

MySQL

關聯式資料庫設計與實作

授課講師

蔡宜璋

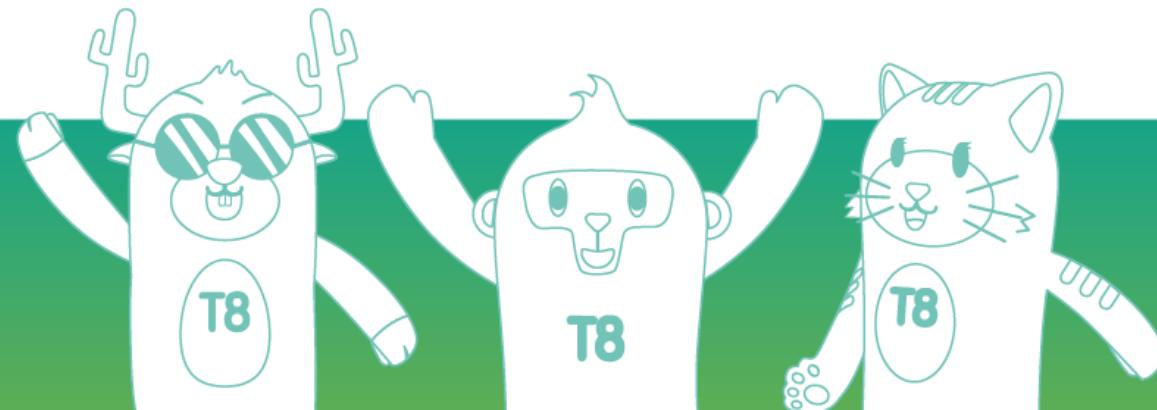
教材編寫

蔡宜璋

緯
育 *TibaMe*

即學 · 即戰 · 即就業

<https://www.tibame.com/>



課程大綱

- ◆ Module 1. 資料庫
- ◆ Module 2. 關聯式資料庫簡介
- ◆ Module 3. MySQL簡介與安裝
- ◆ Module 4 ~ 5. 資料庫管理
- ◆ Module 6. 資料表管理基礎
- ◆ Module 7 ~ 8. 欄位的資料型態
- ◆ Module 9. 資料表處理
- ◆ Module 10 ~ 15. 資料查詢
- ◆ Module 16 ~ 20. 資料連結
- ◆ Module 21. 新增紀錄
- ◆ Module 22. 修改及刪除紀錄
- ◆ Module 23 ~ 24. 限制條件
- ◆ Module 25 ~ 27. 交易處理
- ◆ Module 28 ~ 29. 檢視表
- ◆ Module 30. 索引

授課講師介紹



蔡宜璋

專長

機器學習、自然語言、網路資料擷取、行動應用服務、AIoT 物聯網

簡歷

15年 程式開發經驗

10年 講師教學經驗

8年 專案管理經驗

3年 新創參與經驗

老師的話

網路應用服務，都要使用資料庫，學會操作資料庫，開啟數據之門！

聯絡方式

yc0815@gmail.com

學習本課程須知

先備知識

具備操作個人電腦文書編輯能力
具備基礎上網能力

學習目標

- A 資料庫架設管理與規劃
- B 資料定義與查詢(DDL、DQL)
- C 進階資料查詢(DQL)
- D 資料操作(DML)
- E 資料控制與進階定義(DCL、DDL)

學習方式

課程講解
程式範例

須完成哪些作業或考試

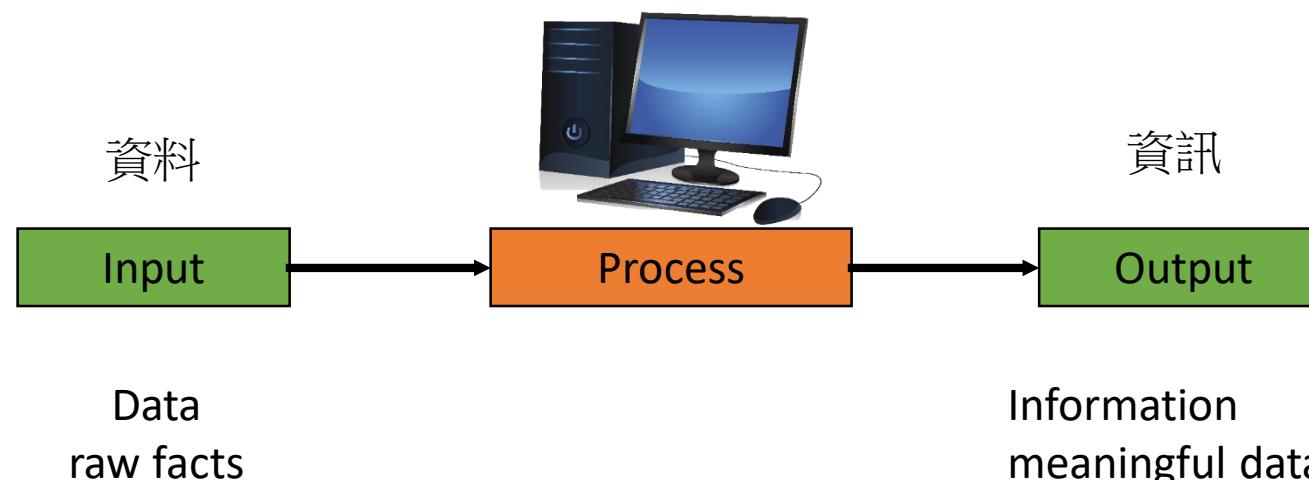
作業練習
筆試

Module 1. 資料庫

- 1-1: 資料處理與資料庫
- 1-2: 關聯式資料庫設計
- 1-3: E-R Model 實體關係圖

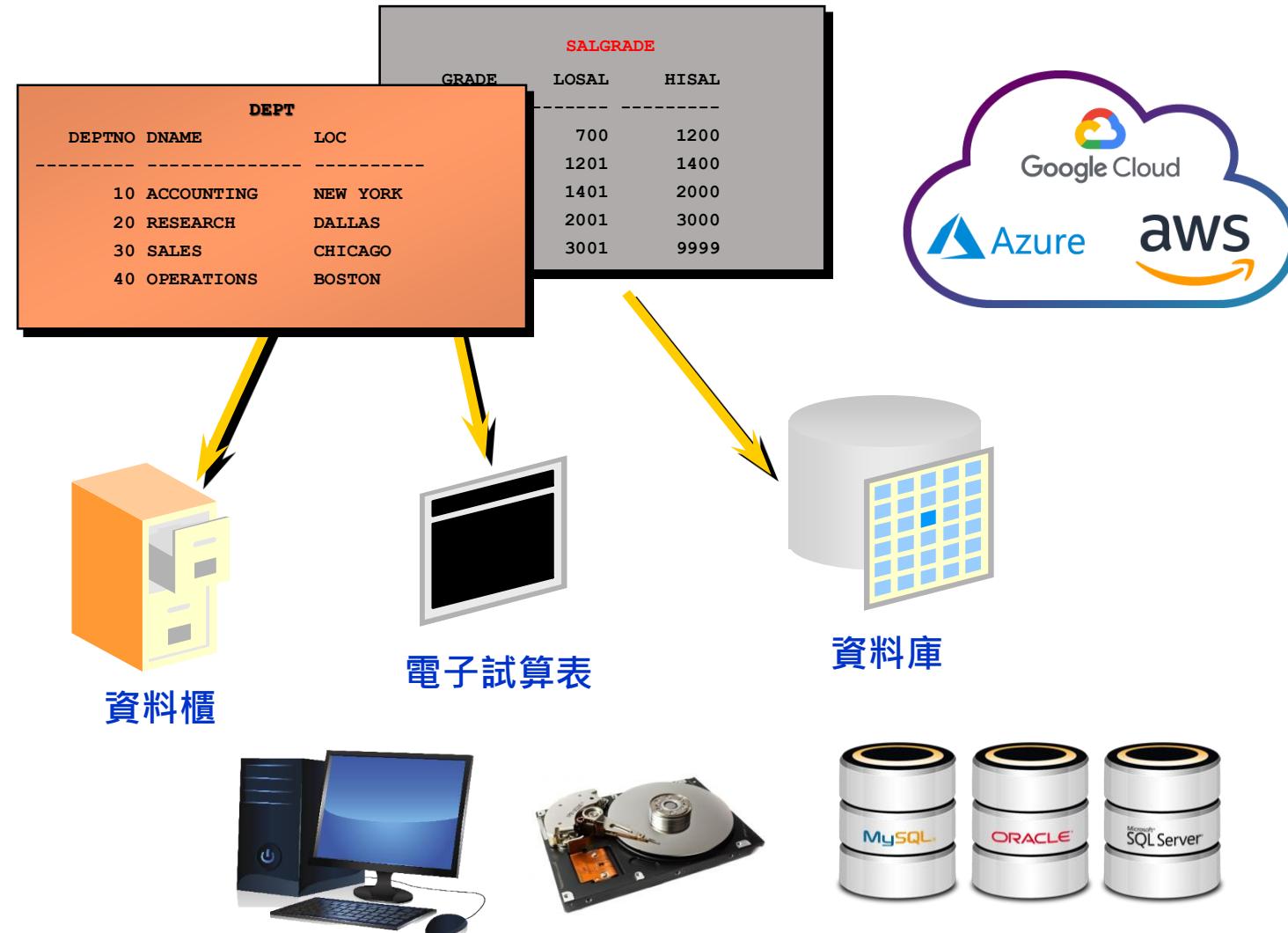
電腦系統(Computer System)

- ▶ 電腦(Computer)
 - 一個電子設備用來處理資料
- ▶ 電子資料處理(Electronic Data Processing)
 - 硬體: physical parts of the computer
 - 軟體: instructions to the computer
 - 資料: raw facts the computer can manipulate
 - 人 : also known as users

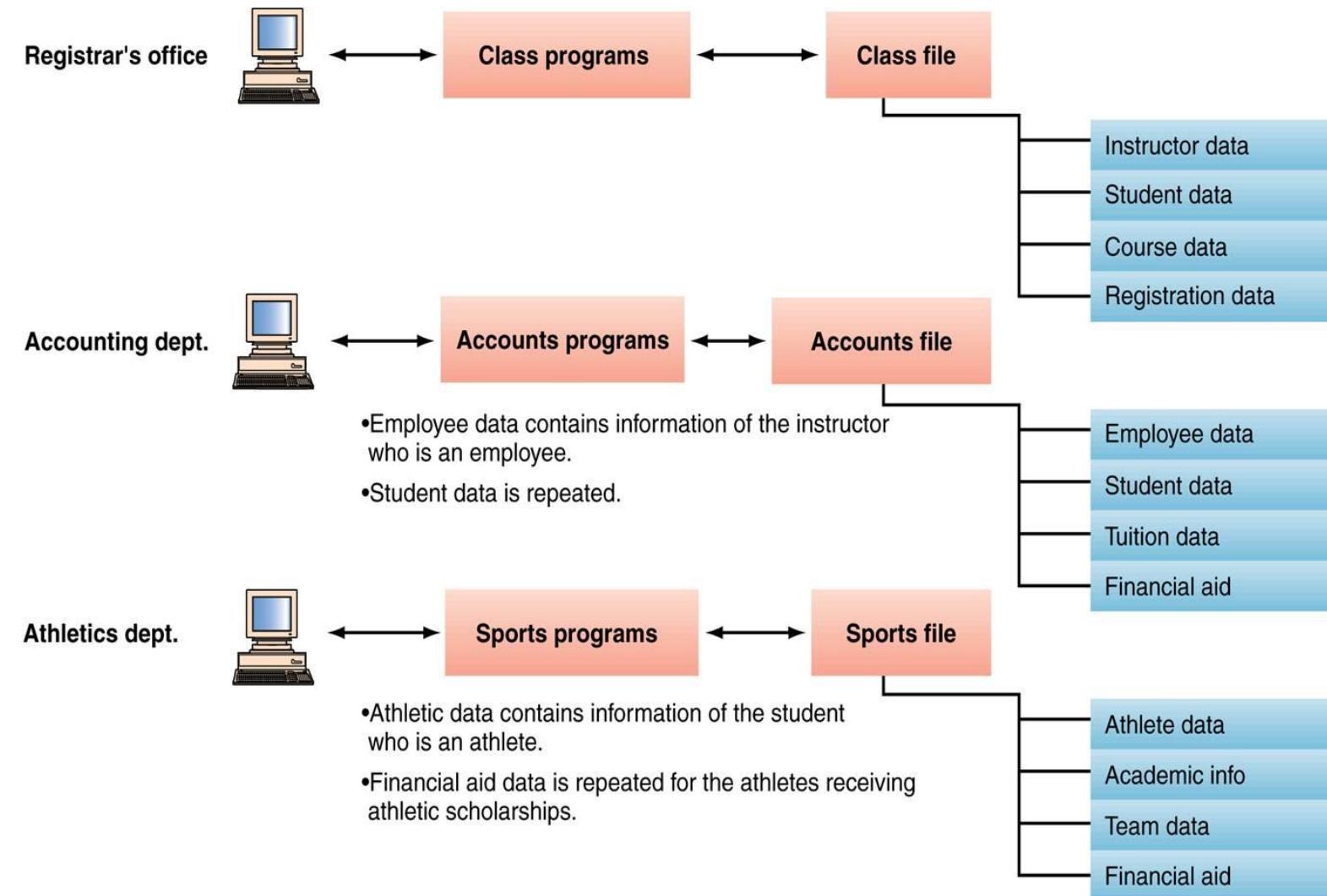


資料處理(Data Processing)

資訊處理方式的演進



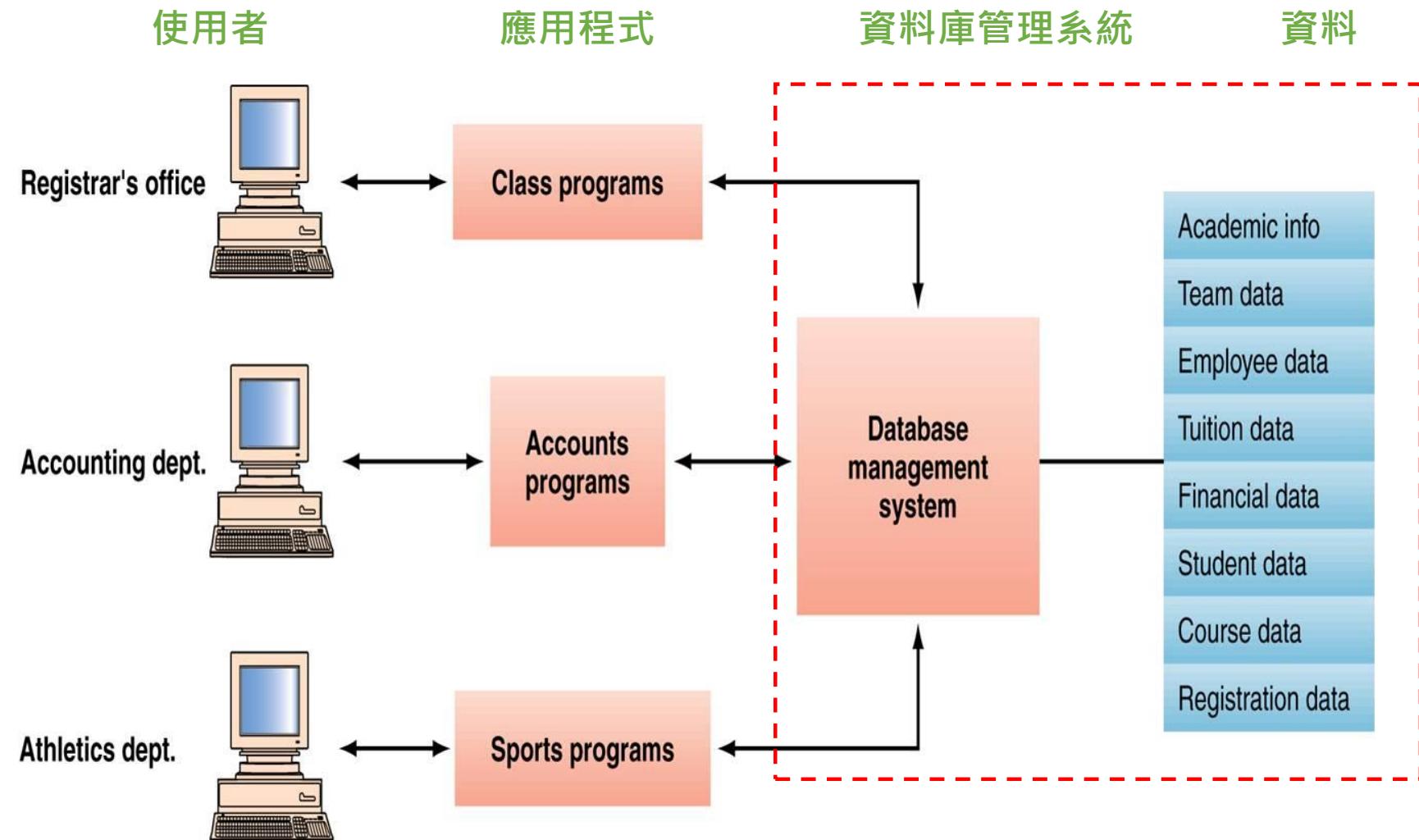
檔案式資訊系統的潛在問題



檔案資訊系統的潛在問題

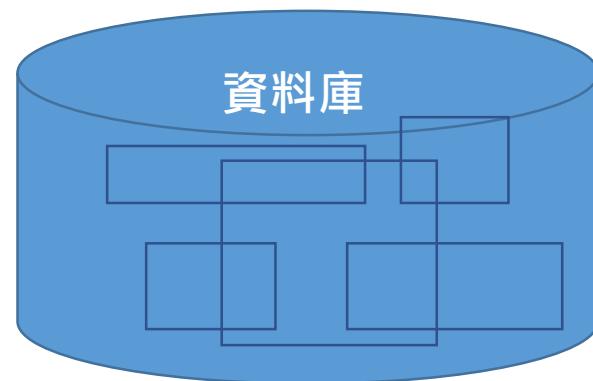
- 資料重複 (Data Redundancy)
- 資料不一致 (Data Inconsistency)
- 資料存取不易 (Difficulty in Accessing Data)
- 資料隔離 (Data Isolation)
- 資料完整性問題 (Integrity Problems)
- 資料更新一致性問題 (Atomicity Problem)
- 多人同時存取的異常 (Concurrent Access anomalies)
- 安全性問題 (Security Problems)

使用資料庫管理系統的資訊系統



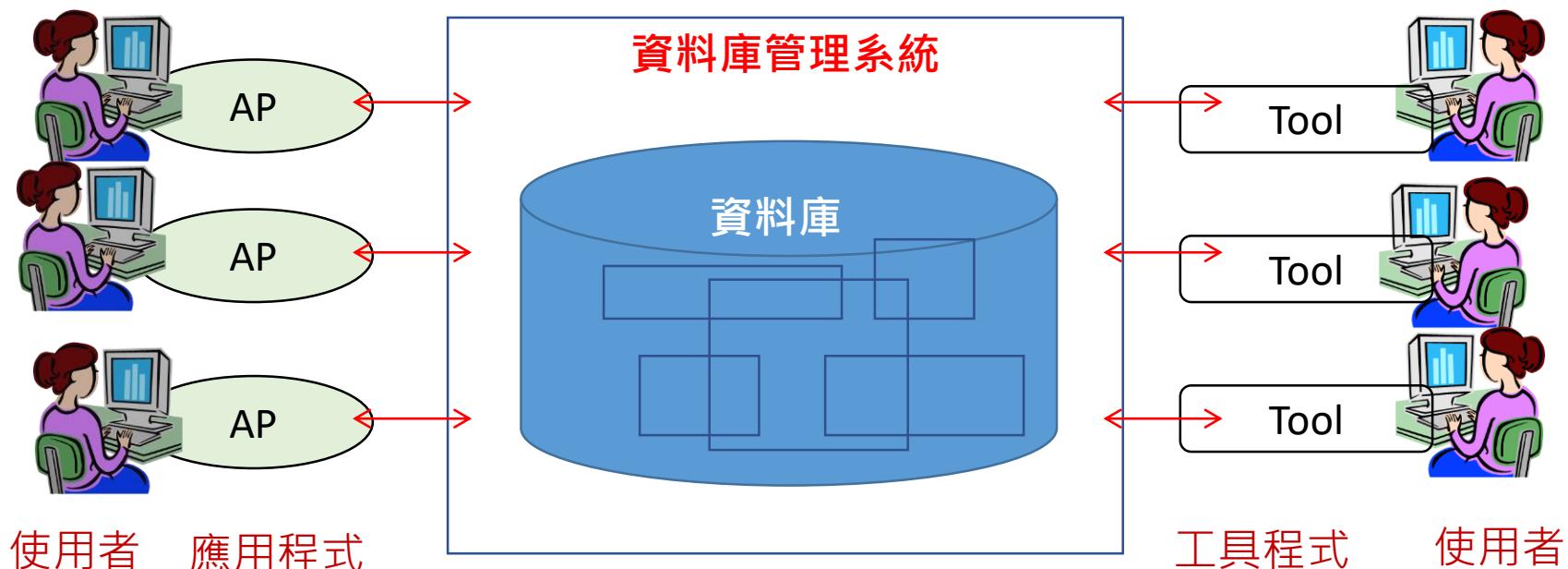
資料庫

- 資料庫
 - Database, DB
 - 相關資料的集合(Collection of interrelated data)
 - 資訊架構(Information architecture)
 - Bit >> Byte >> Field >> Record >> File >> Database



資料庫管理系統

- DataBase Management System, DBMS
 - 一種用來建立與管理資料庫的系統軟體
 - 提供系統化的方式讓使用者及其他程式來存取/管理資料
 - 介於資料庫與用戶或應用程式之間，確保資料一致性與管理方便性。
 - 例如：MySQL, Oracle, MS SQL Server, DB2, ...



資料庫管理系統

- 資料庫管理系統的功能
 - 管理資料 (Management of Data - MOD)
 - 定義資料儲存的結構
 - 提供資料維護的機制
 - 確保資料的安全(系統當機,未經授權的存取,誤用,...)
 - 提供多人使用下的同時存取控制機制

資料庫管理系統的使用者

- 應用程式設計師(Application programmer)
 - 透過DML與資料庫互動(做資料維護)
- 經驗豐富的使用者(Sophisticated users)
 - 使用SQL來完成工作
- 專業使用者(Specialized users)
 - -撰寫不同於傳統資料處理方式的特殊資料庫應用程式
- 一般使用者(Naive users)
 - 使用他人寫好的應用程式與資料庫互動
- 資料庫管理者(Database administrator,DBA)
 - 管理資料庫系統的所有活動

資料庫管理者(Database administrator,DBA)

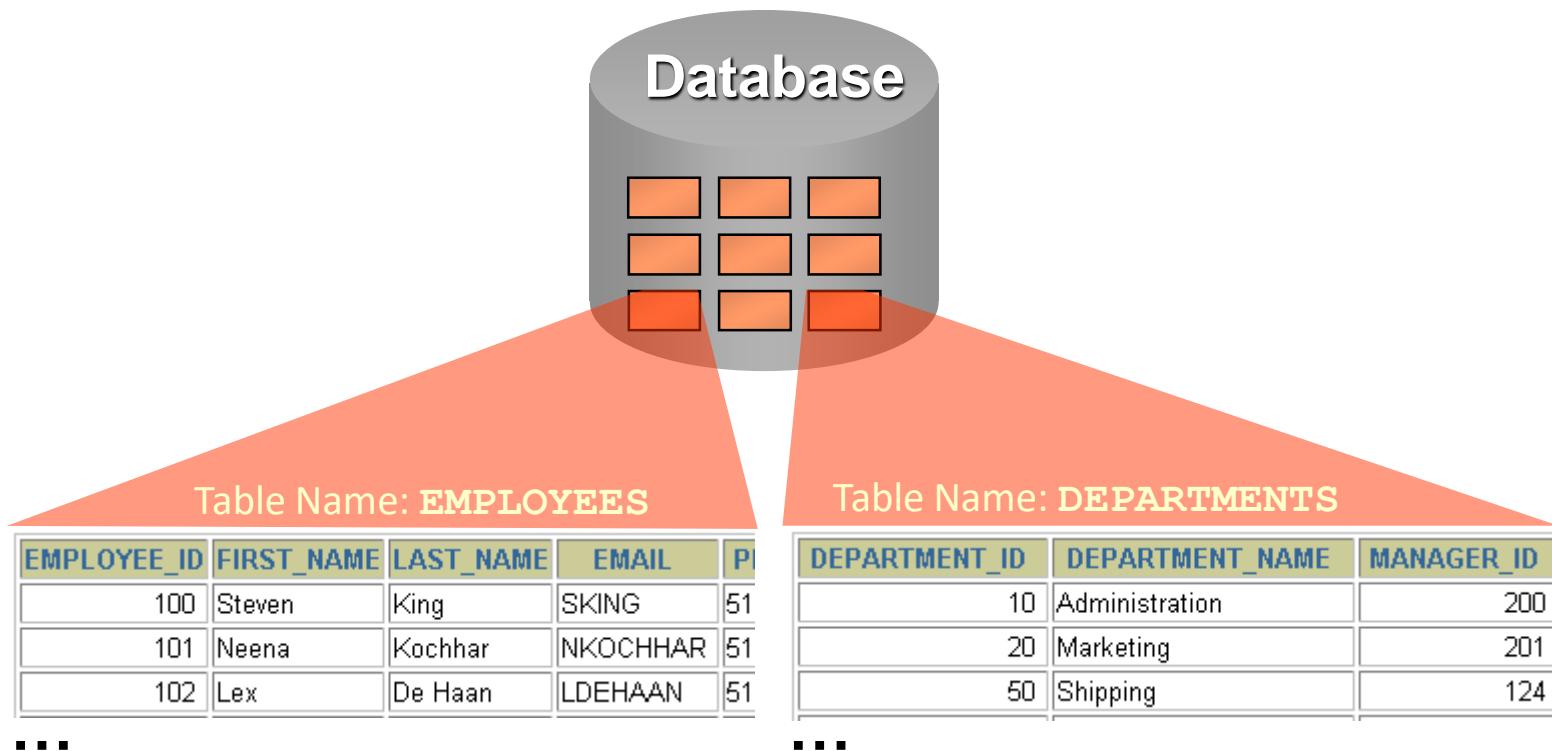
- ▶ 需很瞭解企業的資訊資源和需求(有豐富經驗)
- ▶ 主要職責包括：
 - 定義資料架構(**Schema**)
 - 資料儲存的架構及存取方式的定義
 - 資料架構和實際組成的修訂
 - 使用者授權
 - 設定整合性資料限制條件
 - 與使用者溝通介面
 - 資料庫系統效能的監督與因應需求的改變

關聯式資料庫管理系統(RDBMS)

- 根據1970年Dr. E.F. Codd 提出的**關聯式資料模型**所發展出來的**資料庫管理系統**
- 關聯式資料模型主要組成：
 - 物件或關聯的集合(Collection of objects or relations)
 - 關聯運算方法(Set of operators to act on the relations)
 - 資料整合條件(Data integrity for accuracy and consistency)
- 關聯式資料庫管理系統
 - 資料庫物件(Database Objects)
 - 資料運算(Data Operations)
 - 資料庫限制條件(Database Constraints)

關聯式資料庫

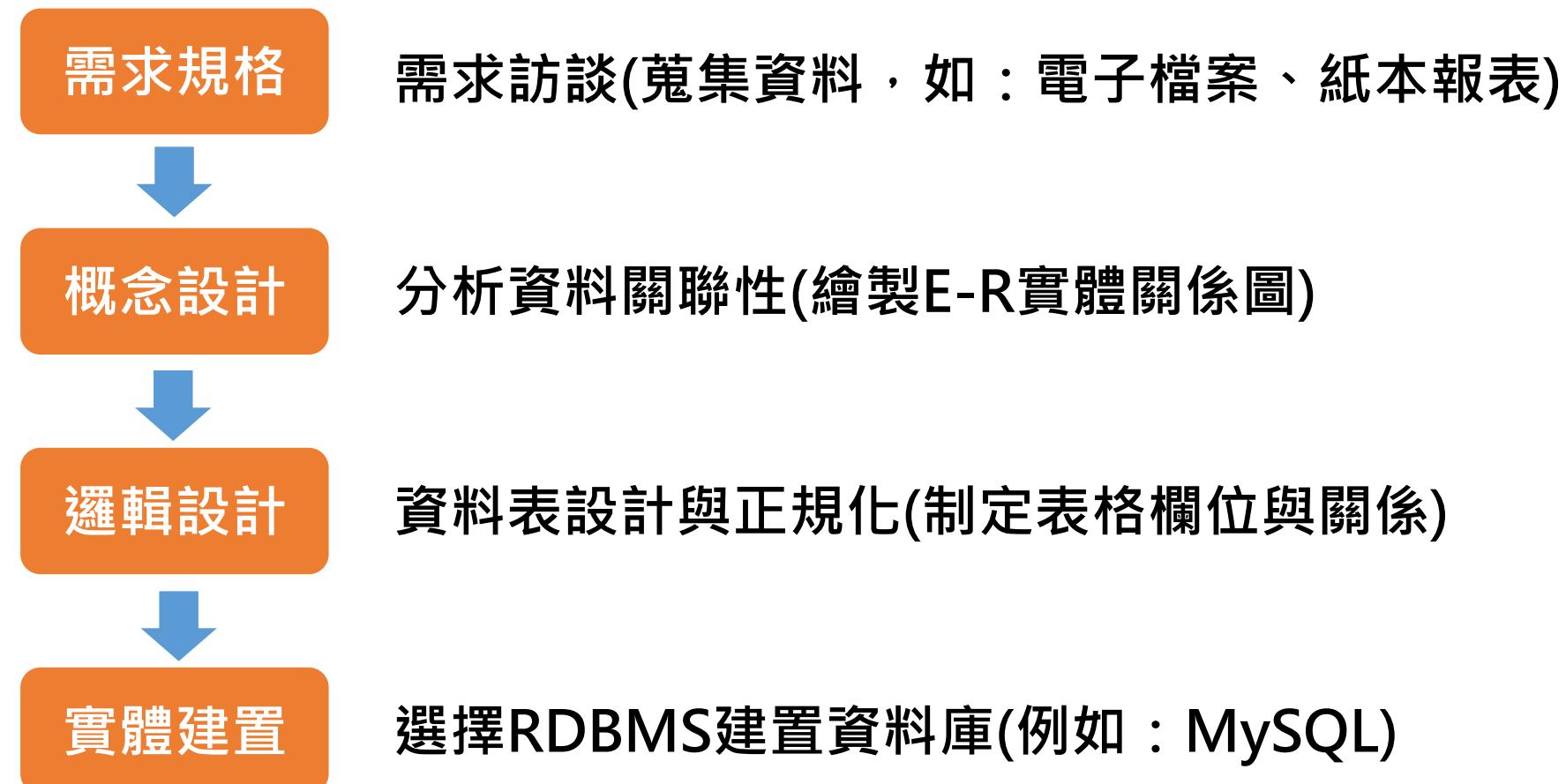
- 主要的資料庫物件：資料表(table)



關聯式資料庫管理系統的功能

- 儲存使用者的資料
- 提供安全與穩定的存取系統
- 提供多人線上同時存取
- 提供權限來做資料庫的管理
- 提供資料備份與回復的功能
- 支援結構化查詢語言 (Structured Query Language, SQL)

關聯式資料庫的規劃與設計

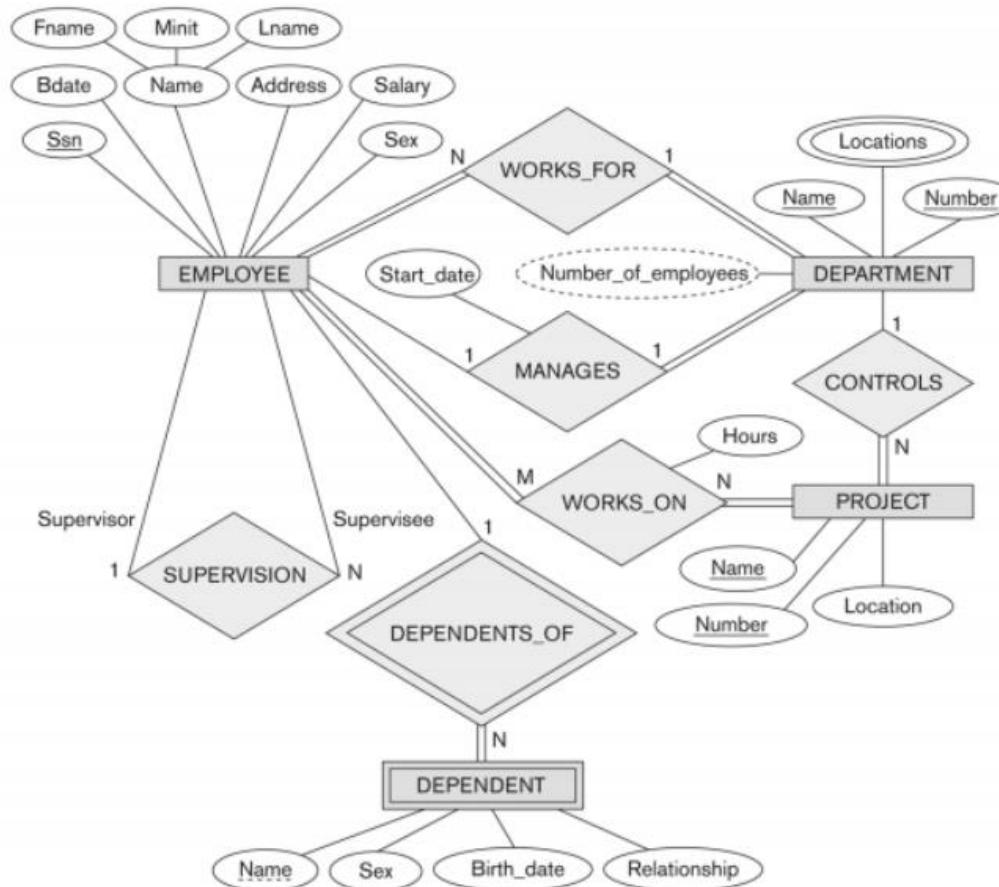
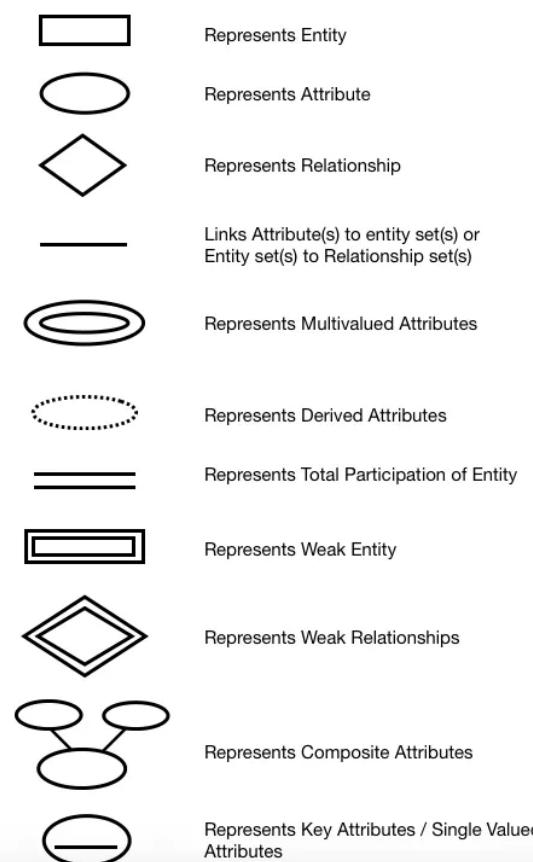


實體關係圖(E-R Model)

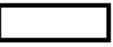
- 實體關係圖是一種圖形化的分析方法
- 透過「標準化圖示」將「需求情境」轉成實體關係圖
- 利於將需求轉為概念設計，分析資料之間的關聯
- 實體關係圖可轉換成為關聯式資料表

E-R 關係圖

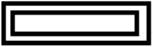
- 透過標準化圖示，將「需求情境」轉成標準的實體關係圖(E-R Model)
- 利於在設計階段，討論與分析資料之間的關聯
- 最後可將實體關係圖轉換成為關聯式資料表



E-R 關係圖示說明

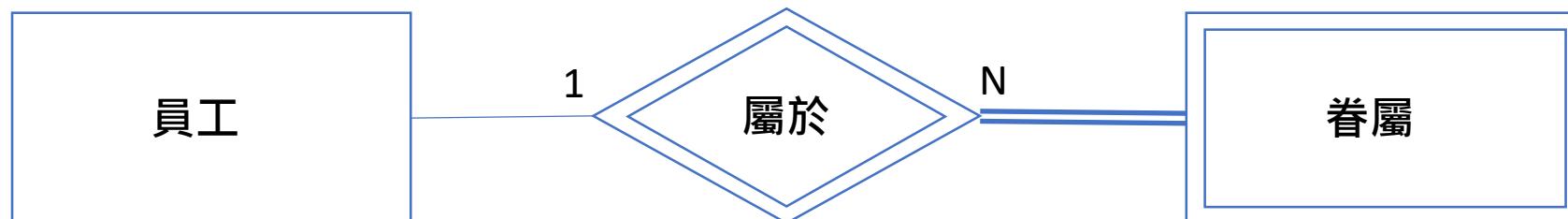
- 實體(Entity) 

表示真實世界裡個體(無形或有形)，如：員工、部門、專案、...。

- 弱實體(Weak Entity) 

- 弱關係 (Weak Relationship) 

表示該實體是依附別的實體而存在，如：眷屬是依附員工個體而存在。



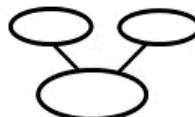
E-R 關係圖示說明

- 屬性(Attribute)



用來描述實體的特性，如：員工的員編、姓名、薪資、地址、...。

- 複合屬性(Composite Attribute)



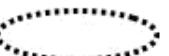
用來描述屬性再細分更小的屬性，如：姓名屬性再區分為姓與名。

- 多值屬性(Multivalued attributes)



該類屬性中，可包含一個以上的值，如：員工實體的電話屬性，可包含一個以上的值。

- 衍生屬性(Derived attributes)



這個屬性的值可以“完全”由其它屬性之值計算出來。可不用存於資料庫。如：年齡可由生日計算。

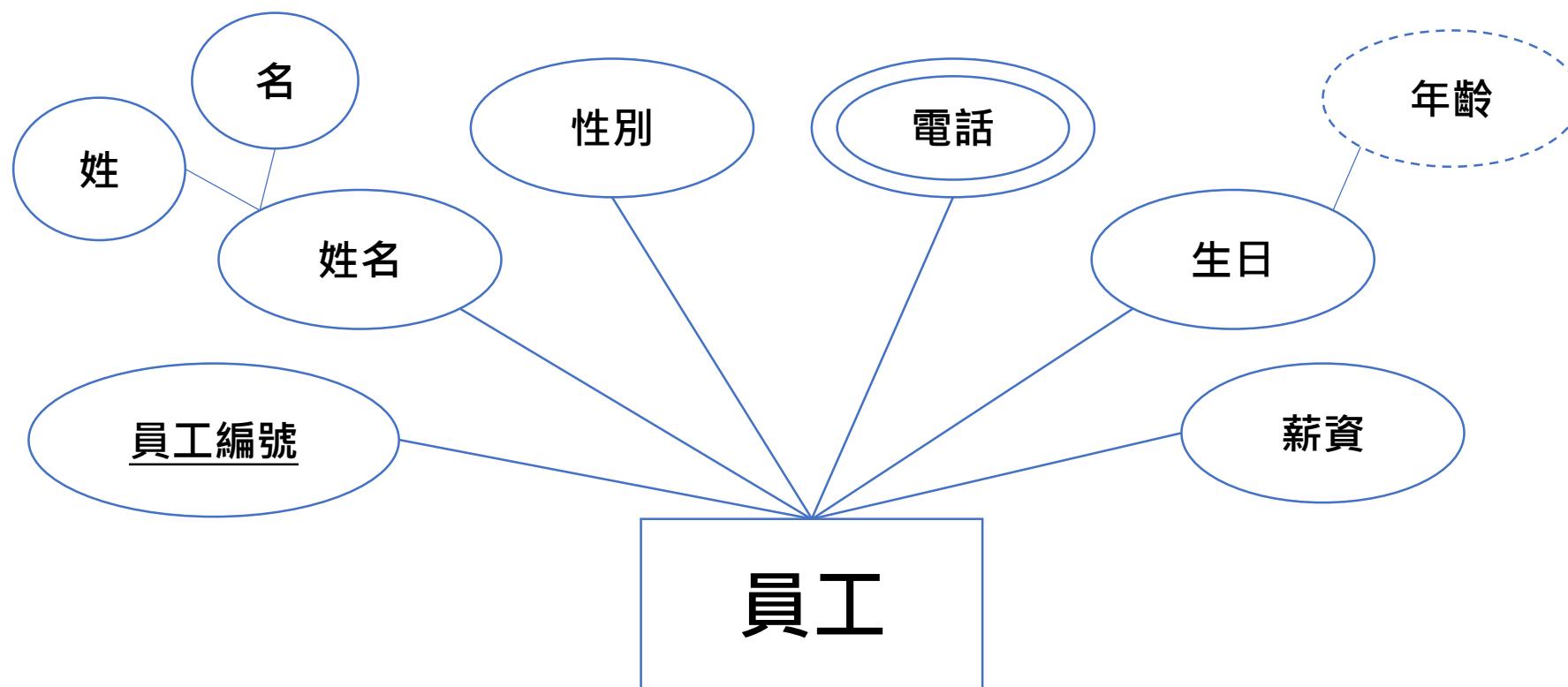
- 鍵值屬性(Key attributes)



用來辨識在實體集合中唯一性的屬性，如：員編是員工實體中唯一性的辨識。

實體的屬性

例如：觀察公司紙本員工資料表，可以聯想出以員工當作實體的相關屬性



實體與實體的關係

- 關係 

不同實體(Entities)之間的關聯(結合、關係集合)

- 關係類型

一對一關係 (1:1)

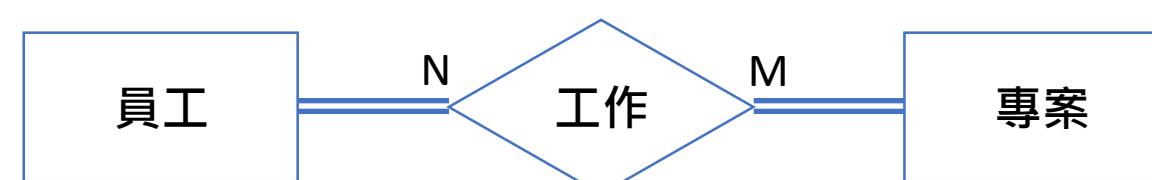
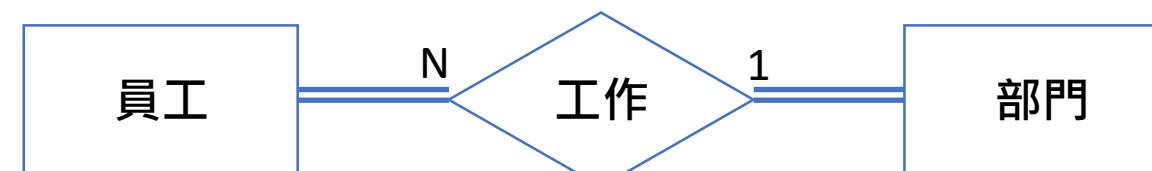
一對多關係 (1:N)

多對多關係 (N:M)

- 參與程度

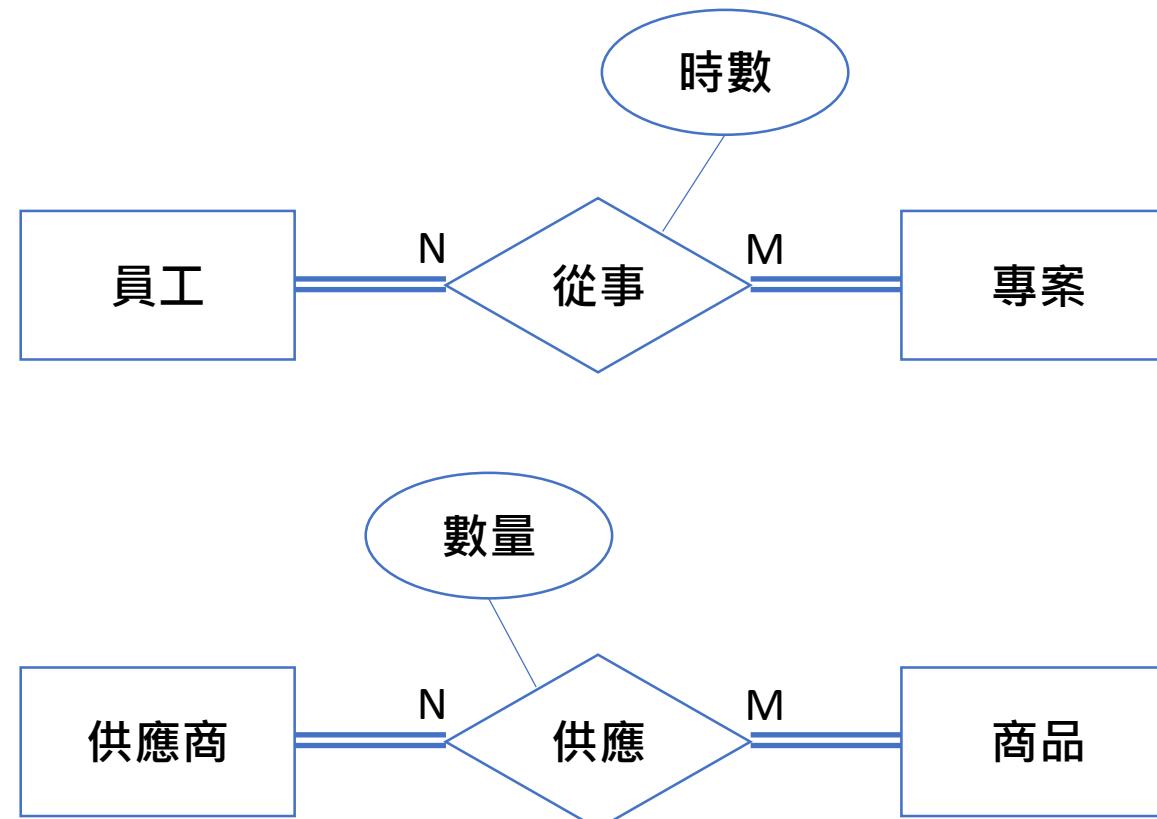
部分參與 —

全部參與 ==



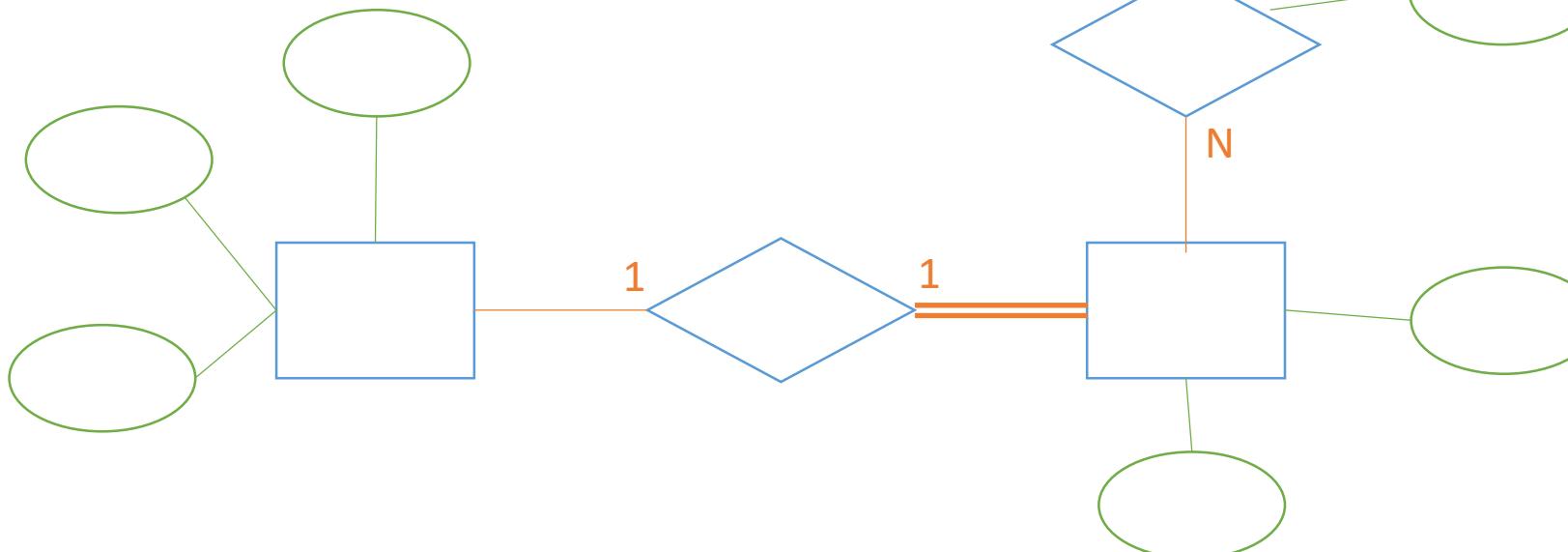
關係的屬性

- 關係也可能有一些屬性
該屬性會隨著該關係進行時改變。



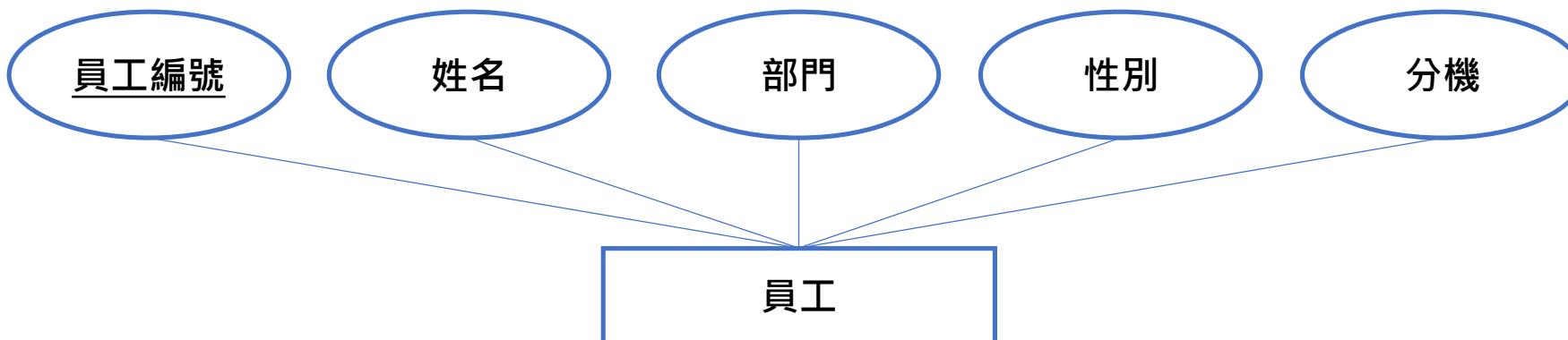
實體關係圖的發想順序

1. 根據需求找出實體與彼此關係
2. 定義關係類型與參與程度
3. 長出實體與關係上的屬性



E-R Model 轉換為資料表

- 實體轉換成資料表：以「員工實體與其屬性」為例，可轉換成「員工資料表」

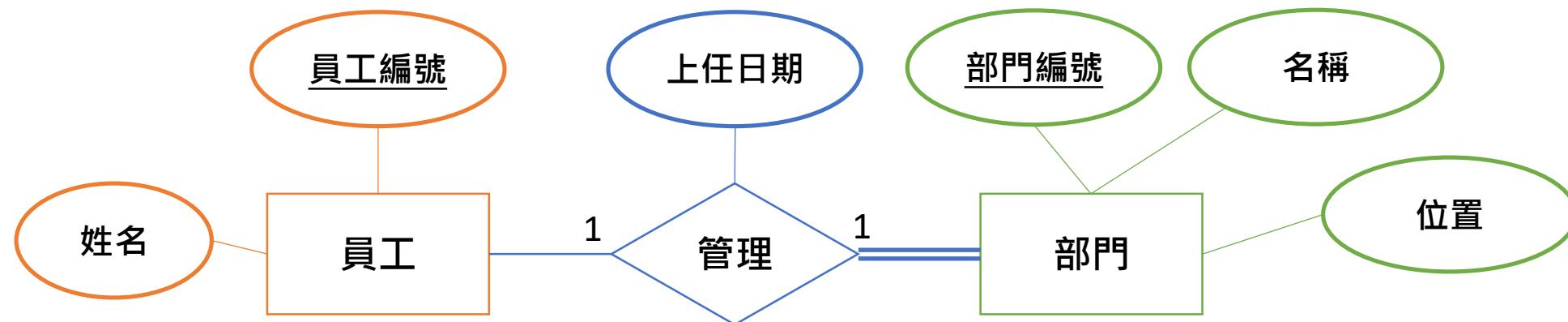


員工資料表

員工編號	姓名	部門	性別	分機
E01	Richard	Finance	M	8001
E02	Mary	Marketing	F	8002
E03	John	R&D	M	8003
...

E-R Model 轉換為資料表

- 一對一關係換成資料表



員工資料表

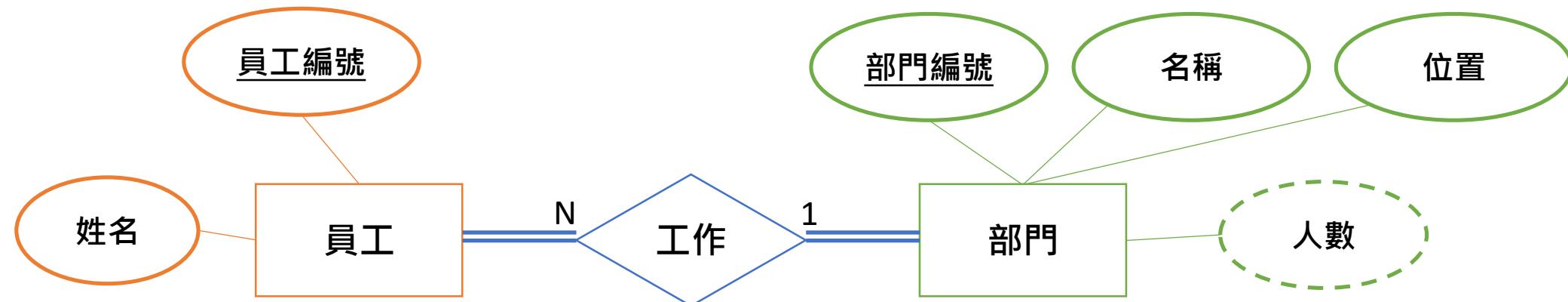
員工編號	姓名
E01	Richard
E02	Mary
E03	John
...	...

部門資料表

部門編號	名稱	位置	管理員工	上任日期
D01	Finance	10F	E01	2018/01/01
D02	Marketing	9F	E02	2019/04/05
D03	R&D	8F	E03	2018/03/05
...

E-R Model 轉換為資料表

- 一對多關係換成資料表



員工資料表

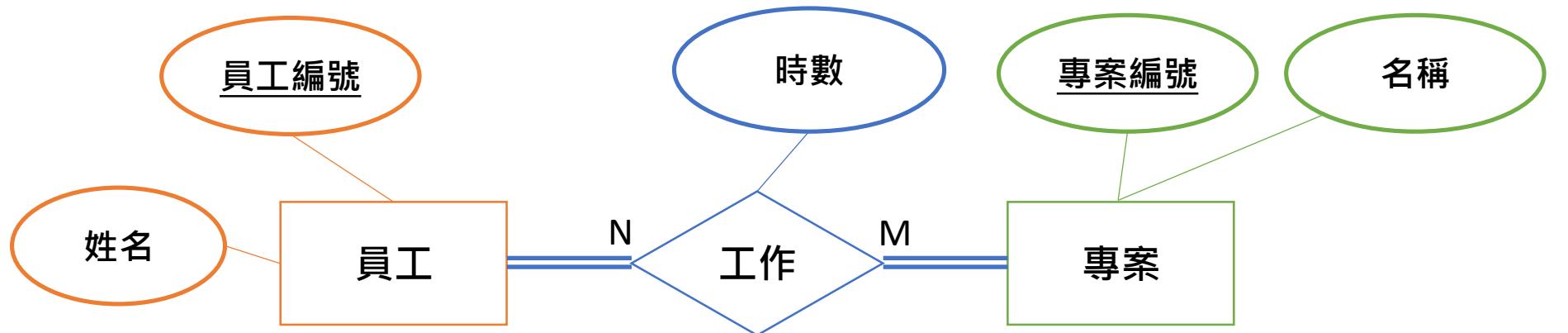
員工編號	姓名	工作部門
E01	Richard	D01
E02	Mary	D02
E03	John	D03
...

部門資料表

部門編號	名稱	位置
D01	Finance	10F
D02	Marketing	9F
D03	R&D	8F
...

E-R Model 轉換為資料表

- 多對多關係換成資料表



員工資料表

員工編號	姓名
E01	Richard
E02	Mary
E03	John
...	...

工作資料表

員工編號	專案編號	時數
E01	P01	8
E02	P01	4
E02	P03	4
...

專案資料表

專案編號	名稱
P01	財務專案
P02	行銷專案
P03	產品專案
...	...

繪製練習

請根據TibaMe的**學員(學號、姓名、電話)**與**班級(班級代碼、班級名稱、地點)**之間的關係，繪製實體關係圖，並根據該圖轉換成資料表。

Module 2.

關聯式資料庫簡介

- 2-1: 資料表正規化**
- 2-2: 資料表主鍵與外部鍵的關係**
- 2-3: SQL語言與資料庫**

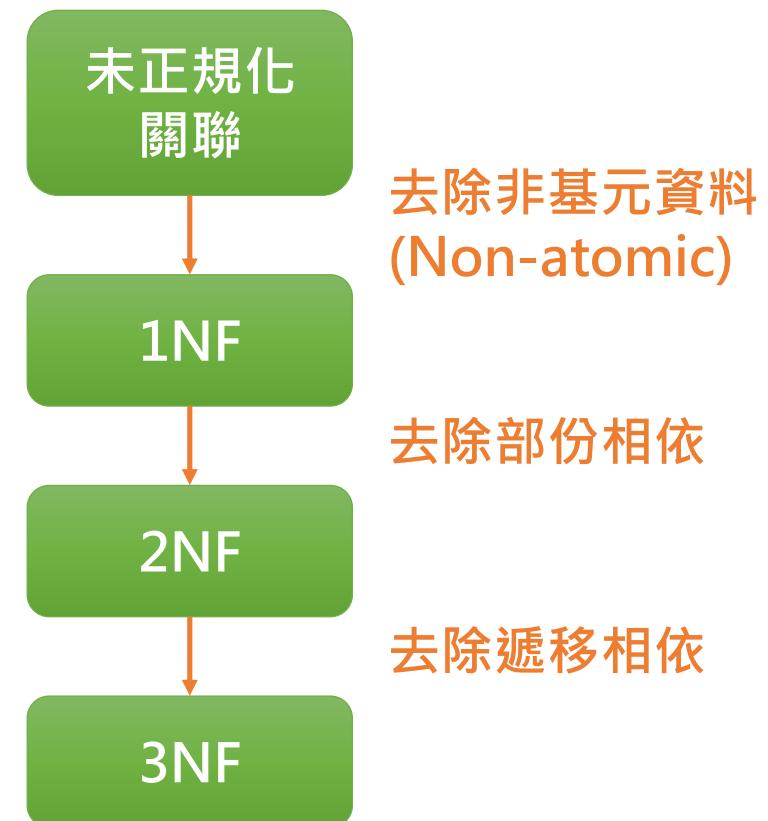
正規化(Normalization)

- 由 E. F. Codd 於 1971 年提出，根據不同的相依性來分割資料表欄位的關聯
 - 完成「良好邏輯設計」的關聯式資料模型
1. 降低不必要的資料重複儲存
 2. 避免更新造成資料異常

員工編號	姓名	部門	位置	職級	分紅	電話
E01	Richard	Finance	10F	S	20%	(02)2203-1001, 0912-3456
E02	Mary	Marketing	9F	A	10%	(02)2203-9001, 0912-1234
E03	John	R&D	8F	A	10%	(02)2203-8001
E04	Tom	R&D	8F	B	5%	(02)2203-8002
E05	Joe	Marketing	9F	C	3%	(02)2203-9002 0912-7890
E06	Cindy	Finance	10F	B	5%	(02)2203-1002
...

正規化的型式

- 正規化是以「漸進」的方式，進行表格分割，每步都會滿足某項「正規化類型」。
- 實務上，避免表格分割過細，影響查詢效率，通常做到第三正規化。



第一正規化型式 (1NF, First Normal Form)

資料表每個欄位內容都是基元值(Atomic Value)，則資料表滿足1NF。

員工編號	姓名	部門	位置	職級	分紅	電話
E01	Richard	Finance	10F	S	20%	(02)2203-1001, 0912-3456
E02	Mary	Marketing	9F	A	10%	(02)2203-9001, 0912-1234
E03	John	R&D	8F	A	10%	(02)2203-8001
E04	Tom	R&D	8F	B	5%	(02)2203-8002
E05	Joe	Marketing	9F	C	3%	(02)2203-9002 0912-7890
E06	Cindy	Finance	10F	B	5%	(02)2203-1002
...

