db/LIB
資料庫管理程式庫

■適用編譯型式的BASIC版本
■讓您用BASIC設計類似dBASE
的資料庫管理系統

張其邦　編譯

目 錄

如何使用本手册

如果您對資料庫（database）或檔案管理（file management）的理論並不熟悉，那麼我們建議您仔細地閱讀「資料庫管理的簡介」以及「編製索引的簡介」這兩節。這兩節內會告訴您一些關於資料庫管理所要做的事情，以及將db/LIB程式庫當作一個應用系統發展工具所能完成工作的一個基本概念。

即使您對檔案管理有一些經驗，也許會需要參看「如何使用 db／LIB程式庫」來瞭解這個程式對程式發展有多少幫助，以及程式需要去執行那一些工作，使得這個程式庫的功能得以完全發揮。

「db/LIB程式庫的功能」這一節內包含了在此程式庫內，每一個程序確實的用途及語法。我們希望您在建立及修正第一個資料庫應用系統時，能夠經常參考這一節。

在「資料庫管理的基本概念」這一節中討論的是，如何將db/LIB程式庫的呼叫組合成可以用來建立眞正資料庫管理應用系統的方法。本節中包括如何在典型的資料庫管理應用系統中從事基本工作的建議，這使得您不必再花費許多時間來“重新發展”許多標準的程序。

「安裝、編譯及連結」一節中想要提供的是，在不同的選擇項下將程式庫連結所需的語法。請注意這一節會因爲不同的Basic 編譯程式所提供的Basic 版本而有所不同。

「技術資料」一節包含資料緩衝區如何記錄讀及寫的作業、程式庫在記憶體存放的位置、資料庫檔案的技術極限，以及其他主題的細節。這些資料十分

是提供給專家使用的，對於正常情況下程式庫的使用者而言是不需要的。

當您遇到一個由程式庫或 DOS 傳回的錯誤訊息時，請參考「錯誤訊息」這一節。其中包含的資料應該可以幫助您決定該採取什麼動作。

請參考「詞彙」這一節以瞭解一些在資料庫管理、程式庫、Basic或電腦所使用的術語之意義。

「程式庫參考速見」這一節內包含有每一個程式庫呼叫的基本語法，使您能夠很快地查看所有需求的引數。

資料庫管理的簡介

注意：若您對資料庫管理應用系統的設計及執行相當陌生，那麼這一節將會介紹如何從事這些工作所需的資訊。若您對資料庫發展有相當的認知，那麼可以跳過這一節，而不致失去操作 db/LIB程式庫所需的重要知識。

簡　介

在今日微電腦軟體發展中最重要而且最具生產力的領域，就是資料庫的管理以及應用系統的發展。這個程式庫的目的就是要幫助您在使用Basic 語言，來建立應用程式時所需的資料庫管理技巧。

一個資料庫就是一個已儲存資料的組合，可以經由一種有組織的作業來處理使用。在現今的安排情況下，db/LIB程式庫提供了資料的處理能力，而Basic 程式則提供有組織的作業。

何謂資料庫

通常資料庫這個術語所指的，就是一個單一而存有資料的 DOS 檔案（例如，DBF檔）。正確地說，一個資料庫是所有檔案的集合，檔案內存著所有依照某一保存記錄原則而儲存的資料，它們組合而成爲一個資料庫。

一個簡單的資料庫例子就是一個裝滿了檔案夾的檔案櫃，所有的檔案夾都

表示同一類的資料 —— 而且儲存在內的資料也都是同一種格式。

資料庫管理系統

一個資料庫管理系統是整組的程式，而且不是管理在資料庫內的資料之電腦作業。Basic 程式就是DBMS的主要部份，它的主要責任是管理使用者介面及維護所有有關驗證、解釋，以及將資料輸出做爲有用資料的內部意義及關係。

關聯式理論(Relational theory)

在設計及執行電腦檔案的資料庫管理時，有許多種的方法；但在 db/LIB 程式庫中所採用的是關聯式的模式。有許多術語在關聯式資料庫管理中是相當常見的，我們在書中經常會使用，相信其中許多都是您所熟悉的。

