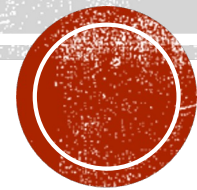


# 關聯式資料庫

林新德

shinder.lin@gmail.com



參考專案 <https://bitbucket.org/lsd0125/mfee29-php/>

# 1. 資料庫

- <https://www.oracle.com/tw/database/what-is-database/>
- 資料庫是結構化的資訊或資料集合，通常以電子方式儲存在電腦系統中。
- 資料庫通常由資料庫管理系統 (DBMS) 控制。
- 資料和 **DBMS** 以及與之關聯的應用程式統稱為資料庫系統，通常簡稱為資料庫。
- 目前資料庫系統大致分為兩類：關聯式資料庫 (RDBMS) 和 NO-SQL。



# 1.1 關聯式資料庫

- <https://www.oracle.com/tw/database/what-is-a-relational-database/>
- 關係式資料庫是一種資料庫，用於儲存並存取相關的資料點。
- 關聯式資料庫的基礎建立於關聯模型之上，以直接且直覺的方式於資料表上顯示資料。
- 在關係式資料庫中，資料表中的每一列資料行都是一條記錄，並有唯一的ID，稱為索引鍵。
- 資料表的資料列中存放著資料的屬性，每條記錄通常有一個屬性值，這樣就很容易建立資料點之間的關係。
- 資料庫之下可以有許多資料表，資料表內可以有許多資料項目 (rows)。
- 資料表之間以「外鍵」為關聯關係。



## 1.2 實體關係模型

- <https://zh.wikipedia.org/zh-tw/ER模型>
- Entity-relationship model，簡稱「ER模型」。
- 實體（Entity）表示一個離散物件。
- 實體可以被視為是名詞，如：學生、電腦、雇員、歌曲、訂單。
- 關聯描述了兩個或更多實體相互之間的關係。
- 如：在一家「公司」可以擁有多台「電腦」、一個部門可以有多個「雇員」、一個「商品」可以擁有多個「標籤」、一個「標籤」可以包含多個「商品」。
- 每個實體應該要有一個唯一標識特性的屬性，此屬性會稱為主鍵。
- 實體關聯圖不展示單一的實體或關聯的單一的實例。它們展示實體集合和關聯集合。



## 1.3 常見的關聯式資料庫

- MySQL（社群版授權，可免費使用）。
- MariaDB（可免費使用，創辦人同 MySQL）。
- PostgreSQL（社群軟體，可免費使用）。
- Microsoft SQL Server（商業軟體，Express版可免費使用）。
- Oracle（商業軟體，有可免費使用的開發版）。
- DB2（IBM 商業軟體）。
- SQLite（開源軟體，常用於手機或資料量較少的系統）。

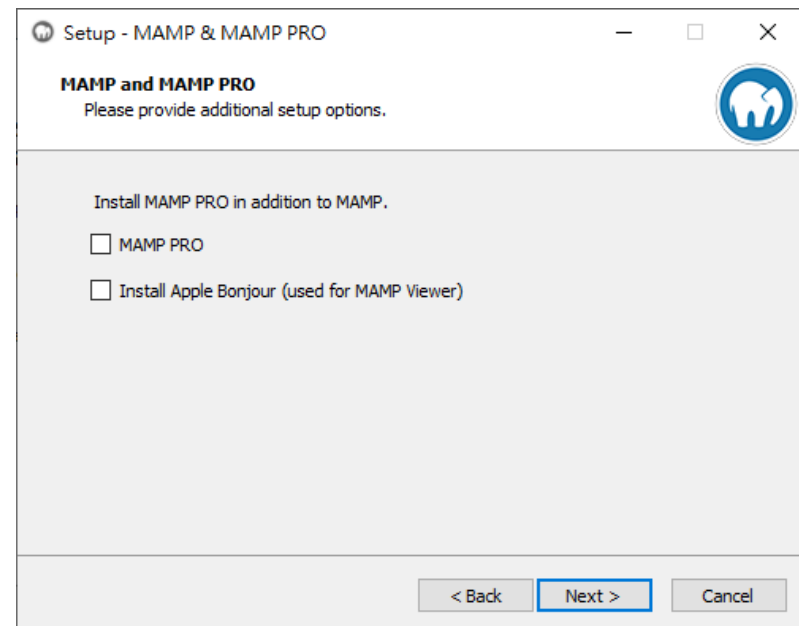
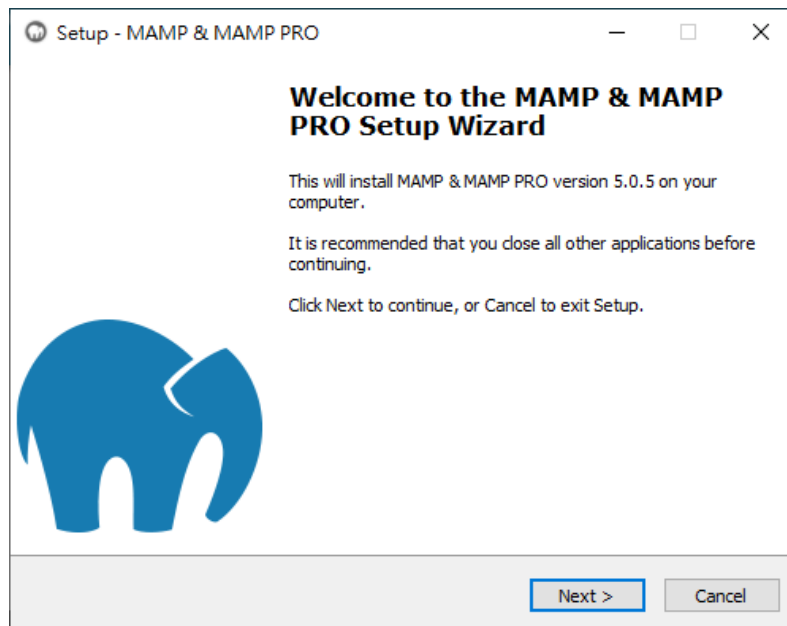


## 2. 安裝資料庫

- 可使用安裝包軟體安裝，例如：XAMPP，MAMP 等。
- 在此我們使用MAMP <https://www.mamp.info/en/downloads/>
- 注意 MacOS 不同晶片的安裝檔不同

