



Uppsala, Sweden, by Frank S.C. Tseng  
(<http://www2.nkfust.edu.tw/~imfrank>)

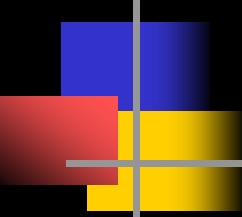
# 第一章 資料庫系統



# 本章內容

---

- 1.1 前言
- 1.2 資料庫系統的組成
- 1.3 使用資料庫系統的優點
- 1.4 使用資料庫系統的缺點與注意事項
- 1.5 資料獨立 (Data Independence)
- 1.6 完整的資料庫系統架構
- 1.7 資料庫管理系統



# 前言

- 「資料」(Data)：用以表示某項事實的語言或符號。
- 「人力資源 + 有效資料」為企業最重要資產。
- 資料 (Data)→ 資訊 (Information)→ 知識 (Knowledge)→智識 (Intelligence) →智慧 (Wisdom)。
- 資料庫管理系統→資料倉儲系統→知識管理系統。
- 使用資料庫乃企業電腦化後的必然結果。
- 架在資料庫上可開發更多元化的系統：
  - 決策支援系統 (DSS)、策略資訊系統 (SIS)、專家系統 (ES)、地理資訊系統 (GIS)、主管查詢系統 (EIS)，...



# 資料的分類 (型態觀點)

- 「資料」(Data)：用以表示某項事實的語言或符號。
- 資料主要用來記錄人類活動範圍中各種相關事物之概念。
- 從型態來看，電腦可以處理的資料有以下兩大類：
  - 量化資料 (Quantitative Data)：以數值所表示的資料
  - 質化資料 (Qualitative Data)：以文字所表示的資料



# 資料的分類 (型態觀點)

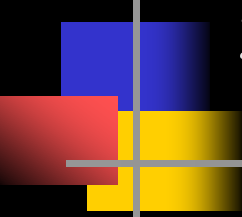
- 量化資料 (Quantitative Data)：以數值型態所表示的資料，可能會具有某種機率上的分佈狀況 (Probability Distribution)，又可以分成：
  - 連續型資料 (Continuous Data)：具連續性而且不可數的數字資料。例如： $\{3.14159, \dots\}$
  - 離散型資料 (Discrete Data)：可以一個個區分出來的數字資料。例如： $\{1, 3, 5, 7, 9, \dots\}$
- To be continued...



# 資料的分類 (型態觀點)

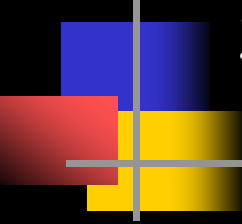
- 質化資料 (Qualitative Data)：以符號型態所表示的資料，又可以分成：
  - 無序型資料 (Nominal Data)：僅有名義 (Nominal) 上的差別，並沒有順序上的差別。如：個人興趣有{'爬山', '音樂', '閱讀'}，或是車款顏色有{'紅', '白', '藍', '黑', '綠'}。
  - 有序型資料 (Ordinal Data)：資料有順序上的意義。如：月份{'1月', '2月', ..., '12月'}，或是學生身分有{'博士生', '碩士生', '大學部'}。





# 資料的分類 (明確性觀點)

- 如果用資料是否明確的觀點來看，又可分成
  - 「明確資料」 (Definite Data)：在各條件均充分下所取得的資料。
  - 「不明確資料」 (Uncertain Data)：依不明確狀況有多種型式，以下介紹常被討論的兩種：
    - 隨機型 (或機率型) 資料 (Probabilistic Data)
    - 模糊性資料 (Fuzzy Data)



# 資料的分類 (明確性觀點)

- **明確資料 (Definite Data)**：在各條件均充分下所取得的資料。用**集合論 (Set Theory)** 來描述/解釋。
- **隨機性資料 (Probabilistic Data)**：在某種條件不充分的情況下，對資料賦予機率，以表示其隨機程度。可以用數學上的**機率論 (Probability)** 以及**統計學 (Statistics)** 來加以描述與解釋。
- **模糊性資料 (Fuzzy Data)**：用來記錄事件本身是模糊的，但發生與否則是明確而非隨機的情況。用數學上的**模糊集合理論 (Fuzzy Set Theory)** 來描述與解釋。





# 資料的分類（結構觀點）

- 「用來描述資料的資料」稱為「詮釋資料」(Metadata)、或「元資料」、「前設資料」、「後設資料」。
- 從結構觀點來看，資料又可分成
  - 結構化資料 (Structured Data)：具有固定結構的資料。例如：可以用表格型式呈現的資料。
  - 半結構化資料 (Semi-Structured Data)：屬於非純表格型式、也非純文字型式的資料，如：HTML、XML文件。這類文件由可表列的「資料」與其「詮釋資料」所構成。
  - 非結構化資料 (Non-Structured Data)：資料沒有固定的格式。這類資料如：文件、圖像、聲音、影片等。以文件為例，就有純文字檔、Word 文件、pdf 檔等不同的格式。



# 知識管理？從資料、文件角度出發！

- 網際網路蓬勃發展，大幅提昇知識的流通速度
- 掌握致勝關鍵，重點在於掌握知識 (Knowledge)。
- 現代化組織管理的重要議題：知識管理 (KM, Knowledge Management)。
- 落實有效知識管理：以善用知識、分享知識，提昇組織競爭優勢！
- Data → Information (Know what!) → Knowledge (Know how!) → Intelligence → Wisdom (Know why?)
- 知識讓我們知道 How？而智慧則讓我們了解 Why？
- 擁有知識只能讓我們「將事情做對」(Do the thing right!)；
- 具備智慧則可以讓我們選擇「做對的事情」(Do the right thing!)

