

多媒體網頁設計及互動程式設計文憑課程

Part II: Interactive Application Development (Lesson 22)

Content:

1. Introduction to PHP and server-side programming / Installation and Configuration of Apache 2 and PHP 5 on Windows platform
2. Basic PHP Syntax : Data Types, Variables, Arrays, Function and Loop
3. Introduction of Database Programming and MySQL / Installation and Configuration of MySQL 5 and related utilities
4. Implementing user authentication and authorization using cookie and session (Building a membership system)
5. SQL & Database Programming in PHP I (Corporate News with Content Management System)
6. SQL & Database Programming in PHP II (Search function, Interactive Enquiry Forms, Online Questionnaire and Hit Counter)
7. Online Product Catalogue with Content Management System
8. Shopping Cart and Payment Gateway Integration
9. Interactive Voting System and Interactive Bidding System
10. Online Forum Setup and Configuration using phpBB



Contents

3.1 關聯式資料庫 (RDBMS) 4

資料.....	4
資料模型.....	4
關聯式系統.....	5
表格.....	5
運算.....	6
INDEX.....	8
INDEX 的優點.....	8
INDEX 的缺點.....	10
用不用 INDEX ?.....	10
我會用 INDEX.....	11
我不用 INDEX.....	11
KEY.....	12
候選鍵 (CANDIDATE KEY).....	12
主鍵 (PRIMARY KEY) 與候補鍵 (ALTERNATE KEY).....	13
外鍵 (FOREIGN KEY).....	14
資料正規化 (NORMALIZATION).....	16
何謂正規化.....	16
目的.....	16
步驟.....	16
實例探討.....	17
第一正規化 (FIRST NORMAL FORM).....	18
第二正規化 (SECOND NORMAL FORM).....	20
第三正規化 (THIRD NORMAL FORM).....	22

3.2 結構化查詢語言 – SQL..... 25

何謂 SQL (結構化查詢語言).....	25
SQL 的分類.....	26
資料定義語言 (DATA DEFINITION LANGUAGE , DDL).....	26

資料操作語言 (DATA MANIPULATION LANGUAGE , DML)	26
資料控制語言 (DATA CONTROL LANGUAGE , DCL)	26
MySQL 的命名法則	26
建立、移除與選擇資料庫	27
建立、修改與刪除資料表	28
SQL 檢索資料	29
SELECT 語法	29
SELECT 進階	30
SQL 新增與異動資料	32
3.3 MYSQL 資料庫	34
MySQL 簡介	34
欄位型態	34
數字型態	35
日期與時間型態	37
字串型態	38
TEXT 與 BLOB	39
ENUM 與 SET	39
存取 MySQL	41
建立資料庫連線	41
選定欲存取的資料庫	42
執行查詢動作	42
取出查詢結果	43
中斷資料庫連線	44
3.4 安裝 MYSQL 附屬工具	45
MySQL ADMINISTRATOR (資料庫管理員)	45
MySQL QUERY BROWSER	51
PHPMYADMIN	57
3.5 練習	58
3.6 資源	61

3.1 關聯式資料庫 (RDBMS)

資料

在日常生活中，我們會面對許多不同型態的資料，如親朋好友的通訊資料、個人行程計畫等等。為了保存這些資料，我們會自定一些規則，將它們有組織地記錄在紙張或電腦上，以便將來取用。我們用這樣的方法長期保存資料，就算是一種「資料庫」。

資料模型

前面提到，我們會自定一些資料記錄的規則，將資料庫以固定的架構來組成；而用來表示資料庫如何組成的架構，稱為資料模型。關於資料模型的理論不少，其中最為著名的是「關聯式資料模型」(relational model of data)，這是 E. F. Codd 博士 (數學家，IBM 的研究人員) 於 1970 年在「A Relational Model of Data for Large Shared Data Banks」這篇論文中所提出的。

這項理論隨後不斷地被討論與修正，到 1980 年前後開始有「關連式」的資料庫產品上市；自此之後，資料庫方面的發展與研究幾乎都是「關連式」的天下了。

關聯式系統

什麼樣的資料庫系統才是「關連式」的呢？簡單地說，「關連式系統」就是：

使用者看到的都是表格。使用者可使用的運算子，都是從舊表格中產生新表格。這些運算子至少包括 RESTRICT(SELECT)、PROJECT 與 JOIN。

換句話說，「關連式系統」的特徵就是其利用表格來呈現資料，然後將表格視為集合來進行處理。當要操作資料時，便是針對表格去執行以集合理論為基礎的數學運算，而其執行結果還是表格。

表格

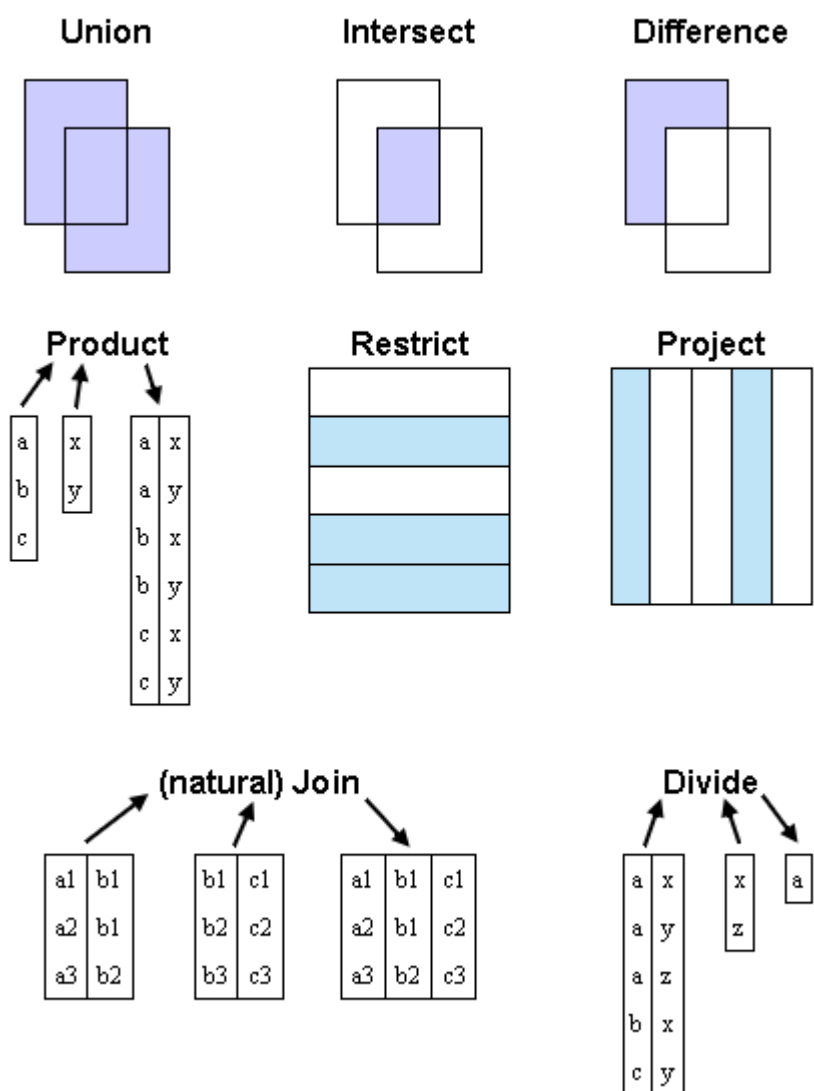
下圖是一個「表格」(table)，其中縱向的稱為「行」(column)，或是稱為「欄」(field)，存放著相同性質的資料。橫向的稱為「列」(row)，或是「記錄」(record)，裡頭包含許多不同性質的資料項目。這個表格有 5 欄 7 列。

The diagram shows a table with 5 columns and 7 rows. Labels with arrows point to specific parts of the table: 'Record' points to a horizontal row, 'Data Value' points to a single cell, 'Field' points to a column header, and 'Table' points to the entire grid.

Company	Address	City	State	Zip	Phone
2nd Byte Computers	6055 35th Ave SW #104	Seattle	WA	98126	(206)932-8816
Des Donnée Concepts	1314 East Pike	Seattle	WA	98122	(206)860-4333
3 Com Corp	10900 NE 8th St #900	Bellevue	WA	98004	(425)455-8530
4 I Software	7216 26th Ave NE	Seattle	WA	98115	(206)364-2337
4THPASS	83 S King St #100	Seattle	WA	98104	(206)749-9070
80-20 Software	3055 112th Ave NE #120	Bellevue	WA	98004	(425)739-6767
A & E Systems	11820 Northup Way #E108	Bellevue	WA	98005	(425)401-7171

運算

關連式資料模型是以數學的集合理論為基礎，Codd 博士定義了八個運算子 (operator)，用來對關聯式資料模型做數學運算。運算後的結果會再度成為一個資料表 (relations in, relations out)，而且這個資料表也能再度進行運算。透過這樣不斷地運算，便可導出所需的結果來。這八個運算子之中，前四個是傳統的集合運算(union, intersection, difference, and Cartesian product)，後四個則是特殊的關聯式運算(restrict, project, join, and divide)。



舉個例子來看，從通訊錄中取出所有性別為「女性」的資料列，這就是「restrict」運算；若取出「姓名」與「電話」這兩欄的所有資料，則屬於「project」運算。

Index

Index 的優點

先看看這個例子：請在下表中找出學號為「75120」者的成績。

75312	Chen	80
75524	Chuang	95
75207	Yeh	92
75302	Lee	90
75101	Chuang	89
75303	Ho	90
75120	Lin	92
75313	Chen	88

從表格的開頭逐筆尋找，您將在第 7 筆的位置發現目標，您總共找了 7 次。若針對「學號」這一欄建立「索引」(Index) 的話，則會像以下這個樣子：除了右側的原始表格之外，還會多出一個如左邊這樣的 index table；在 index table 中，學號已經被排序過了，但它仍記錄著每個學號與原始表格之間的對應關係。

75101		75312	Chen	80
75120		75524	Chuang	95
75207		75207	Yeh	92
75302		75302	Lee	90
75303		75101	Chuang	89
75312		75303	Ho	90
75313		75120	Lin	92
75524		75313	Chen	88

同樣要找出學號為「75120」者的成績，有了 index 之後將有些不同，我們改從 index table 中著手。

由於 index table 中的資料已經被排序過了，我們不必自表格開頭循序尋找。以「二分法」為例，8 筆資料取半數，直接看 index table 的第 4 筆記錄 (75302)；我們的目標 (75120) 比它小，所以再取 4 的半數，看看第 2 筆記錄 (75120)，結果就找到了。整個過程中，我們只找了 2 次。

有了 index 以後，不見得每次都會比較快，若是要找「75312」的話，在沒有 index 的情況下，反而可以一次命中目標。

那麼 index 對搜尋速度到底有沒有幫助呢？我們以數學的觀點來看這個問題。假如某個表格有 N 筆記錄，在沒有 index 的情況下，想找到特定記錄，最快的是 1 次 (目標恰好在開頭處)，最慢的則是 N 次 (目標恰好在末尾處)，平均得花上 $N/2$ 次才能找到目標。

有了 index，再使用「二分法」來搜尋的話，因為每找一次，即可去除一半的資料，所以平均只要 $\log_2 N$ 次即可。

若某個表格有 1024 筆記錄，沒 index 的話，平均得找 512 次 ($1024/2$) 才能找到；有了 index 之後，則可以在 10 次 ($\log_2 1024$) 以內交差。

Index 的缺點

效率差很多吧！真是這樣的話，我們將表格中的每個欄位都設為 index 的話，豈不是最好了嗎？這倒不見得，index 雖然有加速搜尋的優點，但也有以下兩項缺點：

1. 需要更多資料儲存的空間

本來只有一個原始表格，當我們將「學號」設為 index 之後，就多了一個 index table；若再將「姓名」設為 index 的話，又會多出一個 index table 來。在儲存成本日漸下滑的今日，您或許不太介意這種問題，但第二項缺點則是您不可忽視的。

2. 增加資料異動所需的時間

當我們在原始表格中加入一筆「75121」的成績記錄時，相關的 index table 也需要跟著同步更新。越多欄位被設為 index 的話，需要同步更新的 index table 也就更多，這樣一來，就會花費更多資料處理的時間。

用不用 Index ？

Index 的使用有提升搜尋效率的優點，但異動時間與儲存空間的耗費卻也是不可漠視的缺點。那麼，到底用不用 index 呢？其實當只有少量資料時，index 的好處與壞處都讓人感受不到，唯有在面對大量資料時，您才需要煩惱這個問題。

我會用 Index

當我將全高雄縣的師生基本資料建檔，且置放於同一表格時，這可能會有近十萬筆資料，不用 index 是不成的了。但要怎麼用才適當呢？首先，我會將「身分證字號」這種確定不重複的欄位設為 Primary key (同時具有 index 效果)；其次，將經常被用來檢索的欄位設為 index key，如「學校代號」。至於像「地址」這種又長又是中文內容的欄位，最好別打它的主意。

我不用 Index

有些行政應用系統注重資料安全性的問題，操作人員的一舉一動都會被記錄下來；在使用尖峰時段，同一秒之中就會有多筆資料同時被寫入，一天下來可能會有數十萬筆資料。面對這種資料表，我不會加上任何 index，因為它如同流水帳一般，不斷地新增資料，但無需進行異動。

像這樣的資料表，我會以「天」為單位，每天分別建立一個表格，日後需要查詢歷史資料時，再將舊表格取出來用。您或許會問：光是一天的資料就有數十萬筆了，沒有 index key 的話，查詢的效率豈不是很糟糕嗎？沒關係，因為它是歷史資料，已經沒有異動的需要了，這時候再為它加上 index key 的話，就利遠大於弊了。

Key

候選鍵 (Candidate Key)

什麼是「候選鍵」？簡單地說，它就是具有資格成為 primary key 的「候選人」。「候選鍵」可以由一至多個欄位組成，但它必須同時符合以下兩項條件才行。

身分證字號 班級 座號 姓名 成績

身分證字號	班級	座號	姓名	成績
T120123456	3-1	1	陳小華	89
E120654321	3-1	2	張小文	90
T120989898	3-1	3	林小明	78
...
S220567890	3-2	1	葉小花	95

1. 「唯一性」(uniqueness):

在被選定為「候選鍵」的欄位中，沒有任何兩組相同的記錄。以上表為例，「身分證字號」可以做為候選鍵，「班級 + 座號」也可以；但單單只有「班級」或只有「座號」都不行，因為「班級」這欄的有相同的記錄，「座號」這欄也是。

2. 「最簡性」(irreducibility):

由多欄位組成的「候選鍵」，其中的任何欄位組合都不能具有唯一性。

同樣以上表為例，「身分證字號 + 姓名」不可以做為候選鍵，因為「身分證字號」本身即具有唯一性了。

A	B	C
...
...

