5

系統模型（System models）

學習重點

1. 認識程序塑模的工作
2. 學習系統模型的分析
3. 了解資料流程圖（DFD，data flow diagram）的繪製與使用
4. 認識資料塑模（data modeling）的工作
5. 了解資料塑模程序
6. 學習資料塑模的方法
7. 認識 ER model
8. 學習 ER model 的表示方法
9. 了解資料塑模與特徵工程的差異。

學習地圖

5.1 資訊系統開發的架構
5.2 程序塑模（process modeling）
5.3 結構化的資料流程圖
5.4 商業程序塑模（BPM，business process modeling）
5.5 資料塑模（data modeling）
5.6 資料模型（data model）簡介
5.7 資料塑模 vs. 特徵工程 (feature engineering)

在軟體開發的過程中，經常會碰到需要繪製所謂的系統模型（system models）的情況，問題是系統模型到底是什麼？該如何產生與繪製呢？我們下面會解釋系統模型，然後介紹系統塑模（system modeling）中的兩大工作：程序塑模（process modeling）與資料塑模（data modeling）。

絕大多數的資訊系統都要倚賴資料庫系統來儲存與處理大量的資料，在系統分析的過程中有很多的時間花在資料塑模上，我們必須了解組織的資訊系統使用那些資料、資料之間有什麼樣的關係，以及資料本身具有什麼樣的特徵與結構，然後用適當的方式表達出來。

5.1 資訊系統開發的架構

圖 5-1 顯示資訊系統開發的架構，裡頭的每個方格都會產生各種模型，代表參與者在開發環境中運用某些方法，針對系統特性所發展出來的模型。雖然有很多方格，但這並不代表內容雜亂，因爲整個系統開發的過程有開發的方法論來引導，每一種模型出現的階段、提供的功能與扮演的角色，都已經有明確的規範。

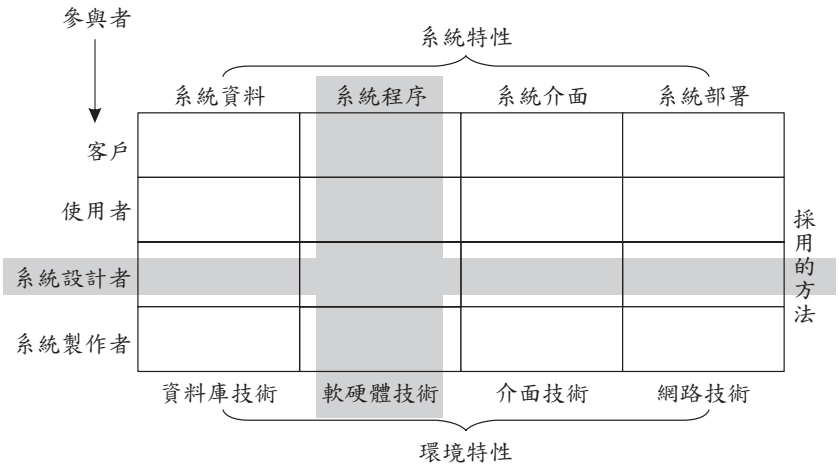


圖 5-1　資訊系統開發的架構

通常客戶與使用者在資訊系統開發的過程中常扮演關鍵的角色，提出主要的需求，系統設計者往往要爲了特定的需求絞盡腦汁，畢竟同一個問題解決的方法很多，客戶通常都希望功能齊全，但是花費的成本不高。因此，光是描述使用者的需求或許還不是難事，最難的還是在於如何找到滿足需求的最有效的

設計方式。舉一個簡單的例子，客戶可能在爲員工加保時，可選擇月保或是日保，但是對某些員工來說，因特殊的原因只能月保，不能日保，對於客戶來說，一般都會要求系統設計者在加保的介面中設計成由使用者選擇日保或是月保，對於系統設計而言，這個問題沒有那麼單純，因爲使用者未必能記住所有只能月保的員工，由使用者自行選擇容易出錯，但是若是每次系統都要檢查，則會影響系統的效能，這時候就要花點時間想出一個好的解決方法，跟客戶討論。

延伸思考

對於同樣的一件事，不同的人往往會有不同的看法，有時候思考的角度不一樣，看法也會不一樣。資訊系統那麼複雜，參與的人那麼多，衍生出來的想法當然更多了，這也是系統分析與設計工作難度的所在。不過，複雜的事物經過抽絲剝繭以後，也可以變得很單純，這就要靠系統分析師清楚的判斷與思考，再加上專業背景與豐富的經驗技巧，才能勝任。

5.1.1 系統模型（system models）的定義

模型反映事實，是事實的一種表示方式，建立模型就是在描述這個現實世界的某一部分，建立模型的過程也常稱爲塑模（modeling），對現有的系統塑模（system models）來說，可以讓我們更深入地了解這些系統，對於尚未建立的系統塑模而言，等於是爲未來的系統進行規劃。現代人常爲自己的住家裝潢，在裝潢以前做的思考、規劃與設計就是一種塑模的過程，塑模做的好壞跟未來裝潢的成敗有密切的關係。從另外一個觀點來看，系統模型把原來取得的需求資訊結構化，更清楚地描述一個軟體系統的特徵。

