資料庫應用
DataBase System Application

劉和師 老師

2020/09/01

何謂資料庫?

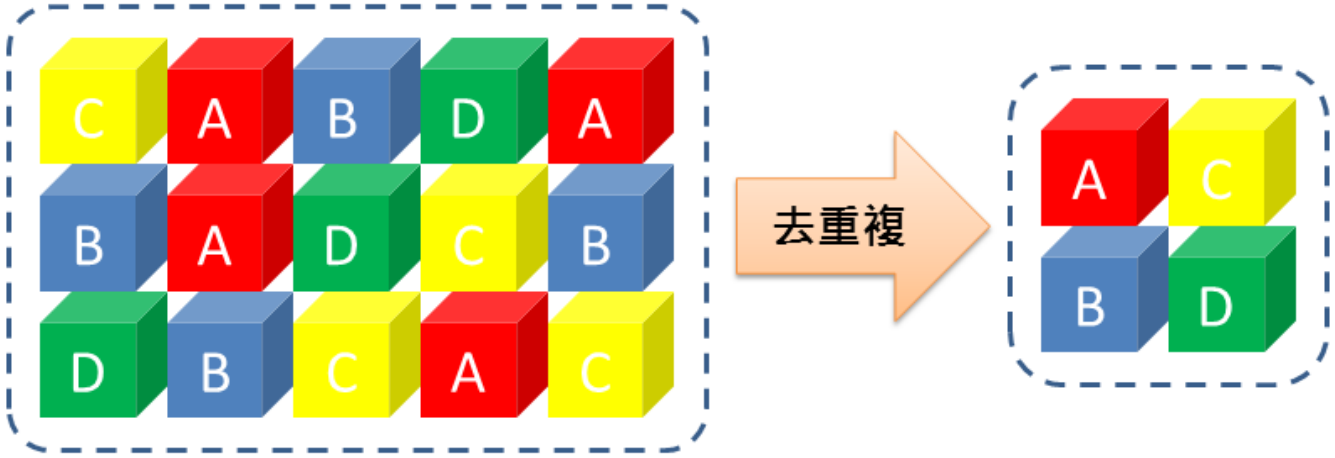
- 廣義的定義：資料庫就是儲存資料的地方
- 狹義的定義：資料庫是由一群相關資料的集合體
- 請說說你所想到的「資料庫」?

資料庫的好處

- 降低資料的重複性
- 達成資料的一致性
- 達成資料的共享性
- 達成資料的獨立性
- 達成資料的完整性
- 避免紙張與空間的浪費
- 達成資料的安全性

資料庫的好處

- 降低資料的重複性：
- 例如：教務處與學務處都有學生基本資料，其實不需要兩邊都儲存。增加了儲存空間及維護的資料麻煩。



資料庫的好處

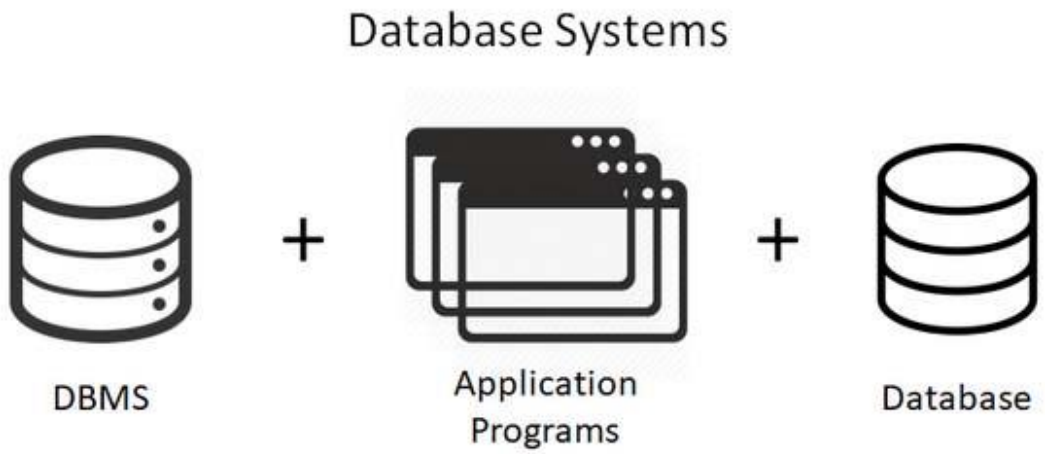
- 達成資料的一致性：
- 例如：學生改名了，教務處改了，學務處卻沒改，造成資料不一致。

資料庫的好處

- 達成資料的共享性：
- 資料集中後便於分享及管理

資料庫的好處

- 達成資料的獨立性：
- 資料與程式分開儲存，修改應用程式與資料本身無關。



資料庫的好處

- 達成資料的完整性：
- 確保資料在新增、修改、刪除時能保持正確而不產生錯誤。正確的資料才是可信賴的資料。

資料庫的好處

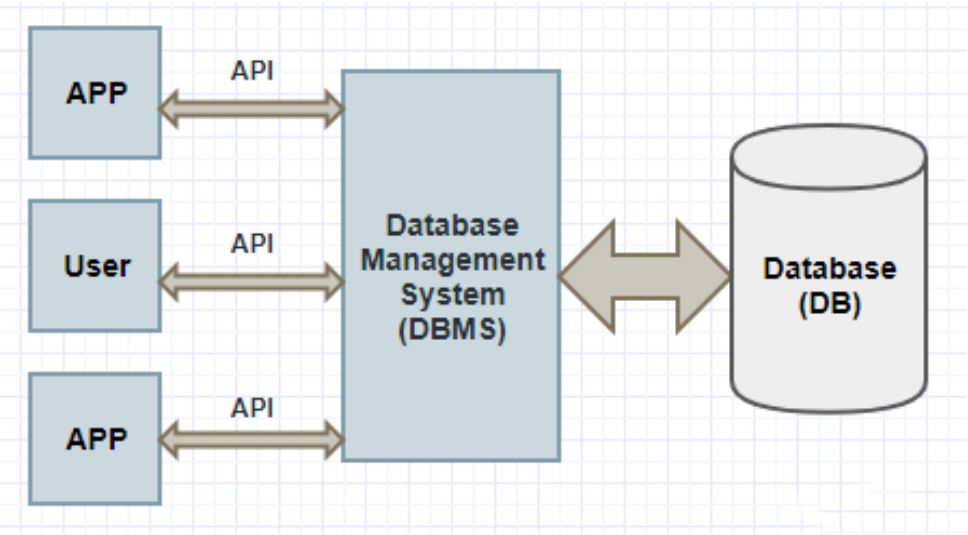
- 避免紙張與空間的浪費：
- 你可以想想以前醫院的病歷表多占空間，以前圖書館書籍書目的查詢都用書目卡，好大一櫃。

資料庫的好處

- 達成資料的安全性：
- 資料庫的資料是重要資產，除了防止非法存取，還要定期備份，以確保資料安全。

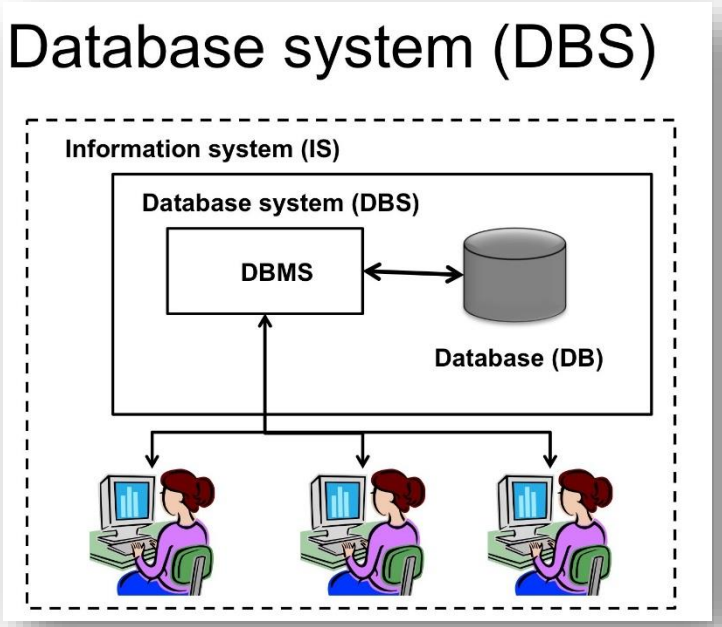
資料庫系統(DataBase System)

- 資料庫系統的組成：
  - 資料庫系統(DataBase System，DBS)：是指DB + DBMS
  - 資料庫(DataBase，DB)：一群相關資料的集合體。
  - 資料庫管理系統(DataBase Management System，DBMS)：管理這些資料庫檔案的軟體。

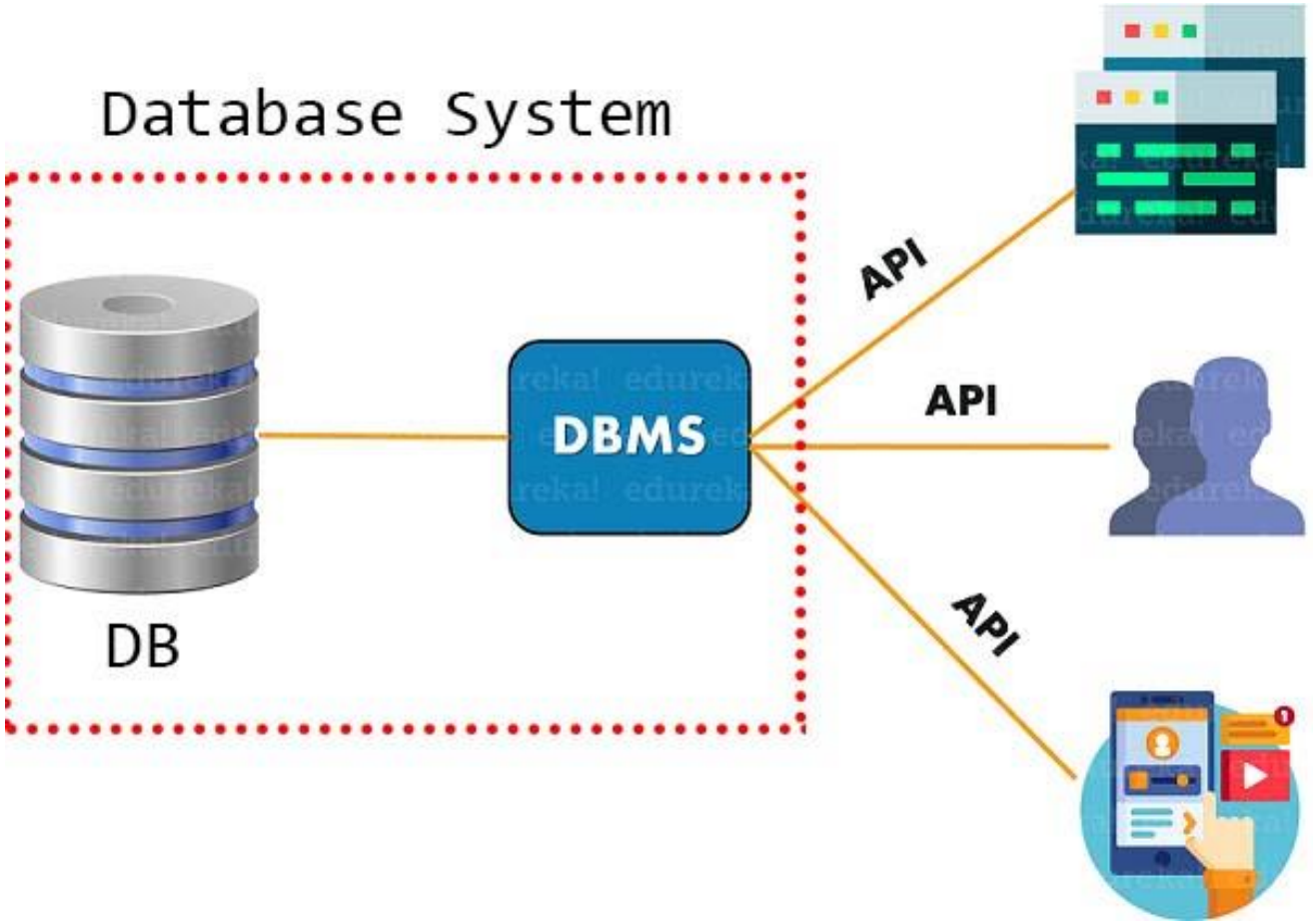


資料庫系統(DataBase System)

- 嚴格來說，一個資料庫系統包括：
  - 資料：即資料庫檔案
  - 硬體：伺服器、硬碟等一切週邊設備
  - 軟體：DBMS及相關應用程式
  - 使用者：
    - 一般使用者(End User)
    - 程式設計師(Programmer)
    - 資料庫管理師(DBA)



資料庫管理系統(DBMS)

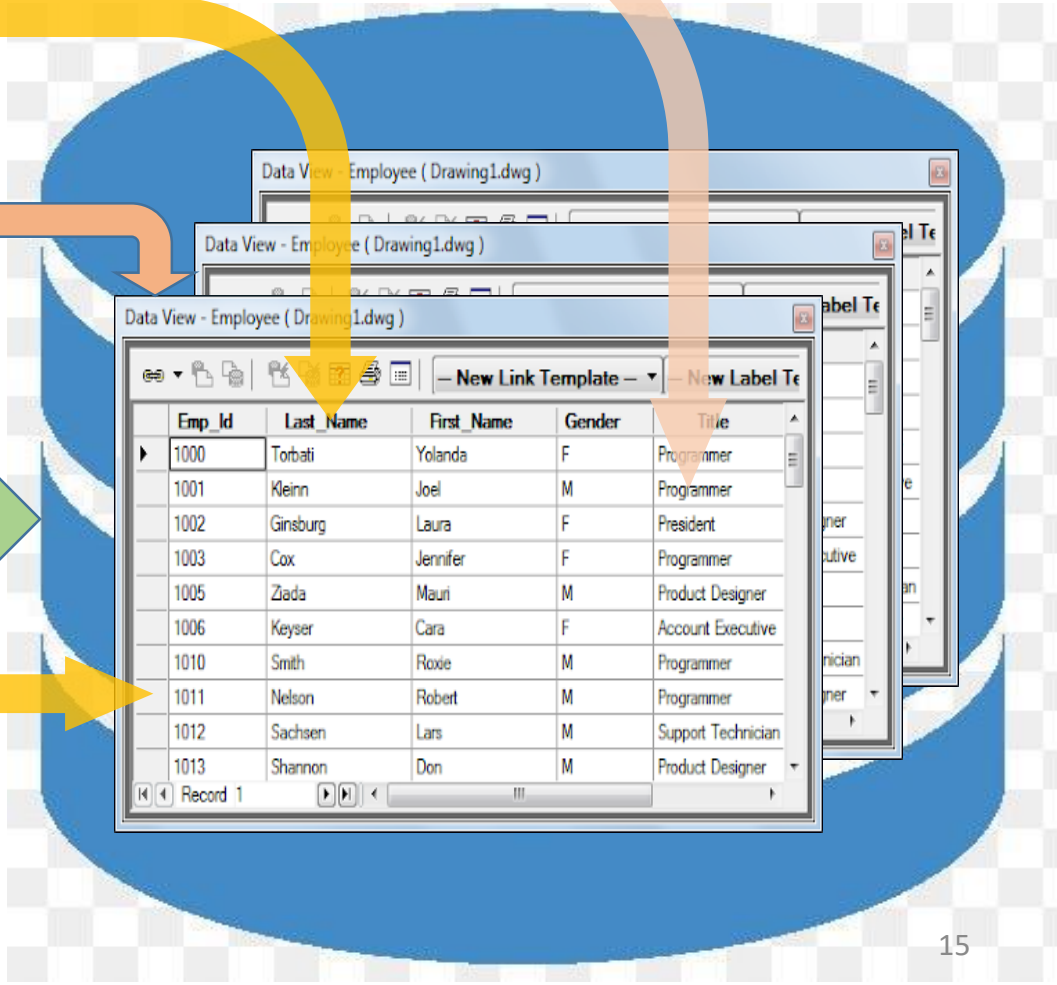


DBMS的功能

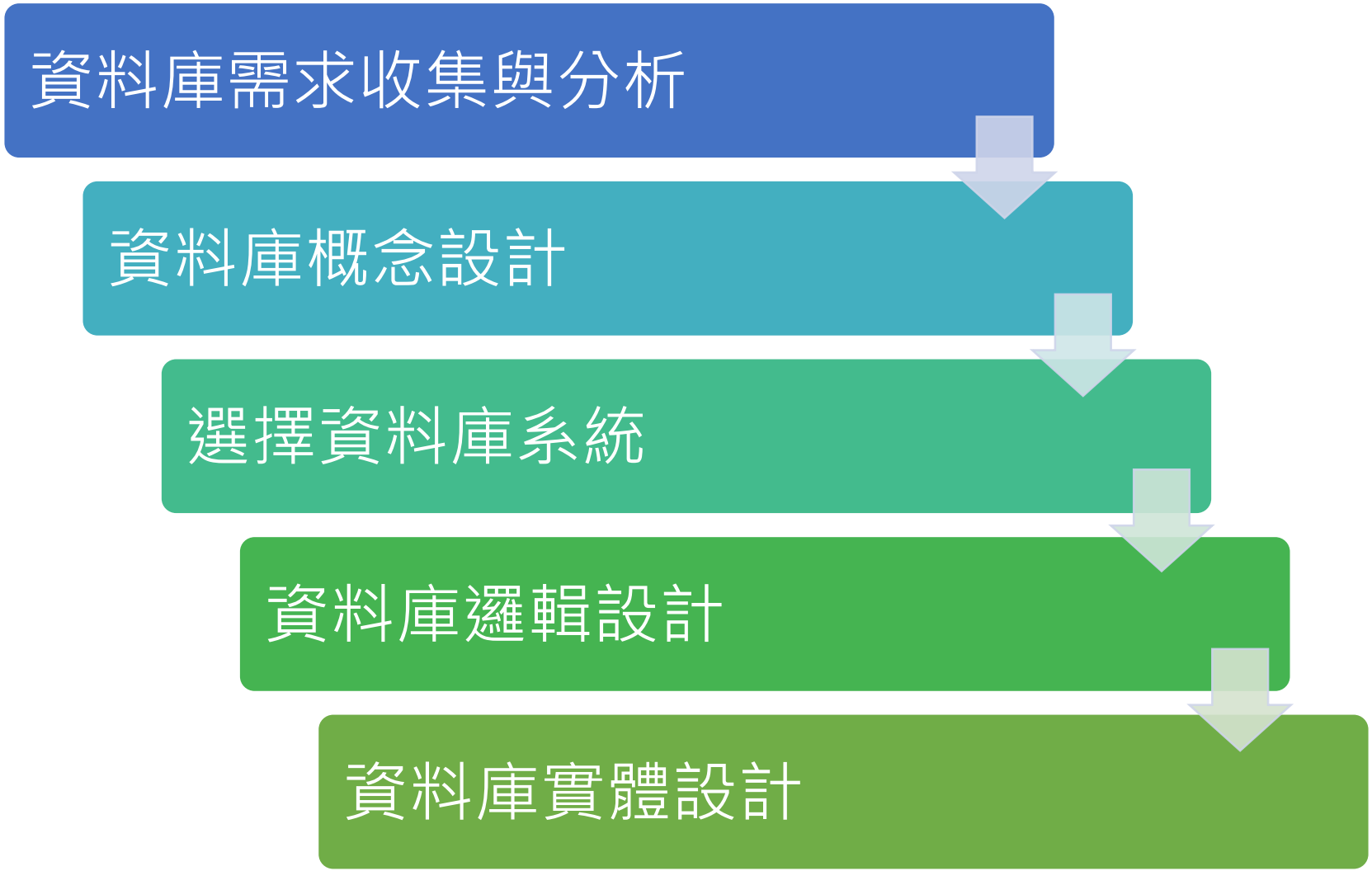
- 資料的定義
- 資料的操作
  - 新增、刪除、修改、查詢、列印報表
- 重複性的控制
- 表示資料之間的複雜關係
- 實施完整性限制
- 提供備份與回復的能力

資料庫的階層

- 字元(Character)
- 欄位(Field)
- 紀錄(Record)
- 資料表(Table)
- 資料庫(Database)



資料庫的設計

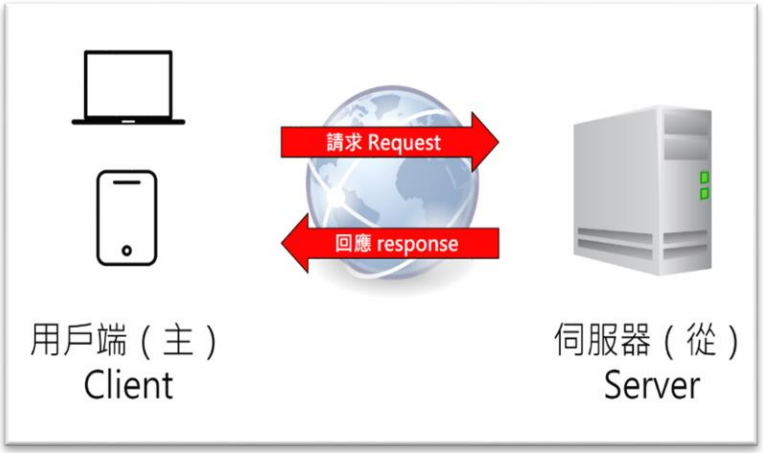


資料庫系統的架構

- 單機架構：
  - 指資料庫與應用程式集中於同一台電腦
- 優點：資料保密性高。
- 缺點：資料無法分享，中大型公司無法適用。

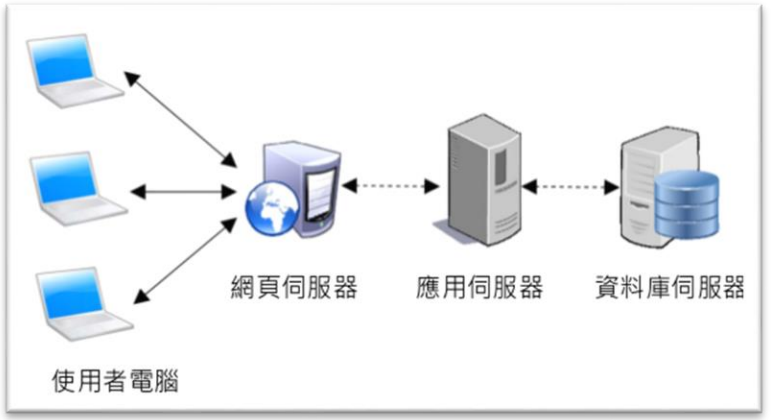
資料庫系統的架構

- 主從式架構：
  - 指資料庫系統獨立放在一台「資料庫伺服器」中，使用者利用本機端應用程式透過網路連接到資料庫。
- 優點：可避免資料的重複以達到資料的一致性，可達成資料共享。
- 缺點：應用程式改版時使用者端需一一安裝，耗費維護時間與精力。



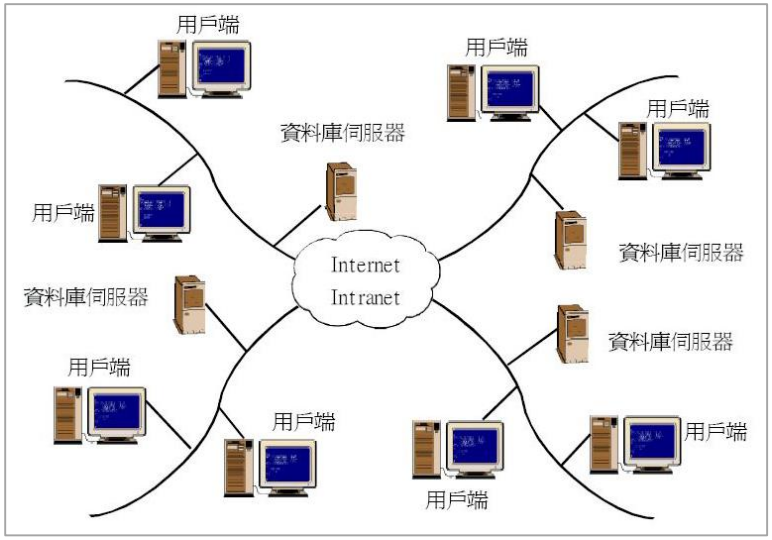
資料庫系統的架構

- 三層式架構：
  - 資料庫系統獨立放在一台「資料庫伺服器」中，應用程式亦獨立放在一台「應用程式伺服器」中，使用者透過網路連接到應用程式伺服器，再由應用程式伺服器去存取資料庫。
- 優點：包含主從式架構的優點，資料分享為全球性的，更新應用程式時只需更新「應用程式伺服器」即可。
- 缺點：伺服器負荷加重，安全性問題增加。



資料庫系統的架構

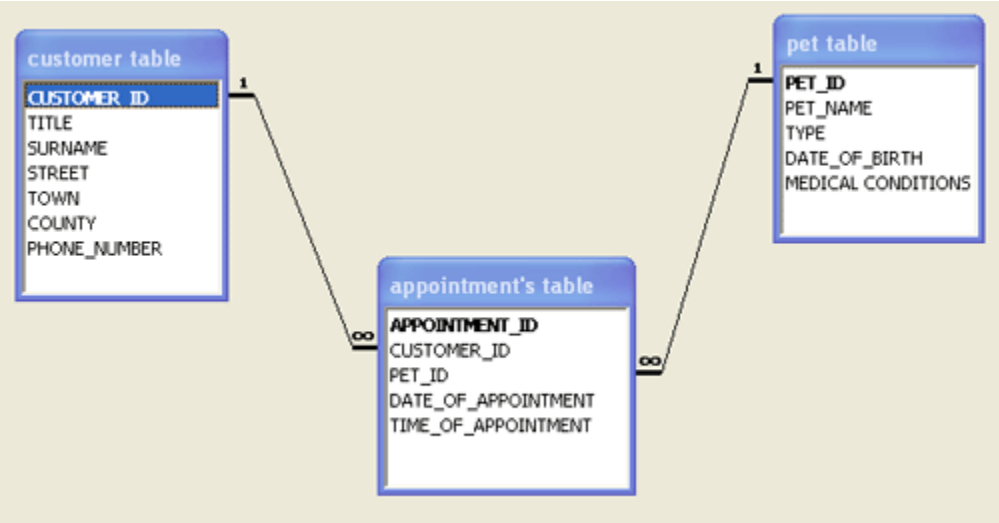
- 分散式架構：
  - 是主從式架構的延伸，當公司規模較大時，各部門分散在不同地區，因此，各部門就會有自己的資料庫系統需求。
- 優點：資料處理速度快，效率較佳，適合分權組織型態，減少主機負荷。
- 缺點：資料分散，易造成資料不一致。



休息一下~

關聯式資料庫

- 是由兩個或兩個以上的資料表組合而成，而資料表之間是透過相同的欄位值(即「外鍵」參考「主鍵」)來連結，以這種方式來存放資料的資料庫就稱關聯式資料庫(Relational database)
- 目前大部分資料庫都採用此方式。



關聯式資料庫

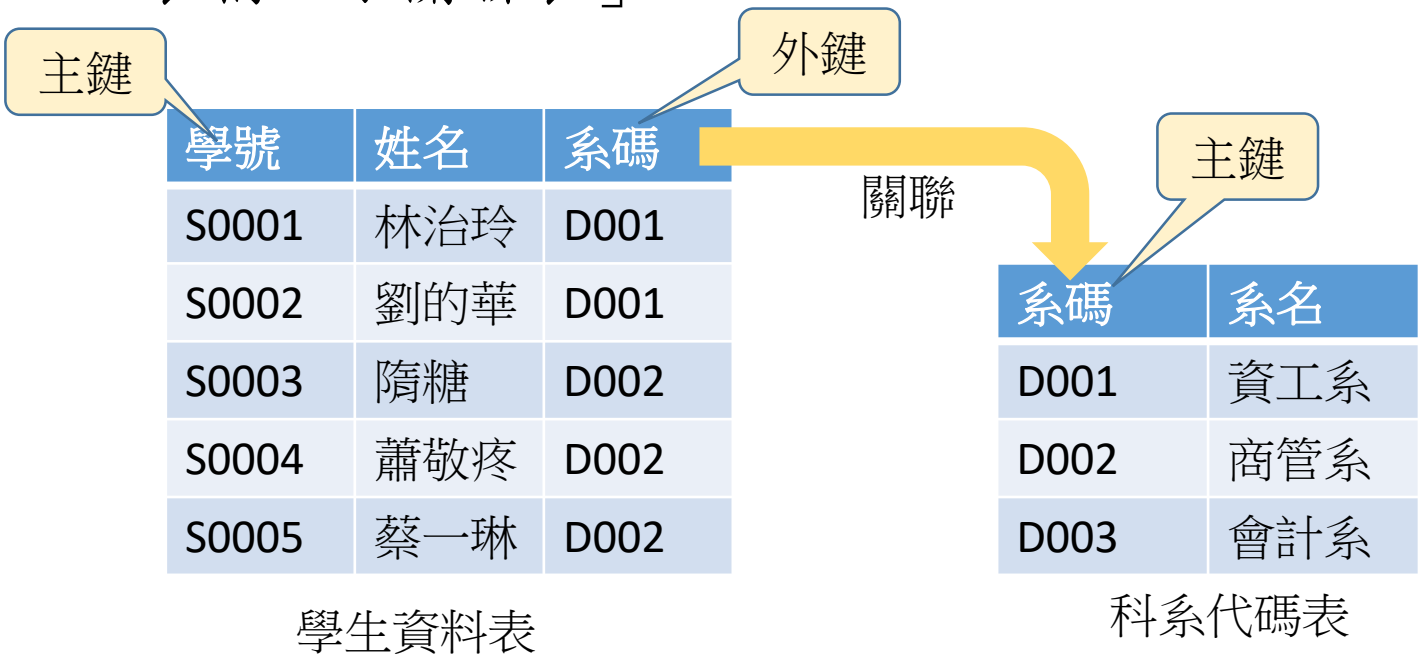
- 主鍵(Primary Key)：
  - 是指用來識別記錄的唯一性，它不可以重複或空直(Null)，例如身份證字號、學生的學號，簡稱PK。

- 挑選主鍵三原則：
  - 一、固定不會再變更的值。
  - 二、單一欄位
  - 三、不可以空值或重複

- (身分證號也可能更改，該如何處理?)

關聯式資料庫

- 外鍵(Foreign Key)：
  - 是指用來建立資料表之間的關係，其外鍵內涵值必須與另一個資料表的主鍵相同。
  - 主鍵所在的資料表稱「父關聯表」，外鍵所在的資料表稱「子關聯表」。

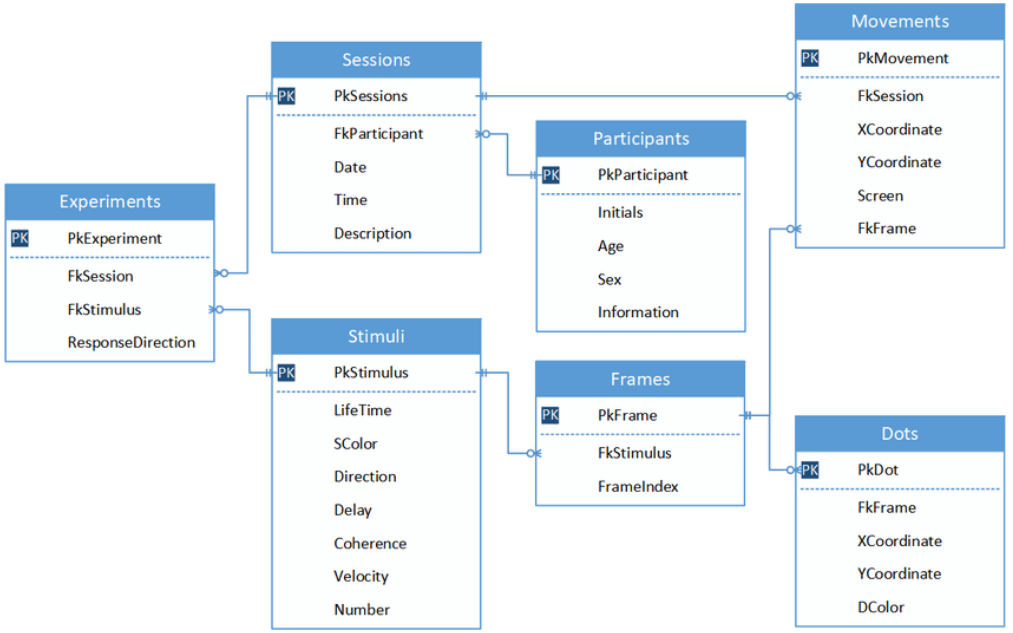


學生資料表

科系代碼表

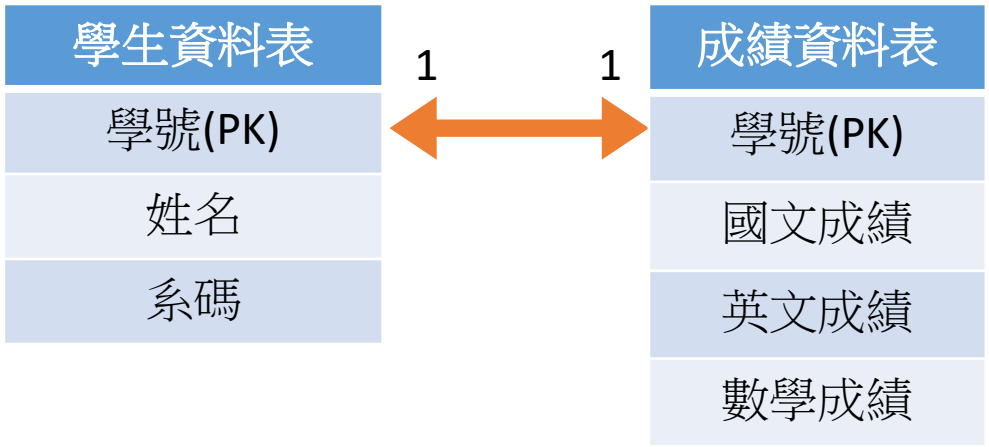
關聯式資料庫

- 關聯式資料庫的種類：
  - 1.一對一的關聯(1：1)
  - 2.一對多的關聯(1：M)
  - 3.多對多的關聯(M：N)



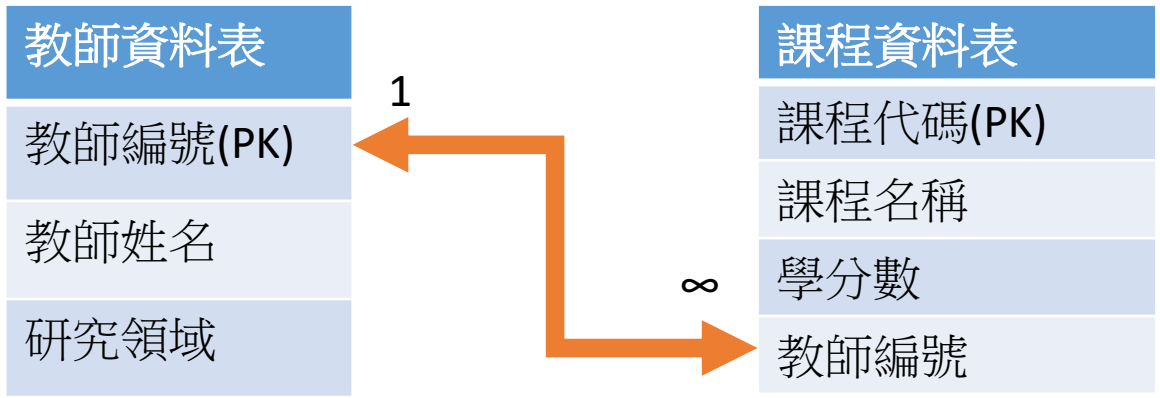
關聯式資料庫

- 關聯式資料庫的種類：
  - 1. 一對一的關聯(1：1)：
    - 假設有A、B兩個資料表，A的一筆紀錄只能對應到B的一筆紀錄，並且B的一筆紀錄也只能對應到A的一筆紀錄。



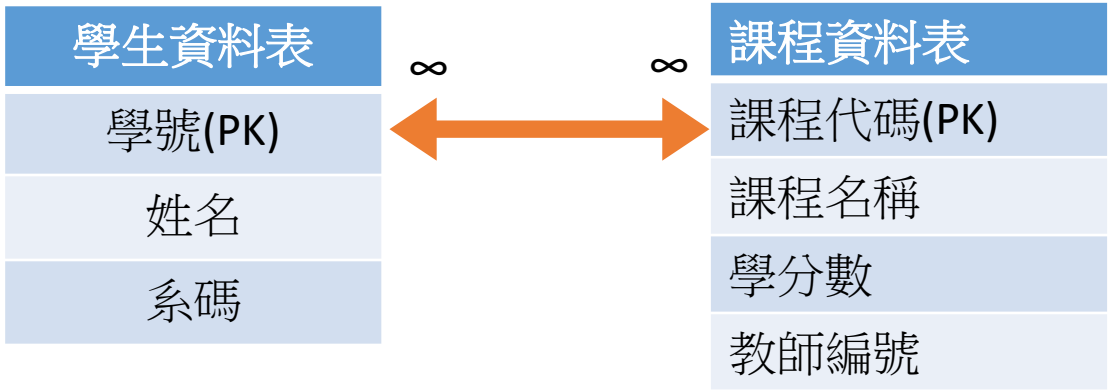
關聯式資料庫

- 關聯式資料庫的種類：
  - 2.一對多的關聯(1：M)：
    - 假設有A、B兩個資料表， A的一筆紀錄可以對應到B的多筆紀錄，而B的一筆紀錄卻只能對應到A的一筆紀錄。



關聯式資料庫

- 關聯式資料庫的種類：
  - 3. 多對多的關聯(M：N)：
    - 假設有A、B兩個資料表， A的一筆紀錄可以對應到B的多筆紀錄，而B的一筆紀錄也能對應到A的多筆紀錄。



關聯式資料完整性規則

- 完整性規則是用來確保資料的一致性與完整性，以避免經過新增、修改及刪除等動作之後，產生異常的現象。
  - 1. 實體完整性規則：是指針對單一資料表，主鍵必須具有「唯一性」，且「不可為空值」。
  - 2. 參考完整性規則：是指針對多個資料表，「子關聯表」的「外鍵」的欄位值一定要存在於「父關聯表」中。
  - 3. 值域完整性規則：是指針對單一資料表，同一資料欄位中的資料屬性必須要相同。

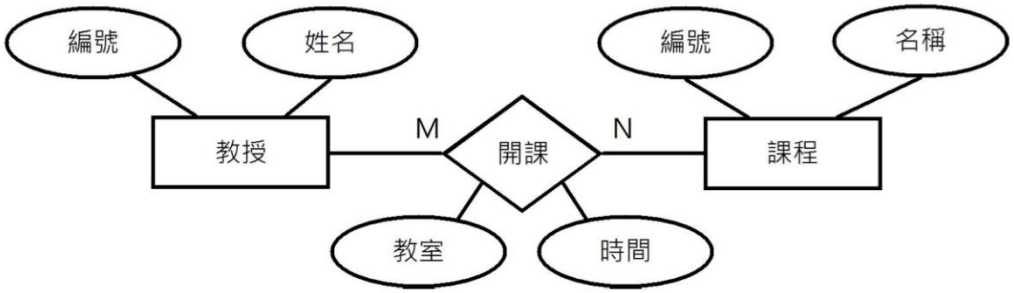
討論

- 如果一個商店，讓你設計資料庫，你覺得：
  - 1. 需要記錄那些資料?
  - 2. 這些資料可能是甚麼性值?
  - 3. 那些可以當主鍵?
  - 4. 需要區分幾個資料表比較好?

休息一下~

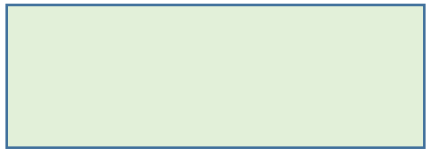
ER Model 實體關係圖

- 實體關係模式(Entity-Relation Model)是利用圖形化的表示法，可以很容易地讓一般非技術人員所了解。因此實體關係模式可視為設計者與使用者溝通的工具與橋梁。
- 例如：學生、課程、老師、員工、產品等都是屬於實體，在實務需求上我們可以將實體轉換成各種資料表。

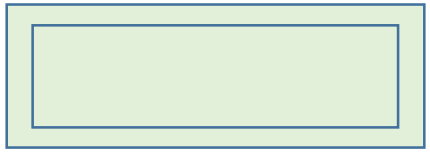


ER Model 實體關係圖

- 實體(Entity)：
  - 是用以描述真實世界的物件，它可以用來描述實際存在的事物(如學生)，也可以是邏輯抽象的概念(如課程)。
  - 必須是以「名詞」來命名，不可以用形容詞或動詞。
- 強實體：不需要依附其他實體而存在的實體(例如學生)。



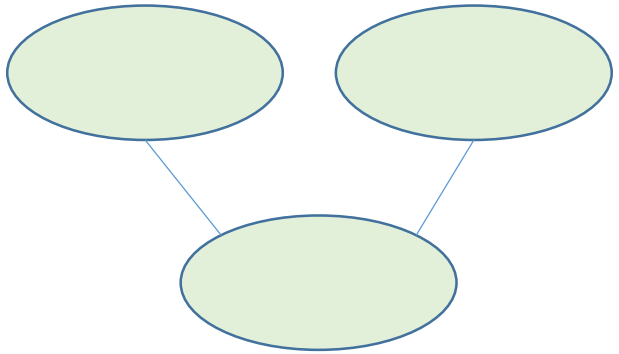
- 弱實體：需要依賴其他實體而存在的實體(例如教職員眷屬)。



ER Model 實體關係圖

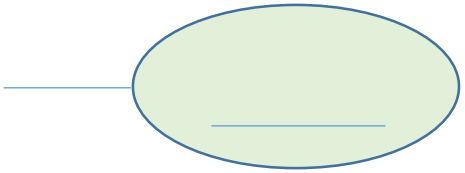
- 屬性(Attribute)：
  - 用來描述實體的性質。
  - 例如姓名、學號、科系等，是用來描述學生這個實體的性質。
- 簡單屬性：指已經不能再分割為更小單位的屬性(例如學號)。

- 複合屬性：由兩個或兩個以上的屬性組合而成，並且代表將來該屬性可以進一步切割(例如地址)。



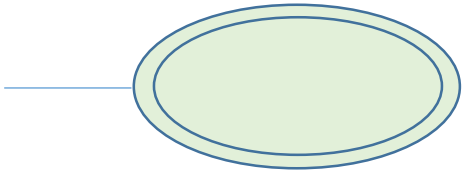
ER Model 實體關係圖

- 鍵屬性：指該屬性的值在某個環境下具有唯一性。

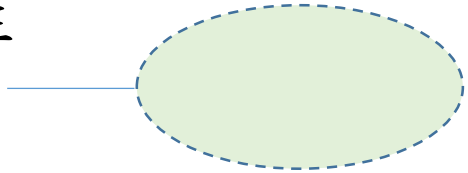


- 單值屬性：指屬性中只會存在一個單一值(例如學號)。

- 多值屬性：指屬性中會存在多個值(如連絡電話)。

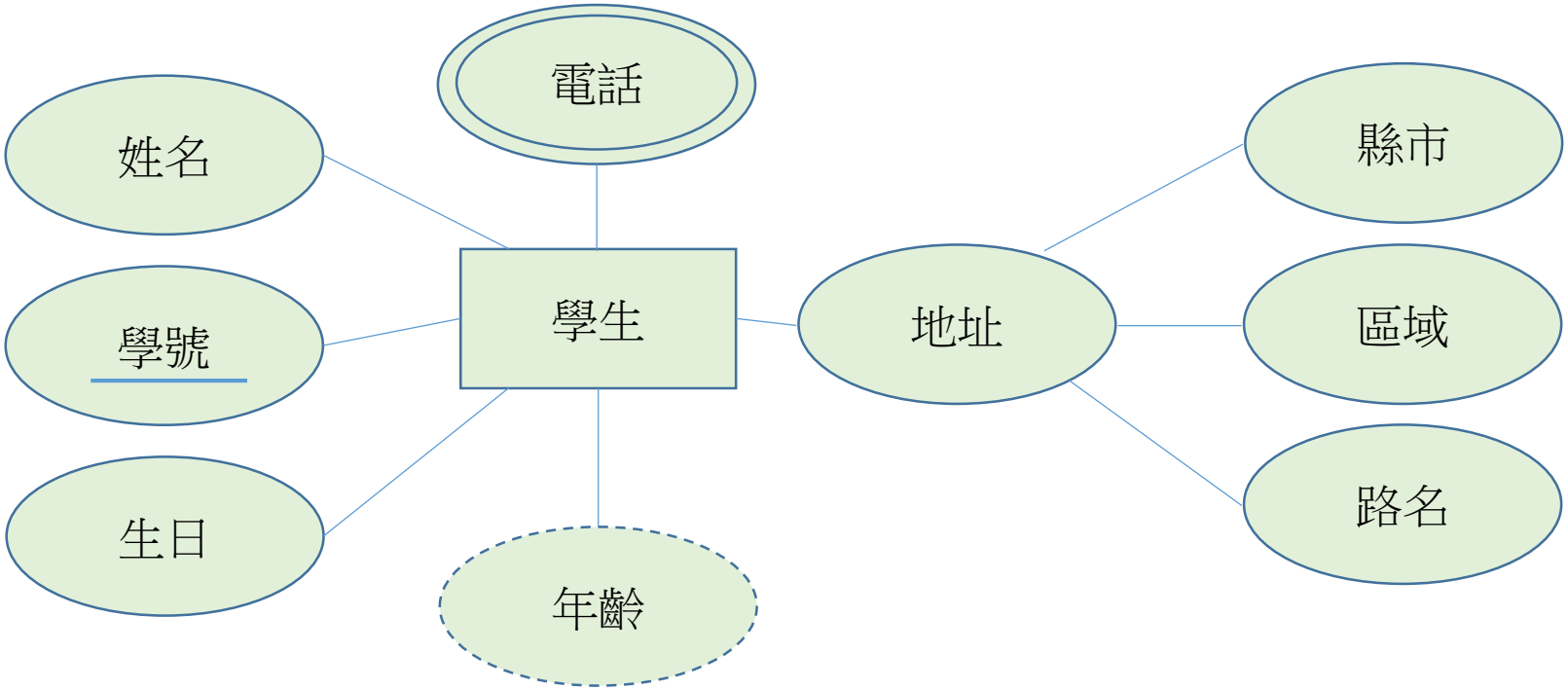


- 衍生屬性：可由其它欄位或屬性計算而得(如年齡)



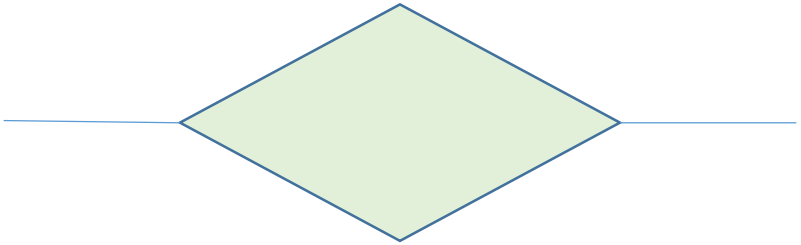
ER Model 實體關係圖

- 例：一個「學生」實體與他的屬性的實體與屬性圖：



ER Model 實體關係圖

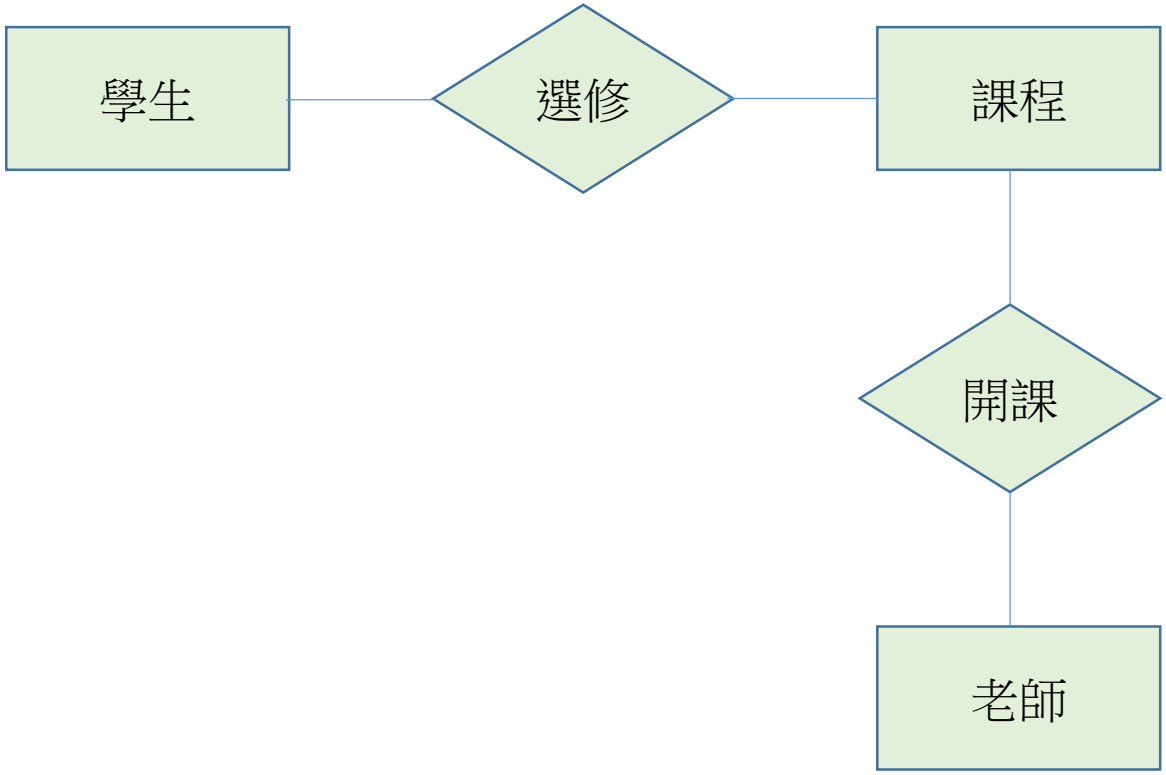
- 關係(Relation)：
  - 是指用來表達兩個實體之間所隱含的關聯性。



  - 使用足以說明關聯性的「動詞」或「動詞片語」命名。例如：選修、開課、分配等。

ER Model 實體關係圖

- 一個ER Model的例子：



ER Model 實體關係圖

- 一對一的關係(1:1)：

- 例如：

ER Model 實體關係圖

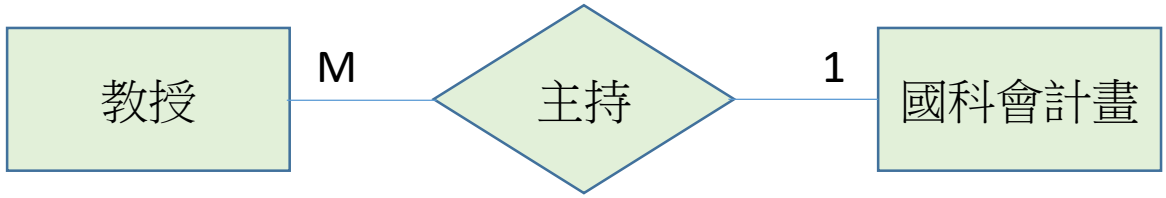
- 一對多的關係(1:M)：

- 例如：

ER Model 實體關係圖

- 多對一的關係(M:1)：

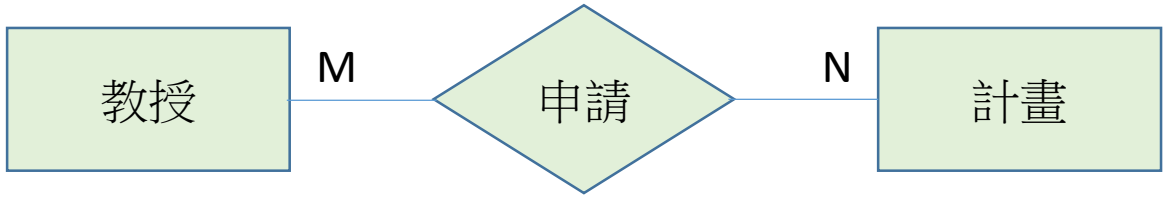
- 例如：



ER Model 實體關係圖

- 多對多的關係(M:N)：

- 例如：



ER Model 實體關係圖

- 利用「雞爪圖基數性」來表示實體之間的關係。
- 強制單基數：
  - 指一個實體參與其關係的案例最少一個，最多也一個。

- 例如：
  - 每一位老師僅能分配一間研究室，每一間研究室一定要分配給一個老師。

ER Model 實體關係圖

- 強制多基數：
  - 指一個實體參與其關係的案例最少一個，最多可以有多個。

- 例如：
  - 每一位教授最少要指導一位研究生，也可以多位，但每一位研究生只能被一位教授指導。

ER Model 實體關係圖

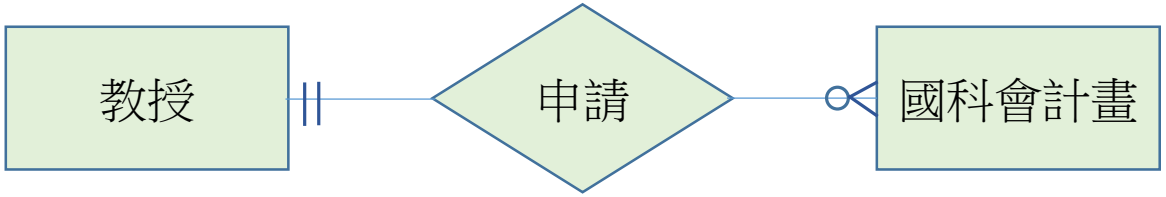
- 選擇單基數：
  - 指一個實體參與其關係的案例最少0個，最多一個。

- 例如：
  - 每一位教授可以分配一個助教，也可以沒有，但每一位助教只能分配給一個教授。

ER Model 實體關係圖

- 選擇多基數：
  - 指一個實體參與其關係的案例最少0個，最多有多個。

- 例如：
  - 每一位教授可以申請多個國科會計畫，也可以不申請，但每一件計畫至少要有一位教授來申請。



ER Model 實體關係圖

- 綱要(Schema)：
  - 是資料庫中全體資料的邏輯結構和特徵描述，它只是型態的描述，不涉及具體的值。
  - 例如我們需要一個儲存老師編號的欄位，那「欄位名稱」就用「老師編號」(實務上欄位名稱盡量用英文啦)，然後指定可以儲存5個字元(Char型態)。
  - 綱要就是對資料表(Table)的描述。
  - 例如：

#	名稱	數據類型	長度/設置	無符號的	允許NU...	填零	默認
1	老師編號	CHAR	5	☐	☐	☐	無默認值
2	老師姓名	CHAR	4	☐	☐	☐	無默認值
3	研究領域	CHAR	10	☐	☑	☐	NULL
4	薪資	INT	11	☐	☑	☐	NULL

ER Model 實體關係圖

- 實例(Instance)：
  - 綱要的具體值稱為綱要的一個實例。
  - 就是資料表裡面的每筆紀錄，這裡會看到真實的資料，而不是資料性質的描述。
  - 例如：

