TR-81-005

萬用資料輸入系統之設計

本研究報告部分爲國家科學委員會贊助之研究計劃
「散佈資料基系統的設計和建立」 NSC70-0404-E001-01

內容

萬用資料輸入系統之設計

林慶良　柯志昇　洪永常
陳守文　王秋鳳

中央研究院資訊科學研究所

摘　要

本篇論文介紹中央研究院資訊所研製的資料輸入系統—GPDES

GPDES包括有資料描述語言，語言轉譯器，表檔解譯器，畫面描述語言諸部份。

介面程式將 GPDES 與關係資料庫系統—RDBMS連線； RDBMS 有完整的資料操作語言及詢取能力。

GPDES設計重點强調系統的多用途性，操作簡易性，資料文書可讀性，及獨立性。

GPDES與 RDBMS已實際的應用在中央研究院資訊所圖書資料的電腦化作業上。

一、緒言

中央研究院資訊科學研究所於四年前成立籌備處之初，首先面對的問題是圖書館的建立。爲了將來便於管理，我們決定利用計算機來管理圖書資料。尤其是，初期只有幾百本圖書，計算機化較容易進行。

一九八〇年八月，我們開始著手籌劃及訓練事宜。經過詳細的研究考慮，決定利用 PDP11/70小型計算機配上UNIX系統。UNIX系統裡，有C-語言可以用來編寫必要的程式。因此，我們的第一步工作就是學習UNIX系統和C-語言。一個月後，參加此工作計劃的同仁都已能夠操作UNIX系統，並利用C-語言編寫程式。同一時間，我們著手規劃整個圖書館系統的軟體。

圖書館系統的軟體，主要分爲三部份：一是資料輸入，二是資料庫，三是資訊詢取。一年來，前兩部份大致已完成，第三部份的工作正在進行中。

在資料輸入方面，我們訂下的第一個原則就是：資料輸入的程式必須具有通用性；在資料庫方面，決定採用關係模式資料庫。一九八〇年九月，我們開始研習資料庫系統的設計，採用自修和研討並進的方式，由同仁們輪流講解有關資料庫方面的文章。同年年底，完成訓練，準備工作。

