關聯式邏輯資料庫設計

作者：朱阿水

摘　要

283-318

關聯式邏輯資料庫之設計爲一複雜的問題，本文討論此一問題，以多值相依來做分解關聯之依據，故結果之基元關聯均在第四正規形式。

此外，亦提供一擴增式實體關係圖及其相應的實際需求描述語言來做爲需求描述之工具，因此可讓設計人員以較自然且直接的方式來記錄、描述需求，並經由所製作之邏輯資料庫設計工具設計出關聯式邏輯資料庫。

1.緒論

1.1　問題背景與研究動機

邏輯資料庫設計的工作主要把多個個別應用所涉及之資料及其關係與限制設計成一個整體性的資訊結構，再將之定義成一個綱目。一個好的資料庫應能滿足所有原始的應用，並能應付一個全新的應用，故邏輯資料庫設計乃在設計出一個好的綱目，使得資料庫能展現最佳的資料獨立性（data independency）〔1〕，並具有充分的穩定性（stability）。

使用資料庫可減少資料的重複儲存、避免資料不一致、共享資料等優點，但資料庫的設計卻是一繁複的工作，尤其當資料項多達幾百項時，以人力從頭到尾來處理這些資料及其關係，是費時費力的。因此，發展一套系統化、電腦化的設計方法與工具以改善傳統的、直觀的設計方法是必須的〔2〕。目前已有一些階層式與網狀式資料庫之電腦化設計工具〔3，4〕但關聯式者卻沒什麼進展，此乃研究動機之一。

關聯式邏輯資料庫之設計方法大致分爲三種：

(1)以資料及其關係爲基礎之直接重疊併合式（merge）〔5，6〕。

(2)以函數性相依（functional dependency）爲語意基元（semantic primi-

tives ）之合成式（ synthesis ）〔 7 , 8 , 9 〕。

⑶以一大的關聯（ relation ）或平面表（ flat table ）及於其內成立之相依爲基礎之分解式（ decomposition ）〔 1 , 10 , 11 〕。

併合式是以較直觀的方式來達成設計，而合成式雖然較具系統化，但仍有難以找盡全部函數性相依、求函數性相依集合之極小含蓋（ cover ）、不易表達多值映對（ multivalued mapping ）、屬性之語意有重複（ overlooding ）之可能〔 12 〕及只得到第三正規形式（ third normal from ）之關聯等等限制與缺點。

分解式雖無合成式之屬性語意重複的問題，但也需要找出平面上所存在的所有相依，尤其在分解時要考慮該關聯的鍵〔 1 , 13 〕。

關聯式資料庫最具數學理論，研究者確信可找到更佳的演算過程以獲得系統化、電腦化的設計方法，此乃研究動機之二。

1.2 研究目的

基於以上研究動機，本研究目的如下：

⑴提出一套基於分解式而可以改進上述缺點的關聯式邏輯資料庫設計方法，使關聯式邏輯資料庫的設計方法趨於系統化。

⑵製作一設計工具，使關聯式邏輯資料庫的設計成爲電腦化、自動化。

1.3 研究範圍與限制

針對研究目的，訂定研究範圍如下：

⑴關聯式資料庫設計分邏輯上的設計（ logical design ）與實體上的設計（ physical design ），本研究只限於邏輯上的設計。

⑵資料庫之邏輯上的設計方法很多，依本研究之目的，故只使用了合成技術與分解技術。

⑶本研究不包括有關原始資料之同名異義（ homonyms ）或異名同義（ synonyms ）以及使用上的某些衝突（ conflict ）之研究。

1.4 研究方法與步驟

本研究主要採用「理論分析法」，其步驟如下：

⑴蒐集有關資料庫理論與實務的文獻資料與已完成專案計劃報告，以及軟體工具（ software tool ）的製作理論之文獻。

(2)研閱、分析並整理收集之文獻資料與報告。

(3)思考並設計系統化、電腦化的關聯式邏輯資料庫設計理論與方法。

(4)以數學及資料結構表現出設計理論之演算過程。

(5)以PASCAL程式語言實現該設計理論並製作成一設計工具。

(6)撰寫研究報告。

1.5 名詞釋義

(1)關聯

設A_1，A_2，…，A_n為屬性（attribute），DOM（A_1），DOM（A_2），…，DOM（A_n）為屬性之定義域，則DOM(A_1)，DOM(A_2)，…，DOM(A_n)之卡廸遜乘積（cartesian product）的所有部分集合均稱為在這些定義域上的一個關聯，記為

$$r(A_1, A_2, \cdots, A_n) \subseteq DOM(A_1) \times DOM(A_2) \times \cdots \times DOM(A_n)$$

其中之n稱為關聯r之度數（degree），所有r之元素

$$t_i(A_1, A_2, \cdots, A_n), \quad 1 \leq i \leq m$$

稱為關聯r之維（tuple），記為

$$t_i(A_1, A_2, \cdots, A_n) \in r(A_1, A_2, \cdots, A_n)$$

或

$$t_i \in r$$

其中之m稱為關聯r之基數（cardinality）。由於r上存在某些制約（constraint），故定義一關聯構成（relation scheme）

$$R(\Omega, DOM, C)$$

其中

Ω：為屬性集合

DOM：為定義域所成集合

C：為相依（dependency）所成集合

若$r(\Omega)$滿足於C，則稱$r(\Omega)$為$R(\Omega, DOM, C)$的一個外觀（extension），記為

$$r \in R$$

在r中有某些屬性之值是可唯一辨識的，故可用來分辨關聯的每一維，此稱為r之待用鍵（condidate key），可選定其一為主鍵（primary key）。即若以t〔A〕

表維 t 相應於屬性 A 之值，則 t_i〔 KEY-ATTRIBUTE 〕可唯一決定該維 t_i ，$t_i \in \gamma$。

⑵關聯式資料庫

關聯式資料庫構成（ database sheme ）爲

$$DB_1 = R\ (\ \Omega\ ,\ DOM\ ,\ C\ )$$

或

$$DB_2 = (\ \rho\ ,\ C_\rho\ )$$

其中

$$\rho = \{\ R_i\ (\ \Omega_i\ ,\ DOM\ ,\ C_i\ )\ ,\ 1 \leq i \leq k\ \}$$

C_ρ 表 ρ 之元素間的一些限制，如可合併、可聯集等。

⑶函數性相依（ functional dependency, FD ）

於關聯 γ 中，對於所有 t〔 X 〕至多只有一個 t〔 Y 〕與之對應，X ，$Y \subseteq \Omega$。亦即

$$t_1\ ,\ t_2 \in \gamma$$

若

$$t_1\ [\ X\ ] = t_2\ [\ X\ ]$$

則

$$t_1\ [\ Y\ ] = t_2\ [\ Y\ ]$$

稱X 函數性決定Y 或Y 函數性相依於X ，以

$$FD : X \rightarrow Y$$

表之。X ，Y 分別稱爲FD 之左側（ left-side ），右側（ right-side ）。

⑷多值相依（ multivalued dependency, MVD ）

設 $X \twoheadrightarrow Y$ 表從X 到Y 之一多值映對（ multivalued mapping ），於 γ（ Ω ）上若存在二個互爲獨立的多值映對：

$$X \twoheadrightarrow Y\ ,\ X \twoheadrightarrow Z$$

則 γ（ Ω ）存在一個多值相依，記爲

$$MVD : X \twoheadrightarrow Y \mid Z$$

亦即對任何

$$t_1\ ,\ t_2 \in \gamma$$

若

$$t_1\ [\ X\ ] = t_2\ [\ X\ ]$$

則存在一維 t ，

$$t \in r$$

$$t[X] = t_1[X]$$

且

$$t[Y] = t_1[Y]$$

且

$$t[Z] = t_2[Z]$$

⑸正規化（ normalization ）

正規化是造成較高層次正規形式（ normal gorm ）之關聯的過程，此可經由合成與分解來達成。

1.6 論文內容架構

本文於第二節討論關聯式邏輯資料庫的設計方法，其中包括說明擴增式實體關係圖及實際需求描述語言之使用。第三節簡述一個邏輯資料庫設計工具的製作，第四節舉一實例說明工具之使用，最後本文結論及未來展望。

2.關聯式邏輯資料庫設計工具之製作

關聯式邏輯資料庫設計的目的在找到資料庫構成（ database scheme ）

$DB = (\rho, C_\rho)$

設計過程中，主要的關鍵在於是否完整收集全部的屬性與相依，由於直接在一大的表或關聯上描述資料之間的關係可能會使相依收集不完整，故此處提出以擴增式實體係圖（ EERD ）及相應的實際需求描述語言（ RRDL ）來做爲需求描述之工具，期能自然地揭發有意義的相依。

整個設計過程包括實際需求收集與產生關聯式邏輯資料庫兩大部分。

⑴實際需求收集：

由於實際需求不是以電腦爲主（ computer-oriented)之資訊層次，因此最好以較能表現資料原始語意的模型來表現，如實體關係模型（ entity-relationship model)〔 5 〕及語意資料模型（ semantic data model)〔 14 〕等，以模型所提供的資料關係圖或語言來記錄需求。本文中使用一擴增式實體關係圖（ Extented Entity-Relationship Diagram, EERD)及其對應的實際需求描述語言（ Real world Requirement Description Language, RRDL)來描述每一應用之資料。

⑵產生關聯式邏輯資料庫：

此工作由一邏輯資料庫設計工具（ Logical Database Design Tool, LDBDT ）來達成，此工具係以多值相依（ multivalued dependency)〔 11 〕來做爲正規化（ normalization)〔 1 〕之依據，故能保證邏輯資料庫之每一關聯都在第四正規形式（ fourth normal form, 4NF)〔 11 〕。

2.1 需求描述工具

EERD可幫助我們迅速地表現個別應用之資料及其關係，再輔以RRDL將之表現爲語言形式。

⑴擴增式實體關係圖（ EERD ）主要由節點（ node)與弧（ arc)所形成。節點包括實體集E，關係集R，屬性A之值集V。而屬性爲從E或R到V的映對。弧包括無向弧及有向弧，無向弧存於實體集與關係集之間，其上的字母表該關係集內，各實體集可能結合之實體數目，亦即表示了結合形態（ association type)。有向實線弧從E到V或從R到V，並標示屬性名稱於弧旁邊，弧旁之字母表屬性之映對關係。有向虛線

弧由V到E表該屬性可唯一辨識該實體，可做爲實體鍵〔15〕。另一有向虛線弧由V_i到V_j表屬性ATT_i可以決定屬性ATT_j。各符號如圖2－1所示，其中ATT_1，ATT_6分別爲E_1，E_2之實體鍵，而ATT_8可決定ATT_9。

