

Go 程式設計課 05

Go 與資料庫



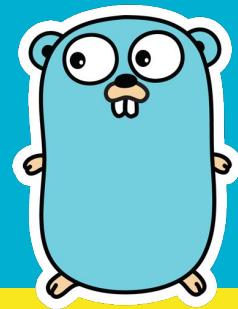
大綱

- ❖ 介紹資料庫
 - 嵌入式資料庫
 - 資料庫
 - SQL vs non SQL
- ❖ entity diagram 教學
 - 設計資料庫
 - 地震 Get and Create



大綱

- ❖ SQL 語法介紹
- ❖ ORM framework
- ❖ DB migration
- ❖ 快取資料庫介紹
- ❖ 資料庫實作



資料庫

資料庫 (Database) 是儲存在電腦中的資料集合，我們可以透過撰寫 SQL 有效率地對資料庫中的數據進行資料操作。大至銀行的存款資訊與交易資訊、小至手機的通話紀錄與通訊錄，背後都有資料庫在運作支撐。

本質上是能更高效操作 Disk (數據儲存裝置) 的工具。

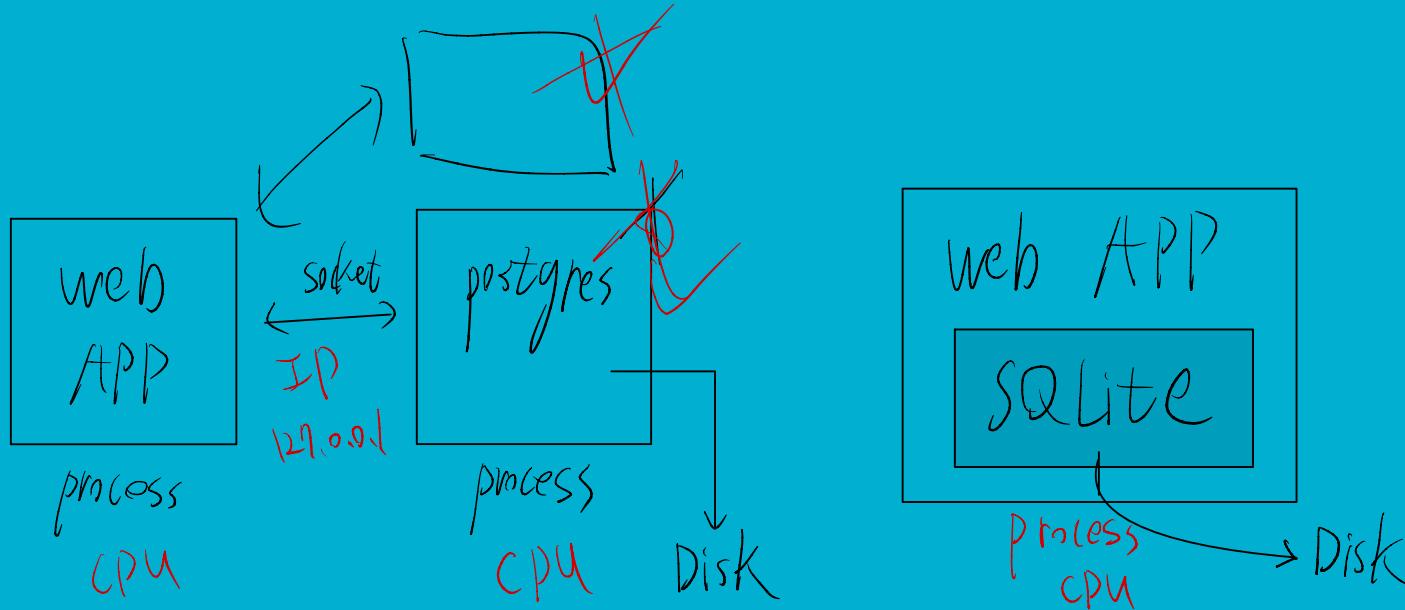
具有以下兩個特徵的資料集合會被稱為資料庫：

1. 資料觀測值具有屬性 (attributes)
2. 儲存有元資料 (Metadata)。元資料 (Metadata) 常見的解釋為「data about data」、「描述資料的資料」



嵌入式資料庫 vs 傳統資料庫

- 傳統資料庫: 獨立的伺服器程序, 通過網絡與應用程式通信
- 嵌入式資料庫: 與應用程式緊密整合, 由應用直接運行的資料庫



嵌入式資料庫的特點

- 無需獨立的伺服器,減少了複雜性
- 資料儲存在本地端檔案中,避免網絡通信開銷
- 通常體積小巧,資源佔用低
- 部署方便,適合嵌入到應用程式中

常用在: 主機遊戲數據存儲, 高性能分散式應用數據存儲, 嵌入式設備存儲

block chain!

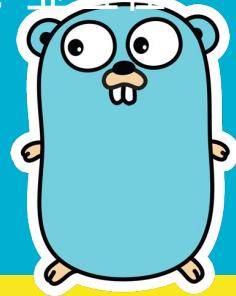


關聯式資料庫



目前有許多不同類型的資料庫技術，每個類型都有不同的儲存和讀取資料的方式，但到目前為止，最流行的還是**關聯式資料庫(RDBMS, Relational Database Management System)**