第一正規化型式 (1NF, First Normal Form)

員工編號	姓名	部門	位置	性別	電話	手機
E01	Richard	Finance	10F	M	(02)2203-1001	0912-3456
E02	Mary	Marketing	9F	F	(02)2203-9001	0912-1234
E03	John	R&D	8F	M	(02)2203-8001	Null
E04	Tom	R&D	8F	M	(02)2203-8002	0912-7890
E05	Joe	Marketing	9F	M	(02)2203-9002	Null
E06	Cindy	Finance	10F	F	(02)2203-1002	Null
...

第二正規化型式 (2NF, Second Normal Form)

資料表須符合第一正規化特性外，所有主鍵以外的欄位，都必須與主鍵功能相依，則稱該資料表滿足 2NF。

員工編號	姓名	部門	位置	職級	分紅	電話	手機
E01	Richard	Finance	10F	S	20%	(02)2203-1001	0912-3456
E02	Mary	Marketing	9F	A	10%	(02)2203-9001	0912-1234
E03	John	R&D	8F	A	10%	(02)2203-8001	Null
E04	Tom	R&D	8F	B	5%	(02)2203-8002	0912-7890
E05	Joe	Marketing	9F	C	3%	(02)2203-9002	Null
E06	Cindy	Finance	10F	B	5%	(02)2203-1002	Null
...

第二正規化型式 (2NF, Second Normal Form)

員工編號	姓名	職級	分紅	電話	手機
E01	Richard	S	20%	(02)2203-1001	0912-3456
E02	Mary	A	10%	(02)2203-9001	0912-1234
E03	John	A	10%	(02)2203-8001	Null
E04	Tom	B	5%	(02)2203-8002	0912-7890
E05	Joe	C	3%	(02)2203-9002	Null
E06	Cindy	B	5%	(02)2203-1002	Null
...

部門	位置
Finance	10F
Marketing	9F
R&D	8F
R&D	8F
Marketing	9F
Finance	10F
...

第二正規化型式 (2NF, Second Normal Form)

員工編號	姓名	職級	分紅	電話	手機	部門
E01	Richard	S	20%	D01
E02	Mary	A	10%	D02
E03	John	A	10%	D03
E04	Tom	B	5%	D03
E05	Joe	C	3%	D02
E06	Cindy	B	5%	D01
...

部門編號	部門	位置
D01	Finance	10F
D02	Marketing	9F
D03	R&D	8F
...

第三正規化型式 (3NF, Third Normal Form)

資料表符合第二正規化，且所有主鍵以外的欄位都僅能與主鍵功能相依，且沒有與其它欄位相依，則稱該資料表滿足 3NF。』

員工編號	姓名	職級	分紅	電話	手機
E01	Richard	S	20%	(02)2203-1001	0912-3456
E02	Mary	A	10%	(02)2203-9001	0912-1234
E03	John	A	10%	(02)2203-8001	Null
E04	Tom	B	5%	(02)2203-8002	0912-7890
E05	Joe	C	3%	(02)2203-9002	Null
E06	Cindy	B	5%	(02)2203-1002	Null
...

第三正規化型式 (3NF, Third Normal Form)

員工編號	姓名	職級	電話	手機
E01	Richard	S	(02)2203-1001	0912-3456
E02	Mary	A	(02)2203-9001	0912-1234
E03	John	A	(02)2203-8001	Null
E04	Tom	B	(02)2203-8002	0912-7890
E05	Joe	C	(02)2203-9002	Null
E06	Cindy	B	(02)2203-1002	Null
...

職級	分紅
S	20%
A	10%
A	10%
B	5%
C	3%
B	5%
...	...

第三正規化型式 (3NF, Third Normal Form)

員工編號	姓名	職級	電話	手機	
E01	Richard	S	(02)2203-1001	0912-3456	
E02	Mary	A	(02)2203-9001	0912-1234	
E03	John	A	(02)2203-8001	Null	
E04	Tom	B	(02)2203-8002	0912-7890	
E05	Joe	C	(02)2203-9002	Null	
E06	Cindy	B	(02)2203-1002	Null	
...	

職級	分紅
S	20%
A	10%
B	5%
C	3%

資料正規化練習

請針對下列資料表**逐步**進行正規化，結果請滿足**第三正規化**型式。

客戶編號	客戶名稱	客戶地址	運費	商品編號	商品名稱	訂購數量	訂購日期
C01	David	台北	100	P01	乾洗手	10, 20	3/1, 3/5
C01	David	台北	100	P02	酒精	15	3/1
C02	Jerry	台中	200	P03	口罩	20, 40	3/6, 3/8

資料表

- 關聯式資料庫中最主要的資料庫物件
- 資料儲存在資料表中
- 欄位(Column)
 - 每一個欄位都必須命名而且必須設定資料型態和資料長度, 用來存放欄位資料.
- 資料列(Row)
 - 資料列是資料表中的一筆記錄資料

Bit → Byte → Field → Record → File → Database

Bit → Byte → Column → Row → Table → Database

資料表的相關用語

(1) 主鍵(Primary Key)

- 用於識別每一橫列的欄位組合，其特性如下
 - 具有唯一性(Unique)
 - 必填，不為空值(Not Null)

(2) 直欄(Column)

- 為實體(Entity)或表格中的屬性，其特性如下
 - 具有相同的資料型態
 - 儲存相似的內容資料

(3) 橫列(Row)

- 由1個以上的直欄所組成的一筆記錄(Record)
- 作為SQL查詢所回傳的資料單位

(4) 欄位(Field)

- 直欄與橫列交會的地方，作為使用者儲存資料的地方

(5) 空值(NULL)

- 欄位如果沒有儲存資料時稱空值

(6) 外來鍵(Foreign Key)

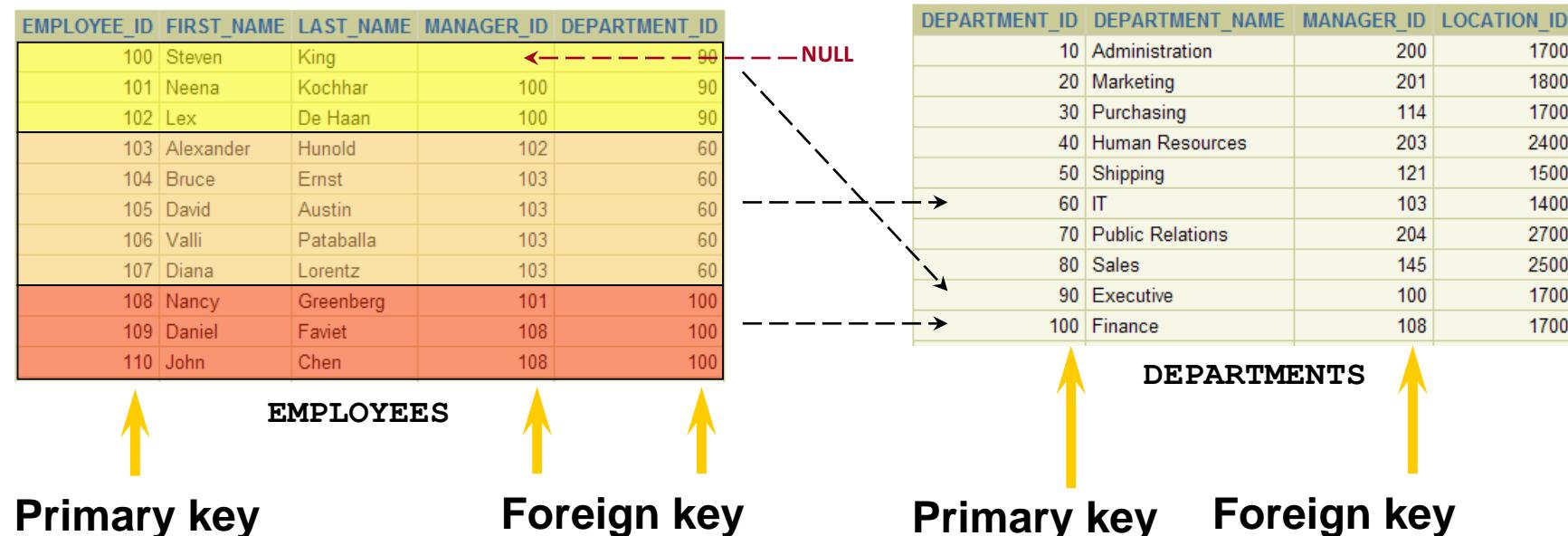
- 欄位所輸入的資料必需出現在所參照的表格中主鍵或唯一鍵的鍵值
- 欄位亦可為空值

EMPLOYEE_ID	LAST_NAME	FIRST_NAME	SALARY	COMMISSION_PCT	DEPARTMENT_ID
100	King	Steven	24000		90
101	Kochhar	Neena	17000		90
102	De Haan	Lex	17000		60
103	Hunold	Alexander	9000		60
104	Ernst	Bruce	6000		60
107	Lorentz	Diana	4200		60
124	Mourgos	Kevin	5800		50
141	Rajs	Trenna	3500		50
142	Davies	Curtis	3100		50
143	Matos	Randall	2600		50
144	Vargas	Peter	2500		50
149	Zlotkey	Eleni	10500	.2	80
174	Abel	Ellen	11000	.3	80
176	Taylor	Jonathon	8600	.2	80
178	Grant	Kimberely	7000	.15	
200	Whalen	Jennifer	4400		10
201	Hartstein	Michael	13000		20
202	Fay	Pat	6000		20
205	Higgins	Shelley	12000		110
206	Gietz	William	8300		110

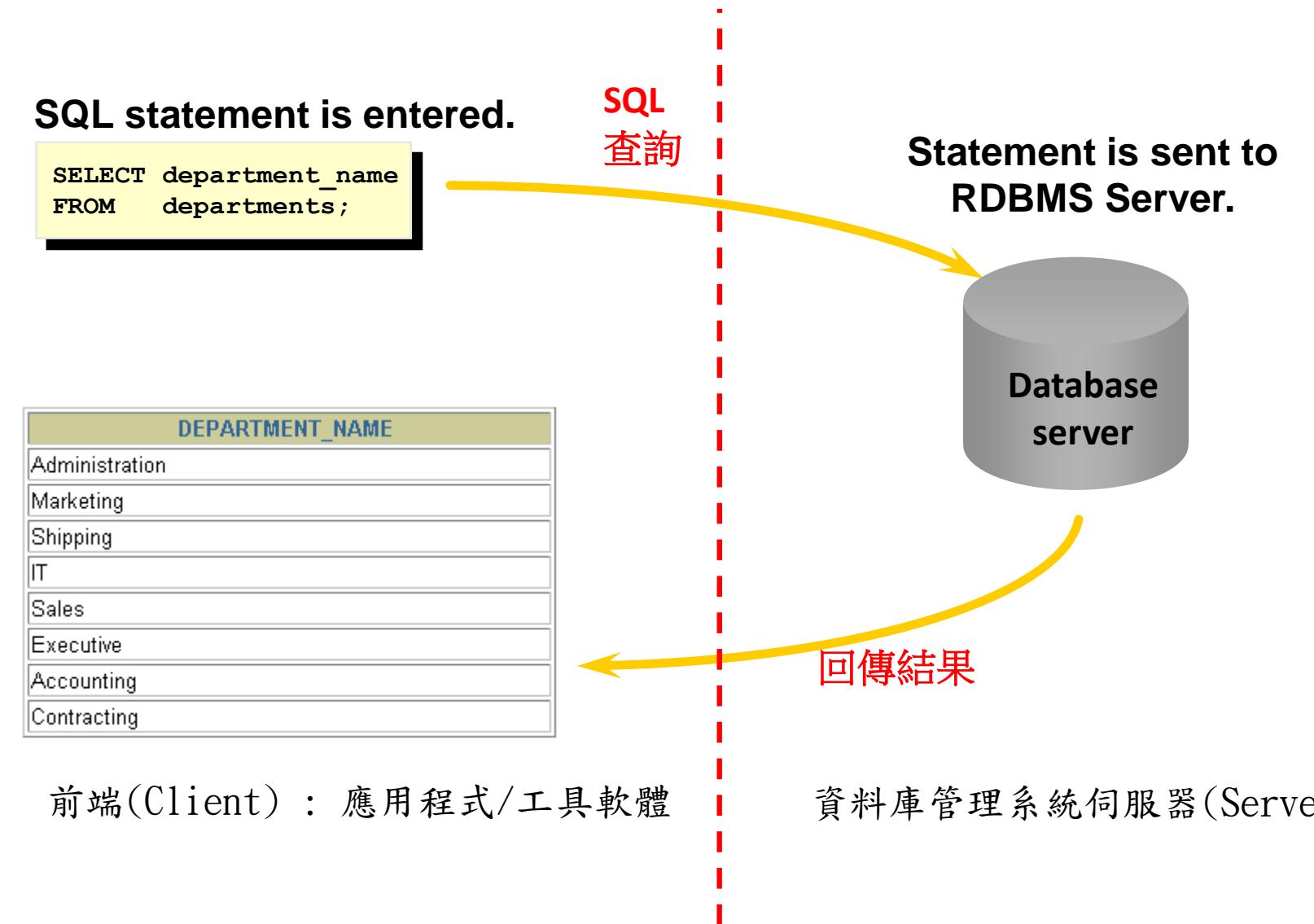
2-2: 資料表主鍵與外部鍵的關係

資料表的關聯

- 透過主鍵與外來鍵，建立不同資料表間的關係。
- 每個外來鍵只能參考一個主鍵或唯一鍵。



關聯式資料庫管理系統的使用方式



結構化查詢語言(SQL)簡介

- IBM早期使用其開發的資料庫系統(System R)中的規範語言(SEQUEL)，1980年改名為SQL。
- 1979年ORACLE公司首先提供商用的SQL，IBM公司在DB2和SQL/DS資料庫系統中也實現了SQL。
- 1986年10月美國ANSI對SQL進行規範後，此作為關聯式資料庫管理系統(RDBMS)的標準語言(ANSI X3. 135-1986)。
- 1987年得到國際標準組織(ISO)的支持，成為國際標準。
- 1989年美國ANSI採納在ANSI X3.135-1989報告中定義的關係數據庫管理系統的SQL標準語言，稱為ANSI SQL 89。
- 美國國家標準局陸續制定出ANSI SQL-92,以定義出SQL的關鍵字與語法標準。其後更陸續發展出SQL-99,及目前最新版的SQL-2003

結構化查詢語言(SQL)簡介

- SQL 是第四代高階的程式語言
- 使用SQL指令來操作資料庫
- 不要求使用者指定其資料的存放方法
- 不需要使用者瞭解其具體的資料存放方式
- 不同底層結構的資料庫，可用相同SQL語言來操作
- SQL 語言可以寫出非常複雜的語句

SQL語言的分類

- ▶ 資料查詢語言(DQL : Data Query Language)
 - SELECT : 查詢資料庫中表格的資料
- ▶ 資料定義語言(DDL : Data Definition Language)
 - CREATE : 可建立資料庫中的物件
 - ALTER : 可變更資料庫中物件的結構
 - DROP : 可刪除資料庫中的物件
 - RENAME : 可變更資料庫中的物件名稱
 - TRUNCATE : 清除資料庫中表格內的資料
- ▶ 資料操作語言(DML : Data Manipulation Language)
 - INSERT : 可新增資料庫表格內的資料
 - UPDATE : 可更新資料庫表格內的資料
 - DELETE : 可刪除資料庫表格內的資料
- ▶ 資料控制語言(DCL : Data Control Language)
 - GRANT : 授予權限
 - REVOKE : 除移權限

Module 3.

MySQL簡介與安裝

3-1: MySQL源起與特性

3-2: MySQL下載

3-3: MySQL安裝

MySQL的歷史

- ▶ MySQL是目前很受歡迎的資料庫管理系統之一
 - 原本是一個開放原始碼的關聯式資料庫管理系統，原開發者瑞典的 MySQL AB公司
 - 2000年7月加入了GNU Public License (GNU GPL)
 - 該公司於2008年被昇陽微系統（Sun Microsystems）收購
 - 2009年，甲骨文公司（Oracle）收購昇陽微系統公司，MySQL成為Oracle旗下產品
 - MySQL 5.7 版 => MySQL 8.0 版

[MySQL Community Server \(GPL\)](#)

(Current Generally Available Release: 5.7.9)

MySQL Community Server is the world's most popular open source database.



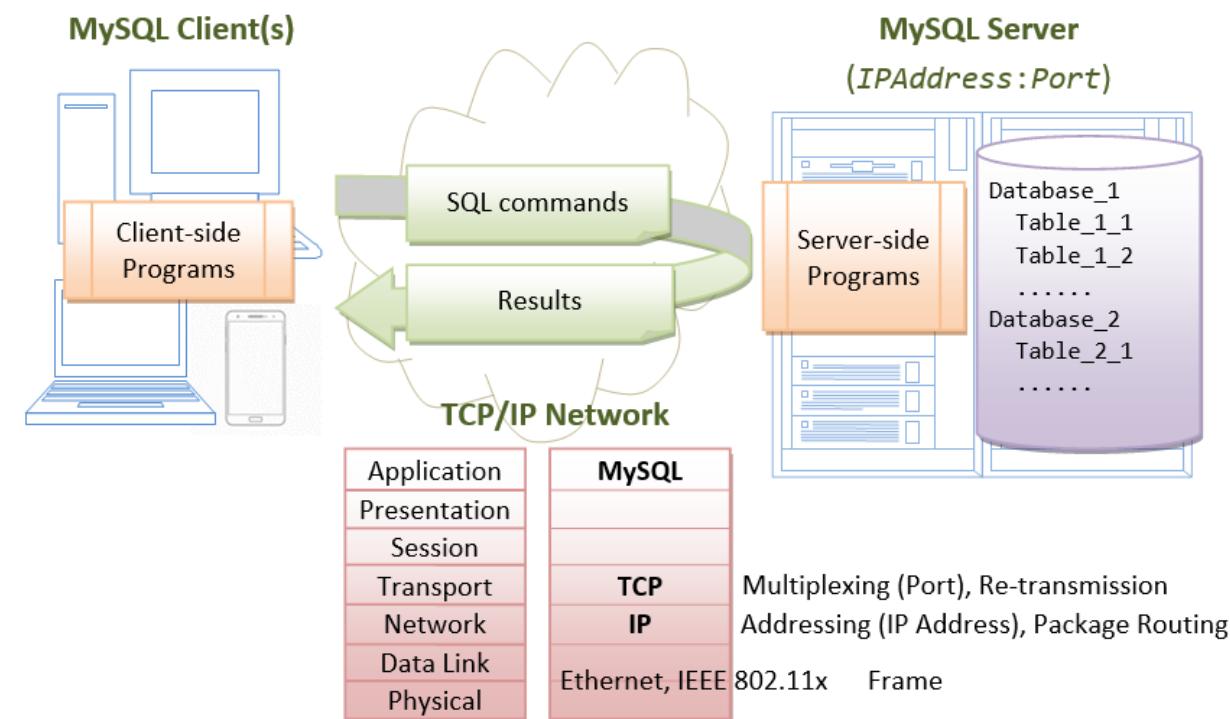
MySQL關聯式資料庫管理系統(RDBMS)

- ▶ MySQL是一個關聯式資料庫管理系統
 - 開放原始碼(Community Server)
 - 速度快的，可靠的，且易於使用
 - 主從式架構(Client/Server)
 - 大量的網路貢獻者，提供MySQL相關軟體
 - [MariaDB](#)是MySQL分支出來的產品，是MySQL創辦人 Michael Widenius 的另一套開放源碼資料庫。

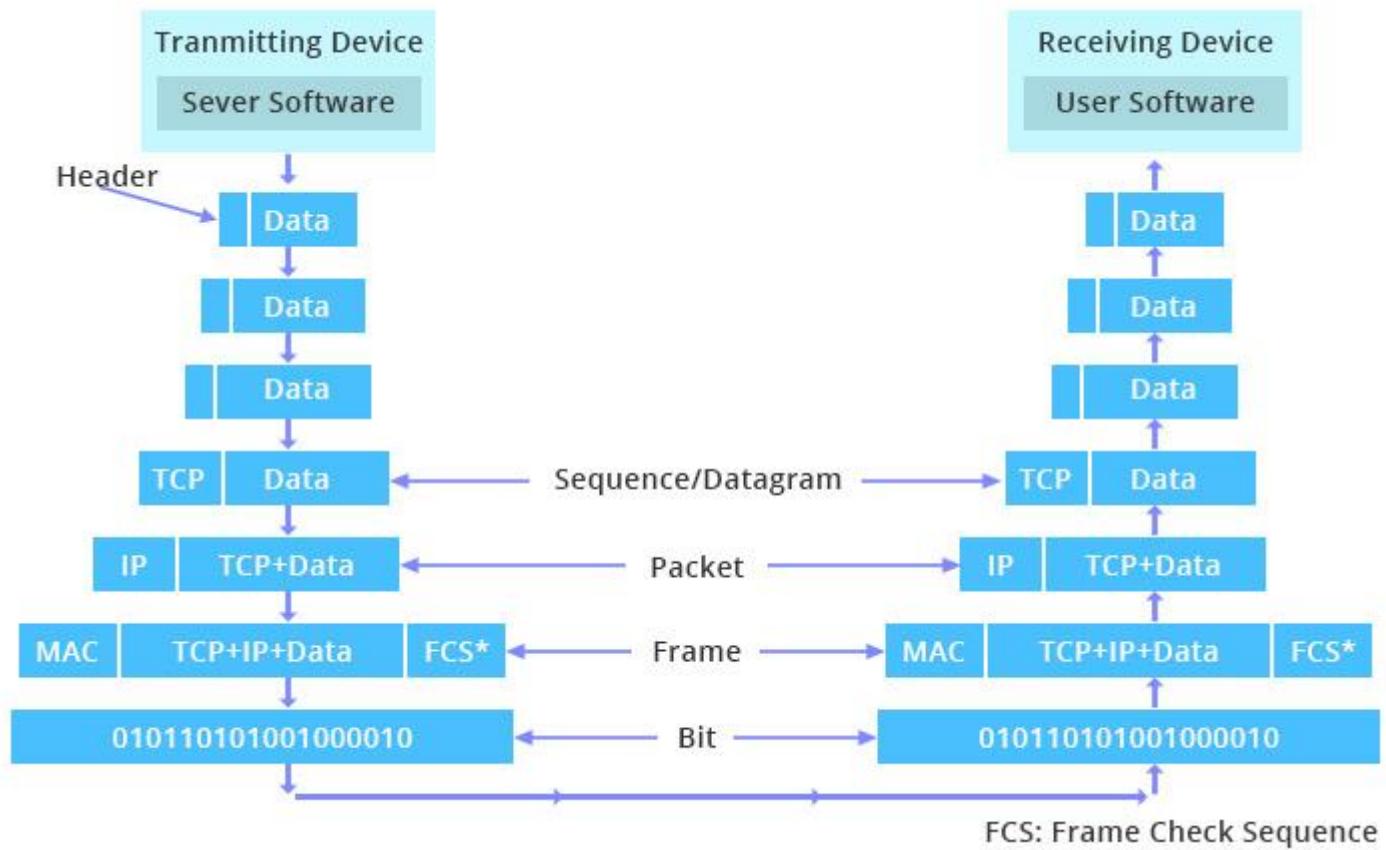


連接使用 MySQL

- ▶ 以TCP/IP為基礎的主從式架構(Client/Server)
- ▶ 工具軟體
 - 圖形化使用者介面(Graphic User Interface)
 - 命令列工具程式
- ▶ 客戶端應用程式
 - PHP ,Python , Java,



TCP/IP協定與網路封包的傳送



Module 3.

MySQL簡介與安裝

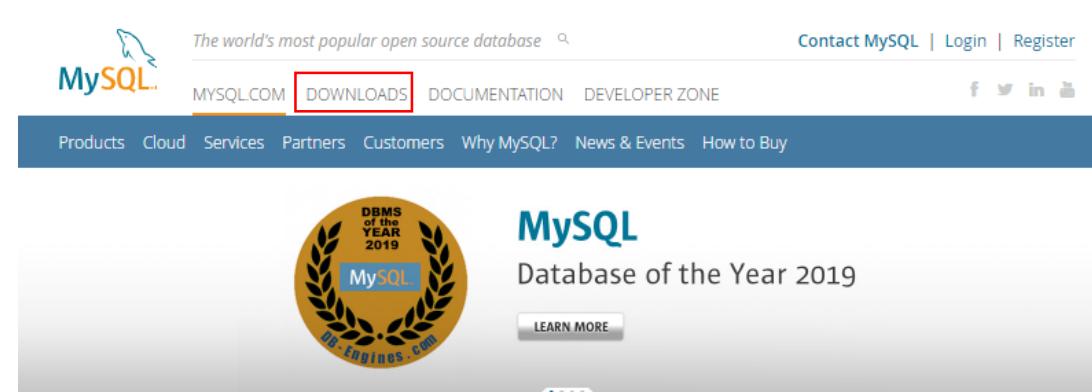
3-1: MySQL源起與特性

3-2: MySQL下載

3-3: MySQL安裝

下載 MySQL Installer (安裝程式)

<https://www.mysql.com/>



The screenshot shows the official MySQL website at <https://www.mysql.com/>. The top navigation bar includes links for MySQL.COM, DOWNLOADS (which is highlighted with a red box), DOCUMENTATION, and DEVELOPER ZONE. Below the navigation is a secondary menu with links for Products, Cloud, Services, Partners, Customers, Why MySQL?, News & Events, and How to Buy. A prominent banner in the center features a gold circular award badge for "DBMS of the YEAR 2019" from DB-Engines.com, with the text "MySQL Database of the Year 2019" and a "LEARN MORE" button. Below the banner, there are three product sections: "MySQL Enterprise Edition" (described as the most comprehensive set of advanced features), "MySQL Cluster CGE" (described as enabling users to meet database challenges with scalability, uptime, and agility), and "MySQL for OEM/ISV" (described as being used by over 2000 ISVs, OEMs, and VARs to make their applications more competitive). At the bottom of the page, there are links for "Free Webinars" and "White Papers".

MySQL Community (GPL) Downloads

The screenshot shows the MySQL website's 'Downloads' section. At the top, there's a navigation bar with the MySQL logo, a search bar, and links for 'Contact MySQL', 'Login', and 'Register'. Below the navigation, there are tabs for 'MYSQL.COM', 'DOWNLOADS' (which is selected), 'DOCUMENTATION', and 'DEVELOPER ZONE'. Social media icons for Facebook, Twitter, LinkedIn, and YouTube are also present.

MySQL Enterprise Edition

MySQL Enterprise Edition includes the most comprehensive set of advanced features and management tools for MySQL.

Features listed include:

- MySQL Database
- MySQL Storage Engines (InnoDB, MyISAM, etc.)
- MySQL Connectors (JDBC, ODBC, .Net, etc.)
- MySQL Replication
- MySQL Partitioning
- MySQL Router
- MySQL Shell
- MySQL Workbench
- 24x7 Technical Support
- MySQL Enterprise Backup
- MySQL Enterprise Monitor
- MySQL Enterprise HA
- MySQL Enterprise Transparent Data Encryption (TDE)
- MySQL Enterprise Masking and De-identification
- MySQL Enterprise Firewall
- MySQL Enterprise Encryption
- MySQL Enterprise Audit

Buttons for 'Learn More', 'Customer Download', and 'Trial Download' are shown at the bottom of the main section.

Contact Sales

USA: +1-866-221-0634
Canada: +1-866-221-0634

Germany: +49 89 143 01280
France: +33 1 57 60 83 57
Italy: +39 02 249 59 120
UK: +44 207 553 8447

Japan: 0120-065556
China: 10800-811-0823
India: 0008001005870

[More Countries »](#)

MySQL Cluster CGE

MySQL Cluster is a real-time open source transactional database designed for fast, always-on access to data under high throughput conditions.

MySQL Cluster
MySQL Cluster Manager
Plus, everything in MySQL Enterprise Edition

[Learn More »](#)
[Customer Download » \(Select Patches & Updates Tab, Product Search\)](#)
[Trial Download »](#)

[MySQL Community \(GPL\) Downloads »](#)

下載 MySQL Installer (安裝程式)

<https://dev.mysql.com/downloads/installer/>

④ MySQL Community Downloads

◀ MySQL Installer

The screenshot shows the MySQL Community Downloads page. At the top, there are tabs for "General Availability (GA) Releases" (which is selected), "Archives", and a help icon. Below the tabs, the title "MySQL Installer 8.0.20" is displayed. A dropdown menu "Select Operating System" is set to "Microsoft Windows". To the right, a link says "Looking for previous GA versions?". Two download options are listed for "Windows (x86, 32-bit), MSI Installer":

- (mysql-installer-web-community-8.0.20.0.msi) Version 8.0.20, 24.4M, Download button, MD5: 26ae47807122bf0052b99ebf893b0dac | Signature
- (mysql-installer-community-8.0.20.0.msi) Version 8.0.20, 420.6M, Download button, MD5: a69c77fe737654d8931079b4623b9e1a | Signature

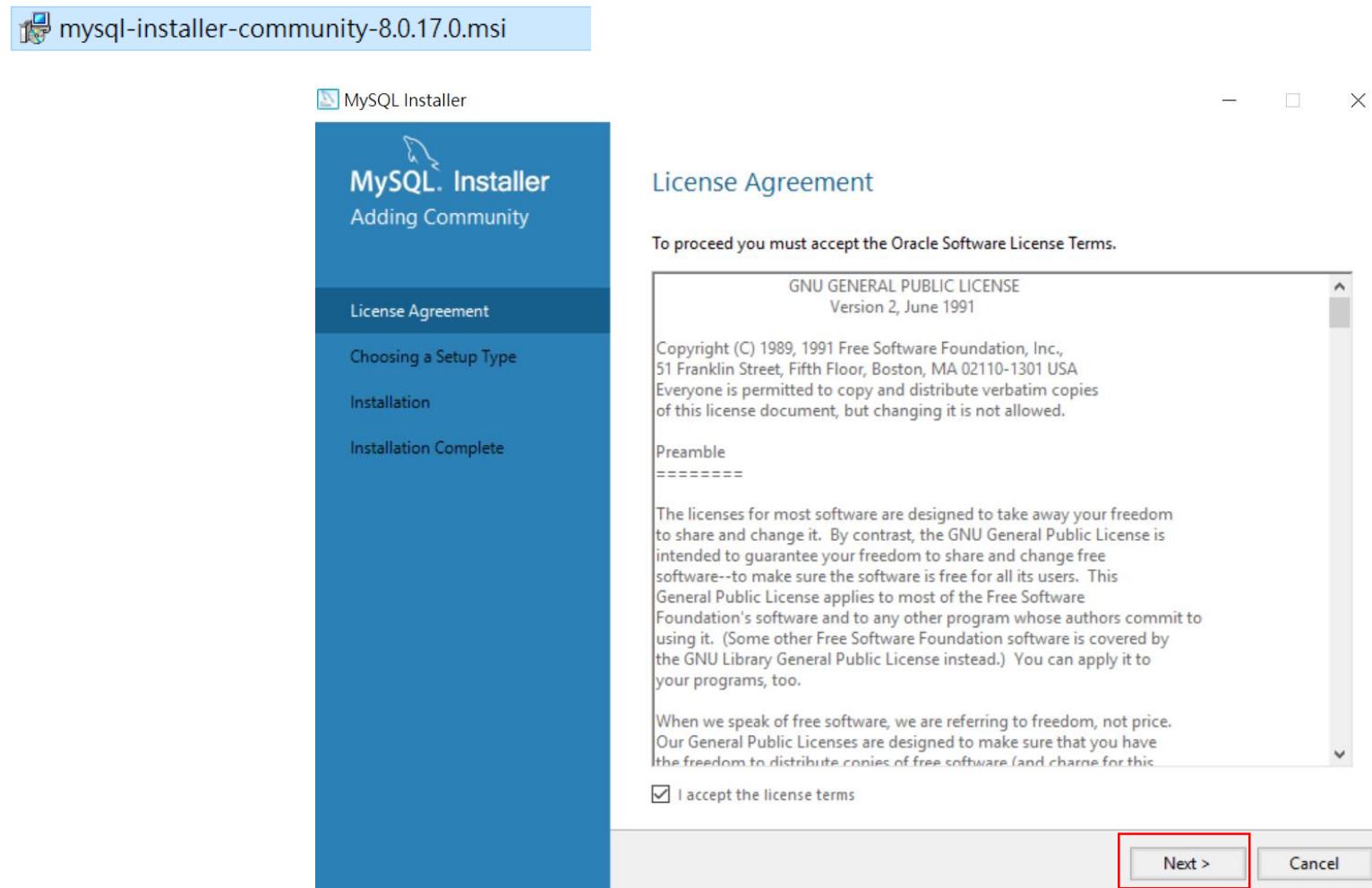
A note at the bottom suggests using MD5 checksums and GnuPG signatures for package integrity.

ORACLE © 2020, Oracle Corporation and/or its affiliates

[Legal Policies](#) | [Your Privacy Rights](#) | [Terms of Use](#) | [Trademark Policy](#) | [Contributor Agreement](#) | [Cookie偏好](#)

