TR-88-009

基於語意模型之資料庫管理系統設計

基於語意模型之資

料庫管理系統設計

主持人：郭德盛

研究助理：洪炯宗

隋寶華

劉伴煌

中央研究院資訊科學研究所

INSTITUTE OF INFORMATION SCIENCE

ACADEMIA SINICA

中華民國七十七年六月

摘　要

關聯型資料庫系統是目前被廣泛使用的一種資料庫系統，但它仍然有許多缺點，語意的缺乏是最嚴重的一項。本文提出一種語意模型，它是根據RM/T 模型改良而來，它能讓使用者處理更多的語意。

目前所有俱備實體觀念的資料模型在定義實體類型之間的關係時，僅說明實體之間有何種對應關係存在而已。本文則認爲在定義特性實體類型時，除了說明其依賴那一種實體類型外，仍須說明特性實體類型的重覆性，來限制特性實體出現的次數；在定義結合實體類型時，除了指出結合的參與者外，也必須說明參與者的連接性及重覆性，如此，不僅明白的表示出參與者之間的對應也能限制實體出現的次數，進而表示出更多的語意。

在許多系統中都採用了代理的觀念，以代理爲實體的永久識別字。此舉雖改善了人爲識別字的缺點，但卻沒有人針對此一特性提出運算子。本文則提出一種根據代理值運作的基本運算子，此運算子不像關聯運算子僅是爲了資料處理，它能讓使用者在以代理爲實體識別字及看不見代理值的環境下，執行一些語意處理的動作。另外，爲了使用者的方便，

本文亦提出巨運算子。以上兩種運算子都能以關聯運算子建立，證明確實可在關聯系統上執行。結構查詢語言（ SQL ）雖然其結構很簡潔，但不能很直接的表示出使用者的想法，使用者常須使用到子查詢。本文則提出一種能很自然、很直接表示出使用者想法的資料處理語言，它能在很多情形下讓使用者省略子查詢的使用。

本文所提的語意模型實際上是建立在關聯系統之上。我們利用關聯模型定義所提出的語意模型，也就是完全利用關聯來建立資料典以描述本文所提出的資料模型。本文所定義之資料庫系統是採用自述方式。

ABSTRACT

Relational database system is widely used today,it has many shortcomings,semantic scantiness is the most serious problem. A semantic data model,based on RM/T is proposed,it can handle more semantic than RM/T does in some aspects.

All data models with notion of entity specifying mapping only when defining relationship. Defining a characteristic entity type must specify not only the entity type it described,but also specify the multiplicity of the characteristic entity type is supposed in this thesis. Also,in defining a associative entity type most specify not only the associative entity types in association,but also specify the connectivity and multiplicity of each participant is suggested in this thesis. A result of defining multiplicity and connectivity is that more semantic can be handled.

Many systems use surrogate as permanent entity identifier,though this removed the disadvantages caused by user-defined primary key,it is not fully utilized by system. A set of primitive operators,whose operations are based on surrogate values,is proposed. User can use primitive operators perform semantic manipulation. Macro operators,for user convenience,is also introduced. Both macro operators and primitive operators can be

implemented in terms of RDB operators,showing that they can be performed in a relational system. Although SQL is easy to use,it can't express user's thoughts naturally, subquery is often used in SQL. A data language which can express user's thoughts directly and naturally is introduced,subquery will not be necessary in some circumstances.

The semantic data model proposed is built on a relational system. We define semantic data model by relational data model,i.e.,the dictionary of the semantic database system is defined by relations. The semantic database system is self-describing.

目　　錄

第一章　緒　論

隨著時代的進步，人們處理的資料量愈來愈大，爲了快速及正確的處理資料，傳統的檔案系統已不能滿足人們的要求，代之而起的是目前大家所熟知的資料庫系統。

資料庫的定義，根據〔4〕資料庫是某一特定企業內應用系統所使用的運作資料的集合。此處的特定企業可以是製造工廠、銀行、醫院、學校或者政府部門等，而其所使用的運作資料可能是產品資料、會計資料、病歷資料、學生資料或是計劃資料等，但這些資料並不包含輸入、輸出及暫時性資料。以一間製造工廠爲例，除了有關作業員、零件、計劃等這些基本資料以外，作業員、零件及計劃之間的關係也可視爲工廠的運作資料而加以儲存。

一般而言，資料庫具有下列優點：

(1)減少重覆資料：

在非資料庫系統，每一種應用都有其專屬的檔案，以致造成儲存之資料有相當的重覆，而浪費了儲存空間。爲了減少資料重覆及不浪費儲存空間，基本上，相同資料在資料庫中僅存一份。也有時候爲了某些因素，資料也可能重覆，這些重覆資料則須非常小心的控制。

(2)避免資料不一致：

如果每一筆資料只存一份則不會有資料不一致的現象發生。當資料有重覆儲存時，任何資料的更正，其重覆之資料亦必須隨之更正，則所有重覆的資料將一致。

(3)資料共用：

資料不僅可被現存的應用系統共用，新的應用系統也可以使用，而不需要額外的儲存資料。

(4)資料標準化：

由於資料庫集中管理，資料可以標準化。資料標準化對於資料交換，合併都非常重要。

(5)資料保密：

因爲資料庫的資料是大家共用，但並非每個人都可以使用任何一筆資料，有的資料僅提供給某些特定使用者使用。資料庫系統爲了達到上述功能，提供許多措施，以確保資料的保密。

(6)保持資料的完整性：

完整性的問題在於確定資料庫內的資料是正確的。爲了完整性，所有進入資料庫的資料或更正的資料都將經過完整性的檢查，來確保資料的正確。

(7)資料獨立性：

資料獨立就是應用不受資料儲存結構、存取方式的改變而有所影響。其主要原因在於資料庫系統的結構，根據ANSI / SPARC〔 21 〕的報告，資料庫的結構可分爲三層（ 圖(1) ）：

①外部階層（ EXTERNAL LEVEL ）；

②觀念階層（ CONCEPTUAL LEVEL ）；

③內部階層（ INTERNAL LEVEL ）.

所有下一層結構敍述的改變經轉換軟體，可以使上一層不受影響。

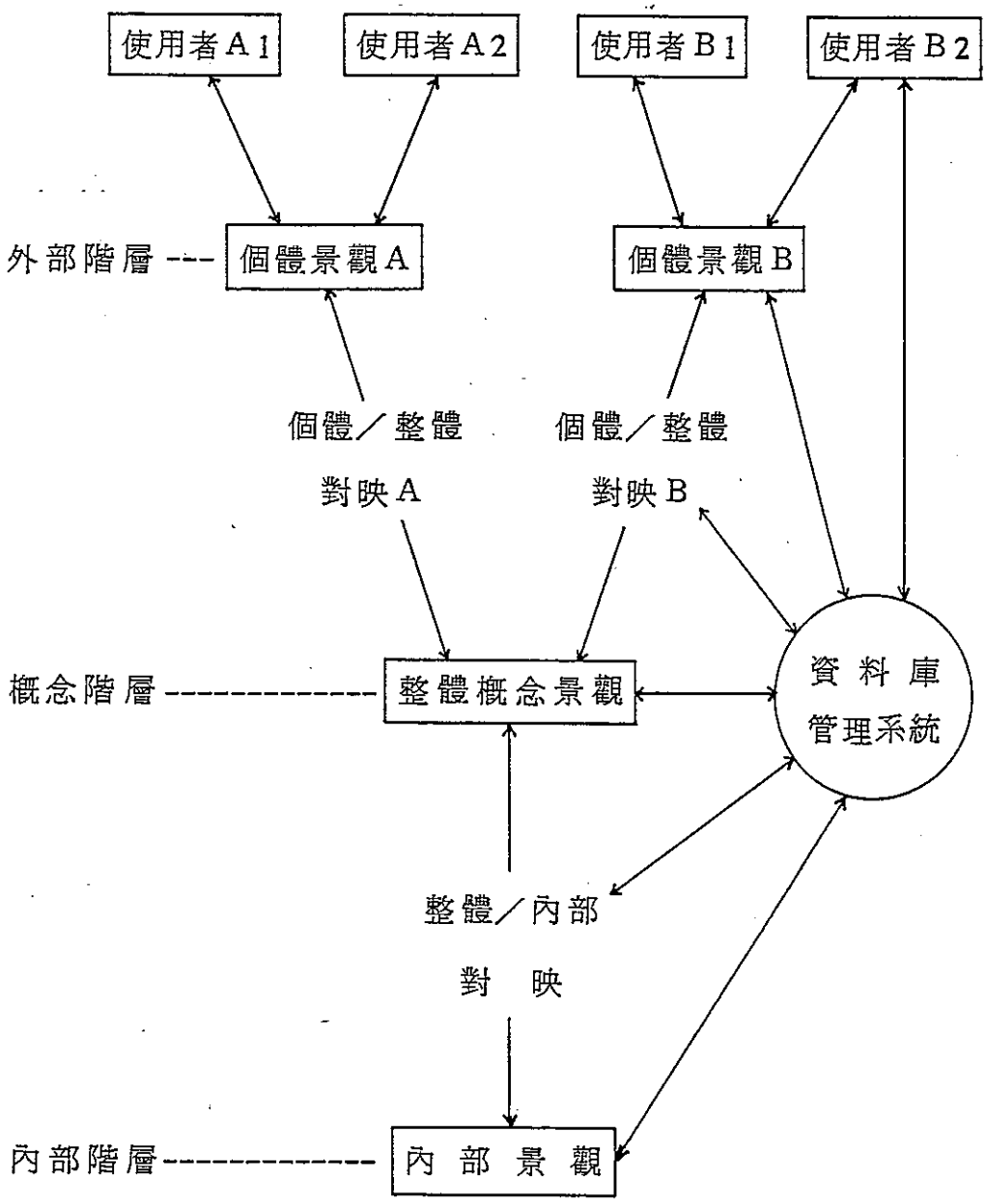

圖1　典型資料庫系統結構

根據資料庫系統所使用的資料結構（ DATA STRUCTURES ）及提供給使用者的運算子（ OPERATORS ），資料庫大致可分成：

(1)關聯型（ RELATIONAL ）資料庫系統；

(2)層次型（ HIERARCHIC ）資料庫系統；

(3)網狀型（ NETWORK ）資料庫系統。

就以上三種類型的資料庫系統之中，以關聯型最受人們歡迎而廣被使用。最主要是因爲關聯型資料庫使用了大家所熟悉的表格爲其資料結構，以及它提供了使用者簡單、易學、高階層的資料處理語言。

資料庫系統雖然有前面所提到的這麼多好處，它仍然也有一些能力限制。以目前大家最常使用的關聯型資料庫而言，它的最大缺點就是語意（ SEMANTICS ）的缺乏。例如：工程師是雇員的子集合，電子工程師和機械工程師又是工程師的子集合，這類的關係如何明白的表示？另外，雇員的小孩與雇員之間附屬的關係又如何表示？這些都是關聯型資料庫在語意方面的弱點。因爲這方面的缺點，有許多人提出了一些新的資料模型（ DATA MODEL ）及觀念，希望在語意的表示及處理能力能有所改良。以下我們將簡單討論一些探討語意方面的報告。

(1)基本語意模型(THE "BASIC SEMANTIC MODEL")

Schmid 和 Swenson 兩人在語意探討的領域中是較早期的工作者，他們提出了基本語意模型(BASIC SEMANTIC MODEL)〔13〕。在他們的想法中，世界包含了物體(OBJECT)及結合(ASSOCIATIONS)。物體又可以區分爲獨立物體(IN-DEPENDENT OBJECT)及依賴物體(DEPENDENT OBJECT)兩種。這兩者差別是依賴物體必須存在依賴(EXISTENCE-DEPENDENT)某獨立物體，而獨立物體則可以單獨存在，不須依賴其它物體。例如，在一人事資料庫中，雇員可能是獨立物體，而雇員的小孩則可能是依賴物體，這表示雇員的小孩被記錄在資料庫之內的先決條件是雇員必須被記錄在資料庫之內。

根據關聯(RELATION)所表示的訊息類別，關聯被分成五種類型。例如，類型一用來表示獨立性物體，類型五則用來表示結合。因爲基本語意模式的基本資料結構乃是關聯，所以 Schmid 和 Swenson 並沒有提出新的運算子，乃採用一般關聯型運算子。

基本語意模型有一個特性，物體與結合被認爲是不同的東西。物體可以有屬性(PROPERTY)，結合不可以有屬性；物體可以參與其它的結合，結合本身則不能再參與其它的結合。因爲這個特性使得有些語意不易表示，而成爲它在語

意表示方面的一個缺點。

(2)實體—關係模型(THE ENTITY-RELATIONSHIP MODEL)

實體—關係模型(E-R MODEL)〔3〕是 PIN-SHAN CHEN 於1976年提出。它的基本觀念與基本語意模型的觀念所差有限。 E-R model 認爲世界包含了實體(ENTITIES)和關係(RELATIONSHIPS)。它是根據集合原理(SET THEOREM)發展出的一套能達到高度資料獨立性的模式。

實體是一可以清楚辨認的事物。例如，特定的人，公司或者事件都是一個實體。實體可分成正規(REGULAR)實體及弱(WEAK)實體。關係則是實體之間的結合。例如，"父親—兒子"則是一種二個"人"這種實體之間的關係。在 E-R model 中，實體與關係都可以擁有屬性。這一點雖然比基本語意模式好，但是，關係仍然不可以參與其他的關係，這也是 E-R model 的一項缺點。

(3)資料抽象(DATA ABSTRACTION)

Smith 和 Smith 在1977介紹了資料抽象〔8〕的概念。抽象的目的在於讓使用者僅考慮有關的細節而不考慮無關的細節。資料抽象可分成二種，集合(AGGREGATION)及歸納(GENERALIZATION)。

在集合(AGGREGATION)的概念下，物體之間的關係

被認爲是一種較高層次的物體。例如，結合也可被視爲物體的一種。因爲這個概念使得結合本身也可以參與其它的結合，使資料庫系統在語意方面前進不少。

在歸納（ GENERALIZATION ）的概念下，一組相同的物體可以被視爲一個較高層次的物體，歸納可分成兩種。其中之一，由特定的物體歸納出所有那種類型的物體。事實上，在基本關聯模式中集合同一種格式的資料到一個關聯之中便完成此類的歸納。另一種歸納的觀念是由不同類型的物體歸納出一種共同、高一層次的物體。例如，我們可以由國內旅行與國外旅行歸納出旅行這種物體。因爲歸納的觀念在資料庫系統中才有了超類型（ SUPERTYPE ）及子類型（ SUBTYPE ）的想法。由前面的討論可以知道，資料抽象對於資料庫在語意方面的提昇有很大的影響。

