

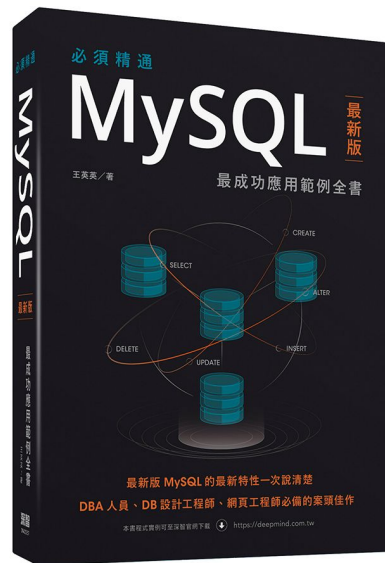
MySQL 資料庫系統建置與管理

講師: 戴天恩

jerryoop66@gmail.com

本課程提供資料

- MySQL 電子檔講義(本講義)
- 參考書: 必須精通 MySQL 最新版:最成功應用範例全書
- 參考書提供課後自修使用
- MySQL 安裝文件
- MySQL 移除文件



資料庫(Database)的概念

- Database: 儲存資料的地方, 具有一定的分類、排序方法。
例如: 電話簿、網站資料。
- 通常會有方便的功能做資料的查詢、修改。
- 例如: 查詢這個月生日的會員、根據會員資料查找有問題的訂單...等
- 資料庫中可以有許多表格(Table), 代表不同分類的資料。



資料庫管理系統(DBMS)

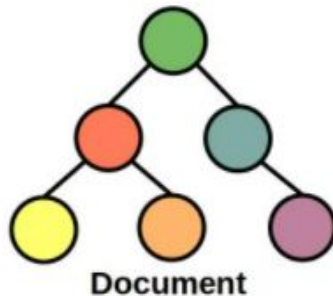
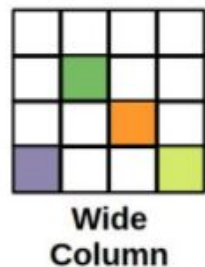
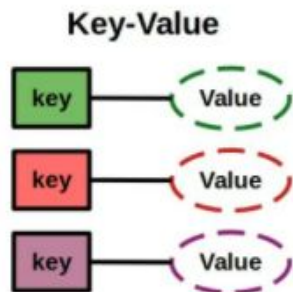
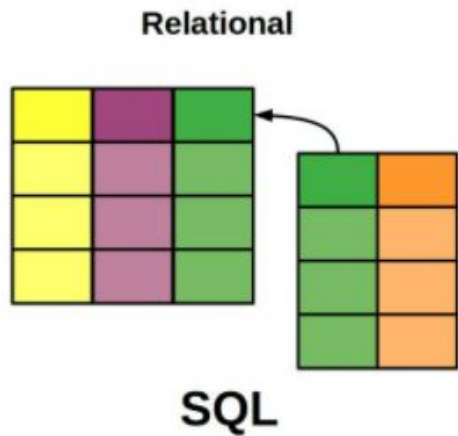
- 資料庫管理系統(DBMS): Database Management System
- 實作各種豐富的功能, 實作方法會因廠商不同而有所差異。
- 常見的 DBMS 有: MySQL, Oracle, SQL Server, PostgreSQL
- 本課程使用 MySQL 8.0
- MySQL 屬於關聯式資料庫, 更精確地說是 RDBMS: (Relational Database Management System)



關聯式資料庫 vs. 非關聯式資料庫

關聯式資料庫 (SQL)
Relational databases

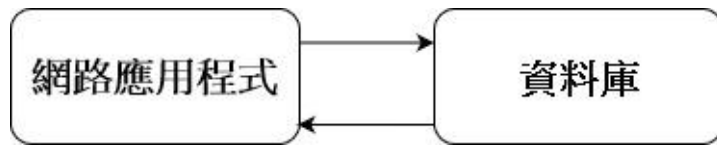
非關聯式 (NoSQL databases)



