

# 資料庫基礎

---

1. 資料庫基本概念
2. SQL 基礎語法
3. 基本的資料庫管理

# 什麼是資料庫？

1. 資料庫是一種資料集合，用於儲存、查詢和管理大量資料。
2. 它使用結構化查詢語言（SQL）等工具來操作資料，並確保資料的一致性、完整性和安全性。
3. 資料庫系統（如MySQL、PostgreSQL、Oracle）支援多用戶同時存取，適用於各種應用，包括企業管理、電子商務和科學研究分析。

# 資料庫能幫助我們做什麼？

1. 有條理地儲存大量資料：不再是雜亂無章的筆記，而是結構化的資訊。
2. 快速查詢和檢索：你可以輕易地找到你需要的那份資料。
3. 確保資料的一致性與準確性：避免重複或錯誤的資料。
4. 多人共享與協作：多個人可以同時安全地存取和使用資料。
5. 試著舉網站的例子，並說說看你覺得那個資料庫可能儲存了哪些資訊？

# 資料庫種類

## 1. 關連式資料庫 (Relational Databases)

- 一個個表格組成的，每個表格都有欄位和記錄，而且表格之間可以互相連結。
- 你可以想像它像一個個 Excel 工作表，但它們之間有著「關係」。

## 2. 非關連式資料庫 (NoSQL Databases)

- 「NoSQL」其實是 "Not Only SQL" 的縮寫，代表它們不只使用傳統的表格結構。
- 沒有固定的表格結構，而是以鍵值對 (Key-Value)、文件 (Document)、圖形 (Graph) 等形式儲存資料。

# 資料庫種類

## 1. 關連式資料庫 (MySQL、PostgreSQL、Oracle、SQL Server)

- 優點：資料結構清晰，容易理解和管理，而且非常適合處理結構化、有明確關係的資料。
- 缺點：面對非結構化或資料量極大的情況時，可能會比較吃力。

## 2. 非關連式資料庫 (MongoDB、Redis)

- 優點：彈性高，適合處理大量非結構化或半結構化的資料，擴展性強。
- 缺點：資料一致性管理較複雜，有時學習曲線較陡峭。

# 以下系統適合哪一種資料庫開發？

1. 校務資料庫
2. 圖書館系統
3. Fackbook 與 IG

# SQL 是什麼？

---

1. SQL (Structured Query Language)
2. 是一種用於管理關聯式資料庫的程式語言。

# 資料庫系統語言的類型

## 1. 資料定義 (DDL)

- 定義資料庫的結構，例如建立表格、定義每個欄位的資料類型（這是數字還是文字？）、設定哪些欄位是主鍵等等。

## 2. 資料操作 (DML)

- 對資料進行查詢 (Query)、新增 (Insert)、修改 (Update) 和刪除 (Delete) 的操作。

## 3. 資料控制 (DCL)

- 管理使用者權限，確保只有被授權的人才能存取和修改資料。同時也處理資料的安全性、完整性與可用性等問題。



# 安全的資料庫特性

1. 資料機密性：確保僅授權用戶能夠訪問資料庫，防止未授權用戶的存取。這通常通過身份驗證和訪問控制來實現，如用戶名/密碼、雙因素驗證等。
2. 資料完整性：保證資料在傳輸和存儲過程中不被未經授權的修改或損壞。使用校驗和（checksum）、雜湊函數和數位簽章等技術來確保資料的一致性和準確性。
3. 資料可用性：確保資料在需要時可供授權用戶存取。這包括保護資料庫免受各種攻擊（如拒絕服務攻擊）以及災難恢復計劃和資料備份。

# 常用基本關鍵字

- DDL、DCL、DML
- 使用者、角色、語系
- 資料庫、資料表、資料欄位、資料型態
- 主鍵、外鍵、複合鍵、索引
- 檢視、預存程序、觸發、函數
- 行、列、紀錄

# 關聯式資料庫的核心概念

1. 表格：EXCEL 工作表，如 `users`、`user_profiles`、`orders` 等
2. 欄位：EXCEL 欄位，如 `id`、`username`、`email` 等
3. 紀錄：EXCEL 行，如 `U1234`、`Andy`、`andy@em1.com` 等
4. 主鍵：代表識別一行資料唯一的值且非空，如 `學號`、`身分證` 號等。