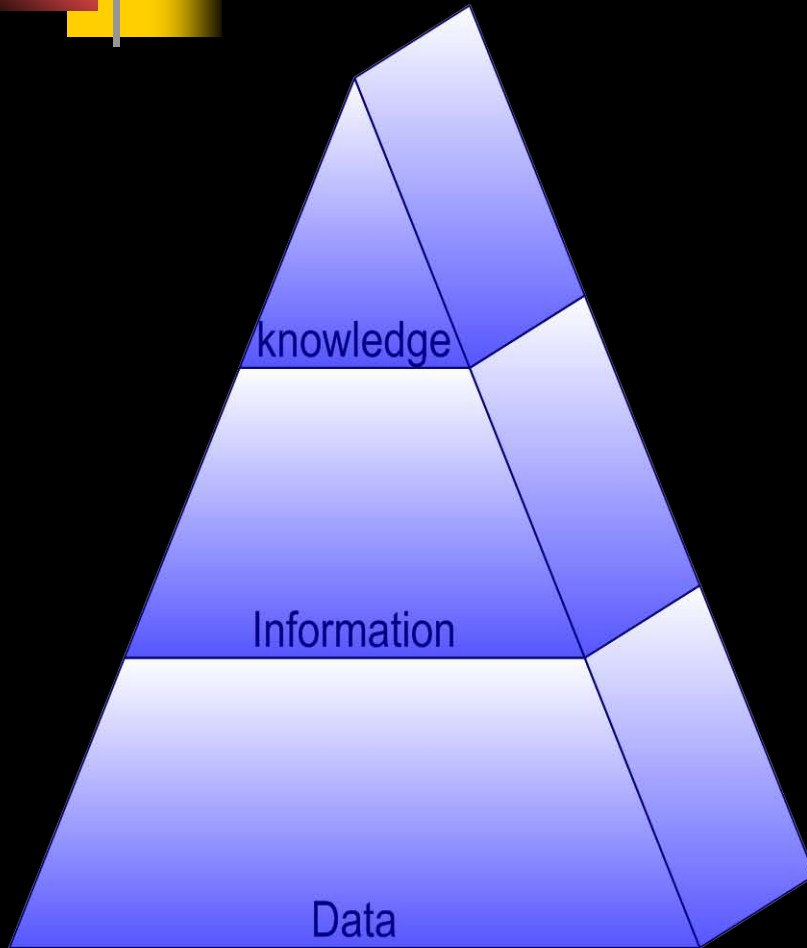


# 資訊管理的目標

- 資訊管理 (Information Management) 追求的目標：
  - 收集資料 (Data) 、
  - 產生資訊 (Information) ，
  - 由資訊衍生出具有價值的知識 (Knowledge) ，
  - 最後轉化成為商業應用上的商業智慧 (BI, Business Intelligence) 並予以妥善管理與分享。
  - 知識的建構過程: 內隱→外顯→文件化→標準化 (SOP)
  - 昨天的資料→今天的資訊→明天的知識→後天的智慧



# 系統階層關係



Knowledge Management Systems

Expert Systems

Decision Support Systems

Management Information Systems

Database Systems

Database Management Systems



# 資料庫系統的組成

- 建構資料庫系統的目的：
  - 透過電腦化方式將資料集中控制、管理。
  - 讓應用程式不受資料的真實存放方式所牽絆，達成所謂的「**資料獨立**」(Data Independence)。
- 組成資料庫系統的四大部分：
  - 使用者 (Users)、資料 (Data)、硬體 (Hardware)、軟體 (Software)



# 使用者

---

- 直接使用者 (End Users)：直接使用 SQL、Form 與資料庫溝通的使用者。
- 應用程式 (Application Programs)：透過程式介面與資料庫溝通的各類軟體程式。
  - 使用專屬的資料庫程式庫：Native DB Access Library
  - 嵌入式 SQL (Embedded SQL)：結合高階語言與查詢語言
  - 使用標準資料庫存取程式庫：ODBC/JDBC 介面
- 資料庫管理師 (DBA—Data Base Administrator)：負責管理、維護資料庫，是很重要的角色。





# 直接使用者

---

- 查詢方式：
  - SQL (Structural Query Language) 、
  - 表單 (Menu-Driven) 或
  - 表格 (Form-Driven) 方式
- SQL 將在第七章詳述