SQL 常見的資料庫廠商：
MySQL, MariaDB
Oracle
PostgreSQL
Microsoft SQL Server

NoSQL 常見的資料庫廠商：
MongoDB
Amazon DynamoDB
Redis
ElasticSearch

資料庫在整個網路應用程式的位置



全端課程地圖：<https://www.iiiedu.org.tw/fuen/>

目前開發技術詳細資訊：<https://github.com/kamranahmedse/developer-roadmap>

SQL 簡介

Structured Query Language(SQL):結構化查詢語言, 由IBM公司於1970年代初所開發, 用來存取資料庫內的資料。它包括了

- 用來產生、修改與刪除表格、索引、視野等資料定義語言(DDL)
- 用來新增、修改與刪除表格資料的資料處理語言(DML)
- 使用 SELECT 敘述來查詢資料庫所含資訊的資料查詢語言(DQL)
- 用來控制交易(Transaction)之進行的交易控制敘述
- 用來授與及取回使用者權限、設定密碼的資料控制言(DCL)

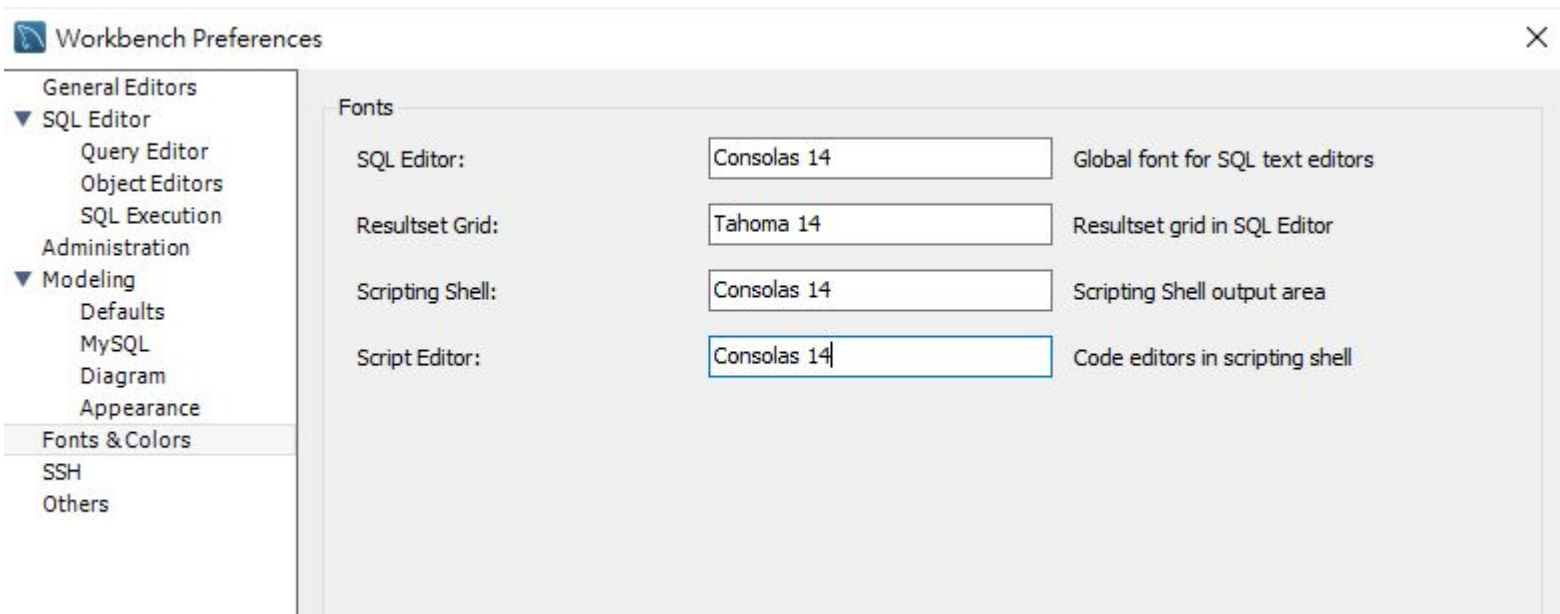
SQL指令可被用來產生、刪除資料庫中的表格, 以及讀取、新增、修改、刪除表格內的資料