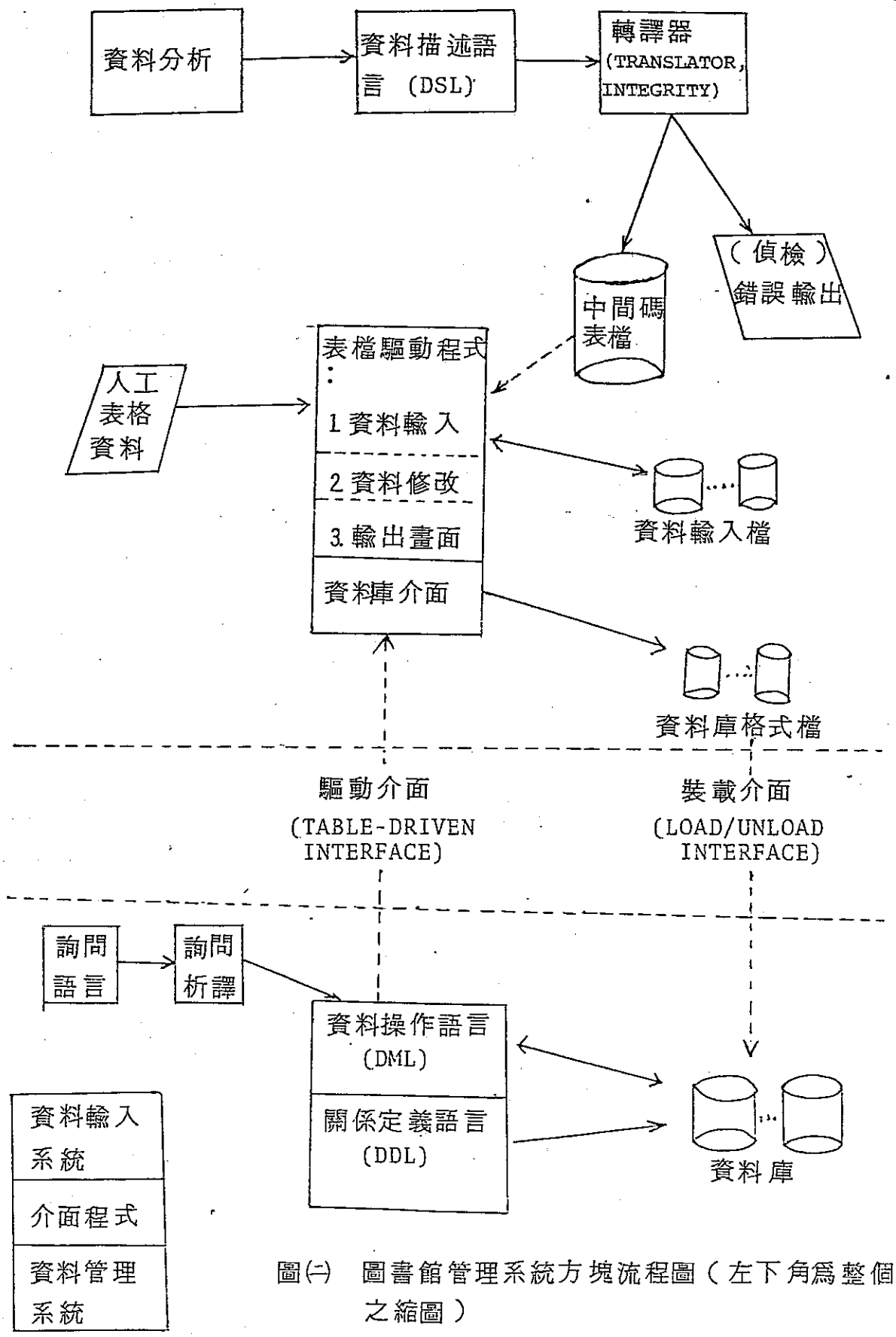
一九八一年二月，資料輸入的程式編寫完成，並試用於圖書館系統上，效果尚稱滿意。同年三月間自美國伊利諾大學獲得一個半完成的關係模式資料庫系統，經過三個月的修改和部份重新設計，配合資料輸入程式，構成了一套簡單的圖書館管理系統。目前此系統內存有六百本圖書的資料，而每天都不斷地增加新購入的書籍。

以下，第二節簡述整個圖書館管理系統的結構，
第三節描述資料輸入系統的原理和機能，
第四節描述資料庫系統的機能，
第五節簡短的結論。

二、圖書館管理系統的結構

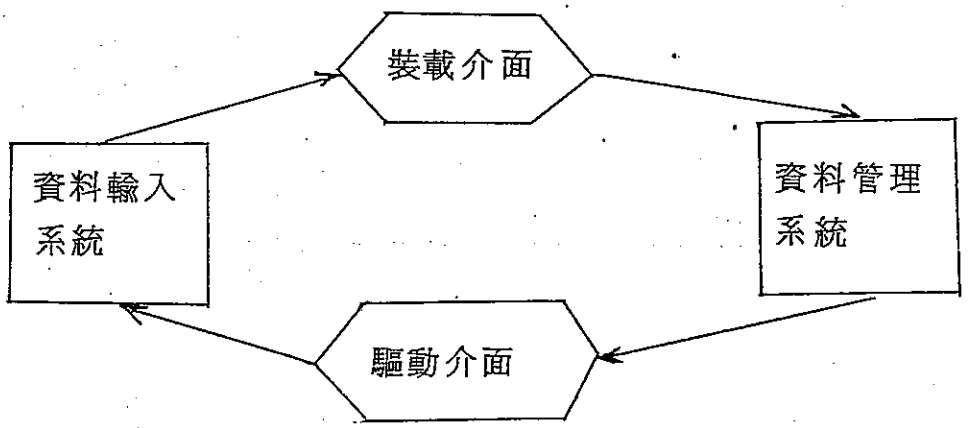
圖書館系統的軟體，主要分爲三部份：一是資料輸入，二是資料庫，三是資訊詢取。後二者我們合稱爲資料管理系統。前者稱之爲資料輸入系統。在兩系統之間依存著介面程式，以整合此兩系統而成一完整的圖書館管理系統。圖一說明在圖書館管理系統內，資料輸入系統及資料庫管理系統間的概念圖。