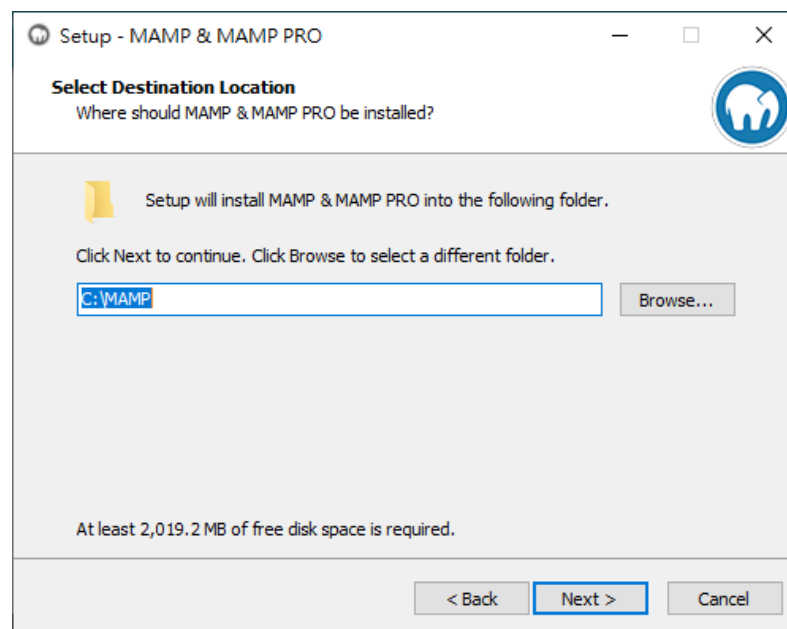
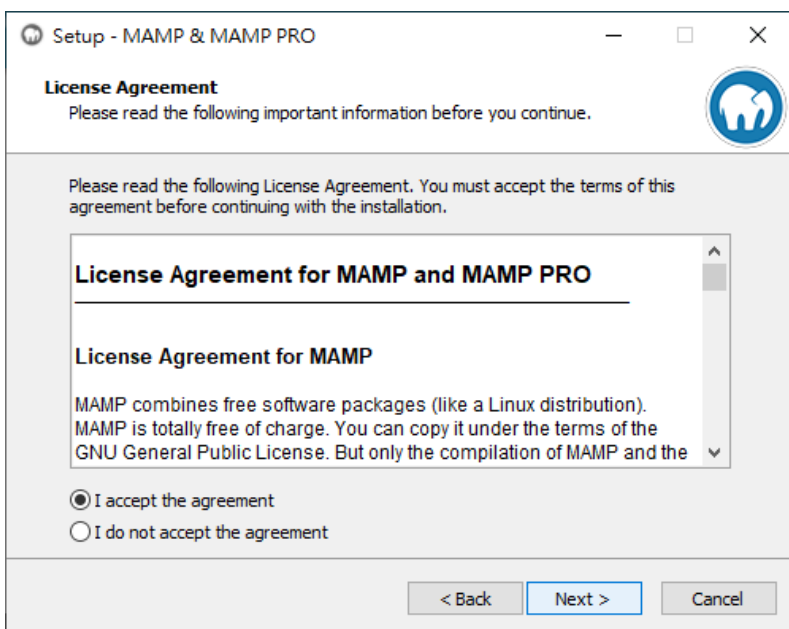


## 2.1 安裝 MAMP



\* 取消勾選 PRO 和 Bonjour

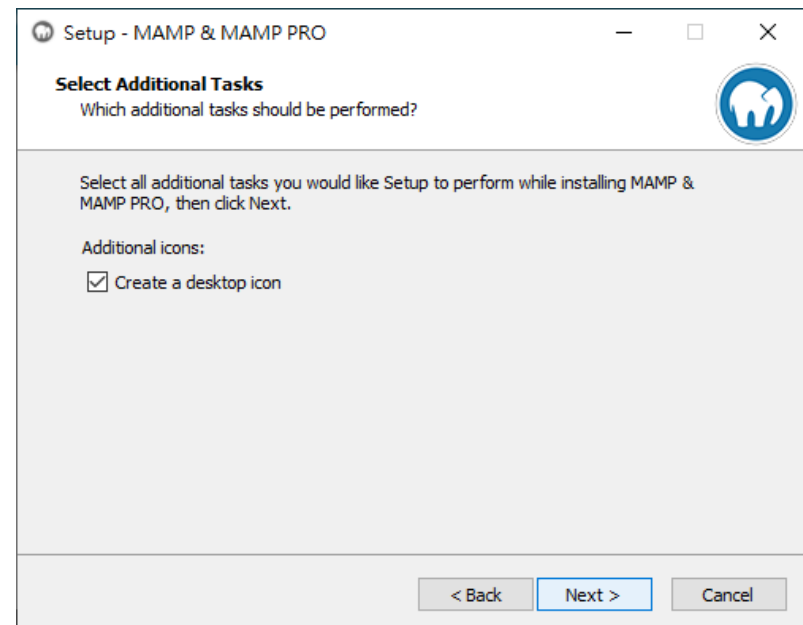
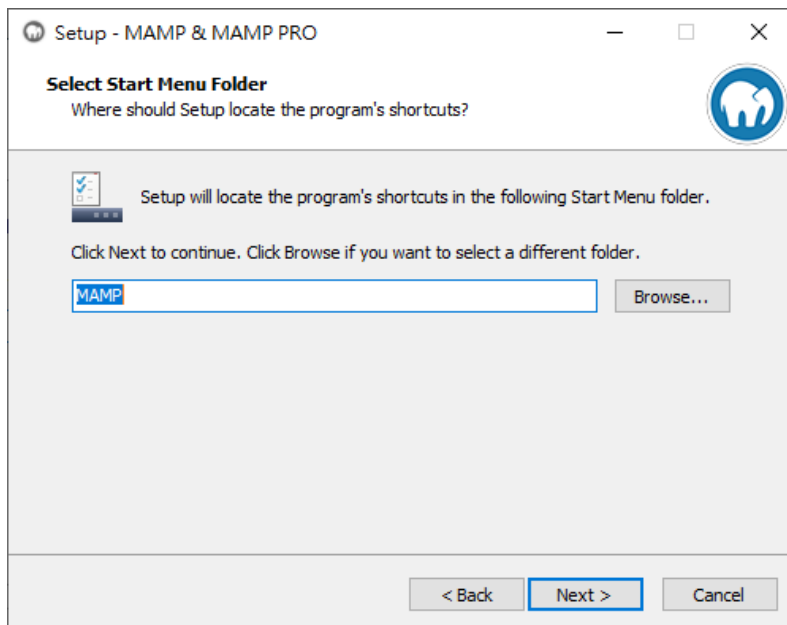


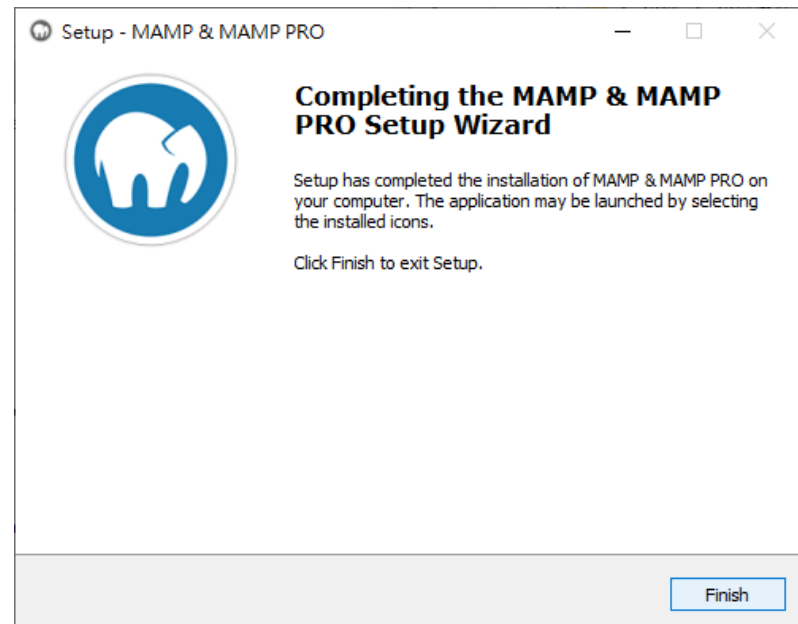
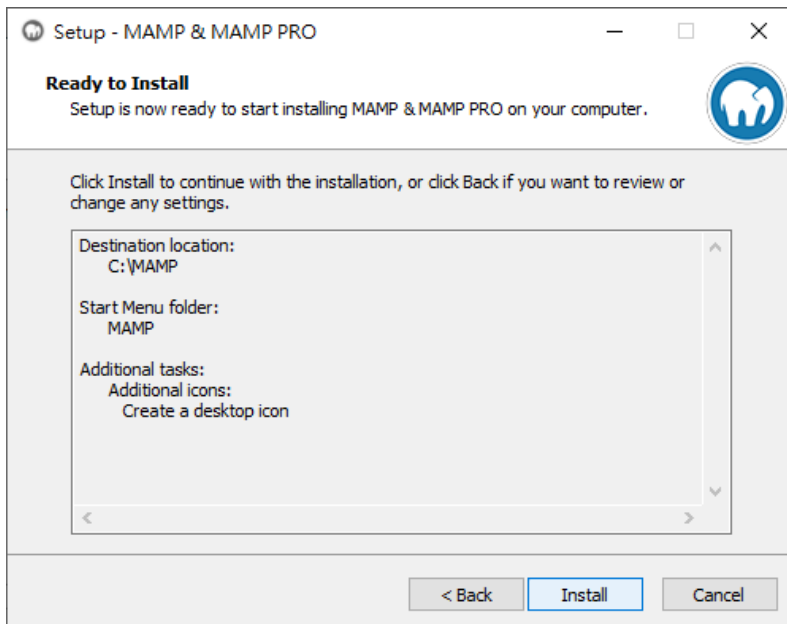