MySQL 的安裝與啟動

請見 MySQL 安裝文件

圖形化工具 Workbench

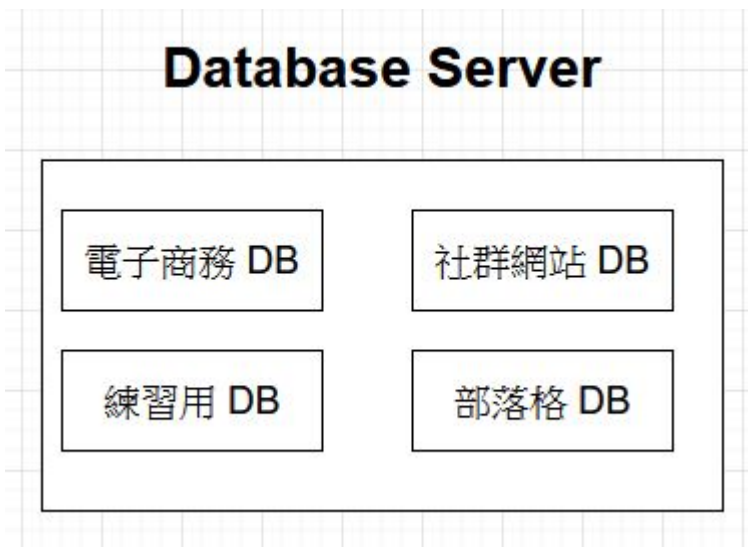
設定字型大小: Edit > Preference...



資料庫 Database

MySQL 與 Database

- 前面提過, MySQL 為一種資料庫管理系統(DBMS), 當 MySQL 啟動時會啟動一個 Database Server, 這個 Server 裡面可以有多个 Database 。
- 不同的應用程式通常會使用不同的 Database (DB)。
- 有時用會 Schema 來描述 Database 。



Database 基本指令

- 顯示所有資料庫：`SHOW DATABASES;`
- 建立 employee 資料庫：`CREATE DATABASE MyShop;`
- 使用 employee 資料庫：`USE MyShop;`
- 移除 employee 資料庫：`DROP DATABASE MyShop;`

註：最後一定要有分號，否則不會執行

註：語法英文大小寫皆可

CREATE DATABASE 注意事項

- CREATE DATABASE <name>; 英文大小寫皆可。
- create database drinkShop; > 可以
- 可以使用底線命名: create database drink_shop; > 可以
- <name> 不可以有空格: create database my member; > 不行

查看現在使用的資料庫

當 MySQL 管理系統內有許多資料庫，有時需要查看目前所使用的資料庫，則可使用以下語法

```
SELECT database();
```

註：

- 這邊的 database() 一定要小寫，因為是 MySQL 的內建函數
- 若現在無使用的 Database 則會回傳 NULL

資料表(表格) Tables

資料表(Tables) 簡介

- 資料表又稱表格，原文為 Tables。
- 資料表為 SQL 中最核心的單位。
- 在關聯式資料庫內，資料庫(Database)裡面通常會有一堆資料表(Tables)。
- 資料表內的資料，會有一致的格式。
- 若直接刪除資料庫，裡面的資料表會一起刪除。

資料表(Tables) 簡介

假設這是一個顧客定位的資料表。

memberId	memberName	memberLocation	memberOrder
1001	小明	東京	3
1002	小花	台中市	5
1003	Amy	USA	8

註：

- 資料庫名稱、表格名稱、欄位名稱請使用英文命名，避免發生不可預期的問題。

資料表(Tables) 簡介- 欄位 (Columns)

memberId	memberName	memberLocation	memberOrder
----------	------------	----------------	-------------

說明:

- 欄位原文為 Columns 或俗稱 headers (表頭)
- 欄位請使用英文命名。

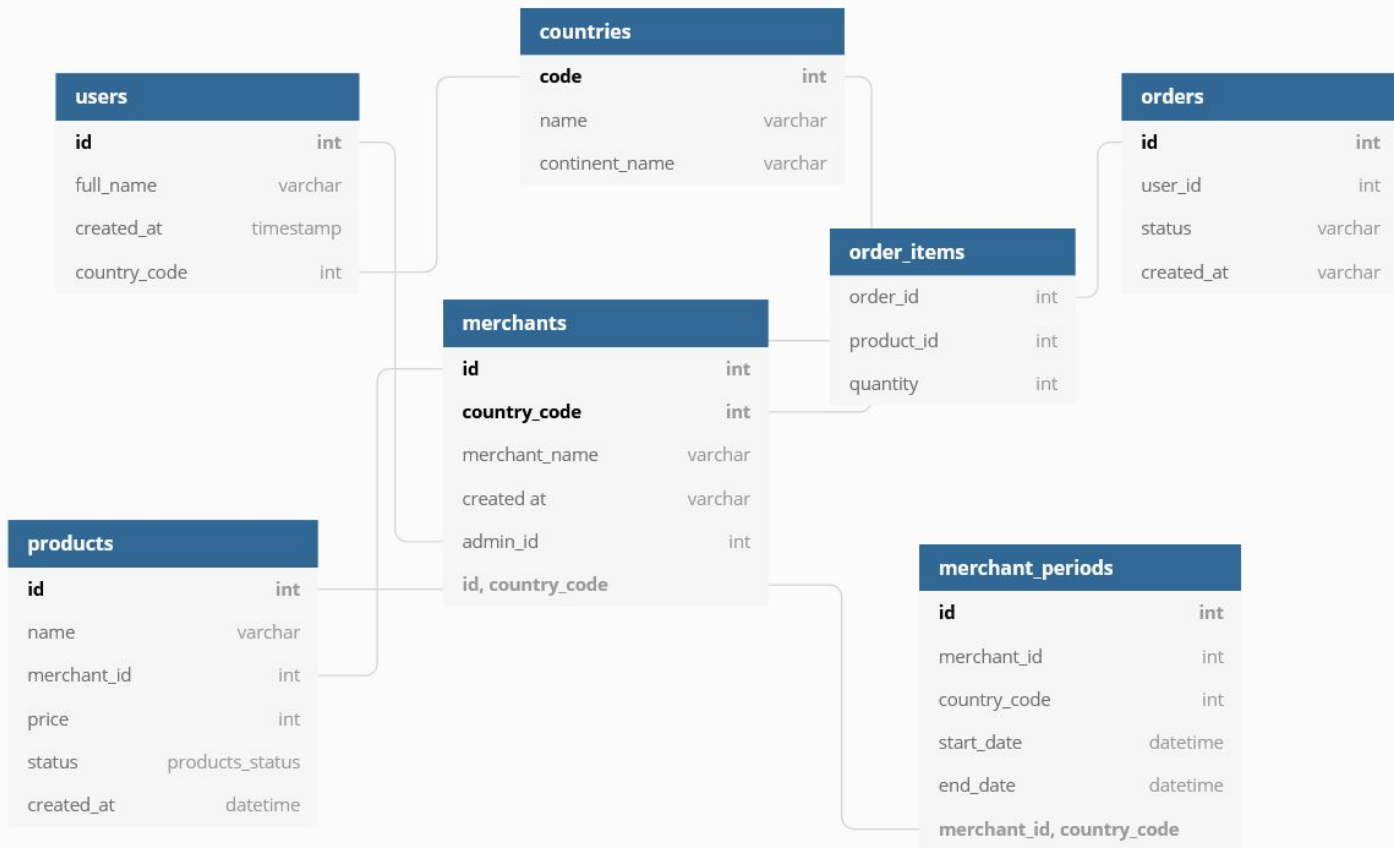
資料表(Tables) 簡介 - 真實資料(rows)

1 row
→

memberId	memberName	memberLocation	memberOrder
1001	小明	東京	3
1002	小花	台中市	5
1003	Amy	USA	8

說明: 1 row 為一筆資料, 也可以稱一列資料。

範例:



通常一個應用程式的資料表會比這個複雜

資料表(Tables)