圖二剖析各系統詳細之操作方塊圖。資料輸入系統包括有：(1)資料欄分析步驟，(2)資料描述語言，(3)轉譯器，(4)表檔驅動輸入程式，(5)表檔驅動修



圖(二) 圖書館管理系統方塊流程圖(左下角爲整個之縮圖)

改程式，(6)畫面輸出程式，(7)資料庫介面程式等部份。在第三節中我們會有更清楚的說明。資料管理系統由(1)資料操作語言(DML) (2)關係定義語言(DDL) (3)詢問語言(4)詢問析譯等構成。在第四節中我們將一一說明這些構成元件的內容。



圖一　資料輸入及管理系統關係

三、資料輸入系統

資料處理的第一步工作就是資料輸入(data entry)　，然而這個工作往往被一般人以笨拙的方式解決（例如：將資料打在卡片上，然後集體輸入）。隨著計算機工業的發展，連綫作業輸入的要求與日俱增；提供智慧型的資料輸入系統給使用者爲勢在必行。

連綫資料輸入系統的目的在於：㈠讓使用者以最簡單、方便的方式輸入資料，㈡系統自動核對資料的正確性。一般而言，資料俱有下列之特性：

(1)資料格式 (data format)── 資料可能是整數，浮點數，文字串，・・・等等不同種類。

(2)資料連屬性 (data association)── 資料項間往往有某種特定的關係存在，這些關係的存在影響到使用者輸入資料的次序和內容。

(3)資料項型 (data type)── 資料的輸入可能是鍵入字串，單次選擇，多項選擇，是非問答，・・・等等。以選擇方式鍵入的資料，其內容往往影響到和它有關的其他資料的出現與否。換言之，資料內容的輸入通常需要遵照某些特定的順序或規則；而這些順序或規則往往因應用範圍不同有所差異。

現行一般之資料輸入系統大多是因應某種需要而製作程式。這樣的程式有很多缺乏：

(1)系統不易適應應用範圍的變更。換言之，一旦應用範圍的資料項目有所變

動，則必須重新編寫程式。

(2)缺乏一般性。由於同一系統程式只能適用一種應用範圍，開發系統的花費過鉅。

(3)系統開發的時間太長。

(4)維護困難，系統的正確性不易核對。

針對上述問題，本文描述一個俱相當可靠一般性的連綫資料輸入系統 —— GPDES(General Purpose Data Entry System) 是一表格驅動系統，它提供一種描述語言 DSL(Data Specification Language)讓系統設計者用來描述資料項的格式、型別、連屬性、值域，。。。等等。不同的應用對象，只要利用 DSL重新描述即可，毋須另外設計全套的資料輸入系統。除了俱有通用性外，由於整個系統內資料項的特性和連屬性都可用高階語言描述，可讀性高，且修改容易。DSL 的翻譯器俱有偵錯的能力，一旦設計者在描述上犯了錯誤，翻譯器即列出錯誤的情形；因此，只要設計者對整個應用範圍瞭解得很清楚，必可在數小時內設計出一個資料輸入系統。

利用DSL 描述所得的資料輸入表(data entry schema) 將被翻譯成一組表檔。這一組表檔是用來驅動資料輸入的順序，並且檢核資料內容的正誤。

除了DSL, GPDES另外提供一種高階語言LSL(Layout Specification Language)用以描述報表的格式。設計者利用LSL 可輕易地設計出各式各樣的輸出表格，以滿足使用者的需要。

