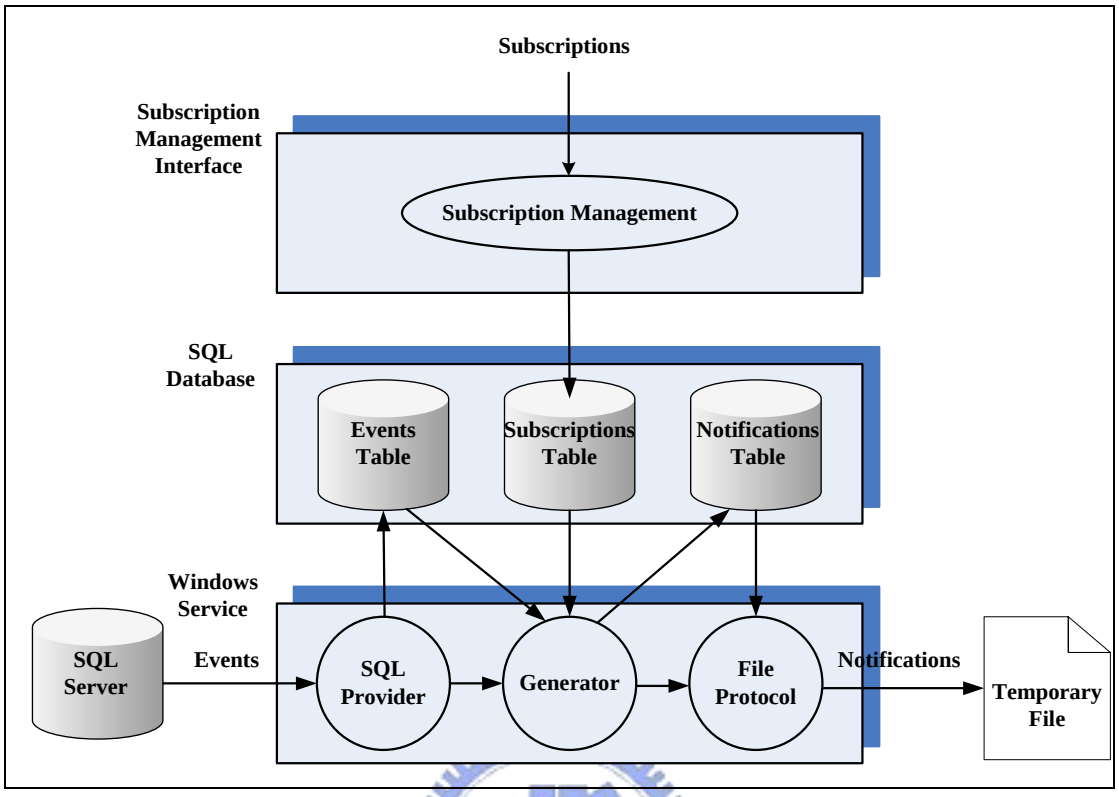


圖 34：事件接收元件架構圖

4.2.1.2. Subscription Management 元件

Subscription Management 主要功能爲提供 SQL Server 的管理者一個方便使用的介面，可用來管理訂閱者、訂閱內容以及訂閱裝置。

而在 DEMCD 架構中，Subscription Management 的功能很簡單，因爲訂閱者就是管理者自己本身，管理者可以設定系統想要散佈的資料當作爲訂閱內容，而訂閱裝置就是本機自己本身。

4.2.1.3. Event Provider 元件

Event Provider 可從許多不同的資料來源來收集事件，在 DEMCD 架構中，Event Provider 主要的資料來源爲 SQL Server，透過 SSNS 中內建的 SQLProvider 可收從 SQL Server 中收集有興趣的資料，一旦這些受到關注的資料變更，SQL Server 會主動傳送 Event 給 Event Provider。

SQLProvier 可以週期性地執行 SQL 查詢，此查詢可以由應用軟體來定義。查詢可以從來源資料表中取得資料，當查詢的資料回傳時，SQL Provider 會送出整批的事件至 SQL-NS 應用軟體。查詢和送出查詢結果這兩個動作，都是

發生在資料庫中，因此 SQLProvider 主要是促使事件能夠從來源資料表送至 SQL-NS 應用程式，而不須在資料庫中記錄任何資料。

4.2.1.4. Generator 元件

Generator 可將 Events table 與 Subscript table 資料作 SQL join，所產生的結果就是通知的內容。再將產生的內容寫入 Notifications table 中。

4.2.1.5. Distributor 元件

Distributor 主要是將通知散佈至訂閱者的手上，在 DEMCD 架構中，Distributor 的目的地爲本機的 Temporary File，所以使用 SSNS 內建的 File Protocol 將通知寫入 Temporary File 之中。

4.2.2. DEMCD—訊息傳送/接收模組

4.2.2.1. 模組功能說明

因爲資訊的傳遞是 MOM 架構，所以在訊息傳送與接收元件的設計上，以 JMS API 爲主，可分成三部份，JMS Provider、JMS Producer 以及 JMS Consumer。

