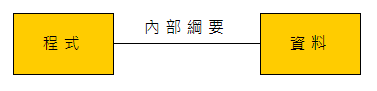
**資料庫的架構/種類/功能**

1. 架構發展
   1. 無綱要架構(Non-Schema Architecture) : 程式與資料沒有分開



* 1. 單一網要架構(One-Tier Schema Architecture) : 程式與資料分開，可從檔案讀取或由鍵盤輸入



* 1. 兩層綱要架構(Two-Tier Schema Architecture) : 產生外部綱要(External Schema)使資料看起來符合使用者需求與內部綱要(Internal Schema)負責實際資料在檔案中的儲存結構；當資料的儲存結構或方式改變時，應用程式也必須跟著修改。這也就是所謂的「資料相依」(Data Dependence)



* 1. 三層綱要架構(Three-Tier Schema Architecture) :

資料庫網要(Database Schema)是指整個資料庫的描述，即描述整個資料庫儲存定義資料

綱要(Schema) : 描述定義的資料

ANSI/SPARC 架構 :

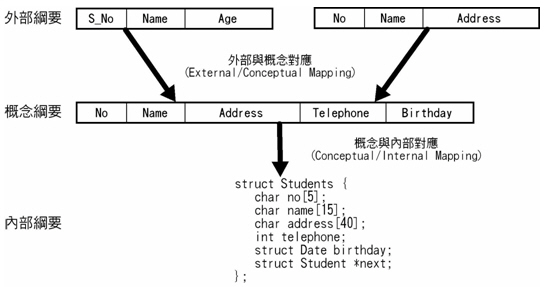
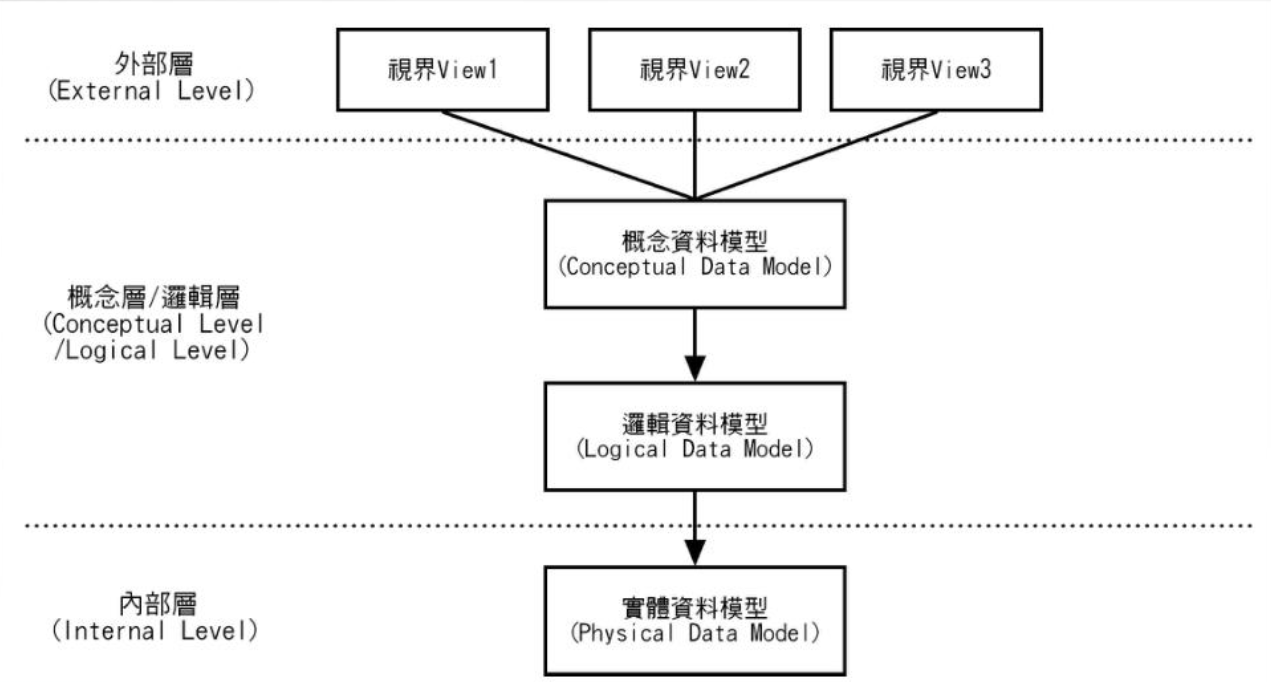
1. 外部層(The External Level) : 代表使用者所看到的資料庫系統，會因為不同的使用介面與應用程式而不同。
2. 概念層(The Conceptual Level) :資料庫內容的表示方式，形式與實際資料的儲存方式沒有關係，它與使用者所看到的「外部視觀」完全不同，只有一個view