以下，3·1 節描述DSL 的內容、原理和功能，

3·2 節描述 LSL 的內容、原理和功能，

3·3 節以 DSL 和 LSL 設計圖書館管理系統的資料輸入系統，

3·4 節檢討 GPDES 的能力限制及在資料修改方面所遇到的困難。

3·1 資料描述語言(Data Specification Language)

通常我們定義一在眞實世界中(real world) 單獨存在的物件；如人，公司部門，。。。等等爲一組件 (entity)。每一組件賦與一組的形容特性，稱爲組件的屬性(Attribute)，各屬性俱有下二性質：

1 屬性間的關係有1:m 及 n:m 的關係。

2 屬性關係有(1)靜態的 (static) —— 關係可以確定，且不隨時間遷移，及

(2)動態的(dynamic) —— 關係隨時間而遷移。

3. 屬性爲一不可分的個體——屬性不是由二個或二個以上的屬性組合。

一般說來，動態屬性關係較靜態屬性關係複雜。然而，在動態時間間隔很長時，在這一間隔內我們以靜態屬性關係來處理這一問題，因此，本文中

我們均以靜態來描述屬性關係。

資料描述語言即一高階之特殊語言，用以描述組件內屬性間之關係；以下，我們定義描述語言之資料標準表格 (normal form)描述語言的意義，轉換器輸出表檔格式，及資料輸入，修改原理。

(a) 描述語言句型(syntax) 及意義。

在附錄㈠中詳附了描述語言句型 (syntax) 的波格標準表示法 (Bakus Normal Form).

對於資料屬性的描述有下列六項：

(1) {[key]} 項名( )：

資料屬性的名稱，有 [key] 表示一鎖屬性項

(2) { type is { str / int / char / float } { = {初值域列} } } ;

資料屬性的類別有，str (字串) ，int(整數) ，char(字母) ，float (浮點) 。大括號表示可省略。中括號表示存在時僅選其一。初值域列爲規定此一屬性之值域，必須要同前面之str,int,... 等在意義上相符合。設錯值 (Default) 爲整數。

(3) { pic is 寬度格式 } ;

資料屬性格式之寬度格式，省略時設錯值爲寬度不限。

{sel is { uni / mul } ;}

資料屬性以單項或多項鍵入選擇；設錯值爲uni 。

(5) { case is {下一項名：(條件列) , ... } } ;

資料屬性下一鍵入及選擇屬性項之決定。設錯值爲列式下一屬性 (或條件爲假) 。

(6) ! 輸入路徑終止。

圖四是在圖書館作業上，描述圖書資料的一部份。

```
[key]regist.no( )
type is str;
pic is 4;
sel is uni;
group( )
type is str = {"textbook","reference","journal"};
 ;
check-out( )
type is char = {'y','n'};
case is {
     ISBN:(check-out == 'n')}
 ;
borrower( )
 ;
ISBN( )
 !
```

圖 (四) 資料描述語言

(b) 內示表格(Internal Representation)

描述語言經過轉譯器的轉譯而成一標準的表檔內碼，轉譯器並能偵檢語法之錯誤及保証語意上的不相混淆。爲求系統之易展性(extensibility)及彈性，我們選用內指令之格式：

內指令名～運作元1～運作元2～ ... \n

內指令名 = {type,name,pic,sel,case,pass,break}

運作元依附於內指令。

每一內指令均以'\n'表示結束。

轉譯器技巧我們將用 Top-down Recursive descent 之parsing technique 及用 Syntax-Directed translation （語法直接轉譯）(請參考 AHO and Ullman:principles of compiler design)

(c) 表檔驅動輸入：

表檔驅動輸入乃按照內指令之功作(action)以輔助資料之輸入，決定輸入路徑。

表㈠說明各內指令及其功作(action)

內指令名	功　作
name	輸出項名
type	決定初值與否，以作鍵入及選擇輸入
pic	偵查輸入長度
sel	決定多項或單項輸入
case	依據輸入之值，決定下一輸入項
pass	未作用
break	輸入完成