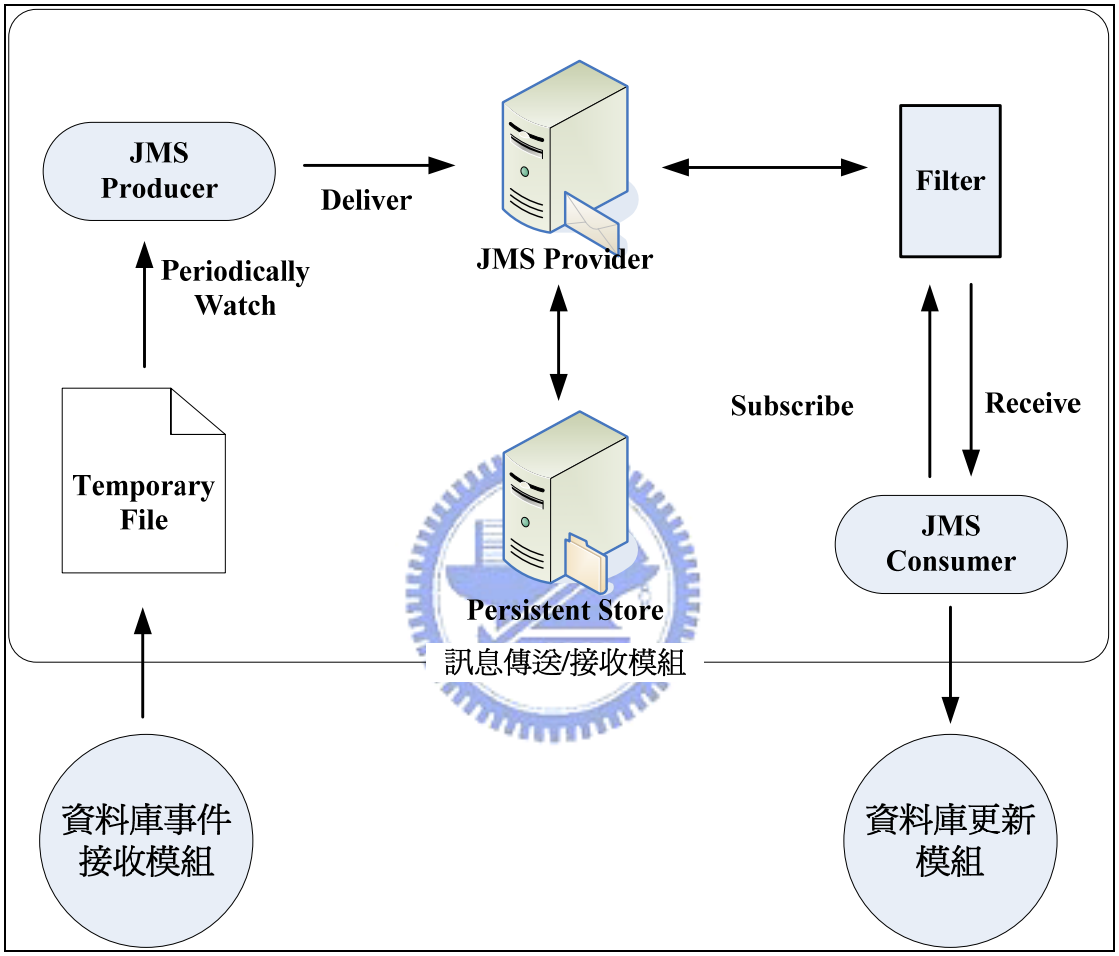


圖 35：傳送接收元件

JMS Provider 提供訊息傳遞的服務，JMS Producer 負責將資料庫事件接收模組所蒐集到的事件資訊發送給 JMS Provider，最後 JMS Consumer 從 JMS Provider 接收訊息之後，將訊息的內容傳遞給資料庫更新模組。以下分別詳細探討這三部份所負責的功能：

4.2.2.2. DEMCD—JMS Provider 元件

JMS Provider 負責訊息的傳送功能，將訊息從 Publisher 傳送到 Subscriber 手上。JMS Provider 根據訂閱者訂閱的資料，來決定訊息的傳送目的。

爲了提高傳遞訊息的可靠度，JMS Provider 具有 Persistent Store 機制來儲存 JMS Producer 所送出的訊息，以免訊息因爲障礙而遺失。

除此之外，JMS Provider 必須具有 filter 的功能。Filter 有存取訂閱者資訊的權限，因此可以得知訂閱者的訂閱內容，當訊息要傳送給訂閱者時，filter 可以過濾非訂閱者所訂閱的內容，可節省傳送的頻寬，亦可增加訊息的保密性。