(4) RM / T 模型

在 Codd 所提的擴充關聯型模型 RM / T〔 7 〕之中，假設世界上的任何事物都可以用實體（ ENTITIES ）表示。例如，某一特定的人、公司、關係都是一個實體。所以RM/T 中的實體實際上包括了 E-R model 中的實體及關係。因爲任何一種關係也被視成一種實體，所以 RM/T 沒有 E-R model 在語意上限制關係不可參與其它關係的缺點。實體共分成中心（

KERNEL ）實體，特性（ CHARACTERISTIC ）實體及結合（ ASSOCIATIVE ）實體三種。另外，RM/T 中也加入了超類型／子類型（ SUPERTYPE/SUBTYPE ）的觀念。因此，RM/T 模式幾乎已包括了基本語意模式、E-R 模型及資料抽象在語意方面的能力。雖然RM/T模型幾乎已包括了基本語意模型、E-R 模型及資料抽象觀念在語意方面的表示及處理能力，它仍然有一些缺點。例如，當特性實體的個數受到限制時，在RM/T 中如何表示？結合實體類型中參與者之間的對應如何表示？參與者出現的次數又如何加以限制？這種對個數、次數加以限制的語意在RM/T 都未討論，也就無法表示此類語意。RM/T 模型定義了一些運算子，但這些運算子並未充份的運用RM/T 本身所提到的代理，以致使用者也無法利用此一特性，而且這些運算子也不俱備語意處理能力。以上所提RM/T的缺點也就是本文認爲應該加以改進的地方。

本文所提的資料模型係由RM/T 模型改良而來。我們強調使用者在定義特性實體類型時，除了說明特性實體類型是描述那一種實體類型外，仍須限制特性實體出現的次數，因此使用者必須定義特性實體的重覆性。經由定義重覆性的動作達到使用者限制特性實體出現最多次數的語意。在結合實體類型中，使用者不僅須指明結合的參與者爲那些實體類型，

更須進一步說明參與者的連接性〔20〕及重覆性。因爲在眞實世界中，在某些情況下結合的參與者之間會有一定的限制，如果不指明這些限制，便無法眞正的表示出結合實體的性質。定義了參與者之間的連接性後，系統便能決定參與者之間對應次數是否超過限制；定義參與者的重覆性後，系統就有一定的根據來限制實體出現的次數。有了連接性及重覆性，才算完整的表示出結合實體的特性。本文所提的資料模型是以代理爲實體的永久識別字，代理值使用者是無法控制的。針對此特性，我們提出一套完全根據代理值運作的基本運算子，它俱有語意處理能力。例如：找出同時擁有二種身份的實體，或找出擁有某一種身份或另一種身份，但不同時俱備兩個身份的實體，以上動作都可以利用基本運算子完成。在本文中，我們也提出如何利用關聯型系統來建立一個語意模型〔9，22〕，以增加系統處理語意的能力。

這篇報告共分六章。第一章爲緒論。第二章討論資料模型，對於我們所提出資料模型的運算子將在第三章中加以說明。第四章討論系統架構及資料典（DATA DICTIONARY）。第五章的重點爲資料語言，分爲資料定義語言（DATA DEFINITION LANGUAGE）及資料處理語言（DATA MANIPULATION LANGUAGE）。最後一章爲結論。

第二章　資料模型

一種資料模型基本上包含三大部份：

(1)一組物體類型(OBJECT TYPES)；

(2)一組運算子；

(3)一組完整性規則 (INTEGRITY RULES) 。

物體類型是資料模型的基本建造元件。換言之，任何滿足某種資料模型之資料庫的邏輯結構必須是完全由此資料模型的物體類型所構成。例如，關聯模型〔 10 〕的物體類型爲關聯(RELATIONS) 及定義域(DOMAINS)，則關聯型資料庫僅可由關聯及定義域構成。運算子則提供處理資料庫的方法。完整性規則是爲了限制資料庫的內容，以確保資料庫內資料的正確。

在這一章中我們詳細討論一種由RM/T 模型改良形成的資料模型。與RM/T 最主要的差別在特性實體(CHARACTERISTIC ENTITY) 及結合實體(ASSOCIATION ENTITY) 增加一些更能處理語意的能力。

在這種資料模型下基本假設是任何有形、無形的東西都可以視爲實體(ENTITY) 。實體在資料庫中是以實體關聯(E-RELATION) 及屬性關聯(P-RELATION) 表示，實體關聯及

屬性關聯都是關聯的一種特別型式。實體關聯是記載實體的存在，屬性關聯則是用來記錄實體的屬性。實體之間也有各種不同的關係存在。例如，幾種實體可以經由結合（ ASSOCIATION ）連在一起；一種實體類型可爲另一種實體類型的子類型（ SUBTYPE ）；一種實體類型也可以有另一種實體類型附屬於它。以上所提的各種特性將在下面各節中討論。

2-1 實體（ ENTITIES ）

如同前面所說的，任何東西都可以視爲實體。但要注意一點，實體的觀念是相對的，並非絕對的。因爲有些事物被某一羣人認爲是實體，但可能被另一羣人認爲是實體的屬性。基本上，我們可以說實體是任何可以區別的物體。這所說的＂物體＂可以是實在的（ CONCRETE ），也可以是抽象的（ ABSTRACT ）。此節所敍述實體觀念係採用 RM/T 模型對實體之敍述。

實體可以區分成不同的實體類型（ ENTITY TYPES ）。例如，每一名勞工都是＂勞工＂這種實體類型的一個實例（ INSTANCE ）；每一個父—子關係都是＂父—子＂這種結合實體類型的一個實例。區分不同的實體類型可以使資料庫的結構比較簡單，因爲同類型的實體都有一些相同的屬性必須加

以記載。例如，所有的學生都有學號、姓名、年級等屬性。同類型的實體集中在一起表示，就可以用一種格式的關聯來表示。

實體除了可以區分成不同類型，也可以根據其性質加以區分成特性實體（ CHARACTERISTIC ENTITIES ）、結合實體（ ASSOCIATION ENTITIES ）及中心實體（ KERNEL ENTITIES ）〔 7 〕。其定義如下：

特性實體：一個特性實體也是一個實體，它主要是用來描述另一個實體。例如，訂購單上的每一單項訂購可以是一個特性質體，用來描述＂訂購單＂這個實體。

結合實體：結合實體也是實體，它主要是表示許多實體之間一種關係。例如，一個船運（ SHIPMENT ）表示一個供應者（ SUPPLIER ）與一種零件（ PART ）之間的關係。

中心實體：中心實體也是實體，它既不是特性實體，也不是結合實體。例如，訂購單、供應者及零件都可以是中心實體。

中心實體才是一個資料庫眞正記載的重心。雖然結合實體的參與者（ PARTICIPANT ）並不一定是中心實體，它卻是

由連接各種實體而形成的一種實體；一個特性實體所依賴的對象並不限制是中心實體，但它總是必須依賴另一個實體才可以存在資料庫之中。

一種實體類型X可以有一種或多種的子類型（ SUBPYPES ）X1，X2，X3，……。當實體類型X1的每一個實體都是實體類型X的一個實體，我們就叫X1這種實體類型是X實體類型的子類型。例如，〝電子工程師〞及〝機械工程師〞這兩種實體類型都是〝工程師〞這種實體類型的子類型，因爲每一位電子工程師及每一位機械工程師都是一工程師。當實體類型X1是實體類型X的子類型，則實體類型X便是實體類型X1的超類型（ SUPERTYPE ）。不論是中心實體類型、特性實體類型或結合實體類型都可以有子實體類型（ SUBENTITY TYPES ）。子實體類型繼承了超實體類型的性質。換言之，中心實體類型之子實體類型的性質乃爲中心實體類型；結合實體類型之子實體類型仍是結合實體類型；特性實體類型的子實體類型仍是特性實體類性。

不同實體類型有可能會互相重疊（ OVERLAP ）〔 8 〕。例如，某一特定人可以是公司的雇員，也可以同時是公司的股東。則〝雇員〞這種實體類型就與〝股東〞這種實體類型互相重疊。因爲在眞實世界中，這種同時擁有幾種身份的情

形是可能的，所以在資料庫中我們也必須考慮此項事實。一個實體同時屬於二種實體類型時，在資料庫中會有什麼影響？下一節我們將討論如何解決此現象。

2-2 代理（ SURROGATES ）

在關聯型資料庫中，每一個實體都是以關聯（ RELATION ）中的一個維（ TUPLE ）來表示，如圖2-1 。其中 SUPPLIER 這個關聯有五個維，就是它的基數（ CARDINALITY ）爲五，也就表示有五位供應商。PART 及 SHIPMENT 的基數均爲四，表示有四種零件及四次船運。在眞實世界中實體是可以區別的，所以它們在資料庫中的表示也必須要可以區別。在關聯型資料模型中，這種辨認的工作是經由使用者定義及控制的主鍵（ PRIMARY KEY ）來完成。如圖2-1 中，供應商是由 S# 來辨認；零件是根據P# 來辨認，S# 及 P#的値是使用者決定。辨認工作經由使用者所定義的主鍵來完成可能會有某些缺點。因爲主鍵是使用者所決定，它們很容易被更改。而這些更改在資料庫中可能會引起一些錯誤。例如，當供應商的 S#由S5 改成 S6 時，SHIPMENT 這個關聯所存的資料便有問題。此時 SP# 爲 SP3 的船運卻仍然使用舊的對外鍵（ FOREIGN KEY ）" S5 "，而造成資料的錯誤。另外，使用者可能會用不同的鍵

SUPPLIER

S#	SNAME	STATUS	CITY
S1	SMITH	20	N.Y.
S2	JOHN	30	N.Y.
S3	MARY	20	T.P.
S4	TOM	20	T.P.
S5	PETER	30	N.Y.

PART

P#	PNAME	WEIGHT	CITY
P1	PN1	20	N.Y.
P2	PN2	30	T.P.
P3	PN3	100	N.Y.
P4	PN4	10	T.P.

SHIPMENT

SP#	S#	P#	QTY
SP1	S1	P1	100
SP2	S1	P2	100
SP3	S3	P2	200
SP4	S4	P3	200

圖 2-1 a SUPPLIER-and-PART database

來辨認實體。例如，有些情況下，使用者利用S#來辨認供應商，在另外一些情況下，使用者利用供應商的名字來辨認供應商，而造成資料的混亂、錯誤。

代理（ SURROGATE ）便可以用來解決上面那些問題。本文對代理的定義、使用與RM/T模型一致。代理是系統產生的實體識別字（ IDENTIFIER ）。對每一個被記錄在資料庫的實體，都有一個代理與之對應。每次使用者插入（ INSERT ）一新的實體到資料庫中，系統便會產生一新的代理，此代理的值與資料庫中所有使用中及使用過的代理值都不相同。代理值是永遠不會改變的，它永遠對應著它所代表的實體。使用者雖然可以讓系統產生新的代理及毀掉代理（刪除實體），但使用者卻看不到代理也無法控制代理的值，也就無法改變代理值。根據使用者對資料庫熟悉的程度，有的使用者甚至不知道有代理的存在，代理完全由系統控制。因爲代理是用來辨認實體，所以在資料庫中，實體的辨認及實體參考（REFERENCE ）都是經由代理來完成，換言之，在資料庫中主鍵及對外鍵都是代理。

代理值的定義域稱之爲實體定義域（ E-DOMAIN ）。任何定義於實體定義域的特性（ ATTRIBUTE ）我們稱之爲實體特性（ E-ATTRIBUTE ）。我們將以＂@＂符號當成實體定義域的

名字。另外，所有實體特性名字的最後一個符號也必須是〝@〞。上面的規定只是爲系統操作的方便。

在資料庫中，每一種實體類型都有一個實體關聯（E-RELATION ）。它是一個一度（ ONE DEGREE ）的關聯，用來記錄目前在資料庫中屬於此種實體類型之實體的代理值。通常一種實體類型的名字與其實體關聯的名字一樣；實體關聯唯一特性的名字比實體關聯多了〝@〞符號。

我們所定義的資料模型除了關聯模型的二項完整性規則（ INTEGRITY RULE）外，仍有其它的完整性規則，將隨各章節的內容依順介紹。

關聯模型的二項完整性規則如下：

完整性規則一：

主鍵值的組成成分不能是虛有值。

完整性規則二：

假如基礎關聯 R2 包含了一個能配合另一個基礎關聯 R1 的主鍵之對外鍵，則 R2 中每一個對外鍵必須是(1)等於中某一維的主鍵值，或(2)虛有值。

因爲實體關聯是用來存代理值，代理值又是永久性的實體識別字，不容許使用者更改，所以針對實體關聯的完整性規則如下：

完整性規則三〔 7 〕：

實體關聯允許插入和刪除，但不允許更新。

實體關聯當然也不允許虛有值（ NULL ）的存在。

下面我們舉一個有關代理和實體關聯的例子。

假設我們要建立一個供應商及零件的資料庫，首先我們要告知系統有三種實體類型存在：

```
CREATE ENTITY TYPE SUPPLIER;
CREATE ENTITY TYPE PART;
CREATE ASSOCIATIVE ENTITY TYPE SHIPMENT
        PARTICIPANT(SUPPLIER:S@: ...,
                    PART:P@:...);
```

（以上三個指令的語法是按照本文所定義的資料定義語言（ DATA DEFINITION LANGUAGE ）而寫，資料定義語言將在後面章節加以介紹）。上面三個指令執行的結果將產生 SUPPLIER 、 PART 及SHIPMENT 三個空的實體關聯，它們特性的名字分別爲 SUPPLIER@ 、 PART@ 及 SHIPMENT@ （因爲 SHIPMENT 爲一結合實體，系統同時還會爲SHIPMENT 建立一個屬性關聯SHIPMENT-A-SGATE ，它的特性分別爲 SHIPMENT@ 、 S@ 及 P@ ，其中 S@ 、 P@ 是使用者定義）。此刻使用者則可以利用插入運算子使系統對特定的 SUPPLIER 、 PART 及SHIPMENT 產生代理（ 插入運

算子將在後面介紹）。

在前面我們曾經允許一個實體可以同時屬於幾種實體類型。例如，當我們插入（ INSERT ）一位公司的雇員，他又同時是公司的股東，則此雇員的代理值必須同時出現在＂雇員＂及＂股東＂這兩個實體關聯之中。國內旅行和國外旅行都是旅行的子類型，則不論我們插入一個實體到國內旅行或國外旅行，系統都會產生一個新的代理值，而將代理值插入＂國內旅行＂或＂國外旅行＂實體關聯之中，同時將新的代理值插入＂旅行＂這個實體關聯之中。所以一個實體不論其同時屬於多少種實體類型，它的代理值只有一個。