此外，可視關係集爲一實體集，而可再與另一實體作結合，此稱爲關係式實體集E_R，以菱形外加方形來表示，如圖2－2所示。

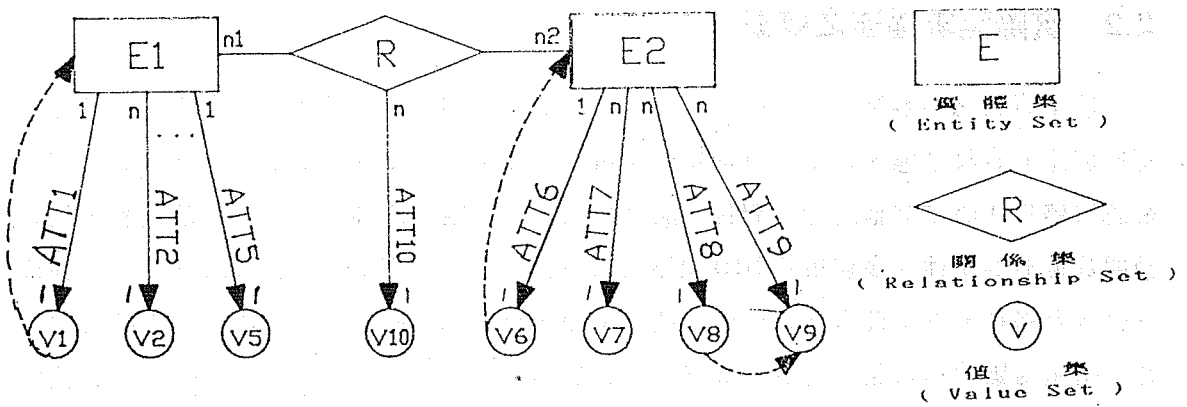


圖2－1　EERD之符號

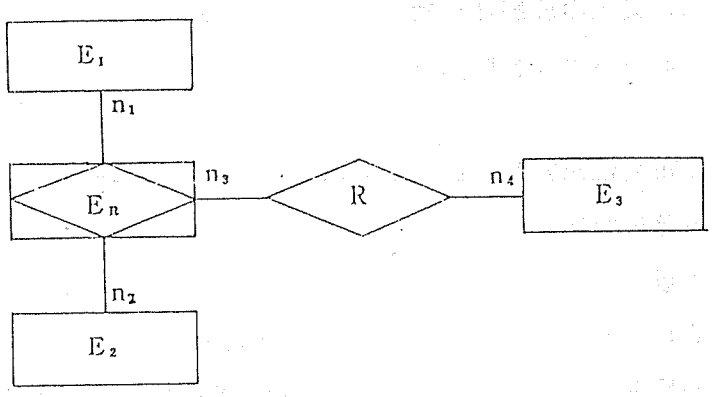


圖2－2　關係式實體集E_R

(2)實際需求描述語言（RRDL）

RRDL把個別之EERD所表示的資料以語言形式來表現，一個以RRDL所記錄的內容可包含任何數目的個別應用，每一個應用由鍵字（keyword）“DFVEW”開始描述，每一個應用給予一個名稱，一個應用涉及很多實體集及關係集〔5〕，並以鍵字“

EDVEW ”結束該描述。

每一個實體集由鍵字“ DEFENT ”開始描述，並由“ EDENT ”結束對一實體集的描述。其中所含的屬性使用“ ATT ”來描述，用“ KEY ”來定出實體鍵，並以“ DET ”來列出可能的決定子屬性（ determinant ）。

每一關係集由“ RLP ”開始描述，其中要列出所涉之實體集、結合型態，及可能擁有的屬性。RRDL格式如圖 2 － 3 所示，於〔 16 〕中有其BNF 表示法。

2.2 實際需求描述之收集

收集資料之內容應含有實體集、屬性及關係集。對每一實體集給予一名稱，及擁有那些屬性，資料型態、長度、可能單位、重複出現數（ occurrence ），及是否被用來辨認實體集或決定其他屬性。對每一關係集給予一名稱，所涉及之實體集的名稱，結合型態及擁有的屬性。並使用EERD記下資料及其關係，再用RRDL表成語言形式，而成一輸入資料檔，再從此資料檔取得一些有用的資料做爲自動化設計之用。此包括屬性集合、基礎函數性相依集合及多重基礎多值相依集合，以下說明如何到RRDL之資料檔取得這些資料。

(1)屬性集合

一個RRDL資料檔記錄著多個個別應用，每一個別應用中所有實體集及所有關係集之屬性，均需加以收集，由於屬性可能存於二個以上的個別應用或實體集內，故需把重複之屬性去除，便成爲極小之屬性集合。

(2)相依集合

由於相依隱藏在實體與屬性、決定子屬性與被決定屬性之間，以及實體集之結合關係中，以下分四部分說明之：

(a)設實體集以

$$E(K, S_1, S_2, \cdots, S_m, M_1, M_2, \cdots, M_n)$$

來表示，K表實體鍵屬性，S_i 表 occurrence $= 1$ 之屬性，而M_j 表 occurrence > 1 之屬性，則存在函數性相依FD 或多值映對MF 如下：

$$FD : K \rightarrow S_i, \quad 1 \leq i \leq m$$

$$MF : K \twoheadrightarrow M_j, \quad 1 \leq j \leq n$$

(b)在實體集內宣告之決定子屬性，其格式爲

DET：左側屬性 ATT_l ，映對型態，右側屬性 ATT_r 則存在

FD ： $ATT_l \rightarrow ATT_r$ ，若映對型態爲“ ONE ”

```
DBCTE : database_name {            };
  TGDBMS : dbms_type    {            };
  DFVEW  : view_name ,priority {            };
           DFENT :entity_name , cardinality  {       };
                  ATT : att_name , value_type ,length,
                        unit , occurrence {    };
                  ATT :
                    .
                    .
                  ATT :
                  KEY : attn_list  {                 };
                  DET : left_attn_list ,mapping_type,
                        right_attn_list {       };
                  DET :
                    .
                    .
           EDENT ;
           ...
           .....
           DFENT :
             :
             :
           EDENT ;
           RLP : relationship_name {    };
                 RLPFORM     : relationship_form {     };
                 INVOLENT    : entity_name_list {    };
                 ASSOCTYPE   : assocation_type    {    };
                 RLPATT      : att_name ,value_type,
                               length, unit, occurrence{ };
                 RLPATT      :
                   :
                   :
           AOQ : operation_type, object_list, frequency;
           AOQ :
             :
             :
  EDVEW {    };
  :
  :
  DFVEW : ..........
      .
      .
      .
  EDVEW ;
EDCTE .
```

圖2－3　RRDL之格式

MF ： $ATT_1 \rightarrow\!\!\!\!\rightarrow ATT_r$ ，若映對型態爲“ MANY ”

由於視實體集內之屬性存在某種關係，當一決定子屬性明顯地決定某屬性，則該決定子亦決定其他屬性，故存在

MF ： $ATT_1 \rightarrow\!\!\!\!\rightarrow \Omega_e - (ATT_1 \cdot ATT_r)$

Ω_e 表該實體集之全部屬性。

(c)在關係集中，依不同的結合型態分爲下列四種：

Ⅰ、若 ASSOCTYPE = “ ONE , ONE ” ，則存在

FD ： $K_1 \rightarrow K_2$

FD ： $K_2 \rightarrow K_1$

K_1 , K_2 爲實體集 E_1 , E_2 之實體鍵屬性。

Ⅱ、若 ASSOCTYPE = “ MANY , ONE ” ，則存在

FD ： $K_1 \rightarrow K_2$,

MF ： $K_2 \rightarrow\!\!\!\!\rightarrow K_1$,

Ⅲ、若 ASSOCTYPE = “ ONE , MANY ” ，則存在

MF ： $K_1 \rightarrow\!\!\!\!\rightarrow K_2$

FD ： $K_2 \rightarrow K_1$

Ⅳ、若 ASSOCTYPE = “ MANY , MANY ” ，則存在

MF ： $K_1 \rightarrow\!\!\!\!\rightarrow K_2$

MF ： $K_2 \rightarrow\!\!\!\!\rightarrow K_1$

(d)若關係集擁有屬性 RLPATT ，則存在

FD ： $K_1 \cdot K_2 \rightarrow$ RLPATT ，若 occurrence = 1 ；

MF ： $K_1 \cdot K_2 \rightarrow\!\!\!\!\rightarrow$ RLPATT ，若 occurence > 1 ；

假設整個 RRDL 所描述到的屬性均有某種關係，當 $K_1 \cdot K_2$ 明顯地決定 RLPATT ，則 $K_1 \cdot K_2$ 亦隱含地決定其他實體鍵屬性。若所有實體鍵屬性所成集合 KSET ，其部分集合爲 $KSET_i$ ， $1 \leq i \leq n$ ，則存在

MF ： $KSET_1 \rightarrow\!\!\!\!\rightarrow KSET_2$ ，

故對於擁有屬性 RLPATT 的關係集，應再收集

MF ： $K_1 \cdot K_2 \rightarrow\!\!\!\!\rightarrow KSET - K_1 \cdot K_2$

所收集之 FD 集合以 C_{FD} 表之，由於可視 FD 爲 MF ，故可假設 MF 之集合爲 MVD 集合，並以 C_{MVD} 表示之。但 C_{FD} ， C_{MVD} 中可能有非基礎元素存在，即若

f_1 ： $X \rightarrow Y$

$f_2 : X' \rightarrow Y$

均屬於C_{FD}，而$X' \supseteq X$，稱f_2爲非基礎FD，可視其爲多餘（ redundancy ）。

以下定義基礎相依，並把非基礎相依去除。

【定義2－1】偏序集（ partial order set ）$PO = (C, \leq)$，

$C = \{ (P, Q) \mid P, Q \in \Omega \}$

對於

$(X, Y), (U, V) \in C$，

若

$X \cup U = U$且$Y \cup V = V$

則稱

$(X, Y) \leq (U, V)$

以有序對

$< (X, Y), (U, V) >$

表之

即 $< (X, Y), (U, Y) > \in PO$。

若FD：$X \rightarrow Y$表爲(X, Y)，則可定義$R(\Omega)$上之基礎FD（ elementary FD，EFD）之集合爲：

$C_{EFD} = \{ FD : (X, Y) \mid \min (< (X, Y), (U, V) > \in (C_{FD}, \leq)) \}$，

min函數取該偏序之最小元素。

同理，$R(\Omega)$上之基礎MVD（ elementary MVD， EMVD ）之集合爲：

$C_{EMVD} = \{ MVD : (X, Y) \mid \min (< (X, Y), (U, V) > \in (C_{MVD}, \leq)) \}$，

C_{EFD}，C_{EMVD}爲具有

$(C_{EFD} \cup C_{EMVD})^+ \supseteq (C_{FD} \cup C_{MVD})$

之性質，故可自C_{FD}，C_{MVD}中去除非基礎者。

由於視FD爲MVD，故$R(\Omega)$上所有相依集合爲

$$C = \{ MVD \mid MVD_i : X_i \twoheadrightarrow Y_{i1} \mid Y_{i2} \mid \cdots \mid Y_{in},\ X_i Y_{i1} Y_{i2} \cdots Y_{in} = \Omega_i,\ 1 \leq i \leq K, \text{且} \bigcup_{i=1}^{K} \Omega_i = \Omega \}$$