4.2.2.3. DEMCD—JMS Producer 元件

JSM Producer 爲訊息傳遞系統中的最前線，負責訊息的來源處理，位於 Publish 端的 JMS Producer 不斷地輪詢 Temporary File，檢查檔案最後更新時間，若發現時間改變，就表示『資料庫事件接收模組』已經將最新的通知寫入 Temporary File，此時開啓檔案，讀取最新的內容。

JMS Producer 在取得最新的資訊之後，將資料以 Persistent message 方式封裝，增加資料的可靠度，並且傳送給 JMS Provider，完成傳送資訊的任務。

4.2.2.4. DEMCD—JMS Consumer 元件

JMS Consumer 位於 subscribe 端，負責接收訊息，並且將訊息傳送給『資料庫更新模組』做處理。

4.2.3. 資料庫更新模組

4.2.3.1. 模組功能說明

資料庫更新模組負責更新本地端資料庫的任務，提供資料庫更新、恢復資料庫內容、執行策略設定以及資料轉換的功能。

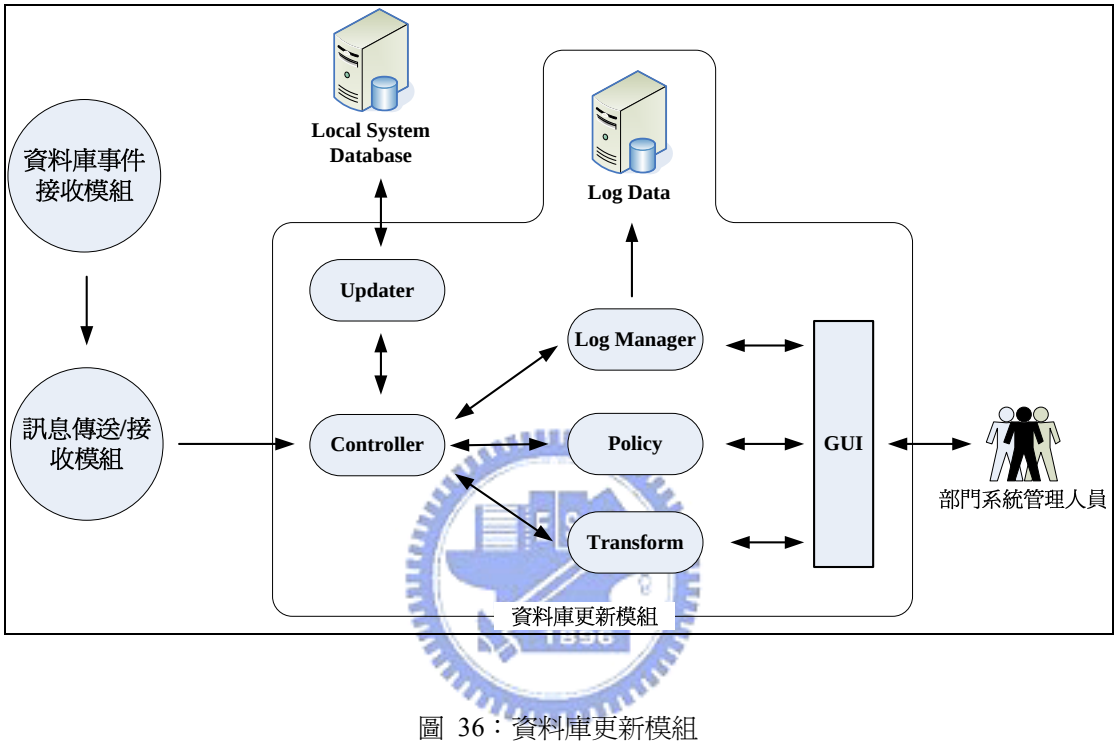


圖 36：資料庫更新模組

資料庫更新模組主要的組成單位有：1) Controller 2) Log Manager 3) Updater 4) Policy 5) Transform。以下詳述這些單位的功能：

4.2.3.2. Controller

主要負責資料庫更新模組運作的流程，透過 Controller 訊息可以在其他單位之間做傳遞。

4.2.3.3. Log Manager & Log Database

當接收到訊息時，Controller 會在第一時間透過 Log Manager 將訊息儲存至 Log Database，然後再回 Ack 給『訊息傳送/接收模組』，以完成接收訊息的動作。因爲有 Log Database 的機制，所以可以提供相當程度的可靠性。除此之外，先將訊息儲存起來，可以提供執行上的彈性，讓使用者決定何時將資訊更新至本地端的資料庫。

4.2.3.4. Updater

Updater 可以將訊息更新至本地端的資料庫。Updater 可以處理從 Controller 接收的訊息，轉換成 SQL 語法，透過 JDBC 更新本地端資料庫。

4.2.3.5. Policy

Policy 單元可提供系統管理者自行設定執行更新資料庫的策略，Controller 依照 Policy 中的設定，決定何時更新本地端的資料庫。例如，當收到新訊息時，系統管理者可透過 Log Manager 查看收到的訊息，並且自行手動更新資料庫，或者決定由系統自行更新資料庫。