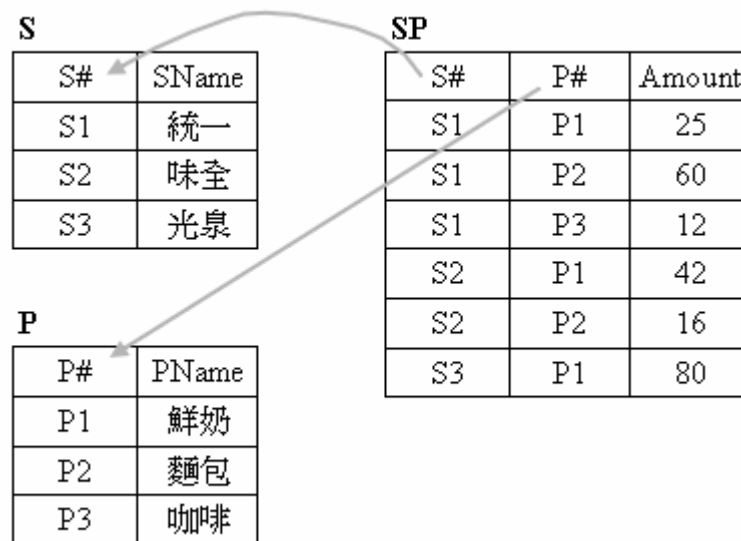
具有 n 個欄位的表格，最多可能擁有 $2^n - 1$ 個「候選鍵」。以上表為例，「A」、「B」、「C」、「A + B」、「A + C」、「B + C」與「A + B + C」都可能成為「候選鍵」。

主鍵（Primary Key）與候補鍵（Alternate Key）

您可以在一個表格中挑選一組「候選鍵」做為「主鍵」，其他沒被選上的「候選鍵」，則被稱為「候補鍵」。

有了「候選鍵」之後，我們可以用來區別每筆記錄，不致弄錯欲異動的對象。因此，雖說每個表格不見得一定要有「主鍵」，但一定至少要有一組「候選鍵」。

外鍵 (Foreign Key)



上圖之中有三個表格，分別是「供應商」(S)、「產品」(P)與「銷售統計」(SP)。其中，SP 的 S# 與 P# 是「外鍵」，它們分別對應到 S 的 S# 與 P 的 P#。S 的 S# 與 P 的 P# 都必須是「候選鍵」，在 SP 與它們對應時，才不會發生錯誤。

當 SP 表的 S# 值為 S1 時，在 S 表的 S# 之中，一定要有 S1 這個值。若將 S 表的 S1 值給刪除了，則 SP 表中 S# 為 S1 的所有資料列需要一併刪除。也就是說，資料庫中不能含有任何未相配的外鍵值，這樣才能維持參考完整性 (referential integrity)。

在異動「外鍵」所對應的內容時，必須連帶異動「外鍵」本身的值，或是做一些必要的限制。例如，欲異動 S 之 S# 的某筆資料時，需連帶異動 SP 之 S# 中相關的資料；或者是限制只能異動還沒被 SP 對應到的其它 S# (如 S4)。

在這樣的條件下，勢必將加重資料庫系統運作上的負擔。為了維持高的運作效率，MySQL 至 3.x 版為止，仍未真正地提供「外鍵」的功能。

資料正規化 (Normalization)

何謂正規化

將表格細分成多個更小的表格，直到每個表格只描述一種事實為止，這一連串的調整過程就稱為資料正規化 (Normalization)。

目的

正規化的目的何在？簡單的說就是要將資料的重覆性降至最低 (避免資料重複的狀況發生)。倘若在不同的表格中都有學生的姓名時，一旦有個學生改名了，則必須同步更改多個表格的內容；修改的過程中若稍有遺漏，有些資料沒更正，就會發生不一致的狀況。因此，避免資料重複是相當重要的。

步驟

- 第一正規化 (First Normal Form , 簡稱 1NF。由 E. F. Codd 提出)
- 第二正規化 (Second Normal Form , 簡稱 2NF。由 E. F. Codd 提出)
- 第三正規化 (Third Normal Form , 簡稱 3NF。由 E. F. Codd 提出)
- 第四正規化 (Fourth Normal Form , 簡稱 4NF。由 R. Fagin 提出)
- 第五正規化 (Fifth Normal Form , 簡稱 5NF。由 R. Fagin 提出)

實例探討

假設我們將要設計一個成績單郵寄列印系統，需要學號、地址、郵遞區號、學科代碼與各科成績等資料，而初步搜集到的原始資料如下表所示：

Stu_no	City	ZIP	Subject_no, Score
75312	上海	400	(S5302, 89), (S5345, 90), (S8005, 78), (S3581, 80), (M1201, 65), (M5251, 95)
75524	北京	800	(S5302, 88)
75302	天津	830	(S5302, 98), (S5345, 90), (S3581, 84), (M5251, 85)

接下來，我們將依序探討 1NF, 2NF 與 3NF 的過程。

第一正規化 (First Normal Form)

第一正規化的表格最重要的是能滿足「每個欄位只能含有一個值」這個條件。

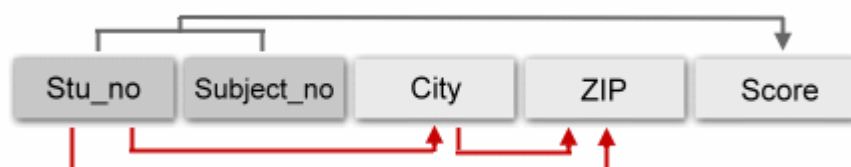
在正規化之後，我們將表格命名為 A：

《A》

Stu_no	City	ZIP	Subject_no	Score
75312	上海	400	S5302	89
75312	上海	400	S5345	90
75312	上海	400	S8005	78
75312	上海	400	S3581	80
75312	上海	400	M1201	65
75312	上海	400	M5251	95
75524	北京	800	S5302	88
75302	天津	830	S5302	98
75302	天津	830	S5345	90
75302	天津	830	S3581	84
75302	天津	830	M5251	85

結果探討：

在同一學生只能選修同科目一次的條件下，「Stu_no」加上「Subject_no」可以做為 A 的主鍵 (Primary key)。我們以下圖來說明主鍵與其他欄位之間在功能上的相依關係 (Functional Dependency)：



在 A 之中係以 (Stu_no, Subject_no) 為 Primary key , 但從上圖看來 , 有三項「功能相依」關係是錯誤的 (如紅線所示) , City 與 ZIP 的值與 Subject_no 絲毫無關。在這樣的架構下 , 將產生下列問題 :

1. 無法單獨新增一筆學生資料。因為 Subject_no 是 Primary key 之一 , 不能為空值 (Null); 因此 , 一個未修習任何課程學生的資料 , 將無法寫入 A。
2. 無法單獨刪除一筆成績資料。如果我們打算刪除 (75524, S5302) 這筆資料的話 , 該生的地址資料也將一併消失。
3. 需要同步異動的資料太多。假如 75312 這個學生搬家了 , 那麼我們得異動其中的 6 筆紀錄。

因此 , 我們得繼續進行 2NF。

第二正規化 (Second Normal Form)

一個表格必須滿意第一正規化的條件，並且非主鍵的欄位都要對主鍵有「完全地功能性相依 (Fully Functional Dependency)」關係，才能算是達到第二正規化。

在正規化之後，我們將表格 A 一分為二，並分別命名為 B1 與 B2：

《B1》

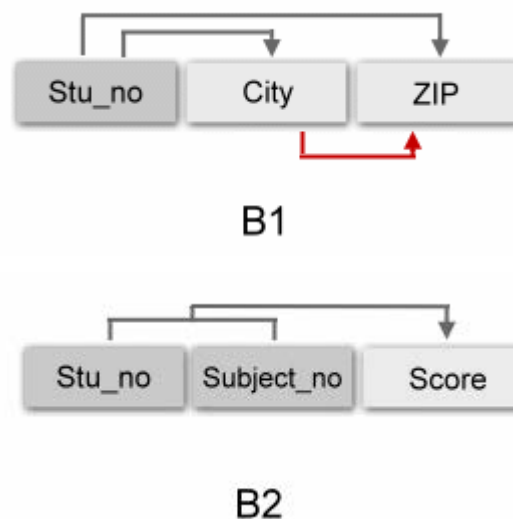
Stu_no	City	ZIP
75312	上海	400
75524	北京	800
75302	天津	830

《B2》

Stu_no	Subject_no	Score
75312	S5302	89
75312	S5345	90
75312	S8005	78
75312	S3581	80
75312	M1201	65
75312	M5251	95
75524	S5302	88
75302	S5302	98
75302	S5345	90
75302	S3581	84
75302	M5251	85

結果探討：

在經過 2NF 之後，先前的「無法單獨新增一筆學生資料」與「無法單獨刪除一筆成績資料」的問題都解決了。我們再看看 B1 與 B2 各欄位和主鍵之間在功能上的相依關係 (Functional Dependency)：



在一個表格中，如果某一欄位值可決定其他欄位值；而這些欄位中又存在某一欄位可以決定剩餘欄位的值，稱為「遞移相依性 (Transitive Dependency)」。若有此一情況發生，在異動資料時，可能會造成其他資料不一致的現象。在 B1 之中便有「遞移相依性」關係存在：B1.Stu_no -> B1.City 且 B1.City -> B1.ZIP。在這樣的架構下，將產生下列問題：

1. 無法單獨新增一筆縣市資料。因為 Stu_no 是 Primary key，不能為空值 (Null)；因此，若無任何學生居住的某個縣市，其郵遞區號資料將無法被事先建立。
2. 無法單獨刪除一筆學生資料。如果我們打算刪除 75524 這筆資料的話，該生所在的高雄市郵遞區號資料也將一併消失。

3. 仍有需要同步異動的資料。假如台中市的郵遞區號修改了，且住在該地區的學生又不只一位時，那麼我們又得異動多筆紀錄了。

因此，我們還得繼續進行 3NF。

第三正規化 (Third Normal Form)

一個表格必須滿意第二正規化的條件，並且消除「遞移相依」現象，意即非主鍵的欄位之間沒有「完全地功能性相依」關係，才能算是達到第三正規化。

已合乎 2NF 的表格 B1，正規化之後，我們將表格 B1 再度一分為二，並分別命名為 C1 與 C2：

《C1》

Stu_no	City
75312	上海
75524	北京
75302	天津

《C2》

City	ZIP
上海	400
北京	800
天津	830

結果探討

在經過 3NF 之後，先前的「無法單獨新增一筆縣市資料」與「無法單獨刪除一筆學生資料」的問題都解決了，需要同步異動大量資料的情況似乎也不復存在了。我們再以表格與欄位間的相依關係來看看正規化的結果：

Stu_no	City
75312	上海
75524	北京
75302	天津

City	ZIP
上海	400
北京	800
天津	830

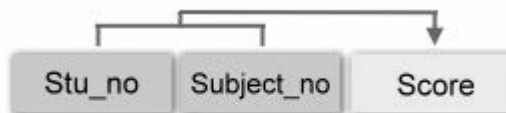
Stu_no	Subject_no	Score
75312	S5302	89
75312	S5345	90
75312	S8005	78
75312	S3581	80
75312	M1201	65
75312	M5251	95
75524	S5302	88
75302	S5302	98
75302	S5345	90
75302	S3581	84
75302	M5251	85



C1



C2



B2

一般表格進行至第三正規化時，多半沒有什麼狀況了；倘若仍有異常狀況發生，則需繼續進行 BCNF，甚至於 4NF 與 5NF，關於這個部份請自行參閱相關書籍。

正規化只是建立資料表的原則，而非鐵律。切莫因為過度正規化，反而導致資料存取的效率下降。有時在優先考量執行效率的前提下，我們還必須做適當的反正規化 (Denormalize)。

3.2 結構化查詢語言 – SQL

何謂 SQL (結構化查詢語言)