5.1.2 邏輯模型與實體模型

在系統分析與設計的領域中，常會把各種模型區分爲邏輯模型（logical model）與實體模型（physical model）；邏輯模型描述一個系統本身以及系統的功能，跟系統是如何製作出來的是無關的，換句話說，有了邏輯模型以後，系統用什麼方式製作都可以。邏輯模型也常稱爲概念模型（conceptual model）或商業模型（business model），所以當我們在想像室內的裝潢時，並不需要先確定要找那一家裝潢公司。

實體模型所描述的可能會包括邏輯模型的內容，再加上系統實際上製作的方法與技術，所以實體模型會跟製作的方法與技術有關，必須考量技術上的限制，因此實體模型也常稱爲實作模型（implementation model）或是技術模型（technical model）。眞正要確定裝潢時，同樣需要考量到各種實際的因素，例如屋齡、材料、預算等。

系統分析師很早就發現邏輯模型跟實體模型應該要分開，所以在實務上邏輯模型會用來描述商業與營運上的需求，而實體模型則專注於技術上的設計與考量；系統分析師的工作偏向於邏輯模型的建立，原因如下：

1. 純粹邏輯模型的思考重視概念、需求與功能的了解，不受製作方法與技術的干擾或誤導，邏輯模型鼓勵創意思考與腦力激盪。
2. 邏輯模型讓我們更專注於需求的了解，降低了忽略或誤解實際需求的風險，讓需求的分析更完整、精確與一致。
3. 邏輯模型可以用來跟眞正的使用者或客戶進行溝通，這種溝通會排除技術性的細節，更容易掌握使用者的需求，提昇溝通的效率。

追根究底

到底邏輯模型跟實體模型有什麼實質的差異呢？在設計學生成績系統的時候，我們會談到 60 分為及格的標準，成績的範圍是 0 到 100，50 分或 50 分以上才能補考，這些都是邏輯模型探討的內容。實體模型當然要把這些概念也描述出來，不過實體模型還要考慮到成績的資料要如何表示，例如到小數底下第幾位，或是成績單的格式如何等問題。

5.2 程序塑模（process modeling）

程序塑模是針對系統程序（system process）的結構、資料流程（data flow）、邏輯（logic）與步驟加以組織與記錄的技巧，不同的系統開發方法也衍生出不一樣的程序塑模的方式。

5.2.1 程序（process）的概念

程序可以看成是一個資訊系統的基本組成，所以把描述一個資訊系統的程序都找出來，就能把資訊系統給拼湊出來。程序本身會針對描述商業事件

（business events）應該產生的回應，把資料轉變成有用的資訊。在程序塑模的過程中，除了了解程序本身的特性與功能以外，還要確立程序與周圍環境、其他系統，以及其他程序的關係。

一個系統（system）本身其實就是一個程序（process），圖 5-2 以一個長方形框出一個系統的範圍，這個範圍之外的是系統的環境，系統與環境之間可以透過輸入（input）與輸出（output）來進行溝通，由於環境持續在變化中，所以系統要透過回饋與控制來調整自己，適應環境。

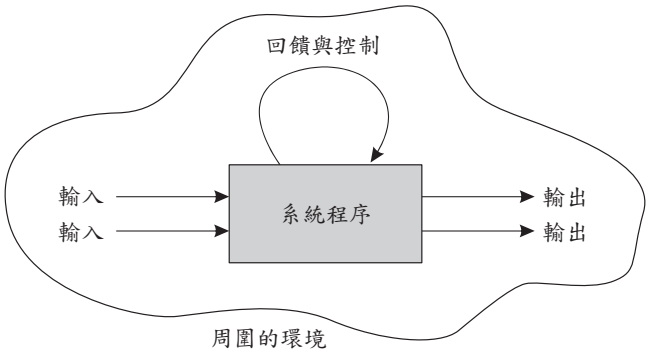
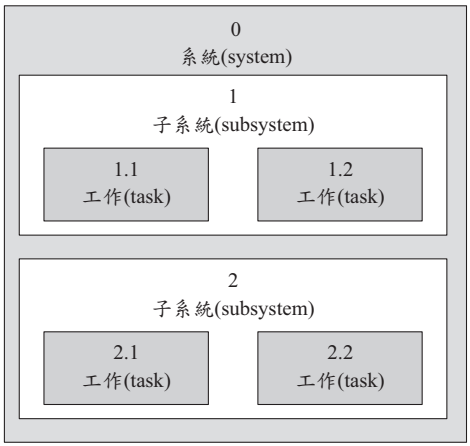


圖 5-2　系統（system）與程序（process）的概念

程序（process）代表因應輸入或發生的事件而完成的一段工作，在程序塑模中同樣有邏輯程序塑模（logical process modeling）與實體程序塑模（physical process modeling）之分，邏輯程序強調完成的是什麼工作，實體程序則會更進一步地說明完成的方法（即如何完成），以及由誰完成。

5.2.2　程序的分解（process decomposition）

一個複雜的系統是很難用少數的程序來描述的，通常需要分解成組成系統的子系統（component subsystem），甚至於要把子系統再進一步地分解成更小的子系統，一直到子系統本身在表示上沒有困難，不會因爲太複雜而難以處理。



5.2.3 資料流程圖（data flow diagram）

程序塑模來自傳統的軟體工程方法，多年來衍生出各式各樣的表示方法，包括程式結構圖（ program structure charts ）、資料流程圖、邏輯流程圖（ logic flowcharts ）與決策表格（ decision table ）等。以資料流程圖來說，主要是用來描述系統裡頭資料的流動，以及系統所進行的工作。一般的資料流程圖常使用以下的表示方式，如圖 5-3 所示：