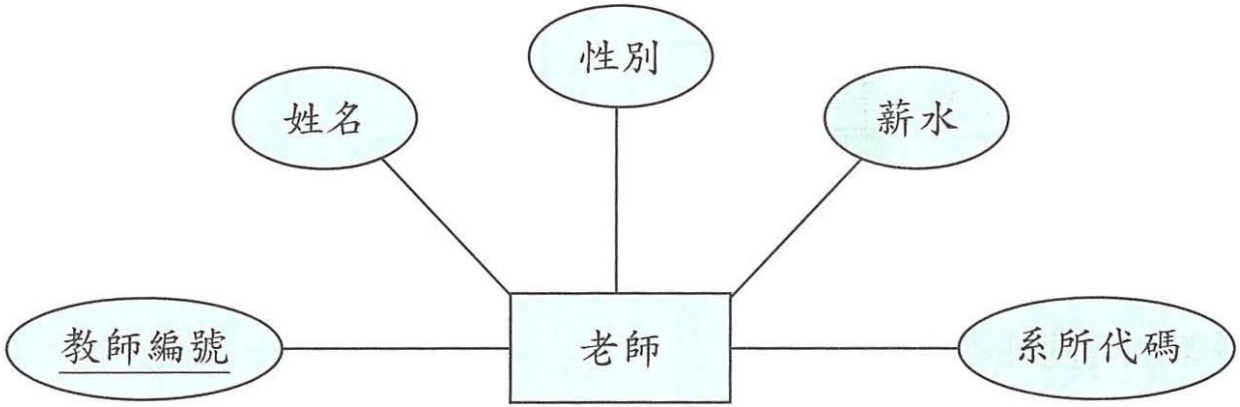
老師編號	老師姓名	研究領域	薪資
T0001	一心	手機程式設計	68000
T0002	二聖	機器人程式設計	75000
T0003	三多	物聯網	85000
T0004	四維	大數據分析	95000
T0005	五福	雲端資料庫系統	110000

ER Model 實體關係圖

- 將實體與屬性轉換為資料表與欄位：
  - 1. 每一個「實體」名稱轉換成一個「表格」名稱。
  - 2. 每一個實體的「屬性」轉換為該表格的「欄位」名稱。
  - 3. 每一個實體的「鍵值屬性」轉換為「主鍵欄位」。
  - 4. 如果鍵值屬性為「複合欄位」，則這複合屬性的所有欄位皆為主索引欄位。

ER Model 實體關係圖

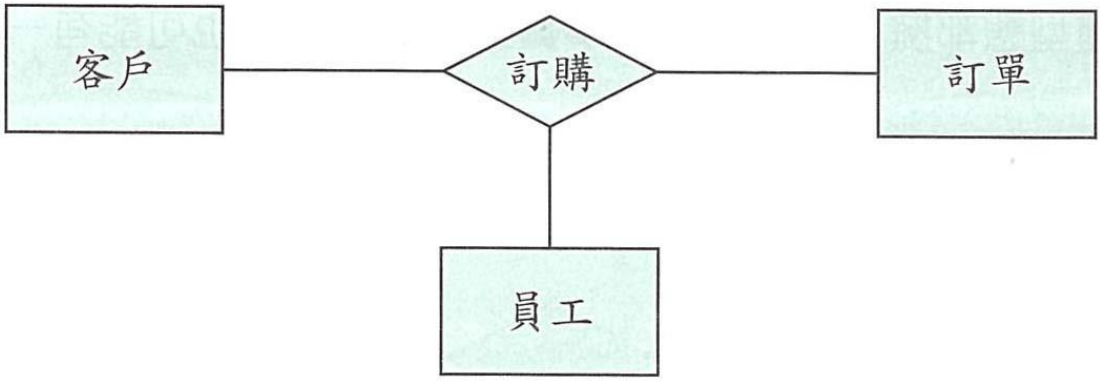
- 將下列ER圖轉換成資料表：



欄位名稱	類型	長度	允許Null	. . .
教師編號	CHAR	5	N	
姓名	CHAR	10	N	
性別	CHAR	1	N	
薪水	INT	11	Y	
系所代碼	Char	5	N	

ER Model 實體關係圖

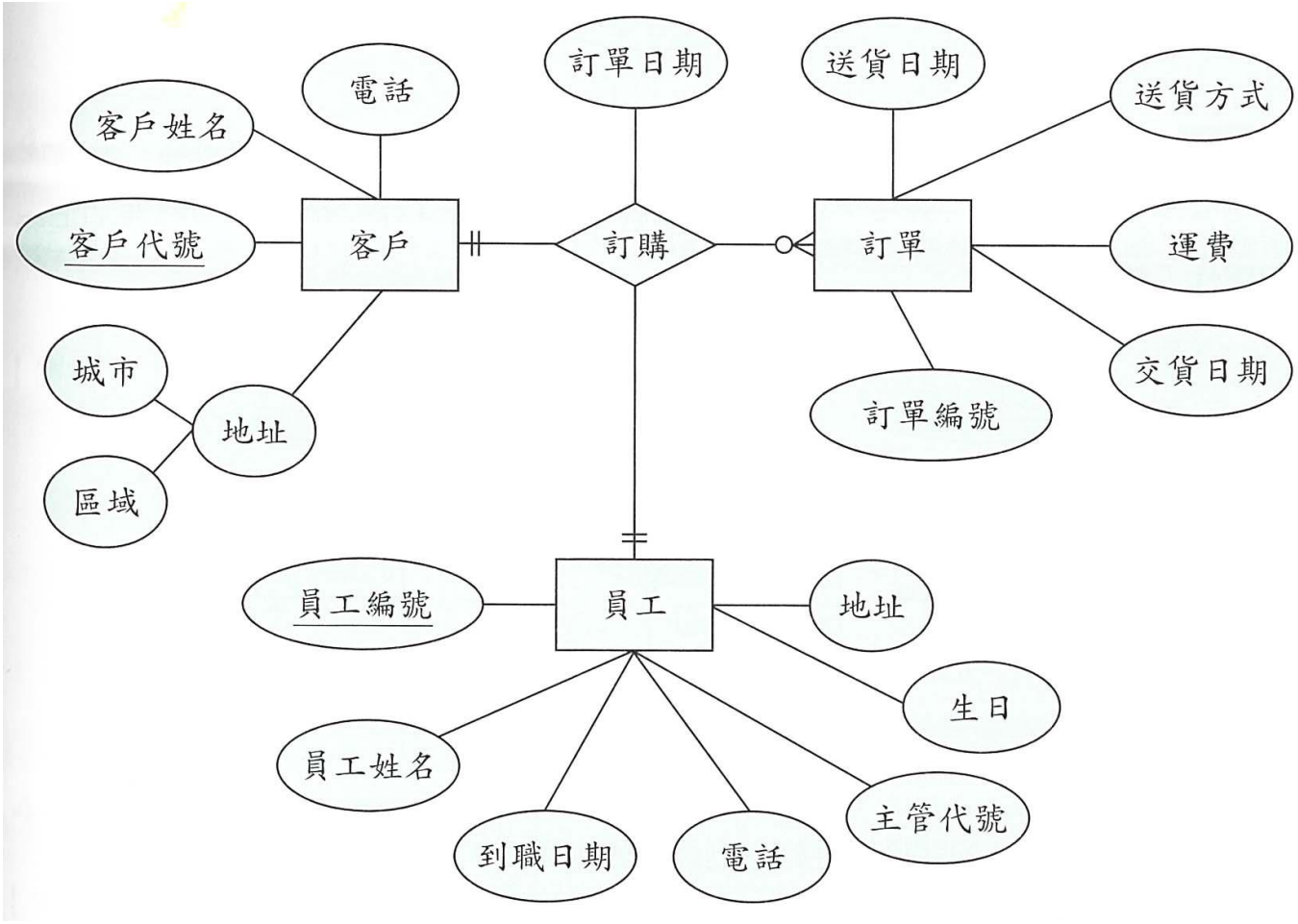
- 案例一：假設客戶、員工與訂單之間的關係為「訂購」，如圖示



- 假設每一位客戶可以訂購一張以上的訂單，也可以沒有下任何訂單，但是，每一張訂單必須會有一位客戶的訂購資料，並且每一張訂單必須要有一位員工負責客戶的訂購資料。請畫出它的ER Model。

ER Model 實體關係圖

- 案例一解答：



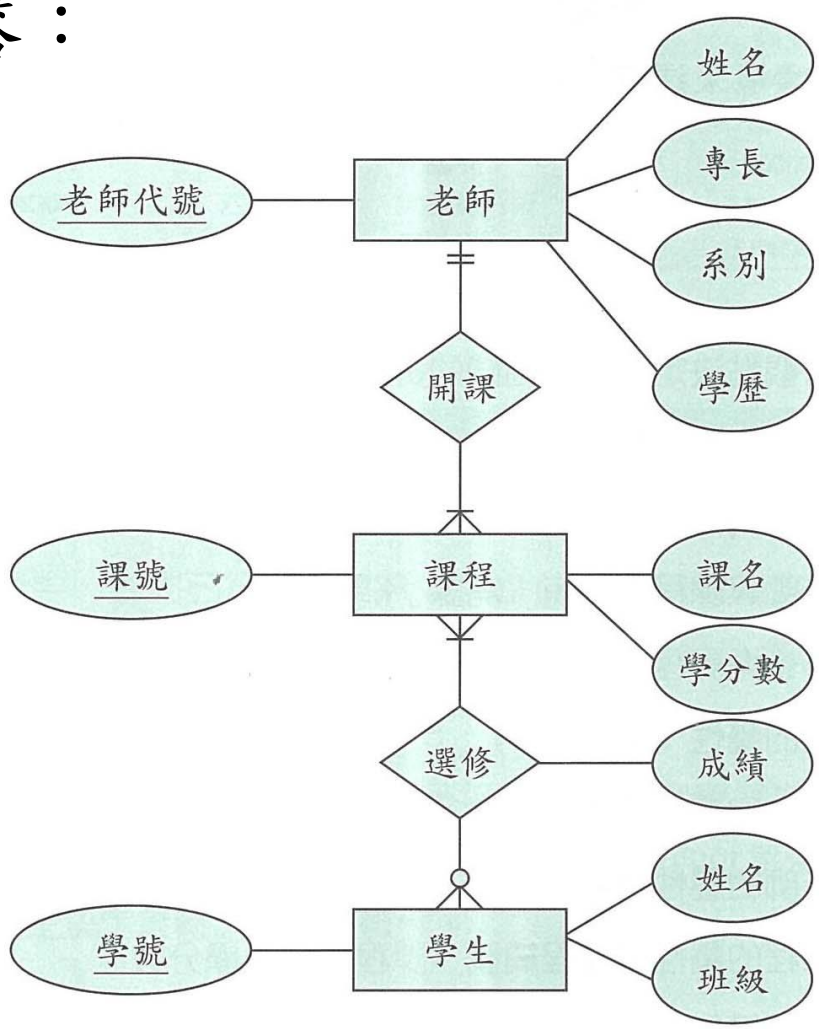
ER Model 實體關係圖

- 案例二：
  - 情境一：假設每一位「老師」必須要開設一門以上的「課程」，並且每一課程只能有一位老師來開課，不能有多位老師開設相同的課程。
  - 情境二：假設每一位「學生」必須要選修一門以上的「課程」，而每一門課程可以被多位學生來選修。

- 請依照以上兩個情境來建立學生、老師、課程之選課的資料庫系統ER圖。

ER Model 實體關係圖

- 案例二解答：



休息一下~

資料庫正規化

- 正規化的概念：
  - 資料庫是用來存放資料的地方，建置之前妥善規劃資料庫綱要(Schema)是很重要的。至於建置規劃的好不好，就要使用正規化(Normalization)的方法來檢視了。

- 正規化的目的：
  - 正規化是為了降低資料的重複性與避免更新異常的情況發生。
  - 正規化的精神是讓資料庫中重複的資料欄位減到最少，並且能快速找到資料，提高效能。

資料庫正規化

- 降低資料重複性：例如以下的資料表設計造成重複的資料，應使用正規化方法，將其分成三個資料表。

資料庫正規化

- 避免資料異常更新：
  - 新增異常：
    - 在新增某些資料時必須同時新增其他資料，否則會產生新增異常現象。
  - 修改異常：
    - 在修改某些資料時必須同時修改其他資料，否則會產生修改異常現象。
  - 刪除異常：
    - 在刪除某些資料時必須同時刪除其他資料，否則會產生刪除異常現象。

資料庫正規化

- 假設某學校開設課程，學員課程收費表如下，我們逐一探討各項異常。

學員課程收費表

學號	課號	學分費
S0001	C001	3000
S0002	C002	4000
S0003	C001	3000
S0004	C003	5000
S0005	C002	4000

資料庫正規化

- 新增異常：
  - 假設要新增C004課程，但此課程無法立即新增到資料表，除非至少有一位學員選修了C004這門課。

	學號	課號	學分費
#1	S0001	C001	3000
#2	S0002	C002	4000
#3	S0003	C001	3000
#4	S0004	C003	5000
#5	S0005	C002	4000

無法新增

#6	Null	C004	3000

資料庫正規化

- 修改異常：
  - 假如C002課程的學分費改為4500元時，若此課程有多位學員選修，可能會有紀錄未修改到，造成資料不一致。

	學號	課號	學分費
#1	S0001	C001	3000
#2	S0002	C002	4000 調整→ 4500
#3	S0003	C001	3000
#4	S0004	C003	5000
#5	S0005	C002	4000 忘了調整

造成 C002 課程的學分費不一致現象

資料庫正規化

- 刪除異常：
  - 假設學員S0004退選時，同時也刪除了C003這門課程，由於這門課只有S0004這位學員選修，若刪除這一筆，將失去C003這門課及其學分費資訊。

	學號	課號	學分費
#1	S0001	C001	3000
#2	S0002	C002	4000
#3	S0003	C001	3000
#4	~~S0004~~	~~C003~~	~~5000~~
#5	S0005	C002	4000

失去 C003 課程及其相關資訊

資料庫正規化

- 解決方法：
  - 採用正規化的手段，將資料表分開成兩個。

資料庫正規化

- 功能相依(Functional Dependence; FD)的概念：
  - 是指資料表中各個欄位之間的相依性，亦即某欄位不能單獨存在，必須要和其他欄位一起存在時才有意義。
  - 例如：

姓名	地址	電話	生日	性別

  - 地址不能單獨存在，一定要與姓名一起才有意義，其他欄位也是，若失去了姓名欄位就都沒有意義了。

資料庫正規化

- 功能相依(FD)的表示方式：
  - 假設有一個資料表R，有三個欄位，分別為X,Y,Z，則可以用下式表示：

$$R = \{X,Y,Z\} \text{ 或 } R(X,Y,Z)$$

  - 式 $X \rightarrow Y$ ：表示X決定Y，X稱決定因素，Y稱相依因素，也就是Y相依於X，Y不能單獨存在。

資料庫正規化

- 完全功能相依：
  - 假設關聯表R(X,Y,Z)存在一組功能相依(X,Y)→Z，則移除X或Y會造成Z存在沒有意義。
  - 例如：

學號	課號	成績
X	Y	Z

  - 不管移除學號(X)或課號(Y)，都將使成績(Z)變得沒有意義(因為不知道是誰的成績或哪門課的成績)。
  - 亦即成績(Z)完全功能相依於{學號(X)，課號(Y)}

資料庫正規化

- 部分功能相依：
  - 假設關聯表R(X,Y,Z)存在一組功能相依(X,Y)→Z，則移除X或Y時，Z仍能存在，此時稱Z部分功能相依於(X,Y)。
  - 例如：

學號	身分證號	姓名
X	Y	Z

  - 不管單獨移除學號(X)或身分證號(Y)，姓名(Z)都可以繼續存在(因為只要知道學號或身分證號，姓名就有意義)。
  - 亦即姓名(Z)部分功能相依於{學號(X)，身分證號(Y)}

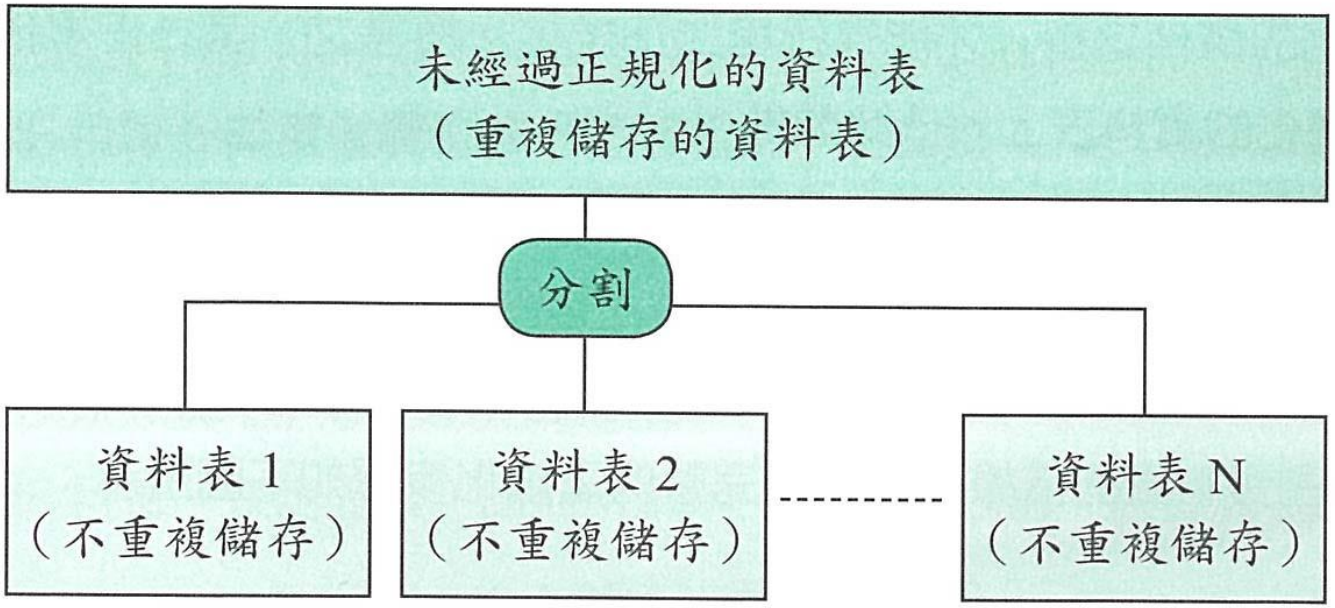
資料庫正規化

- 遞移相依：
  - 是指在兩個欄位間並非直接相依，而是借助第三個欄位來達成資料相依的關係。
  - 例如關聯表R(X,Y,Z)，包含一組相依X→Y，Y→Z，則X→Z成立，此時我們稱Z遞移相依於X。

  - 因為課程代號可以決定老師編號，並且老師編號又可以決定老師姓名，所以課程代號與老師姓名之間存在遞移相依。

資料庫正規化

- 資料庫正規化(Normalization)：
  - 是指將原先關聯(表格)的所有資訊，在分解之後，仍能由數個新關聯(表格)中經過合併得到相同的資訊。
  - 正規化就是對一個「非正規化」的原始資料表格，進行一連串的分割，分割成數個不重複的資料表格。

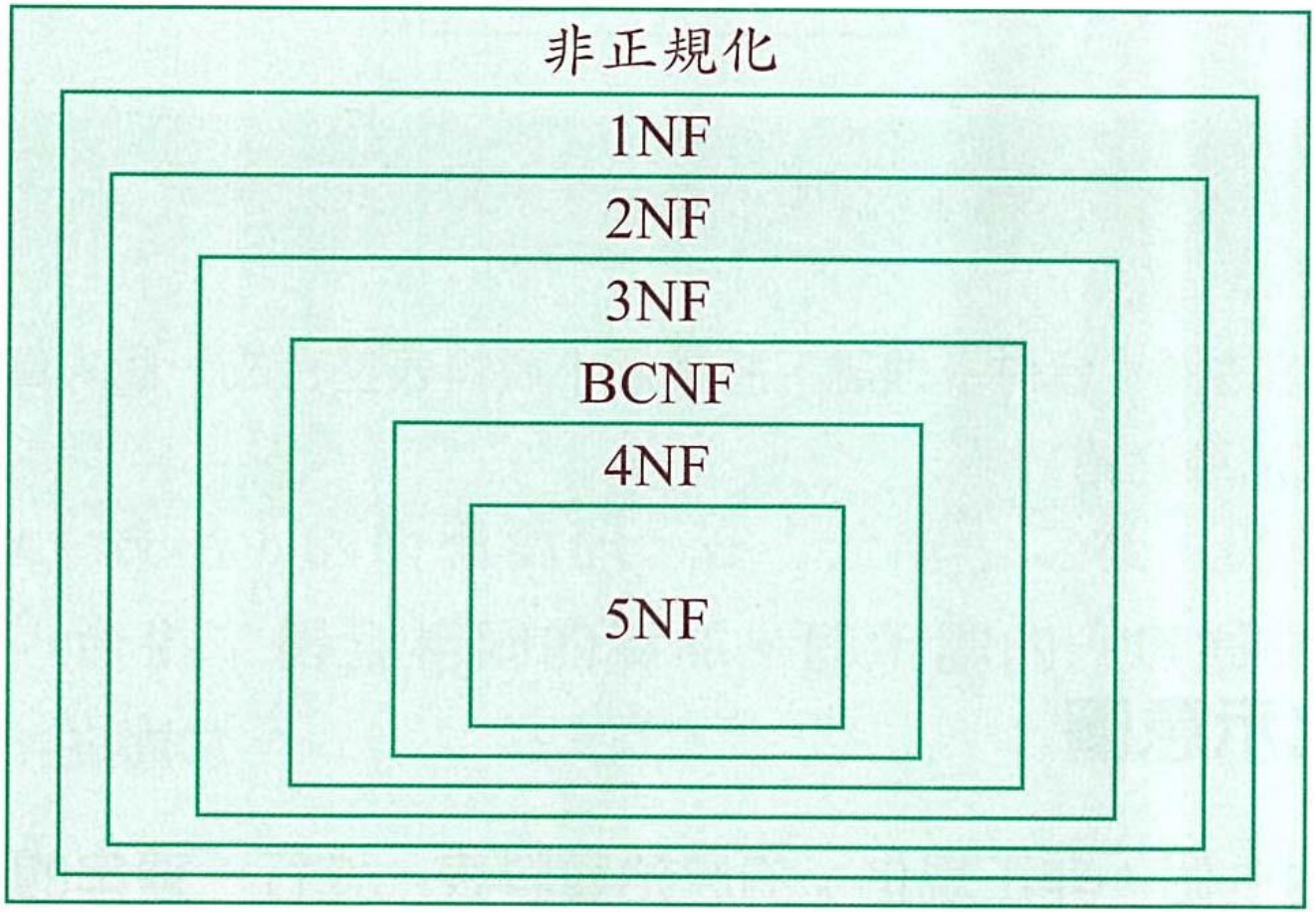


資料庫正規化

- 正規化的原則：
  - 資料庫在正規化時會有一些規則，每條規則都稱為「正規形式」。如果符合第一條規則，則資料庫就稱為「第一正規化形式(1NF)」，如果符合第二條規則，則資料庫就稱為「第二正規化形式(2NF)」，依此類推。
  - 最多可以進行到第五正規化形式，但一般做到BCNF就可以了。

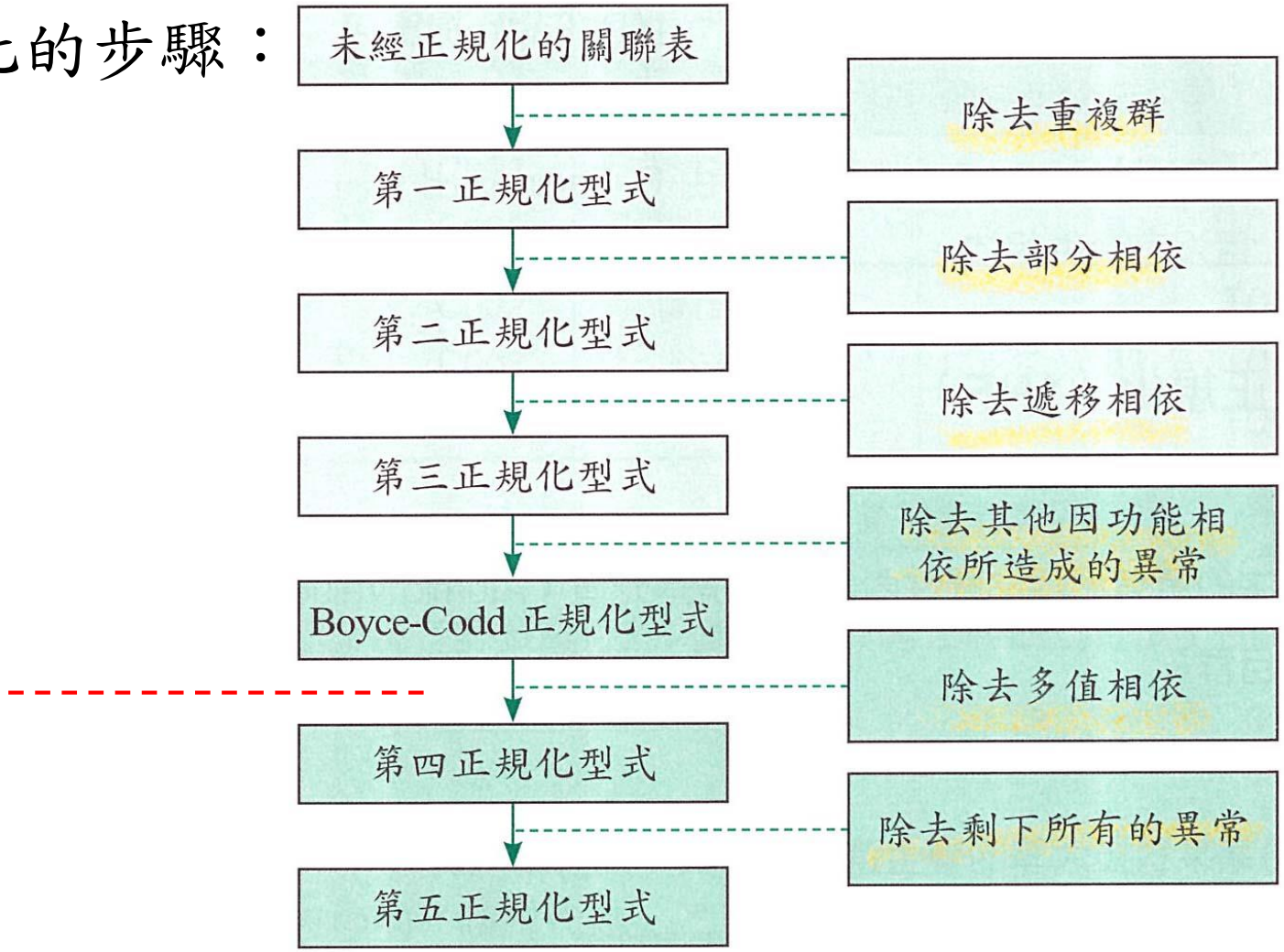
資料庫正規化

- 正規化的原則：



資料庫正規化

- 正規化的步驟：

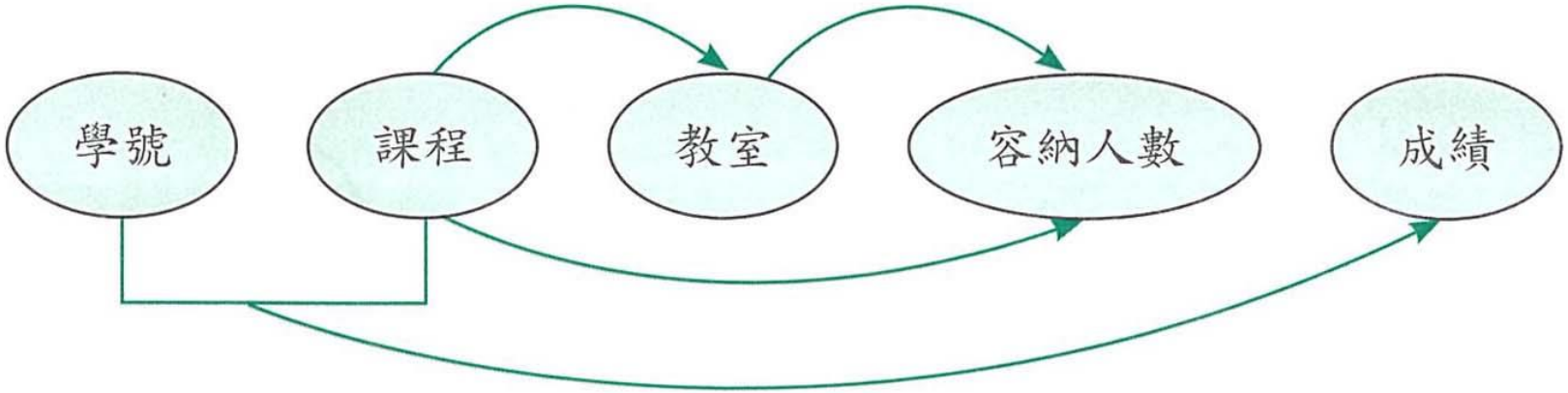


資料庫正規化

- 正規化步驟：
  - 第一正規化：排除所有重複的項目。
  - 第二正規化：符合1NF且每一非鍵值欄位完全功能相依於主鍵，即不可有欄位部分功能相依於主鍵。
  - 第三正規化：符合2NF且每一非鍵值欄位非遞移相依於主鍵，即去除遞移相依問題。
  - **BCNF正規化**：(Boyce-Codd Normal Form;BCNF)，符合2NF且每一決定因素都是候選鍵。
  - **第四正規化**：符合BCNF，再除去所有的多值相依。
  - **第五正規化**：符合4NF且沒有合併相依。

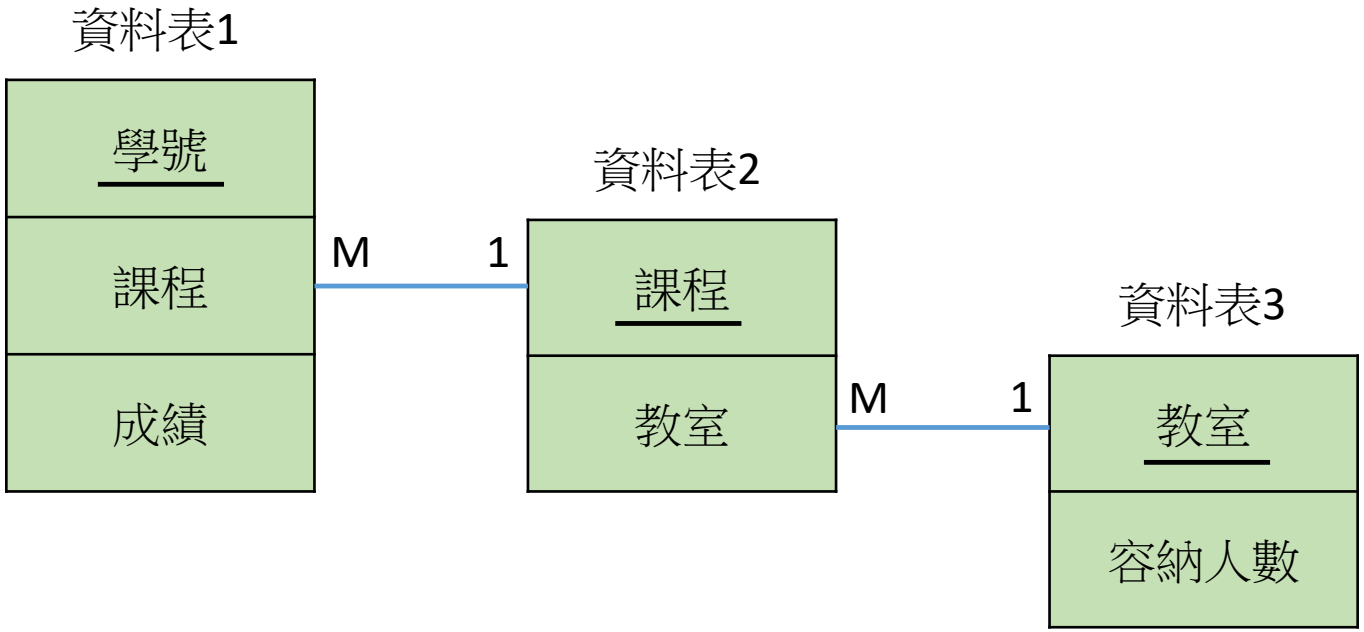
資料庫正規化

- 練習一：
  - 請將下列的1NF關係正規化為3NF。



資料庫正規化

- 練習一解答參考：



資料庫正規化

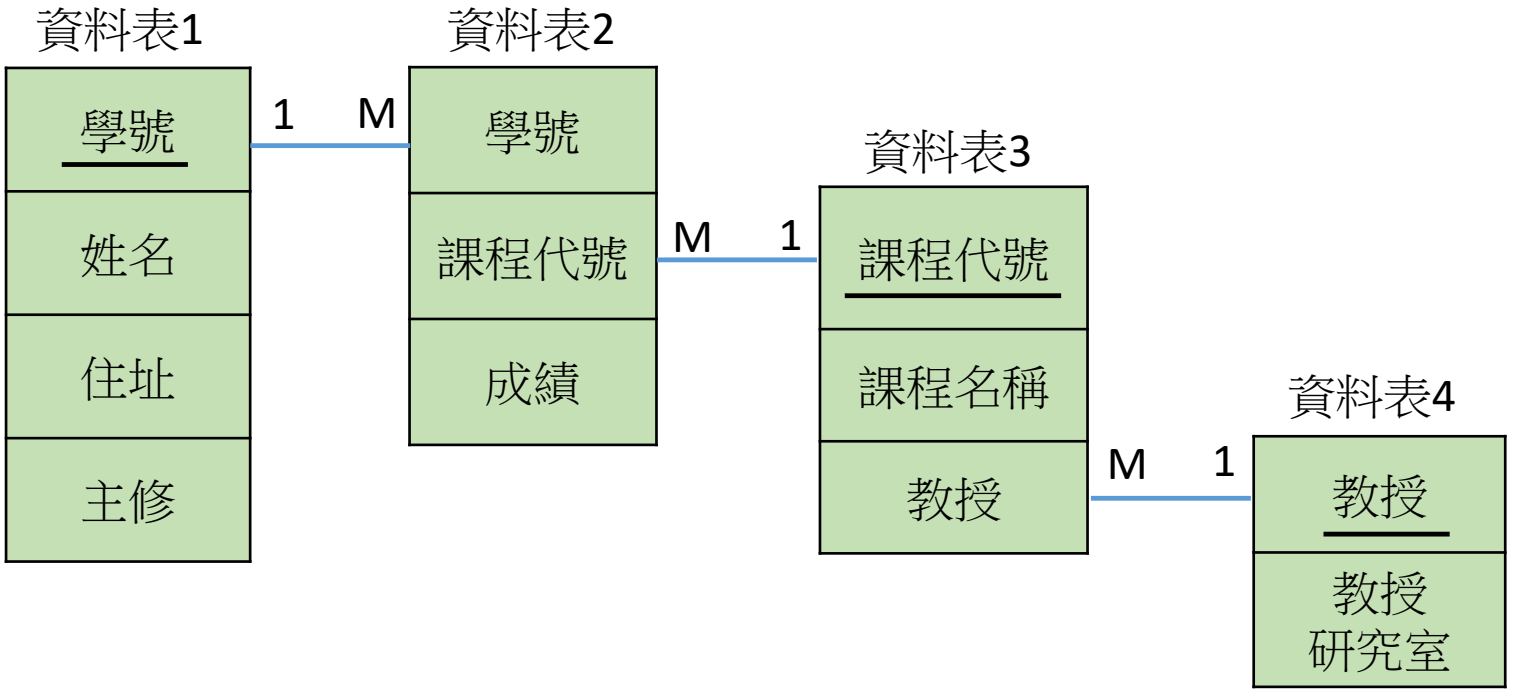
- 練習二：
  - 將下列使用者觀點的資料正規化為3NF。

姓名：李春雄　　主修：資管所
學號：D9309105
住址：台北市文山區基隆路

課程代號	課程名稱	教授	教授研究室	成績
C001	高等資料庫	李教授	IB301	90
C004	高等演算法	王教授	IB403	88
C006	分散式系統	陳教授	IB501	85

資料庫正規化

- 練習二解答參考：



休息一下~

常用資料庫介紹

- Microsoft Access
  - 微軟的單機版簡易資料庫
- MySQL / MariaDB
  - 廣受歡迎的免費資料庫(我們要用這個來上課)
- MS SQL Server
  - 微軟的要錢的企業用SQL資料庫
- SQLite
  - 開放原始碼的簡易資料庫
- Oracle
  - 有名的甲骨文公司的老牌資料庫

MySQL / MariaDB

- MySQL資料庫是網路上最受歡迎的免費開源資料庫，性能高、可靠性佳，可以跟一般市售的資料庫媲美。
- 因為太好用了，在2009年被Oracle公司收購，並提高了商用版的價格，雖然仍提供免費版本，但大家有些擔心，怕Oracle公司改變政策。
- MySQL的創始人麥克爾·維德紐斯以MySQL為基礎，成立分支計劃MariaDB，並與MySQL完全相容。
- 維基百科也是用MySQL系統喔，並於2013年遷移至MariaDB。

MySQL / MariaDB

- Maria是MySQL創作者麥克爾·維德紐斯(芬蘭人)的女兒，所以他的新系統就叫MariaDB了。
- 之後課程不管使用MySQL或MariaDB，我們統稱MySQL。

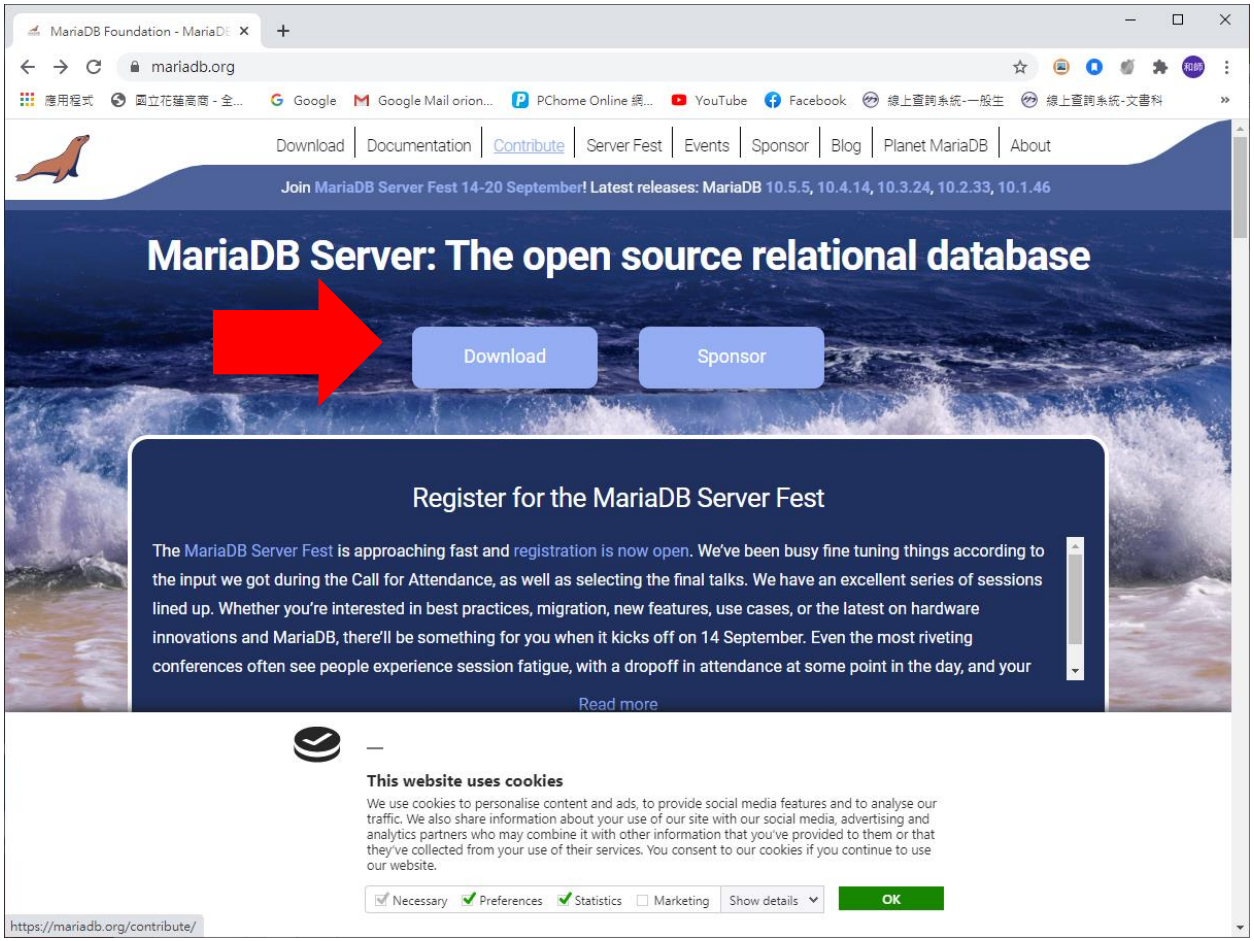
MySQL/MariaDB安裝

- 第一種方式：單獨下載MySQL/MariaDB，只安裝資料庫。
- 網址：https://dev.mysql.com/downloads/installer/
- 網址：https://mariadb.org/

- (網頁每年可能會長的不一樣)