# 應用程式

- 以高階語言呼叫專屬資料庫存取程式庫 (Native DB Access Library)，如：Sybase 系統的 DB-Library、CT-Library；或透過系統所提供的第四代語言 (Fourth Generation Language，簡稱 4GL) 來完成
  - 4GL：說明 “What to do?” 即可，而非 “How to do?”
- 「嵌入式結構化查詢語言」 (Embedded SQL)
- 使用標準資料庫存取程式庫: ODBC/JDBC/OLE DB 介面



# 資料庫管理師

---

- 扮演使用者與資料庫管理系統間的橋樑
- 工作相當專業化 (大多數系統有認證考試)
- 從業人員最好具備企業與電腦方面的專長
- 資訊管理的核心
- 工作份量繁重，通常由一組專業人士所組成
- 主要工作如下頁所示：



# 資料庫管理師的工作

---

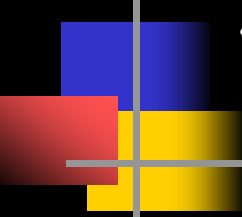
- 決定資料的儲存結構
- 提供一個中央控制的整合資料庫，協調各部門以決定資料的呈現方式
- 負責維護資料庫的綱要 (Schema)
- 監督並調整資料庫的效能
- 規劃適當的防範措施以防止系統發生錯誤與資料的損毀



# 資料庫管理師的工作 (續)

---

- 安全性控制 (Security Control) 與整合性檢查 (Integrity Checking)
- 對資料作備份 (Backup)，與回復原貌 (Recovery)
- 舊資料的轉換：保持原來的資料得以在新系統上正常運作
- 這些工作需要借助某些軟體工具來完成，如下頁所述



# 資料庫管理師的工具

---

- 大量的資料拷貝與轉換 (Bulk Copy)
- 對整個資料庫做備份的工具或指令 (Dump and Backup Utility)
- 製作磁碟複本 (Disk Mirroring) 的指令
- 統計、分析資料庫的使用狀況
- 稽核追蹤 (Audit Trail) 的功能
- 管理系統目錄 (System Catalog 或稱為 Data Dictionary) 的工具





# 資料

- 「運算資料」 (Operational Data)：資料庫中所存放的資料，也是使用者可以存取、運算的資料，此類資料必須要完全整合，以提供使用者共享資料的目標。「系統目錄」(屬於一種「詮釋資料」 Metadata) 也屬此類。
- 「異動記錄」 (Transaction Log)：依照使用者所下達之命令，而自行產生的記錄資料，對於資料庫管理師 (DBA) 而言則有重大意義
- 備份的對象：「運算資料」與「異動記錄」



# 硬體

---

- 電腦主機
- 磁碟機, 共同儲存設備 NAS (Network Attached Storage), SAN (Storage Area Network), ...
- 光碟機、光碟櫃
- 備份裝置：磁帶機、可讀寫光碟機：絕對必要
- 異地備援計畫 (Remote Backup)...
- UPS (Uninterruptible Power System) 有必要、
- 千萬不可以用買個人電腦的角度來看企業的資訊系統規劃



# 軟體

- 資料庫管理系統 (DBMS—DataBase Management System): 如 Oracle, SQL Server
- 架在 DBMS 上的應用程式 (Application Programs), 我們也稱之為 “資料庫系統”: 如人事, 行銷, 會計等系統
- 中介軟體 (Middleware): 某些介於上述兩者之間的軟體, 以專門負責像: 「負載平衡」(Load Balancing) 或異質性資料整合的工作 (如: ODBC 資料庫驅動程式)



# 應用程式

---

- 公文系統、學術研討會管理系統
- 人事管理系統、教務系統、設備管理系統
- 會計資訊系統、進-銷-存管理系統
- 電子佈告欄系統、部落格 (BLOG) 系統
- 基金投資分析系統、產學合作與學生實習媒合系統
- ... (請同學自行組成 3 ~ 5 人一組，學期末前使用 SQL Server 2008 配合選定的開發平台或語言開發一個上述範例程式進行展示)



# 使用資料庫系統的優點

---

- 簡潔性
- 增快擷取資料的速度
- 最新的資料 (Up-to-Date Information)
- 減少大量重覆儲存的資料
- 減少不一致的資料
- 資料得以共享
- 達成文書資料標準化的目的



# 使用資料庫系統的優點 (續)

---

- 保密性 (Security) 提高
- 資料具有整合性 (Integrity)
- 提供決策支援 (Decision Support) 服務
- 「資料獨立」 (Data Independence)
- 透過適當的資料模式 (Data Models) 將資料之間的複雜關係表現出來
- 快速備份 (Backup) 與回復 (Recovery) 資料庫
- 提供不同的視界 (View)





# 使用資料庫系統的 缺點與注意事項

- 「水能載舟，亦能覆舟。」
- 企業仰賴電腦越深，就有越危險的潛在危機
- 電腦化的首要重點不是儘速建立功能強大的資訊系統，而是仔細的分析、檢討如何防範電腦對企業可能造成的所有傷害。
- 對資料庫管理系統的產品要先探討其缺點何在？再要求其優點