資料庫檔案

一個資料庫檔案就是一個具有相同格式（或結構）的資料記錄之組合，而且它們之間都有一些相關性；例如，都是客戶。一個資料庫檔案可以有許多筆記錄、少許記錄，或根本沒有記錄。所有的資料庫檔案都有一個 DOS 型式的名稱及 .DBF 的附加名（extension）。

記　錄

一筆記錄（record）在一個資料庫檔案內，通常是代表人、事或物的輸入。在關聯式資料庫內，所有記錄的大小及格式都是相同的，也就是說，每一筆記錄內的資料格式都是相同的。例如：名稱、地址、城市、州、郵遞區號、電話。

所有的記錄都由資料庫指定一個記錄號碼，經由這個記錄號碼就可以找到這筆記錄。然而，記錄號碼並不與資料一起儲存，而且也不該被視爲與某一特

定的記錄資料永久連結在一起。

欄 位

一個欄位（field）是一筆記錄內資料的一部份。欄位裡的資料永遠存在記錄中相同的相關位置內，而且對無一筆記錄而言，它具有相同的字元數目。一個字元是一個可以輸入到欄位內的字母、數字或符號。

到目前爲止，我們可以用以下的例子做爲一個資料庫檔案：

```
FIELD =>      NAME       ADDRESS      PHONE
Recno# 1   <name data> <address data> <phone data>
Recno# 2   <name data> <address data> <phone data>
Recno# 3   <name data> <address data> <phone data>
```

結構及表頭

我們可以建立許多具有任何數目、安排過的欄位及記錄的不同資料庫檔案，並且依據我們的理想對所有的檔案工作使用相同的程式。但是由於電腦本身無法在人名及地址間分辨出它們的不同，因此，我們需要建立一個排好次序的道路名稱對照圖，來告訴程式如何在欄位及記錄內找到特定的資料。

欄位及其長度的整體安排就稱爲結構（structure）。一個結構永遠在檔案建立時，並且在資料輸入前定義。關於一個資料庫檔案的結構資料是存在檔案的表頭（header）內。表頭的實際管理是由程式庫自動處理的。程式庫內具有可以傳回表頭內資料的常式。

關聯式模式的特性

關聯式資料庫模式具有某種希望達成的意義及目標，其中有許多是關於記錄儲存的方式，以及如何設計應用系統來管理記錄方面的。其中影響到關聯式資料庫模式的基本原則是：

相關於任何資料庫內部結構的資料，都應該存在資料庫檔案本身，並且不能儲存在用來管理它的程式內。也就是說，資料永遠是與程式分開來使用及維護的。

資料庫檔案會爲每一筆定義的記錄配置空間，而不論其中是否存有資料。也就是說，會有許多的空白" "存在檔案內。

所有的記錄都可由記錄號碼而找到，而此號碼通常都經由一個索引檔（index file）內取得。在磁碟上的一筆記錄之實際位置，是將每一筆記錄的長度乘上記錄號碼，再加上表頭的長度而找到的。因此使用一個程式來處理記錄時，記錄號碼是相當重要的，但不要將記錄號碼當做是永久，也不要將它視爲是實際的資料。

在關聯式模式中所有的欄位資料都是由欄位名稱找到，而不是經由它們的絕對位置。程式庫使用欄位名稱及記錄號碼來取得需求的資料。

在設計一個關聯式資料庫時，我們通常儘可能地要求除了被其它檔案用來當做資料鍵的欄位外，資料庫中不應該有任何重複的資料，這樣做可以讓我們省卻更新記錄副本的種種麻煩。這個目標在實務應用上並不容易達成，但一般來說，在一個資料庫內只要資料能夠在一個已知的位置上迅速地被查詢到，就不應該製作其副本。

應用程式的發展

除了與程式庫在一起作爲範例的程式之外，程式發展的工作是身爲一個使用Basic 應用程式庫的程式設計師，所必須做的工作。若針對利用資料庫管理技術的應用系統而言，應用系統程式的功能可以分爲四個主要的類別。將這四個工作組合應用是使用者程式的責任。