# 設計一個資料表儲存「會員資料」

1. 表格名稱？
2. 包含哪些欄位？
3. 主鍵？

簡單來說，資料模型設計就是把真實世界中的事物和它們之間的關係，用一種結構化的方式表示出來，讓電腦能夠理解和儲存。

# 資料模型設計

1. 定義資料的結構：決定多少表格？多少欄位？欄位資料類型？
2. 定義資料的關係：不同表格之間的關係，如「課程」、「學生」之間的關聯？一個學生多門課，一門課多個學生。
3. 確保資料的完整性：避免重複資料、錯誤、遺失。

# 以選課系統為例

1. 現有「學生」（學號、姓名）、「課程」（課程編號、課程名稱）表格
2. 若要紀錄學生選了哪些課的資訊，需要建立什麼表格？
3. 包含哪些欄位？
4. 與「學生」和「課程」的哪些欄位有關係？
5. 結合兩個資料表的關係？有什麼關鍵字可以使用？

# SQL 語言的基礎 (SELECT)

```
SELECT 欄位1, 欄位2, ...  
FROM 表格名稱  
WHERE 條件;
```

1. **SELECT** 表示選取所有列出的「欄位」資料，欄位名稱以 **,** 區隔。
2. **FROM**：從哪個表格選取資料
3. **WHERE**：設定條件來篩選資料

```
WHERE 欄位1='A' AND 欄位2='B' OR 欄位3='C'
```

# 表格 Students 包含 Sid , Name , Age , Major

1. 選取學生的所有欄位的資訊

```
SELECT * FROM Students;
```

2. 選取學生姓名和年齡的資訊

```
SELECT Name, Age FROM Students;
```



# 表格 Courses 包含 Cid , CName , Credit

1. 選取課程的所有欄位的資訊

```
SELECT * FROM Courses ;
```

2. 選取課程名稱和學分數的資訊

```
SELECT CName , Credit FROM Courses ;
```

# 選取主修 (Major) 「資安」 學生姓名與學號

```
SELECT SD, Name  
FROM Students  
WHERE Major = '資安'  
-- 加上年齡為 20 歲的條件
```

# 不確定的資料查詢

1. 關鍵字 `LIKE` 運算子和 `%` 萬用字元（代表任意長度字串）
2. `LIKE '%關鍵字%'`
3. `LIKE '%關鍵字'`
4. `LIKE '關鍵字%'`

# 不確定的資料查詢

1. AI與 資安 的關係、AI世代的 資安 、 資安 與AI的關係
2. LIKE '%資安%'
3. LIKE '%資安'
4. LIKE '資安%'
5. 有修關鍵字包含 '資安' 且年齡為 20 歲的學生

# 資料新增 (INSERT)

## 1. 關鍵字 `INSERT INTO`

```
INSERT INTO 表格名稱 (欄位1, 欄位2, ...)  
VALUES (值1, 值2, ...);
```

## 2. 新增一位學生

```
INSERT INTO Students (Sid, Name, Major, Age)  
VALUES ('S001', '王小明', 'AI與資安的關係', 20);
```

# 資料修改 (UPDATE)

## 1. 關鍵字 UPDATE

UPDATE 表格名稱

SET 欄位1 = 新值1, 欄位2 = 新值2, ...

WHERE 條件;

## 2. 把學生「王小明」的名字改成「王大明」

```
UPDATE Students
```

```
SET Name = '王大明'
```

```
WHERE Name = '王小明';
```

```
-- 所有主修「AI與資安的關係」改為「AI和資安的關係」
```

# 資料刪除 (DELETE)

## 1. 關鍵字 `DELETE FROM`

`DELETE FROM` 表格名稱  
`WHERE` 條件;

## 2. 刪除學號為 `S001` 的學生

```
DELETE FROM Students  
WHERE Sid = 'S001';
```

# 資料排序

1. 關鍵字 `ORDER BY` 、 `ASC` 升冪、 `DESC` 降冪

```
SELECT 欄位1, 欄位2, ...  
FROM 表格名稱  
ORDER BY 欄位1 ASC, 欄位2 DESC
```

2. 請將學生照年齡大至小與學號小至大排序

```
SELECT *  
FROM Students  
ORDER BY Age ASC DESC, Sid;
```



# 資料限制 LIMIT 數量

## 1. 關鍵字

SELECT 欄位1, 欄位2, ...