2-3 屬性（ PROPERTIES ）

在此所謂的屬性是指實體立即的（ IMMEDIATE ），單值的（ SINGLE-VALUED ）屬性，也就是RM/T 模型的立即屬性。對於結合實體類型及特性實體類型的屬性關聯產生則有不同方式。一實體的立即屬性僅爲此實體的屬性，而不包含其特性實體的屬性。例如，供應商號碼、供應商姓名、狀況、城市都是供應商的屬性。多值（ MULTIVALUED ）屬性在後面一節將會討論。

一種實體類型的屬性將以一組屬性關聯（ P-RELATIONS ）

來表示。以供應商這種實體類型而言，我們可能有三個屬性關聯如下：

SKN(SUPPLIER@,S#,SNAME)

ST(SUPPLIER@,STATUS)

SC(SUPPLIER@,CITY)

至於如何把那些屬性集中在一個屬性關聯則是資料庫設計者的責任。把全部屬性集中在一個屬性關聯或每一個屬性在一個屬性關聯都有可能。無論如何安排屬性，這一組屬性關聯必須滿足下面條件：

(1)這一組屬性關聯中的每一個屬性關聯之主鍵名字與此實體類型的實體關聯之特性名字一樣。

(2)除了(1)中所說的以外，其它特性都不能有相同的名字。

前面所提供應商的三個屬性關聯可經由以下指令定義：

```
CREATE P_RELATION SKN FOR SUPPLIER
        (S#: SNO,
         SNAME: SNAME);
CREATE P_RELATION ST FOR SUPPLIER
        (STATUS: STATUS);
CREATE P_RELATION SC FOR SUPPLIER
        (CITY: LOCATION);
```

（假設其中的 SNO, SNAME, STATUS, LOCATION 這些定義域已經定義過了。）執行以上三個 CREATE P-RELATION 的指令後，系統便已爲＂供應商＂實體類型產生了三個屬性關聯，不過到目前爲止它們 是空的。在 SKN ， ST 及 SC 這三個關聯屬性中都有一個 SUPPLIER@ 的特性，但我們在前面指令中並未指出有 SUPPLIER@ 這個特性。因爲在 CREATE P-RELATION 中，我們借著＂FOR SUPPLIER ＂指出這些屬性關聯是屬於 SUPPLIER 這種實體類型，則系統自動在每個屬性關聯中加入＂SUPPLIER@ ＂這個特性。

此刻使用者便可以利用插入指令把供應商的資料記錄在屬性關聯之中。例如：

```
INSERT
INTO  SUPPLIER(S#,SNAME,STATUS,CITY)
VALUES("S1","SMITH",20,"LONDON");
```

系統便會產生一個新的代理，假設爲 alpha ，而將之插入SUPPLIER 這個實體關聯；再將 alpha， S1 ，SMITH 插入 SKN 這個屬性關聯；再將 alpha ， 20 插入 ST 這個屬性關聯；再將 alpha ， LONDON 插入 SC 這個屬性關聯。其結果如圖 2-2 。

對於＂零件＂這種實體類型我們可以定義其屬性關聯如下：

E_RELATION:

SUPPLIER
SUPPLIER@
alpha

P_RELATION:

SKN

SUPPLIER@	S#	SNAME
alpha	S1	SMITH

ST

SUPPLIER@	STATUS
alpha	20

SC

SUPPLIER@	CITY
alpha	LONDON

圖 2-2

E_RELATION:

PRAT
PART@
beta

P_RELATION:

PKN

PART@	P#	PNAME
beta	P1	PN1

PW

PART@	WEIGHT
beta	100

PC

PART@	CITY
beta	N.Y.

圖 2-3

```
CREATE P_RELATION PKN FOR PART
(P#: PNO,
 PNAME: PNAME);
CREATE P_RELATION PW FOR PRAT
(WEIGHT: WEIGHT);
CREATE P_RELATION PC FOR PRAT
(CITY: LOCATION);
```

結果如圖2-3。（圖2-3中假設已有一種零件存在。）

至於船運之屬性關聯的定義如下：

```
CREATE P_RELATION SPQ FOR SHIPMENT
(QTY: QUANTITY);
```

雖然我們爲"船運"這種實體類型僅定義了一個屬性關聯 SPQ，然而實際上船運卻有二個屬性關聯，分別爲 SHIPMENT-A-SGATE 及 SPQ。其中 SHIPMENT-A-SGATE 是系統在使用者定義"SHIPMENT"這種實體類型時便已產生（參考2-2節）。這種專門爲了參考（REFERENCE）的屬性關聯由系統負責產生，則使用者在定義屬性關聯便不必考慮參考的問題。資料庫中有關船運的部份如圖2-4。

假設我們要對船運插入一個由供應商 SMITH 及零件 PN1 所形成的船運，數量爲 100。我們可用以下的插入指令：

E_RELARION:

SHIPMENT
SHIPMENT@

P_RELATION :

SHIPMENT_A_SGATE		
SHIPMENT@	S@	P@

SPQ	
SHIPMENT@	QTY

圖 2-4

E_RELATION

SHIPMENT
SHIPMENT@
gamma

P_RELATION:

SHIPMENT_A_SGATE		
SHIPMENT@	S@	P@
gamma	alpha	beta

SPQ	
SHIPMENT@	QTY
gamma	100

圖 2-5

```
INSERT
INTO  SHIPMENT(QTY)
VALUES (100)
WHERE  (PARTICIPANT:S.SNAME="SMITH"  AND
        PARRICIPANT:P.PNAME="PN1");
```

系統產生新的代表，假設爲 gamma ，並插入 SHIPMENT 實體關聯中，將 gamma ，alpha ，beta 插入 SHIPMENT-A-SGATE，將 gamma ，100 插入 SPQ 之中，結果如圖 2-5 。在上面插入指令中 WHERE 句子，主要是要系統找出參與 SP 的參與者（ PARTICIPANTS ）的代表值。（參與一個結合實體的實體，我們稱之爲參與者）

一種實體類型可以是另一種實體類型的子類型。例如＂國外旅行＂則是＂旅行＂的子類型。在這種超類型／子類型的關係中，我們規定子類型的屬性不可以與其超類型的屬性有共同的特性。（超類型／子類型在 2-6 節中討論）

因爲在資料庫中，每一個屬性值必須描述一個已存在資料庫中的實體。所以，我們有了完整性規則四。

完整性規則四：

假如一個屬性關聯中，某一個維（ TUPLE ）的主鍵值爲

t ，則 t 必須存在於對應此屬性關聯的實體關聯中。

2-4 特性實體

所謂的特性實體，它主要的目的是用來描述另一個較大（ SUPERIOR ）的實體〔 5 〕，在RM/T模型中，特性實體主要是爲了解決多值屬性的問題，但它也具備了解決依賴實體的特性。特性實體是附屬於，且存在依賴那個較大實體。例如，採購單上的每一採購行都可以視爲特性實體，而用來描述採購單。採購單則可以視爲較大的實體。

我們之所以需要特性實體的原因之一是因爲有些實體可能會有多值屬性（ MULTIVALUED PROPERTIES ）〔 7 〕，而我們所說實體的屬性是指立即的，單值的屬性。特性實體便可以解決多值屬性的問題，也就是在傳統資料處理中大家所熟知的重覆組（ REPEATING GROUP ）問題。RM/T模型雖然利用特性實體解決了多值屬性的問題，但我們認爲還有一些有關特性實體在語意方面的問題仍未解決。例如，一實體所能擁有某一種特定特性實體的個數是多少？到目前爲止，尚未有任何報告討論此問題。

我們以採購單的例子來說明基本關聯模型及RM/T模型如何處理重覆組的問題，以及RM/T未能解決的語意問題。

在基本關聯模型我們將以二個關聯來表示此情況：

```
ORDER(ORDER#,CUSTOMER,DATE,TOTAL_AMOUNT)
        PRIMARY KEY(ORDER#)

ORDERLINE(ORDER#,LINE#,ITEM,QTY,PRICE,AMOUNT)
        PRIMARY KEY(ORDER#,LINE#)
```

採購單的立即的，單值的屬性爲採購單號碼、顧客、日期及總計。採購單的多值屬性則以採購行表示。採購單號碼，採購行號碼、項目、數量、單價及小計則是採購行的單值屬屬性。

在RM/T模型中，採購單將被視爲較大的實體，它的屬性有採購單號碼、顧客、日期及總計。採購行被視爲特性實體，它是存在依賴採購單。使用者使用以下指令定義採購單與採購行之間的關係，以及它們的屬性關聯（爲說明方便，假設所有屬性在一個屬性關聯之中）。

```
CREATE E_RELATION ORDER;
CREATE P_RELATION ORDER_PROPERTIES
                    FOR E_RELATION ORDER
      PROPERTIES(ORDER#        DOMAIN(...),
                 CUSTOMER      DOMAIN(...),
                 DATE          DOMAIN(...),
                 TOTAL_AMOUNT  DOMAIN(...));
```

```
CREATE E_RELATION ORDLINE
                  CHARACTERISTIC OF ORDER;
CREATE P_RELATION ORDLINE_PROPERTIES
                  FOR E_RELATION ORDLINE
      PROPERTIES(SURROGATE:   FOR ORDER,
                  ITEM:       DOMAIN(...),
                  QTY:        DOMAIN(...),
                  PRICE:      DOMAIN(...),
                  AMOUNT:     DOMAIN(...));
```

(此例子摘錄自"An Introduction to Database Systems" C.J. DATE, VOL.II)資料庫的內容如圖 2-6。

如上例，當採購行與採購單的關係除了存在依賴，及重覆組以外沒有其它的限制，RM/T 可以完全解決。但在實際生活中，我們常常會限制特性實體出現的次數。例如，一張採購單的採購行最多可以有八行，則填入第九行時便出現錯誤；自動提款卡使用六次後，便需到櫃台節算，否則第七次使用自動提款卡便無法提款。以上這種"最多八行"及"最多六次"的語意應如何表達？所以我們在定義特性實體類型時，除了指明存在依賴那一種實體類型外，也必須指明其重覆

E_RELATION:

ORDER	ORDLINE
ORDER¢	ORDLINE¢
beta	gamma
	delta

P_RELATIONS:

ORDER_PROPERTIES

ORDER¢	ORDER#	CUSTOMER	DATE	TOTAL_AMOUNT
beta	1	...	...	...

ORDLINE_PROPERTIES

ORDLINE¢	ORDER¢	ITEM	QTY	PRICE	AMOUNT
gamma	beta				
delta	beta				

圖 2-6 （摘錄自〔 5 〕）

性（ MULTIPLICITY ），如此才能完全表示出被依賴實體及依賴實體之間的關係。

```
CREATE ENTITY TYPE ORDER;
CREATE P_RELATION ORDER_PROPERTIES FOR ORDER
       (ORDER#:          ...,
        CUSTOMER:        ...,
        DATE:            ...,
        TOTAL_AMOUNT:    ...);
CREATE CHARACTERISTIC ENTITY TYPE  ODRLINE:8
       OF ORDER;
CREATE P_RELATION ORDLINE_PROPERTIES FOR ORDLINE
       (ITEM:           ...,
        QTY:            ...,
        PRICE:          ...,
        AMOUNT:         ...);
```

我們在定義ORDLINE 實體類型時，不僅標明了ORDLINE 是特性實體類型及生存依賴 ORDER 實體類型，且經由＂：8 ＂表示其重覆性是八。重覆性的指明可有可無，如果在定義特性實體類型時不指出其重覆性，則表示不限制其重覆性。系統可以根據使用者所執行的資料定義語言得知特性實體類型的重覆性，加以記錄在資料典(DATA DICTIONARY)之中，便可

以限制特性實體的出現次數，而達到表示＂最多八行＂及＂最多6次＂的語意。在前面雖然僅爲 ORDLINE 定義一個屬性關聯，而系統卻爲 ORDLINE 產生二個屬性關聯，分別爲 ORDLINE-C-SGATE及 ORDLINE PROPERTIES，如圖 2-7 。

在插入一個特性實體時，因爲所有的實體特性（ E-ATTRIBUTES ）都是系統負責，使用者必須說明此特性實體是生存依賴那一個實體。例如：

```
INSERT
INTO ORDLINE(ITEM,QTY,PRICE,AMOUNT)
VALUES("BIKE",10,500,5000)
WHERE SUPERIOR:ORDER.ORDER#=1;
```