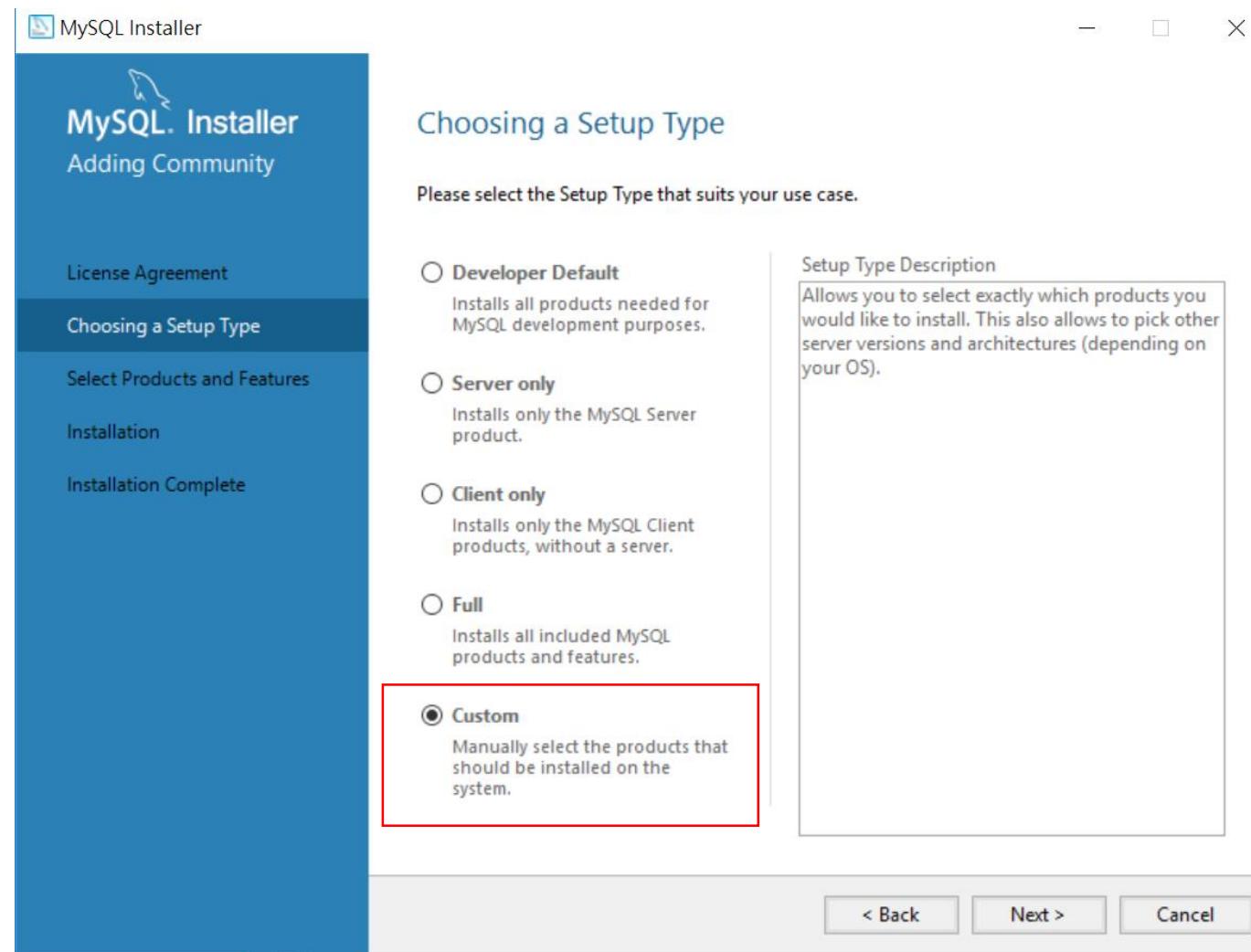
3-3: MySQL安裝

安裝MySQL

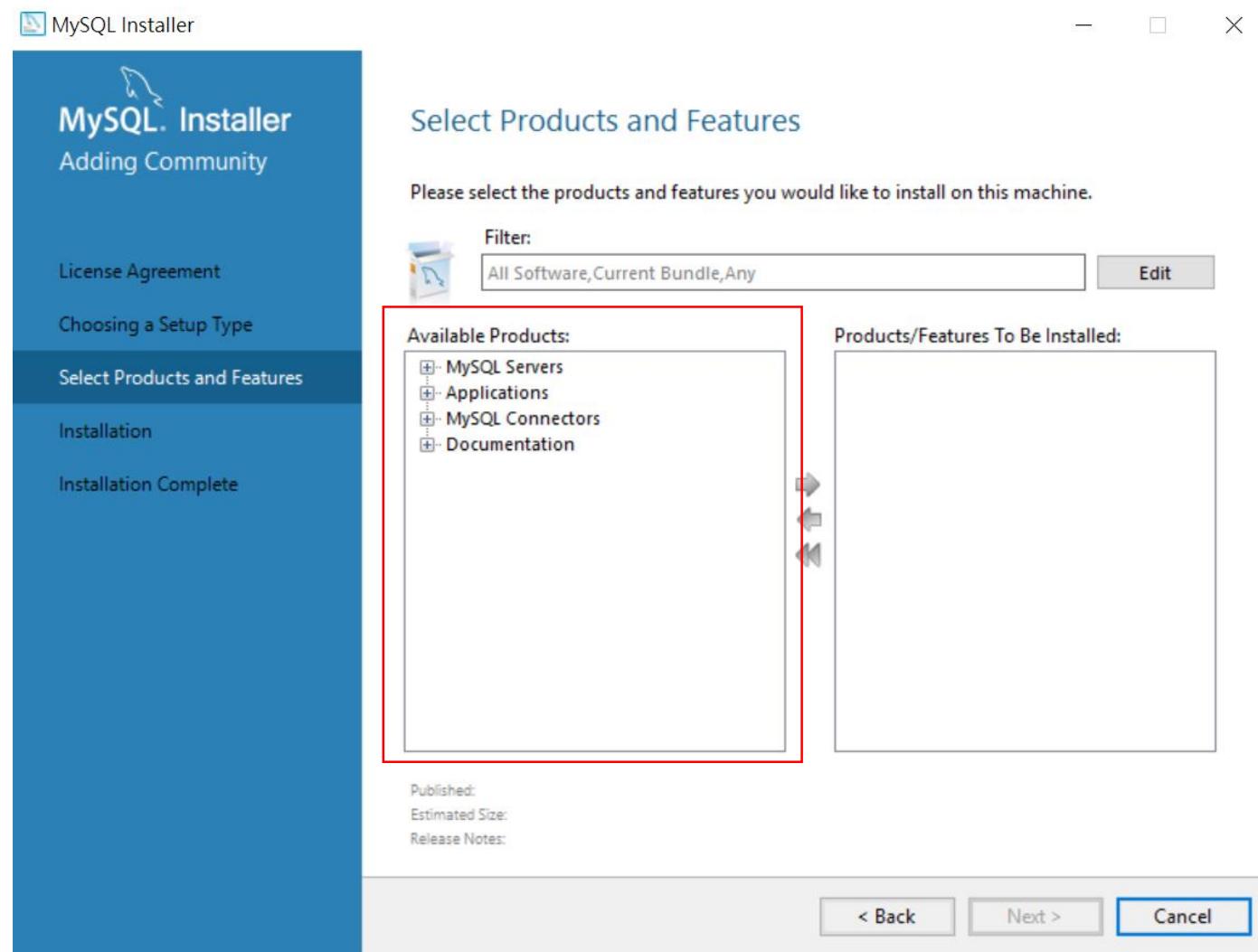


3-3: MySQL安裝

安裝MySQL

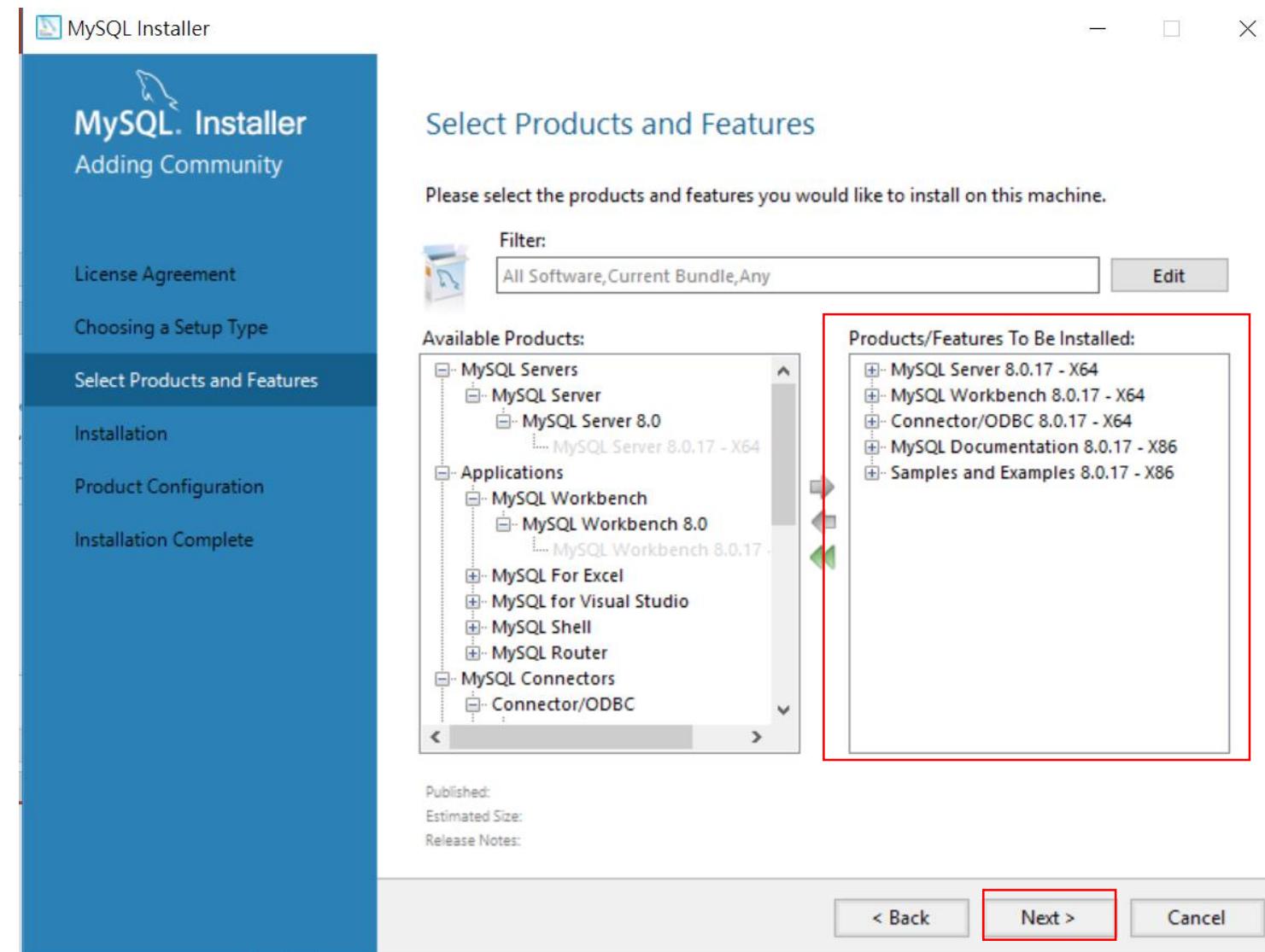


安裝MySQL

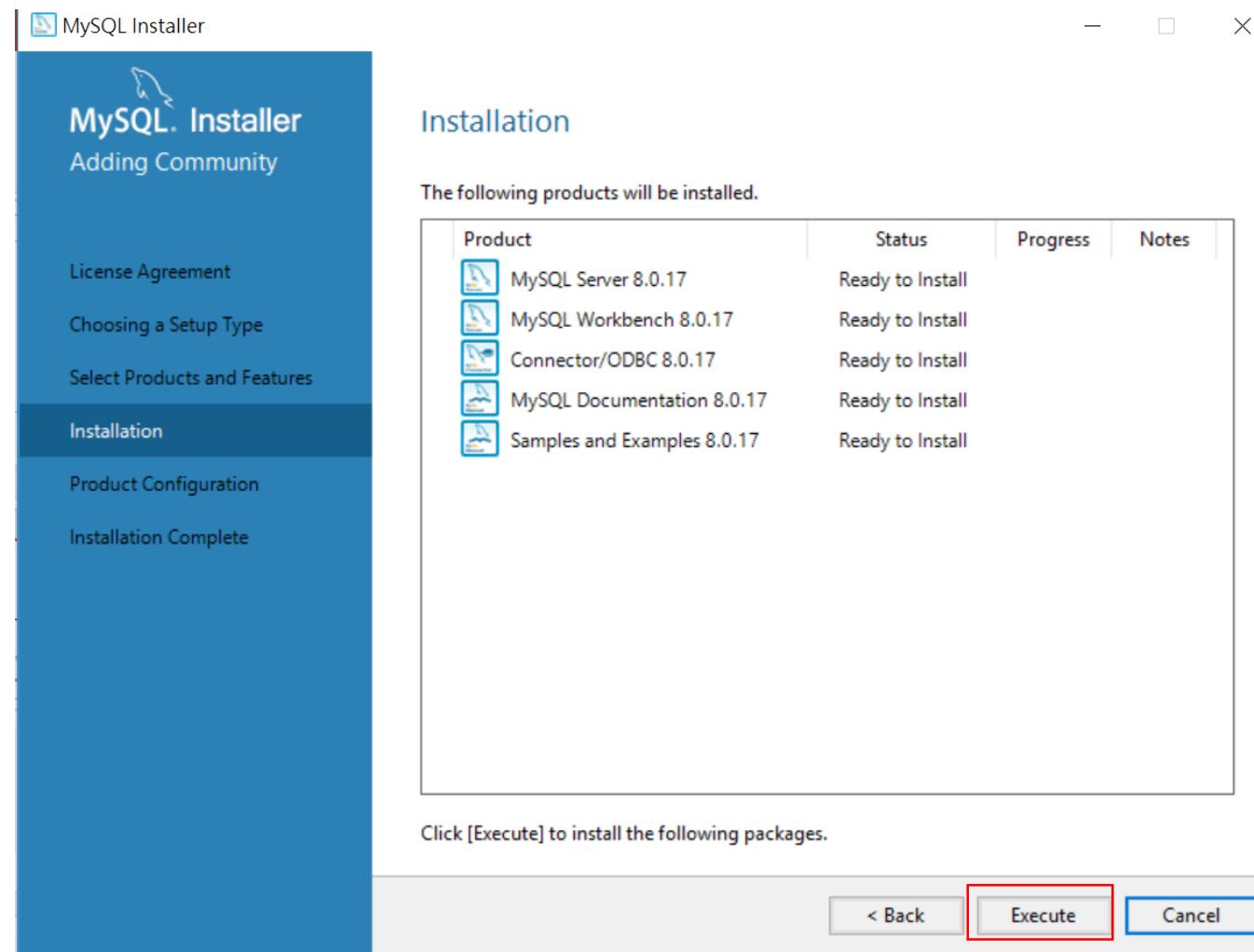


3-3: MySQL安裝

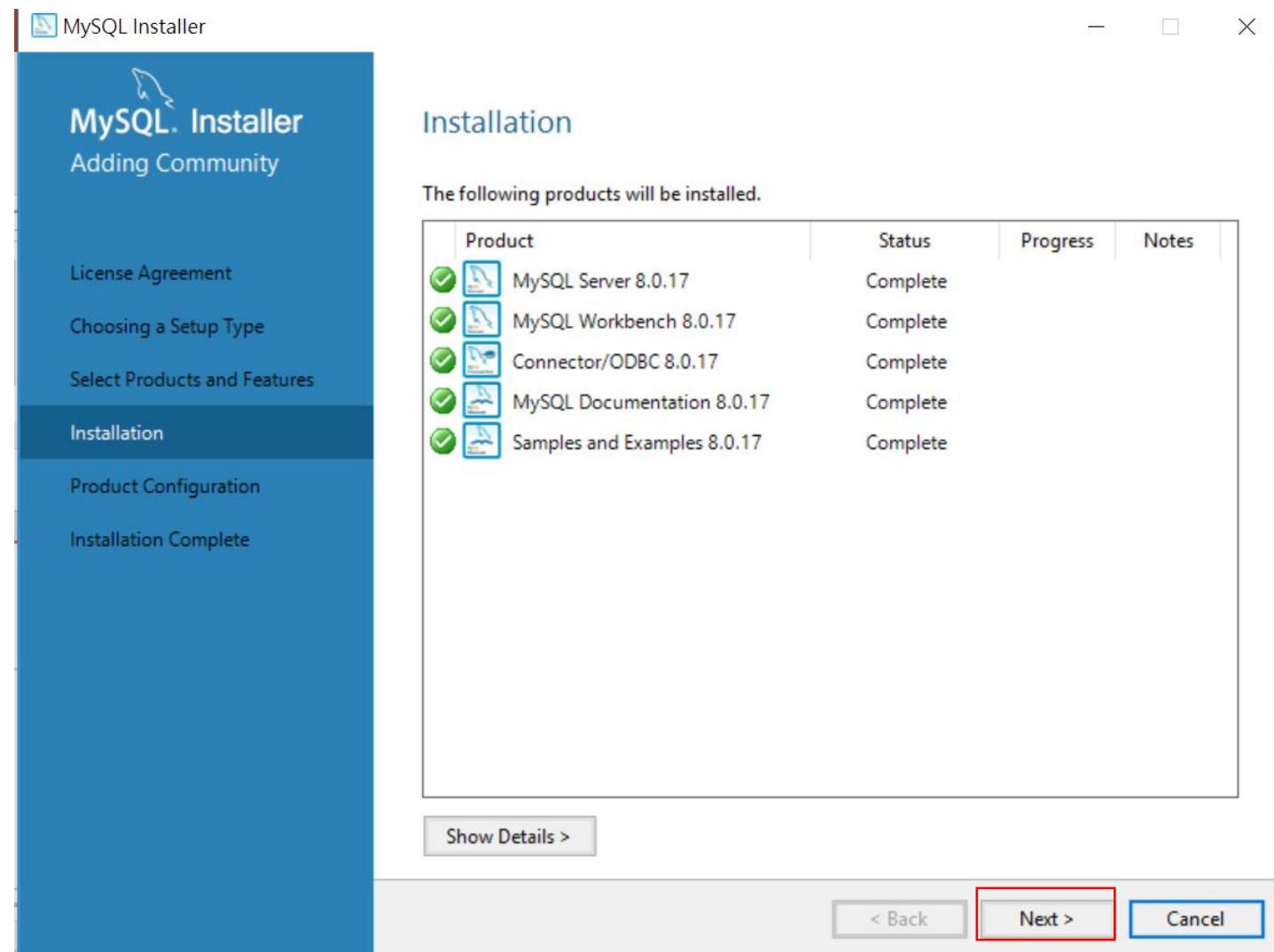
安裝MySQL



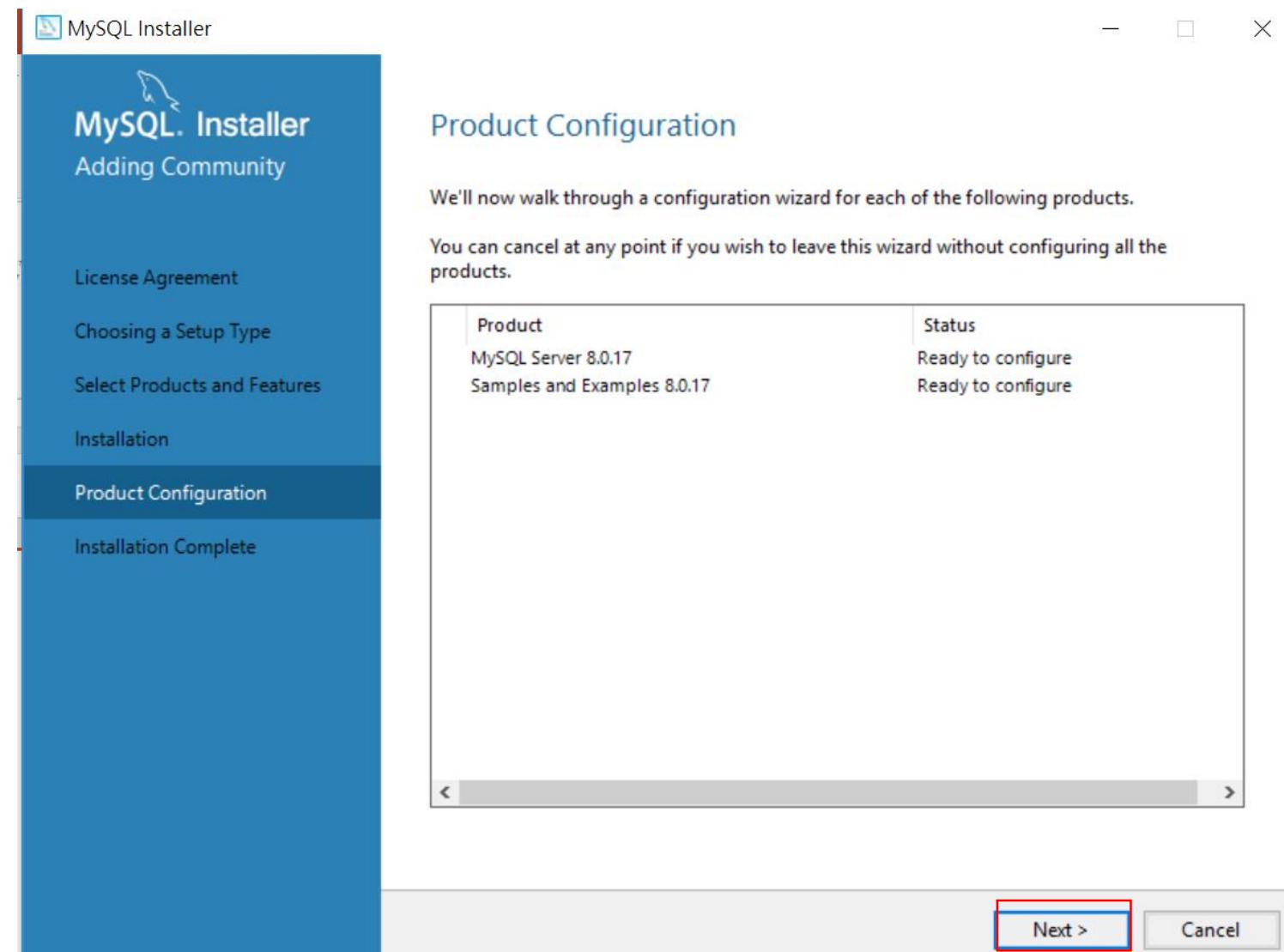
安裝MySQL



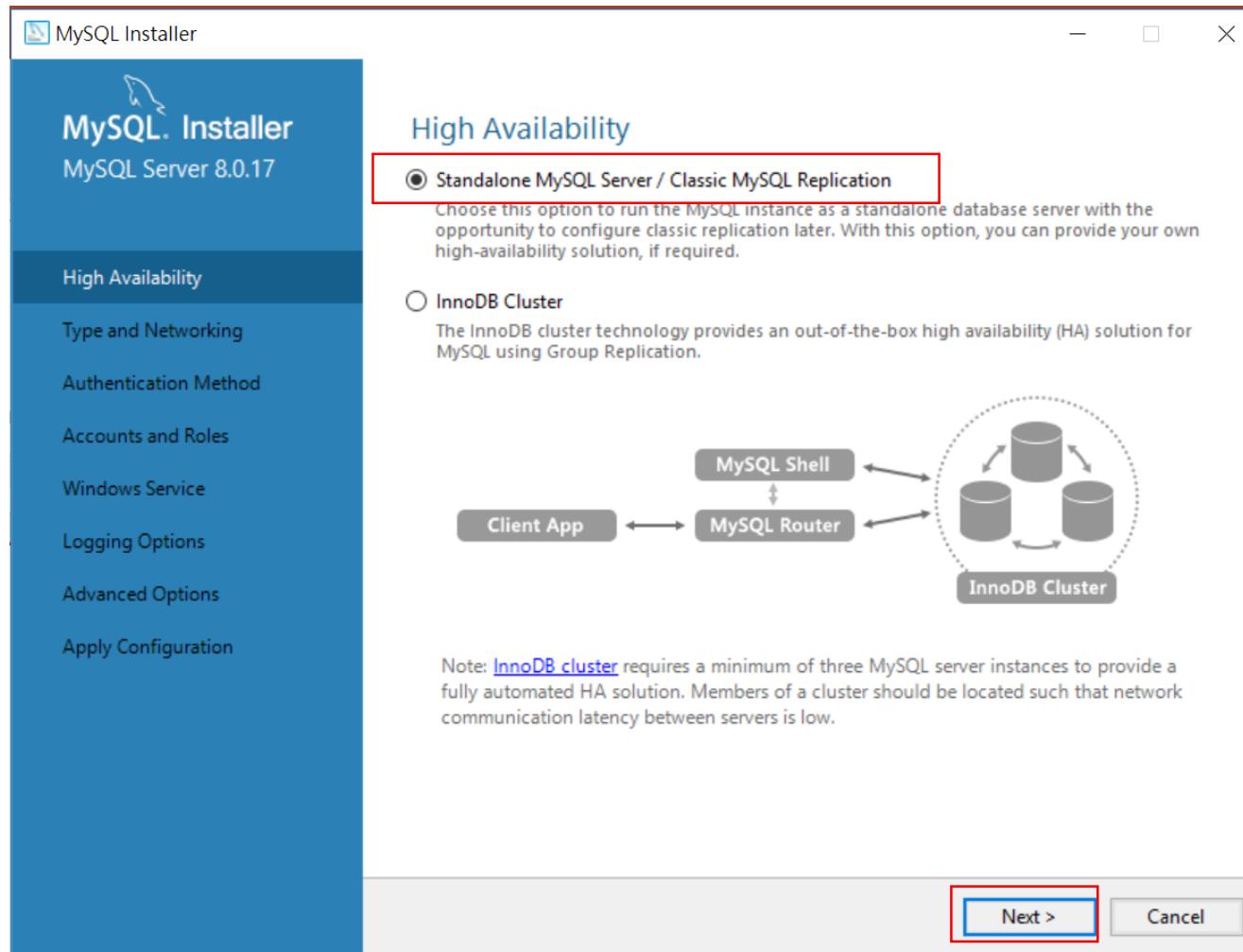
安裝MySQL



安裝MySQL

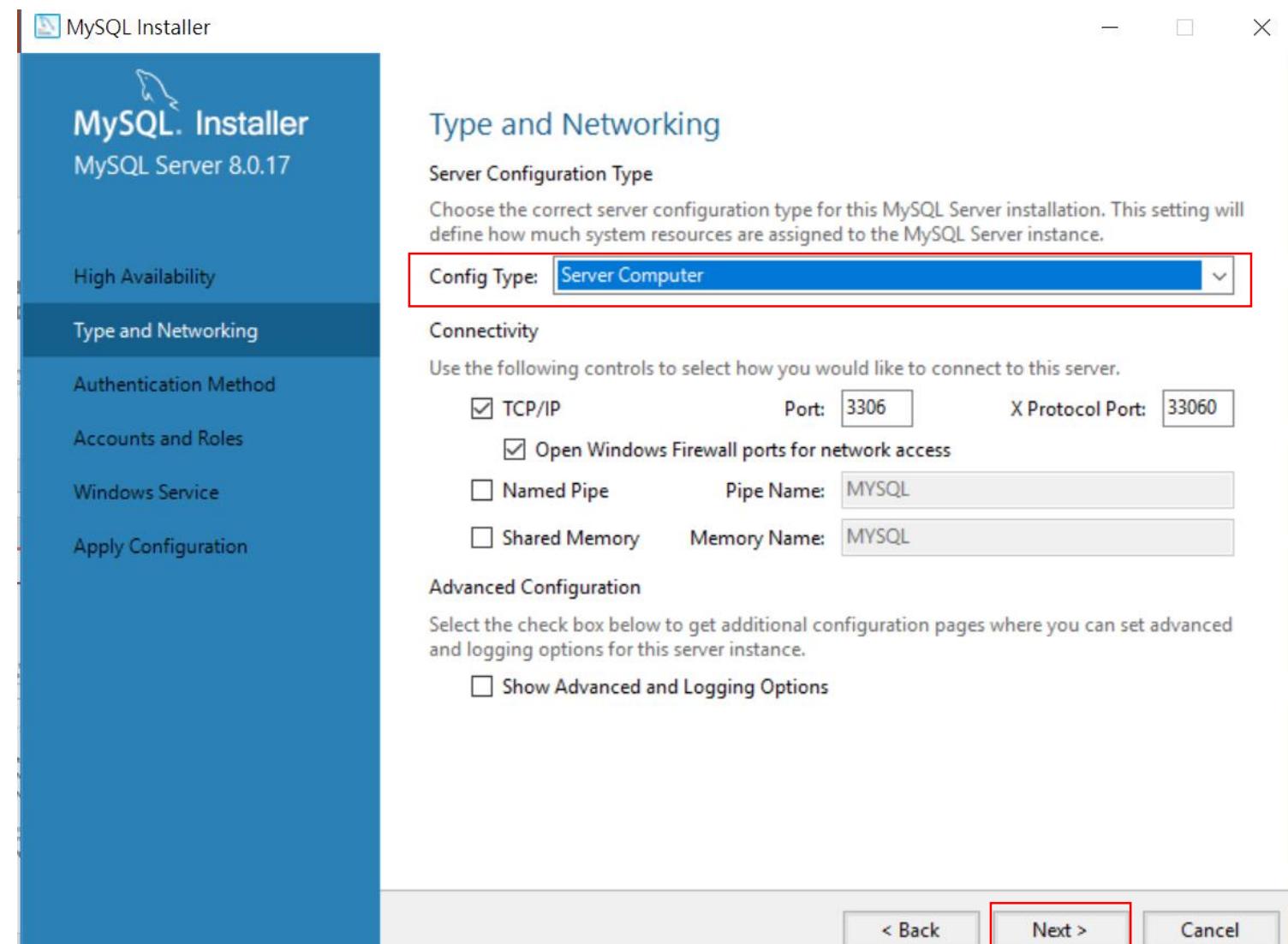


安裝MySQL

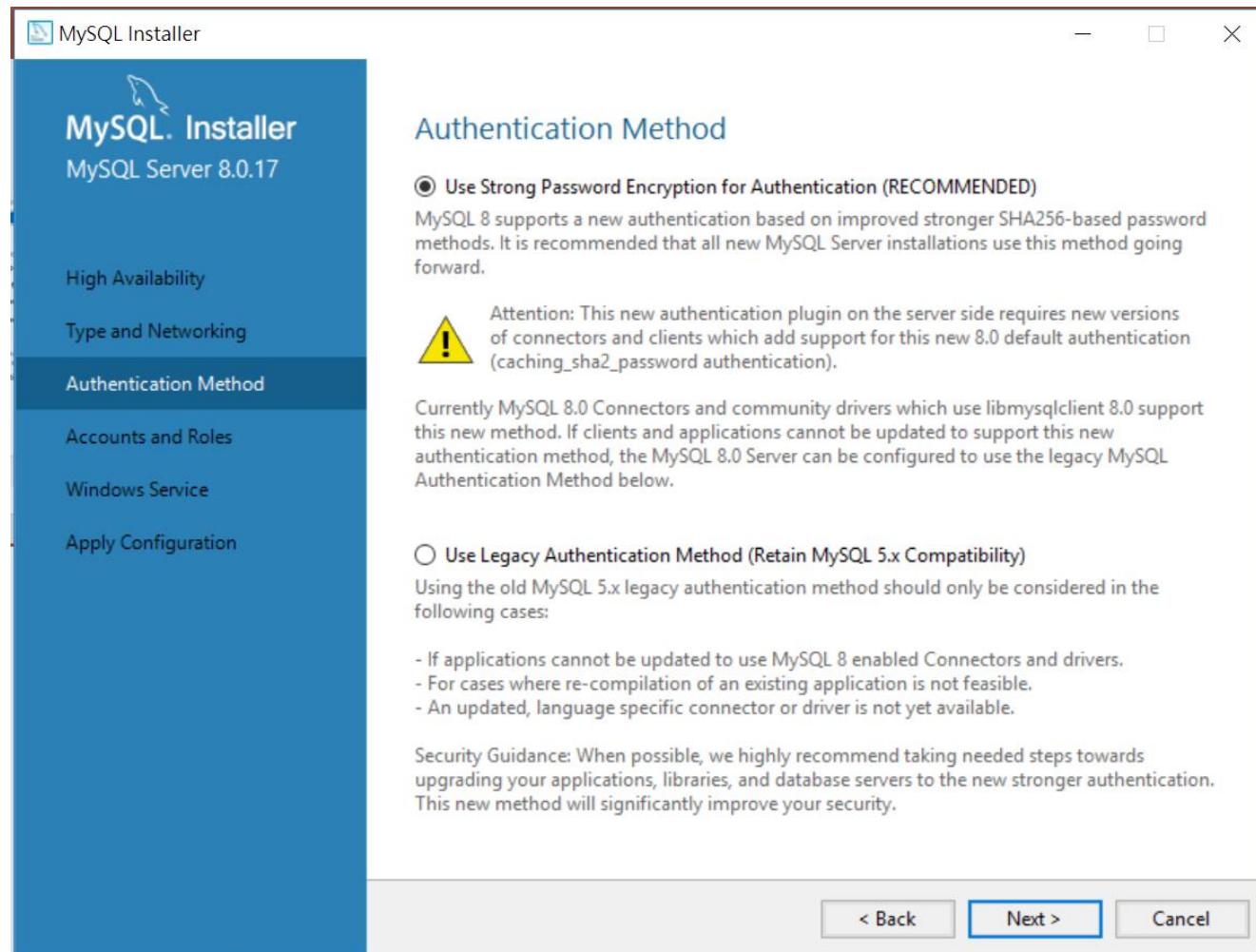


3-3: MySQL安裝

安裝MySQL

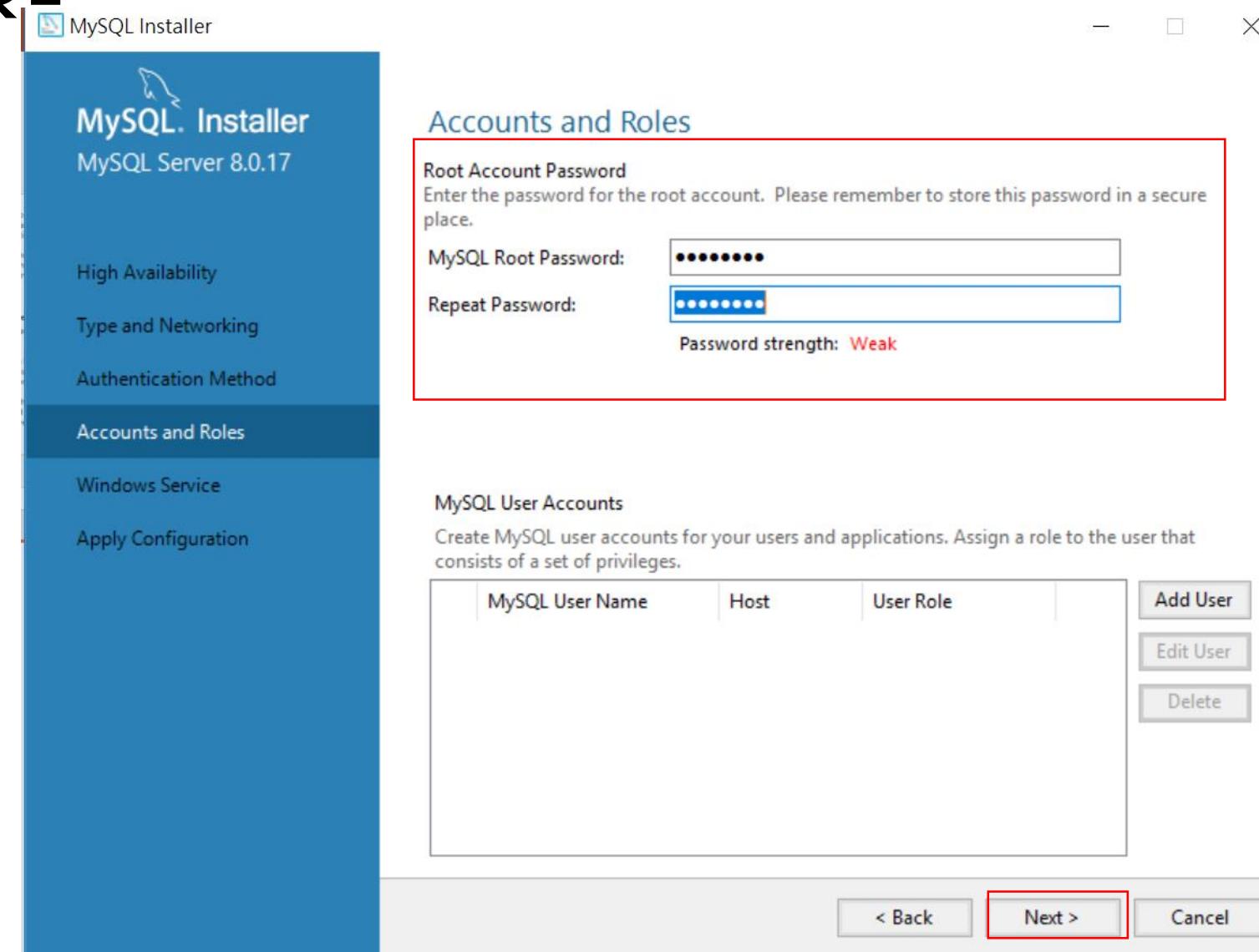


安裝MySQL

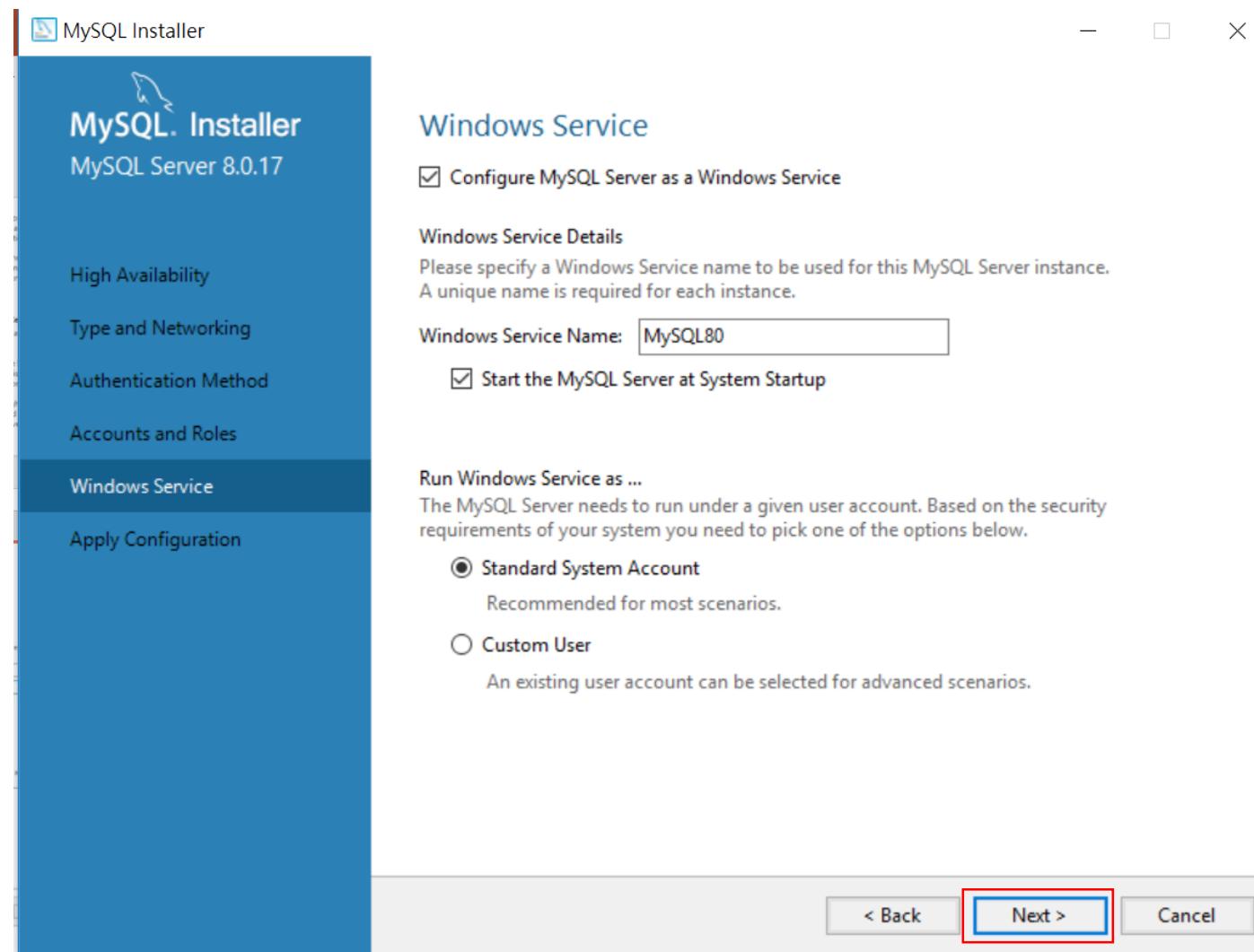


3-3: MySQL安裝

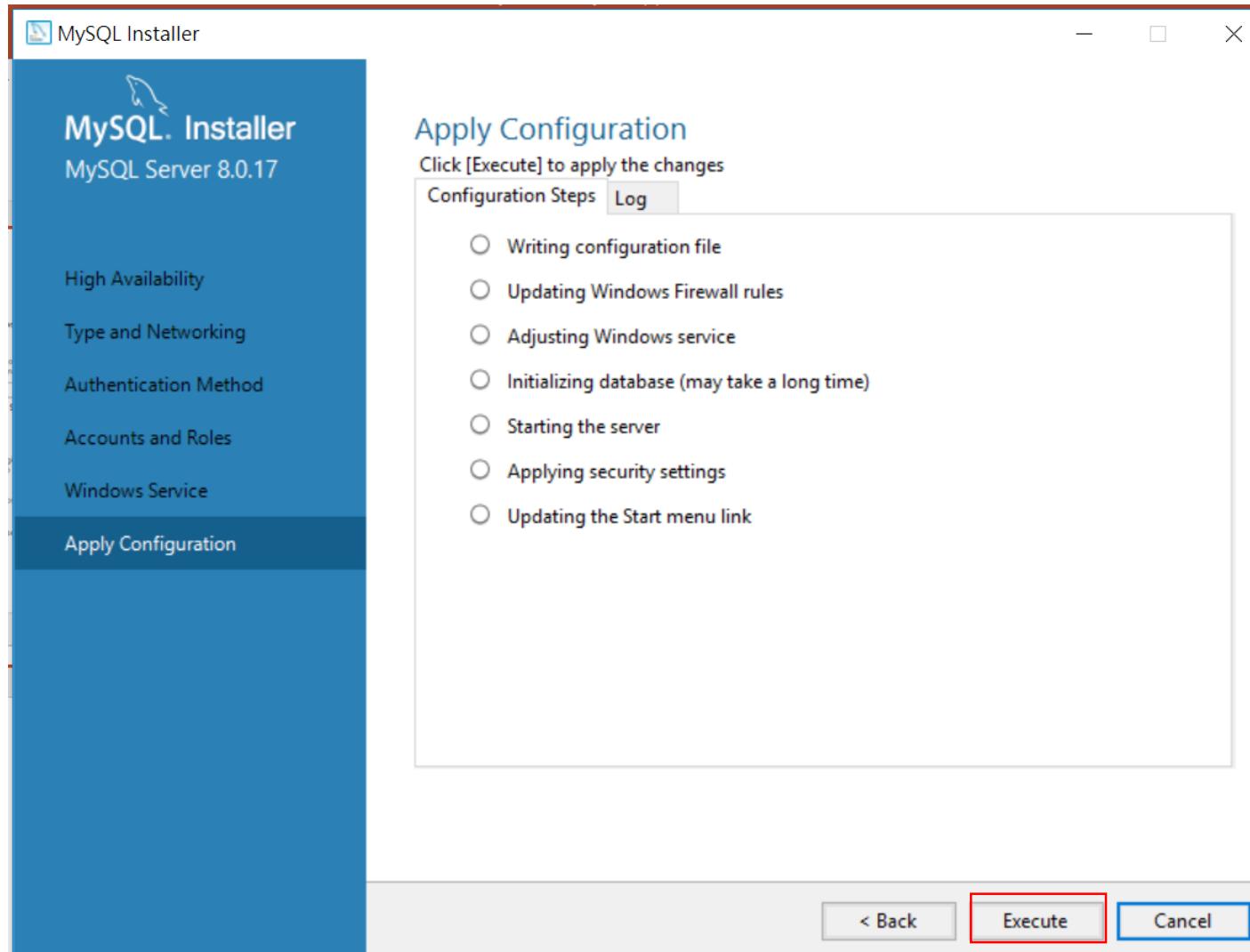
安裝MySQL



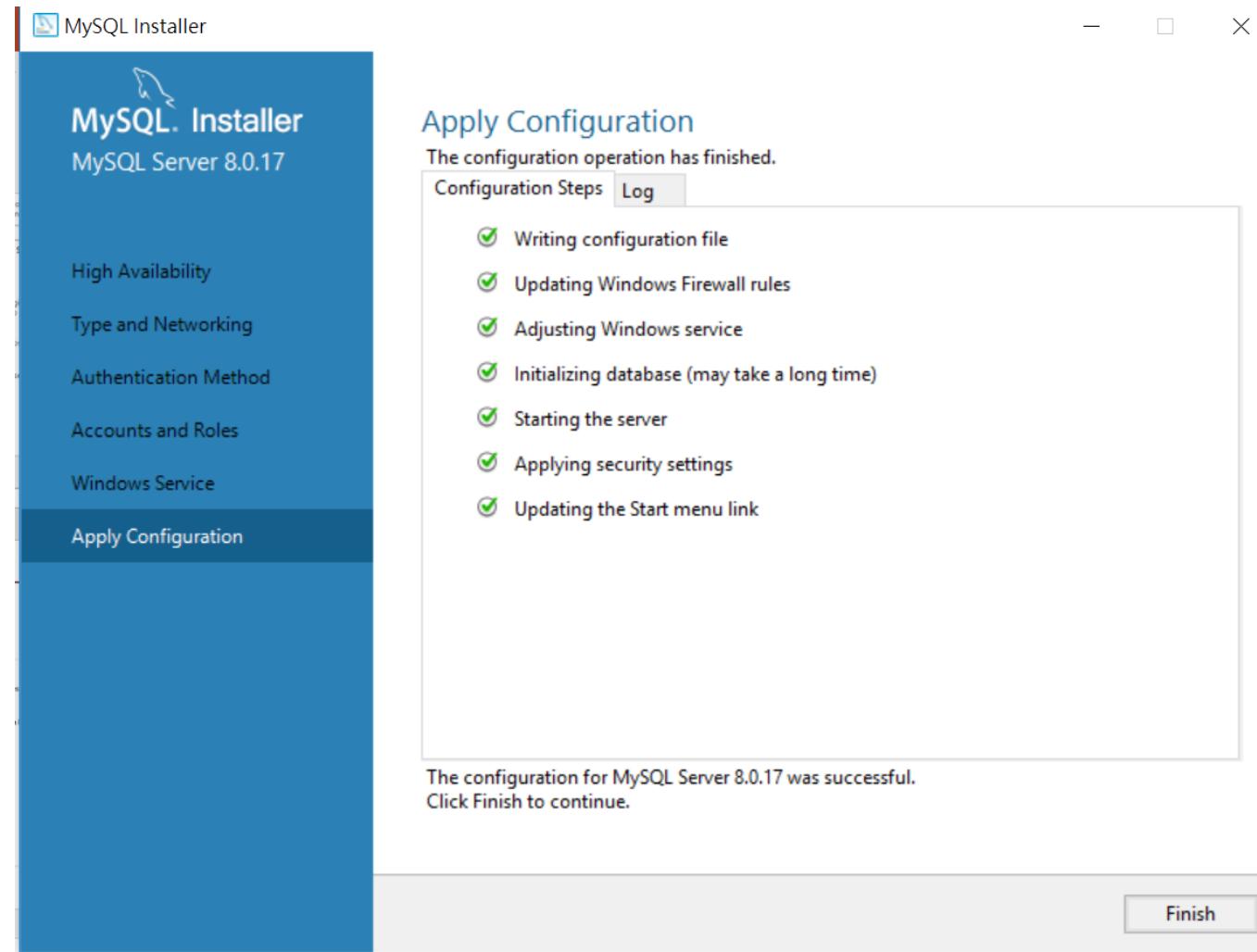
安裝MySQL



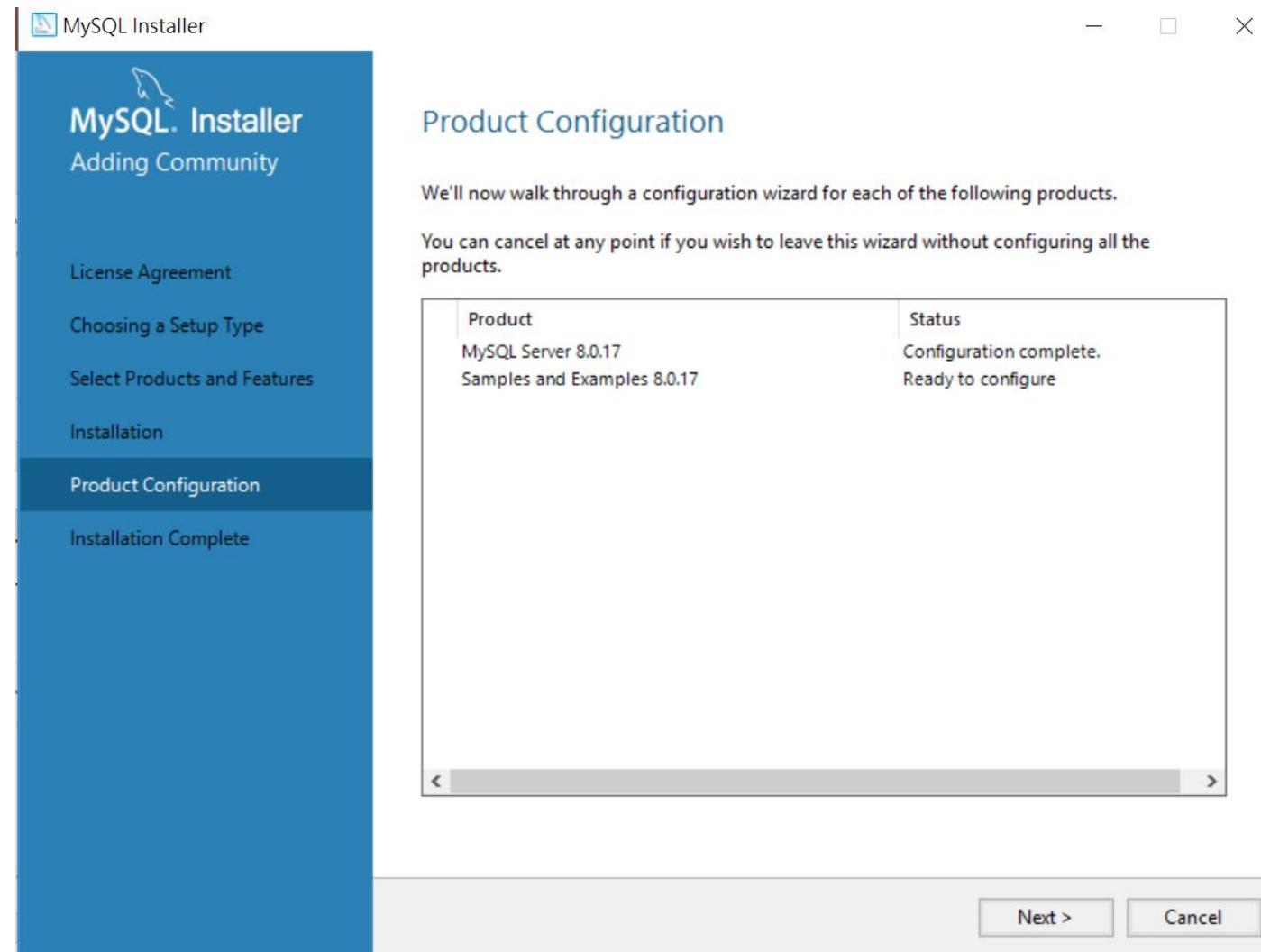
安裝MySQL



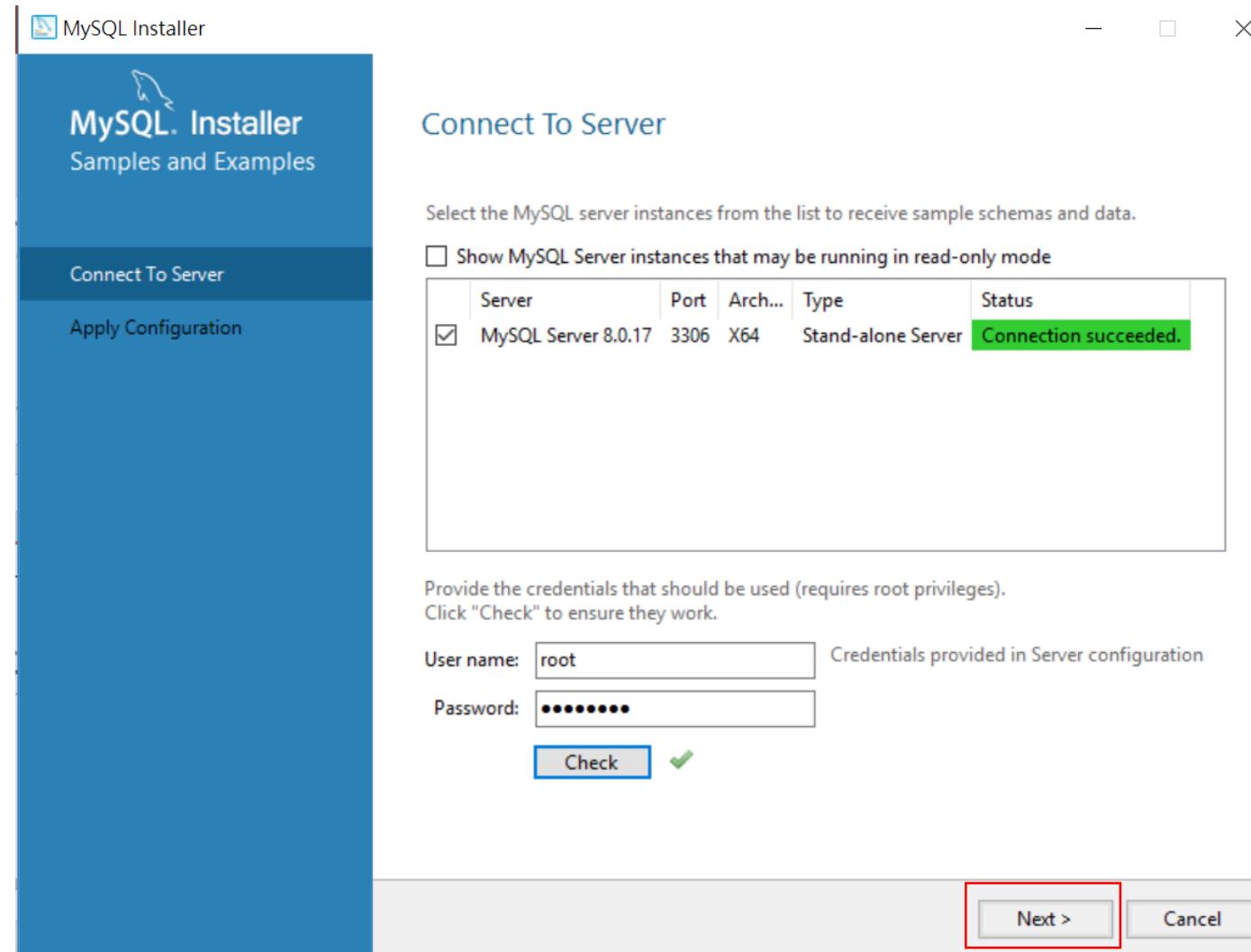
安裝MySQL



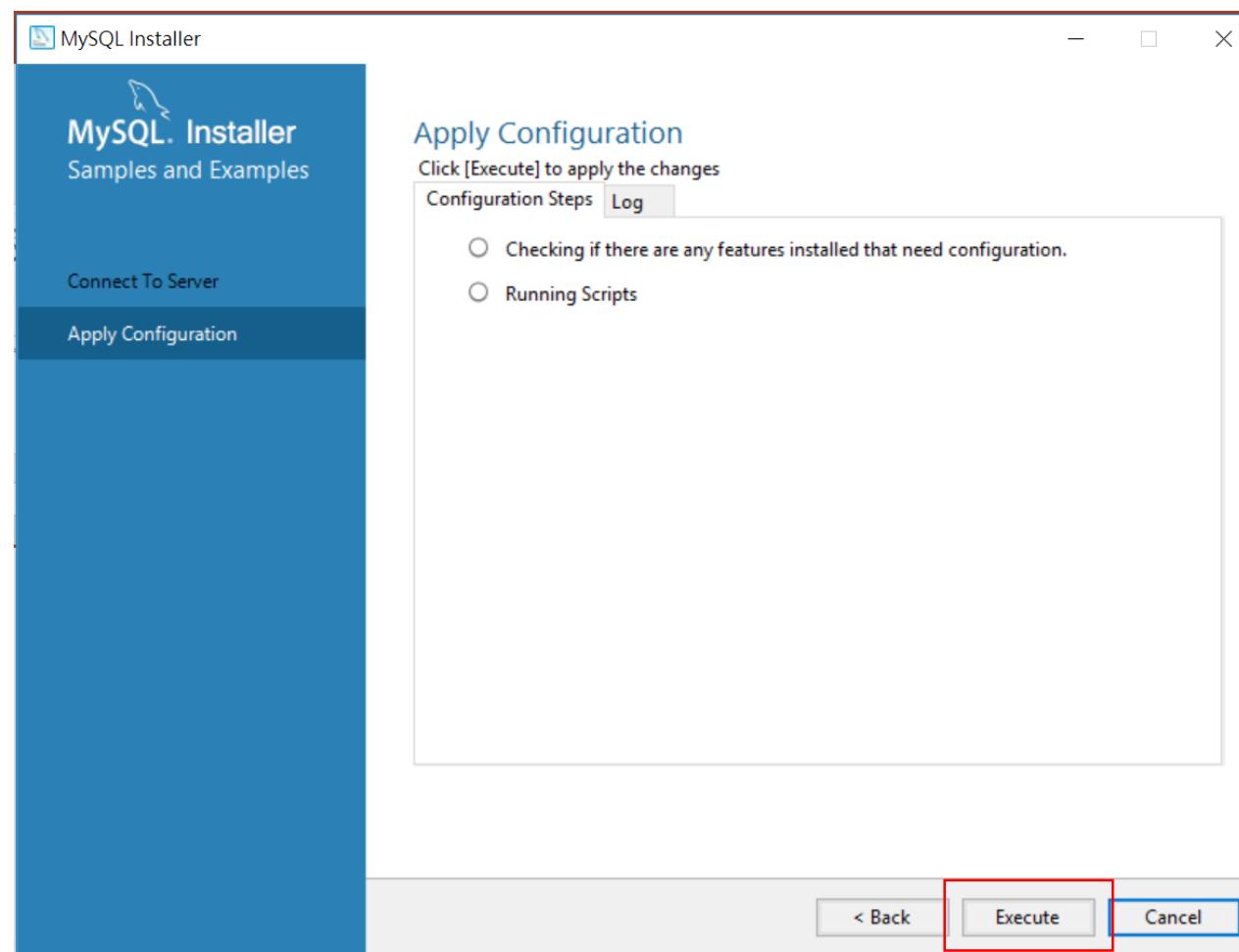
安裝MySQL



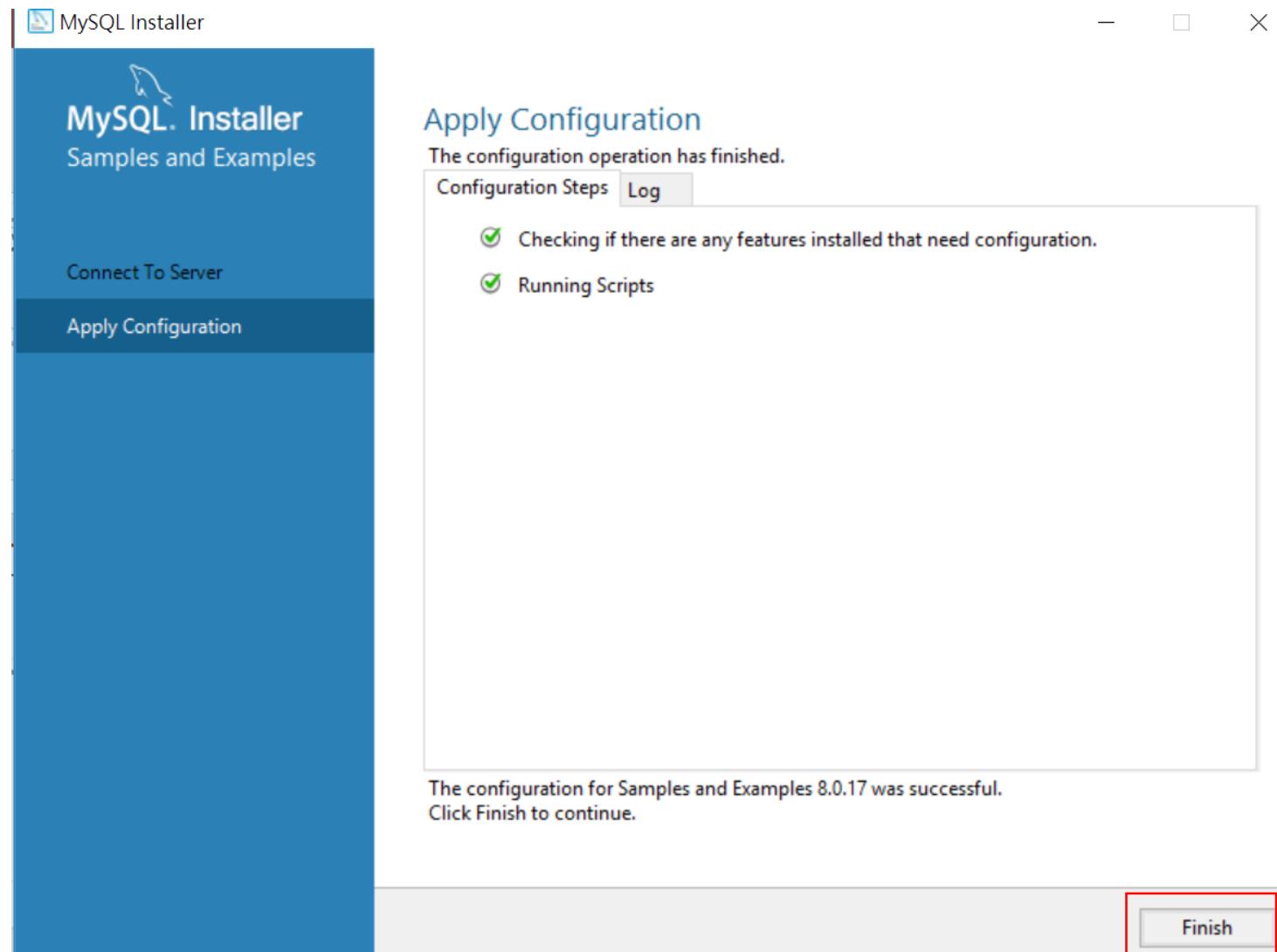
安裝MySQL



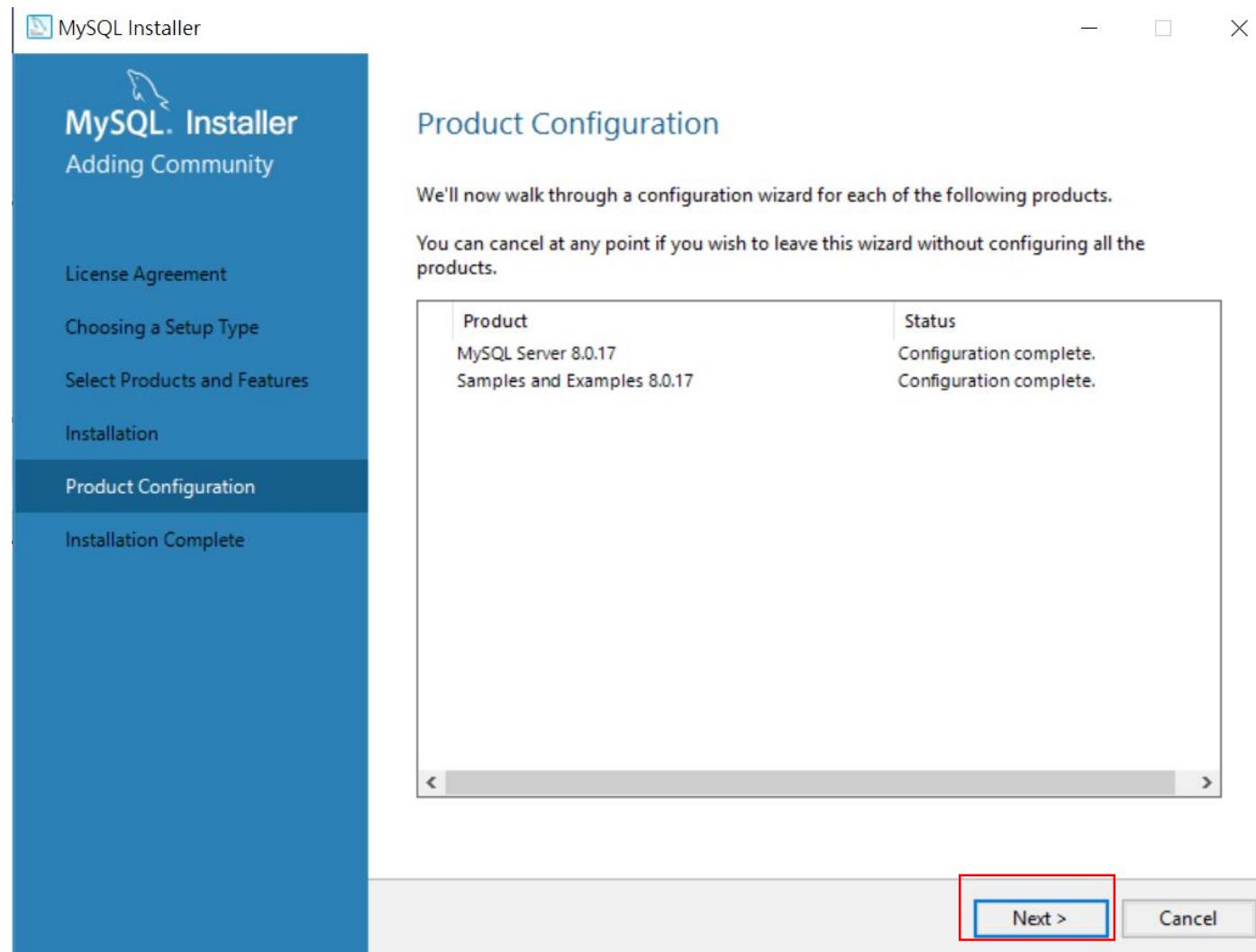
安裝MySQL



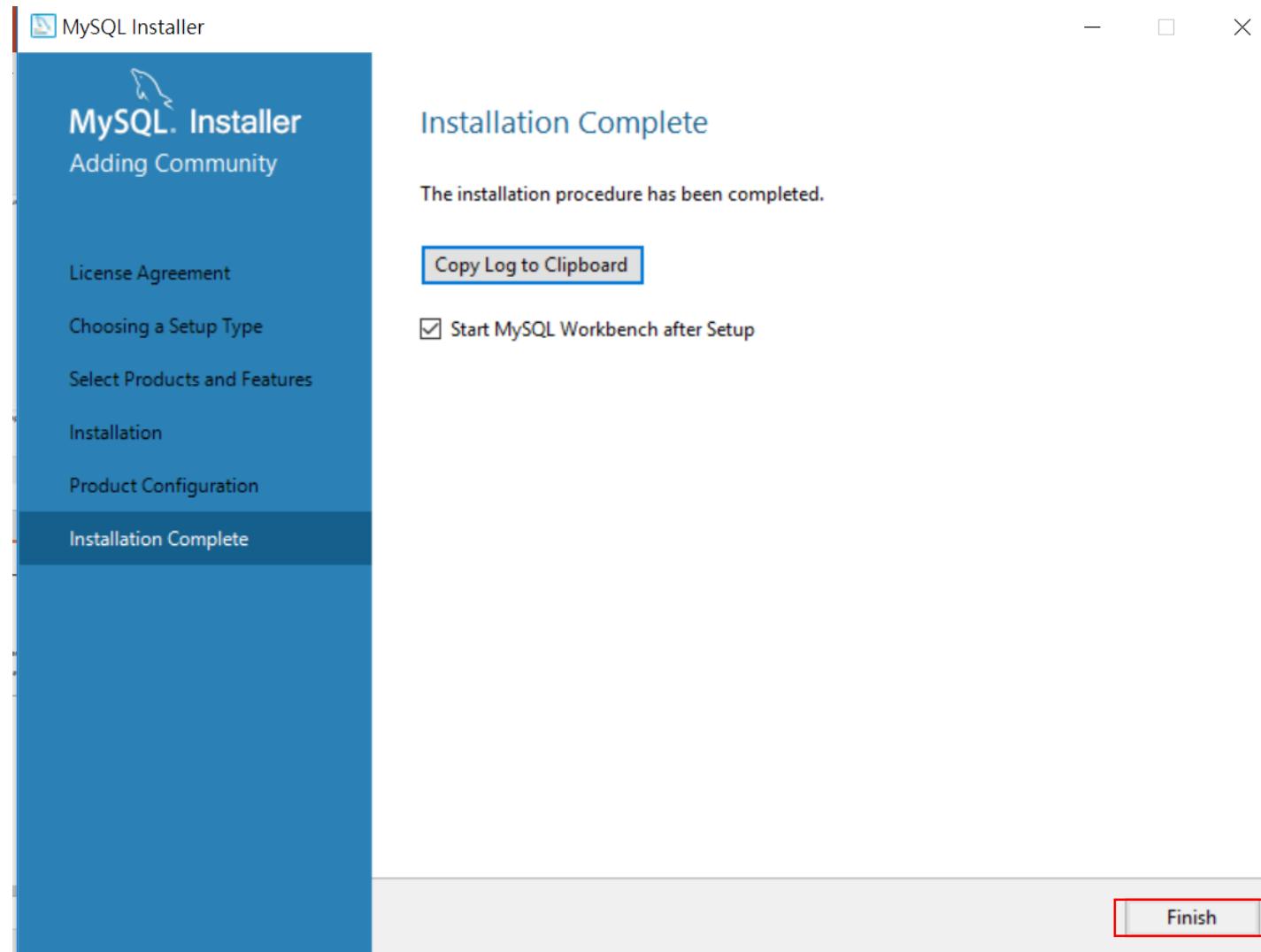
安裝MySQL



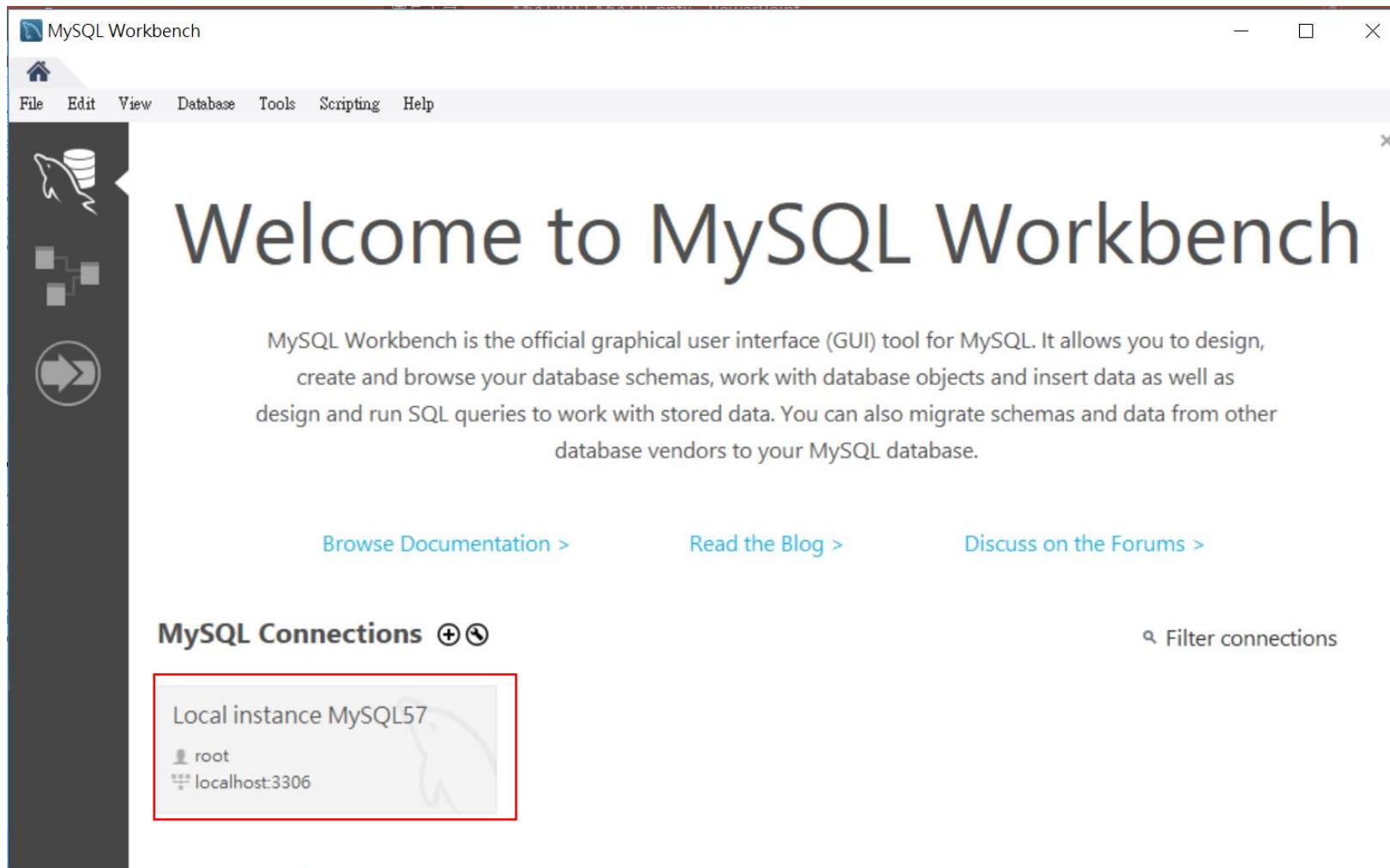
安裝MySQL



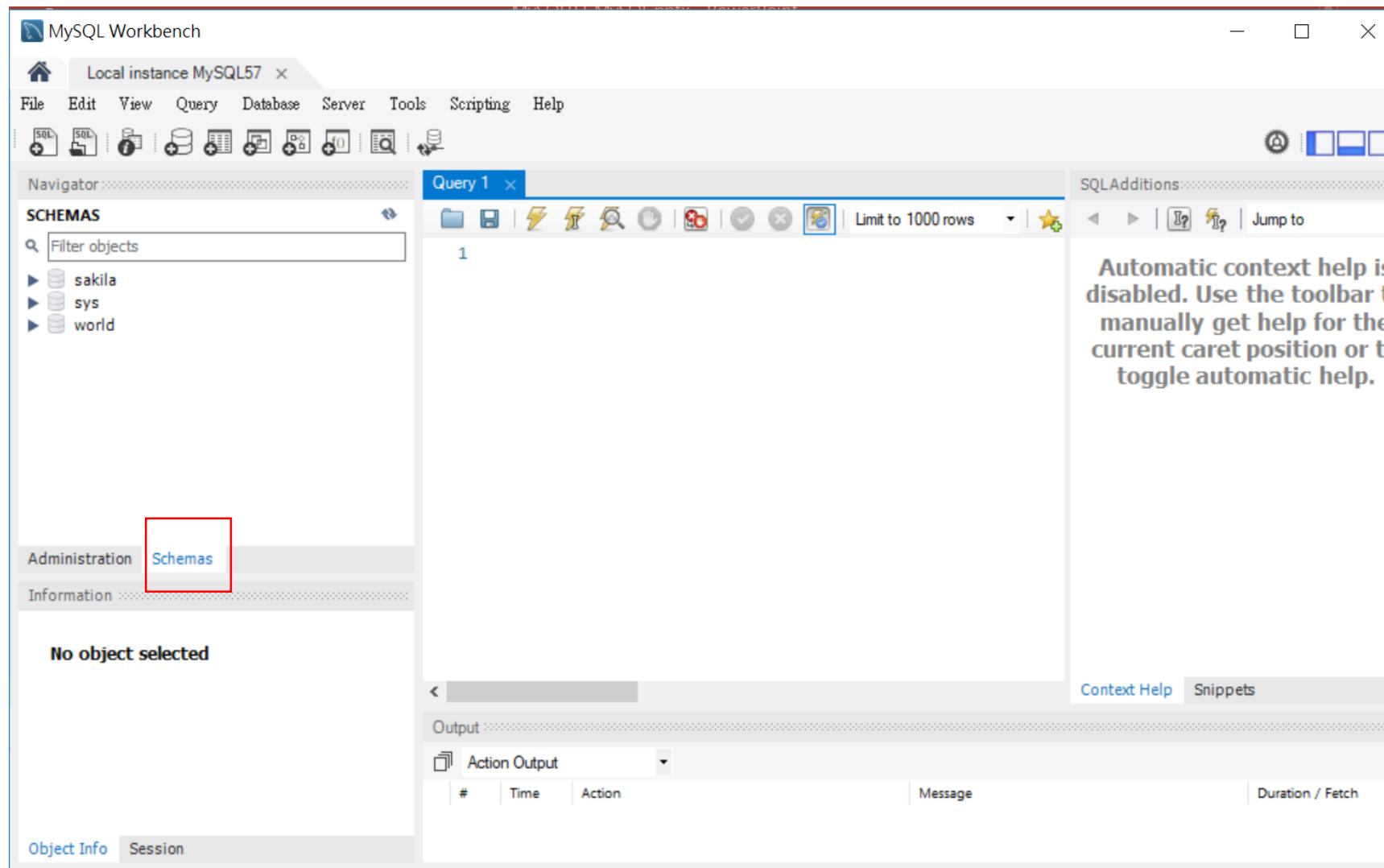
安裝MySQL



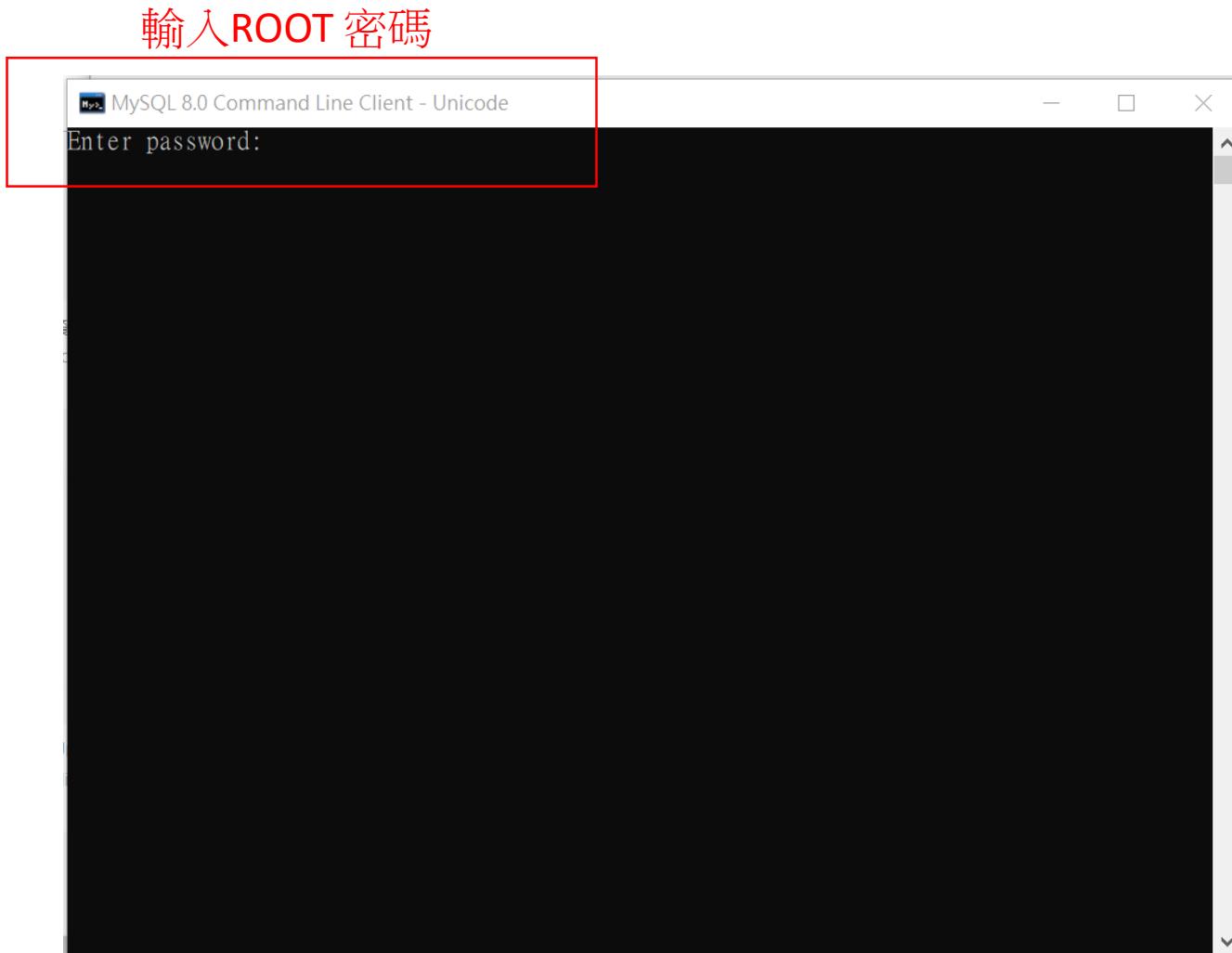
MySQL Workbench



MySQL Workbench

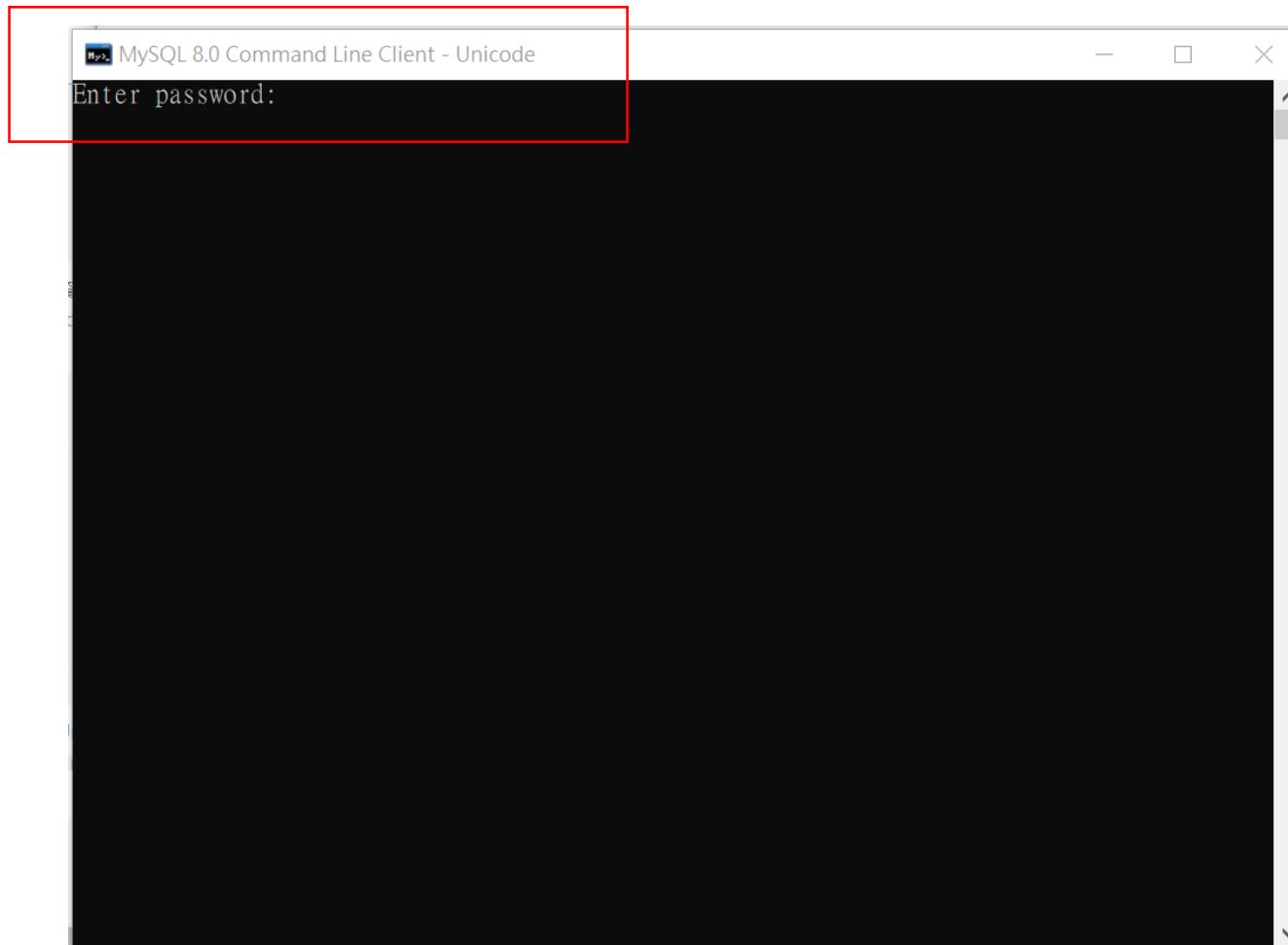


安MySQL 8.0 Command Line Client

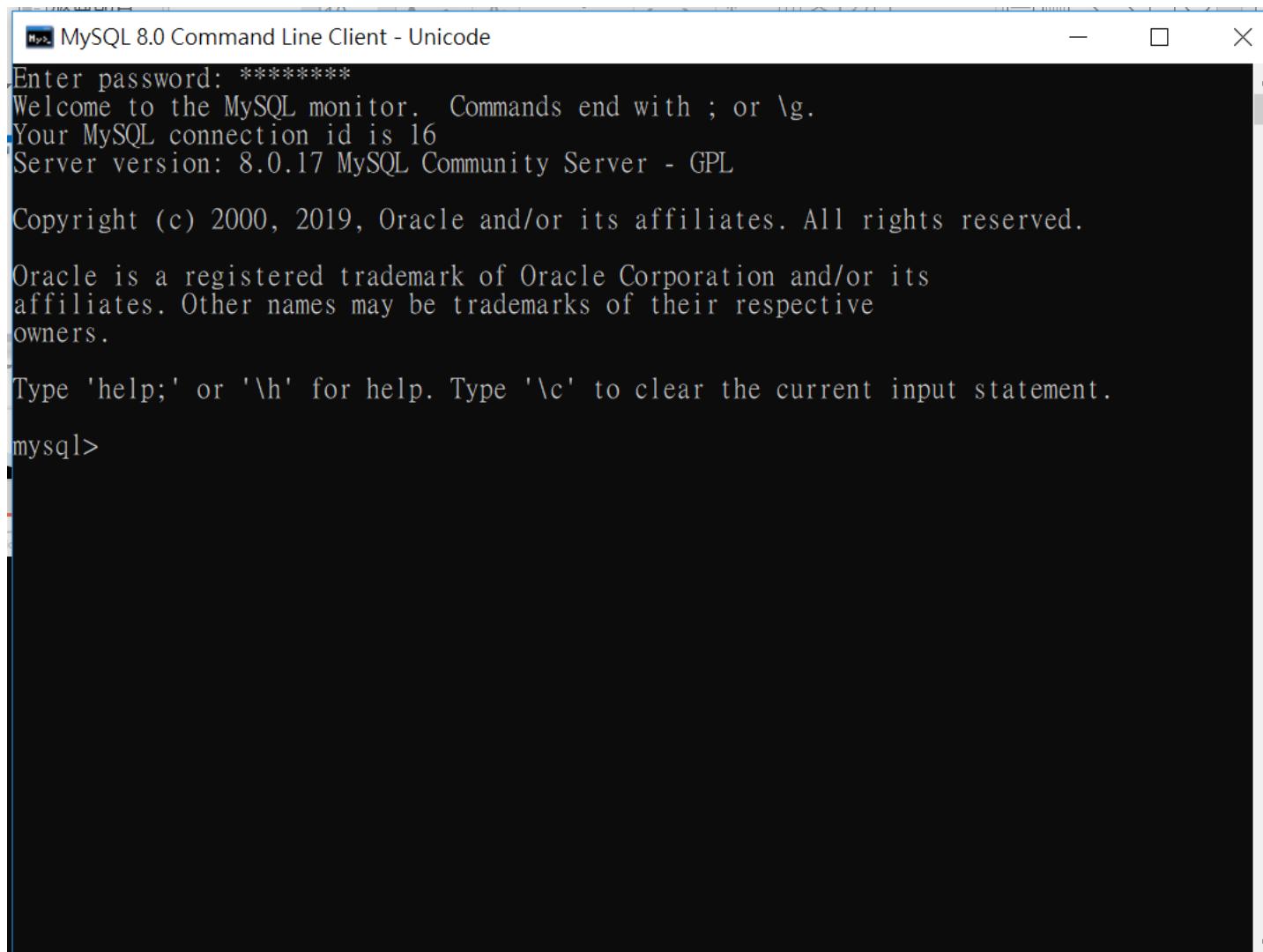


安MySQL 8.0 Command Line Client

輸入ROOT 密碼

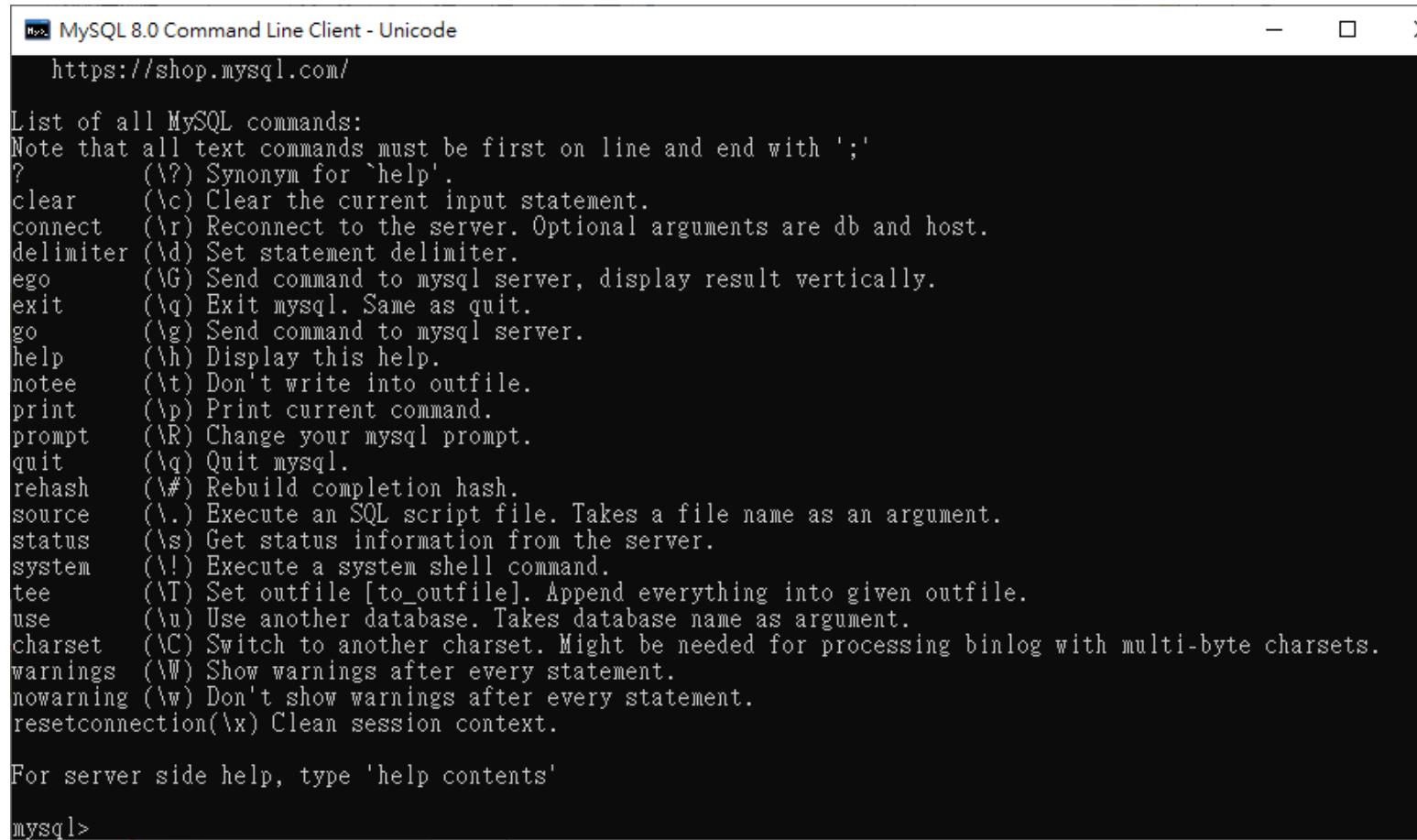


安MySQL 8.0 Command Line Client



請使用MySQL 8.0 Command Line Client

- 以root身分登入
- 執行 help; 顯示出MySQL的命令清單
- 並且輸入quit 登出



The screenshot shows a terminal window titled "MySQL 8.0 Command Line Client - Unicode". The URL "https://shop.mysql.com/" is visible in the address bar. The main content is a list of MySQL commands:

```
List of all MySQL commands:
Note that all text commands must be first on line and end with ';'
?      (\?) Synonym for `help'.
clear  (\c) Clear the current input statement.
connect (\r) Reconnect to the server. Optional arguments are db and host.
delimiter (\d) Set statement delimiter.
ego    (\G) Send command to mysql server, display result vertically.
exit   (\q) Exit mysql. Same as quit.
go     (\g) Send command to mysql server.
help   (\h) Display this help.
notee  (\t) Don't write into outfile.
print   (\p) Print current command.
prompt  (\R) Change your mysql prompt.
quit   (\q) Quit mysql.
rehash  (\#) Rebuild completion hash.
source  (\.) Execute an SQL script file. Takes a file name as an argument.
status  (\s) Get status information from the server.
system  (!!) Execute a system shell command.
tee    (\T) Set outfile [to_outfile]. Append everything into given outfile.
use    (\u) Use another database. Takes database name as argument.
charset (\C) Switch to another charset. Might be needed for processing binlog with multi-byte charsets.
warnings (\W) Show warnings after every statement.
nowarning (\w) Don't show warnings after every statement.
resetconnection(\x) Clean session context.

For server side help, type 'help contents'

mysql>
```