4.2.3.6. Transform

來源端的欄位與本地端的欄位不一定相同，爲了解決此問題，所以設計 Transform 單位。透過 Transform 單元，系統管理者可以自行定義欄位的對應關係，如此一來可以解決欄位差異的問題。

4.2.4. DEMCD—教職員升遷案例

在這章節以教職員升遷爲例子，模擬 DEMCD 架構運作流程，使得架構的功能更容易了解。

人事室資料管理系統在固定時間更新共同資料庫，此時常駐在共同資料庫的 SQL-NS 應用軟體—資料庫事件接收模組，會立即察覺到共同資料庫有進行更新，並且將此更新的相關資料送出至 JMS Provider，再轉送給資料的訂閱者—課務組，最後課務組接收到更新的資訊並且更新教師鐘點合計系統的資料。

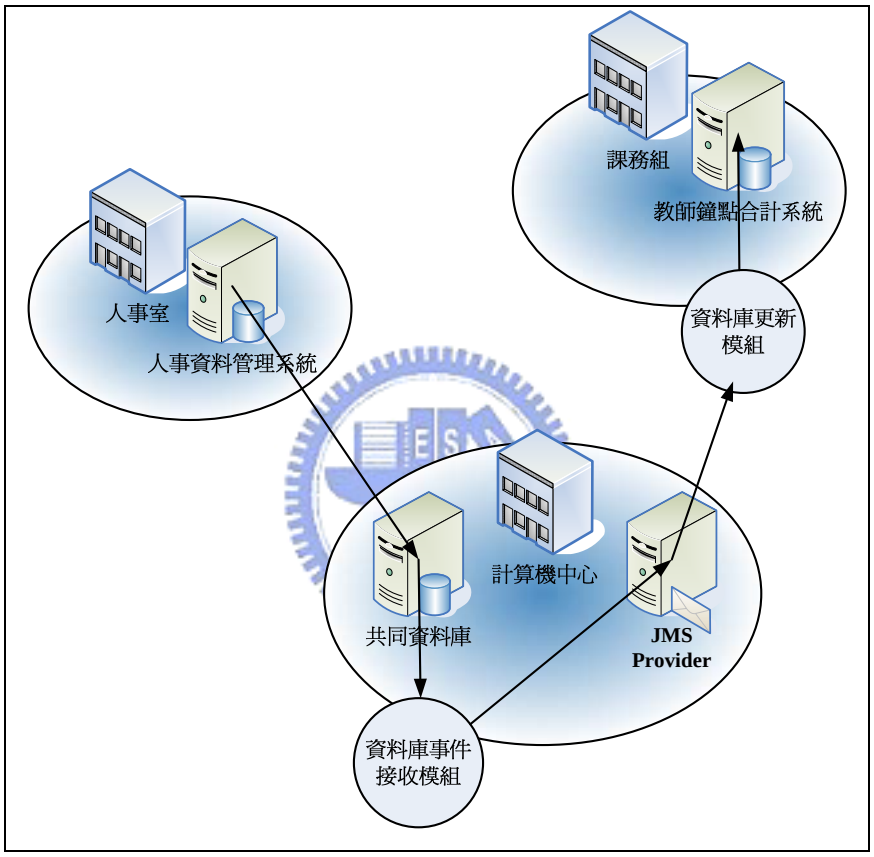


圖 37：教職員升遷案例-全覽圖

資料庫事件接收模組中的 Subscription 內容爲 ”ALL”，因爲對於共同資料庫的資料表來說，所有的資料都可能被其他部門所使用，所以在 Subscription 的設定上爲 ALL，也就是說，所有的 Event 都爲 Notification。