若$n = 1$，則稱爲一個單一MVD（ single MVD ），即

$MVD_i : X_i \twoheadrightarrow Y_{i1}$；

若$n > 1$，則稱爲多重MVD（ multiple MVD ），即

$MVD_i : X_i \twoheadrightarrow Y_{i1}, X_i \twoheadrightarrow Y_{i2}, \cdots, X_i \twoheadrightarrow Y_{in}$ 。

若－MVD：$X \twoheadrightarrow Y \mid Z$，$Z = \phi$，即 $XY = \Omega$，

稱此種MVD爲平凡MVD（ trivial MVD ）。又若關聯 $r(\Omega)$

$r(\Omega) \in R(\Omega, DOM, C)$

只存在平凡MVD，則 $r(\Omega)$ 爲一基元關聯（ atomic relation ），且在 4 NF〔 17 〕。

單一MVD在分解過程中終會成爲平凡MVD，故可先予以去除，使MVD集合均爲多重基礎MVD（ mutiple elementary MVD, MEMVD ），以 C_{MEMVD} 表之。

2.3 分解與資訊之保存

分解爲正規化所用的一種方法，分解關聯可基於 FD〔 1 , 18 〕、MVD〔 12 , 17 , 19 〕、或其組合〔 20 〕來達成。分解須先找出關聯中獨立成分的相依〔 21 〕，且須確保能恢復原有之資訊〔 22 〕。

設 Ω_1 , Ω_2 爲 Ω 之部分集合，若且唯若

$$\Omega_1 \cap \Omega_2 \rightarrow \Omega_1 - \Omega_2$$

或

$$\Omega_1 \cap \Omega_2 \rightarrow \Omega_2 - \Omega_1$$

則 $R(\Omega)$ 可分解爲兩個無損失合併（ lossless joining ）的子關聯（ sub-relation ）$R_1(\Omega_1)$ 及 $R_2(\Omega_2)$〔 23 〕。或可用 $R(\Omega)$ 上之一函數性相依

$$FD : X \rightarrow Y$$

來分解 $R(\Omega)$ 成爲 $R_1(\Omega_1)$ 及 $R_2(\Omega_2)$，其中

$$\Omega_1 = X \cdot Y, \quad \Omega_2 = X \cdot (\Omega - Y) = \Omega - Y,$$

並稱 $R(\Omega)$ 保有一分解：

$$DEC_1 = (\Omega_1, \Omega_2)。$$

而 $$DEC_2 = (\Omega_3, \Omega_4)$$

$$= (\Omega_1 W, \Omega_2 W)$$

$$= (XYW, \Omega - Y)$$

亦爲 $R(\Omega)$ 之一分解，但 DEC_2 具有多餘性，故若 $R(\Omega)$ 中存在有函數性相依：

$$X \rightarrow Y$$

$$XW \rightarrow Y$$

因 $XW \supseteq X$ 則所有 $XW \rightarrow Y$

稱爲非基礎函數相依（ non-elementary FD ）。

又根據Fagin之MVD分解定理〔 11 〕，若且唯若 r（ Ω ）中存在多值相依

MVD：$X \twoheadrightarrow Y \mid Z$，

則

$\pi_{XY}(r) \bowtie \pi_{XZ}(r) = r$，$\pi$、$\bowtie$分別爲投影（ projection ）與自然合併（ natural join ）運算。

故可用MVD來分解關聯。

設 r 被分解爲子關聯

r_i，$1 \leq i \leq K$，$K \in I^+$（正整數）

這些子關聯應能保存或恢復 r 之資訊，此包括屬性資料與相依。

若

$$\bowtie_{i=1}^{K} r_i = r$$

則稱該分解爲一無損失合併之分解，否則稱爲有損失合併（ lossy join ）之分解。例如把R（ XYZ ）及其上之

FD：$X \rightarrow Y$

分解爲 R_1（ XY ）及 R_2（ YZ ），此爲一有損失合併之分解。

設 C_i 表各 r_i（ Ω_i ）中之相依，則其聯集要能等於或蘊含（ implied ）C，即

$$(\bigcup_{i=1}^{K} C_i)^+ \supseteq C。$$

故是否成功地分解一關聯，要測試

$$(\bigcup_{i=1}^{K} C_i)^+ \supseteq C$$

是否成立，若成立，表該分解爲一能保存相依（ preserving dependency ）之分解。

例如把R（ XYZ ）及其上之

$C = \{ X \rightarrow Y ; Y \rightarrow Z ; X \rightarrow Z \}$

分解爲：

R_1（ XY ），$C_1 = \{ X \rightarrow Y \}$

R_2（ XZ ），$C_2 = \{ X \rightarrow Z \}$

此爲一無損失合併之分解，但卻不能保存相依。但若分解爲：

R_1（ XY ），$C_1 = \{ X \rightarrow Y \}$；

R_2（YZ），$C_2 = \{ Y \rightarrow Z \}$；

此爲一無損失合併且保存相依之分解，因爲

$(C_1 \cup C_2)^+ = \{ X \rightarrow Y ; Y \rightarrow Z \}^+ \supseteq \{ X \rightarrow Y ; Y \rightarrow Z ; X \rightarrow Z \}$

2.4 分解之過程

根據已收集完整的屬性集合Ω，基礎函數性相依集合C_{EFD}，及多重基礎多值相依集合C_{MEMVD}，可視之爲形成一個大的關聯構成

R（Ω，DOM，C），

其中

$C = C_{EFD} \cup C_{MEMVD}$。

已知以

MVD：$X \twoheadrightarrow Y$

去分解R（Ω），若合於保存相依之條件，則可得兩子關聯R_1（Ω_1），R_2（Ω_2），分別承有R（Ω）中某些FD及MVD，以C_{1FD}，C_{2FD}，C_{1MVD}，C_{2MVD}表之

當 $C_{1FD} \cup C_{2FD} \neq C_{FD}$，

則要求

$(C_{1FD} \cup C_{2FD})^+ \supseteq C_{FD}$

成立，否則表示R（Ω）此時不能以該MVD來分解。同理，分解結果亦要求能保存MVD。

整個分解之過程如圖2－4之流程圖所示，此爲一遞迴式分解動作，每一子關聯都被處理直到成爲不能分解的基元關聯爲止，亦即關聯之$C_{MVD} = \phi$。若$C_{MVD} \neq \phi$，則選一MVD繼續分解，故我們將發現整個分解結果會如圖2－5所示之二元樹狀之形式。