Module 4. 資料庫管理1

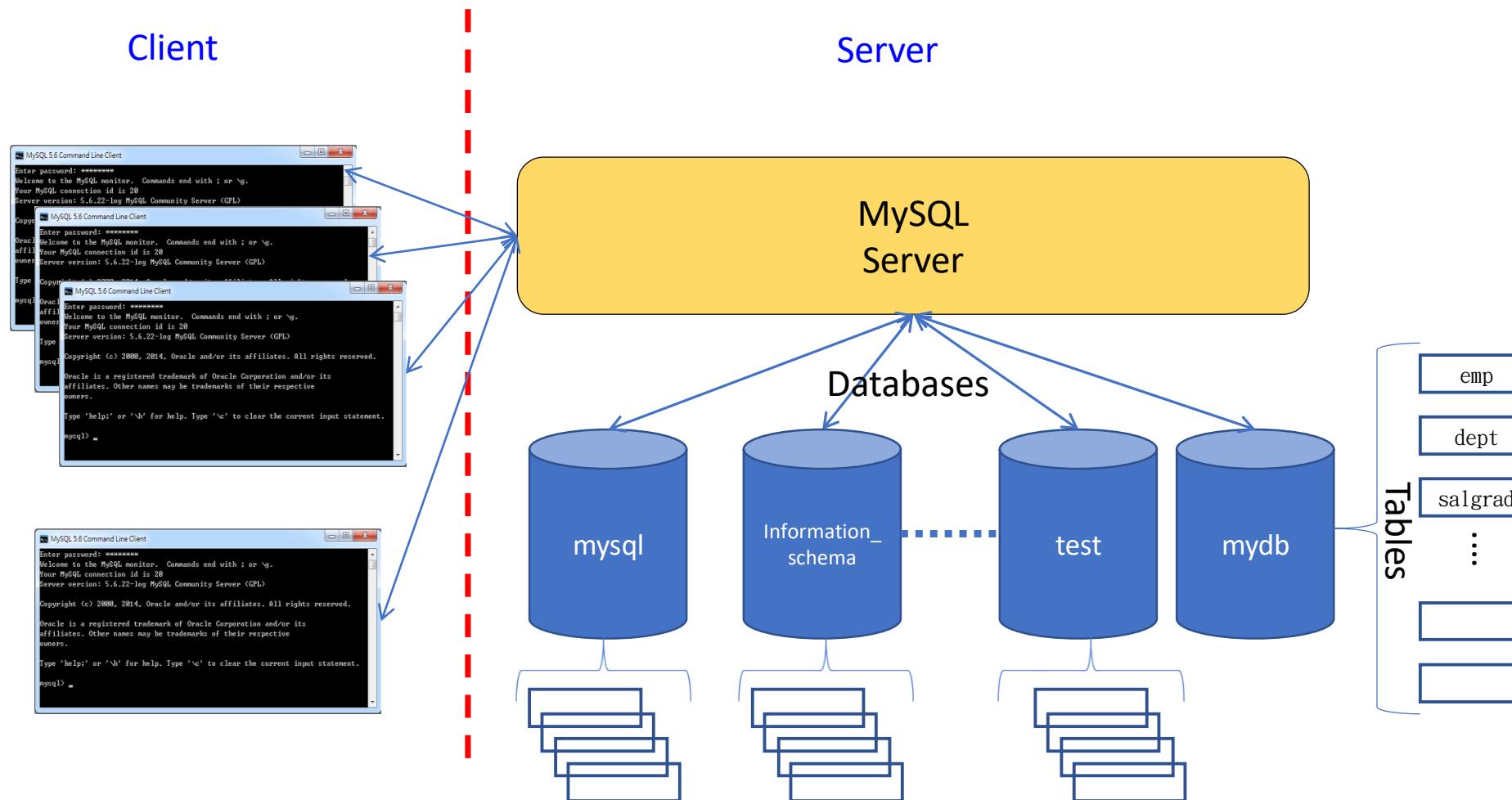
- 4-1: MySQL的預設資料庫**
- 4-2: 新增資料庫程序**
- 4-3: 字元集、Collation 、 SCHEMA**

資料庫物件(Database Objects)

- ▶ 資料庫物件是定義在資料庫中用來儲存資料或存取資料的物件
- ▶ 常見的資料庫物件：
 - 資料表(table)、資料限制條件(Constraints)
 - 視觀表(View)、索引(Index)
 - 使用者帳號(User Account)
 - 內儲程式(Stored Porgram) …等
- ▶ 資料庫物件的維護命令
 - 新建物件 : CREATE
 - 修改物件 : ALTER
 - 刪除物件 : DROP

4-1: MySQL的預設資料庫

MySQL 與 Command Line Client 的互動架構



使用 MySQL 命令建立範例資料庫

1. 使用 CREATE DATABASE 命令來建立一個新的資料庫

```
CREATE DATABASE [IF NOT EXISTS] databaseName;
```

```
CREATE DATABASE mydb;
```

自行設定資料
庫名稱

2. 使用 SOURCE 命令來執行一個SQL命令檔

```
SOURCE FileName;
```

絕對路徑檔名包
含路徑檔案名稱

```
source c:\mysql_demo\ld.sql
```

使用 MySQL 命令顯示與使用資料庫

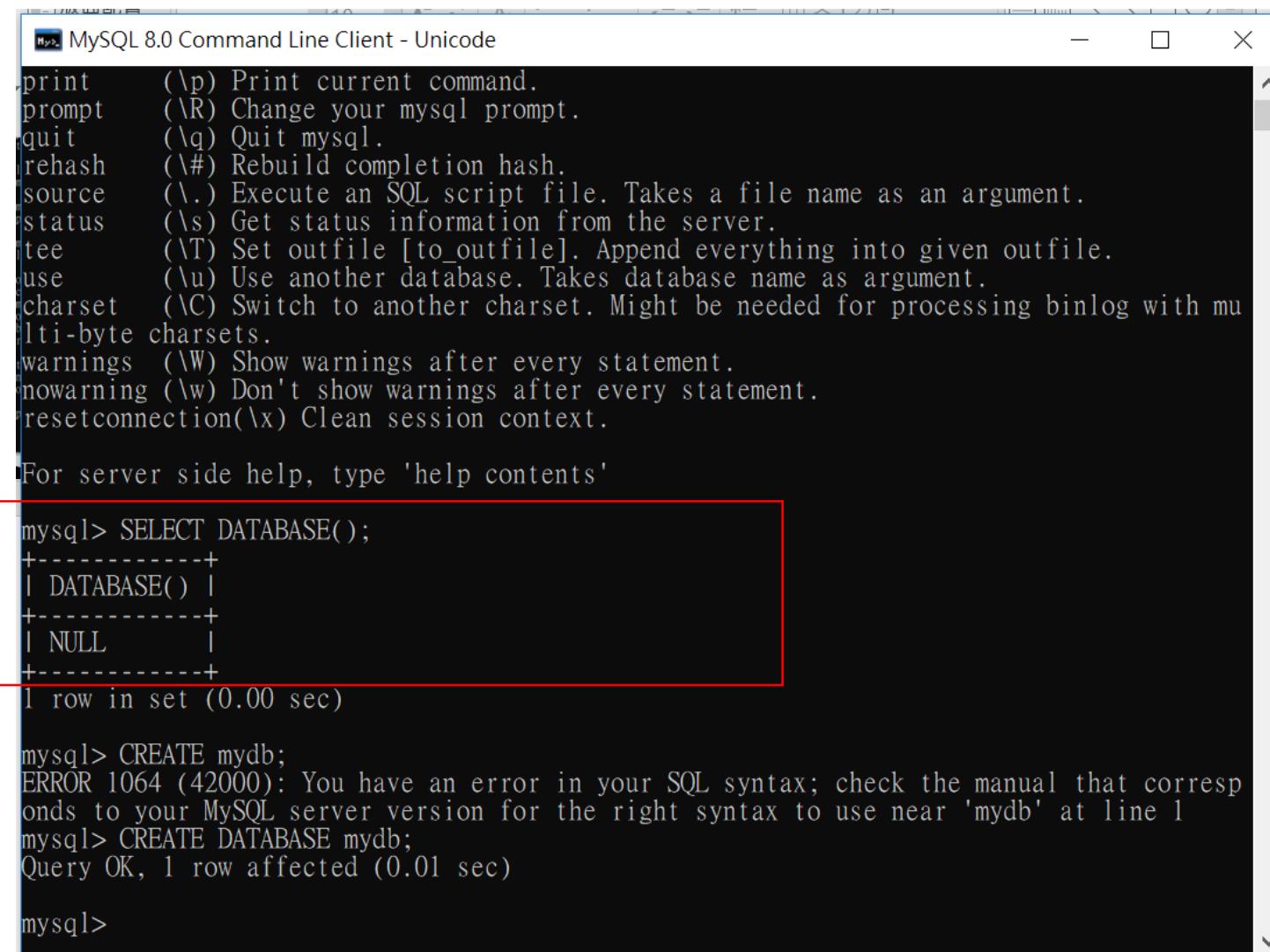
▶ SHOW DATABASES

- 列出目前伺服器現存的所有資料庫(databases)
- mysql> show databases;

▶ USE [database]

- 設定預設存取的資料庫(default database)
- mysql> use mydb;

使用 MySQL 命令查詢資料庫狀況



The screenshot shows a terminal window titled "MySQL 8.0 Command Line Client - Unicode". It displays the following text:

```
print      (\p) Print current command.
prompt     (\R) Change your mysql prompt.
quit       (\q) Quit mysql.
rehash     (\#) Rebuild completion hash.
source     (\.) Execute an SQL script file. Takes a file name as an argument.
status     (\s) Get status information from the server.
tee        (\T) Set outfile [to_outfile]. Append everything into given outfile.
use        (\u) Use another database. Takes database name as argument.
charset    (\C) Switch to another charset. Might be needed for processing binlog with mu
lti-byte charsets.
warnings   (\W) Show warnings after every statement.
nowarning  (\w) Don't show warnings after every statement.
resetconnection(\x) Clean session context.

For server side help, type 'help contents'

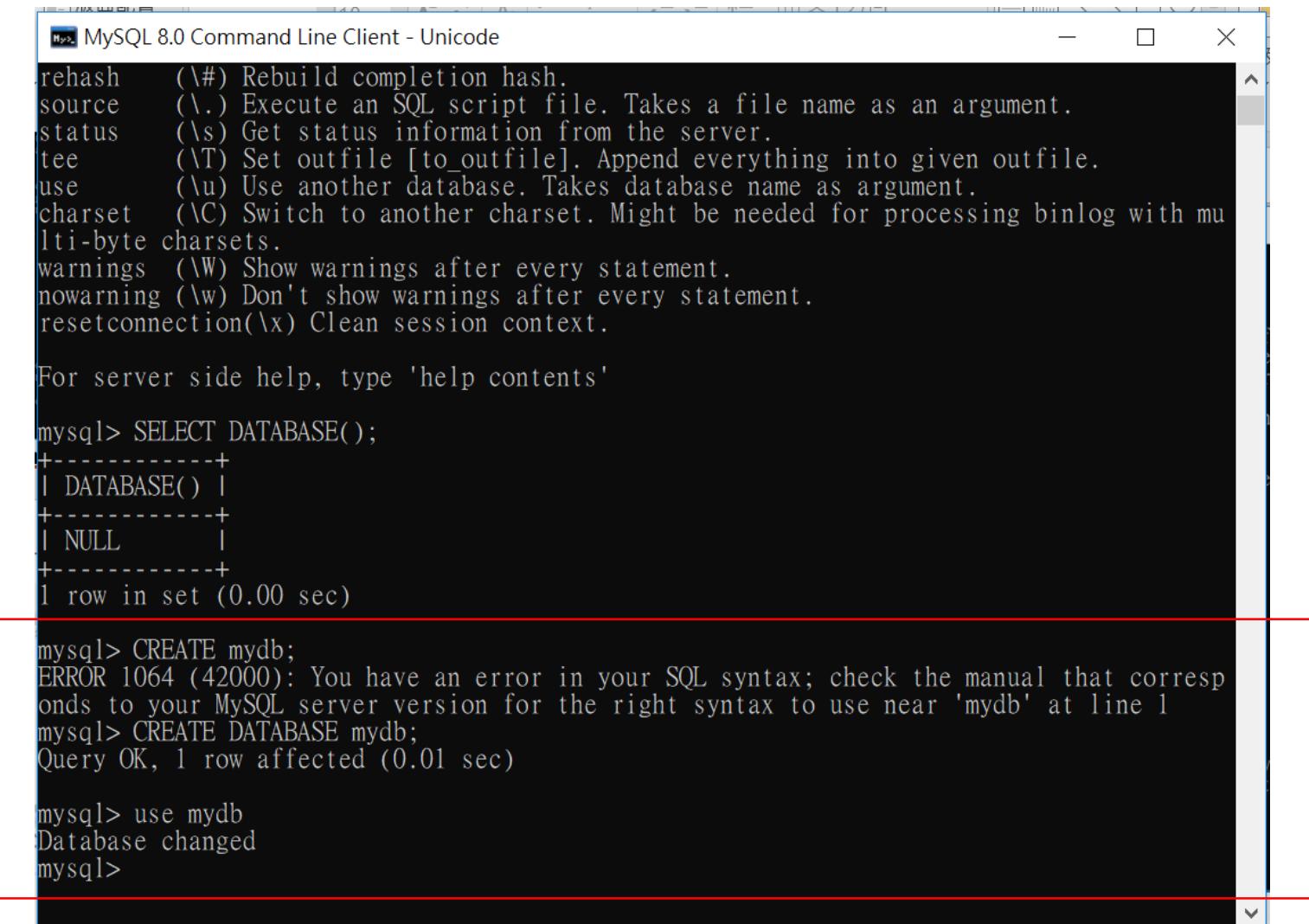
mysql> SELECT DATABASE();
+-----+
| DATABASE() |
+-----+
| NULL      |
+-----+
1 row in set (0.00 sec)

mysql> CREATE mydb;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'mydb' at line 1
mysql> CREATE DATABASE mydb;
Query OK, 1 row affected (0.01 sec)

mysql>
```

A red box highlights the output of the `SELECT DATABASE();` query, which shows a single row with a value of `NULL`.

使用 MySQL 命令建立與選定範例資料庫



The screenshot shows a terminal window titled "MySQL 8.0 Command Line Client - Unicode". It displays the following MySQL session:

```
rehash      (\#) Rebuild completion hash.
source      (\.) Execute an SQL script file. Takes a file name as an argument.
status      (\s) Get status information from the server.
tee         (\T) Set outfile [to_outfile]. Append everything into given outfile.
use         (\u) Use another database. Takes database name as argument.
charset     (\C) Switch to another charset. Might be needed for processing binlog with mu
lti-byte charsets.
warnings   (\W) Show warnings after every statement.
nowarning  (\w) Don't show warnings after every statement.
resetconnection(\x) Clean session context.

For server side help, type 'help contents'

mysql> SELECT DATABASE();
+-----+
| DATABASE() |
+-----+
| NULL      |
+-----+
1 row in set (0.00 sec)

mysql> CREATE mydb;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'mydb' at line 1
mysql> CREATE DATABASE mydb;
Query OK, 1 row affected (0.01 sec)

mysql> use mydb
Database changed
mysql>
```

A red rectangular box highlights the error message and the successful creation of the database.

使用MySQL命令 TEE/NOTEE – 將執行結果留存

- MySQL Command Line Client

```
TEE <FileName>
...
...
NOTE
```

- 存成文字檔

The screenshot shows a MySQL command-line interface window titled "MySQL 5.6 Command Line Client". The session starts with the command `tee g:\aa`, which is highlighted with a red arrow and labeled "開始留存" (Start Saving). This command directs the output to a file named "aa" located at "g:\". The next command, `select * from dept;`, retrieves data from the "dept" table and displays it in a tabular format:

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

There are 4 rows in the set, completed in 0.01 seconds.

Following this, the command `show databases;` is run, listing the available databases:

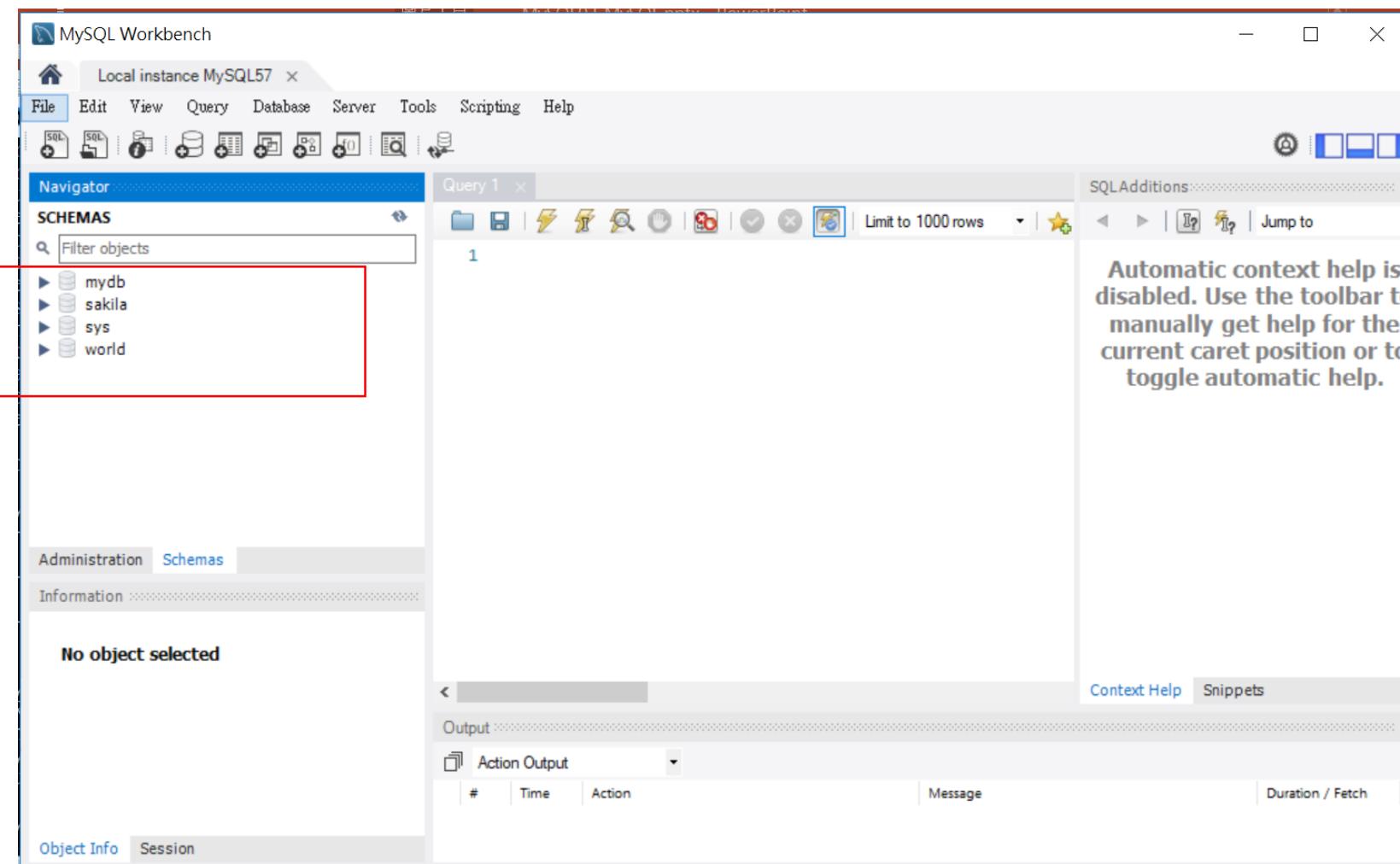
Database
information_schema
exp
mysql
performance_schema
sakila
sports
test
world

There are 8 rows in the set, completed in 0.02 seconds.

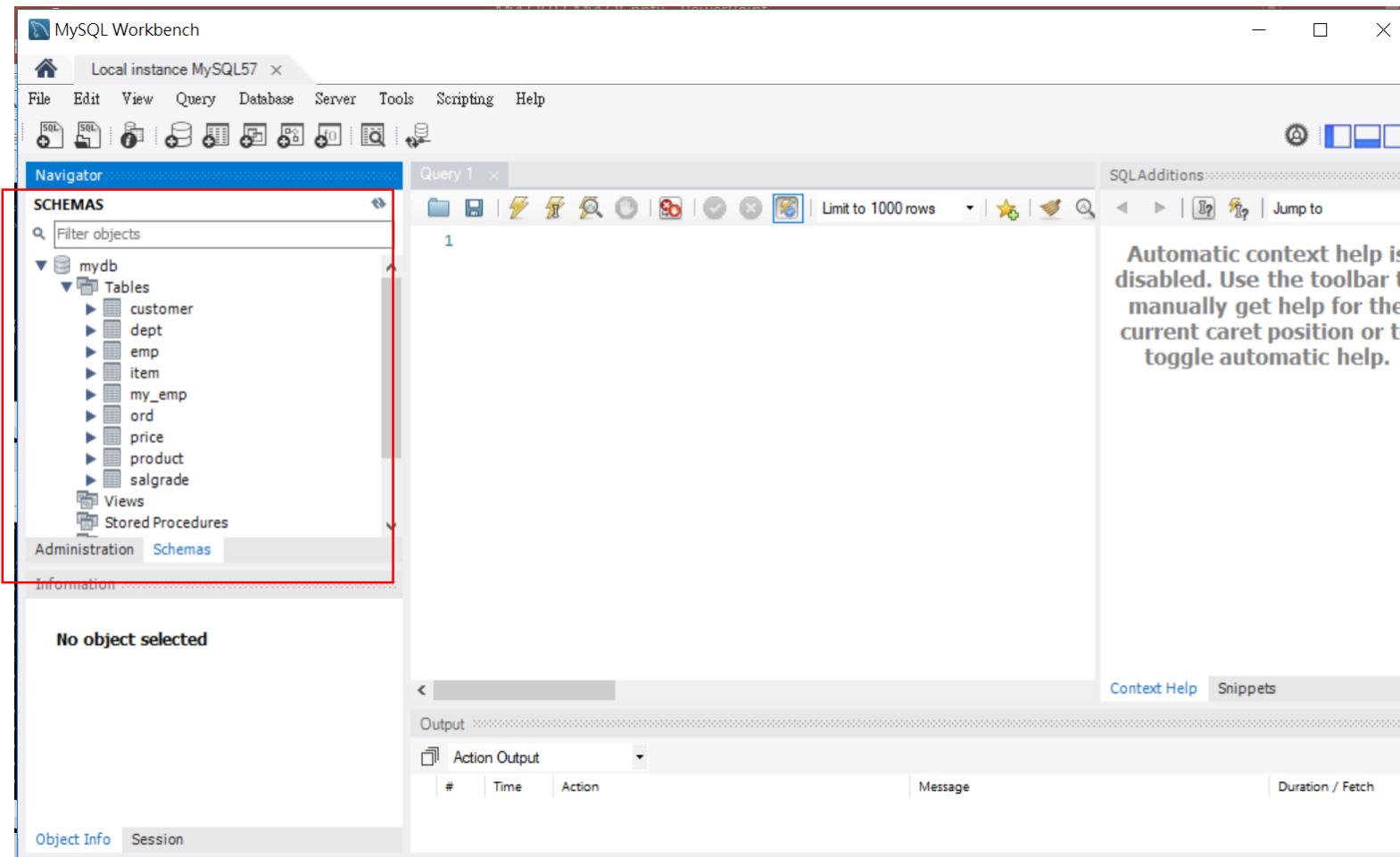
Finally, the command `notee` is run, resulting in the message "Outfile disabled.", which is highlighted with a red arrow and labeled "結束留存" (End Saving).

4-2: 新增資料庫程序

使用 MySQL Workbench 列出資料庫

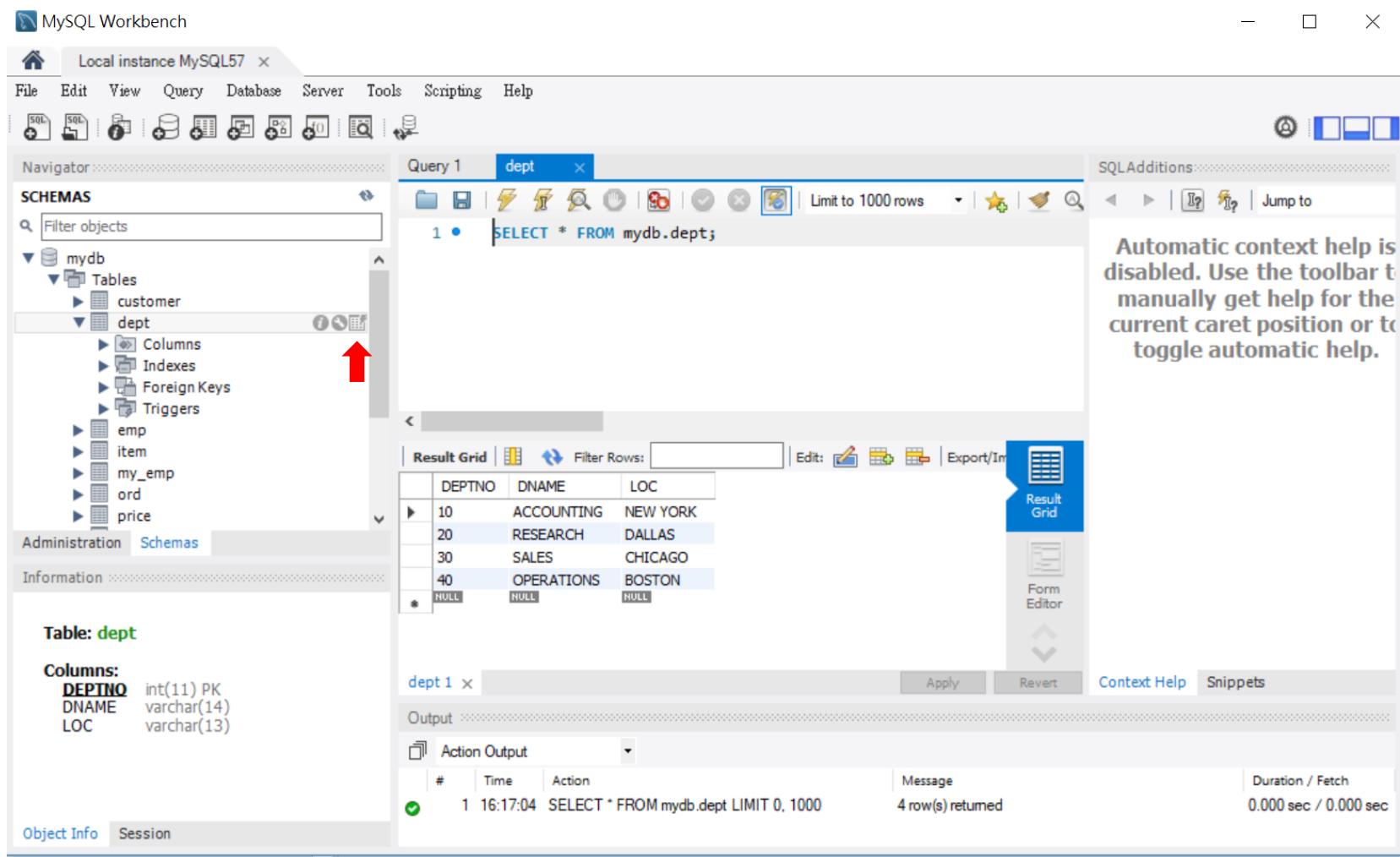


使用 MySQL Workbench 確認資料庫內容



4-2: 新增資料庫程序

使用 MySQL Workbench 確認資料庫內容



4-2: 新增資料庫程序

範例資料庫主要資料表

EMP(員工資料)

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		17-NOV-81	5000		10
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30

DEPTNO	DNAME	LOC	7566	03-DEC-81	3000		20
10	ACCOUNTING	NEW YORK	7902	17-DEC-80	800		20
20	RESEARCH	DALLAS	7566	09-DEC-82	3000		20
30	SALES	CHICAGO	7788	12-JAN-83	1100		20
40	OPERATIONS	BOSTON	7782	23-JAN-83			

DEPT(部門資料)

GRADE	LOSAL	HISAL
1	700	1200
2	1201	1400
3	1401	2000
4	2001	3000
5	3001	9999

SALGRADE(薪資等級對照資料)

MySQL命令 顯示資料表

- SHOW TABLES
 - 列出預設資料庫中所有資料表
 - mysql> show tables;
- DESC[RIBE] table
 - 列出指定資料表的欄位資訊
- Select * from dept;
 - 列出資料表資料

```
mysql> show tables;
+-----+
| Tables_in_sample |
+-----+
| bonus
| customer
| dept
| dummy
```

```
mysql> describe dept;
+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default |
+-----+-----+-----+-----+
| DEPTNO | smallint(6) | PRI | 0 |
| DNAME  | varchar(14) | YES | NULL |
| LOC    | varchar(13) | YES | NULL |
+-----+-----+-----+-----+
4 rows in set (0.04 sec)
```

```
mysql>
```

- **字元集(Char Set)**
 - 定義了字元與字元的編碼
 - SHOW CHARSET;
- **定序(Collation)**
 - 定義了字元的比較規則
 - 字尾_ci，不分大小寫(Case Insensitive)
 - 字尾_cs，區分大小寫(Case Sensitive)

```
mysql> SHOW CHARSET LIKE 'utf8%';
+-----+-----+-----+-----+
| Charset | Description | Default collation | Maxlen |
+-----+-----+-----+-----+
| utf8    | UTF-8 Unicode | utf8_general_ci   |      3  |
| utf8mb4 | UTF-8 Unicode | utf8mb4_0900_ai_ci |      4  |
+-----+-----+-----+-----+
```

結構(Schema)

- Schema(資料庫結構) 等同於資料庫(database)

- 建立一個 my_db 的資料庫

CREATE SCHEMA my_db;

等同於

CREATE DATABASE my_db;

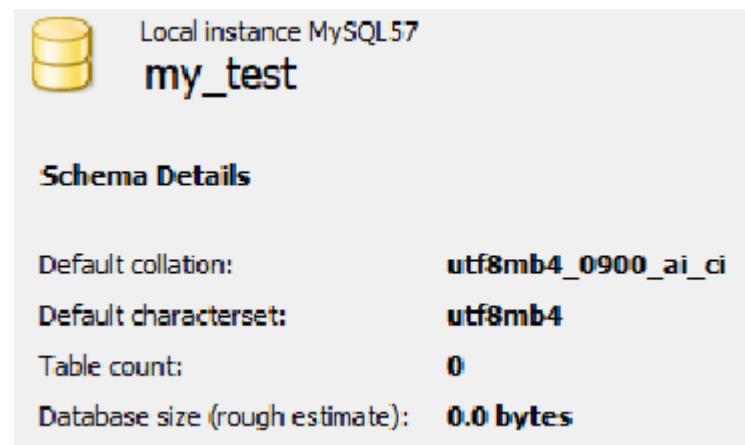
Module 5. 資料庫管理2

- 5-1: LAB:新增資料庫**
- 5-2: 修改資料庫與刪除資料庫**
- 5-3: 切換資料庫與顯示所有資料庫**

5-1: LAB:新增資料庫

- 新增資料庫

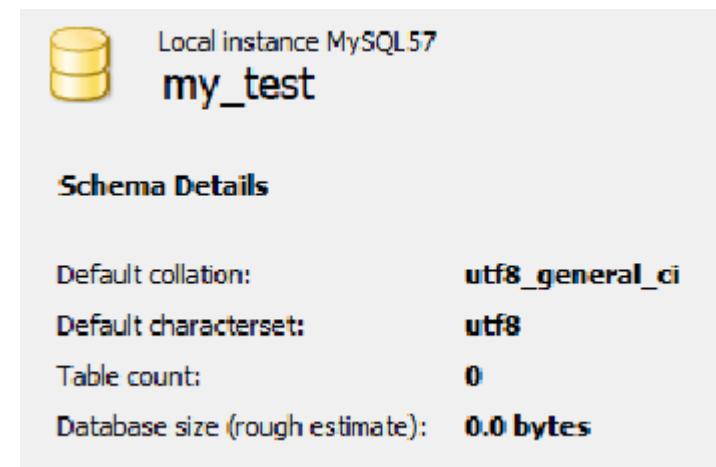
```
CREATE DATABASE my_test;
```



5-2: 修改資料庫與刪除資料庫

- 修改資料庫

```
ALTER DATABASE my_test
CHARACTER SET utf8
COLLATE utf8_general_ci ;
```



- 刪除資料庫

```
DROP DATABASE my_test;
```

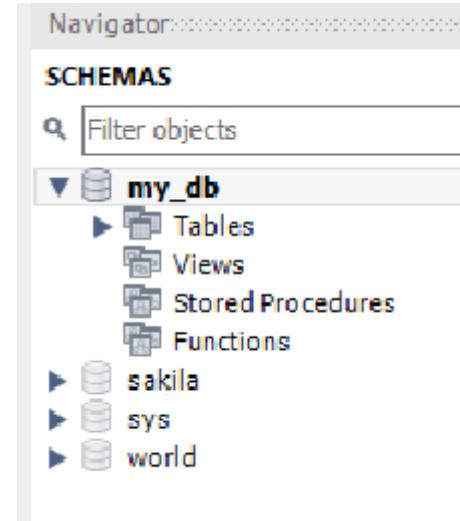
5-3: 切換資料庫與顯示所有資料庫

- 切換資料庫

`USE my_db;`

- 顯示所有資料庫

`SHOW DATABASES;`



Database
information_schema
my_db
mysql
performance_schema
sakila
sys
world

Module 6.

資料表管理基礎

6-1: SQL Statements撰寫規則

6-2: Naming Rule

6-3: Storage engine

撰寫SQL命令

- 指令**不分大小寫**
- 敘述**可以編寫成多行**
- 習慣上每個子句以**一行表示**
- 關鍵字(Keywords)不可以分割成多行
- 以**分號(;)作為敘述的結束符號**並執行該敘述
- 使用縮排來提升敘述的可讀性

SQL 註解方式

This comment continues to the end of line

-- This comment continues to the end of line

/* this is an in-line comment */

資料查詢 – SELECT 敘述

指令	表達式	功用
SELECT	<select list>	給定查詢的資料項目 Defines which columns to return
FROM	<table source>	給定資料來源 Defines table(s) to query
WHERE	<search condition>	給定查詢/過濾資料條件 Filters rows using a predicate
GROUP BY	<group by list>	資料分組設定 Arranges rows by groups
HAVING	<search condition>	給定分組資料查詢/過濾條件 Filters groups using a predicate
ORDER BY	<order by list>	給定查詢結果排序方式 Sorts the output

SELECT - 查詢資料的指令

▶ 語法

```
SELECT * | { [DISTINCT] column|expression [alias], ... }  
FROM table;
```

▶ 子句

- SELECT 指定檢視的資料項
 - 欄位(Column name)、常值(Literal)、運算式(expression)、函數(Function)
- FROM 指定資料的來源
 - 資料表(Table)、虛擬資料表(Virtual Table)、視觀表(View)

▶ 敘述

- 由二個以上子句所組成
- 資料項間以逗號(,)來區隔
- 以分號作為結束的符號

```
SELECT ename, job, sal  
FROM emp;
```

選取資料表中的所有(欄位)資料

```
SELECT *
FROM TableName;
```

	說明
SELECT	選取資料項(欄位)
*	資料表中所有的欄位
FROM	指定查詢的資料來源(資料表)
TableName	資料表名稱

```
mysql> select * from dept;
+-----+-----+-----+
| DEPTNO | DNAME        | LOC      |
+-----+-----+-----+
|      10 | ACCOUNTING   | NEW YORK |
|      20 | RESEARCH     | DALLAS   |
|      30 | SALES        | CHICAGO  |
|      40 | OPERATIONS   | BOSTON   |
+-----+-----+-----+
```

選取指定欄位資料

```
SELECT column, ...
FROM   TableName;
```

	說明
SELECT	選取資料項(欄位)
ColumnName	資料表中欄位名稱
FROM	指定查詢的資料來源(資料表)
TableName	資料表名稱

```
SELECT empno, ename, sal
FROM  emp;
```

欄位表頭 (Column Heading)

mysql> select deptno, DNAME from dept;	
deptno	DNAME
10	ACCOUNTING
20	RESEARCH
30	SALES
40	OPERATIONS

- ▶ 欄位表頭顯示的方式
 - Column 大寫 則以大寫表示
 - column 小寫 則以小寫表示
 - * 以 DESC 命令所見為準

SQL 一般命名規則

- 保證名字獨一無二且不是關鍵字(保留字)。
- 保證名字長度不超過 30 個字節。
- 名字要以字母開頭，不能以下底線結尾。
- 名字中只能使用字母、數字和下底線。
- 不要在名字中出現連續下底線，這樣很難辨認。
- 在名字中需要空格的地方用下底線代替。
- 盡量避免使用縮寫詞，縮寫時需要簡明易懂。

資料表的儲存引擎

- ▶ MySQL 支援多種儲存引擎，以便處理多種類型資料表
- ▶ MySQL 支援非交易以及交易(Transaction)二種儲存引擎
- ▶ 資料表的儲存引擎類型如下

SHOW ENGINES;

儲存引擎類型	支援交易	說明
ISAM	N	最初的儲存引擎
MyISAM	N	二元可攜式儲存引擎,取代ISAM
BDB	Y	用 page locking 作安全交易
InnoDB	Y	用 row locking 作安全交易並支援外來鍵功能
MEMOARY/HEAP	N	資料儲存在記憶體中,但關機後立即消失
MERGE/MRG_MyISAM	N	將多個資料表的資料合併(UNION)成為一個資料表

MyISAM Table Type

- MySQL 舊版預設的儲存引擎(storage engine)
 - MyISAM Table Type
- 以 ISAM(Indexed Sequential Access Method) 為基礎
- 一個資料表由 3 個檔案構成
 - *.frm Table 的結構
 - *.MYD Table 的資料
 - *.MYI Table 的 Index(使用 B-tree Indexes)
- 可以使用 ENGINE 來指定

```
CREATE TABLE T
  ( i int )
ENGINE = MyISAM;
```
- 此類型並不支援交易處理

InnoDB Table Type

- ▶ InnoDB Table Type 提供交易儲存引擎(預設)
 - Commit, rollback, crash recovery capabilities
 - Row Level Locking
 - No Locking in Select statement
 - Support Foreign key constraints
 - Data and Indexes 存在資料表空間(Tablespace)
 - Table Structure 同 MyISAM

資料表的儲存引擎

```
CREATE TABLE DEPT (
    DEPTNO      INT NOT NULL,
    DNAME        VARCHAR(14),
    LOC          VARCHAR(13),
    CONSTRAINT  DEPT_PRIMARY_KEY
    PRIMARY KEY (DEPTNO)
) ENGINE = INNODB ;
```

Table Details	
Engine:	InnoDB
Row format:	Dynamic
Column count:	3
Table rows:	4
AVG row length:	4096
Data length:	16.0 KiB
Index length:	0.0 bytes

Module 7. 欄位的資料型態1

- 7-1: 數字型態：整數型態、浮點數型態
- 7-2: 日期與時間型態
- 7-3: 字串型態

欄位資料型態(Column Data Types)

數值資料	日期時間資料	文數字資料
TINYINT	DATE	CHAR(M)
SMALLINT	DATETIME	VARCHAR(M)
MEDIUMINT	TIMESTAMP	TINYBLOB, TINYTEXT
INT, INTEGER	TIME	BLOB, TEXT
BIGINT	YEAR	MEDIUMBLOB, MEDIUMTEXT
FLOAT(p)		LONGBLOB, LONGTEXT
FLOAT		
DOUBLE [PRECISION], item REAL		
DECIMAL(M,D), NUMERIC(M,D)		

整數型態

- ▶ 整數(Integer)
 - TINYINT, SMALLINT, MEDIUMINT
 - INT/INTEGER, BIGINT
- ▶ 整數資料-資料長度與有效範圍

Type	Bytes	Minimum Value (Signed)	Maximum Value (Signed)
TINYINT	1	-128	127
SMALLINT	2	-32768	32767
MEDIUMINT	3	-8388608	8388607
INT	4	-2147483648	2147483647
BIGINT	8	-9223372036854775808	9223372036854775807

浮點數型態

- ▶ 實數
 - DECIMAL(p, s), NUMERIC(p, s)
 - FLOAT, DOUBLE PRECISION(p, s), DOUBLE
- ▶ 精準實數(DECIMAL & NUMERIC)
 - DECIMAL與NUMERIC型態是相同的作法
 - 儲存時以字串儲存，而不是二元浮點方式
 - 最大的範圍同DOUBLE
 - 可以指定精確度及小數位數

DECIMAL [(p [,s])]

NUMERIC [(p [,s])]

- 總長度(p)=整數+小數(s) 預設值為10
- 範例：DECIMAL(5, 2) 範圍999.99 ~ -999.99

浮點數型態

FLOAT [(p)]

DOUBLE [(p)]

- p 指定精確長度,決定儲存的大小
- 資料長度與有效範圍
 - 單精確數 4bytes p=0~23
 - 雙精確數 8bytes p=24~53

日期與時間型態

▶ 日期與時間資料

- DATE
- DATETIME
- TIMESTAMP
- TIME
- YEAR

型態	資訊	格式	範圍	儲存大小
DATETIME	DATE & TIME	'YYYY-MM-DD HH:MM:SS'	'1000-01-01 00:00:00'至 '9999-12-31 23:59:59'	8 bytes
TIMESTAMP	DATE & TIME	'YYYY-MM-DD HH:MM:SS'	'1000-01-01 00:00:00'至 '9999-12-31 23:59:59'	4 bytes
DATE	DATE	'YYYY-MM-DD'	'1000-01-01'~'9999-12-31'	3 bytes
TIME	TIME	'HH:MM:SS'	'-838:59:59'~'838:59:59'	3 bytes
YEAR	YEAR	'YYYY'	1901~2155	1 byte

DateTime資料格式

格式	範例	日期時間
'YYYY-MM-DD HH:MM:SS'	'1998-12-31 11:30:45'	1998-12-31 11:30:45
'YY-MM-DD HH:MM:SS'	'98-12-31 11:30:45'	1998-12-31 11:30:45
'YYYY-MM-DD'	'1998-12-31'	1998-12-31
'YY-MM-DD'	'98-12-31'	1998-12-31
'YYYYMMDDHHMMSS'	'19970523091528'	1997-05-23 09:15:28
'YYMMDDHHMMSS'	'970523091528'	1997-05-23 09:15:28
'YYYYMMDD'	'19981231'	1998-12-31
'YYMMDD'	'981231'	1998-12-31
YYYYMMDDHHMMSS	19830905132800	1983-09-05 13:28:00
YYMMDDHHMMSS	830905132800	1983-09-05 13:28:00
YYYYMMDD	19830905	1983-09-05
YYMMDD	830905	1983-09-05

DateTime資料以字串型態輸入查詢

- 'YYMMDD'

```
mysql> SELECT empno, ename, job, hiredate,sal
      -> FROM emp
      -> WHERE hiredate='811203';
+-----+-----+-----+-----+-----+
| empno | ename | job      | hiredate           | sal    |
+-----+-----+-----+-----+-----+
| 7900  | JAMES | CLERK    | 1981-12-03 00:00:00 | 950.00 |
| 7902  | FORD   | ANALYST  | 1981-12-03 00:00:00 | 3000.00 |
+-----+-----+-----+-----+-----+
```

- 'YYYYMMDD'

```
mysql> SELECT empno, ename, job, hiredate,sal
      -> FROM emp
      -> WHERE hiredate='19811203';
+-----+-----+-----+-----+-----+
| empno | ename | job      | hiredate           | sal    |
+-----+-----+-----+-----+-----+
| 7900  | JAMES | CLERK    | 1981-12-03 00:00:00 | 950.00 |
| 7902  | FORD   | ANALYST  | 1981-12-03 00:00:00 | 3000.00 |
+-----+-----+-----+-----+-----+
```

日期時間的分隔符號

- 你可將-或:符號改成其他任何符號

字串格式	日期/時間
'98-12-31 11:30:45'	1998-12-31 11:30:45
'98.12.31 11+30+45'	1998-12-31 11:30:45
'98/12/31 11*30*45'	1998-12-31 11:30:45
'98@12@31 11^30^45'	1998-12-31 11:30:45

- 所有不合法的格式, 格式均以0表示

資料型態	日期/時間
DATE	0000-00-00
DATETIME	0000-00-00 00:00:00
TIMESTAMP	0000000000000000

日期時間的分隔符號

.YY-MM-DD

```
mysql> SELECT empno, ename, job, hiredate, sal
      -> FROM emp
      -> WHERE hiredate='81-12-03';
+-----+-----+-----+-----+-----+
| empno | ename | job      | hiredate          | sal    |
+-----+-----+-----+-----+-----+
|  7900 | JAMES | CLERK    | 1981-12-03 00:00:00 |  950.00 |
|  7902 | FORD   | ANALYST  | 1981-12-03 00:00:00 | 3000.00 |
+-----+-----+-----+-----+-----+
```

• YY#MM#DD

```
mysql> SELECT empno, ename, job, hiredate, sal
      -> FROM emp
      -> WHERE hiredate='81#12#03';
+-----+-----+-----+-----+-----+
| empno | ename | job      | hiredate          | sal    |
+-----+-----+-----+-----+-----+
|  7900 | JAMES | CLERK    | 1981-12-03 00:00:00 |  950.00 |
|  7902 | FORD   | ANALYST  | 1981-12-03 00:00:00 | 3000.00 |
+-----+-----+-----+-----+-----+
```

DateTime 資料位數

- ▶ 可用一位數表示月，日，時，分，秒
 - '1979-6-9' = '1979-06-09'
 - '1979-10-30 1:2:3' = '1979-10-30 01:02:03'
- ▶ 若年為2位數時
 - 00-69 轉成 2000~2069
 - 70-99 轉成 1970~1999
 - 可用6, 8, 12, 14等長度自動判斷

長度	判斷格式	判斷長度(年)
6	YYMMDD	2
8	YYYYMMDD	4
12	YYMMDDHHMMSS	2
14	YYYYMMDDHHMMSS	4

- ▶ 型態互換注意事項
 - DATE 轉 DATETIME, TIMESTAMP 時間單位為 '00:00:00'
 - DATETIME, TIMESTAMP 轉 DATE 截斷時間單位

TIME 資料型態

▶ 顯示格式及範圍

- 'HH:MM:SS'
- '-838:59:59' ~ '838:59:59'

▶ 輸入字串格式

- 'D HH:MM:SS.fraction'
- 'D HH:MM:SS'
- 'D HH:MM'
- 'D HH'
- 'HH:MM:SS.fraction' 或 'HHMMSS.fraction'
- 'HH:MM:SS' 或 'HHMMSS'
- 'HH:MM' 或 'HHMM'
- 'SS'

▶ 注意：MySQL 不會儲存 fraction 部份

TIME 資料型態

▶ 輸入字串格式轉換原則

D : 0 ~ 34 days (1 D = 24 hours)

格式	範例	時間
'D HH:MM:SS.fraction'	'1 06:10:10.123000'	30:10:10.123000
'D HH:MM:SS'	'1 06:10:10'	30:10:10
'D HH:MM'	'1 06:10'	30:10:00
'D HH'	'1 06'	30:00:00
'HH:MM:SS.fraction'	'06:10:10.123000'	06:10:10.123000
'HHMMSS.fraction'	'061010.123000'	06:10:10.123000
'HH:MM:SS'	'06:10:10'	06:10:10
'HHMMSS'	'061010'	06:10:10
'HH:MM'	'06:10'	06:10:00
'MMSS'	'0610'	00:06:10 (注意)
'SS'	'10'	00:00:10 (注意)

所有無法自動轉換的字串，皆轉成00:00:00

YEAR資料型態

▶ 顯示格式及範圍

- 'YYYY'
- 1901~2155

▶ 自動轉換原則

格式	範例	時間
'YYYY'	'1901'~'2155' '0000'	1901~2155 0000
'YY'	'00'~'69' '70'~'99'	2000~2069 1970~1999
YYYY	1901~2155 1700	1901~2155 0000
YY	1~69 70~99 0	2001~2069 1970~1999 0000

對於無法自動轉換的字串，皆轉成0000

文字資料 (Character Types)

類型	儲存方式	儲存大小	範圍
CHAR(N)	固定size 不足部份以 空白 填入 顯示時將 空白 截斷	字元數	0~255
VARCHAR(N)	變動size(空白移除) 空白部份自動截斷 顯示時將 空白 截斷	字元數+ 1byte	0~255
TINYTEXT TEXT MEDIUMTEXT LONGTEXT	Character 分大小寫	L+1 bytes L+2 bytes L+3 bytes L+4 bytes	L < 2^8 L < 2^16 L < 2^16 L < 2^16

Module 8. 欄位的資料型態2

8-1: 二元碼(binary)型態

8-2: 列舉型態

8-3: 集合型態

二元碼型態

- ▶ 二元碼型態是一種二進位字串型態，適合存放音樂或圖片資料。

類型	儲存方式	儲存大小	範圍
TINYBLOB	Binary 不分大小寫	L+1 bytes	$L < 2^8$
BLOB		L+2 bytes	$L < 2^{16}$
MEDIUMBLOB		L+3 bytes	$L < 2^{16}$
LONGBLOB		L+4 bytes	$L < 2^{16}$

列舉型態

- ▶ 列舉型態是一種特殊的資料表欄位型態，欄位的值只能是列舉值其中之一。

```
ENUM ('value1', 'value2', 'value3', ... )
```

```
CREATE TABLE clothes (
    id          INT      PRIMARY KEY AUTO_INCREMENT,
    brand       VARCHAR(255) NOT NULL,
    size        ENUM('S', 'M', 'L', 'XL') NOT NULL
);
```

MySQL會給列舉值一個編號：1, 2, 3, 4

集合型態

- ▶ 集合型態的資料表欄位，可存放零個或多個集合成員資料。

```
CREATE TABLE table_test(
    id INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
    activities SET('Travel', 'Sports', 'Dancing', 'Fine Dining')
);
```

MySQL會給成員一個編號：1, 2, 4, 8

table_test

id	activities
1	Sports
2	Travel,Sports
3	Travel,Sports,Dancing
4	Travel,Dancing

Module 9. 資料表處理

- 9-1: 新增資料表**
- 9-2: 建置資料表**
- 9-3: 修改資料表的設計**

資料表(Database Tables)

- ▶ 最主要的資料庫物件(Database Object)
- ▶ 資料庫中儲存資料的物件
- ▶ 主要由欄位(Column)及資料列(Row)所組成
- ▶ 欄位(Column)
 - 每一個欄位都必須命名而且必須設定資料型態和資料長度，用來存放欄位資料。
- ▶ 資料列(Row)
 - 資料列是資料表中的一筆記錄資料

欄位資料型態(Column Data Types)

數值資料	日期時間資料	文數字資料
TINYINT	DATE	CHAR(M)
SMALLINT	DATETIME	VARCHAR(M)
MEDIUMINT	TIMESTAMP	TINYBLOB, TINYTEXT
INT, INTEGER	TIME	BLOB, TEXT
BIGINT	YEAR	MEDIUMBLOB, MEDIUMTEXT
FLOAT(p)		LONGBLOB, LONGTEXT
FLOAT		
DOUBLE [PRECISION], item REAL		
DECIMAL(M,D), NUMERIC(M,D)		

定義資料表(Defining Tables)

必要項目(Required Elements)	選擇性項目(Optional Elements)
<ul style="list-style-type: none">• 資料表名稱(Table name)• 欄位名稱(Column names)• 欄位資料型態(Column data types)• 欄位資料長度(column size)	<ul style="list-style-type: none">• 欄位預設值(Default)• 資料檢查條件(Constraints)<ul style="list-style-type: none">• 不允許空值(NOT NULL)• 主鍵(Primary key)• 唯一鍵(Unique key)• 外來鍵(Foreign key)

新建資料表 – CREATE TABLE

```
CREATE TABLE TableName  
( ColumnName type, . . . );
```

- TableName 表格名稱
- ColumnName 欄位名稱
- Type 欄位資料型態[(長度)]

```
CREATE TABLE DEPT  
( DEPTNO SMALLINT(4),  
  DNAME  VARCHAR(14),  
  LOC    VARCHAR(13)  
);
```

顯示資料表結構(Table Structure)

- ▶ 使用 DESCRIBE 命令查詢資料表的結構(欄位資訊)

```
DESC [RIBE] tablename
```

- ▶ 查看資料表 dept 的結構(欄位資訊)

```
DESCRIBE dept
```

Field	Type	Null	Key	Default	Extra
DEPTNO	int(11)	NO	PRI	NULL	
DNAME	varchar(14)	YES		NULL	
LOC	varchar(13)	YES		NULL	

新建資料表的注意事項

- ▶ 要有建立資料表的權限
- ▶ 必需指定資料表與欄位名稱，最大長度64 bytes
- ▶ 以字母開頭，符號可用_,\$
- ▶ 不可使用系統保留字
- ▶ 在同一個資料庫中，**不可有相同名稱的資料表**

IF NOT EXISTS 選項

- 可判斷資料表是否存在

```
CREATE TABLE [IF NOT EXISTS] TableName  
( ColumnName type, ...);
```

- 若資料表已存在，則不建立資料表
- 若資料表不存在，則建立資料表

```
CREATE TABLE IF NOT EXISTS DEPT  
( DEPTNO SMALLINT(4),  
  DNAME  VARCHAR(14),  
  LOC    VARCHAR(13)  
);
```

設定資料表儲存引擎

```
CREATE TABLE TableName
( ColumnName type, . . . )
    {ENGINE = {BDB|HEAP|ISAM|InnoDB|
MERGE|MRG_MYISAM|MYISAM};
```

```
CREATE TABLE IF NOT EXISTS dept
( deptno smallint(4),
  dname  varchar(14),
  loc    varchar(13)
) ENGINE = InnoDB;
```

註：沒有指定 ENGINE 時，其預定的類型由參數 default-storage-engine 控制

資料表的維護(Managing Tables)

- ▶ 資料表可以修改其結構
 - 新增, 修改, 刪除 欄位
 - 新增, 刪除 資料檢查條件
- ▶ 修改資料表名稱或欄位名與註解
 - Rename
 - Comment
- ▶ 可以刪除資料表的結構或資料
 - 刪除資料
 - DELETE , TRUNCATE
 - 刪除資料及結構
 - DROP TABLE

新增一個新欄位 (Adding a Column)

▶ 加入新欄位

```
ALTER TABLE TableName  
ADD [COLUMN] Column_Definition [FIRST|AFTER column];  
OR  
ALTER TABLE TableName  
ADD [COLUMN] (Column_Definition, ...);
```

- Column_Definition 欄位的定義
- FIRST 新增在第一個欄位
- AFTER column 新增在指定欄位後
- 預設為新增在最後一個欄位之後

9-3: 修改資料表的設計

```
CREATE TABLE EMP10
( EMPNO int(11) NOT NULL,
  ENAME varchar(10),
  JOB varchar(9),
  SAL decimal(7,2)
);
```

```
mysql> desc emp10;
+-----+-----+-----+-----+-----+
| Field | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| EMPNO | int(11)   | NO   |     | NULL    |       |
| ENAME | varchar(10) | YES  |     | NULL    |       |
| JOB   | varchar(9)  | YES  |     | NULL    |       |
| SAL   | decimal(7,2) | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

9-3: 修改資料表的設計