SQL 是「結構化查詢語言」(Structured Query Language) 的簡稱，讀作「Ess Que Ell」或「sequel」。SQL 最初是由 IBM 的研究中心在 1970 年代初期所開發的，是專門用於關連式資料庫的一種查詢語言。利用 SQL 可以用來定義資料庫結構、建立表格、指定欄位型態與長度，也能新增、異動或查詢資料，它已經成為關聯式資料庫的標準語言。

SQL 的標準化作業，主要是由 ANSI (美國國家標準學會) 與 ISO (國際標準組織) 這兩個組織所推動的。最初是在 1986 年由 ANSI 制定其標準化規格，隨後在 1992 年時再推出更新的版本，就是所謂的「SQL-92」、「SQL/92」或「SQL2」。目前，新一代的規格仍在持續發展中，即「SQL-99」規格。

目前市面上的資料庫產品幾乎都以支援 SQL-92 為主，但卻沒有任何一家支援完整的 SQL-92。它們都有少部份的功能不支援，但在某些方面的支援卻比 SQL-92 來得多。例如 IBM 的 DB2 不支援 SQL-92 所有的完整性法則，但在 VIEW 更新法則的支援上，卻比 SQL-92 多。像這些由軟體商獨自擴充的 SQL 是不具相容性的，如：Oracle 的 PL/SQL、Microsoft SQL Server 的 Transact-SQL。

SQL 的分類

資料定義語言 (Data Definition Language, DDL)

可以用來建立、更改或刪除 table、schema、domain、index 與 view。主要指令有三：CREATE、ALTER 與 DROP。

資料操作語言 (Data Manipulation Language, DML)

DML 係用來操作資料。主要指令有四：SELECT、INSERT、UPDATE 和 DELETE。

資料控制語言 (Data Control Language, DCL)

DCL 提供資料庫的安全性。主要指令有四：COMMIT、ROLLBACK、GRANT 和 REVOKE。

MySQL 的命名法則

在 MySQL 裡，database, table, index, column 與 alias 都依循以下的命名原則：

- Identifier Max length Allowed characters
- Database 64 除了「/」、「\」與「.»以外的其它字元
- Table 64 除了「/」與「.»以外的其它字元
- Column 64 所有字元
- Alias 255 所有字元

此外，您也不能使用 ASCII(0) 或 ASCII(255) 字元為上述任一 identifier 命名。如果 identifier 是一個限制詞或包含特殊字符，使用時必須用「`」來引用它，例如：

```
SELECT * FROM `select` WHERE `select`.id > 100
```

MySQL 對關鍵字與函數名稱是不分大小寫的，SELECT、select 與 sElEcT 的作用均相同。此外，index 與 column 也是不分大小寫。

Alias 則是大小寫分明的。即使以相同的字母來為 alias 命名時，大小寫相異便會被視為不同的 alias。

至於 database 與 table 的名稱就要視情況而定了。在 Windows 作業系統下，不區分大小寫；在 UNIX 系列的作業系統下，則會區分大小寫。

以下將介紹部份 MySQL 所支援的 SQL 敘述，其中有些用法可能不適用於其它的資料庫產品。

建立、移除與選擇資料庫

CREATE DATABASE：建立資料庫。

```
CREATE DATABASE [IF NOT EXISTS] db_name
```

DROP DATABASE：移除資料庫。

```
DROP DATABASE [IF EXISTS] db_name
```

USE：選取將連線的資料庫。

```
USE db_name
```

建立、修改與刪除資料表

CREATE TABLE：建立資料表

其實在實作時，有許多設定可以直接引用其預設值，因此語法得以簡化不少。以下就是兩個 CREATE TABLE 的使用實例：

```
CREATE TABLE friend (
    sn INT(4) NOT NULL AUTO_INCREMENT,
    realname CHAR(10),
    address CHAR(50),
    phone CHAR(15),
    PRIMARY KEY (sn)
);
CREATE TABLE new_friend SELECT * FROM old_friend
```

ALTER TABLE：修改現有的資料表

ALTER TABLE 在 MySQL 裡功用很多，不但可以用來建立或刪除 index，也可以用來更改資料表的名稱或結構。要使用時，您只要指定 tbl_name，然後就可以在後頭寫上一到多組異動的敘述。例如：

```
增加兩個欄位：ALTER TABLE friend ADD email CHAR(50), ADD age INT
刪除一個欄位：ALTER TABLE friend DROP email
加上 INDEX：ALTER TABLE friend ADD INDEX (realname)
刪除 INDEX：ALTER TABLE friend DROP INDEX realname
```

DROP TABLE：移除資料表

```
DROP TABLE tbl_name
```

SQL 檢索資料

以下將繼續介紹部份 MySQL 所支援的 SQL 敘述，其中有些用法可能不適用於其它的資料庫產品。

SELECT 語法

SELECT 被用來取得我們所要的資料。它是 SQL 中最常用，也是最複雜的指令。我們將語法中最常用的部份取出，可以簡化成為以下這種形式：

SELECT column_list	要檢索哪些欄位
FROM table_list	要以哪些資料表為目標
WHERE primary_constraint	必須滿足哪些條件
GROUP BY grouping_columns	如何產生 group
ORDER BY sorting_columns	如何對結果排序
HAVING secondary_constraint	其次要滿足哪些條件
LIMIT count	限制資料輸出的筆數

以實例來看：

例一，讀取所有資料，但不設定條件：

SELECT * FROM friend

例二，讀取部份欄位的資料，且指定條件：

SELECT realname, address FROM friend WHERE age > 20

SELECT 進階

ORDER BY

如果我們希望查詢所得的資料能依序排列的話，可以使用 ORDER BY 來達成目的。

例一，升冪排序（這是預設的排列方式，因此 ASC 可以省略不寫）：

```
SELECT first_name, address FROM friend ORDER BY city ASC
```

例二，降冪排序：

```
SELECT first_name, address FROM friend ORDER BY city DESC
```

例三，同時對兩個欄位排序（前者優先）：

```
SELECT first_name, address FROM friend ORDER BY city DESC, age
```

GROUP BY

我們可以使用 GROUP BY 來指定某個特定欄位，以便將查詢結果區分為若干個群組，然後對這些群組進行計算。

例一，從通訊錄中列出所有親友居住的縣市名稱，並求出各地居住的人數：

```
SELECT city, COUNT(*) FROM friend GROUP BY city
```

例二，依親友所居住的縣市來區分，並求出各地親友的平均年齡：

```
SELECT city, AVG(age) FROM friend GROUP BY city
```

例三，依親友所居住的縣市來區分，並求出各地親友家中成員的總數：

```
SELECT city, SUM(member) FROM friend GROUP BY city
```

HAVING

HAVING 的功能類似於 WHERE，可以用來規範資料被讀取的條件，但它必須與 GROUP BY 搭配，不能用來取代 WHERE。而且 HAVING 是在資料被取出之後，才再次進行篩選的動作。

例如，以學生的分組為單位，列出測驗平均成績高於 60 分的各組與其成績：

```
SELECT group_num, AVG(score) FROM exam  
GROUP BY group_num HAVING AVG(score) > 60
```

DISTINCT

假如我們在意的是「某個欄位裡是否有某些特定的值」，而不在乎它出現多少次的話，我們可以使用 DISTINCT 來剔除欄位中重複的值。

例如，從通訊錄中列出所有親友居住的縣市名稱：

```
SELECT DISTINCT city FROM friend
```

LIMIT

如果在某項查詢結果中，我們想要的僅是其中一部份時，可以使用 LIMIT 來限制資料被讀取的筆數。

例如，找出座號最前面的五位學生姓名：

```
SELECT realname FROM student ORDER BY num LIMIT 0, 5
```

在此例中，LIMIT 後方的 0 代表「從第 0 筆資料開始」，5 表示「取出 5 筆資料」。

SQL 新增與異動資料

INSERT 語法

INSERT 語法是將資料寫入資料表中，最簡便的一種方法。

```
INSERT [LOW_PRIORITY | DELAYED] [IGNORE]
      [INTO] tbl_name [(col_name,...)]
      VALUES ((expression | DEFAULT),...),(...),...
or
INSERT [LOW_PRIORITY | DELAYED] [IGNORE]
      [INTO] tbl_name [(col_name,...)]
      SELECT ...
or
INSERT [LOW_PRIORITY | DELAYED] [IGNORE]
      [INTO] tbl_name
      SET col_name=(expression | DEFAULT), ...
```

例如：

```
INSERT INTO salary VALUES ( '01', '3000' )
INSERT INTO salary VALUES ( '01', '3000' ), ( '02', '2000' )
INSERT INTO salary ( employee_id, amount ) VALUES ( '01', '3000' )
INSERT INTO salary SELECT employee_id, amount FROM other_table
INSERT INTO salary SET employee_id = '01', amount = '3000'
```


UPDATE 語法

UPDATE 可以被用來異動資料表中現存的資料。

```
UPDATE [LOW_PRIORITY] [IGNORE] tbl_name
    SET col_name1 = expr1 [, col_name2 = expr2, ...]
    [WHERE where_definition]
    [ORDER BY ...]
    [LIMIT #]
```

例如：

```
UPDATE friend SET phone = '07-7235300', city = 'Kaohsiung'
WHERE realname = 'David Chan'
UPDATE friend SET age = age + 1
```

DELETE 語法

使用 DELETE 可以自資料表中刪除部份的資料列。

```
DELETE [LOW_PRIORITY] [QUICK] FROM table_name
    [WHERE where_definition]
    [ORDER BY ...]
    [LIMIT rows]
or
DELETE [LOW_PRIORITY] [QUICK] table_name[*] [,table_name[*] ...]
    FROM table-references
    [WHERE where_definition]
or
DELETE [LOW_PRIORITY] [QUICK]
    FROM table_name[*], [table_name[*] ...]
    USING table-references
    [WHERE where_definition]
```

例如：

```
DELETE FROM friend WHERE realname = 'David Chan'
```

3.3 MySQL 資料庫

MySQL 簡介

MySQL 是一個快速、多執行緒 (multithread)、多使用者且功能強大的關聯式資料庫管理系統 (relational database management system, RDBMS)，可以與 C、C++、Java、Perl、PHP 等語言很容易的連結，可以運行於多種平台上，例如：Solaris、RedHat、Linux、FreeBSD、OS/2、Windows 等等。

MySQL 並不是一個 Open Source 的計劃，因為其版權在某些情況下是需要付費的，例如將它與其他產品包裝販售。不過，大體上來說，個人及非營利單位使用它是免費的，而 MySQL 所收取的授權費主要來協助 MySQL 研發所需，使 MySQL 得以更加茁壯。

請注意，「MySQL」的官方發音是「My Ess Que Ell」，不是「my sequel」。

欄位型態

建立資料表時，我們必須定義每個欄位所採用的資料型態，以下將介紹 MySQL 所支援的各種欄位型態。

數字型態

型態	空間需求	範圍
TINYINT[(M)]	1 byte	Signed: -128 to 127 (-2^7 to 2^7-1) Unsigned: 0 to 255 (0 to 2^8-1)
SMALLINT[(M)]	2 bytes	Signed: -32768 to 32767 (-2^{15} to $2^{15}-1$) Unsigned: 0 to 65535 (0 to $2^{16}-1$)
MEDIUMINT[(M)]	3 bytes	Signed: -8388608 to 8388607 (-2^{23} to $2^{23}-1$) Unsigned: 0 to 16777215 (0 to $2^{24}-1$)
INT[(M)] INTEGER[(M)]	4 bytes	Signed: -2147483648 to 2147483647 (-2^{31} to $2^{31}-1$) Unsigned: 0 to 4294967295 (0 to $2^{32}-1$)
BIGINT[(M)]	8 bytes	Signed: -9223372036854775808 to 9223372036854775807 (-2^{63} to $2^{63}-1$) Unsigned: 0 to 18446744073709551615 (0 to $2^{64}-1$)
FLOAT(precision)	4 or 8	若 precision ≤ 24 的話，視為 FLOAT（單精數） 若 $25 \leq \text{precision} \leq 53$ 的話，則視為 DOUBLE（倍精數）
FLOAT[(M,D)]	4 bytes	$\pm 1.175494351\text{E}-38$ $\pm 3.402823466\text{E}+38$
DOUBLE[(M,D)] REAL[(M,D)]	8 bytes	$\pm 1.7976931348623157\text{E}+308$ $\pm 2.2250738585072014\text{E}-308$
DECIMAL[(M[,D])] DEC[(M[,D])] NUMERIC[(M[,D])]	M+2	依 M 與 D 值而定

上表中的 M 代表「最大顯示寬度」，其值不得大於 255。無論您將欄位的型態設定為「INT(4)」或「INT(5)」，都不影響它儲存數值的能力；但在顯示時，就可以發現其差異了。我們將兩個欄位分別設為「INT(4)」與「INT(5) ZEROFILL」（ZEROFILL 會在前頭的空位補 0），然後兩者均

存入數字 4，在顯示其資料時，前者是「4」，後者則是「00004」。上表中的 D 代表「小數位數」，其值不得大於 30，也不能大於 M-2。

如果您將上述欄位型態設定為 UNSIGNED 的話，對整數型態 (TINYINT, SMALLINT, MEDIUMINT, INT, and BIGINT) 而言，可以儲存較大的正整數；對其餘的浮點數型態而言，則可以避免被存入負的數值。