# 使用資料庫系統的缺點

- 若無良好的控制，安全堪慮
- 若無良好的控制，資料的正確性也令人擔憂
- 額外的預算：前後採購的軟、硬體不相容問題，人員教育訓練問題，使用人員的排斥等。
- 資訊管理部門扛起所有責任，負擔越來越重
- 系統一旦停擺，整個企業組織也整個跟著癱瘓



建議在電腦化之前要事先規劃替代方案



# 使用資料庫系統的缺點 (續)

---

- 過度膨脹，引發管理上的困難
- 每隔一段時間便檢討、並規劃未來的發展藍圖
- “制度面” 與 “人” 的因素才是最大變數
- 技術反而不是系統建置與運轉時的最大考量
- 管理與技術並重，避免人為的錯誤與阻撓



# 資料安全的維護

---

- 威脅資料安全的原因
  - 天然災害：適度預防並加強資料的備份。
  - 機件故障：適度預防並加強資料的備份。
  - 人為過失：透過加強訓練來避免。
  - 惡意破壞：最令人頭痛的一大問題。
- 惡意破壞尚待立法及適當的宣導與觀念建立



# 電腦犯罪的類型

---

- 員工因貪心、經濟困難、個人因素挺而走險
- 無意間發現資料安全上的漏洞，覺得好玩
- 私人恩怨，致使離職後進行對原公司的報復
- 商場競爭上的工業或國際間諜
- 想瞬間成名的「電腦駭客」(Hacker)
- 喜歡捉弄他人電腦系統的犯罪奇才
- 請不要將電腦犯罪當成是一種榮耀



# 制訂安全對策的考量重點

- 選用能正確反應資訊安全需求的設備
- 系統發展階段將資訊安全的需求加入
- 日常運作即配合人員管制、使用管理與系統稽核
- 制定的各種安全基準與制度
- 由另一組稽核人員定期與不定期追蹤、檢查，以評估系統的安全性





# 制訂安全對策考量重點 (續)

---

- 隨時檢討、引進新的安全裝備及實行辦法
- 對磁碟機做妥善的規劃，製作磁碟複本 (Disk Mirroring)，以保護資料
- 制定完整的備份策略，定期備份
- 規劃可人工處理部份替代方案，以防萬一



# 可能產生的行政問題

---

- 不切實際的期望
- 期望「立竿見影」，造成規劃步調出現落差
- 主管的異動造成系統目標與政策不連貫
- 球員兼裁判的弊病發生
- 只加強軟、硬體設備，忽略了「人」的因素
- 高階主管對資訊安全的不了解
- 資訊安全常被列為次要的議題



# 可能產生的行政問題 (續)

---

- 絕對避免私相授受電腦使用權利的情況
- 不了解資訊安全之政策、程序、準則與懲戒
- 沒有定期接受最新的資訊安全技術訓練
- 管理系統與資料的權利義務未界定清楚
- 聘用員工時了解員工過去的記錄
- 未報告單位中的電腦犯罪、舞弊或偷竊事件
- . . .



# 資料獨立 (Data Independence)

- **反義詞** 「資料相依」 (Data Dependence) : 只要資料的儲存結構 (Internal Data Structure) 或存取方式 (Access Method) 一更改，則該應用程式就必須要跟著修改
- 「**資料獨立**」 (Data Independence) : 應用程式不會因內層結構的改變而需更改
- 我們希望所有應用程式的開發都是「**資料獨立**」的，因此大多數系統架構設計也都是朝這個方向在努力



# 動態鏈結串列 vs. 樹狀結構

- 儲存結構所使用的是動態鏈結串列 (Linked List)

```
struct student {  
    int    no;  
    char   name[10];  
    int    age;  
    char   address[30];  
    char   phone[10];  
    student *next;  
};
```

```
pointer = head;  
found = FALSE;  
while (pointer != NULL) {  
    if (pointer->age != 21) { pointer = pointer->next; }  
    else { found = TRUE; }  
} /* not found */
```