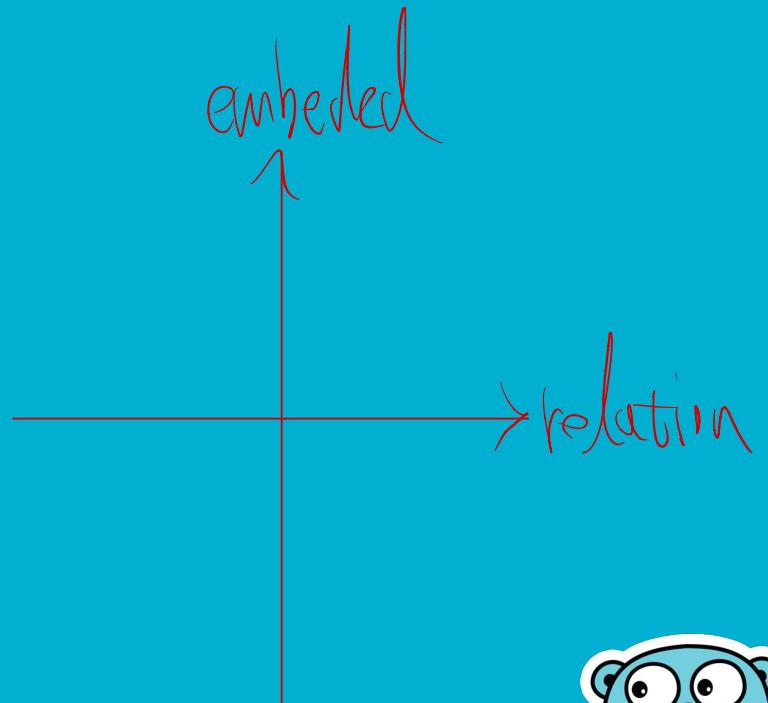
在關聯式資料庫中，資料存儲在 Table 中，Table 類似於 Excel 的表單，由欄和列組成的多個儲存格，Table 中有一個唯一的 key 來標識每一行數據。這些唯一的 key 可以用來將 Table 連接在一起。正是這種結構使我們能夠辨識和存取資料，建立起資料庫中其他區塊之間的關係。也正是這些關係，使得關聯式資料庫變得如此強大，並且在資料庫中提供資料的一致性。



關聯式資料庫

❖ 常見的關聯式資料庫管理系統：

- Oracle Database
- SQL Server
- DB2 ←最老的之一
- SQLite
- MySQL
- PostgreSQL



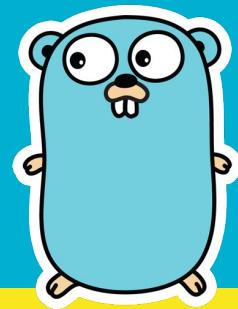
SQL

結構化查詢語言 SQL，全名為 Structured Query Language，是一個能針對資料庫中的數據進行 資料操作 的語言。早於 1970 年代問世，50 年後仍然是資料科學與軟體開發從業者最重要的技能之一。

具體而言，資料操作 可以再細分為創造(Create)、查詢(Read)、更新(Update)、刪除(Delete) 四個動詞，舉例來說，在使用任何的網頁或手機應用程式時，我們的滑鼠點擊與手勢觸控都會被轉換成 CRUD：

- 創造 Create：發佈新的動態
- 查詢 Read：瀏覽追蹤對象的動態
- 更新 Update：編輯先前發佈動態的內容
- 刪除 Delete：撤掉先前所發佈的動態

Join



noSQL

1. migrate easy
2. ↓ delay
3. non-strated data

NoSQL DBMS, Not only SQL Database Management System

4. 大流量(?)

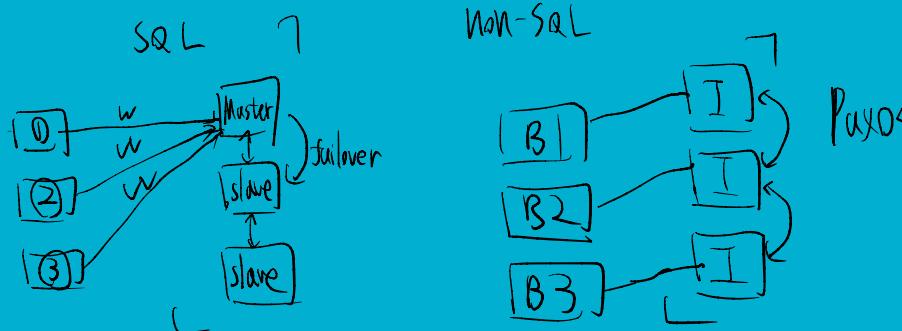
現代應用程式面臨的若干挑戰可透過NoSQL 資料庫來解決。例如，應用程式處理不同來源(如社交媒體、智慧感應器和第三方資料庫)的大量資料。所有這些不同的資料都不適合關聯式模型。強制執行表格式結構可能會導致冗餘、資料重複以及大規模效能問題。

NoSQL 資料庫是為非關聯式資料模型而建立，並且具有構建新型應用程式的彈性結構描述。這些資料庫在開發的容易性、功能性和大規模效能方面廣受肯定。NoSQL 資料庫的優勢如下。

NoSQL是Not Only SQL的縮寫，和剛才前面介紹的不同是，NoSQL 資料庫是一種非關聯式資料庫，將資料儲存為類似JSON 的文件，並對資料進行查詢，這是一個 document 資料庫模型，在這個模型中，資料並不存儲在Table 中，document 是 key-value 的有序集合。資料庫中的每個document 不需要具有相同的數據結構，你可以將資料儲存在JSON、XML文件甚至CSV文件。



noSQL



- **靈活性**
 - NoSQL 資料庫整體而言提供促進更快速及更能反覆開發的彈性結構描述。具彈性的資料模型讓 NoSQL 資料庫成為半結構和非結構式資料的理想資料庫。
- **可擴展性**
 - NoSQL 資料庫通常的設計都能透過硬體的分散式叢集來橫向擴展，而不必藉由增加昂貴和重量級的伺服器來進行縱向擴展。有些雲端供應商背後將這些操作處理成全受管服務。
- **高效能**
 - NoSQL 資料庫針對特定資料模型和存取模式進行優化。相較於嘗試使用關聯式資料庫來執行類似的功能，這可實現更高的效能。
- **高功能性**
 - NoSQL 資料庫提供專為各別資料模型而建造的高功能 API 和資料。
 - *high level API* ↔ 原生 ORM
 - backend → SQL → DB

backend → NoSQL-LIB → 中間 → [DB]
backend → SQL → [DB]