1. 圓角的方形代表程序（process），程序會對資料進行處理。
2. 方形代表外部程式（ external agents ），可能是資料的來源（ source ）或是接受資料的地方（sink）。
3. 開放式的區域代表資料儲存區（data store），表示檔案與資料庫。
4. 帶箭頭的線段表示程序的資料流（data flow）、輸入，或輸出。

圖 5-3 的 DFD 表示法採用的是 Gane & Sarson 的表達方式，DeMarco & Yourdon 也有提出類似的表達方式，雖然略有不同，但是畫出來的 DFD 大同小異，基本的原則是一樣的。

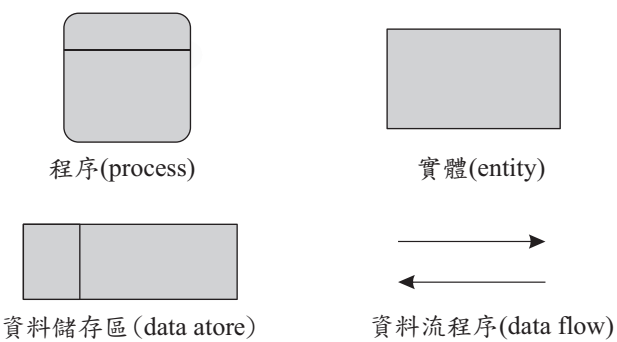


圖 5-3　資料流程圖的表示法

資料流程圖的繪製必須遵循一些規則，圖 5-4 顯示的是其中的一個規則，這些規則的訂定是爲了保持資料流程圖的正確性，但是完全遵循規則並不保證資料流程圖是絕對正確的。圖 5-5 顯示一個資料流程圖的實例。

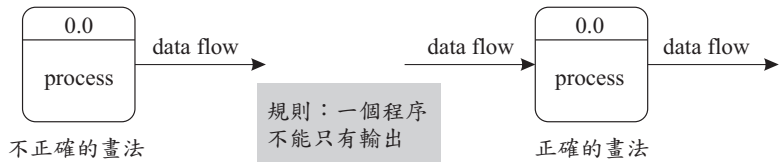


圖 5-4　資料流程圖繪製的規則

追根究底

資料流程圖與系統流程圖（system flowchart）有什麼差異呢？在資料流程圖流行以前，很多分析師使用系統流程圖，但是在表示法上並沒有統一，而且系統流程圖描述太多實體的細節，超出系統分析階段希望專注的重點，所以後來就比較少人再使用系統流程圖。

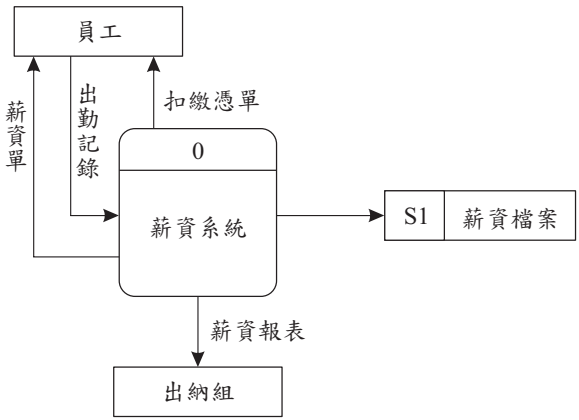


圖 5-5　資料流程圖的實例

5.2.4　描述程序的邏輯（process logic）

資料流程圖明確地找出系統的程序，也標示出程序之間的關係，但是每個程序內部詳細運作的狀況並沒有記錄，這一部分要靠程序的邏輯塑模（logic modeling）來發掘，有一些工具可以運用，這些工具通常不必跟任何的程式語言有關，主要用來當成分析者與使用者之間的一種溝通方法，常見的工具包括：純粹結構化的語言描述、決策表格（decision table）、決策樹（decision tree）與狀態變化圖（state-transition diagram）等。

5.3 結構化的資料流程圖

資料流程圖算是一種結構化分析的技巧，進行系統分析的時候，可以配合分析的工作把資訊系統的資料流的邏輯畫出來，由粗略到詳細，形成一種類似於階層式的結構。圖 5-6 顯示的是一個環境圖（context diagram），代表一個組織資訊系統的概觀，只有單一的程序，這個程序代表整個系統。環境圖裡頭沒有資料儲存區，因爲我們假設資料儲存區包含在系統中，代表 source/sink 的實體描述系統與所在環境之間的關係。

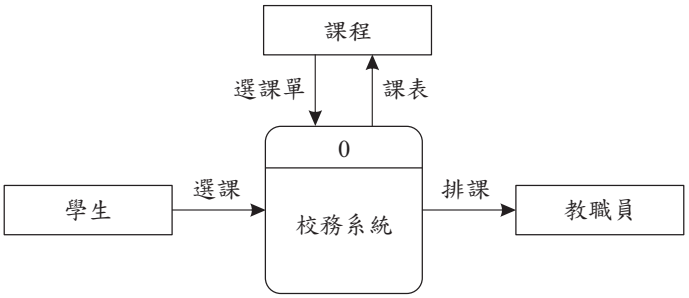


圖 5-6　環境圖（context diagram）

接下來系統分析師要了解 process 0 是由那些其他的程序所組成的，圖 5-7 顯示第 0 層的資料流程圖（ level-0 DFD ），原來的 source/sink 都維持一樣，但是原來的 process 0 現在變成了 3 個程序，這表示我們以這 3 個程序來表達原來 process 0 的功能，等於有更多的細節。