(1)決定子關聯之基礎函數性相依

設企圖以一MVD

g：$X \twoheadrightarrow Y$

分解R（Ω）爲R_1（Ω_1），R_2（Ω_2），則這些子關聯之C_{1FD}，C_{2FD}可由以下之敍述來決定之。

【敍述2－1】C_{FD}爲R（Ω）上之FD集合，且

$f : X \rightarrow Y \in C_{FD}$

則X相對於C_{FD}之封閉

X^+爲

Ⅰ、$X^+ \supseteq X$ ，

Ⅱ、每一$W \rightarrow A \in C_{FD}$ ，$W \subseteq X^+$ ，$A \in X^+$ 。

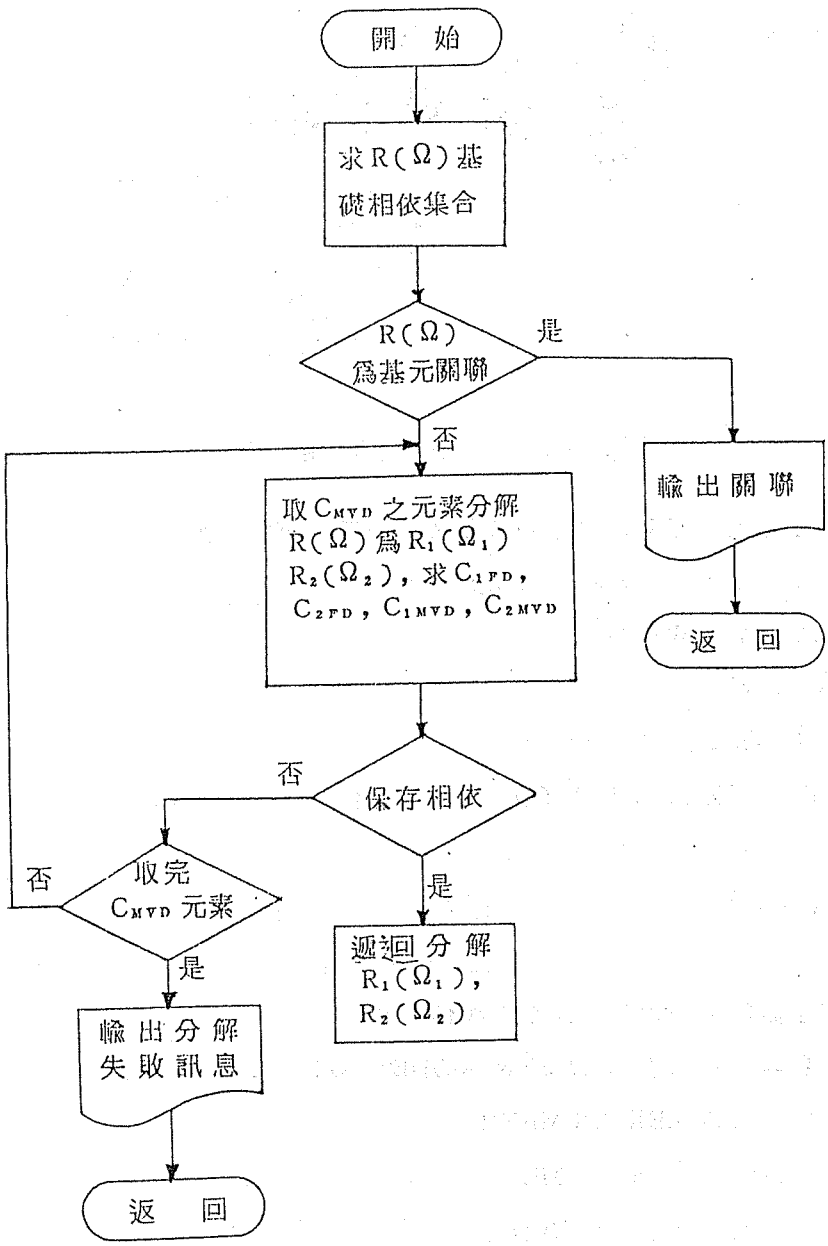


圖 2 — 4　分解之流程

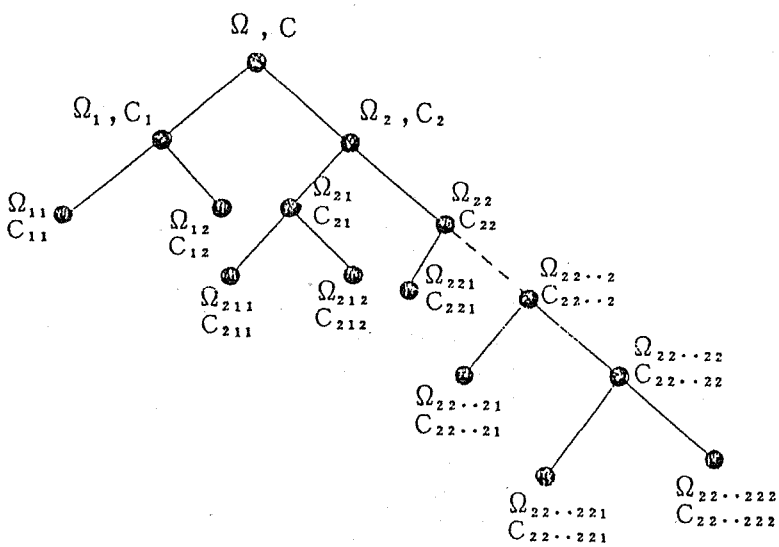


圖 2 － 5　二元樹狀之分解結果

【敍述 2 － 2 】f ：X→Y 爲在R（Ω）上成立之基礎 FD ，即

f ∈ C_{FD} ，其範圍（ scope ）爲

XY＝W ，W⊆Ω ，

若　　$\Omega_1 \supseteq W$

則　　f 亦在 R_1（Ω_1 ）中成立。

令R（Ω）之FD 集合爲 C_{FD}，且 R_1（Ω_1 ）與 R_2（Ω_2 ）者分別爲 C_{1FD} 與 C_{2FD} ，則由敍述 2 － 2 知：

C_{1FD} ＝｛ f ：X→Y ｜ f ∈ C_{FD} 且 X ◦ Y ⊆ Ω_1 ｝，

C_{2FD} ＝｛ f ：X→Y ｜ f ∈ C_{FD} 且 X ◦ Y ⊆ Ω_2 ｝。

以下以一例子說明求子關聯之基礎函數性相依。

【例 2 － 1 】令Ω＝（ CUST#, CUSTN, MODEL, QTY, TEC#, TECN ）

CUST# ：CUSTOMER NUMBER,

CUSTN ：CUSTOMER NAME,

MODEL ：COMPUTER MODEL,

QTY　：QUANTITY OF A GIVEN MODEL,

TEC#　：TECHNICIAN NUMBER

TECN　：TECHNICIAN NAME

宣告之相依有下列六個：

CUST# → CUSTN ；　　CUST#, MODEL → QTY ；

TEC# → TECN；　　CUST# ↠ MODEL , QTY ；

CUST# ↠ TEC# , TECN ；

TEC# ↠ CUST# , CUSTN , MODEL , QTY ；

則 C_{FD} = { CUST# → CUSTN ; CUST# , MODEL → QTY ; TEC# → TECN; }

C_{MVD} = { CUST# ↠ CUSTN ; TEC# ↠ TECN ; CUST# ↠ MODEL , QTY;
TEC# ↠ CUST# , CUSTN , MODEL , QTY ; CUST# ↠ TEC# ,
TECN ; }

若以MVD：CUST# ↠ MODEL , QTY 去分解 R (Ω)，則

Ω_1 = (CUST# , MODEL , QTY)

Ω_2 = (CUST# , CUSTN , TEC# , TECN)

C_{1FD} = { CUST# , MODEL → QTY ; }

C_{2FD} = { CUST# → CUSTN ; TEC# → TECN ; }

⑵決定子關聯之多重基礎多值相依

以MVD，**g** ：X ↠ Y 去分解關聯 R (Ω)，各子關聯所承有的MVD：C_{1MVD} 及 C_{2MVD}，與求出 C_{1FD} , C_{2FD} 者不同，設

Ω = { X , Y , Z , W , T }

C_{MVD} = { $\mathbf{g}_1$ ： X ↠ YT ;
$\mathbf{g}_2$ ： X ↠ ZT ;
$\mathbf{g}_3$ ： X ↠ W }

若以 $\mathbf{g}_1$ ： X → YT 去分解 R (Ω)，則

Ω_1 = XYT , Ω_2 = XWZ ,

依照求 C_{2FD} 方式只得

C_{2MVD} = { X ↠ W ; }

根據 R (Ω) 之隱藏式多值相依 (embedded MVD) 〔 11 〕

C_{2MVD} 應爲：

C_{2MVD} = { X ↠ W ; X ↠ Z ; }

因此，求 C_{1MVD} , C_{2MVD} 可用下列二式：

C_{1MVD} = { MVD ： X ↠ Y` | **g** ： X ↠ Y ∈ C_{MVD} , X ⊆ Ω_2 ,
Y` = Y ∩ Ω_1 ≠ ϕ }

$$C_{2MVD} = \{ MVD : X \twoheadrightarrow Y` \mid g : X \twoheadrightarrow Y \in C_{MVD} , X \subseteq \Omega_2 ,$$
$$Y` = Y \cap \Omega_2 \neq \phi \}$$

故仍如例 2 － 1 以

$$MVD : CUST\# \twoheadrightarrow MODEL , QTY$$

分解R（Ω）得

$$C_{1MVD} = \{ CUST\# \twoheadrightarrow MODEL , QTY ; \}$$
$$C_{2MVD} = \{ CUST\# \twoheadrightarrow CUSTN ; CUST\# \twoheadrightarrow TEC\# , TECN ;$$
$$TEC\# \twoheadrightarrow TECN ; TEC\# \twoheadrightarrow CUST\# , CUSTN \} ,$$

⑶函數性相依保存測試與封閉之求法

當

$$C_{1FD} \cup C_{2FD} \neq C_{FD} ,$$

表示R（Ω）不一定能分解爲R_1（Ω_1）與R_2（Ω_2），但若$C_{1FD} \cup C_{2FD}$之含蓋（cover）能包含C_{FD}，則此分解仍能保存相依。

由於求含蓋爲一很費時之問題〔24，25，26〕，例如

$$C_{1FD} \cup C_{2FD} = \{ X \rightarrow Y_i , 1 \leq i \leq n \} ,$$

則 $(C_{1FD} \cup C_{2FD})^+ = \{ X - W_j , 1 \leq j \leq 2^n \}$

故此處並不用此方法求出含蓋

$$(C_{1FD} \cup C_{2FD})^+ ,$$

而是利用歸屬問題（membership problem）〔25，27〕來測試某一FD

$$h : X \rightarrow Y$$

是否屬於$(C_{1FD} \cup C_{2FD})^+$。由敍述 2 － 2，若

$$h : X \rightarrow Y$$

之範圍X Y包含於Ω，表h在R（Ω）上成立。故若

$$h \notin (C_{1FD} \cup C_{2FD}) ,$$

則可重複應用Armstrong三公設及FD推演法則〔23〕於$(C_{1FD} \cup C_{2FD})$之上，求X之封閉$X^+ = Q$，$Q \subseteq \Omega$，

若

$Y \subseteq Q$，即表示

$XY \subseteq XQ \subseteq \Omega$，

故

$h : X \rightarrow Y \in (C_{1FD} \cup C_{2FD})^+$。

以下爲一求屬性X之封閉的演算法

【演算法 2 － 1 】求屬性X相對於$C_{1FD} \cup C_{2FD}$之封閉X^+。

輸入：屬性集合Ω，在R（Ω）上成立之FD集合$C_{1FD} \cup C_{2FD}$及屬性X，$X \subseteq \Omega$。

輸出：X之封閉，X^+

方法：利用下列步驟求出一連串的$X^{(i)}$，

$$X^{(i)} \subseteq X^{(i+1)} \subseteq \Omega$$

直到 $X^{(i+1)} = X^{(i)}$爲止

則 $X^+ = X^{(i)}$

步驟：

Ⅰ、$i \leftarrow 0$，$X^{(i)} \leftarrow X$

Ⅱ、$X^{(i+1)} \leftarrow X^{(i)} \cup A$；其中$A \subseteq Z$，$W \subseteq X^{(i)}$

每一$W \rightarrow Z \in (C_{1FD} \cup C_{2FD})$

Ⅲ、檢查是否$X^{(i+1)} = X^{(i)}$

若眞，則已求得$X^+ = X^{(i)}$

若假，則回頭到（Ⅱ）。

(4)多值相依保存測試與相依基之求法

由於視FD爲MVD，故將$C_{1FD} \cup C_{2FD}$聯上C_{1MVD}及C_{2MVD}得總MVD集合。

$$C_{TMVD} = (C_{1FD} \cup C_{2FD} \cup C_{1MVD} \cup C_{2MVD})$$

之後，檢查原來R（Ω）上之C_{MVD}是否被其所蘊含，即$C_{TMVD}{}^+ \supseteq C_{MVD}$是否成立，此可由Armstrong公設及MVD推演法則〔17，23〕來達成。

求某MVD集合之含蓋亦極爲費時，故此處並不求$C_{TMVD}{}^+$而是對每一MVD

$$g : X \twoheadrightarrow Y \in C_{MVD}$$

檢查其是否在C_{TMVD}中，若不在其中，則求X相對於C_{TMVD}之相依基（dependency basis）〔25，26〕，再檢查Y是否爲某些相依基之聯集，若爲眞，表該MVD

$$g : X \twoheadrightarrow Y \in C_{TMVD}{}^+$$

即C_{MVD}蘊含$X \twoheadrightarrow Y$。

以下說明相依基及相依基之求法。

【敍述 2 － 3 】設Ω爲屬性集合

$$G = \{ \Omega_i \subseteq \Omega, 1 \leq i \leq K, \bigcup_{i=1}^{K} \Omega_i = \Omega \}$$

G對於聯集（union）、交集（intersection）、差集（difference）等運算

具封閉性，即

$$\Omega_i \cup \Omega_j \text{、} \Omega_i \cap \Omega_j \text{、} \Omega_i - \Omega_j \in G$$

則存在 $B = \{ b_i , i = 1 \cdots n \}$, $b_i \cap b_j = \phi$, $i \neq j$

使每一$\Omega_i \in G$均爲B中某些元素之聯集，稱B爲G之基（ basis ）。

設 $|G| = K$, $K \in I^+$ ，則當需要時，可對其B之元素取聯集 ，加入G，使$|G| > K$。

已知Ω，C_{TMVD}及另一MVD

$$g : X \twoheadrightarrow \Omega \text{`} , X \subseteq \Omega$$

假設可用MVD推演法則求得X之多值相依集合

$$G = \{ \Omega_i \mid \forall (X \twoheadrightarrow \Omega_i) \in C_{TMVD}{}^+ \} ,$$

由敍述2－3知G有B，稱B爲X對於C_{TMVD}之相依基（ dependency basis ）。若Ω′爲B之某些元素之聯集，表$\Omega' \in G$，即

$$X \twoheadrightarrow \Omega' \in C_{TMVD}{}^+ \text{。}$$

以下爲求相依基之演算法並舉實例說明求相依基。

【演算法2－2】求X對於C_{TMVD}之相依基，DFPB(X)。

輸入：屬性集合Ω，在R（Ω）上成立之MVD：C_{TMVD}，以及X，$X \subseteq \Omega$。

輸出：X對於C_{TMVD}之相依基，DEPB(X)。

方法：利用MVD推演法則及下列步驟完成之。

步驟：

Ⅰ、令$DEPB(X) = \{ Y , \Omega - X'Y \mid \forall (X' \twoheadrightarrow Y) \in C_{TMVD}$ 且$X' \subseteq X \}$。