SQL vs noSQL : SQL

- **效能:**
 - 效能取決於磁碟子系統
 - 若要達到頂級效能,通常必須針對查詢、索引及表格結構進行優化 → *table + index*
- **擴展:**
 - 通常透過增加硬體運算能力縱向擴展
 - 以新增唯讀工作負載複本的方式橫向擴展
read-write 分流
- **API:**
 - 存放和擷取資料的請求是透過符合結構式查詢語言 (SQL) 的查詢進行通訊
 - 這些查詢是由關聯式資料庫剖析和執行
- **最佳工作負載:**
 - 交易性以及高度一致性的線上交易處理 (OLTP) 應用程式 ← 真正拿來用, ACID
 - 線上分析處理 (OLAP)

Analytic

ref: <https://aws.amazon.com/tw/nosql/>

- **資料模型:**
 - 關聯式模型將資料標準化,成為由列和欄組成的表格 *用 key 建立 relation*
 - 結構描述嚴格定義表格、列、欄、索引、表格之間的關係,以及其他資料庫元素
 - 強化資料庫表格間的參考完整性
- **ACID 屬性:***原子*
 - 提供單元性、一致性、隔離性和耐用性 (ACID) 的屬性
 - 單元性要求交易完整執行或完全不執行
 - 一致性要求進行交易時資料,必須符合資料庫結構描述
 - 隔離性要求並行的交易必須分開執行
 - 耐用性要求從意外的系統故障或停電狀況還原成上個已知狀態的能力

SQL vs noSQL : noSQL

- **最佳工作負載:**
 - 包含低延遲應用程式的多樣資料存取模式
 - 進行半結構資料的分析
- **資料模型:**
 - 提供鍵值、文件、圖形和資料欄等多種資料模型
 - 具有最佳化的效能與規模
- **ACID 屬性:**
 - 透過鬆綁部分關聯式資料庫的 ACID 屬性來取捨,以達到能夠水平擴展的更具彈性的資料模型
 - 成為水平擴展超過單執行個體上限的高輸送量、低延遲使用案例的最佳選擇
- **效能:**
 - 效能通常會受到基礎硬體叢集大小、網路延遲,以及呼叫應用程式的影響
- **擴展:**
 - 通常可分割
 - 存取模式可透過使用分散式架構來橫向擴展,以近乎無限規模的方式提供一致效能來增加資料輸送量
- **API:**
 - 以物件為基礎的 API 讓應用程式開發人員可輕鬆存放和擷取資料結構
 - 應用程式可透過分區索引鍵 查詢鍵值組、欄集,或包含序列化應用程式物件與屬性的半結構化文件

ref: <https://aws.amazon.com/tw/nosql/>

SQL vs noSQL : 挑選

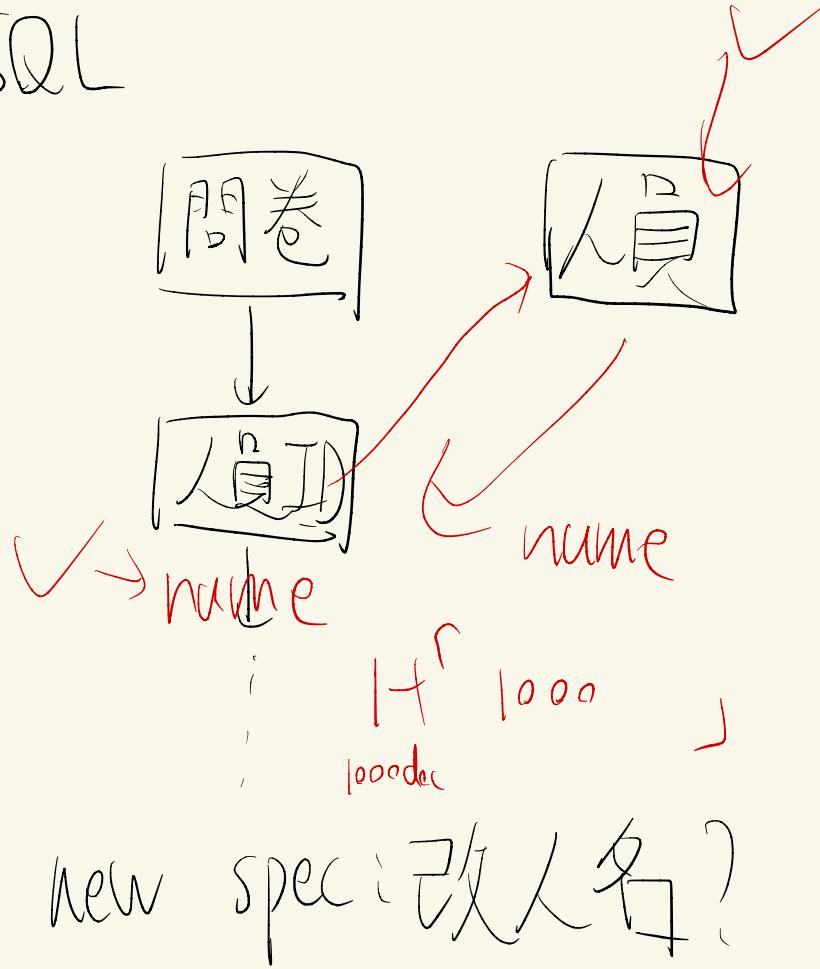
我該挑 SQL 還是 noSQL 來用？

- 數據是否需要關聯查詢 (Join), 要就是 SQL
- 數據的 ACID 要求, 是否有大量 race condition, 有就是 SQL
- 是否結構化, 有就是 SQL