Level-0 DFD 算是系統最頂層的程序表示圖，從圖 5-7 可以看到 data store 也出現了。接下來系統分析師可以進一步地深入了解資訊系統，畫出 level-1、level-2、… 、level-n 的 DFD，產生詳細的系統程序模型，這就是一種系統化與結構化的系統分析方法。這裡要特別注意 DFD 中所描述的資料流並沒有說明發生的時間、資料量或是資料流發生的頻率，所以有很多實際的細節沒有包含在裡頭，這是因爲 DFD 描述的是邏輯的資料流程，強調的重點不同。

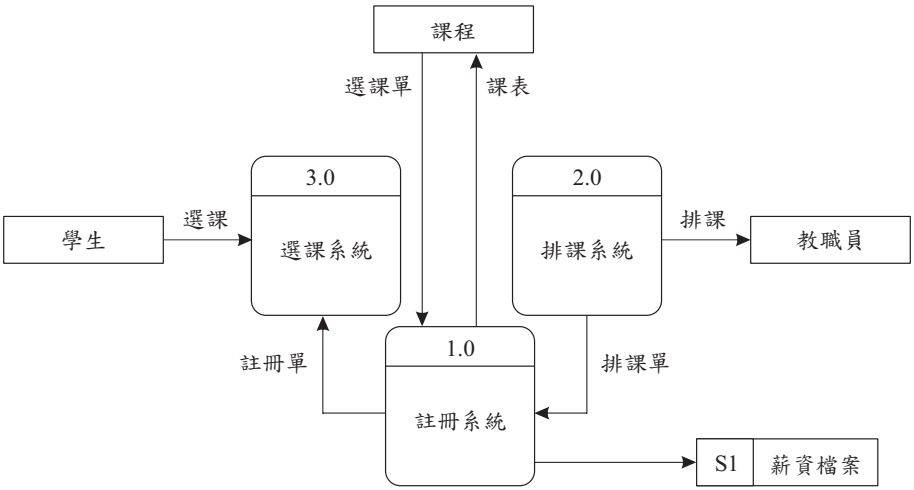


圖 5-7　第 0 層的資料流程圖（level-0 DFD）

5.4　商業程序塑模（BPM，business process modeling）

商業程序塑模（ BPM ）的目的在於描述企業現在與未來的程序，讓目前進行中的程序能夠被分析與改善。BPM 通常由商業的分析師與管理階層負責，試著改善程序的效率與品質，這種改善不見得需要資訊科技，不過資訊科技常在 BPM 中扮演重要的角色。商業程序管理（ BPM，business process management ）的縮寫也是 BPM，可以看成是涵蓋商業程序塑模的領域。

UML 可以用來描述描述 BPM 的結果，MDA（ model-driven architecture ）與 SOA（ service-oriented architecture ）也都是跟 BPM 相關的資訊技術，BPM 強調的是企業架構中跟程序相關的部分，當企業有大的變動時，BPM 扮演重要的角色；例如兩個企業合併時，兩者的程序都需要經過審慎的評估，這樣管理階層才能有效地避免一些重複的作業。一個商業程序有下列的特徵：

1. 有一個目標。
2. 有特定的輸入。
3. 有特定的輸出。

4. 有使用的資源。

5. 包含一些工作或活動，而且是按照某種順序執行的。

6. 可能對組織內部多個部門產生影響。

7. 對於組織內部或外部的參與者產生某種價值。

簡單地說，一個商業程序包括一些活動，目的在於爲客戶與市場產生特定的輸出，強調的是工作在組織內完成的方式，對於得到的產品不見得需要詳細的描述，主要關心的是工作執行的時間、地點與順序，開始與結束的時機，以及輸入與輸出。

商業程序其實是整個企業架構的一部分，但是商業程序的層次比較高，企業的應用、技術與資料的存在是爲了支援商業程序的進行，當企業要更新再造時，關鍵也在於商業程序的再造，不是多花錢改善硬體設備就能解決問題的，所以軟體專案的開發一定要跟商業程序的目標結合。

5.5 資料塑模（data modeling）

資料塑模是針對系統資料的特性進行組織與記錄的技巧，也常稱爲資料庫塑模（database modeling），因爲資料模型在實作上通常都會選擇資料庫的技術。資料塑模也常被稱爲資訊塑模（information modeling）。有很多人認爲資料塑模是各種塑模技術中最重要的，原因如下：

1. 資料是很多資訊系統的核心，由多個程序共用，反映出商業系統的主要需求。

2. 資料的定義在整個資訊系統中是比較穩定而少有改變的，會比系統的程序快確認，而且規模也比其他的模型要小。

3. 資料塑模建立了後續溝通的共同術語與規則，一旦完成了資料模型以後，系統分析者會更容易進一步地了解系統的各種需求。

在不同的方法論中採用的資料塑模的方法也會不一樣，早期資訊系統的開發常用 ER 模型（entity-relationship model）來進行資料塑模，而 ER 模型

正是一般資料庫系統中常用來進行概念塑模的工具，以實體（entity）與關係（relationship）的觀念爲基礎；假如採用的是物件導向的分析與設計，則以類別（class）與物件（object）的觀念爲基礎，跟 ER 模型有顯著的差異。不過一般說來，**有一些資料塑模的觀念與技巧是不會因為資料模型的差異而不同的，例如溝通的技巧、取得資訊的方式、分析的方式等**。