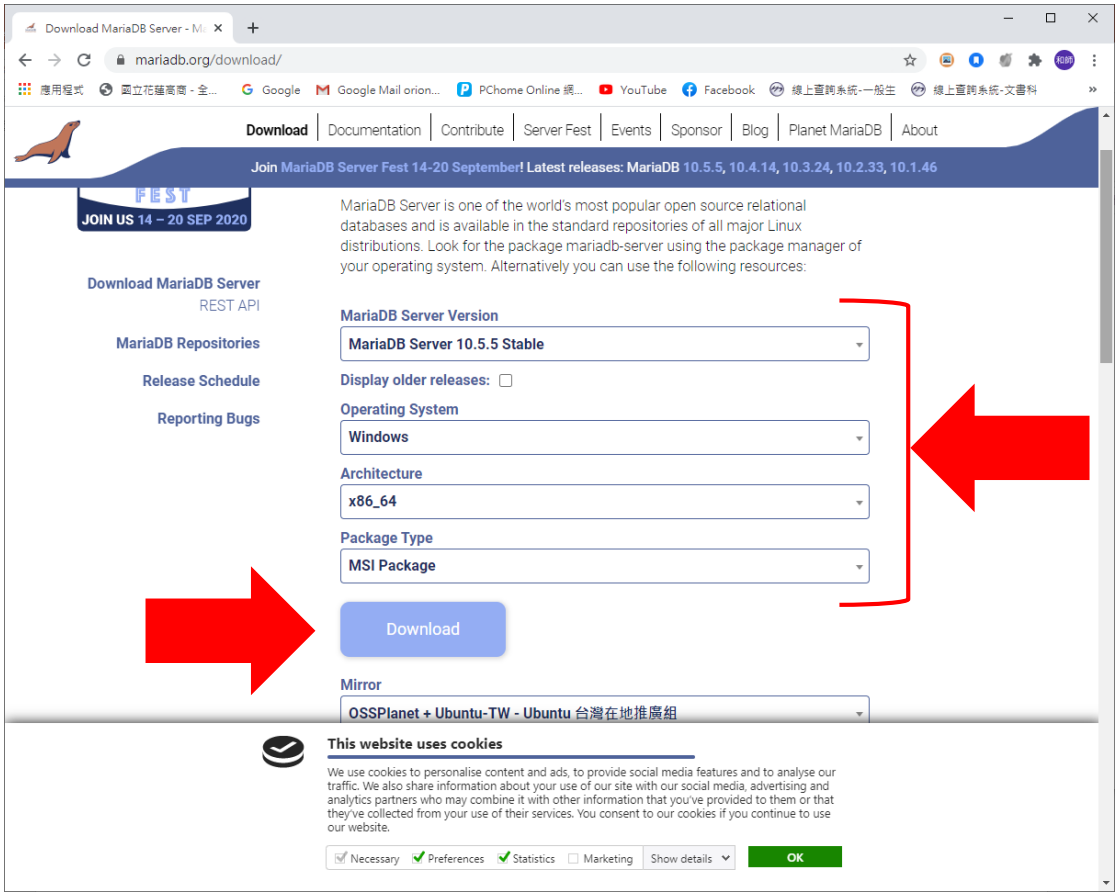
MySQL/MariaDB安裝

- 連線到MariaDB官網，點選下載



MySQL/MariaDB安裝

- 雖然會自動偵測你的系統，列出適合你的，但還是要檢查一下。

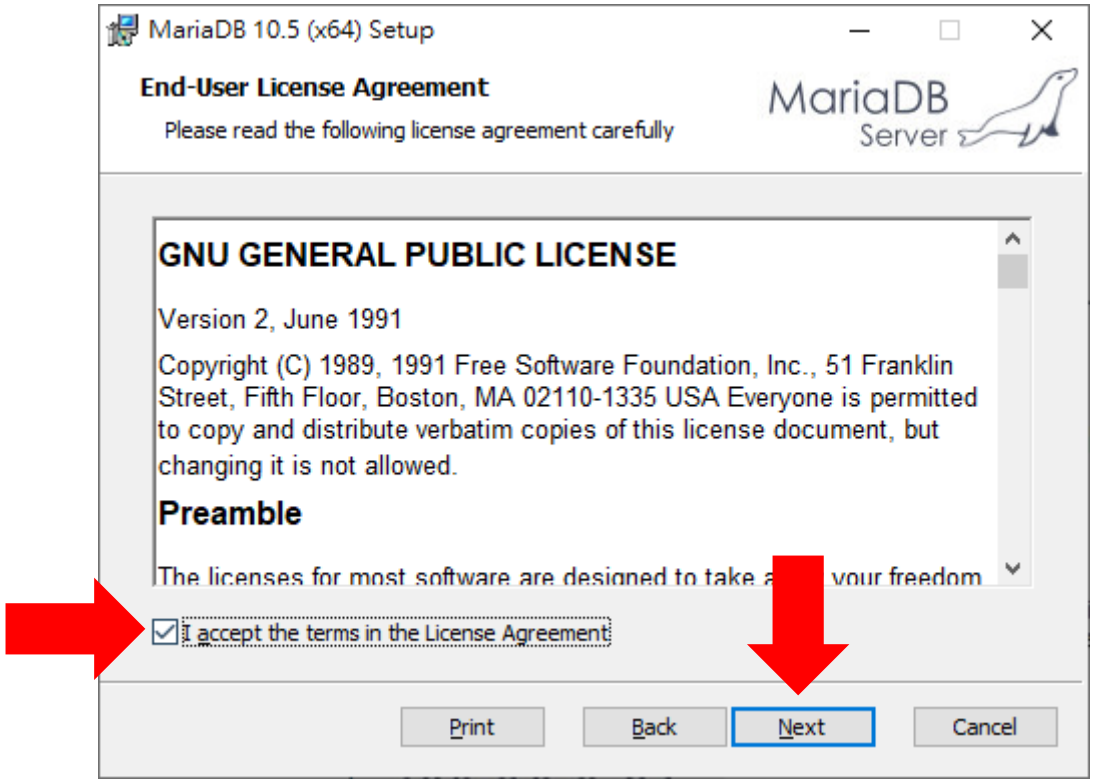


MySQL/MariaDB安裝

- 開始安裝~

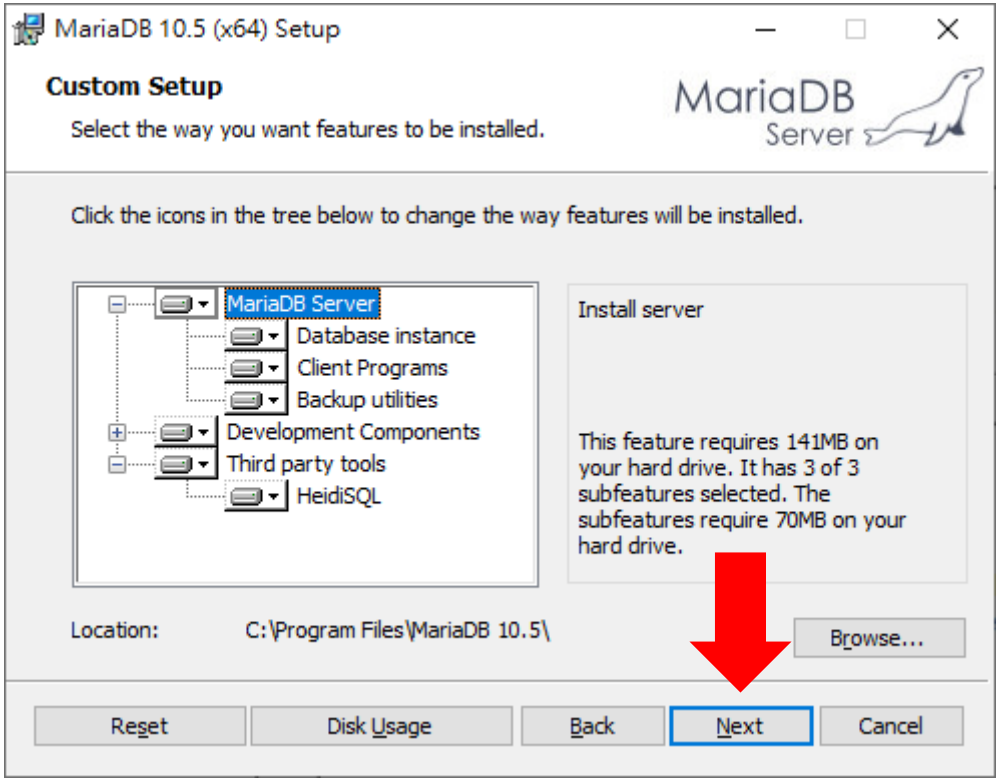
MySQL/MariaDB安裝

- 版權說明，打勾接受



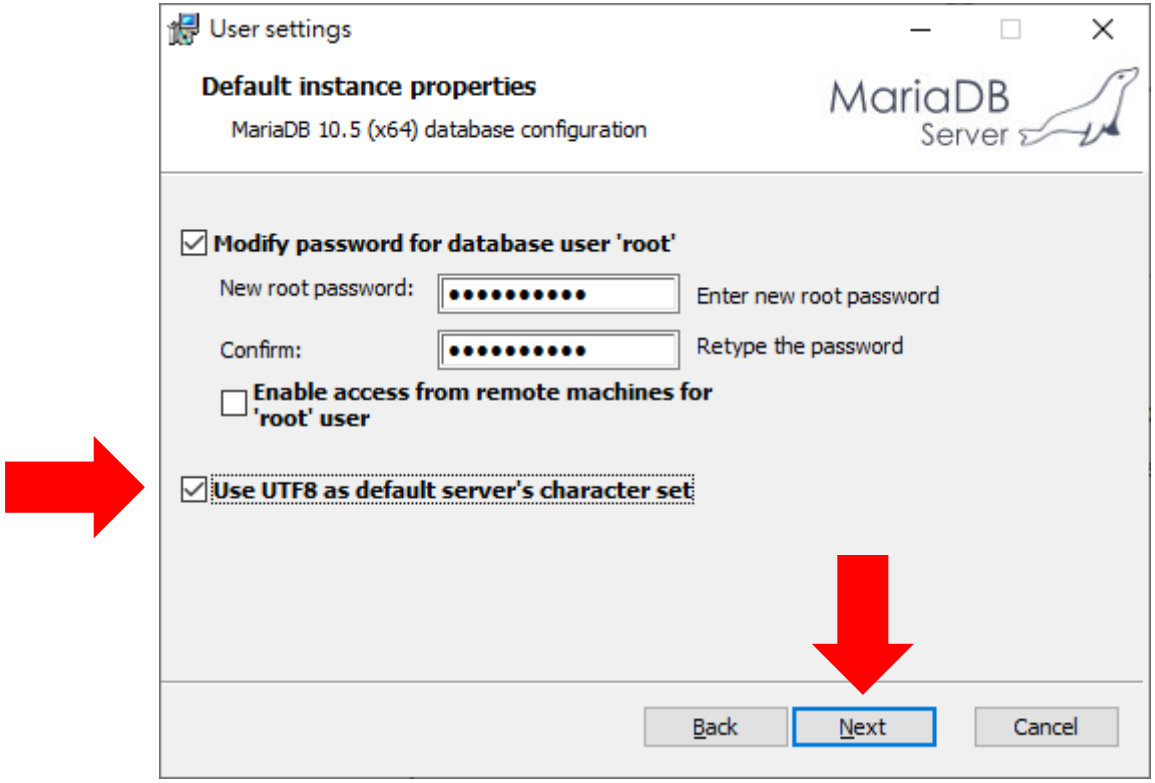
MySQL/MariaDB安裝

- 要安裝那些東西呢?照它內定的就好



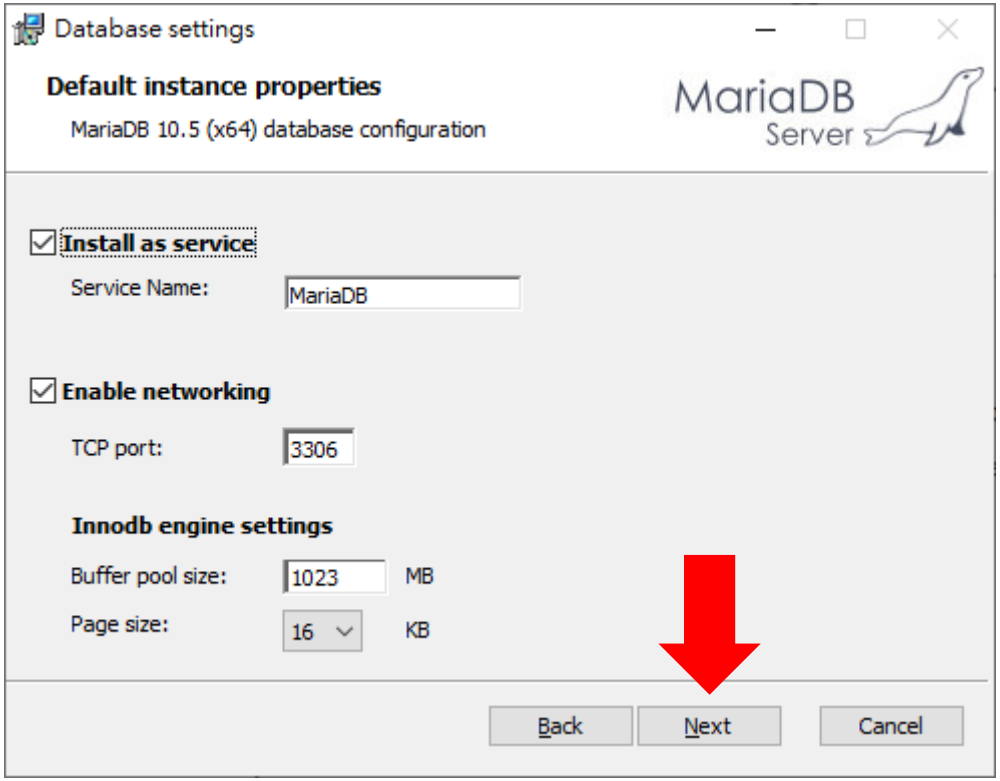
MySQL/MariaDB安裝

- 輸入最高權限管理者root的密碼，然後看你要不要設定UTF8為預設的編碼方式。



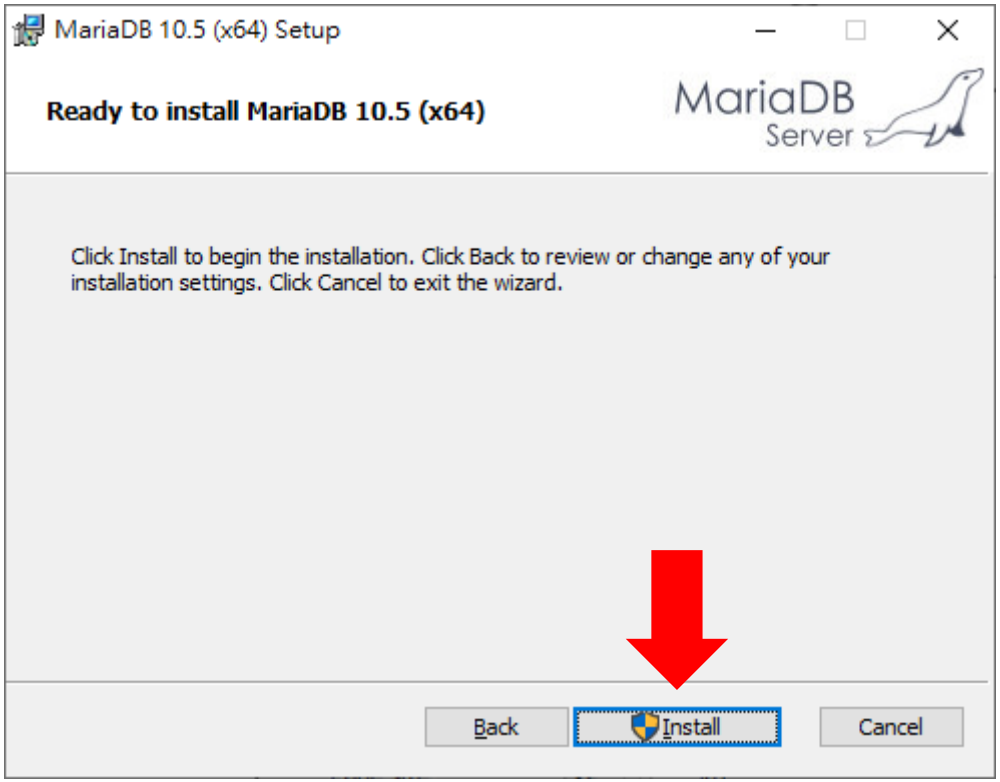
MySQL/MariaDB安裝

- 基本上伺服器名稱、埠號等都用內定值就好



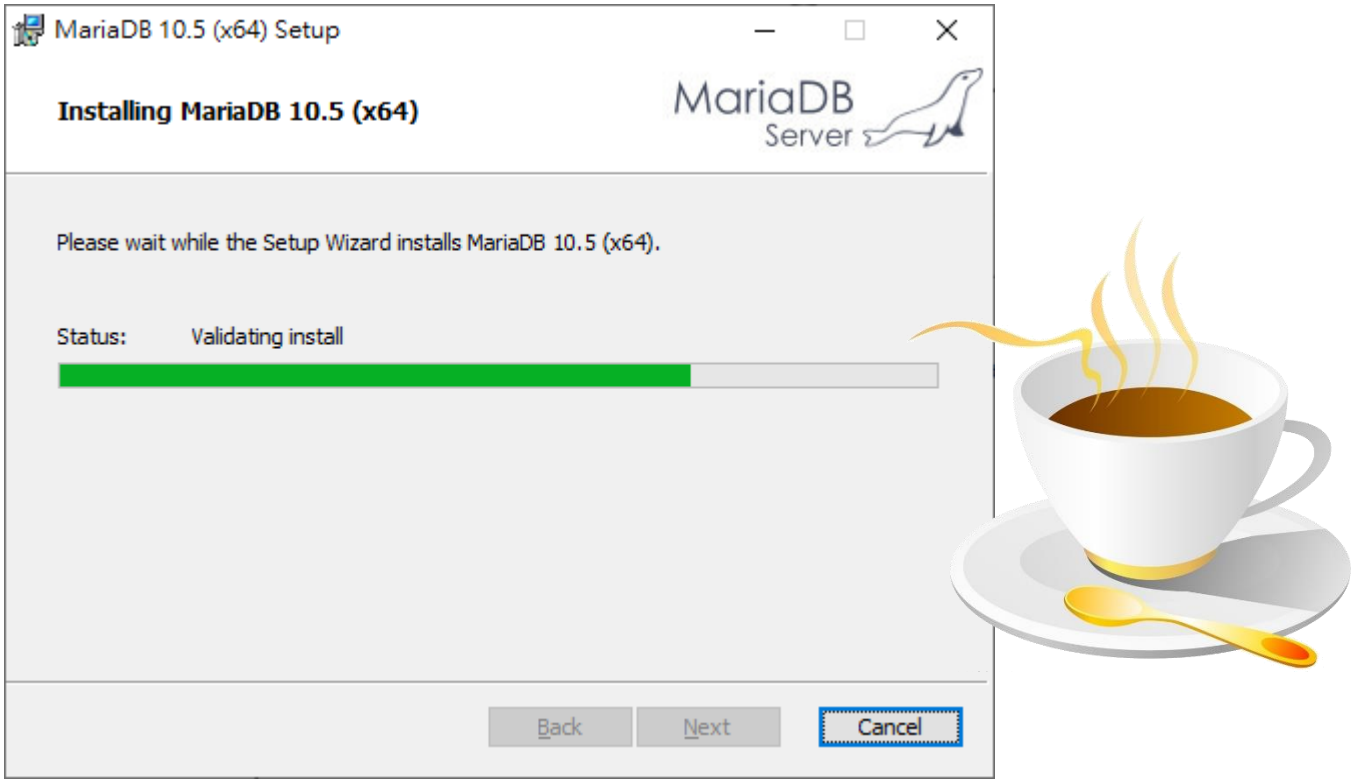
MySQL/MariaDB安裝

- 都設定好了，開始安裝



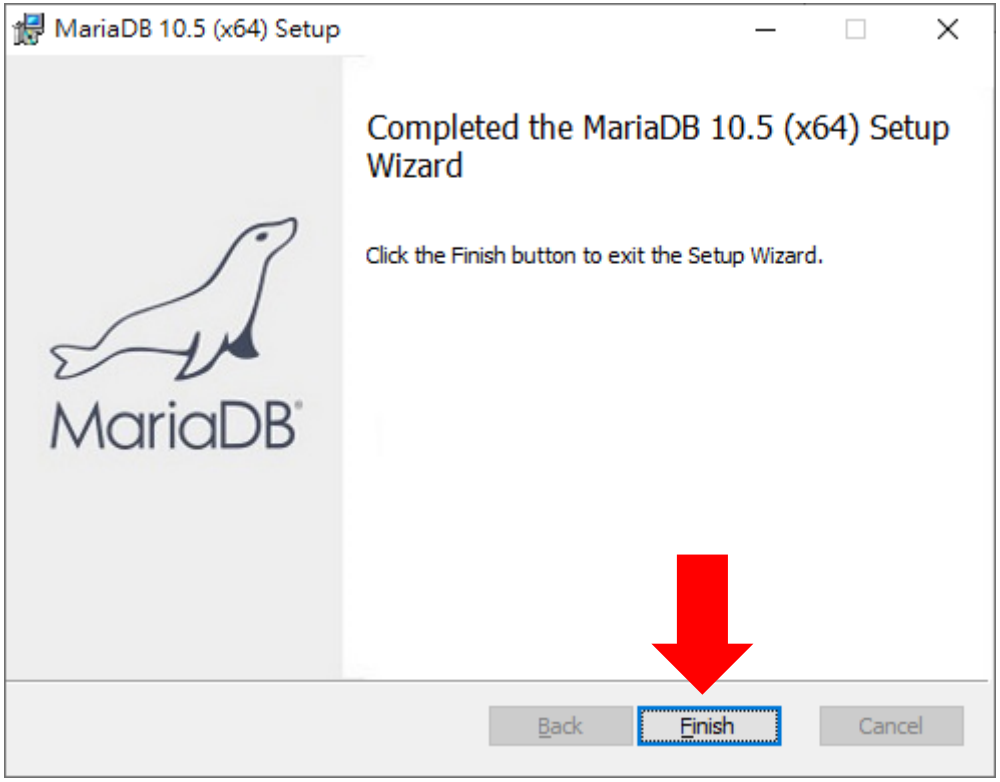
MySQL/MariaDB安裝

- 咖啡時間~



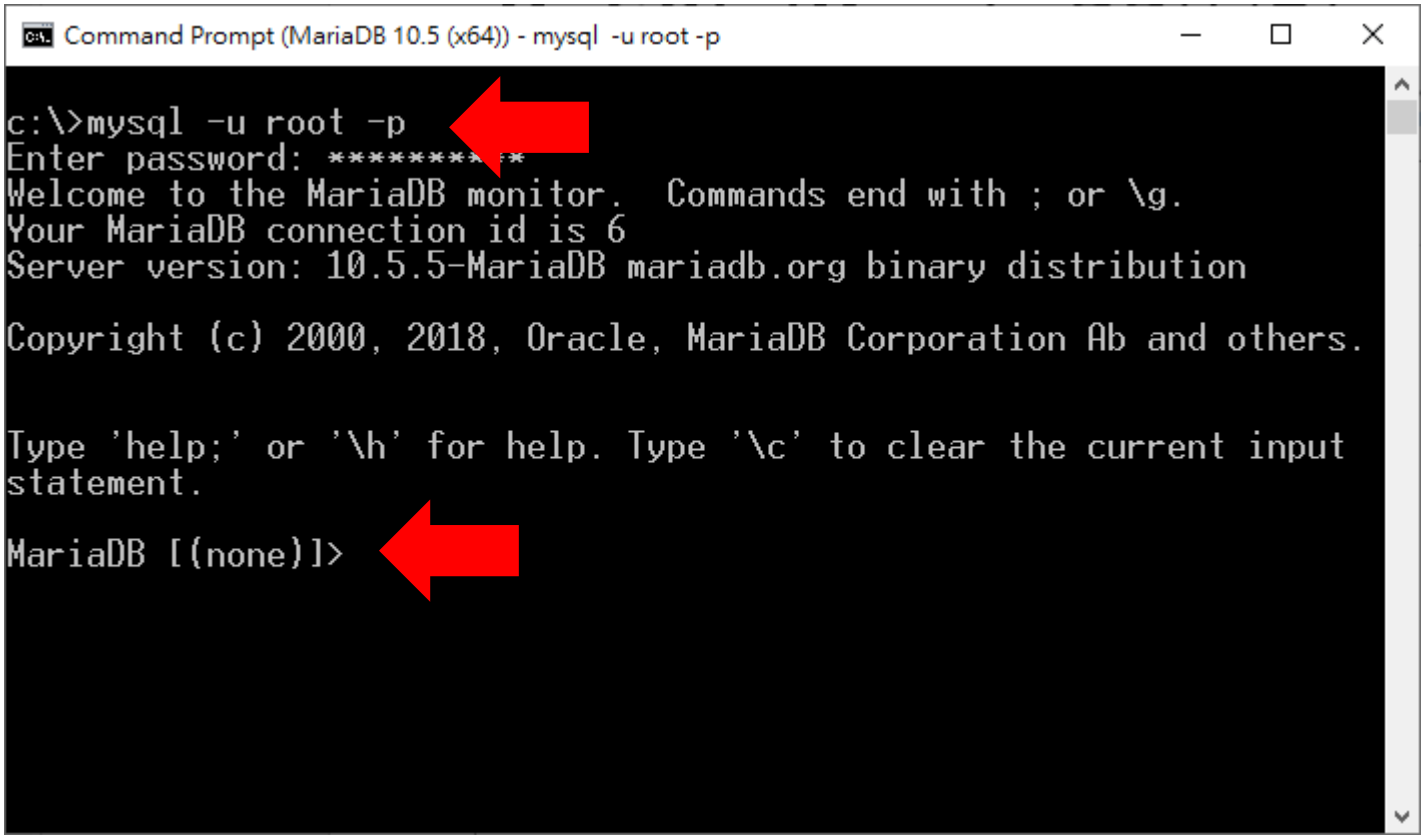
MySQL/MariaDB安裝

- 安裝完成(咖啡都還沒泡好呢)



MySQL/MariaDB安裝

- 輸入指令 `mysql -u root -p` ，啟動MariaDB看看是否運作。它其實和MySQL幾乎是一樣的。

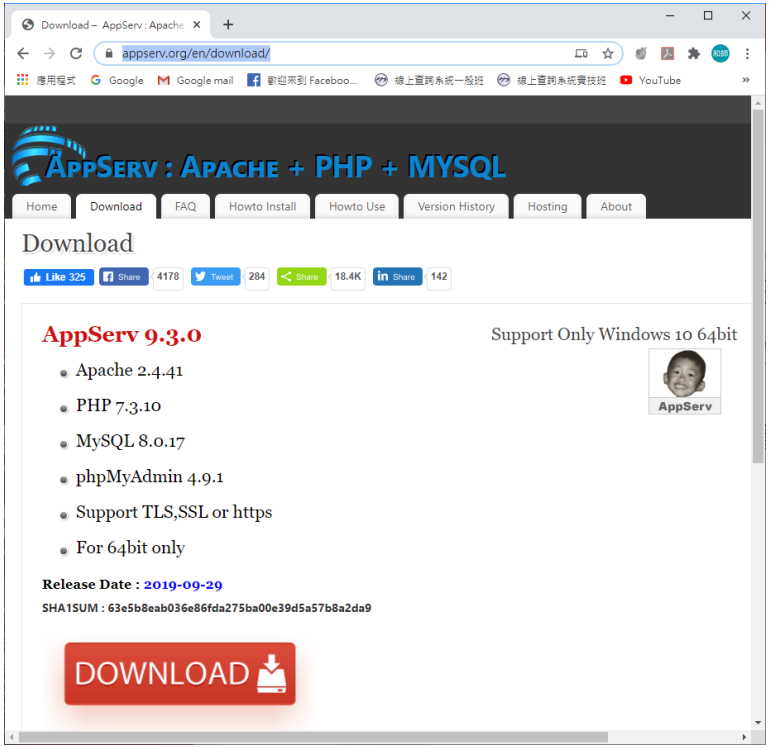


MySQL/MariaDB安裝

- 第二種方式：同時安裝Web Server+PHP+MySQL。
- 網址：https://www.appserv.org/en/download/

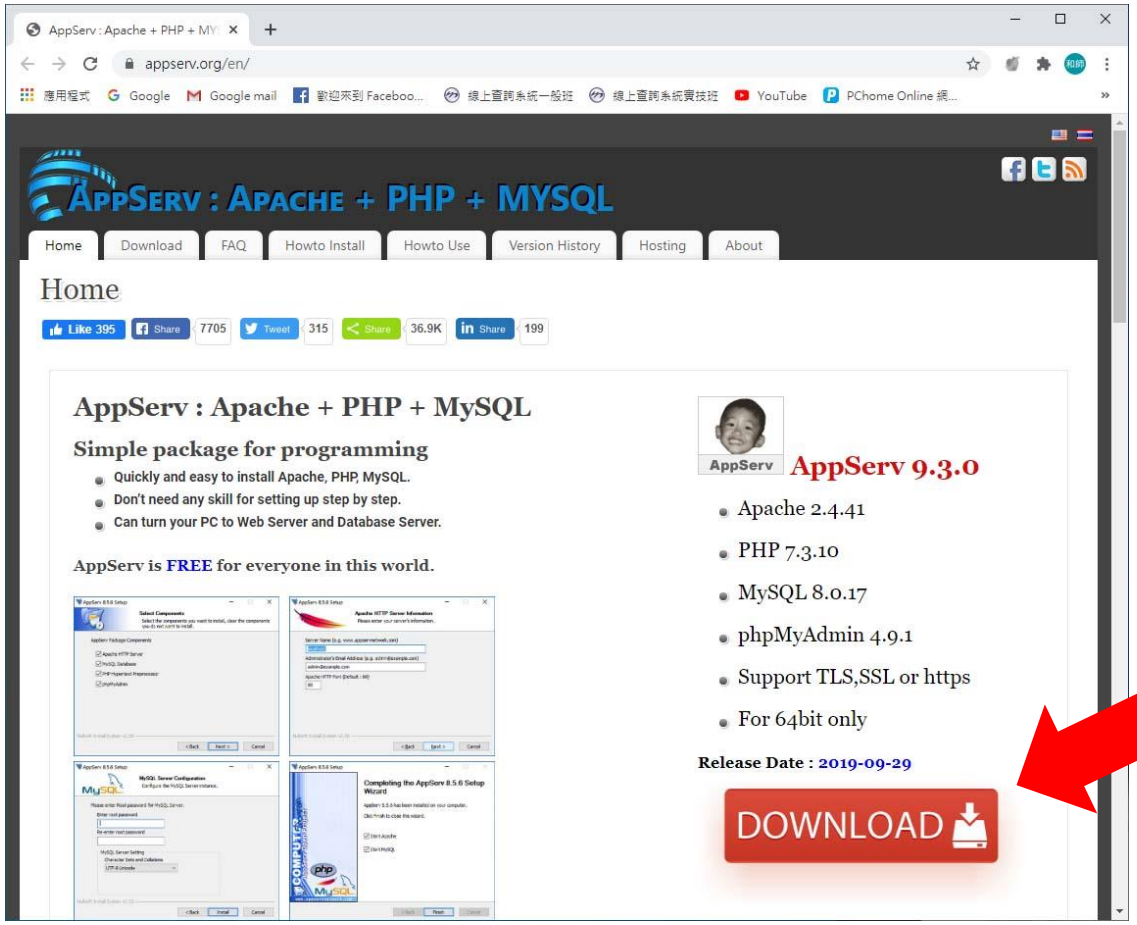
- (網頁每年可能會長的不一樣)

- 安裝畫面看操作



Appserv安裝過程

- 至https://www.appserv.org/en/下載檔案

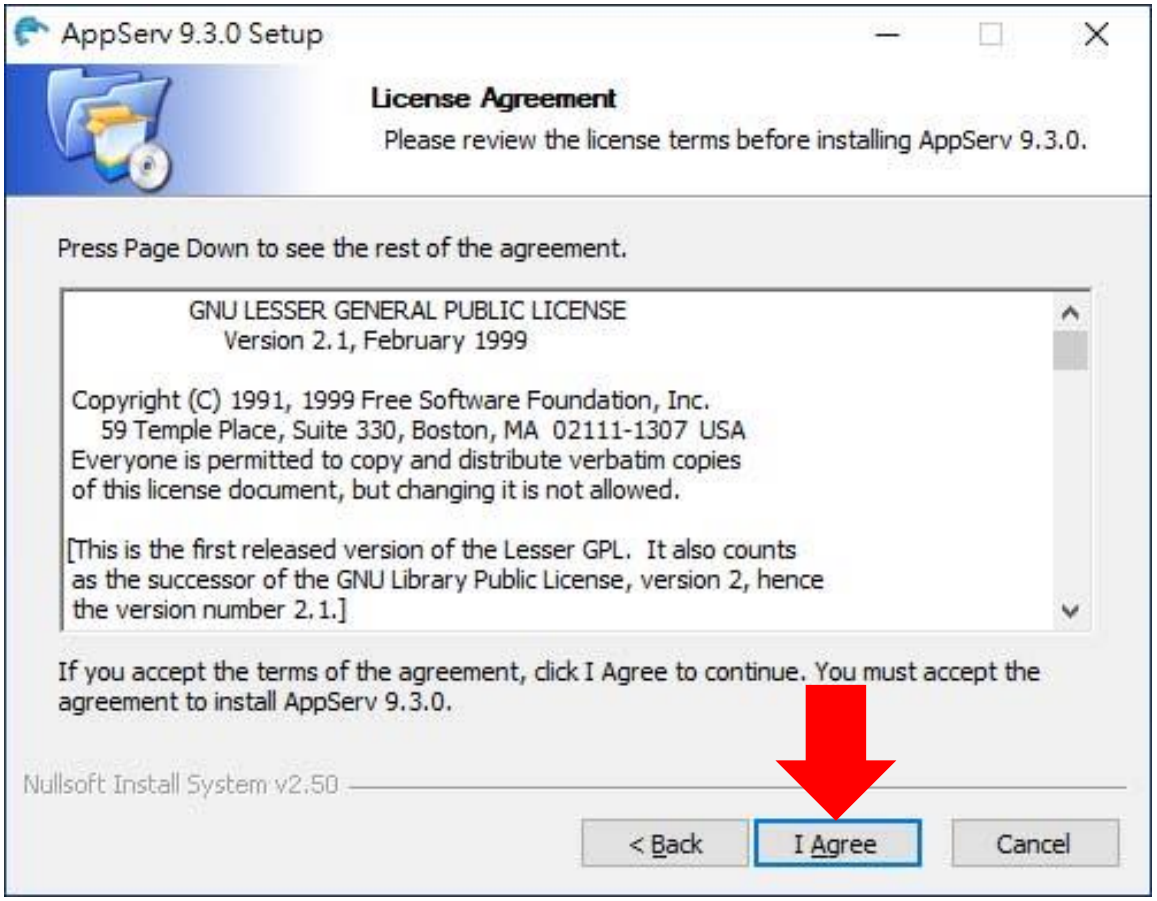


Appserv安裝過程

- 下載完成後點兩下執行安裝

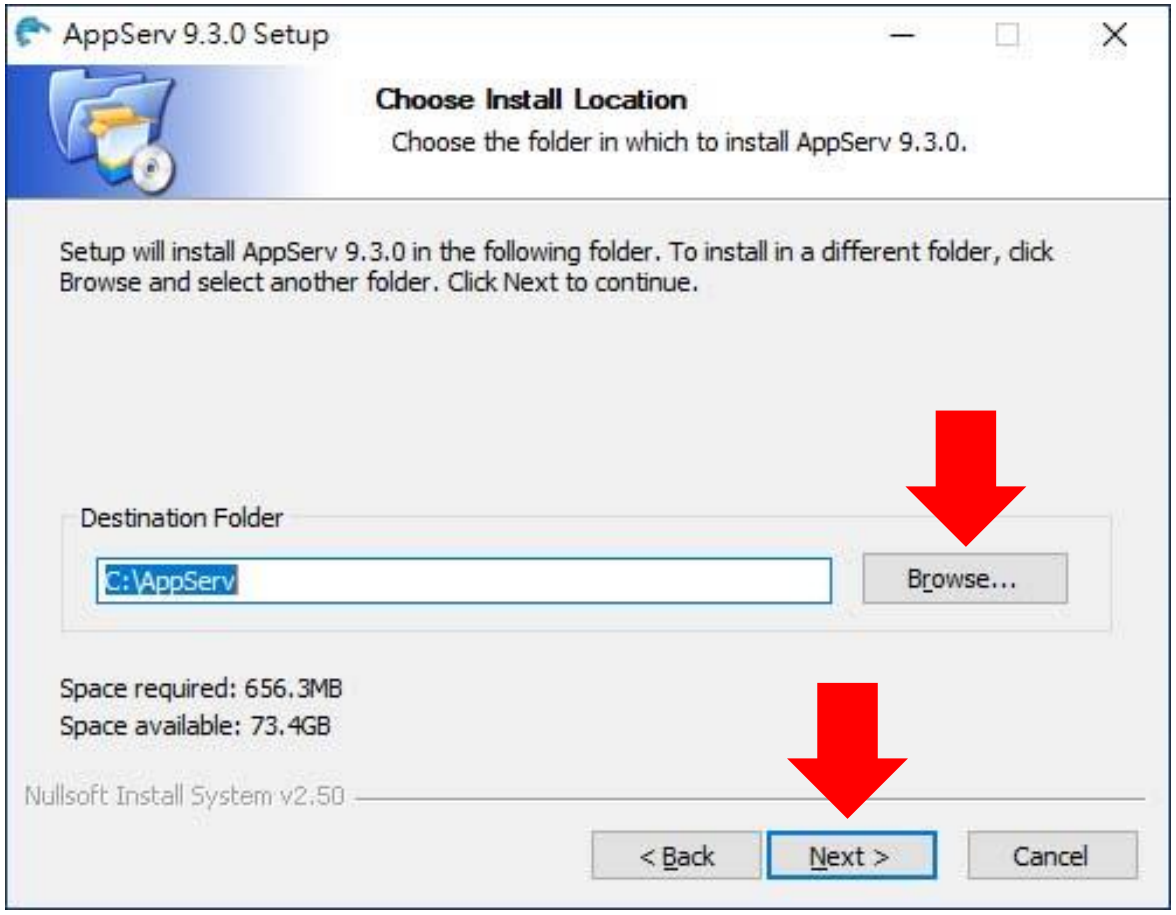
Appserv安裝過程

- 版權聲明，看看就好，反正一定要同意



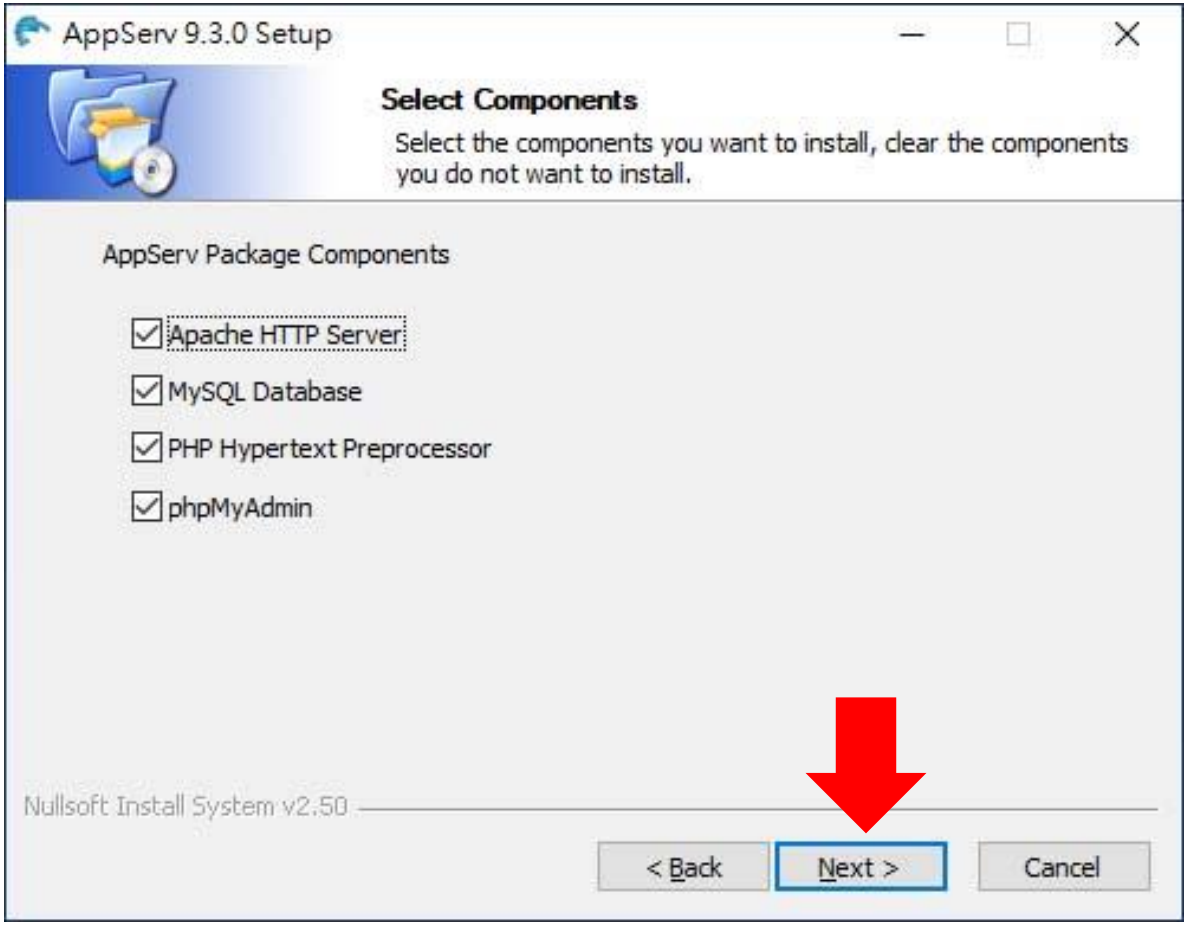
Appserv安裝過程

- 選擇要安裝在何處



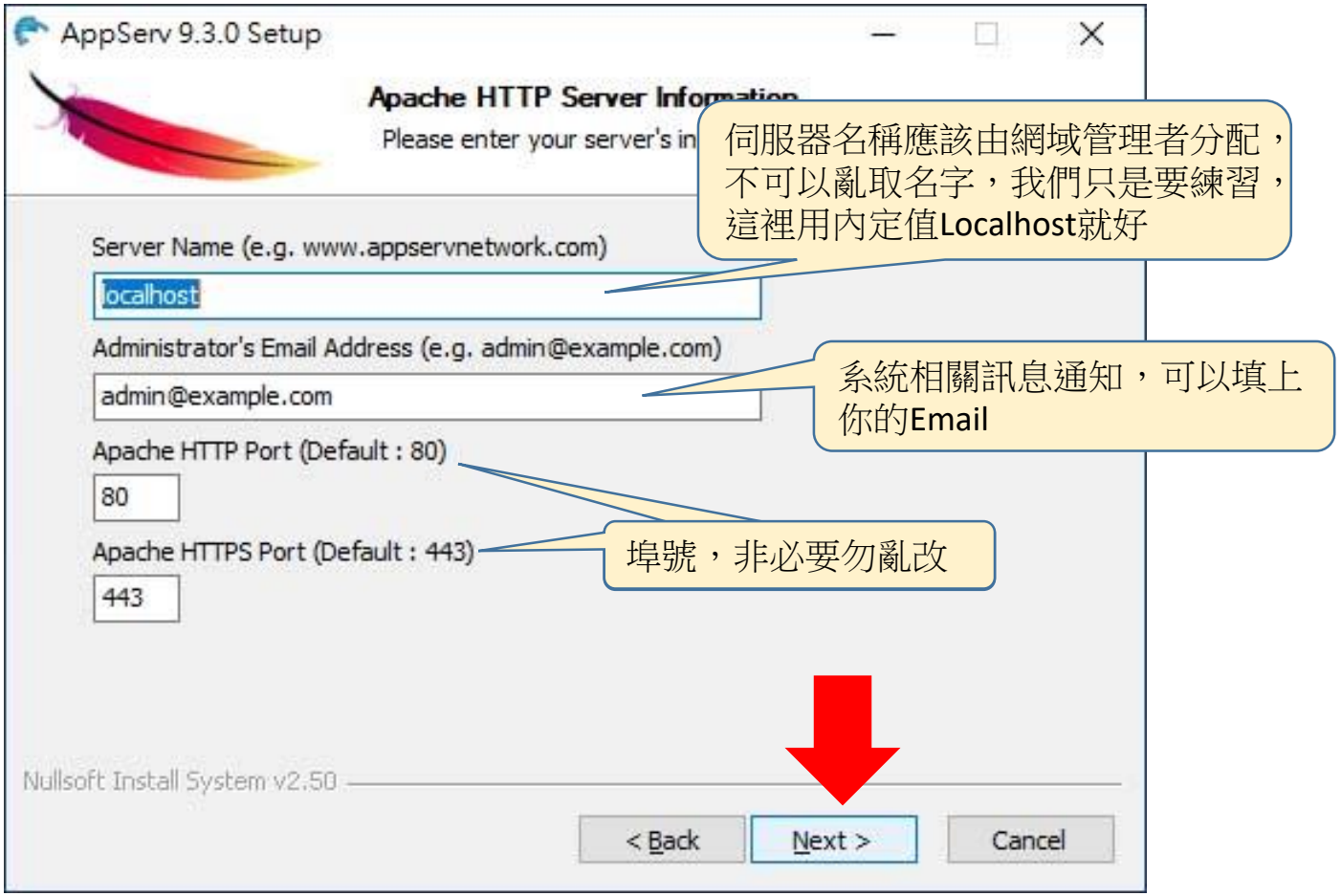
Appserv安裝過程

- 要安裝那些元件，應該都要啦



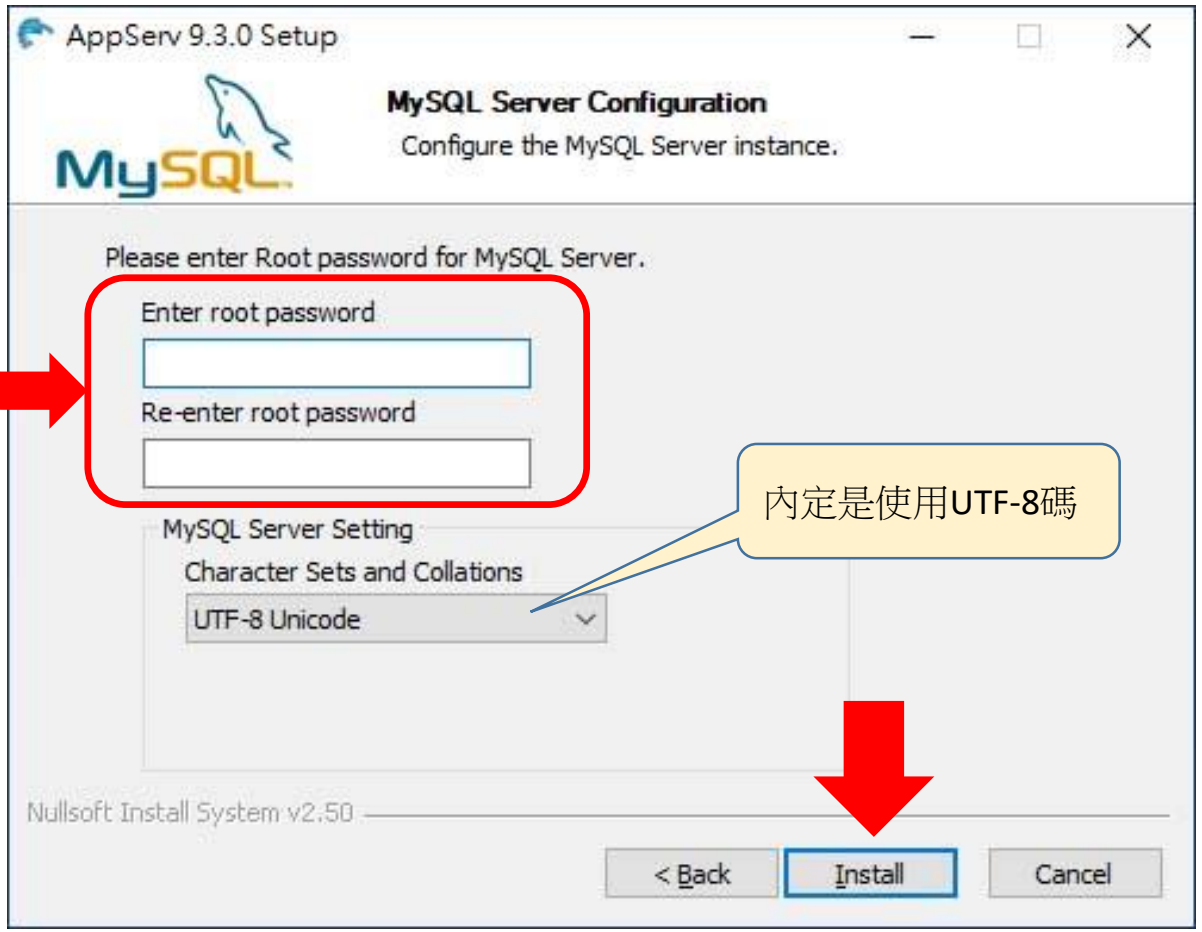
Appserv安裝過程

- 可以指定伺服器的名字、管理者Email、埠號等



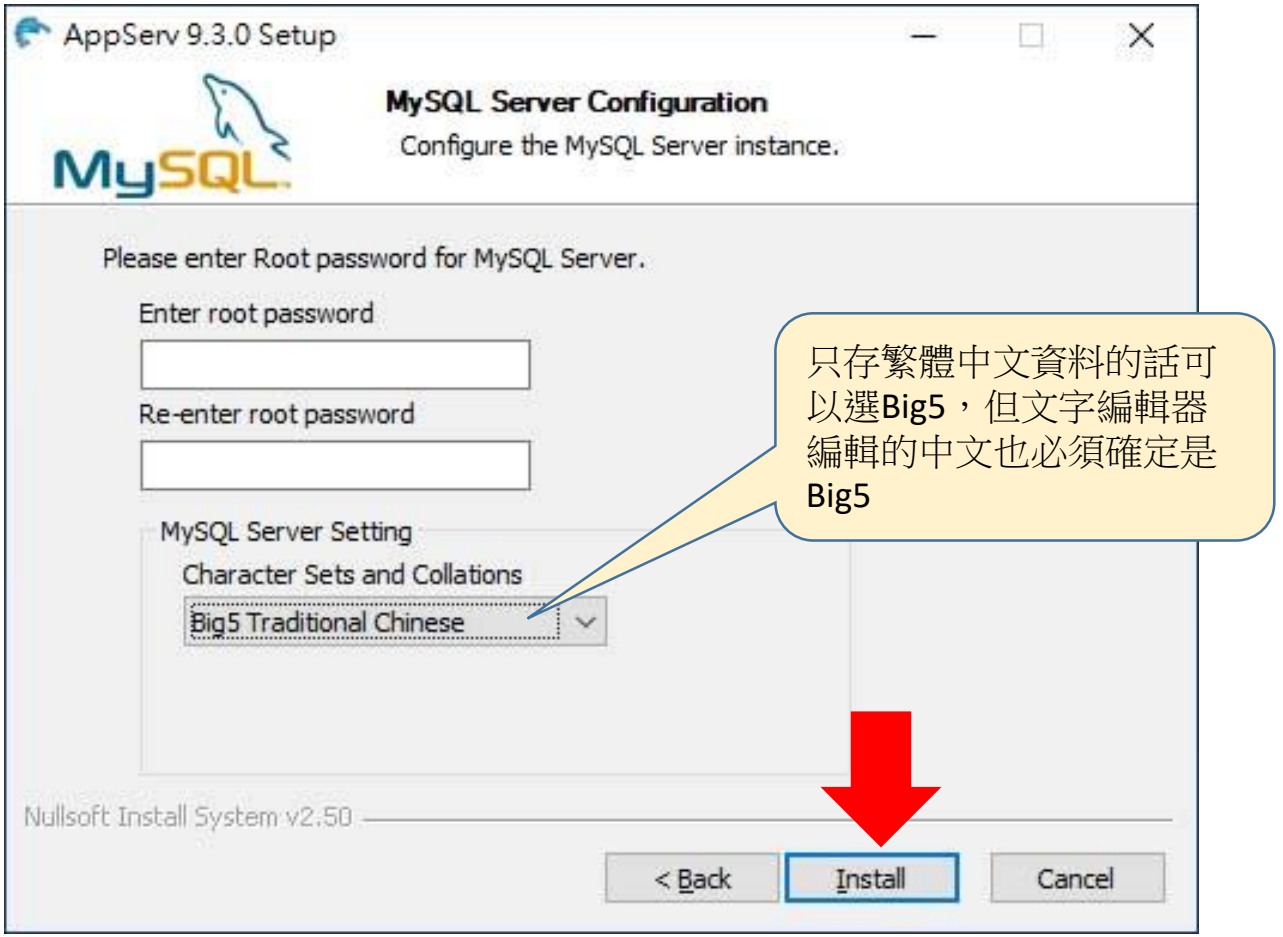
Appserv安裝過程

- 這裡很重要，要設定MySQL最高權限管理者(root)的密碼



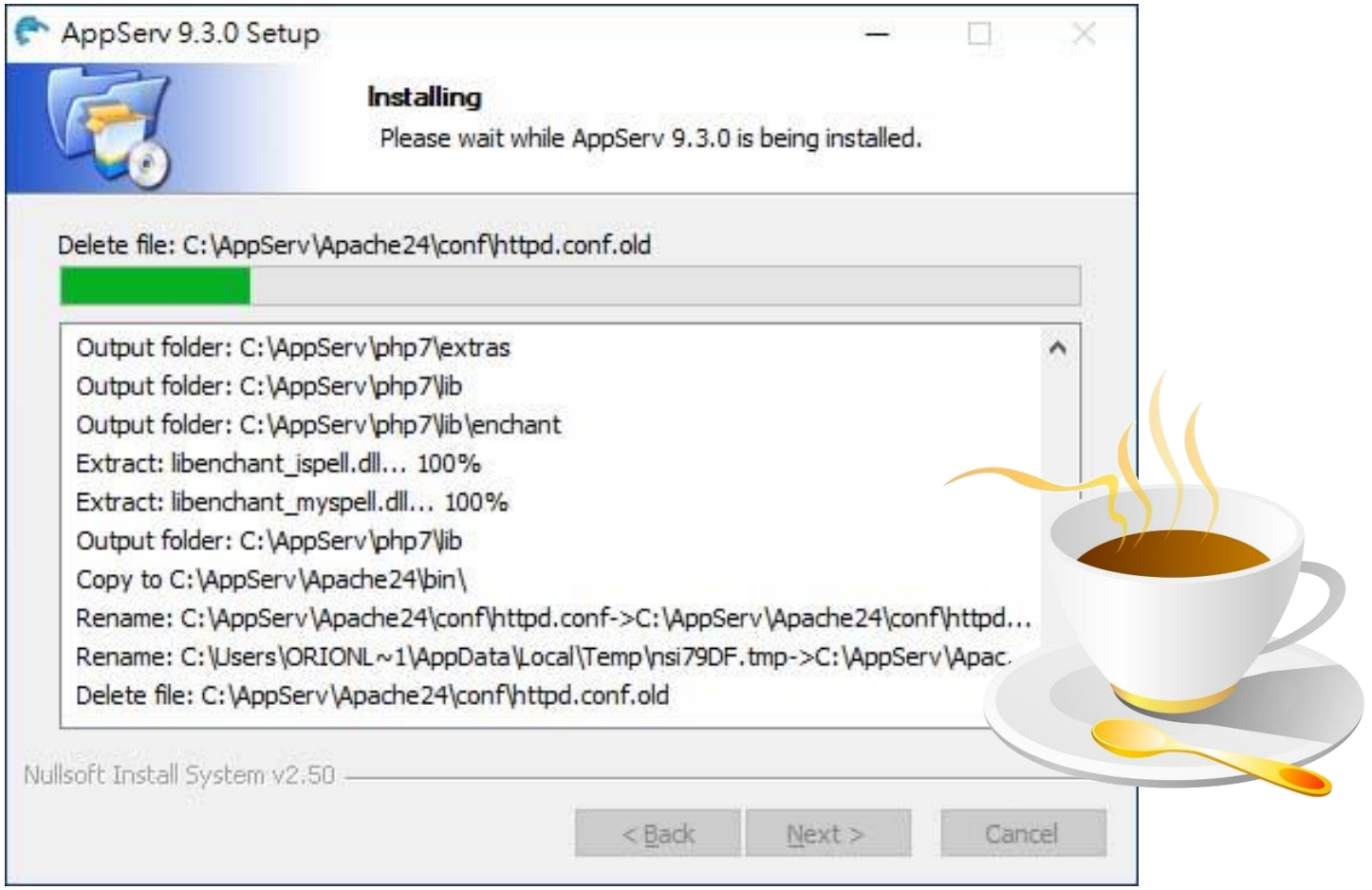
Appserv安裝過程

- 如果只會存繁體中文資料，也可以選Big5碼



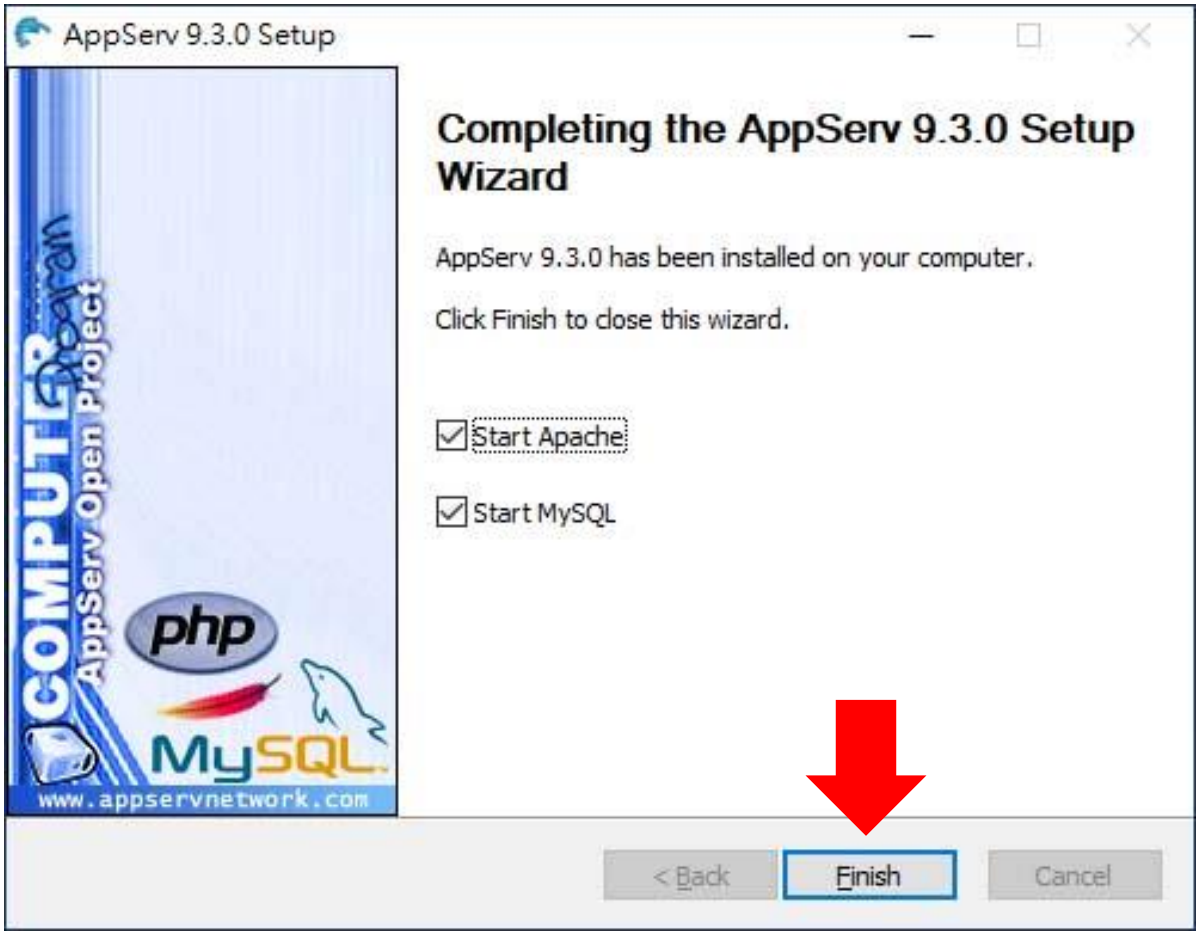
Appserv安裝過程

- 開始安裝，等它跑完，不會很久



Appserv安裝過程

- 安裝完成，之後只要開機就會自行啟動

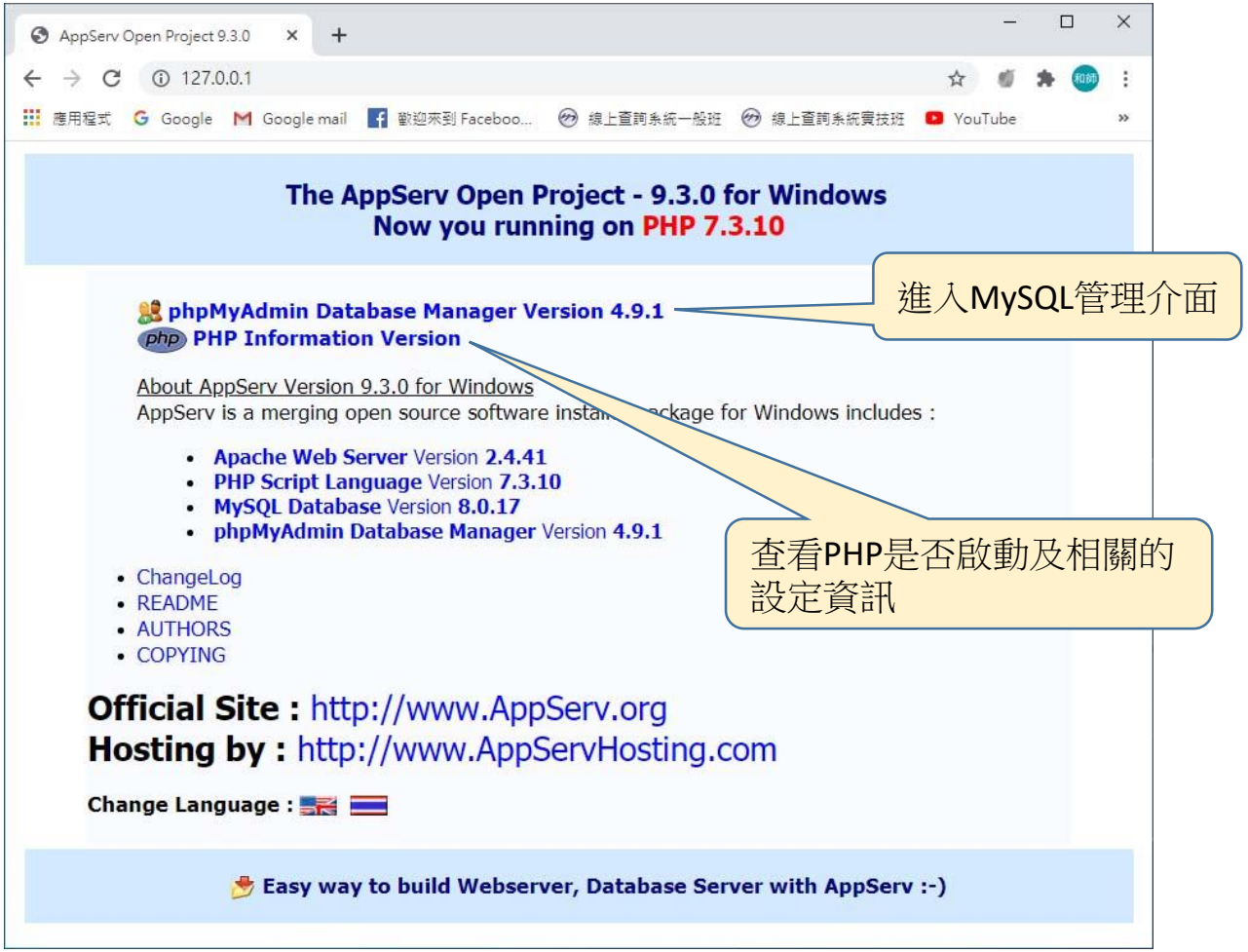


Appserv安裝過程

- 因為網頁伺服器和資料庫要讓外界存取，所以跳出Windows防火牆設定畫面

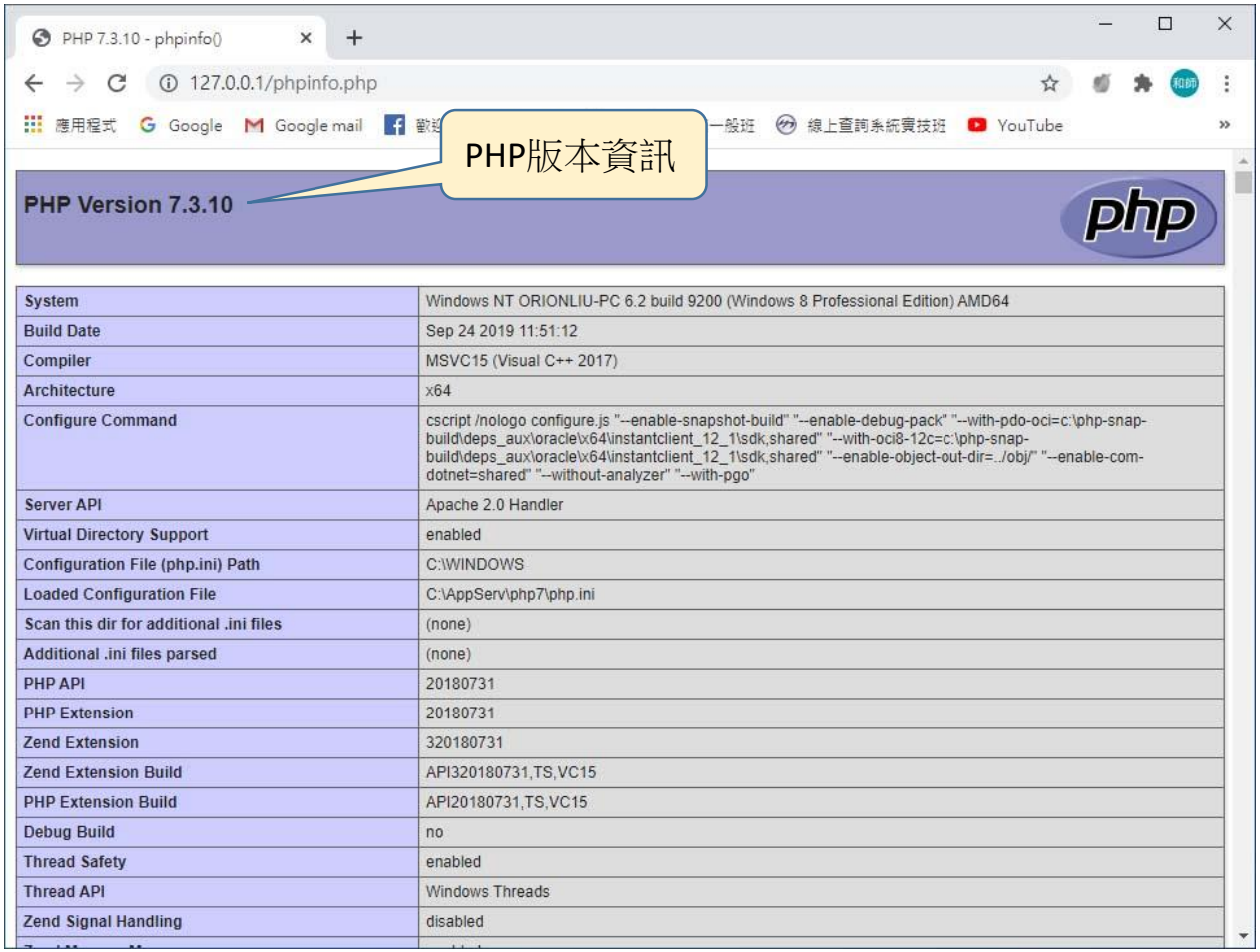
Appserv安裝過程

- 開啟瀏覽器，輸入127.0.0.1或localhost測試



Appserv安裝過程

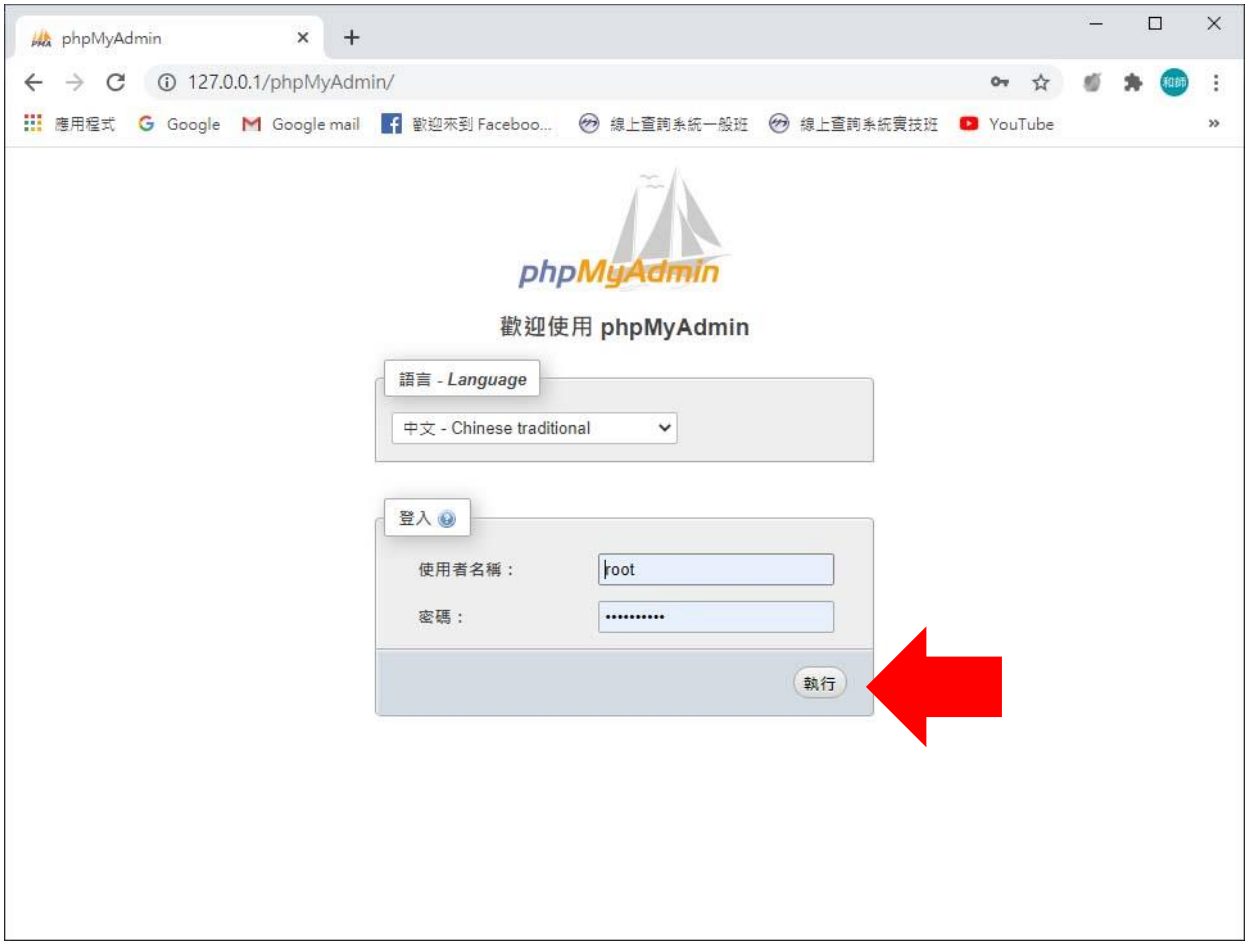
- PHP若啟動正常應看到此畫面



System	Windows NT ORIONLIU-PC 6.2 build 9200 (Windows 8 Professional Edition) AMD64
Build Date	Sep 24 2019 11:51:12
Compiler	MSVC15 (Visual C++ 2017)
Architecture	x64
Configure Command	cscript /nologo configure.js "--enable-snapshot-build" "--enable-debug-pack" "--with-pdo-oci=c:\php-snap-build\deps_aux\oracle\x64\instantclient_12_1\sdk,shared" "--with-oci8-12c=c:\php-snap-build\deps_aux\oracle\x64\instantclient_12_1\sdk,shared" "--enable-object-out-dir=../obj/" "--enable-com-dotnet=shared" "--without-analyzer" "--with-pgo"
Server API	Apache 2.0 Handler
Virtual Directory Support	enabled
Configuration File (php.ini) Path	C:\WINDOWS
Loaded Configuration File	C:\AppServ\php7\php.ini
Scan this dir for additional .ini files	(none)
Additional .ini files parsed	(none)
PHP API	20180731
PHP Extension	20180731
Zend Extension	320180731
Zend Extension Build	API320180731,TS,VC15
PHP Extension Build	API20180731,TS,VC15
Debug Build	no
Thread Safety	enabled
Thread API	Windows Threads
Zend Signal Handling	disabled