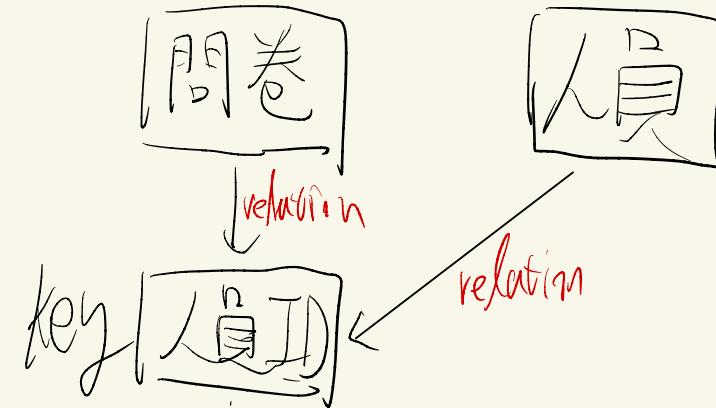
如過上述皆否則選 noSQL。

我的個人經驗: 無腦選 SQL 即為正解 !

NoSQL



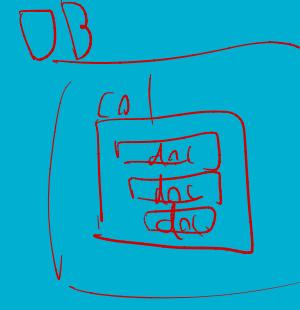
SQL



ACID →

mongo database 教學

- MongoDB 是一種常見的 NoSQL 文件型資料庫。
- 它使用類似 JSON 的 BSON 格式來儲存資料,每個資料庫包含多個集合,每個集合包含多個文件。
- **核心概念**
 - Database: 最大單位, 可以放下多個 Collection
 - Collection: 表格(集合), 可以放下多個 Document
 - Document: 類似 Json 格式, 可以放入多種屬性
- **常見操作**
 - MongoClient.Database("test") : 客戶端連線 Database
 - db.Collection("users") : 客戶端連線 Collection
 - collection.InsertOne : Collection 插入新 document
 - collection.DeleteOne : Collection 刪除 document, 要指定過濾條件
 - collection.Find : Collection 查找 document 列表, 可以設定過濾條件



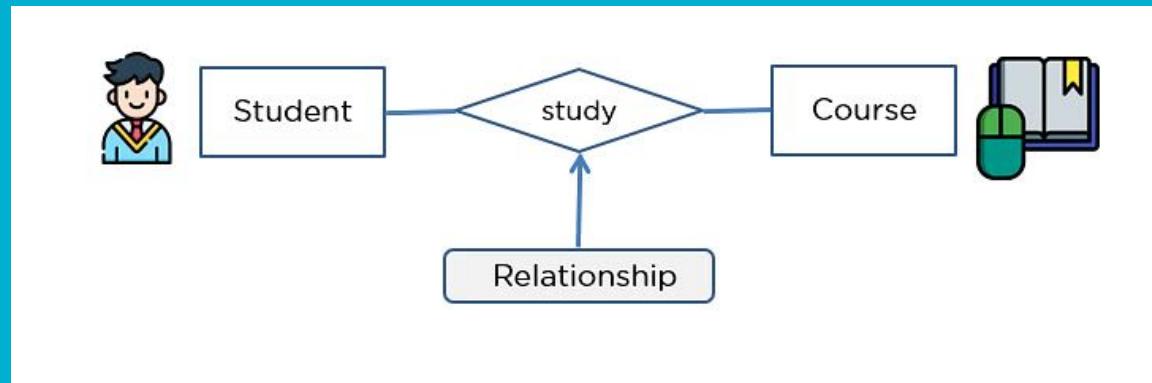
跳至 Lab 完成 mongo 練習 !



Entity Diagram 教學

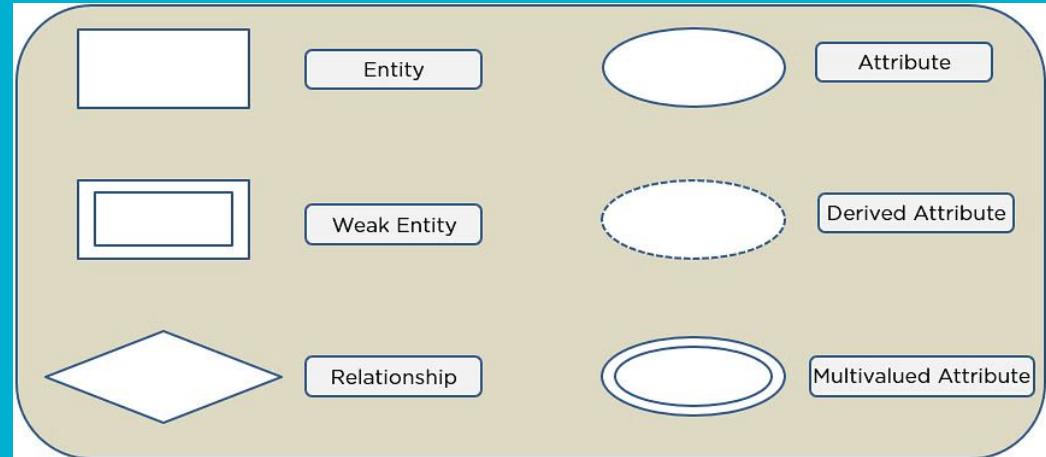
Entity Relationship Diagram

- 實體關聯圖(Entity Relationship Diagram, 簡稱 ERD)是關聯式資料庫的設計圖，在實體關聯圖裡，一個實體(entity)會對應一個資料表(table)。
- 而實體關聯圖當中有不同的格式，分別對應到「實體(Entity)」、「屬性(Attribute)」、「主鍵(Primary Key)」、「實體之間的關係(Relation)」等。