\* 按裝使用預設的資料夾



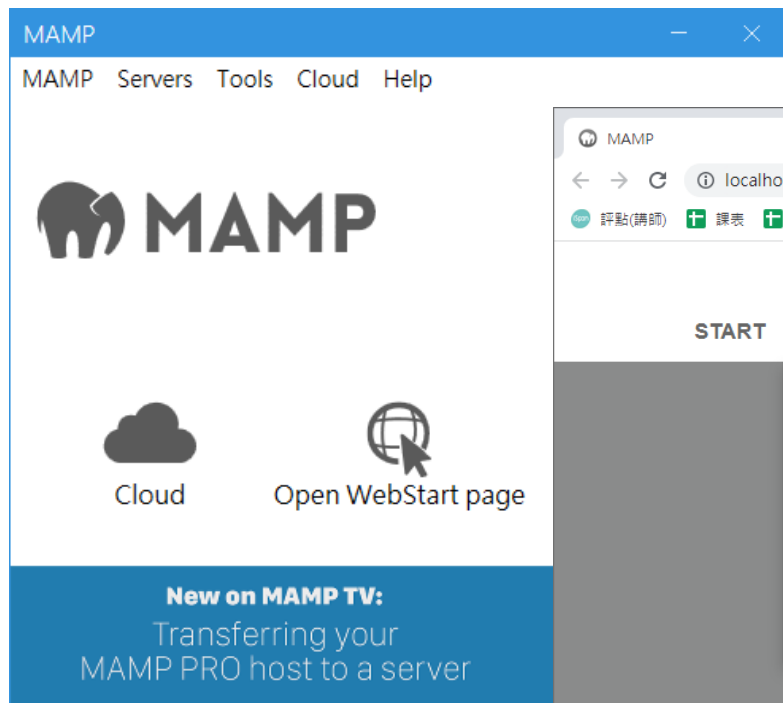




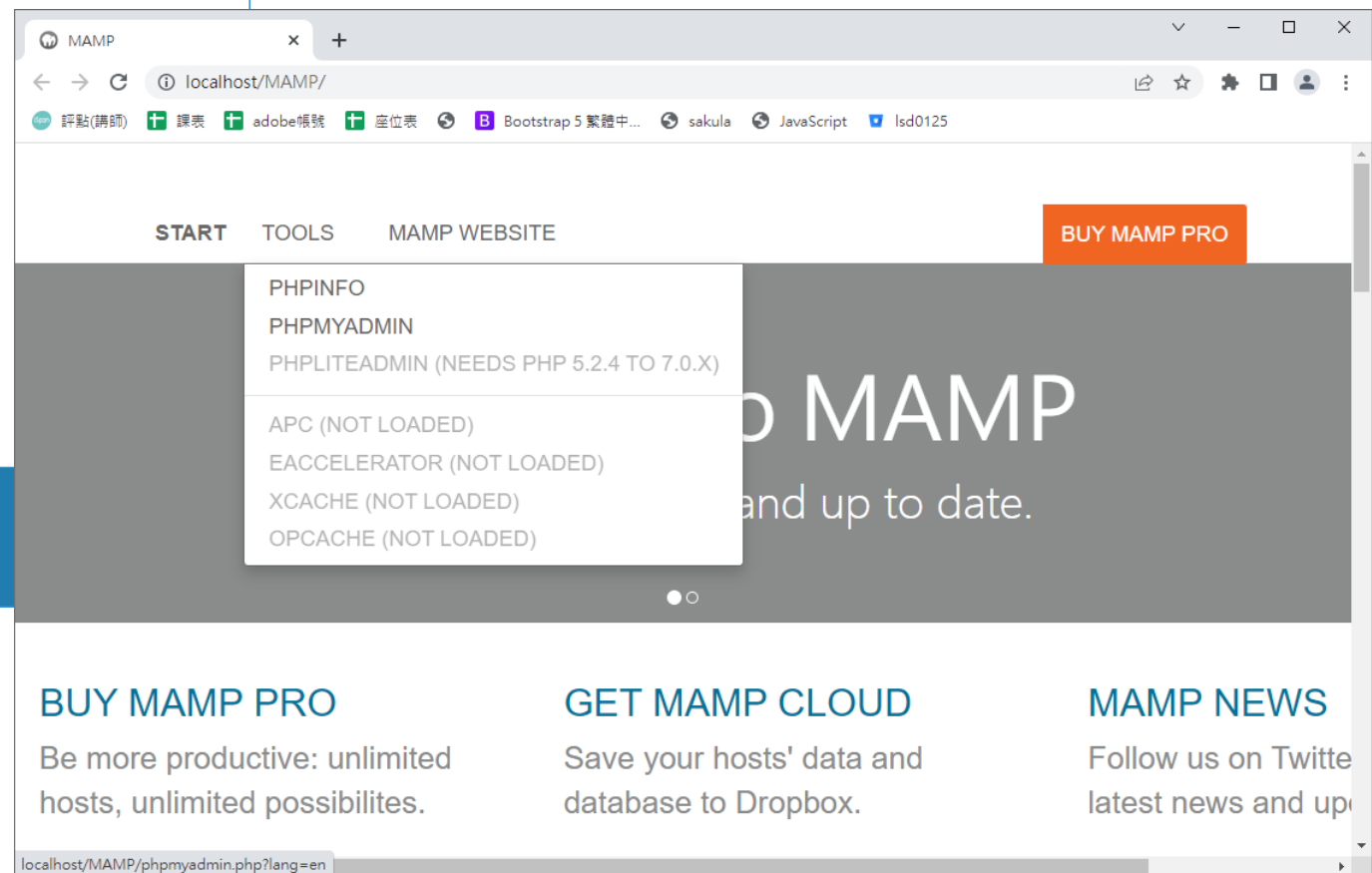


# 第一次啟動

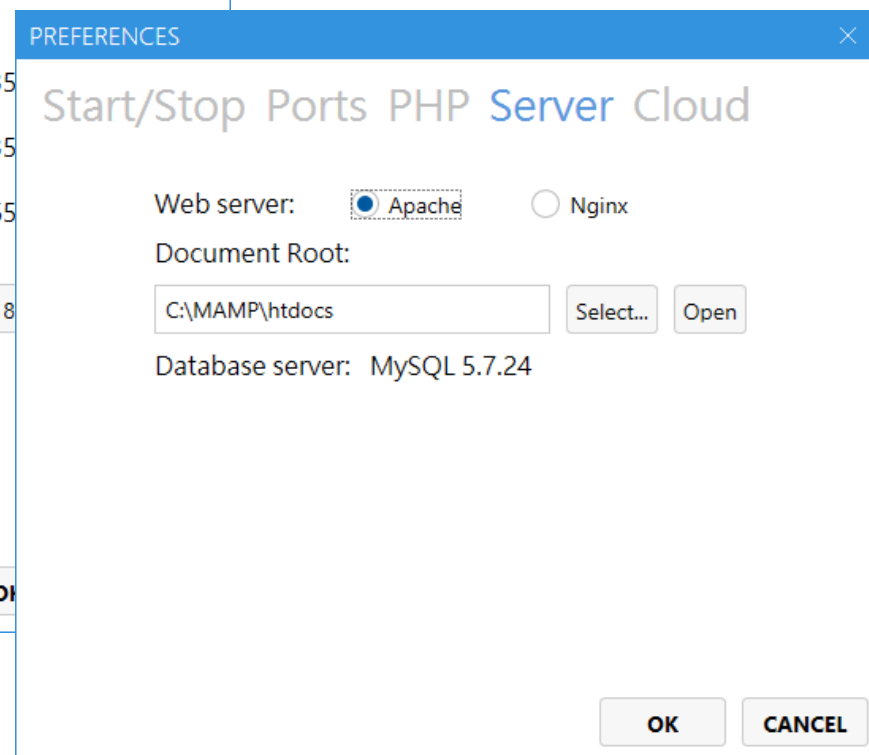
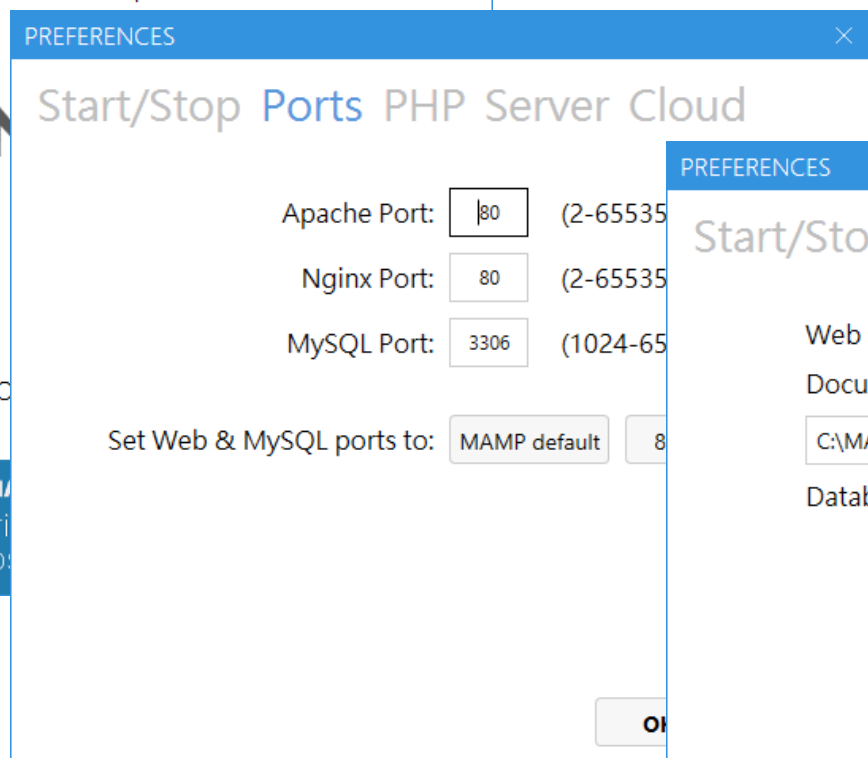
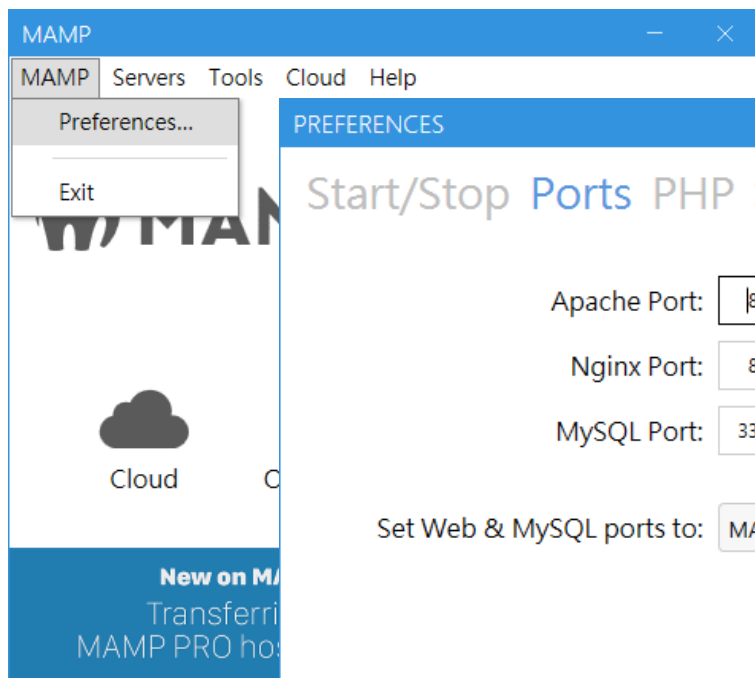




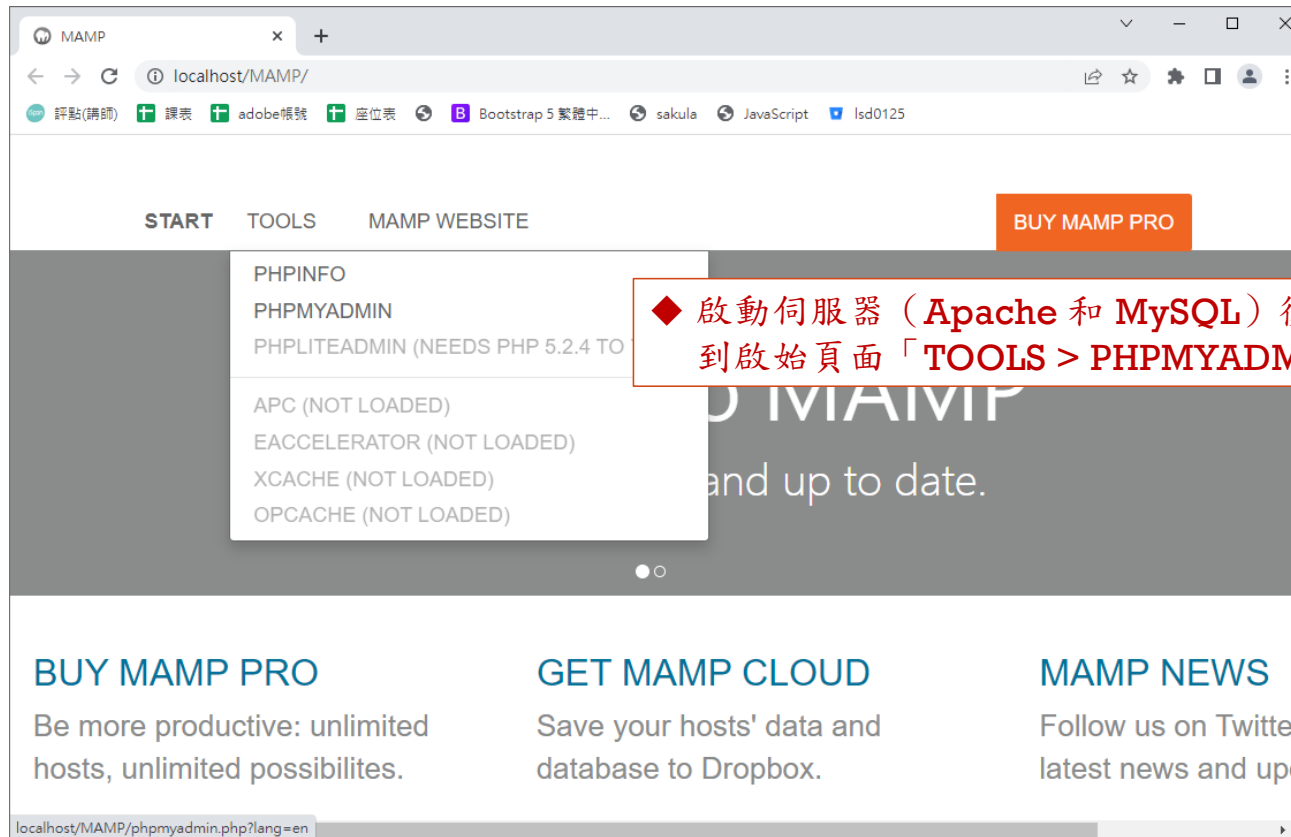
\* 開啟 WebStart page



\* 偏好設定



## 2.2 使用 PhpMyAdmin



## 2.3 MySQL, MariaDB 管理工具

- MySQL Workbench：MySQL 官方提供的管理工具。
- HeidiSQL：只支援 Windows
- DBeaver Community：免費、多平台。
- Adminer：Web 介面，需要 PHP 環境。
- phpMyAdmin：Web 介面，需要 PHP 環境。（\* 功能多、容易上手）