```
mysql> ALTER TABLE EMP10
      -> ADD COLUMN MGR SMALLINT;
Query OK, 0 rows affected (0.36 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

新增在
最後一個欄位之後

```
mysql> desc emp10;
+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| EMPNO | int(11)       | NO   |     | NULL    |       |
| ENAME | varchar(10)  | YES  |     | NULL    |       |
| JOB   | varchar(9)    | YES  |     | NULL    |       |
| SAL   | decimal(7,2)  | YES  |     | NULL    |       |
| MGR   | smallint(6)   | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)
```

9-3: 修改資料表的設計

```
mysql> desc emp10;
+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| EMPNO | int(11)       | NO   |     | NULL    |       |
| ENAME | varchar(10)   | YES  |     | NULL    |       |
| JOB   | varchar(9)      | YES  |     | NULL    |       |
| SAL   | decimal(7,2)    | YES  |     | NULL    |       |
| MGR   | smallint(6)    | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+
```

```
mysql> ALTER TABLE EMP10
      -> ADD COLUMN HIREDATE DATE AFTER JOB;
Query OK, 0 rows affected (0.34 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

新增在 JOB
欄位之後

```
mysql> desc emp10;
+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| EMPNO | int(11)       | NO   |     | NULL    |       |
| ENAME | varchar(10)   | YES  |     | NULL    |       |
| JOB   | varchar(9)      | YES  |     | NULL    |       |
| HIREDATE | date          | YES  |     | NULL    |       |
| SAL   | decimal(7,2)    | YES  |     | NULL    |       |
| MGR   | smallint(6)    | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)
```

9-3: 修改資料表的設計

新增為第一個欄位

```
mysql> ALTER TABLE emp10
      -> ADD PHONE varchar(12) DEFAULT '02-66310000' FIRST;
Query OK, 0 rows affected (0.39 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

```
mysql> desc emp10;
+-----+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| PHONE | varchar(12) | YES |     | 02-66310000 |       |
| EMPNO | int(11)    | NO  |     | NULL     |       |
| ENAME | varchar(10) | YES |     | NULL     |       |
| JOB   | varchar(9)  | YES |     | NULL     |       |
| HIREDATE | date | YES |     | NULL     |       |
| SAL   | decimal(7,2) | YES |     | NULL     |       |
| MGR   | smallint(6) | YES |     | NULL     |       |
| COMM  | int(11)    | YES |     | NULL     |       |
| DEPTNO | smallint(6) | YES |     | NULL     |       |
| EMAIL | varchar(200) | YES |     | NULL     |       |
+-----+-----+-----+-----+-----+-----+
10 rows in set (0.00 sec)
```

新增多個欄位

```
mysql> ALTER TABLE EMP10
      -> ADD COLUMN(COMM INT, DEPTNO SMALLINT, EMAIL VARCHAR(200));
Query OK, 0 rows affected (0.37 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

```
mysql> desc emp10;
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| EMPNO | int(11)       | NO   |     | NULL    |       |
| ENAME | varchar(10)   | YES  |     | NULL    |       |
| JOB   | varchar(9)    | YES  |     | NULL    |       |
| HIREDATE | date        | YES  |     | NULL    |       |
| SAL   | decimal(7,2)  | YES  |     | NULL    |       |
| MGR   | smallint(6)   | YES  |     | NULL    |       |
| COMM  | int(11)       | YES  |     | NULL    |       |
| DEPTNO | smallint(6)  | YES  |     | NULL    |       |
| EMAIL | varchar(200)  | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
9 rows in set (0.00 sec)
```

修改欄位定義(Modifying a column)

▶ 修改欄位定義

```
ALTER TABLE TableName
ALTER [COLUMN] column {SET DEFAULT literal|DROP DEFAULT};

ALTER TABLE TableName
MODIFY [COLUMN] column_definition [FIRST|AFTER column];
```

- 修改欄位預設值
- 修改欄位資料型態
- 修改欄位順序

修改欄位預設值（取消預設值設定）

```
mysql> ALTER TABLE EMP10
      -> ALTER PHONE DROP DEFAULT;
Query OK, 0 rows affected (0.07 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

```
mysql> desc emp10;
+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| PHONE | varchar(12) | YES |   | NULL    |       |
| EMPNO | int(11)    | NO  |   | NULL    |       |
| ENAME | varchar(10) | YES |   | NULL    |       |
| JOB   | varchar(9)  | YES |   | NULL    |       |
| HIREDATE | date | YES |   | NULL    |       |
| SAL   | decimal(7,2) | YES |   | NULL    |       |
| MGR   | smallint(6) | YES |   | NULL    |       |
| COMM  | int(11)    | YES |   | NULL    |       |
| DEPTNO | smallint(6) | YES |   | NULL    |       |
| EMAIL | varchar(200) | YES |   | NULL    |       |
+-----+-----+-----+-----+-----+
10 rows in set (0.00 sec)
```

9-3: 修改資料表的設計

修改欄位資料型態/順序

```
mysql> ALTER TABLE EMP10
      -> MODIFY COLUMN MGR INT AFTER JOB;
Query OK, 3 rows affected (0.54 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

```
mysql> desc emp10;
+-----+-----+-----+-----+-----+
| Field | Type       | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| PHONE | varchar(12) | YES  |     | NULL    |       |
| EMPNO | int(11)     | NO   |     | NULL    |       |
| ENAME | varchar(10) | YES  |     | NULL    |       |
| JOB   | varchar(9)  | YES  |     | NULL    |       |
| MGR   | int(11)     | YES  |     | NULL    |       |
| HIREDATE | date     | YES  |     | NULL    |       |
| SAL   | decimal(7,2) | YES  |     | NULL    |       |
| COMM  | int(11)     | YES  |     | NULL    |       |
| DEPTNO | smallint(6) | YES  |     | NULL    |       |
| EMAIL | varchar(200) | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+
10 rows in set (0.00 sec)
```

9-3: 修改資料表的設計

修改欄位資料型態 / NOT NULL

```
mysql> ALTER TABLE EMP10
      -> MODIFY COLUMN ENAME VARCHAR(20) NOT NULL;
Query OK, 3 rows affected (0.54 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

```
mysql> desc emp10;
+-----+-----+-----+-----+-----+
| Field | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| PHONE | varchar(12) | YES  |     | NULL    |       |
| EMPNO | int(11)    | NO   |     | NULL    |       |
| ENAME | varchar(20) | NO   |     | NULL    |       |
| JOB   | varchar(9)  | YES  |     | NULL    |       |
| MGR   | int(11)    | YES  |     | NULL    |       |
| HIREDATE | date    | YES  |     | NULL    |       |
| SAL   | decimal(7,2) | YES  |     | NULL    |       |
| COMM  | int(11)    | YES  |     | NULL    |       |
| DEPTNO | smallint(6) | YES  |     | NULL    |       |
| EMAIL | varchar(200) | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+
10 rows in set (0.00 sec)
```

更改欄位名稱/定義

► 更改欄位名稱/定義

```
ALTER TABLE TableName  
CHANGE [COLUMN] old_column New_column_definition;
```

- **old_column** 舊欄位名稱
- **New_column_definition** 新欄位定義

9-3: 修改資料表的設計

修改欄位名稱與資料型態

```
mysql> ALTER TABLE EMP10
      -> CHANGE COLUMN SAL SALARY SMALLINT;
Query OK, 3 rows affected (0.54 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

```
mysql> desc emp10;
+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| PHONE | varchar(12)   | YES  |     | NULL    |       |
| EMPNO | int(11)        | NO   |     | NULL    |       |
| ENAME | varchar(20)    | NO   |     | NULL    |       |
| JOB   | varchar(9)      | YES  |     | NULL    |       |
| MGR   | int(11)        | YES  |     | NULL    |       |
| HIREDATE | date         | YES  |     | NULL    |       |
| SALARY | smallint(6)   | YES  |     | NULL    |       |
| COMM  | int(11)        | YES  |     | NULL    |       |
| DEPTNO | smallint(6)   | YES  |     | NULL    |       |
| EMAIL | varchar(200)   | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+
10 rows in set (0.00 sec)
```

刪除欄位(Droping a column)

▶ 刪除欄位

```
ALTER TABLE TableName  
DROP [COLUMN] ColumnName;
```

- 欄位有無資料皆可刪除，資料表最少留一個欄位

```
mysql> ALTER TABLE dept1  
-> DROP COLUMN loc;  
Query OK, 0 rows affected (0.46 sec)  
Records: 0  Duplicates: 0  Warnings: 0
```

```
mysql> desc dept1;  
+-----+-----+-----+-----+-----+  
| Field | Type      | Null | Key | Default | Extra |  
+-----+-----+-----+-----+-----+  
| DEPTNO | int(11)   | NO   | PRI | NULL    |       |  
| DNAME  | varchar(14) | YES  |     | NULL    |       |  
+-----+-----+-----+-----+-----+  
2 rows in set (0.00 sec)
```

資料表改名(Managing the Name of Table)

- ▶ 資料表重新命名

```
ALTER TABLE TableName  
RENAME [TO] new_table;
```

- ▶ 將 EMP10 改名為 EMP10A

```
ALTER TABLE EMP10  
RENAME TO EMP10A;
```

截斷資料表中的資料 (Truncating a Table)

- ▶ 刪除資料表中的所有資料
- ▶ 釋出所使用的磁碟空間
- ▶ 保留資料表的結構

```
TRUNCATE TABLE TableName;
```

```
TRUNCATE TABLE EMP10A;
```

刪除資料表(Droping a Table)

- ▶ 刪除資料表物件
 - 刪除資料表的所有資料及結構

```
DROP TABLE TableName;
```

```
DROP TABLE EMP10A;
```

修改資料表及欄位註解(Managing Comment)

- 註解
 - 資料表註解

```
ALTER TABLE EMP10
    COMMENT 'FOR DEPARTMENT NUMBER 10';
```

- 欄位註解

```
ALTER TABLE EMP10
    MODIFY COLUMN EMPNO INT COMMENT 'EMPLOYEE NUMBER';
```

- 查詢註解

```
SHOW TABLE STATUS;
SHOW FULL COLUMNS FROM tbl_name;
```

變更資料表儲存引擎類型

- ▶ 變更資料表儲存引擎類型

```
ALTER TABLE table
ENGINE = {MyISAM|InnoDB};
```

```
CREATE TABLE TX
( NO      INT AUTO_INCREMENT NOT NULL PRIMARY KEY,
DTIMES  DATETIME );

SHOW TABLE STATUS FROM SAMPLE LIKE 'TX';

ALTER TABLE TX ENGINE=InnoDB;

SHOW TABLE STATUS FROM SAMPLE LIKE 'TX';
```

Module 10.

資料查詢1

- 10-1: 欄位別名
- 10-2: 排除重複 DISTINCT
- 10-3: 算術運算子、空值、常值

欄位別名(Column Alias)

- 一般欄位 (General Columns)
- 運算式欄位 (Expression Column)
- 中文及特殊字或空白使用雙引號 “ ”

Column | expression [AS] alias

```
mysql> SELECT empno "編號" , ename Name, sal AS SALARY, sal*12 "Ann Sal"
-> FROM emp;
+-----+-----+-----+
| 編號 | Name   | SALARY | Ann Sal |
+-----+-----+-----+
| 7839 | KING    | 5000   | 60000  |
| 7698 | BLAKE   | 2850   | 34200  |
| 7782 | CLARK   | 2450   | 29400  |
| 7566 | JONES   | 2975   | 35700  |
| 7654 | MARTIN  | 1250   | 15000  |
| 7499 | ALLEN   | 1600   | 19200  |
| 7844 | TURNER  | 1500   | 18000  |
| 7900 | JAMES   | 950    | 11400  |
| 7521 | WARD    | 1250   | 15000  |
| 7902 | FORD    | 3000   | 36000  |
| 7369 | SMITH   | 800    | 9600   |
| 7788 | SCOTT   | 3000   | 36000  |
| 7876 | ADAMS   | 1100   | 13200  |
| 7934 | MILLER  | 1300   | 15600  |
+-----+-----+-----+
14 rows in set (0.01 sec)
```

排除重複資料列

```
SELECT DISTINCT <column list>
FROM <table or view>
```

```
mysql> SELECT job FROM emp;
+-----+
| job   |
+-----+
| PRESIDENT |
| MANAGER   |
| MANAGER   |
| MANAGER   |
| SALESMAN  |
| SALESMAN  |
| SALESMAN  |
| CLERK     |
| SALESMAN  |
| ANALYST   |
| CLERK     |
| ANALYST   |
| CLERK     |
| CLERK     |
+-----+
14 rows in set (0.00 sec)
```

```
mysql> SELECT DISTINCT job FROM emp;
+-----+
| job   |
+-----+
| PRESIDENT |
| MANAGER   |
| SALESMAN  |
| CLERK     |
| ANALYST   |
+-----+
5 rows in set (0.01 sec)
```

算術與運算子

▶ 運算式

- 運算元(operand) 運算子(operator)
- $5 + 3$

▶ 運算式中的運算元(operand)資料型態要相同

▶ 算術運算子

- 加法 +
- 減法 -
- 乘法 *
- 除法 /

數值資料運算

```
mysql> SELECT ename, sal, sal+3000
      -> FROM emp;
+-----+-----+-----+
| ename | sal   | sal+3000 |
+-----+-----+-----+
| SMITH | 800.00 | 3800.00 |
| ALLEN | 1600.00 | 4600.00 |
| WARD  | 1250.00 | 4250.00 |
| JONES | 2975.00 | 5975.00 |
| MARTIN | 1250.00 | 4250.00 |
| BLAKE | 2850.00 | 5850.00 |
| CLARK | 2450.00 | 5450.00 |
| SCOTT | 3000.00 | 6000.00 |
| KING  | 5000.00 | 8000.00 |
| TURNER | 1500.00 | 4500.00 |
| ADAMS | 1100.00 | 4100.00 |
| JAMES | 950.00 | 3950.00 |
| FORD  | 3000.00 | 6000.00 |
| MILLER | 1300.00 | 4300.00 |
+-----+-----+-----+
14 rows in set (0.00 sec)
```

算術運算子優先順序

- ▶ 算術運算子優先順序
 - $*$, $/$ 優先於 $+$, $-$
 - $*$ 與 $/$ 同時出現時，由左至右
 - $+$ 與 $-$ 同時出現時，由左至右
- ▶ 使用括號(Using Parentheses)
 - 小刮弧
 - 改變運算順序
 - $3 * 4 + 2 / 3$
 - $3 * (4+2) / 3$
 - 意義更清楚
 - $3 * 4 + 2 / 3$
 - $(3 * 4) + (2 / 3)$

使用括號改變優先順序

```
mysql> SELECT ename, sal, 12*sal+100
      -> FROM emp;
+-----+-----+-----+
| ename | sal   | 12*sal+100 |
+-----+-----+-----+
| SMITH | 800.00 | 9700.00  |
| ALLEN | 1600.00 | 19300.00 |
| WARD  | 1250.00 | 15100.00 |
| JONES  | 2975.00 | 35800.00 |
| MARTIN | 1250.00 | 15100.00 |
| BLAKE  | 2850.00 | 34300.00 |
| CLARK  | 2450.00 | 29500.00 |
| SCOTT  | 3000.00 | 36100.00 |
| KING   | 5000.00 | 60100.00 |
| TURNER | 1500.00 | 18100.00 |
| ADAMS  | 1100.00 | 13300.00 |
| JAMES  | 950.00  | 11500.00 |
| FORD   | 3000.00 | 36100.00 |
| MILLER | 1300.00 | 15700.00 |
+-----+-----+-----+
14 rows in set (0.00 sec)
```

```
mysql> SELECT ename, sal, 12*(sal+100)
      -> FROM emp;
+-----+-----+-----+
| ename | sal   | 12*(sal+100) |
+-----+-----+-----+
| SMITH | 800.00 | 10800.00 |
| ALLEN | 1600.00 | 20400.00 |
| WARD  | 1250.00 | 16200.00 |
| JONES  | 2975.00 | 36900.00 |
| MARTIN | 1250.00 | 16200.00 |
| BLAKE  | 2850.00 | 35400.00 |
| CLARK  | 2450.00 | 30600.00 |
| SCOTT  | 3000.00 | 37200.00 |
| KING   | 5000.00 | 61200.00 |
| TURNER | 1500.00 | 19200.00 |
| ADAMS  | 1100.00 | 14400.00 |
| JAMES  | 950.00  | 12600.00 |
| FORD   | 3000.00 | 37200.00 |
| MILLER | 1300.00 | 16800.00 |
+-----+-----+-----+
14 rows in set (0.00 sec)
```

空值(Null Value)

- ▶ **NULL** 值是在新增資料時沒有設定欄位值，資料庫管理系統便設定該欄位的內容為NULL
 - 無法使用
 - 無法指派
 - 無法運算
 - 與 0 和空白不相同
- ▶ 所有型態皆可以為空值
- ▶ 運算式中若含有null value時，其結果為null

空值的運算

mysql> SELECT empno, ename, sal, comm, **sal+comm** ←
-> FROM emp;

Expression Column

empno	ename	sal	comm	sal+comm
7369	SMITH	800.00	NULL	NULL
7499	ALLEN	1600.00	300.00	1900.00
7521	WARD	1250.00	500.00	1750.00
7566	JONES	2975.00	NULL	NULL
7654	MARTIN	1250.00	1400.00	2650.00
7698	BLAKE	2850.00	NULL	NULL
7782	CLARK	2450.00	NULL	NULL
7788	SCOTT	3000.00	NULL	NULL
7839	KING	5000.00	NULL	NULL
7844	TURNER	1500.00	0.00	1500.00
7876	ADAMS	1100.00	NULL	NULL
7900	JAMES	950.00	NULL	NULL
7902	FORD	3000.00	NULL	NULL
7934	MILLER	1300.00	NULL	NULL

14 rows in set (0.00 sec)

常值(Literal)

- ▶ 不變的資料值，不儲存於資料庫內部
- ▶ 類型分為
 - 數值常值:任何數值
 - 5 -24 3.65 -8
 - 字串常值:於一對單引號內所輸入的文字
 - ‘book’ ‘員工編號’
 - 日期常值:於一對單引號內輸入的日期資料(需符合日期格式)
 - ‘19951015’

```
mysql> SELECT concat(ename , ' job is ' , job) Jobs
      -> FROM emp;
+-----+
| Jobs           |
+-----+
| SMITH job is CLERK   |
| ALLEN job is SALESMAN |
| WARD job is SALESMAN  |
| JONES job is MANAGER  |
| MARTIN job is SALESMAN |
| ... ... ... ... ... |
| FORD job is ANALYST   |
| MILLER job is CLERK    |
+-----+
```

Module 11.

資料查詢2

11-1: WHERE Clause

11-2: 比較運算子、邏輯運算子

11-3: LIKE 運算子

資料查詢 – SELECT 敘述

子句(Element)	Expression	Role
SELECT	<select list>	給定查詢的資料項目 Defines which columns to return
FROM	<table source>	給定資料來源 Defines table(s) to query
WHERE	<search condition>	給定查詢/過濾資料條件 Filters rows using a predicate
GROUP BY	<group by list>	資料分組設定 Arranges rows by groups
HAVING	<search condition>	給定分組資料查詢/過濾條件 Filters groups using a predicate
ORDER BY	<order by list>	給定查詢結果排序方式 Sorts the output

11-2: WHERE Clause

- ▶ 篩選資料列 - 設定篩選條件
 - 依查詢條件篩選符合條件的資料列

empno	ename	job	deptno
7839	KING	PRESIDENT	10
7698	BLAKE	MANAGER	30
7782	CLARK	MANAGER	10
7566	JONES	MANAGER	20
7654	MARTIN	SALESMAN	30
7499	ALLEN	SALESMAN	30
7844	TURNER	SALESMAN	30
7900	JAMES	CLERK	30
7521	WARD	SALESMAN	30
7902	FORD	ANALYST	20
7369	SMITH	CLERK	20
7788	SCOTT	ANALYST	20
7876	ADAMS	CLERK	20
7934	MILLER	CLERK	10

```
Select empno, ename, job, deptno  
From emp;
```

```
Select empno, ename, job, deptno  
From emp  
Where deptno = 10;
```

只查詢部門代號為10的員工資料

empno	ename	job	deptno
7839	KING	PRESIDENT	10
7782	CLARK	MANAGER	10
7934	MILLER	CLERK	10

使用 WHERE 子句設定查詢條件

```
SELECT column, ...
FROM table
WHERE conditions;
```

- 置於 FROM 子句後
- 條件子句(Conditions)
 - 邏輯值：真(TRUE)、假(FALSE)、空值(NULL)
- 比較運算式/邏輯運算式/ SQL特定運算式
 - 運算元可以是欄位、運算式、函數、常數

```
mysql> SELECT empno,ename,job,deptno
-> FROM emp
-> WHERE deptno = 10;
+-----+-----+-----+-----+
| empno | ename  | job    | deptno |
+-----+-----+-----+-----+
| 7782  | CLARK  | MANAGER |      10 |
| 7839  | KING   | PRESIDENT |      10 |
| 7934  | MILLER | CLERK   |      10 |
+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

比較運算子(Comparison Operators)

- ▶ 比較運算子使用在二個資料項的比較大小
 - 數值資料：值
 - 字串資料：內碼(預設)
 - 日期時間資料：世紀、年、月、日、時、分、秒

Operator	Meaning
=	等於 (Equal to)
>	大於 (Greater than)
>=	大於或等於 (Greater than or equal to)
<	小於 (Less than)
<=	小於或等於 (Less than or equal to)
<>	不等於 (Not equal to)

使用比較運算做條件查詢

- 數值資料-列出薪水大於等於3000的員工

```
mysql> SELECT empno, ename, job, sal
-> FROM emp
-> WHERE sal >= 3000;
+-----+-----+-----+
| empno | ename | job      | sal   |
+-----+-----+-----+
| 7839  | KING   | PRESIDENT | 5000 |
| 7902  | FORD   | ANALYST   | 3000 |
| 7788  | SCOTT  | ANALYST   | 3000 |
+-----+-----+-----+
```

- 日期資料- 1981-12-03 進公司的員工

```
mysql> SELECT empno,ename,job,deptno, hiredate
-> FROM emp
-> WHERE hiredate = '1981-12-03';
+-----+-----+-----+-----+
| empno | ename | job      | deptno | hiredate        |
+-----+-----+-----+-----+
| 7900  | JAMES  | CLERK    |     30 | 1981-12-03 00:00:00 |
| 7902  | FORD   | ANALYST  |     20 | 1981-12-03 00:00:00 |
+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

使用比較運算做條件查詢

- 字串資料-列出 KING 的資料

```
mysql> SELECT empno,ename,job,deptno, hiredate, sal
-> FROM emp
-> WHERE ename = 'KING';
+-----+-----+-----+-----+-----+
| empno | ename | job      | deptno | hiredate           | sal   |
+-----+-----+-----+-----+-----+
|    7839 | KING  | PRESIDENT |       10 | 1981-11-17 00:00:00 | 5000.00 |
+-----+-----+-----+-----+-----+
```

- 相同資料型態的欄位資料

```
mysql> SELECT ename, sal, comm
-> FROM emp
-> WHERE sal <= comm;
+-----+-----+-----+
| ename | sal     | comm    |
+-----+-----+-----+
| MARTIN | 1250.00 | 1400.00 |
+-----+-----+-----+
1 row in set (0.00 sec)
```

邏輯運算子(Logical Operators)

- ▶ 若有一個以上的條件運算必須使用邏輯運算子結合成一個運算結果

邏輯運算子	運算結果
AND	Returns TRUE if both component conditions are TRUE
OR	Returns TRUE if either component condition is TRUE
NOT	Returns TRUE if the following condition is FALSE

AND

- AND 必須二個運算元都是真(TRUE)才會符合查詢條件

```
mysql> SELECT empno, ename, job, sal
-> FROM emp
-> WHERE sal>=1100 AND job='CLERK';
+-----+-----+-----+-----+
| empno | ename  | job    | sal   |
+-----+-----+-----+-----+
|  7876 | ADAMS  | CLERK | 1100.00 |
|  7934 | MILLER | CLERK | 1300.00 |
+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

AND

- 列出薪水>2000且職務為manager的員工

```
mysql> SELECT empno, ename, job, sal, mgr
-> FROM emp
-> WHERE sal > 2000 AND job = 'MANAGER';
+-----+-----+-----+-----+-----+
| empno | ename | job      | sal     | mgr   |
+-----+-----+-----+-----+-----+
|    7566 | JONES | MANAGER | 2975.00 | 7839 |
|    7698 | BLAKE | MANAGER | 2850.00 | 7839 |
|    7782 | CLARK | MANAGER | 2450.00 | 7839 |
+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

11-2: 比較運算子、邏輯運算子

OR

- OR 只要任一個運算元是真(TRUE)就符合查詢條件

```
mysql> SELECT empno, ename, job, sal
-> FROM emp
-> WHERE sal>=1100 OR job='CLERK';
+-----+-----+-----+-----+
| empno | ename  | job    | sal   |
+-----+-----+-----+-----+
| 7369  | SMITH  | CLERK  | 800.00 |
| 7499  | ALLEN  | SALESMAN | 1600.00 |
| 7521  | WARD   | SALESMAN | 1250.00 |
| 7566  | JONES  | MANAGER | 2975.00 |
| 7654  | MARTIN | SALESMAN | 1250.00 |
| 7698  | BLAKE  | MANAGER | 2850.00 |
| 7782  | CLARK  | MANAGER | 2450.00 |
| 7788  | SCOTT  | ANALYST | 3000.00 |
| 7839  | KING   | PRESIDENT | 5000.00 |
| 7844  | TURNER | SALESMAN | 1500.00 |
| 7876  | ADAMS  | CLERK   | 1100.00 |
| 7900  | JAMES  | CLERK   | 950.00  |
| 7902  | FORD   | ANALYST | 3000.00 |
| 7934  | MILLER | CLERK   | 1300.00 |
+-----+-----+-----+-----+
14 rows in set (0.00 sec)
```

OR

- 列出薪水>2000或職務為manager的員工

```
mysql> SELECT empno, ename, job, sal, mgr
-> FROM emp
-> WHERE sal > 2000 OR job = 'MANAGER';
+-----+-----+-----+-----+-----+
| empno | ename | job       | sal      | mgr   |
+-----+-----+-----+-----+-----+
|    7566 | JONES | MANAGER  | 2975.00 | 7839 |
|    7698 | BLAKE | MANAGER  | 2850.00 | 7839 |
|    7782 | CLARK | MANAGER  | 2450.00 | 7839 |
|    7788 | SCOTT | ANALYST  | 3000.00 | 7566 |
|    7839 | KING  | PRESIDENT | 5000.00 | NULL  |
|    7902 | FORD  | ANALYST  | 3000.00 | 7566 |
+-----+-----+-----+-----+-----+
6 rows in set (0.00 sec)
```

NOT

- NOT 反向運算,只有一個運算元

```
mysql> SELECT empno, ename, job, sal, mgr
-> FROM emp
-> WHERE NOT (sal > 2000 OR job = 'MANAGER');
+-----+-----+-----+-----+
| empno | ename  | job    | sal   | mgr  |
+-----+-----+-----+-----+
| 7369  | SMITH  | CLERK  | 800.00 | 7902 |
| 7499  | ALLEN  | SALESMAN | 1600.00 | 7698 |
| 7521  | WARD   | SALESMAN | 1250.00 | 7698 |
| 7654  | MARTIN | SALESMAN | 1250.00 | 7698 |
| 7844  | TURNER | SALESMAN | 1500.00 | 7698 |
| 7876  | ADAMS  | CLERK   | 1100.00 | 7788 |
| 7900  | JAMES   | CLERK   | 950.00  | 7698 |
| 7934  | MILLER | CLERK   | 1300.00 | 7782 |
+-----+-----+-----+-----+
8 rows in set (0.00 sec)
```

運算子執行優先順序

Order Evaluated	Operator
1	Arithmetic operators
2	Concatenation operator
3	Comparison conditions
4	IS [NOT] NULL, LIKE, [NOT] IN
5	[NOT] BETWEEN
6	NOT logical condition
7	AND logical condition
8	OR logical condition

Override rules of precedence by using parentheses.

- 可使用小括號改變執行運算順序

運算子執行優先順序

▶ AND → OR

```
mysql> SELECT empno, ename, job, sal
    -> FROM emp
    -> WHERE job = 'MANAGER'
    ->      OR job = 'SALESMAN'
    ->      AND sal < 1500;
+-----+-----+-----+-----+
| empno | ename  | job       | sal   |
+-----+-----+-----+-----+
| 7698  | BLAKE   | MANAGER   | 2850  |
| 7782  | CLARK   | MANAGER   | 2450  |
| 7566  | JONES   | MANAGER   | 2975  |
| 7654  | MARTIN  | SALESMAN  | 1250  |
| 7521  | WARD    | SALESMAN  | 1250  |
+-----+-----+-----+-----+
5 rows in set (0.00 sec)
```



運算子執行優先順序

- ▶ 利用小括號改變運算順序

```
mysql> SELECT empno, ename, job, sal
-> FROM emp
-> WHERE (job = 'MANAGER'
->       OR job = 'SALESMAN') 1.OR
->       AND sal < 1500; 2.AND
+-----+-----+-----+-----+
| empno | ename   | job        | sal   |
+-----+-----+-----+-----+
|    7654 | MARTIN | SALESMAN | 1250 |
|    7521 | WARD    | SALESMAN | 1250 |
+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

LIKE 運算子

- 模糊比對-萬用字元查詢

```
expr LIKE 'pattern' ESCAPE 'character'
```

- pattern

%	百分比	任意字元
__	底線	一個字元

- ESCAPE 'character'

定義跳脫字元

- &%
- &_

Example : ESCAPE '&'

'%'
'_'

LIKE 運算子

- 列出姓名以A開頭的員工

```
mysql> SELECT empno, ename, job, sal
-> FROM emp
-> WHERE ename LIKE 'A%';
+-----+-----+-----+-----+
| empno | ename | job      | sal     |
+-----+-----+-----+-----+
|    7499 | ALLEN | SALESMAN | 1600.00 |
|    7876 | ADAMS | CLERK    | 1100.00 |
+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

LIKE 運算子

- 列出姓名以N結尾的員工

```
mysql> SELECT empno, ename, job, sal
-> FROM emp
-> WHERE ename LIKE '%N';
+-----+-----+-----+-----+
| empno | ename   | job      | sal    |
+-----+-----+-----+-----+
| 7499  | ALLEN   | SALESMAN | 1600.00 |
| 7654  | MARTIN  | SALESMAN | 1250.00 |
+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

LIKE 運算子

- 列出姓名中含有T的員工

```
mysql> SELECT empno, ename, job, sal
-> FROM emp
-> WHERE ename LIKE '%T%';
+-----+-----+-----+-----+
| empno | ename  | job    | sal   |
+-----+-----+-----+-----+
| 7369  | SMITH  | CLERK  | 800.00 |
| 7654  | MARTIN | SALESMAN | 1250.00 |
| 7788  | SCOTT  | ANALYST | 3000.00 |
| 7844  | TURNER | SALESMAN | 1500.00 |
+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

LIKE 運算子

- 列出姓名第二個字為A的員工

```
mysql> SELECT empno, ename, job, sal
-> FROM emp
-> WHERE ename LIKE '_A%';
+-----+-----+-----+-----+
| empno | ename   | job      | sal    |
+-----+-----+-----+-----+
| 7521  | WARD    | SALESMAN | 1250.00 |
| 7654  | MARTIN  | SALESMAN | 1250.00 |
| 7900  | JAMES    | CLERK    | 950.00  |
+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

LIKE 運算子與 ESCAPE

```
mysql> SELECT *
-> FROM liquors
-> WHERE content LIKE '%3&%' ESCAPE '&';
```

```
mysql> SELECT *
-> FROM liquors
-> WHERE content LIKE '%3\%' ESCAPE '\';
```

Module 12.

資料查詢3

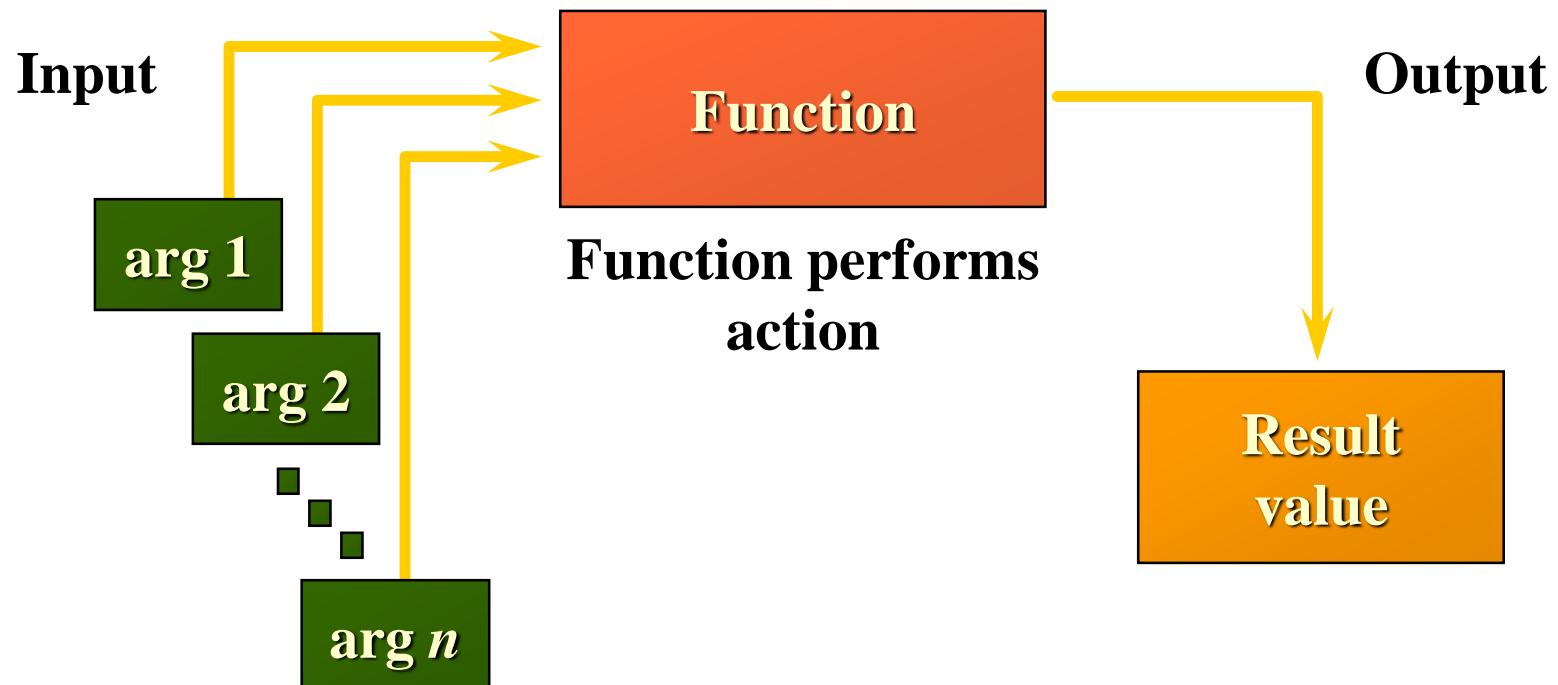
12-1: String Functions

12-2: Date and Time Functions

12-3: General Functions

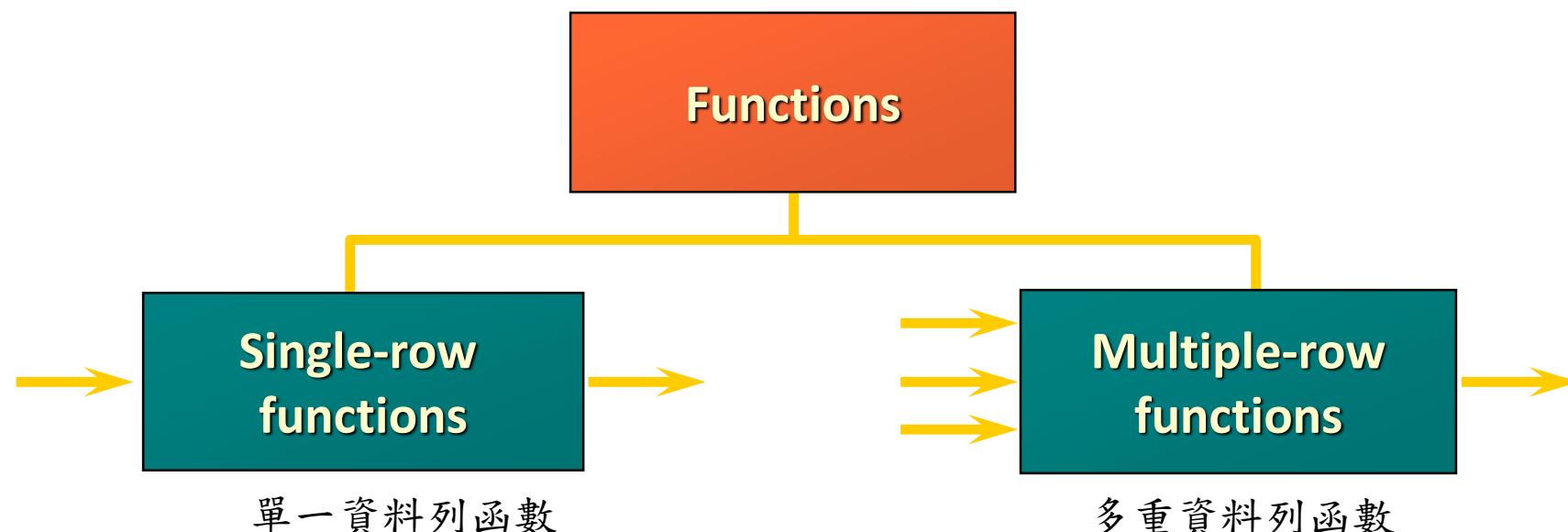
函數(Functions)

- 強化資料處理與運算的能力
- 可接受使用者輸入參數並回傳結果



SQL 函數

- 單一資料列函數(Single-Row Functions)
 - 每筆記錄(row)執行一次,傳回一個結果
- 多重資料列函數(Multiple-Row Functions)
 - 多筆記錄(rows)執行一次，傳回一個結果資料彙總



SQL字串函數

函數	功能
LENGTH(str)	字串長度
CHAR_LENGTH(str)	字元個數
LCASE(str) LOWER(str)	將字串轉成小寫字母
UCASE(str) UPPER(str)	將字串轉成大寫字母
ASCII(char)	將 char 轉成字元代碼
CONCAT(str1,str2,...)	字串連結
FIELD(str, str1, str2, str3,...)	傳回 str 在 str list 的位置
INSERT(str, pos, len, newstr)	在指定區間插入一字串
LEFT(str, len)	由左至右傳回指定字元數
RIGHT(str, len)	由右至左傳回指定字元數
REVERSE(str)	將字串反轉
LPAD(str, len, padstr)	將字串不足指定長度以指定字元向左補齊
RPAD(str,len,padstr)	將字串不足指定長度以指定字元向右補齊

SQL字串函數

函數	功能
SUBSTRING(str, pos) SUBSTRING(str FROM pos) SUBSTRING(str, pos, len) SUBSTRING(str FROM pos FOR len)	截取字串部份字元
REPEAT(str, count)	將指定字串重複指定次數
SPACE(N)	將空白字元重複指定次數
INSTR(str, substr) LOCATE(substr, str[, pos])	傳回指定字元在字串中出現的位置
REPLACE(str, from_str, to_str)	將新字串替代舊字串
LTRIM(str)	去除字串左邊空白字元
RTRIM(str)	去除字串右邊空白字元
TRIM([[BOTH LEADING TRAILING] [remstr] FROM] str)	去除字串二端空白字元
strcmp(expr1,expr2)	expr1=expr2 return 0 expr1>expr2 return 1 expr1<expr2 return -1

SQL字串函數的套用

Function	Result
CONCAT('Good', 'String')	GoodString
SUBSTRING('String',1,3)	Str
LENGTH('String')	6
INSTR('String', 'r')	3
LPAD(sal,10,'*')	*****5000
TRIM('S' FROM 'SSMITH')	MITH

欄位套用字串函數

```
mysql> SELECT concat(deptno,dname) Department
      -> FROM dept;
+-----+
| Department |
+-----+
| 10ACCOUNTING |
| 20RESEARCH |
| 30SALES |
| 40OPERATIONS |
+-----+
4 rows in set (0.00 sec)
```

欄位套用字串函數產生新欄位

```
mysql> select ename, sal, lpad(sal,10,'#'), rpad(sal,10,'#')
   -> from emp;
+-----+-----+-----+-----+
| ename | sal      | lpad(sal,10,'#') | rpad(sal,10,'#') |
+-----+-----+-----+-----+
| SMITH | 800.00   | #####800.00    | 800.00####    |
| ALLEN | 1600.00  | ###1600.00   | 1600.00###   |
| WARD  | 1250.00  | ###1250.00   | 1250.00###   |
| JONES | 2975.00  | ###2975.00   | 2975.00###   |
| MARTIN | 1250.00 | ###1250.00 | 1250.00### |
| BLAKE | 2850.00 | ###2850.00 | 2850.00### |
| CLARK | 2450.00 | ###2450.00 | 2450.00### |
| SCOTT | 3000.00 | ###3000.00 | 3000.00### |
| KING  | 5000.00 | ###5000.00 | 5000.00### |
| TURNER | 1500.00 | ###1500.00 | 1500.00### |
| ADAMS | 1100.00 | ###1100.00 | 1100.00### |
| JAMES | 950.00  | ###950.00   | 950.00###   |
| FORD  | 3000.00 | ###3000.00 | 3000.00### |
| MILLER | 1300.00 | ###1300.00 | 1300.00### |
+-----+-----+-----+-----+
```

不同子句的使用字串函數

```
mysql> select ename, sal, concat(job, ' in department ',deptno) Job_dep
-> from emp
-> where substr(ename,1,1)='M';
+-----+-----+-----+
| ename | sal    | Job_dep          |
+-----+-----+-----+
| MARTIN | 1250.00 | SALESMAN in department 30 |
| MILLER | 1300.00 | CLERK in department 10   |
+-----+-----+-----+
```

數值函數(Numeric functions)

函數	功能
ROUND(X,D)	四捨五入
TRUNCATE(X,D)	無條件捨去
MOD(N,M)	求餘數
CEIL(X)	傳回不小於 X 的最小整數值
FLOOR(X)	傳回不大於 X 的最大整數值
POWER(X,Y)	X的Y次方
SQRT(X)	平方根
ABS(X)	絕對值
SIGN(X)	正負數
RAND() RAND(N)	亂數
PI()	圓周率
RADIANS(X)	度數轉徑值
DEGREES(X)	徑值轉度數

數值函數(Numeric functions)

```
mysql> select ename, sal, repeat('*',round(sal/100,0))
-> from emp
-> where deptno=10;
+-----+-----+
| ename | sal      | repeat('*',round(sal/100,0)) |
+-----+-----+
| CLARK | 2450.00 | *****
| KING  | 5000.00 | ***** ***** ***** ***** ***** ***** ***** |
| MILLER | 1300.00 | *****
+-----+-----+
```

```
mysql> select ename, sal, sal+floor(rand()*1000)
-> from emp
-> where deptno=20;
+-----+-----+
| ename | sal      | sal+floor(rand()*1000) |
+-----+-----+
| SMITH | 800.00 | 1775.00 |
| JONES | 2975.00 | 3716.00 |
| SCOTT | 3000.00 | 3780.00 |
| ADAMS | 1100.00 | 1780.00 |
| FORD  | 3000.00 | 3060.00 |
+-----+-----+
```

日期/時間函數(Date and Time functions)

函數	功能
CURDATE()	現在日期
CURTIME()	現在時間
CURRENT_TIMESTAMP()	現在日期與時間
NOW()	現在日期與時間
UTC_DATE()	格林威治日期
UTC_TIME()	格林威治時間
UTC_TIMESTAMP()	格林威治日期與時
YEAR(date)	傳回指定日期的年份
MONTH(date)	傳回指定日期的月份
DAY(date)	傳回指定日期的日份
HOUR(time)	傳回指定日期的時
MINUTE(time)	傳回指定日期的分
SECOND(time)	傳回指定日期的秒

日期/時間函數(Date and Time functions)

函數	功能
TIME(expr)	傳回時間單位
MICROSECOND(expr)	傳回指定日期的微秒
EXTRACT(type FROM date)	傳回指定的日期單位
DAYNAME(date)	星期名稱(Monday, …)
MONTHNAME(date)	月份名稱
DAYOFWEEK(date)	一週中的第幾天(星期日是第一天)
DAYOFMONTH(date)	月份中的第幾天
DAYOFYEAR(date)	一年中的第幾天
WEEK(date[, mode])	傳回周數0~52
WEEKDAY(date)	傳回星期索引(0, 一, 6, 日)
WEEKOFYEAR(date)	一年中的第幾週 MySQL 4.1.1
YEARWEEK(date[, start])	Returns year and week

計算目前日期是一週的第幾天、全年第幾週

```
mysql> SELECT curdate(), DAYOFWEEK(curdate()),
->           weekday(curdate()), dayofmonth(curdate()), dayofyear(curdate());
+-----+-----+-----+-----+
| curdate() | DAYOFWEEK(curdate()) | weekday(curdate()) | dayofmonth(curdate()) | dayofyear(curdate()) |
+-----+-----+-----+-----+
| 2015-03-15 |             1 |              6 |            15 |            74 |
+-----+-----+-----+-----+
```

```
Mysql> SELECT curdate(), YEARWEEK(curdate());
+-----+-----+
| curdate() | YEARWEEK(curdate()) |
+-----+-----+
| 2015-03-15 |          201511 |
+-----+-----+
1 row in set (0.00 sec)
```

日期/時間函數(Date and Time functions)

函數	功能
DATEDIFF(expr1, expr2)	二個日期相減
ADDDATE(date, INTERVAL expr type)	日期加法
SUBDATE(date, INTERVAL expr type)	日期減法
ADDTIME(expr1, expr2)	時間加法
SUBTIME(expr1, expr2)	時間減法
TIMEDIFF(expr1, expr2)	時間減法
TIMESTAMP(expr1, expr2)	時間加法
TIMESTAMPADD(interval, int_expr, datetime_expr)	時間加法
TIMESTAMPDIFF(interval, datetime_expr1, datetime_expr2)	時間減法
PERIOD_DIFF(P1, P2)	月份(年月-年月)

▶ 查詢現在系統日期時間

```
mysql> select now(),DAYNAME(now()),
       EXTRACT(month FROM curdate()), DATEDIFF(curdate(),'20150317') ;
+-----+-----+-----+
| now() | DAYNAME(now()) | EXTRACT(month FROM curdate()) | DATEDIFF(curdate(),'20150317') |
+-----+-----+-----+
| 2015-03-15 11:54:45 | Sunday | 3 | -2 |
+-----+-----+-----+
```

▶ 日期時間計算

```
mysql> SELECT curdate(), adddate(curdate(), INTERVAL 3 DAY),
       adddate(curdate(), INTERVAL 3 month), adddate(curdate(), INTERVAL 3 year);
+-----+-----+-----+
| curdate() | adddate(curdate(), INTERVAL 3 DAY) | adddate(curdate(), INTERVAL 3 month) | adddate(curdate(), INTERVAL 3 year) |
+-----+-----+-----+
| 2015-03-14 | 2015-03-17 | 2015-06-14 | 2018-03-14 |
+-----+-----+-----+
```

► 計算雇用半年後的日期

```
mysql> SELECT ename, hiredate, ADDDATE(hiredate, INTERVAL 6 Month)
-> FROM emp;
+-----+-----+-----+
| ename | hiredate          | ADDDATE(hiredate, INTERVAL 6 Month) |
+-----+-----+-----+
| ALLEN | 1981-02-20 00:00:00 | 1981-08-20 00:00:00
| JONES | 1981-04-02 00:00:00 | 1981-10-02 00:00:00
| MARTIN | 1981-09-28 00:00:00 | 1982-03-28 00:00:00
| BLAKE | 1981-05-01 00:00:00 | 1981-11-01 00:00:00
| CLARK | 1981-06-09 00:00:00 | 1981-12-09 00:00:00
| SCOTT | 1982-12-09 00:00:00 | 1983-06-09 00:00:00
| KING | 1981-11-17 00:00:00 | 1982-05-17 00:00:00
| TURNER | 1981-09-08 00:00:00 | 1982-03-08 00:00:00
| ADAMS | 1983-01-12 00:00:00 | 1983-07-12 00:00:00
| MARY | 1981-12-03 00:00:00 | 1982-06-03 00:00:00
| MILLER | 1982-01-23 00:00:00 | 1982-07-23 00:00:00
+-----+-----+-----+
11 rows in set (0.00 sec)
```

日期/時間函數(Date and Time functions)

函數	功能
DATE(expr)	將字串轉成日期
STR_TO_DATE(str, format)	將字串轉成日期
MAKEDATE(year, dayofyear)	將日期字串轉成日期
MAKETIME(hour, minute, second)	將時間字串轉成時間
DATE_FORMAT(date, format)	將日期格式化成字串
TIME_FORMAT(time, format)	將時間格式化成字串
TO_DAYS(date)	距啟算日的總天數
FROM_DAYS(N)	將總天數轉成某日期
LAST_DAY(date)	指定日期月份的最後一天
QUARTER(date)	一年中的第幾季
TIME_TO_SEC(time)	換算成一天總秒數
SEC_TO_TIME(seconds)	總秒數換算成一天時間

12-2: Date and Time Functions

- 日期格式化成字串

DATE_FORMAT (curdate() ,‘ %D of %M , %Y’);

- 格式化符號

Specifier	Meaning
%a	Three-characters abbreviated weekday name e.g., Mon, Tue, Wed, etc.
%b	Three-characters abbreviated month name e.g., Jan, Feb, Mar, etc.
%c	Month in numeric e.g., 1, 2, 3...12
%D	Day of the month with English suffix e.g., 0th, 1st, 2nd, etc.
%d	Day of the month with leading zero if it is 1 number e.g., 00, 01,02, ...31
%e	Day of the month without leading zero e.g., 1,2,...31
%f	Microseconds in the range of 000000..999999
%H	Hour with 24-hour format with leading zero e.g., 00..23
%h	Hour with 12-hour format with leading zero e.g., 01, 02...12
%l	Same as %h
%i	Minutes with leading zero e.g., 00, 01,...59
%j	Day of year with leading zero e.g., 001,002,...366
%k	Hour in 24-hour format without leading zero e.g., 0,1,2...23
%l	Hour in 12-hour format without leading zero e.g., 1,2...12

12-2: Date and Time Functions

- 格式化符號(續上頁)

Specifier	Meaning
%M	Full month name e.g., January, February,...December
%m	Month name with leading zero e.g., 00,01,02,...12
%p	AM or PM, depending on other time specifiers
%r	Time in 12-hour format hh:mm:ss AM or PM
%S	Seconds with leading zero 00,01,...59
%s	Same as %S
%T	Time in 24-hour format hh:mm:ss
%U	Week number with leading zero when the first day of week is Sunday e.g., 00,01,02...53
%u	Week number with leading zero when the first day of week is Monday e.g., 00,01,02...53
%V	Same as %U; it is used with %X
%v	Same as %u; it is used with %x
%W	Full name of weekday e.g., Sunday, Monday,..., Saturday
%w	Weekday in number (0=Sunday, 1= Monday,etc.)
%X	Year for the week in four digits where the first day of the week is Sunday; often used with %V
%x	Year for the week, where the first day of the week is Monday, four digits; used with %v
%Y	Four digits year e.g., 2000, 2001,...etc.
%y	Two digits year e.g., 10,11,12, etc.
%%	Add percentage (%) character to the output

▶ 格式化日期以字串顯示

```
mysql> SELECT curdate(),Date_format(curdate(),' %W %D %M %Y');
+-----+-----+
| curdate() | Date_format(curdate(),' %W %D %M %Y') |
+-----+-----+
| 2015-03-14 | Saturday 14th March 2015 |
+-----+-----+
```

▶ 加入字串常數

```
mysql> SELECT curdate(),Date_format(curdate(),' %W ,the %D of %M, %Y');
+-----+-----+
| curdate() | Date_format(curdate(),' %W ,the %D of %M, %Y') |
+-----+-----+
| 2015-03-14 | Saturday ,the 14th of March, 2015 |
+-----+-----+
```

系統資訊函數(Information Functions)

函數	功能
USER()	連線的使用者
VERSION()	資料庫版本
DATABASE()	連線的資料庫
CONNECTION_ID()	連線的Connection ID
CHARSET(str)	字串所屬的字元集

```
mysql> SELECT USER(),VERSION(),DATABASE(),
->           CONNECTION_ID(), CHARSET('abc');
+-----+-----+-----+-----+-----+
| USER() | VERSION() | DATABASE() | CONNECTION_ID() | CHARSET('abc') |
+-----+-----+-----+-----+-----+
| jih@localhost | 5.6.22-log | exp | 3 | utf8 |
+-----+-----+-----+-----+-----+
```

- ▶ 流程控制函數(Control Flow Functions)
 - IF(expr1, expr2, expr3)
 - IFNULL(expr1, expr2)
 - NULLIF(expr1, expr2)

IF(expr1,expr2,expr3)

If expr1 is TRUE

then

 returns expr2,

else

 returns expr3.

```
mysql> SELECT IF(1>2,2,3) x1,
    ->      IF(1<2,'yes','no') x2,
    ->      IF(STRCMP('test','test1'),'no','yes') x3;
+----+----+----+
| x1 | x2 | x3 |
+----+----+----+
|   3 | yes | no |
+----+----+----+
```

IFNULL(expr1,expr2)

If expr1 **is not** NULL,

Then

returns expr1

Return type : numeric or string

else

returns expr2

```
mysql> SELECT IFNULL(1,0),
->           IFNULL(NULL,10),
->           IFNULL(1/0,10),
->           IFNULL(1/0,'yes');
+-----+-----+-----+-----+
| IFNULL(1,0) | IFNULL(NULL,10) | IFNULL(1/0,10) | IFNULL(1/0,'yes') |
+-----+-----+-----+-----+
|          1 |        10          |      10.00    |     yes       |
+-----+-----+-----+-----+
```

NULLIF (expr1,expr2)

If expr1 = expr2 ,

Then

 returns **NULL**

else

Return type : numeric or string

 returns **expr1**

```
mysql> SELECT NULLIF(1,1), NULLIF(1,0), NULLIF(0,1);
+-----+-----+-----+
| NULLIF(1,1) | NULLIF(1,0) | NULLIF(0,1) |
+-----+-----+-----+
|      NULL   |         1 |         0 |
+-----+-----+-----+
```

Module 13.