系統根據 WHERE 句子便可以找到較大（ SUPERIOR ）實體的代理值，填入 ORDLINE 實體類型的屬性關聯中。

特性實體類型既然也是一種類型，所以本身也可以有特性實體類型來描述自己，這種性質可以繼續擴展下去。基本上，特性實體類型與被其描述的較大實體類型不會有相同的屬性。

完整性規則五：

除非特性實體所描述的實體存在資料庫中，否則本身無

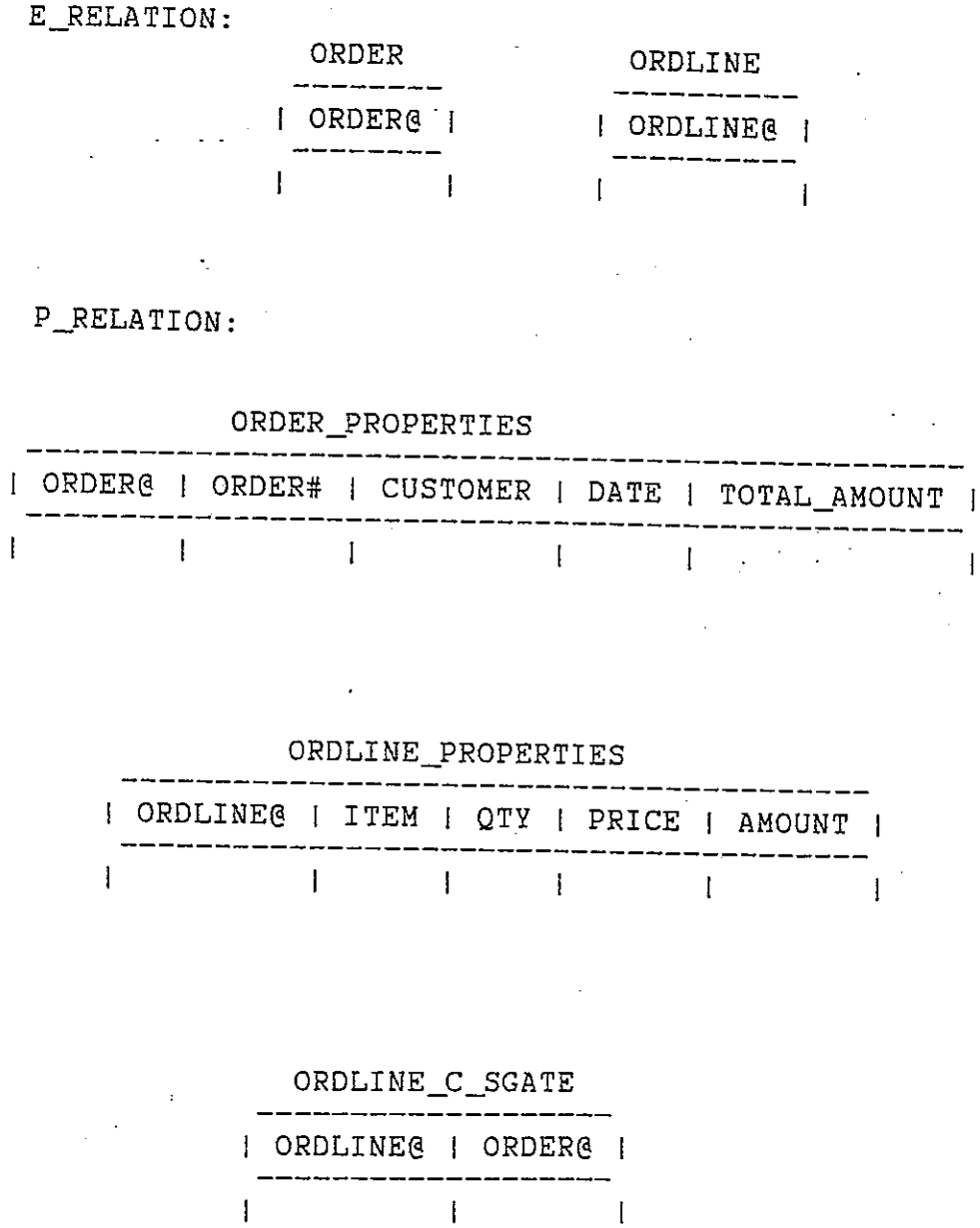
E_RELATION:
ORDER
ORDER@
ORDLINE
ORDLINE@
P_RELATION:
ORDER_PROPERTIES
ORDER@ | ORDER# | CUSTOMER | DATE | TOTAL_AMOUNT
ORDLINE_PROPERTIES
ORDLINE@ | ITEM | QTY | PRICE | AMOUNT
ORDLINE_C_SGATE
ORDLINE@ | ORDER@

圖 2-7

法單獨存在資料庫中。

由完整性規則五我們知道，當被描述的較大實體類型中某一特例被刪除後，描述此一特例的特性實體也將被刪除。

2-5 結合實體 (ASSOCIATION ENTITIES)

結合是多種互相獨立實體之間的一種關係。例如，船運則是由供應商和零件結合的一種關係。參與結合的實體類型稱之爲此結合的參與者(PARTICIPANTS)。在一結合中，參與者之間的對應關係是多對多的對應關係。此處所謂的"多"可以是1，2，……。結合雖然是一種關係，但我們將之視爲一種實體，結合實體。結合既然是實體，它就可以參與其它的結合，它也可以有特性實體來描述自己。同樣的，它也可以有屬性，所以在資料庫中，我們也是以實體關聯及一組屬性關聯來表示結合。在定義結合時，我們除了說明結合的參與者，乃需指明參與者的連接性(CONNECTIVITY)〔20〕及重覆性(MULTIPLICITY)。我們用例子說明連接性及重覆性的重要。在〔20〕中，雖提及連接性，並未說明連接性對語意的重要及應用。下一段說明連接性及重覆性在語意上的作用。

假設車架和引擎合在一起形成一輛車子。則車子是車架

和引擎之間的一種結合。另外，船運是供應商和零件之間的一種結合。雖然車子及船運都是結合，但這種結合的參與者卻有不同的限制。如圖 2-8 ，供應商 S1可以和零件 P1 及 P2 結合，形成船運 SP1 及 SP2。這是因爲供應商 S1 可以供應零件 P1 也可以供應零件 P2 。但引擎 E1 則不能和車身 F1 結合形成車子 C1 ，再去和車身 F2 結合形成車子 C2，因爲車身和引擎之間的對映是 1 對 1 的關係。可知圖 2-8 中，關聯 CAR-A-SGATE 的資料有錯。由上面說明，我們知道在定義一種結合時，使用者必須告訴系統參與者之間的對應是多少，也就是各個參與者的連接性是多少。如上例，引擎及車架的連接性都是 1 。否則此種錯誤系統是無法發現的。由此可知，定義結合實體類型時，說明參與者的連接性是必須的。

僅說明了參與者的連接性是不夠的，在有些情況下，乃無法完整表示結合的各種性質。假設結合 " P-C-S " 是由教授、科目及學生三種實體結合而成，它們的連接性分別爲 1 ，2 和 3 。這表示一位教授最多可以指導二種科目，指導三位學生；每種科目最多有三位學生可以選修；每位學生最多可以選修二種科目。根據教授、科目、學生的連接性，它們之間的結合最多有 6 種可能性。假設教授爲了本身的研究，一週之內只願意指導五次（不論學生與科目的組合）。這種

SUPPLIER

SUPPLIER@
S1
S2
S3
S4

PART

PART@
P1
P2
P3
P4

SP_A_SGATE

SP@	SUPPLIER@	PART@
SP1	S1	P1
SP2	S1	P2
SP3	S3	P3
SP4	S4	P4

ENGINE

ENGINE@
E1
E2
E3
E4

FRAME

FRAME@
F1
F2
F3
F4

CAR_A_SGATE

CAR@	ENGINE@	FRAME@
C1	E1	F1
C2	E1	F2
C3	E3	F3
C4	E4	F4

圖 2-8

"最多只願指導五次"的語意，僅憑連接性是無法表示的。爲了能表示"最多只願指導五次"的語意，則在定義一種結合時，除了說明參與者的連接性外，也必須定義參與者的重覆性。

```
CREATE ASSOCIATIVE ENTITY TYPE P_C_S
        PARTICIPANT( PROFESSOR:P@:1:5,
                     COURSE   :C@:2:3,
                     STUDENT  :S@:3:2);
```

定義了結合實體類型 P-C-S 。參與者教授、科目、學生的連接性分別爲 1，2 及 3；重覆性分別爲 5，3 及 2。如此便表示了教授最多指導五次的語意。如圖 2-9 中，當對 P-C-S-A-SGATE 插入第 6 個 tuples 便錯誤。

P_C_S

P_C_S@
1
2
.
.
.
.

P_C_S_A_SGATE

P_C_S@	P@	C@	S@
1	P1	C1	S1
2	P1	C1	S2
3	P1	C1	S3
4	P1	C2	S1
5	P1	C2	S2
6	P1	C2	S3

圖 2-9

完整性規則六：

一結合實體類型實例存在資料庫中的先決條件是參與此結合實體實例的參與者已存在資料庫之中。

2-6 實體類型（ ENTITY TYPES ）

任何一個實體至少是屬於一種實體類型。例如，一輛特定車子就是＂車子＂這種實體類性的一個實例。一個實體也可以同時屬於幾種實體類型。例如，一位特定雇員可以同時俱有下列身份：

(1)雇員；

(2)經理（經理是雇員的子類型）；

(3)工程師（工程師也是雇員的子類型）；

(4)電子工程師（電子工程師是工程師的子類型）；

(5)股東。

雖然同時有五種身份，但還是同一個人；所以在雇員、經理、工程師、電子工程師及股東這五種實體類型中，它的代理值（ SURROGATE VALUE ）是相同的。在整個資料庫中，是否代表同一實體完全根據代理值是否相同而定。

一種實體類型可以分成許多不同種（ CATEGORIES ）的子類型。如圖2-10之中，經理及非經理形成一種子類型；工程

師、秘書和程式設計員又形成另一種子類型（相對於超類型雇員）。電子工程師和機械工程師形成工程師的一種子類型。超類型中的一個實例最多可以是每個不同種子類型的一個實例。如圖2-10中，一位雇員可以是經理又是工程師，但不能同時是工程師和程式設計員。當超類型中的每一個實例都某種子類型的一個實例時（每一位雇員一定是經理或非經理），我們稱這組子類型分割（PARTITION）此超類型（經理和非經理分割雇員）。

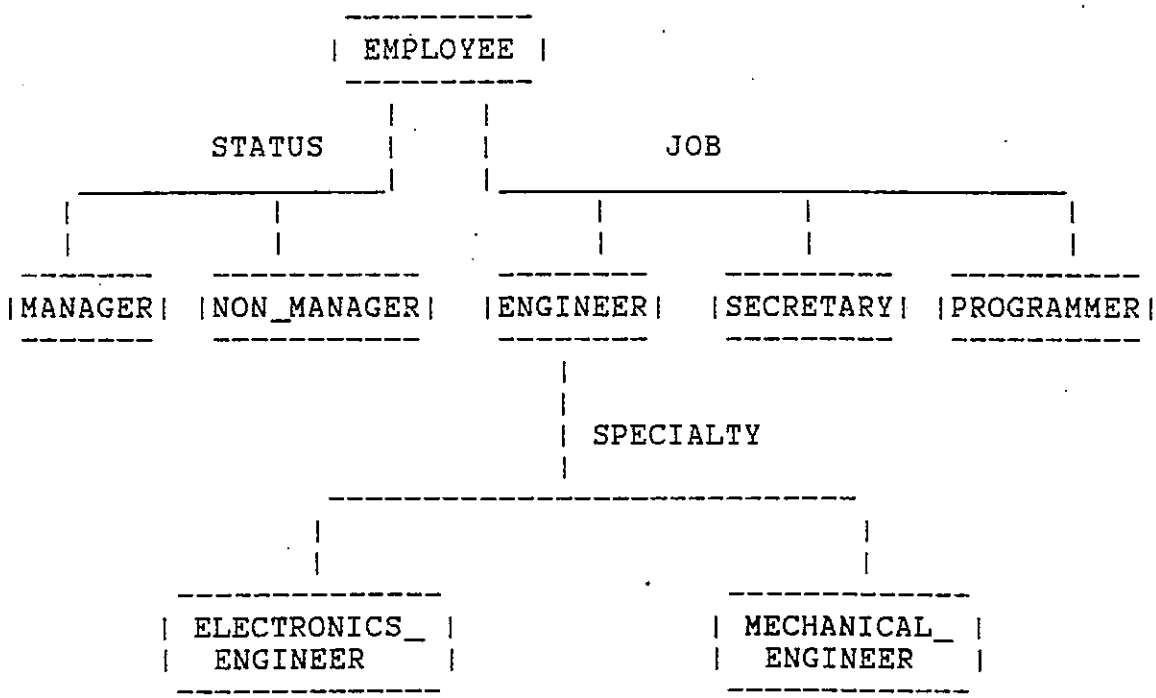

圖 2-10

把實體類型分成超類型及子類型的理由之一是爲了避免虛有值的出現。把實體共同的屬性歸納出形成超類型的屬性，特有的屬性則爲子類型的屬性。例如，姓名是所有雇員（不論是經理或非經理，工程師或秘書或程式設計員）都有的屬性，則把＂姓名＂當成是雇員的屬性；假如＂預算＂是經理人員才有的屬性，而我們把＂預算＂當成雇員的屬性時，則所有非經理人員＂預算＂這項屬性的值都將是虛有值。

對於圖 2-10 中各種實體類型的定義如下（省略屬性關聯的定義）：

```
CREATE ENTITY TYPE EMPLOYEE;
CREATE SPAN CATEGORY STATUS FOR EMPLOYEE;
CREATE CATEGORY JOB FOR EMPLOYEE;
CREATE SUBENTITY TYPE MANAGER OF EMPLOYEE
        PER STATUS;
CREATE SUBENTITY TYPE NON_MANAGER OF EMPLOYEE
        PER STATUS;
CREATE SUBENTITY TYPE ENGINEER OF EMPLOYEE
        PER JOB;
CREATE SUBENTITY TYPE SECRETARY OF EMPLOYEE
        PER JOB;
CREATE SUBENTITY TYPE PROGRAMMER OF EMPLOYEE
        PER JOB;
```

```
CREATE CATEGORY SPECIALTY FOR ENGINEER;
CREATE SUBENTITY TYPE ELECTRONICS_ENGINEER OF ENGINEER
       PER SPECIALTY;
CREATE SUBENTITY TYPE MECHANICAL_ENGINEER OF ENGINEER
       PER SPECIALTY;

DEFINE EMPLOYEE OVERLAP WITH STOCKHOLDER;
```

完整性規則七：

所有子類型實例的代理值都必須存在其所有超類型的實體關聯中。

如同前面所說的，一個實體可以同時屬於不同的實體類型。這些實體類型可以是超類型／子類型的關係，也可以是互相獨立的實體類型。當互相獨立的實體類型有相同的實體時，使用者必須利用 DEFINE OVERLAP 的指令告訴系統那些互相獨立的實體類型相互重疊。例如，

```
DEFINE EMPLOYEE OVERLAP WITH STOCKHOLDER;
```

超類型／子類型一定互相重疊，但使用者不必再告訴系統兩者重疊。因爲在定義子類型時，系統便將兩者的關係記錄在資料典（ DATA DICTIONARY ）中的 SG 關聯之內。

資料模型的物體類型（OBJECT TYPES）及完整性規則在此章中都已介紹，至於運算子將在下一章討論。

第三章　運算子

在前一章中，討論了我們所提出資料模型的物體類型（OBJECT TYPES）及完整性規則。這一章我們將討論運算子。在我們所提出的資料模型的運算子，基本上可分成三大類。因爲我們是在關聯資料模型下發展我們的資料模型，所以關聯資料模型的運算子〔7〕是我們所提運算子的一部份。爲了使用者操作方便，我們還增加了巨運算子（MACRO OPERATORS）。另外，針對我們所定義的資料模型提出了一些基本運算子（PRIMITIVE OPERATORS）。這些基本運算子具有處理語意的能力。在討論運算子之前，我們先討論虛有值（NULL VALUE）〔7〕。因爲運算子執行的結果中可能會有虛有值的出現。

3-1 虛有值

在此節中我們將討論虛有值。在一個屬性關聯中除了主鍵（PRIMARY KEY）及對外鍵（FOREIGN KEYS）（定義在實體定義域之中）外，其它屬性都可能有虛有值的出現。當一個實體的某屬性值爲虛有值時，表示實體的這個屬性值到目前乃不知道或不具備此屬性。本文中有關虛有值的應用及討論