FROM 表格名稱

WHERE 條件

ORDER BY 排序欄位

LIMIT 數量; -- 通常會與 ORDER BY 一起使用，以確保結果有意義

## 2. 年齡最大的 3 位學生

# 資料限制 LIMIT 偏移量, 數量

## 1. 偏移量由 0 開始

```
SELECT 欄位1, 欄位2, ...  
FROM 表格名稱  
ORDER BY 排序欄位  
LIMIT 偏移量, 數量;
```

## 2. 依據年齡排序取得學生資料的第2頁每頁20筆

```
-- ...  
LIMIT 20, 20; -- 第三頁 ?
```

# 聚合函數

1. COUNT()：計算紀錄數量

1. COUNT(\*)：所有紀錄

2. COUNT(欄位名稱)：非空欄位的紀錄

3. COUNT(DISTINCT 欄位名稱)：欄位值不重複的紀錄

2. SUM(學分)：學分總和

3. AVG(年齡)：平均年齡

4. MAX(學分)、MIN(年齡)：學分最大值、年齡最小值

# 分組

## 1. 關鍵字 `GROUP BY` 、 `HAVING`

```
SELECT 欄位1, 聚合函數(欄位2), ...  
FROM 表格名稱  
WHERE 條件  
GROUP BY 欄位1, ...  
HAVING 聚合函數條件; -- 可選, 用於篩選分組後的結果  
ORDER BY 排序欄位; -- 可選
```

## 2. 計算課程修課人數

# 分組過濾 (HAVING)

1. **WHERE** 用於分組 **前** 篩選
2. **HAVING** 用來分組 **後** 篩選
3. 選取課程修課人數大於10人的課程，並排序

# 跨表查詢 (JOIN)

1. INNER JOIN
2. LEFT JOIN
3. RIGHT JOIN

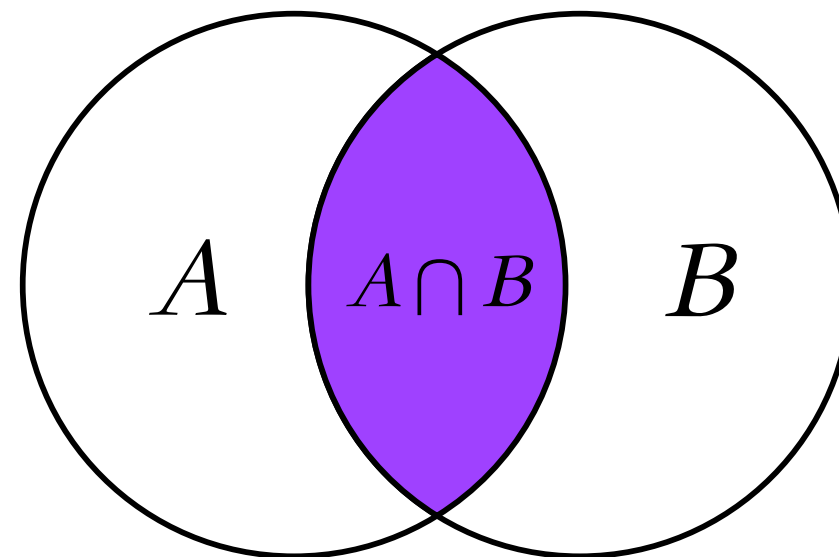
# INNER JOIN

```
SELECT 欄位列表  
FROM 表格A  
INNER JOIN 表格B ON 表格A.共同欄位 = 表格B.共同欄位;
```

# INNER JOIN

1. A : users
2. B : profiles
3.  $A \cap B$  : users + profiles

```
SELECT A.username, B.bio  
FROM users as A  
INNER JOIN profiles as B  
ON A.user_id = B.user_id;
```