資料查詢4

13-1: BETWEEN 運算子

13-2: IN 運算子

13-3: IS NULL 運算子

SQL特定運算子

- **LIKE** 運算子: 萬用字元查詢
- **BETWEEN** 運算子: 一個連續區間值的查詢
- **IN** 運算子: 列舉值的查詢
- **IS NULL** 運算子: 空值(**NULL**)的查詢

Operator	Meaning
BETWEEN <low_val> AND <hi_val>	Between two values (inclusive)
IN (list)	Match any of a list of values
LIKE	Match a character pattern
IS NULL	Is a null value

BETWEEN AND 運算子

- 連續區間的條件判斷

```
expr BETWEEN x1 AND x2
```

- expr 介於x1及x2之間(包含x1 & x2)
- x1 必需小於 x2
- 可使用在數值、日期及字元資料
 - WHERE sal BETWEEN 1000 AND 3000
 - WHERE hiredate BETWEEN '1981-01-01' AND '1981-06-30'
 - WHERE ename BETWEEN 'A' AND 'E'

使用 BETWEEN AND 查詢

- ▶ 查詢薪水介於 2000 到 3500 之間的員工

```
mysql> SELECT empno, ename, job, sal
-> FROM emp
-> WHERE sal BETWEEN 2000 AND 3500;
+-----+-----+-----+-----+
| empno | ename | job      | sal    |
+-----+-----+-----+-----+
| 7566  | JONES | MANAGER | 2975.00 |
| 7698  | BLAKE | MANAGER | 2850.00 |
| 7782  | CLARK | MANAGER | 2450.00 |
| 7788  | SCOTT | ANALYST | 3000.00 |
| 7902  | FORD   | ANALYST | 3000.00 |
+-----+-----+-----+-----+
5 rows in set (0.00 sec)
```

IN 運算子

- 列舉值的查詢

```
expr IN (value1, value2, ...)
```

- value 所成的串列
- expr 和 value list 的資料型態要相同
- 可使用在數值、日期及字元資料
 - WHERE sal IN (100, 200, 300)
 - WHERE code IN ('A','B','C')
 - WHERE hiredate IN ('1981-05-01','1981-10-03')

使用IN查詢

- ▶ 列出職務為 salesman 或 manager 的員工

```
mysql> SELECT empno, ename, job, sal
-> FROM emp
-> WHERE job IN ('SALESMAN', 'MANAGER');
+-----+-----+-----+-----+
| empno | ename  | job    | sal   |
+-----+-----+-----+-----+
| 7499  | ALLEN  | SALESMAN | 1600.00 |
| 7521  | WARD   | SALESMAN | 1250.00 |
| 7566  | JONES  | MANAGER  | 2975.00 |
| 7654  | MARTIN | SALESMAN | 1250.00 |
| 7698  | BLAKE  | MANAGER  | 2850.00 |
| 7782  | CLARK  | MANAGER  | 2450.00 |
| 7844  | TURNER | SALESMAN | 1500.00 |
+-----+-----+-----+-----+
7 rows in set (0.00 sec)
```

IS NULL 運算子

► 空值運算子(判斷資料是否為NULL)

expr IS NULL

- NULL 專用的運算子
- $(NULL = NULL) \rightarrow NULL$
- NULL 與空白和0不相同
- 任何型態欄位皆可以為 NULL value

使用IS NULL查詢

- 找出公司老闆的資料(mgr 是 NULL的員工)

```
mysql> SELECT empno, ename, job, sal, mgr
-> FROM emp
-> WHERE mgr IS NULL;
+-----+-----+-----+-----+-----+
| empno | ename | job      | sal     | mgr   |
+-----+-----+-----+-----+-----+
|    7839 | KING  | PRESIDENT | 5000.00 | NULL  |
+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

與 NOT 搭配

- NOT BETWEEN .. AND ..
- NOT IN
- NOT LIKE
- IS NOT NULL

▶ 列出除了 manager 和 salesman 以外的員工

```
mysql> SELECT empno, ename, job, sal, deptno
-> FROM emp
-> WHERE job NOT IN ('SALESMAN', 'MANAGER');
+-----+-----+-----+-----+-----+
| empno | ename  | job       | sal     | deptno |
+-----+-----+-----+-----+-----+
| 7369 | SMITH   | CLERK    | 800.00  | 20    |
| 7788 | SCOTT   | ANALYST  | 3000.00 | 20    |
| 7839 | KING    | PRESIDENT | 5000.00 | 10    |
| 7876 | ADAMS   | CLERK    | 1100.00 | 20    |
| 7900 | JAMES   | CLERK    | 950.00  | 30    |
| 7902 | FORD    | ANALYST  | 3000.00 | 20    |
| 7934 | MILLER  | CLERK    | 1300.00 | 10    |
+-----+-----+-----+-----+-----+
7 rows in set (0.00 sec)
```

Module 14.

資料查詢4

- 14-1:CASE 運算式**
- 14-2: ORDER BY Clause**
- 14-3: LIMIT Clause**

CASE 運算式

- ▶ 列舉式 CASE (Simple CASE)
 - 與一串列舉的值做比較
 - 傳回第一個(值相等)的回傳值
 - 若都不相等則傳回 ELSE 的回傳值, 若無給定 ELSE 則傳回 NULL 值
- ▶ 條件式CASE (Searched CASE)
 - 與一串列舉的條件做比較
 - 傳回第一個(條件運算結果=真)的回傳值
 - 若都不符合則傳回 ELSE 的回傳值, 若無給定 ELSE 則傳回 NULL 值

列舉式 CASE (Simple CASE)

CASE expr

 WHEN v1 THEN r1

 [WHEN v2 THEN r2]

 ...

 [ELSE r]

END

```
mysql> SELECT empno, ename, sal, job,
->           CASE job
->             WHEN 'PRESIDENT' THEN sal*1.5
->             WHEN 'MANAGER' THEN sal*1.3
->             WHEN 'ANALYST' THEN sal*1.2
->             ELSE sal
->           END new_sal
->         FROM emp;
```

empno	ename	sal	job	new_sal
7839	KING	5000	PRESIDENT	7500.0
7698	BLAKE	2850	MANAGER	3705.0
7782	CLARK	2450	MANAGER	3185.0
7566	JONES	2975	MANAGER	3867.5
7654	MARTIN	1250	SALESMAN	1250.0
7499	ALLEN	1600	SALESMAN	1600.0
7844	TURNER	1500	SALESMAN	1500.0
7900	JAMES	950	CLERK	950.0
7521	WARD	1250	SALESMAN	1250.0
7902	FORD	3000	ANALYST	3600.0
7369	SMITH	800	CLERK	800.0
7788	SCOTT	3000	ANALYST	3600.0
7876	ADAMS	1100	CLERK	1100.0
7934	MILLER	1300	CLERK	1300.0

14 rows in set (0.00 sec)

條件式CASE (Searched CASE)

```
CASE
    WHEN condition1 THEN r1
    [WHEN condition2 THEN r2]
    ...
    [ELSE r]
END
```

```
mysql> SELECT empno, ename, sal,
->           CASE
->             WHEN sal BETWEEN 0      AND 1000 THEN 'A'
->             WHEN sal BETWEEN 1001 AND 2000 THEN 'B'
->             WHEN sal BETWEEN 2001 AND 3000 THEN 'C'
->             WHEN sal BETWEEN 3001 AND 4000 THEN 'D'
->             ELSE 'E'
->           END sal
->         FROM emp;
+-----+-----+-----+
| empno | ename  | sal   | sal  |
+-----+-----+-----+
|  7839 | KING   | 5000  | E    |
|  7698 | BLAKE  | 2850  | C    |
|  7782 | CLARK  | 2450  | C    |
|  7566 | JONES  | 2975  | C    |
|  7654 | MARTIN | 1250  | B    |
|  7499 | ALLEN  | 1600  | B    |
|  7844 | TURNER | 1500  | B    |
|  7900 | JAMES  |  950  | A    |
|  7521 | WARD   | 1250  | B    |
|  7902 | FORD   | 3000  | C    |
|  7369 | SMITH  |  800  | A    |
|  7788 | SCOTT  | 3000  | C    |
|  7876 | ADAMS  | 1100  | B    |
|  7934 | MILLER | 1300  | B    |
+-----+-----+-----+
14 rows in set (0.00 sec)
```

ORDER BY 子句

- ▶ 使用 ORDER BY 子句來做資料排序

```
SELECT column, ...
FROM table
[WHERE conditions]
ORDER BY {column|alias|expression|position [ASC|DESC], ...};
```

- 放置於SELECT敘述的最後一行
- ▶ 依指定欄位或運算式的資料值來排序
- ▶ 排序方式
 - ASC 升冪 Ascending 由小到大 [預設]
 - DESC 降冪 Descending 由大到小
- ▶ 若有空值(NULL)時, 升冪在最前面, 降冪在最下面
- ▶ 排序資料
 - 欄位/別名/運算式/位置

資料排序

▶ 資料未排序前

empno	ename	job	sal
7839	KING	PRESIDENT	5000
7698	BLAKE	MANAGER	2850
7566	JONES	MANAGER	2975
7902	FORD	ANALYST	3000
7788	SCOTT	ANALYST	3000

▶ 資料依薪資由低至高排列

empno	ename	job	sal
7698	BLAKE	MANAGER	2850
7566	JONES	MANAGER	2975
7902	FORD	ANALYST	3000
7788	SCOTT	ANALYST	3000
7839	KING	PRESIDENT	5000



使用欄位值排序 – 由小至大

- 依 select list 中的欄位

```
mysql> SELECT empno, ename, sal
-> FROM emp
-> WHERE deptno = 10
-> ORDER BY sal;
+-----+-----+-----+
| empno | ename  | sal   |
+-----+-----+-----+
|  7934 | MILLER | 1300 |
|  7782 | CLARK  | 2450 |
|  7839 | KING    | 5000 |
+-----+-----+-----+
3 rows in set (0.01 sec)
```

使用欄位值排序 – 由大至小

- 降冪 Descending

```
mysql> SELECT empno, ename, sal
-> FROM emp
-> WHERE deptno = 10
-> ORDER BY sal DESC;
+-----+-----+-----+
| empno | ename  | sal   |
+-----+-----+-----+
| 7839  | KING    | 5000  |
| 7782  | CLARK   | 2450  |
| 7934  | MILLER  | 1300  |
+-----+-----+-----+
3 rows in set (0.00 sec)
```

使用欄位值排序 – 由小至大

- 不在 select list 的欄位

```
mysql> SELECT empno, ename, sal
-> FROM emp
-> WHERE deptno = 10
-> ORDER BY hiredate;
+-----+-----+-----+
| empno | ename  | sal   |
+-----+-----+-----+
| 7782  | CLARK  | 2450 |
| 7839  | KING   | 5000 |
| 7934  | MILLER | 1300 |
+-----+-----+-----+
3 rows in set (0.00 sec)
```

- 依運算式排序

```
mysql> SELECT empno, ename, sal+comm bonus
-> FROM emp
-> WHERE deptno = 30
-> ORDER BY sal+comm;
+-----+-----+-----+
| empno | ename  | bonus  |
+-----+-----+-----+
| 7698  | BLAKE  | NULL   |
| 7900  | JAMES   | NULL   |
| 7844  | TURNER | 1500   |
| 7521  | WARD    | 1750   |
| 7499  | ALLEN   | 1900   |
| 7654  | MARTIN | 2650   |
+-----+-----+-----+
6 rows in set (0.01 sec)
```

- 依資料項在list中的位置順序

```
mysql> SELECT empno, ename, sal
-> FROM emp
-> WHERE deptno = 10
-> ORDER BY 3;
+-----+-----+-----+
| empno | ename  | sal   |
+-----+-----+-----+
| 7934  | MILLER | 1300  |
| 7782  | CLARK   | 2450  |
| 7839  | KING    | 5000  |
+-----+-----+-----+
3 rows in set (0.00 sec)
```

註：不可以為 0 或 大於 column list 的總欄位數

- 依欄位在list中的位置順序

```
mysql> SELECT *
-> FROM emp
-> WHERE deptno = 20
-> ORDER BY 3;
+-----+-----+-----+-----+-----+-----+-----+
| EMPNO | ENAME | JOB      | MGR    | HIREDATE | SAL     | COMM   | DEPTNO |
+-----+-----+-----+-----+-----+-----+-----+
| 7902  | FORD   | ANALYST  | 7566   | 1981-10-03 | 3000   | NULL   | 20      |
| 7788  | SCOTT  | ANALYST  | 7566   | 1982-10-09 | 3000   | NULL   | 20      |
| 7369  | SMITH  | CLERK    | 7902   | 1980-10-17 | 800    | NULL   | 20      |
| 7876  | ADAMS  | CLERK    | 7788   | 1983-01-12 | 1100   | NULL   | 20      |
| 7566  | JONES  | MANAGER  | 7839   | 1981-04-02 | 2975   | NULL   | 20      |
+-----+-----+-----+-----+-----+-----+-----+
5 rows in set (0.01 sec)
```

- 超出欄位的總數

```
mysql> SELECT *
->   FROM emp
->  WHERE deptno = 20
-> ORDER BY 9;
ERROR 1054 (42S22): Unknown column '9' in 'order clause'
```

```
mysql> SELECT empno, ename, sal
->   FROM emp
->  WHERE deptno = 10
-> ORDER BY 4;
ERROR 1054 (42S22): Unknown column '4' in 'order clause'
```

14-2: ORDER BY Clause

- 多個欄位的排序
 - 每一個欄位皆可指定降冪或升冪

```
mysql> SELECT *
-> FROM emp
-> ORDER BY deptno, job, 6 DESC, 1;
+-----+-----+-----+-----+-----+-----+-----+
| EMPNO | ENAME  | JOB    | MGR   | HIREDATE | SAL    | COMM   | DEPTNO |
+-----+-----+-----+-----+-----+-----+-----+
| 7934  | MILLER | CLERK  | 7782  | 1982-01-23 | 1300  | NULL   | 10     |
| 7782  | CLARK   | MANAGER | 7839  | 1981-06-09 | 2450  | NULL   | 10     |
| 7839  | KING    | PRESIDENT | NULL  | 1981-11-18 | 5000  | NULL   | 10     |
| 7788  | SCOTT   | ANALYST | 7566  | 1982-10-09 | 3000  | NULL   | 20     |
| 7902  | FORD    | ANALYST | 7566  | 1981-10-03 | 3000  | NULL   | 20     |
| 7876  | ADAMS   | CLERK  | 7788  | 1983-01-12 | 1100  | NULL   | 20     |
| 7369  | SMITH   | CLERK  | 7902  | 1980-10-17 | 800   | NULL   | 20     |
| 7566  | JONES   | MANAGER | 7839  | 1981-04-02 | 2975  | NULL   | 20     |
| 7900  | JAMES   | CLERK  | 7698  | 1981-10-03 | 950   | NULL   | 30     |
| 7698  | BLAKE   | MANAGER | 7839  | 1981-05-01 | 2850  | NULL   | 30     |
| 7499  | ALLEN   | SALESMAN | 7698  | 1981-02-20 | 1600  | 300    | 30     |
| 7844  | TURNER  | SALESMAN | 7698  | 1981-09-08 | 1500  | 0      | 30     |
| 7521  | WARD    | SALESMAN | 7698  | 1981-02-22 | 1250  | 500    | 30     |
| 7654  | MARTIN  | SALESMAN | 7698  | 1981-09-28 | 1250  | 1400   | 30     |
+-----+-----+-----+-----+-----+-----+-----+
14 rows in set (0.00 sec)
```

限制查詢結果的筆數

- SELECT 命令可以使用 LIMIT 子句來設定查詢結果傳回的資料筆數

```
SELECT [DISTINCT] <column list>
FROM <table or view>
LIMIT offset, count;
```

- offset : 傳回資料的開始位置, 由 0 起算, ≥ 0
- count : 傳回最大的資料筆數
- LIMIT 後面只加一個整數, 代表要傳回幾筆資料
 - 例如 : limit 5 代表從第一筆開始最多傳回5筆
- OFFSET 可以把它想成要略過筆數
 - 以offset=2為例, 可想成在找到的資料筆數中略過前二筆, 即是由第三筆開始回傳
- 使用LIMIT語句時通常會伴隨著「ORDER BY」

限制查詢結果的筆數

```
mysql> select ename, sal, job
      -> from emp;
+-----+-----+-----+
| ename | sal   | job    |
+-----+-----+-----+
| ALLEN | 2000.00 | SALESMAN |
| JONES  | 2975.00 | MANAGER  |
| MARTIN | 1250.00 | SALESMAN |
| BLAKE   | 2850.00 | MANAGER  |
| CLARK   | 2450.00 | MANAGER  |
| SCOTT   | 3000.00 | ANALYST |
| KING    | 5000.00 | PRESIDENT |
| TURNER  | 1500.00 | SALESMAN |
| ADAMS   | 1100.00 | CLERK   |
| MARY    | 3000.00 | ANALYST |
| MILLER  | 1300.00 | CLERK   |
+-----+-----+-----+
11 rows in set (0.00 sec)
```

```
mysql> select ename, sal, job
      -> from emp
      -> limit 5;
+-----+-----+-----+
| ename | sal   | job    |
+-----+-----+-----+
| ALLEN | 2000.00 | SALESMAN |
| JONES  | 2975.00 | MANAGER  |
| MARTIN | 1250.00 | SALESMAN |
| BLAKE   | 2850.00 | MANAGER  |
| CLARK   | 2450.00 | MANAGER  |
+-----+-----+-----+
5 rows in set (0.00 sec)
```

```
mysql> select ename, sal, job
      -> from emp
      -> limit 3,5;
+-----+-----+-----+
| ename | sal   | job    |
+-----+-----+-----+
| BLAKE | 2850.00 | MANAGER |
| CLARK  | 2450.00 | MANAGER |
| SCOTT  | 3000.00 | ANALYST |
| KING   | 5000.00 | PRESIDENT |
| TURNER | 1500.00 | SALESMAN |
+-----+-----+-----+
5 rows in set (0.00 sec)
```

資料排行榜

- ORDER BY 加 LIMIT 子句

```
mysql> select ename, sal, job
-> from emp
-> order by sal desc;
+-----+-----+-----+
| ename | sal   | job    |
+-----+-----+-----+
| KING  | 5000.00 | PRESIDENT |
| SCOTT | 3000.00 | ANALYST   |
| MARY   | 3000.00 | ANALYST   |
| JONES  | 2975.00 | MANAGER   |
| BLAKE  | 2850.00 | MANAGER   |
| CLARK  | 2450.00 | MANAGER   |
| ALLEN  | 2000.00 | SALESMAN  |
| TURNER | 1500.00 | SALESMAN  |
| MILLER | 1300.00 | CLERK     |
| MARTIN | 1250.00 | SALESMAN  |
| ADAMS  | 1100.00 | CLERK     |
+-----+-----+-----+
11 rows in set (0.00 sec)
```

薪資最高前5名

```
mysql> select ename, sal, job
-> from emp
-> order by sal desc
-> limit 5;
+-----+-----+-----+
| ename | sal   | job    |
+-----+-----+-----+
| KING  | 5000.00 | PRESIDENT |
| MARY   | 3000.00 | ANALYST   |
| SCOTT | 3000.00 | ANALYST   |
| JONES  | 2975.00 | MANAGER   |
| BLAKE  | 2850.00 | MANAGER   |
+-----+-----+-----+
5 rows in set (0.00 sec)
```

薪資最低前5名

```
mysql> select ename, sal, job
-> from emp
-> order by sal asc
-> limit 5;
+-----+-----+-----+
| ename | sal   | job    |
+-----+-----+-----+
| ADAMS | 1100.00 | CLERK   |
| MARTIN | 1250.00 | SALESMAN |
| MILLER | 1300.00 | CLERK   |
| TURNER | 1500.00 | SALESMAN |
| ALLEN  | 2000.00 | SALESMAN |
+-----+-----+-----+
5 rows in set (0.00 sec)
```

Module 15.

資料查詢6

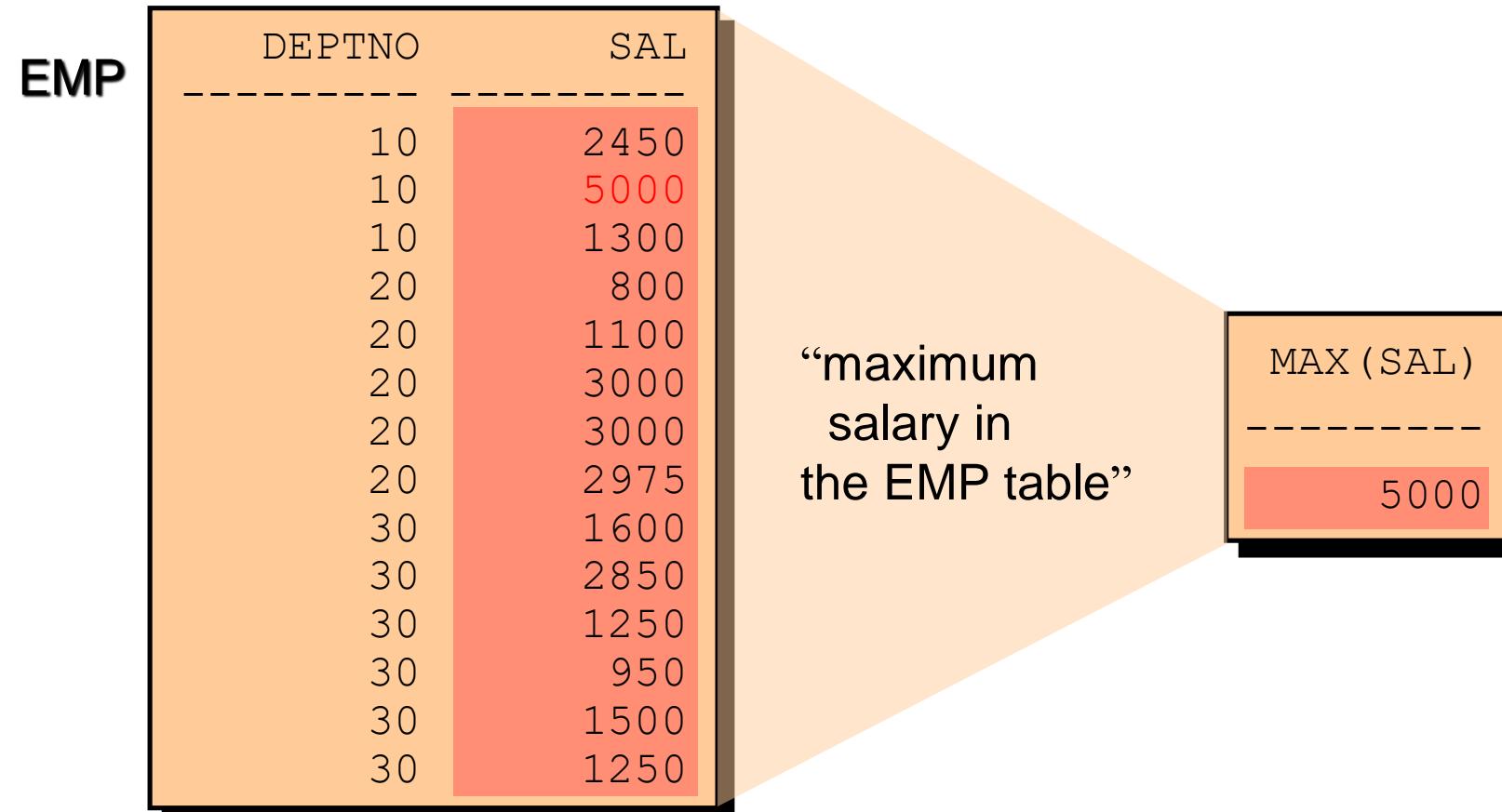
15-1: Aggregate Function, COUNT

15-2: MAX, MIN, SUM, AVG

15-3: GROUP BY Clause

彙總函數(Aggregate Function)

- 彙總函數主要提供資料彙總功能，如：計算平均、資料加總…等
- 多筆記錄(rows)執行一次，傳回一個結果



彙總函數

函數	說明
COUNT(*)	回傳資料的筆數
COUNT(column)	回傳欄位 不為空值 的筆數
COUNT(DISTINCT column)	回傳欄位 去除重複列不為空值 的筆數
MAX(column)	回傳欄位中的最大值
MIN(column)	回傳欄位中的最小值
SUM(column)	回傳欄位的加總
AVG(column)	回傳欄位的平均值

COUNT(*)

- 傳回資料表中的資料列數

```
mysql> SELECT *
      -> FROM emp
      -> WHERE deptno=10;
+-----+-----+-----+-----+-----+-----+-----+-----+
| EMPNO | ENAME  | JOB       | MGR     | HIREDATE | SAL     | COMM    | DEPTNO |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 7839  | KING    | PRESIDENT | NULL    | 1981-11-18 | 5000   | NULL    | 10      |
| 7782  | CLARK   | MANAGER   | 7839   | 1981-06-09 | 2450   | NULL    | 10      |
| 7934  | MILLER  | CLERK    | 7782   | 1982-01-23 | 1300   | NULL    | 10      |
+-----+-----+-----+-----+-----+-----+-----+-----+
mysql> SELECT count(*)
      -> FROM emp
      -> WHERE deptno=10;
+-----+
| count(*) |
+-----+
|      3   |
+-----+
```

COUNT(column|expr)

- 傳回欄位或運算式不為空值的資料列數

```
mysql> SELECT comm
      -> FROM emp
      -> WHERE deptno=30;
+-----+
| comm |
+-----+
| NULL |
| 1400 |
| 300  |
| 0    |
| NULL |
| 500  |
+-----+
```

```
mysql> SELECT COUNT(comm)
      -> FROM emp
      -> WHERE deptno=30;
+-----+
| COUNT(comm) |
+-----+
|          4 |
+-----+
```

Number of columns
(not include null value)

COUNT(DISTINCT column|expr)

- 傳回欄位或運算式中去除重複資料的資料列數, 但不包含空值

```
mysql> SELECT DISTINCT comm
      -> FROM emp
      -> WHERE deptno=30;
+-----+
| comm |
+-----+
| NULL |
| 1400 |
| 300  |
| 0    |
| 500  |
+-----+
```

```
mysql> SELECT COUNT(DISTINCT comm)
      -> FROM emp
      -> WHERE deptno=30;
+-----+
| COUNT(DISTINCT comm) |
+-----+
| 4                   |
+-----+
```

Number of distinct column value
(not include null value)

MAX(column|expr) Function

- 傳回欄位或運算式中最大值

```
mysql> SELECT sal
      ->   FROM emp
      -> WHERE deptno=30;
+-----+
| sal |
+-----+
| 2850 |
| 1250 |
| 1600 |
| 1500 |
| 950  |
| 1250 |
+-----+
```

```
mysql> SELECT MAX(sal)
      ->   FROM emp
      -> WHERE deptno=30;
+-----+
| MAX(sal) |
+-----+
|     2850  |
+-----+
```

註：NULL Value皆忽略不計入

MIN(column|expr) Function

- 傳回欄位或運算式中最小值

```
mysql> SELECT sal
      -> FROM emp
      -> WHERE deptno=30;
+-----+
| sal |
+-----+
| 2850 |
| 1250 |
| 1600 |
| 1500 |
| 950  |
| 1250 |
+-----+
```

```
mysql> SELECT MIN(sal)
      -> FROM emp
      -> WHERE deptno=30;
+-----+
| MIN(sal) |
+-----+
|      950 |
+-----+
```

註：NULL Value皆忽略不計入

SUM(column|expr) Function

- 將欄位或運算式加總(數值資料)

```
mysql> SELECT sal
      -> FROM emp
      -> WHERE deptno=30;
+-----+
| sal |
+-----+
| 2850 |
| 1250 |
| 1600 |
| 1500 |
| 950  |
| 1250 |
+-----+
```

```
mysql> SELECT SUM(sal)
      -> FROM emp
      -> WHERE deptno=30;
+-----+
| SUM(sal) |
+-----+
|    9400  |
+-----+
```

註：NULL Value 皆忽略不計入

AVG(column|expr) Function

- 計算欄位或運算式的平均值(數值資料)

```
mysql> SELECT sal
      ->   FROM emp
      -> WHERE deptno=30;
+-----+
| sal |
+-----+
| 2850 |
| 1250 |
| 1600 |
| 1500 |
| 950  |
| 1250 |
+-----+
```

```
mysql> SELECT AVG(sal)
      ->   FROM emp
      -> WHERE deptno=30;
+-----+
| AVG(sal) |
+-----+
| 1566.6667 |
+-----+
```

註：NULL value皆忽略不計入

AVG(column|expr) Function

- 計算公式

```
AVG(column) = SUM(column) / COUNT(column)
```

```
mysql> SELECT comm
      -> FROM emp
      -> WHERE deptno=30;
+-----+
| comm |
+-----+
| NULL |
| 1400 |
| 300  |
| 0   |
| NULL |
| 500  |
+-----+
```

```
mysql> SELECT AVG(comm)
      -> FROM emp
      -> WHERE deptno=30;
+-----+
| AVG(comm) |
+-----+
| 550.0000 |
+-----+
```

註：NULL value皆忽略不計入

AVG with IFNULL Function

- 平均值計算公式

平均值 (Average) = $\text{SUM}(\text{column}) / \text{COUNT}(\star)$ 或
= $\text{AVG}(\text{IFNULL}(\text{column}|\text{expr}, 0))$

```
mysql> SELECT SUM(comm)/COUNT(*) AVG
      -> FROM emp
      -> WHERE deptno=30;
+-----+
| AVG   |
+-----+
| 366.67 |
+-----+
```

```
mysql> SELECT AVG(IFNULL(comm,0)) AVG
      -> FROM emp
      -> WHERE deptno=30;
+-----+
| AVG   |
+-----+
| 366.6667 |
+-----+
```

資料彙總

```
mysql> SELECT SUM(SAL), MIN(SAL), MAX(SAL), AVG(SAL), COUNT(*)
-> FROM emp WHERE deptno=30;
+-----+-----+-----+-----+-----+
| SUM(SAL) | MIN(SAL) | MAX(SAL) | AVG(SAL) | COUNT(*) |
+-----+-----+-----+-----+-----+
|      9400 |       950 |     2850 | 1566.6667 |        6 |
+-----+-----+-----+-----+-----+
```

GROUP BY 子句

子句(Element)	Expression	Role
SELECT	<select list>	給定查詢的資料項目 Defines which columns to return
FROM	<table source>	給定資料來源 Defines table(s) to query
WHERE	<search condition>	給定查詢/過濾資料條件 Filters rows using a predicate
GROUP BY	<group by list>	資料分組設定 Arranges rows by groups
HAVING	<search condition>	給定分組資料查詢/過濾條件 Filters groups using a predicate
ORDER BY	<order by list>	給定查詢結果排序方式 Sorts the output

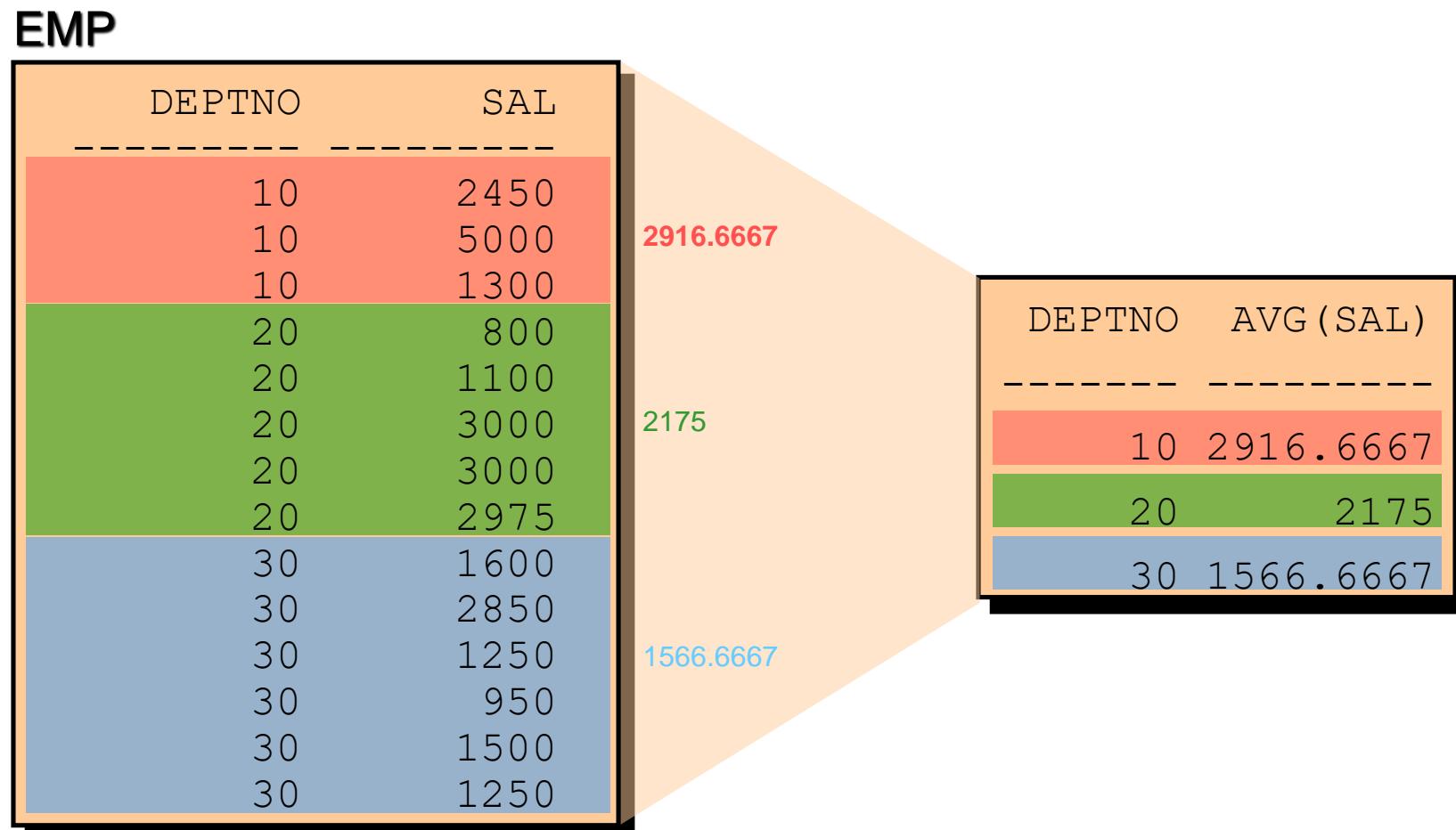
資料分組(Grouping Data)

- 依資料內容來分組，分組後再做資料彙總

deptno	sal		deptno	SUM(sal)
10	5000	部門 10	10	8750
10	1300			
10	2450			
20	2975			
20	1100			
20	3000	部門 20	20	10875
20	800			
20	3000			
30	1250			
30	2850			
30	1500	部門 30	30	9400
30	1600			
30	1250			
30	950			

GROUP BY 子句

- 使用 **GROUP BY** 子句將表格(table)中的記錄(rows)依資料內容分為不同的群組
- 若無 **GROUP BY** 子句，則整個表格就是一組



GROUP BY 子句

- ▶ GROUP BY子句將表格(table)中的記錄(rows)依 GROUP BY後資料項的相同內容分為一個群組

```
SELECT      column, group_function
FROM        table
[WHERE      condition]
[GROUP BY   group_by_expression]
[HAVING     group_condition]
[ORDER BY   column];
```

- ▶ GROUP BY 分組後再做各組的資料彙總
 - *group_by_expression* 即為排序的依據(ASC)
 - 可以使用ORDER BY改變預設排序的結果

```
SELECT      deptno, AVG(sal) AS AvgSal
FROM        emp
GROUP BY   deptno;
```

資料分組使用GROUP BY 子句

```
mysql> SELECT deptno, SUM(sal)
-> FROM emp
-> GROUP BY deptno;
+-----+-----+
| deptno | SUM(sal) |
+-----+-----+
|      10 |     8750 |
|      20 |    10875 |
|      30 |     9400 |
+-----+-----+
```

有排序(升冪ASC)

```
mysql> SELECT deptno, SUM(sal)
-> FROM emp;
ERROR 1140 (42000): Mixing of GROUP columns (MIN(),MAX(),COUNT())...
with no GROUP columns is illegal if there is no GROUP BY clause
```

資料分組 - NULL

- ▶ NULL值在 GROUP BY 子句的資料分組

```
mysql> SELECT comm, COUNT(*)
-> FROM emp
-> GROUP BY comm;
+-----+-----+
| comm | COUNT(*) |
+-----+-----+
| NULL |      10 |
| 0    |       1 |
| 300  |       1 |
| 500  |       1 |
| 1400 |       1 |
+-----+-----+
```

NULL 自己也算資料分組的一組

資料分組 – 使用ORDER BY

- ▶ 改變預設排序的結果

```
mysql> SELECT deptno, SUM(sal)
-> FROM emp
-> GROUP BY deptno
-> ORDER BY 2;
+-----+-----+
| deptno | SUM(sal) |
+-----+-----+
|      10 |     8750 |
|      30 |     9400 |
|      20 |    10875 |
+-----+-----+
3 rows in set (0.00 sec)
```



升冪 ASC

資料分組-多欄位(Grouping of Multiple Column)

- 分成更小的群組資料

```
mysql> SELECT deptno, job, count(*)  
      -> FROM emp  
      -> GROUP BY deptno, job  
      -> ORDER BY deptno;
```

deptno	job	count(*)
10	CLERK	1
10	MANAGER	1
10	PRESIDENT	1
20	ANALYST	2
20	CLERK	2
20	MANAGER	1
30	CLERK	1
30	MANAGER	1
30	SALESMAN	4

總人員應和 COUNT(*)= 14 相同

篩選分組資料(Filtering Group Results)

EMP

DEPTNO	SAL
10	2450
10	5000
10	1300
20	800
20	1100
20	3000
20	3000
20	2975
30	1600
30	2850
30	1250
30	950
30	1500
30	1250

5000

3000

2850

“maximum salary per department greater than \$2900”

DEPTNO	MAX (SAL)
10	5000
20	3000

```
SELECT deptno, max(sal) SalMax
FROM emp
GROUP BY deptno
HAVING max(sal)>2900;
```

分組資料過濾條件(HAVING子句)

- ▶ 使用 HAVING 子句設定條件來篩選回傳的分組資料

```
SELECT      column, group_function
FROM        table
[WHERE       condition]
[GROUP BY   group_by_expression]
[HAVING     group_condition]
[ORDER BY   column];
```

- WHERE子句中不可以出現群組函數

使用 HAVING 子句篩選分組資料

- ▶ 依職務分類，列出人數>3的類別

```
mysql> SELECT job, COUNT(*) CNT
      -> FROM emp
      -> GROUP BY job
      -> ORDER BY CNT;
```

job	CNT
PRESIDENT	1
ANALYST	2
MANAGER	3
SALESMAN	4
CLERK	4

```
mysql> SELECT job, COUNT(*) CNT
      -> FROM emp
      -> GROUP BY job
      -> HAVING COUNT(*) > 3;
```

job	CNT
CLERK	4
SALESMAN	4

篩選分組資料的錯誤(Error in Restricting)

- WHERE子句中不可以出現群組函數

```
mysql> SELECT job, COUNT(*) CNT
->     FROM emp
-> WHERE COUNT(*) > 3
-> GROUP BY job;
ERROR 1111 (HY000): Invalid use of group function
```

分組資料的字串連結函數

- GROUP_CONCAT 群組函數

```
GROUP_CONCAT([DISTINCT] expr [,expr ...]
              [ORDER BY {unsigned_integer | column | expr}
                [ASC | DESC] [,column ...]])
              [SEPARATOR string])
```

- | | |
|--------------------|------------|
| • DISTINCT | 去除連結字串重複資料 |
| • ORDER BY | 排序連結字串 |
| • unsigned_integer | 無負號 |
| • column, expr | 欄位名稱或運算式 |
| • SEPARATOR | 設定連結字串間隔 |
| • string | 連結字串間隔字串 |

分組資料的字串連結函數

- ▶ 使用 GROUP_CONCAT 群組函數列出各部門所有的職務列表

```
mysql> SELECT deptno, GROUP_CONCAT(job SEPARATOR ',') JOBS
-> FROM emp
-> GROUP BY deptno;
+-----+-----+
| deptno | JOBS
+-----+-----+
|      10 | PRESIDENT,CLERK,MANAGER
|      20 | MANAGER,CLERK,ANALYST,CLERK,ANALYST
|      30 | SALESMAN,MANAGER,SALESMAN,SALESMAN,SALESMAN,CLERK
+-----+-----+
```

分組資料的字串連結函數

- ▶ 列出和部門有關的職務(去除重複資料+排序)

```
mysql> SELECT deptno,
      ->   GROUP_CONCAT(DISTINCT job ORDER BY job ASC SEPARATOR ',') JOBS
      ->   FROM emp
      ->   GROUP BY deptno;
+-----+-----+
| deptno | JOBS           |
+-----+-----+
|      10 | CLERK,MANAGER,PRESIDENT |
|      20 | ANALYST,CLERK,MANAGER  |
|      30 | CLERK,MANAGER,SALESMAN |
+-----+-----+
3 rows in set (0.01 sec)
```

Module 16.

資料連結1

16-1: Join的概念

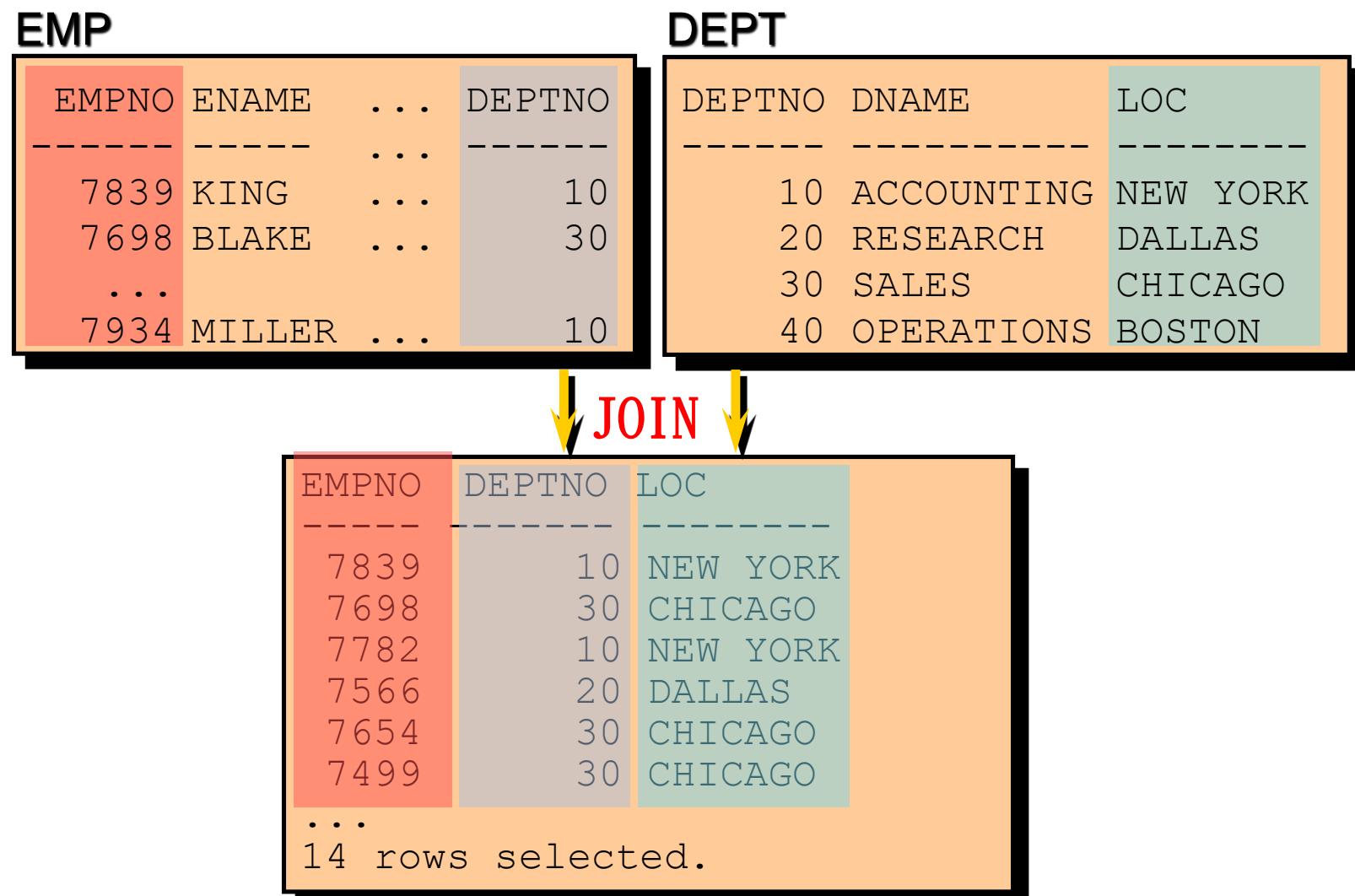
16-2: Cross Join

16-3: Inner join-Equal join

FROM 子句

子句(Element)	Expression	Role
SELECT	<select list>	給定查詢的資料項目 <small>Defines which columns to return</small>
FROM	<table source>	給定資料來源 <small>Defines table(s) to query</small>
WHERE	<search condition>	給定查詢/過濾資料條件 <small>Filters rows using a predicate</small>
GROUP BY	<group by list>	資料分組設定 <small>Arranges rows by groups</small>
HAVING	<search condition>	給定分組資料查詢/過濾條件 <small>Filters groups using a predicate</small>
ORDER BY	<order by list>	給定查詢結果排序方式 <small>Sorts the output</small>

從多個資料表查詢資料



連結多個表格的運算

- ▶ 連結查詢-用來查詢多個資料表中的資料
 - 將多個資料表中的資料利用連結運算(Join)結合成一個虛擬表格(Virtual Table)
- ▶ 直積運算(Cartesian Product)
 - 合併二個表格中的所有資料

Name	Product
Davis	Alice Mutton
Funk	Crab Meat
King	Ipooh Coffee

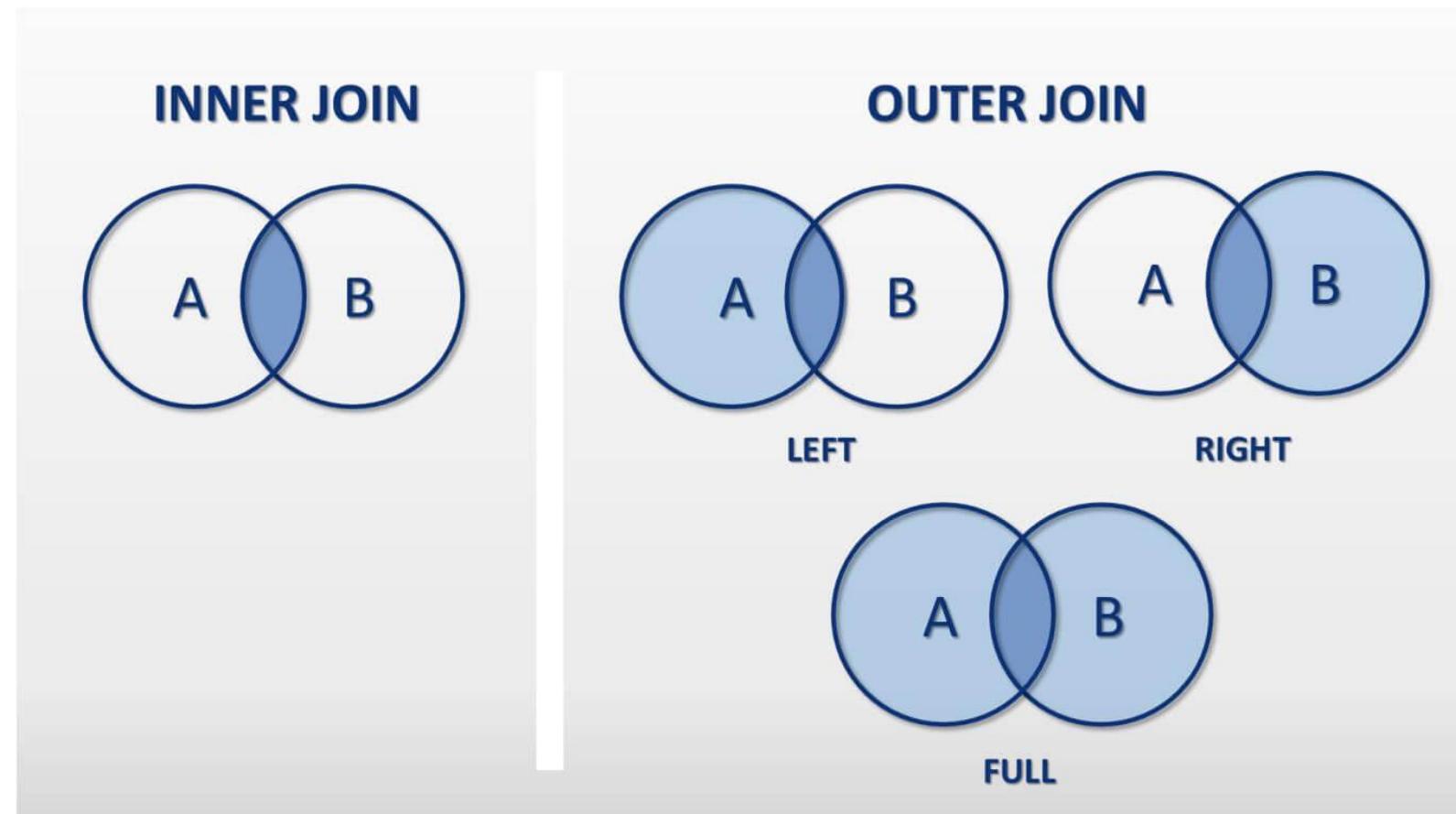
X 個欄位  Y 個欄位  X+Y 個欄位
M 筆記錄  N 筆記錄  M*N 筆記錄

Name	Product
Davis	Alice Mutton
Davis	Crab Meat
Davis	Ipooh Coffee
Funk	Alice Mutton
Funk	Crab Meat
Funk	Ipooh Coffee
King	Alice Mutton
King	Crab Meat
King	Ipooh Coffee

連結運算方式(Join Types)

- ▶ 交叉連結 Cross Join
- ▶ 內部連結 Inner Join (又稱一般連結)
 - 自然連結 Natural Join
 - 相等連結 Equal Joins
 - 不相等連結 Non-Equal Joins
- ▶ 外部連結 Outer Join
 - 左邊外部連結 Left Outer Joins
 - 右邊外部連結 Right Outer Joins
 - 完全外部連結 Full Outer Joins(5.1才支援)
- ▶ 自我連結 Self Joins

連結的比較



交叉連結(Cross Join)

- ▶ 無條件連結
 - 第一個資料表所有的rows會與第二個資料表中每一個row合併(Cartesian Product)
- ▶ ANSI SQL-92 語法:

```
SELECT ...
FROM table1 CROSS JOIN table2
...
```

- ▶ ANSI SQL-89 語法:

```
SELECT ...
FROM table1, table2
...
```

交叉連結(Cross Join)

- ▶ 使用二個表格的資料來產生測試用資料

```
SELECT *
FROM emp e1 CROSS JOIN dept d1;
```

```
SELECT e1.ename, e2.job
FROM emp e1 CROSS JOIN emp e2;
```

不明確的欄位名稱(Ambiguous Column Names)

- ▶ 若一個以上的表格中有相同的欄位名稱
 - 附加表格名稱來分辨相同的欄位名稱

Table_prefixes.Column_Name

- ▶ **附加表格名稱可提升執行效能**

```
SELECT ename, sal, deptno, dname
FROM dept JOIN emp ON dept.deptno = emp.deptno;
```

```
SELECT emp.ename, emp.sal, emp.deptno, dept.dname
FROM dept JOIN emp ON dept.deptno = emp.deptno;
```

- ▶ 表格別名(table alias)

```
SELECT e.ename, e.sal, e.deptno, d.dname
FROM dept d JOIN emp e ON d.deptno = e.deptno;
```

內部連結(Inner Join)

- ▶ 產生只有符合條件連結(join conditions)的資料
 - 第一個資料表所有的rows會與第二個資料表中每一個row做連結條件測試，結果為真(TRUE)才會合併產生一筆新紀錄(row)
 - 如何設定連結條件
 - SQL-92 語法使用 **ON** 子句 (preferred)
 - SQL-89 語法使用 **WHERE** 子句
- ▶ 為何使用 ON 子句來設定連結條件?
 - 可以跟資料查詢條件分開來以免造成混淆

自然連結(Natural Join)

- ▶ 連結條件(Join Condition)
 - 使用表格內之同名欄位作為連結條件

```
SELECT ...  
FROM table1 NATURAL JOIN table2  
...
```

- 若欄位型態不同時，則產生錯誤訊息

```
SELECT e.ename, e.job, e.sal, e.deptno, d.dname, d.loc  
FROM emp e NATURAL JOIN dept d;
```



```
SELECT e.ename, e.job, e.sal, e.deptno, d.dname, d.loc  
FROM emp e JOIN dept d ON(e.deptno = d.deptno);
```

自然連結(Natural Join)

▶ ANSI SQL-92

```
mysql> SELECT a.empno, a.ename, a.mgr, a.sal, a.deptno, b.deptno, b.dname
-> FROM emp a NATURAL JOIN dept b
-> WHERE a.deptno = 10;
+-----+-----+-----+-----+-----+-----+
| empno | ename | mgr  | sal   | deptno | deptno | dname    |
+-----+-----+-----+-----+-----+-----+
|  7839 | KING  | NULL | 5000 |      10 |      10 | ACCOUNTING |
|  7782 | CLARK | 7839 | 2450 |      10 |      10 | ACCOUNTING |
|  7934 | MILLER | 7782 | 1300 |      10 |      10 | ACCOUNTING |
+-----+-----+-----+-----+-----+-----+
```

▶ ANSI SQL-89

```
SELECT a.empno, a.ename, a.mgr, a.sal, a.deptno, b.deptno, b.dname
FROM emp a, dept b
WHERE a.deptno = b.deptno
      AND a.deptno = 10;
```

與上述結果相同

自然連結(Natural Join)- Using 子句

- 在相同欄位中指定欄位做為連結條件

```
mysql> SELECT a.empno, a.ename, a.mgr, a.sal, a.deptno, b.deptno, b.dname
-> FROM emp a JOIN dept b USING(deptno)
-> WHERE a.deptno = 10;
+-----+-----+-----+-----+-----+-----+
| empno | ename | mgr | sal | deptno | deptno | dname |
+-----+-----+-----+-----+-----+-----+
| 7839 | KING | NULL | 5000 | 10 | 10 | ACCOUNTING |
| 7782 | CLARK | 7839 | 2450 | 10 | 10 | ACCOUNTING |
| 7934 | MILLER | 7782 | 1300 | 10 | 10 | ACCOUNTING |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

不可使用關係名稱

```
mysql> SELECT a.empno, a.ename, a.mgr, a.sal, a.deptno, b.deptno, b.dname
-> FROM emp a JOIN dept b USING(a.deptno)
-> WHERE a.deptno = 10;
ERROR 1064 (42000): You have an error in your SQL syntax. Check the
manual that corresponds to your MySQL server version for the right syntax
to use near '.deptno'
WHERE a.deptno = 10' at line 3mysql>
```

內部連結(Inner Join)語法

- ▶ 將要連結的表格名稱列在 FROM 子句中，並加入 JOIN 運算子及 ON 來設定連結條件
- ▶ 紿定表格別名(Table aliases preferred)
- ▶ 表格先後順序無關

```
SELECT ...
FROM t1 JOIN t2 ON JoinCondions
...
```

```
SELECT e.ename, e.deptno, d.dname
FROM dept d JOIN emp e ON (d.deptno = e.deptno);
```

```
SELECT e.ename, e.deptno, d.dname
FROM dept d , emp e
WHERE d.deptno = e.deptno;
```

內部連結(Inner Join)

▶ 使用表格全名

```
mysql> SELECT emp.empno, emp.ename, emp.mgr, emp.sal, emp.deptno,
->           dept.deptno, dept.dname
-> FROM emp join dept on emp.deptno = dept.deptno;
```

empno	ename	mgr	sal	deptno	deptno	dname
7839	KING	NULL	5000	10	10	ACCOUNTING
7782	CLARK	7839	2450	10	10	ACCOUNTING
7934	MILLER	7782	1300	10	10	ACCOUNTING
7566	JONES	7839	2975	20	20	RESEARCH
7902	FORD	7566	3000	20	20	RESEARCH
7369	SMITH	7902	800	20	20	RESEARCH
7788	SCOTT	7566	3000	20	20	RESEARCH
7876	ADAMS	7788	1100	20	20	RESEARCH
7698	BLAKE	7839	2850	30	30	SALES
7654	MARTIN	7698	1250	30	30	SALES
7499	ALLEN	7698	1600	30	30	SALES
7844	TURNER	7698	1500	30	30	SALES
7900	JAMES	7698	950	30	30	SALES
7521	WARD	7698	1250	30	30	SALES

內部連結(Inner Join)

- ▶ 建立並使用表格別名

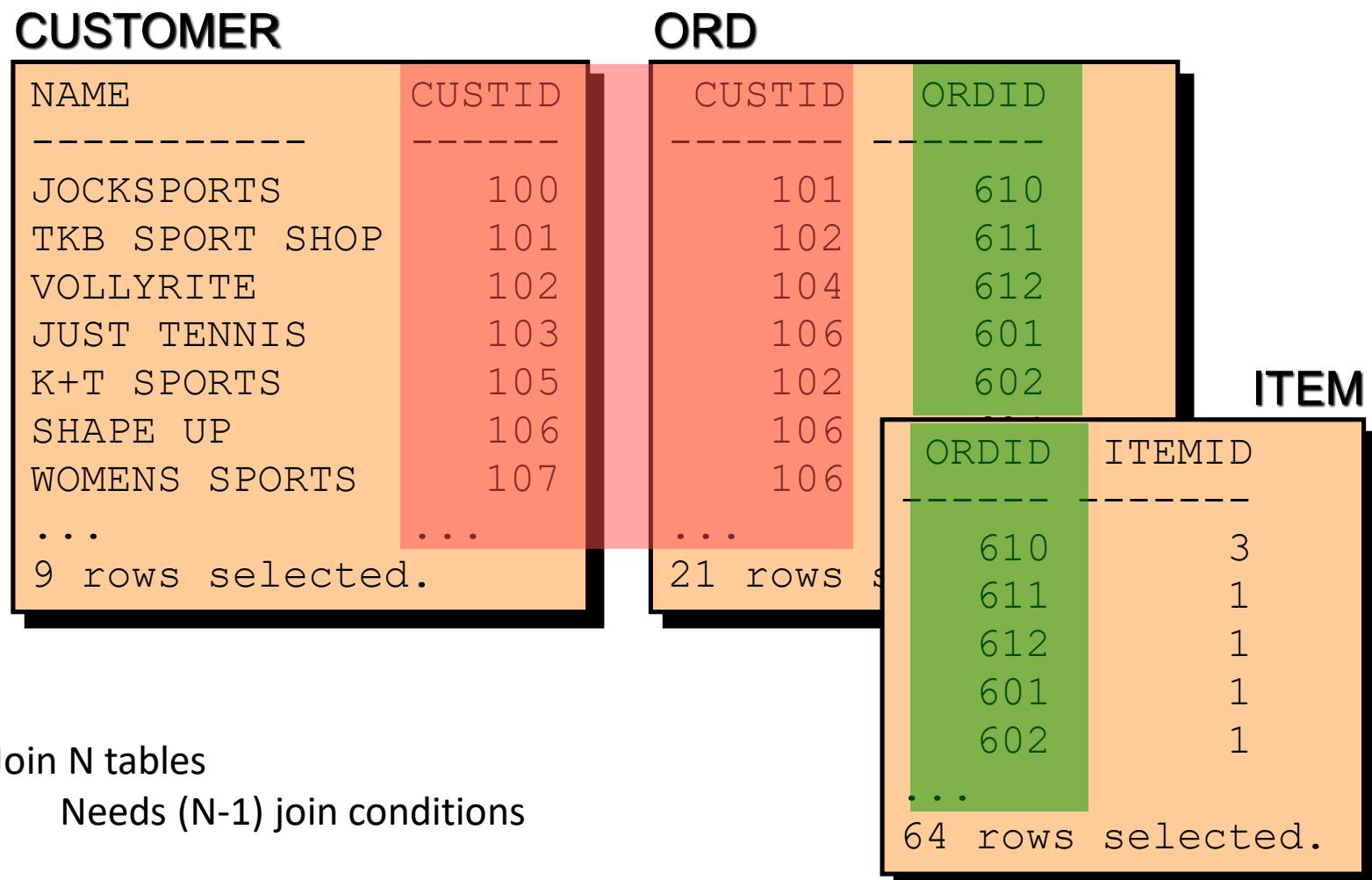
```
mysql> SELECT a.empno, a.ename, a.mgr, a.sal, a.deptno, b.deptno, b.dname
-> FROM emp a JOIN dept b ON (a.deptno = b.deptno);
+-----+-----+-----+-----+-----+-----+
| empno | ename  | mgr   | sal   | deptno | deptno | dname   |
+-----+-----+-----+-----+-----+-----+
| 7839  | KING    | NULL  | 5000  |      10 |      10 | ACCOUNTING |
| 7782  | CLARK   | 7839  | 2450  |      10 |      10 | ACCOUNTING |
| 7934  | MILLER  | 7782  | 1300  |      10 |      10 | ACCOUNTING |
| 7566  | JONES   | 7839  | 2975  |      20 |      20 | RESEARCH   |
| 7902  | FORD    | 7566  | 3000  |      20 |      20 | RESEARCH   |
| 7369  | SMITH   | 7902  | 800   |      20 |      20 | RESEARCH   |
| 7788  | SCOTT   | 7566  | 3000  |      20 |      20 | RESEARCH   |
| 7876  | ADAMS   | 7788  | 1100  |      20 |      20 | RESEARCH   |
| 7698  | BLAKE   | 7839  | 2850  |      30 |      30 | SALES     |
| 7654  | MARTIN  | 7698  | 1250  |      30 |      30 | SALES     |
| 7499  | ALLEN   | 7698  | 1600  |      30 |      30 | SALES     |
| 7844  | TURNER  | 7698  | 1500  |      30 |      30 | SALES     |
| 7900  | JAMES   | 7698  | 950   |      30 |      30 | SALES     |
| 7521  | WARD    | 7698  | 1250  |      30 |      30 | SALES     |
+-----+-----+-----+-----+-----+-----+
14 rows in set (0.00 sec)
```

內部連結(Inner Join)-查詢條件

- ▶ 列出員工姓名KING所在的部門資訊

```
mysql> SELECT a.empno, a.ename, a.mgr, a.sal, a.deptno, b.deptno, b.dname
-> FROM emp a JOIN dept b ON a.deptno = b.deptno
-> WHERE a.ename = 'KING';
+-----+-----+-----+-----+-----+-----+
| empno | ename | mgr  | sal   | deptno | deptno | dname
+-----+-----+-----+-----+-----+-----+
|  7839 | KING  | NULL | 5000 |      10 |      10 | ACCOUNTING |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.01 sec)
```

多個資料表的連結運算



多個資料表的連結運算

► 語法：

```
SELECT ...
FROM t1 JOIN t2 ON t1.columnA = t2.columnA
          JOIN t3 ON t2.columnB = t3.columnB
          JOIN t4 ON t3.columnB = t4.columnB
...
```

```
SELECT c.name, c.custid, o.ordid, i.itemid
FROM customer c JOIN ord o ON c.custid=o.custid
                  JOIN item i ON o.ordid=i.ordid;
```

```
SELECT c.name, c.custid, o.ordid, i.itemid
FROM customer c, ord o, item i
WHERE c.custid=o.custid AND o.ordid=i.ordid ;
```

多個資料表的連結運算

- ▶ 多個資料表的連結(計算員工業績)



```
mysql> SELECT a.empno, a.ename, sum(c.total) total
-> FROM emp a JOIN customer b ON a.empno = b.repid
->           JOIN ord c ON b.custid = c.custid
-> GROUP BY a.empno, a.ename;
+-----+-----+-----+
| empno | ename  | total   |
+-----+-----+-----+
| 7499  | ALLEN  | 7870.80 |
| 7654  | MARTIN | 27775.50 |
| 7844  | TURNER | 58050.90 |
+-----+-----+-----+
3 rows in set (0.08 sec)
```

Module 17.

資料連結2

17-1: Inner join–Non-Equal join

17-2: Outer Join與 UNION

17-3: Self Join

不相等的連結(Non-Equijoins)

- ▶ 連結條件不是使用等號(=)
- ▶ 列出所有員工薪資的等級

EMP

EMPNO	ENAME	SAL
7839	KING	5000
7698	BLAKE	2850
7782	CLARK	2450
7566	JONES	2975
7654	MARTIN	1250
7499	ALLEN	1600
7844	TURNER	1500
7900	JAMES	950
...		
14 rows selected.		