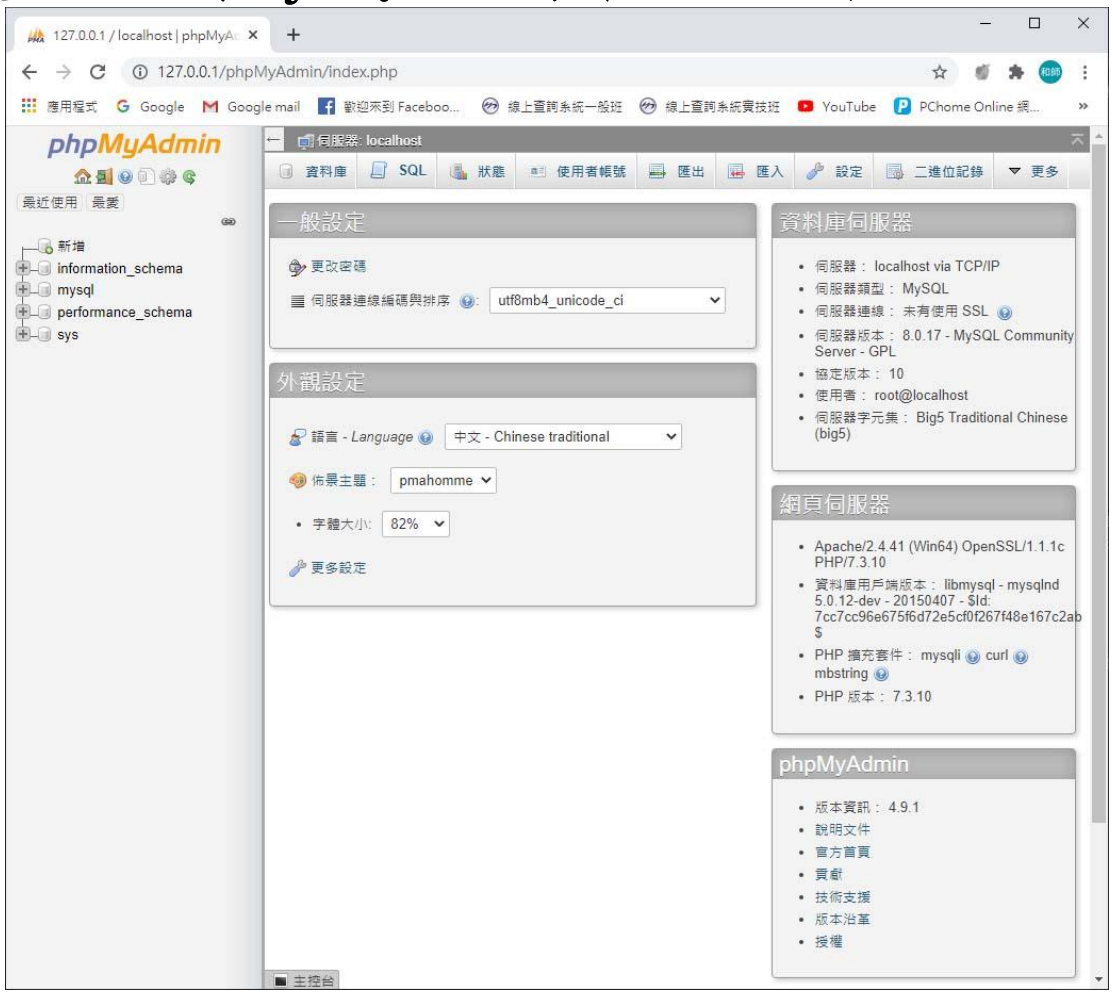
Appserv安裝過程

- 輸入剛才設定帳號root的密碼



Appserv安裝過程

- 進入後可以對MySQL進行管理動作



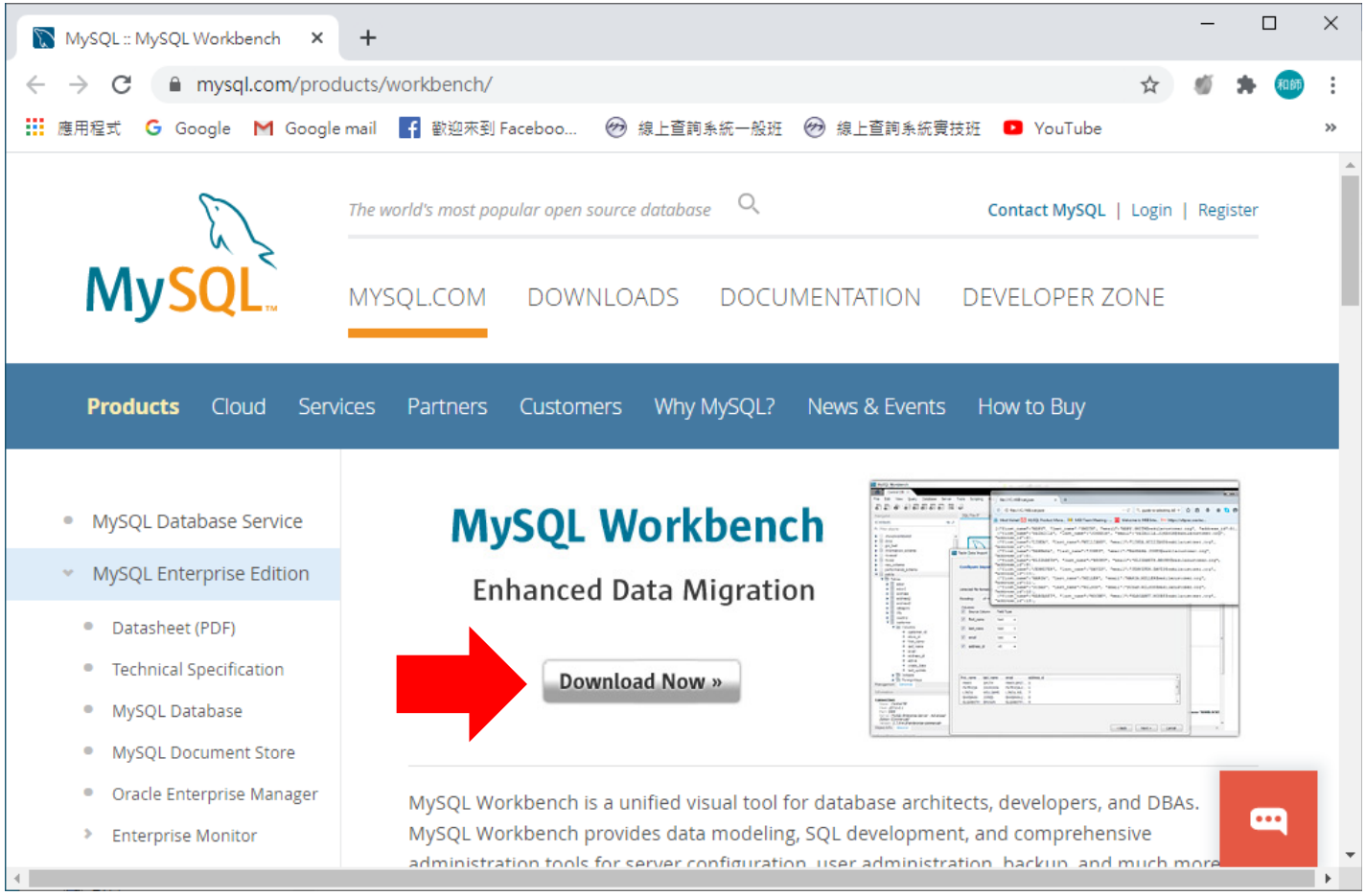
資料庫工具軟體

- 資料庫本身只是放資料的地方，我們還需要一些工具軟體幫我們更方便的操作資料庫。
- 最基本的是利用Telnet軟體(在本機的話就直接用「命令列模式」)登入後以指令操作。

- 視窗工具：
  - MySQL Workbench
    - MySQL官方提供的專門管理MySQL資料庫的工具。
  - HeidiSQL
    - 是一個免費的資料庫管理工具，可以用於MySQL，以及Microsoft SQL Server和PostgreSQL等。

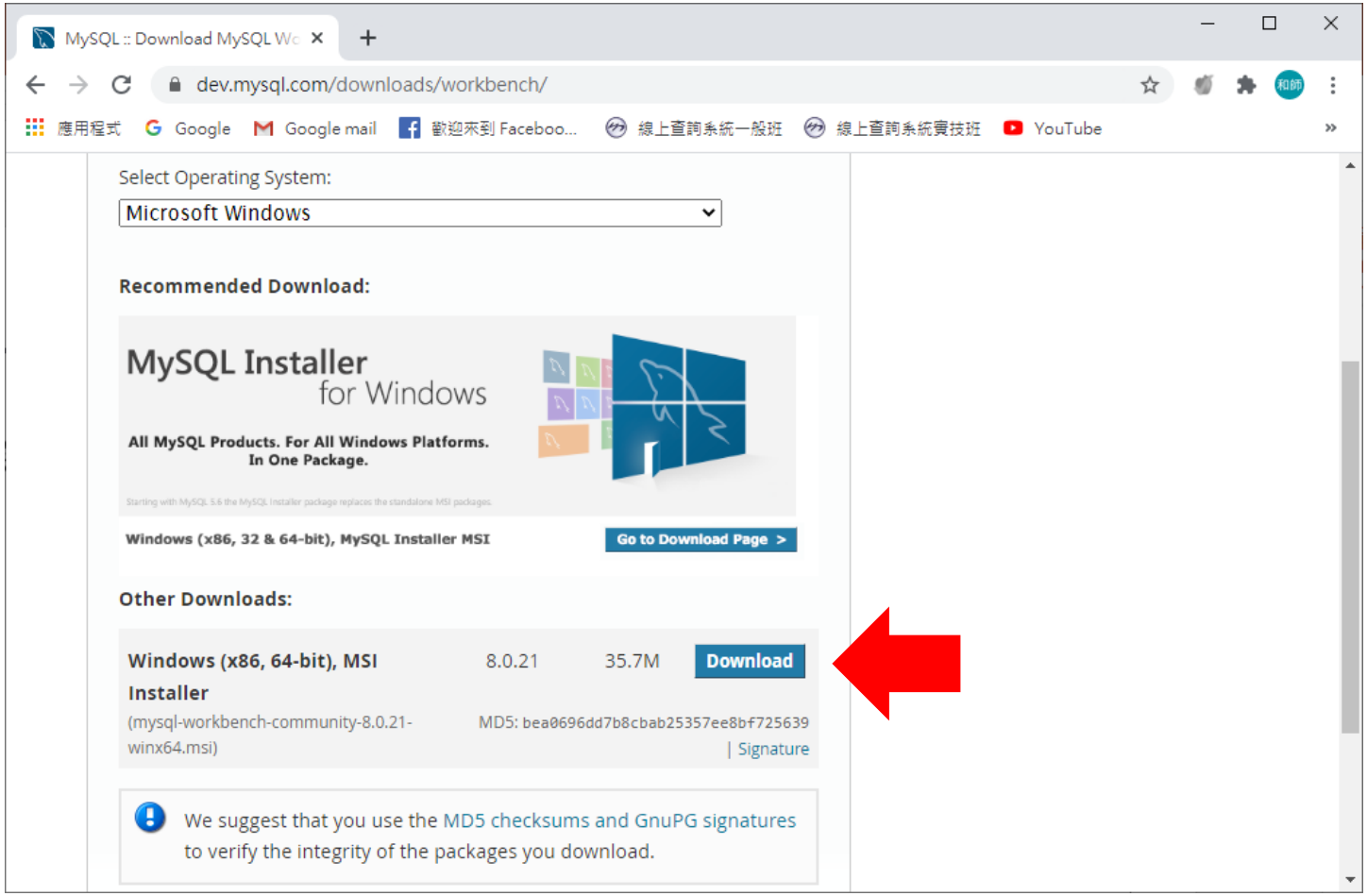
MySQL Workbench安裝

- 到MySQL網站下載MySQL Workbench



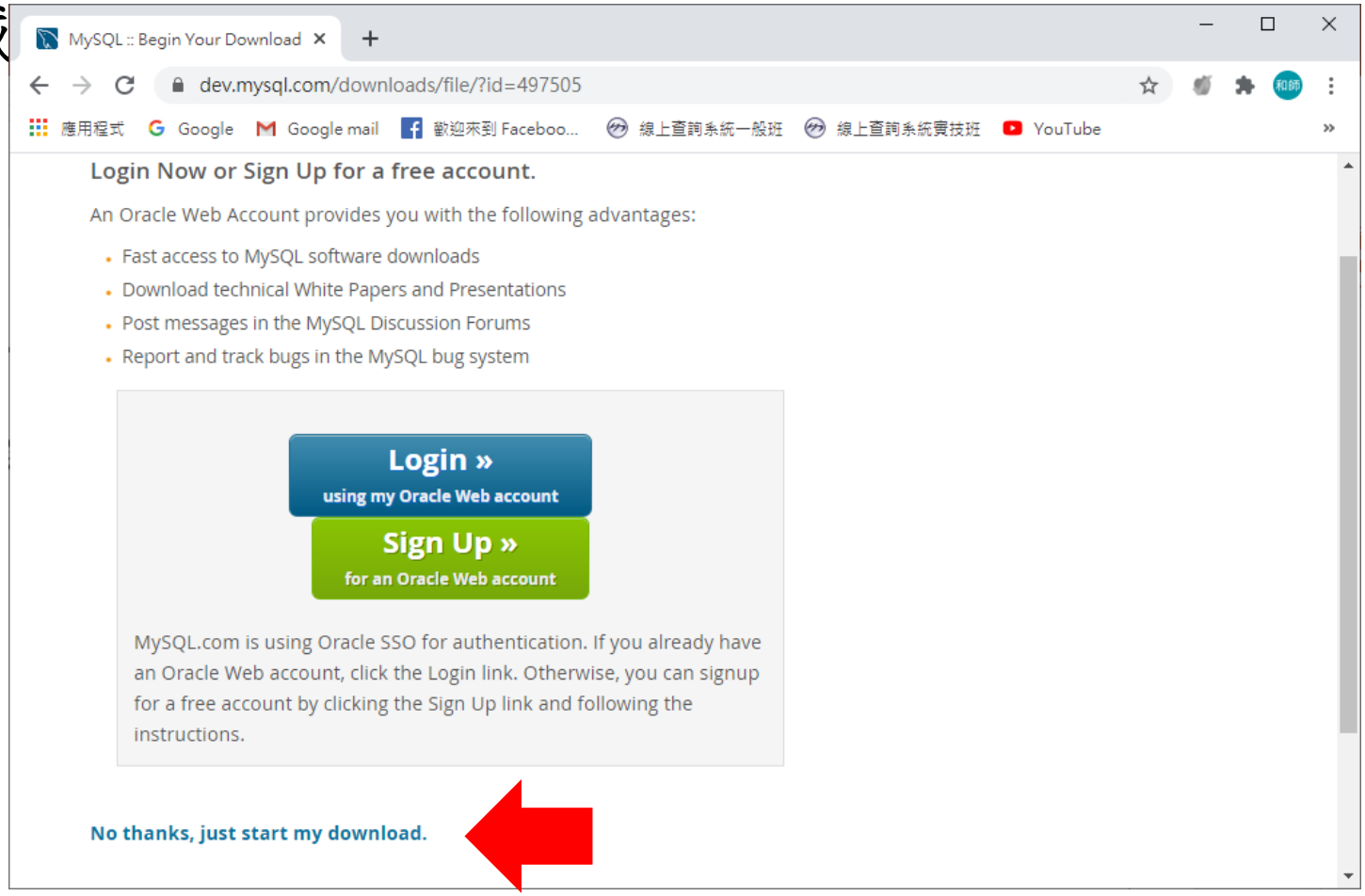
MySQL Workbench安裝

- 網頁會偵測你的作業系統類型，點選Download



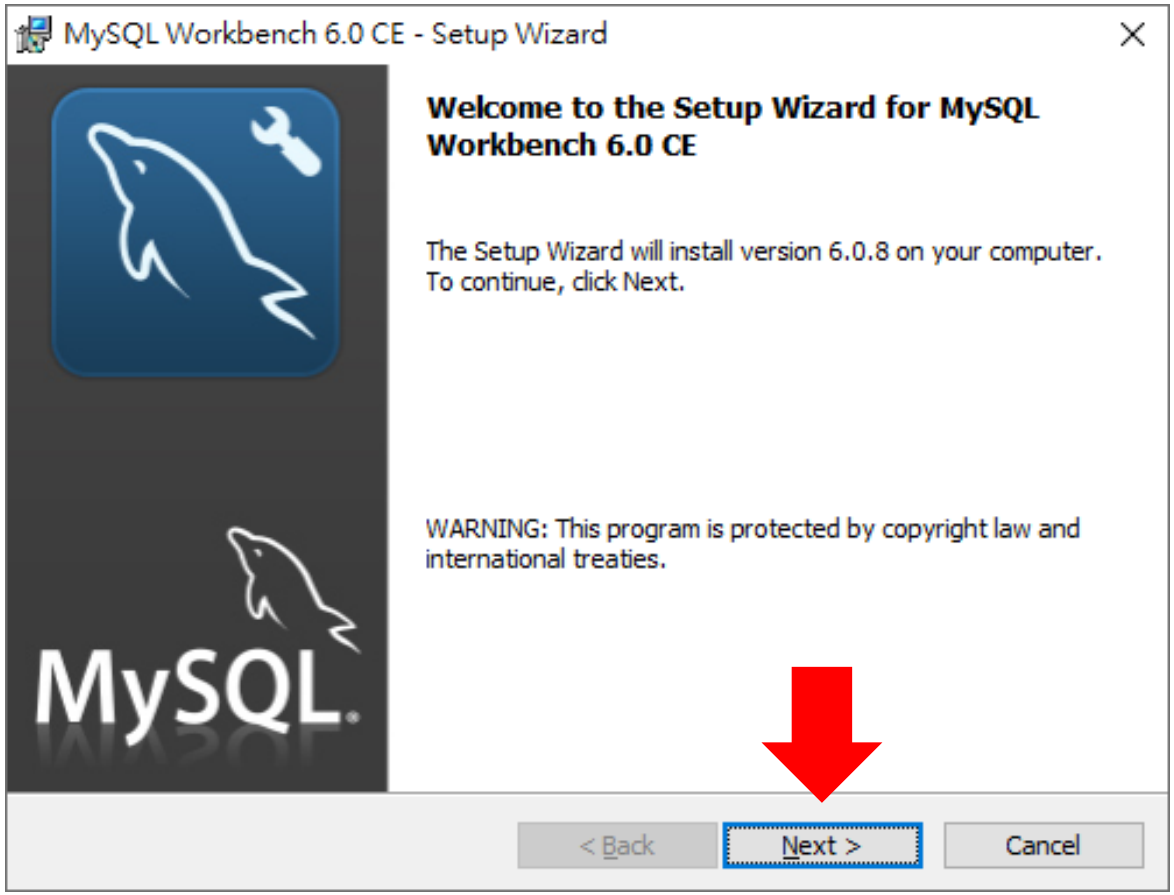
MySQL Workbench安裝

- 會要求登入，略過，直接點「謝謝，我要直接下載



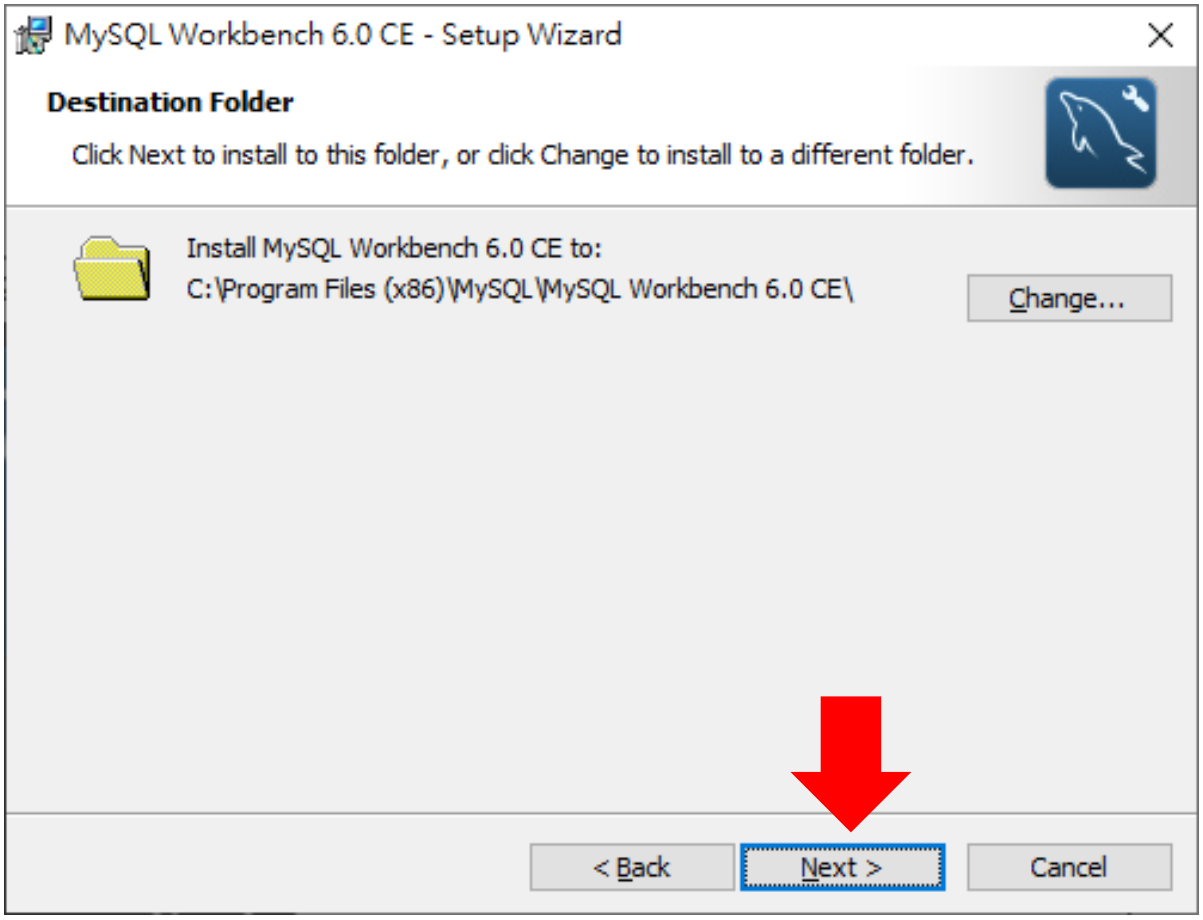
MySQL Workbench安裝

- 開始安裝，不囉嗦，沒什麼要你同意的，按下一步就好



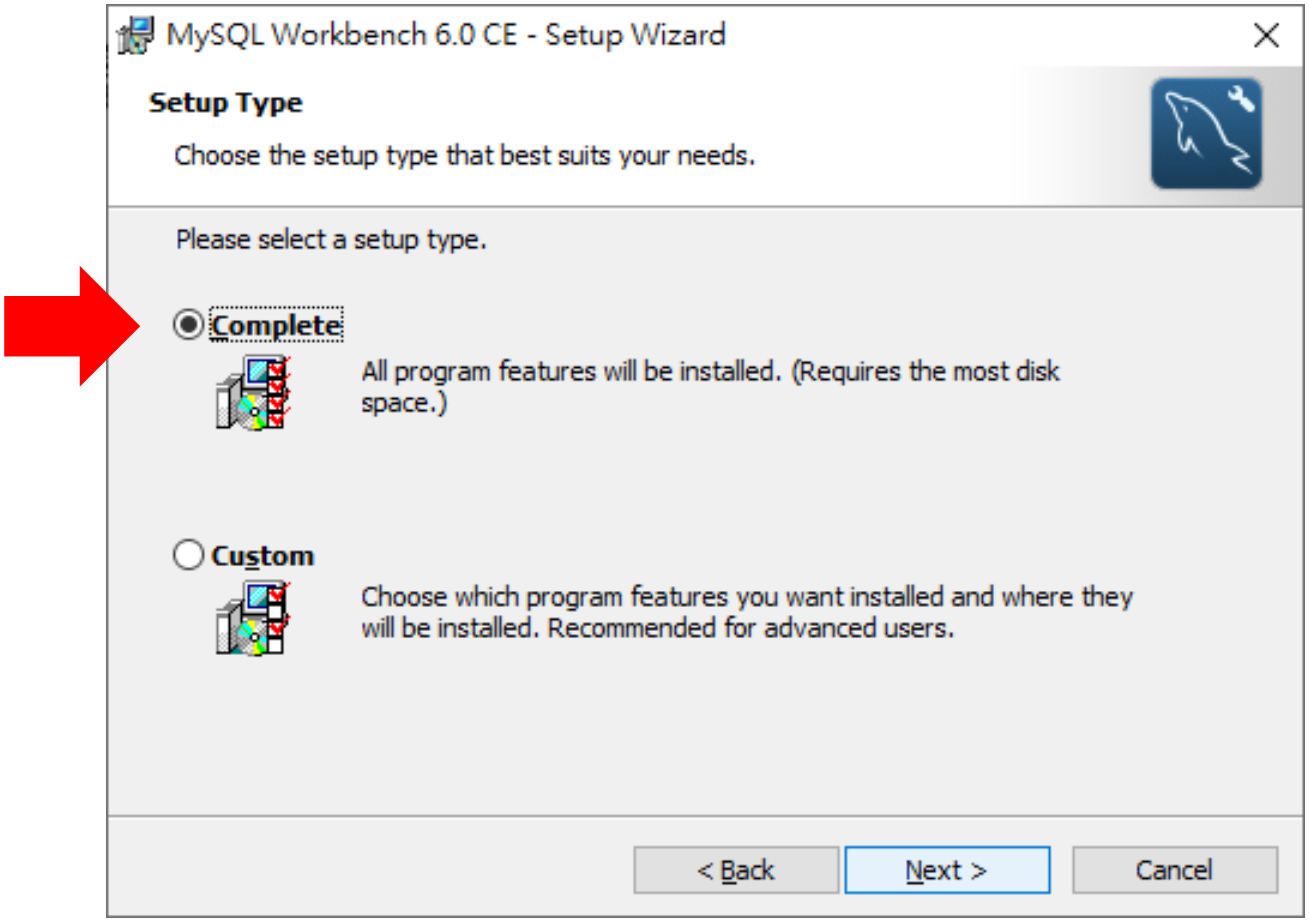
MySQL Workbench安裝

- 選擇安裝在何處，用內定路徑就好



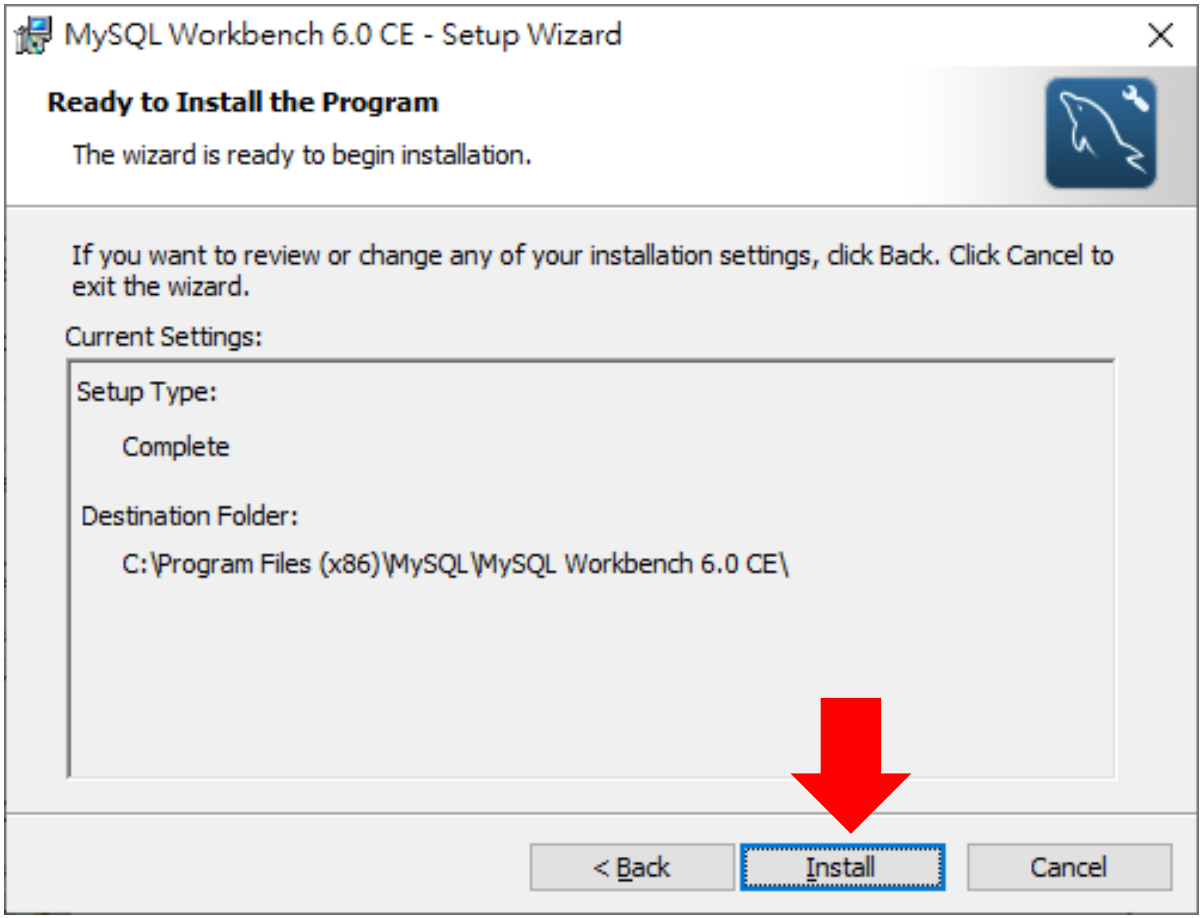
MySQL Workbench安裝

- 要完整安裝還是自訂安裝，選完整安裝



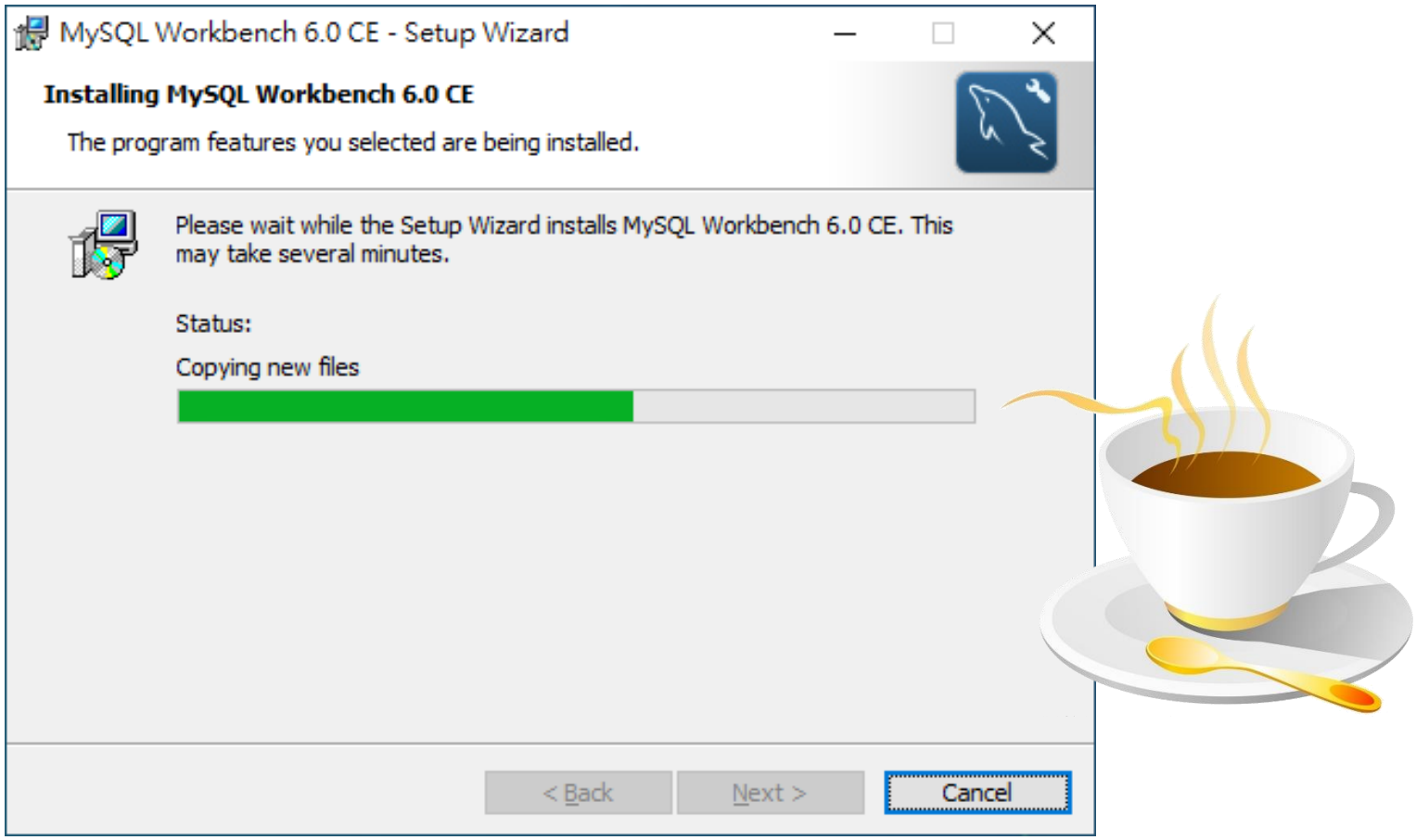
MySQL Workbench安裝

- 如果想修正剛才的選項可以回上一步，OK的話選安裝



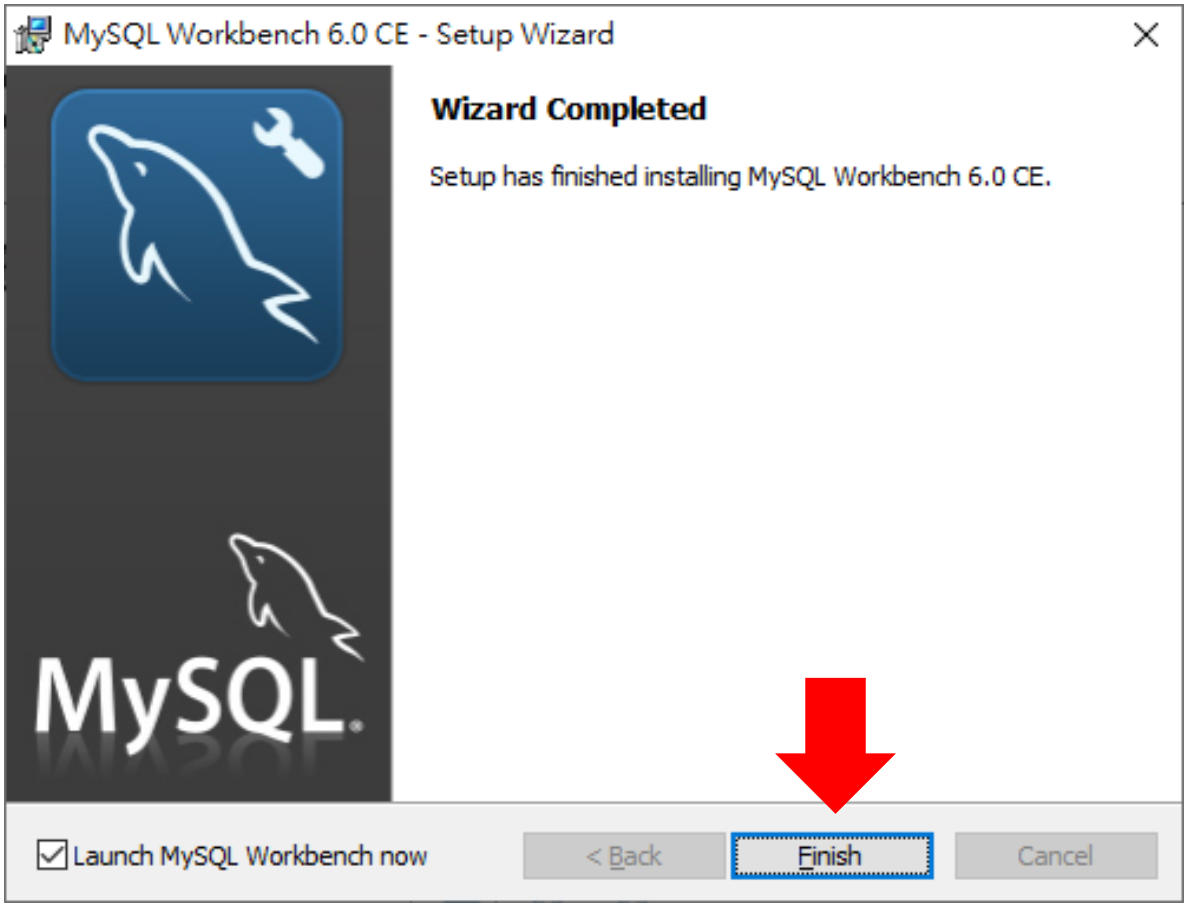
MySQL Workbench安裝

- 安裝中，咖啡時間



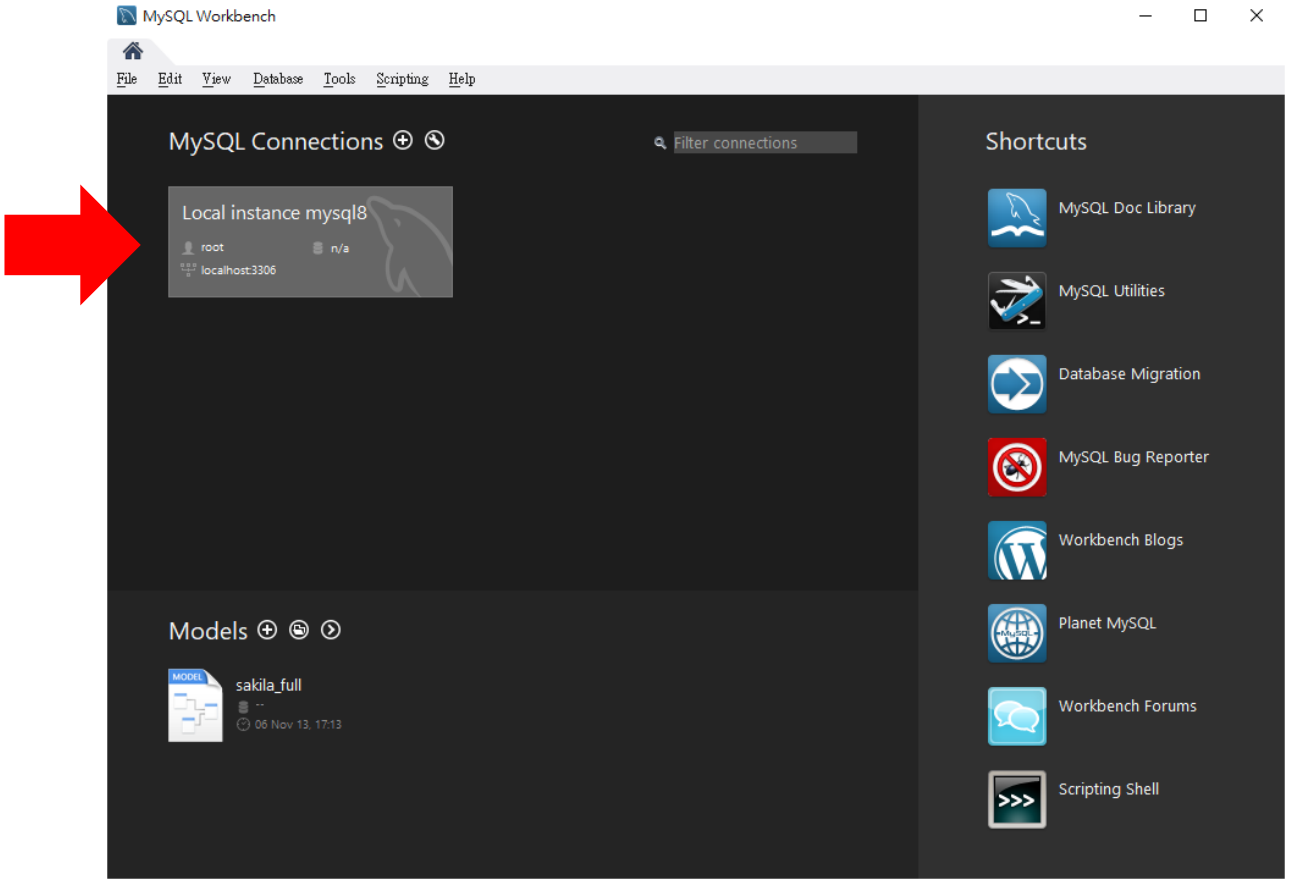
MySQL Workbench安裝

- 安裝完成，簡潔有力，選完成



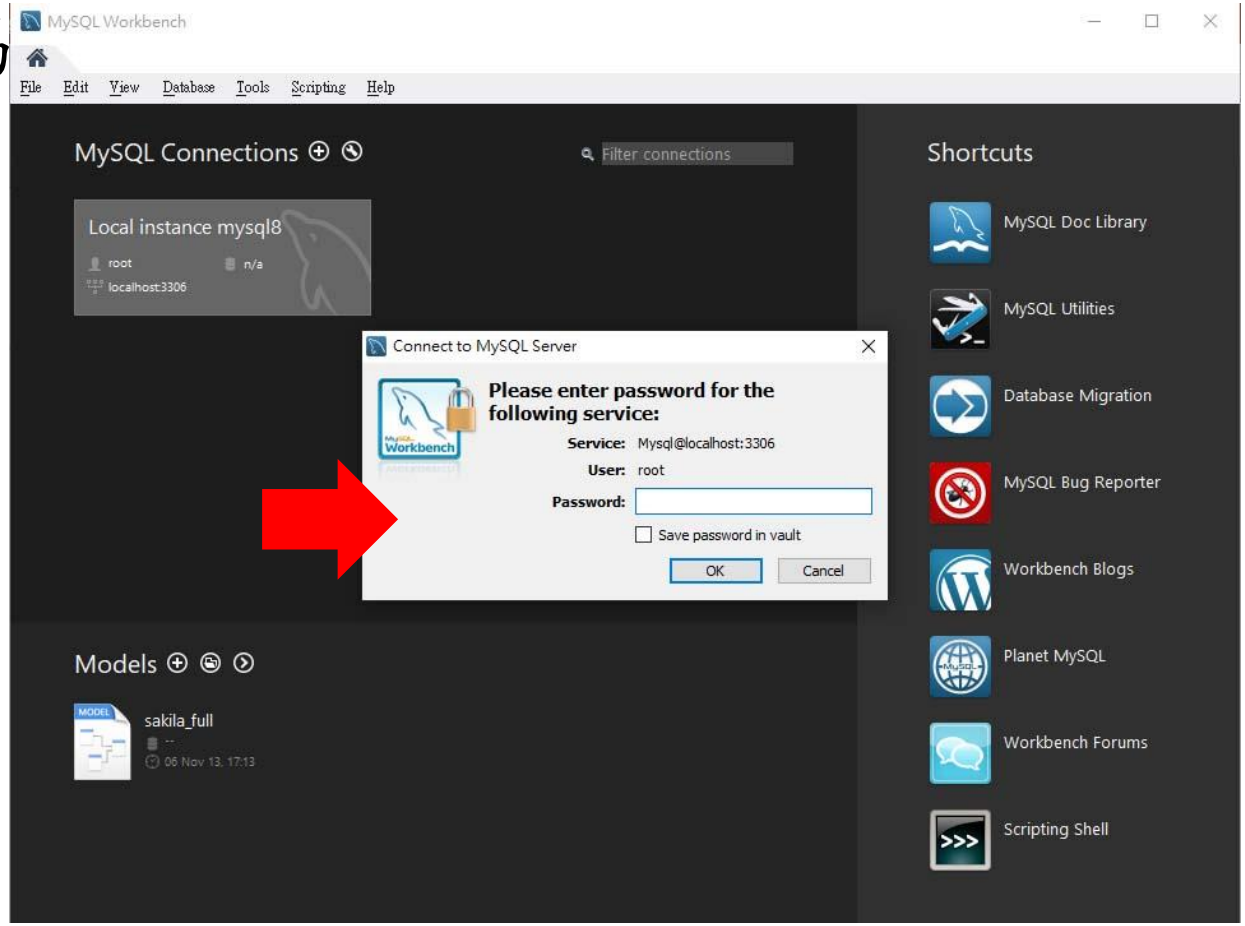
MySQL Workbench安裝

- 執行的畫面，它應該會偵測到本機已經有MySQL了，所以連結資訊已經在畫面上，點選它



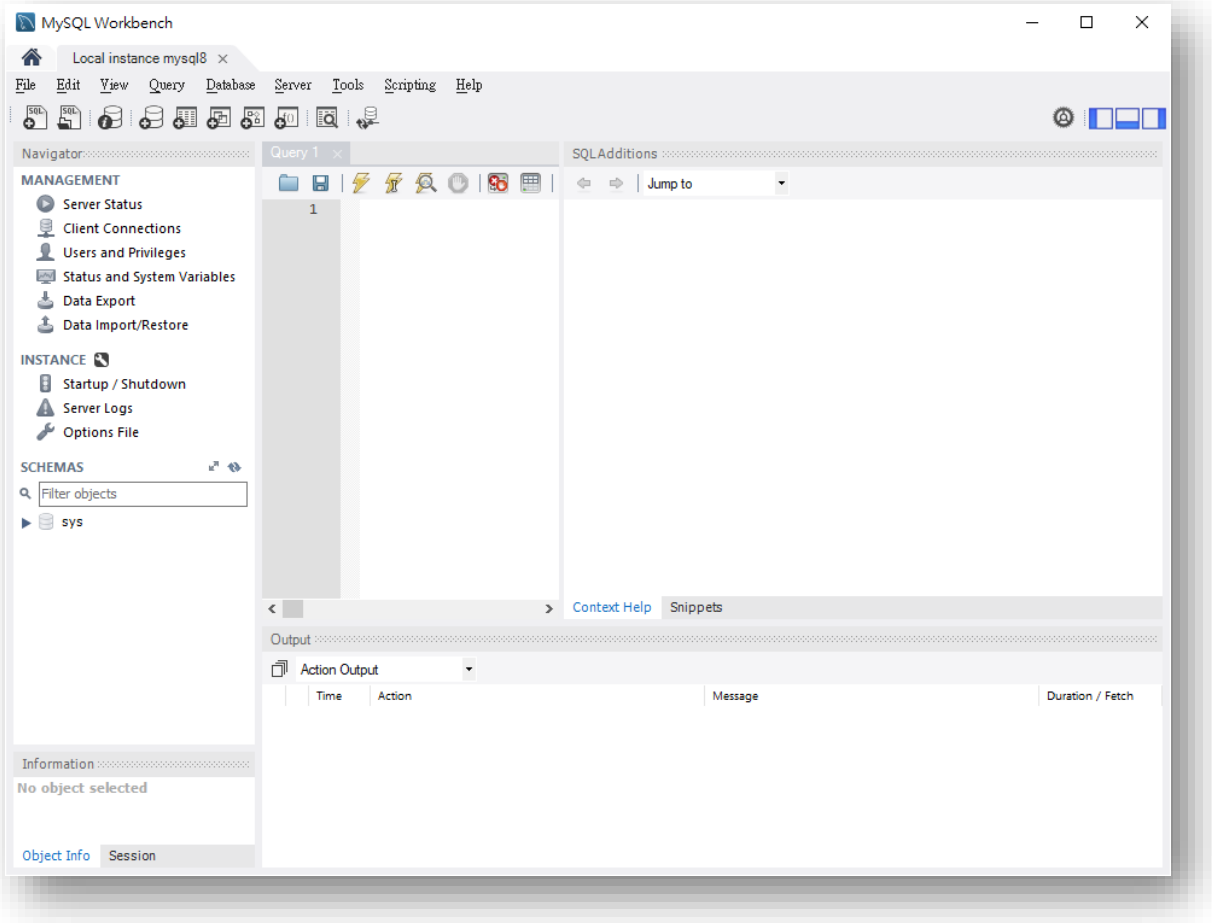
MySQL Workbench安裝

- 進入MySQL前照慣例驗證身分及密碼，輸入root密碼



MySQL Workbench安裝

- 這是一個很方便、功能很強的MySQL管理工具，也可以連線到遠端的MySQL(當然你必須有權限)