若您存入的數值超過該欄位的範圍時，MySQL 只會取其所能處理的最大值。例如，您在 TINYINT 型態的欄位中存入「300」這個值，結果將只剩下「127」。

日期與時間型態

型態	空間需求	範圍
DATE	3 bytes	'1000-01-01' to '9999-12-31'
TIME	3 bytes	'-838:59:59' to '838:59:59'
DATETIME	8 bytes	'1000-01-01 00:00:00' to '9999-12-31 23:59:59'
TIMESTAMP[(M)]	4 bytes	自 1970 年起，至 2037 年的某時
YEAR[(2 4)]	1 bytes	4-digit format: 1901 to 2155 2-digit format: 1970 to 2069

在使用 TIMESTAMP 型態時，您可以指定「最大顯示寬度」，就是 M。不同的 M 值與儲存所需空間無關，而是與顯示的格式有關。請見下表：

型態	顯示格式
TIMESTAMP(14)	YYYYMMDDHHMMSS
TIMESTAMP(12)	YYMMDDHHMMSS
TIMESTAMP(10)	YYMMDDHHMM
TIMESTAMP(8)	YYYYMMDD
TIMESTAMP(6)	YYMMDD
TIMESTAMP(4)	YYMM
TIMESTAMP(2)	YY

字串型態

型態	空間需求	最大長度
CHAR(M)	M bytes	M bytes
VARCHAR(M)	L+1 bytes	M bytes
TINYBLOB, TINYTEXT	L+1 bytes	28-1 bytes
BLOB, TEXT	L+2 bytes	2 ¹⁶ -1 bytes
MEDIUMBLOB, MEDIUMTEXT	L+3 bytes	2 ²⁴ -1 bytes
LONGBLOB, LONGTEXT	L+4 bytes	2 ³² -1 bytes
ENUM('value1','value2',...)	1 or 2 bytes	65535 個成員
SET('value1','value2',...)	1, 2, 3, 4, or 8 bytes	64 個成員

上表中的 L 代表「實際儲存的空間大小」，上述多種型態的空間需求都與實際存入的空間大小有關，意即，它們的空間需求是變動的。

使用 CHAR 及 VARCHAR 型態時，必須宣告「最大儲存長度」，就是上表中的 M。這兩種型態是相似的，所能儲存的最大長度都是 255 bytes。其相異之處在於 CHAR 是個固定長度的型態，而 VARCHAR 是個長度可變的型態。我們從下表來看兩者在資料儲存上的表現：

字串內容	CHAR(4)	空間需求	VARCHAR(4)	空間需求
"	' '	4 bytes	"	1 byte
'ab'	'ab '	4 bytes	'ab'	3 bytes
'abcd'	'abcd'	4 bytes	'abcd'	5 bytes
'abcdefgh'	'abcd'	4 bytes	'abcd'	5 bytes

由於使用 CHAR 型態儲存資料時，MySQL 會以空白字元填補至最大儲存長度，所以無論存入的內容為何，所需的儲存空間都是固定的。當資料內容少於最大儲存長度時，VARCHAR 將可以有效地節省空間，這是它最大的優點；檢索時擁有較佳的效率，則是 CHAR 的長處。

VARCHAR 和各類 TEXT 與 BLOB 都是長度可變的型態。當一資料表中同時選用 CHAR 和這類長度可變的型態時，CHAR 會被自動改為 VARCHAR，除非它的最大儲存長度少於 4。

另一方面，若資料表中 VARCHAR 欄位的最大儲存長度少於 4 時，其型態也會被自動改為 CHAR。因為少於 4 bytes 的欄位在節省空間方面，實在沒多大的效果，不如改為 CHAR 以加快檢索的速度。

TEXT 與 BLOB

BLOB 的全名是「binary large object」，與 TEXT 一樣，都是用來儲存長度較長的字串或是二元資料。兩者大同小異，唯一的差別在於，TEXT 是有區分大小寫的，而 BLOB 不分。

ENUM 與 SET

ENUM 與 SET 是特別的字串型態，有人稱之為「列舉 (enumeration)」型態。這兩種欄位的值只能從固定的項目中挑選，不能隨心所欲的存入資料。舉個例子來看，我們想調查使用者的性別與年齡分布狀況，所以設定了以下兩種欄位：

```
sex ENUM("M", "F")
age ENUM("0-20", "21-30", "31-40", "41-50", "51-100")
```

若使用者填答時輸入「男性，32 歲」，我們則要用程式處理它，分別以 sex 的第 1 個選項與 age 的第 3 個選項存入。像 age 這個欄位，它已經將原始的輸入值以分組的形式來儲存，日後我們將無法得知使用者當時所輸入的精確數值。

ENUM 型態最多可以建立 65535 個不同的成員，而 SET 型態少得多，只能建立 64 個不同的成員。除了「容量」有別之外，兩者還有一項相異之處：SET 型態可以從中挑選一組以上的值，而 ENUM 只能選其一。換句話說，ENUM 型態是單選，而 SET 型態則是複選。

```
subject SET("Chinese", "English", "Math", "Music", "Art", "Sport")
```

這是一個 SET 型態的例子，使用者在從上述 subject 欄位挑選所修習的科目時，就可以同時挑選多個項目。

存取 MySQL

透過網頁存取資料庫內容的程式，多半要遵循以下幾個步驟來進行：

建立資料庫連線

1. 選定欲存取的資料庫
2. 執行查詢動作
3. 取出查詢結果
4. 中斷資料庫連線

以下將針對這幾個步驟一一加以說明。

建立資料庫連線

我們將存取資料庫的過程，想像成一艘貨輪要進港載貨。首先，港務局 (MySQL) 會對我們進行身分驗證，以確定是否為貨物的擁有者。在通過身分驗證之後，我們將會得到張「識別證」。

一般而言，港務局會開放 3306 號港口 (port) 供我們上下貨物，因此我們不必多費唇舌，可以直接駛入其中；否則，我們還要將貨輪駛向特定的港口停放。

```
$link = mysql_connect(HOSTNAME, USERNAME, PASSWORD);  
$link = mysql_connect(HOSTNAME:PORT, USERNAME, PASSWORD);
```

上面這行敘述在對 MySQL 表明存取者的身分。其中的 HOSTNAME 是「主機名稱」，若 Web server 與 MySQL 同在一部機器上的話，可以寫成「localhost」，否則請寫資料庫主機的名稱或 IP。USERNAME 與 PASSWORD 所指的就是要存取 MySQL 的帳號密碼了。

一般的情況下，MySQL 會透過 3306 port 運作。若系統使用這項預設值的話，我們可以不用再加指定；否則請額外加入 PORT 參數來說明，如：「localhost:9000」。

透過 `mysql_connect()` 的運作，MySQL 可以確認資料存取者的身分。成功的話，將回傳一個「link identifier」；否則傳回 FALSE。

選定欲存取的資料庫

取得通行許可之後，接著進入貨物區。這裡可能會有許多間貨品倉庫（資料庫），我們還要向管理員出示「識別證」，並從中挑選一間倉庫。

```
mysql_select_db(DBNAME);  
mysql_select_db(DBNAME, Identifier);
```

上面的 DBNAME 指的是「資料庫名稱」，當面對單一資料庫時，Identifier 可以省略不寫。

由於存取每個資料庫都需要一個個別的 link identifier，所以若我們想同時存取兩個資料庫時，必須建立兩個 link identifier，並在 `mysql_select_db()` 裡指明哪一個 link identifier 將對應到哪一個資料庫。

執行查詢動作

此時，無論是要將貨物搬上船，或是將船上的貨品存放到倉庫中，請將工作清單（QueryString）拿出來，然後就可以開工了。

```
$result = mysql_query(QueryString);  
$result = mysql_query(QueryString, Identifier);  
$result = mysql_db_query(DBNAME, QueryString, Identifier);
```

QueryString 即我們對資料庫的查詢動作，它可能是一段資料查詢的敘述，也可能是新增、修改或刪除等異動資料的要求。當 \$result 為 FALSE 時，表示這個存取動作是失敗的；否則，它將會是一個編號 (ID)。在「取出查詢結果」的步驟，我們得靠它來識別這次的查詢結果。

在使用 mysql_query() 時，Identifier 是可以省略不寫的。若我們不加以指定的話，系統會自動選用最近一次被使用的 link identifier。

如果我們先前「選定欲存取的資料庫」這個步驟略過的話，現在可以改用 mysql_db_query() 來進行資料查詢的動作。也就是說，mysql_db_query() 的功能等同於 mysql_select_db() 加上 mysql_query()。

取出查詢結果

取得貨物以後，拆開來看吧！裡頭分裝成一箱一箱的，我們可以算算有多少箱，也可以再把箱子拆開，取出裡面的貨品來。

在上個步驟中，若我們執行的是查詢動作的話，MySQL 會將結果包裝成一個 result set，並傳回它的編號 (ID)，接下來要讀取資料時，都少不了它。

```
$number_of_rows = mysql_num_rows($result);
```

藉由 mysql_num_rows() 函數，我們可以得知這個 result set 裡頭有多少個資料列 (row)。

最後，再透過迴圈的運作，將 result set 裡的每個資料列一一讀出來。由於每個資料列可能含有多個欄位，所以 \$data 將會是一個陣列；

要一一顯示個別欄位內容的話，我們可以用 `$data[0]`、`$data[1]`、`$data[2]`、..... 的方式。

```
for ( $i=0; $i<$number_of_rows; $i++ ) {  
    $data = mysql_fetch_row($result);  
    echo $data[0];  
    .....  
}
```

要不然，使用以下這種方式也行：將各個欄位的值存入相對應的變數中。

```
for ( $i=0; $i<$number_of_rows; $i++ ) {  
    list($value1, $value2, $value3, .....)= mysql_fetch_row($result);  
    echo $value1;  
    .....  
}
```

中斷資料庫連線

任務達成，向港務局 say good bye 吧！

```
mysql_close(Identifier);
```

`mysql_close()` 可以用來關閉一個由 `mysql_connect()` 所建立的資料庫連結，但是這個動作不是必要的。因為當一支程式執行結束之後，資料庫連結會被自動中斷。

3.4 安裝 MySQL 附屬工具

MySQL Administrator (資料庫管理員)

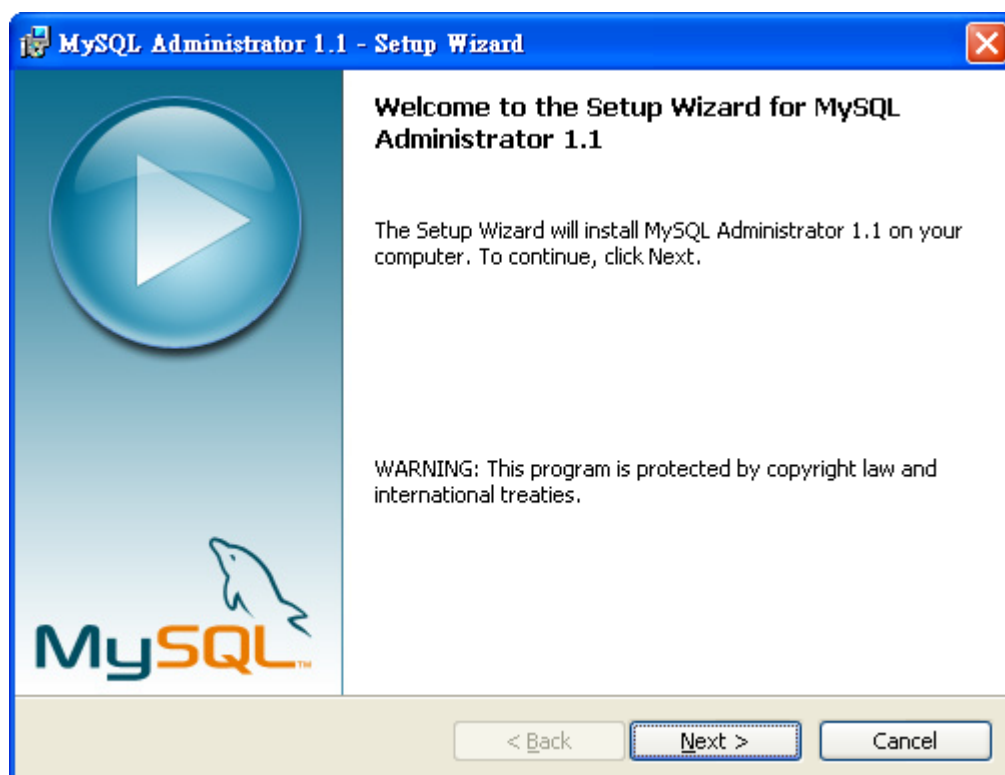
我們可以透過 MySQL Administrator 對而安裝的 MySQL 資料庫進行監察及設定。

1. 取得 MySQL Administrator :