## 2.4 phpMyAdmin 阻擋本機以外的連線

```
# C:\MAMP\conf\apache\httpd.conf
Alias /phpMyAdmin "C:/MAMP/bin/phpMyAdmin/"
<Directory "C:/MAMP/bin/phpMyAdmin/">
    # Options Indexes Includes ExecCGI
    # AllowOverride None
    # Require all granted
    AllowOverride AuthConfig Limit
    Require local
    ErrorDocument 403 /error/HTTP_FORBIDDEN.html.var
</Directory>
```





## 2.5 phpMyAdmin 介面概念

The screenshot shows the phpMyAdmin web interface. On the left is a sidebar with a tree view of databases and tables. The main area displays the 'products' table from the 'proj57' database. The interface includes a top navigation bar, a toolbar, and a main content area with a SQL query editor and a table view. Annotations in red boxes point to specific parts of the interface:

- 1. 左側欄：資料庫列表 (Left sidebar: Database list)
- 2. 麵包屑 (Breadcrumb)
- 3. 對象選單 (Object menu)
- 4. 主要內容區 (Main content area)

The table view shows the following data:

	sid	author	bookname	category_sid	book_id	publish_date	pages	price	isbn	on_st
<input type="checkbox"/>	1	洪一 新、許 瑞珍	圖解C++程式設計	16	PG30036	2010-02-08	624	560	978-986-201-306-9	
<input type="checkbox"/>	2	吳睿紘	圖解資料結構-使用JAVA	1	PG30035	2009-12-15	384	420	978-986-201-281-9	
<input type="checkbox"/>	3	江家 顏、陳	Visual C# 2008網路遊戲程式設計	1	PG30034	2009-11-27	424	480	978-986-201-278-9	

### 3. 建立資料庫



- ① 點選「新增」
- ② 填入所要新增的資料庫名稱「my\_test」
- ③ 點選文字編碼「utf8mb4\_general\_ci」
- ④ 點按「建立」
- 按下「建立」之後，即可建立資料庫，有些版本會顯示建立資料庫的 SQL 語法

```
CREATE DATABASE `my_test` DEFAULT CHARACTER SET utf8mb4  
COLLATE utf8mb4_general_ci;
```



## 4. 建立資料表

- 建立資料表之前，先規劃好用途及 **ER mapping**（Entity-relation mapping）。
- **ER mapping**是所謂實體關係對應。以建立通訊錄為例，通訊錄（簡化後）預計存放料：姓名、**email**、手機號碼、生日、地址和建立時間。
- 這個資料表至少要有 6 欄，外加一個流水號當主鍵，應該要有 7 欄。
- 選定資料庫，並建立新資料表，輸入資料表名稱及欄位數目。



## 4.1 主鍵

伺服器: localhost:3306 > 資料庫: my\_test

結構 SQL 搜尋 查詢 匯出 匯入 操作 權限 預存程序 事件 觸發器 設計器

資料表名稱: address\_book 新增 1 欄位 執行

結構

名稱	類型	長度 / 值	預設值	編碼與排序	屬性	空值 (Null)	索引	A_I	備註
sid	INT		無			<input type="checkbox"/>	PRIMARY	<input checked="" type="checkbox"/>	
							PRIMARY		

- ① 第一個欄位名稱設定為「sid」。
- ② 型態為 INT 整數。
- ③ 選擇 PRIMARY。
- ④ 在跳出的確認面板，按「確定」。
- ⑤ 勾選 A\_I 的核選盒（Auto Increment，自動累加）。

- ◆ 主鍵的意思是該欄位為識別資料的主要欄位，欄位的值不會重複。
- ◆ 主鍵會編列索引以加快資料的查詢，一張資料表最多只會有一欄為主鍵。
- ◆ 若其它欄也要有索引的功能，可設定為索引鍵。
- ◆ 設定為索引鍵可加快查詢，同時也會較耗費磁碟資源。



## 4.2 其它欄位

名稱	類型	長度 / 值	預設值	編碼與排序	屬性	空值 (Null)	索引	AI	備註
sid	INT		無			<input type="checkbox"/>	PRIMARY	<input checked="" type="checkbox"/>	
							PRIMARY		
name	VARCHAR	255	無			<input type="checkbox"/>	---	<input type="checkbox"/>	
email	VARCHAR	255	無			<input type="checkbox"/>	---	<input type="checkbox"/>	
mobile	VARCHAR	20	無			<input type="checkbox"/>	---	<input type="checkbox"/>	
birthday	DATE		無			<input checked="" type="checkbox"/>	---	<input type="checkbox"/>	
address	VARCHAR	255	無			<input type="checkbox"/>	---	<input type="checkbox"/>	
created_at	DATETIME		無			<input type="checkbox"/>	---	<input type="checkbox"/>	

資料表備註：

編碼與排序：

儲存引擎：

utf8mb4\_general\_ci

InnoDB



## 4.2 其它欄位

- ① 填入各欄位名稱。
- ② 選擇適當的資料類型。
- ③ 設定資料長度，尤其是 **VARCHAR** 必須設定長度。
- ④ 時間格式的非必填欄位，請設定為「可以是空值」。
- ⑤ 編碼與排序選擇「**utf8mb4\_general\_ci**」。
- ⑥ 儲存引擎，通常使用「**InnoDB**」不需變更。

- 時間格式的「預設值」可以選「**CURRENT\_TIMESTAMP**」，以使用資料建立時的當下時間。
- 「編碼與排序」，是資料以什麼編碼來看待。編碼從頭到尾都統一用 **utf8** 以避免不必要的亂碼發生。每個資料欄都有「編碼」的設定，只要設定資料表的編碼即可。
- 欄位名稱可不可以使用中文？可以，資料庫名稱、資料表名稱和欄位名稱都可以是中文，不過不建議使用中文，以避免造成不必要的困擾。
- 如果在建立資料表的過程，發現當初設想的欄位數太少，可以在「新增？個欄位」下填入要增加的欄位數，並按「執行」。若欄位數太多，沒用到的只要欄位名稱留白即可忽略。



## 4.3 建立資料表：預覽 SQL

```
CREATE TABLE `my_test`.`address_book`  
(  
  `sid` INT NOT NULL AUTO_INCREMENT ,  
  `name` VARCHAR(255) NOT NULL ,  
  `email` VARCHAR(255) NOT NULL ,  
  `mobile` VARCHAR(20) NOT NULL ,  
  `birthday` DATE NULL ,  
  `address` VARCHAR(255) NOT NULL ,  
  `created_at` DATETIME NOT NULL ,  
  PRIMARY KEY (`sid`)  
)  
ENGINE = InnoDB CHARSET=utf8mb4 COLLATE utf8mb4_general_ci;
```