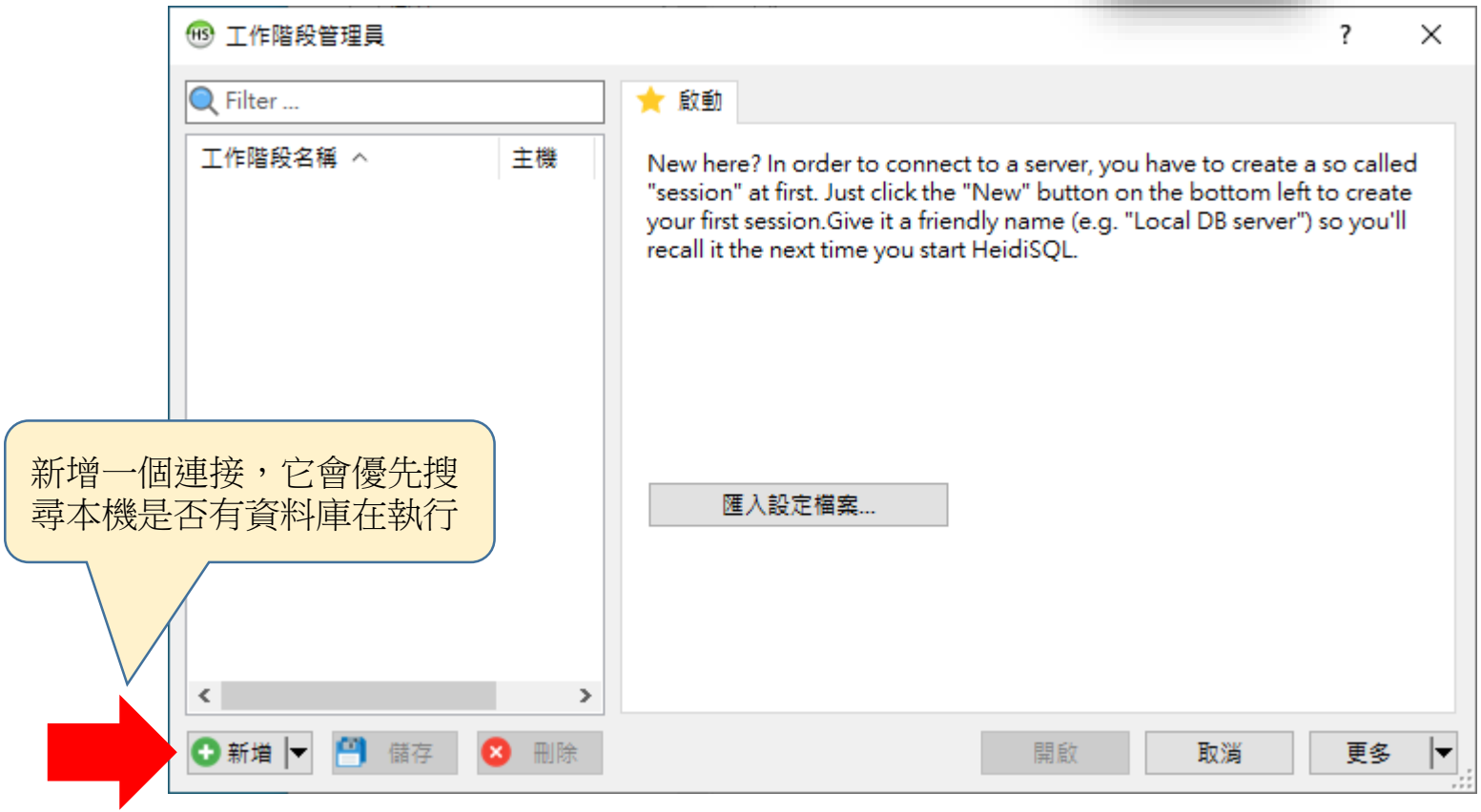


HeidiSQL

- 到https://www.heidisql.com/網站下載免安裝版

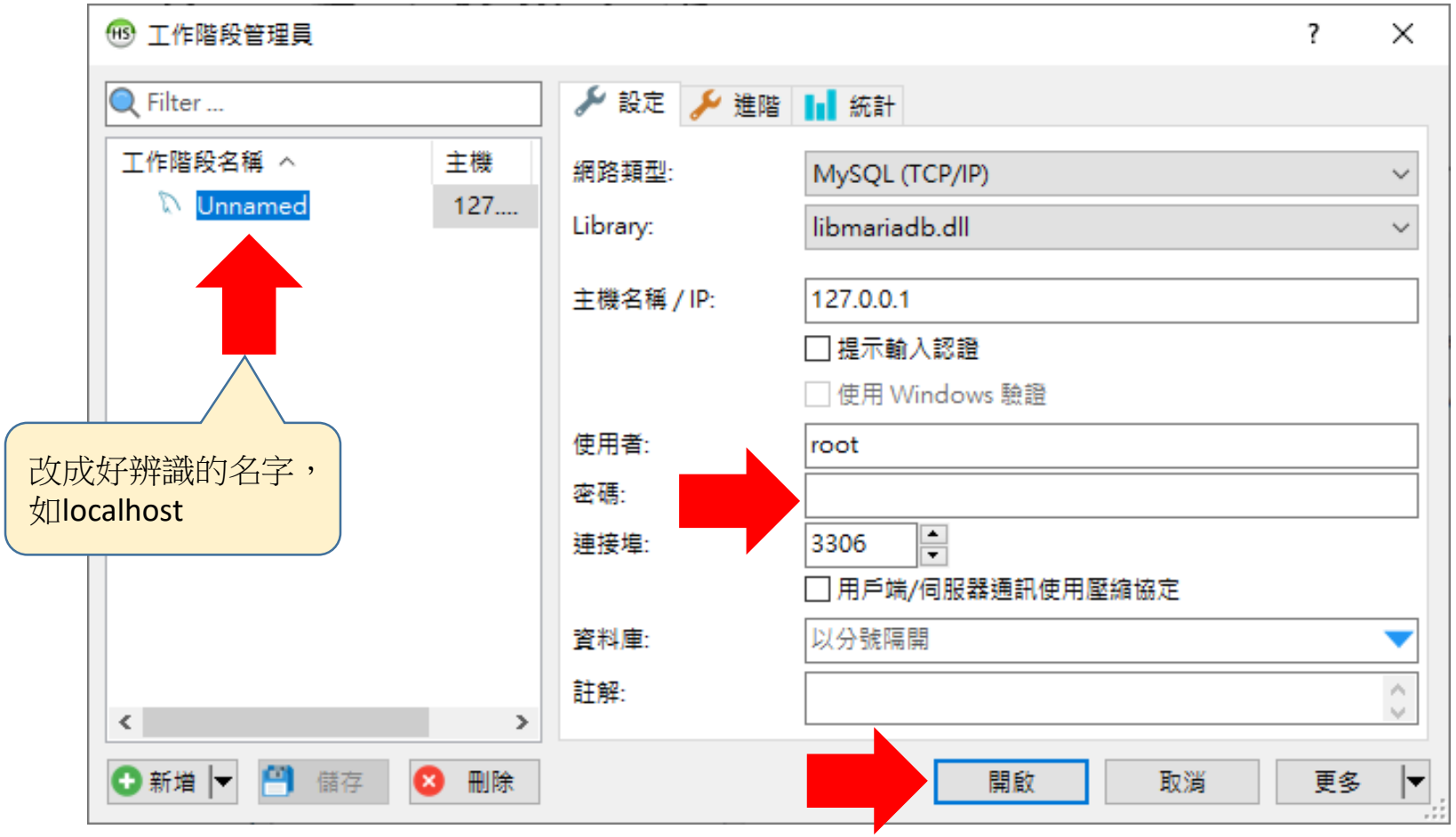
HeidiSQL

- 它是一個壓縮檔(.zip)，解開後



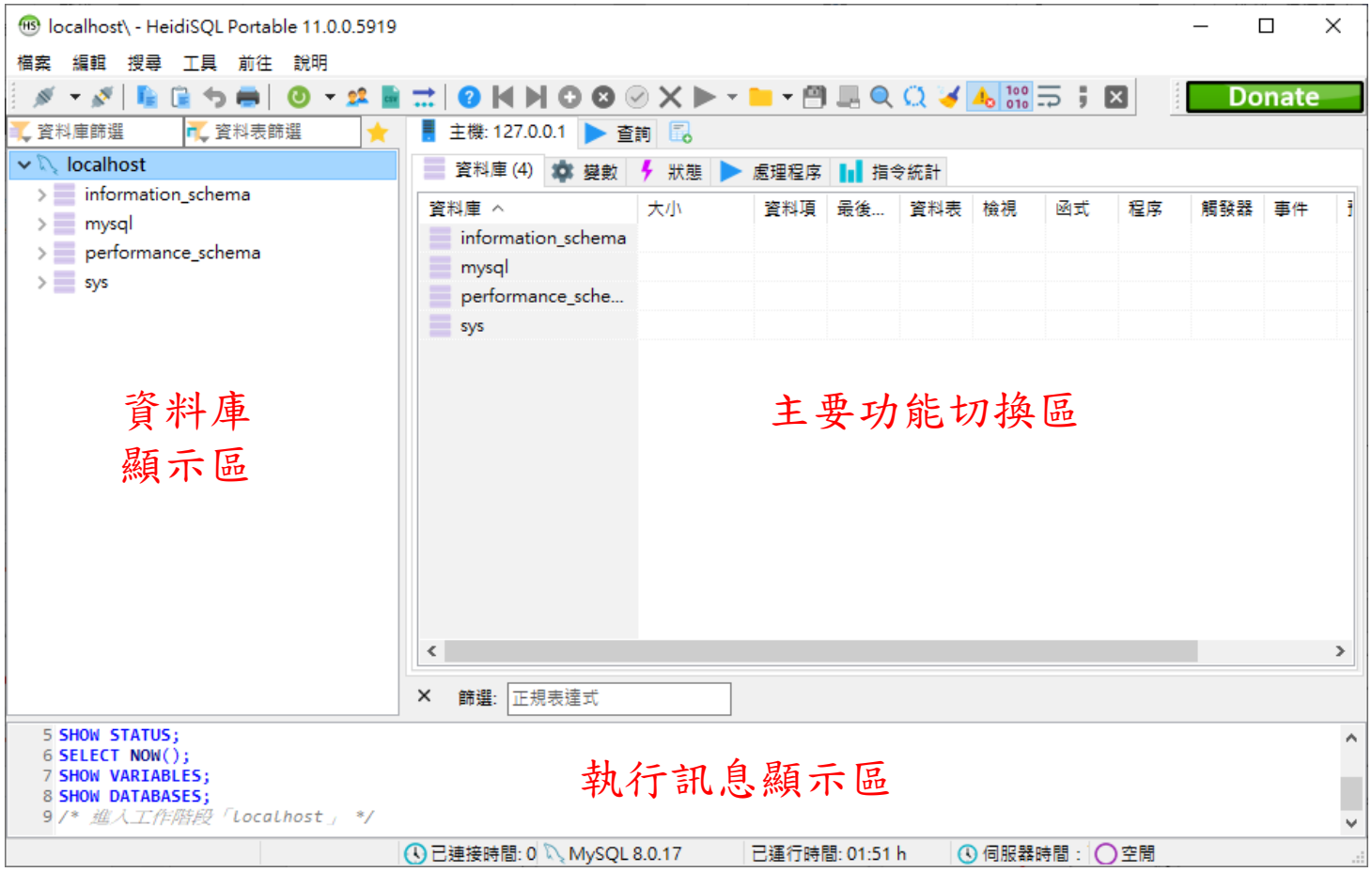
HeidiSQL

- 找到本機的MySQL，輸入密碼即可連接



HeidiSQL

- 功能清晰的主畫面



HeidiSQL


localhost\我的商店\ - HeidiSQL 11.0.0.5919
檔案 編輯 搜尋 工具 前往 說明
執行
資料庫篩選
資料表篩選
主機: 127.0.0.1
資料庫: 我的商店
查詢*
localhost
information_schema 0 B
mysql
performance_schema
sys
我的商店 128.0 ...
分店負責人清單 16.0 KiB
商品清單 16.0 KiB
負責人清單 16.0 KiB
販賣資料 64.0 KiB
顧客清單 16.0 KiB
1 SELECT * FROM 商品清單
選「查詢」，在此
輸入SQL指令
Filter ...
欄位
SQL 函式
SQL 關鍵字
SQL 片段
查詢歷史
查詢分析
連結參數
商品清單 (10r × 5c)
商品ID 商品名稱 群組名稱 進貨單價 販賣單價
1 桌上型電腦 電腦主機 150,000
10 網路卡 網路設備 15,000
2 筆記型電腦 電腦主機 230,000 270,000
已建立自己的資料庫
SQL執行結果
篩選: 正規表達式
21 SHOW TRIGGERS FROM `information_schema`;
22 SHOW EVENTS FROM `information_schema`;
23 SELECT *, EVENT_SCHEMA AS `Db`, EVENT_NAME AS `Name` FROM information_schema.`EVENTS` WHERE `EVENT_SCHEMA`='我
24 /* Loading file "C:\Users\Orion Liu\AppData\Roaming\HeidiSQL\Backups\query-tab-2020-09-12_21-23-40-589.sql" (2
25 SELECT * FROM 商品清單;
26 /* 受影響行數: 0 資料行數: 10 警告: 0 本操作所需時間： 1 查詢: 0.000 秒 */
r1 : c19 (24 B) 已連接時間: MySQL 8.0.17 已運行時間: 4 天, 23:4 伺服器時間 空閒

休息一下~

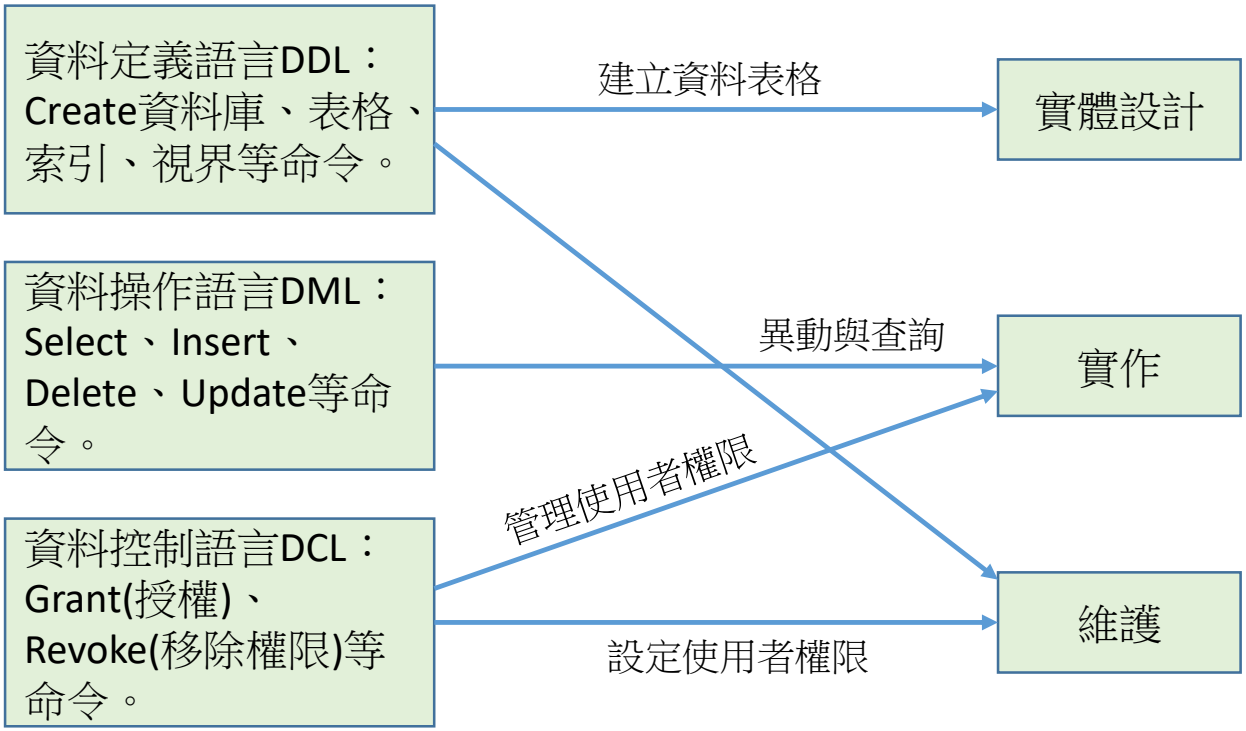
SQL語言簡介

- 結構化查詢語言SQL(Structured Query Language)：是一種與資料庫溝通的共通語言，是專門為資料庫處理而設計的第四代「非程序性」查詢語言。
- 所有的資料庫軟體都支援此查詢語言，所以要操作資料庫一定要熟悉SQL。

- SQL提供三種命令：
  - 資料定義語言(Data Definition Language, DDL)
  - 資料操縱語言(Data Manipulation Language, DML)
  - 資料控制語言((Data Control Language, DCL)

SQL語言簡介

- DDL、DML與DCL：



資料定義語言DDL

- 用來定義資料庫、資料表(含欄位名稱、資料型態及設定完整性限制)。
- DDL提供三種指令：

	Database(資料庫)	Table(表格)	View(視界)
新增	Create Database	Create Table	Create View
修改	Alter Database	Alter Table	Alter View
刪除	Drop Database	Drop Table	Drop View

資料定義語言DDL

- MySQL其它相關指令：

show databases	列出所有資料庫
use 資料庫名稱	使用/開啟指定的資料庫
show tables	列出使用中資料庫的資料表格
describe / desc 資料表名稱	列出資料表欄位資訊
\q	離開MySQL(DOS模式時)

資料定義語言DDL

- Create Database(建立資料庫)基本語法：

```
Create Database [ IF NOT EXISTS ] 資料庫名稱;
```

  - Ex:建立一個資料庫，名稱是「我的商店」，如果不存在的話。

```
Create Database IF NOT EXISTS 我的商店;
```

  - 直接用 `Create Database 我的商店;` 如果該資料庫已經存在則會有錯誤訊息。

資料定義語言DDL

- 實際操作：
  - 進入命列列模式，輸入下列指令，再輸入密碼

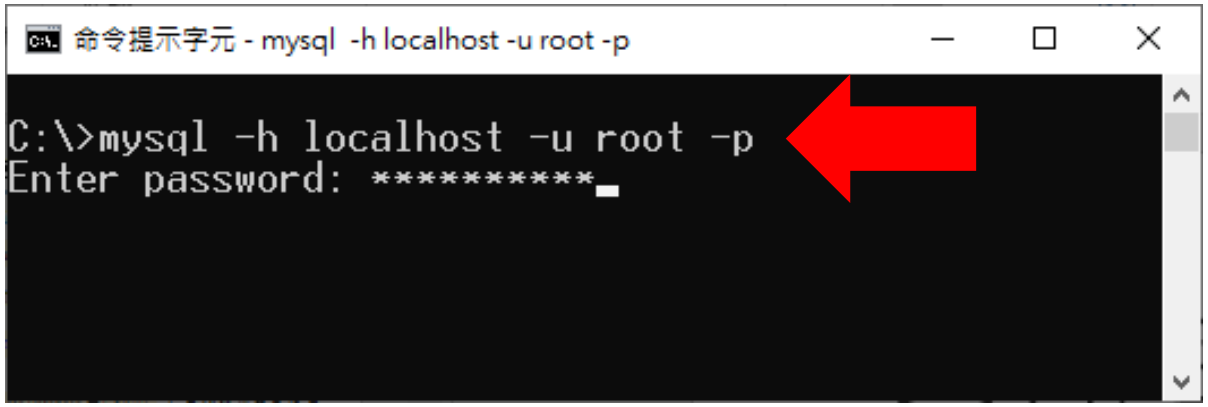
```
C:\mysql -h localhost -u root -p
```

執行 MySQL

要連線的主機或IP (本機時可省略)

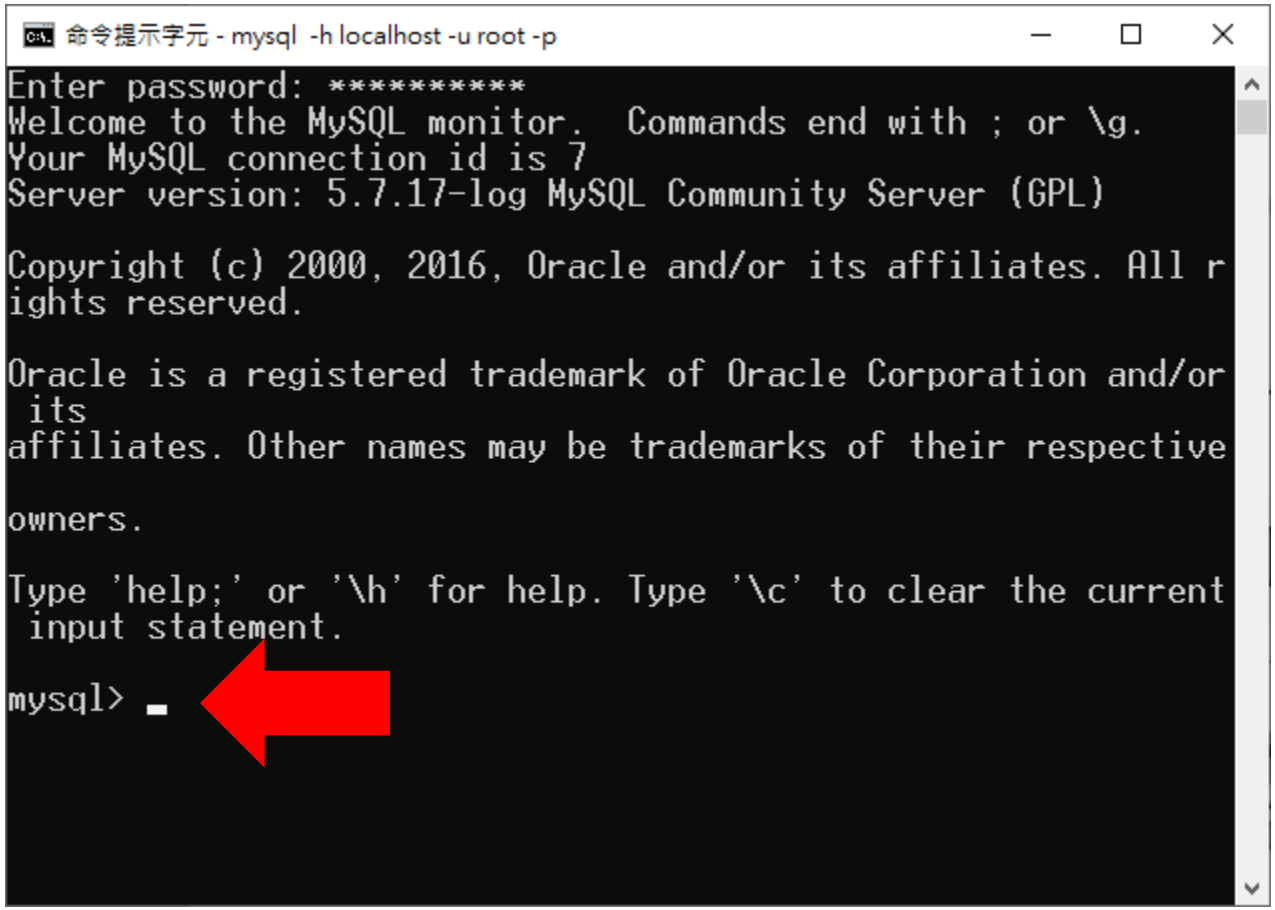
使用者名稱

需輸入密碼



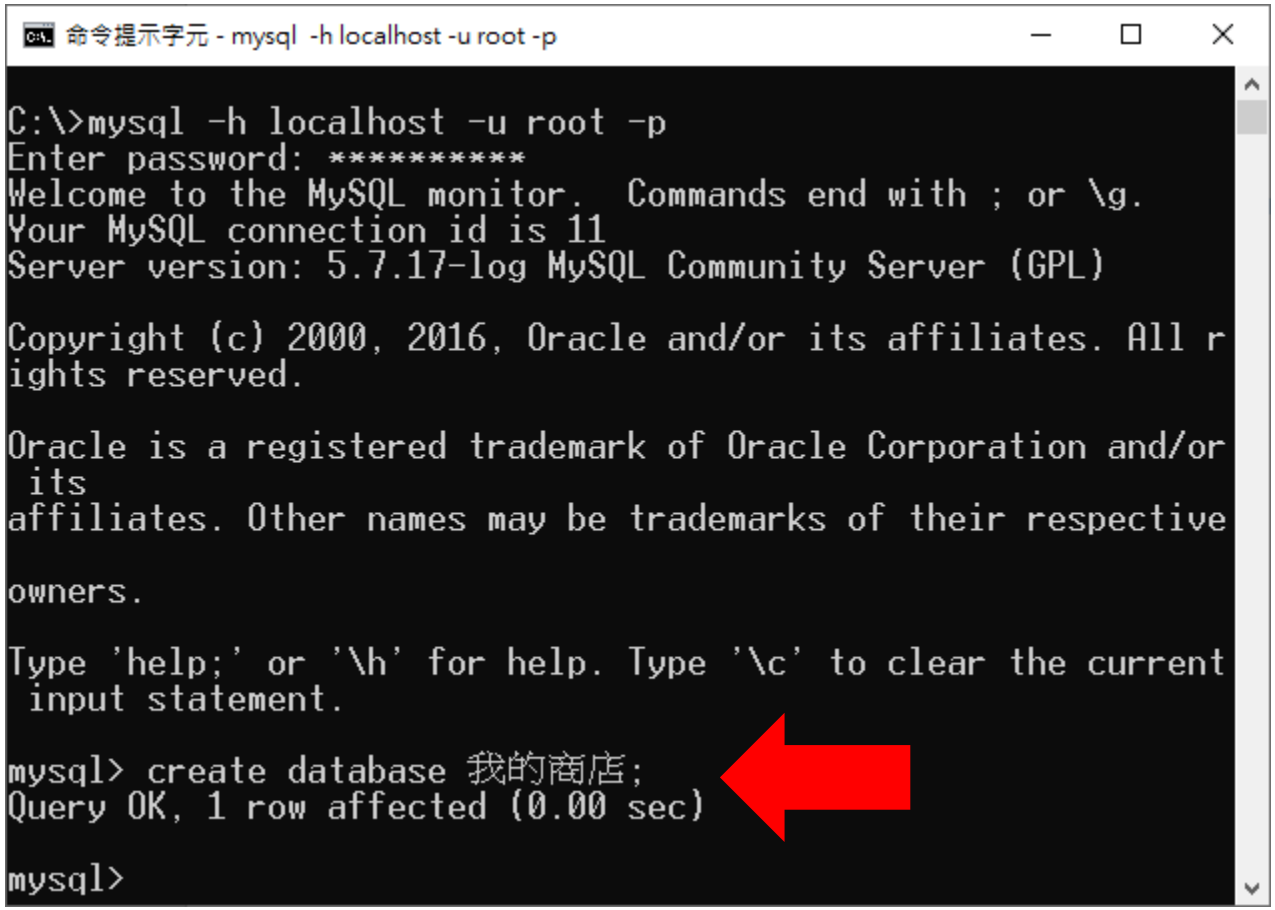
資料定義語言DDL

- 成功進入後會出現提示字元「 mysql>_ 」：



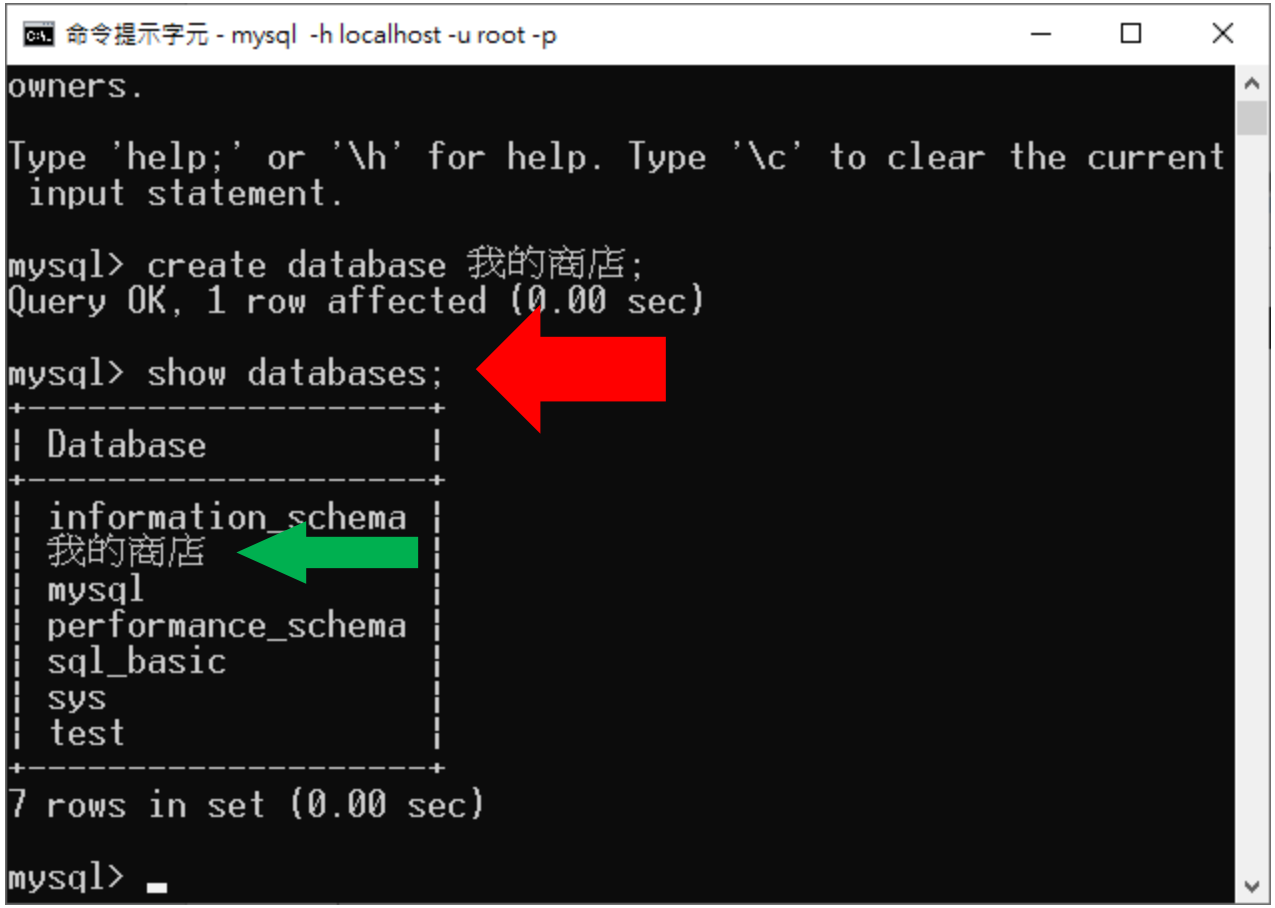
資料定義語言DDL

- 建立「我的商店」資料庫



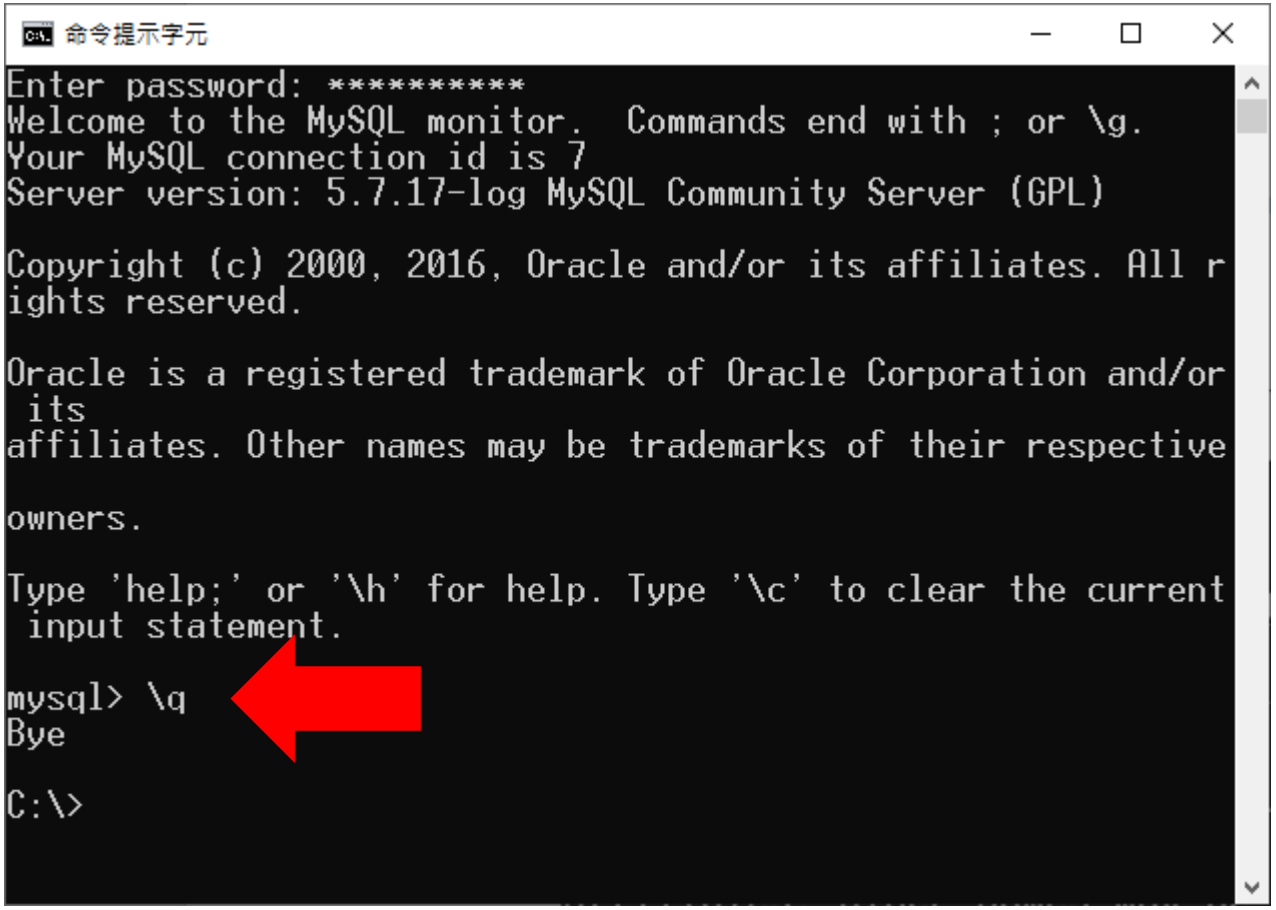
資料定義語言DDL

- 用「show databases;」查看有哪些資料庫：



資料定義語言DDL

- 輸入「 \q 」可以結束MySQL。



資料定義語言DDL

- Alter database(修改資料庫)基本語法：
  - 資料庫一旦建立就不要再改名，尤其有資料後更是不能隨便亂改，所以這個指令只讓你改改預設字元。

```
Alter Database  資料庫名稱
Character set 字元集名稱;
```

  - Ex:將「我的商店」的字元集改成UTF8。

```
Alter Database 我的商店 character set utf8;
```

資料定義語言DDL

- Drop database(刪除資料庫)基本語法：
  - 注意一旦刪除，所有資料也會消失，沒有後悔的機會。

```
Drop Database [ IF EXISTS ] 資料庫名稱;
```

  - Ex:將「我的商店」資料庫刪除：

```
Drop Database  IF EXISTS  我的商店;
```

資料定義語言DDL

- Create Table(建立資料表)：
  - 建立新資料表步驟：
    - 1.決定資料表名稱及相關欄位名稱。
    - 2.決定欄位的資料型態。
    - 3.決定欄位的限制(指定值域)。
    - 4.決定那些欄位可以是空值(NULL)或不允許空值(NULL)。
    - 5.找出必須具有唯一值得欄位(主鍵)。
    - 6.找出主鍵-外鍵配對(兩個表格)。
    - 7.決定預設值(欄位值的初值設定)。
  - 好好規劃，決定後非絕對必要就不要再修改。

資料定義語言DDL

- Create Table(建立資料表)基本語法：

【格式】

```
Create Table 資料表
(欄位{資料型態 | 定義域}[NULL|NOT NULL][預設值][定義整合限制]
    ┆
Primary Key(欄位集合)   ←當主鍵
Unique(欄位集合)        ←當候選鍵
Foreign Key(欄位集合)   References 基本表(屬性集合)←當外鍵
  [ON Delete 選項] [ON Update 選項]
)
```

【符號說明】

- { | } 代表在大括號內的項目是必要項，但可以擇一。
- [ ] 代表在中括號內的項目是非必要項，依實際情況來選擇。

1. PRIMARY KEY 用來定義某一欄位為主鍵，不可為空值
2. UNIQUE 用來定義某一欄位具有唯一的索引值，可以為空值
3. NULL/NOT NULL 可以為空值 / 不可為空值
4. FOREIGN KEY 用來定義某一欄位為外部鍵

資料定義語言DDL

- Create Table(建立資料表)基本語法：
  - Ex:建立三個資料表如下

學生表			
欄位名稱	資料型態	空值	預設值
學號(PK)	字元(8)		
姓名	字元(4)	NOT NULL	
電話	字元(12)		
地址	字元(20)		

課程表			
欄位名稱	資料型態	空值	預設值
課號(PK)	字元(5)		
課名	字元(20)	NOT NULL	
學分數	整數		3
必選修	字元(2)		

選課表			
欄位名稱	資料型態	空值	預設值
學號(PK)	字元(8)		
課號	字元(5)		
成績	整數	NOT NULL	
選課日期	日期		系統時間

資料定義語言DDL

- Create Table(建立資料表)基本語法：
  - 建立「學生表」：

學生表			
欄位名稱	資料型態	空值	預設值
學號(PK)	字元(8)		
姓名	字元(4)	NOT NULL	
電話	字元(12)		
地址	字元(20)		

```
use 資料庫名稱;

Create table 學生表(
     學號   char(8),
     姓名   char(4) NOT NULL,
     電話   char(12),
     地址   char(20),
     primary key(學號)
)
```

最後一行不用逗號

資料定義語言DDL

- Create Table(建立資料表)基本語法：
  - 建立「課程表」：

課程表			
欄位名稱	資料型態	空值	預設值
課號(PK)	字元(5)		
課名	字元(20)	NOT NULL	
學分數	整數		3
必選修	字元(2)		

```
use 資料庫名稱;

Create table 課程表(
     課號   char(5),
     課名   char(20) NOT NULL,
     學分數   INT default 3,
     必選修   char(2),
     primary key(課號)
)
```

資料定義語言DDL

- Create Table(建立資料表)基本語法：
  - 建立「選課表」：

選課表			
欄位名稱	資料型態	空值	預設值
學號(PK)	字元(8)		
課號	字元(5)		
成績	整數	NOT NULL	
選課日期	日期		系統時間

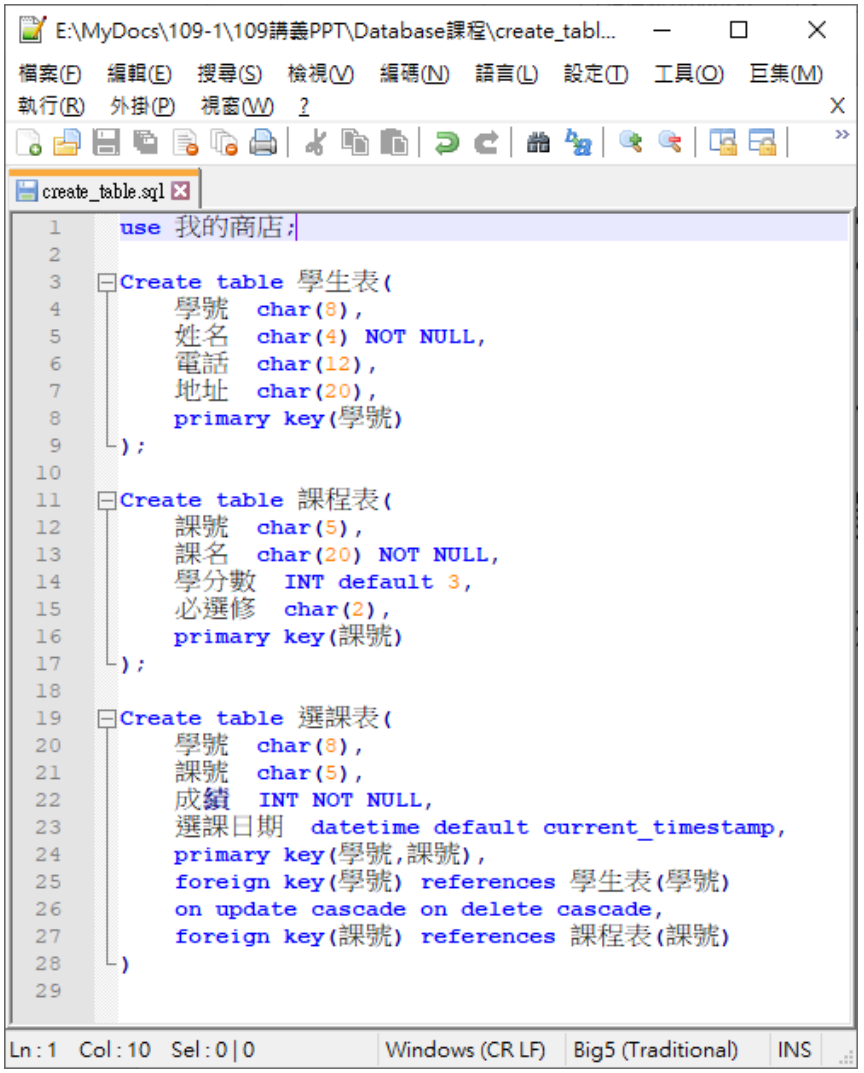
```
use 資料庫名稱;

Create table 選課表(
    學號   char(8),
    課號   char(5),
    成績   INT NOT NULL,
    選課日期   datetime default current_timestamp,
    primary key(學號,課號),
    foreign key(學號) references 學生表(學號)
    on update cascade on delete cascade,
    foreign key(課號) references 課程表(課號)
)
```

沒有逗號，表示核下一行是同一個命令

資料定義語言DDL

- 由於建立資料表命令稍長，可以先在記事本編輯好，再一次匯入MySQL。

（進入MySQL>模式很難編修較長的指令）

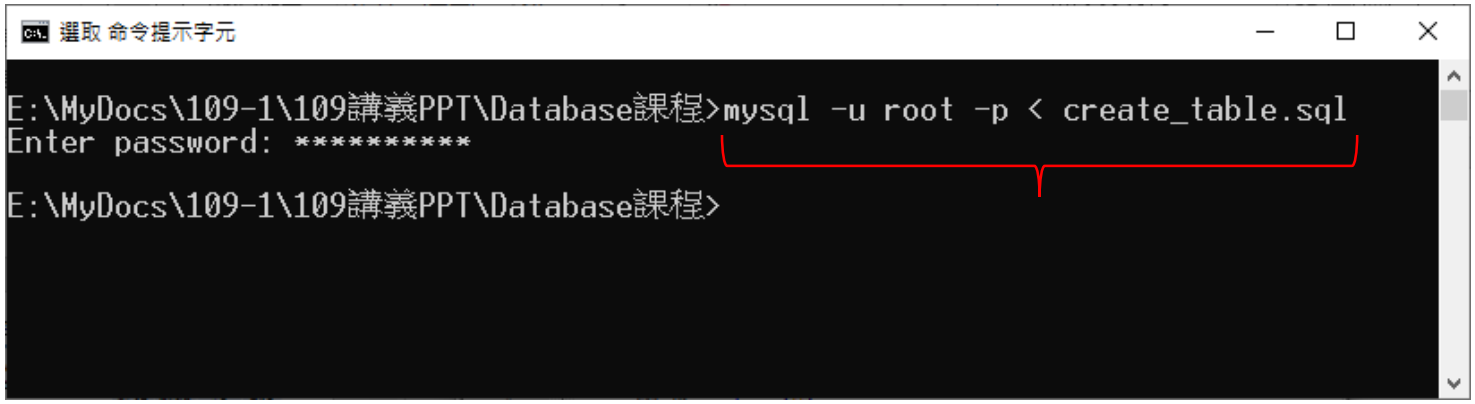


資料定義語言DDL

- 例如存成「create_table.sql」，使用下列指令，一次匯入MySQL去執行。

```
mysql –u root –p < create_table.sql
```

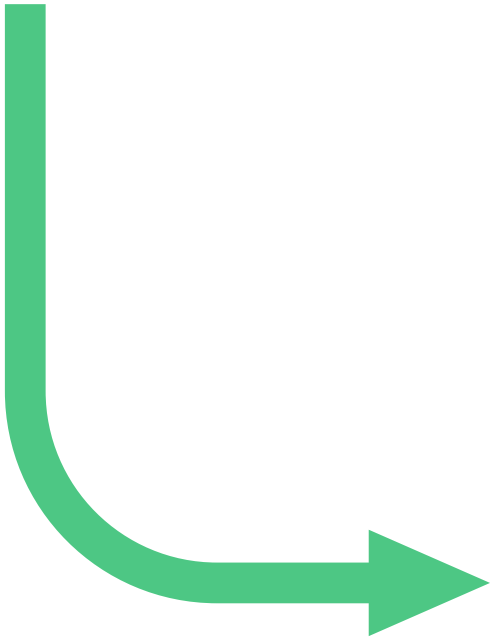
導入此檔案的意思



- 如果沒有錯誤訊息就是成功了。

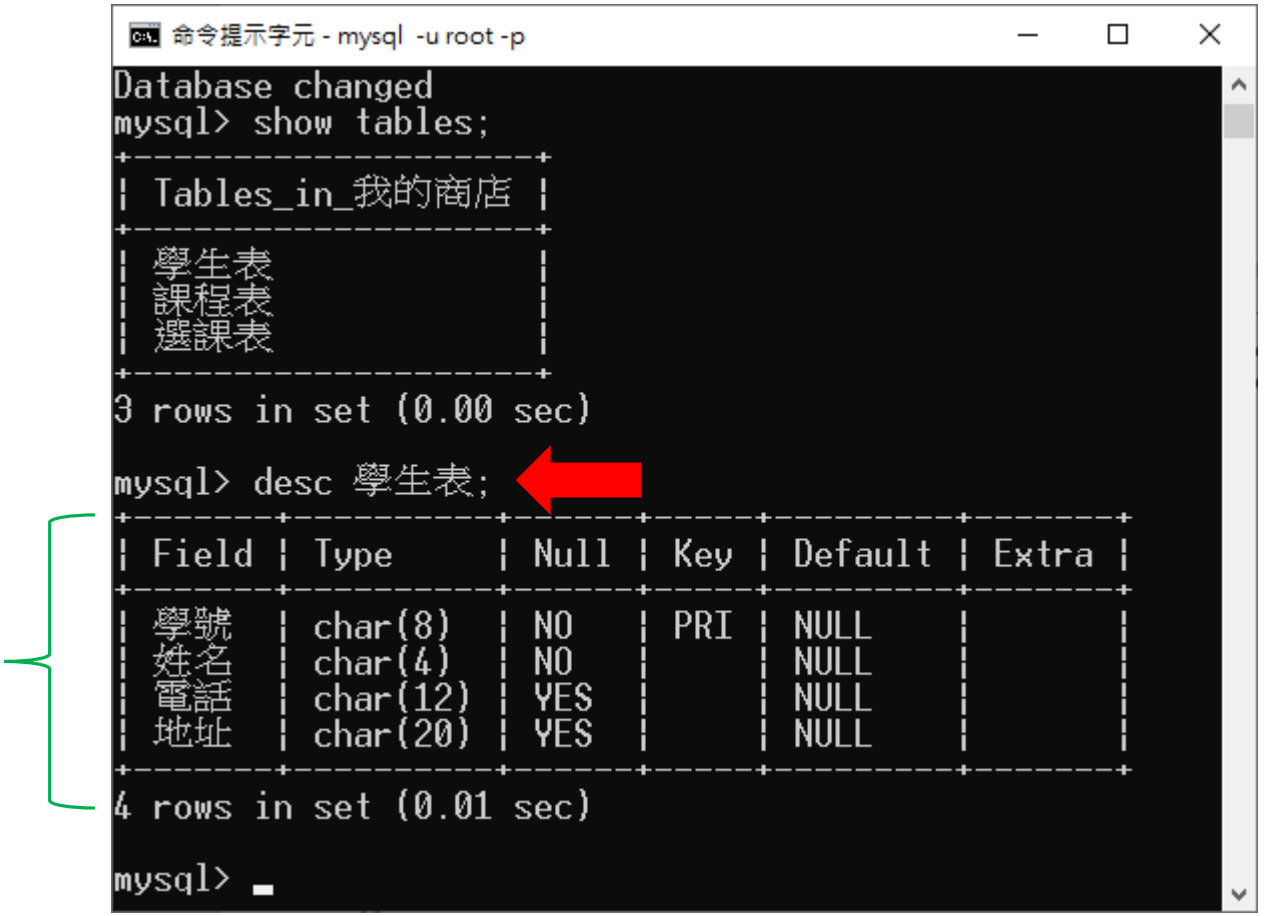
資料定義語言DDL

- 查看一下資料庫：
  確實有三個資料表了。



資料定義語言DDL

- 查看一下資料表內容：



資料定義語言DDL

- Alter table(修改資料表)基本語法：
  - 資料表一旦建立就盡量不要再修改，尤其有資料後，一定要注意不能因修改而損害到原有的資料。且修改資料表無可避免的一定會需要修改相對的應用程式。

```
ALTER TABLE 資料表名稱
ADD 欄位名稱 { 資料型態 | 定義域 } [NULL|NOT NULL ]
    [ 預設值 ]
```

```
ALTER TABLE 資料表名稱
Modify 欄位名稱 { 資料型態 | 定義域 } [NULL|NOT NULL ]
       [ 預設值 ]
```

```
ALTER TABLE 資料表名稱
Drop 欄位名稱
```

資料定義語言DDL

- Alter table 新增欄位：
  - 在原來的學生表中新增一個「電子信箱」欄位。

學生表			
欄位名稱	資料型態	空值	預設值
學號(PK)	字元(8)		
姓名	字元(4)	NOT NULL	
電話	字元(12)		
地址	字元(20)		
電子信箱	字元(50)		

```
alter table 學生表
  add 電子信箱 char(50);
```

- 通常因法令或適用原因改變而須修改，但不會一次修改多個欄位(不然一開始的設計和正規化怎麼做的?)

資料定義語言DDL

- Alter table 新增欄位：
  - 在原來的學生表中再新增一個「性別」欄位，預設值為「女」。

學生表			
欄位名稱	資料型態	空值	預設值
學號(PK)	字元(8)		
姓名	字元(4)	NOT NULL	
電話	字元(12)		
地址	字元(20)		
電子信箱	字元(50)		
性別	字元(1)		女

```
alter table 學生表 add 性別 char(1) default '女';
```

資料定義語言DDL

- Alter table 修改欄位：
  - 將原來學生表中的「地址」之資料型態修改為50字元，且不能為空值。

學生表			
欄位名稱	資料型態	空值	預設值
學號(PK)	字元(8)		
姓名	字元(4)	NOT NULL	
電話	字元(12)		
地址	字元(50)	NOT NULL	
電子信箱	字元(50)		
性別	字元(1)		女

```
alter table 學生表
modify 地址 char(50) NOT NULL;
```

資料定義語言DDL

- Alter table 刪除欄位：
  - 將學生表中的「電子信箱」欄位刪除。

學生表			
欄位名稱	資料型態	空值	預設值
學號(PK)	字元(8)		
姓名	字元(4)	NOT NULL	
電話	字元(12)		
地址	字元(20)		
電子信箱	字元(50)		
性別	字元(1)		女

```
alter table 學生表
  drop 電子信箱;
```

- 注意：一旦刪除，資料是無法回復的。

資料定義語言DDL

- Drop table(刪除資料表)基本語法：
  - 直接刪除整個資料表，包含定義及所有資料，且無法回復。
  - 資料表必須沒有被其它子關聯表參考時才可被刪除。

```
drop table 選課表;
```

學生表

欄位	資料型態	空值	預設值
學號(PK)	字元(8)		
姓名	字元(4)	NOT NULL	
電話	字元(12)		
地址	字元(20)		

課程表

欄位	資料型態	空值	預設值
課號(PK)	字元(5)		
課名	字元(20)	NOT NULL	
學分數	整數		3
必選修	字元(2)		

選課表

欄位	資料型態	空值	預設值
學號(PK)	字元(8)		
課號	字元(5)		
成績	整數	NOT NULL	
選課日期	日期		系統時間

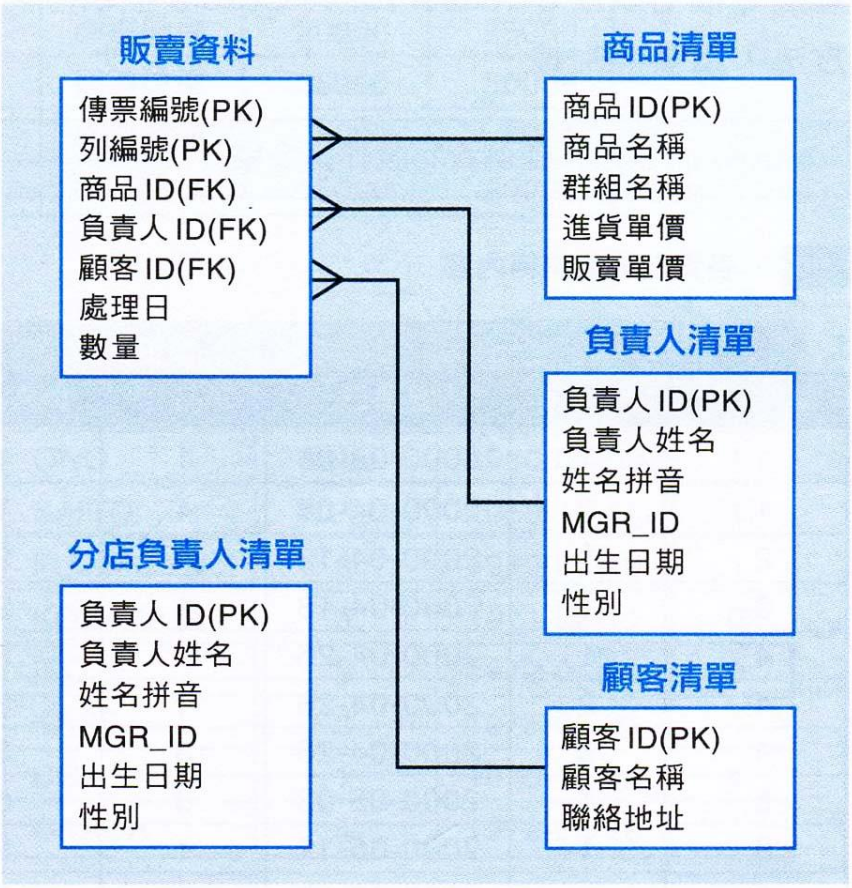
休息一下~

資料操作語言DML

- 透過DML指令，可以對資料表的紀錄進行查詢、新增、修改、刪除等動作。
- 四種基本指令：
  - SELECT(查詢)
  - INSERT(新增)
  - UPDATE(修改)
  - DELETE(刪除)
- DDL指令通常只在建立資料庫、資料表時用到，之後的操作幾乎都是這四個DML指令。

建立一個練習資料庫

- 為方便接下來的許多練習，我們需要一個已知紀錄內容的練習資料庫，這樣才知道指令執行結果是否正確。
- 請依ER圖建立資料庫：
  - 資料庫名稱：
    - 我的商店
  - 資料表：
    - 1. 販賣資料
    - 2. 商品清單
    - 3. 負責人清單
    - 4. 分店負責人清單
    - 5. 顧客清單



建立一個練習資料庫

- 先確認資料庫是否已存在，若有(可能是之前練習的)，先刪除再重建。

```
drop database if exists 我的商店;
create database 我的商店;
use 我的商店;
```

- 刪掉「我的商店」資料庫後再新增，並指定使用這個資料庫(因為MySQL裡面可能有多個資料庫，必須指定你要使用哪一個)。

建立一個練習資料庫

- 建立「販賣資料」表：

販賣資料			
欄位名稱	資料型態	空值	預設值
傳票編號(PK)	字元(5)	NOT NULL	
列編號(PK)	字元(5)	NOT NULL	
處理日	日期時間		
商品ID(FK)	字元(5)		
負責人ID(FK)	字元(5)		
顧客ID(FK)	字元(5)		
數量	整數		

```
create table 販賣資料(
    傳票編號 char(5) NOT NULL,
    列編號 char(5) NOT NULL,
    處理日 date,
    商品ID char(5),
    負責人ID char(5),
    顧客ID char(5),
    數量 int,
    primary key (傳票編號,列編號),
    foreign key (商品ID) references 商品清單 (商品ID),
    foreign key (負責人ID) references 負責人清單 (負責人ID),
    foreign key (顧客ID) references 顧客清單 (顧客ID)
);
```

建立一個練習資料庫

- 建立「商品清單」表：

```
create table 商品清單(
    商品ID char(5),
    商品名稱 char(20),
    群組名稱 char(10),
    進貨單價 int,
    販賣單價 int,
    primary key (商品ID)
);
```

商品清單			
欄位名稱	資料型態	空值	預設值
商品ID(PK)	字元(5)	NOT NULL	
商品名稱	字元(20)		
群組名稱	字元(10)		
進貨單價	整數		
販賣單價	整數		

建立一個練習資料庫

- 建立「負責人清單」表：

```
create table 負責人清單(
    負責人ID char(5),
    負責人姓名 char(20),
    姓名拼音 char(20),
    MGR_ID char(5),
    出生日期 date,
    性別 char(2),
    primary key (負責人ID)
);
```

負責人清單			
欄位名稱	資料型態	空值	預設值
負責人ID(PK)	字元(5)	NOT NULL	
負責人姓名	字元(20)		
姓名拼音	字元(20)		
MGR_ID	字元(5)		
出生日期	日期時間		
性別	字元(2)		

建立一個練習資料庫

- 建立「分店負責人清單」表：

```
create table 分店負責人清單(
    分店負責人ID char(5),
    分店負責人姓名 char(20),
    姓名拼音 char(20),
    MGR_ID char(5),
    出生日期 date,
    性別 char(2),
    primary key (分店負責人ID)
);
```

分店負責人清單			
欄位名稱	資料型態	空值	預設值
分店負責人ID(PK)	字元(5)	NOT NULL	
分店負責人姓名	字元(20)		
姓名拼音	字元(20)		
MGR_ID	字元(5)		
出生日期	日期時間		
性別	字元(2)		

建立一個練習資料庫

- 建立「顧客清單」表：

```
create table 顧客清單(
    顧客ID char(5),
    顧客名稱 char(20),
    聯絡地址 char(20),
    primary key (顧客ID)
);
```

顧客清單			
欄位名稱	資料型態	空值	預設值
顧客ID(PK)	字元(5)	NOT NULL	
顧客名稱	字元(20)		
聯絡地址	字元(20)		

建立一個練習資料庫

- 建立資料庫及資料表：

```
drop database if exists 我的商店;
create database 我的商店;
use 我的商店;

create table 顧客清單(
    顧客ID char(5),
    顧客名稱 char(20),
    聯絡地址 char(20),
    primary key (顧客ID)
);

create table 分店負責人清單(
    分店負責人ID char(5),
    分店負責人姓名 char(20),
    姓名拼音 char(20),
    MGR_ID char(5),
    出生日期 date,
    性別 char(2),
    primary key (分店負責人ID)
);
```

```
create table 負責人清單(
    負責人ID char(5),
    負責人姓名 char(20),
    姓名拼音 char(20),
    MGR_ID char(5),
    出生日期 date,
    性別 char(2),
    primary key (負責人ID)
);

create table 商品清單(
    商品ID char(5),
    商品名稱 char(20),
    群組名稱 char(10),
    進貨單價 int,
    販賣單價 int,
    primary key (商品ID)
);
```

建立一個練習資料庫

- 建立資料庫及資料表：

```
create table 販賣資料(
    傳票編號 char(5),
    列編號 char(5),
    處理日 date,
    商品ID char(5),
    負責人ID char(5),
    顧客ID char(5),
    數量 int,
    primary key (傳票編號,列編號),
    foreign key (商品ID) references 商品清單 (商品ID),
    foreign key (負責人ID) references 負責人清單 (負責人ID),
    foreign key (顧客ID) references 顧客清單 (顧客ID)
);
```