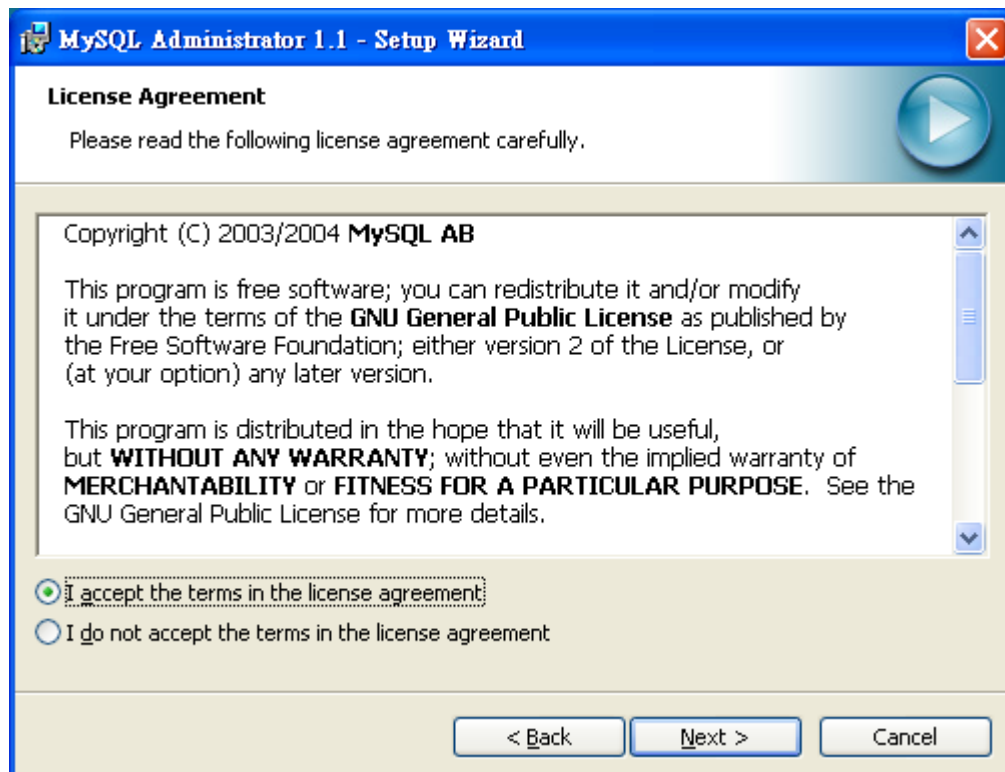
<http://mysql.isu.edu.tw/Downloads/MySQLAdministrationSuite/mysql-administrator-1.1.9-win.msi>

<http://mysql.mirror.vmmatrix.net/Downloads/MySQLAdministrationSuite/mysql-administrator-1.1.9-win.msi>

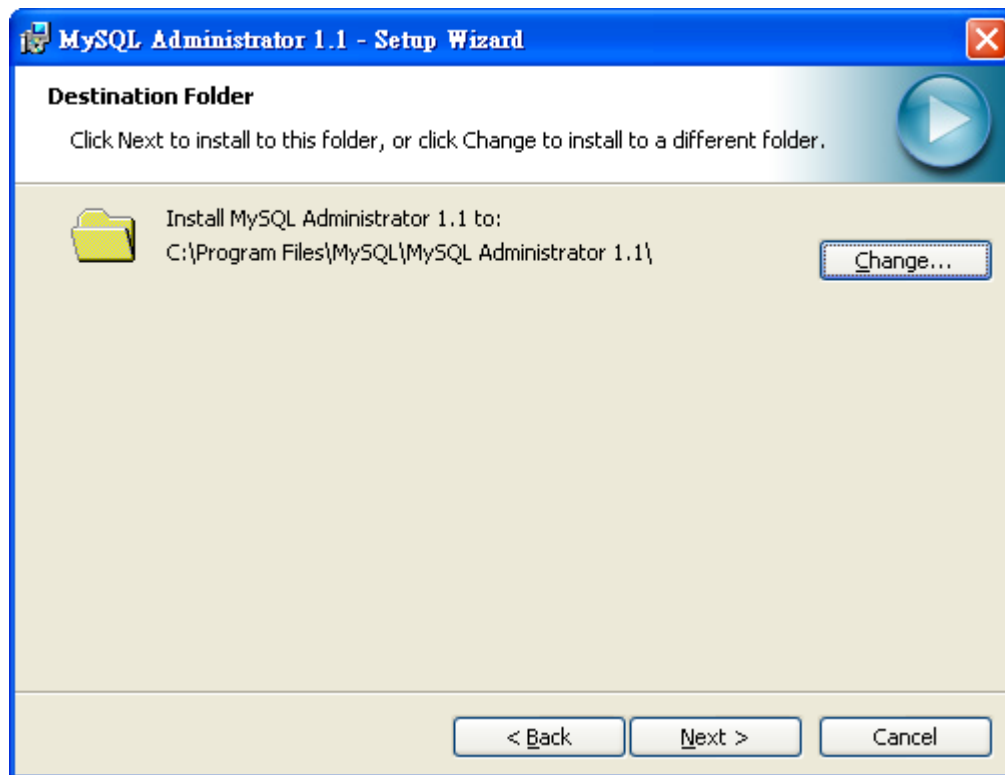
2. 下載完成後執行 mysql-administrator-1.1.9-win.msi 檔。



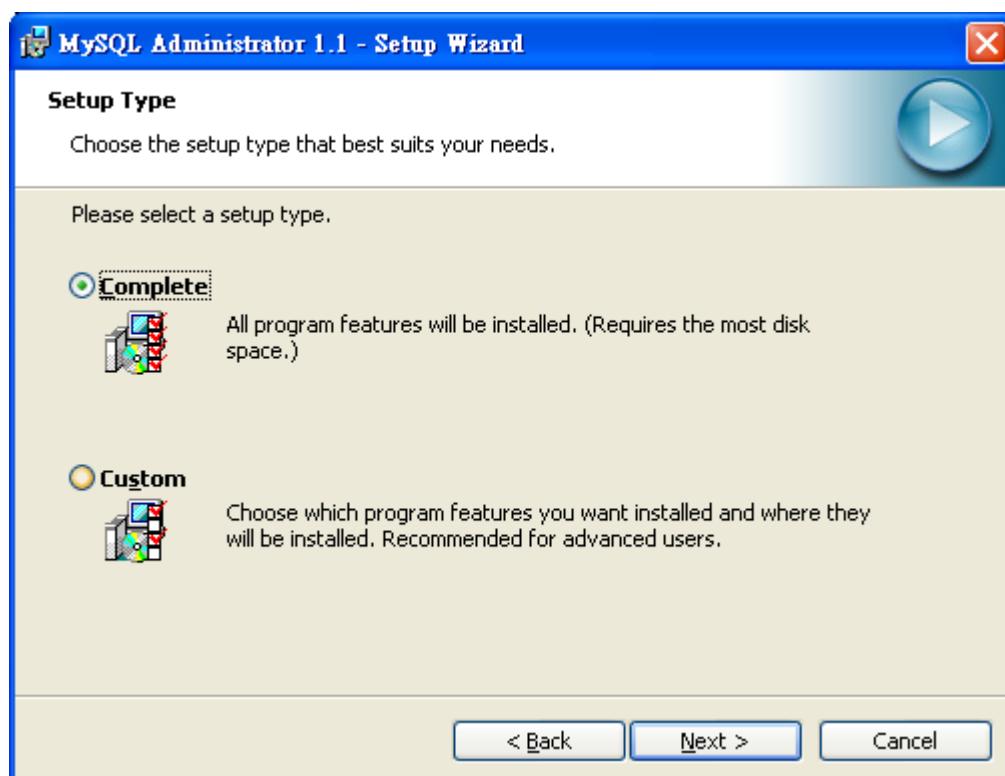
3. 按 Next 後選擇 I agree the terms in the license agreement 。然後按 Next 。



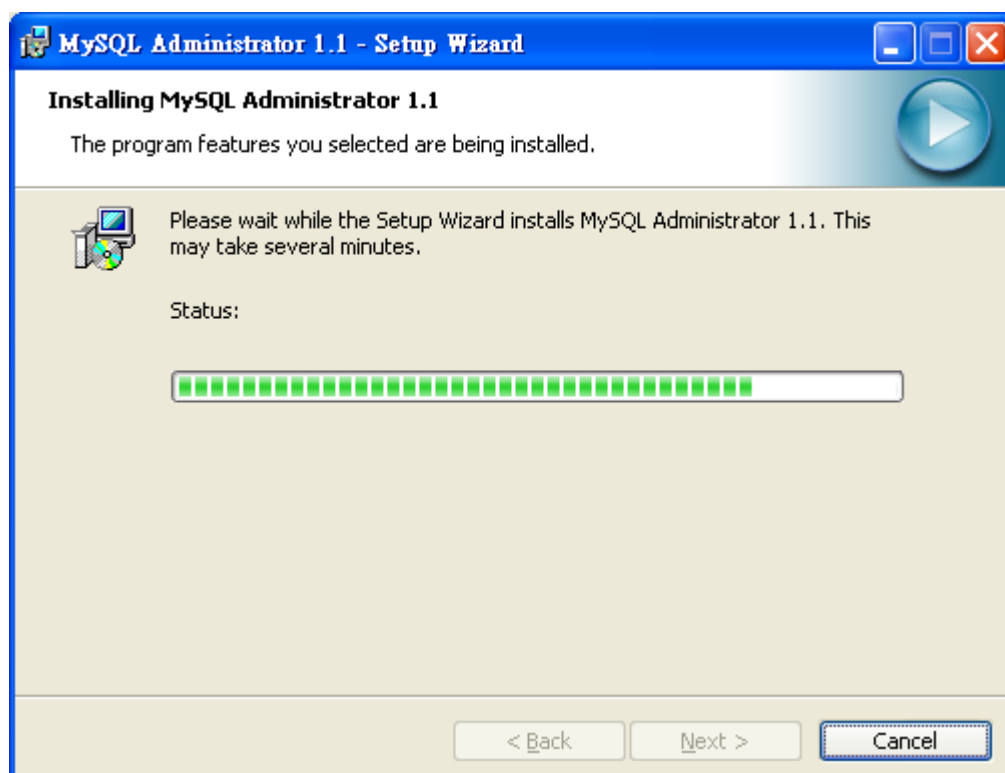
4. 選擇要安裝的位置： C:\Program Files\MySQL\MySQL Administrator 1.1\ 。完成後按 Next 。