係根據RM/T模型。爲什麼要有虛有值的觀念呢？因爲在眞實世界上有些資料並不完整，但我們卻需要一些處理它們的方法。例如，在登記人事資料時可能會發生"年、月、日不詳"的情形，此時我們便可以用虛有值代替。在本報告文中用"？"符號表示虛有值。

假設 theta 是＝，～＝，＞，＞＝，＜或＜＝這六個比較關係之一時，且 x 或 y 或 x 和 y 兩者的值是虛有值時，運算式 x theta y 的結果是什麼？因爲當運算元的值爲虛有值時，運算式 x theta y 的結果可能是眞（ truth ），也可能是假（ false ）。所以，當 x 或 y 或 x 和 y 兩者的值爲虛有值時，我們定義運算式 x theta y 的結果也爲虛有值。爲了能正確地處理虛有值，我們採用了三值邏輯（ three-valued logic ）來取代大家熟悉的二值邏輯（ two volued logic）。三值邏輯的眞值表（ truth table ）如下：

AND	T	?	F
T	T	?	F
?	?	?	F
F	F	F	F

OR	T	?	F
T	T	T	T
?	T	?	?
F	T	?	F

NOT	
T	F
?	?
F	T

3-2 關聯型資料庫系統運算子（ RDB OPERATOR ）

此處我們完全以例題來說明關聯型資料庫系統的運算子。必須注意一點的就是當使用者在用 RDB 運算子時，他並沒有實體的觀念，不像在使用基本運算子時，使用者必須有實體的觀念。 RDB 運算子僅僅是資料處理，不具備語意處理能力。雖然使用者看不見代理值，系統在執行 RDB 運算子時，將代理值視爲資料一同處理。 RDB 運算子中的 UNION, INTERSECTION 及 DIFFERENCE 的運算元須爲聯集適合（ UNION-COMPATIBLE ）關聯，也就是關聯的度必須一樣，且對應的特性需定義在相同的定義域。在本節中，引用了〔 7 ，26 〕中的一些符號及例題。

假設資料庫內容如下：

R

A	B	C
p	1	2
p	2	1
q	1	2
r	2	5
r	2	3

S

A	B	C
p	1	2
q	1	2
q	2	3
r	2	3
r	3	5

X

A	B
a	1
b	2

Y

C	D
c	3
d	4

UNION:

R U S

A	B	C
p	1	2
p	2	1
q	1	2
r	2	5
r	2	3
q	2	3
r	3	5

INTERSECTION:

R ∩ S

A	B	C
p	1	2
q	1	2
r	2	3

DIFFERENCE:

R - S

A	B	C
p	2	1
r	2	5

S - R

A	B	C
q	2	3
r	3	5

CARTESIAN PRODUCTION:

CARTESIAN PRODUCTION (X,Y)

X.A	X.B	Y.C	Y.D
a	1	c	3
a	1	d	4
b	2	c	3
b	2	d	4

THETA-SELECT : THETA 可以是＝，～＝，＞，＞＝，＜，＜＝這六種關係之一。

THETA-SELECT:

R[A~=r]

A	B	C
p	1	2
p	2	1
q	1	2

R[A=r]

A	B	C
r	2	5
r	2	3

R[B>C]

A	B	C
p	2	1

PROJECTION : 當 PROJECTION 後產生重覆維(DUPLICATE TUPLES)時，則多餘的重覆維被刪掉。

PROJECTION:

R[A,B]

A	B
p	1
p	2
q	1
r	2

R[B]

B
1
2

THETA-JOIN : THETA 可以是＝，～＝，＞，＞＝，＜，＜

=這六種關係之一。當 THETA 爲= 時，稱之爲 EQUI-JOIN 。

假設資料庫內尙如下：

R

A	B	C
p	1	2
p	2	1
q	1	2
r	2	5
r	3	3

S

D	E
2	u
3	v
4	u

THETA-JOIN:

R[C>D]S

A	B	C	D	E
r	3	3	2	u
r	2	5	2	u
r	2	5	3	v
r	2	5	4	u

NATURAL-JOIN: 將 EQUI-JOIN 中重覆的行（ COLUMNS ）去掉便成爲 NATURAL JOIN 。

R[C*D]S

A	B	C	E
p	1	2	u
q	1	2	u
r	3	3	v

假設資料庫內容如下：

R

A	B
p	1
p	2
p	3
q	1
r	1
r	3

S

C
1
3

DIVIDE:

R[B/C]S

A
p
r

OUTER UNION:

假設

R

A	B	C
p	1	2
p	2	1
q	1	2

S

B	D
2	u
3	v

先將關聯 R 、 S 擴展成聯集適合關聯如下：

A	B	C	D
p	1	2	?
p	2	1	?
q	1	2	?

A	B	C	D
?	2	?	u
?	3	?	v

再去執行 UNION 運算子。所得結果爲 OUTER UNION，如下：

R ⓤ S

A	B	C	D
p	1	2	?
p	2	1	?
q	1	2	?
?	2	?	u
?	3	?	v

OUTER INTERSECTION 及 OUTER DIFFERENCE 可用與 OUTER UNION 同樣方法定義。

OUTER THERA-JOIN : THETA 可以爲＝，～＝，＞，＞＝，＜，＜＝這六者關係之一。當 THETA 爲＝時，稱之爲 OUTER EQUI-JOIN 。

假設：

S

S#	CITY
s7	c1
s8	c2
s9	c3

P

P#	CITY
p7	c1
p8	c4
p9	c5

OUTER EQUI-JOIN OF S and P ON S,CITY and P. CITY :

S[S.CITY ⊜ P.CITY]P

S.S#	S.CITY	P.P#	P.CITY
s7	c1	p7	c1
s8	c2	?	?
s9	c3	?	?
?	?	p8	c4
?	?	p9	c5

OUTER NATURAL-JOIN OF S and P on S. CITY and P. CITY :

S[S.CITY ⊛ P.CITY]P

S.S#	S.CITY	P.P#
s7	c1	p7
s8	c2	?
s9	c3	?
?	c4	p8
?	c5	p9

3-3 基本運算子

在傳統關聯型資料庫中，使用者在使用運算子時，把資料庫看成是由一堆一堆資料形成，運算子僅是把一堆堆的資料加以合併或是分離或是摘取一部份。本文認爲任一種語意模型，僅使用資料處理運算子是不夠的，因爲語意模型的重點在於語意的表示處理，必須有處理語意能力的運算子。在此處我們所提出的基本運算子便俱有一些語意處理的功能。這些運算子是根據代理值運作，它們是唯一運用到代理爲實體永久識別字這種觀念的運算子。下面將是基本運算子的詳細介紹。

假設目前資料庫中包含有經理及工程師二種實體類型。表示如下：

經理　　工程師

M

M@	NAME	AGE
1	N1	41
2	N2	42
3	N3	43
4	N4	44

E

E@	ADDRESS
3	A1
4	A2
7	A3
8	A4

其中@符號表示代理值。（在關聯中，M 表示 MANAGER，E 表示 ENGINEER）

E_inter:

E_inter(MANAGER,ENGINEER)

M.M@	M.NAME	M.AGE	E.ADDRESS
3	N3	43	A1
4	N4	44	A2

E-inter可以找出同時爲某幾種實體類型的實體，而將其屬性列出。如上例將找到是經理又是工程師的人，且將其屬性值列出。E-inter 與傳統的聯合（JOIN）似乎功能一樣，但其實不然。聯合的目的在於把幾個關聯的特性接合在一起，而 E-inter 的主要目的在把同時爲幾種實體類型的實體找

出。既然同時具有幾種實體類型的性質，它們的屬性當然同時顯示出來。另外必須強調的一點是代理值：使用者看不到，所以使用者無法利用其它方法達到 E-inter 的功能。

E_diff:

E_diff(MANAGER,ENGINEER)

M@	NAME	AGE
1	N1	41
2	N2	42

找出是經理但不是工程師的人。E-diff 的運算元有先後順序，不同的運算元順序將有不同的結果。E-diff (MANAGER, ENGINEER) 並不等於 E-diff (ENGINEER, MANAGER) 。E-diff 運算子的運算元（當以關聯來看時）並不需要俱有相同的度 (DEGREE) 。

E_incl:

E_incl(MANAGER,ENGINEER)

M.M@	M.NAME	M.AGE	E.ADDRESS
1	N1	41	?
2	N2	42	?
3	N3	43	A1
4	N4	44	A2
7	?	?	A3
8	?	?	A4

找出擁有經理或工程師或同時擁有二種身份的人。

的運算元先後順序無關緊要，而且不需要度相同（當以關聯眼光來看時）。

E_excl:

E_excl(MANAGER,ENGINEER)

M.M@	M.NAME	M.AGE	E.ADDRESS
1	N1	41	?
2	N2	42	?
7	?	?	A3
8	?	?	A4

找出有經理身份或工程師身份，但不同時俱有二者身份

的人，運算元的先後順序沒有關係。

E_sel:

E_sel(MANAGER,AGE > 42)

M@	NAME	AGE
3	N3	43
4	N4	44

E-sel 與關聯型資料模型的選擇（ SELECT ）運算子功能一樣。

假設

S

S@	NAME
1	A
2	B
3	C

P

P@	NAME
4	D
5	E
6	F

SP

SP@	S@	P@	QTY
7	1	4	100
8	1	5	100
9	2	5	100

船運是一結合實體類型，其參與者（ paticipant ）爲供應商及零件，（ 關聯中 S 表示 SUPPLIER ， P 表示 PART ， SP 表示 SHIPMENT ）。

E_ref:

E_ref(SHIPMENT,SUPPLIER)

SP.SP@	SP.S@	SP.P@	SP.QTY	S.S@	S.NAME
7	1	4	100	1	A
8	1	5	100	1	A
9	2	5	100	2	B

找出所有供應商與船運的關係，並非所有船運與供應商的各種組合。

E_ref(SHIPMENT,SUPPLIER,PART)

SP.SP@	SP.S@	SP.P@	SP.QTY	S.S@	S.NAME	P.P@	P.NAME
7	1	4	100	1	A	4	D
8	1	5	100	1	A	5	E
9	2	5	100	2	B	5	E

找出供應商，零件及船運三者的關係，並非三者之間的組合。

假設目前資料庫的內容如下：

E

E@	NAME
1	A
2	B
3	C

C

C@	E@	AGE
4	1	10
5	1	15
6	2	20

小孩是雇員的特性實體。（E 表示 EMPLOYEE，C 表示 CHILDREN）。找出小孩與雇員之間的關係。

E_ref(CHILDREN,EMPLOYEE)

C.C@	C.E@	C.AGE	E.E@	E.NAME
4	1	10	1	A
5	1	15	1	A
6	2	20	2	B

由以上三例題可知 E-ref 的運算元是結合實體類型與其參與者或者是特性實體類型與其對應之較大實體類型。E_ref 的第一個運算元必須是結合實體類型或特性實體類型。

屬性選擇：

P_sel:

P_sel(MANAGER,NAME)

M@	NAME
1	N1
2	N2
3	N3
4	N4

以上所提的基本運算子在執行時均是根據代理值運作而得到結果，所以運算後的結果有意義。重覆性的維是否會被刪除，完全由系統根據代理值而定。因爲不同的代理值代表

不同的實體。

在這一節中，我們介紹了七個基本運算子。如何將基本運算子以 RDB 運算子完成如下面表示。

基本運算子的運算元爲實體類型。系統可經由資料典中 PG 關聯找到實體類型的屬性關聯，將所有屬性關聯結合成爲一個關聯來代表實體類型。所以在下面我們以一個關聯來代表一種實體類型。

我們先介紹一些在此節中所用到的符號：

R，S 爲關聯

```
R[A * B]S : NATURAL JOIN OF RELATION R AND S ON ATTRIBUTE A AND B.

R[A ⊛ B]S : OUTER NATURAL JOIN OF RELATION R AND S ON ATTRIBUTE
            A AND B.

ATTNAME(R): ATTRIBUTE NAMES OF RELATION R.

R@        : APPEND SYMBOL "@" TO R.

PJ[R,A]   : PROJECTION OF A SET A OF ATTRIBUTES FROM RELATION R.

SL[R,theta comparison]: SELECT OF TUPLES SATISFYING theta
                        comparison FROM RELATION R.

U         : SET UNION.

CAR(R,S): CARTESIAN PRODUCT OF RELATION R AND S.
```

基本運算子	相對 RDB 運算子
E_inter(R,S)	R[R@ * S@]S;

E_diff(R,S)	temp1:=R[R@ * S@]S;
	temp2:=PJ[temp1,ATTNAME(R)];
	R-temp2;

E_incl(R,S)	R[R@ ⊛ S@]S;
E_excl(R,S)	temp1:=R[R@ * S@]S; temp2:=R[R@ ⊛ S@]S; temp2-temp1;
E_sel(R,theta comparison)	SL[R,theta comparison];
P_sel(R,selector)	PJ[R,R@ U selector];
E_ref(R,S)	temp1:=CAR(R,S); SL[temp1,R.S@=S.S@];
E_ref(R,S,T)	temp1:=CAR(R,S,T); SL(temp1,R.S@=S.S@ AND R.T@=T.T@];

（以上相對 RDB 運算子是由系統執行，所以可以使用代理值）

3-4 巨運算子

一種實體類型在資料庫中是以一個實體關聯及數個屬性關聯表示。這些屬性關聯與實體關聯的關係記錄在資料典的 PG 關聯之中。另外，實體類型互相的關係也都記錄在資料典的 CG, SG, AG 等關聯之中。當使用者要找一實體類型的所有屬性，或是找一結合實體類型的參與者時，都必須透過資料典。而且這些動作常需要數個 RDB 運算子連在一起執行才能達到目的。爲了使用者的方便，系統提供了巨運算子。例如，PROPERTY ， C-PROPERTY 等巨運算子。

假設資料庫中現有 EMPLOYEE, SALES, PARTS, CHILDREN, SHIPMENT 及 STOCKHOLDER 六種實體。 SALES 是 EHPLOYEE 在 JOB 種類之下的子類型； CHILDREN 是 EMPLOYEE 的特性實體類型；STOCKHOLDER 與 EMPLOYEE 是互相獨立且重疊的實體類型； SHIPMENT 是 PART 與 SALES 結合而成的結合實體類型。這六種實體類型以圖 3-1 的方式存在資料庫之中（在關聯的特性名稱中，E 表示 EMPLOYEE ， S 表示 SALES ， P 表示 PART ， C 表示 CHILDREN SP 表示 SHIPMENT ， ST 表示 STOCKHOLDER），對應的資料典內容如圖 3-2（只顯示 PG, CG, AG, SG 及 OVERLAP 關聯，暫不考慮連接性及重覆性）。

EMPLOYEE

E@
1
2
3
4
5

E_TN

E@	NAME	STATUS
1	N1	10
2	N2	20
3	N3	10
4	N4	20
5	N5	20

E_C

E@	CITY
1	T.P.
2	T.N.
3	T.P.
4	T.C.
5	T.C.