SALGRADE

GRADE	LOSAL	HISAL
1	700	1200
2	1201	1400
3	1401	2000
4	2001	3000
5	3001	9999

“salary in the EMP
table is between
low salary and high
salary in the SALGRADE
table”

不相等的連結(Non-Equijoins)

- ▶ 連結條件的運算子(Operators)
 - \geq 或 \leq
 - BETWEEN AND

```
SELECT a.empno, a.ename, a.sal, b.grade
FROM emp a join salgrade b on
(a.sal BETWEEN b.losal AND b.hisal);
```

- ▶ 若符合條件的資料超過一筆以上時，因向量積關係，資料可能會不正確。

不相等的連結(Non-Equijoins)

- ▶ 列出所有員工薪資的等級

```
mysql> SELECT a.empno, a.ename, a.sal, b.grade
-> FROM emp a JOIN salgrade b ON a.sal BETWEEN b.losal AND b.hisal;
+-----+-----+-----+
| empno | ename  | sal   | grade |
+-----+-----+-----+
| 7900  | JAMES   | 950   | 1     |
| 7369  | SMITH   | 800   | 1     |
| 7876  | ADAMS   | 1100  | 1     |
| 7654  | MARTIN  | 1250  | 2     |
| 7521  | WARD    | 1250  | 2     |
| 7934  | MILLER  | 1300  | 2     |
| 7499  | ALLEN   | 1600  | 3     |
| 7844  | TURNER  | 1500  | 3     |
| 7698  | BLAKE   | 2850  | 4     |
| 7782  | CLARK   | 2450  | 4     |
| 7566  | JONES   | 2975  | 4     |
| 7902  | FORD    | 3000  | 4     |
| 7788  | SCOTT   | 3000  | 4     |
| 7839  | KING    | 5000  | 5     |
+-----+-----+-----+
14 rows in set (0.18 sec)
```

外部連結(Outer Join)

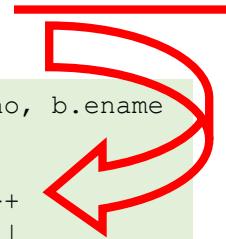
DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

```
mysql> SELECT a.deptno, a.dname, b.empno, b.ename
-> FROM dept a JOIN emp b
->      ON a.deptno = b.deptno;
```

deptno	dname	empno	ename
10	ACCOUNTING	7839	KING
10	ACCOUNTING	7782	CLARK
10	ACCOUNTING	7934	MILLER
20	RESEARCH	7566	JONES
20	RESEARCH	7902	FORD
20	RESEARCH	7369	SMITH
20	RESEARCH	7788	SCOTT
20	RESEARCH	7876	ADAMS
30	SALES	7698	BLAKE
30	SALES	7654	MARTIN
30	SALES	7499	ALLEN
30	SALES	7844	TURNER
30	SALES	7900	JAMES
30	SALES	7521	WARD

14 rows in set (0.00 sec)

無40部門之員工



deptno	empno	ename
10	7839	KING
10	7934	MILLER
10	7782	CLARK
20	7566	JONES
20	7876	ADAMS
20	7788	SCOTT
20	7369	SMITH
20	7902	FORD
30	7521	WARD
30	7698	BLAKE
30	7844	TURNER
30	7499	ALLEN
30	7654	MARTIN
30	7900	JAMES

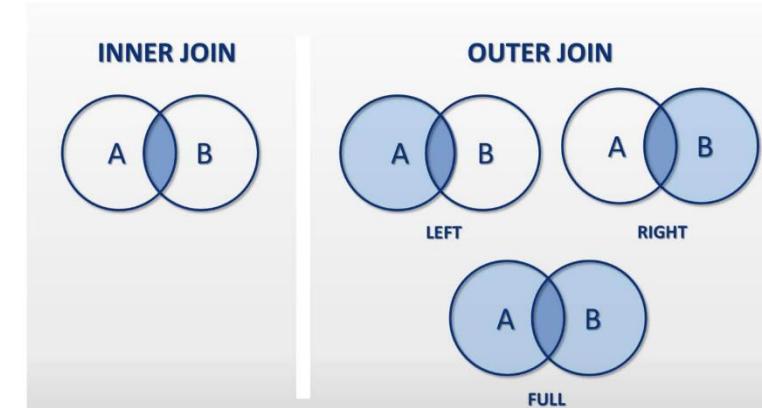
此連結為相等連結，40部門無法顯示

外部連結(Outer Join)

▶ 外部連結

```
SELECT alias.column, alias.column, ...
FROM table1 [AS] alias1
    { [LEFT|RIGHT|FULL [OUTER]] JOIN }
        table2 [AS] alias2
    ON alias1.column = alias2.column
WHERE ...;
```

- 指定一資料表，當連結條件符合時則顯示，若與第二個資料表都不符合時則以null value顯示
- OUTER可以省略不寫
- Types
 - LEFT OUTER JOIN
 - RIGHT OUTER JOIN
 - FULL OUTER JOIN



外部連結(Outer Join)

▶ 列出所有部門下的員工

```
mysql> SELECT a.deptno, a.dname, b.empno, b.ename
-> FROM dept a LEFT OUTER JOIN emp b
->      ON a.deptno = b.deptno;
```

deptno	dname	empno	ename
10	ACCOUNTING	7839	KING
10	ACCOUNTING	7782	CLARK
10	ACCOUNTING	7934	MILLER
20	RESEARCH	7566	JONES
20	RESEARCH	7902	FORD
20	RESEARCH	7369	SMITH
20	RESEARCH	7788	SCOTT
20	RESEARCH	7876	ADAMS
30	SALES	7698	BLAKE
30	SALES	7654	MARTIN
30	SALES	7499	ALLEN
30	SALES	7844	TURNER
30	SALES	7900	JAMES
30	SALES	7521	WARD
40	OPERATIONS	NULL	NULL

Not match的資料以NULL來表示

外部連結(Outer Join)

- ▶ 列出沒有員工的部門

```
mysql> SELECT e.ename, e.deptno, d.dname, d.loc
-> FROM emp e RIGHT OUTER JOIN dept d
->           ON e.deptno = d.deptno
-> WHERE e.empno is NULL;
+-----+-----+-----+-----+
| ename | deptno | dname      | loc       |
+-----+-----+-----+-----+
| NULL  |     NULL | OPERATIONS | BOSTON   |
+-----+-----+-----+-----+
1 row in set (0.01 sec)
```

合併查詢(UNION)

- ▶ 把多個查詢結果，合併到一個結果

```
SELECT ...
UNION [DISTINCT | ALL]
SELECT ...
UNION [DISTINCT | ALL]
SELECT ...
```

```
mysql> SELECT * FROM emp where ENAME like 'A%' UNION SELECT * FROM emp where ENAME like 'S%';
+-----+-----+-----+-----+-----+-----+-----+-----+
| EMPNO | ENAME | JOB      | MGR    | HIREDATE          | SAL     | COMM    | DEPTNO |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 7499  | ALLEN | SALESMAN | 7698   | 1981-02-20 00:00:00 | 1600.00 | 300.00  | 30     |
| 7876  | ADAMS  | CLERK    | 7788   | 1983-01-12 00:00:00 | 1100.00 | NULL    | 20     |
| 7369  | SMITH  | CLERK    | 7902   | 1980-12-17 00:00:00 | 800.00  | NULL    | 20     |
| 7788  | SCOTT  | ANALYST  | 7566   | 1982-12-09 00:00:00 | 3000.00 | NULL    | 20     |
+-----+-----+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

自我連結(Self Joins)

- ▶ 使用同一表格做連結運算
- ▶ 使用表格別名來建立二個相同資料表的連結運算

```
SELECT ...
FROM Table1 A1 JOIN Table1 A2 ON A1.Col1 = A2.Col2
...
```

EMP (WORKER)			EMP (MANAGER)	
EMPNO	ENAME	MGR	EMPNO	ENAME
7839	KING		7839	KING
7698	BLAKE	7839	7839	KING
7782	CLARK	7839	7839	KING
7566	JONES	7839	7839	KING
7654	MARTIN	7698	7698	BLAKE
7499	ALLEN	7698	7698	BLAKE

“MGR in the WORKER table is equal to EMPNO in the MANAGER table”

自我連結(Self Joins)

- ▶ 列出員工與他的主管的資訊

empno	ename	mgr
7839	KING	NULL
7788	SCOTT	7566
7902	FORD	7566
7521	WARD	7698
7900	JAMES	7698
7844	TURNER	7698
7499	ALLEN	7698
7654	MARTIN	7698
7934	MILLER	7782
7876	ADAMS	7788
7566	JONES	7839
7782	CLARK	7839
7698	BLAKE	7839
7369	SMITH	7902

員工

empno	ename
7369	SMITH
7499	ALLEN
7521	WARD
7654	MARTIN
7698	BLAKE
7782	CLARK
7788	SCOTT
7839	KING
7844	TURNER
7876	ADAMS
7900	JAMES
7902	FORD
7934	MILLER

主管

你將發現資料少了一列， Why?

```
mysql> SELECT a.empno, a.ename,
->           a.mgr, b.ename
->      FROM emp a, emp b
->     WHERE a.mgr = b.empno
->    ORDER BY a.mgr;
+-----+-----+-----+-----+
| empno | ename | mgr   | ename |
+-----+-----+-----+-----+
| 7788  | SCOTT | 7566  | JONES |
| 7902  | FORD   | 7566  | JONES |
| 7521  | WARD   | 7698  | BLAKE  |
| 7900  | JAMES  | 7698  | BLAKE  |
| 7844  | TURNER | 7698  | BLAKE  |
| 7499  | ALLEN  | 7698  | BLAKE  |
| 7654  | MARTIN | 7698  | BLAKE  |
| 7934  | MILLER | 7782  | CLARK  |
| 7876  | ADAMS  | 7788  | SCOTT  |
| 7566  | JONES  | 7839  | KING   |
| 7782  | CLARK  | 7839  | KING   |
| 7698  | BLAKE  | 7839  | KING   |
| 7369  | SMITH  | 7902  | FORD   |
+-----+-----+-----+-----+
13 rows in set (0.00 sec)
```

Module 18.

資料連結3

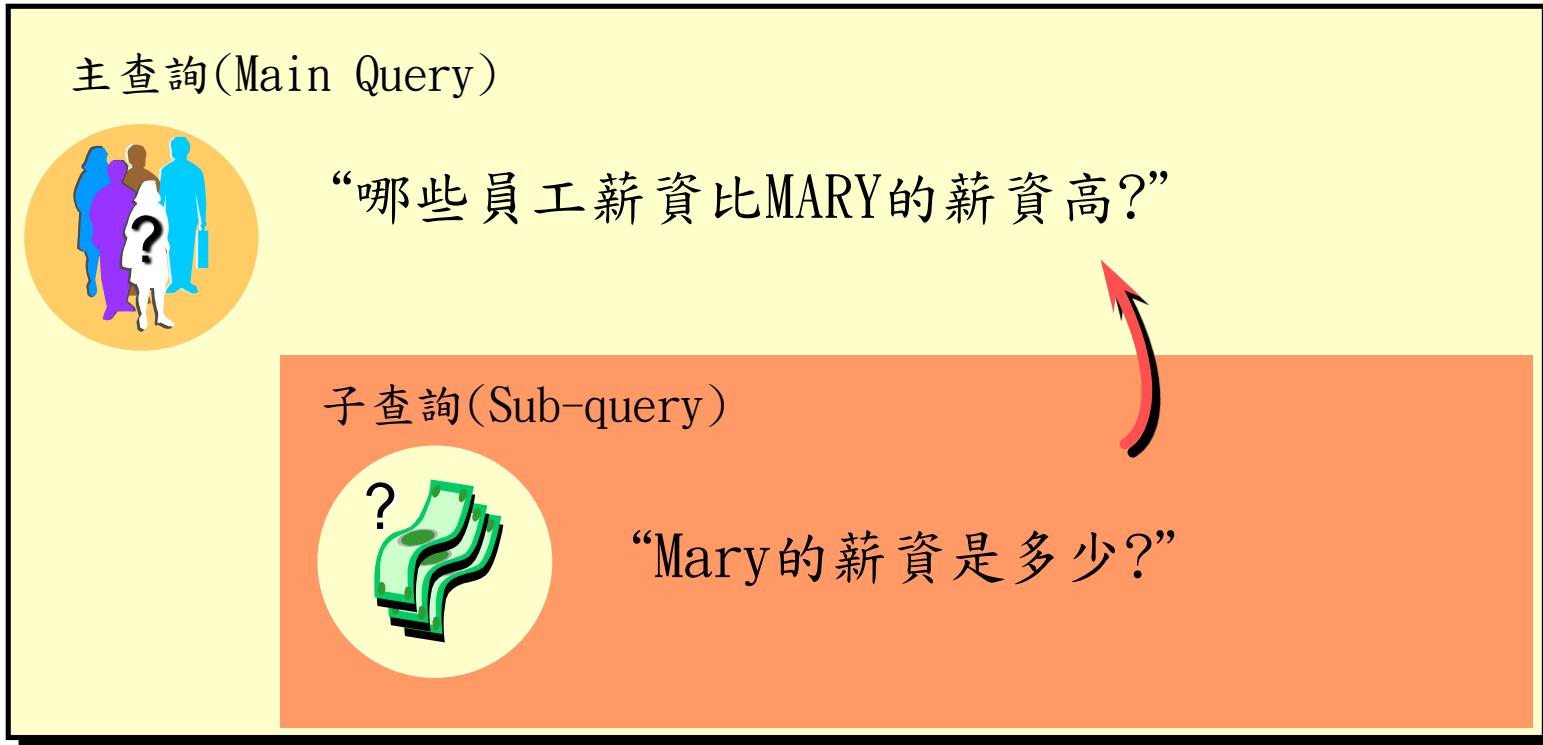
18-1: 子查詢(Subquery)

18-2: Single-Row Subquery語法

18-3: Single-Row Subquery範例

子查詢(Sub-Queries)

- ▶ 利用子查詢來解決另類查詢問題
 - 請找出薪資比MARY高的所有員工資料

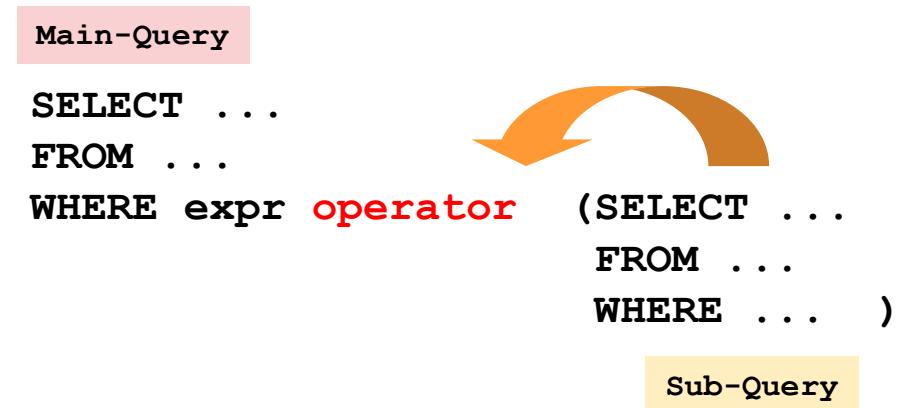


子查詢(Sub-Queries)

- ▶ 子查詢是巢狀查詢:查訥中有查訥
- ▶ 內部查訥(inner query)的結果會傳給外部查訥(outer query)
 - 內部查訥結果當成外部查訥運算式中的運算元
- ▶ 子查詢的種類(依子查詢執行方式分類):
 - 自主子查詢(Self-contained subqueries):與主查訥沒有相關性
 - 相關子查詢(Correlated subqueries):需依賴外部查訥的值
- ▶ 子查詢的型式(依子查詢結果分類):
 - 純量(scalar)
 - 多值(multi-valued)
 - 多欄位(multiple-column)或表格值(table-valued)

子查詢(Sub-Queries)

- ▶ 主查詢中含有另一個查詢(SELECT)
- ▶ 查詢時先執行子查詢，並將結果傳回主查詢



- ▶ 子查詢可以出現的位置
 - WHERE 子句
 - HAVING 子句
 - FROM 子句

子查詢的型式(Types of Subqueries)

- ▶ 單筆記錄子查詢(Single-row subquery)



- ▶ 多筆記錄子查詢(Multiple-row subquery)



- ▶ 表格子查詢(table-valued subquery)



子查詢(Sub-Queries)

- ▶ 子查詢需用()括住
- ▶ 寫在運算子的右邊
- ▶ 子查詢類型
 - 單筆記錄子查詢(Single Row SubQueries)
 - 多筆記錄子查詢(Multiple Row SubQueries)
- ▶ 運算子
 - 單筆記錄運算子(Single Row Operator)
 - >, >=, <, <=, <>, =
 - 多筆記錄運算子(Multiple Row Operator)
 - IN, ANY, ALL

自主子查詢(Self-contained subqueries)

- ▶ 自主子查詢
 - 子查詢執行與主查詢無關(可獨自測試)

```
SELECT    <select_list>
FROM      <table>
WHERE     <expr> operator (SELECT <select_list>
                           FROM   <table>);
```

- ▶ 在執行主查詢前，先只執行子查詢一次，再將子查詢的結果傳回給主查詢使用

單筆記錄子查詢 (Single Row SubQueries)

- ▶ 列出薪水比員工7566高的所有員工

```
mysql> SELECT empno, ename, sal
      -> FROM emp
      -> WHERE sal > (SELECT sal
      ->                   FROM emp
      ->                   WHERE empno = 7566);
+-----+-----+-----+
| empno | ename | sal |
+-----+-----+-----+
| 7839 | KING  | 5000 |
| 7902 | FORD  | 3000 |
| 7788 | SCOTT | 3000 |
+-----+-----+-----+
3 rows in set (0.00 sec)
```

```
mysql> SELECT sal
      -> FROM emp
      -> WHERE empno = 7566;
+-----+
| sal |
+-----+
| 2975 |
+-----+
1 row in set (0.00 sec)
```

```
mysql> SELECT empno, ename, sal
      -> FROM emp
      -> WHERE sal > 2975;
+-----+-----+-----+
| empno | ename | sal |
+-----+-----+-----+
| 7839 | KING  | 5000 |
| 7902 | FORD  | 3000 |
| 7788 | SCOTT | 3000 |
+-----+-----+-----+
3 rows in set (0.00 sec)
```

單筆記錄子查詢 (Single Row SubQueries)

- ▶ 多個子查詢在 SELECT 中

```
mysql> SELECT empno, ename, sal, job
      -> FROM emp
      -> WHERE job = (SELECT job
      ->                   FROM emp
      ->                   WHERE empno = 7499)
      ->     AND sal > (SELECT sal
      ->                   FROM emp
      ->                   WHERE empno = 7934);
+-----+-----+-----+-----+
| empno | ename  | sal   | job    |
+-----+-----+-----+-----+
| 7499  | ALLEN  | 1600  | SALESMAN |
| 7844  | TURNER | 1500  | SALESMAN |
+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

單筆記錄子查詢 (Single Row SubQueries)

- ▶ 列出薪資>全公司平均薪資的員工

```
mysql> SELECT empno, ename, sal
      -> FROM emp
      -> WHERE sal >  2073.2143
      ->           (SELECT AVG(sal)
      ->             FROM emp      );
+-----+-----+-----+
| empno | ename | sal   |
+-----+-----+-----+
| 7839  | KING  | 5000 |
| 7698  | BLAKE | 2850 |
| 7782  | CLARK | 2450 |
| 7566  | JONES | 2975 |
| 7902  | FORD  | 3000 |
| 7788  | SCOTT | 3000 |
+-----+-----+-----+
6 rows in set (0.00 sec)
```

單筆記錄子查詢 (Single Row SubQueries)

- ▶ 應用在 HAVING 子句

```
mysql> SELECT deptno, MIN(sal)
-> FROM emp
-> GROUP BY deptno
-> HAVING MIN(sal) >
->
->
->
+-----+-----+
| deptno | MIN(sal) |
+-----+-----+
|      10 |      1300 |
|      30 |      950  |
+-----+-----+
2 rows in set (0.00 sec)
```



800

```
(SELECT MIN(sal)
FROM emp
WHERE deptno=20);
```

單筆記錄子查詢的錯誤用法

- 多筆記錄子查詢使用了單一列運算子

```
mysql> SELECT empno, ename, sal
-> FROM emp
-> WHERE sal =
->           (SELECT MIN(sal)
->           FROM emp
->           GROUP BY deptno);
ERROR 1241 (21000) : Subquery returns more than 1 row
```

- 若子查詢沒傳回資料，則主查詢傳沒有列傳回

```
mysql> SELECT empno, ename, sal
-> FROM emp
-> WHERE sal > (SELECT sal
->                 FROM emp
->                 WHERE empno = 8000);
Empty set (0.00 sec)
```

Module 19.

資料連結4

19-1: Multiple-Row Subquery範例

19-2: ANY, ALL, IN

19-3: EXISTS

多筆記錄子查詢

- ▶ 列出公司所有主管員工編號與姓名

```
mysql> SELECT empno, ename
-> FROM emp
-> WHERE empno IN
->           (SELECT mgr
->                  FROM emp);
+-----+-----+
| empno | ename |
+-----+-----+
|    7839 | KING   |
|    7698 | BLAKE  |
|    7782 | CLARK  |
|    7566 | JONES  |
|    7902 | FORD   |
|    7788 | SCOTT  |
+-----+-----+
6 rows in set (0.00 sec)
```

NULL 值出現在子查詢中

- ▶ 列出公司所有職員員工編號與姓名

```
mysql> SELECT empno, ename
-> FROM emp
-> WHERE empno NOT IN
->                               (SELECT mgr
->                               FROM emp);
Empty set (0.00 sec)
```

- 當子查詢中含有空值且運算子為 not in 時，主查詢無資料傳回

NULL 值出現在子查詢中

- ▶ 解決方法—排除子查詢空值資料

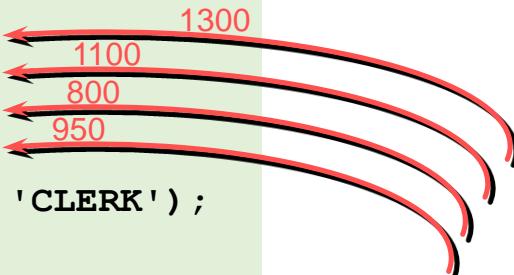
```
mysql> SELECT empno, ename
-> FROM emp
-> WHERE empno NOT IN ( SELECT mgr
->                                     FROM emp
->                                     WHERE mgr IS NOT NULL) ;

+-----+-----+
| empno | ename |
+-----+-----+
|    7654 | MARTIN |
|    7499 | ALLEN  |
|    7844 | TURNER |
|    7900 | JAMES   |
|    7521 | WARD    |
|    7369 | SMITH   |
|    7876 | ADAMS   |
|    7934 | MILLER  |
+-----+-----+
8 rows in set (0.00 sec)
```

多筆記錄子查詢

▶ 使用 ANY

```
mysql> SELECT empno, ename, job
-> FROM emp
-> WHERE sal < ANY (SELECT sal
-> FROM emp
-> WHERE job = 'CLERK');
+-----+-----+-----+
| empno | ename  | job    |
+-----+-----+-----+
| 7369  | SMITH   | CLERK  |
| 7521  | WARD    | SALESMAN|
| 7654  | MARTIN  | SALESMAN|
| 7876  | ADAMS   | CLERK  |
| 7900  | JAMES   | CLERK  |
+-----+-----+-----+
5 rows in set (0.00 sec)
```

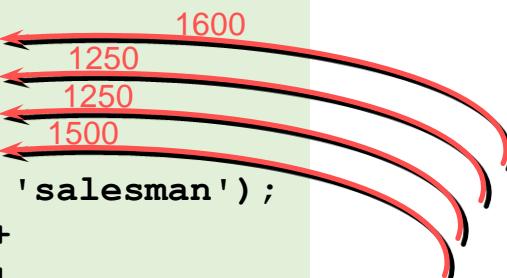


```
mysql> SELECT sal
-> FROM emp
-> WHERE job = 'CLERK';
+-----+
| sal   |
+-----+
| 800.00 |
| 1100.00|
| 950.00 |
| 1300.00|
+-----+
4 rows in set (0.00 sec)
```

多筆記錄子查詢

- ▶ 列出薪水比所有salesman高的員工資料

```
mysql> SELECT empno, ename, job, sal
-> FROM emp
-> WHERE sal > ALL (SELECT sal
->
->                               FROM emp
->                               WHERE job = 'salesman');
+-----+-----+-----+-----+
| empno | ename | job      | sal   |
+-----+-----+-----+-----+
| 7566  | JONES | MANAGER  | 2975.00 |
| 7698  | BLAKE | MANAGER  | 2850.00 |
| 7782  | CLARK | MANAGER  | 2450.00 |
| 7788  | SCOTT | ANALYST  | 3000.00 |
| 7839  | KING   | PRESIDENT | 5000.00 |
| 7902  | FORD   | ANALYST  | 3000.00 |
+-----+-----+-----+-----+
6 rows in set (0.00 sec)
```



使用EXISTS-存在性測試(existence test)

- ▶ EXISTS 不需任何欄位或運算式
- ▶ 因為不傳回資料, 所以一般在子查詢中的 SELECT 資料項都使用星號 asterisk (*)

```
SELECT custid, name
FROM customer AS c
WHERE EXISTS ( SELECT *
                FROM ord AS o
                WHERE c.custid=o.custid) ;
```

```
SELECT custid, name
FROM customer AS c
WHERE NOT EXISTS ( SELECT *
                     FROM ord AS o
                     WHERE c.custid=o.custid) ;
```

使用EXISTS-存在性測試(existence test)

- ▶ 列出下過訂單的客戶資料

```
mysql> SELECT c.custid, c.name
      -> FROM customer AS c
      -> WHERE EXISTS ( SELECT *
      ->                   FROM ord AS o
      ->                   WHERE o.custid = c.custid) ;
+-----+-----+
| custid | name
+-----+-----+
|    100 | JOCKSPORTS
|    101 | TKB SPORT SHOP
|    102 | VOLLYRITE
|    103 | JUST TENNIS
|    104 | EVERY MOUNTAIN
|    105 | K + T SPORTS
|    106 | SHAPE UP
|    107 | WOMENS SPORTS
|    108 | NORTH WOODS HEALTH AND FITNESS SUPPLY CENTER
+-----+-----+
9 rows in set (0.05 sec)
```

使用 NOT EXISTS

- ▶ 列出從未下過訂單的客戶資料

```
mysql> SELECT c.custid, c.name
-> FROM customer AS c
-> WHERE NOT EXISTS ( SELECT *
->                      FROM ord AS o
->                      WHERE o.custid=c.custid) ;
Empty set (0.00 sec)
```

Module 20.

資料連結5

- 20-1: Correlated Subquery語法
- 20-2: Correlated Subquery範例
- 20-3: CREATE TABLE with query

相關子查詢(Correlated subqueries)

- ▶ 相關子查詢必須依賴外部查詢(主查詢)所使用的資料表的資料來執行子查詢
- ▶ 依賴於外部查詢，不能單獨執行
 - 比自主子查詢難測試
- ▶ 執行次數依主查詢的資料筆數而定
 - 一筆資料執行一次子查詢
- ▶ 可能傳回一個值或多個值

撰寫相關子查詢(Correlated Subqueries)

- ▶ 內部查詢如何接收外部查詢資料表的資料
- ▶ 外部查詢如何接受內部查詢執行的結果(scalar or multi-valued)

```
SELECT custid, ordid, orderdate
FROM   ord AS outerorders
WHERE  orderdate =
       (SELECT MAX(orderdate)
        FROM ord AS innerorders
        WHERE innerorders.custid = outerorders.custid)
ORDER BY custid;
```

相關子查詢(Correlated subqueries)

- ▶ 查詢每個客戶最近跟公司下訂單的日期資料

```
mysql> SELECT ordid, custid, orderdate
-> FROM      ord AS O1
-> WHERE     orderdate = ( SELECT MAX(orderdate)
->                           FROM      ord AS O2
->                           WHERE    O2.custid = O1.custid)
-> ORDER BY custid, orderdate;
+-----+-----+-----+
| ordid | custid | orderdate          |
+-----+-----+-----+
|   613 |    100 | 1987-03-12 00:00:00 |
|   601 |    101 | 1987-01-07 00:00:00 |
|   620 |    102 | 1987-02-15 00:00:00 |
|   616 |    103 | 1987-02-03 00:00:00 |
|   617 |    104 | 1987-02-02 00:00:00 |
|   618 |    105 | 1987-02-05 00:00:00 |
|   607 |    106 | 1986-07-14 00:00:00 |
|   619 |    107 | 1987-02-01 00:00:00 |
|   614 |    108 | 1987-02-01 00:00:00 |
+-----+-----+-----+
9 rows in set (0.05 sec)
```

相關子查詢(Correlated subqueries)

- ▶ 查詢各部門薪資最高的員工資料

```
mysql> Select ename, sal, deptno
      -> From emp oe
      -> Where sal = (select max(sal)
      ->                   from emp ie
      ->                   where ie.deptno=oe.deptno)
      -> Order by deptno;
+-----+-----+-----+
| ename | sal     | deptno |
+-----+-----+-----+
| KING  | 5000.00 |      10 |
| SCOTT | 3000.00 |      20 |
| FORD  | 3000.00 |      20 |
| BLAKE | 2850.00 |      30 |
+-----+-----+-----+
4 rows in set (0.02 sec)
```

使用現有資料表來建立新的資料表

▶ 使用子查詢建立新的資料表

```
CREATE TABLE TableName [ (ColumnName, ...) ]  
AS  
SELECT ColumnName, ...  
FROM TableName  
WHERE ...
```

- 可在資料表後指定欄位名稱
- 若不指定，則與子查詢中欄位同名
- 亦可在子查詢中使用欄位別名
- 子查詢的資料會新增到新建的資料表中
- 只有不為空值(Not Null)的檢查條件會被保留在新的資料表中，其餘的Constraints 均不存在

使用現有資料表來建立新的資料表

```
mysql> CREATE TABLE EMP10
      -> AS
      -> SELECT EMPNO, ENAME, JOB, SAL
      -> FROM EMP
      -> WHERE DEPTNO = 10;
Query OK, 3 rows affected (0.26 sec)
Records: 3  Duplicates: 0  Warnings: 0
```

```
mysql> desc emp10;
+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| EMPNO | int(11) | NO  |    | NULL    |       |
| ENAME | varchar(10) | YES |    | NULL    |       |
| JOB   | varchar(9)  | YES |    | NULL    |       |
| SAL   | decimal(7,2) | YES |    | NULL    |       |
+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

NOT NULL Constraint

使用LIKE子句建立資料表

▶ 使用LIKE子句建立資料表

```
CREATE TABLE TableName LIKE Source_TableName;
```

- 使用既存的資料表結構建立新資料表
- 此種方法不會有資料的新增，只有結構定義部份
- 無法指定欄位名稱，和來源資料表欄位同名
- 只有主鍵與不為空值會被保留，其餘的Constraints均不存在
- 要新增資料請再使用 Insert into with Sub-Query 方法

使用現有資料表來建立新的資料表

```
mysql> create table emp1 like emp;  
Query OK, 0 rows affected (0.27 sec)
```

```
mysql> desc emp1;
```

Field	Type	Null	Key	Default	Extra
EMPNO	int(11)	NO	PRI	NULL	
ENAME	varchar(10)	YES		NULL	
JOB	varchar(9)	YES		NULL	
MGR	int(11)	YES	MUL	NULL	
HIREDATE	datetime	YES		NULL	
SAL	decimal(7,2)	YES		NULL	
COMM	decimal(7,2)	YES		NULL	
DEPTNO	int(11)	NO	MUL	NULL	

NOT NULL & PRIMARY KEY

```
mysql> select * from emp1;  
Empty set (0.00 sec)
```

Module 21.

資料操作語法

21-1: INSERT Statement

21-2: UPDATE Statement

21-3: DELETE Statement

資料處理語言(DML)

- ▶ 資料處理語言(Data Manipulation Language)
 - **INSERT**: 新增資料到資料表中
 - **UPDATE**: 修改資料表中的資料
 - **DELETE**: 刪除資料表中的資料

21-1: INSERT Statement

新增一筆記錄到資料表中

```
INSERT INTO dept(deptno, dname, loc)  
VALUES (50, 'DEVELOPMENT', 'DETROIT');
```

DEPT		
DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

“...insert a new row
into DEPT table...”



DEPT		
DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON
50	DEVELOPMENT	DETROIT

INSERT INTO 命令

```
INSERT INTO table [(column, ...)]  
VALUES (value, ...);
```

- 欄位與值的數量需相同，位置對應且型態要相容
- 可以用指定欄位或不指定欄位的方式來新增資料
 - 不指定欄位時，資料表中所有的欄位均要給值
 - 使用 DESC 指令可以得知欄位值的順序

```
INSERT INTO dept(deptno, dname, loc) ← 指定欄位  
VALUES(50, 'MIS', 'NEW YORK');
```

```
INSERT INTO dept ← 不指定欄位  
VALUES(50, 'MIS', 'NEW YORK');
```

```
mysql> DESC DEPT;  
+-----+-----+-----+-----+-----+  
| Field | Type | Null | Key | Default | Extra |  
+-----+-----+-----+-----+-----+  
| DEPTNO | smallint(6) | YES | PRI | 0 |  
| DNAME | varchar(14) | YES | | NULL |  
| LOC | varchar(13) | YES | | NULL |  
+-----+-----+-----+-----+-----+
```

21-1: INSERT Statement

► 新增空值到資料表的方法

- 未列舉的欄位，新增資料時將值填入預設值(default value)，若該欄位沒有設定預設值則填入空值(NULL)

```
mysql> INSERT INTO dept(deptno, dname)
-> VALUES(60, 'MIS');
Query OK, 1 row affected (0.00 sec)
```

- 使用 NULL 關鍵字

```
mysql> INSERT INTO dept(deptno, dname, loc)
-> VALUES(70, 'HR', NULL);
Query OK, 1 row affected (0.00 sec)
```

```
mysql> SELECT *
->   FROM DEPT
-> WHERE DEPTNO=70;
+-----+-----+-----+
| DEPTNO | DNAME  | LOC   |
+-----+-----+-----+
|    70  | HR     | (NULL) |
+-----+-----+-----+
1 row in set (0.00 sec)
```

```
mysql> SELECT *
->   FROM DEPT
-> WHERE DEPTNO=60;
+-----+-----+-----+
| DEPTNO | DNAME  | LOC   |
+-----+-----+-----+
|    60  | MIS    | (NULL) |
+-----+-----+-----+
1 row in set (0.00 sec)
```

▶ 新增日期資料的方法

- 可以使用 `NOW()` 或 `CURRENT_DATE()` … 等函數
- 用字串輸入, 請參照資料型態章節

```
INSERT INTO EMP(empno,ename,job,mgr,hiredate,sal,comm,deptno)
VALUES(8001,'JAMES','ANALYST',7839,CURRENT_DATE(),2500,NULL,20);
```

```
INSERT INTO EMP(empno,ename,job,mgr,hiredate,sal,comm,deptno)
VALUES(8002,'JOHN','SALESMAN',7839,'1983-03-25',1400,500,30);
```

```
mysql> SELECT *
-> FROM emp
-> WHERE empno > 8000;
+-----+-----+-----+-----+-----+-----+-----+-----+
| EMPNO | ENAME | JOB      | MGR   | HIREDATE        | SAL   | COMM | DEPTNO |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 8001  | JAMES | ANALYST | 7839 | 2004-07-05     | 2500  | NULL  |    20  |
| 8002  | JOHN  | SALESMAN | 7839 | 1983-03-25     | 1400  | 500   |    30  |
+-----+-----+-----+-----+-----+-----+-----+-----+
2 rows in set (0.01 sec)
```

從其他資料表複製資料

- ▶ 使用子查詢將其他資料表中的資料新增到指定的資料表

```
INSERT INTO table [(column, ...)]  
    SELECT column, ...  
    FROM table  
    WHERE ...
```

Sub-Query

- 範例：

```
INSERT INTO bonus (ename, job, sal, comm)  
    SELECT ename, job, sal, comm  
    FROM emp  
    WHERE deptno=10;
```

```
mysql> SELECT * FROM bonus;  
+-----+-----+-----+-----+  
| ENAME | JOB      | SAL   | COMM  |  
+-----+-----+-----+-----+  
| KING  | PRESIDENT | 5000 | NULL  |  
| CLARK | MANAGER   | 2450 | NULL  |  
| MILLER | CLERK    | 1300 | NULL  |  
+-----+-----+-----+-----+  
3 rows in set (0.00 sec)
```

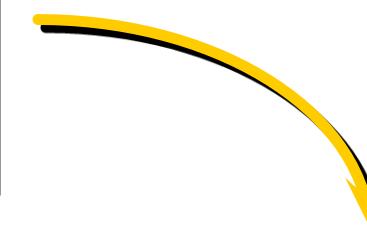
修改資料表中的資料

EMP

EMPNO	ENAME	JOB	...	DEPTNO
7839	KING	PRESIDENT		10
7698	BLAKE	MANAGER		30
7782	CLARK	MANAGER		10
7566	JONES	MANAGER		20
...				

```
UPDATE emp  
SET deptno = 20  
WHERE empno = 7782;
```

“...update a row
in EMP table...”



EMP

EMPNO	ENAME	JOB	...	DEPTNO
7839	KING	PRESIDENT		10
7698	BLAKE	MANAGER		30
7782	CLARK	MANAGER		20
7566	JONES	MANAGER		20
...				

修改資料表中的資料

```
UPDATE table  
SET column=value, [column=value,] ...  
WHERE conditions;
```

- 欄位與值的型態應相同
- 一次可以修改多個欄位, 用逗號隔開
- 若遺漏了WHERE子句則會更改資料表中所有的rows

UPDATE emp
SET sal = 1000
WHERE empno = 7900;

```
mysql> SELECT * FROM emp WHERE empno = 7900;  
+-----+-----+-----+-----+-----+-----+-----+-----+  
| EMPNO | ENAME | JOB   | MGR  | HIREDATE | SAL    | COMM  | DEPTNO |  
+-----+-----+-----+-----+-----+-----+-----+-----+  
| 7900  | JAMES  | CLERK | 7698 | 1981-10-03 | 1000  | NULL   | 30    |  
+-----+-----+-----+-----+-----+-----+-----+-----+  
1 row in set (0.01 sec)
```

同時更新兩個欄位資料

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT	NULL	1981-11-18	5000	NULL	10
7698	BLAKE	MANAGER	7839	1981-05-01	2850	NULL	30
7782	CLARK	MANAGER	7839	1981-06-09	2450	NULL	10
7566	JONES	MANAGER	7839	1981-04-02	2975	NULL	20
7654	MARTIN	SALESMAN	7698	1981-09-28	1250	1400	30
7499	ALLEN	SALESMAN	7698	1981-02-20	1600	300	30
7844	TURNER	SALESMAN	7698	1981-09-08	1500	0	30
7900	JAMES	CLERK	7698	1981-10-03	1000	NULL	30
7521	WARD	SALESMAN	7698	1981-02-22	1250	500	30
7902	FORD	ANALYST	7566	1981-10-03	3000	NULL	20
7369	SMITH	SALESMAN	7902	1980-10-17	800	NULL	30
7788	SCOTT	ANALYST	7566	1982-10-09	3000	NULL	20
7876	ADAMS	CLERK	7788	1983-01-12	1100	NULL	20
7934	MILLER	CLERK	7782	1982-01-23	1300	NULL	10

```
UPDATE emp
SET job = 'SALESMAN',
deptno = 30
WHERE empno = 7369;
```

使用子查詢來更新資料

▶ 使用子查詢

- 與子查詢合併使用時，二者間資料表不可以是同一個

▶ 當取值使用

```
UPDATE emp
SET deptno = (SELECT deptno
                FROM dept
                WHERE dname='HR')
WHERE empno = 7900;
```

EMPNO	ENAME	JOB	SAL	DEPTNO
7566	JONES	MANAGER	2975	20
...				
7902	FORD	ANALYST	3000	20
7698	BLAKE	MANAGER	2850	30
7521	WARD	SALESMAN	1250	30
7900	JAMES	CLERK	1000	70
...				
8002	JOHN	SALESMAN	1400	30

• 當條件使用

```
UPDATE emp
SET sal = sal + 500
WHERE deptno = (SELECT deptno
                  FROM dept
                  WHERE dname='SALES');
```

以下為30部門員工異動後的薪水參考

EMPNO	SAL	SAL+500
7698	2850	3350
7654	1250	1750
7499	1600	2100
7844	1500	2000
7900	1000	1500
7521	1250	1750
8002	1400	1900

7 rows in set (0.00 sec)

刪除資料表中的資料

DEPT

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON
50	DEVELOPMENT	DETROIT
60	MIS	
...		

```
DELETE FROM DEPT  
WHERE deptno = 50;
```

“...delete a row
from DEPT table...”

DEPT

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON
60	MIS	
...		

DELETE FROM 命令

```
DELETE FROM table  
[WHERE conditions];
```

- 刪除資料表中的資料
- 若遺漏了 WHERE 子句時，則刪除資料表中所有的資料

```
DELETE FROM dept  
WHERE deptno = 70;
```

```
mysql> SELECT * FROM dept;  
+-----+-----+-----+  
| DEPTNO | DNAME      | LOC       |  
+-----+-----+-----+  
| 10    | ACCOUNTING | NEW YORK |  
| 20    | RESEARCH   | DALLAS    |  
| 30    | SALES      | CHICAGO   |  
| 40    | OPERATIONS | BOSTON    |  
| 50    | MIS         | NEW YORK |  
| 60    | MIS         | NULL      |  
+-----+-----+-----+  
6 rows in set (0.00 sec)
```

部門70的資料已被刪除

刪除資料表中的資料

- ▶ 指定刪除的對象

```
DELETE FROM dept  
WHERE dname = 'Finance';
```

- ▶ 省略WHERE子句刪除表格中全部的資料

```
DELETE FROM dept;
```

- ▶ 使用子查詢結果作為刪除資料的對像(條件)

```
DELETE FROM emp  
WHERE deptno = (SELECT deptno  
                  FROM dept  
                 WHERE dname LIKE '%Public%');
```

Module 22.

DML常見的錯誤

22-1: INSERT 常見的錯誤

22-2: UPDATE 常見的錯誤

22-3: DELETE 常見的錯誤

新增、修改與刪除資料的錯誤

▶ 常見的錯誤

- 違反資料檢查條件(Constraint)
 - 鍵值重複—違反主鍵(primary key)或唯一鍵(Unique)
 - 違反外來鍵(Foreign Key)
 - 對非空值(NOT NULL)的欄位給定空值
- 值過長或值不夠
- 資料型態錯誤

```
UPDATE    emp
SET        deptno = 55
WHERE      deptno = 10;
```

UPDATE emp
*
ERROR at line 1:
ORA-02291: integrity constraint (USR.EMP_DEPTNO_FK) violated - parent key not found

Department number 55 does not exist

新增資料-違反 NOT NULL

- NOT NULL的錯誤

```
mysql> DESC EMP;
+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| EMPNO | smallint(6) | YES | PRI | 0        |          |
| ENAME | varchar(10) | YES |      | NULL     |          |
| JOB   | varchar(9)  | YES |      | NULL     |          |
| MGR   | smallint(6) | YES |      | NULL     |          |
| HIREDATE | date    | YES |      | NULL     |          |
| SAL   | int(11)  | YES |      | NULL     |          |
| COMM  | int(11)  | YES |      | NULL     |          |
| DEPTNO | smallint(6) | YES |      | 0        |          |
+-----+-----+-----+-----+-----+
8 rows in set (0.00 sec)

mysql> INSERT INTO emp(empno, ename, job, deptno )
->   VALUES(9001, 'SANDY', 'CLERK', NULL);
ERROR 1048 (23000): Column 'DEPTNO' cannot be null
```

新增資料-違反外來鍵

- ▶ 新增資料時的錯誤
 - 違反外來鍵(Foreign Key)的錯誤

TABLE : PKT		TABLE : FKT	
PK		FK	C2
1		1	ROW1
2		2	ROW2
3		2	ROW3
		1	ROW4
		3	ROW5
		3	ROW6
		3	ROW7
		4	ROW8

X

```
INSERT INTO FKT
Values (4,'ROW8')
```

在PKT TABLE中並沒有 4 這筆資料, 所以無法新增

修改資料-違反NOT NULL

▶ NOT NULL的錯誤

```
mysql> DESC EMP;
+-----+-----+-----+-----+-----+
| Field | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| EMPNO | smallint(6) | YES  | PRI | 0       |       |
| ENAME | varchar(10) | YES  |     | NULL    |       |
| JOB   | varchar(9)  | YES  |     | NULL    |       |
| MGR   | smallint(6) | YES  |     | NULL    |       |
| HIREDATE | date     | YES  |     | NULL    |       |
| SAL   | int(11)    | YES  |     | NULL    |       |
| COMM  | int(11)    | YES  |     | NULL    |       |
| DEPTNO | smallint(6) | YES  |     | 0       |       |
+-----+-----+-----+-----+-----+
8 rows in set (0.00 sec)
```

```
UPDATE EMP
  SET DEPTNO = NULL
 WHERE EMPNO = 8001;
```

修改資料-違反外來鍵

- ▶ 修改資料時的錯誤
 - 違反整合資料檢查條件(Integrity Constraint Error)

TABLE : PKT		TABLE : FKT	
PK		FK	C2
1			ROW1
2		2	ROW2
3		2	ROW3
		1	ROW4
		4	ROW5
		3	ROW6
		3	ROW7

在PKT TABLE中並沒有 4 這筆資料, 所以無法更新

刪除資料-違反外來鍵

- ▶ 刪除資料的錯誤
 - Integrity Constraint Error

TABLE : PK		TABLE : FKT	
T		+-----+-----+	
+----+	PK	FK	C2
1	+----+	+----+	+----+
2		1	ROW1
3	↑	2	ROW2
+----+		2	ROW3
		1	ROW4
		3	ROW5
		3	ROW6
		3	ROW7
共3筆資料		+-----+	

在FKT TABLE中有3筆資料參考著PKT TABLE中 PK=3 這筆資料，故刪除PKT TABLE中 PK=3 時會產生錯誤訊息，如下：

```
mysql> DELETE FROM pkt
-> WHERE pk = 3;
ERROR 1217 (23000): Cannot delete or update a parent row: a foreign key
constraint fails
```

Module 23.

限制條件1

- 23-1: DEFAULT Constraint**
- 23-2: NOT NULL Constraint**
- 23-3: UNIQUE KEY Constraint**

主要的限制條件

- ▶ 用來檢查資料表的資料 (data validation)
 - 欄位預設值(Default)
 - 不允許空值(NOT NULL)
 - 唯一鍵(Unique key)
 - 主鍵(Primary key)
 - 外來鍵(Foreign key)

欄位預設值(DEFAULT option)

▶ 設定欄位預設值

```
ColumnName type [NOT NULL] | [NULL] DEFAULT expr
```

- Expr 需是常值(Literal), 不可以是函數或運算式
- MySQL 資料庫中, 所有欄位皆有預設值, 如果不設定其預設值為空值(NULL)

```
CREATE TABLE EMP
(
    ...,
    BIRTHDATE DATE,
    SEX        CHAR(1) DEFAULT 'M'
    ...
);
```

初值為NULL

初值為 'M'

不允許空值(NOT NULL constraint)

- ▶ 不為空值檢查(一定要輸入資料)

```
ColumnName type [NOT NULL] | [NULL]
```

- NOT NULL: 該column所有的rows均要輸入Data

```
CREATE TABLE EMP
(
    EMPNO      SMALLINT      NOT NULL,
    ENAME      VARCHAR(10)   NOT NULL,
    JOB        VARCHAR(9),
    MGR        SMALLINT,
    HIREDATE   DATE,
    SAL         INT,
    COMM        INT,
    DEPTNO     SMALLINT
);
```

唯一鍵 (UNIQUE constraint)

▶ 唯一鍵

```
ColumnName type UNIQUE
```

- 使用唯一鍵時，值可以是空值
- 一個資料表中可以多組唯一鍵
- 適用一個欄位構成的唯一鍵

```
CREATE TABLE EMP
(
    . . . ,
    EMAIL      VARCHAR(200) UNIQUE ,
    . . .
)
```

唯一鍵 (UNIQUE constraint)

- ▶ 複合唯一鍵(由多個欄位構成)

```
CREATE TABLE TableName
(
    ...,
    ColumnName type ,
    ...,
    [CONSTRAINT const_name] UNIQUE(column,...)
);
```

- const_name 為唯一鍵命名

```
CREATE TABLE ITEM1
(
    ORDID          INT      NOT NULL,
    ITEMID         SMALLINT NOT NULL,
    PRODID         SMALLINT,
    CONSTRAINT UK_ITEM_ORDID_PRODID UNIQUE (ORDID,PRODID)
);
```

Module 24.

限制條件2

24-1: PRIMARY KEY Constraint

24-2: FOREIGN KEY Constraint

24-3: 變更資料表的限制條件

主鍵 (PRIMARY KEY constraint)

► 主鍵(PK)

ColumnName type PRIMARY KEY

- 一個資料表中只能有一個主鍵
- 適用一個欄位構成的主鍵

```
CREATE TABLE DEPT122
(
    DEPTNO SMALLINT(4) PRIMARY KEY,
    DNAME  VARCHAR(14),
    LOC    VARCHAR(13)
);
```

主鍵 (PRIMARY KEY constraint)

- ▶ 複合主鍵(由多個欄位構成)

```
CREATE TABLE table
(
    ...,
    ColumnName type NOT NULL,
    ...,
    [CONSTRAINT const_name] PRIMARY KEY(column,...)
);
```

- **CONSTRAINT const_name**: 為主鍵命名

```
CREATE TABLE ITEM12
(
    ORDID          INT      NOT NULL,
    ITEMID         SMALLINT NOT NULL,
    ...
    CONSTRAINT PK_ITEM_ORDID_ITEMID PRIMARY KEY(ORDID,ITEMID)
);
```

AUTO_INCREMENT 選項

► 欄位資料自動編號

```
ColumnName type AUTO_INCREMENT
```

- 欄位的資料型態必需是數值型態
- 預設起始值是0, 也可自設起始值
- 需與Primary key或Unique key配合使用
- LAST_INSERT_ID() : 取得最近一次自動產生的編號

```
CREATE TABLE ORD2
(  ORDID      INT AUTO_INCREMENT PRIMARY KEY,
    ORD_DATE   DATE,
    ...
);
```

AUTO_INCREMENT 選項

```
CREATE TABLE TableName
(Column Name type AUTO_INCREMENT,
 . . . ,
) AUTO_INCREMENT = start ;
```

```
CREATE TABLE ORD2
( ORDID      INT      AUTO_INCREMENT PRIMARY KEY,
  ORD_DATE   DATE,
  . . . ,
) AUTO_INCREMENT = 101 ;
```

外來鍵(Foreign key constraint)

▶ 外來鍵

```
CREATE TABLE table
(
    ...,
    [CONSTRAINT const_name]
        FOREIGN KEY(fk_column,...) REFERENCES parent(column,...)
) ENGINE=InnoDB;
```

- 外來鍵與參考的資料表須是 InnoDB 儲存引擎類型
- 外來鍵的欄位數需和 parent table 的主鍵或唯一鍵欄位數量相同

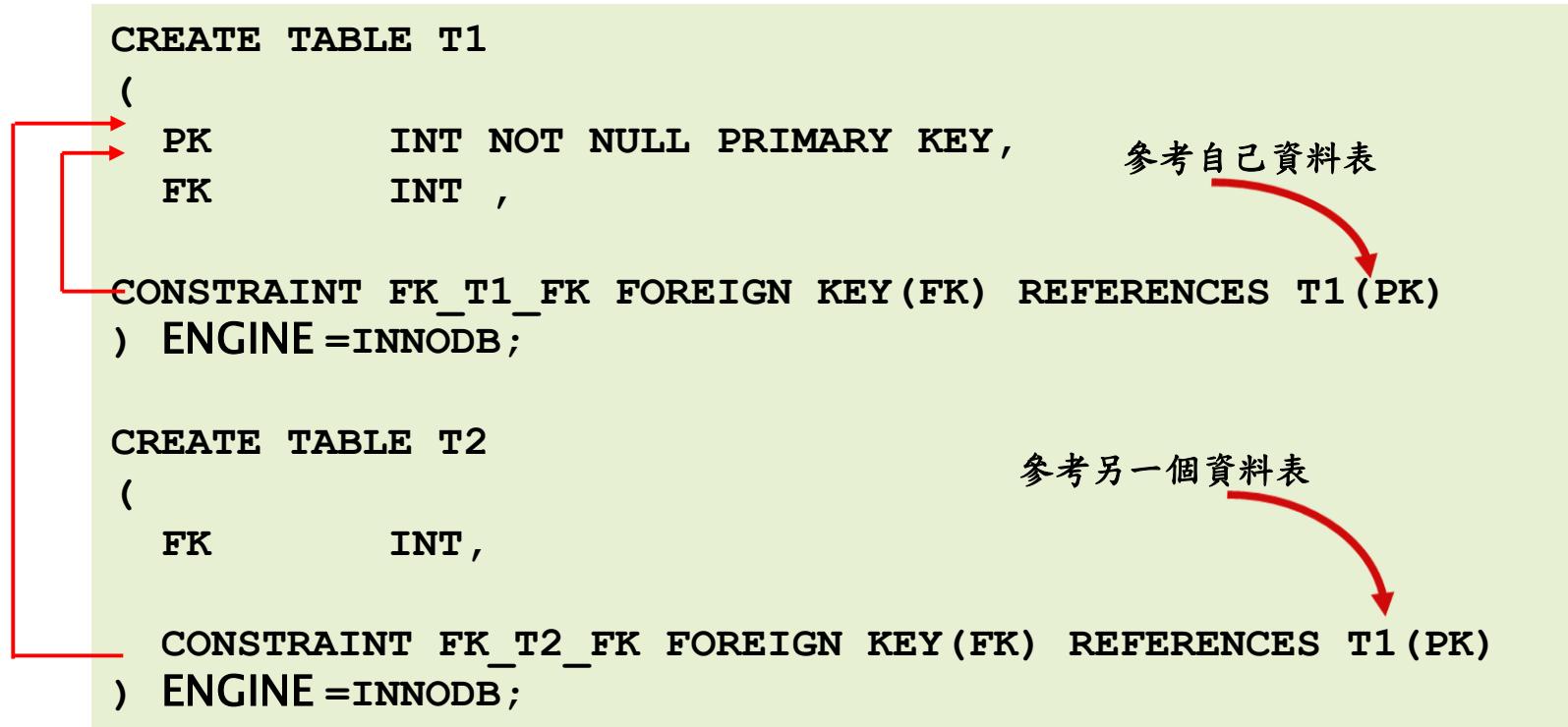
外來鍵(Foreign key constraint)

```
CREATE TABLE T1
(
    PK      INT NOT NULL PRIMARY KEY,
    FK      INT ,
    CONSTRAINT FK_T1_FK FOREIGN KEY(FK) REFERENCES T1(PK)
) ENGINE=INNODB;

CREATE TABLE T2
(
    FK      INT ,
    CONSTRAINT FK_T2_FK FOREIGN KEY(FK) REFERENCES T1(PK)
) ENGINE=INNODB;
```

參考自己資料表

參考另一個資料表



外來鍵(Foreign key constraint)

► 外來鍵資料的刪除與修改

```
CREATE TABLE table
(
    ...,
    fk_definition
    [ON DELETE {RESTRICT|CASCADE|SET NULL}] |
    [ON UPDATE {RESTRICT|CASCADE|SET NULL}]
);
```

- ON DELETE 資料刪除時
- ON UPDATE 資料更新時

- RESTRICT 限制
- CASCADE 相依性
- SET NULL 設成空值

外來鍵(Foreign key constraint)

- ▶ ON DELETE CASCADE
 - 相依性刪除

```
DROP TABLE T2;
DROP TABLE T1;
CREATE TABLE T1
( PK    SMALLINT NOT NULL PRIMARY KEY
) ENGINE = INNODB;

CREATE TABLE T2
( FK    SMALLINT,
  C2    CHAR(2),
  FOREIGN KEY(FK) REFERENCES T1(PK) ON DELETE CASCADE
) ENGINE = INNODB;

INSERT INTO T1 VALUES(1),(2),(3);
INSERT INTO T2 VALUES(1,'A1'),(2,'A2'),(2,'A2'),(3,'A3');
```

外來鍵(Foreign key constraint)

- ON DELETE SET NULL
 - 相依性刪除

```
DROP TABLE T2;
DROP TABLE T1;
CREATE TABLE T1
( PK    SMALLINT NOT NULL PRIMARY KEY
) ENGINE = INNODB;

CREATE TABLE T2
( FK    SMALLINT,
C2    CHAR(2),

FOREIGN KEY(FK) REFERENCES T1(PK) ON DELETE SET NULL
) ENGINE = INNODB;

INSERT INTO T1 VALUES(1),(2),(3);
INSERT INTO T2 VALUES(1,'A1'),(2,'A2'),(2,'A2'),(3,'A3');
```

外來鍵(Foreign key constraint)

- ▶ ON UPDATE CASCADE
 - 相依性修改

```
DROP TABLE T2;
DROP TABLE T1;
CREATE TABLE T1
( PK      SMALLINT NOT NULL PRIMARY KEY
) ENGINE = INNODB;

CREATE TABLE T2
( FK      SMALLINT,
C2      CHAR(2),

FOREIGN KEY(FK) REFERENCES T1(PK) ON UPDATE CASCADE
) ENGINE = INNODB;

INSERT INTO T1 VALUES(1),(2),(3);
INSERT INTO T2 VALUES(1,'A1'),(2,'A2'),(2,'A2'),(3,'A3');
```

外來鍵(Foreign key constraint)

- ▶ ON UPDATE SET NULL
 - 外來鍵修改時設為空值

```
DROP TABLE T2;
DROP TABLE T1;
CREATE TABLE T1
( PK      SMALLINT NOT NULL PRIMARY KEY
) ENGINE = INNODB;

CREATE TABLE T2
( FK      SMALLINT,
 C2      CHAR(2),

    FOREIGN KEY(FK) REFERENCES T1(PK) ON UPDATE SET NULL
) ENGINE = INNODB;

INSERT INTO T1 VALUES(1),(2),(3);
INSERT INTO T2 VALUES(1,'A1'),(2,'A2'),(2,'A2'),(3,'A3');
```

外來鍵(Foreign key constraint)

► 範例

```
CREATE TABLE DEPARTMENT
(
    DEPTNO      SMALLINT NOT NULL PRIMARY KEY,
    DNAME       VARCHAR(14),
    LOC         VARCHAR(13)
) ENGINE = INNODB;

CREATE TABLE EMPLOYEE
(
    EMPNO       SMALLINT NOT NULL PRIMARY KEY,
    ENAME       VARCHAR(14),
    JOB         VARCHAR(13),
    MGR         SMALLINT,
    HIREDATE    DATE,
    SAL          INT,
    COMM         INT,
    DEPTNO      SMALLINT,
    EMAL        VARCHAR(200) UNIQUE,
    CONSTRAINT FK_EMP_MGR FOREIGN KEY(MGR) REFERENCES EMPLOYEE(EMPNO),
    CONSTRAINT FK_EMP_DEPTNO FOREIGN KEY(DEPTNO) REFERENCES DEPARTMENT(DEPTNO)
) ENGINE = INNODB;
```

新增資料檢查條件(Adding a constraint)

- ## ▶ 加入資料檢查條件

```
ALTER TABLE TableName  
    ADD [CONSTRAINT const name] constraint type;
```

- const_name 限制條件名稱
 - constraint_type 限制條件種類
 - NOT NULL constraint 請用MODIFY句子

```
/* 新增主鍵 */
ALTER TABLE EMP10
    ADD CONSTRAINT PK_EMP10_EMPNO PRIMARY KEY(EMPNO);
/* 新增外來鍵 */
ALTER TABLE EMP10
    ADD CONSTRAINT FK_EMP10_MGR FOREIGN KEY(MGR)
        REFERENCES EMP10(EMPNO);
/* 新增唯一鍵 */
ALTER TABLE EMP10
    ADD CONSTRAINT UK_EMP10_EMAIL UNIQUE(EMAIL);
```

刪除資料檢查條件(Drop a constraint)

```
ALTER TABLE table
DROP [PRIMARY KEY|FOREIGN KEY ConstraintName |
      INDEX ConstraintName];
```

```
/* 刪除外來鍵 */
ALTER TABLE EMP10,
    DROP FOREIGN KEY FK_EMP10_MGR;

/* 刪除主鍵 */
ALTER TABLE EMP10
    DROP PRIMARY KEY;

/* 刪除唯一鍵 */
ALTER TABLE EMP10
    DROP INDEX UK_EMP10_EMAIL;
```

刪除資料限制條件(Drop a constraint)

- ▶ 刪除UNIQUE key

```
mysql> ALTER TABLE EMP10
      -> ADD CONSTRAINT EMP10_ENAME_UK UNIQUE(ENAME) ;
Query OK, 0 rows affected (0.60 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> UPDATE EMP10
      -> SET ENAME='CLARK'
      -> WHERE EMPNO=7934;
ERROR 1062 (23000): Duplicate entry 'CLARK' for key 'EMP10_ENAME_UK'

mysql> ALTER TABLE EMP10
      -> DROP INDEX EMP10_ENAME_UK;
Query OK, 3 rows affected (0.59 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> UPDATE EMP10
      -> SET ENAME='CLARK'
      -> WHERE EMPNO=7934;
Query OK, 1 row affected (0.04 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

Module 25.

交易處理1

- 25-1: 交易處理概念**
- 25-2: Commit與Rollback**
- 25-3: 交易啟動與結束的方式**

資料庫交易(Database Transactions)

- ▶ 交易(Database Transactions)
 - 改變資料庫中現有資料
- ▶ 依不同類型的指令所產生的交易：
 - DML 交易
 - DDL 交易
 - DCL 交易
- ▶ MySQL 預設的交易處理方式：
 - DDL & DCL 交易 - Auto commit(自動確認)
 - DML 交易 - Auto commit(自動確認)



User Commit (使用者確認或放棄)

交易控制(TTransactions Control)

- ▶ 交易控制的主要目的
 - 維持資料庫中資料的一致性(Consistency)和正確性(Correctness)
- ▶ 何謂交易(Database Transactions)
 - 是多個 SQL statements(DML)所構成的邏輯工作單位
 - 交易(Transaction) 提供了一個資料處理的邏輯單元(Logic Unit)
 - 在該邏輯單元中如果全部執行成功，則會確定交易期間所修改的所有資料正式成為資料庫的內容
 - 如果有發生錯誤，則必須取消或回復該交易期間內所有的資料修改
 - 交易的結果：成功或失敗

交易的特性 (Properties of Transaction)

- ▶ 資料庫交易需符合**ACID**特性
- ▶ 單元性 (**A**tomicity)
 - 一個交易是一個單元 (Atomic) 的處理。如果是正常情況，則整個交易應完整的被執行。否則，發生任一錯誤情況時，則將交易所做的資料更新復原到交易開始前狀況 (undo)。因此，單元性也稱之為不可部份完成性。
- ▶ 一致性 (**C**onsistency)
 - 一個完整的交易能使資料庫從一個一致性狀態 (consistent state) 轉換到另一個一致性狀態。所謂一致性狀態是指資料庫的資料滿足資料庫綱要所定義的所有限制，例如整合限制條件。
- ▶ 隔離性 (**I**solation)
 - 一個執行中的交易不應被其他同步執行中的交易所影響，即交易間應具有隔離機制。
- ▶ 耐久性 (**D**urability)
 - 一個交易成功的執行「認可」 (commit) 命令後，它對資料庫所作的任何資料更新處理，均反應在資料庫中。這些更新動作不會受「認可」後的任何錯誤所影響。

MySQL 交易控制(Transactions Control)

- ▶ MySQL 只有 InnoDB Type 才支援 交易處理
- ▶ MySQL 預設的交易控制
 - 自動確認(AUTOCOMMIT=1)
- ▶ MySQL 的交易控制
 - 起始：
 - SET AUTOCOMMIT = 0 - 改變預設交易模式
 - 使用 BEGIN | START TRANSACTION
 - 結束：
 - COMMIT/ROLLBACK
 - Auto-Commit Command
 - DDL, DCL
 - SET AUTOCOMMIT

資料在交易中的狀態

- ▶ 交易過程中
 - DML 會產生 row locking(列的鎖定)
 - 可以檢視資料的改變
 - 可做邏輯交易單位(savepoint 儲存點)的測試
 - 其他連線(sessions)並無法看到資料的改變(讀取舊值)
 - Row Locking 並不會影響到其他的資料查詢
 - 不可有自動確認交易指令在其中，否則系統會確認並結束交易

確認交易(COMMIT)後的資料狀態

- ▶ 確認交易： COMMIT
 - 將資料儲存到資料庫中
 - 釋放交易過程中的 Locking
 - 刪除設定的儲存點(savepoint)
 - 其他連線(sessions)的使用者將可以看到變更後的資料
 - 無法再復原(Rollback)資料
 - 此交易立即結束

放棄交易(ROLLBACK)後的資料狀態

- ▶ 放棄交易 : ROLLBACK
 - 取消交易中資料的變更
 - 釋放交易過程中的 Locking
 - 刪除設定的儲存點(savepoint)
 - 其他連線(sessions)的使用者立即可對資料做操作(DML)
此交易立即結束

Module 26.