而在比對的規則設計上，爲了擷取在共同資料庫公務人員資料表中的所有資料，將公務人員的資料表設定爲資料來源，語法如圖 39 所示。

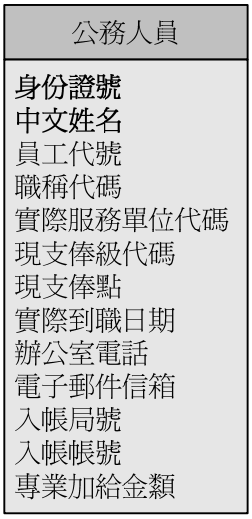


圖 38：共同資料庫-公務人員資料表

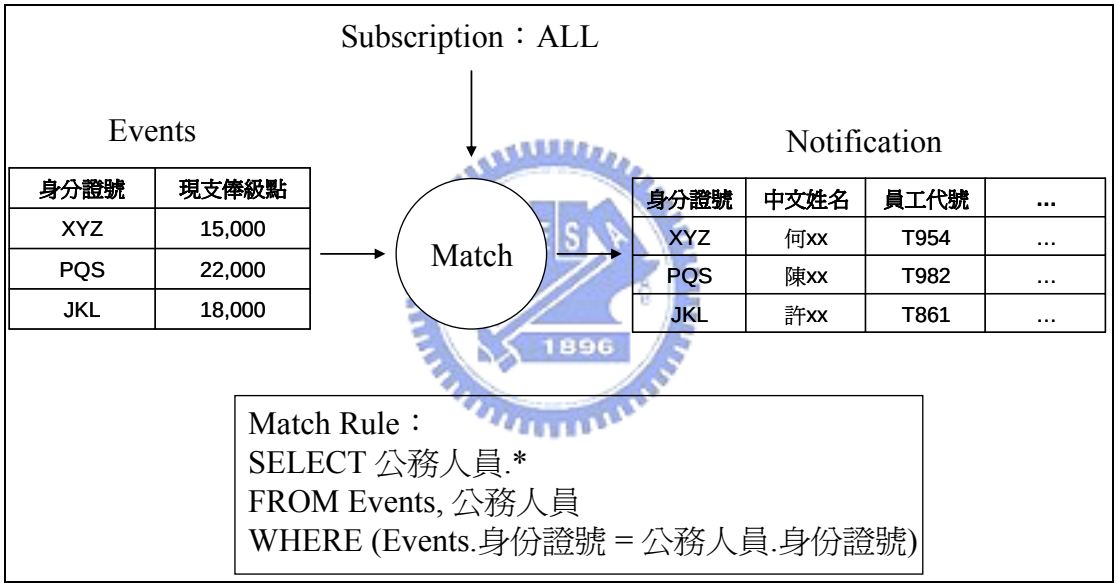


圖 39：教職員升遷案例—Match

資料庫事件接收模組將 XML 格式的 Notification 寫入 Temporary File 之中，JMS Producer 會不斷地 Polling Temporary File，所以可以立即得到 Notification 的資訊，並且以 JMS Message 格式封裝，傳送給 JMS Provider。運作過程如圖 40 所示。

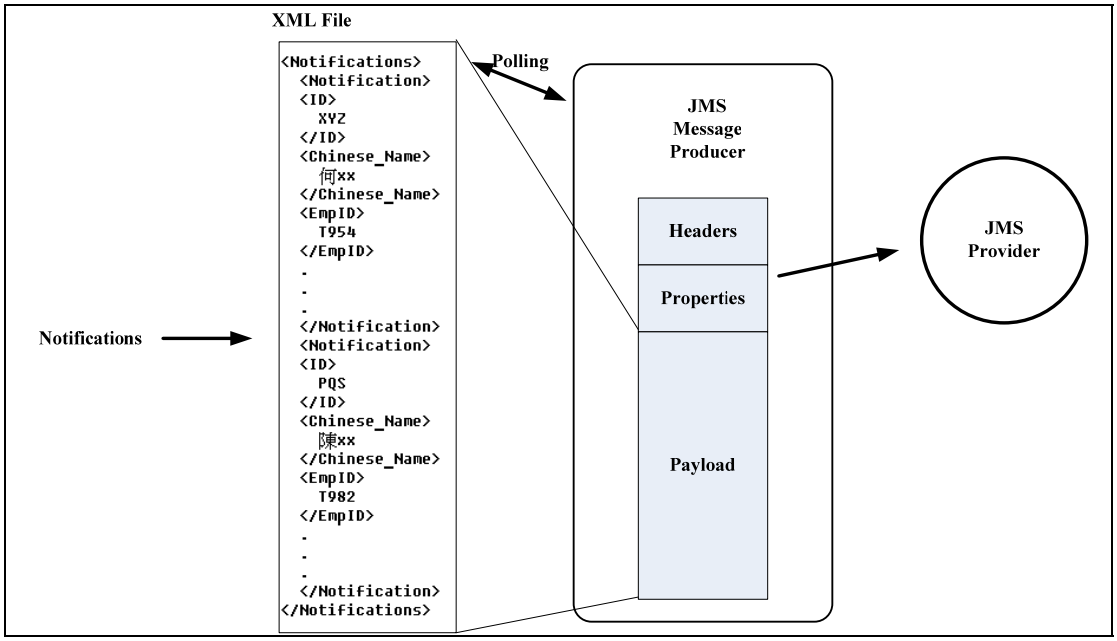


圖 40：教職員升遷案例—Polling Temporary File and Submit

當 JMS Provider 要傳送訊息之前，會先將此訊息做 filter 的動作。在 filter 元件中，有 subscriber 的相關資訊，因此可以將訊息依照 subscriber 的內容來過濾訊息，這樣的過慮動作可以節省不必要的資訊傳遞，亦可以避免有其他的資訊傳送給 subscriber，達到一定效果的資料隱密性。如圖 41 所示：