- 插入新資料請參閱附檔。

建立一個練習資料庫

- 紀錄內容：1.販賣資料

1. 販賣資料

傳票編號	列編號	處理日	商品ID	負責人ID	顧客ID	數量
1	1	2000-04-06	1	1	2	3
1	2	2000-04-06	4	1	2	3
2	1	2000-04-12	1	2	1	1
3	1	2000-04-18	1	2	2	1
4	1	2000-04-26	2	3	4	1
4	2	2000-04-26	7	3	4	1
4	3	2000-04-26	8	3	4	1
5	1	2000-05-08	3	6	1	3
6	1	2000-05-12	1	2	5	1
6	2	2000-05-12	3	2	5	2
7	1	2000-05-19	2	5	4	1
8	1	2000-05-22	2	6	1	1
9	1	2000-05-25	5	8	2	5
10	1	2000-06-02	5	2	1	1
11	1	2000-06-06	2	3	3	2
11	2	2000-06-06	10	3	3	1
12	1	2000-06-12	2	6	2	1
13	1	2000-06-15	9	7	5	5
13	2	2000-06-15	2	7	5	2
13	3	2000-06-15	10	7	5	1

建立一個練習資料庫

- 紀錄內容：2. 商品清單

2. 商品清單

商品 ID	商品名稱	群組名稱	進貨單價	販賣單價
1	桌上型電腦	電腦主機	150000	180000
2	筆記型電腦	電腦主機	230000	270000
3	17 吋螢幕	周邊設備	40000	50000
4	19 吋螢幕	周邊設備	80000	95000
5	15 吋液晶螢幕	周邊設備	100000	120000
6	數位相機	周邊設備	NULL	NULL
7	印表機	周邊設備	20000	25000
8	掃描器	周邊設備	25000	30000
9	HUB	網路設備	5000	7000
10	網路卡	網路設備	15000	20000

建立一個練習資料庫

- 紀錄內容：3. 負責人清單

3. 負責人清單

負責人 ID	負責人姓名	姓名拼音	MGR_ID	出生日期	性別
1	鈴木	SUSUKI	NULL	1960-01-23	1
2	小野	ONO	NULL	1960-08-02	1
3	齋藤	SAITO	NULL	1963-10-15	1
4	藤本	FUJIMOTO	3	1972-07-18	1
5	小林	KOBAYASHI	3	1971-02-11	0
6	伊藤	ITO	2	1972-04-01	0
7	佐瀨	SASE	2	1975-02-21	1
8	宇賀神	UGAJIN	1	1975-12-22	1
9	岡田	OKADA	4	1972-03-18	1

建立一個練習資料庫

- 紀錄內容：4. 分店負責人清單

4. 分店負責人清單

分店負責人 ID	分店負責人姓名	姓名拼音	MGR_ID	出生日期	性別
4	藤本	FUJIMOTO	NULL	1972-07-18	1
9	岡田	OKADA	4	1972-03-18	1
10	田中	TANAKA	9	1975-05-23	1
11	井上	INOUE	9	1980-02-18	1
12	佐佐木	SASAKI	9	1968-10-10	1

- 紀錄內容：5. 顧客清單

5. 顧客清單

顧客 ID	顧客名稱	聯絡地址
1	Waikiki 軟體	090-AAAA-AAAA
2	鈴木商事	090-BBBB-BBBB
3	齋藤模型店	090-CCCC-CCCC
4	MicroHard	090-DDDD-DDDD
5	Lanru	090-EEEE-EEEE

建立一個練習資料庫

- 在操作資料庫的過程中，對每個資料表的內容、性質等都要很清楚。

販賣資料

欄位名稱	資料型態	空值	預設值
傳票編號(PK)	字元(5)	NOT NULL	
列編號(PK)	字元(5)	NOT NULL	
處理日	日期時間		
商品ID(FK)	字元(5)		
負責人ID(FK)	字元(5)		
顧客ID(FK)	字元(5)		
數量	整數		

商品清單

欄位名稱	資料型態	空值	預設值
商品ID(PK)	字元(5)	NOT NULL	
商品名稱	字元(20)		
群組名稱	字元(10)		
進貨單價	整數		
販賣單價	整數		

顧客清單

欄位名稱	資料型態	空值	預設值
顧客ID(PK)	字元(5)	NOT NULL	
顧客名稱	字元(20)		
聯絡地址	字元(20)		

分店負責人清單

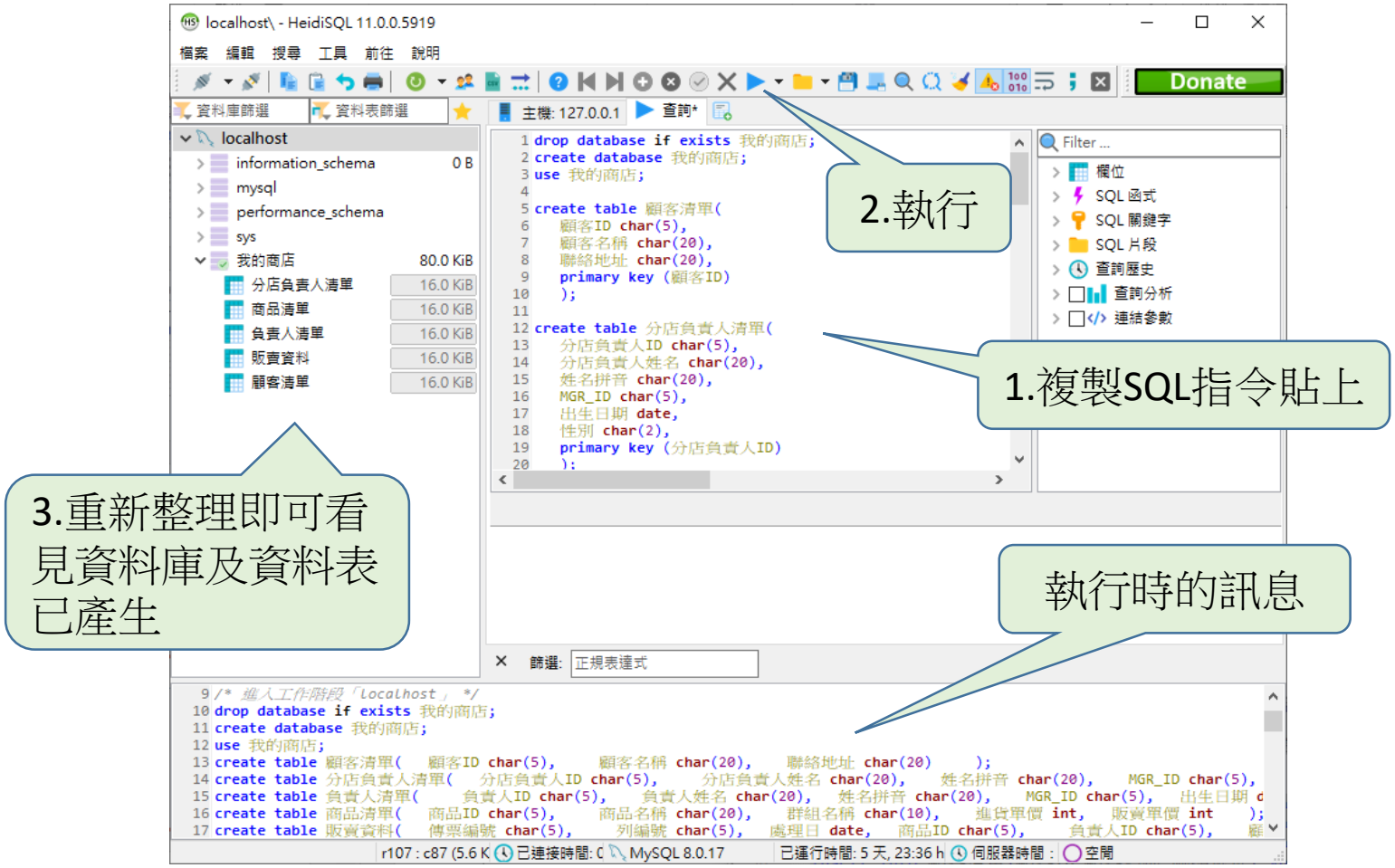
欄位名稱	資料型態	空值	預設值
分店負責人ID(PK)	字元(5)	NOT NULL	
分店負責人姓名	字元(20)		
姓名拼音	字元(20)		
MGR_ID	字元(5)		
出生日期	日期時間		
性別	字元(2)		

負責人清單

欄位名稱	資料型態	空值	預設值
負責人ID(PK)	字元(5)	NOT NULL	
負責人姓名	字元(20)		
姓名拼音	字元(20)		
MGR_ID	字元(5)		
出生日期	日期時間		
性別	字元(2)		

建立一個練習資料庫

- 將檔案mysql_create_table.sql內容複製到HeiDiSQL執行。



SELECT語法基本格式

- 這是SQL基本的搜尋功能，是從資料庫中找出我們要的資料。
- 「搞定了SELECT與法就等於搞定了SQL」！
- 基本格式：

```
SELECT  欄位名稱
FROM    資料表
WHERE   條件
```

SELECT語法基礎

- 取得商品清單的全部資料，「*」代表所有欄位。
- 欄位順序會依資料表格式順序。

```
SELECT * FROM 商品清單
```

商品清單 (10r × 5c)

商品ID	商品名稱	群組名稱	進貨單價	販賣單價
1	桌上型電腦	電腦主機	150,000	180,000
10	網路卡	網路設備	15,000	20,000
2	筆記型電腦	電腦主機	230,000	270,000
3	17吋螢幕	周邊設備	40,000	50,000
4	19吋螢幕	周邊設備	80,000	95,000
5	15吋液晶螢幕	周邊設備	100,000	120,000
6	數位相機	周邊設備	(NULL)	(NULL)
7	印表機	周邊設備	20,000	25,000
8	掃描器	周邊設備	25,000	30,000
9	HUB	網路設備	5,000	7,000

SELECT語法基礎

- LIMIT敘述限制輸出數目：取得商品清單的前5個輸出資料。

```
SELECT *
FROM 商品清單 LIMIT 5
```

商品清單 (5r × 5c)

商品ID	商品名稱	群組名稱	進貨單價	販賣單價
1	桌上型電腦	電腦主機	150,000	180,000
10	網路卡	網路設備	15,000	20,000
2	筆記型電腦	電腦主機	230,000	270,000
3	17吋螢幕	周邊設備	40,000	50,000
4	19吋螢幕	周邊設備	80,000	95,000

SELECT語法基礎

- 取得資料表的某一個欄位全部資料(指定單一欄位)

```
SELECT 商品名稱 FROM 商品清單
```

商品清單 (10r × 1c)

商品名稱
桌上型電腦
網路卡
筆記型電腦
17吋螢幕
19吋螢幕
15吋液晶螢幕
數位相機
印表機
掃描器
HUB

SELECT語法基礎

- 取得資料表某些欄位全部資料(指定多個欄位)。
- 欄位順序依SELECT指令順序。

```
SELECT 商品ID,群組名稱,商品名稱,進貨單價,販賣單價
FROM 商品清單
```

商品清單 (10r × 5c)

商品ID	群組名稱	商品名稱	進貨單價	販賣單價
1	電腦主機	桌上型電腦	150,000	180,000
10	網路設備	網路卡	15,000	20,000
2	電腦主機	筆記型電腦	230,000	270,000
3	周邊設備	17吋螢幕	40,000	50,000
4	周邊設備	19吋螢幕	80,000	95,000
5	周邊設備	15吋液晶螢幕	100,000	120,000
6	周邊設備	數位相機	(NULL)	(NULL)
7	周邊設備	印表機	20,000	25,000
8	周邊設備	掃描器	25,000	30,000
9	網路設備	HUB	5,000	7,000

SELECT語法基礎

- 指定搜尋條件：取得販賣單價為100000元以下的商品名稱。

```
SELECT 商品名稱
FROM 商品清單
WHERE 販賣單價 <= 100000
```

>	大於
<	小於
>=	大於等於
<=	小於等於
=	等於
<> 或 !=	不等於

商品清單 (6r × 1c)

商品名稱
網路卡
17吋螢幕
19吋螢幕
印表機
掃描器
HUB

SELECT語法基礎

- 指定搜尋條件(字串)：取得群組名稱為「週邊設備」的商品名稱。

```
SELECT 商品名稱
FROM 商品清單
WHERE 群組名稱 = ‘周邊設備’
```

(單引號)

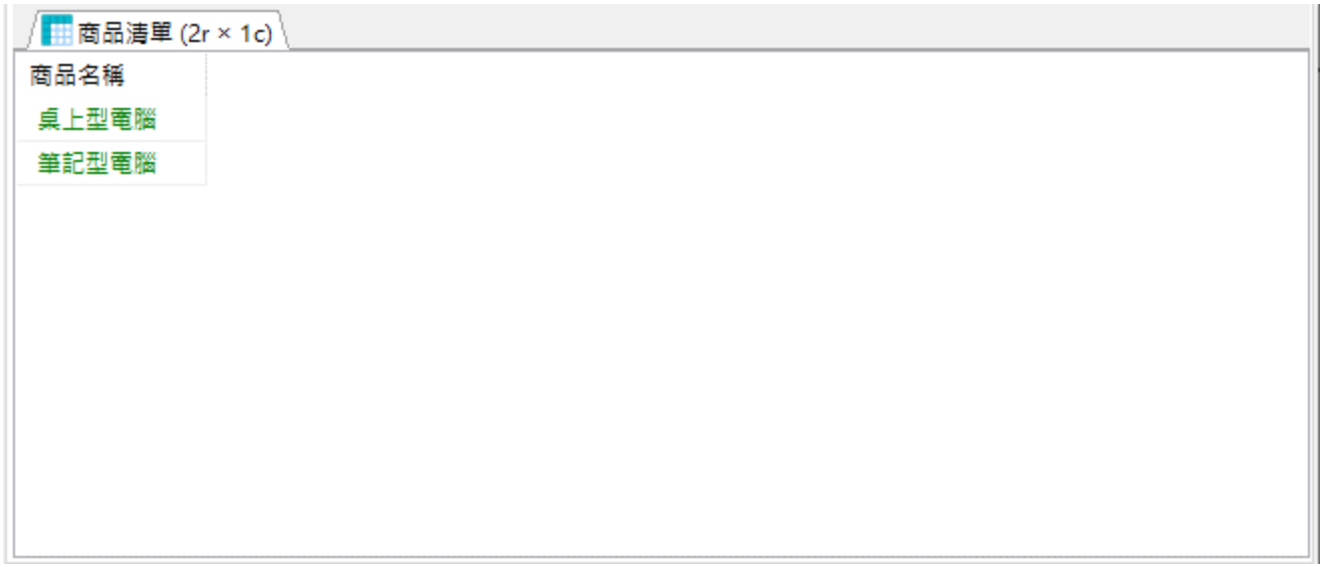
商品清單 (6r × 1c)

商品名稱
17吋螢幕
19吋螢幕
15吋液晶螢幕
數位相機
印表機
掃描器

SELECT語法基礎

- 指定搜尋條件(字串、符合條件)：取得商品名稱中含有「電腦」的商品名稱。

```
SELECT 商品名稱
FROM 商品清單
WHERE 商品名稱 LIKE '%電腦%'
```



SELECT語法基礎

- 當要搜尋的字串需完全符合時用「=」號，例如：

```
群組名稱 = '周邊設備'
```

- 當要搜尋的是部分字串符合時，要用「LIKE」指令加上「%」符號，例如：

```
商品名稱 LIKE '%電腦%'
```

電腦%	查詢以電腦開頭的字串
%電腦%	查詢字串中有包含電腦的字串
%電腦	查詢以電腦結束的字串

SELECT語法基礎

- 指定搜尋條件(日期)：取得四月份的販賣資料。

```
SELECT * FROM 販賣資料
WHERE 處理日 >= '2000-04-01' AND 處理日 <= '2000-04-30'
```

日期格式寫法

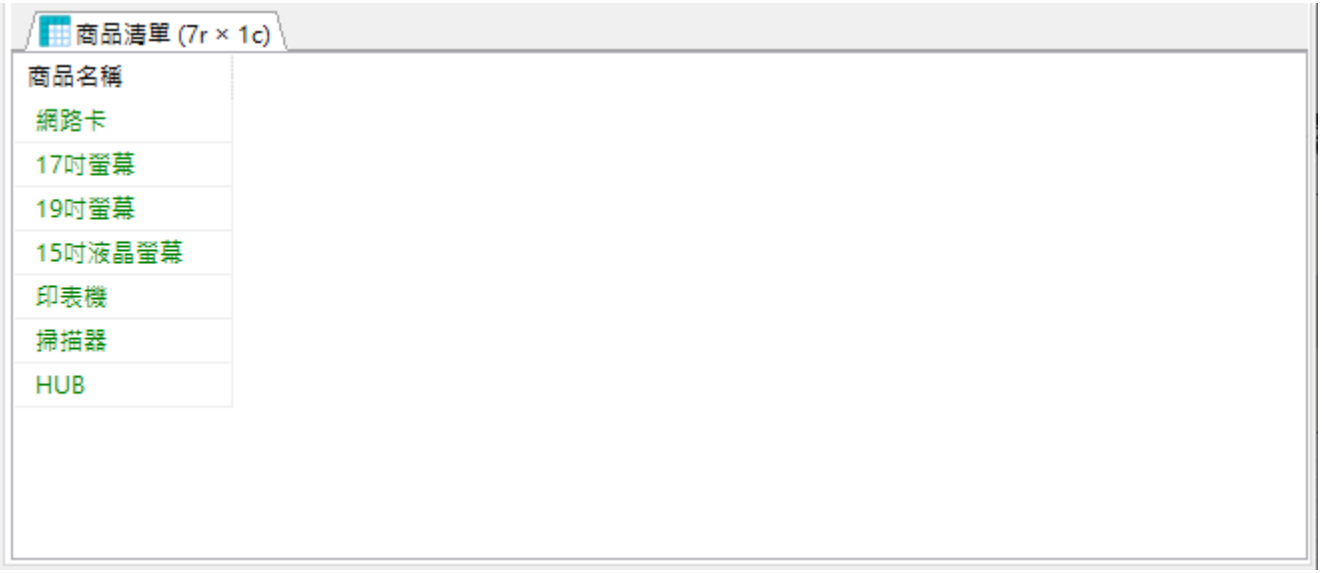
販賣資料 (7r × 7c)

傳票編號	列編號	處理日	商品ID	負責人ID	顧客ID	數量
1	1	2000-04-06	1	1	2	3
1	2	2000-04-06	4	1	2	3
2	1	2000-04-12	1	2	1	1
3	1	2000-04-18	1	2	2	1
4	1	2000-04-26	2	3	4	1
4	2	2000-04-26	7	3	4	1
4	3	2000-04-26	8	3	4	1

SELECT語法基礎

- 指定複數條件(AND、OR)：取得群組名稱為周邊設備或網路設備，且進貨單價為100000元以下的商品名稱。
- AND的優先權比OR高，所以OR結合的條件要框起來。

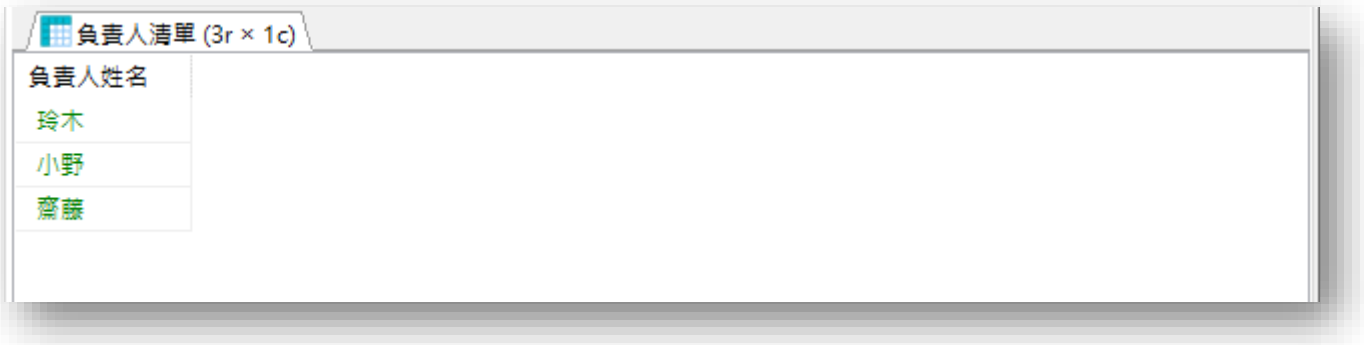
```
SELECT 商品名稱 FROM 商品清單
WHERE (群組名稱 = '周邊設備' OR 群組名稱 = '網路設備')
      AND 進貨單價 <= 100000
```



SELECT語法基礎

- IN的使用方法：取得負責人ID為1、2、3的負責人姓名。
- IN是指之後括號內符合任何一項即可。

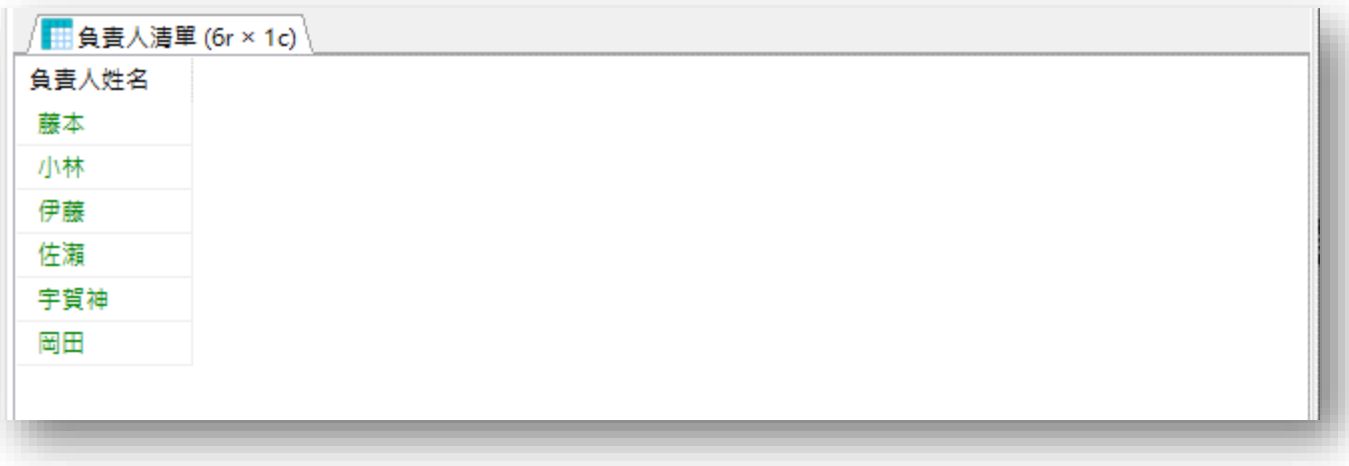
```
SELECT 負責人姓名
FROM 負責人清單
WHERE 負責人ID IN (1,2,3)
```



SELECT語法基礎

- NOT IN的使用方法：取得負責人ID不是1、2、3的負責人姓名。
- NOT IN是指不包含在之後括號內任何一項即可。
- IN與NOT IN是相反的。

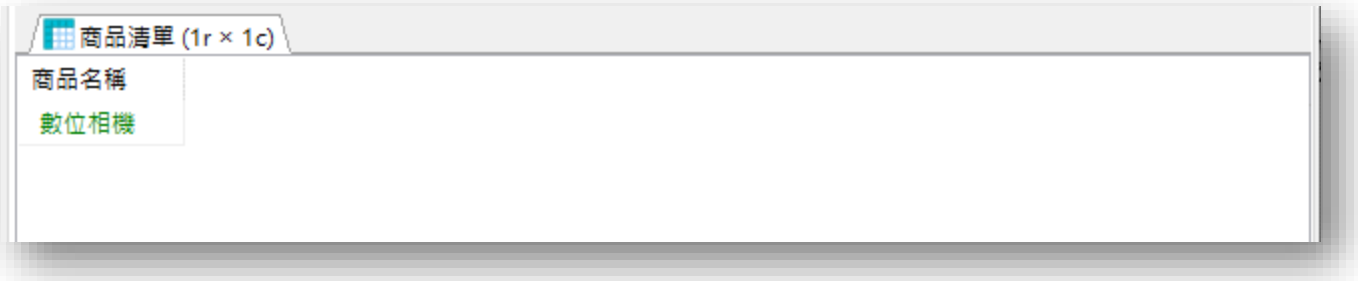
```
SELECT 負責人姓名
FROM 負責人清單
WHERE 負責人ID NOT IN (1,2,3)
```



SELECT語法基礎

- NULL的概念(IS的使用方法)：取得沒有設定販賣單價的商品名稱(即取得販賣單價為NULL的商品)。
- 不可以用 `WHERE 販賣單價 = NULL` 這樣的語法，因為「=、!=、<>」一定要跟確定值比較，而NULL不算確定值，因為它代表「沒有」。

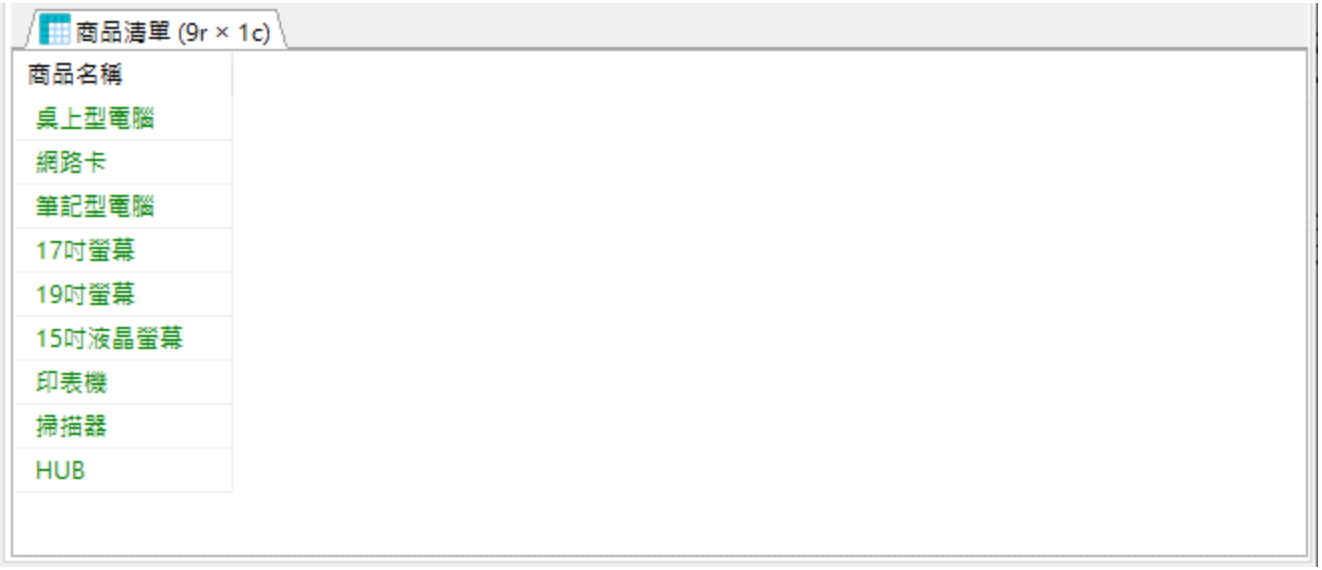
```
SELECT 商品名稱
FROM 商品清單
WHERE 販賣單價 IS NULL
```



SELECT語法基礎

- NULL的概念(IS NOT的使用方法)：取得已設定販賣單價的商品名稱(即取得販賣單價不為NULL的商品)。
- 跟前面相反，不是NULL就在IS前面加上否定運算NOT。

```
SELECT 商品名稱 FROM 商品清單
WHERE 販賣單價 IS NOT NULL
```



SELECT語法基礎

- 複合數個資料表的結合：結合販賣資料與商品清單，並取得處理日、商品ID、商品名稱的列表。

```
SELECT 販賣資料.處理日, 商品清單.商品ID, 商品清單.商品名稱
FROM 販賣資料, 商品清單
WHERE 販賣資料.商品ID = 商品清單.商品ID
```

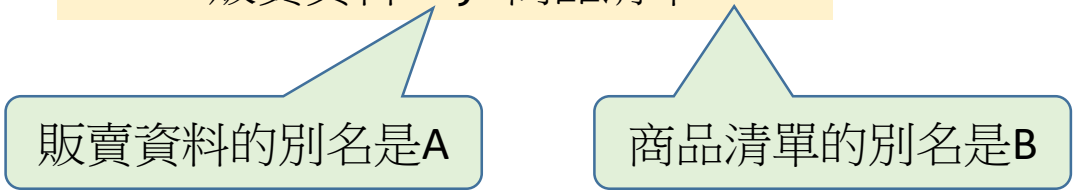
結果 #1 (20r × 3c)

處理日	商品ID	商品名稱
2000-04-06	1	桌上型電腦
2000-04-06	4	19吋螢幕
2000-06-02	5	15吋液晶螢幕
2000-06-06	2	筆記型電腦
2000-06-06	10	網路卡
2000-06-12	2	筆記型電腦
2000-06-15	9	HUB
2000-06-15	2	筆記型電腦
2000-06-15	10	網路卡
2000-04-12	1	桌上型電腦
2000-04-18	1	桌上型電腦
2000-04-26	2	筆記型電腦
2000-04-26	7	印表機
2000-04-26	8	掃描器
2000-05-08	3	17吋螢幕
2000-05-12	1	桌上型電腦
2000-05-12	3	17吋螢幕
2000-05-19	2	筆記型電腦
2000-05-22	2	筆記型電腦
2000-05-25	5	15吋液晶螢幕

SELECT語法基礎

- 複合數個資料表的結合：前頁範例也可以改寫成如下，結果是一樣的。
- 在「FROM 資料表」後面給一個「別名」，之後在整個敘述中就可以用這個別名代表資料表，例如：

```
FROM 販賣資料 A, 商品清單 B
```



- 下列執行結果和前頁相同：

```
SELECT A.處理日, B.商品ID, B.商品名稱
FROM 販賣資料 A, 商品清單 B
WHERE A.商品ID = B.商品ID
```

SELECT語法基礎

- 練習：
- 1.取得由誰販賣(姓名)、何時、賣掉甚麼(商品名稱)的列表。

- 2.取得負責人「鈴木」何時、賣掉甚麼(商品名稱)的列表。

- 3.取得Waikiki軟體於5月份所購入之周邊設備與商品名稱、負責人名稱的列表。

SELECT語法基礎

- 練習(解答)：
- 1.

```
SELECT C.負責人姓名,
       A.處理日,
       B.商品名稱
FROM 販賣資料 A,
     商品清單 B,
     負責人清單 C
WHERE A.商品ID = B.商品ID
  AND A.負責人ID = C.負責人ID
```

販賣資料 (20r × 3c)

負責人姓名	處理日	商品名稱
玲木	2000-04-06	桌上型電腦
玲木	2000-04-06	19吋螢幕
小野	2000-06-02	15吋液晶螢幕
齋藤	2000-06-06	筆記型電腦
齋藤	2000-06-06	網路卡
伊藤	2000-06-12	筆記型電腦
佐瀨	2000-06-15	HUB
佐瀨	2000-06-15	筆記型電腦
佐瀨	2000-06-15	網路卡
小野	2000-04-12	桌上型電腦
小野	2000-04-18	桌上型電腦
齋藤	2000-04-26	筆記型電腦
齋藤	2000-04-26	印表機
齋藤	2000-04-26	掃描器
伊藤	2000-05-08	17吋螢幕
小野	2000-05-12	桌上型電腦
小野	2000-05-12	17吋螢幕
小林	2000-05-19	筆記型電腦
伊藤	2000-05-22	筆記型電腦
宇賀神	2000-05-25	15吋液晶螢幕

SELECT語法基礎

- 練習(解答)：
- 2.

```
SELECT A.處理日, B.商品名稱
FROM 販賣資料 A, 商品清單 B, 負責人清單 C
WHERE A.商品ID = B.商品ID AND A.負責人ID = C.負責人ID
      AND C.負責人姓名 = '玲木'
```

販賣資料 (2r × 2c)

處理日	商品名稱
2000-04-06	桌上型電腦
2000-04-06	19吋螢幕

SELECT語法基礎

- 練習(解答)：
- 3.

```
SELECT B.商品名稱, D.負責人姓名
FROM 販賣資料 A, 商品清單 B, 顧客清單 C, 負責人清單 D
WHERE A.商品ID = B.商品ID
  AND A.顧客ID = C.顧客ID
  AND A.負責人ID = D.負責人ID
  AND B.群組名稱 = '周邊設備'
  AND C.顧客名稱 = 'Waikiki軟體'
  AND A.處理日 >= '2000-05-01' AND A.處理日 <= '2000-05-31'
```

休息一下~

SELECT較複雜且能派上用場語法

- 指定顯示順序(ORDER BY)：以英文字母排序來輸出負責人姓名。
- 注意它是以「內碼」的順序排列，不是筆畫順序。

```
SELECT 姓名拼音, 負責人姓名
FROM 負責人清單
ORDER BY 姓名拼音
```

負責人清單 (9r × 2c)

姓名拼音	負責人姓名
FUJIMOTO	藤本
ITO	伊藤
KOBAYASHI	小林
OKADA	岡田
ONO	小野
SAITO	齋藤
SASE	佐瀨
SUSUKI	玲木
UGAJIN	宇賀神

SELECT較複雜且能派上用場語法

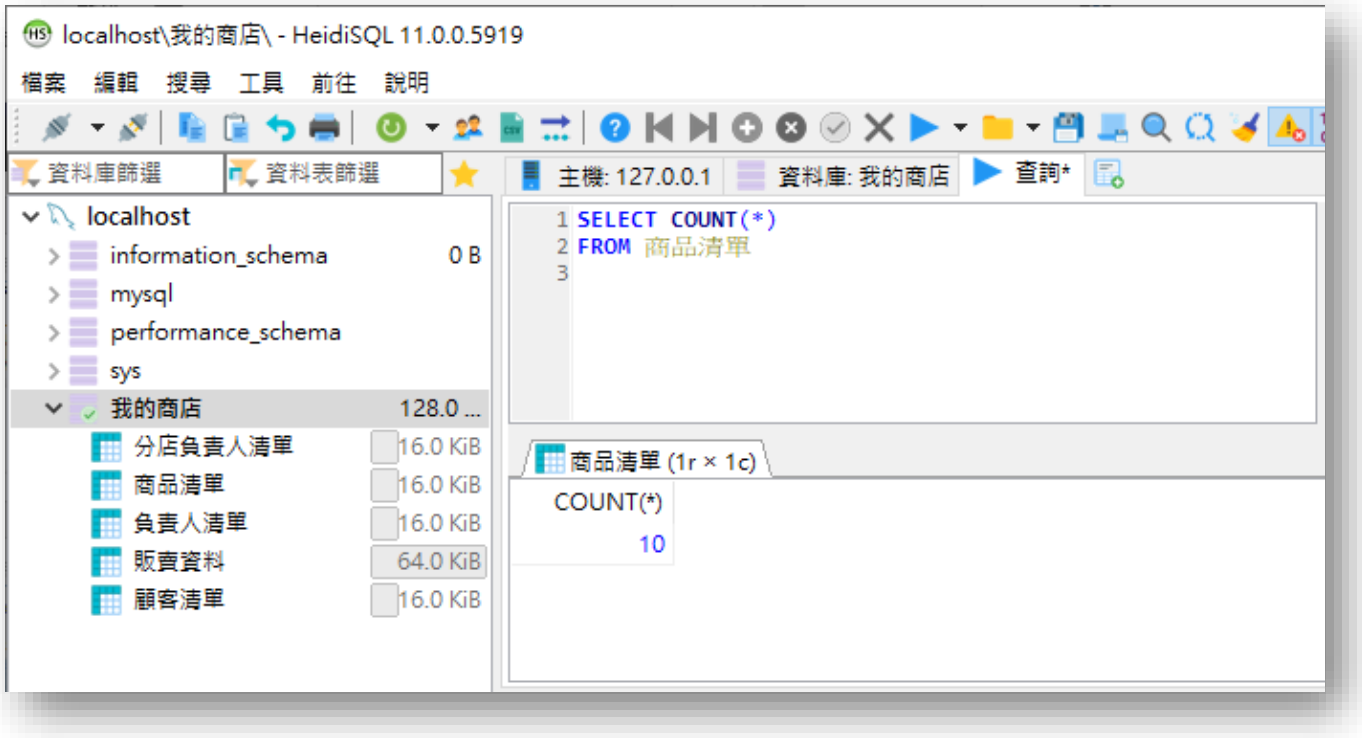
- ORDER BY說明：
  - 「ASC」是昇冪排序，可省略。「DESC」是降冪排序。
  - 可以多個欄位排序，第一欄位相同時再依第二欄位排序，依此類推。
  - 各欄位可以單獨指定昇冪或降冪。

ORDER BY 姓名拼音 ASC	由小到大排序，昇冪排序
ORDER BY 姓名拼音	同上，ASC可省略
ORDER BY 姓名拼音 DESC	由大到小排序，降冪排序
ORDER BY 姓名拼音, 負責人姓名	以姓名拼音昇冪排序，若有相同，則再依負責人姓名昇冪排序
ORDER BY 姓名拼音 DESC, 負責人姓名 ASC	以姓名拼音降冪排序，若有相同，則再依負責人姓名昇冪排序

SELECT較複雜且能派上用場語法

- 取得符合結果的資料數(COUNT())：取得商品清單的所有資料數。
- 注意：值為NULL的資料不會被記入。

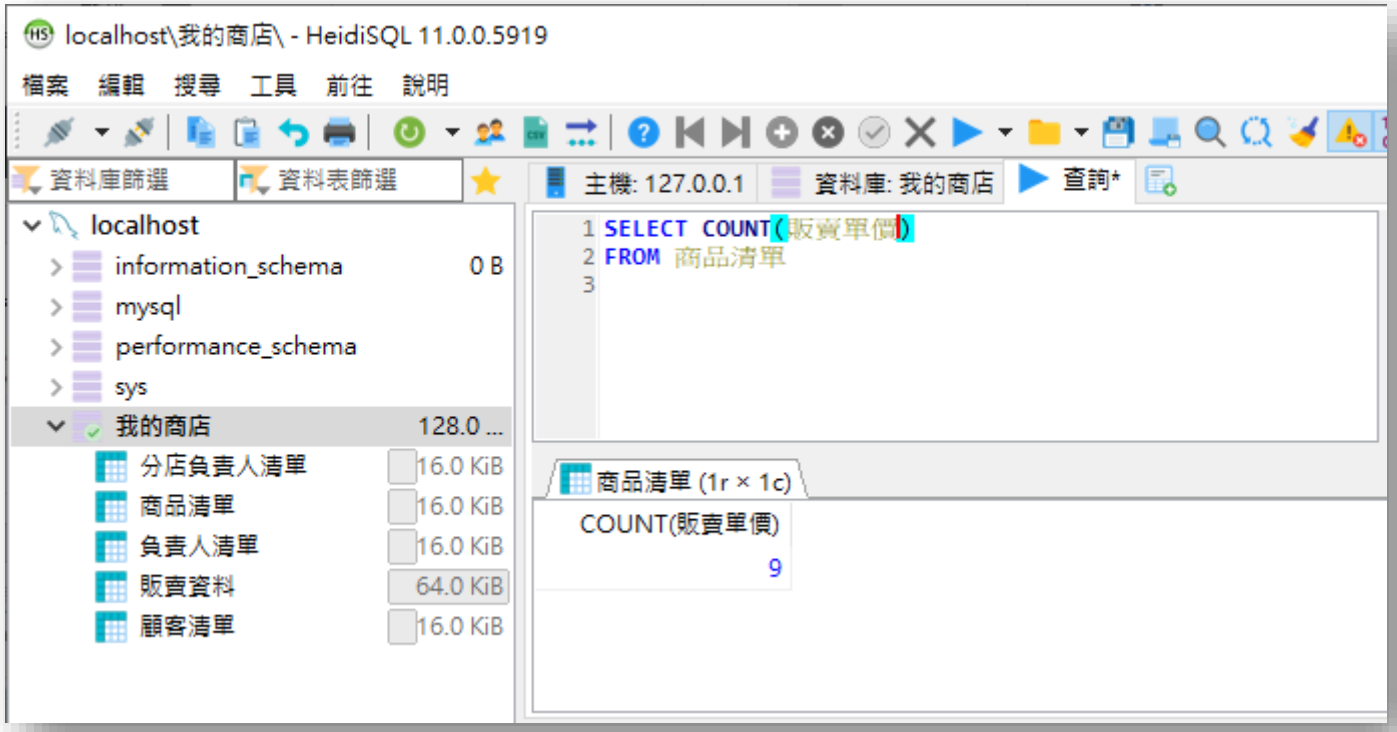
```
SELECT COUNT(*) FROM 商品清單
```



SELECT較複雜且能派上用場語法

- 取得符合結果的資料數(COUNT())：承前頁，若使用下列指令，則商品「數位相機」不會被列入，因為它的「販賣單價」是NULL，故只剩9筆資料。

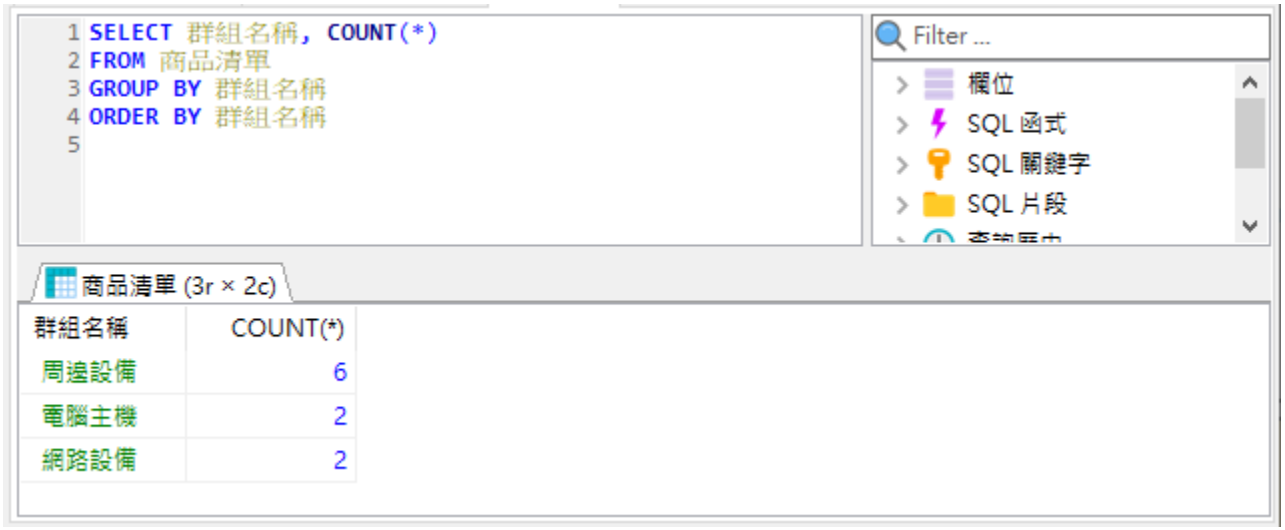
```
SELECT COUNT(販賣單價) FROM 商品清單
```



SELECT較複雜且能派上用場語法

- 群組化(GROUP BY)：針對每一個群組來從商品清單取得商品數量。

```
SELECT 群組名稱, COUNT(*)
FROM 商品清單
GROUP BY 群組名稱
ORDER BY 群組名稱
```



SELECT較複雜且能派上用場語法

- 群組化(GROUP BY)：
  - 一般來說，在資料庫中都會有些資料是可以群組化的，並需要進行統計，像是以月份、商品類型等，所以搞懂GROUP BY是很重要的。
  - GROUP BY的欄位通常也應該是要被SELECT的欄位，否則容易對結果產生意義不清。

拿掉「群組名稱」

```
SELECT        COUNT(*)
FROM 商品清單
GROUP BY 群組名稱
ORDER BY 群組名稱
```

那這些數字是甚麼意思呢?

商品清單 (3r × 1c)

COUNT(*)
6
2
2

SELECT較複雜且能派上用場語法

- GROUP BY敘述與SUM()：由販賣資料取得以商品名稱區別的販賣數量(依照群組名稱排序)

```
SELECT B.商品名稱, SUM(A.數量)
FROM 販賣資料 A, 商品清單 B
WHERE A.商品ID = B.商品ID
GROUP BY B.商品名稱
ORDER BY B.商品名稱
```

由SUM()函數直接計算加總，和GROUP BY敘述一起很好用。

販賣資料 (9r × 2c)

商品名稱	SUM(A.數量)
15吋液晶螢幕	6
17吋螢幕	5
19吋螢幕	3
HUB	5
印表機	1
桌上型電腦	6
掃描器	1
筆記型電腦	8
網路卡	2

SELECT較複雜且能派上用場語法

- HAVING敘述：由販賣資料取得以商品名稱區別的販賣數量(不過只要販賣數量超過5以上的商品)。

```
SELECT B.商品名稱, SUM(A.數量)
FROM 販賣資料 A, 商品清單 B
WHERE A.商品ID = B.商品ID
GROUP BY B.商品名稱
HAVING SUM(A.數量) >= 5
ORDER BY B.商品名稱
```

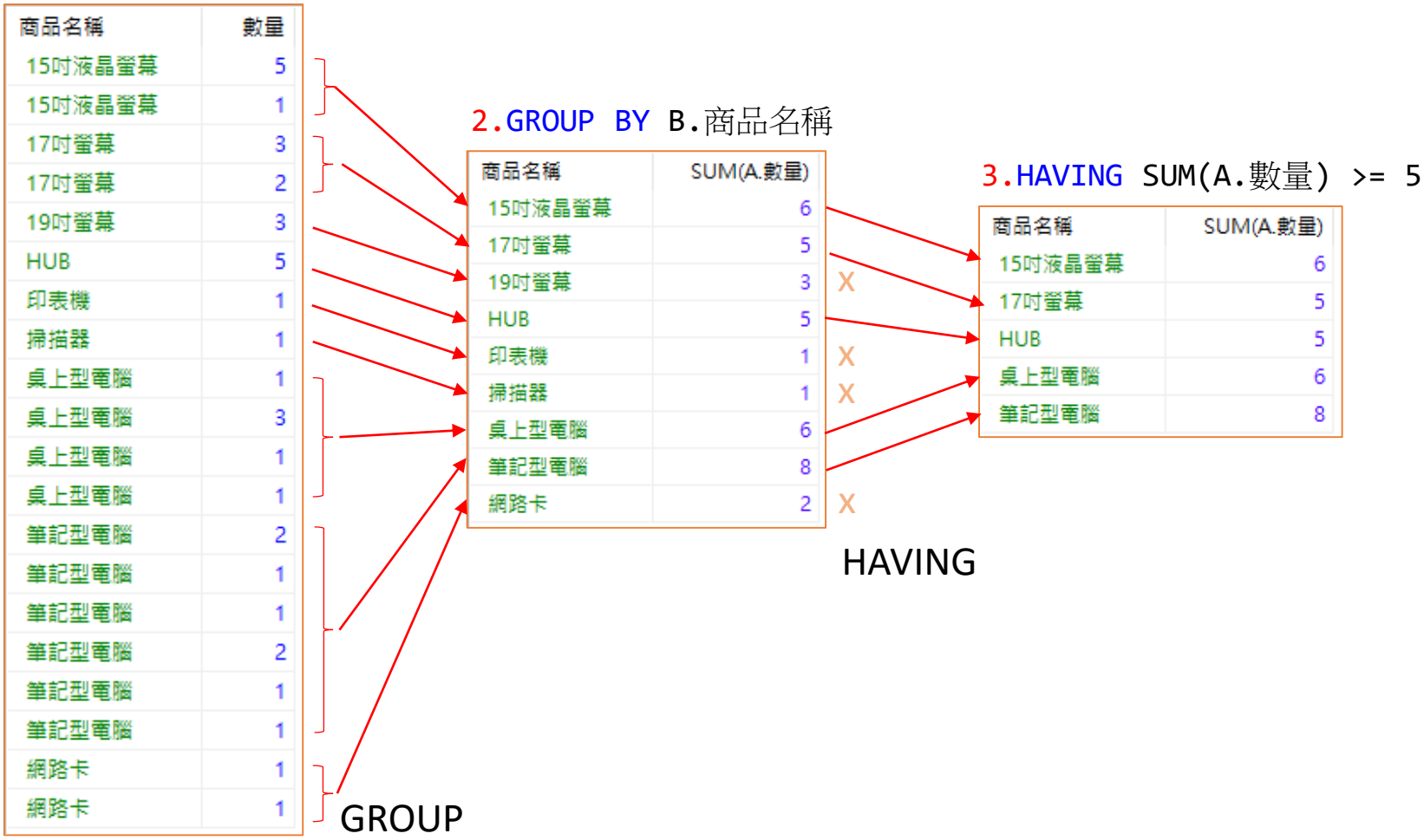
一定要先有GROUP BY群組化後才可以使用HAVING敘述設定條件。

販賣資料 (5r × 2c)

商品名稱	SUM(A.數量)
15吋液晶螢幕	6
17吋螢幕	5
HUB	5
桌上型電腦	6
筆記型電腦	8

SELECT較複雜且能派上用場語法

1. SELECT B.商品名稱,數量 FROM 販賣資料 A, 商品清單 B WHERE A.商品ID = B.商品ID



SELECT較複雜且能派上用場語法

- WHERE 與 HAVING：
  - WHERE是用來給SELECT下條件：

```
SELECT …  FROM …  WHERE …
```

  - HAVING是用來給GROUP BY下條件：

```
SELECT …  FROM …  GROUP BY …  HAVING …
```

SELECT較複雜且能派上用場語法

- DISTINCT敘述：取得4月份有賣出之商品的商品ID(但商品ID不得重複)

```
SELECT DISTINCT 商品ID
FROM 販賣資料
WHERE 處理日 >= ‘2000-04-01’ AND 處理日 <= ‘2000-04-30’
```

- DISTINCT會將重複的資料只輸出一筆。
- 通常不會和GROUP BY一起出現。

SELECT較複雜且能派上用場語法

- MAX()、MIN()、AVG()：取得販賣單價的最大值、最小值、平均值。

```
SELECT MAX(販賣單價), MIN(販賣單價), AVG(販賣單價)
FROM 商品清單
```

商品清單 (1r × 3c)

MAX(販賣單價)	MIN(販賣單價)	AVG(販賣單價)
270,000	7,000	88,555.5556

- 以上函式會把NULL值排除不計入。
- MAX()、MIN()也可以用在日期資料。

SELECT較複雜且能派上用場語法

- 顯示欄位的名稱可以自行命名：同前例，將結果欄位名稱顯示為最高單價、最低單價、平均單價。

```
SELECT MAX(販賣單價) “最高單價”, MIN(販賣單價) “最低單價”,
       AVG(販賣單價) “平均單價”
FROM 商品清單
```

商品清單 (1r × 3c)

最高單價	最低單價	平均單價
270,000	7,000	88,555.5556

- 在SELECT的欄位後面加上雙引號”” 或單引號’’ 字串即可，通常會替計算或函式的結果加上適當的欄位名稱。

SELECT較複雜且能派上用場語法

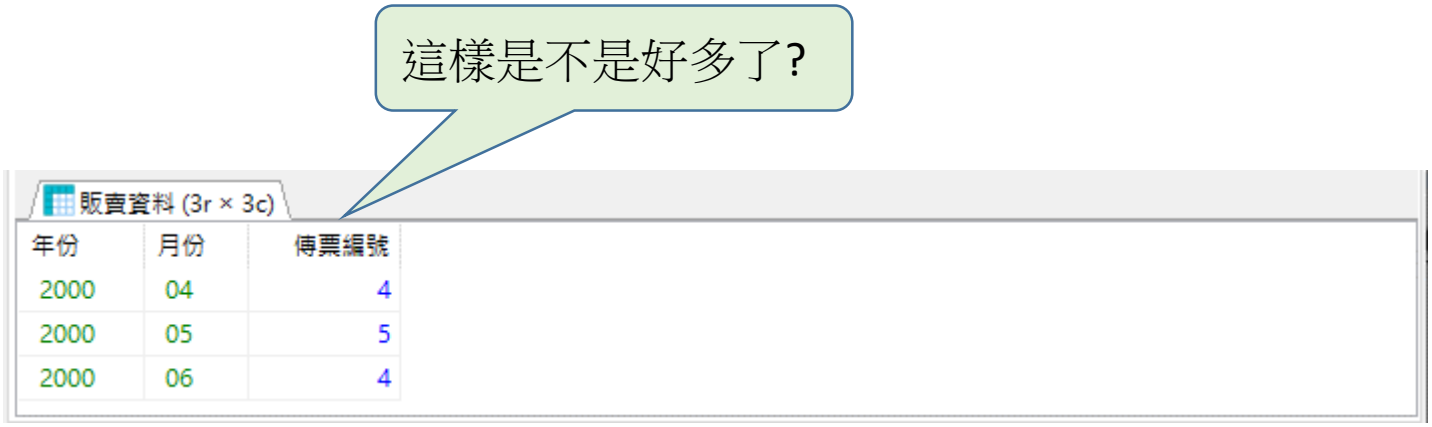
- 日期的加工：輸出月分別傳票。
  - 利用DATE_FORMAT()函式取得年、月、日等資訊。

```
SELECT DATE_FORMAT(處理日,'%Y'), DATE_FORMAT(處理日,'%m'),
       COUNT(DISTINCT 傳票編號)
FROM 販賣資料
GROUP BY DATE_FORMAT(處理日,'%Y'), DATE_FORMAT(處理日,'%m')
```

SELECT較複雜且能派上用場語法

- 日期的加工：將輸出加上適當的欄位名稱

```
SELECT DATE_FORMAT(處理日, '%Y') '年份',
       DATE_FORMAT(處理日, '%m') '月份',
       COUNT(DISTINCT 傳票編號) '傳票編號'
FROM 販賣資料
GROUP BY DATE_FORMAT(處理日,'%Y'), DATE_FORMAT(處理日,'%m')
```



SELECT較複雜且能派上用場語法

- DATE_FORMAT(date,format)函式的參數：
  - 根據format字串格式化date值。下列修飾符可以被用在format字串中。

%M 月名字(January……December)
%W 星期名字(Sunday……Saturday)
%D 有英語首碼的月份的日期(1st, 2nd, 3rd, 等等。)
%Y 年, 數字, 4 位元
%y 年, 數字, 2 位元
%a 縮寫的星期名字(Sun……Sat)
%d 月份中的天數, 數字(00……31)
%e 月份中的天數, 數字(0……31)
%m 月, 數字(01……12)
%c 月, 數字(1……12)
%b 縮寫的月份名字(Jan……Dec)
%j 一年中的天數(001……366)

%H 小時(00……23)
%k 小時(0……23)
%h 小時(01……12)
%I 小時(01……12)
%l 小時(1……12)
%i 分鐘, 數字(00……59)
%r 時間,12 小時(hh:mm:ss [AP]M)
%T 時間,24 小時(hh:mm:ss)
%S 秒(00……59)
%s 秒(00……59)
%p AM或PM
%w 一個星期中的天數(0=Sunday…… 6=Saturday)
%U 星期(0……52), 這裡星期天是星期的第一天
%u 星期(0……52), 這裡星期一是星期的第一天
%% 一個文字“%”。

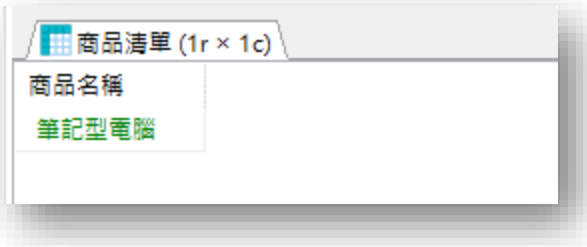
- 本函式只適用於MySQL，其它SQL Server寫法不同。

SELECT較複雜且能派上用場語法

- 子查詢：取得販賣單價最高之商品名稱。

```
SELECT 商品名稱
FROM 商品清單
WHERE 販賣單價 = (SELECT MAX(販賣單價) FROM 商品清單)
```