# 動態鏈結串列 vs. 樹狀結構

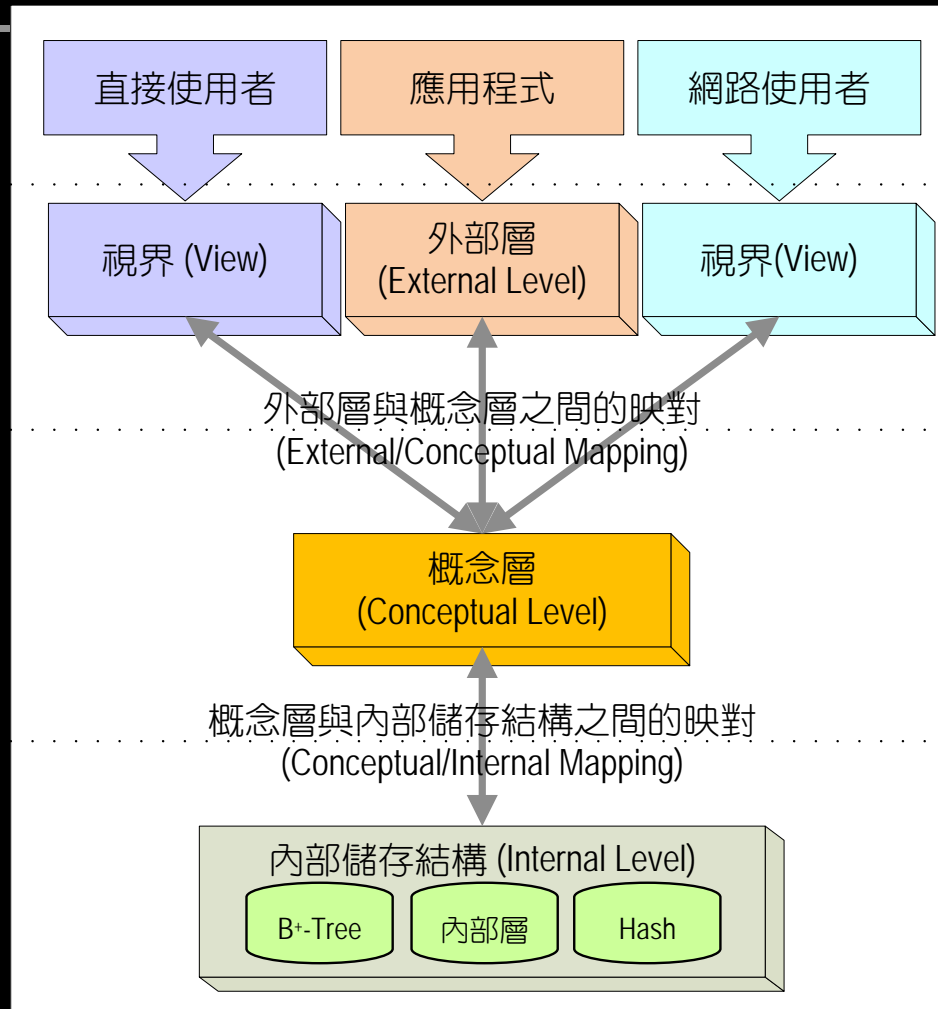
- 儲存結構所使用的是樹狀結構 (Tree Structure)

```
struct student {  
    int    no;  
    char   name[10];  
    int    age;  
    char   address[30];  
    char   phone[10];  
    student *left;  
    student *right;  
};
```

```
pointer = root;  
found = FALSE;  
while (pointer != NULL) {  
    if (pointer->age > 21) { pointer = pointer->left; }  
    else if (pointer->age < 21) { pointer = pointer->right; }  
    else { found = TRUE; /* found */ }  
} /* not found */
```

資料結構一換，許多程式碼就要重寫  
這樣的現象稱為：資料相依 (Data Dependence)

# 完整的資料庫系統架構



## ■ ANSI/SPARC 資料庫系統架構



# ANSI/SPARC 資料庫系統架構

---

- 內部層 (Internal Level)：實際儲存資料的結構
- 外部層 (External Level)：使用者看到的部份。
- 概念層 (Conceptual Level)：為內部層與外部層之間的橋樑 (是資料庫管理師 (DBA) 所看到的整體部份)。





# ANSI/SPARC 架構 (續)

---

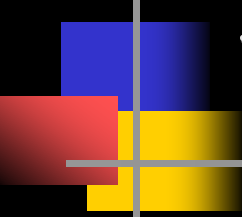
- 各層的資料定義，也就是所謂的綱要 (Schema)
- 概念層以上的綱要設計一般稱為「邏輯資料庫設計」 (Logical Database Design)  
是系統分析的首要工作
- 內部層的綱要與儲存結構之訂定則稱為「實體資料庫設計」 (Physical Database Design)



# 外部層 (The External Level)

---

- 使用者直接面對的是外部層
- 關聯式資料庫系統的外部層是「視界」(Views)
- 基底關聯表 (Base Tables) 對應到「概念層」
- 透過結構化查詢語言 (SQL) 存取資料庫
- 應用程式：使用第四代語言 4GL 或將 SQL 嵌在 C、COBOL 等高階語言中來存取資料
- 終端使用者：直接使用 SQL 或表單型式 Form
- 「視界」是一種「虛構的關聯表」(Virtual Table)



# 嵌入式 SQL vs. DB-Library

---

```
main()
{
    int a;
    if (a != 0) { ... }
    else { ... }
    EXEC SQL Update students set name = 'Frank' where id = 88;
    ...
}
```

- EXEC SQL 可以用呼叫 DB-Library 來取代



# 概念層 (The Conceptual Level)

- 整個資料庫的實體，存放所有內含資訊表式法
- 由資料庫管理師來負責維護與管理
- 以表格方式呈現給使用者看（基底關聯表）
- 包含某些整合限制條件 (Integrity Constraints)

Students

<u>no</u>	name	age	city	...
1	John	23	Taipei	...
2	Mary	21	HsinChu	...



