

## PowerBuilder/Database 程式碼分析

由於民國一百年將至，當初因為千禧蟲所發生的資訊大地震，似乎震波又傳回來，傳回的是民國百年年序問題，只是這個波幅小多了，當初不少系統早已利用千禧蟲的時候，一併解決民國百年年序問題。

從這些事件裡，突顯出一個問題，就是當我們需要修改某特定標的的程式碼時，如何在茫茫大海之中，搜尋這些標的？我們知道千禧年或是民國百年年序問題，所要修改的部份都與日期處理有關，如何將日期處理相關的程式碼精準的找出來，並進而修改？

由此引發的問題，就算找到要修改的部份，如何與此部分有關聯程式，確保都會一併修改？所謂牽一髮動全局，萬一漏掉有關聯的程式碼修改，可能造成所謂的蝴蝶效應，這也許是誇大其詞，但萬一發生，無疑仍是會造成傷害。

### 壹、 搜尋程式碼

怎樣找到被尋找的標的，首先要界定搜尋的範圍。我們以一個例子來說，假如員工資料表有個欄位，叫做”到職日期”，資料型態為：**char(6)**，存放的是民國年資料，例如”981201”，也就是民國年格式”yyymmdd”。假如因為百年年序問題，我們要將資料型態改成 **char(7)**，便能儲存新的民國年格式”yyyymmdd”。所以首先的問題就是要到哪裡去搜尋系統有用到”到職日期”這個欄位的地方。我們以 **PowerBuilder** 語言為例，常見的發生地，可以簡單的區分程式碼和資料庫。基本上，只要將來可能修改或是讀取的部份，就是搜尋的範圍。如下所述：

- 程式碼：
  - **PowerScript (含 embed SQL)**  
所有 **PowerScript** 內，用以暫存和處理”到職日期”資料的 **local** 變數、以及用以讀取 (**Select**、**Insert**、**Update**、**Delete**)資料庫的 **embed SQL** 命令或 **Cursor** 命令。這些變數和命令，都可能會包含”到職日期”資料。
  - **各種 DataWindow 物件**  
**DataWindow** 物件，後端對應到的是資料庫欄位。所以以對應到”到職日期”欄位為例，假如報表內有顯示”到職日期”，或是藉由此 **DataWindow**，新增一筆員工資料到資料庫，這些都會牽涉到”到職日期”，則可能需要修改

- **Structure**

結構的設計用來存放資料或傳遞參數。假如有存放”到職日期”資料，則不管是存放資料到此結構、或到此結構讀取資料者，都可能需要修改。
- **Menu**

Menu 內主要常用的是 **Clicked Event** 和 **Selected Event**，又由於 menu 一定是搭配 Window 物件使用，因此在這些事件，若有使用到”到職日期”資料，或是有和 Window 物件有”到職日期”資料的互相傳遞，則這些都是要修改的部份。
- **User object**

使用者物件可能搭配在應用程式的很多地方，因此很有可能和其他物件會有資料的傳遞及處理。假如這些資料存取”到職日期”，就必須要修改。
- **Function(global、Instance)**

函數的部份包括函數的傳遞參數，以及函數本身的程式碼。只要這兩個部份有包含”到職日期”資料，就需要修改。
- **Variable(global、instance、shared、local)**

變數主要用來暫存資料。也就是說，假如暫存資料的是”到職日期”，則與此變數有關的地方，不管讀寫，都可能會是修改的地方。
- **source of PowerScript objects**

各種 PowerBuilder 物件的 source，不限 PowerScript。例如 DataWindow 物件本身，在 Display Format、Edit Style 和 Validation Rule，就可能與”到職日期”有關。或是 Window 物件上的 editmask 控制項，也可能有關。這些無法顯示在物件的程式碼內(因為沒有程式碼)，而是存在物件的屬性，因此只能藉由 source，得知是否與”到職日期”可能有關。
- **資料庫：**
  - **Stored Procedure**

凡是使用到”到職日期”欄位資料的，都會是要修改的地方
  - **Table Column**

這是最直接的修改，因為就是存放”到職日期”欄位。
  - **Trigger**

凡是使用到”到職日期”欄位資料的，都可能會是要修改的地方

- **Function**  
凡是使用到”到職日期”欄位資料的，都可能會是要修改的地方
- **Cursor**  
凡是使用到”到職日期”欄位資料的，都可能會是要修改的地方
- **View**  
凡是使用到”到職日期”欄位資料的，都可能會是要修改的地方

從以上可以看出，要檢查的範圍還真不小，幾乎就是所有應用系統範圍，而這僅僅只是一個”到職日期”而已。假如你需要找的標的更甚於”到職日期”，這些標的加起來，系統可以說被地毯似的掃過一遍。