表 ㈠ 內指令名及其功作

因而；驅動輸入就是一解譯器(interpreter)

(d) 資料修改

輸入資料會因(1)資料錯誤(2)資料內容更易而需要修改已輸入的資料，在資料系統中資料修改的好壞直接影響到資料的一致性。最笨拙系統的處理方式，是把各有相關資料的修改完全交給使用者，讓使用者根據資料的結構，作一次又一次的修改，這樣一來，不但造成系統之能力限制(相信使用者都會抱怨的)，也使系統製造一些髒資料(dirty data)，另一解決的方式是當資料發生變易時促使系統狀態改變，而激發(trigger)一維護程式以保持系統的一致性。然而自動修改的方式，常隨系統之複雜度而成指數的增加。

在 GPDES中，根據描述語言之路徑，當資料發生變易時，系統自動偵檢路徑，作資料項之自動消去或要求輸入增添的項目，減輕使用者之負担。

以圖㈣之資料語言爲例，圖㈤之樹狀結構爲兩種可能的輸入路徑(path)：一是 regist-no—group—check-out—borrower—ISBN ，一是regist-no—group—check-out—ISBN。當修改 check-out項之值由Y 改成n 時，路徑由 a 改成 b ，仔細的看可知borrower項並不在 b 中，此時系統會自動的將 borrower項消去。反之，check-out 項之值由 n 收成 y 時，路徑由b 改成 a ，其中 borrower 項需作重新輸入，此時系統會自動要求增添borrower項的內容。同

時；GPDES 中亦設計一般編輯器(Editor)的功能。實作 (Implement) 的系統指令有

(1) c—— 修改字串
(2) d—— 消去項中之行
(3) a—— 增加項中之行
(4) t(-)np——顯示路徑
(5) s—— 顯示資料

3·2 輸出畫面

輸出畫面之設計直接可利用系統所提供的描述語言 (LSL)，在很短的時間內即可設計出所要的畫面。

LSL 描述語言包括：

(1)輸出項及其格式（橫式及直式）
(2)標題字串
(3)標題字串與輸出項間的關係
(4)空行及空格
(5)特殊畫面程式

圖(六)係圖四之例子的輸出畫面設計。

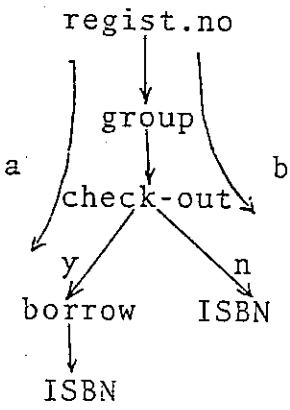


圖 (五) 輸入路徑分析

```
Layout Specification Language{0¦0}\n
\n
\n
\t % regis.no% %group%-*({0&1}%check-out%-*
%borrower%-*){2&0} -- {0¦1}%ISBN%-*
\n
%?output%
```

（圖六） 輸出畫面描述語言

LSL 各元件之說明如下：

(1) Layout Specification Language {0¦0}
標題字串，與其印出之條件。{0¦0} 表示跟前、後項無關。¦ 爲〃或〃，& 爲〃且〃。

(2) \n (line-feed)

(3) \t (tab)