「資料輸入」是建立一個資料庫檔案的基本工作，接收由使用者輸入的資料，將它儲存在適當的檔案內，並且也許再輸入一個索引鍵以幫助將來資料的擷取。

「資料辨別」是輸入資料的應用系統規則，確定這個資料是合理的、正確的、適當的格式化、依據一定的次序接受，或對目前資料庫而言是合適的。可

以在使用者輸入之後立刻執行辨別的工作，或在批次處理後執行，或以合併的方式進行。通常它不能在輸出資料時才執行。

「資料處理」是將資料修正、更新、計算、移動、暫時的複製及備份，以及選取，而使資料較易使用或較正確。

「資料輸出」是選取資料記錄，將它們設定好格式，並且寫出到另一個可供使用的格式裡，通常是報表格式、另一個檔案、或電腦螢幕的處理。有一些記錄的選取及非永久性的修改，會在輸出的時候完成。

設計一個資料庫

在建立一個新的資料庫檔案之前，最好先想想這個資料庫要如何使用，其中存些什麼，以及不需要存些什麼。

一般說來，應該進入一個資料庫檔案內的就是具有同樣特性的記錄，而且具有某一特定的格式。電話目錄、庫存，以及雨量的讀數都是很好的範例。但是一首詩、一隻狗的故事或是一個電腦程式的參考手冊，都不能有效地由BDMS來管理。

要決定在特性方面的相同點並不如想像中的容易。例如，我們可以建立一個名爲“PEOPLE.DBF”的資料庫檔案。這個檔案可能包含著名稱、地址，以及每一個人都知道的資料。我們也希望在每一筆記錄內擁有一個欄位，用來記錄我們與這些人之間的關係，“C”表示玩橋牌的人，“R”表示親戚，“B”表示生意上的往來朋友。我們可以用這些代碼來“過濾”出所要找的人。

這樣的安排對於我們個人的檔案系統也許相當不錯，但若我們的生意逐漸擴展，將發現我們必須追踪類似金額：到期日及交貨：地址之類的問題。修改欄位結構可以增加這些欄位的使用空間，但整體來講，我們會發現用這種型式的檔案來維護朋友資料並不恰當。

因此我們可能會將所有的商業記錄，複製到稱爲CUSTOMERS或VENDORS的新檔案內，而加以修改這個資料庫，然後再將原先的檔案重新命名爲FRIENDS。若我們做的正確，可能仍然可以使用原本設計來處理原先檔案的許多相同程式。

在一個關聯式資料庫內的記錄選取及結構，主要是依據它們的目的、誰要使用這些資料、可用的磁碟空間、處理機速度，以及資料本身的特性而決定的。

資料的驗證

在一個資料庫檔案內，所有使用者的資料都是以ASCII字元的格式儲存的。但有許多的工作需要資料以其他的格式來表示。每一個已定義的欄位型式之特性及其限制，並不由程式庫目錄所維護，而需要在應用系統程式中特別注意。

我們建議您發展經常要使用在程式中，用來驗證輸入到資料庫內的資料，並且可在使用者發生錯誤時，能夠採取適當反應的常式。在定義檔案的欄位型式時，當然是選取驗證常式來處理由使用者輸入資料的基礎。

特別是，數值資料型式“N”，應該爲要計算用的資料所保留。相對的，一個郵遞區號則應該給予字元欄位“C”，因爲它永遠不可能用來計算。雖然一個邏輯欄位“L”只是以一個字元的型式儲存，但使用者的程式也要驗證這個只能由兩個符號所組成的資料，而它們的意義永遠是相反的。日期欄位“D”應該驗證日期是否有意義，而且是否依照正確的格式儲存以供日後的比較使用。

相關的多重檔案

如同我們見到的一般，我們可以將具有相類似結構的資料，存在相同或不同的檔案內。但對於具有緊密關係而性質卻不同的資料，像是客戶以及他們所購買的商品，又該怎麼辦呢？這就是關聯式資料庫模式最有用的地方了。

我們會維護一個稱爲ORDERS的明細檔案，其中含有日期、金額、以及每一筆訂單的資料，但其中會包含曾經下過訂單的每一位客戶的名稱、地址、交貨資料、以及他們所訂購貨品的說明、數目、價格。

這個檔案具有將訂單所有資料放在同一個地方的好處，但卻有資料重複（一般的客戶以及我們所供應的貨品資料）的缺點。更糟的是，我們可能會發現每一次所輸入的重複資料都會有少許的不同。