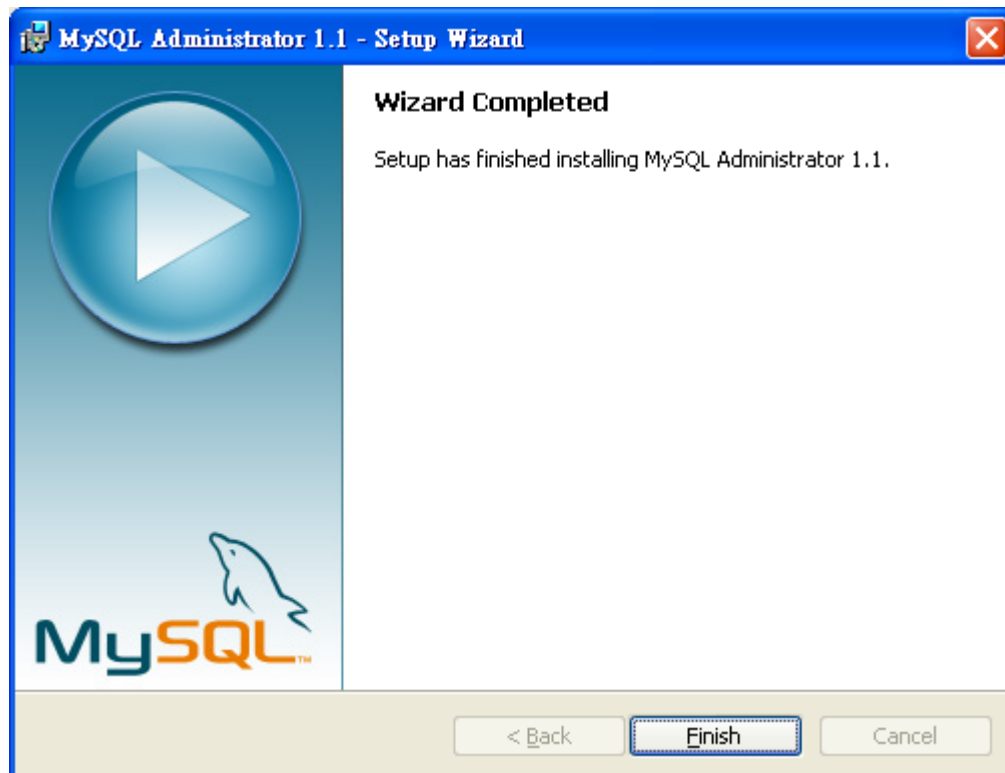
5. 選擇 Complete 並按 Next 。



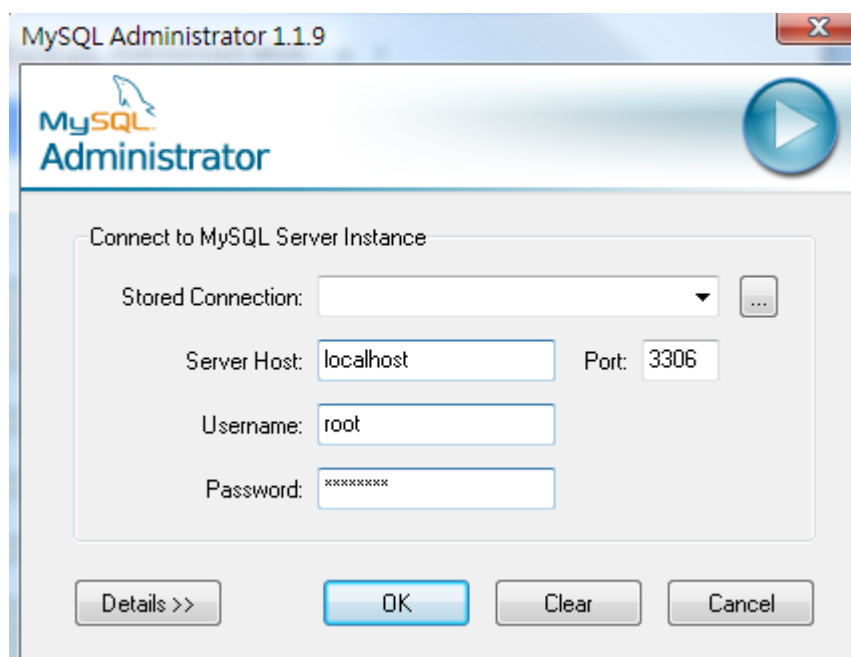
6. 完成後按 Install 。



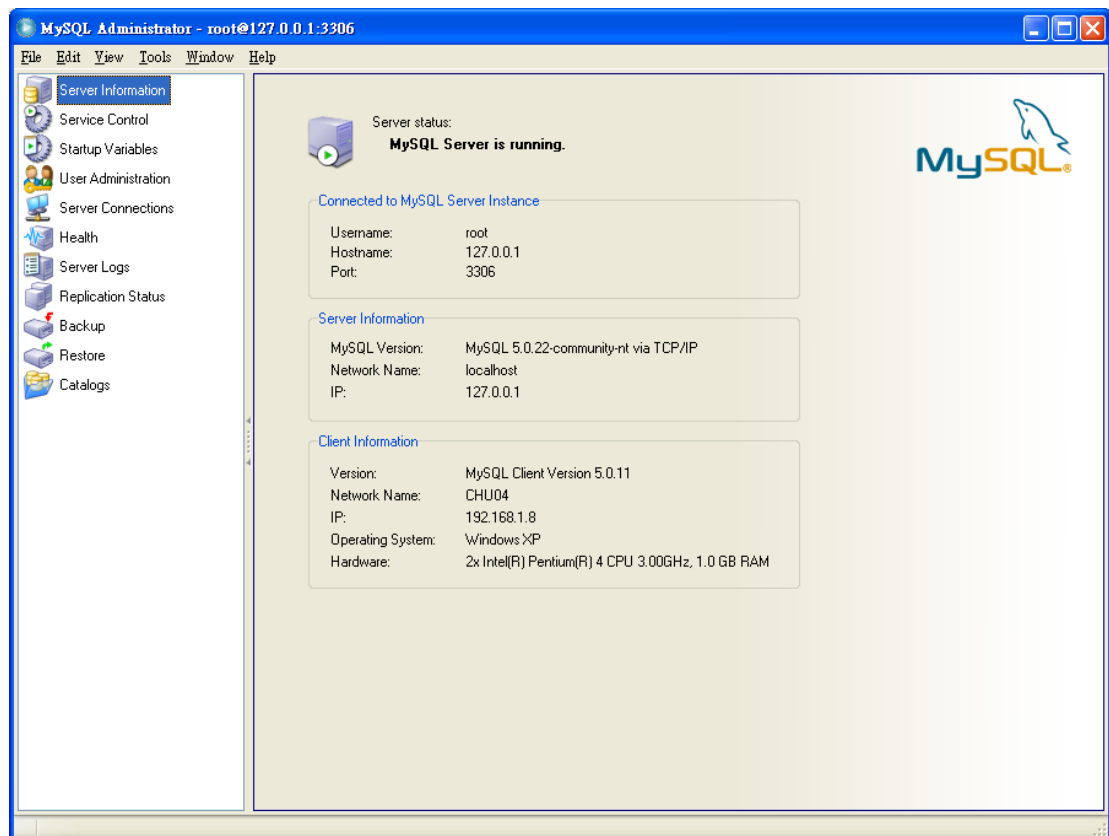
7. MySQL Administrator 安裝完成。



8. 到程式集 -> MySQL 中打開 MySQL Administrator，輸入之前設定的 root 用戶的密碼：password，並按“OK”按鈕。



9. 登入 MySQL Administrator 並連接至 MySQL 資料庫。



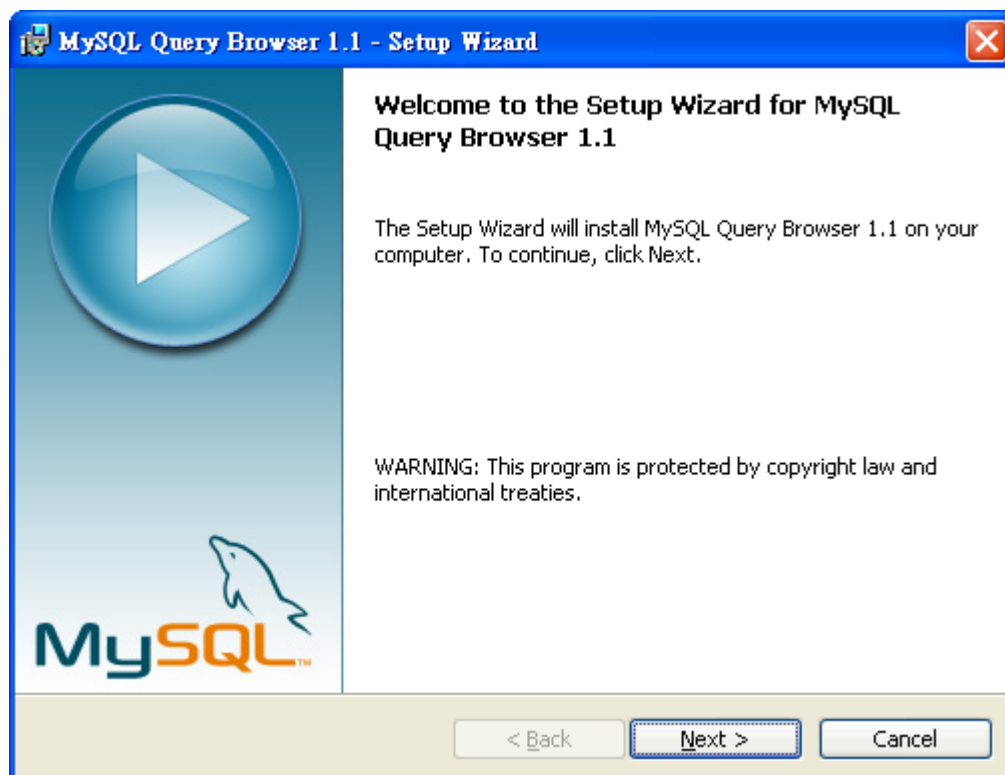
MySQL Query Browser

通過 MySQL Query Browser 我們可以很方便地查詢及修改 MySQL 資料庫。它讓我們可以簡單地對資料表作出查詢，也可以增加、修改或刪除資料表及資料表內的記錄。

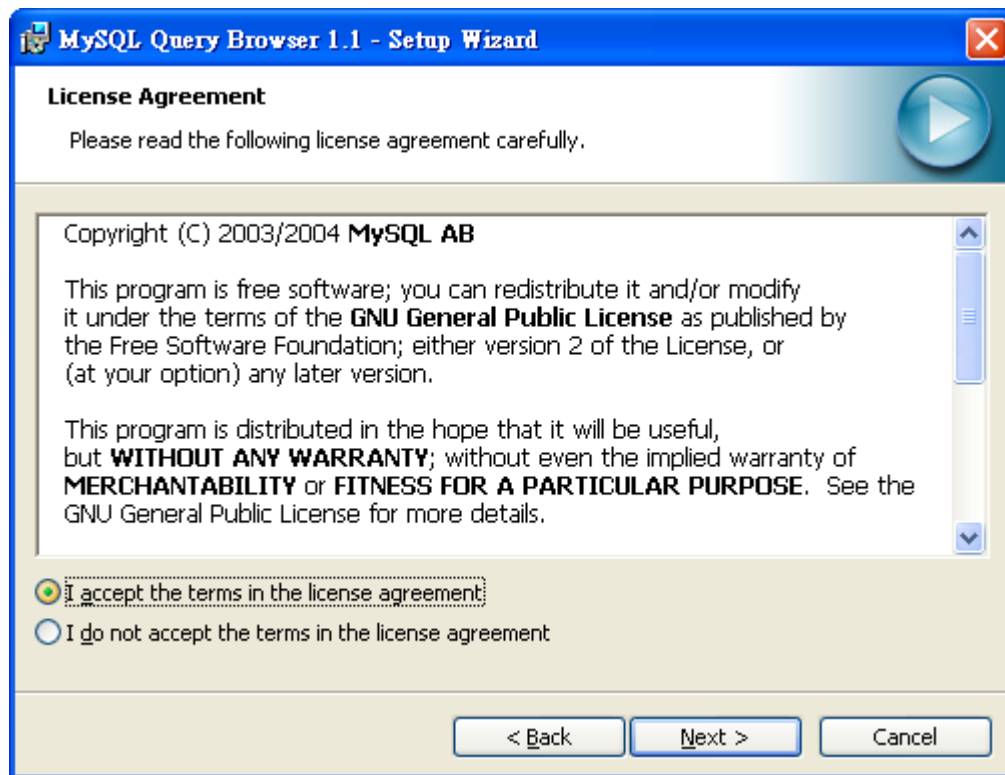
1. 我們可以到以下網址取得 MySQL Query Browser 的 Installer :

<http://dev.mysql.com/get/Downloads/MySQLAdministrationSuite/mysql-query-browser-1.1.20-win.msi/from/http://mysql.cdpa.nsysu.edu.tw/>

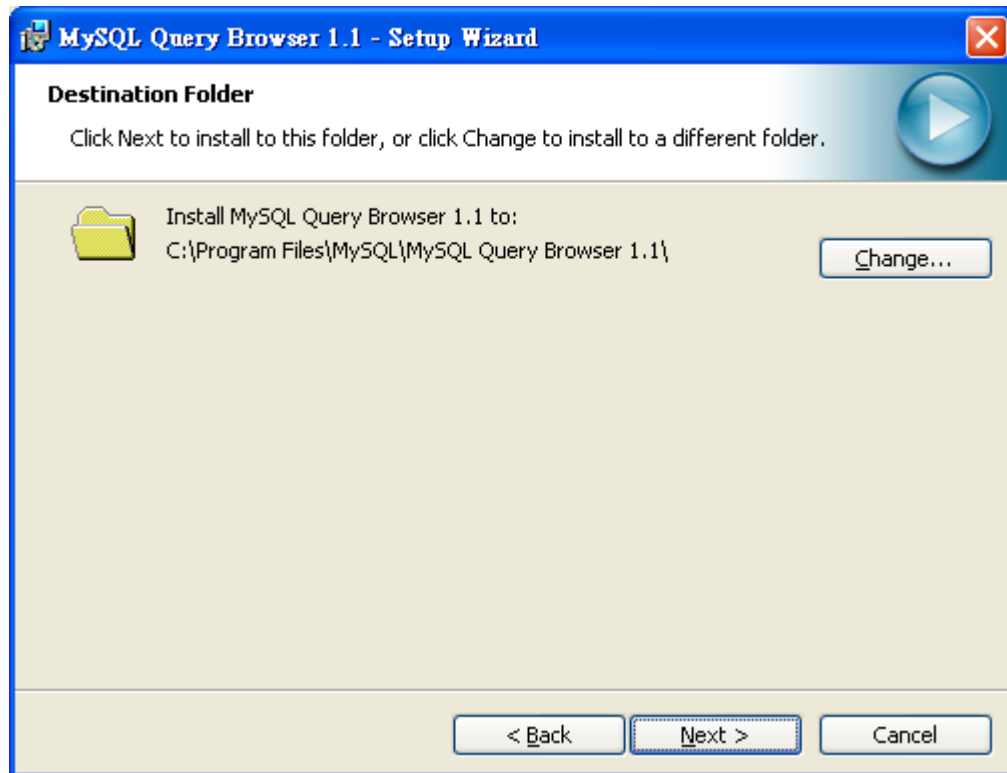
2. 下載完成後，執行 mysql-query-browser-1.1.20-win.msi 檔。



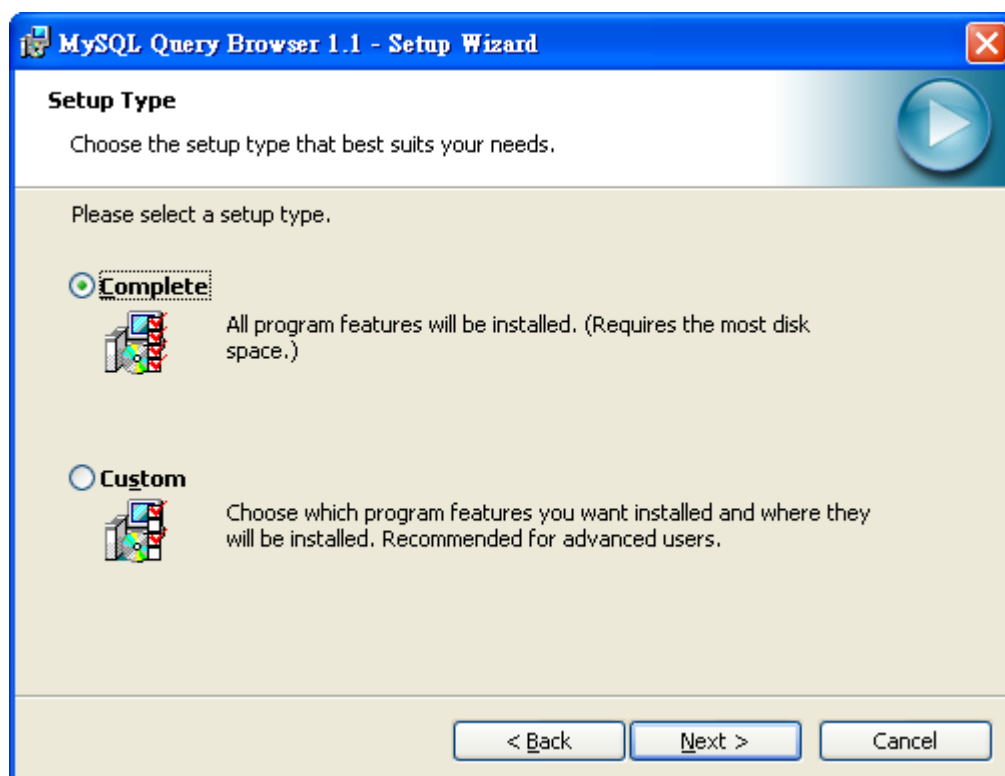
3. 按 Next 後，剔選 I accept the terms in the license agreement 。
然後按 Next 。