資料型別(Data Types)

數字	文字	文字	數字
memberId	memberName	memberLocation	memberOrder
1001	小明	東京	3
1002	小花	台中市	5
1003	Amy	USA	eight (不可)

說明:

- Table 內的資料, 每個欄位的資料必須一致。
- 之後需要運算時, 因格式相同而方便運算。
- 讓資料進去資料庫以前就先設定好格式, 使資料整齊且方便運算處理。

資料表(Tables)

MySQL資料型別(Data Types)

- 有關數字的型別(Numeric Data Types):
INT, SMALLINT, BIGINT, TINYINT, DECIMAL, NUMERIC
- 有關文字的型別(String Data Types):
CHAR, VARCHAR, BINARY, VARBINARY, BLOB, LONGBLOG ...
- 有關日期的型別():
DATE, TIME, DATETIME, TIMESTAMP, YEAR ...

說明: 不用刻意去記每一個型別, 先熟練常用的, 之後遇到再透過官方網站查詢文件。

詳細資料型別: <https://dev.mysql.com/doc/refman/8.0/en/data-types.html>

資料表(Tables)

有關文字、字串相關的資料型別(String Data Types)

- VARCHAR(N): 變動長度的非二進位字串, 會用單引號包起來, e.g: '小明'
N: 可以指定 0~65535
e.g. VARCHAR(50) 表示該欄位可接受 50 個字以內的字串
- CHAR(N): 固定長度的非二進位字串, 長度不足會往右補空格
N: 可以指定 0~255
- BLOB(Binary Large OBject): 可以存放二進位資料(小檔案)。

資料表(Tables)

CHAR VS. VARCHAR

Value	CHAR (4)	Storage Required	VARCHAR (4)	Storage Required
' '	' '	4 bytes	' '	1 byte
'ab '	'ab '	4 bytes	'ab '	3 bytes
'abcd'	'abcd'	4 bytes	'abcd'	5 bytes
'abcdefgh'	'abcd'	4 bytes	'abcd'	5 bytes

由上圖可知，CHAR 型別的資料會與先給 **固定的容量**，若是沒裝滿則會補空格。
而 VARCHAR 會根據裡面裝的內容來調整所需的容量。
因此，實務上來說 VARCHAR 會比較常用。
相同的地方是，這兩者都不能塞入超過預先設定的長度的資料。
e.g. varchar(50) 最多只能給 50 個字元，超過則無法塞入資料。

資料表 (Tables)

有關數字資料型別 (Numeric Data Types)

- INT : 整數型態 (−2147483648 ~ +2147483647)
- DECIMAL(M, D) : 浮點數(含有小數點的數), 最多 38 位數

M : 整數與小數位數的總位數

D : 小數位數(介於 0~M)

e.g. : DECIMAL(5, 2): -999.99 ~ 999.99

- TINYINT : 空間較小的數字, 範圍: -127 ~ 128

TINYINT 常用來表示布林值(Boolean) : 1 為真, 0 為假

範例 : Select TRUE, FALSE, true, false, True, False; (不分大小寫)

預期輸出 :

	TRUE	FALSE	true	false	True	False
▶	1	0	1	0	1	0

資料表(Tables)

日期時間相關的資料型別(Date and Time Types)

- TIME 時間：格式為時分秒: 'HH:MM:SS'

範例: '10:11:12', '22:50:32'

- DATE 日期：格式為年月日: 'YYYY-MM-DD'

範例: '2021-05-15', '2020-03-20', '2008-09-15'

資料表(Tables)

日期時間相關的資料型別(Date and Time Types)

- DATETIME 日期加時間：格式'YYYY-MM-DD HH:MM:SS'，需 8 位元組
範圍: '1001-01-01 00:00:00' ~ '9999-12-31 23:59:59'
範例: '2008-09-15 10:11:12', '2021-05-15 22:50:32'
- TIMESTAMP 時間戳記：格式與DATETIME 相同但範圍較小。若新增時沒給資料，則會自動加入現在時間。需 4 位元組
範圍: '1970-01-01 00:00:01' ~ '2038-01-19 23:59:59'
範例: '2021-05-15 09:00:00', '2020-03-20 22:50:32'
- 特殊：TIMESTAMP 儲存日期時會對目前時區轉換，取出時再轉換成目前時區，所以查詢的時候會因時區的不同，而顯示不同時間。

