資料庫系統筆記 Ch. 2

> 參考書籍：
>
> 1. 《Databases System 7th Edition》— Ramez Elmasri, Shamkant B. Navathe.
> 2. 《數據庫系統基礎 第六版》— Ramez Elmasri, Shamkant B. Navathe，李翔鷹、劉鑌、邱海艷、陳立軍譯
>
> 筆記作者：黃漢軒

資料庫模型、模式與狀態

資料庫模型

- 主要可以分成三種不同類型的資料庫模型：
  - 高層 - 概念資料庫模型（Conceptual Data Model）
  - 低層 - 物理資料庫模型（Physical Data Model）
  - 介於高層與低層之間 - 表示資料庫模型（Representation Data Model）
    - 在資料庫中最常被使用，因為不會因為過於概念而拋棄底層概念，也不會因為過於底層而難以理解。
    - 包含了過時的網狀模型（Network Model）與層次模型（Hierarchical Model）。
- 在模型中可以看到以下的概念：
  - 實體（Entity）：泛指真實世界中的對象或概念。
  - 屬性（Attribute）：泛指真實世界中某個感興趣的特性。
  - 關聯（Relationship)：泛指實體之間的關聯。

資料庫模式

- 對於資料庫來說，資料庫模式（Database Schema）泛指對於資料庫的描述。
- 資料庫模式*一般認定上*並不會被頻繁的改變，它會在規劃資料庫時進行設計，在未來進行微調。
  - 通常伴隨著需求的改變，會導致資料庫會有所改變，稱為模式演化（Schema Evolution）。
- 在描述資料庫中所有表格的圖，我們稱為模式圖（Schema Diagram）
  - 絕大多數情況下，資料庫所有表格都可以呈現出模式圖，如下圖即呈現了一個資料庫中的模式圖。
  - 裡面的資料表格稱為模式構件（Schema Constructor），透過多個模式構件可以產生出模式圖。

STUDENT

Name	Student_number	Class	Major

COURSE

Course_name	Course_number	Credit_hours	Department

PREREQUISITE

Course_number	Prerequisite_number

SECTION

Section_identifier	Course_number	Semester	Year	Instructor

GRADE_REPORT

Student_number	Section_identifier	Grade

資料庫狀態

- 資料庫的資料被稱為一個資料庫狀態（Databases State），或者可以稱為快照（Snapshot）。
- 資料庫狀態，即為資料庫中當前的具體值（Occurrence）或實例（Instance）的集合。
- 在初始灌檔進入資料庫時，資料庫即有第一次版本的資料狀態，稱為初始狀態。
  - 在之後的變更後每次都會產生一個資料狀態，即為版本上的變換。
  - 資料庫會確保每一次的資料狀態都是有效的，保證有效狀態（Valid State）。

三層模式結構與資料獨立性

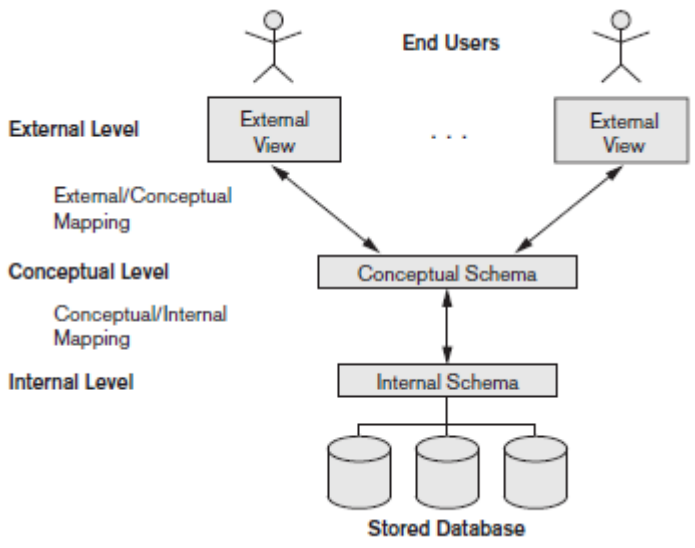
三層模式結構

- 資料庫具有三個主要特性：多視角、資料抽象化、資料獨立性。
- 我們用三層模式結構（Three-schema Architecture），來更好的實作這三個主要的特性，主要分成以下三層：
  - 內層（Internal Level）：也稱內模式（Internal Schema），