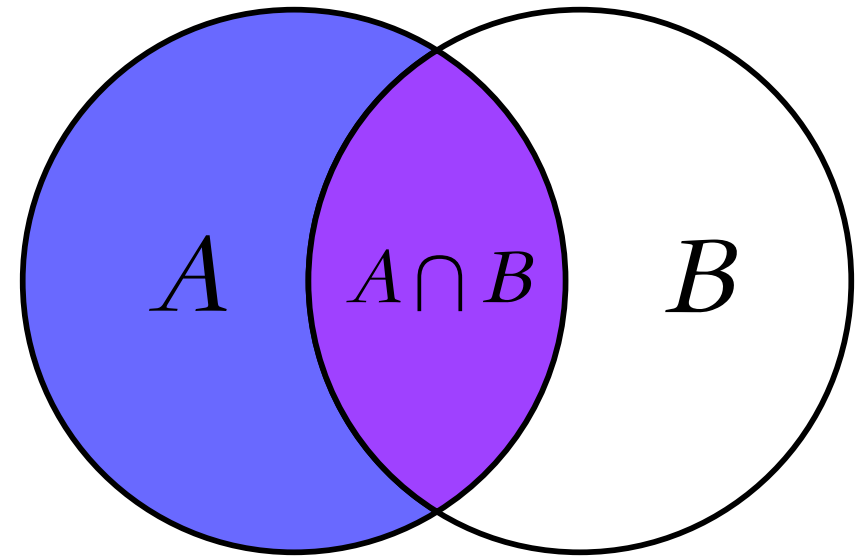
# LEFT JOIN

```
SELECT 欄位列表  
FROM 表格A -- 左側表格  
LEFT JOIN 表格B -- 右側表格  
ON 表格A.共同欄位 = 表格B.共同欄位;
```

# LEFT JOIN

1. A : `users` + NULL `profiles`
2. B : `profiles`
3.  $A \cap B$  : `users` + `profiles`

```
SELECT A.username, B.bio  
FROM users as A  
LEFT JOIN profiles as B  
ON A.user_id = B.user_id;
```



# RIGHT JOIN

```
SELECT 欄位列表  
FROM 表格A -- 左側表格  
RIGHT JOIN 表格B -- 右側表格  
ON 表格A.共同欄位 = 表格B.共同欄位;
```

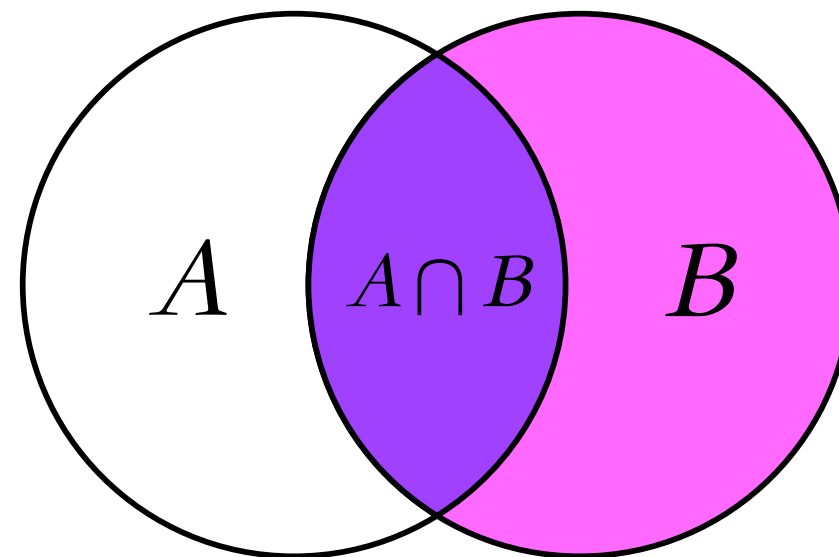
# RIGHT JOIN

1. A : NULL `users` + `profiles`

2. B : `profiles`

3.  $A \cap B$  : `users` + `porfiles`

```
SELECT A.username, B.bio  
FROM users as A  
RIGHT JOIN profiles as B  
ON A.user_id = B.user_id;
```



# UNION 和 UNION ALL

1. 欄位數量必須相同
2. 對應欄位的資料型態必須相容

```
SELECT column1, column2, ...  
FROM table1  
WHERE condition1  
UNION [ALL]  
SELECT column1, column2, ...  
FROM table2  
WHERE condition2;
```

# UNION 和 UNION ALL 差異

1. UNION（去重合併） UNION ALL（保留重複合併）
2. UNION 的執行效率通常會比 UNION ALL 差。

# 條件邏輯判斷 CASE WHEN

```
CASE column_or_expression
  WHEN value1 THEN result1
  WHEN value2 THEN result2
  ...
  [ELSE else_result] -- 可選，如果所有 WHEN 條件都不滿足時的預設值
END
-- 成績大於等於60分 PASS else 'FAIL'
```

## 子查詢

1. 返回單一值：= 、 > 、 < 、 >= 、 <= 、 !=

2. 返回單欄位列表：IN 、 NOT IN 、 ANY 、 ALL

# 鍵 Key

---

1. Primary Key
2. Unique Key
3. Foreign Key
4. Composite Key



# Primary Key (主鍵) :

1. 用於唯一標識表中的每一行資料。
2. 不能包含NULL值。
3. 一個表中只能有一個主鍵。

```
CREATE TABLE users (  
    id INT AUTO_INCREMENT,  
    username VARCHAR(50) NOT NULL,  
    email VARCHAR(50),  
    PRIMARY KEY (id)  
);
```