ERD圖元素簡介

- ❖ 矩形：
 - 表示實體(Entity)類型
- ❖ 橢圓：
 - 代表屬性(Attribute)
- ❖ 菱形：
 - 代表關係類型(Relationship)
- ❖ 線：
 - 將屬性連結到實體類型以及實體類型與其他關係類型

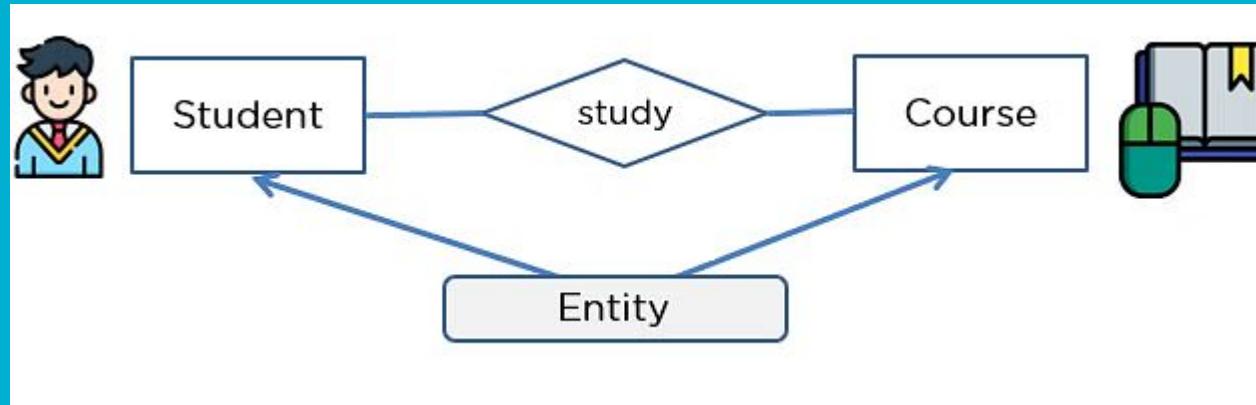


ref:<https://www.simplilearn.com/tutorials/sql-tutorial/er-diagram-in-dbms>



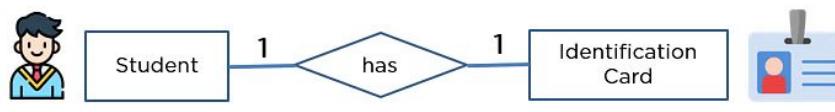
ERD 範例

- 學生”學習”課程
- 學生, 課程 都是實體
- “學習” 是關聯

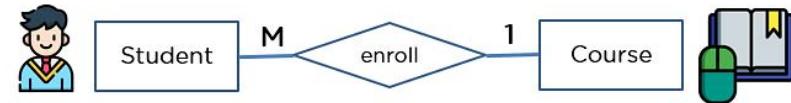


ERD 範例

1對1關係：



多對1關係：



1對多關係：



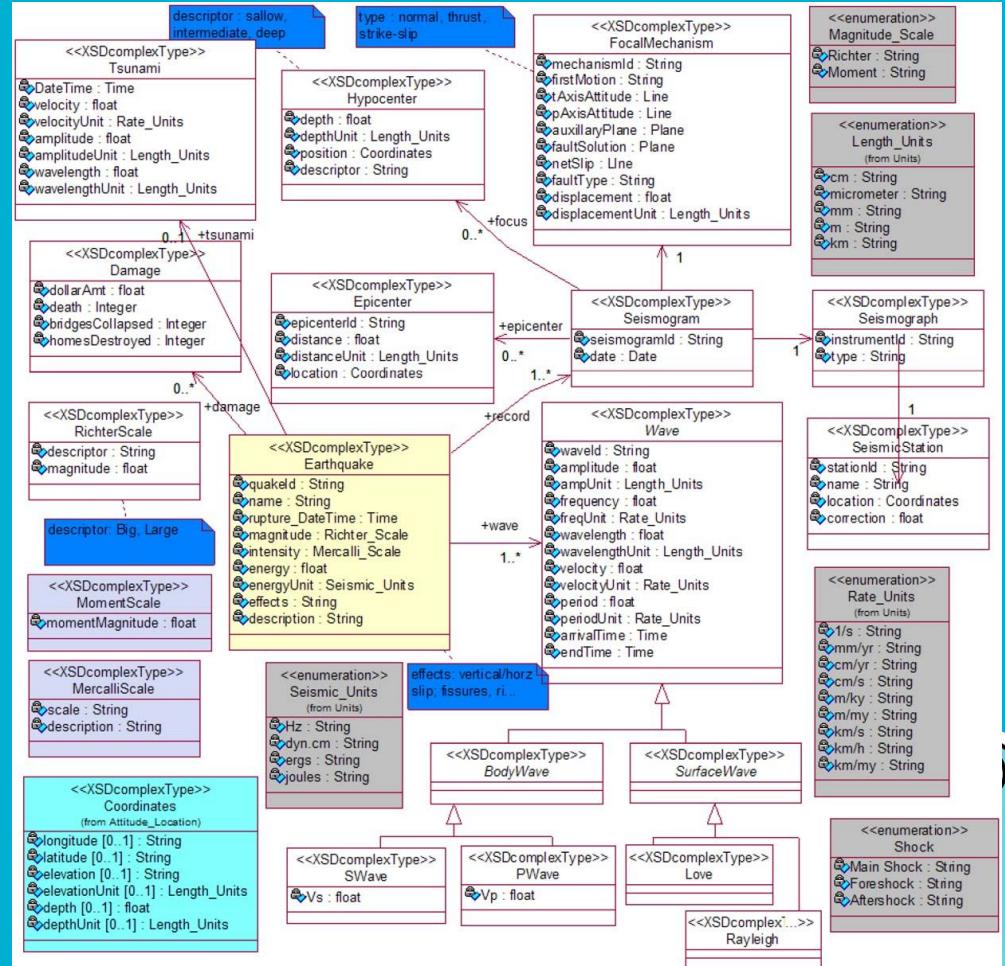
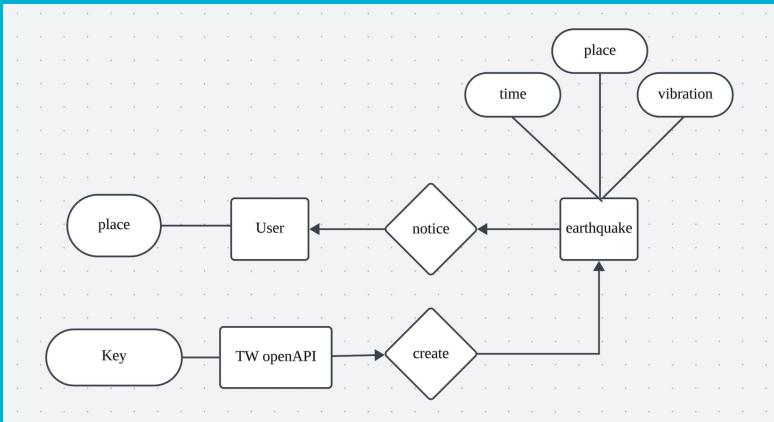
多對多關係：