    使用物理資料庫模型來實作，主要用於描述資料庫的物理儲存特性，資料在實際上儲存於這層。
  - 概念層（Conceptual Level）：也稱概念模式（Conceptual Schema），

    使用表示資料庫模型來實作，主要用於描述資料庫整個結構。
  - 外層（External Level）：也稱視圖層，包含許多個外模式（External Schema）或用戶視角（User View），

    使用概念資料庫模型為基底，再使用表示資料庫模型加以實作，描述使用者感興趣的一部分。
- 在各層之間完成請求與將結果轉換的處理過程稱為映射（Mapping）。



資料獨立性

- 定義資料獨立性為，當改變某一層的模式（Schema）時，不會使他的上層模式發生改變。
- 可以分成兩類資料獨立性：
  - 邏輯資料獨立性（Logical Data Independence）：
    - 概念層改變時，不需要改變外層或應用程式的特性，只須改變與外層的定義與映射關係。
    - 例如：我們在一個表格新增了一欄欄位，在這個表格所做的操作，不需要對外部應用程式進行變更。
  - 物理資料獨立性（Physical Data Independence）：
    - 內層改變時，不需要改變概念層的特性，也不需要改變外層，只需改變與概念層的定義與映射關係。
    - 以內層上來說，即為儲存資料的硬碟的確切位置，較底層的部分。
    - 例如：為了提高查詢效率，我們提供了一種新的存取結構，我們也提供了一個新的路徑供概念層使用（內層映射），這樣對於概念層的情況下不需改變概念層的應用程式，從內層將資料傳送至概念層依然是正常運作的。
- 利用三層結構，使實作資料獨立變得更加簡單。

資料庫語言與介面

資料庫語言

- 對於三層的結構而言：
  - 對於外層與概念層上的溝通，資料庫設計者使用資料定義語言（Data Definition Language, DDL）來進行兩層上的映射。
  - 對於概念層與內層上的溝通，使用儲存定義語言 （Storage Definition Language, SDL）來進行兩層上的映射。
  - 額外會再使用視角定義語言（View Definition Language, VDL）來讓用戶視角與概念層上進行映射。
- 對於使用者說，使用資料操縱語言（Data Manipulation Language, DML），來透過 DBMS 進行資料庫上的 CRUD。
  - DML 又可以分成高等與低等：
    - 高等可以用簡潔的方式指定複雜的資料庫操作，通常指定要查詢哪些東西，稱為描述性語言（例如：SQL）。
    - 低等則是從資料庫查詢單獨的紀錄或對象，並對資料庫進行一系列的操作。
- 現今使用 SQL（Structured Query Language）來綜合 DDL、VDL 與 DML，但不包含 SDL。
  - 為了將 SQL 僅限在概念層與外層上，故不包含在內層的 SDL。

DBMS 介面

分成七類：

- 面相 Web 客戶端或瀏覽器，基於菜單的介面：使用 Browser 來進行資料庫的操作。
- 基於表單的介面：填寫表單來操作資料庫。
- 圖形使用者介面：使用 GUI 來操作資料庫。
- 自然語言介面：使用自然語言來理解與處理使用者描述的操作。
- 語音輸入與輸出：使用語音來進行資料庫的操作。
- 面相簡單參與使用者的介面：單純為了某些特定的 Naive User，給定他們功能性軟體來完成資料庫上的操作。

- 面相 DBA 的介面：包含一些僅由 DBA 能使用的操作，例如創立帳號、設定系統變數等等。