Ⅱ、對DEPB(X)中元素有交集者，用分解律來精細化，使$DEPB(X) = \{ b_1 , b_2 , \cdots , b_n \}$, $b_i \cap b_j = \phi$, $i \neq j$

Ⅲ、找到可能的遞移相依（ transitive dependency ）

$(W \twoheadrightarrow Z) \in C_{TMVD}$ 且$W \subseteq b_i$

把$Z - b_i$加入DEPB(X)，再利用分解律精細化，最後結果爲X對於C_{TMVD}的相依基。

【例2－2】$\Omega = (A , B , C , D , E , H)$

$C_{TMVD} = \{ DE \twoheadrightarrow C , A \twoheadrightarrow BC , A \twoheadrightarrow H \}$,

求$C_{TMVD}{}^+ \supseteq C_{TMVD} \cup \{ A \twoheadrightarrow CDE \}$是否成立？

因$A \twoheadrightarrow CDE \notin C_{TMVD}$，所以求A之相依基

$DEPB(A) = \{ BC , DEH \}$；由於$A \twoheadrightarrow BC \mid \Omega - ABC$

$= \{ BC , DEH , H , BCDE \}$；由於$A \twoheadrightarrow H \mid \Omega - AH$

$= \{ DE , BC , H \}$；用MVD分解律

$= \{ DE , BC , H \} \cup \{ C \}$；由於$A \twoheadrightarrow DE$，$DE \rightarrow G$之遞移

$= \{ DE , BC , H , C \}$

$= \{ DE , B , C , H \}$；用MVD分解律。

由以上之推演，可知CDE爲DEPB(A)之元素DE、C的聯集，

故 $A \twoheadrightarrow CDE \in C_{TMVD}^{+}$

即 $C_{TMVD}^{+} \supseteq C_{TMVD} \cup \{ A \twoheadrightarrow CDE \}$成立。

以上爲測試某分解是否能保存相依，若能保存相依，則分解成功，否則再找另一 MVD 嘗試去分解。如例 2 － 1

$\Omega =$（CUST#，CUSTN，MODEL，QTY，TEC#，TECN），先以多值相依。

MVD：CUST# $\twoheadrightarrow$ TEC#，TECN去分解R（Ω），則

$\Omega_1 =$（CUST#，TEC#，TECN）

$\Omega_2 =$（CUST#，CUSTN，MODEL，QTY）

$C_{1FD} = \{$ TEC# $\rightarrow$ TECN $\}$

$C_{2FD} = \{$ CUST# $\rightarrow$ CUSTN；CUST#，MODEL $\rightarrow$ QTY $\}$

$C_{2MVD} = \{$ CUST# $\twoheadrightarrow$ TEC#，TECN；TEC# $\twoheadrightarrow$ TECN；

TEC# $\twoheadrightarrow$ CUST# $\}$

$C_{2MVD} = \{$ CUST# $\twoheadrightarrow$ CUSTN；CUST# $\twoheadrightarrow$ MODEL，QTY $\}$

因 $C_{1FD} \cup C_{2FD} = C_{FD}$，所以保存FD。又

$C_{TMVD} = C_{1FD} \cup C_{2FD} \cup C_{1MVD} \cup C_{2MVD}$

$= \{$ CUST# $\twoheadrightarrow$ TEC#，TECN，TEC# $\twoheadrightarrow$ TECN；

CUST# $\twoheadrightarrow$ CUSTN；TEC# $\twoheadrightarrow$ CUST#；

CUST# $\twoheadrightarrow$ MODEL，QTY；$\}$

故知 $C_{TMVD} \neq C_{MVD}$，因有一MVD

g：TEC# $\twoheadrightarrow$ CUST#，CUSTN，MODEL，QTY；

$g \in C_{MVD}$，但 $g \notin C_{TMVD}$，故求DEPB（TEC#）：

DEPB(TEC#) $= \{$ TECN；CUST#；MODEL；QTY；CUSTN $\}$

故 $g \in (C_{TMVD})^{+}$，

即 $(C_{TMVD})^{+} \supseteq C_{MVD}$，

故分解成功。

以MVD：CUST#→→TEC#，TECN 分解後所得結果及其繼續做分解之最後結果如圖2－6所示，共有四個基元關聯，分別爲R_1（TEC#，TECN），R_2（TEC#，CUST#），R_3（CUST#，CUSTN）及R_4（CUST#，MODEL，QTY）。

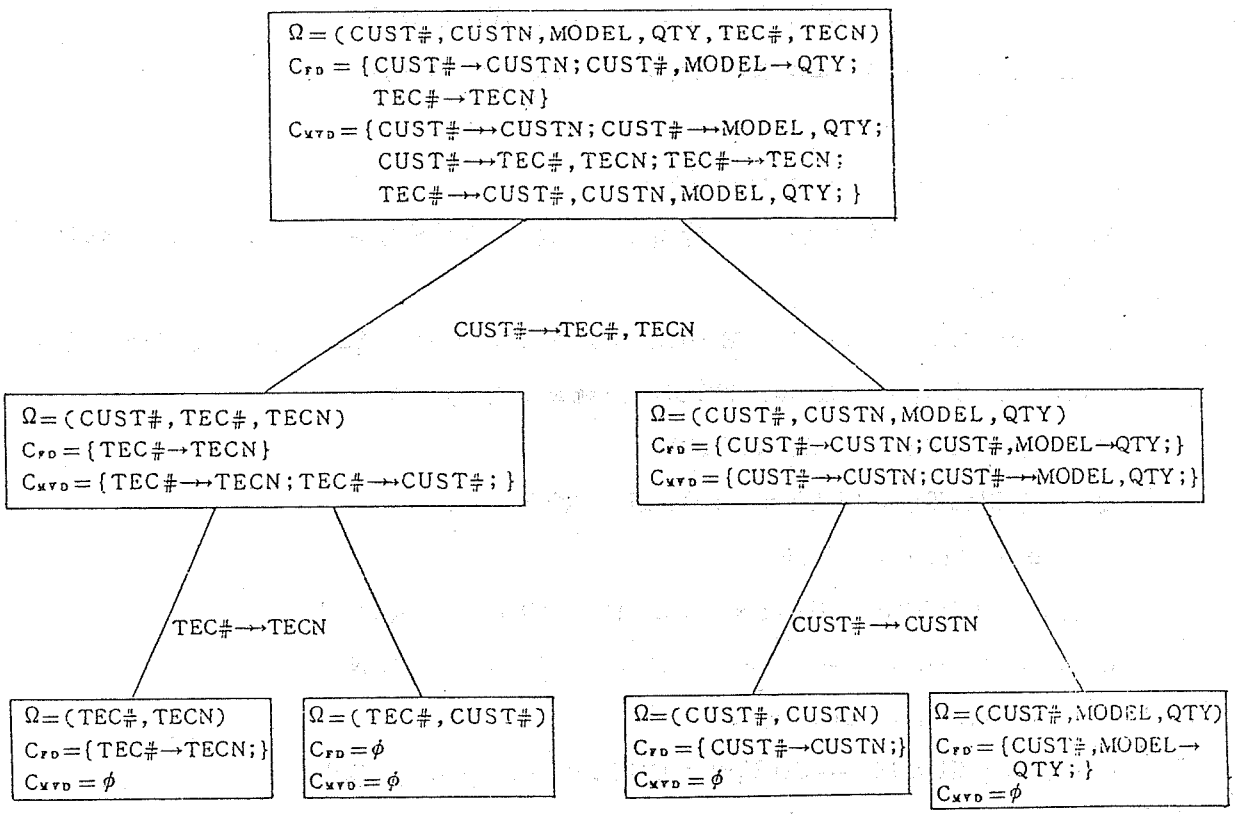


圖2－6　例2－1之分解結果

3.關聯式邏輯資料庫設諸工具之製作

本節說明如何根據上述之原理製作一關聯式邏輯資料庫設計工具（ Logical Database Design Tool, LDBDT ）。由於在實際設計所含之資料爲一變數，因此採用具有動態儲存域（ dynamic extent ）及能處理指標（ pointer ）資料型態的PASCAL程式語言來製作。