繪製ERD圖片

繪製軟體: [LucidChart](#)

- 地震軟體 SPEC
 - 地震查詢



地震觀測網頁

交通部中央氣象署
Central Weather Administration

回首頁 EN 網站導覽 意見箱 常見問答 關於氣象署 小 中 大 ⌂ ⌃ ⌁ ⌂ ⌂

警特報 天氣 生活 地震 海象 氣候 資料 知識與天文 常用服務

> 地震 > 最近地震

最近地震

26 25 24 23 22

小區域

編號 最大震度 詳細資訊

05/12 15:14 NEW 看更多+
地點: 花蓮縣政府南南西方73.4公里
(位於花蓮縣卓溪鄉)
深度: 11.9km
地震規模: 3.5

05/12 09:28 NEW 看更多+
地點: 花蓮縣政府東北方41.3公里
(位於臺灣東部海域)
深度: 11.3km
地震規模: 3.1

05/12 07:48 點我看更多+
地點: 花蓮縣政府北北東方14.2公里
(位於花蓮縣近海)
深度: 18.3km
地震規模: 3.6

05/12 05:47 點我看更多+
地點: 花蓮縣政府西北西方6.3公里
(位於花蓮縣秀林鄉)

小區域

2級

1級

2級

2級

請開啟 Mac App Store 以購買和下載 App*

臺灣地震速報 [4+]

即時的地震倒數預警 APP
Chih-Ping Lin
專為 iPhone 設計
在「天氣」類中排名第 1
★★★★★ 4.5 • 3,947 則評分
免費
在此平台查看: Mac App Store ↗

iPhone 截圖

10:54 台北市 地震速報監測 2023/06/23 02:45:00 發表
目前無發布地震資訊

抵達 0 秒 預估級數 2 級
預計抵達時間 2023/06/23 02:45:00

測報歷史紀錄

10:55 6/23/2023 上午 2:45 發生時間
預測倒數 0

10:55 6/17/2023 上午 9:47 發生時間
預測倒數 24

10:55 6/15/2023 上午 7:59 發生時間
預測倒數 18

10:55 6/11/2023 下午 8:26 發生時間
預測倒數 14

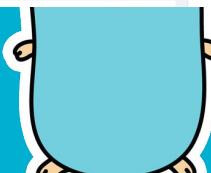
設定 Settings

警報設定
警報位置 台北市

通知設定
預警震度 震度 3(含)以上
4級或以上強震一律推播警報。

歷史設定
歷史記錄顯示 1 年
查看完整速報歷史

關於本程式
版本 1.0



政府資料開放式平台

1. 金鑰申請 : <https://opendata.cwa.gov.tw/>
2. 調用文件:

<https://opendata.cwa.gov.tw/dist/opendata-swagger.html?urls.primaryName=openAPI>

GET /v1/rest/datastore/E-A0015-001 顯著有感地震報告資料-顯著有感地震報告

顯著有感地震報告

Parameters

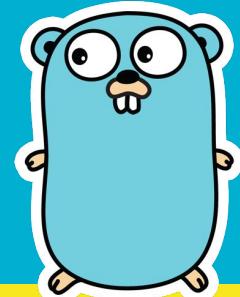
Name	Description
Authorization <small>* required</small>	氣象開放資料平台會員授權碼 string (query) CWA-3453242F-C8C6-4A7A-8C64-24C
limit	number(\$int) (query) 限制最多回傳的資料，預設為回傳全部筆數 limit
offset	number(\$int) (query) 指定從第幾筆後開始回傳，預設為第 0 筆開始回傳 offset