資料模型透過資料抽象化來隱藏詳細儲存的資訊以及呈現資料庫的概念層與使用者需求

概念資料模型(Conceptual Data Model)又稱為高階資料模型(High Level Data Model) : 適合系統分析師與一般非電腦專業人員所使用

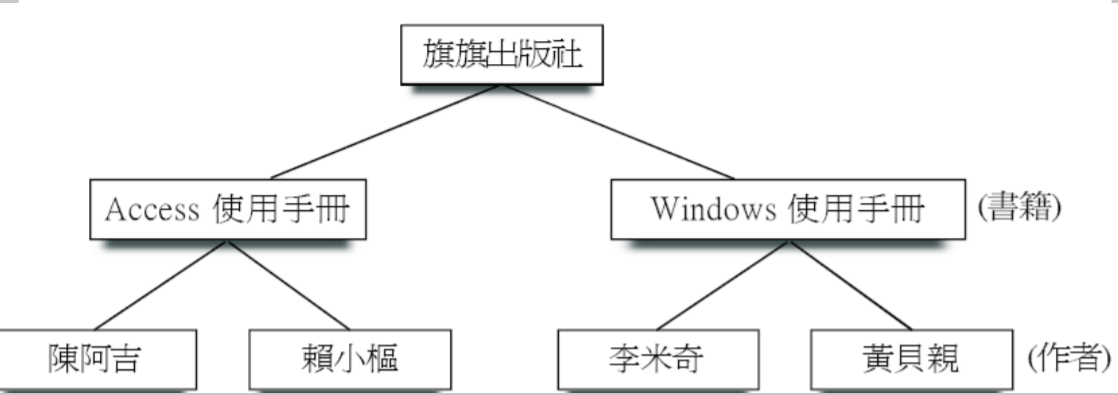
實體資料模型(Physical Data Model)又稱為低階資料模型(Low Level Data Model) : 適合系統分析師與程式設計人員之間使用

1. 內部層(The Internal Level) : 實際儲存的資料結構



1. 三層綱要架構(Three-Tier Schema Architecture)目的
   1. 使用者的角度 : 使用者觀點所看見的資料是獨立的、使用者並不用了解實際資料的儲存方式
   2. 資料庫管理師的角度 : 更改實際儲存結構不會影響使用者觀點的資料、改概念層的完整資料庫也不會影響使用者觀點的資料
2. 資料庫網要之間的對映(Mapping)與資料獨立(Data Independent)
   1. 外部與概念對映(External /Conceptual Mapping) : 所有的外部綱要(部份資料)都是對映到概念網要(完整的資料)以便資料庫管理系統知道如何將外部層的資料連結到哪一部分的概念
   2. 概念與內部對映(Conceptual /Internal Mapping) : 概念網要對映到內部綱要，以便資料庫管理系統可以找到實際儲存裝置的記錄資料後，建立概念網要的邏輯結構
   3. 資料獨立 : 應用程式不會因為資料結構或存取方法改變而必須更改，可以照常運作
3. 邏輯資料獨立性(Logical data independence) : 當更改概念網要一並不會影響外部綱要
4. 實體資料獨立性(Physical data independence) : 當更改內部綱要時並不會影響概念網要
5. 目的 : 使用者觀點所見的資料是獨立的、使用者並不用了解實際資料的儲存方式、更改實際储存結構不會影響使用者親點的資料、更改概念層的完整資料庫也不會影響使用者親點的資料
6. 資料庫的類型
   1. 階層式資料庫(Hierarchical Database) : 是採用樹狀的結構, 將資料分門