LDBDT之程式結構如圖3－1所示，主程式下有五個程序：掃描程序、屬性收集程序、相依收集程序、分解關聯程序、及輸出程序。

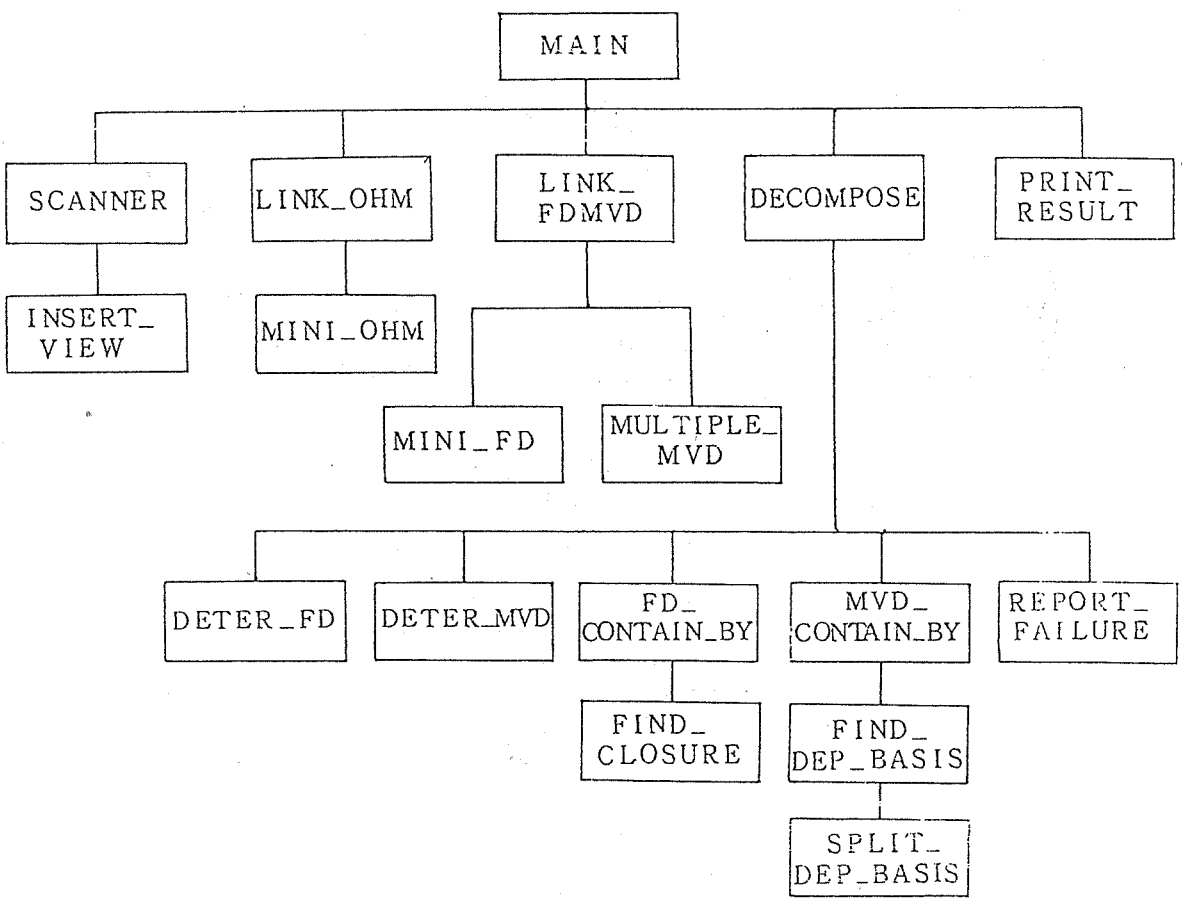


圖3－1　LDBDT之程式結構圖

以下分別說明這五個程序的功能：

(1)掃描程序（ SCANNER ）

負責自RRDL所描述之輸入資料檔讀入需求之內容並處理之。其中包括檢查語法之

錯誤，並輸出錯誤代碼；去除多餘空白；將結果寫到一輸出檔；並把個別應用所含的資料按序安排於一資料結構中。

⑵屬性收集程序（ LINK_OHM ）

收集RRDL 輸入資料檔中所有之屬性，並做極小化處理，即去除重複的屬性。

⑶相依收集程序（ LINK_FDMVD ）

收集輸入資料檔中所含之 FD 及MVD ，做極小化處理，並求基礎FD及多重基礎MVD 等，相依收集之處理如圖 3 － 2 所示。

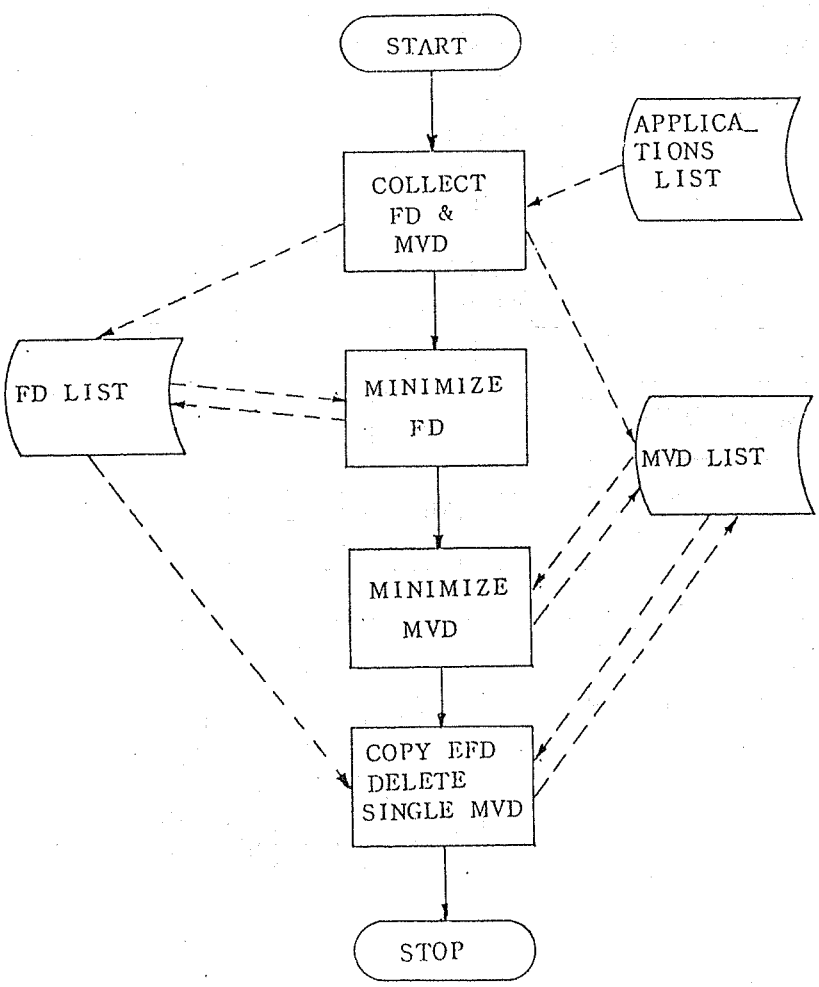
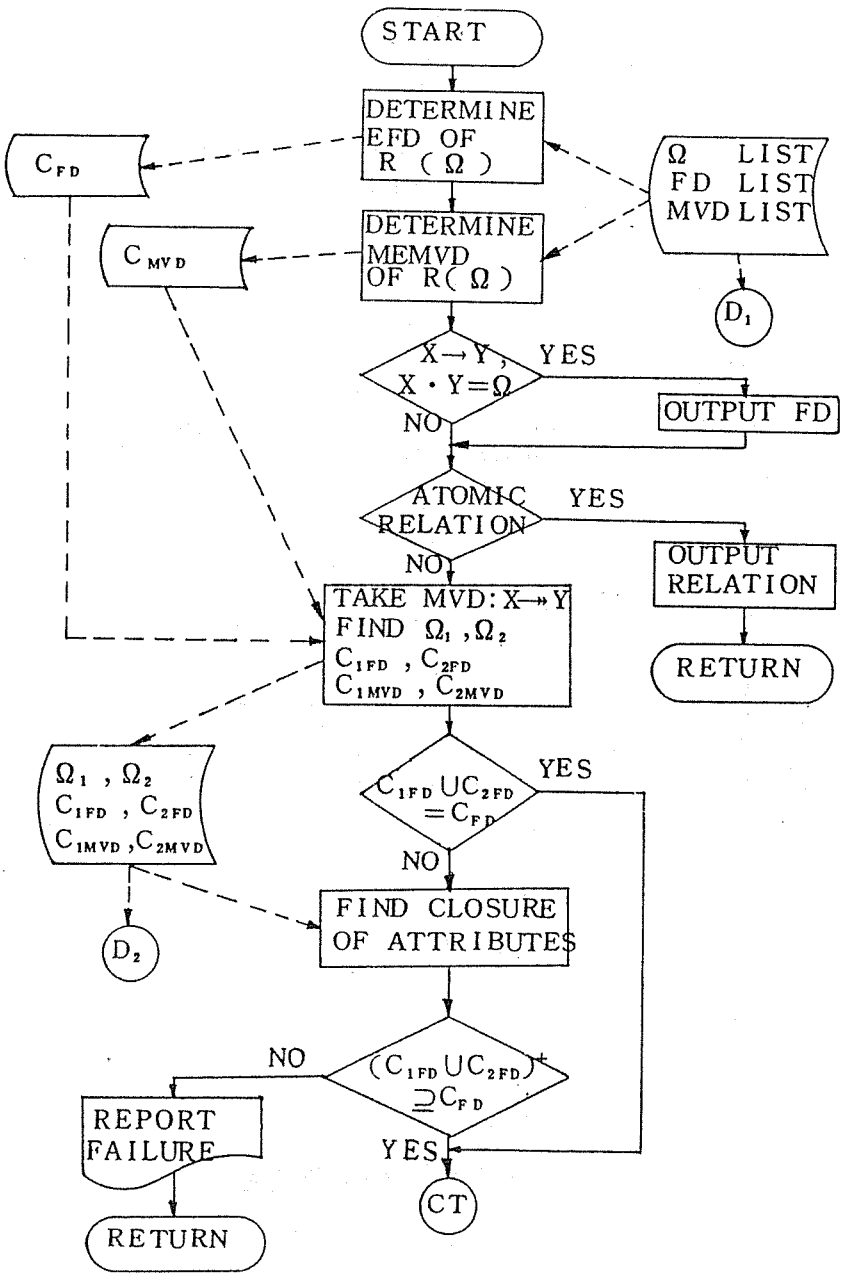


圖 3 － 2　相依收集之處理流程圖

⑷分解關聯程序（ DECOMPOSE ）

把R （ Ω ）依據其上之MVD 來分解爲 較小的子關聯，其中要測試相依之保存，以及包括求封閉與相依基。分解之處理流程如圖 3 － 3 所示。


START
DETERMINE EFD OF R (Ω)
C_{FD}
Ω LIST
FD LIST
MVD LIST
DETERMINE MEMVD OF R(Ω)
C_{MVD}
D_1
X→Y, X·Y=Ω
YES
NO
OUTPUT FD
ATOMIC RELATION
YES
NO
OUTPUT RELATION
RETURN
TAKE MVD: X→→Y
FIND Ω_1, Ω_2
C_{1FD}, C_{2FD}
C_{1MVD}, C_{2MVD}
Ω_1, Ω_2
C_{1FD}, C_{2FD}
C_{1MVD}, C_{2MVD}
$C_{1FD} \cup C_{2FD} = C_{FD}$
YES
NO
D_2
FIND CLOSURE OF ATTRIBUTES
$(C_{1FD} \cup C_{2FD})^+ \supseteq C_{FD}$
NO
YES
REPORT FAILURE
RETURN
CT