# 內部層 (The Internal Level)

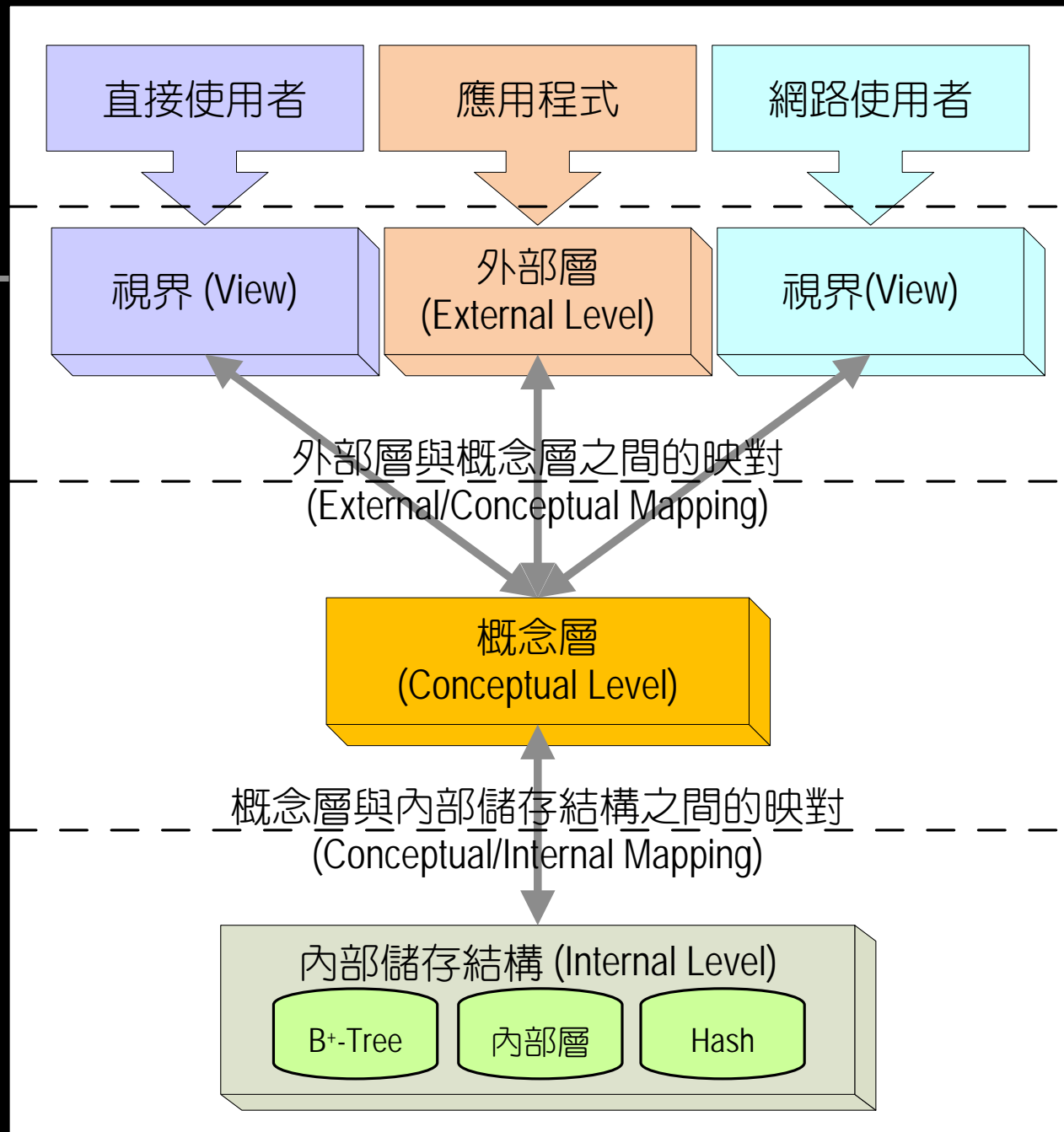
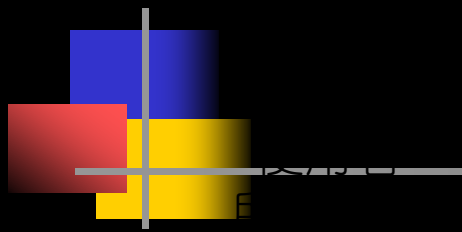
---

- 在「內部層」上，要考慮：
  - 那些資料需要加上索引 (Index)，增進擷取效率
  - 要採用何種內部儲存技術：B+-Tree 或雜湊式的儲存方式 (Hash-Based Methods)？
  - 在磁碟上如何將資料叢聚 (Clustering) 在一起
- 不考慮與裝置有關的存取細節，如：  
磁碟機有幾軌，大小為何，如何劃分等
- 因為：那是 BIOS 與 Driver 的事



# 各層間的映對 (Mapping)

- 各層之間的映對是協助我們達成資料獨立這個目標的主要基礎
- 內層結構改變時，只要改變內部層/概念層之間的映對 (Internal/Conceptual Mapping)  
概念層綱要 (Conceptual Schema) 不需異動
- 概念層結構改變則只要改變概念層/外部層之間的映對 (Conceptual/External Mapping)  
外部層綱要 (External Schema) 可維持不變