## 5. 新增資料

① 點選資料表

② 點選「新增」分頁

③ 填入各欄的值

④ 點按「執行」

sid int(11)

name varchar(255)

email varchar(255)

mobile varchar(20)

birthday date

address varchar(255)

created\_at datetime

sid 欄位的值可以不用填，已經設定為Auto Increment

林小新

shinder.lin@gmail.com

0918000000

1985-11-11

新北市

2022-11-05 17:15:59.01

執行

- 新增資料的 SQL 語法

```
INSERT INTO `address_book`  
    (`sid`, `name`, `email`, `mobile`, `birthday`, `address`, `created_at`)  
VALUES  
    (NULL, '林小新', 'shinder.lin@gmail.com', '0918000000', '1985-11-11',  
     '新北市', '2022-11-05 17:15:59.000000');
```

```
INSERT INTO `address_book`  
    (`name`, `email`, `mobile`, `birthday`, `address`, `created_at`)  
VALUES  
    ('林小新', 'shinder.lin@gmail.com', '0918000000', '1985-11-11',  
     '新北市', NOW());
```

- 請注意，上述的欄位名稱都使用「`」包裹，字串的值用單引號「'」包裹。
- **phpMyAdmin** 的新增頁面每次可以新增2筆資料，如果有多筆資料，用其新增頁面就顯得較為不方便。可以使用文字編輯器，將之前新增資料的 **SQL** 敘述複製貼上，再加以編寫。
- **NOW()** 為 **SQL** 函式，可取得當下的時間。



■ 新增多筆資料的 SQL 語法

```
INSERT INTO `address_book`  
    (`name`, `email`, `mobile`, `birthday`, `address`, `created_at`)  
VALUES  
    ('林小新1', 'shin@test.com', '0918000000', '1985-11-11', '新北市', NOW()),  
    ('林小新2', 'shin@test.com', '0918000000', '1985-11-11', '新北市', NOW()),  
    ('林小新3', 'shin@test.com', '0918000000', '1985-11-11', '新北市', NOW()),  
    ('林小新4', 'shin@test.com', '0918000000', '1985-11-11', '新北市', NOW()),  
    ('林小新5', 'shin@test.com', '0918000000', '1985-11-11', '新北市', NOW()),  
    ('林小新6', 'shin@test.com', '0918000000', '1985-11-11', '新北市', NOW());  
-- 註解  
-- 每組值中間以逗號隔開
```



## 6. 編輯資料

伺服器: localhost:3306 » 資料庫: my\_test » 資料表: address\_book

瀏覽 結構 SQL 搜尋 新增 匯出 匯入 權限 操作 觸發器

1 顯示第 0 - 6 列 (總計 7 筆, 查詢用了 0.0002 秒。)

SELECT \* FROM `address\_book`

☐ 效能分析 [ 行內編輯 ] [ 編輯 ] [ SQL 語句分析 ] [ 建立 P...

☐ 全部顯示 | 資料列數: 25 | 篩選資料列: 搜尋此資料表 | 依主鍵排序: 無

Extra options

2

	sid	name	email	mobile	birthday	address	created_at
<input type="checkbox"/> 編輯 複製 刪除	1	林小新	shinder.lin@gmail.com	0918000000	1985-11-11	新北市	2022-11-05 17:15:59
<input type="checkbox"/> 編輯 複製 刪除	3	林小新	shin@test.com	0918000000	1985-11-11	新北市	2022-11-06 13:20:45
<input type="checkbox"/> 編輯 複製 刪除	4	林小新2	shin@test.com	0918000000	1985-11-11	新北市	2022-11-06 13:20:45
<input type="checkbox"/> 編輯 複製 刪除	5	林小新3	shin@test.com	0918000000	1985-11-11	新北市	2022-11-06 13:20:45
<input type="checkbox"/> 編輯 複製 刪除	6	林小新4	shin@test.com	0918000000	1985-11-11	新北市	2022-11-06 13:20:45

3

① 點選「瀏覽」分頁  
② 點按「編輯」，以進入編輯畫面  
③ 或在欲編輯的欄位內雙擊，可「快速編輯」



- 編輯資料的 SQL 語法
- 以下為修改 **name** 欄位的資料後，按「執行」後得到的語法

```
UPDATE `address_book` SET `name` = '林小新0'  
WHERE `address_book`.`sid` = 1;
```

- 使用一張表，**WHERE**後面的資料表名稱可以省略，通常寫成：

```
UPDATE `address_book` SET `name` = '林小新0' WHERE `sid` = 1;
```



## 7. 刪除資料

←T→		sid	name	email	mobile	birthday	address	created_at
<input type="checkbox"/>	 編輯  複製  刪除	1	林小新0	shinder.lin@gmail.com	0918000000	1985-11-11	新北市	2022-11-05 17:15:59
<input type="checkbox"/>	 編輯  複製  刪除	3	林小新1		0918000000	1985-11-11	新北市	2022-11-06 13:20:45
<input type="checkbox"/>	 編輯  複製  刪除	4	林小新2		0918000000	1985-11-11	新北市	2022-11-06 13:20:45
<input type="checkbox"/>	 編輯  複製  刪除	5	林小新3		0918000000	1985-11-11	新北市	2022-11-06 13:20:45
<input type="checkbox"/>	 編輯  複製  刪除	6	林小新4		0918000000	1985-11-11	新北市	2022-11-06 13:20:45
<input type="checkbox"/>	 編輯  複製  刪除	7	林小新5		0918000000	1985-11-11	新北市	2022-11-06 13:20:45
<input type="checkbox"/>	 編輯  複製  刪除	8	林小新6		0918000000	1985-11-11	新北市	2022-11-06 13:20:45

確認

確定要執行「DELETE FROM address\_book WHERE `address\_book`.`sid` = 6」？

確定 取消

確認  
確定要執行「DELETE FROM  
address\_book WHERE  
`address\_book`.`sid` = 6」?

確定 取消

- ① 點按某筆所欲刪除的資料前的「刪除」
- ② 跳出「確認」面板，點擊「確定」刪除

◆ 刪除資料時，要特別注意，若沒有 **WHERE** 子句，將刪除資料表內的所有資料



## 8. DB 連線用戶帳密

- MySQL/MariaDB 預設的使用者是 **root**，密碼為安裝時所設定。
- 在開發環境時，由於是測試環境，使用 **root** 及簡單密碼是沒問題的。
- 如果有隱私問題，或者要直接用發佈時使用的帳號及密碼，就必須設定使用者和密碼。
- 變更密碼時，最好用密碼複製到文字檔儲存，倘若忘記密碼就必須重新安裝 DB。
- 以下是以 **phpMyAdmin** 為操作說明：
  - 1. 到主目錄頁面，
  - 2. 點選「使用者帳號」，此時可以看到「使用者帳號一覽」裡有多個授權的使用者都是 **root**，可是「從哪登入」的主機是不同的。
- 只要主機的 **IP** 或網域名稱看起來不一樣，就視為不同的主機（不管它們最後是不是指向相同的主機）。



phpMyAdmin

伺服器: localhost:3306

資料庫 SQL 狀態 使用者帳號 匯出 匯入 設定 備援 變數

### 使用者帳號一覽

	使用者名稱	主機名稱	密碼	全域權限	允許授權(Grant)	動作
<input type="checkbox"/>	任何	%	否	USAGE	否	編輯權限 匯出 Lock
<input type="checkbox"/>	mariadb.sys	localhost	否	USAGE	否	編輯權限 匯出 Unlock
<input type="checkbox"/>	root	127.0.0.1	否	ALL PRIVILEGES	是	編輯權限 匯出 Lock
<input type="checkbox"/>	root	::1	否	ALL PRIVILEGES	是	編輯權限 匯出 Lock
<input type="checkbox"/>	root	localhost	是	ALL PRIVILEGES	是	編輯權限 匯出 Lock
<input type="checkbox"/>	root	mini2.local	否	ALL PRIVILEGES	是	編輯權限 匯出 Lock

↑ ☐ 全選 已選擇項目: 匯出

新增

新增使用者帳號

1 2 3 4

- ① 點選「首頁」(小房子)
- ② 點按「使用者帳號」分頁
- ③ 點按某帳號的「編輯權限」
- ④ 或者,「新增使用者帳號」



## 新增使用者帳號

### 登入資訊

使用者名稱： 使用文字方塊 shinder ①

主機名稱： 任何主機 % ②

密碼： 使用文字方塊 ..... 強度： 極差

重新輸入： .....