5.5.1 資料塑模的策略

在系統分析階段，通常會以邏輯資料塑模爲主（logic data modeling），集中在使用者端需求的了解與分析，建立所謂的應用資料模型（application data model）。對於某些人來說，可能還是習慣先畫出系統的程序模型，但是這個階段的資料塑模會有以下幾個好處：

1. 系統分析者可以從資料模型快速而完整地了解商業模型與簡單的需求。
2. 資料模型的建立通常會比程序模型要來的快。
3. 資料模型的規模與篇幅遠比程序模型要小。
4. 程序模型的建立容易碰到一些瓶頸。
5. 資料模型的變動比較小，後面的階段可以沿用。

資料塑模的過程中有一些策略與技巧，一開始了解的時候，可以先針對資料模型中的主要實體進行探索，還不用一下子找出實體的屬性來，此時得到的是所謂的情境資料模型（context data model）。這樣的資料模型可以再經過修改，慢慢地把細節勾勒出來：

1. 建立鍵值資料模型（key-based data model）：以關聯式資料模型來說，資料表格之間的關聯是透過鍵值來建立的，必須確立主鍵（primary key）與外鍵（foreign key）。
2. 建立完整屬性的資料模型（fully attributed data model）：各實體的屬性、屬性的資料型態、屬性值的範圍（domain）、屬性的預設值（default）等，都必須確立，才能建立完整屬性的資料模型。

系統分析階段得到的資料模型純粹用來描述需求，在系統設計的階段必須將邏輯資料模型轉變爲實體資料模型，有時候也叫做資料庫定義（ database schema ），是依據選定的資料庫管理系統將資料模型定義出來。在關聯式資料庫系統中，實體資料模型還可以經過正規化（ normalization ）來改善原來的資料模型。

5.5.2 資料塑模的方法

不同的資料模型通常都會有特定的資料模型的表示法，熟悉這些表示法可以讓我們看得懂別人畫的資料模型圖，或是自己也可以繪製這些圖形，但是資料塑模的關鍵在於要去找出資料模型的內涵來！在物件導向的分析與設計方法中，常透過情節（ scenarios ）與使用案例（ use case ）來了解一個系統的需求，我們要試著找出系統的資料有那些，以 ER 模型來說，實體（entity）是第一個需要找的，以物件導向模型來說，則是要找出類別（class），有一些常用的方法：

1. 跟系統的使用者面談，從溝通裡頭找出關鍵的詞彙，了解各名稱與術語代表的涵義。
2. 在面談中直接詢問使用者有那些資料需要使用、儲存或產生的。
3. 從現有的表單與報表中找出可能需要定義與使用的資料。
4. 透過找到的資料，運用一些技巧或活動來確立實體或類別，例如 CRC、腦力激盪等。
5. 運用 CASE 工具，利用反向工程（ reverse engineering ）的技術來從已經存在的系統或資料庫推演出資料庫定義。

動手動腦

塑模（ modeling ）是非常實務導向的工作，人際溝通的技巧是關鍵，系統分析師不可能對每個應用領域都熟悉，更何況使用者的需求差異性很大。理論上教條式的原則的確有一些引導的作用，但是一個成功的系統分析師絕對需要真正參與系統分析才能得到成長。

5.6 資料模型（data model）簡介

資料庫提供資料的服務與資料的儲存方式隱含在資料庫系統的功能裡。我們可以把這種資料的處理模式稱爲資料抽象化（ data abstraction ），因爲使用者所看到的資料已經處理過，簡化了很多細節。**資料模型（data model）是資料抽象化的基礎**，我們可以把資料模型定義成描述資料庫結構的各種觀念，而資料庫的結構則包括資料的型式、資料之間的關係，以及資料的限制（constraints）。

除此之外，資料模型多半也會包含一些基本的操作，對資料庫進行資料的擷取與更新。**（ Ullman 1988 ）對於資料庫使用的資料模型所下的定義是：一種正式的數學表示法（mathematical formalism），包括兩大組成**。

1. **描述資料的表示法**。
2. **處理資料的一組操作（operations）**。

一般人會想知道的是到底有沒有一種資料模型是最好的，因爲這樣我們只要學會使用一種資料模型就好了。但是事實上並不是如此，各種資料模型都有一些特長與不足，可能可以說某些資料模型比較完善一點，但是很難說那種資料模型是最好的。下面幾項資料模型的特徵也是 Ullman 提出來的，可以用來比較資料模型之間的差異：

1. 目的與用途。
2. 以物件（object）還是以值（value）爲主。
3. 資料重複（redundancy）問題的處理。
4. 多對多（many-many）關係的處理。

5.6.1 資料模型的分類

資料模型用來描述資料庫結構的觀念有時候差異蠻大的，因此產生了很多種資料模型，暫且不論那一種資料模型比較好，我們可以利用資料模型描述的方式來進行各種資料模型的分類：

1. 概念式的資料模型（conceptual data model）：與使用者對於資料了解的方式很相似，也稱爲高階的資料模型（high-level data model）。我們可以使用個體（entity）來表示現實世界中的物件或觀念，個體的特徵則用屬性（attribute）來描述，個體之間的關係（relationship）代表個體之間的互動。