SALES

S@
1
2
4
5

S_P

S@	SEX	AGE
1	M	41
2	F	38
4	F	27
5	M	34

PARTS

P@
6
7
8
9

P_P

P@	WEIGHT	CITY
6	25	T.N.
7	30	T.P.
8	24	T.P.
9	34	T.N.

圖 3-1

CHILDREN

C@
10
11
12

CHILDREN_C_SGATE

C@	E@
10	1
11	1
12	4

C_P

C@	AGE	NAME
10	10	N10
11	12	N11
12	13	N12

SHIPMENT

SP@
13
14
15
16

SHIPMENT_A_SGATE

SP@	S@	P@
13	1	6
14	1	7
15	4	7
16	5	8

SP_QTY

SP@	QTY
13	120
14	150
15	170
16	180

STOCKHOLDER

ST@
2
3
17
18

ST_P

ST@	STOCK
2	10K
3	13K
17	17K
18	18K

圖 3-1 (續)

PG

P_RELNAME	E_RELNAME
E_TN	EMPLOYEE
E_C	EMPLOYEE
S_P	SALES
P_P	PARTS
CHILDREN_C_SGATE	CHILDREN
C_P	CHILDREN
SHIPMENT	SHIPMENT
SP_QTY	SHIPMENT
ST_P	STOCKHOLDER

CG

CHARACTERISTIC_E_RELNAME	SUPERIOR_E_RELNAME	MULTIPLICITY
CHILDREN	EMPLOYEE	

SG

SUB_E_RELNAME	SUPER_E_RELNAME	CATEGORY
SALES	EMPLOYEE	JOB

圖 3-2

AG

ASSOCIATION _E_RELNAME	ASSOCIATION _P_ATTNAME	PARTICIPANT _E_RELNAME	CONNEC- TIVITY	MULTI- PLICITY
SHIPMENT	S@	SALES		
SHIPMENT	P@	PARTS		

OVERLAP

E_RELNAME1	E_RELNAME2
EMPLOYEE	STOCKHOLDER

圖 3-2 （續）

• PROPERTY ：找某一實體類型的立即屬性，並形成一 n-ary 關聯。

例如：

PROPERTY(EMPLOYEE):

E@	NAME	STATUS	CITY
1	N1	10	T.P.
2	N2	20	T.N.
3	N3	10	T.P.
4	N4	20	T.C.
5	N5	20	T.C.

• C-PROPERTY ：找一實體類型的特性實體類型之立即屬性，並形成一 n-ary 的關聯。

例如：

C_PROPERTY(EMPLOYEE):

C@	E@	AGE	NAME
10	1	10	N10
11	1	12	N11
12	4	13	N12

• CHARACTERISTICS ：找一實體類型的特性實體類型。

例如：

CHARACTERISTICS(EMPLOYEE):

CHARACTERISTIC_E_RELNAME
CHILDREN

• ASSOCIATIONS ：找參與者所參與的結合實體類型。

例如： ASSOCIATIONS(SALES):

ASSOCIATION_E_RELNAME
SHIPMENT

ASSOCIATIONS(PARTS):

ASSOCIATION_E_RELNAME
SHIPMENT

- PARTICIPANTS ：找出結合實體類型的參與者。

例如：

PARTICIPANTS(SHIPMENT):

PARTICIPANT_E_RELNAME
SALES
PARTS

- CATEGORY：找出超實體類型的種類。

例如：

CATEGORY(EMPLOYEE):

CATEGORY
JOB

- SUBTYPE ：找出超實體類型在某一種類之下的子實體類型。

例如：

SUBTYPE(EMPLOYEE,JOB):

SUB_E_RELNAME
SALES

- OVERLAP ：找出與某一實體類型互相獨立且重疊的實體類型。

例如：

OVERLAP(EMPLOYEE):

E_RELNAME1
STOCKHOLDER

下面介紹如何實現巨指令。

巨 指 令	系統對應動作
PROPERTY(X)	temp1:=SL[PG,E_RELNAME="X"];
	temp2:= PJ[temp1,P_RELNAME];
	temp3:=DENOTE(temp2);
	COMPRESS(temp3,(OUTER NATURAL-JOIN, "X@"));

(DENOTE：作用於一組關聯名稱，結果爲此組關聯名稱所表示的關聯；

COMPRESS ：對一組關聯執行一指定運算。每次對這一組關聯中任二個關聯執行，再以執行結果代替這二個關聯。重覆執行此運算，直到剩下一個關聯。)

```
C_PROPERTY(X)        temp1:=SL[CG,SUPERIOR_E_RELNAME="X"];
                     temp2:=PJ[temp1,CHARACTERISTIC_E_REL
                              NAME];
                     temp3:=APPLY(PROPERTY,temp2);
                     COMPRESS(temp3,E_incl);
```

(APPLY ：對一組關聯中的每一個關聯執行指定運算。)

巨　指　令	系統對應動作
CHARACTERISTICS(X)	temp1:=SL[CG,SUPERIOR_E_RELNAME="X"]; PJ[temp1,CHARACTERISTIC_E_RELNAME];
ASSOCIATIONS(X)	temp1:=SJ[AG,PARTICIPANT_E_RELNAME="X"]; PJ[temp1,ASSOCIATION_E_RELNAME];

巨　指　令	系統對應動作
PARTICIPANTS(X):	temp1:=SL[AG,ASSOCIATION_E_RELNAME="X"]; PJ[temp1,PARTICIPANT_E_RELNAME];
CATEGORY(X)	temp1:=SL[SG,SUPER_E_RELNAME="X"]; PJ[temp1,CATEGORY];
SUBTYPE(X,Y)	temp1:=SL[SG,SUPER_E_RELNAME="X" AND CATEGORY="Y"]; PJ[temp1,SUB_E_RELNAME];

OVERLAP(X)	temp1:=SL[OVERLAP,E_RELNAME1="X"];
	temp2:=PJ[temp1,E_RELNAME2];
	temp3:=SL[OVERLAP,E_RELNAME2="X"];
	temp4:=PJ[temp3,E_RELNAME1];
	UNION(temp2,temp4);

第四章　系統架構

關聯模型的主要缺點是語意表示的不夠完整，關聯綱要（SCHEMA）無法很完整的表示實體之間的關係及相互限制。加強關聯資料庫在語意方面的能力，基本上可分二種方法〔9〕。其一爲放棄現有的資料庫系統，發展一套新的資料模型，建立新的資料庫系統。第二種方法是採用進化方式，把語意資料模型（SEMANTIC DATA MODEL）的能力加到現存的關聯系統上，使其處理語意的能力增加。因爲目前人們最常使用的是關聯系統，第二種方法可以與現存系統配合，比較實際。我們所採取的方法，基本上屬於第二種方法。利用關聯模型來定義另一種語意模型。系統本身乃是關聯系統，但使用者所面對的卻是俱有語意處理能力的另一種模型。整個系統架構可以看成圖4-1。使用者所定義的任何資料在系統中都是以關聯模型存放，所以系統可以不必更改，能繼續操作。爲了證明我們所定義的資料模型可以在關聯系統上執行，我們不僅把第三章所定義的基本運算子，巨運算子以關聯運算子完成外，對於下一章的資料語言，我們也提出如何將選擇（SELECT）指令轉換成運算子的演算法。

一個自述（SELF-DESCRIBING）〔22〕資料庫系統的資料

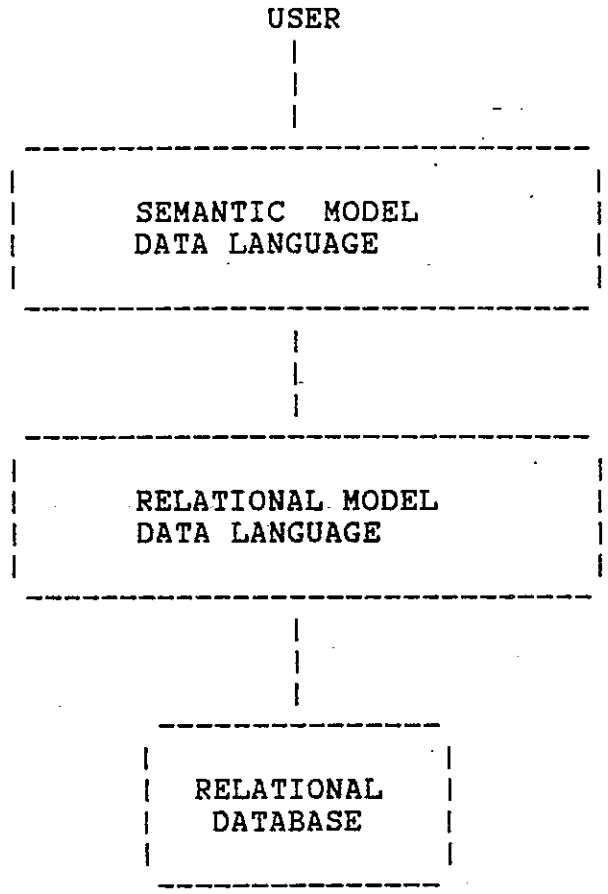

圖 4-1 SYSTEM ARCHITECTURE

典〔 1 〕包含了許多種訊息。它不僅包含了描述應用資料的訊息，也包含了描述資料模型的訊息。在資料典中我們用 DOMAINS, RELATIONS, ATTRIBUTES, PG, AG, SG, OVERLAP, SPAN, SURROGATE等關聯來描述提供給使用者的語意資料模型。而對這些用來描述語意資料模型的關聯，我們也將它們的描述存在資料典之中，所以我們所設計的是一個自述資料庫系統〔 21 〕。資料典的初始內容是資料典本身的關聯的描述，也就是描述資料模型的那些關聯的描述。因此資料典中除了RELATIONS, DOMAINS 及 ATTRIBUTES 這三個關聯有初始資料外，其它所有關聯一開始都是空的。這些初始資料是不可以被更改的。

當資料庫系統開始操作後，隨著使用者定義各種實體類型，資料典的內容也隨之增加。這些內容包含了各種實體類型之間的關係。例如，某種實體類型是另一種實體類型的特性實體；那幾種實體類型結合而成一種結合實體類型；那些實體類型互相重疊等關係。對圖 4-1 中的關聯模型而言，使用者的應用資料及資料典都是它的應用資料。

資料典中除了 DOMAINS, RELATIONS 和 ATTRIBUTES 三個最基本的關聯外，還有 PG, CG, AG, SG, SPAN, OVERLAP及 SURROGATE 等關聯。 DOMAINS, RELATIONS 及 ATTRIBUTES 三個關聯不僅記

錄了應用資料中有那些關聯、定義域、特性及它們之間的關係外，也記錄了資料典本身有那些定義域、關聯、特性和它們之間的關係。資料典中所有關聯的結構和作用如下：

DOMAINS(DOMNAME,DATATYPE,ORDERING)

PRIMARY KEY(DOMNAME)

在資料庫中的每一個定義域都在 DOMAINS 關聯中以一維（ TUPLE ）來登記。其中 ORDERING 特性的值是 "Y" 或 "N" 。當爲 "Y" 時，表示此定義域內的元素可以比較大小和相等。當爲 "N" 時，則僅可以比較是否相等。例如實體定義域（E-domain）內的代理值僅可比較是否相等，無法比較大小。

RELATIONS(RELNAME,RELTYPE)

PRIMARY KEY(RELNAME)

所有的關聯（應用資料及資料典內的關聯）都以 RELATIONS 關聯的一維記錄。關聯型式我們以四位數 A B C D 表示。關聯可分成實體關聯，屬性關聯及資料典關聯。其中實體關聯依其所表示的實體類型再分成結合關聯、中心關聯及特性關聯。每種實體類型都可能是子類型或超類型；也可能參與某一種結合，也可能有特性實體。以上這些關係都記錄在關聯型式中。關聯的分類如表 4-1 。

RELTYPE = ABCD

A = { 1 ：結合實體類型
　　2 ：中心實體類型
　　3 ：特性實體類型

B = { 0 ：不爲子類型及超類型
　　1 ：僅爲子類型
　　2 ：僅爲超類型
　　3 ：爲子類型及超類型

C = { 0 ：本參與結合
　　1 ：參與結合

D = { 0 ：擁有特性實體類型
　　1 ：未擁有特性實體類型

ABCD = 0001　表示屬性關聯

ABCD = 0010　表示資料典關聯

表 4-1

ATTRIBUTES(RELNAME,ATTNAME,DOMNAME,PKEY)

PRIMARY KEY(RELNAME,ATTNAME)

記錄資料庫中每一個特性的名字，及它屬於那一個關聯，定義在那一個定義域及是否爲主鍵。

PG(P_RELNAME,E_RELNAME)

PRIMARY KEY(P_RELNAME)

PG (PROPERTY GRAPH) 關聯用來登記每一個屬性關聯是屬於那一個實體關聯。

CG(CHARACTERISTIC_E_RELNAME,SUPERIOR_E_REL

NAME,MULTIPLICITY)

PRIMARY KEY(CHARACTERISTIC_E_RELNAME)

CG (CHARACTERISTIC GRAPH)關聯用來記錄每一種特性實體類型及被其描述的較大(SUPERIOR)實體類型。特性實體類型的重覆性(MULTIPLICITY)也登記在 CG 關聯中。當重覆性不受限制時，我們以-1 表示。

AG(ASSOCIATION_E_RELNAME,ASSOCIATION_P_ATT

NAME,PARTICIPANT_E_RELNAME,CONNECTIVITY,

MULTIPLICITY)

PRIMARY KEY(ASSOCIATION_E_RELNAME,ASSOCI

ATION_P_ATTNAME)

AG (ASSOCIATION GRAPH) 關聯用一維來記錄資料庫中每一個結合中的每一個參與者。ASSOCIATION-P-ATTNAME 的作用是讓結合用來識別其參與者。每個參與者的連接性及重覆性也必

須加以記錄。

SG(SUB_E_RELNAME,SUPER_E_RELNAME,CATEGORY)