Code Details

200 Response body

```
"MagnitudeType": "芮氏規模",
"MagitudeValue": 4
},
"Intensity": {
"ShakingArea": [
{
"AreaDesc": "南投縣地區",
"CountyName": "南投縣",
"InfoStatus": "observe",
"AreaIntensity": "1級",
"EqStation": [
{
"pga": {
"unit": "gal",
"EWComponent": 2.06,
"NSComponent": 1.22,
"VCComponent": 0.48,
"IntScaleValue": 1.26
},
"pgv": {
"unit": "kine",
"EWComponent": 0.04,
"NSComponent": 0.02,
"VCComponent": 0.01
}
}
]
}
]
```

Response headers



SQL基本語法介紹 (1)

- Init table: <產生一張表格, 可以存放數據, 需要設定欄位名 / 型別 / 限制>

...

```
CREATE TABLE IF NOT EXISTS users (
    id VARCHAR(255) PRIMARY KEY, /* 追加限制 PRIMARY KEY 表示該值唯一, 且搜索加速 */
    name VARCHAR(255) NOT NULL, /* 追加限制 NOT NULL 表示不可以為空 */
    email VARCHAR(255) UNIQUE NOT NULL /* 追加限制 unique 表示該值唯一 */
```

); *Col
name* *type*

constraint

...



SQL基本語法介紹 (2)

- create item: 插數據進表格, 要有表格名稱和各欄位的數據

```

```
INSERT INTO users (name, email, id) VALUES ($1, $2, $3);
```

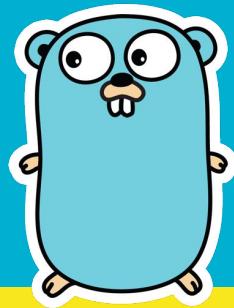
```

- read items: 從表格讀取各欄位數據 要有各欄位的名稱

```

```
SELECT id, name, email FROM users
```

```



SQL基本語法介紹 (3)

- update item: 更新特定表格特定欄位 要用 where 指定目標的條件

...

```
UPDATE users SET email = $1 WHERE id = $2;
```

...

- delete items: 從表格刪除特定數據 要用 where 指定刪除條件

...

```
DELETE FROM users WHERE id = $1;
```

...

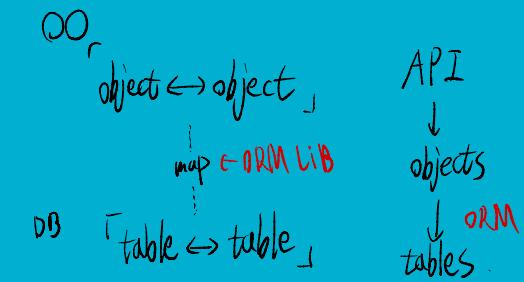
跳至 Lab 完成 postgres 練習！



ORM(Object-Relational Mapping)

ORM (Object-Relational Mapping), 若是照著字義上來翻譯, 那就是「物件關係的映射」, 實際上它其實只是一種概念。在物件兩者之間, 存在著一種對應的映射關係, 這種映射關係的概念, 就是所謂的 ORM。而 ORM 概念的誕生, 其實也是因為物件導向程式開發的崛起。

當物件與表資料之間存在映射關係的時候, 就可以讓我們在開發程式時, 無須操作許多繁瑣的 SQL 語句, 同時在資料的安全面上, 也可以避免 SQL 語句的一些惡意注入攻擊。因此在 ORM 的概念上, 是可以減少那些對資料庫進行基本操作的程式開發。



ORM特性

優點: ORM 可以將資料提取映射成物件的形式

缺點:但如果對於較複雜的操作, 像是多個表之間的 join 或查詢, 相較直接下SQL操作較為不便、效能可能較差



ORM - gorm with golang

- ❖ gorm為常見的Golang ORM函式庫，本身為開源專案，有豐富的官方文件及教學
- ❖ [官網](#)

參考

[:https://pjchender.dev/golang/note-gorm-example/](https://pjchender.dev/golang/note-gorm-example/)

The screenshot shows the GitHub page for the GORM repository. At the top, there's a navigation bar with links to Docs, Gen, Community, API, Contribute, and a search bar. On the right, there's an English language switcher. Below the header, the title "The fantastic ORM library for Golang" is displayed. A prominent button below the title contains the command "\$ go get -u gorm.io/gorm" followed by a blue arrow icon. Underneath the title, there are several GitHub statistics: 36K stars, 3.8K forks, 374 contributors, 5.2K followers, and a link to the V2 RELEASE NOTE with the version V1.25.10. A large white box on the right lists various features of GORM, each preceded by a blue icon:

- Full-Featured ORM
- Associations (has one, has many, belongs to, many to many, polymorphism, single-table inheritance)
- Hooks (before/after create/save/update/delete/find)
- Eager loading with Preload, Joins
- Transactions, Nested Transactions, Save Point, RollbackTo to Saved Point
- Context, Prepared Statement Mode, DryRun Mode
- Batch Insert, FindInBatches, Find/Create with Map, CRUD with SQL Expr and Context Valuer
- SQL Builder, Upsert, Locking, Optimizer/Index/Comment Hints, Named Argument, SubQuery
- Composite Primary Key, Indexes, Constraints
- Auto Migrations
- Logger
- Extendable, flexible plugin API: Database Resolver (multiple databases, read/write splitting) / Prometheus...
- Every feature comes with tests
- Developer Friendly

跳至 Lab 完成 gorm 練習！

DB Migration

資料庫遷移(db migrations)相較於直接進資料庫系統使用 SQL 修改結構，使用 Migrations 可以讓我們有記錄地進行資料庫修改，每次變更就是一筆 Migration 記錄。