認證外掛程式 原生 MySQL 認證

產生密碼： 產生 ③

### 使用者帳號的資料庫

- ☐ 建立與使用者同名的資料庫，並授予所有權限。
- ☐ 給以 帳號\_ 開頭的資料庫 (username\\_%) 授予所有權限。

全域權限 ☐ 全選 ④

注意：MySQL 權限名稱會以英文表示。

☐ 資料

☐ SELECT

☐ 結構

☐ CREATE

☐ 管理

☐ GRANT

資源限制

注意：設定為 0 即代表無限制。

- ① 填入用戶名稱
- ② 使用「%」時，不能由 **localhost** 登入
- ③ 隨機產生高強度密碼
- ④ 全域權限設定，依需求勾選權限



## 8.1 phpMyAdmin 登入模式

- phpMyAdmin 的設定檔為「**config.inc.php**」。注意，另一個檔名有點雷同的「**config.sample.inc.php**」，就如其名是個**sample**，參考用，改這個是無效的。
- 路徑 `C:\MAMP\bin\phpMyAdmin\`
- 用文字編輯器打開 **config.inc.php**，可以看到預設為 **cookie** 登入的方式。

```
$cfg['Servers'][$i]['auth_type'] = 'cookie';
```

- 在開發環境為了方便，會將登入模式設定為 **config**，並將帳號及密碼放入設定檔，以自動授權。

```
$cfg['Servers'][$i]['auth_type'] = 'config';  
$cfg['Servers'][$i]['user'] = 'root';  
$cfg['Servers'][$i]['password'] = '123456';
```



## 9. 資料備份

- 以phpMyAdmin 的「匯出」做備份時，依範圍大小分成以「主機」、「資料庫」，或者以「資料表」為單位。
- 以「主機」匯出，才会有建立資料庫的 **SQL** 語法；以「資料庫」或「資料表」匯出，則否。
- 一般使用「快速」模式，直接以下載 **SQL** 檔的方式匯出。
- 若需要部份資料，或細部的匯出設定，請使用「自訂」模式匯出。
- 除了以 **SQL** 格式做輸出，也可以使用其它格式。不過其它格式大多是為了將資料轉移到不同的軟體（如 **Excel**）。以資料庫備份的角度應該使用 **SQL** 格式匯出。



phpMyAdmin

最近使用 最愛

- 新增
- information\_schema
- mysql
- my\_test
  - 新增
  - address\_book
- performance\_schema
- proj57
- test

伺服器: localhost:3306 » 資料庫: my\_test » 資料表: address\_book

瀏覽 結構 SQL 搜尋 新增 匯出 匯入 權限 操作 觸發器

### 正在匯出「address\_book」資料表的資料列

**匯出方式:**

☒ 快速 - 僅顯示必要的選項  
☐ 自訂 - 顯示所有可用的選項

**格式:**

SQL

**資料列數:**

☒ 傾印所有資料列  
☐ 傾印部份資料列

資料列數: 6

起始列數: 0

**匯出**

匯出「資料表」

- ① 選定資料表
- ② 選擇「匯出」分頁
- ③ 按「匯出」下載 SQL 檔



# 10. 資料還原

- 資料還原就是資料匯入，匯入備份的資料。
- 在 **phpMyAdmin** 做匯入時，同樣依範圍大小分成以「資料庫」為單位，或者以「資料表」為單位。
- 以「資料庫」為單位匯入時，先進入主目錄，再點選「匯入」選項。
- 以「資料表」為單位匯入時，先點選左側資料庫，再點選「匯入」選項。
- 載入的 **SQL** 檔有大小限制。
- 如果 **SQL** 檔太大時，可以使用命令列工具做匯入。



# 11. CRUD

- Create
- Read
- Update
- Delete
- 匯入參考資料到 proj57 資料庫

<https://github.com/shinder/mmmh57-php/blob/master/proj57.sql>

<https://raw.githubusercontent.com/shinder/mmmh57-php/master/proj57.sql>



# 12. 外鍵設定

- 第一正規化
- 第二正規化

設定 **products** 和 **categories** 兩張表的關聯

- ① 選定 **products** 資料表（打開在規劃時有外鍵的資料表）
- ② 選擇「結構」分頁
- ③ 點選「關聯檢視」
- ④ 限制式屬性，使用預設值
- ⑤ 選定外鍵欄位「**category\_sid**」
- ⑥ 選擇 **categories** 資料表
- ⑦ 選擇 **categories** 的主鍵欄位「**sid**」
- ⑧ 按「預覽 SQL」或「儲存」完成設定



← 伺服器：localhost:3306 » 資料庫：proj57 » 資料表：products 1

瀏覽 2 結構 SQL 搜尋 新增 匯出 匯入 權限 操作 觸發器

資料表結構 關聯檢視 3

外鍵限制式

動作	限制式屬性	欄位 ?	外鍵限制式 (INNODB)		
			資料庫	資料表	欄位
	限制式名稱				
ON DELETE	RESTRICT 4	category_sid 5	proj57	categories 6	sid 7
ON UPDATE	RESTRICT				

+ 新增欄位

+ 加入限制式

預覽 SQL 儲存 8





```
ALTER TABLE `products`  
  ADD FOREIGN KEY (`category_sid`)  
    REFERENCES `categories`(`sid`)  
    ON DELETE RESTRICT ON UPDATE RESTRICT;
```

- 限制式屬性有 4 種選擇，差異為何？
- **CASCADE**
- **SET NULL**
- **NO ACTION**
- **RESTRICT**
- 在合併查詢時，有沒有設定外鍵關聯，並不影響資料的「查詢」。
- 然而，會影響資料的「新增」、「修改」和「刪除」。
- 有設定外鍵，可方便使用圖形化工具查看各資料表的關聯，如下頁圖。



伺服器: localhost:3306 資料庫: proj57

結構 SQL 搜尋 查詢 匯出 匯入 操作 權限 預存程序 事件 觸發器 設計器

未命名 \*

proj57 products

- sid : int(11)
- author : varchar(50)
- bookname : varchar(60)
- category\_sid : int(11)
- book\_id : varchar(30)
- publish\_date : date
- pages : int(11)
- price : int(11)
- isbn : varchar(30)
- on\_sale : tinyint(1)
- introduction : text

proj57 categories

- sid : int(11)
- name : varchar(30)
- parent\_sid : int(11)

圖形化查看 products 和 categories 兩張表的關聯

- ① 選定 proj57 資料庫
- ② 點選「設計器」分頁



# 12.1 關聯架構

- 一對一
- 一對多
- 多對多



# 13. Excel 資料匯入 DB

- 將 **.xlsx** 檔案轉存成 **.csv**，再匯入資料庫。
- 注意 **.csv** 檔案的編碼。
- 注意 **utf-8** 編碼的 **.csv** 檔案，是否包含 **BOM**（Byte order mark）。



# 14. SQL 參考

```
-- 基本 SELECT

SELECT 1 + 5;
SELECT RAND(); -- 亂數

SELECT MD5('123456');
-- e10adc3949ba59abbe56e057f20f883e

SELECT SHA1('123456');
-- 7c4a8d09ca3762af61e59520943dc26494f8941b

SELECT NOW(); -- 當下時間

SELECT 1 FROM `categories`;
```



-- 計算小計

```
SELECT `sid`, `order_sid`, `product_sid`, `price`, `quantity`, `price`*`quantity`  
FROM `order_details`;
```