PRIMARY KEY(SUB_E_RELNAME,SUPER_E_RELNAME)

SG (SUBTYPE GRAPH) 關聯登記每一個立即 (IMMEDIATE) 超類型／子類型的關係。

SPAN(SUPER_E_RELNAME,CATEGORY)

PRIMARY KEY(SUPER_E_RELNAME,CATEGORY)

登記超類型被那些種類子類型分割 (PARTITION) 。

OVERLAP(E_RELNAME1,E_RELNAME2)

PRIMARY KEY(E_RELNAME1,E_RELNAME2)

登記互相重疊的獨立實體類型。

SURROGATE(S_VALUE)

PRIMARY KEY(S_VALUE)

SURROGATE 關聯用來產生代理值。每次使用者插入一新的實體，SURROGATE 關聯中 S-VALUE 的值便是新實體的代理值。S-VALUE 每被使用一次，其值加 1。

資料典利用上面這些關聯來記錄所有實體類型之間的關係。關聯系統便可以透過資料典執行完整性規則，以確保資料正確。

第五章　資料語言

在這一章中我們討論的主題有

(1)資料定義語言（ DDL ）；

(2)資料處理語言（ DML ）。

如何定義實體類型，屬性關聯及定義域都將在資料定義語言中一一介紹。資料定義語言也定義了實體類型之間的關係，但如何定義景觀（ VIEW ）及索引（ INDEX ）目前暫不考慮。資料處理語言有四種指令，格式與 SQL 的四個資料處理語言指令相似，但不包含內建函數（ BUILT-IN FUNCTION ），例如： COUNT ， SUM ， AVG ， MAX ， MIN 等功能，也不處理 LIKE 及 EXISTS 等特性。因爲我們必須將資料處理語言轉換成 RDB 運算子或基本運算子然後去執行，而 RBD 運算子及基本運算子並沒有能力安排關聯中維的順序，所以 GROUP BY 也不考慮。我們所提的資料處理語言在某些情況下（後面將舉例說明）可以省略 SQL 中子查詢（ SUBQUERY ）的現象，因此目前我們也不考慮子查詢。由以上簡單介紹，知道我們所提的資料語言並不完整（ NOT COMPLETE ），但對資料定義語言及資料處理語言的基本功能已經具備，而且能夠很直接的表示出使用者的用意。

5-1 資料定義語言

資料定義語言〔 2 , 14 , 16 〕主要在定義實體類型，實體類型之間關係，屬性關聯及定義域等。實體類型可分成中心實體類型，特性實體類型及結合實體類型三種，每一種實體類型又可能有子類型或超類型；也可能擁有特性實體類型附屬於它；也可能是參與某種結合的參與者，這些訊息在定義實體類型時都必須說明。每定義一種實體類型，系統就製造一個實體關聯。因爲實體類型有不同的性質，所以代表它的實體關聯也必須區分成不同性質的關聯。我們用 A B C D 四位數來表示它們的性質，詳細表示法已列在表 4-1 中。

- CREATE ENTITY TYPE

CREATE ENTITY TYPE 可以用來定義中心實體類型，結合實體類型，特性實體類型及子實體類型。每定義一實體類型，系統便製造一個實體關聯。

例一：

```
CREATE ENTITY TYPE EMPLOYEE ;
```

定義了 " EMPLOYEE " 這個中心實體類型。系統產生一個名爲 " EMPLOYEE " 的實體關聯，它擁有一個特性 (ATTRIBUTE)，特性名稱爲 " EMPLOYEE@ " 。目前此實體關聯是空的。

例二：

```
CREATE CHARACTERISTIC ENTITY TYPE  CHILDREN
                OF EMPLOYEE;
```

定義了 " CHILDREN " 這個特性實體類型。系統除了產生一個叫 CHILDREN 的實體關聯外，還產生一個叫 CHILDREN_C_SGATE 的屬性關聯，此屬性關聯有 CHILDREN@ 及 EMPLOYEE@ 二個特性。它主要是用來登記那一個 CHILDREN 實體是附屬於那一個 EMPLOYEE 實體。 CHILDREN 與 EHPLOYEE 實體類型之間的關係在資料典中 CG 關聯加以記錄。

例三：

```
CREATE ASSOCIATIVE ENTITY TYPE SHIPMENT
            PARTICIPANT( SUPPLIER:S@:3:5,
                         PART:P@:2:5);
```

定義一個由 SUPPLIER 及 PART 這二種實體類型結合而成的 SHIPMENT 結合實體類型。系統除了產生 " SHIPMENT " 實體關聯外，還產生一個有 SHIPMENT@ ， S@ 及 P@ 三個特性的屬性關聯 SHIPMENT-A-SGATE . SHIPMENT-A-SGATE 中的 SHIPMENT 爲實體類型名稱，A 表示結合， SGATE 表示代理，因爲此屬性關聯所存的值全是代理值。如同前面第二章資料模型中說明，一結合的參與者必須指明其連接性及重覆性。此例中， SUPPLIER 及 PART 的連接性分別爲 3 和 2 ，重覆性均爲 5 。以上

特性在資料典中 AG 關聯加以登記。

例四：

```
CREATE SUBENTITY TYPE MANAGER OF EMPLOYEE
        PER STATUS;
```

定義 MANAGER 是 EMPLOYEE 在 STATUS種類之下的子實體類型。系統產生 MANAGER 實體關聯。一個子實體類型與其超實體類型俱有同樣性質。例如，EMPLOYEE 是中心實體類型，則 MANAGER也是中心實體類型。子實體類型本身可能參與結合，也可能擁有特性實體類型。所以在資料典中記錄 MANAGER 實體關聯的特性時，千位數必須與 EMPLOYEE 實體關聯一樣，十位數及個位數則目前暫時爲 0（目前乃未參與任何結合，也沒有特性實體類型）。資料典中 SG 關聯會記錄 MANAGER 與 EMPLOYEE 的關係。

• CREATE P-RELATION

利用 CREATE P-RELATION 來定義實體類型的屬性關聯。

例五：

```
CREATE P_RELATION EKN FOR EMPLOYEE
        (NAME   :NAME,
         STATUS:RANK);
```

爲實體類型 EMPLOYEE 定義了一個屬性關聯 EKN。EKN 除

了NAME及STATUS兩個特性外，系統自動加入EMPLOYEE@特性到EKN之中。EMPLOYEE@是爲了指示EKN中每一個維是屬於那一個EMPLOYEE實例。屬性關聯EKN屬於EMPLOYEE實體類型的關係在資料典中PG關聯記錄。

• CREATE DOMAIN

例六：

```
CREATE DOMAIN RANK
        (DATATYPE:INTEGER, ORDERING:"Y");
```

定義一個資料型式爲整數的定義域RANK。ORDERING=Y表示定義域中元素可以比較大小及相等。ORDERING=N表示僅可比較是否相等，不能比較大小。

• CREATE CATEGORY

例七：

```
CREATE SPAN CATEGORY STATUS FOR EMPLOYEE;
```

在EMPLOYEE實體類型下定義了一個種類（SPAN CATEGORY）STATUS。

• DEFING OVERLAP

例八：

```
DEFINE EMPLOYEE OVERLAP WITH STOCKHOLDER;
```

表示EMPLOYEE與STOCKHOLDER這兩個互相獨立的實體類型

擁有相同實體。資料典中 OVERLAP 關聯會登記。

• DROP STATEMENT

DROP指令可以用來去掉實體類型及屬性關聯。

例九：

```
DROP P_RELATION EKN;
```

去掉了 EKN 這個屬性關聯。因爲在資料庫中，屬性關聯的名稱是唯一，所以不須指明被去掉的屬性關聯是屬於那個實體類型。

例十：

```
DROP ENTITY TYPE CHILDREN ;
```

刪掉 CHILDREN 實體類型，附屬於它的屬性關聯也將被刪除。當刪除的實體類型是結合的參與者，則其參與的所有結合均將被刪除；當刪除的實體類型是超實體類型，則其子實體類型亦被刪除；當刪除的實體類型是較大實體類型（ SUPERIOR ENTITY TYPE ），則附屬於它的特性實體類型也將被刪除。所有被刪除的實體類型的屬性關聯全部被刪除。

資料定義語言的語法在附錄一。

5-2 資料處理語言

資料處理語言共有 SELECT , UPDATE , DELETE 及 INSERT 四

種指令。它們處理的對象是實體類型，並非屬性關聯。UPDATE，DELETE 及 INSERT 每次僅能針對一種實體類型執行，SELECT則不受限制。本章最前面曾聲明資料處理語言並不完整，所以以下僅對資料處理語言所能執行的功能加以說明。四種指令的一般形式如下：

一、

```
SELECT selector
FROM   entity type(s)
[WHERE predicate];
```

二、

```
UPDATE entity type
SET    property=expression {,property=expression}
[WHERE predicate];
```

三、

```
DELETE
FROM   entity type
[WHERE predicate];
```

四、

```
INSERT
INTO    entity type[(property {,property})]
VALUES  (constant {,constant})
[WHERE predicate];
```

DML 的語法列在附錄二。

針對 SELECT 指令，我們提出如何將SELECT 指令轉換成運

算子的演算法〔23〕，以證明其確實可行。所以在此先以例題簡單介紹另外三個指令。

• UPDATE

例十一：

```
UPDATE SUPPLIER
SET    STATUS=10
WHERE  STATUS>10;
```

將 SUPPLIER 實體類型中所有 STATUE ＞10 的實例設定其 STATUS 爲 10。在 WHERE 句子中也可能牽涉其它與 SUPPLIER 有關實體類型。

例十二：

```
UPDATE SUPPLIER
SET    STATUS=10
WHERE  ASSOCIATION:SHIPMENT.QTY=100;
```

將所有參與 QTY=100 這種 SHIPMENT 的 SUPPLIER 實例，設定其 STATUS=10 。

• DELETE

例十三：刪除所有年齡超過65歲的員工。

```
DELETE
FROM   EMPLOYEE
WHERE  AGE>65;
```

例十四：刪除有 5 歲以下孩童的員工。

```
DELETE
FROM   EMPLOYEE
WHERE  CHARACTERISTIC:CHILDREN.AGE<5;
```

- INSERT

例十五：插入一位新員工，姓名＝ JOHN ，年齡＝50 ，城市：台北。

```
INSERT
INTO   EMPLOYEE(NAME,AGE,CITY)
VALUES ("JOHN",50,"T.P.");
```

例十六：插入一船運，由 S1 SUPPLIER 及 P1 PART 所結合而成，數量＝100 。

```
INSERT
INTO   SHIPMENT(QTY)
VALUES (100)
WHERE (PARTICIPANT:SUPPLIER.NAME="S1"  AND
       PARTICIPANT:PART.NAME="P1");
```

在提出轉換 SELECT 指令成運算子的演算法〔23〕之前，我們先定義一些符號（圖 5-1 ）。其中 S-COMP 在後面還會介紹其動作。

演算法 1 ：

假設 Q是一 SELECT指令。

```
IF EMPTY(Q.<predicate>)
 THEN
     PJ(Q.<selector>)
      CAR( PROTY (Q.<entity type list>))
 ELSE              /* with predicate */
   IF EMPTY( DPTOR(Q.<predicate>))
    THEN  /* no descriptor */
        PJ(Q.<selector>)
        SL(Q.<predicate>)
        CAR( PROTY(Q.<entity type list>))
    ELSE  /* with predicate,descriptor */
        PJ(Q.<selector>)
        SL(Q.<predicate>)
        CAR( S_COMP( Q.<predicate>,Q.<entity type list> U
                                EXENTYPE(Q.<predicate>)))
```

在此我們認爲所有 SELECT 指令都有 SQL 中 " DISTINCT " 關鍵字。也就是當執行結果有相同維時，重覆維會被去掉。

SUMMARY OF NOTATION:

PROTY(set of entity types): return immediate property relation for each of the entity types.

EXENTYPE(predicate): is the set of entity types whose property appear in the predicate.

EMPTY(string): is a boolean function, which return T if it is applied to a empty string, F otherwise.

DPTOR(predicate): is the name of the descriptor in the predicate.

S_COMP(predicate,set of entity types): perform E_inter or E_ref or both on some of the relations in the set according to the descriptor in the predicate.

SL(P): selection of tuples satisfying predicate P.

PJ(A): projection of set A of properties.

U : set union.

圖 5-1

在圖 5-1 中，我們僅說S-COMP 是在一組實體類型上執行 E-inter 或 E-ref或 E-inter 與E-ref 的動作，至於何時執行 E-inter 或 E-ref 則未說明。演算法 2 說明S-COMP 根據何種因素去執行 E-inter 或 E-ref基本指令。

演算法 2 ：

```
IF (SUPERTYPE-id: OR SUBTYPE-id: OR OVERLAP:) IN
      DPTOR(predicate)
THEN      E_inter
IF (ASSOCIATION: OR PARTICIPANTS: OR CHARACTERISTIC:
    OR SUPERIOR:) IN DPTOR(predicate)
THEN      E_ref
```

在演算法 2 中的 THEN E-inter 表示執行 E-inter 基本指令，但並不是對整組實體類型都執行E-inter，僅對Predicate 中提到的實體類型與其超類型或子類型或重疊的實體類型執行。THEN E-ref表示執行E-ref 基本指令，也不是對整組實體類型執行，僅對 Predicate 中提到的實體類型與其結合實體類型或與其參與者或與其特性實體類型或與其較大(SUPERIOR) 實體類型執行。當E-inter 與E-ref 都須執行時，E-inter 必須先執行。我們將以例題來說明SELECT 指令，及其

經過演算法 1 與演算法 2 後對應的運算子，以及在那些情況下可以省略子查詢。

假設資料庫內容如下：

EMPLOYEE ， CUSTOMER ， ITEM 與 SALES 是四種中心實體類型；

CHILDREN 是 EMPLOYEE 的特性實體類型；

SHIPMENT 是由 SALES 及 ITEM 形成的結合實體類型；

SALES 是 EMPLOYEE 在 JOB 種類之下的子類型。

它們的內容如下（實體特性名稱縮寫）：

EMPLOYEE

E@	NAME	CITY
1	N1	T.P.
2	N2	T.P.
3	N3	T.C.
4	N4	T.N.
5	N5	T.N.

CUSTOMER

CS@	NAME	CITY
16	N16	T.P.
17	N17	T.P.
18	N18	T.C.
19	N19	T.N.