在沒有 Migration 之前，若手動修改資料庫，就必須通知其他開發者也進行一樣的修改步驟。另外，在正式佈署的伺服器上，你也必須追蹤並執行同樣的變更才行。而這些步驟如果沒有記錄下來，就很容易出錯。



DB Migration tool

Migrations 會自動追蹤哪些變更已經執行過了、那些還沒有，你只要新增 Migration 檔案，然後執行 `rake db:migrate` 就搞定了。

它會自己搞清楚該跑哪些 migrations，如此所有的開發者和正式佈署的伺服器上，就可以輕易的同步最新的資料庫結構。

另外一個優點是：Migration 是獨立於資料庫系統的，所以你不需要煩惱各種資料庫系統的語法差異，像是不同型態之類的。當然，如果要針對某個特定資料庫系統撰寫專屬功能的話，還是可以透過直接寫 SQL 的方式。

original

$SQL \rightarrow DB$

new

$migration\ Lib \rightarrow SQL \rightarrow DB$



簡介快取 (cache)

cache 80% (1/100)
↑①

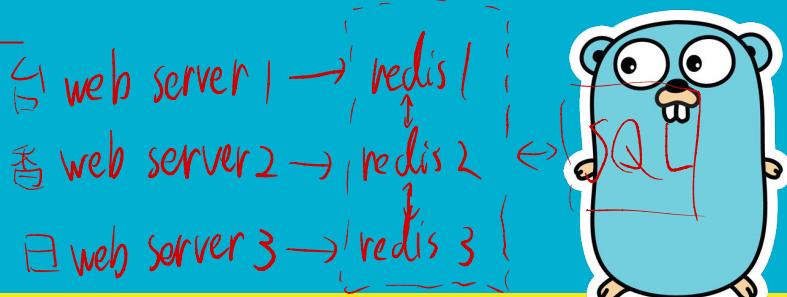
8/2 法則

API → Web Server → ② → DB

$$100 = 80 \times \frac{1}{100} + 20 + \frac{1}{1000} = 0.8 + 20 + 0.001 = 20.801$$

real

- 本機快取是比較基本的資料快取方式，將資料存在 Web Application 的記憶體中。如果是單一站台架構，沒有要同步快取資料，用本機快取應該都能滿足需求。
- ~~若有短時間內大量存取的需求，卻又不想如此頻繁的對硬碟讀寫，這種場合可以使用Redis資料庫暫時存放資料，等一段時間後再一齊寫入硬碟裡。~~
- 當網站有橫向擴充，架設多個站台需求時，**分散式快取**就是一個很好的同步快取資料解決方案。基本上就是 NoSQL 的概念，把分散式快取的資料位置，指向外部的儲存空間，如：SQL Server、Redis 等等。

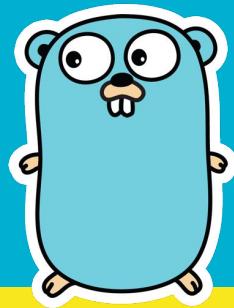


Redis介紹



Redis存放資料的速度極快，是物理層面上的快，因為Redis對記憶體(Memory)進行操作

Redis將資料存放於記憶體中的特點為：讀寫速度比存放於磁碟快、容量較小，且資料在斷電就揮發(Volatile)不見了(若沒及時寫進磁碟中的話)



用Go實作Redis

使用:go-redis/redis套件操作Redis

語法:\$ go get github.com/go-redis/redis

```
package user

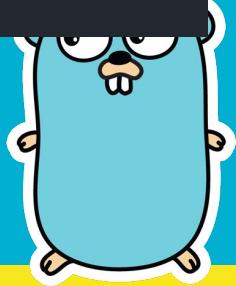
import (
    "db/model"
    "github.com/redis/go-redis/v9"
)
import rdb "db/repository/redis"

type RedisUserService struct {
    alias UserServiceType
    client *redis.Client
}

func (rds *RedisUserService) InitTable() error {
    // No need to create table in Redis
    return nil
}

func (rds *RedisUserService) GetType() UserServiceType {
    return rds.alias
}
```

跳至 Lab 完成 redis 練習！



Lab

各種資料庫試寫



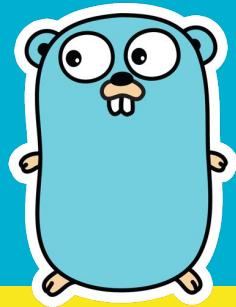
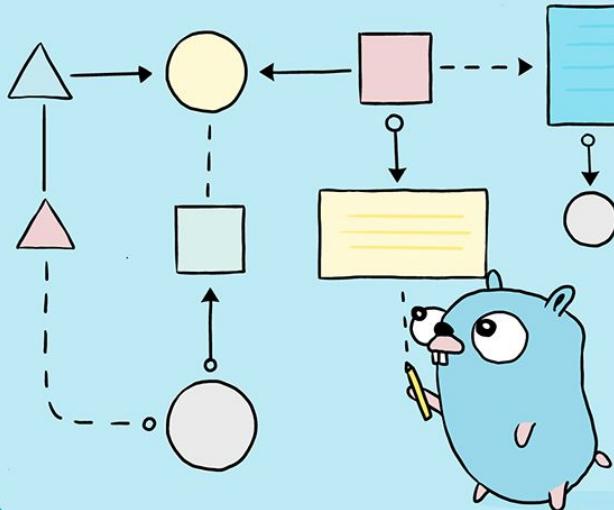
資料庫練習

1. mongo
2. postgres
3. gorm
4. redis

Link: <https://github.com/leon123858/go-tutorial/tree/main/db>



Q&A 時間



作業: 資料庫練習

整合 mongo + postgres + gorm + redis 的資料庫練習

Link: <https://github.com/leon123858/go-tutorial/tree/main/db>