## 貳、 衝突分析(Impact Analysis)

搜尋到標的後，接下來就是程式修改。此時重點不在於修改內容本身，而是該標的修改後，是否與其他物件有關聯？假如有關聯，都必須修改，確保一致性。所以這階段最重要的問題在於，你如何得知與此標的有關聯的其他物件？你需要做的就是”衝突分析”。”衝突分析”可以讓你知道不同物件之間的關連，例如函數的呼叫關係(Calling)、變數的使用關係(Using)、或是 DataWindow 控制項的引用關係(referencing)...等等。衝突分析也是一種雙向多維分析，你不單可以知道物件 A 使用了哪些物件，也可以藉此知道有哪些物件使用了 A 物件，這是一種多對多的雙向關聯分析。

假如你找到一個global function需要修改，其中的參數會傳入”到職日期”資料，此global function回傳值是”調薪日期”資料，同樣的char(6)、”yymmdd”格式。你會針對此函數，將日期資料調整成char(7)新的格式。接下來，你就必須做衝突分析，以便知道在這個系統裡面，有哪些地方有呼叫到此global function，然後一併修改。

另外，在PowerBuilder程式內，有不少程式是牽涉到資料庫物件，最常見的就是DataWindow物件，其他也包括Query物件和embed SQL，這些和程式都是緊密結合在一起，在做衝突分析的時候，尤其要被分析到，才不會程式和資料庫分離，雙頭馬車，各自發展。

不僅如此，Table的欄位定義、stored procedure等資料庫物件也都是會碰到相同的問題，都需要一併修改，都需要做衝突分析。衝突分析是一個很大的挑戰，這裡提到的僅僅是以百年年序問題為例，但是在一般程式開發過程，這樣的修改更是普遍。系統在分析和設計階段，原本就是一直在修改，例如資料表格的改變、資料欄位的改變、函數簽名(Function Signature)的改變、程式繼承的關係、結構(structure)的改變...等等，這都是一直在進行著；萬一修改的部份是共同

元件，影響將會更巨大，你必須告知所有程式設計師，必須配合修改，以確保一致性。試想，對於這些改變，程式設計師唯一能做的，就是自己的大腦，自己做衝突分析，自己去修改其他關聯的物件，以確保一致性。可是，一旦人員異動呢？亦或是大腦忘記一項重要的修改呢？

## 參、 程式架構

在程式撰寫的過程中，從雛型到完成。衝突分析可讓程式設計師了解局部的架構，便於特定問題的程式撰寫。但是假如對於整體架構有清楚的輪廓，對於開發的效率和品質，將會有很大的正面影響。筆者認為，所有專案可以改善的地方，都是看不到的地方。對於系統整體架構能夠有清楚的了解與撰寫，大家用同樣的語言解讀系統架構，對系統建立過程是有正向的結果，這也是一種看不到的地方，也是促進專案成功的因素之一。

程式設計師能夠了解整體架構，可加速程式開發速度，減低程式出錯數目，增進品質，減低將來維護的次數和時間，這些都是一連串的連鎖反應，所降低的隱藏成本非常可觀。對於 PowerBuilder 而言，這些架構的了解，最常使用的功能就是 **Browser Painter**，裡面可以查看應用系統所包含的所有物件，此物件的屬性、函數和事件，以及物件的繼承關係等等。**Browser Painter** 是一個很好的功能，提供各種物件的資訊，也能夠使用 **Help** 即時查詢你想要的資訊。

但是 **Browser Painter** 尚有美中不足的地方，就是無法結合衝突分析。**Browser Painter** 主要針對單一物件，與其他物件的關聯目前只有繼承關係，但是假如加上分析不同物件之間的關連(如前述，包括 **Calling**、**Using** 和 **Referencing**)讓程式設計師不但知道單一物件在系統內的位置，也可知道整體系統的框架，若把這樣的關聯再以文件製作出來，甚至放在網站上，提供即時資訊的分享，對於專案開發的程式撰寫流程，以及進入專案後續的維護階段，都一直會有很好的協助效果。

## 肆、 資料庫物件衝突分析

程式碼可以做這樣的分析和搜尋，同樣的，資料庫應該也要提供如此的功能，這對於大量將運算核心放在資料庫伺服器的企業，會非常有幫助。資料庫運算核心包括 **Stored Procedure** 和 **Trigger**，這些物件常見的問題就是修改。要修改這些物件，因為資料庫並沒有針對這類物件做註解，頂多是本身內部的註解，所以企業通常會額外製作文件，用以說明這些 **Stored Procedure/Trigger** 的目的和功能。這些是一般的作法，但是問題在於該物件和其他物件的關係，很難用文件說明出來。例如，當你修改 **Stored Procedure A**(簡稱 **SP A**)，你頂多紀錄該 **SP A** 會呼叫 **SP B** 和 **SP C**，但是，你可否知道，有哪些 **Stored Procedure** 有呼叫 **SP A**？