# 資料庫管理系統

---

- 資料庫管理系統 (Database Management System, 簡稱 DBMS):
  - 負責處理使用者存取資料庫要求的套裝軟體。
  - 分析查詢指令、
  - 檢查使用者的外部層綱要、相對的外部層/概念層映對、概念層綱要、概念層/內部層映對、以及資料的儲存結構，
  - 對儲存的資料庫執行指令所傳達的動作。
  - 當同時有大量使用者湧入時，必須做好「異動管理」(Transaction Management)





# 異動管理 (Transaction Management)

---

- 系統中可能有數個使用者同時對資料庫下達命令，要求資料庫管理系統完成工作。
- 對於資料庫的一個完整動作稱為一個「異動」(Transactions)
- 資料庫管理系統必須有效地做異動的管理，以防止同時執行的異動因交錯執行而發生不可原諒的錯誤。
- 異動的定義 (四大特性)：ACID [Jim Gray (1981)]  
1998 年 Jim Gray 獲得 Turing Awards.  
(<http://www.acm.org/awards/taward.html>)



# 異動的單元性 (Atomicity)

- 一個異動中的所有運算動作不是完全做完，便是完全不做！
- 提款
  - 由銀行帳戶扣錢，
  - 給錢，
- 此兩動作必須全部做完（稱為委任Commit），或完全不做（稱為撤回 Abort 或 Rollback）
- 只做其中一個動作會引起爭議

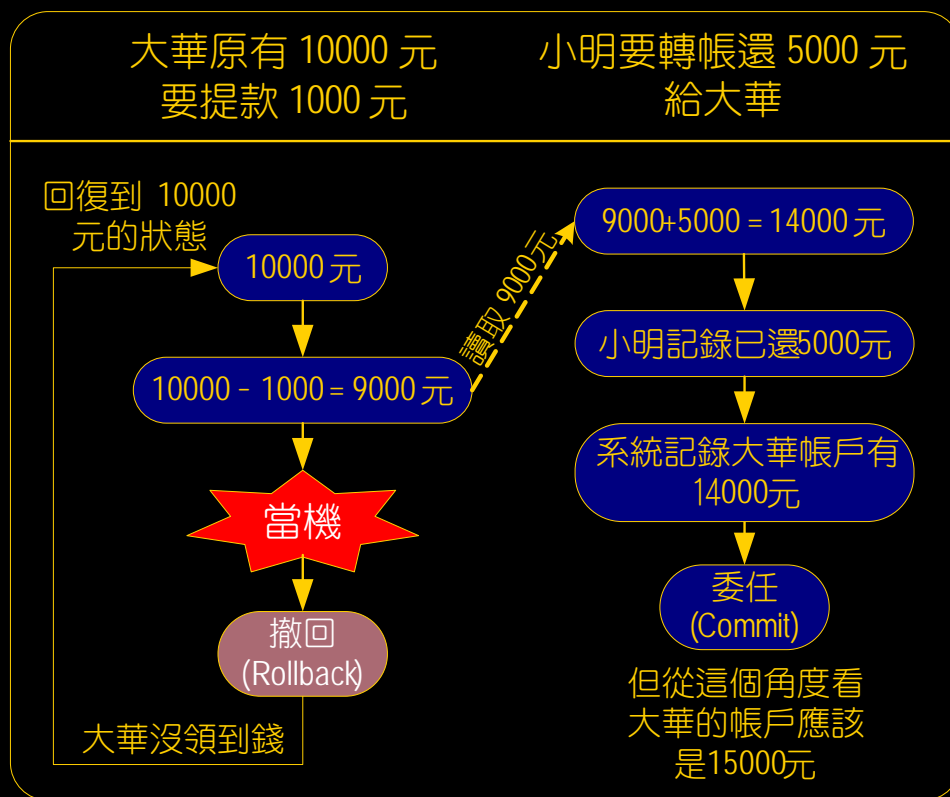


# 異動的一致性 (Consistency)

- 多個異動可以同時執行，但是要作好並行控制 (Concurrency Control)，產生宛如異動循序個別執行的一致性 (Consistency) 結果  
(將 DB 從某個一致性狀態帶往另一個一致性狀態)
- 舉例來說，下面兩件工作可以同時進行，但是結果必須與按某種順序排隊執行的一樣（誰先都算對）
  - 大華先提款
  - 小明轉帳

# 異動的隔離性 (Isolation)

異動在尚未委任前的中間執行結果不得讓其它同時在執行的異動存取





# 異動的持續性 (Durability)

- 如果異動執行過程完全正常，但在透過「委任」(Commit) 命令指揮資料庫管理系統，將異動結果反應到資料庫的過程中，系統卻發生錯誤，那麼在系統回復後，就應當將原來未做的後續動作完成。
- 「異動記錄」(Transaction Log) 是維持此特性的主角
  - 異動過程要不斷寫入異動記錄
  - 若遇當機，再根據異動記錄來回復



# 異動的定義

- 異動的範圍是由人來定義，而非任意將某些指令運算集合起來。
- 使用 BEGIN TRANSACTION...  
配合 COMMIT Transaction 或  
是 ROLLBACK Transaction 指令來規範

## BEGIN Transaction

Select ...