交易處理2

26-1: 外顯式交易

26-2: 隱含式交易

26-3: 交易儲存點

交易控制(Controlling Transactions)

► 設定交易區塊(TRANSACTION BLOCK)

```
START TRANSACTION | BEGIN;  
    dml_statement...;  
{ COMMIT | ROLLBACK };
```

- 適合單次交易的處理

```
START TRANSACTION;  
    INSERT INTO TX VALUES (NULL,NOW());  
    SELECT * FROM TX;  
ROLLBACK;  
    SELECT * FROM TX;
```

交易控制(Controlling Transactions)

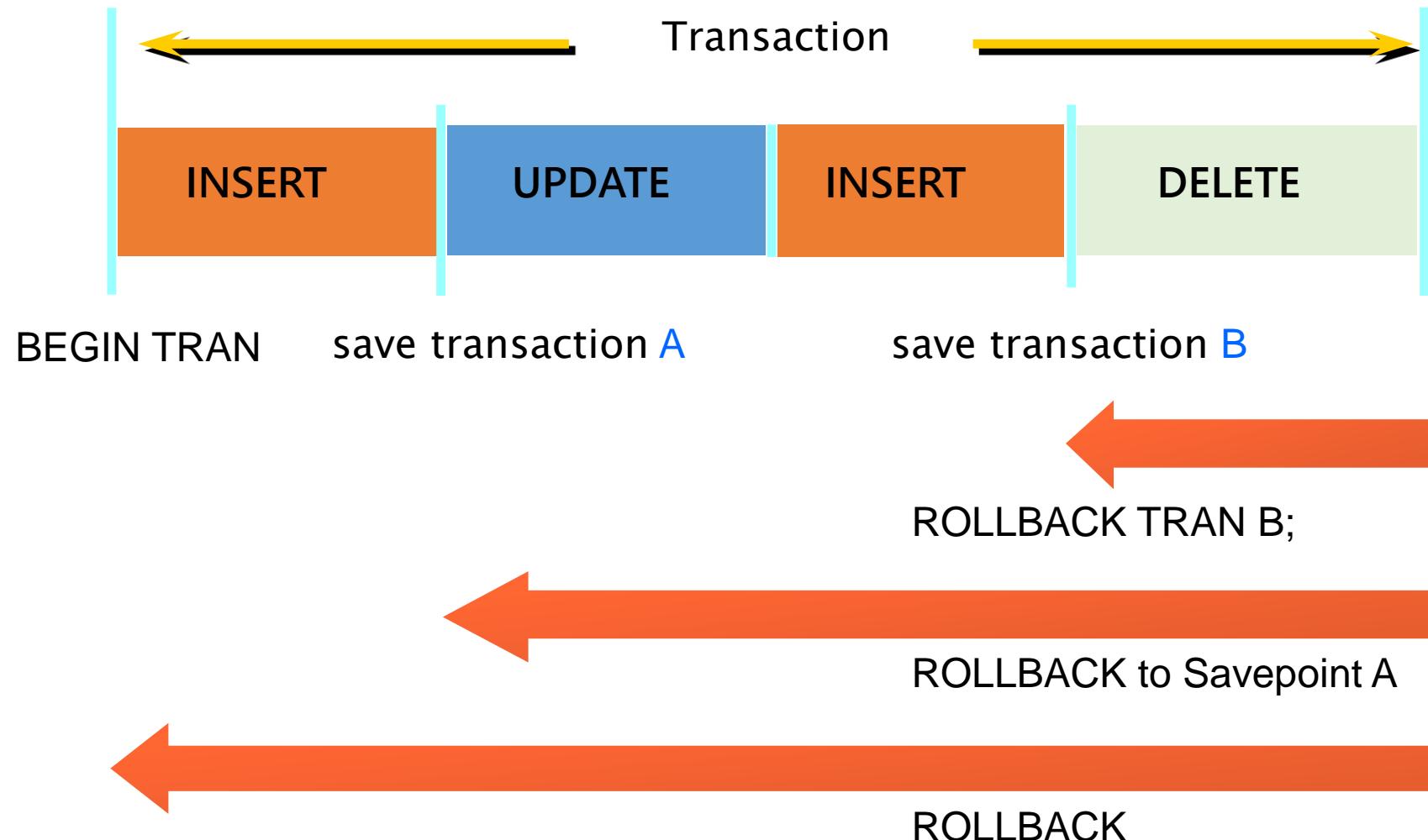
▶ 取消自動確認(DISABLE AUTOCOMMIT)

```
SET AUTOCOMMIT = {0|1}; /*預設為:1, 自動確認*/  
SELECT @@AUTOCOMMIT;      /*查詢目前設定值 */
```

- 使用安全交易，可執行一連串的交易

```
SET AUTOCOMMIT = 0;                      /* 安全交易控制開始 */  
INSERT INTO TX VALUES(NULL,NOW());/* 交易開始 */  
SELECT * FROM TX;  
ROLLBACK;                                /* 交易倒回 */  
SELECT * FROM TX;  
/* ...仍可繼續其他交易的進行 */  
  
SET AUTOCOMMIT = 1; /* 安全交易控制結束 */
```

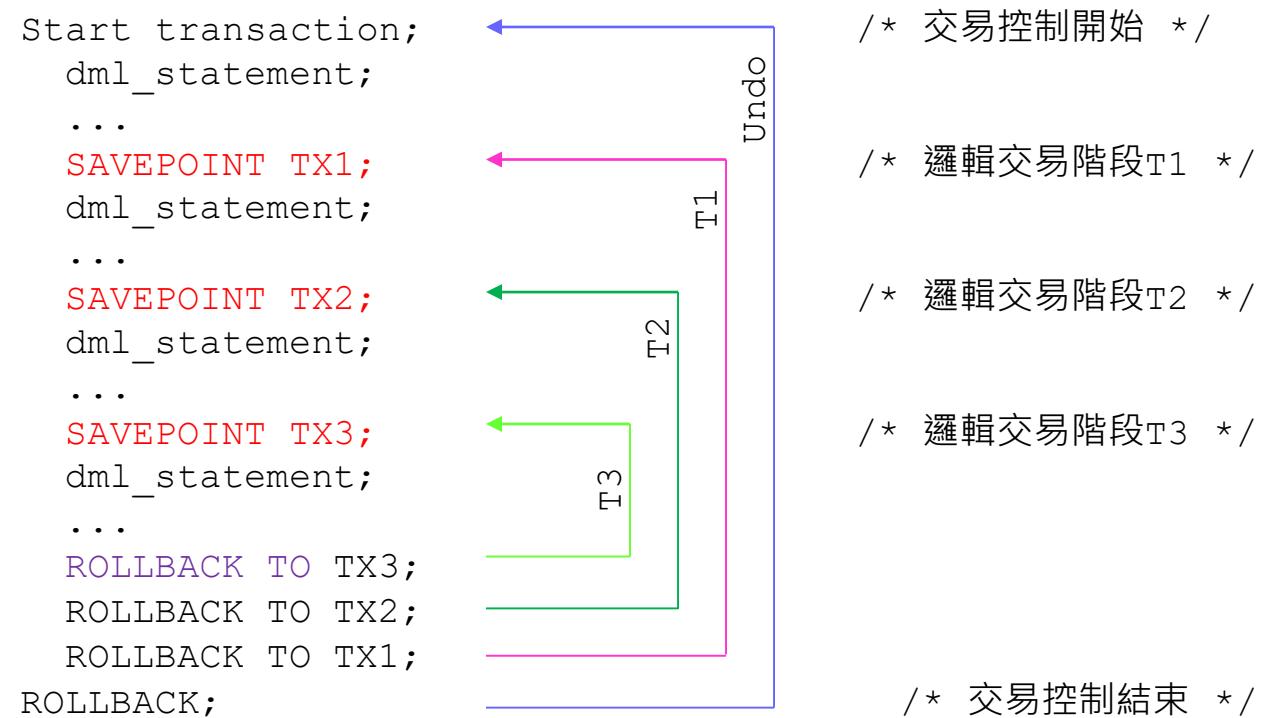
交易控制(Controlling Transactions)



交易儲存點(SAVEPOINT)

▶ 交易儲存點

- 在交易中，可以有很多個statement，用邏輯的交易階段控制有助於交易的進行，儲存點即是實作的方法。



Module 27.

資料庫安全管理

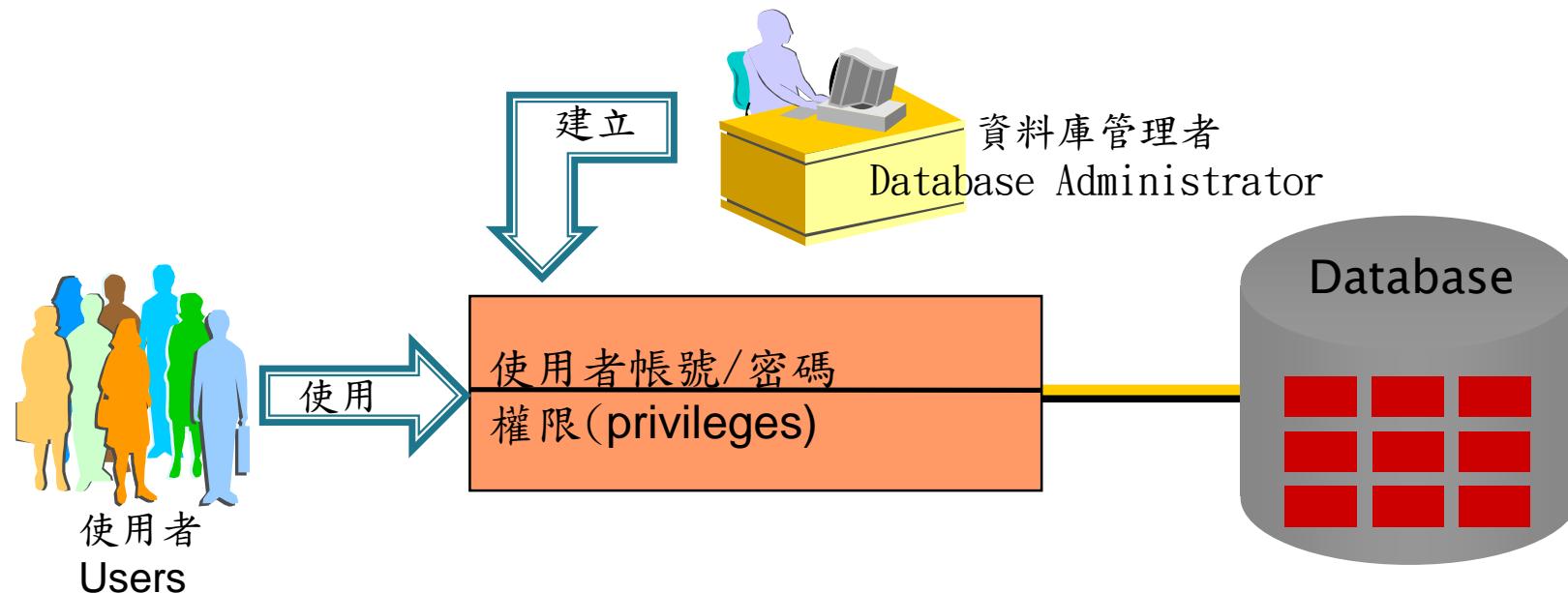
- 27-1: MySQL 資料庫安全管理
- 27-2: MySQL 帳號與權限管理
- 27-3: 連線MySQL資料庫

資料庫安全(Database Security)

- ▶ 資料庫安全主要有二大議題：
 - 系統安全(System security)
 - 資料庫系統層次的連線許可與使用權限
 - 如：使用者帳號/密碼，磁碟空間的使用及使用者在系統內可以操作的事項
 - 資料安全(Data security)
 - 使用者對資料庫物件的存取與使用，及對資料庫物件可以的操作方式

使用者權限管控(Controlling User Access)

- 在多人使用的環境中，必須要建立資料庫存取與使用的安全機制
 - 認證(Authentication) & 授權(Authorization)



權限(Privileges)

- ▶ 權限(Privileges)是執行特定SQL命令的權利
- ▶ 系統權限(System privileges)
 - 設定系統安全(System security)
 - 存取資料庫的權利
- ▶ 物件權限(Object privileges)
 - 設定資料安全(Data security)
 - 維護資料庫物件的權利

MySQL的管理方式

- ▶ MySQL利用帳號及權限來管理資料庫的安全
 - 帳號:控管使用者連線
 - 權限:管理使用者對資料庫中的資料存取
- ▶ MySQL權限系統的主要功能：
 - 由使用者帳號及使用者的連接主機進行認證
 - 再給予通過認證的使用者資料庫的存取權限
 - 如新增、修改、刪除、選取等的權限
(delete, update, select, insert, drop, alter, create)

MySQL 使用者帳號(User Account)

- ▶ 使用者須利用一個使用者帳號來連線
- ▶ 使用者帳號由兩部分組成
 - UserName@HostName
 - UserName : 使用者名稱
 - HostName : 可登入 MySQL 的機器名稱,
% 任何主機, localhost 本機
 - MySQL 資料庫權限設定會將帳號的權限設定給主機或遠端主機
 - root@% - 可在任一台主機登入的帳號root
 - tony@localhost - 只能在本機登入的帳號 tony

建立使用者帳號(Create an User Account)

```
use mysql;
```

```
CREATE USER UserAccount [IDENTIFIED BY 'Password'];
UserAccount : UserName@HostName
```

```
mysql> CREATE USER mary@localhost;
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> SELECT HOST, USER FROM USER;
+-----+-----+          檢視現有使用者
| HOST    | USER   |
+-----+-----+
| %       | tony   |
| 127.0.0.1 | root   |
| ::1     | root   |
| localhost | tony   |
| localhost | root   |
+-----+-----+
5 rows in set (0.00 sec)
```

```
mysql> SELECT HOST, USER FROM USER;
+-----+-----+
| HOST    | USER   |
+-----+-----+
| %       | tony   |
| 127.0.0.1 | root   |
| ::1     | root   |
| localhost | tony   |
| localhost | mary   |
| localhost | root   |
+-----+-----+
6 rows in set (0.00 sec)
```

在localhost使用mary登入伺服器，無任何權限，但TEST資料庫除外

405

建立使用者帳號(Create an User Account)

- ▶ 建立使用者帳號/密碼

```
mysql> CREATE USER mary1@localhost IDENTIFIED BY '1234';
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> SELECT HOST, USER, PASSWORD FROM USER;
+-----+-----+
| HOST    | USER   | PASSWORD
+-----+-----+
| localhost | root   | *6E3E8EA42CA5C64539019349DC9CCB7F0C233550 |
| 127.0.0.1 | root   | *6E3E8EA42CA5C64539019349DC9CCB7F0C233550 |
| ::1      | root   | *6E3E8EA42CA5C64539019349DC9CCB7F0C233550 |
| %        | tony   | *732FFF3F2E6794510CDBB851251B016D90EED29 |
| localhost | tony   | *732FFF3F2E6794510CDBB851251B016D90EED29 |
| localhost | mary   | *
| localhost | mary1  | *0DCA45A7255ABCB6D9413F73E63EB50A7AF67C8A |
+-----+-----+
7 rows in set (0.00 sec)
```

PASSWORD 改名為
authentication_string

MySQL 帳號權限管理

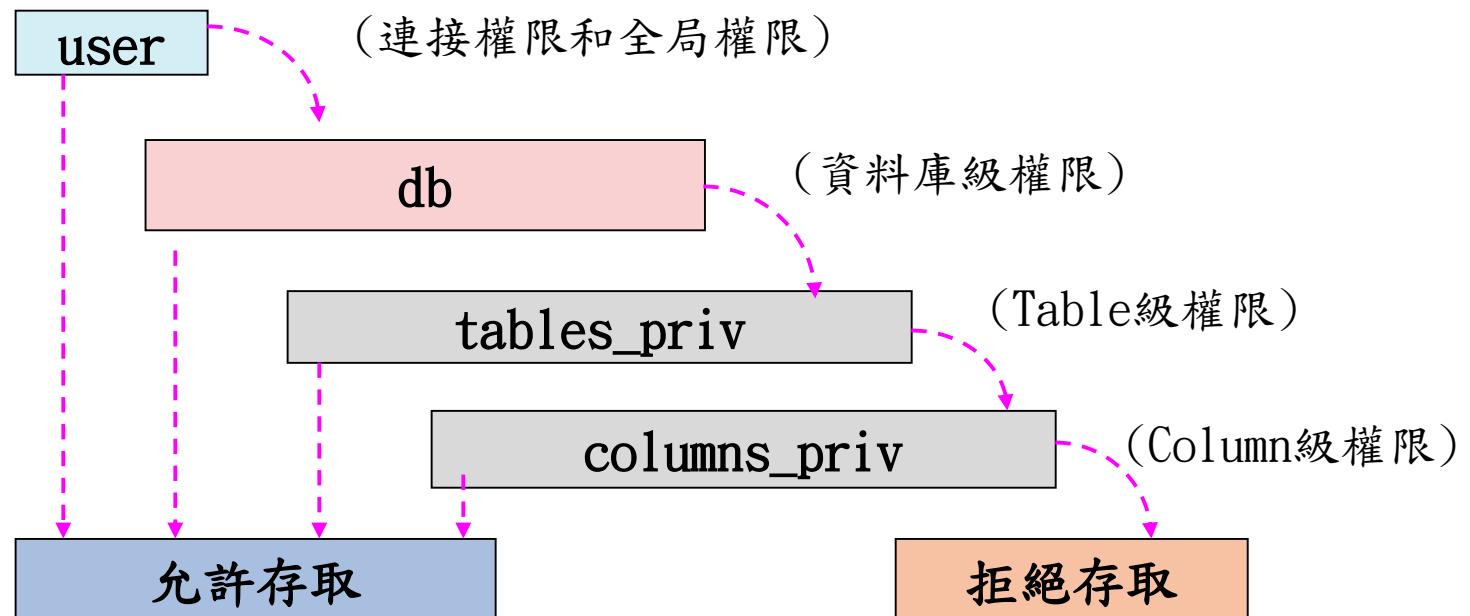
- ▶ 帳號及權限記錄在mysql資料庫中的資料表

Table Name	儲存內容
user	紀錄所有使用者帳號密碼及在Server中的權限 (全域性的權限)
db	紀錄使用者對一個資料庫的權限
tables_priv	紀錄使用者對資料表的權限
columns_priv	紀錄使用者對特定欄位的權限
Procs_priv	紀錄使用者對store procedure的權限

```
USE mysql;  
SHOW TABLES;
```

MySQL的權限驗證處理

- ▶ MySQL 權限的驗證處理順序



權限授權 (Grant 命令)

▶ 授權命令

```
GRANT [ALL| privs [columns]] [, . . .]
ON {table | * | *.* | database.*}
TO user, . . .
[WITH GRANT OPTION];
```

- ALL 所有權限
- privs 個別權限
- columns 指定欄位
- ON 授權等級
- WITH GRANT OPTION 再授權的權力
- 任何使用者均可存取以 test 開頭的資料庫

授權等級(Grant Level)

- ▶ 授權等級(由ON子句設置)
 - 全域授權與特定物件授權

授權等級	說明	權限資料表	語法
全域授權 (Server)	所有資料庫	mysql.users	GRANT ALL ON *.* REVOKE ALL ON *.*
特定物件授權 Database	特定資料庫	mysql.db	GRANT ALL ON database.* GRANT ALL ON * ←表示目前的資料庫 REVOKE ALL ON database.*
特定物件授權 Table	特定資料表	mysql.tables_priv	GRANT ALL ON database.table REVOKE ALL ON database.table
特定物件授權 Column	資料表中特定欄位	mysql.columns_priv	GRANT ALL ON database.table(column) REVOKE ALL ON database.table(column)

MySQL 權限-使用者層級權限

▶ 授權給一般使用者的權限

權限	權限說明
ALTER	可修改表和索引的結構
CREATE	創建資料庫和 Table
DELETE	刪除 Table 資料
DROP	刪除 Table
INDEX	建立或刪除索引
INSERT	新增 Table 資料
REFERENCES	(暫時不支持)
SELECT	查詢 Table 資料
UPDATE	更新 Table 資料
ALL	所有權限，但不包括 GRANT。
USAGE	“無權限”的權限

MySQL 權限-管理者層級權限

▶ 授權給需擔任系統管理者的權限

權限	權限說明
CREATE TEMPORARY TABLES	創建臨時 Table
FILE	允許使用SELECT ... INTO OUTFILE and LOAD DATA INFILE
GRANT OPTION	可把本帳號的權限授予其它用戶
LOCK TABLES	鎖定指定 Table
PROCESS	查看正在執行中的Session資訊
RELOAD	重新加載權限表或刷新日誌及緩衝區
REPLICATION CLIENT	可查詢主/從伺服器主機名
REPLICATION SLAVE	運行一個鏡像從伺服器
SHOW DATABASES	可執行SHOW DATABASES指令
SHUTDOWN	關閉資料庫伺服器
SUPER	可用kill終止Session 以及進行超級用戶操作

MySQL權限管理-新建使用者帳號&授權

- ▶ GRANT指令也可以新增使用者與權限的調整
 - 當使用者不存在時會建立帳號，若存在則會變更權限

```
GRANT privileges (columns) #privileges表示授予的權限，columns表示作用的列(可選)
ON what #設置權限級別，全局級、資料庫級、Table級和Column級
TO account #權限授予的用戶，用“user_name” @ “host_name” 用戶名@主機名
IDENTIFIED BY 'password' #設置用戶帳號密碼
```

```
mysql>grant all on db.* to 'test'@'localhost'
        identified by 'test';
```

```
mysql>grant all on db.* to 'test'@'%'
        identified by 'test';
```

GRANT 權限
ON 資料庫. 資料表
TO 帳號@主機
IDENTIFIED BY ‘密碼’

新增使用者帳號並授權

- ▶ 新增使用者(Foreign user)
 - 新增 whitedog 使用者並指定密碼為 abcxyz，讓 whitedog 僅可從 192.168.1.100 連到此電腦的 MySQL，並操作任何 Database 及 Table。

```
mysql> grant all on *.* to whitedog@192.168.1.100  
        identified by 'abcxyz';
```

- ▶ 新增 twocats 使用者並指定密碼為 abcxyz，讓 twocats 可從任何 IP 連到此電腦的 MySQL，並操作任何 Database 及 Table

```
mysql> grant all on *.* to twocats@"%"  
        identified by 'abcxyz';
```

新增使用者帳號並授權

- 新增 lazycat 使用者並指定密碼為 sleeping，讓 lazycat 可從任何 IP 連到此電腦的 MySQL，並對 yd702 資料庫底下的所有資料表有 select, insert, update, delete, create, drop 之權限

```
mysql> grant select,insert,update,delete,create,drop
      on yd702.*
      to lazycat@'%'
      identified by 'sleeping';
```

- 將 MySQL 內所有的資料庫及資料表權限，都開放給 whitedog，且 whitedog 密碼為 IlovePU(注意：大小寫有別)

```
mysql> grant all on *.* to whitedog@localhost
      identified by 'IlovePU';
```

移除權限(Revoking a Privilege)

- ▶ 使用「revoke on」指令移除使用者權限

```
REVOKE [ALL] privs[columns],...
ON {table | * | *.* | database.*}
FROM user, ...;
```

SQL statement	說明
REVOKE ALL ON sample.bonus FROM odxxx;	撤消sample database bonus資料表所有的權限
REVOKE GRANT OPTION ON sample.emp FROM odxxx;	撤消使用者odxxx授予其他人權限的能力

- ▶ Revoke ON 指令只能移除帳號權限，但無法將帳號刪除。

修改帳號密碼(Changing Password)

▶ 修改使用者帳號密碼

```
SET PASSWORD = PASSWORD(password);
```

```
SET PASSWORD FOR user = PASSWORD(password);
```

```
UPDATE mysql.user
SET PASSWORD = PASSWORD('password')
WHERE USER = 'user';
FLUSH PRIVILEGES;
```

- user 使用者名稱
- PASSWORD() 密碼函數
- password 使用者密碼
- FLUSH PRIVILEGES 同步更新權限(記憶體與權限表)
- 必需具備存取mysql database的權限才能更改密碼

修改帳號密碼(Changing Password)

- ▶ 變更目前連線使用者密碼

```
SET PASSWORD = password('mary123');
```

- ▶ 變更指定使用者密碼

```
SET PASSWORD FOR odxxx = password('newpwd');
```

```
UPDATE MYSQL.USER
SET PASSWORD = password('')
WHERE USER = odxxx;
FLUSH PRIVILEGES;
```

- ▶ 使用USER()函數變更目前連線使用者密碼

```
UPDATE MYSQL.USER
SET PASSWORD = password('')
WHERE USER = user();
FLUSH PRIVILEGES;
```

移除使用者帳號(Dropping a User)

- ▶ 刪除一個使用者帳號

```
DROP USER UserAccount;
```

```
mysql> drop user mary1@localhost;
Query OK, 0 rows affected (0.00 sec)
```

- ▶ 刪除一個使用者帳號

```
use mysql
delete from user where user='mary';
flush privileges;
```

連線到遠端MySQL資料庫

▶ 連線到 MySQL 資料庫

```
mysql [.exe] [options] [database]
```

options	縮寫	說明
--help	-?	線上輔助說明
--user	-u	使用者名稱
--password	-p	使用者密碼
--host	-h	伺服器位置(IP或HOST)
--port	-P	連接埠(3306)

```
mysql --user=odxxx --password=odxxx --host=10.130.11.100  
or  
mysql -u odxxx -p odxxx -h 10.130.11.100
```

環境變數PATH要加入
C:\Program Files\MySQL\MySQL Server 8.0\bin

密碼若以 -p 縮寫表示時，密碼緊接其後

Module 28.

檢視表1

- 28-1: View概念**
- 28-2: CREATE View語法**
- 28-3: 建立View範例**

View(視觀表)

- ▶ View是SELECT查詢結果動態組合生成的虛擬資料表(Virtual Table)

EMP Table

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		17-NOV-81	5000		10
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
					7600	20-FEB-81	1600
					1600	300	30
EMPNO	ENAME	JOB					
7839	KING	PRESIDENT			0	0	30
7782	CLARK	MANAGER			0	500	30
7934	MILLER	CLERK			0		20
7670	ADAMS	CLERK	7782	12-JAN-83	1100		20
7934	MILLER	CLERK	7782	23-JAN-82	1300	10	

- ▶ 視觀表(View)物件中只有查詢定義(SELECT)，而該查詢所使用的資料表稱為基底資料表(Based Tables)

```
CREATE VIEW empvu10
AS
SELECT empno, ename, sal, job
FROM emp
WHERE deptno = 10;
```

基底資料表
(Based Tables)

- 視觀表所呈現的結果和資料表(Table)類似
 - 由 rows and columns 所組成
 - Views本身並不儲存任何的資料，基底資料表(Based Tables)才是真正儲存資料的地方
- ▶ 使用上有如的資料表(table)
 - 所有的查詢語法(SELECT)都可以在View上操作
 - DML-新增、刪除、更新資料，則會受限制

資料表(table)是一種實體結構(Physical Structure)

視觀表(View)是一種虛擬結構(Virtual Structure)

- ▶ 可以加強資料庫的安全性
 - View 可以將實體資料表結構隱藏起來，同時限制使用者只可以檢視及使用哪些資料表欄位。
 - 檢視表可以是唯讀的，亦即外部使用者無法直接透過 View 去修改內部資料。
- ▶ 將複雜的查詢(SELECT)包裝在 View 中，可以簡化查詢的複雜度
- ▶ 當資料表結構有變更時，只需要更改 View 的設定，而不需更改程式
- ▶ 同一個資料表可建立多個視觀表

建立View

► CREATE VIEW 命令

```
CREATE [OR REPLACE] VIEW view_name [(column[,...n ])]
AS
<select_statement>
[WITH CHECK OPTION];
```

- 如果加上「**OR REPLACE**」子句的意思就是**若同名的 View 已經存在就覆蓋取代它**
 - 如果 View 已存在，可以把 **CREATE OR REPLACE VIEW** 當做是 **ALTER VIEW**
- 如果 View 不存在，**CREATE OR REPLACE VIEW** 如同 **CREATE VIEW**
- 基底資料表(Based Tables)
 - 已存在資料表或檢視表
- 不可以使用 **subquery** 在**FROM** 子句中
- 不要有**ORDER BY**子句，若存在**ORDER BY** 也將被忽略

建立視觀表(Creating a View)

- ▶ 使用基底資料表(Based Tables)的欄位名稱

```
mysql> CREATE VIEW empvu10
      -> AS
      ->     SELECT *
      ->     FROM   emp
      ->     WHERE  deptno = 10;
```

```
mysql> SELECT * FROM empvu10;
+-----+-----+-----+-----+-----+-----+-----+-----+
| EMPNO | ENAME | JOB       | MGR    | HIREDATE          | SAL     | COMM   | DEPTNO |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 7782  | CLARK  | MANAGER   | 7839   | 1981-06-09 00:00:00 | 2450.00 | NULL   | 10      |
| 7839  | KING    | PRESIDENT | NULL    | 1981-11-17 00:00:00 | 5000.00 | NULL   | 10      |
| 7934  | MILLER | CLERK     | 7782   | 1982-01-23 00:00:00 | 1300.00 | NULL   | 10      |
+-----+-----+-----+-----+-----+-----+-----+-----+
```

建立視觀表-設定欄位名稱

```
mysql> CREATE VIEW salvu20 (EMPLOYEE_NO, EMPLOYEE, ANNUAL_SAL)
-> AS
-> SELECT empno, ename, sal*12
-> FROM emp
-> WHERE deptno = 20;
```

```
mysql> desc salvu20;
+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| EMPLOYEE_NO | int(11) | NO  |   | NULL    |       |
| EMPLOYEE    | varchar(10) | YES |   | NULL    |       |
| ANNUAL_SAL  | decimal(9,2) | YES |   | NULL    |       |
+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

```
mysql> select * from salvu20;
+-----+-----+-----+
| EMPLOYEE_NO | EMPLOYEE | ANNUAL_SAL |
+-----+-----+-----+
|      7369 | SMITH    | 9600.00   |
|      7566 | JONES    | 35700.00  |
|      7788 | SCOTT    | 36000.00  |
|      7876 | ADAMS    | 13200.00  |
|      7902 | FORD     | 36000.00  |
+-----+-----+-----+
5 rows in set (0.00 sec)
```

建立視觀表-使用別名當欄位名稱

```
mysql> CREATE VIEW salvu30
-> AS
-> SELECT empno EMPLOYEE_NUMBER, ename NAME, sal SALARY
-> FROM emp
-> WHERE deptno = 30;
```

```
mysql> desc salvu30;
+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| EMPLOYEE_NUMBER | int(11) | NO  |   | NULL    |       |
| NAME            | varchar(10) | YES |   | NULL    |       |
| SALARY          | decimal(7,2) | YES |   | NULL    |       |
+-----+-----+-----+-----+
```

```
mysql> select * from salvu30;
+-----+-----+-----+
| EMPLOYEE_NUMBER | NAME   | SALARY |
+-----+-----+-----+
|        7499 | ALLEN | 1600.00 |
|        7521 | WARD  | 1250.00 |
|        7654 | MARTIN | 1250.00 |
|        7698 | BLAKE | 2850.00 |
|        7844 | TURNER | 1500.00 |
|        7900 | JAMES | 950.00  |
+-----+-----+-----+
```

視觀表類別(View types)

- ▶ 簡單視觀表(Simple Views)
 - 只有一個基底資料表(Based Table)-沒有 JOIN
 - 沒有使用函數或做資料分組
 - 可更新的檢視表(Updateable View)
- ▶ 複雜視觀表(Complex Views)：
 - 一個以上的基底資料表(Based Table)- JOIN
 - 有使用函數或做資料分組
 - 不可更新的檢視表

建立視觀表 - 複雜視觀表(Complex Views)

```
mysql> CREATE VIEW dept_sum_vu(name, minsal, maxsal, avgsal)
-> AS
->   SELECT d.dname, MIN(e.sal), MAX(e.sal), AVG(e.sal)
->   FROM emp e JOIN dept d ON(e.deptno = d.deptno)
->   GROUP BY d.dname;
```

```
mysql> desc dept_sum_vu;
+-----+-----+-----+-----+-----+
| Field | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| name  | varchar(14) | YES  |     | NULL    |       |
| minsal | decimal(7,2) | YES  |     | NULL    |       |
| maxsal | decimal(7,2) | YES  |     | NULL    |       |
| avgsal | decimal(11,6) | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+
```

```
mysql> select * from dept_sum_vu;
+-----+-----+-----+-----+
| name      | minsal | maxsal | avgsal |
+-----+-----+-----+-----+
| ACCOUNTING | 1300.00 | 5000.00 | 2916.666667 |
| RESEARCH   | 800.00  | 3000.00 | 2175.000000 |
| SALES      | 950.00  | 2850.00 | 1566.666667 |
+-----+-----+-----+-----+
```

內嵌視觀表(Inline Views)

- ▶ 內嵌視觀表(Inline Views)是在FROM子句內的SELECT命令，用來定義外部 SELECT 命令的一個資料來源(Table value sub-query)

```
mysql> SELECT a.ename, a.sal, a.deptno, b.avgsal
    -> FROM emp a Join (SELECT deptno, avg(sal) avgsal
    ->                      FROM emp
    ->                      GROUP BY deptno) b
    ->                  ON (a.deptno = b.deptno)
    -> WHERE a.sal < b.avgsal;
+-----+-----+-----+-----+
| ename | sal      | deptno | avgsal      |
+-----+-----+-----+-----+
| CLARK | 2450.00 |      10 | 2916.666667 |
| MILLER | 1300.00 |      10 | 2916.666667 |
| ADAMS | 1100.00 |      20 | 2518.750000 |
| MARTIN | 1250.00 |      30 | 1900.000000 |
| TURNER | 1500.00 |      30 | 1900.000000 |
+-----+-----+-----+-----+
```

Module 29.

檢視表2

- 29-1: 使用View查詢
- 29-2: 使用View維護資料
- 29-3: 修改與刪除View

視觀表(View)的使用

- ▶ View 就像是一個Table, 大部份使用Table可以完成的工作，也可以透過View來完成
- ▶ 視觀表(View)除了可以查詢外也可以透過它來做資料維護
 - 使用View來新增、修改或刪除資料
- ▶ 可更新的視觀表(Updateable View)
 - 不可以包含計算或函數的欄位
 - 只有一個基底資料表(Based Tables)

視觀表(View) - 查詢

- ▶ 查詢功能與資料表完全一樣

```
mysql> SELECT COUNT(*), SUM(SALARY) SUM_SAL_30, AVG(SALARY) AVG_SAL_30
      -> FROM SALVU30;
+-----+-----+-----+
| COUNT(*) | SUM_SAL_30 | AVG_SAL_30 |
+-----+-----+-----+
|       6 |     9400.00 | 1566.666667 |
+-----+-----+-----+
1 row in set (0.00 sec)
```

```
mysql> SELECT NAME, AVGSAL
      -> FROM DEPT_SUM_VU
      -> WHERE AVGSAL>2000;
+-----+-----+
| NAME        | AVGSAL      |
+-----+-----+
| ACCOUNTING | 2916.666667 |
| RESEARCH    | 2175.000000 |
+-----+-----+
2 rows in set (0.00 sec)
```

可更新的視觀表(Updateable View)

- ▶ 查詢VIEW是否可執行DML命令 ?
 - INFORMATION_SCHEMA.VIEWS

```
use mysql;
```

```
mysql> select table_name, is_updatable
    -> from information_schema.views;
+-----+-----+
| table_name | is_updatable |
+-----+-----+
| dept_sum_vu | NO |
| empvu10 | YES |
| salvu20 | YES |
| salvu30 | YES |
| actor_info | NO |
| customer_list | YES |
| film_list | NO |
| nicer_but_slower_film_list | NO |
| sales_by_film_category | NO |
| sales_by_store | NO |
| staff_list | YES |
+-----+-----+
```

視觀表(View)-資料更新

```
mysql> UPDATE salvu30
-> SET SALARY = 2000
-> WHERE EMPLOYEE_NUMBER=7499;
Query OK, 1 row affected (0.03 sec)
Rows matched: 1    Changed: 1    Warnings: 0
```

VIEW

```
mysql> SELECT * FROM SALVU30;
+-----+-----+-----+
| EMPLOYEE_NUMBER | NAME   | SALARY |
+-----+-----+-----+
|      7499 | ALLEN | 2000.00 |
|      7521 | WARD   | 1250.00 |
|      7654 | MARTIN | 1250.00 |
|      7698 | BLAKE  | 2850.00 |
|      7844 | TURNER | 1500.00 |
|      7900 | JAMES   | 950.00 |
+-----+-----+-----+
6 rows in set (0.00 sec)
```

```
mysql> SELECT EMPNO, ENAME, JOB, SAL
-> FROM EMP
-> WHERE EMPNO=7499;
+-----+-----+-----+-----+
| EMPNO | ENAME | JOB       | SAL    |
+-----+-----+-----+-----+
| 7499 | ALLEN | SALESMAN | 2000.00 |
+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

Based Table

視觀表(View)-資料更新

```
mysql> UPDATE SALVU20
-> SET EMPLOYEE='MARY'
-> WHERE EMPLOYEE_NO=7902;
Query OK, 1 row affected (0.03 sec)
Rows matched: 1    Changed: 1    Warnings: 0
```

非計算欄位
可更新

```
mysql> SELECT * FROM SALVU20;
+-----+-----+-----+
| EMPLOYEE_NO | EMPLOYEE | ANNUAL_SAL |
+-----+-----+-----+
|      7369 | SMITH     |   9600.00 |
|      7566 | JONES     |  35700.00 |
|      7788 | SCOTT     |  36000.00 |
|      7876 | ADAMS     |  13200.00 |
|      7902 | MARY      |  36000.00 |
+-----+-----+-----+
5 rows in set (0.00 sec)
```

計算欄位
不可更新

```
mysql> UPDATE SALVU20
-> SET ANNUAL_SAL=48000
-> WHERE EMPLOYEE_NO=7902;
ERROR 1348 (HY000): Column 'ANNUAL_SAL' is not updatable
```

視觀表(View)-新增資料

```
mysql> DESC EMP;
+-----+-----+-----+-----+-----+
| Field | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| EMPNO | int(11)   | NO   | PRI | NULL    |       |
| ENAME | varchar(10)| YES  |     | NULL    |       |
| JOB   | varchar(9) | YES  |     | NULL    |       |
| MGR   | int(11)   | YES  | MUL | NULL    |       |
| HIREDATE | datetime | YES  |     | NULL    |       |
| SAL   | decimal(7,2)| YES  |     | NULL    |       |
| COMM  | decimal(7,2)| YES  |     | NULL    |       |
| DEPTNO | int(11)   | NO   | MUL | NULL    |       |
+-----+-----+-----+-----+-----+
8 rows in set (0.00 sec)
```

```
mysql> DESC EMPVU10;
+-----+-----+-----+-----+-----+
| Field | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| EMPNO | int(11)   | NO   |     | NULL    |       |
| ENAME | varchar(10)| YES  |     | NULL    |       |
| JOB   | varchar(9) | YES  |     | NULL    |       |
| MGR   | int(11)   | YES  |     | NULL    |       |
| HIREDATE | datetime | YES  |     | NULL    |       |
| SAL   | decimal(7,2)| YES  |     | NULL    |       |
| COMM  | decimal(7,2)| YES  |     | NULL    |       |
| DEPTNO | int(11)   | NO   |     | NULL    |       |
+-----+-----+-----+-----+-----+
8 rows in set (0.00 sec)
```

視觀表(View)-新增資料

```
mysql> INSERT INTO EMPVU10 (EMPNO, ENAME, SAL, DEPTNO)
-> VALUES (9700, 'KEN', 3800, 10);
Query OK, 1 row affected (0.04 sec)
```

```
mysql> SELECT * FROM EMPVU10;
+-----+-----+-----+-----+-----+-----+-----+-----+
| EMPNO | ENAME  | JOB   | MGR   | HIREDATE        | SAL    | COMM  | DEPTNO |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 7782  | CLARK  | MANAGER | 7839  | 1981-06-09 00:00:00 | 2450.00 | NULL  | 10     |
| 7839  | KING   | PRESIDENT | NULL  | 1981-11-17 00:00:00 | 5000.00 | NULL  | 10     |
| 7934  | MILLER | CLERK   | 7782  | 1982-01-23 00:00:00 | 1300.00 | NULL  | 10     |
| 9700  | KEN    | NULL   | NULL  | NULL            | 3800.00 | NULL  | 10     |
+-----+-----+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

視觀表(View)-新增資料

```
mysql> INSERT INTO SALVU20
-> VALUES(9800, 'JACKSON', 24000);
ERROR 1471 (HY000): The target table SALVU20 of the
INSERT is not insertable-into
```

```
mysql> DESC SALVU20;
+-----+-----+-----+-----+-----+
| Field      | Type       | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| EMPLOYEE_NO | int(11)    | NO   |     | NULL    |       |
| EMPLOYEE    | varchar(10) | YES  |     | NULL    |       |
| ANNUAL_SAL  | decimal(9,2) | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

有計算欄位
不可新增資料

視觀表(View)-刪除資料

```
mysql> DELETE FROM EMPVU10  
      -> WHERE EMPNO=9700;  
Query OK, 1 row affected  
(0.10 sec)
```

```
mysql> DELETE FROM SALVU30  
      -> WHERE NAME='JAMES';  
Query OK, 1 row affected (0.03  
sec)
```

```
mysql> SELECT * FROM SALVU20;  
+-----+-----+-----+  
| EMPLOYEE_NO | EMPLOYEE | ANNUAL_SAL |  
+-----+-----+-----+  
| 7369 | SMITH | 9600.00 |  
| 7566 | JONES | 35700.00 |  
| 7788 | SCOTT | 36000.00 |  
| 7876 | ADAMS | 13200.00 |  
| 7902 | MARY | 36000.00 |  
+-----+-----+-----+  
5 rows in set (0.00 sec)
```

```
mysql> DELETE FROM SALVU20  
      -> WHERE EMPLOYEE_NO='7902';  
ERROR 1451 (23000): Cannot delete or update a parent row: a foreign key constraint fails (`exp`.`emp`, CONSTRAINT `EMP_MGR_FK` FOREIGN KEY (`MGR`) REFERENCES `mp`(`EMPNO`))
```

```
mysql> DELETE FROM SALVU20  
      -> WHERE EMPLOYEE_NO=7369;  
Query OK, 1 row affected (0.04 sec)
```

違反基底資料表
的資料檢查條件

複雜視觀表(Complex Views)

```
mysql> select * from dept_sum_vu;
+-----+-----+-----+
| name | minsal | maxsal | avgsal |
+-----+-----+-----+
| ACCOUNTING | 1300.00 | 5000.00 | 2916.666667 |
| RESEARCH | 1100.00 | 3000.00 | 2518.750000 |
| SALES | 1250.00 | 2850.00 | 1770.000000 |
+-----+-----+-----+
3 rows in set (0.00 sec)
```

無法使用
DML

```
mysql> INSERT INTO dept_sum_vu
-> VALUES('EDUCATION',1200,3000,2800);
ERROR 1471 (HY000): The target table dept_sum_vu of the INSERT is not insertable-into
```

```
mysql> UPDATE dept_sum_vu
-> SET MAXSAL=4800
-> WHERE NAME='SALES';
ERROR 1288 (HY000): The target table dept_sum_vu of the UPDATE is not updatable
```

```
mysql> DELETE FROM dept_sum_vu
-> WHERE NAME='RESEARCH';
ERROR 1288 (HY000): The target table dept_sum_vu of the DELETE is not updatable
```

修改視觀表(Altering the Definition of a View)

- ▶ 修改已建立的視觀表內容
 - 直接修改內容，不需要刪除後再重建
 - 保留原有的權限管控內容
 - 取代整個原有的內容

```
ALTER VIEW view_name [( column [ ,...n ] )]  
AS  
    <select_statement>  
[ WITH CHECK OPTION ];
```

修改視觀表(Altering the Definition of a View)

- ▶ 修改視觀表內容，增加 WITH CHECK OPTION 選項

```
mysql> ALTER VIEW salvu30
      -> AS
      -> SELECT empno EMPLOYEE_NUMBER, ename NAME, sal SALARY, deptno
      -> FROM emp
      -> WHERE deptno = 30
      -> WITH CHECK OPTION;
```

```
mysql> SELECT * FROM SALVU30;
+-----+-----+-----+-----+
| EMPLOYEE_NUMBER | NAME    | SALARY | deptno |
+-----+-----+-----+-----+
|        7499 | ALLEN  | 2000.00 |      30 |
|        7521 | WARD   | 1250.00 |      30 |
|        7654 | MARTIN | 1250.00 |      30 |
|        7698 | BLAKE  | 2850.00 |      30 |
|        7844 | TURNER | 1500.00 |      30 |
+-----+-----+-----+-----+
```

WITH CHECK OPTION 選項

```
mysql> SELECT * FROM SALVU30;
+-----+-----+-----+-----+
| EMPLOYEE_NUMBER | NAME    | SALARY | deptno |
+-----+-----+-----+-----+
|      7499 | ALLEN   | 2000.00 |      30 |
|      7521 | WARD    | 1250.00 |      30 |
|      7654 | MARTIN  | 1250.00 |      30 |
|      7698 | BLAKE   | 2850.00 |      30 |
|      7844 | TURNER  | 1500.00 |      30 |
+-----+-----+-----+-----+
```

```
mysql> DELETE FROM salvu30 WHERE EMPLOYEE_NUMBER=7521;
Query OK, 1 row affected (0.04 sec)
```

```
mysql> UPDATE salvu30
      -> SET SALARY = 2000
      -> WHERE EMPLOYEE_NUMBER=7499;
Query OK, 0 rows affected (0.03 sec)
Rows matched: 1    Changed: 0    Warnings: 0
```

資料更新時不可違反WHERE子句的條件
...
WHERE deptno = 30
WITH CHECK OPTION;

```
mysql> UPDATE salvu30
      -> SET deptno = 10
      -> WHERE EMPLOYEE NUMBER=7499;
ERROR 1369 (HY000): CHECK OPTION failed 'exp.salu30'
```

刪除視觀表(Removing a View)

- ▶ 從資料庫中刪除視觀表
 - 不會刪除任何資料

```
DROP VIEW view_name;
```

```
DROP VIEW dept_sum_vu;
```

Module 30. 索引

30-1: Index

30-2: 建立Index

30-3: 刪除Index

索引(Index)

▶ 什麼是索引

- 是一種檔案(大都是B-Tree結構), 鍵值具有排序的特性
- 用來加快資料的搜尋
- 沒有索引, 搜尋資料以循序的方式進行(Full Table Scan)

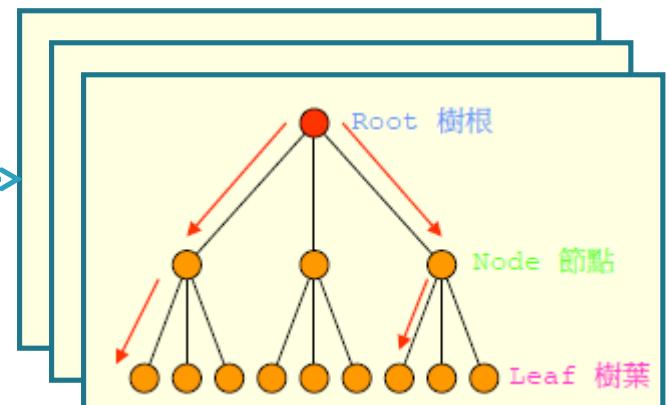
▶ 索引分為三種

- 主鍵索引 (primary key)
- 唯一索引 (unique index)
- 非唯一索引(non-unique index)

資料表(table)

EMPLOYEE_ID	LAST_NAME	FIRST_NAME	SALARY	COMMISSION_PCT	DEPARTMENT_ID
100	King	Steven	24000		90
101	Kochhar	Neena	17000		90
102	De Haan	Lex	17000		90
103	Hunold	Alexander	9000		60
104	Ernst	Bruce	6000		60
107	Lorentz	Diana	4200		60
124	Moursig	Kevin	5800		50
141	Rajs	Trenna	3500		50
142	Davies	Curtis	3100		50
143	Matos	Randall	2600		50
144	Vargas	Peter	2500		50
149	Zlotkey	Eleni	10500	.2	80
174	Abel	Ellen	11000	.3	80
176	Taylor	Jonathon	8600	.2	80
178	Grant	Kimberely	7000	.15	
200	Whalen	Jennifer	4400		10
201	Hartstein	Michael	13000		20
202	Fay	Pat	6000		20
205	Higgins	Shelley	12000		110
206	Gietz	William	8300		110

索引(Index)



索引的特色

- ▶ 索引(Index)可以由一個欄位或多個欄位所構成
- ▶ 索引(Index)的使用與維護由系統負責
- ▶ 建立PK, UK時, 系統會自動建立索引(Index)
- ▶ 使用者可以自行建立索引(Index)
- ▶ MySQL中建立FK時, 需建立索引(Index)方便搜尋
- ▶ PK, UK被捨棄時, 索引(Index)一併刪除
- ▶ FK被捨棄時, 並不會連同索引(Index)一併刪除
- ▶ 刪除資料表時, 所有的索引(Index)一併刪除

建立索引的考量

- ▶ 適合建立索引的時機
 - 經常出現在 WHERE 句子中的欄位
 - 多個資料表連結時(PK與FK之關係)
 - 經常使用 MAX(), MIN() Group Functions
 - 值域分布很廣的欄位
 - 空值多的欄位
 - 查詢比低於2%時(筆數/總比數)
- ▶ 不適合建立索引的時機
 - 不經常出現在WHERE句子中的欄位
 - 使用像%string%的搜尋
 - 查詢比大於5%時(筆數/總比數)
 - 資料表異動頻繁時

建立索引(Creating an Index)

► 建立索引

```
CREATE [UNIQUE] INDEX index  
ON TableName(column [length] [ASC|DESC], .);
```

- 可以指定某一個欄位為建立索引的欄位
- 字串型態欄位的部份資料建立索引
- 索引資料，可由小到大，或由大到小排列
- $\text{Index}(A, B) \neq \text{Index}(A) + \text{Index}(B)$
- $\text{Index}(A, B) \neq \text{Index}(B, A)$
- $\text{Index}(A) \neq \text{Index}(A, B)$

► 查看索引

```
SHOW INDEX FROM TableName;
```

建立索引(Creating an Index)

```
mysql> SHOW INDEX FROM emp;
+-----+-----+-----+-----+
| Table | Non_unique | Key_name      | Seq_in_index | Column_name | ...
+-----+-----+-----+-----+
| emp   |          0 | PRIMARY       |            1 | EMPNO      | ...
| emp   |          1 | EMP_DEPTNO_FK |            1 | DEPTNO     | ...
| emp   |          1 | EMP_MGR_FK    |            1 | MGR        | ...
+-----+-----+-----+-----+
3 rows in set (0.02 sec)
```

```
mysql> CREATE INDEX ind_emp_ename ON emp(ename);
Query OK, 0 rows affected (0.23 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

```
mysql> SHOW INDEX FROM emp;
+-----+-----+-----+-----+
| Table | Non_unique | Key_name      | Seq_in_index | Column_name | ...
+-----+-----+-----+-----+
| emp   |          0 | PRIMARY       |            1 | EMPNO      | ...
| emp   |          1 | EMP_DEPTNO_FK |            1 | DEPTNO     | ...
| emp   |          1 | EMP_MGR_FK    |            1 | MGR        | ...
| emp   |          1 | ind_emp_ename |            1 | ENAME      | ...
+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

建立索引 - 複合索引鍵

```
mysql> CREATE INDEX ind_emp_dept_job ON emp(deptno, job);
Query OK, 0 rows affected (0.25 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

```
mysql> SHOW INDEX FROM emp;
+-----+-----+-----+-----+
| Table | Non_unique | Key_name          | Seq_in_index | Column_name |
+-----+-----+-----+-----+
| emp  |      0 | PRIMARY           |            1 | EMPNO       |
| emp  |      1 | EMP_MGR_FK        |            1 | MGR          |
| emp  |      1 | ind_emp_ename     |            1 | ENAME        |
| emp  |      1 | ind_emp_dept_job  |            1 | DEPTNO      |
| emp  |      1 | ind_emp_dept_job  |            2 | JOB          |
+-----+-----+-----+-----+
5 rows in set (0.00 sec)
```

刪除索引(Droping an Index)

```
ALTER TABLE table DROP INDEX IndexName;  
or  
DROP INDEX IndexName ON table;
```

```
mysql> ALTER TABLE emp DROP INDEX ind_emp_ename;  
Query OK, 0 rows affected (0.11 sec)  
Records: 0  Duplicates: 0  Warnings: 0
```

```
mysql> DROP INDEX emp_deptno_fk ON emp;  
ERROR 1553 (HY000) : Cannot drop index  
'ind_emp_dept_job': needed in a foreign key constraint
```

附件

1. 建立一個查詢來顯示部門(dept)資料表中的所有資料。
2. 建立一個查詢來顯示每一位員工的姓名(name)、職稱(job)、進公司日期(hire date)及員工編號(employee number)，並將且員工編號顯示在最前面。
3. 建立一個查詢來顯示所有員工所擔任的職稱有哪些？（重複的資料只顯示一次）
4. 建立一個查詢來顯示每一位員工的姓名(name)、職稱(job)、進公司日期(hire date)及員工編號(employee number)，並將且員工編號顯示在最前面。並將資料表頭重新命名為：Emp#, Employee, Job, Hire Date。
5. 建立一個查詢將姓名(name)及職稱(job)串接為依個資料項(資料中間利用一個空白和一個逗號做區隔)，並將表頭重新命名為 Employee and Title。

建立查詢指令以顯示下列各題描述之資料：

1. 顯示出所有員工薪資超過2850元的員工之姓名和薪資。
2. 顯示員工編號為7566員工的姓名和他所屬的部門編號。
3. 顯示薪資不介於1500~2850元之間的所有員工之姓名和薪資。
4. 顯示於1981年2月20日和1981年5月1日間進入公司的員工之姓名，職稱和進公司日期，並依進公司日期由小到大排序。
5. 顯示部門10和30所有員工之姓名和他所屬的部門編號，並依名字依英文字母順序排序。
6. 顯示薪資超過1500 “且” 在10 “或” 30部門工作員工之姓名和薪資，把分別把表頭命名為Employee和 Monthly Salary。
7. 顯示於1982年進公司的所有員工之姓名，職稱和進公司日期。
8. 顯示沒有主管的員工之姓名和職稱。
9. 顯示所有有賺取佣金的員工之姓名，薪資和佣金，並以薪資加佣金作降冪排列。
10. 顯示所有名字裡第三個英文字母為A的員工之姓名與職稱。
11. 顯示名字裡有兩個 “L” 且在30部門工作或經理是7782的員工之姓名，經理員工編號及所屬的部門編號。
12. 顯示職稱為Clerk或Analyst且薪水不等於1000, 3000, 5000的員工之姓名，職稱和薪資。
13. 顯示佣金比薪水的1.1倍還多的員工之姓名，薪資和佣金。

建立查詢指令以顯示下列各題描述之資料：

1. 顯示系統目前的日期並將表頭命名為” 系統日期 ” 。
2. 顯示所有員工之員工編號, 姓名, 薪資及將薪資增加15%並且以整數表示，並將表頭命名為” New Salary ” 。
3. 接續第二題，增加一個資料項表頭命名為Increase (將New Salary 減掉 salary 的值)。
4. 顯示員工的姓名, 進公司日期, 檢討薪資的日期(指在進公司工作六個月後的第一個星期一)，將該欄命名為 REVIEW，並自訂日期格式為：Sunday, the Seventh of September。(星期幾， 幾月幾日)。
5. 顯示每位員工的姓名，資料項(MONTHS_WORKED):計算到今天為止工作了幾個月(將月數四捨五入到整數)
6. 顯示如下格式：<員工姓名> earns <薪水> monthly but wants <3倍的薪水>. 並將表頭顯示為Dream Salaries。
7. 顯示所有員工之姓名和薪資，設定薪資長度為15個字元並且在左邊加上\$符號，將表頭命名為SALARY。
8. 顯示員工之姓名, 進公司日期，資料項(DAY):顯示員工被雇用的那天為星期幾，並以星期一作為一週的起始日，依星期排序。
9. 顯示員工的姓名和名為COMM的欄位:顯示佣金額，如果該員工沒有賺取佣金則顯示"No Commission."
10. 顯示資料項命名為 EMPLOYEE_AND THEIR SALARIES 的資料來顯示所有員工之名字和薪資，且用星號來表示他們的薪資，每一個星號表示100元，並以薪資由高到低來顯示。

建立查詢指令以顯示下列各題描述之資料：

1. 顯示所有員工的最高、最低、總和及平均薪資，依序將表頭命名為 Maximum, Minimum, Sum 和 Average，請將結果顯示為四捨五入的整數。
2. 顯示每種職稱的最低、最高、總和及平均薪水。
3. 顯示每種職稱的人數。
4. 顯示資料項命名為Number of Managers來表示擔任主管的人數。
5. 顯示資料項命名為DIFFERENCE的資料來表示公司中最高和最低薪水間的差額。
6. 顯示每位主管的員工編號及該主管下屬員工最低的薪資，排除沒有主管和下屬員工最低薪資少於1000的主管，並以下屬員工最低薪資作降冪排列。
7. 顯示在1980, 1981, 1982, 1983年進公司的員工數量，並給該資料項一個合適的名稱。

1. 顯示所有員工之姓名, 所屬部門編號, 部門名稱及部門所在地點。
2. 顯示所有有賺取佣金的員工之姓名, 佣金金額, 部門名稱及部門所在地點。
3. 顯示姓名中包含有” A” 的員工之姓名及部門名稱。
4. 顯示所有在” DALLAS” 工作的員工之姓名, 職稱, 部門編號及部門名稱
5. 顯示出表頭名為: Employee, Emp#, Manager, Mgr#, 分別表示所有員工之姓名, 員工編號, 主管名字, 主管的員工編號。
6. 顯示出SALGRADE資料表的結構, 並建立一查詢顯示所有員工之姓名, 職稱, 部門名稱, 薪資及薪資等級。
7. 顯示出表頭名為: Employee, Emp Hiredate, Manager, Mgr Hiredate的資料項, 來顯示所有比他的主管還要早進公司的員工之姓名, 進公司日期和主管之姓名及進公司日期。
8. 顯示出表頭名為: dname, loc, Number of People, Salary的資料來顯示所有部門之部門名稱, 部門所在地點, 部門員工數量及部門員工的平均薪資, 平均薪資四捨五入取到小數第二位。

1. 顯示和Blake同部門的所有員工之姓名和進公司日期。
2. 顯示所有在Blake之後進公司的員工之姓名及進公司日期。
3. 顯示薪資比公司平均薪資高的所有員工之員工編號, 姓名和薪資，並依薪資由高到低排列。
4. 顯示和姓名中包含 T 的人再相同部門工作的所有員工之員工編號和姓名。
5. 顯示在Dallas工作的所有員工之姓名, 部門編號和職稱。
6. 顯示直屬於” King” 的員工之姓名和薪資。
7. 顯示銷售部門” Sales” 所有員工之部門編號, 姓名和職稱。
8. 顯示薪資比公司平均薪資還要高且和名字中有 T 的人在相同部門上班的所有員工之員工編號, 姓名和薪資。
9. 顯示和有賺取佣金的員工之部門編號和薪資都相同的員工之姓名, 部門編號和薪資。
10. 顯示和在Dallas工作的員工之薪資和佣金都相同的員工之姓名, 部門編號和薪資。
11. 顯示薪資和佣金都和Scott相同的所有員工之姓名, 進公司日期和薪資。(不要在結果中顯示Scott的資料)
12. 顯示薪資比所有職稱是” Clerks” 還高的員工之姓名, 進公司日期和薪資，並將結果依薪資由高至低顯示。



THE END