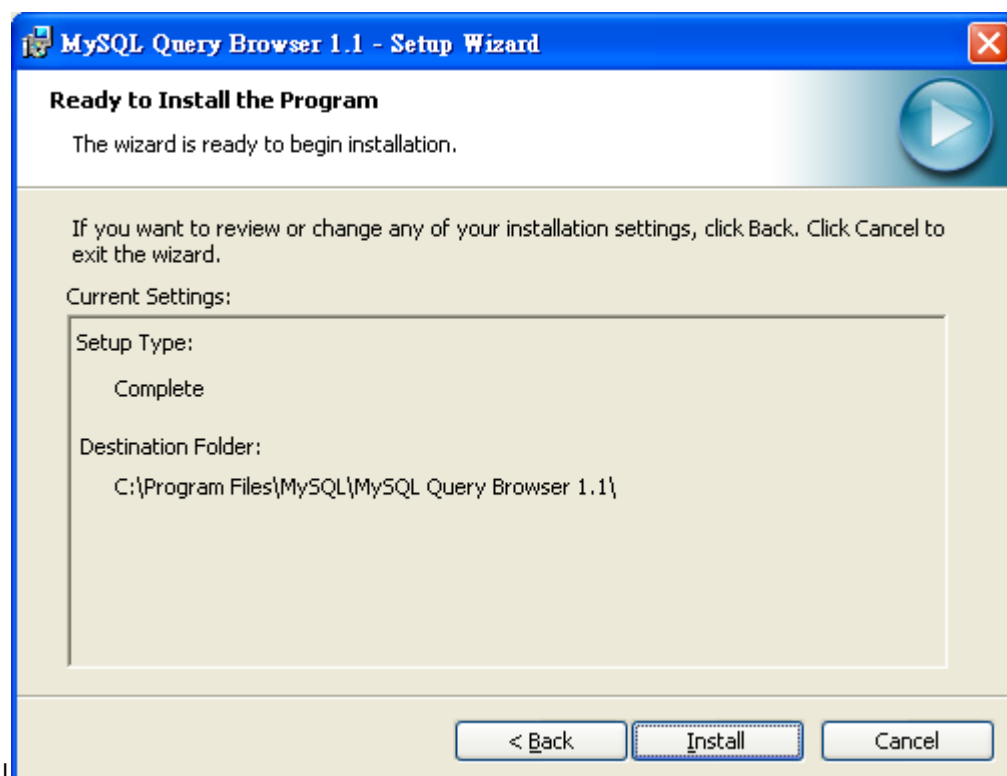
4. 選擇要安裝的位置 : C:\Program Files\MySQL\MySQL Query Browser 1.1\ 。然後按 Next 。



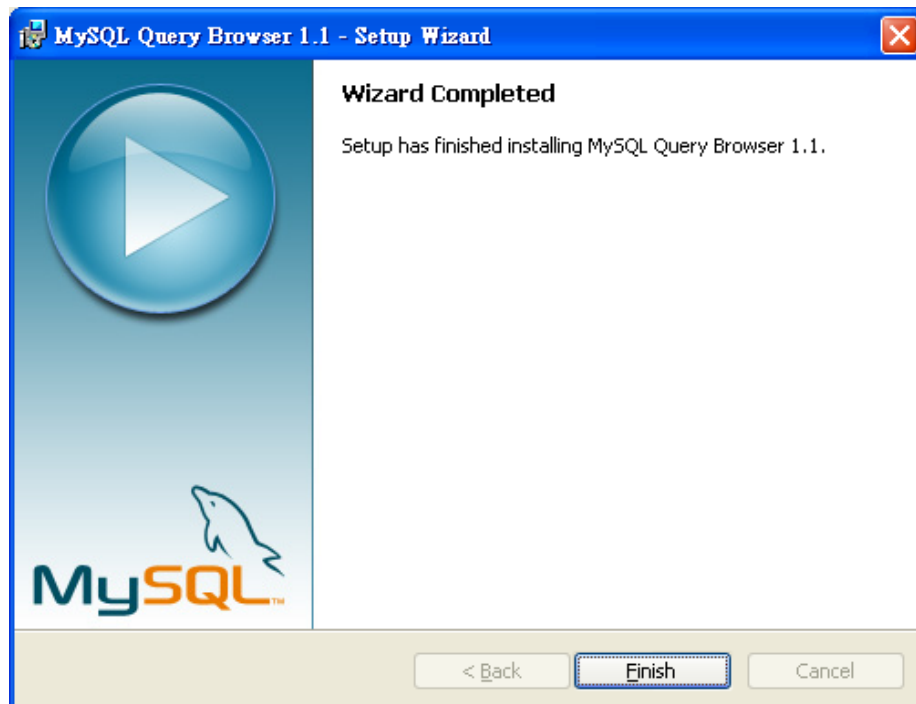
5. 選擇 Complete 並按 Next 。



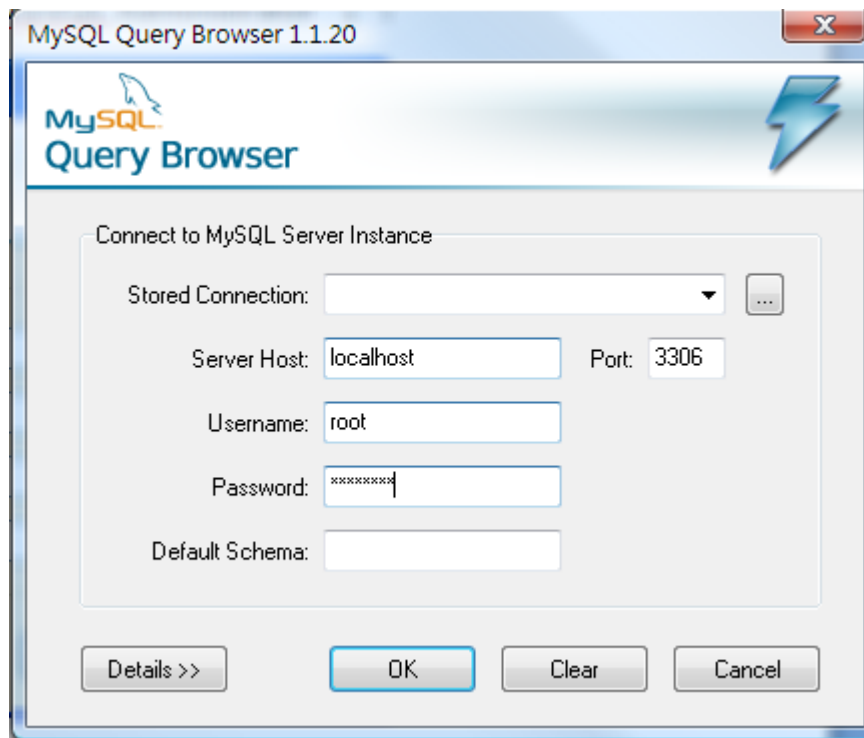
6. 確定資料後，按 Install 以進行安裝。



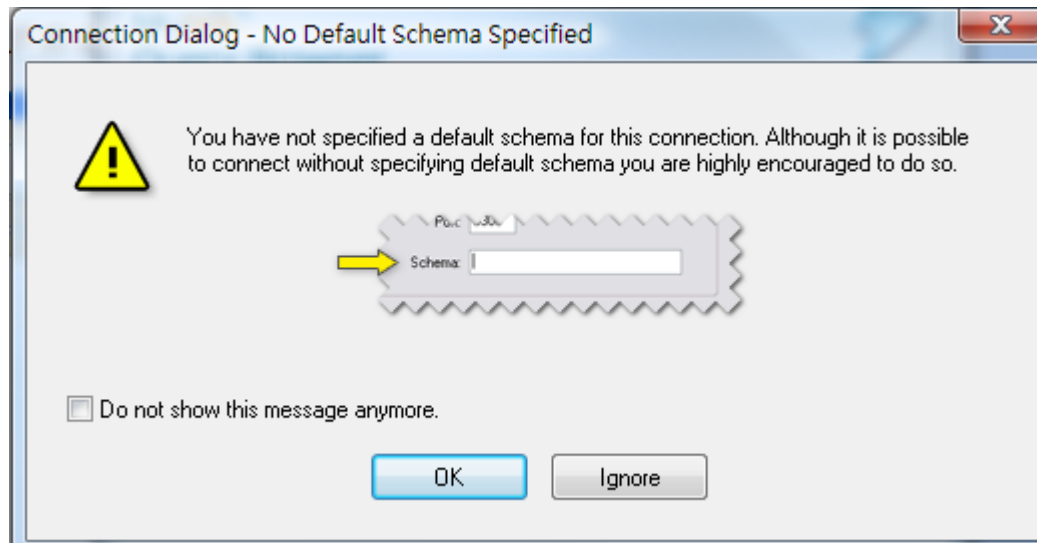
7. 安裝完成後按 Finish 。



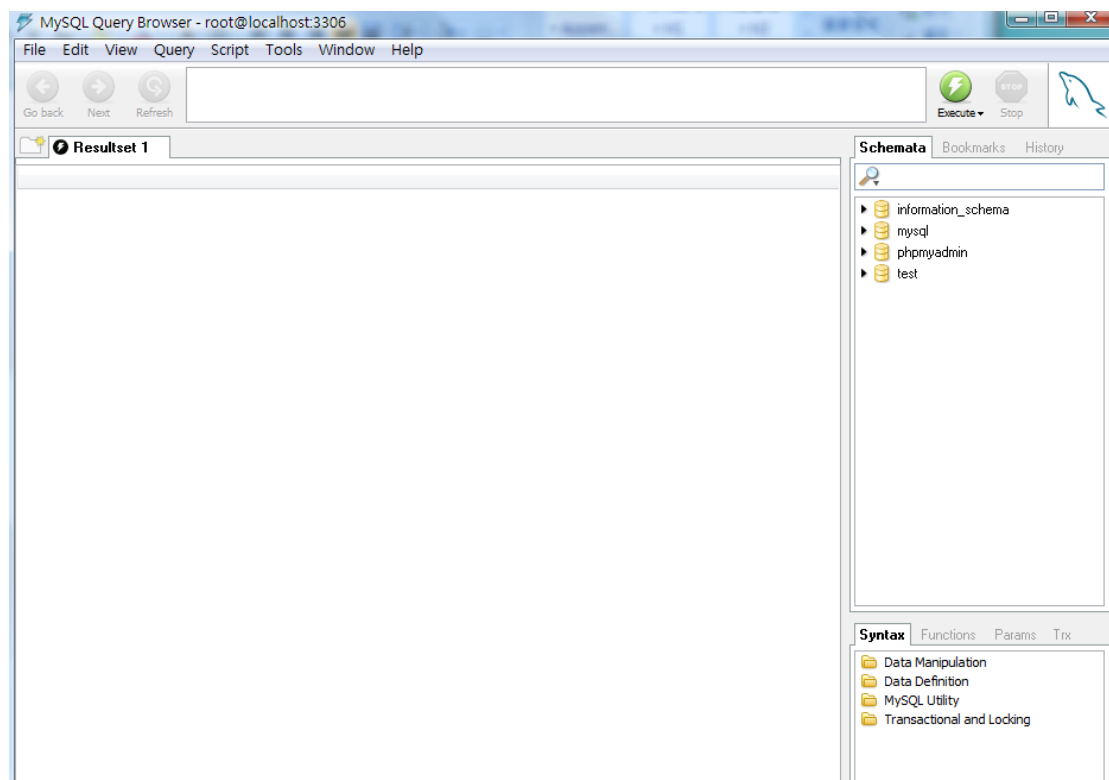
8. 到程式集 -> MySQL 中打開 MySQL Query Browser，輸入之前設定的 root 用戶的密碼：password，並按“OK”按鈕。