# Unique Key（唯一鍵）：

1. 保證列中的所有值唯一。
2. 可以包含NULL值，且一個表中可以有多个唯一鍵。

```
CREATE TABLE users (  
    id INT,  
    username VARCHAR(50),  
    email VARCHAR(50) UNIQUE  
);
```

# Foreign Key（外鍵）：

1. 用於保持兩個表之間的參照完整性。
2. 外鍵在一個表中指向另一個表中的主鍵。

```
CREATE TABLE orders (  
    order_id INT PRIMARY KEY,  
    user_id INT,  
    FOREIGN KEY (user_id) REFERENCES users(id)  
);
```

# Composite Key（複合鍵）：

1. 使用多個欄位來唯一標識一行資料。
2. 可以用來建立複合主鍵或複合唯一鍵。

```
CREATE TABLE user_roles (  
    user_id INT,  
    role_id INT,  
    PRIMARY KEY (user_id, role_id)  
);
```

# 索引 (Indexes)

1. 目的：大幅提升資料查詢 `SELECT` 的速度
2. 資料庫裡的 `目錄` 或 `捷徑`
3. 當為某個欄位建立索引時，DBMS 會為這些欄位的值建立一個排序好的副本，並儲存這些值所對應的原始資料行的物理位置。當你執行查詢時，DBMS 可以直接透過這個索引來快速定位資料，而不需要掃描整個資料表。

# 索引優點

1. 減少掃描的資料量
2. 加速排序 (ORDER BY)
3. 加速聚合函數 (GROUP BY)
4. 加入 JOIN 操作

# 索引缺點

1. 佔用空間
2. 降低寫入（INSERT、UPDATE、DELETE）速度
3. 維護成本，索引重建

# 索引排序

```
CREATE INDEX idx_bookname_asc ON Books (BookName ASC);  
CREATE INDEX idx_bookname_desc ON Books (BookName DESC);
```



# 如何透過AI建立索引

1. 告知使用何種資料庫（MySQL）與版本（8.X）
2. 提供 T-SQL
3. 協助建立索引



# 資料完整性與約束 (Constraints)

1. 主鍵 (Primary Key Constraints)：唯一且非空
2. 外鍵 (Foreign Key Constraints)：指向其他表格的主鍵
  - 選課紀錄表中 **Sid** 與 **Cid** 為外鍵，不存在的 **Sid** 與 **Cid** 無法新增。
3. 唯一性 (Unique Constraints)：唯一且可空
4. 非空約束 (NOT NULL Constraint)：欄位值必須有值
5. 檢查 (Check Constraints)：符合條件才可儲存

# 外鍵約束與刪除

1. NO ACTION 或 RESTRICT (預設行為，或最嚴格的)：阻止刪除
2. CASCADE (級聯刪除)：自動刪除相關記錄
3. SET NULL (設為空值)：將外鍵欄位設為 NULL
4. SET DEFAULT (設為預設值)：將外鍵欄位設為預設值

# 非空約束與預設值（個人習慣）

1. 字串或大部分型態：設定非空預設 
2. 整數與浮點：非空預設 ，若有負值不設定非空
3. 時間：視需求設定可空或非空或是預設值現在時間
4. 特殊格式不設定非空

# 資料庫的安全性

1. 身份驗證 (Authentication)：帳密確認
2. 授權 (Authorization)：權限確認，決定可以執行哪些操作與哪些資料
3. 加密 (Encryption)：敏感資料加密
4. 稽核 (Auditing)：記錄所有對資料庫的操作，萬一被惡意攻擊，可使用稽核紀錄追查。

# 資料庫備份與還原

## 1. 備份 Backup：

- 包含資料與結構（含資料庫程式）複製到另外一個位置
- 定期、安全、異地、有效、多份
- 根據不同的資料特性訂定不同的資料庫備份策略，搭配成本考量做適當的策略訂定。

## 2. 還原 Recovery：

- 資料庫故障時，用 **最近** 的備份還原，使其恢復正常運作
- 故障：物理性、邏輯性（資料有誤、誤刪、誤植或是被駭客惡意串改）