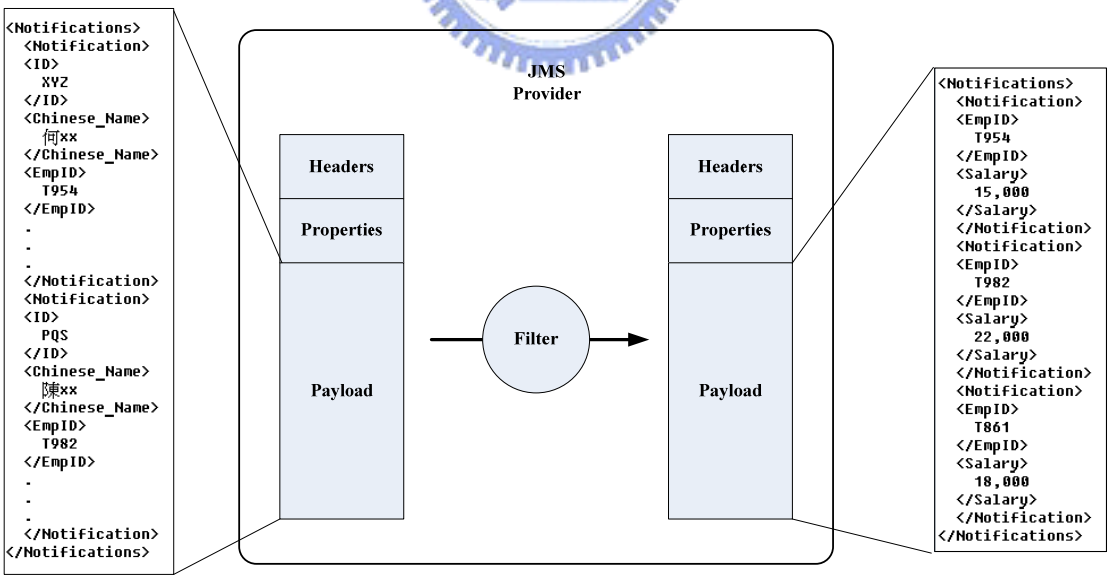


圖 41：教職員升遷案例—Filter

在 JMS Consumer 接收到訊息時，會馬上將此訊息藉由 Log Manager 儲存在 Log Database 之中的 ”新資料” 欄位。除了儲存訊息之外，也立即以 Primary key 查詢本地資料庫的相關資料，並將資料儲存在 Log Database 的 ”舊資料” 欄位，以提供日後系統資料回復的功能。

另外 Log Database 還有 ”執行狀態” 與 ”接受訊息時間” 兩個欄位，執行狀態可以記錄此筆資料更新本地資料庫的執行情況，可分爲 “已執行”、”不執行” 以及未執行” 三種。執行的決定，將依 Policy 元件的設定來執行。接收訊息時間，可提供本地系統管理者在閱覽資料時的一個參考。如圖 42 所示。