9. 按 “Ignore” 按鈕。

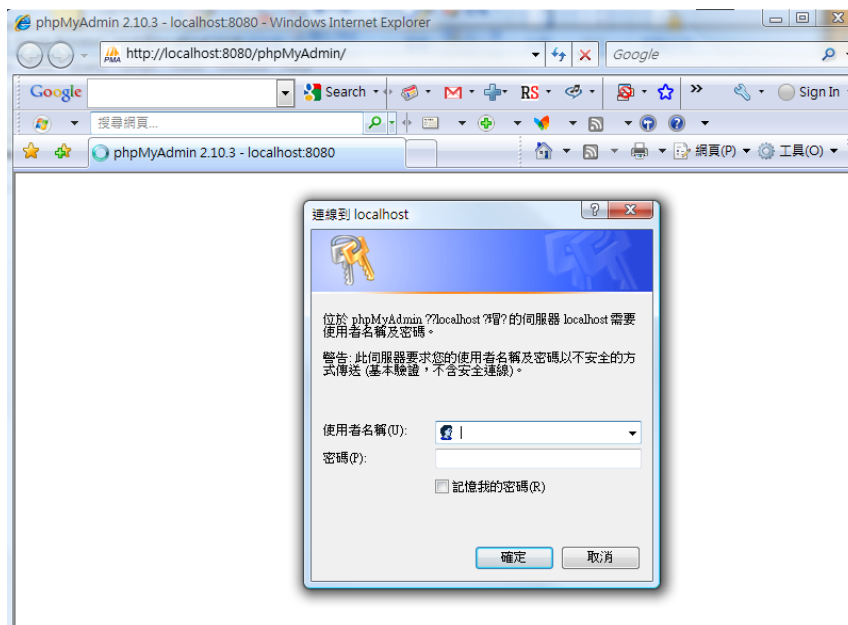


10. 登入 MySQL Query Browser 並連接至 MySQL 資料庫。

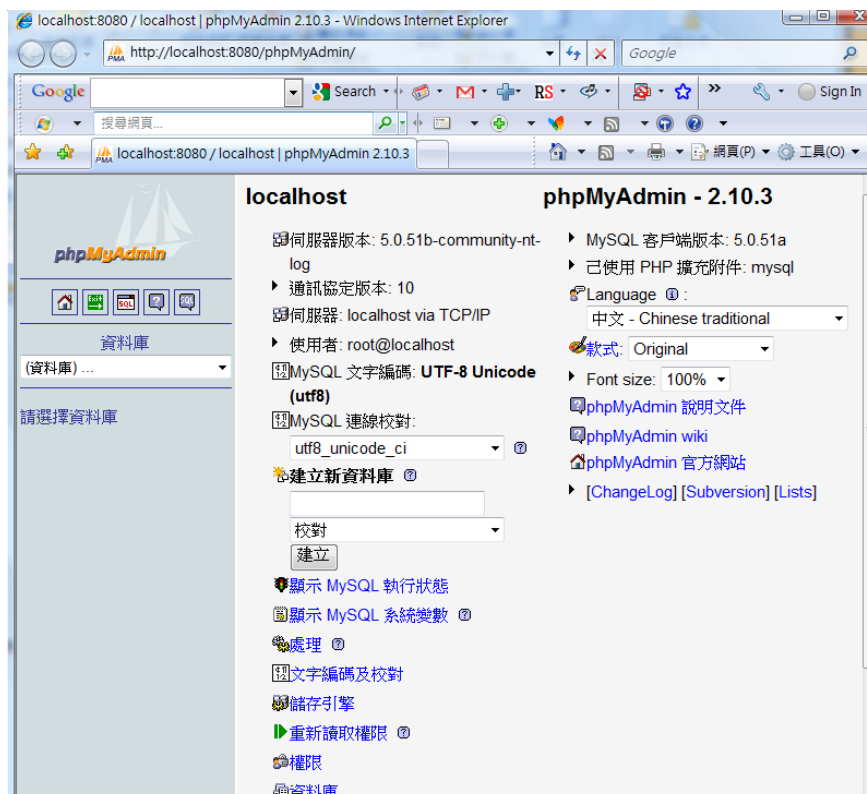


PHPMYAdmin

1. 瀏覽 <http://localhost:8080/phpMyAdmin/>, login: root, password: password



2. PHPMyAdmin Site



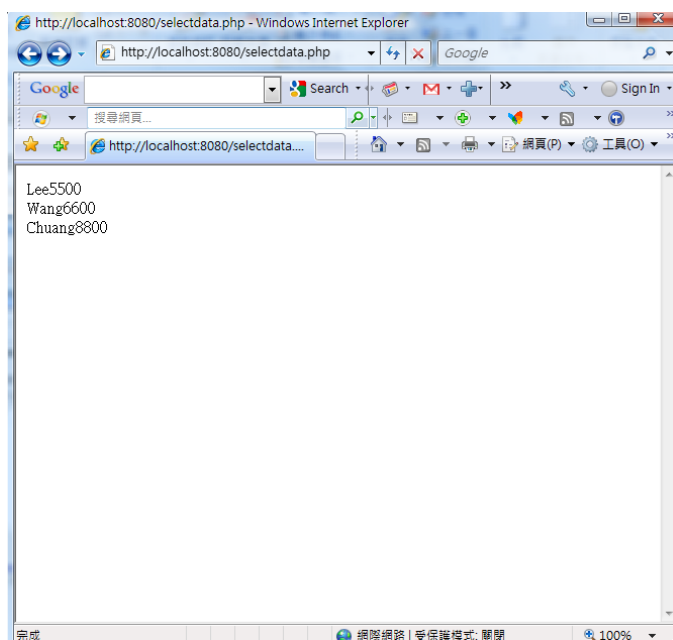
3.5 練習

使用 **PHP** 從數據庫取得數據

1. 把以下程式儲存成 `selectdata.php` 在 `C:\AppServ\www` 文件夾中。

```
<?php
$link = mysql_connect("localhost", "root", "password");
mysql_select_db("test");
$sql = "SELECT realname, salary FROM employee";
$result = mysql_query($sql);
$number_of_rows = mysql_num_rows($result);
for ( $i=0; $i<$number_of_rows; $i++ ) {
    $data = mysql_fetch_row($result);
    echo $data[0].$data[1]."<br>";
}
mysql_close();
?>
```

2. 打開瀏覽器並瀏覽至 `http://localhost:8080/ selectdata.php`

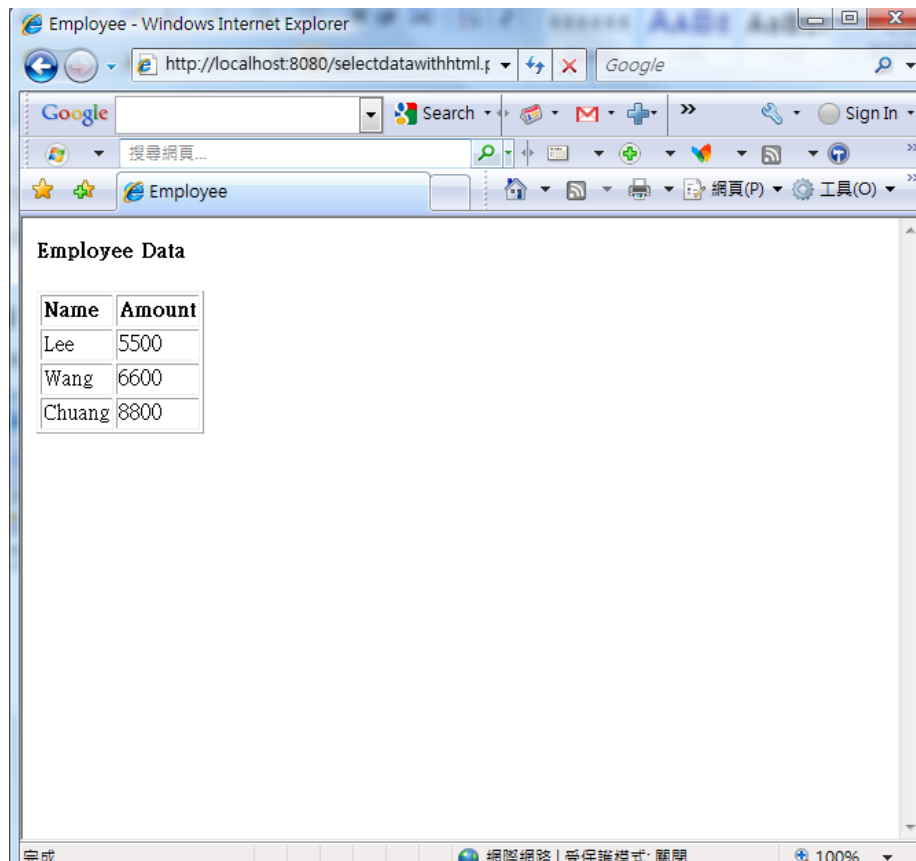


3. 把以下程式儲存成 `selectdataWithHTML.php` 在 `C:\AppServ\www` 文件夾中。

```
<html>
<head></head>
<title>Employee</title>
<body>
<?php
$link = mysql_connect("localhost", "root", "password");
mysql_select_db("test", $link);
$sql = "SELECT realname, salary FROM employee";
$result = mysql_query($sql);
$number_of_rows = mysql_num_rows($result);
?>
<b>Employee Data</b>
<br>
<br>
<table border=1>
<tr>
<td><b>Name</b></td>
<td><b>Amount</b></td>
</tr>
<?php
    for ( $i=0; $i<$number_of_rows; $i++ ) {
        $data = mysql_fetch_row($result);
    ?>
    <tr>
    <td><?=$data[0]?></td>
    <td><?=$data[1]?></td>
    </tr>
    <?
        }
    mysql_close($link);
    ?>
</table>
</body>
</html>
```

4. 打開瀏覽器並瀏覽至

<http://localhost:8080/selectdataWithHTML.php>



3.6 資源

1. 打開瀏覽器並瀏覽至

<http://www1.fevaworks.com/course/cefwebeng/>

Username: cefwebeng

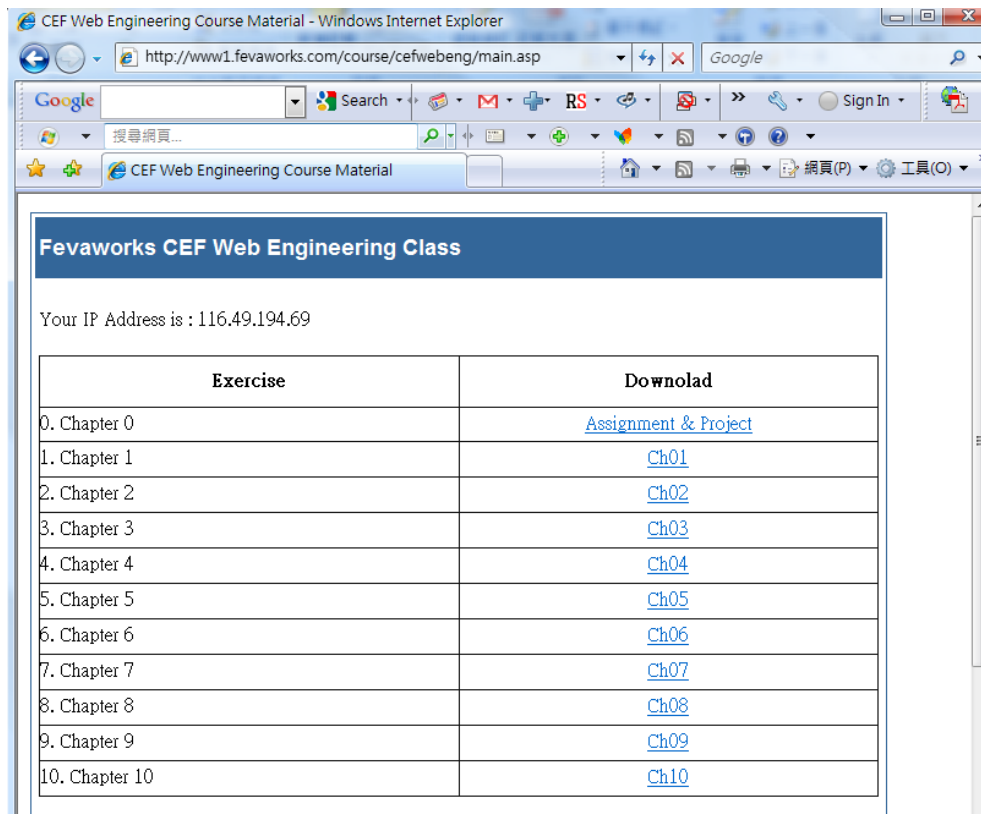
Password: joelam

The screenshot shows a Windows Internet Explorer browser window displaying the login page for the CEF Web Engineering Course Material. The browser's address bar shows the URL <http://www1.fevaworks.com/course/cefwebeng/>. The page features the FEVAWORKS logo and the title "CEF Web Engineering Course Material". Below the title is a login form with the following fields and buttons:

Login	
Username:	<input type="text"/>
Password:	<input type="password"/>
<input type="button" value="Login Now"/> <input type="button" value="Reset"/>	

Below the login form, a message states: "Please call 3106 8211 if you don't know the username & password". At the bottom of the page, there is a banner for THOMSON PROMETRIC, with the text "now at Fevaworks".

2. 下載 Chapter 3 “Ch03”



CEF Web Engineering Course Material - Windows Internet Explorer

http://www1.fevaworks.com/course/cefwebeng/main.asp

Google

Google

Search

Sign In

CEF Web Engineering Course Material

Fevaworks CEF Web Engineering Class

Your IP Address is : 116.49.194.69

Exercise	Downolad
0. Chapter 0	Assignment & Project
1. Chapter 1	Ch01
2. Chapter 2	Ch02
3. Chapter 3	Ch03
4. Chapter 4	Ch04
5. Chapter 5	Ch05
6. Chapter 6	Ch06
7. Chapter 7	Ch07
8. Chapter 8	Ch08
9. Chapter 9	Ch09
10. Chapter 10	Ch10

3. 解壓縮并且打開文件夾