例如，“Smith Eng. Co.”可能是代表著“Smith Engineering Corporation”，或是“Smith Engine Company”。電腦將無法找到由Smith公司所下的訂單，或者將所有具有Smith名稱的6個客戶完全視爲一樣。

由互相關聯的觀點來看，正確進行的方式是要維護三個 DBF 檔案，並且將它們之間的關聯性建立起來。我們使用選取一個“鍵”欄位的方式來完成這個工作，使我們可以到其他檔案去查看記錄，以找到所需的其他資料。

我們可把檔案建立成下面的方式：

CUSTOMERS	ORDERS	INVENTORY
C1 Smith Corp....	# 1000021, C2, P1058	P1053 Drill ....
C3 Jones	# 1000022, C7, P1053	P1020 Saw, 5" ....
C2 Smith Eng	# 1000023, C7, P1026	P1011 Saw, 8" ....
C7 Allen	# 1000024, C3, P1026	P1058 Cord ....
	# 1000025, C2, P1011	P1026 Pump ....
	# 1000026, C1, P1053	

以最後一筆訂單做爲例子，（#1000026），我們並不需要將客戶名稱作儲存（以及其他固定的資料，如“....”），而是建立一個單一的客戶號碼，“C1”。當我們需要所有的資料時，可以在客戶檔案中查看。同樣地，零件號碼，“P1053”，指向在庫存檔案內的一個單一的零件。

當要發出每一筆訂單時，我們可查看客戶清單，選取正確的客戶，只輸入客戶號碼“C2”，然後程式就會去查看其他所有的資料（包含名稱、地址、信用條件等等），並且列印出貨單，但只把客戶號碼與訂單記錄存在一起。

同樣的情況對庫存也是有效的，我們應該只把零件號碼與訂單記錄存在一起，並且使這個應用程式去查看成本、價格、說明，以及可資利用的各種情況。

這種方法的最大好處就是可以避免重複的說明資料，以便我們可以使用已經存在電腦內的資料，而不必每次去輸入它們，並且可以避免因爲重複輸入而造成輸入的錯誤。藉著使用單一的客戶及零件號碼（“鍵”），我們就可以得到組合所有客戶訂單的報表，而且即使它們具有相似的名稱也不會造成任何混淆。

另一個好處是，我們現在有了獨立的庫存及客戶檔案，它們可以經由自己的軟體程式來進行維護的工作。

在程式庫中的兩個檔案之間所產生的關係，通常有以下幾種（這兩種檔案

一般稱爲“主檔”及“查看檔”）：

一對一 -- 對於在主資料庫檔案內的每一筆記錄而言，它是獨一的，而且在查看檔案中也只有一筆相關的記錄。

一對多 -- 對每一筆在主資料庫檔案內的記錄而言，在查看檔內有一筆或許多筆相關的記錄。

多對一 -- 在主資料庫檔案內的許多筆記錄，共用在查看檔內的一筆記錄。

在我們的範例中，客戶及訂單間的關係是一對多；而在訂單及庫存間的關係則是多對一。在範例中，客戶及庫存之間並沒有任何定義的關係。

編製索引的簡介

簡　介

編製索引（indexing）的意思就是，我們可以藉由它在一個資料庫檔案中，依照一定的順序查看記錄，將記錄依照邏輯分類的方式組合起來供以後使用，並且使搜尋某一筆記錄的速度加快。

使用索引意味著，記錄是如何新增或存放在資料庫內並不重要，索引可以用任何順序來表示。db/LIB的索引系統可以建立能夠將記錄依照任何順序排列的任何數目之索引檔。

索引循序檔（index sequential file）的一般理論就是，我們用儲存在磁碟位置或記憶體位置內的索引所佔用之空間，來交換資料庫應用程式整體執行上的改進。

索引檔

索引檔就是一個爲單一資料庫檔案所建立的獨特 DOS 檔。索引檔中儲存著每一筆資料記錄內的一小部份資料，也就是鍵（key）以及相關記錄的記錄號碼。因此，對在資料庫內的每一筆記錄而言，在索引檔內都有一個“鍵”輸入口（亦即，我們可以依照鍵值找到相對應的記錄）。索引檔內存有一個表頭，