由於資料庫沒有這樣的機制，筆者經常看到有不少企業撰寫了上千個 **Stored Procedure**，

也有很厚的使用手冊，但是同樣的，裡面也存放了 **unused Stored Procedure**，成為潛在的 **Garbage**，因為這些 **Stored Procedure** 手冊找不到，又看不出誰使用了它，不敢亂刪。

## 伍、 程式碼分析解決方案

要解決這類的問題，倍力基於處理民國百年年序的問題，已經累積不少的經驗，也已提供相關的解決方案，對於想要快速搜尋問題點，又可避免程式修改遺漏，已經有萬全的準備。

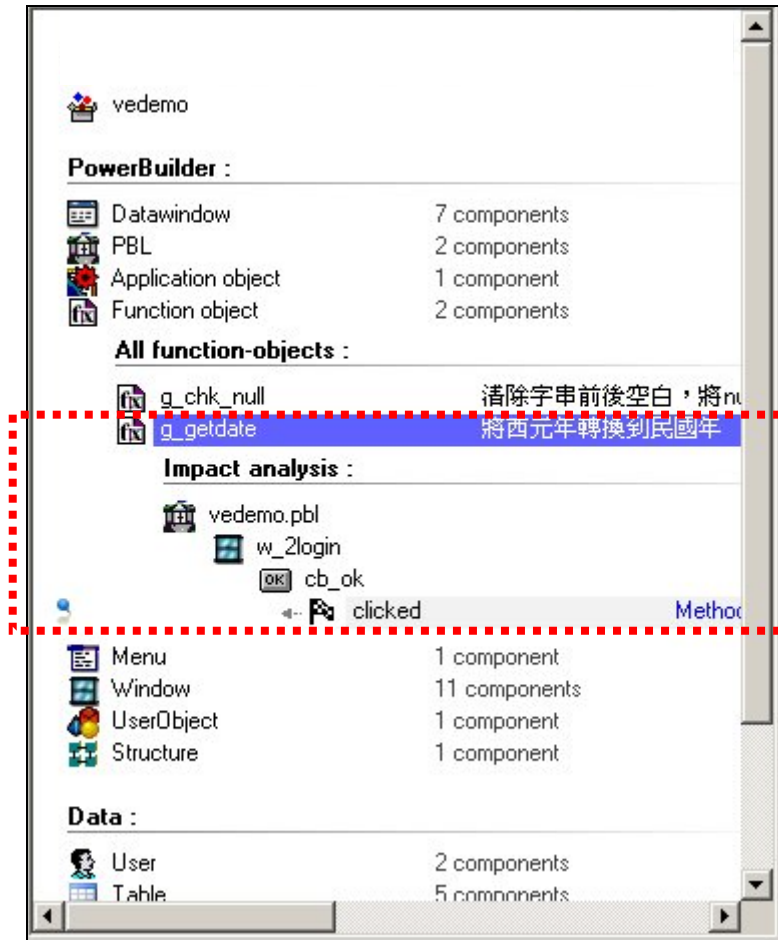
此方案功能眾多，除了前述搜尋標的、衝擊分析、呈現系統架構和資料庫物件分析外，還可將這些結果，製作成文件，變成知識庫。這有利於新人快速進入專案，也讓人員異動離開時，保留知識。另外，包括 **Dead Code** 的功能，可以將沒有使用到的變數、參數和 **Function** 等，全部列出來，供你檢視，這有降低將來維護上的困擾。

茲將此解決方案，在此節錄出來，希望讀者能快速了解其功能。

- **資料庫分析**：可針對資料庫物件做關聯性分析，包括：
  - 分析資料庫物件內的關連，提供增修物件的參考
  - 資料庫物件和 **PowerBuilder** 程式的關連，保持資料庫和程式的一制性。
  
- **衝擊分析 (Impact Analysis)**：只要一個動作，就可以幫您列出所有與該程式有關的元件，並為您智慧分析每個改變可能帶來的衝擊。
  - 在數分鐘內就可以幫您完整分析您的應用程式。
  - 您可以隨意選擇應用程式的任何項目來分析。
  - 可以把您所選項目有關的所有物件，完整列出所有元素。