2. 實體資料模型（physical data model）：可以描述資料在電腦中如何儲存的細節，也稱爲低階的資料模型（low-level data model）。以記錄（record）的儲存方式爲例，實體資料模型會記載記錄格式、記錄的順序與存取路徑（access path）等資訊。

3. 實作資料模型（implementation data model）：描述的方式可能可以讓一般的使用者了解，但是也包含了一些有關於資料儲存的細節，因此，在實作上有很大的功用與彈性。

從以上的分類來看，關聯式的資料模型、網路式的資料模型與階層的資料模型都算是實作資料模型，在傳統的商業 DBMS 的產品中使用的最廣泛。由於實作資料模型通常都使用記錄（record）的結構來表示資料，所以也稱爲 record-based data model。**Entity-Relationship model 屬於概念式的資料模型，物件導向資料模型比較接近概念式的資料模型，但具有實作資料模型的特性。這裡要注意上面的分類只是文獻一般的探討，方便說明，我們不必太在意到底某一種資料模型該歸屬於那一類。**

5.6.2 資料獨立性（data independence）的觀念

three-schema architecture 可以用來解釋**資料獨立性（data independence）的觀念，主要的關鍵在於我們能在不同的層次（level）上改變對應的 schema，而不會影響其他層次上的 schema**。問題是資料獨立性有什麼好處呢？我們先來看看所謂的 schema 的定義：

1. internal schema：描述資料庫的實體儲存結構（physical storage structure），使用 physical data model 來描述資料儲存的細節與資料庫存取的路徑。

2. conceptual schema：描述資料的屬性（entities）、資料型式（data types）、關聯（relationships）、限制條件（constraints）與使用者的操作，使用 high-level data model。所定義的資料針對所有使用者的需求。

3. external schema：描述資料庫的一部分，可能只對某些使用者有用。

圖 5-8 畫出 three-schema architecture，我們可以把這樣的架構看成是一種 DBMS 的架構，不過顯然這樣的看法比較偏向資料的角度，與系統的功能似乎關係不大。three-schema architecture 的主要目的是要把應用系統與實體的資料庫分開，這種分隔是要讓資料的使用更簡化。我們可以在實際的 DBMS 中看到這樣的架構存在。

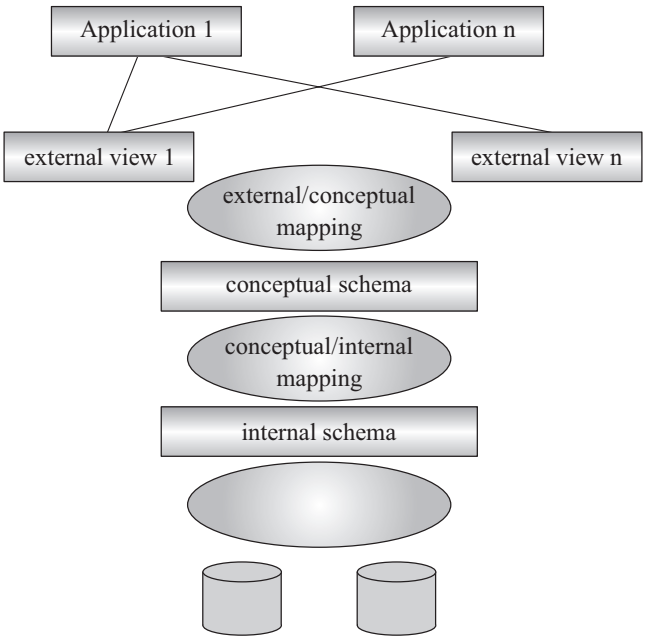


圖 5-8　three-schema architecture

軟體開發工具也支援 three-schema architecture，所以運用資料模型建立的 conceptual schema 通常可以運用軟體開發工具自動轉換成某種 DBMS 接受的 internal schema，例如 PowerDesigner 把 CDM 轉成 PDM。這裡要注意 3 種 schema 都是用來描述資料的，所以說起來是資料的定義，眞正的資料是儲存在實體的儲存層次中，要把資料送到應用程式裡還要經過一些轉換與對應

（ mapping ）的程序，這些過程會花費額外的時間與處理。當然，這樣的架構主要的精神還是在於讓所得到的好處遠超過必須付出的成本。

資料獨立性（ data independence ）是資料庫系統中相當重要的觀念，我們可

思考問題

這裡有幾個值得思考的問題，首先是有關於 three-schema architecture，這種層次化的分類有其優點，可是那是純粹以資料模型的觀點來看的，假如以系統的功能來看，會有類似的層次嗎？另外一個問題是有關於資料模型的建立，假如從早期 relational database design 的觀點來看，似乎可以從作業中的表單與報表資料配合溝通來了解問題領域的資料特徵，然後進入資料庫設計，而 relational database design 又有傷腦筋的正規化（ normalization ）檢查，理論上有嚴謹的定義，但卻缺乏簡易可循的方法來驗證，在物件導向的分析與設計方法中能避免這些麻煩嗎？要如何滿足正規化的要求？由於大多數的應用系統底下採用的還是 relational database，**在物件導向的分析與設計的開發過程中，relational database 的部分是如何產生出來的**？

以用 three-schema architecture 來解釋，也就是當下層的 schema 改變時不會影響到上層的 schema，有兩種資料獨立性的定義：

1. logical data independence：當 conceptual schema 改變時不需要牽動 external schema 或 application program。例如我們改變了資料記錄中的欄位定義，沒引用到相關資料的資料集（ view ）不受影響，用到改變欄位的資料集需要重新定義，但是不會影響到使用該資料集的應用。