別類儲存在不同的階層下

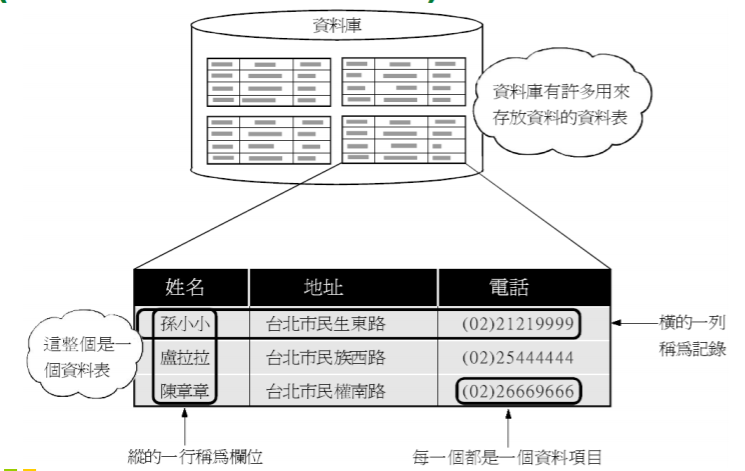


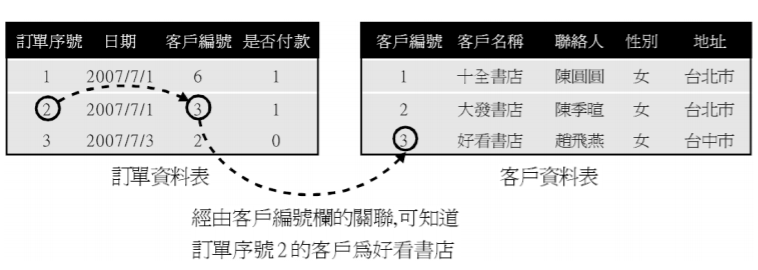
* 1. 網狀式資料庫 (Network Database)



* 1. 關聯式資料庫(Relational Database) : 以 2 維的矩陣來儲存資料，而儲

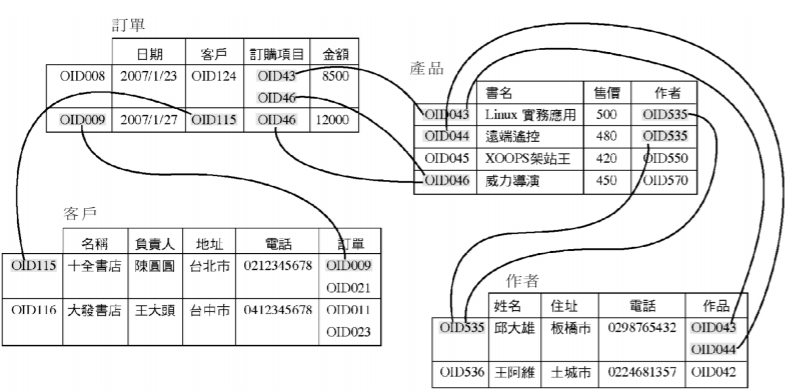
存在欄、列裡的資料必會有所 “關聯”





🡪 Microsoft SQL Server、SyBase、Informix、MySQL、PostgreSQL、Access...等

* 1. 物件導向式資料庫(Object-Oriented Database) : 物件導向資料庫是以物件導向的方式來設計資料庫, 其中包含了物件的屬性、方法、類別、繼承等特性



1. 處理架構
   1. 單機架構
   2. 大型主機 / 終端機架構
   3. 主從式架構 (Client / Server)
   4. 分散式架構