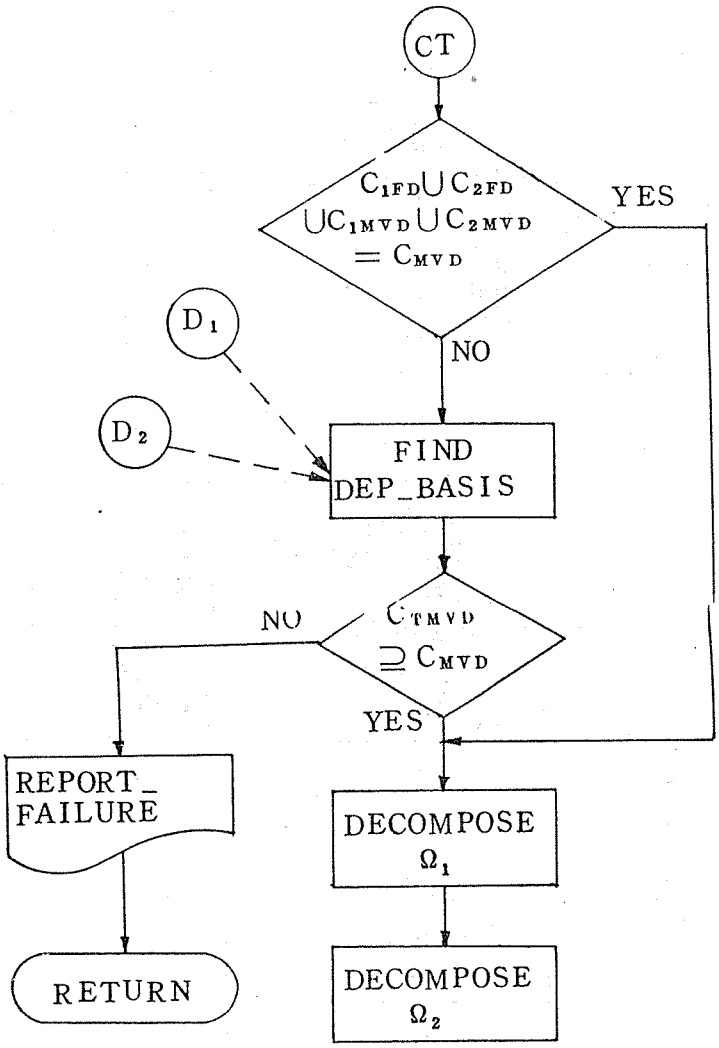


圖 3 — 3　分解關聯之處理流程圖

⑸輸出程序（ PRINT_RESULT ）

把分解所得結果，包括子關聯及其上之 FD ，寫到一個輸出資料檔，以便印出。工具之整體處理流程如圖 3 － 4 所示，於〔 16 〕中有大部分的演算過程。

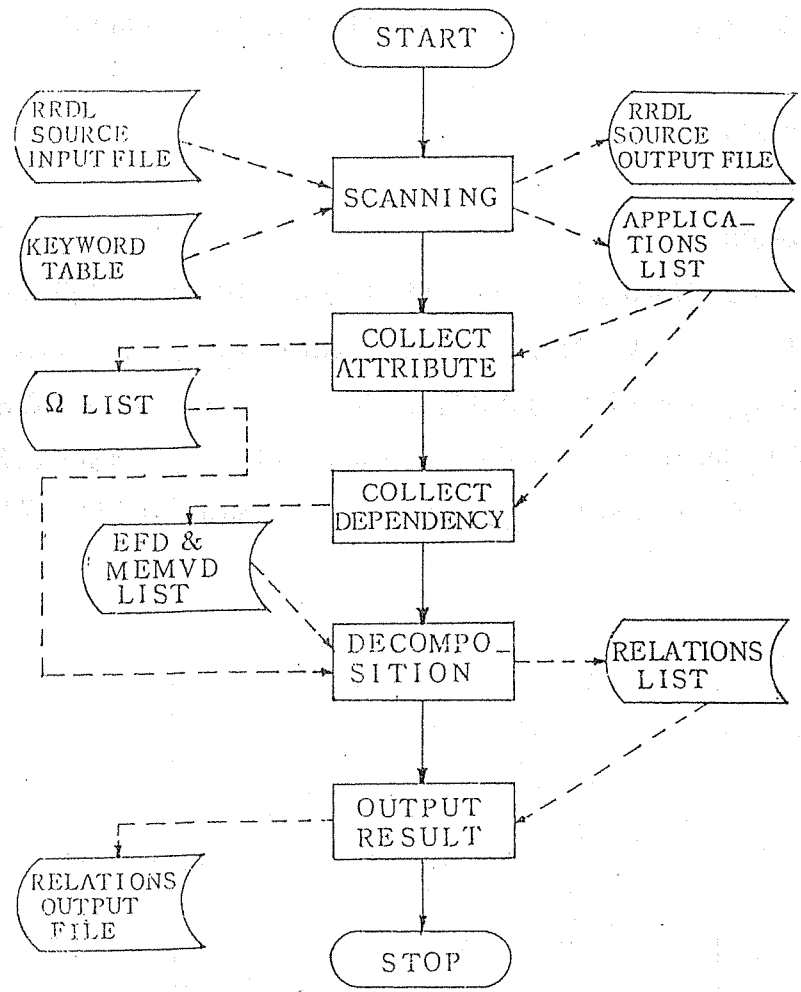


圖 3-4　LDBDT之處理流程圖

圖 3 － 4　LDBDT之處理流程圖

4.實例說明

本節以一實例說明完整的邏輯資料庫之設計過程。

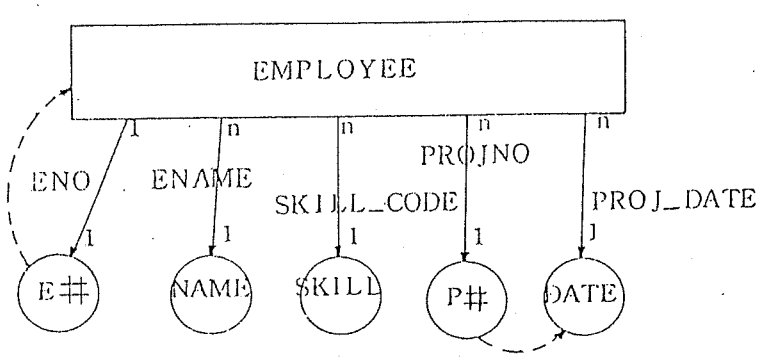
設某公司要查詢有關於計劃、參與人員及所用零件之資料，使用EERD記錄下列三個應用：

⑴查詢員工與計劃

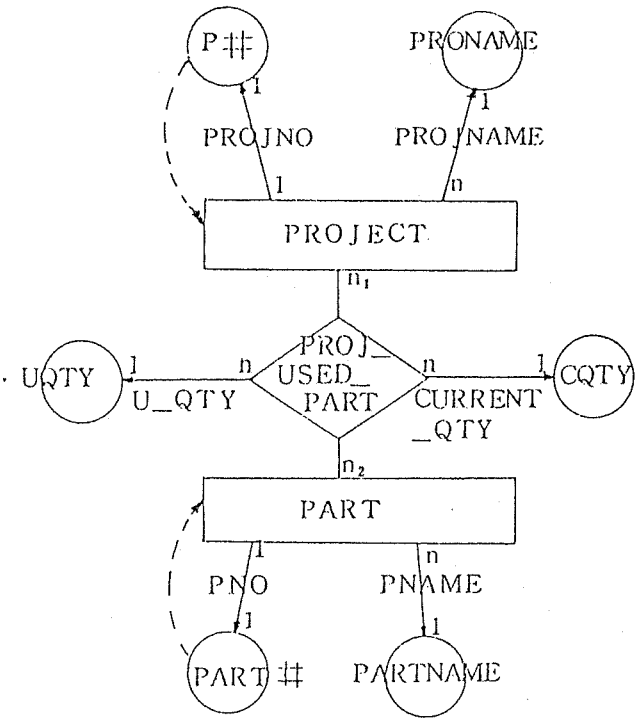
⑵查詢計劃與零件

⑶查詢零件與供應商

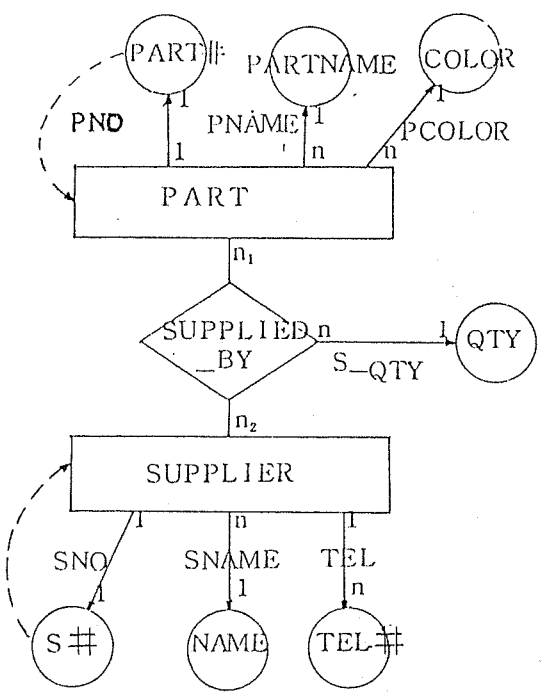
首先把每一應用以EERD記錄下來，三個應用之EERD如圖4－1所示。其次把EERD轉成RRDL之語言形式，如圖4－2所示。RRDL所描述之輸入資料檔經LDBDT來收集之極小屬性集合、FD集合、MVD集合分別如圖4－3，圖4－4及如圖4－5所示。求出之EFD、MEMVD集合分別如圖4－6，及如圖4－7所示。經分解而得之基元關聯與FD如圖4－8所示，最後經輸出程序併合基元關聯而輸出之關聯如圖4－9所示。



(a)應用(1)



(b)應用(2)



(c)應用(3)

圖 4 — 1　實例之EERD

```
DBCTE : ENG_SCHDULE ;

  TGDBMS : RELATIONAL;

  DFVIEW  : EMP_PROJ, 3 ;

    DFENT : EMPLOYEE, 100;
            ATT : ENO, C, 10, 1;
            ATT : ENAME, C, 32, 1;
            ATT : SKILL_CODE, C, 7, 1;
            ATT : PROJNO, C, 8, 1;
            ATT : PROJ_DATE, C, 8, 1;
            KEY : ENO;
            DET : PROJNO, ONE, PROJ_DATE;
    EDENT;

    AOQ : FIND EMPLOYEE, 100/DAY;
  EDVEW;

  DFVIEW :PROJECT_PART, 1;

    DFENT : PROJECT,21;
            ATT :PROJNO,C,8,1;
            ATT :PROJNAME,C,32,1;
            KEY : PROJNO;
    EDENT;

    DFENT : PART,105;
            ATT :PNO,C,10,1;
            ATT : PNAME, C, 32,1;
            KEY : PNO;
    EDENT;

    RLP : PROJ_USED_PART;
          RLPFORM  : BINARY;
          INVOLENT :PROJECT, PART;
          ASSOCTYPE: MNY,MNY;
          RLPATT   : U_QTY,I, 4, CPS, 1;
          RLPATT   : CURRENT_QTY,I, 4, CPS, 1;
    AOQ: FIND CURRENT_QTY OF PART BACK PROJECT, 50/DAY;
  EDVEW;

  DFVIEW : PART_SUPPLIER, 2;
    DFENT: PART, 150;
           ATT: PNO, C,8, 1;
           ATT: PNAME, C, 32, 1;
           ATT: PCOLOR, C, 10, 1;
           KEY: PNO;
    EDENT;

    DFENT : SUPPLIER,50;
            ATT: SNO, C, 10, 1;
            ATT: SNAME, C, 32,1;
            ATT: TEL, C,10, 11;
            KEY: SNO;
    EDENT;

    RLP : SUPPLIED_BY;
          RLPFORM  : BINARY;
          INVOLENT : PART,SUPPLIER;
          ASSOCTYPE: MNY,MNY;
          RLPATT   : S_QTY,I, 6, CPS, 1;
    AOQ :FIND  SUPPLIER,50/DAY;
  EDVEW;

EDCTE .
```

圖4－2　實例之RRDL的描述

```
* THE ATTRIBUTES LIST *
    ENO
    ENAME
    SKILL_CODE
    PROJNO
    PROJ_DATE
    PROJNAME
    PNO
    PNAME
    U_QTY
    CURRENT_QTY
    PCOLOR
    SNO
    SNAME
    TEL
    S_QTY
```

圖 4 — 3　實例之極小屬性集合

```
* THE FD LIST *
    ENO --> ENO
    ENO --> ENAME
    ENO --> SKILL_CODE
    ENO --> PROJNO
    ENO -- > PROJ_DATE
    PROJNO --> PROJNO
    PROJNO --> PROJNAME
    PNO --> PNO
    PNO --> PNAME
    PNO --> PNO
    PNO --> PNAME
    PNO --> PCOLOR
    SNO --> SNO
    SNO --> SNAME
    PROJNO --> PROJ_DATE
    PROJNO,PNO --> U_QTY
    PROJNO,PNO --> CURRENT_QTY
    PNO,SNO --> S_QTY
```

圖 4 — 4　實例之 FD 集合

```
* THE MVD LIST *
    SNO -->> TEL
    PROJNO -->> ENO,ENAME,SKILL_CODE
    PROJNO -->> PNO
    PNO -->> PROJNO
    PNO -->> SNO
    SNO -->> PNO
    PROJNO,PNO -->> ENO,SNO
    PNO,SNO -->> ENO,PROJNO
```