2. physical data independence：internal schema 改變時不需要牽動 conceptual schema 或 external schema。例如我們對資料庫檔案進行了一些整理，建立新的存取架構，而上層的 conceptual schema 並沒有改變。

5.6.3 傳統的資料模型

從資料庫發展的歷史來看，Network Data Model 與 Hierarchical Data Model 在 1960 年代就出現了，等於早期在資料庫的領域中，大家對於資料的描述與看法都圍繞在這些資料模型中，1970 年 relational data model 的出現對於資料庫

系統產生了很大的影響。我們把這些出現較早的資料模型歸類於傳統的資料模型。一般說來，早期的資料模型不像近代的資料模型那麼複雜，但是以資料模型的特徵來說，都是五臟俱全。

各種資料模型的差異在什麼地方呢？基本上，資料模型賦與我們描述資料的能力，是否能很清楚地描述資料決定於資料模型所提供的描述方法，進階的資料模型在描述能力上通常會比較好一些，至少會比較完整。**以 semantic data model 為例，裡頭就引入了類別（class）與次類別（subclass）的觀念，成為物件導向資料模型中沿用的特性**。關聯式資料模型（relational data model）的地位有稍微特殊一點，因爲關聯式資料模型也出現得很早，但是模型本身具有簡易的觀念與完整的理論基礎，所以在商業市場上大受歡迎。

5.6.3.1 個體 - 關係模型（ER model, Entity-Relationship model）

個體 - 關係模型（ER model, Entity-Relationship model）是資料庫應用在開發時經常使用的資料建模（data modeling）方法，許多資料庫的設計工具就是使用 ER model 爲基礎。其實 ER 模型有再被擴充爲 EER（enhanced ER），不過 ER 原本的描述能力已經相當完整。在資料庫系統的領域中，ER model 有相當重要的地位，一般認爲它的 modeling 能力很強。

個體（entity）與屬性（attributes）

ER 模型利用個體（entity）、屬性（attributes）與關係（relationship）來表示資料，entity 代表現實世界上的事物，有獨立存在的特性，因此一個 entity 可以用來代表一個人、一輛車等事物，或是代表一種存在的觀念，例如一個課程、一家公司等。至於 entity 的詳細特徵則使用 attributes 來描述，譬如員工是 entity，則該員工的姓名、性別、出生日期等資料就是 attributes，attributes 的值就是儲存在資料庫裡的資料。到目前爲止，ER model 跟一般的資料模型都蠻相似的。不過 ER model 對於 attributes types 做了以下的分類：

- Simple attributes 與 composite attributes：假如一個 attribute 需要再由幾個 attributes 來描述，則該 attribute 就是一種 composite attribute，至於單純的 attribute 就不需要再由其他的 attributes 來描述。

- Single-valued attributes 與 multivalued attributes：有的 attribute 只有單一的值，但是有的 attribute 卻有多個值，例如某人的學歷可能就由多個值所組成，這樣的 attribute 就是 multivalued attributes。

- Stored attributes 與 derived attributes：當我們知道某位員工的出生年月日以後，可以直接計算其年齡。因此年齡可以看成是一種 derived attribute，出生年月日則是 stored attribute。

ER model 也支援空值（null value）的觀念，null 是一種很特別的值，假如某位員工最高學歷是高中，則其大學學歷的欄位無法填入，這並不代表資料是空的，這時候就可以填入 null，所以 null value 跟沒有值是不同的。

擁有相同的 attributes 的 entity 可以說屬於同一種 entity type，在 ER diagram 中，entity type 用方形（ rectangular box ）來表示，裡頭有 entity type 的名稱，其 attributes 則以橢圓形來表示，裡頭有 attributes 的名稱，而且以直線連到所屬的 entity type。

注意 entity 與 entity type 的差別，entity 是實際的資料，代表 entity type 的一個 instance，就好像程式語言中的 data type 與 variable，data type 是型式，variable 代表實際的資料。Relationship 表示 entity types 之間的關係，relationship 本身也可以有 attributes。從資料庫的觀點來看，屬於某種 entity type 的 entities 的集合也稱爲 entity set，代表資料庫目前的內容。

名稱急轉彎

ER model 中屬於相同的 entity type 的 entities 組成 entity set，我們說 entity type 是這些 entity 的定義（ schema ），也稱為其 intension，而 entity set entity set 則是該 entity type 的 extension。試著記得這些名稱，讀原文書或是 paper 時會有幫助的。

Entity type 的 attributes 中有一個或多個 attributes 組成 key attribute，其值在 entity set 中是唯一的，就像 relational table 中的 primary key 一樣。在 ER diagram 中，key attribute 會加底線來表示。不要小看 key attribute，我們知道物件導向系統中的物件通常會有 unique identifier，所以可以區分不同的物件。

ER model 中的 entity 假如所含的 attribute values 都一樣的話，其實是無法區分不同的 entities，只是有了 key attribute 的觀念的話，倒是可以比照 relational database 的情況，以集合的 no duplicate 與 key 的 uniqueness 來區分不同的 entities。自己沒有 key attributes 的 entity type 也稱爲 weak entity types，一般的 entity type，即 regular entity type，有 key attributes，也稱爲 strong entity types。