-- 訂單編號為 11 的小計

```
SELECT  
    `sid`, `order_sid`,  
    `product_sid`, `price`,  
    `quantity`, `price`*`quantity`  
FROM `order_details` WHERE `order_sid`=11 ;
```

-- 訂單編號為 11 的總計價格

```
SELECT  
    SUM(`price`*`quantity`)  
FROM `order_details` WHERE `order_sid`=11 ;
```

```
SELECT  
    `sid`, `order_sid`,  
    `product_sid`, `price`,  
    `quantity`,  
    SUM(`price`*`quantity`)  
FROM `order_details` WHERE `order_sid`=11 ;
```



-- 計算數量

```
SELECT COUNT(*) FROM `products`;  
SELECT COUNT(`sid`) FROM `products`;  
SELECT COUNT(1) FROM `products`;
```

-- 合併查詢：通常不這樣子用，說明用

```
SELECT * FROM `products` JOIN `categories`;  
SELECT COUNT(1) FROM `products` JOIN `categories`;
```

-- 合併查詢：一般使用方式

```
SELECT *  
FROM `products`  
    JOIN `categories`  
        ON `products`.`category_sid`=`categories`.`sid`;
```

-- 欄位命名的一種方式：product\_id, product\_name, product\_price

```
SELECT `products`.*, `categories`.`name`  
FROM `products`  
    JOIN `categories`  
        ON `products`.`category_sid`=`categories`.`sid`;
```



```
-- 資料表別名
SELECT p.*, c.`name`
FROM `products` AS p
      JOIN `categories` AS c
        ON p.`category_sid`=c.`sid`;

-- 欄位別名
SELECT p.*, c.`name` AS 分類名稱
FROM `products` AS p
      JOIN `categories` AS c
        ON p.`category_sid`=c.`sid`;

SELECT p.*, c.`name` 分類名稱
FROM `products` p
      JOIN `categories` c
        ON p.`category_sid`=c.`sid`;
```





```
-- left outer join
```

```
SELECT p.*, c.`name` 分類名稱  
FROM `products` p  
      LEFT JOIN `categories` c  
            ON p.`category_sid`=c.`sid`;
```

```
SELECT p.*, c.`name` 分類名稱  
FROM `categories` c  
      LEFT JOIN `products` p  
            ON p.`category_sid`=c.`sid`;
```



-- 取得某筆訂單的內容

```
SELECT o.*, od.price, od.quantity, p.bookname
FROM orders o
      JOIN order_details od
          ON o.sid = od.order_sid
      JOIN products p
          ON p.sid = od.product_sid
WHERE o.sid = 11;
```

-- 取得某個會員所有訂單細目

```
SELECT o.*, od.price, od.quantity, p.bookname
FROM orders o
      JOIN order_details od
          ON o.sid = od.order_sid
      JOIN products p
          ON p.sid = od.product_sid
WHERE o.member_sid = 1;
```



-- 編號 1 會員 買過哪些商品

```
SELECT od.product_sid, p.bookname, p.price
FROM `orders` o
    JOIN `order_details` od
        ON o.sid=od.order_sid
    JOIN `products` p
        ON p.sid=od.product_sid
WHERE o.`member_sid`=1
GROUP BY od.product_sid;
```

-- 沒有對應到分類的商品

```
SELECT p.*, c.`name`
FROM `products` p
    LEFT JOIN `categories` c
        ON p.`category_sid`=c.`sid`
WHERE c.`name` IS NULL;
```



-- 不為零

```
SELECT * FROM `categories` WHERE parent_sid != 0;
```

```
SELECT * FROM `categories` WHERE parent_sid <> 0;
```

-- LIKE

```
SELECT * FROM `products` WHERE `author` LIKE '吳睿紘';
```

```
SELECT * FROM `products` WHERE `author` LIKE '陳%';
```

```
SELECT * FROM `products` WHERE `author` LIKE '%陳%';
```

```
SELECT * FROM `products` WHERE `author` LIKE '%陳%' OR `bookname` LIKE '%陳%';
```

-- IN

```
SELECT * FROM `products` WHERE sid=6 OR sid=2 OR sid=3;
```

```
SELECT * FROM `products` WHERE sid IN (6, 2, 3);
```

```
SELECT * FROM `products` WHERE sid IN (6, 2, 3) ORDER BY `sid` ASC; -- 升冪
```

```
SELECT * FROM `products` WHERE sid IN (6, 2, 3) ORDER BY `sid` DESC; -- 降冪
```

```
SELECT * FROM `products` WHERE sid IN (6, 2, 3, 10, 15) ORDER BY RAND();
```



```
-- GROUP BY: 群組
-- 各分類的數量
SELECT category_sid, COUNT(1) num FROM `products` GROUP BY `category_sid`;

SELECT p.category_sid, COUNT(1) num, c.name
      FROM `products` p
          JOIN `categories` c
              ON p.`category_sid`=c.sid
      GROUP BY p.`category_sid`;

-- MySQL 8 無法執行
SELECT *, COUNT(1) num FROM `products` GROUP BY `category_sid`;

-- MySQL 8 的錯誤訊息: #1055 - Expression #1 of SELECT list is not in GROUP
BY clause and contains nonaggregated column 'mfeel9.products.sid' which is
not functionally dependent on columns in GROUP BY clause; this is
incompatible with sql_mode=only_full_group_by
-- 解決方式: https://stackoverflow.com/questions/41887460/select-list-is-not-in-group-by-clause-and-contains-nonaggregated-column-inc
-- SET GLOBAL sql_mode=(SELECT REPLACE(@@sql_mode,'ONLY_FULL_GROUP_BY',''));
```



-- 時間

```
SELECT * FROM `orders` WHERE `order_date` > '2016-12-31' AND  
      `order_date` <= '2017-12-31';
```

```
SELECT * FROM `orders` WHERE `order_date` > '2017-09-30' AND  
      `order_date` <= '2017-10-31';
```

-- 錯誤的寫法

```
-- SELECT * FROM `orders` WHERE `order_date` > '2017-09' AND `order_date` <= '2017-10';  
-- date, datetime 做比較運算時, 比較值至少要 'YYYY-MM-DD'
```

-- 2017 年之後的訂單

```
SELECT * FROM `orders` WHERE `order_date` > '2017-01-01';
```

-- 2017-10-03 當天的所有訂單

```
SELECT * FROM `orders` WHERE `order_date` >= '2017-10-03'  
      AND `order_date` < '2017-10-04';
```

-- 2016-06 月份

```
SELECT * FROM `orders` WHERE `order_date` >= '2016-06-01'  
      AND `order_date` < '2016-07-01';
```



-- 子查詢

```
SELECT `product_sid` FROM `order_details` WHERE `order_sid`=11;
```

```
SELECT * FROM `products` WHERE sid IN (  
    SELECT `product_sid` FROM `order_details` WHERE `order_sid`=11  
);
```

```
SELECT p.*, od.price od_price FROM products p JOIN (  
    SELECT `product_sid`, `price` FROM `order_details` WHERE `order_sid`=11  
) od ON p.sid = od.product_sid;
```



```
-- 建立檢視表
CREATE VIEW test_view AS SELECT * FROM `categories`;

SELECT od.*, p.bookname FROM `order_details` od
      JOIN products p ON od.product_sid=p.sid;

CREATE VIEW detail_view AS
      SELECT od.*, p.bookname FROM `order_details` od
      JOIN `products` p ON od.product_sid=p.sid;
```

```
-- 款式，型號，顏色

-- 購物車：多種型態的商品、多個購物車

-- 「標籤」資料表結構
```





# 參考資料

- <https://www.w3schools.com/sql/>
- <https://mariadb.com/kb/en/useful-mariadb-queries/>
- <https://dev.mysql.com/doc/refman/8.0/en/sql-data-manipulation-statements.html>