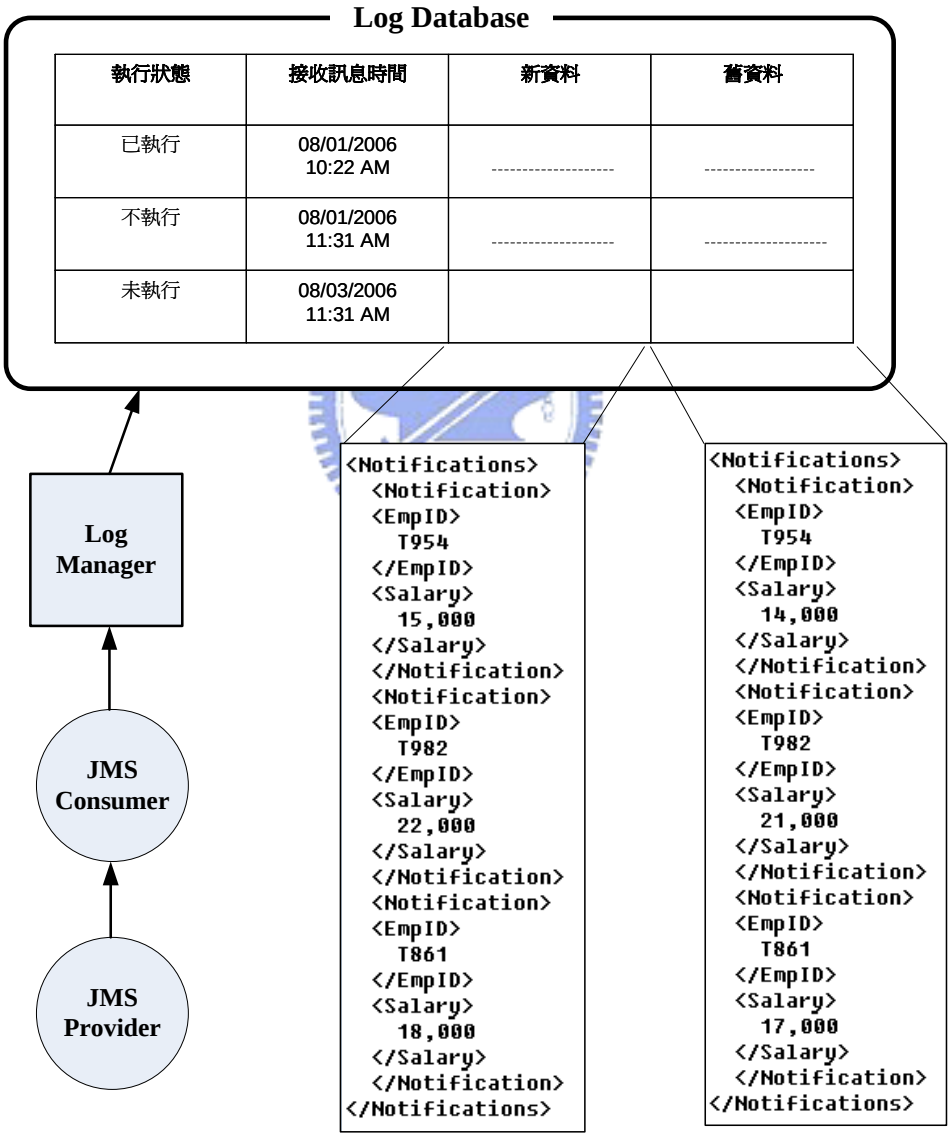


圖 42：教職員升遷案例—Log Manager and Log Database

最後則是介紹 Transform 的範例，在更新本地資料庫之前，會將新的資訊格式轉換成本地資料庫所符合的格式，職員的薪水在原本的資料庫中是以元爲單位，但是在本地資料庫中，則是以千元爲單位。如圖 43 所示。

關於各部門所維護的部門代碼不一致的問題，可以在此處獲得解決。

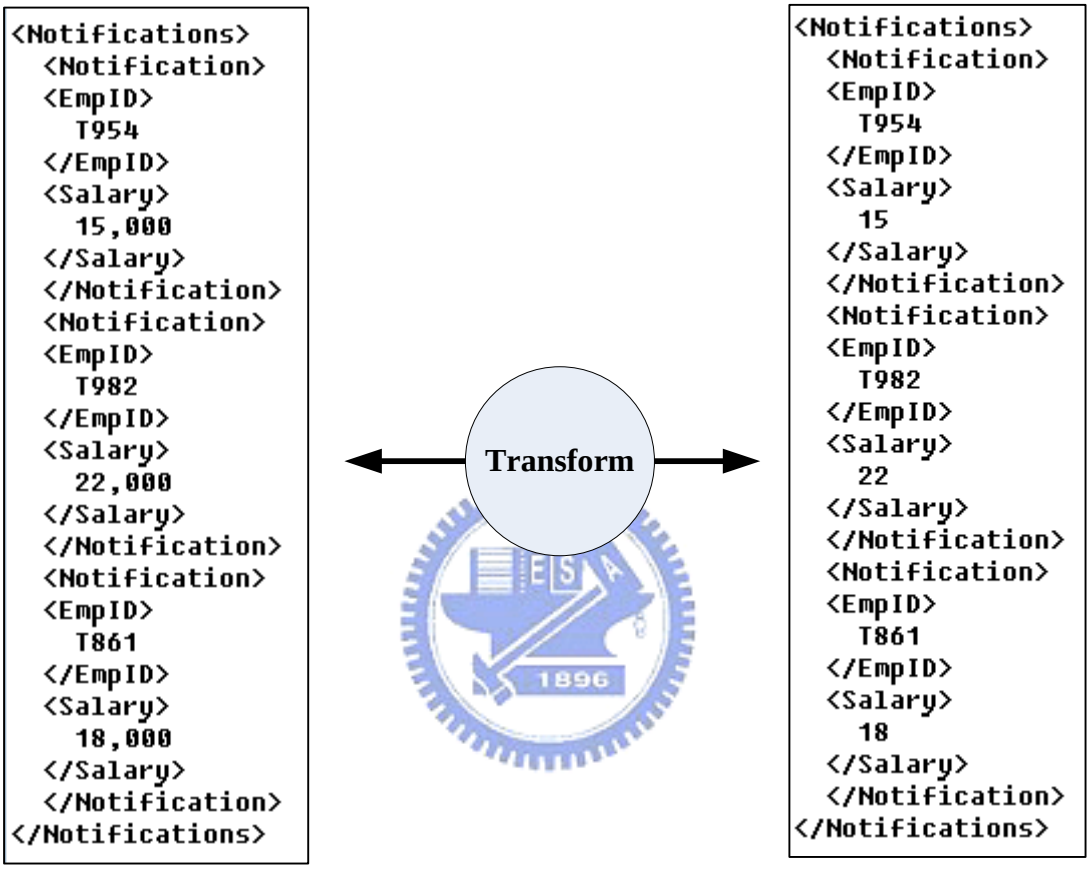


圖 43：教職員升遷案例—Transform

第5章 結論與未來工作

在此章節中，將本研究做個總結，描述本研究提出的 DEMCD 架構所能夠解決的問題，並且說明本研究以後繼續發展的方向。

5.1. 結論

學校各部門的系統，在處理獨立的業務上都可以無障礙地達成，但是在處理部門之間的業務上，運作遭遇到瓶頸。爲了解決目前學校現有系統在整合的過程中所遭遇到的問題。首先透過需求訪問的方式，深入了解校內各部門的情況，並且分析其需求，並且找出問題的癥結所在。