- 子查詢就是SELECT語法中還有一個SELECT，內側的查詢會先被執行，然後再將結果交給母查詢進行處理。

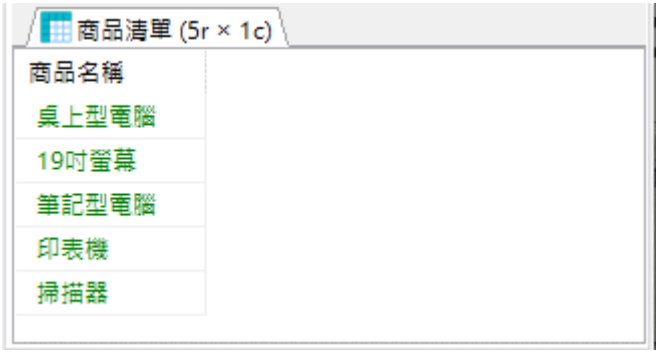


SELECT較複雜且能派上用場語法

- 子查詢：取得4月份有販賣之商品的商品名稱。

```
SELECT 商品名稱 FROM 商品清單
WHERE 商品ID IN
  (SELECT DISTINCT 商品ID FROM 販賣資料
   WHERE 處理日 >= '2000-04-01' AND 處理日 <= '2000-04-30')
```

- 本例當然可以不需要用子查詢的方式來寫，但要注意IN的使用，因為子查詢會傳回多個值，所以要用IN，不能用 =。



SELECT較複雜且能派上用場語法

- 練習：
- 1.取得4月份有賣出之商品的商品ID、商品名稱、負責人姓名(但商品ID不得重複)

- 2.請輸出單月的購買單價有500000以上的顧客的購買年份、月份，還有顧客名稱、購買單價合計。(購買單價合計是指 單價 * 數量)

SELECT較複雜且能派上用場語法

- 練習(解答)：
- 1.

```
SELECT DISTINCT A.商品ID, B.商品名稱, C.負責人姓名
FROM 販賣資料 A, 商品清單 B, 負責人清單 C
WHERE A.商品ID=B.商品ID AND A.負責人ID = C.負責人ID
AND (處理日 >= '2000-04-01' AND 處理日 <= '2000-04-30')
ORDER BY A.商品ID
```

販賣資料 (6r × 3c)

商品ID	商品名稱	負責人姓名
1	桌上型電腦	小野
1	桌上型電腦	玲木
2	筆記型電腦	齋藤
4	19吋螢幕	玲木
7	印表機	齋藤
8	掃描器	齋藤

SELECT較複雜且能派上用場語法

- 練習(解答)：
- 2.

```
SELECT DATE_FORMAT(A.處理日,'%Y'), DATE_FORMAT(A.處理日,'%c'),
            B.顧客名稱, SUM(C.販賣單價 * A.數量)
FROM 販賣資料 A, 顧客清單 B, 商品清單 C
WHERE A.顧客ID = B.顧客ID  AND  A.商品ID = C.商品ID
GROUP BY DATE_FORMAT(A.處理日,'%Y'), DATE_FORMAT(A.處理日,'%c'),
         B.顧客名稱
HAVING SUM(C.販賣單價 * A.數量) >= 500000
```

販賣資料 (4r × 4c)

DATE_FORMAT(A.處理日,'%Y')	DATE_FORMAT(A.處理日,'%c')	顧客名稱	SUM(C.販賣單價 * A.數量)
2000	4	玲木商事	1,005,000
2000	5	玲木商事	600,000
2000	6	Lanru	595,000
2000	6	齋藤模型店	560,000

休息一下~

SELECT進階語法

- EXISTS敘述：顯示有販賣業績的負責人之負責人ID及負責人姓名。EXISTS是檢查是否包含在其中的意思。

```
SELECT 負責人ID, 負責人姓名 FROM 負責人清單
WHERE EXISTS (
  SELECT * FROM 販賣資料
  WHERE 販賣資料.負責人ID = 負責人清單.負責人ID )
```

負責人清單 (7r × 2c)

負責人ID	負責人姓名
1	玲木
2	小野
3	齋藤
5	小林
6	伊藤
7	佐瀨
8	宇賀神

- 這道指令的白話文就是：「列出負責人清單中的負責人ID和負責人姓名，如果他們有出現在子查詢中的話」。

SELECT進階語法

- NOT EXISTS敘述：顯示沒有販賣業績的負責人之負責人ID及負責人姓名。

```
SELECT 負責人ID, 負責人姓名 FROM 負責人清單
WHERE NOT EXISTS (
   SELECT * FROM 販賣資料
   WHERE 販賣資料.負責人ID = 負責人清單.負責人ID )
```

負責人清單 (2r × 2c)

負責人ID	負責人姓名
4	藤本
9	岡田

- 與EXISTS相反，NOT EXISTS是不包含在其中意思。
- EXISTS與IN(NOT EXISTS與NOT IN)的結果會相同，但EXISTS敘述的效率較高。

SELECT進階語法

- SELECT語法中的條件分歧(CASE的使用)：

```
SELECT 負責人姓名 ,
  CASE
    WHEN 性別 = 0 THEN '女性'
    WHEN 性別 = 1 THEN '男性'
    ELSE NULL
  END '性別'
FROM 負責人清單
ORDER BY 負責人姓名
```

負責人清單 (9r × 2c)

負責人姓名	性別
小林	女性
小野	男性
宇賀神	男性
伊藤	女性
佐瀨	男性
岡田	男性
玲木	男性
齋藤	男性
藤本	男性

- 最先符合條件的將成為CASE的傳回值。
- 若無符合則傳回NULL，但若有ELSE敘述，則傳回ELSE指定的值。

SELECT進階語法

- 一些日期函數的操作：
  - 取得今天日期

```
SELECT CURDATE() '今天日期'
```

  - 取得今天日期及三天後日期(加上3天)

```
SELECT CURDATE() '今天日期',
       DATE_ADD(CURDATE(), INTERVAL 3 DAY) '三天後日期'
```

結果 #1 (1r × 2c)

今天日期	三天後日期
2020-09-17	2020-09-20

  - 取得今天日期及三天前日期(減去3天)

```
SELECT CURDATE() '今天日期',
       DATE_ADD(CURDATE(), INTERVAL -3 DAY) '三天前日期'
```

結果 #1 (1r × 2c)

今天日期	三天前日期
2020-09-17	2020-09-14

SELECT進階語法

- 一些日期函數的操作：
  - 取得現在日期時間

```
SELECT NOW() '現在日期時間'
```

  - 取得現在日期時間及10分鐘後時間(加上10分鐘)

```
SELECT NOW() '現在日期時間',
DATE_ADD(NOW(), INTERVAL 10 MINUTE) '10分鐘後時間'
```

結果 #1 (1r × 2c)

現在日期時間	10分鐘後時間
2020-09-17 16:37:28	2020-09-17 16:47:28

  - 取得現在日期時間及10分鐘前時間(減去10分鐘)

```
SELECT NOW() '現在日期時間',
DATE_ADD(NOW(), INTERVAL -10 MINUTE) '10分鐘後時間'
```

結果 #1 (1r × 2c)

現在日期時間	10分鐘後時間
2020-09-17 16:42:00	2020-09-17 16:32:00

SELECT進階語法

- DATE_ADD(日期時間,INTERVAL 數值 單位)函式可用單位：

```
MICROSECOND 毫秒
SECOND 秒
MINUTE 分
HOUR   時
DAY    日
WEEK   週
MONTH  月
QUARTER  季
YEAR  年
SECOND_MICROSECOND
```

```
MINUTE_MICROSECOND
MINUTE_SECOND
HOUR_MICROSECOND
HOUR_SECOND
HOUR_MINUTE
DAY_MICROSECOND
DAY_SECOND
DAY_MINUTE
DAY_HOUR
YEAR_MONTH
```

SELECT進階語法

- UNION(集合運算子)：聯集，同時取得負責人清單的負責人ID與負責人姓名，以及分店負責人清單的分店負責人ID與分店負責人姓名。

```
SELECT 負責人ID, 負責人姓名
FROM 負責人清單
UNION
SELECT 分店負責人ID, 分店負責人姓名
FROM 分店負責人清單
```

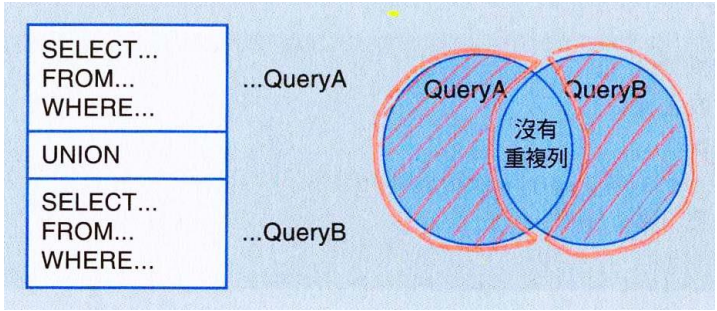
負責人清單 (12r × 2c)

負責人ID	負責人姓名
1	玲木
2	小野
3	齋藤
4	藤本
5	小林
6	伊藤
7	佐瀨
8	宇賀神
9	岡田
10	田中
11	井上
12	佐佐木

- UNION敘述是將兩個SELECT的結果聯集起來，就是將兩個資料表的記錄都全部結合成在一起，如果有重複記錄，只顯示其中一筆。

SELECT進階語法

- UNION敘述：
  - 去掉重複列後A和B所有資料的集合。



  - 兩個SELECT的欄位數目和格式必須相同，如果不相等需補上NULL使其相等。

```
SELECT 欄位一, 欄位二
FROM ...
UNION
SELECT 欄位一, 欄位二
FROM ...
ORDER BY ...
```

兩個SELECT的欄位數目和格式必須相同

  - 如果有ORDER BY要放在最後一個SELECT後面。

SELECT進階語法

- 練習：
- 1.取得月份別的商品別販賣數量。

- 2.取得包含於負責人清單但不包含於分店負責人清單中的負責人總販賣業績(販賣單價*數量)。

SELECT進階語法

- 練習1(解答)：取得月份別的商品別販賣數量。

```
SELECT B.商品名稱 '商品名稱',
  SUM( CASE
         WHEN DATE_FORMAT(A.處理日,'%Y') = 2000
              AND DATE_FORMAT(A.處理日,'%c') = 4
         THEN A.數量
         ELSE 0
       END ) '2000年4月',
  SUM( CASE
         WHEN DATE_FORMAT(A.處理日,'%Y') = 2000
              AND DATE_FORMAT(A.處理日,'%c') = 5
         THEN A.數量
         ELSE 0
       END ) '2000年5月',
  SUM( CASE
         WHEN DATE_FORMAT(A.處理日,'%Y') = 2000
              AND DATE_FORMAT(A.處理日,'%c') = 6
         THEN A.數量
         ELSE 0
       END ) '2000年6月'
FROM 販賣資料 A, 商品清單 B
WHERE A.商品ID = B.商品ID
GROUP BY B.商品名稱
ORDER BY B.商品名稱
```

販賣資料 (9r × 4c)

商品名稱	2000年4月	2000年5月	2000年6月
15吋液晶螢幕	0	5	1
17吋螢幕	0	5	0
19吋螢幕	3	0	0
HUB	0	0	5
印表機	1	0	0
桌上型電腦	5	1	0
掃描器	1	0	0
筆記型電腦	1	2	5
網路卡	0	0	2

SELECT進階語法

- 練習2(解答)：取得包含於負責人清單但不包含於分店負責人清單中的負責人總販賣業績(販賣單價*數量)。

```
SELECT SUM(B.販賣單價 * A.數量) '總業績'
FROM 販賣資料 A, 商品清單 B
WHERE A.商品ID = B.商品ID
  AND A.負責人ID IN (SELECT 負責人ID FROM 負責人清單)
  AND A.負責人ID NOT IN (SELECT 分店負責人ID FROM 分店負責人清單)
```

休息一下~

SELECT字串操作函式

- 字串操作除了可以任意改變顯示的方式外，也可以用在程式碼檢查等方面，是非常方便的功能。

- 雖然字串結果也可以交給程式去操作，但直接在SQL語法中就做好效率較高。

SELECT字串操作函式

- 字串操作(連結)：

```
SELECT CONCAT(負責人姓名,'敬啟') "負責人姓名(敬啟)"
FROM 負責人清單
```

負責人清單 (9r × 1c)

負責人姓名(敬啟)
玲木敬啟
小野敬啟
齋藤敬啟
藤本敬啟
小林敬啟
伊藤敬啟
佐瀨敬啟
宇賀神敬啟
岡田敬啟

- 用CONCAT(字串1,字串2,…)函式連接字串。
- 不要串接NULL值。

SELECT字串操作函式

- ASCII()函式將英文字母轉換成其編碼的數值。
- CHAR()函式將數值轉成相對應的字母。

```
SELECT ASCII('A') "C_A", ASCII('a') "C_a", ASCII('0') "C_0",
       CHAR(65) "C_65", CHAR(97) "C_97", CHAR(48) "C_48"
```

結果 #1 (1r × 6c)

C_A	C_a	C_0	C_65	C_97	C_48
65	97	48	A	a	0

SELECT字串操作函式

- 取得字串長度：顯示商品名稱與其字串長度。
  - 1.LENGTH()：不管中英文，
    依照其byte數算。

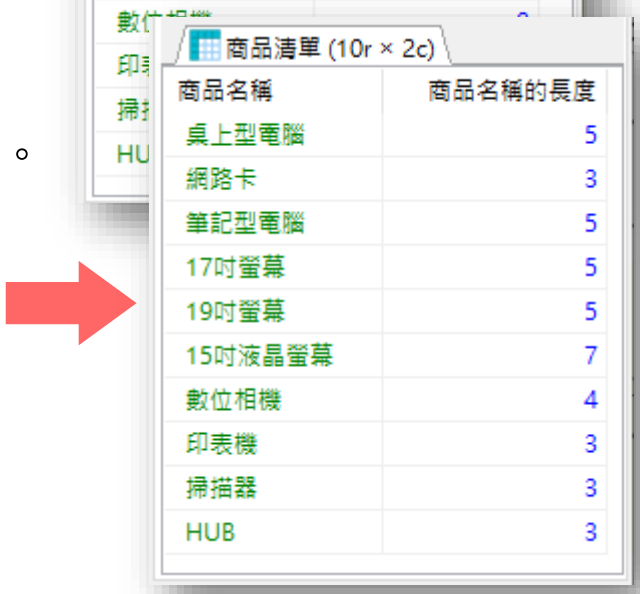
```
SELECT 商品名稱,
       LENGTH(商品名稱)
       "商品名稱的長度"
FROM 商品清單
```

商品清單 (10r × 2c)

商品名稱	商品名稱的長度
桌上型電腦	10
網路卡	6
筆記型電腦	10
17吋螢幕	8
19吋螢幕	8
15吋液晶螢幕	12

  - 2.CHAR_LENGTH()：不管中英文，
    一個字符算一個。

```
SELECT 商品名稱,
       CHAR_LENGTH(商品名稱)
       "商品名稱的長度"
FROM 商品清單
```



商品清單 (10r × 2c)

商品名稱	商品名稱的長度
桌上型電腦	5
網路卡	3
筆記型電腦	5
17吋螢幕	5
19吋螢幕	5
15吋液晶螢幕	7
數位相機	4
印表機	3
掃描器	3
HUB	3

SELECT字串操作函式

- 取得部分字串：
  - LEFT(字串, n)：自字串左邊開始取n個字。
  - 例：LEFT(“筆記型電腦”, 2) → “筆記”

  - RIGHT(字串, n)：自字串右邊開始取n個字。
  - 例：RIGHT(“筆記型電腦”, 2) → “電腦”

  - SUBSTR(字串,m, n)：自字串左邊第m個字開始取n個字。
  - 例：SUBSTR(“筆記型電腦”, 3, 2) → “型電”

1 2 3 4 5

SELECT字串操作函式

- 大小寫變換：
  - LOWER(字串)轉成小寫：
  - 例：LOWER(“MySql”) → “mysql”

  - UPPER(字串)轉成大寫：
  - 例：UPPER(“MySql”) → “MYSQL”

  - 只會改變顯示結果，不會改變原始資料。

SELECT字串操作函式

- 去除空白字元：
  - LTRIM()去除左邊空白：
  - 例：LTRIM(“ MySQL ”) → “MySQL “
  - RTRIM()去除右邊空白：
  - 例：RTRIM(“ MySQL ”) → “ MySQL“
  - TRIM()去除頭尾空白：
  - 例：TRIM(“ MySQL ”) → “MySQL”
  - RTRIM()在去除字串尾端附帶的空白時很常用。

SELECT字串操作函式

- 將數值轉成字串：
  - 使用CONCAT(數值,'')：
  - 例： SELECT CONCAT(123, '') → "123"

  - 使用CAST(數值 AS CHAR)：
  - 例： SELECT CAST(123 AS CHAR) → "123"

  - 使用在需要將數值與字串串接在一起顯示時，例：

```
SELECT 商品名稱, CONCAT(販賣單價,'元')
FROM 商品清單
```

商品清單 (10r × 2c)

商品名稱	CONCAT(販賣單價,'元')
桌上型電腦	180000元
網路卡	20000元
筆記型電腦	270000元
17吋螢幕	50000元
19吋螢幕	95000元
15吋液晶螢幕	120000元
數位相機	(NULL)
印表機	25000元
掃描器	30000元
HUB	7000元

SELECT字串操作函式

- REPLACE()取代字串：

```
SELECT REPLACE(商品名稱,'電腦','PC') "商品名稱"
FROM 商品清單
```

- 顯示出來的內容會被取代，但並不會改變原始資料。

休息一下~

資料異動指令

- SELECT指令並不會更動原有的資料，只是做查詢的動作，就是幫我們把資料撈出來，再交給執行此查詢的程式去做後續處理。
- 即使下錯指令也不太會傷到原始資料。

- 而異動指令(INSERT、UPDATE、DELETE)是直接更動資料庫內原始資料，使用時要謹慎。

- UPDATE、DELETE一但下錯指令，資料的損失會很嚴重。

INSERT指令(新增資料)

- INSERT指令：

```
INSERT INTO 資料表名稱(欄位1, 欄位2, 欄位3, ...)
                    VALUES( 值1,   值2,   值3, …)
```

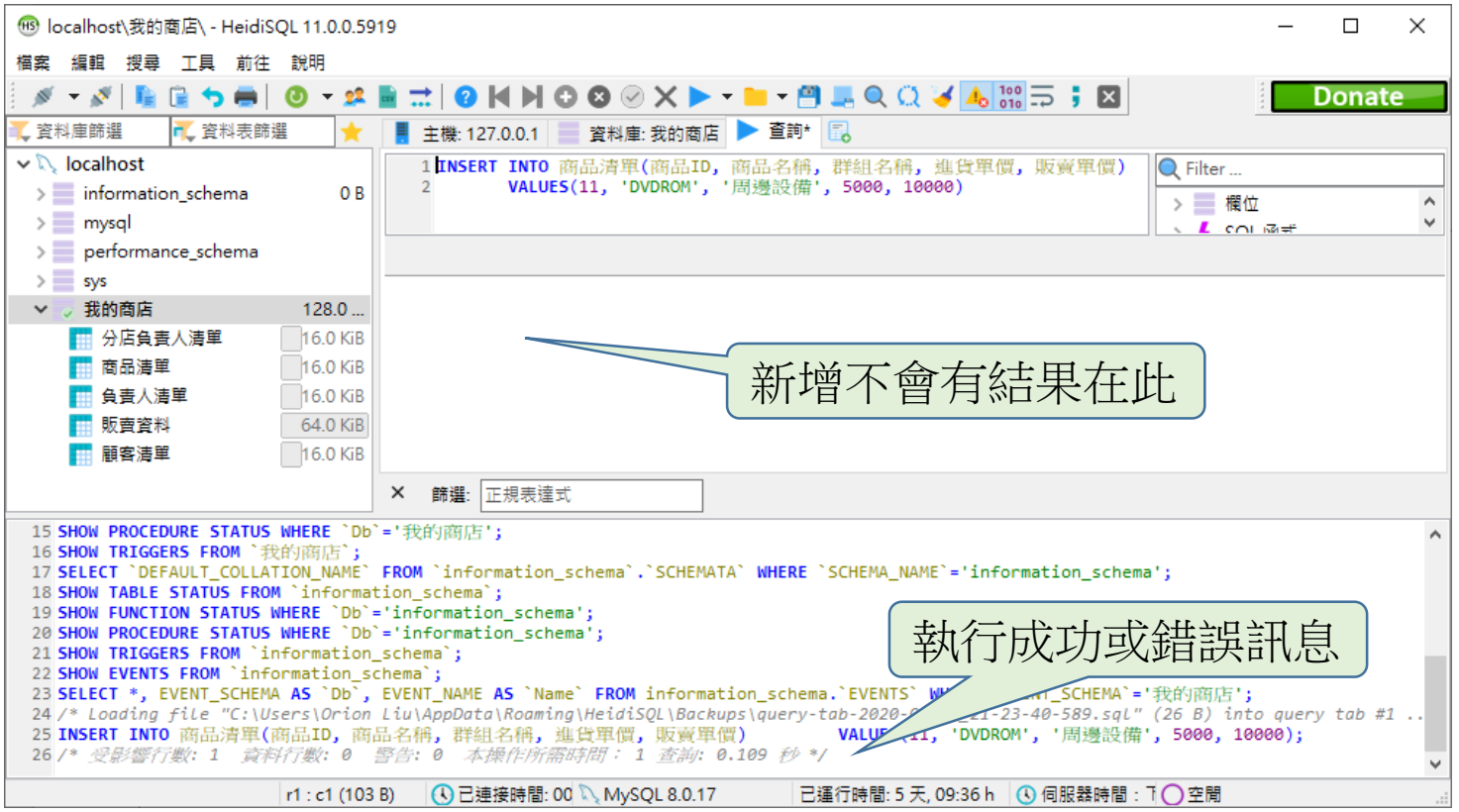
  - 要寫入的值必須與欄位數目、順序、型態相同。
  - 未指明的欄位將自動填入NULL(若是NOT NULL欄位則一定要有輸入值)。
  - 欄位名稱若省略，則會依定義資料表時的順序依序填入，但不建議這樣用。

```
INSERT INTO 資料表名稱 VALUES(值1, 值2, 值3, …)
```

INSERT指令(新增資料)

- 在商品清單中增加新的商品

```
INSERT INTO 商品清單(商品ID, 商品名稱, 群組名稱, 進貨單價, 販賣單價)
        VALUES(11, 'DVDROM', '周邊設備', 5000, 10000)
```



INSERT指令(新增資料)

- 查看結果是否正確新增了一筆資料：

```
SELECT * FROM 商品清單
```

正確的新增了，但因為商品ID欄位是CHAR型態，故排列順序是依照”文字“排列

商品清單 (11r × 5c)

商品ID	商品名稱	群組名稱	進貨單價	販賣單價
1	桌上型電腦	電腦主機	150,000	180,000
10	網路卡	網路設備	15,000	20,000
11	DVDROM	周邊設備	5,000	10,000
2	筆記型電腦	電腦主機	230,000	270,000
3	17吋螢幕	周邊設備	40,000	50,000
4	19吋螢幕	周邊設備	80,000	95,000
5	15吋液晶螢幕	周邊設備	100,000	120,000
6	數位相機	周邊設備	(NULL)	(NULL)
7	印表機	周邊設備	20,000	25,000
8	掃描器	周邊設備	25,000	30,000
9	HUB	網路設備	5,000	7,000

INSERT指令(新增資料)

- 使用了SELECT的INSERT指令：
  - 請把分店負責人清單中負責人ID為10、11的資料新增到負責人清單中。

```
INSERT INTO 負責人清單(負責人ID, 負責人姓名,
                       姓名拼音, MGR_ID, 出生日期, 性別)
SELECT 分店負責人ID, 分店負責人姓名,
       姓名拼音, MGR_ID, 出生日期, 性別
FROM 分店負責人清單
WHERE 分店負責人ID IN (10, 11)
```

  - 當需要從別的資料表取得所需新增的資料時，可以用SELECT取代VALUES敘述，當有多數個結果時，都會新增進去。
  - 注意相對的欄位及其值，不要對應錯了。

INSERT指令(新增資料)

- 查看是否正確新增了兩筆：

```
SELECT * FROM 負責人清單
ORDER BY CAST(負責人ID AS SIGNED)
```

因為負責人ID是CHAR型態，我們將它轉為數值型態再來排序

負責人清單 (11r × 6c)

負責人ID	負責人姓名	姓名拼音	MGR_ID	出生日期	性別
1	玲木	SUSUKI	(NULL)	1960-01-23	1
2	小野	ONO	(NULL)	1960-08-02	1
3	齋藤	SAITO	(NULL)	1963-10-15	1
4	藤本	FUJIMOTO	3	1972-07-18	1
5	小林	KOBAYASHI	3	1971-02-11	0
6	伊藤	ITO	2	1972-04-01	0
7	佐瀨	SASE	2	1975-02-21	1
8	宇賀神	UGAJIN	1	1975-12-22	1
9	岡田	OKADA	4	1972-03-18	1
10	田中	TANAKA	9	1975-05-23	1
11	井上	INOUE	9	1980-02-18	1

正確的新增了10和11

INSERT指令(新增資料)

- 暫時資料表：
  - 有時我們需要的資料可能來自很多資料表的欄位，而獲得結果之後又需要再次處理，這時我們可以將SELECT的結果暫時存放到一個暫時的資料表，做後續使用。
  - 該資料表只會在這次的連結(Session)有效，一旦結束連結就會自動刪除(不過最好還是在結束前手動刪除)。
  - 先以CREATE TEMPORARY TABLE指令建立暫時資料表(不會顯示在資料表區)，再用INSERT INTO指令將資料存入暫時資料表。
  - 暫時資料表在SHOW TABLES時是看不見的。

INSERT指令(新增資料)

- 取得群組名稱為「周邊設備」之商品的商品名稱ID、商品名稱，並存放到暫時資料表「周邊設備」中。
  - 1.建立暫時資料表：

```
CREATE TEMPORARY TABLE 周邊設備
(   商品ID char(5),
    商品名稱 char(20)   )
```

  - 2.將資料存入暫時資料表：

```
INSERT INTO 周邊設備
SELECT 商品ID, 商品名稱
FROM 商品清單
WHERE 群組名稱 = '周邊設備'
```

周邊設備 (7r × 2c)

商品ID	商品名稱
11	DVDROM
3	17吋螢幕
4	19吋螢幕
5	15吋液晶螢幕
6	數位相機
7	印表機
8	掃描器

  - 暫時資料表「周邊設備」在連線結束後會自動刪除。

UPDATE指令(修改資料)

- UPDATE指令語法：

```
UPDATE 資料表 SET 欄位1=值1, 欄位2=值2, ...
WHERE 條件
```

  - 指定資料表，然後SET 欄位=新的值 即可修改該欄位的內容為新值。

  - WHERE 條件 很重要，如果沒有加上條件設定，會將資料表內所有該欄位的紀錄全部修改，這應該不會常發生，除非你確定是要修改所有紀錄。

UPDATE指令(修改資料)

- 將桌上型電腦(商品ID=1)的進貨單價改成100000。

```
UPDATE 商品清單 SET 進貨單價 = 100000
WHERE 商品ID = 1
```

- 修改前：

- 修改後：

UPDATE指令(修改資料)

- 將全部商品的販賣單價設定為進貨單價的130%。

```
UPDATE 商品清單 SET 販賣單價 = 進貨單價 * 1.3
```

商品清單 (11r × 5c)

商品ID	商品名稱	群組名稱	進貨單價	販賣單價
1	桌上型電腦	電腦主機	100,000	130,000
10	網路卡	網路設備	15,000	19,500
11	DVDROM	周邊設備	5,000	6,500
2	筆記型電腦	電腦主機	230,000	299,000
3	17吋螢幕	周邊設備	40,000	52,000
4	19吋螢幕	周邊設備	80,000	104,000
5	15吋液晶螢幕	周邊設備	100,000	130,000
6	數位相機	周邊設備	(NULL)	(NULL)
7	印表機	周邊設備	20,000	26,000
8	掃描器	周邊設備	25,000	32,500
9	HUB	網路設備	5,000	6,500

- 沒有WHERE條件的敘述要謹慎使用。

UPDATE指令(修改資料)

- 請將數位相機(商品ID=6)的進貨單價更新為50000、販賣單價更新為70000。
  - 修改多個欄位：

```
UPDATE 商品清單 SET 進貨單價 = 50000, 販賣單價=70000
WHERE 商品ID = 6
```

- 修改前：

商品清單 (1r × 5c)

商品ID	商品名稱	群組名稱	進貨單價	販賣單價
6	數位相機	周邊設備	(NULL)	(NULL)

- 修改後：

商品清單 (1r × 5c)

商品ID	商品名稱	群組名稱	進貨單價	販賣單價
6	數位相機	周邊設備	50,000	70,000

UPDATE指令(修改資料)

- UPDATE使用CASE語法：將販賣單價未滿30000的提高20%。30000以上，未滿50000的降低販賣單價20%。

```
UPDATE 商品清單 SET 販賣單價 = (
  CASE
    WHEN 販賣單價 < 30000
         THEN 販賣單價 * 1.2
    WHEN 販賣單價 >= 30000 AND 販賣單價 <= 50000
         THEN 販賣單價 * 0.8
    ELSE 販賣單價
  END )
```

- 在此例ELSE一定要加上，否則前兩項條件都不符合時將會傳回NULL而被填入更新資料。

商品清單 (11r × 5c)

商品ID	商品名稱	群組名稱	進貨單價	販賣單價
1	桌上型電腦	電腦主機	100,000	130,000
10	網路卡	網路設備	15,000	23,400
11	DVDROM	周邊設備	5,000	7,800
2	筆記型電腦	電腦主機	230,000	299,000
3	17吋螢幕	周邊設備	40,000	52,000
4	19吋螢幕	周邊設備	80,000	104,000
5	15吋液晶螢幕	周邊設備	100,000	130,000
6	數位相機	周邊設備	50,000	70,000
7	印表機	周邊設備	20,000	31,200
8	掃描器	周邊設備	25,000	26,000
9	HUB	網路設備	5,000	7,800

DELETE指令(刪除資料)

- 刪除指令要謹慎使用，刪錯了是不能回復的，除非已做了相關ROLLBACK(回復)措施。

- WHERE條件也是一定要的，沒設條件的話會刪除所有資料，除非你確定要這麼做。

- 若需刪除全部資料時，不如直接DROP TABLE再重做，效果相同，速度還比較快。若需刪除的筆數相當多時(十萬、百萬筆)，也可考慮這做法。

DELETE指令(刪除資料)

- 刪除所有資料：

```
DELETE FROM 販賣資料
```

（小心使用）

- 從分店負責人清單當中把佐佐木(分店負責人ID=12)刪除。

```
DELETE FROM 分店負責人清單
WHERE 分店負責人ID = 12
```

  - 執行結果：

分店負責人清單 (4r × 6c)

分店負責人ID	分店負責人姓名	姓名拼音	MGR_ID	出生日期	性別
10	田中	TANAKA	9	1975-05-23	1
11	井上	INOUE	9	1980-02-18	1
4	藤本	FUJIMOTO	(NULL)	1972-07-18	1
9	岡田	OKADA	4	1972-03-18	1

DELETE指令(刪除資料)

- 依SELECT子查詢刪除資料：將負責人清單中負責人ID與分店負責人清單中分店負責人ID重複的紀錄刪除(就是分店負責人不能出現在負責人清單中)。

```
DELETE FROM 負責人清單
WHERE 負責人ID IN(
   SELECT 分店負責人ID
   FROM 分店負責人清單
)
```

負責人清單 (7r × 6c)

負責人ID	負責人姓名	姓名拼音	MGR_ID	出生日期	性別
1	玲木	SUSUKI	(NULL)	1960-01-23	1
2	小野	ONO	(NULL)	1960-08-02	1
3	齋藤	SAITO	(NULL)	1963-10-15	1
5	小林	KOBAYASHI	3	1971-02-11	0
6	伊藤	ITO	2	1972-04-01	0
7	佐瀨	SASE	2	1975-02-21	1
8	宇賀神	UGAJIN	1	1975-12-22	1

分店負責人清單 (5r × 6c)

分店負責人ID	分店負責人姓名	姓名拼音	MGR_ID	出生日期	性別
10	田中	TANAKA	9	1975-05-23	1
11	井上	INOUE	9	1980-02-18	1
12	佐佐木	SASAKI	9	1968-10-10	1
4	藤本	FUJIMOTO	(NULL)	1972-07-18	1
9	岡田	OKADA	4	1972-03-18	1

執行後負責人ID 4和9已經刪除了

DELETE指令(刪除資料)

- 為了資料的完整性，當要刪除的資料主鍵是別的資料表的外鍵，而且在別的資料表有紀錄時，是無法刪除的，要先解除外鍵參考關係才行，但恐有資料完整性的問題。
- 刪除外鍵命令：
  - 例：刪除販賣清單中連結到商品清單中的商品ID外鍵。

```
ALTER TABLE 販賣資料
DROP FOREIGN KEY 販賣資料_ibfk_1
```

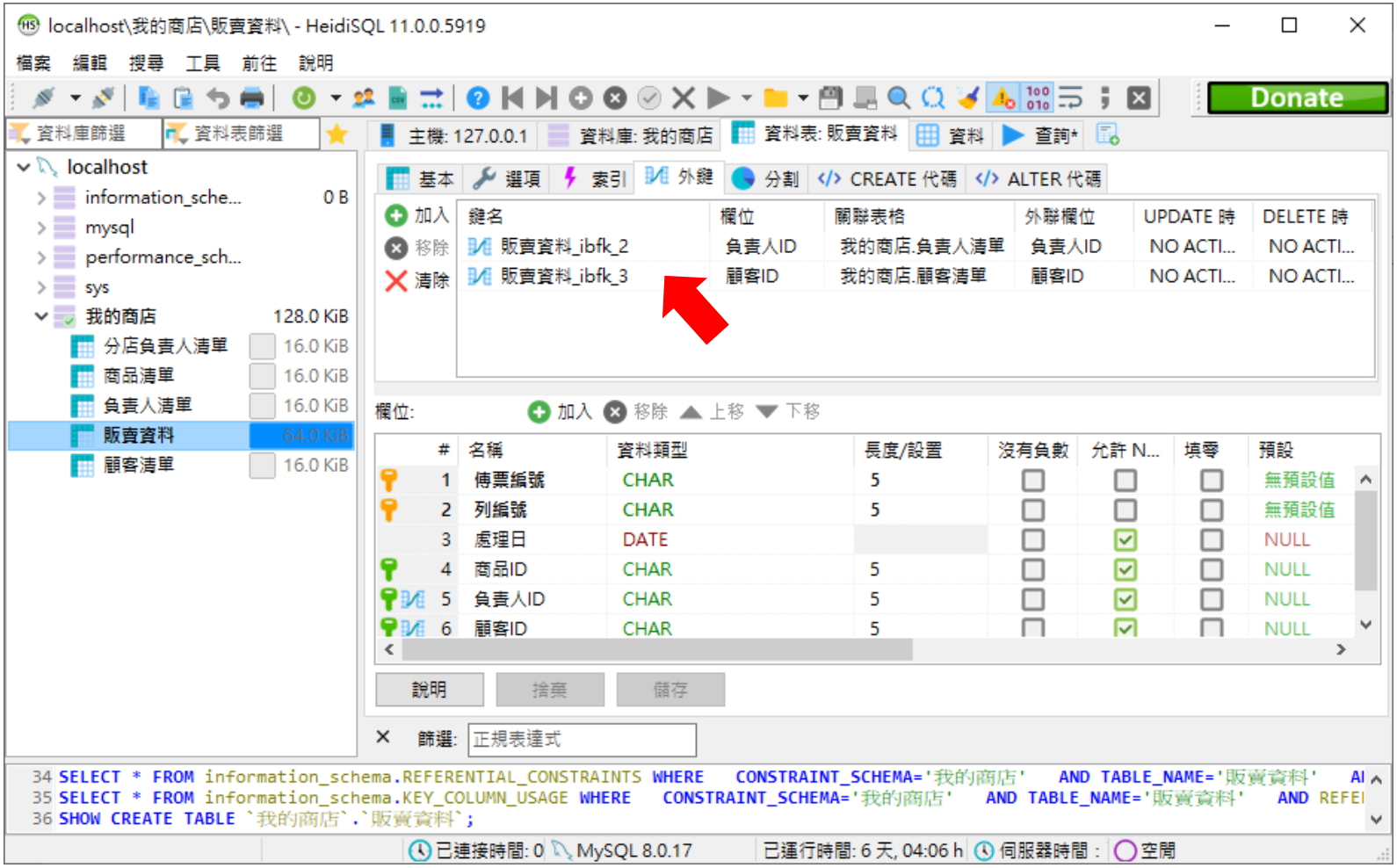
外鍵名稱請到工具軟體查一下，
MySQL中一般以 資料表名稱_ibfk_n
來命名(n是第幾個外鍵)

DELETE指令(刪除資料)

- 從HeidiSQL查看資料表詳細資料：

DELETE指令(刪除資料)

- 商品ID的外鍵已刪除：



DELETE指令(刪除資料)

- 由商品清單來刪除群組名稱有重複的資料(與販賣清單的外鍵關係已移除，否則不給執行)。

```
DELETE FROM 商品清單
WHERE 商品ID NOT IN(
  SELECT * FROM (
    SELECT MIN(商品ID) FROM 商品清單
    GROUP BY 群組名稱
  ) AS P
)
```

群組名稱沒有重複的了

商品清單 (3r × 5c)

商品ID	商品名稱	群組名稱	進貨單價	販賣單價
1	桌上型電腦	電腦主機	150,000	180,000
10	網路卡	網路設備	15,000	20,000
3	17吋螢幕	周邊設備	40,000	50,000

- 比較好的例子是在名片卡管理資料中，進行重複登錄的資料刪除處理，可以用姓名+電話號碼等條件來做重複檢查，然後將重複的資料刪除。或不小心重複輸入了資料，也可以用者種方法來刪除。

ROLLBACK和COMMIT

- 在資料庫中「交易」(Transaction)是很重要的，如果一筆異動的動作確實完成了，就會送出COMMIT完成交易，如果遭到中斷或無法完成，就要送出ROLLBACK，回復到異動前的狀態。尤其現在網路發達，很多Transaction都透過網路進行，正確性就很重要了。

- 試想你在提款機提款，都輸入完成，結果在吐錢的霎那忽然停電了!錢沒拿到!那存款簿裡扣錢了沒?（提款機會等到你把錢取走才送出COMMIT的）

資料異動指令

- 練習：
- 1.請將販賣資料的負責人ID變更為現在負責人之上司(MGR_ID)的ID，但無上司時(MGR_ID = NULL)，其負責人ID就維持不變。

- 2.統計2000年4月的商品別販賣數量，然後將其結果儲存到暫時資料表「商品別販賣數量」中。

資料異動指令

- 練習1(解答)：

```
UPDATE 販賣資料
SET 負責人ID = (
  SELECT CASE
      WHEN 負責人清單.MGR_ID IS NULL THEN 負責人清單.負責人ID
      ELSE 負責人清單.MGR_ID
      END
   FROM 負責人清單
   WHERE 負責人清單.負責人ID = 販賣資料.負責人ID
)
```

販賣資料 (20r × 7c)

傳票編號	列編號	處理日	商品ID	負責人ID	顧客ID	數量
1	1	2000-04-06	1	1	2	3
1	2	2000-04-06	4	1	2	3
10	1	2000-06-02	5	2	1	1
11	1	2000-06-06	2	3	3	2
11	2	2000-06-06	10	3	3	1
12	1	2000-06-12	2	2	2	1
13	1	2000-06-15	9	2	5	5
13	2	2000-06-15	2	2	5	2
13	3	2000-06-15	10	2	5	1
2	1	2000-04-12	1	2	1	1

資料異動指令

- 練習2(解答)：

```
DROP TEMPORARY TABLE IF EXISTS 商品別販賣數量;

CREATE TEMPORARY TABLE 商品別販賣數量 (
  商品ID CHAR(5),
  商品別數量 INT
);

INSERT INTO 商品別販賣數量
  SELECT 商品ID, SUM(數量)
  FROM 販賣資料
  WHERE 處理日>='2000-04-01' AND 處理日<='2000-04-30'
  GROUP BY 商品ID;

SELECT * FROM 商品別販賣數量;
```

- 一次要執行多道SQL敘述時，每一敘述結尾須加上分號「;」。

休息一下~

資料控制語言DCL

- 資料控制語言（Data Control Language）：
  - 在SQL語言中，是一種可對資料存取權進行控制的指令，它可以控制特定使用者帳戶對資料表、檢視表、預存程序、使用者自訂函式等資料庫物件的控制權。

  - 常見的指令有：
    GRANT 賦予使用者使用權限
    REVOKE 取消使用者的使用權限
    COMMIT 完成交易作業
    ROLLBACK 交易作業異常，將已變動的資料回復到交易開始的狀態

資料控制語言DCL

- MySQL對資料安全及其權限管理是非常嚴格的。
- 權限管理主要有以下這些：
  - 1.誰可以登入？可以從哪裡登入？
    - 本機？遠端連線？
    - 一般來說基於安全，root帳號只准許本機連線，不可以透過遠端連線。
  - 2.登入後可以使用哪些資料庫、哪些資料表？
  - 3.可以對資料表執行那些操作？
    - 是否准許SELECT、CREATE、DELETE、UPDATE、ALTER等操作
  - 4.使用者的權限是否可以授權給別的用戶？

資料控制語言DCL

- MySQL新增使用者：
  - 指令：

```
CREATE USER '使用者帳號'@'連線位置' IDENTIFIED BY '密碼';
```

  - 除了帳號名稱外，還要指定可連線的位置。
  - 例如：
    - John@localhost，則John只能從本機連線，遠端無法登入。
    - John@203.68.153.100，則John可以從指定的IP連線進來。
    - John@%，則John可以從任何位置連線進來(危險，不建議)。

    - 以上，John會有三個帳號，有不同的連線位置及可以有不同的密碼。

資料控制語言DCL

- GRANT指令：
  - 新增使用者後還要設定其可使用的資料庫和資料表及相關權限。
  - 新增使用者/授予使用者各項權限。
  - 語法：

```
GRANT privilege,[privilege],.. ON privilege_level
TO user [IDENTIFIED BY password]
[REQUIRE tsl_option]
[WITH [GRANT_OPTION | resource_option]];
```

  - 說明：（下一頁）

資料控制語言DCL

- 在GRANT關鍵字之後指定一個或多個權限，每個權限以逗號分隔。
- 接下來，指定權限套用的資料庫及資料表。MySQL支援全域性(*.*)，資料庫(database.*)，資料表(database.table)和列級別。
- 然後，放置要授予權限的使用者。如果使用者已經存在，則GRANT語句會修改其權限。如不存在，則GRANT語句將建立一個新使用者。可選的條件IDENTIFIED BY允許為使用者設定新密碼。
- 之後，可指定使用者是否必須通過安全連線(如SSL，X059等)連線到資料庫伺服器。
- 最後，可選的WITH GRANT OPTION子句允許此使用者授予其他使用者或從其他使用者刪除您擁有的許可權。此外，可以使用WITH子句來分配MySQL資料庫伺服器的資源，例如，設定使用者每小時可以使用多少個連線或語句。這在MySQL共用託管等共用環境中非常有用。

資料控制語言DCL

- MySQL 有這些常見的權限類型：
  - ALL PRIVILEGES - 所有的權限
  - CREATE - 可以建立資料表或資料庫的權限
  - DROP - 可以刪除資料表或資料庫的權限
  - DELETE - 可以在資料表中刪除資料的權限
  - INSERT - 可以新增資料到資料表的權限
  - SELECT - 可以查詢資料表的權限
  - UPDATE - 可以更新資料表中的資料的權限
  - GRANT OPTION - 可以授權使用權限給其他使用者的權限
- 你可以用逗點分隔多個權限，例如：

```
GRANT SELECT,INSERT ON 我的商店.* TO 'John'@'%';
```

資料控制語言DCL

- GRANT範例：

```
GRANT ALL PRIVILEGES ON *.* TO 'John'@'localhost' WITH GRANT OPTION;
```

  - 給予使用者John@localhost全域最高權限(含管理權限)

```
GRANT ALL PRIVILEGES ON mydb.* TO 'Mary'@'localhost';
```

  - 給予使用者Mary@localhost存取資料庫mydb的所有權限（不含管理權限）

```
GRANT SELECT, INSERT, UPDATE, DELETE ON mydb.*
TO 'myaccount'@'localhost';
```

  - 給予myaccount@localhost存取資料庫mydb的SELECT、INSERT、UPDATE、DELETE權限。

資料控制語言DCL

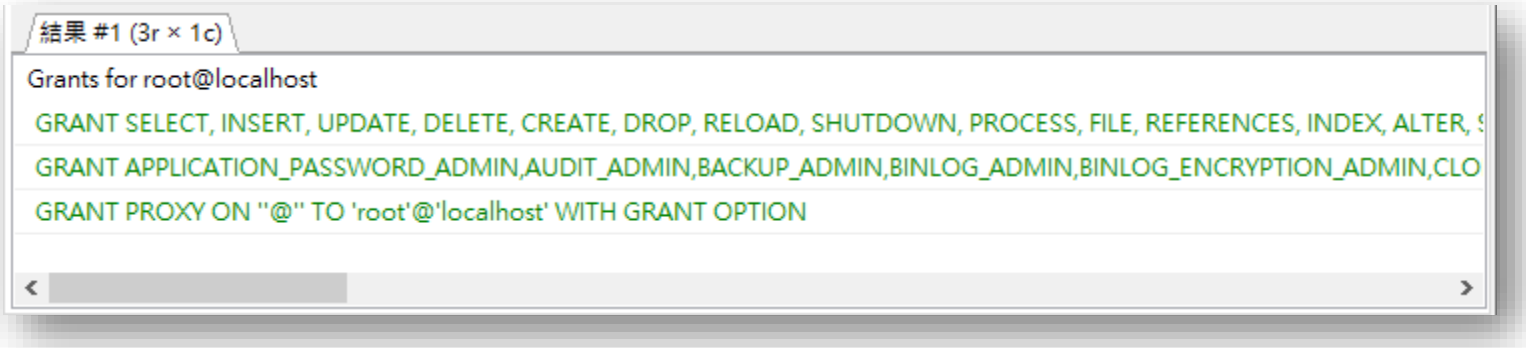
- 查看目前有哪些使用者：

```
SELECT User,Host FROM mysql.user;
```

- 查看指定帳號的權限：

```
SHOW GRANTS FOR root@localhost;
```

  - 執行結果



資料控制語言DCL

- 刪除使用者：
  - 直接刪除使用者，若有多筆記得要一一刪除：

```
DROP USER '使用者帳號'@'連線位置';
```

    - 例如： DROP USER 'John'@'%'

  - 只刪除其權限，但保留帳號：

```
REVOKE ALL PRIVILEGES, GRANT OPTION
FROM '使用者帳號'@'連線位置';
```

    - 類似停權的意思。

  - 執行GRANT、REVOKE後需下達flush privileges指令，以更新異動後的授權權限。

休息一下~

預儲程序(Stored Procedure)

- 是在資料庫儲存較複雜的SQL敘述，以便外部程式呼叫，可以視為資料庫的一種函式或子程式。
- 對於常用且固定的操作，就可以寫成預儲程式存放在資料庫裡(不是應用程式裡)。
- 優點：預存程序可封裝，並隱藏複雜的商業邏輯。預存程序可以回傳值，並可以接受參數。
- 缺點：預存程序往往客製化於特定的資料庫上，因為支援的程式語言不同。當切換到其他廠商的資料庫系統時，需要重寫原有的預存程序。

預儲程序(Stored Procedure)

- 預儲程序分成「函式」與「程序」兩種。

- 「函式」與「程序」的主要差異：

	程序	函式
呼叫	只能以 *CALL* 呼叫	可在所有 SQL 指令內呼叫。
返回	可以傳回 SELECT 的查詢結果。	以 *RETURN* 傳回單一值；傳回值的資料型態必須在函式宣告中以 *RETURNS* 指定。
參數	可以使用傳值、參考參數 (*IN, OUT, INOUT*)。	只能使用傳值參數。
允許使用的指令	所有 SQL 指令，包括 *SELECT* 、*INSERT* 、*UPDATE* 、*DELETE* 、*CREATE TABLE* 等。	不能使用存取資料表的指令。
呼叫其他函式、程序	可以呼叫其他函式、程序。	只能呼叫函式，不能呼叫程序。

預儲程序(Stored Procedure)

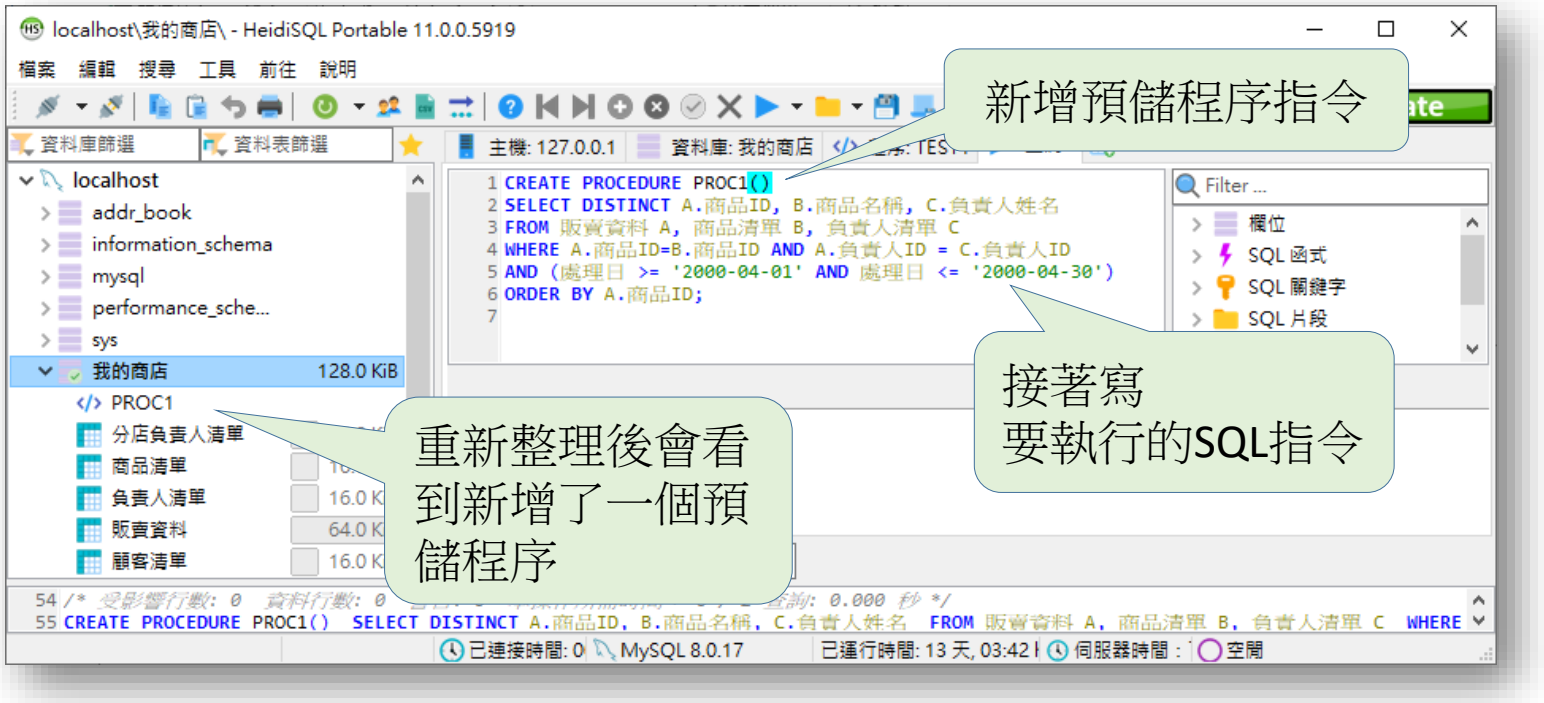
- 新增一個預儲程序： `CREATE PROCEDURE 程序名稱()`
  - (若程序名稱已存在會有錯誤訊息)

- 將前面示範的例子：「請輸出單月的購買單價有500000以上的顧客的購買年份、月份，還有顧客名稱、購買單價合計。(購買單價合計是指 單價 * 數量)」寫成預儲程序。

```
CREATE PROCEDURE PROC1()
SELECT DISTINCT A.商品ID, B.商品名稱, C.負責人姓名
FROM 販賣資料 A, 商品清單 B, 負責人清單 C
WHERE A.商品ID=B.商品ID AND A.負責人ID = C.負責人ID
AND (處理日 >= '2000-04-01' AND 處理日 <= '2000-04-30')
ORDER BY A.商品ID
```

預儲程序(Stored Procedure)

- 在HeidiSQL工具中執行建立：

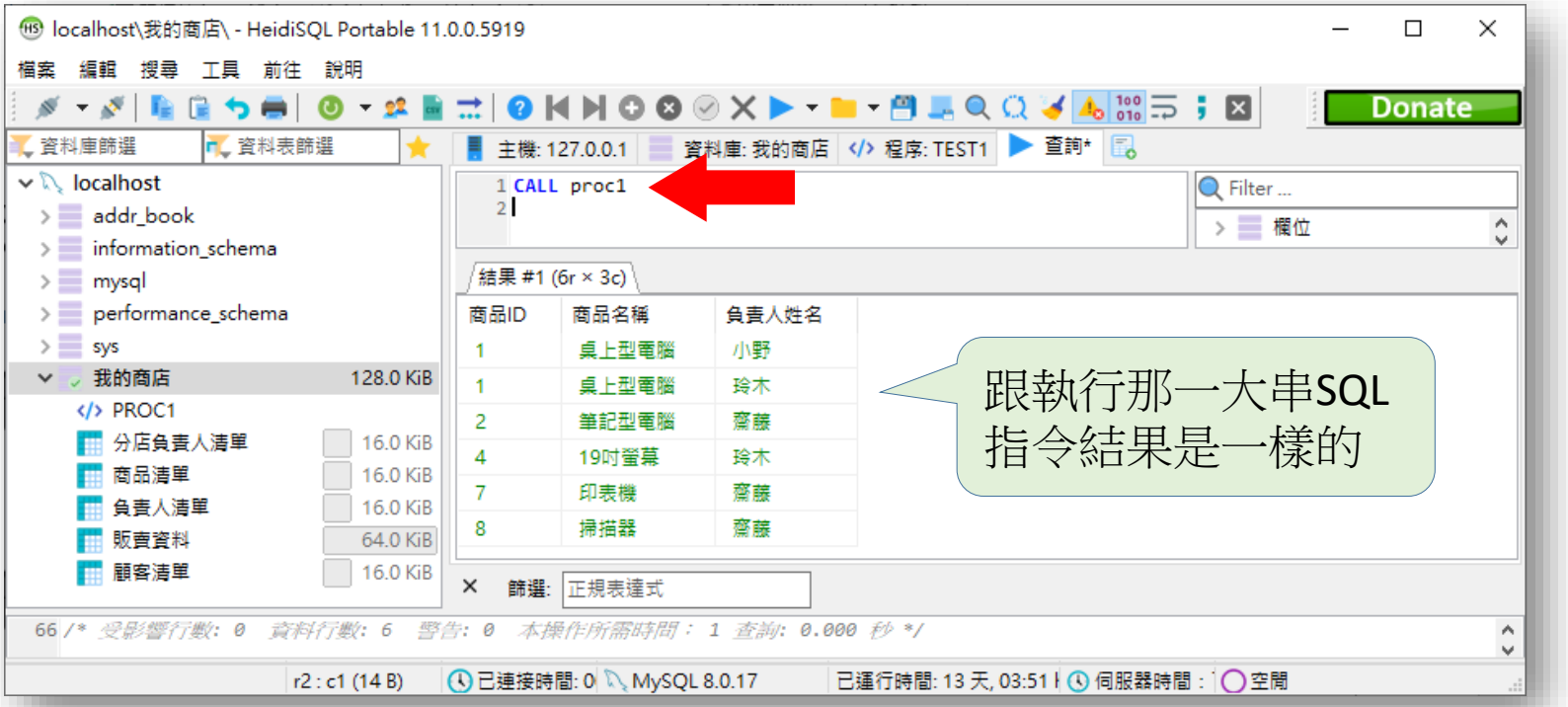


預儲程序(Stored Procedure)