圖 4 — 5　實例之 MVD 集合

```
* THE EFD LIST *

  ENO --> ENAME
  ENO --> SKILL_CODE
  ENO --> PROJNO
  ENO --> PROJ_DATE
  PROJNO --> PROJNAME
  PNO --> PNAME
  PNO --> PCOLOR
  SNO --> SNAME
  PROJNO --> PROJ_DATE
  PROJNO,PNO --> U_QTY
  PROJNO,PNO --> CURRENT_QTY
  PNO,SNO --> S_QTY
```

圖 4 — 6　實例之EFD集合

```
* THE MEMVD LIST *

  ENO -->> ENAME
  ENO -->> SKILL_CODE
  ENO -->> PROJNO
  ENO -->> PROJ_DATE
  PROJNO -->> PROJNAME
  PNO -->> PNAME
  PNO -->> PCOLOR
  SNO -->> SNAME
  PROJNO -->> PROJ_DATE
  PROJNO,PNO -->> U_QTY
  PROJNO,PNO -->> CURRENT_QTY
  PNO,SNO -->> S_QTY
  SNO -->> TEL
  PROJNO -->> ENO,ENAME,SKILL_CODE
  PROJNO -->> PNO
  PNO -->> PROJNO
  PNO -->> SNO
  SNO -->> PNO
  PROJNO,PNO -->> ENO,SNO
  PNO,SNO -->> ENO,PROJNO
```

圖 4 — 7　實例之MEMVD集合

* ATOMIC_RELATION *	* FD on the ATOMIC_RELATION *
(ENO, ENAME)	ENO --> ENAME;
(ENO, SKILL_CODE)	ENO --> SKILL_CODE;
(PROJNO, PROJNAME)	PROJNO --> PROJNAME;
(PNO, PNAME)	PNO --> PNAME;
(PNO, PCOLOR)	PNO --> PCOLOR;
(SNO, SNAME)	SNO --> SNAME;
(PROJNO, PROJ_DATE)	PROJNO --> PROJ_DATE;
(PROJNO, PNO, U_QTY)	PROJNO,PNO --> U_QTY;
(PROJNO, PNO, CURRENT_QTY)	PROJNO,PNO --> CURRENT_QTY;
(PNO, SNO, S_QTY)	PNO,SNO --> S_QTY;
(SNO, TEL)	
(PROJNO, ENO)	ENO --> PROJNO;

圖 4 — 8　實例分解後之基元關聯

```
  * RELATION *                          * FD on the RELATION *
-------------------------------------------------------------
( ENO, ENAME, SKILL_CODE, PROJNO )

                                  ENO --> ENAME;
                                  ENO --> SKILL_CODE;
                                  ENO --> PROJNO;
-------------------------------------------------------------
( PROJNO, PROJNAME, PROJ_DATE )

                                  PROJNO --> PROJNAME;
                                  PROJNO --> PROJ_DATE;
-------------------------------------------------------------
( PNO, PNAME, PCOLOR )

                                  PNO --> PNAME;
                                  PNO --> PCOLOR;
-------------------------------------------------------------
( SNO, SNAME )

                                  SNO --> SNAME;
-------------------------------------------------------------
( PROJNO, PNO, U_QTY, CURRENT_QTY )

                                  PROJNO,PNO --> U_QTY;
                                  PROJNO,PNO --> CURRENT_QTY;
-------------------------------------------------------------
( PNO, SNO, S_QTY )

                                  PNO,SNO --> S_QTY;
-------------------------------------------------------------
( SNO, TEL )
-------------------------------------------------------------
```

圖 4 — 9　實例分解結果輸出之關聯

5.結論與建議

5.1 結論

在本文中，我們討論了關聯式邏輯資料庫的設計方法，並提供一套設計工具，其中包括了使用擴增式實體關係圖與實際需求描述語言來做爲需求描述，及使用分解方式來得到關聯式資料庫。使用此種工具來做需求描述，比直接使用一個大的平面表或關聯及其內成立的相依（ dependency ）來做資料描述更能捕捉資料之語意，也更能自然地發現存在於資料之間的關係。而所提之設計工具能自動地收集屬性、基礎函數性相依、多重基礎多值相依等對於分解關聯有用之資料，並以多值相依來分解關聯，產生第四正規形式之關聯的輸出結果。

5.2 建議

利用這種演算法式的設計方式，不管資料項多達幾百項，仍可很快的得到輸出結果。雖然＂演算過程＂之自動化程度很高，但在進入演算程序之前我們已假設：

⑴資料所存在的某些衝突（ conflict）已在需求收集、分析時解決。這些衝突包括同名異義（ homonyms ）、及異名同義（ synonyms ）以及資料之間結合關係的不一致〔 28 〕。

⑵對於相同之資訊需求，雖然可用多個不同的擴增式實體關係圖來記錄，但我們認爲每一個都已攜帶了正確且有用的資訊〔 29 〕，但這些均是需要再加以研究的內容。

一個設計出來的結果，除了其結構正確外，亦應能符合操作上的需求，故應在實際需求描述語言中加入所有可能操作之相關資料，使能在輸出的關聯上做一些效度測試，如此，才能保證所設計之資料庫能滿足所有操作需求，但本工具未能提供這方面的功能。

最後，提出兩點做爲今後研究之參考：

⑴研究需求描述工具，以直接交談方式來解決衝突，並獲得資料之規格。

⑵研究能藉輸出之關聯及操作之相關資料來驗證輸出結果之效度並發展出應用程式。

參考資料

1. Codd, E. F., " A Relational Model of Data for Large Shared Data Banks, " Commun. ACM, Vol.13, No.6, pp.377-387, Jun. 1970.
2. Navathe, S., " Important Issues in Database Design Methodologies and Tools, " COMPUTER-ADIDED DATABASE DESIGN: The DATAID Projest, North-Holland, pp.199-212, 1985.
3. Raver, N. and G. U. Hubbard, " Automated Logical Database Design: Concepts and Applications, " IBM Syst. J. No.3, pp.287-312, 1977.
4. 陳祥義，"電腦輔助邏輯資料庫設計方法，"臺灣，國立臺灣大學電機工程學碩士論文，民國七十年五月。
5. Chen, P. P., " The Entity-Relationship Model--Toward a Unified View of Data, " ACM Trans. Database Syst., Vol.1, No.1, pp.9-36, Mar. 1976.
6. Bert, M. N., G. Ciardo, B. Demo and P. Ciolito, " The Logical Design in the DATAID Project: The Easymap System," COMPUTER-ADIDED DATABASE DESIGN: The DATAID projest,North-Holland, pp.97-114. 1985.
7. Bernstein, P. A., " Synthesizing Third Normal Form Relations from Functional Dependencies, " ACM Trans. Database Syst., Vol.1, No.1, pp.277-298, Dec. 1976.
8. Vetter, M. and R. N. Maddison, Database Design Methodology, Prentice-Hall International Inc., Englewood Cliffs, New Jersey, 1981.
9. Chung, S. K. and W. H. Cheng, " Database Skeleton and Its Appliation to Logical Database Synthesis, " IEEE Trans. Software Eng., Vol. Se-4, No.1, pp.18-30, Jan. 1978.
10. Delobel, C. and R. G. Casey, " Decomposition of a Data Base and the Theory of Boolean Switching Functions, " IBM J. Res. Develop., pp.374-386, Sept. 1973.
11. Fagin, R., " Multivalued Dependencies and a New Normal Form for

Relational Databases, " ACM Trans. Database Syst., Vol. 2, No. 3, pp.262-278, Sept. 1977.

12. Fagin, R., " The Decomposition versus Synthetic Approach to Relational Database Design, " in Proc. 3rd Int. Conf. on Very Large Data Bases, Tokyo, Japan, Oct. 1977.

13. Date, C.J., An Introduction to Databases Systems, Vol.1, Fourth Edition, Addison-Wesley Publishing Company, Reading, Massachusetts, 1986.

14. Hammer, M. and D. Mcleod, " Database Description with SDM: A Semantic Database Model, " ACM Trans. Database Syst., Vol.6, No.3, pp.351-386, Sept. 1981.

15. Chung, I., F. Nakamura and P. P. Chen, " A Decomposition of Relations Using the Entity-Relationship Approach, " Entity-Relationship Approach to Information Modeling and Analysis, ER Institute, pp.151-173, 1981.

16. 朱阿水，" 關聯式邏輯資料庫設計工具 "，臺灣，國立臺灣工業技術學院工程技術研究所電子組碩士論文，民國七十五年六月。

17. Zaniolo, C. and M. A. Melkanoff, " On the Design of Relational Database Schemata, " ACM Trans. Database Syst., Vol.6, No.1, pp.1-47, Mar. 1981.

18. Armstrong, W. W. and C. Delobel, " Decompositions and Functional Dependencies in Relations, " ACM Trans. Database Syst., Vol. 5, No.4, pp.404-430, Dec. 1980.

19. Lien, Y. E., " Hierarchical Schemata for Relational Databases, " ACM Trans. Database Syst., Vol.6, No.1, pp.48-69, Mar. 1981.

20. Paredaens, J. and D. Janssens, " Decompositions of Relations: A Comprehensive Approach, " Advances in Database Theory, Vol.1, Plenum Press, New York, pp.73-100, Sept. 1980.

21. Rissanen, J., " Independent Components of Relations, " ACM Trans. Database Syst., Vol.2, No.4, pp.317-325, Dec. 1977.

22. Maier, D., A. O. Mendelzon, F. Sadri and J. D. Ullman, " Adequacy of

Decompositions of Relational Databases, " Advances in Database Theory, Vol.1, Plenum Press, New York, pp.101-114, Sept. 1980.

23. Ullman, J. D., Principles of Database Systems, Second Edition, Pitman Publishing Limited, London, 1982.

24. Maier, D., A. O. Mendelzon and Y. Sagiv, " Testing Implications of Data Dependencies, " ACM Trans. Database Syst., Vol.4, No.4, pp. 455-469, Dec. 1979.

25. Beeri, C., " On the Membership Problem for Functional and Multivalued Dependencies in Relational Databases, " ACM Trans. Database Syst., Vol.5, No.3, pp.241-259, Sept. 1980.

26. Maier, D., The Theory of Relational Databases, Pitman Publishing Ltd., London, 1984.

27. Beeri, C. and P. A. Bernstein, " Computational Problems Related to the Design of Normal From Relational Schemas, " ACM Trans. Database Syst., Vol.4, No.1, pp.30-59, Mar. 1979.

28. Batini, C. and M. Lenzerini, " A Methodology for Data Schema Integration in the Entity-Relationship Model, " IEEE Trans. Software Eng., Vol. Se-10, No.6, pp.650-662, Nov. 1984.

29. Jajodia, S., P. A. Ng and F. N. Springsteel, " The Problem of Equivalence for Entity-Relationship Diagrams," IEEE Trans. Software Eng., Vol. Se-9, No.5, pp.617-630, Sept. 1983.