(4) %group%-*

項名及其印出之格式（— ：橫式; | ：直式）。

(5) %?output%

輸出特殊程式，無法由(1)～(4)方法描述者。

3·3 圖書館管理系統之資料輸入

在圖書館專業人員的協助下，我們很快的獲取圖書館管理系統編目卡片的格式，在一張編目卡內，我們搜取了29項屬性來描述一本書，圖(七)爲資料描述語言。圖(八)爲輸出畫面描述語言。圖(九)爲詢取後印出之編目卡片。

```
[key] regist.no( ) /* registration number */
type is str;
pic is 4;
;
group( )
type is str = {"textbook","reference","journal"};
;
class.no.( ) /* classification code */
type is str;
;
cutter-no.( ) /* curteis code */
type is str;
;
author( )
type is str;
sel is mul;
;
authorship( )
type is str;
sel is mul;
;
title( )
type is str;
sel is mul;
edition( )
type is str;
sel is mul;
```

```
 ;
pub.place( )
type is str;
sel is mul;
 ;
publisher( )
type is str;
sel is mul;
 ;
pub.date( )
pic is 4;
sel is mul;
 ;
preface( )
type is str;
sel is mul;
 ;
page( )
 ;
illustration( )
type is str = {"ill."," "};
 ;
height( )
 ;
width( )
sel is mul;
 ;
subject-heading( )
type is str;
sel is mul;
 ;
copies( )
type is str;
sel is mul;
 ;
location( )
type is str;
 ;
```

```
check-out( )
type is char = {'y','n'};
case is {series-name: (check-out = = 'n')};
 ;
borrower( )
type is str;
 ;
date-due( )
type is str;
 ;
reserver( )
type is str;
sel is mul;
 ;
series-name( )
type is str;
sel is mul;
 ;
series-no.( )
type is str;
sel is mul;
 ;
note( )
type is str;
sel is mul;
 ;
reprint( )
type is str = {"Reprinted", " "};
case is {ISBN:(reprint ! = "reprinted")};
 ;
reprinter( )
type is str;
 ;
ISBN( ) /* international standard book number */
type is str;
sel is mul;
!
```

圖 (七) 圖書資料描述語言

```
%class.no.%-*
%cutter-no.%-* %author%-*
          {0&1}%title%-* / by {0&1}%authorship%-*.
{1&0} -- {0&1}%edition%-* ed. {1|0}-- {0&3}%pub.place%-*
:  {1&0}%publiher%-*, {1&0}%pub.date%-*.{1&0}
%regist.no%        {0|0}%preface%-*, {1|0}%page%-* p.{1&0}
: {0&1}%illustration%-* ; {0&0}%height%-* {0&1}%wide%-*cm.
{0|0} -- ({0&2}%series-name%-* ; {1&1}%series-no.%-*){2|0}
%copies%!*
                %language%-*.{1&0}

                %reprint%-* by {1&1}%reprinter%-*.{1&0}

                %subject-heading% *

                %ISBN%-*
borrower : {0%1}%borrower%-* {1&1}%date-due%-*

location : {0&1}%location%-*
```

圖 (八) 畫面輸出描述語言

```
130
K51         Kindred, Alton R
                Introduction to computers / by Alton R.
            Kindred. -- New Jersey : Prentice-Hall, 1976.
0348            vi, 538 p. : ill. ; 21cm.


                Reprinted by Mei Ya.

                1. computer fundamentals.


location : 1
```

圖 (九) 印出之編目卡片

3·4 系統能力限制

GPDES目前已試用於圖書館系統，人事資料，財產資料，並接中文終端機，我們發現：

(1)資料描述語言應擴充至多表格之結構描述。

(2)資料之修改沒有更明晰的描述，對多項錯誤資料之修改可能會有問題。

這些問題，都是在我們設計關係資料庫 (RDBMS) 時列入設計考慮的因素。

四、資料庫管理系統

由 GPDES 輸入的資料將透過一個界面程式直接存入一個關係模式資料庫 (RDBMS). RDBMS 提供一套指令讓使用者定義資料的格式及資料項間的關係。除了定義指令外，還有插入、刪除、取出、修改等指令讓使用者用以更動資料檔。至於資料運算則係以關係代數指令來完成；使用者可依其需要，鍵入一串關係代數指令，以取出所需資料。下面舉出兩個例子說明關係代數指令的效用：

依關係模式資料庫的理論，吾人可將圖書的屬性資料分成下列六個關係：

```
BS(REGIST-NO,GROUP,CLASS-NO,CUTTER-NO,AUTHOR, TITLE,
   EDITION,PUB-PLACE,PUBLISHER,PUB-DATE)
BD(REGIST-NO,PREFACE,PAGE,ILLUSTRATION,HEIGHT,WIDTH,
   LOCATION,SERIES-NAME,SERIES-NO,REPRINT,REPRINTER,
   ISBN)
AUS(REGIST-NO,AUTHORSHIP)
SUB(REGIST-NO,SUB-HEAD)
CHK(REGIST-NO,CHKOUT,BORROWER,BOR-DATE)
RSV(REGIST-NO,RESERVER)
```