- 呼叫一個預儲程序：

```
CAll 程序名稱()
```

- 呼叫剛才建立的程序：(無參數時括號可省略)



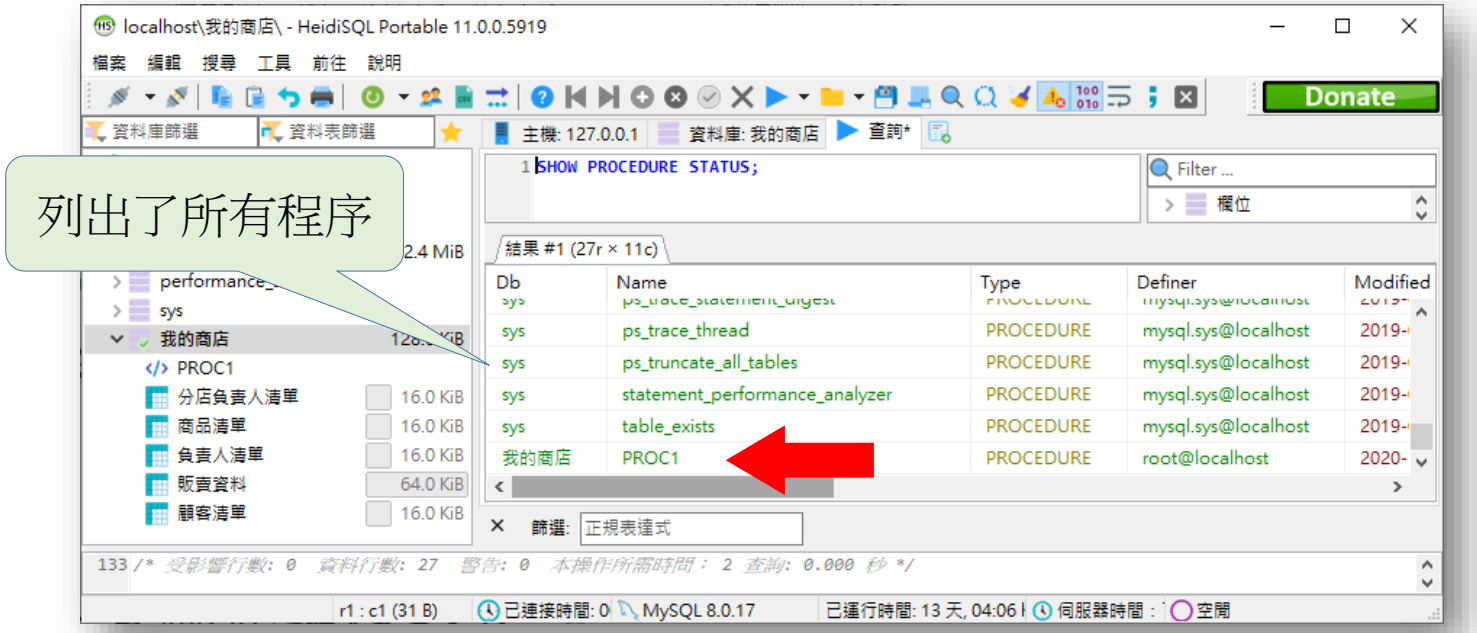
預儲程序(Stored Procedure)

- 刪除一個預儲程序：(注意，無法回復)

```
DROP PROCEDURE IF EXISTS 程序名稱
```

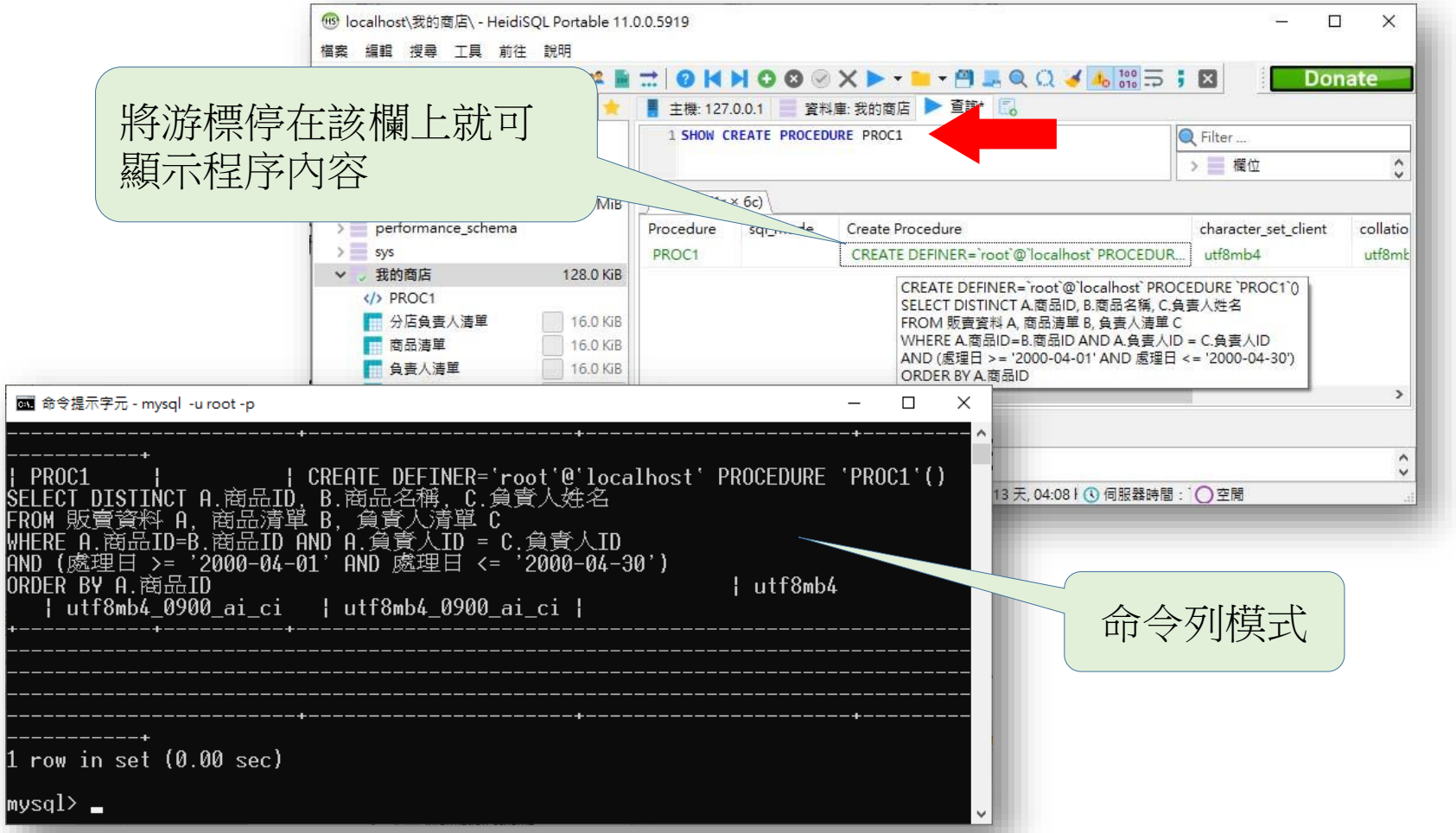
- 查看有哪些預儲程序：

```
SHOW PROCEDURE STATUS
```



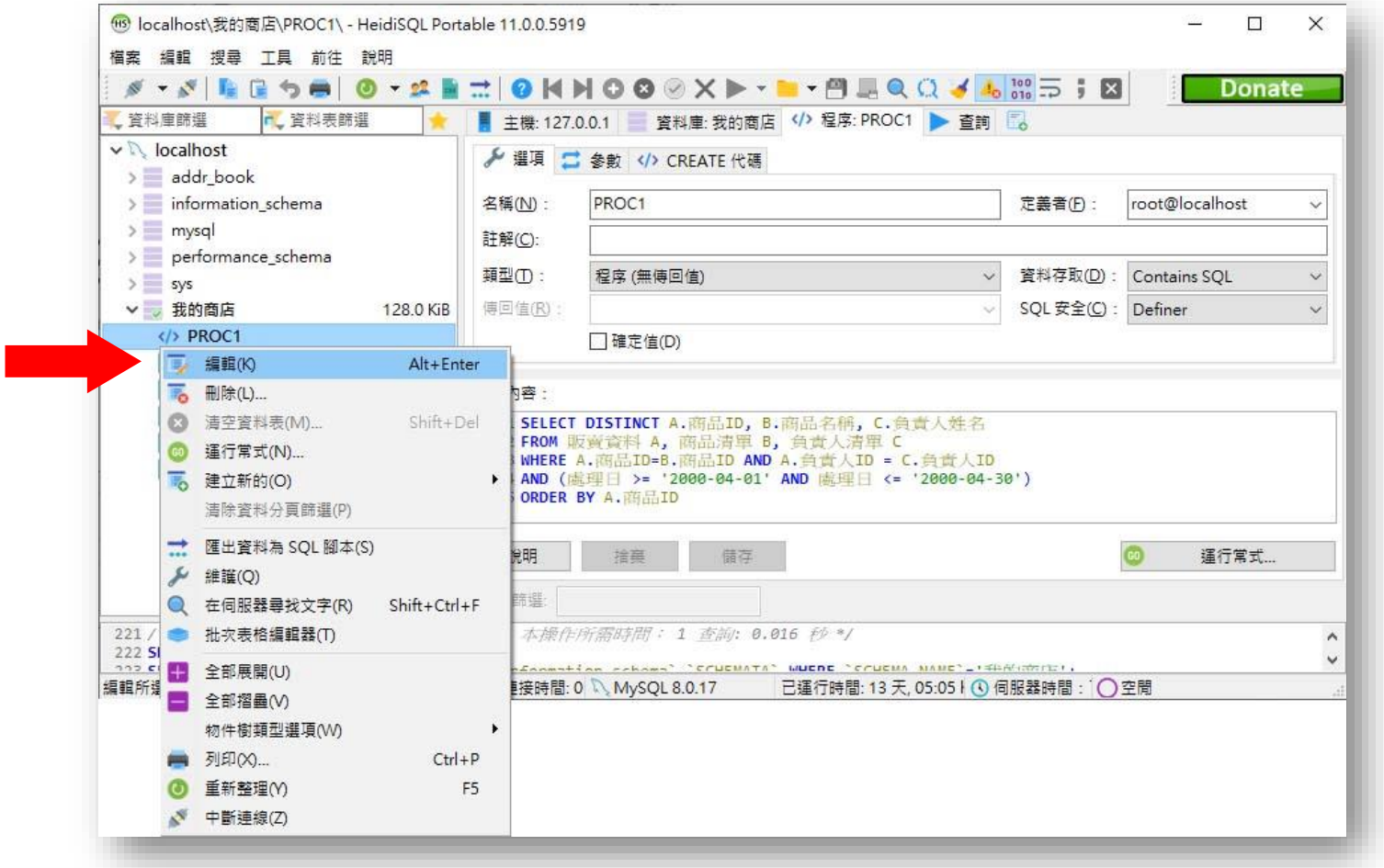
預儲程序(Stored Procedure)

- 查看特定程序內容： `SHOW CREATE PROCEDURE 程序名稱`



預儲程序(Stored Procedure)

- 在HeidiSQL工具裡可以在左邊程序名稱上按右鍵選編輯即可。



預儲程序(Stored Procedure)

- 預儲程序的結果以資料表的形式傳回，(與SELECT指令一樣)。

- 預儲程序無法和其他SQL指令結合，只能單獨執行，例如以下這種寫法是錯誤的：

```
SELECT * FROM 商品清單
WHERE 商品ID IN ( CALL PROC1 )
```

預儲程序(Stored Procedure)

- 有參數的預儲程序：

```
CREATE PROCEDURE 程序名稱(IN 參數名稱 資料型態)
```

- 例：依傳入的負責人名稱取得該負責人何時、賣掉甚麼(商品名稱)的列表。

```
CREATE PROCEDURE PROC2(IN NAME CHAR(20))
SELECT A.處理日, B.商品名稱
FROM 販賣資料 A, 商品清單 B, 負責人清單 C
WHERE A.商品ID = B.商品ID AND A.負責人ID = C.負責人ID
      AND C.負責人姓名 = NAME;
```

- 執行：

```
CALL PROC2('小野')
```

預儲程序(Stored Procedure)

- 傳入多個參數：

```
CREATE PROCEDURE 程序名稱(IN 參數名稱1 資料型態1,
                          IN 參數名稱2 資料型態2, ..... )
```

- 依前例，傳入負責人名稱及要查詢的月份：

```
CREATE PROCEDURE PROC3(IN NAME1 CHAR(20),IN MON INT)
SELECT A.處理日, B.商品名稱, C.負責人姓名
FROM 販賣資料 A, 商品清單 B, 負責人清單 C
WHERE A.商品ID = B.商品ID AND A.負責人ID = C.負責人ID
      AND C.負責人姓名 = NAME1
      AND DATE_FORMAT(A.處理日,'%c') = MON;
```

- 執行：

```
CALL PROC3('小野', 5)
```

預儲程序(Stored Procedure)

- 指定參數名稱時，小心避開欄位、資料表名稱，以免撰寫敘述時搞混。

- IN或OUT或INOUT：決定參數的用途，是只用來傳入資料、傳出資料或雙向傳送資料，省略時預設是IN。

預儲程序(Stored Procedure)

- 建立預儲函式的語法和建立預儲程序差不多：

```
CREATE FUNCTION 函式名稱(參數1, 參數2, ...) RETURN 資料型態
```

- 例：傳入西元日期，傳回民國日期。

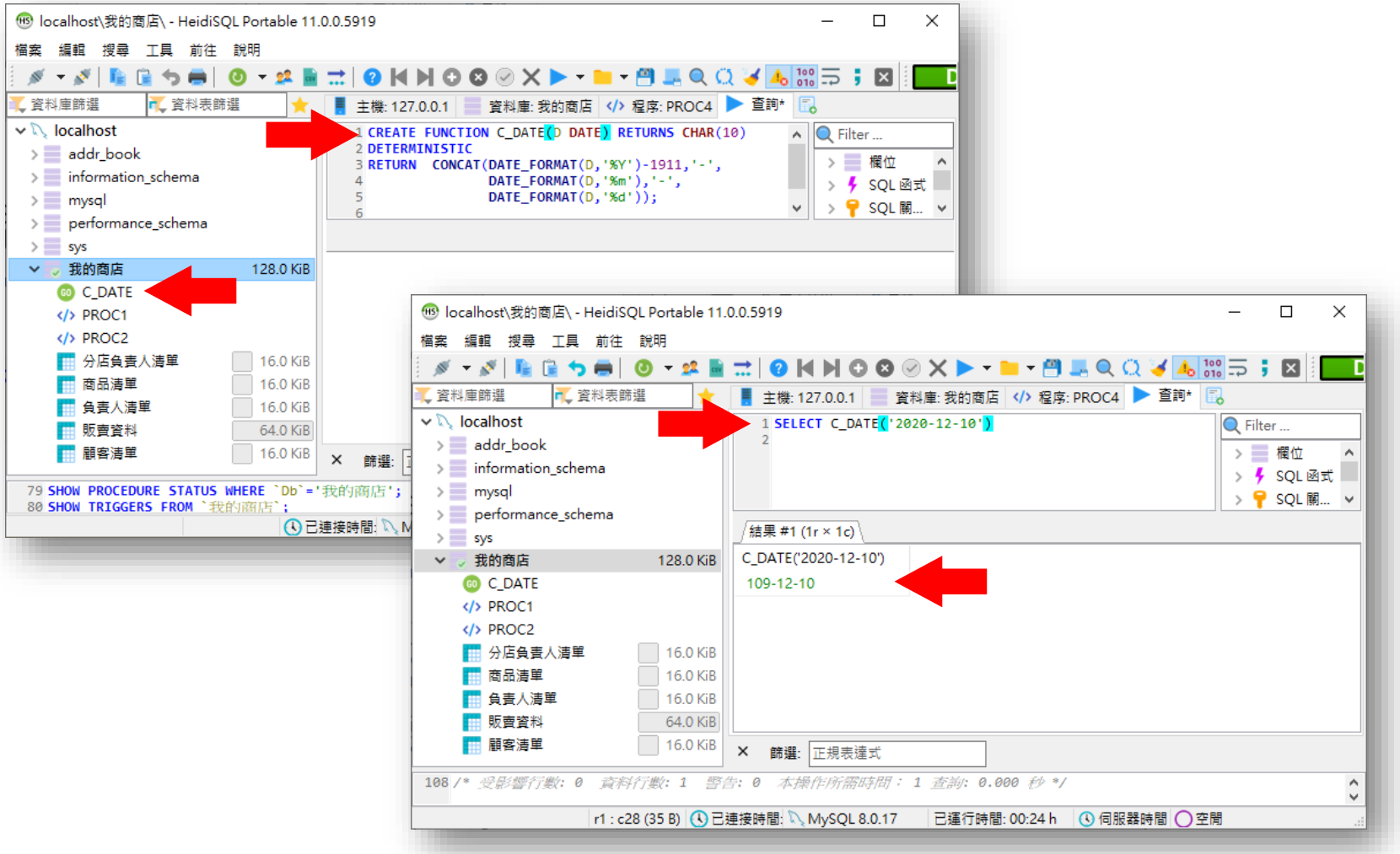
```
CREATE FUNCTION C_DATE(D DATE) RETURNS CHAR(10)
DETERMINISTIC
RETURN CONCAT(DATE_FORMAT(D,'%Y')-1911,'-',
              DATE_FORMAT(D,'%m'),'-',
              DATE_FORMAT(D,'%d'));
```

- 函式的呼叫是用SELECT：

```
SELECT C_DATE('2020-12-10')
```

預儲程序(Stored Procedure)

- 執行結果：



預儲程序(Stored Procedure)

- 預儲程序和預儲函式有各自的空間，所以名稱可以相同，不會干擾。(只會干擾寫程式的人吧)

- 在MySQL建立預儲函式時，要加一行DETERMINISTIC。
- 若有多行，要加上BEGIN...END敘述，並用DELIMITER指令指定結束字符(預設是分號;)。

```
DELIMITER //
CREATE FUNCTION 函式名稱(參數) RETURNS 資料型態
DETERMINISTIC
BEGIN
    :
    敘述;
    :
END //
```

指定結束字符是 //

不可省略

函式定義結束

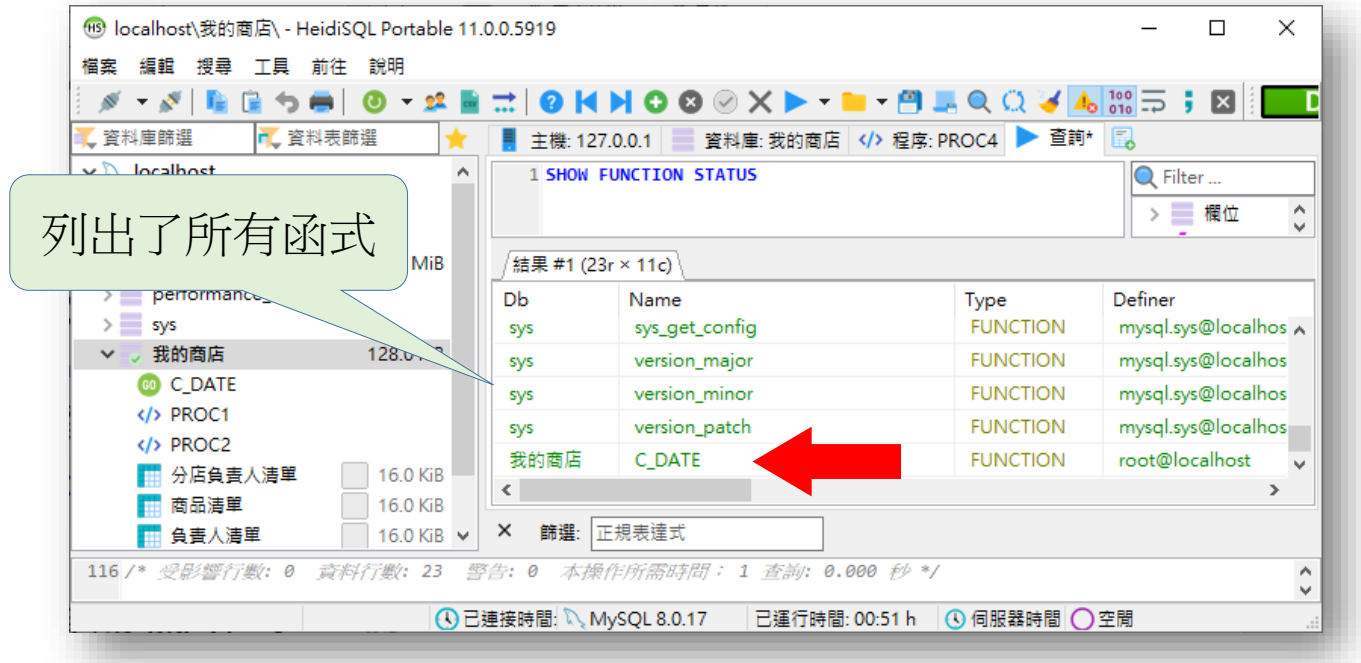
預儲程序(Stored Procedure)

- 刪除一個預儲函式：(注意，無法回復)

```
DROP FUNCTION IF EXISTS 程序名稱
```

- 查看有哪些預儲函式：

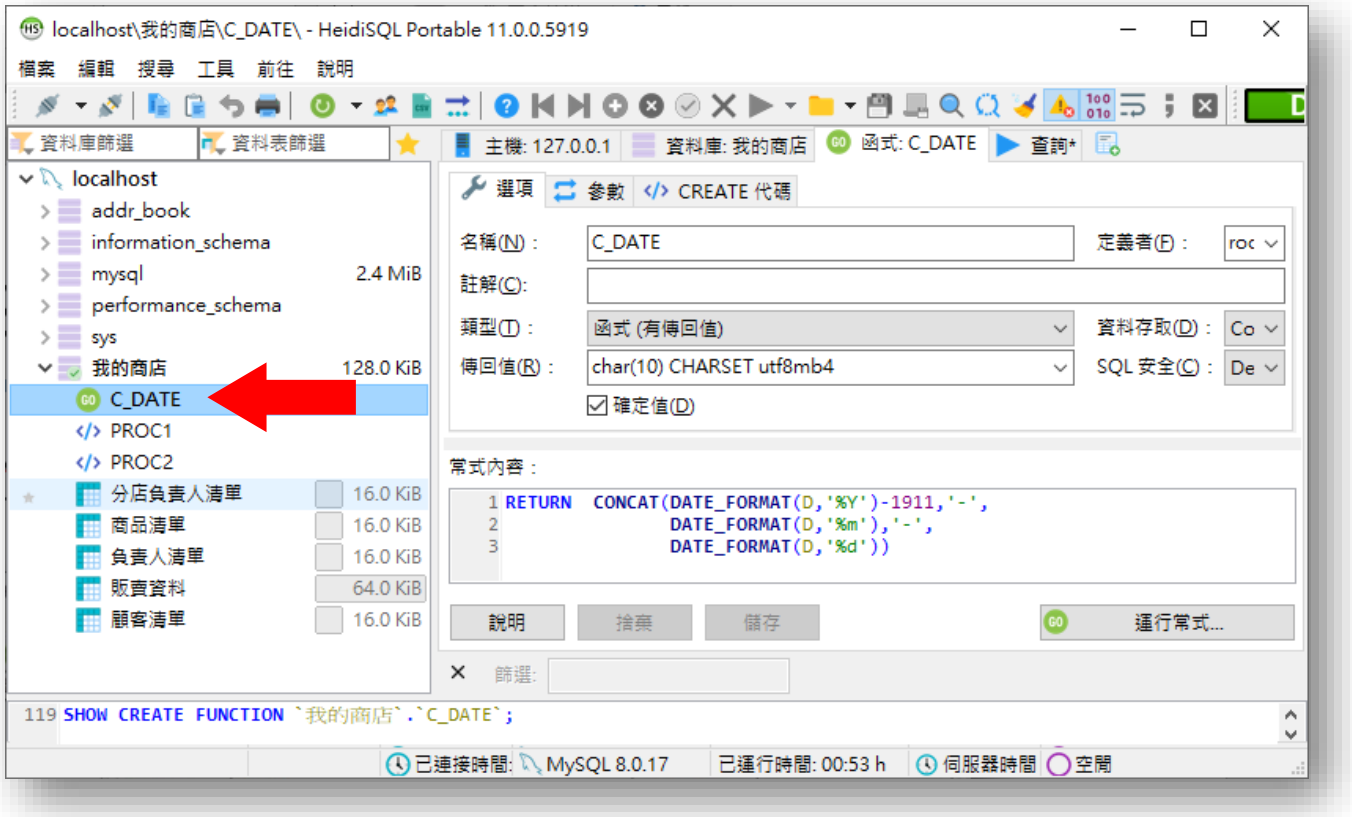
```
SHOW FUNCTION STATUS
```



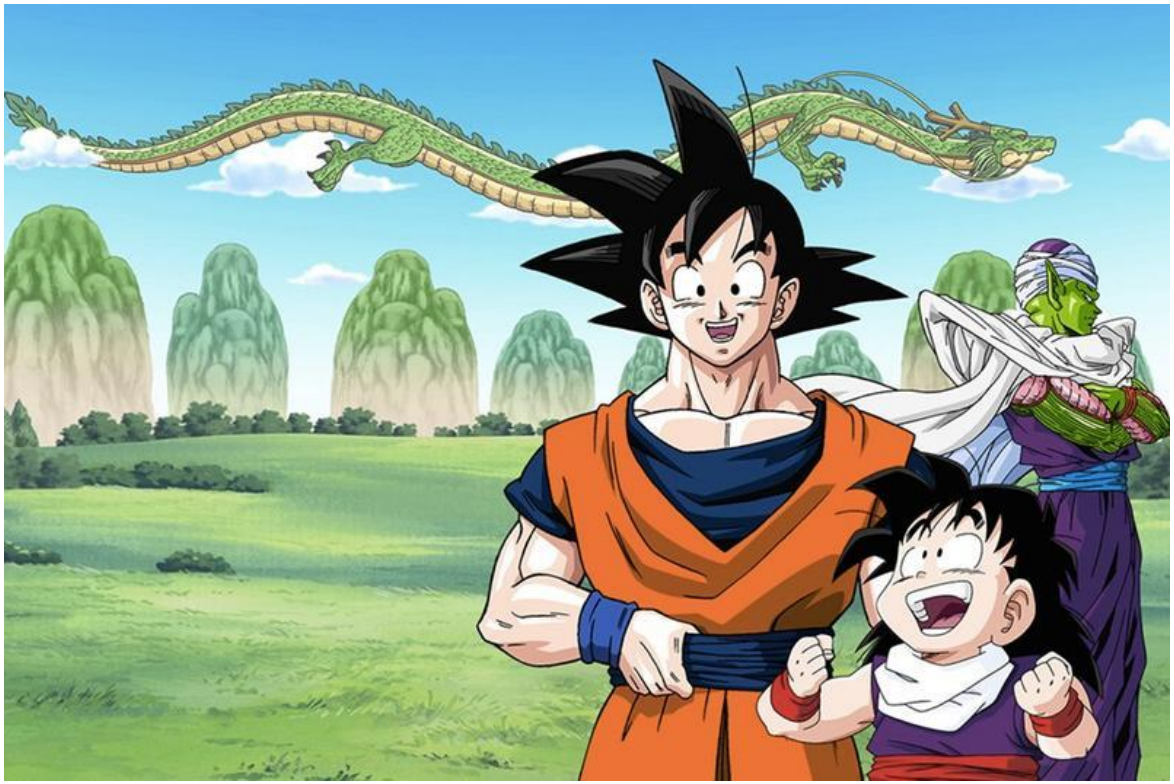
預儲程序(Stored Procedure)

- 查看特定函式內容：`SHOW CREATE FUNCTION 函式名稱`

- 或在左邊函式名稱上按滑鼠右鍵選「編輯」。



休息一下~



- 關於MySQL的一些補充

MySQL資料型態

- 除了前面用過的基本型態，在此再詳列MySQL提供的各種資料型態以供參考。

- MySQL中定義資料欄位的型態對資料庫的優化是非常重要的。

- MySQL支援多種型態，大致可以分為三類：
  - 數值
  - 日期/時間
  - 字串(字元)

MySQL資料型態

- 數值型態：

型別	大小	範圍（有符號）	範圍（無符號）	用途
TINYINT	1 位元組	(-128，127)	(0，255)	小整數值
SMALLINT	2 位元組	(-32 768，32 767)	(0，65 535)	大整數值
MEDIUMINT	3 位元組	(-8 388 608，8 388 607)	(0，16 777 215)	大整數值
INT或 INTEGER	4 位元組	(-2 147 483 648，2 147 483 647)	(0，4 294 967 295)	大整數值
BIGINT	8 位元組	(-9,223,372,036,854,775,808，9 223 372 036 854 775 807)	(0，18 446 744 073 709 551 615)	極大整數值
FLOAT	4 位元組	(-3.402 823 466 E+38，-1.175 494 351 E-38)，0，(1.175 494 351 E-38，3.402 823 466 351 E+38)	0，(1.175 494 351 E-38，3.402 823 466 E+38)	單精度 浮點數值
DOUBLE	8 位元組	(-1.797 693 134 862 315 7 E+308，-2.225 073 858 507 201 4 E-308)，0，(2.225 073 858 507 201 4 E-308，1.797 693 134 862 315 7 E+308)	0，(2.225 073 858 507 201 4 E-308，1.797 693 134 862 315 7 E+308)	雙精度 浮點數值
DECIMAL	對DECIMAL(M,D)，如果M>D，為M+2否則為D+2	依賴於M和D的值	依賴於M和D的值	小數值

MySQL資料型態

- 日期/時間型態：

型別	大小 (位元組)	範圍	格式	用途
DATE	3	1000-01-01/9999-12-31	YYYY-MM-DD	日期值
TIME	3	'-838:59:59'/'838:59:59'	HH:MM:SS	時間值或持續時間
YEAR	1	1901/2155	YYYY	年份值
DATETIME	8	1000-01-01 00:00:00/9999-12-31 23:59:59	YYYY-MM-DD HH:MM:SS	混合日期和時間值
TIMESTAMP	4	1970-01-01 00:00:00/2038 結束時間是第 **2147483647** 秒，北京時間 **2038-1-19 11:14:07**，格林尼治時間 2038年1月19日 凌晨 03:14:07	YYYYMMDD HHMMSS	混合日期和時間值，時間戳

MySQL資料型態

- 字串型態：

型別	大小	用途
CHAR	0-255位元組	定長字串
VARCHAR	0-65535 位元組	變長字串
TINYBLOB	0-255位元組	不超過 255 個字元的二進位制字串
TINYTEXT	0-255位元組	短文字字串
BLOB	0-65 535位元組	二進位制形式的長文字資料
TEXT	0-65 535位元組	長文字資料
MEDIUMBLOB	0-16 777 215位元組	二進位制形式的中等長度文字資料
MEDIUMTEXT	0-16 777 215位元組	中等長度文字資料
LONGBLOB	0-4 294 967 295位元組	二進位制形式的極大文字資料
LONGTEXT	0-4 294 967 295位元組	極大文字資料

MySQL資料型態

- 在INT欄位後面加上AUTO_INCREMENT，在每次新增一筆紀錄時，該欄位執會自動加一，不用輸入，但一個資料表只能有一個AUTO_INCREMENT欄位。通常會用在想當成主鍵的欄位上。

- CHAR()是固定寬度的文字欄位，占用固定的儲存空間。VARCHAR()則是可變動長短的文字欄位，最大可存65,535byte，其佔用空間依實際儲存的資料而定。

- 注意資料排序是依其內碼的大小，尤其是中文，Big5碼和Unicode碼排序結果也不同。

- 事實上，大部份的資料型別都差異不大，除非有特別的原因，否則不建議在資料庫存入大型資料，那一定會影響資料庫的效能。

MySQL資料庫管理

- 更改root的密碼：
  - MySQL在安裝時會要求設定root密碼，如果之後需要更換密碼，在命令列模式使用以下指令：

```
C:\>mysqladmin -u root -p舊密碼 password 新密碼
```

-p和舊密碼之間不要有空白

- 更改一般使用者密碼：
  - 以root身分登入MySQL後執行

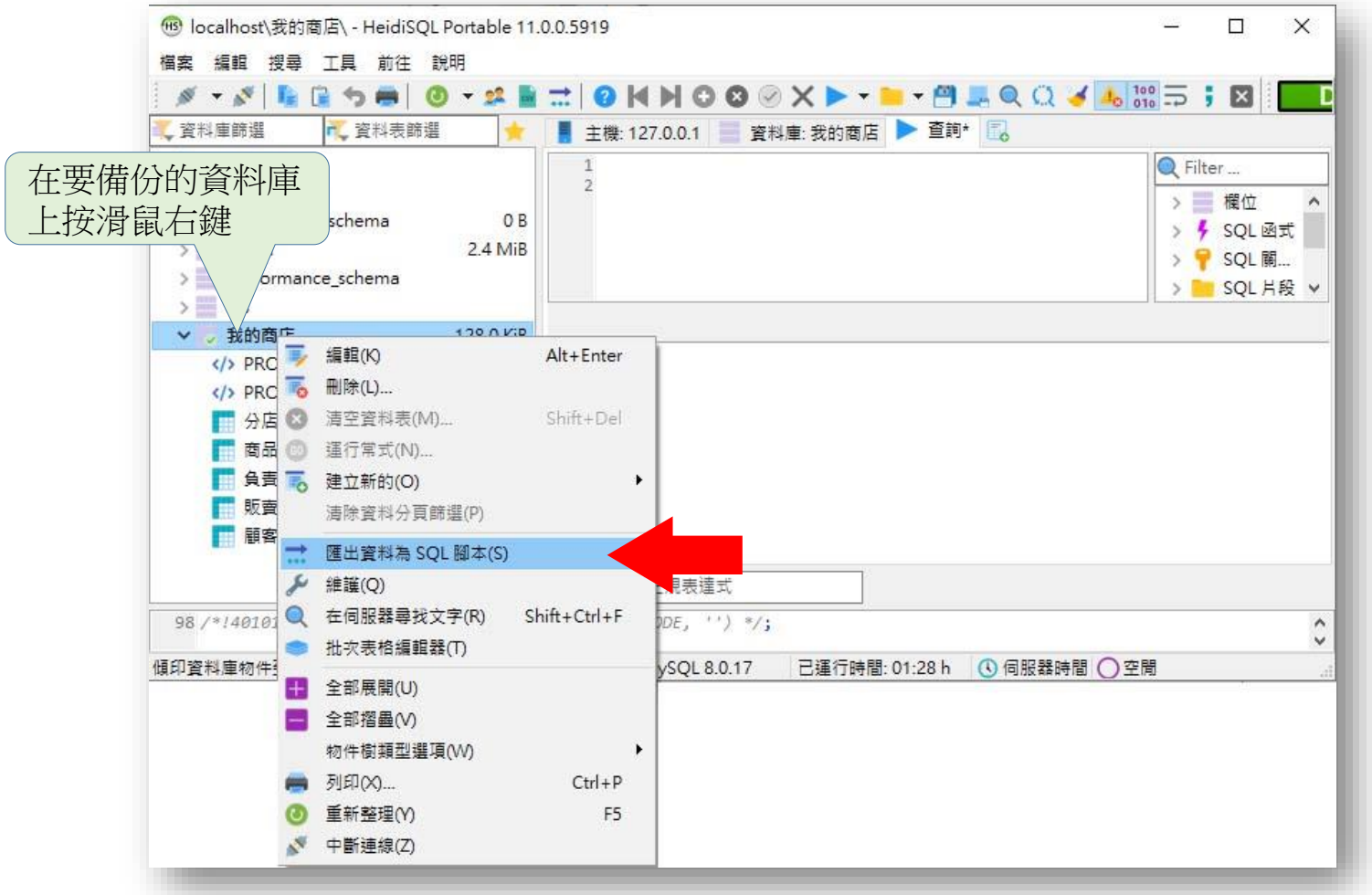
```
ALTER USER 帳號@主機 IDENTIFIED BY '新密碼';
```

  - 例：

```
ALTER USER orion@localhost IDENTIFIED BY '123456';
```

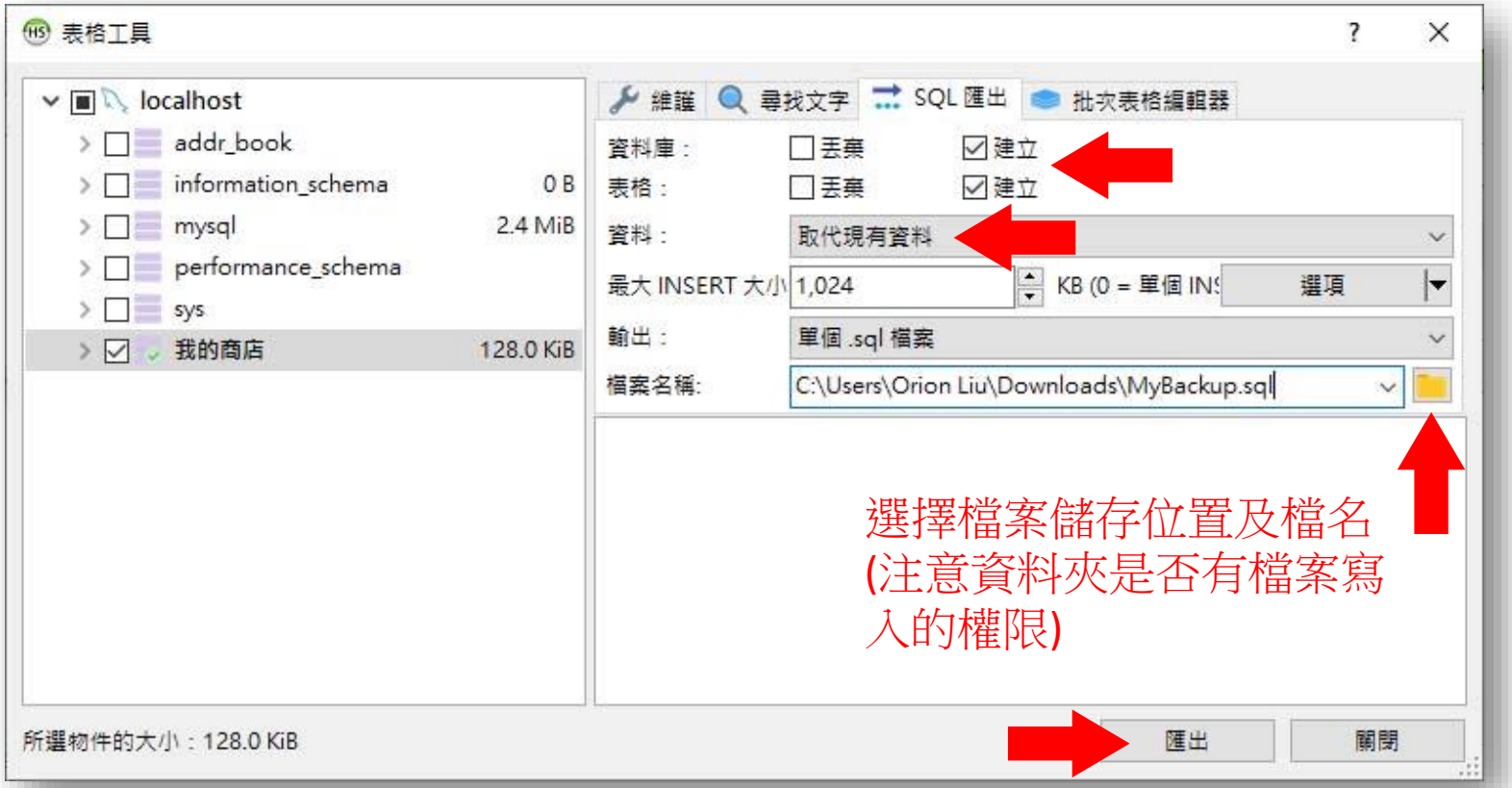
MySQL資料庫管理

- 備份資料庫(使用HeidiSQL工具)：



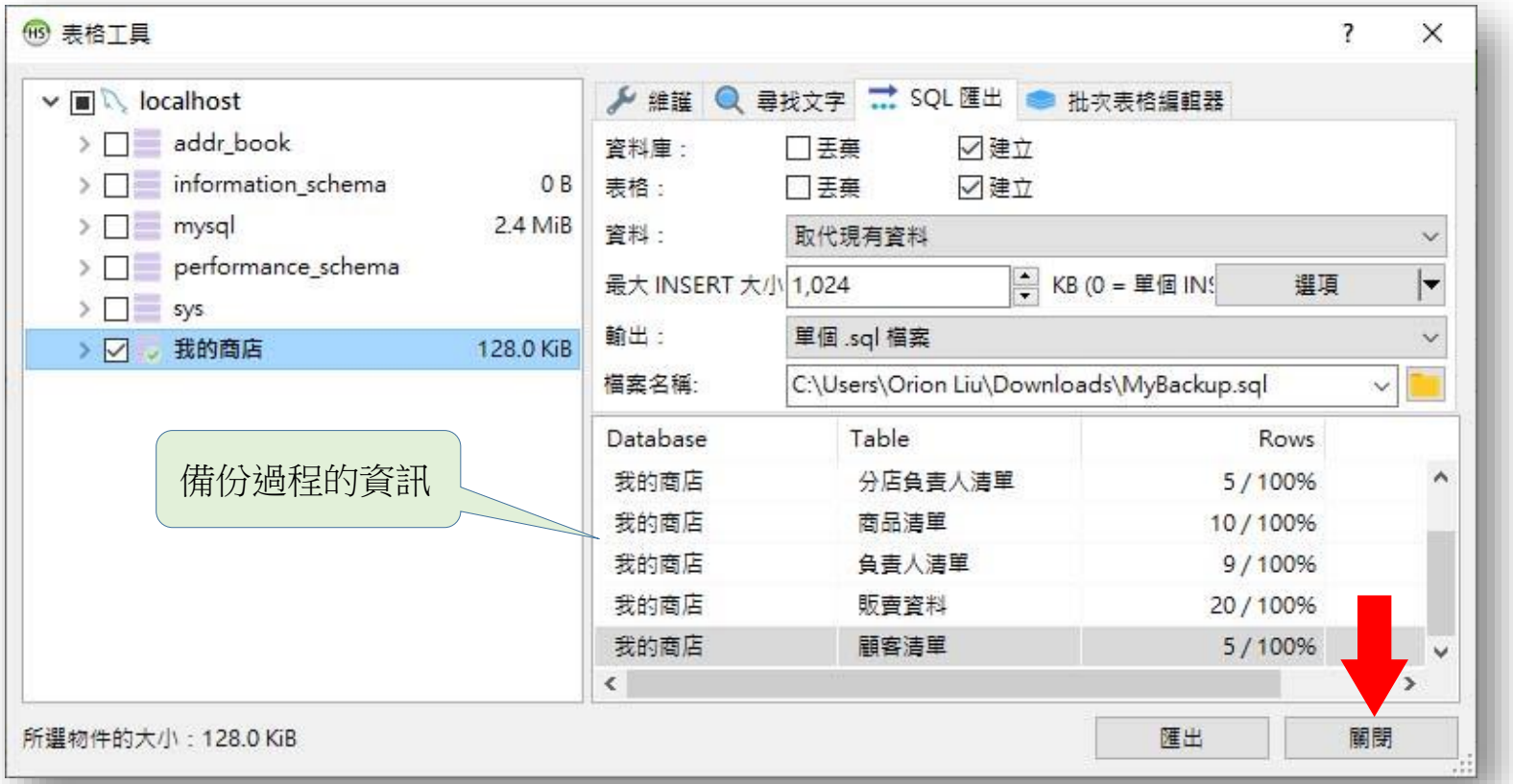
MySQL資料庫管理

- 備份資料庫(使用HeidiSQL工具)：



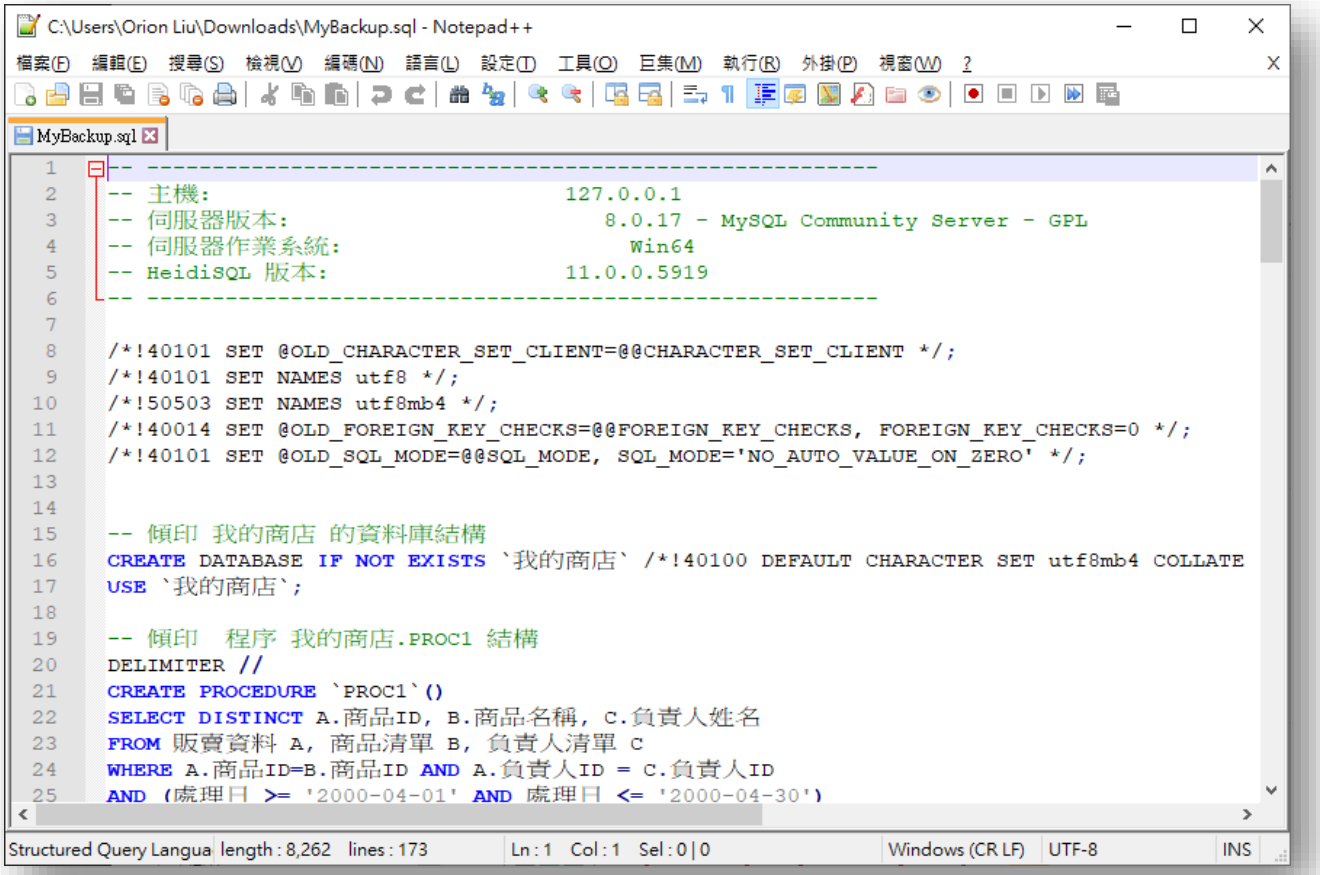
MySQL資料庫管理

- 備份資料庫(使用HeidiSQL工具)：



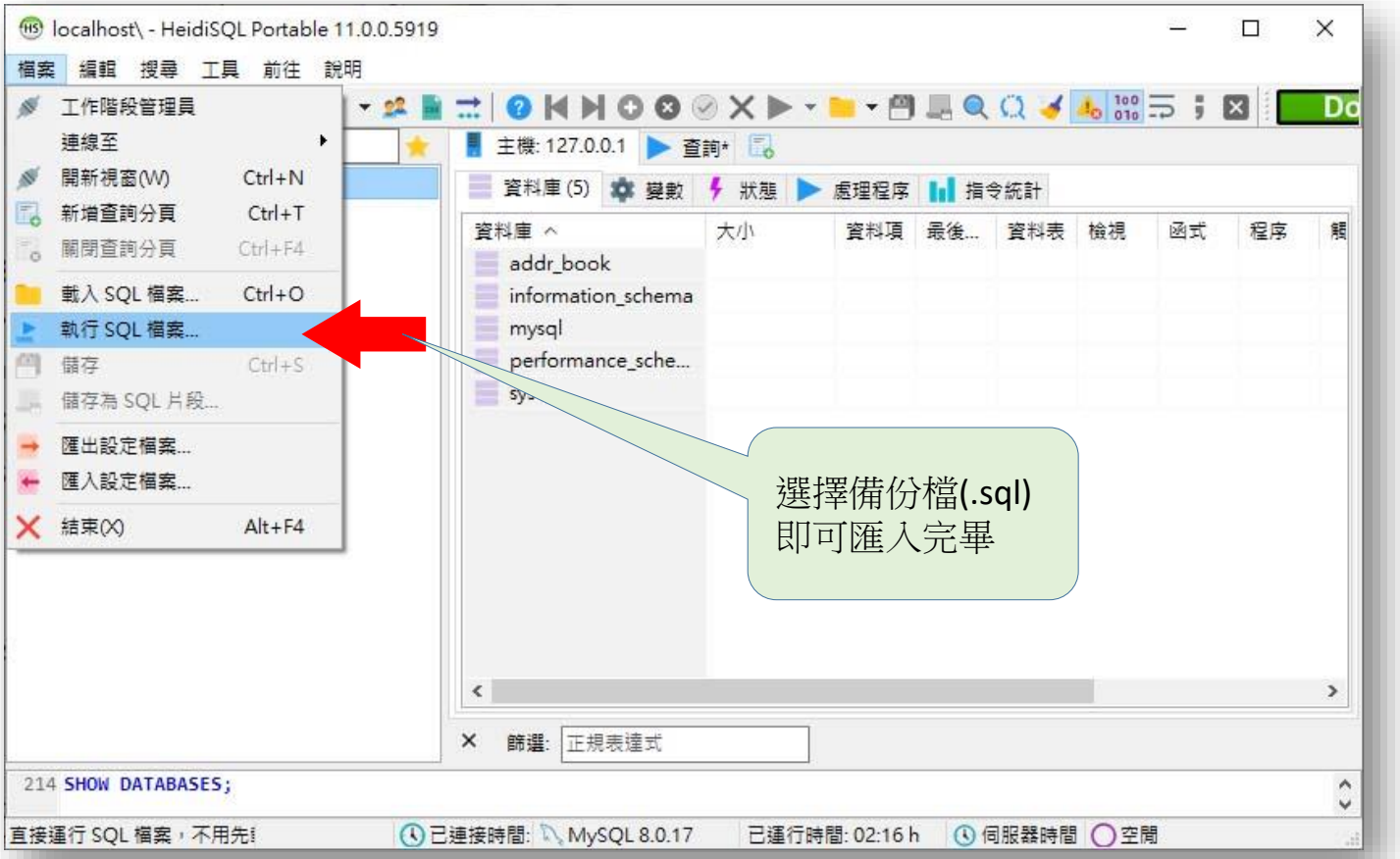
MySQL資料庫管理

- 備份檔就是一個文字格式的SQL命令檔，要還原資料庫時將此檔案匯入即可。



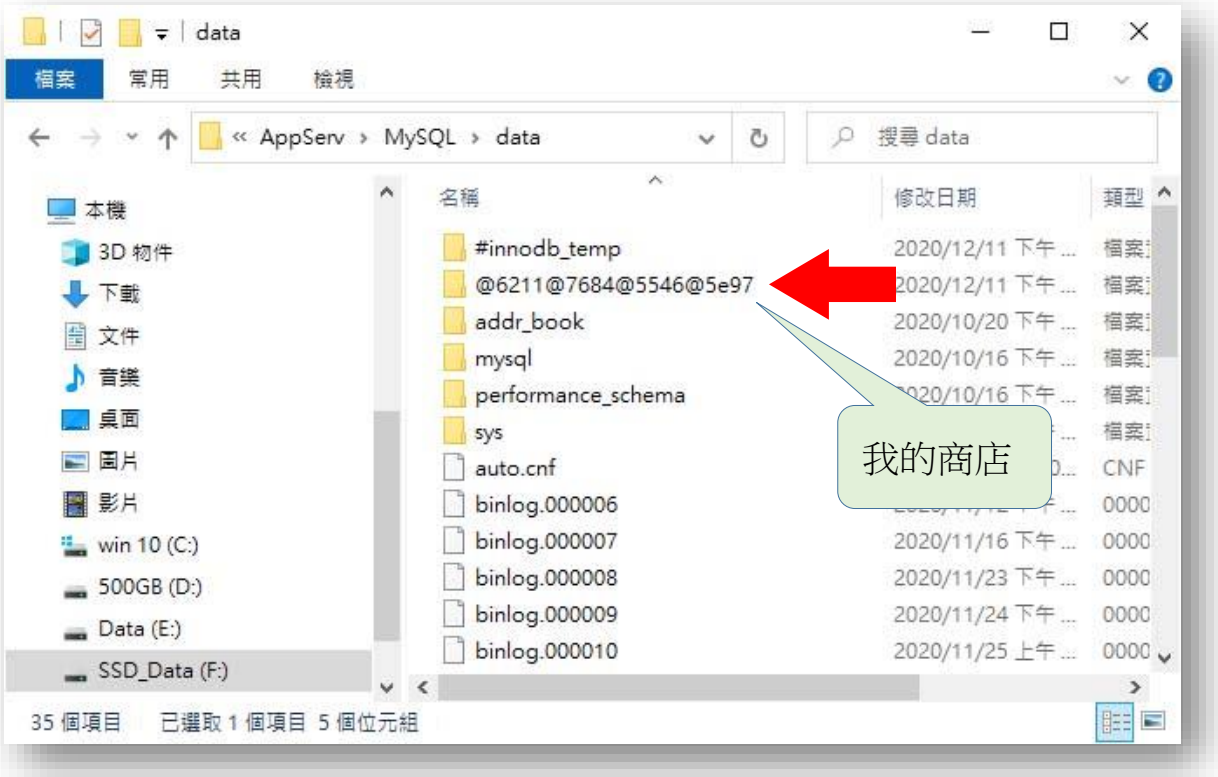
MySQL資料庫管理

- 匯入/還原資料庫(使用HeidiSQL工具)：
  - 只要簡單的一個動作就可以了。



MySQL資料庫管理

- 關於使用的編碼：
  - 不論是Win7或是Win10，在繁体中文系統下，檔案名稱皆使用Big5編碼。而MySQL就會很雞婆的把資料庫檔名使用Big5來代替，如圖：



MySQL資料庫管理

- 關於使用的編碼：
  - 一開始使用MySQL時，是可以很順利的操作。但若想要把資料庫備份出來，不使用如HeidiSQL等其他工具的話，必需使用如下指令：

```
C:\>mysqldump -u root -p --routines
              --default-character-set=big5
              中文資料庫名稱 > 檔名.sql
```

  - 上面的 default-character-set=big5 是指定備份的資料庫中文檔案名為Big5。

MySQL資料庫管理

- 關於使用的編碼：
  - 這下問題又來了，資料表或是其中的欄位也是中文的話，MySQL系統卻使用utf8，而且是不可變更的，所以會造成備份到一半就發生錯誤。如果把 default-character-set改成utf8，則中文資料庫名稱又找不到而無法備份了。
  - 使用MySQL官方的Workbench工具也是不行的。

- 所以不管是資料庫名稱、資料表名稱、欄位名稱，在MySQL中都不要使用中文來命名。

下課~~~