參考畫面：

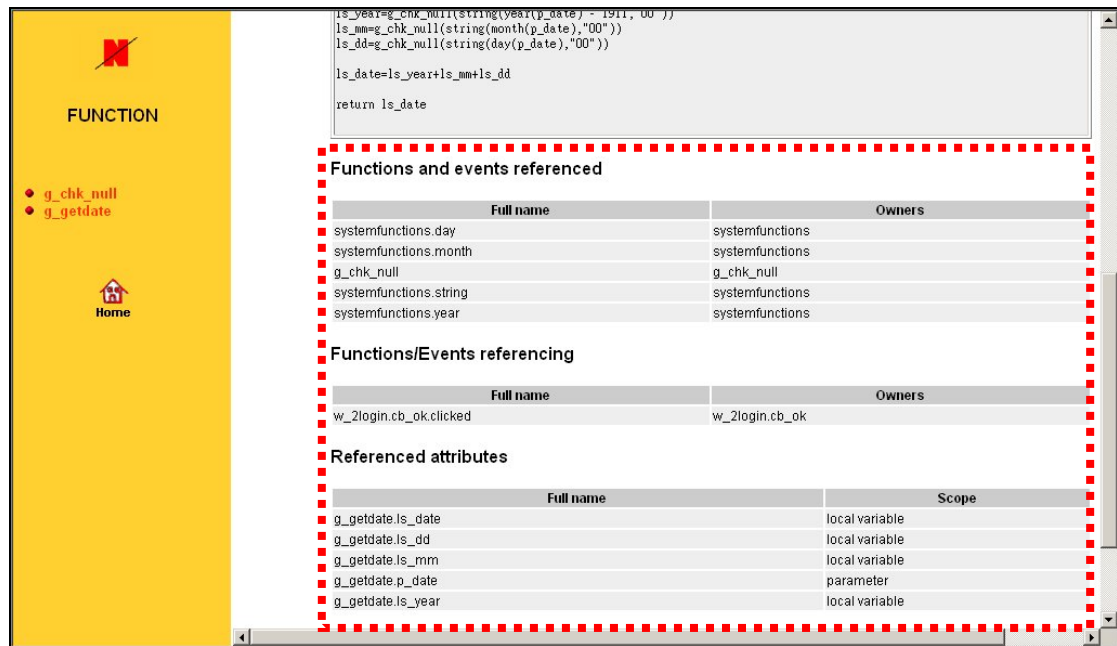
針對函數 `g_getdate()` 做衝突分析的結果



圖一、g\_getdate()函數的衝突分析

- **程式碼自動文件化(Source Code Documentation)**：可以在短短幾分鐘內幫您自動產生文件，並且隨時更新，讓您不再苦於文件製作，可以花更多心思在更重要的事情上。
  - 多達 30 種以上的範本選擇
  - 一些範本可產生 Word 或 HTML 格式的程式碼文件。
  - 一些範本可以幫您檢視程式碼。
  - 提供精靈幫您量身訂做您要的文件結果

參考畫面：g\_getdate()函數資料，其中有顯示呼叫/被呼叫的關連



圖二：文件範例

- **檢視您的應用程式 (Browse through your application)**：提供 Navigation bar、Source Browser 以及 Previews 等功能，可以幫您輕易地檢視您的程式碼。

可幫助您：

- 呈現應用程式的架構與其他元件的關聯。
- 發覺繼承關係並且掌握程式框架。
- 事先預覽程式視窗，以便視覺化其呈現之面貌且便於測試。

- **改善程式碼品質(Code Review)**：在開發的每個階段都緊密協助您：
  - 開發階段前：可以幫您定義開發規則。(您也可以使用預設規則)
  - 開發階段中：可以幫您執行這些規則並且修正錯誤。
  - 開發階段後：可以幫您在整個應用程式中搜尋出一些沒有符合規則、功能有限的問題元件。
- **偵測多餘的程式碼 (Dead Code)**：Dead Code 指的就是一些從來沒有被呼叫或執行過的功能、參數、變數等。把這些 dead code 找出來並從系統中移除，對系統效能將有顯著的幫助，而且可以讓維護工作變得更容易。

若還想更進一步了解此解決方案，可進一步與蔽公司聯絡。

台北市大安區仁愛路四段 85 號 12 樓 B Tel: +886-2-2731-6868  
[pmd@mpinfo.com.tw](mailto:pmd@mpinfo.com.tw)

筆者認為，當程式在撰寫的時候，不管是在開發階段，或是維護階段，會有不同原因需要修改，此時常常需要參考其他物件，一併修改。若有一個功能，可以隨時協助你應該參考的物件，你不用再單獨靠自己的記憶力和經驗來找到其他物件，或是花費時間，一個一個瀏覽物件來修改，這個功能的價值將會非常重要。試想，假如你維護他人所寫的系統，你根本不知道修改該功能所造成的影響，很多是要到 **Compiler** 之後，你才知道錯誤發生在哪裡，這往往又會花費許多寶貴時間了。

## 結論

其實程式檢視是一個容易為人忽略的問題，但這是程式設計師天天在做的事情。對於以專案開發為主的團隊，這類的工具其實非常有幫助，對於專案的品質、成本和時間，都有一定的協助。據統計，若對程式檢視可以提供完整查詢和分析，這可以節省 30% 以上的開發和維護成本，而這也是筆者希望帶給讀者這個解決方案最主要的目的。