例一、取出所有圖書的AUTHOR 和 TITLE

```
Project "AUTHOR,TITLE" OF BS INTO R
Print R
```

例二、取出所有 Printice-Hall 公司出版的圖書之 AUTHOR,CUTTER-NO, 和 ISBN.

```
project "REGIST-NO,AUTHOR,CUTTER-NO,PUBLISHER" of BS
        into R1
project "REGIST-NO,ISBN" of BD into R2
restrict R3 = R1 "PUBLISHER == 'Printice Hall'"
join R4 = R1 (*REGIST-NO;REGIST-NO) R3
print R4
```

五、結論

在這一篇論文裡，我們描述了中央研究院資訊科學研究所圖書館管理系統的軟體和硬體架構。整個系統包括資料輸入、資料庫和資料詢取三大部份。資料輸入的程式已全部完成；目前使用的資料庫系統有很多缺點，因此，自今年六月起重新設計一個關係模式資料庫管理系統，預計明年暑期完工；資訊詢取的程式由於牽涉到比較複雜的理論，尚在研究試驗中。整個系統目前已能滿足一般圖書館作業的需求。圖書館管理系統的中文化是我們下一個預定達成的目標。我們要特別强調一點：這個系統的設計具有〃萬用〃的特性。換言之，本系統不僅可使用在圖書館資料管理上，同時可使用在其他各方面的資訊管理；諸如全國人才資料檔案、計劃追踪管制檔案、儀器管理檔案、醫藥管理・・。等等。除具有〃一般性〃外，利用本系統，可在很短的時間內完成資料輸入系統的設計，是本系統的最大優點。

文後，我們非常感謝方鳳梅小姐、陳瓊音小姐辛勞的爲本文打字，與資訊所同仁的鼓勵，使本文得以完成。

六、參考文獻

1. H. V. Aho and J. D. Ullman "Principles of Compiler Design", Prentice-Hall, Englewood Cliffs, N. J. 1977.
2. C. J. Date, "An Introduction to Database Systems", Addison-Wesley, Reading, MA, 1977.
3. G. Wiederhold, "Database Design", McGraw-Hill, Inc., N. J., 1977.

Appendix

SYNTAX RULE:

1. context:= comment | item-block | comment context | item-block contex
2. comment:= /* char-string */
3. item-block:= header body-expr (seperator end-mark)
4. end-mark:= . .
5. header:= item-id( )
6. body-expr:= sub-expr ⌒(null)
7. sub-exp:= type-exp | pic-exp | sel-exp | case-exp | comment
8. seperator:= ;
9. type-exp:= type is type-declar
10. type-declar:= (int | str | char | float) assignment seperator
11. assignment:= ={const-list} |⌒
12. const-list:= const , const-list | const
13. const:= integer | character | string | float
14. pic-exp:= pic is format seperator
15. format: = integer | float
16. case-exp:= case is {case-exp-list}
17. case-exp-list:= case-exp | case-exp , case-exp list
18. case-exp:= item-id:(condition exp)
19. condition-exp:=relation-exp logical-op condition-exp | relation-exp
20. relation-exp:= item-id relation-op rvalue
21. relation-op:= > | >= | <= | != | < | ==
22. logical-op:= && | ||
23. rvalue:= integer | string | character | float
24. integer:= digit | digit integer
25. float:= .integer | integer. | integer.integer
26. character:= 'letter'
27. string:= "character-sting"
28. digit:= 0 1 1 1 2 1 3 1 4 1 5 1 6 1 7 1 8 1 9
29. character-string:= (letter|digit)*
30. letter:= a | b | c | ..... | . | - | space
31. item-id:= character-string
32. sel-exp:= sel is (mul | uni) seperator