關係（relationship）

entity types 之間的關係叫做 relationship types，一個 relationship type 定義兩個或多個 entity types 之間的關係，像圖 5-9 中的供貨就是一個 relationship type，定義供貨商、訂單與商品之間的關係，R1 是一個 relationship instance，建立 S1、O1 與 P1 的關聯。由於參與這個關聯的 entity type 有 3 個，所以算是一種 ternary relationship。一個 entity type 參與的 relationship instance 的數目稱爲 cardinality ratio。像供貨商與商品之間的關聯的 cardinality ratio 爲 M:N。

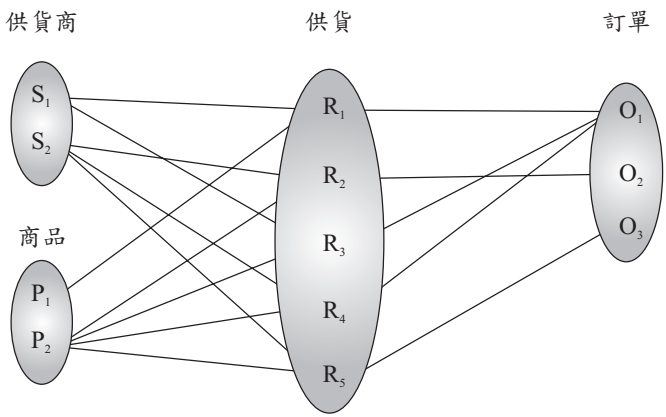


圖 5-9　三重關聯（ternary relationship）

在遞迴關聯（recursive relationship）中，一種 entity type 參與同一種 relationship type 兩次，例如圖 5-10 中的員工 entity type 與管理 relationship type，這時候要引入所謂的角色（role）的觀念，以圖 5-10 的例子來說，一個經理級的員工會管理其他的員工，所以在管理的關聯中扮演第 1 種角色，被管理的員工則扮演第 2 種角色，這樣圖 5-10 就可以很清楚地表示管理上的階層關係。

這裡要注意 entity type 與 relationship type 都是指類型（type），不是實際的資料，而 entity 指實際的資料，relationship 代表實際資料之間的關係。

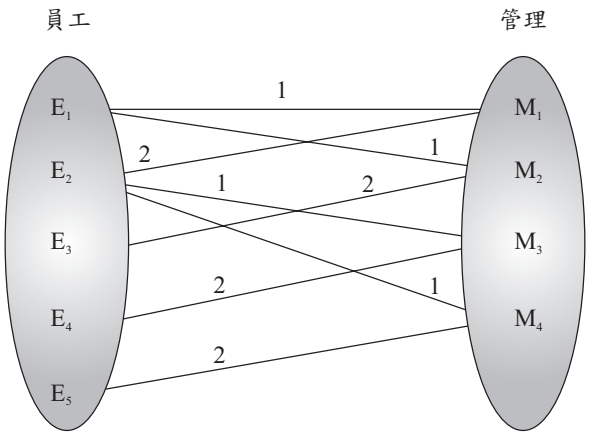


圖 5-10　遞迴關聯（recursive relationship）

表示法（notation）

圖 5-11 顯示 ER diagram 的基本表示法，用這樣的表示法可以畫出資料模型圖，讓人了解資料的定義與資料之間的關聯。這跟其他的資料模型都有一些相當類似的地方，假如需要眞正地使用 ER model，就要熟悉這樣的表示法。

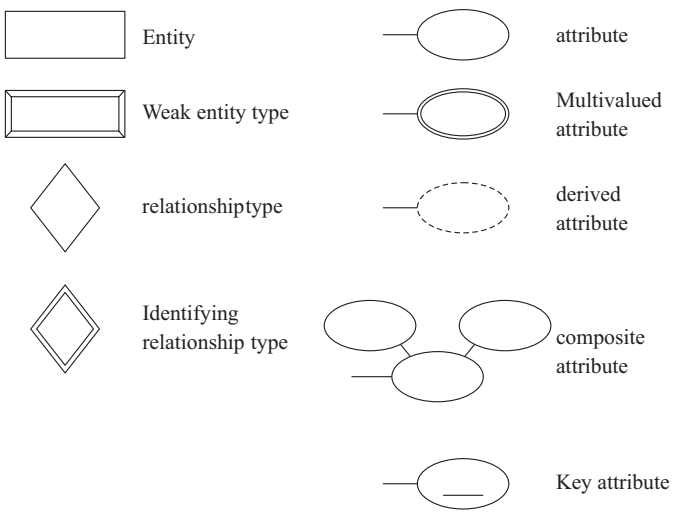


圖 5-11　ER diagram 的表示法

學習評量

1. 試說明系統模型有什麼樣的功能。
2. 試說明資料流程圖描述的是什麼。
3. 資料流程圖對於後續的軟體系統開發程序有什麼影響？
4. 試說明程序的邏輯塑模（logic modeling）會得到的結果。
5. 爲什麼資料流程圖算是一種結構化分析的技巧？
6. 試描述資料塑模（data modeling）的工作。
7. 試由一些企業使用的表單中描述裡頭使用的資料的特徵。
8. ER model 跟關聯式的資料模型（relational data model）有什麼差異？
9. 在系統分析的過程中，有那些問題可以幫助我們了解一個資訊系統的資料特徵？
10. 請參考 ER model 的表示法，試著畫出一個自己熟悉的系統的 ER 模型圖。思考一下，模型裡頭的內容對於系統開發提供了什麼樣的有用訊息。