ITEM

I@	TYPE
10	SPORT
11	CAR
12	COMPUTER

CHILDREN

C@	E@	NAME	AGE
6	1	N6	10
7	2	N7	11
8	2	N8	12

SALSE

S@	RANK
2	10
3	20
4	10

SHIPMENT

SP@	S@	P@	QTY
13	2	10	100
14	2	11	150
15	3	11	150

例十七：找出住在台北的雇員姓名。

```
SELECT NAME
FROM EMPLOYEE
WHERE CITY="T.P.";
```

R:

NAME
N1
N2

```
temp1:=PROPERTY(EMPLOYEE);
temp2:=CAR(temp1);
temp3:=SL[temp2,CITY="T.P."];
R:=PJ[temp3,[NAME]];
```

例十八：找出住在同一城市中所有雇員與顧客姓名的組合。

```
SELECT EMPLOYEE.NAME,CUSTOMER.NAME
FROM   EMPLOYEE,CUSTOMER
WHERE  EMPLOYEE.CITY=CUSTOMER.CITY;
```

R:

EMPLOYEE.NAME	CUSTOMER.NAME
N1	N16
N1	N17
.	.
.	.
.	.
N5	N19

```
temp1:=APPLY(PROPERTY,{EMPLOYEE,CUSTOMER});
temp2:=CAR(temp1);
temp3:=SL[temp2,EMPLOYEE.CITY=CUSTOMER.CITY];
R:=PJ[temp3,[EMPLOYEE.NAME,CUSTOMER.NAME]];
```

例十九：找出擁有小孩姓名爲N7的雇員所居住之城市。

```
SELECT CITY
FROM   EMPLOYEE
WHERE  CHARACTERISTIC:CHILDREN.NAME="N7";
```

R:

EMPLOYEE.CITY
T.P.

```
temp1:=E_ref(CHILDREN,EMPLOYEE);
temp2:=CAR(temp1);
temp3:=SL[temp2,CHILDREN.NAME="N7"];
R:=PJ[temp3,[EMPLOYEE.CITY]];
```

此例題如果以關聯資料庫的 DML 執行，則可能需要子查詢，假設 CHILDREN 關聯如下：

CNAME	AGE	PNAME
N6	10	N1
N7	11	N2
N8	12	N2
N9	13	N3

```
SELECT CITY
FROM   EMPLOYEE
WHERE  NAME =
        ( SELECT PNAME
          FROM   CHILDREN
          WHERE  CNAME="N7");
```

例二十：找出雇員 N2 的小孩的年齡。

```
SELECT AGE
FROM   CHILDREN
WHERE  SUPERIOR:EMPLOYEE.NAME="N2";

R    --------------
     | CHILDREN.AGE |
     --------------
     |      11      |
     --------------
     |      12      |
     --------------

temp1:=E_ref(CHILDREN,EMPLOYEE);
temp2:=CAR(temp1);
temp3:=SL[temp2,EMPLOYEE.NAME="N2"];
R:=PJ[temp3,[CHILDREN.AGE]];
```

例二十一：找出住在台北的銷售員的階級。

```
SELECT RANK
FROM   SALSE
WHERE  SUPERTYPE-JOB:EMPLOYEE.CITY="T.P.";

R    ------------
     | SALES.RANK |
     ------------
     |     10     |
     ------------
```

```
temp1:=E_inter(EMPLOYEE,SALSE);
temp2:=CAR(temp1);
temp3:=SL[temp2,EMPLOYEE.CITY="T.P."];
R:=PJ[temp3,[SALSE.RANK]];
```

如以 SQL 中的 SELECT 執行此例題，則依據 SALES 及 EMPLOYEE 兩個關聯的結構而有可能需要子查詢。

例二十二：找出有銷售汽車的銷售員的階級。

```
SELECT RANK
FROM   SALES
WHERE  ASSOCIATION:SHIPMENT.ITEM.TYPE="CAR";
```

R

SALSE.RANK
10
20

```
temp1:=E_ref(SHIPMENT,ITEM);
temp2:=CAR(temp1);
temp3:=SL[temp2,ITEM.TYPE="CAR"];
R:=PJ[temp3,[SALES.RANK]];
```

此例題以 SQL 的 SELECT 指令執行，則須要子查詢。

假如：

SALES

S#	RANK
2	10
3	20
4	10

ITEM

I#	TYPE
10	SPORT
11	CAR
12	COMPUTER

SHIPMENT

SP#	S#	I#	QTY
13	2	10	100
14	2	11	150
15	3	11	150

```
SELECT  RANK
FROM    SALES
WHERE   S# IN
        ( SELECT S#
          FROM   SHIPMENT
          WHERE  I# =
                 ( SELECT I#
                   FROM   ITEM
                   WHERE  TYPE="CAR"));
```

例二十三：找出階級爲10的銷售員所賣出各種項目的數量。

```
SELECT QTY
FROM   SHIPMENT
WHERE  PARTICIPANT:SALES.RANK=10;
```

R

SHIPMENT.QTY
100
150

```
temp1:=E_ref(SHIPMENT,SALES);
temp2:=CAR(temp1);
temp3:=SL[temp2,SALES.RANK=10];
R:=PJ[temp3,[SHIPMENT.QTY]];
```

此例題以 SQL 的SELECT 指令執行，則需要子查詢。

假如 SALES ，SHIP 關聯如例二十二中所示。

```
SELECT QTY
FROM   SHIPMENT
WHERE  S# IN
       ( SELECT S#
         FROM   SALES
         WHERE  RANK=10);
```

例二十四：找出銷售任一種項目，數量爲 150，並居位在台北之銷售員的階級。

```
SELECT RANK
FROM   SALES
WHERE  (SUPERTYPE-JOB:EMPLOYEE.CITY="T.P."
        AND ASSOCIATION:SHIPMENT.QTY=150);
```

R

SALES.RANK
10

```
temp1:=E_inter(EMPLOYEE,SALES);
temp2:=E_ref(SHIPMENT,TEMP1);
temp3:=CAR(temp2);
temp4:=SL[temp3,EMPLOYEE.CITY="T.P." AND
                SHIPMENT.QTY=150];
R:=PJ[temp4,[SALES.RANK]];
```

第六章　結　論

本文提出一種語意模型，此語意模型是根據RM/T模型改良而來。特性實體在定義時，除了指明其描述那一種實體類型，還必須定義特性實體的重覆性，限制實體所能擁有特性實體的個數，如此，便能完整的定義特性實體與被描述實體之間的關係。對於結合實體類型不僅須說明是由那些實體類型結合而成，還須定義參與者的連接性及重覆性。藉著連接性及重覆性的定義，使用者便能表示更多的語意，系統也能根據連接性及重覆性的值處理語意及確保資料正確，達到加強表示語意及處理語意的目的。根據代理值運作的基本運算子，完全利用了代理爲實體永久證別字的這種觀念，因而具備了語意處理的能力。巨運算子則簡化了使用者的工作。根據資料模型所定義的資料語言能很直接表示出使用者的想法，資料定義語言讓使用者可以定義各種性質的實體類型，以及它們之間的各種關係；在使用資料處理語言時，使用者可以很自然的表示其動機，在某些情況下也能省略子查詢的使用，減少使用者的必須使用子查詢的機會，增加工作效率。利用關聯模型來定義語意模型，也就是利用關聯建立語意資料庫系統的資料典，如此系統本身雖然是關聯型系統，使用

者面對的是另一種具有語意處理能力的資料模型，進而使關聯系統俱備了語意處理的能力。

雖然本文改良了RM/T 模型，但認爲還有一些問題應繼續研究探討。

一、語意處理及表示能力。雖然我們增進了RM/T特性實體類型及結合實體類型能表示的語意，但仍無法將眞實世界所有的語意都明白表示。我們知道語意的發展是沒有限制的、發展的愈完全，就愈能誠實、完整的反映眞實世界的各種情形。

二、基本運算子雖俱備語意處理能力，但它所能處理的語意是否能滿足所有要求？發展一套根據代理值運作的完整運算子是迫切需要的。

三、本文所提之資料處理語言並未考慮內建函數以及 GROUP BY 等特性。將這些特性加入後，便是一套很完整的資料處理語言。

附錄一

D D L syntax

```
<DDL-statement>::= <create entity type>
                 | <define domain>
                 | <create P_relation>
                 | <create category>
                 | <define overlap>
                 | <drop-statement>
<create entity type >::= CREATE ENTITY TYPE  id;
                       | CREATE CHARACTERISTIC ENTITY TYPE
                        id [:<multiplicity>] OF id;
                       | CREATE ASSOCIATIVE ENTITY TYPE id
                        PARTICIPANT( { < participant_spec> } );
                       | CREATE SUBENTITY TYPE id OF id PER id;
<define domain>::=DEFINE DOMAIN id (DATA TYPE:<datatype>,
                                    ORDERING:<yorn>);
<create P_relation>::= CREATE P_RELATION id FOR id
                     (  < property_definition_list >);
<create category>::= CREATE [ SPAN ] CATEGORY id FOR id;
<define overlap>::= DEFINE id  OVERLAP WITH id;
<drop_statement>::= DROP ENTITY TYPE id;
                   |DROP P_RELATION id;
<participant spec>::=id:id[:[<connectivity>]:[<multiplicity>]]
<yorn>::= Y | N
<property_defenition>::= id : id
<multiplicity>::= INTEGER
<connectivity>::= INTEGER
```

附錄二

D M L syntax

```
<DML-statement>::= <select> | <update> | <delete> | <insert>

<select>::= SELECT <selector>
            FROM id_list
            [WHERE <predicate>];

<update>::=UPDATE id
           SET id = <exp> {,id = <exp>}
           [WHERE <predicate>];

<delete>::=DELETE
           FROM id
           [WHERE <predicate>];

<insert>::=INSERT
           INTO id [ id_list ]
           VALUES (< constant_list >)
           [WHERE <predicate>];

<selector>::=<proper_spec_list>

<predicate>::= (<predicate> <boolean> <predicate>)

             | [<descriptor>]<property-spec><comp-op>[<descriptor>]

               <property-spec>

             | [<descriptor>]<property-spec><comp-op><constant>

<descriptor>::=ASSOCIATION:|PARTICIPANT:|CHARACTERISTIC:|SUPERIOR:

              |SUPERTYPE-id:|SUBTYPE-id:|OVERLAPTYPE:

<property_spec>::=id | <entity_spec>.id

<boolean>::= AND

<comp-op>::= = | ~= | > | >= | < | <=

<entity_spec>::=id | <entity_spec>.id

<exp>::= <exp> + <term> | <exp> - <term> | <term>

<term>::= <term> * <factor> | <term> / <factor> | <factor>

<factor>::= <constant> | <property_spec> | <(exp)>
```

參考文獻：

1. Allen,F.W.,Loomis,M.E.S.,and Mannino,M.V. The integrated dictionary/directory system. ACM Comput.Surv.14,2(June 1982),245-286.

2. Chamberlin,D.D.,et al. SEQUEL 2: A unified approach to data definition,manipulation,and control. IBM J.Res.Dev.20,6(Nov. 1976),560-575.

3. Chen,p. The entity-relationship model:Toward a unified view of data. ACM Trans.Database Syst.1,1(Mar.1976),9-36.

4. Date,C.J. An Introduction to Database Systems,Volume 1. 3rd ed. Addison-Wesley,Reading,Mass.,1982.

5. Date,C.J.An Introduction to Database Systems,volume 2.Addison -Wesley.

6. Hammer,M.,and McLeod,D. Database description with SDM:A semantic database model. ACM Trans.Database Syst.6,3 (Sept.1981),351-386.

7. Codd,E.F. Extending the database relational model to capture more meaning.ACM Trans.Database Syst.4,4(Dec.1979),397-434.

8. Smith,J.M., and Smith,D.C. Database abstractions:Aggregations and Generalization. ACM Trans.Database Syst.2,2(June 1977), 105-133.

9. Tsur,S.,and Zaniolo,C. An implementation of GEM-Supporting a semantic data model on a relational back-end. In SIGMOD Record 14,2(June 1984),286-295.

10. Codd,E.F. A relational model of data for large shared data banks. Commun.ACM 13,6(June 1970),377-387.

11. Ullman,J.D. Principles of Database Systems,2nd ed. Computer Science Press, Rockville,Md.,1982.

12. Zaniolo,c., and Melkanoff,M.A. On the design of relational database schemata. ACM Trans.Database Syst.6,1(Mar. 1981), 1-47.

13. Schmid,H.A., and Swenson,J.R. On the semantics of the relational data model. In Proc.ACM SIGMOD Int.Conf. Management of Data,San Jose,Calif.,1975.

14. Shipman,D.W. The functional data model and the data language DAPLEX.ACM Trans.Database Syst.6,1(March 1981),140-173.

15. Stonebraker,M.,Wong,E.,Kreps,P.and Held,G. The design and implementation of INGRES. ACM Trans.Database Syst.6,1(Sep. 1976),189-222.

16. Ullman,J.D. Implementation of logical query languages for database. ACM Trans.Database Syst.10,3(Sep.1985),289-321.

17. Maier,D.,Ullman,J.D.,and Vardi,M.Y. On the foundations of the universal relational model.ACM Trans.Database Syst.9, 2(June 1984),283-308.

18. Korth,H.F.,Kuper,G.M.,Feigenbaum,J.,Gelder,A.V.,and Ullman ,J.D.System/U: a database system based on the universal relation assumption.ACM Trans.Database Syst.9,3(Sep.1984), 331-347.

19. Osborn,S.L. and Heaven,T.E. The design of a relational data base system with abstract data types and domains.ACM Trans. Database Syst.11,3(Sep.1986),357-373.

20. Jeorey,T.J.,Yang,D., and Fry,J.P. A logical design methodology for relational database using the extended entity-relationship model.ACM Comput.Surv.18,2(June 1986) ,197-222.

21. DAFTG of the ANSI/X3/SPARC Database System Study Group. Reference model for DBMS standardization. In SIGMOD Record 15,1(March 1986),19-58.

22. Mark,L. and Roussopoulos,N. Metadata management.IEEE Computer,19,12(DEC.1986),26-36.

23. Ceri,S.,and Gottlob,G. Translating SQL into relational algebra:optimization,semantics,and equivalence of SQL queries.IEEE Trans.Soft.Eng.SE-11,4(April 1985),324-345.

24. Farmer,D.B.,King,R.and Myers,D.A. The semantic database constructor.IEEE Trans.Soft.Eng.SE-11,7(July 1985),583-591.

25. Parent,C.,and Spaccapietra,S. An algebra for a general entity-relationship model.IEEE Trans.Soft.Eng.SE-11,7 (July 1985),634-643.

26. Pirotte,A. A precise definition of basic relational notations and of the relational algebra. In ACM SIGMOD Record 13,1(Sep. 1982),30-45.