在本研究中，針對問題我們提出了一個架構—DEMCD。DEMCD 兩大核心技術是 JMS 與 SSNS。透過這樣的架構，可以提供各部門之間資訊交換的管道，解決部門之間因爲異質平台所造成的問題。

DEMCD 架構最初的構想就是以迅速更新資料爲主，所以利用 SSNS 所提供的功能，來產生資料庫變更事件的觸發，最後再將這些更新的資訊，以主動式的 MOM 傳遞這些資訊給所需要的部門。所以在 DEMCD 架構中，不同部門之間的系統資料，可以在最快速的時間之內，達成資料庫內容的一致性。

因爲 DEMCD 具有如此的特性，所以可以解決以下問題：

- 資料過時
  因爲 DEMCD 可以最快速的方式來達到資料的一致性。

- 異質平台，無法有效地交換資料
  DEMCD 以 Java 爲主要的撰寫語言，所以可以跨平台運作。

- 相同資料在不同資料庫中定義的欄位名稱不同
  在 DEMCD 架構中的『資料庫更新模組』中設計了資料轉換的功能，所以使用者可以自主性的設定欄位之間的對應。

- 部分資訊的交換尚未完全電子化
  只要具備執行 Java 的環境，都可以使用此架構，而且可以不受目前系統的限制，直接取得資料庫中的資料。

5.2. 未來工作

5.2.1. 部門系統描述平台

DEMCD 架構中，一開始的前置作業是很重要的，如果沒有充分了解不同部門之間的系統，很難讓其他部門的管理者有效地使用他們部門的資料，所以在未來工作中，希望能夠提供部門描述他們自己系統的平台，透過這個平台，部門系統管理者可以在上面分享管理自己系統的資訊，除此之外，亦可以查詢其他部門系統的資料。

透過這樣的一個平台，不同部門之間，可以更了解彼此有何資訊可供交流，而不必盲目的獨自開發可能已經存在的系統。

5.2.2. 系統安全性整合

部門之間在交換的資料，可能是具有機密性的，例如，人事資料、薪資…等。爲了保障資料的隱密性，以免被第三者取得，希望將來可以在架構上，加上加密的機制，使本系統架構能更加完整。

參考文獻

[1] National Chiao Tung University, http://www.nctu.edu.tw

[2] MOM, http://en.wikipedia.org/wiki/Message_Oriented_Middleware

[3] MOM, http://dsonline.computer.org/portal/site/dsonline/menuitem.9ed3d9924aeb0dcd82ccc6716bbe36ec/index.jsp?&pName=dso_level1&path=dsonline/topics/middleware&file=intro_MOM.xml&xsl=article.xsl&

[4] MOM, http://www.4k-associates.com/moma.html

[5] MOM, http://looselycoupled.com/glossary/MOM

[6] SQL Server Notification Services, Shyam Pather - Microsoft SQL Server 2005 Notification Services [book]

[7] SQL Server Notification Services, http://www.microsoft.com/sql/technologies/notification/default.mspx

[8] Requirement analysis, http://en.wikipedia.org/wiki/Requirements_analysis

[9] 文書組, http://www.ga.nctu.edu.tw/ga1/

[10] 文書組, 交通大學文書組組長戴淑欣小姐提供資訊

[11] 文書組, 交通大學計算機中心陳玉萍小姐提供資訊

[12] 圖書館, http://www.lib.nctu.edu.tw/

[13] 圖書館, 交通大學浩然圖書館副館長柯皓仁先生、簡玉菱小姐提供資訊

[14] 課務組, http://www.cc.nctu.edu.tw/%7Echcourse/

[15] 課務組, 交通大學課務組組長陳莉平小姐、黃雅坪小姐提供資訊

[16] 駐警隊, http://www.ga.nctu.edu.tw/ga8/

[17] 駐警隊, 交通大學計算機中心詹巧玲小姐提供資訊

[18] 共同資料庫, 交通大學計算機中心尤淑芬小姐提供資訊

[19] JMS, http://java.sun.com/products/jms/

[20] JMS, Richard Monson-Haefel & David A. Chappell - JAVA Message Service [book]

[21] Letting Java in on SQL Server Notifications, http://www.devx.com/Java/Article/28139