From ...

Where;

IF (exists (...)) ...

...

## COMMIT Transaction

*/\* 成功 \*/*

ELSE ...

...

## ROLLBACK Transaction

*/\* 失敗 \*/*

END IF



# 資料庫管理系統的基本功能

---

- 有組織地將資料儲存起來，並具備快速的資料存取技巧。
- 有效地管理資料庫的綱要
- 提供一套高階的查詢語言
  - 資料定義語言 (Data Definition Lang.，簡稱 DDL)
  - 資料操作語言 (Data Manipulation Lang.，DML)
  - 資料控制語言 (Data Control Language，DCL)



# 資料庫管理系統的基本功能(續)

- 資料的安全管制 (Security Control)
  - 建立使用者的通行密碼 (Password)。
  - 針對資料的建立 (Insert)、刪除 (Delete)、查詢 (Select)、修改 (Update) 等使用權。
  - 使用視界 (Views) 來遮蔽 (Mask) 機密資料不給使用者看或查詢。

透過前述的 DCL 或 DML 來完成

- 備份、效能監控、確保資料正確性之工具





# 資料庫管理系統 vs. 檔案系統

---

- 資料庫管理系統是架在檔案系統上的軟體
- 有些資料庫管理系統為了效率與安全性上的考量，直接捨棄檔案系統不用，自行控制磁碟與磁帶機等週邊
- 資料庫管理系統比檔案系統多出了許多的功能如下頁所示



## 資料庫管理系統 vs. 檔案系統 (續)

---

- 提供定義及處理資料記錄與欄位的功能。
- 提供資料記錄與欄位的保密與整合功能。
- 提供系統目錄的管理功能。
- 提供異動管理 (Transaction Management) 與資料回復 (Recovery) 的功能。
- 提供優秀而快速的資料存取與管理功能。



# 檔案系統的限制

---

- 不知道檔案的內容與邏輯結構
- 沒有 (或僅有少許) 資料整合與資料欄位、記錄的安全控制。
- 沒有提供異動管理與資料回復的功能
- 沒有存放欄位、記錄之邏輯關係的系統目錄
- 難以達成資料獨立的目標



# 著名資料庫管理系統產品

---

- Oracle — 「美商甲骨文公司」。<http://www.oracle.com>。
- Informix-OnLine Workgroup Server (OWS)、Informix SE (Standard Engine)、Informix Dynamic Server (IDS) — 「英孚美公司」(<http://www.informix.com>) 已被 IBM 購併
- Sybase Adaptive Server — 「賽貝斯公司」。  
(<http://www.sybase.com>)



## 著名的資料庫管理系統產品(續)

- Ingres — 加州柏克萊大學的 M. Stonebraker 教授所發展，現在是 Computer Associates 的一個部門。  
(<http://www.naiua.org>, [www.ca.com](http://www.ca.com))
  - Ingres 加上了物件的功能後，稱為 Postgres，現在是開放原始碼軟體，並改名稱為 PostgreSQL (<http://www.postgresql.org>)
- Microsoft SQL Server 2008 — Microsoft 公司所發展，最早是和 Sybase 合作發展。(http://www.microsoft.com)
- DB2 Universal Server — IBM 公司所發展。  
(<http://www.software.ibm.com/data/db2/index.html>)。



# 著名的資料庫管理系統產品

## (<http://www.connectionstrings.com>)

- Progress — Progress Software Corp. (<http://www.progress.com>)
- Gupta SQL Base — 由 Gupta Corp. (<http://www.gupta.com>)
- Teradata Database — 原由 AT & T Global Information Solutions 所發展，目前併入 NCR (<http://www.ncr.com>)。
- Rdb — 由 Dec 所發展 (<http://www.digital.com>)，先前由 Compaq 併購，目前併入 HP (<http://www.hp.com>)
- DBMaker — 由凌群電腦公司發展 (台灣本土公司所研發) (<http://www.dbmaker.com.tw>)
- MySQL — 算是目前最受歡迎的開放原始碼 (Open Source) 資料庫管理系統，有許多的商業應用都是使用 MySQL 配合 PHP (PHP: Hypertext Processing) 所開發出來的 (<http://www.mysql.com>)。

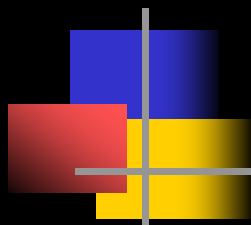


# 後關聯式資料庫管理系統

## Post-Relational DBMSs

---

- 推理式資料庫系統/專家系統
- 可擴充式資料庫系統
- 物件導向式資料庫系統
- 單一關聯表資料庫系統
- 歷史/時間資料庫系統
- 影像/空間資料庫系統
- 主動式資料庫管理系統



本章結束  
The End.