詳細資料型別：<https://dev.mysql.com/doc/refman/8.0/en/data-types.html>

資料表(Tables)

資料型別(Data Types)

- 學會基本型別後, 回顧我們之前的表格, 思考該用哪些型別。

INT	VARCHAR(50)	VARCHAR(50)	INT
memberId	memberName	memberLocation	memberOrder
1001	'小明'	'東京'	3
1002	'小花'	'台中市'	5
1003	'Amy'	'USA'	8

資料表(Tables)

創建資料表(Create Table)

- 在之前課程初步了解資料表(表格)後，以下說明創建表格的方式。
- 創建資料表語法：

```
CREATE TABLE tableName(  
    column_name1 data_type,  
    column_name2 data_type,  
    ... ,  
);
```

範例: CREATE TABLE members(member_name varchar(50), member_age INT);

下一步說明如何得知是否成功創建表格。

資料表(Tables)

查詢資料表欄位資訊

- SHOW TABLES; -- 查詢目前 DATABASE 內的所有資料表
- SHOW COLUMNS FROM <table_name>; -- 查看該資料表內欄位

簡短版1 : DESC <table_name>;

簡短版2 : DESCRIBE <table_name>;

Field	Type	Null	Key	Default	Extra
memberName	varchar(50)	YES		NULL	
age	int	YES		NULL	

資料表(Tables) 實作

請實作出以下表格 (table_name = members), 先做欄位就好, 不用加資料進去

memberId	memberName	memberLocation	memberOrder
1001	'小明'	'東京'	3
1002	'小花'	'台中市'	5
1003	'Amy'	'USA'	8

做完以後使用上個段落學到的, 查看欄位詳細資訊,
然後刪除表格。

新增資料(Insert Data)

新增資料(Insert Data)

本節介紹：

- 前面章節已學習如何創建資料庫、資料表
- 本章節主要說明如何將資料放入資料表，以及需要注意的細節。
- 注意命名資料時避免使用停用字。
- 停用字有哪些？

<https://dev.mysql.com/doc/refman/8.0/en/fulltext-stopwords.html>

新增資料(Insert Data)

Insert 基本語法：

```
INSERT INTO <table_name>(column_name1, column_name2, ...)  
VALUES(value1, value1, ...);
```

範例: 加入之前課程 1001 這筆資料的語法

```
INSERT INTO members(memberId, memberName, memberLocation, memberOrder)  
VALUES(1001,'小明', '東京', 3);
```

memberId	memberName	memberLocation	memberOrder
1001	'小明'	'東京'	3
1002	'小花'	'台中市'	5
1003	'Amy'	'USA'	8

新增資料(Insert Data)

```
INSERT INTO members(memberId, memberName, memberLocation, memberOrder)
VALUES(1003,'Amy', 'USA', 8);
```

memberId	memberName	memberLocation	memberOrder
1003	'Amy'	'USA'	8

注意事項:

- 輸入的欄位順序、值的順序是很重要的，不可以隨意擺放。

新增資料(Insert Data)

簡易地查看資料

這邊先簡單介紹查詢表格內所有資料的語法，後面章節會詳細介紹查詢(Select)

```
SELECT * FROM table_name
```

```
SELECT * FROM members
```

預期輸出

memberId	memberName	memberLocation	memberOrder
1001	小明	東京	3

新增資料(Insert Data)

一次新增多筆資料

第一筆 VALUES 加**逗號**後，可以直接往下寫，就可以新增多筆資料：

```
INSERT INTO table_name  
      (column_name, column_name)  
VALUES  (value, value),  
        (value, value),  
        (value, value);
```

新增資料(Insert Data)

一次新增多筆資料

第一筆 VALUES 加**逗號**後, 可以直接往下寫, 就可以新增多筆資料:

```
INSERT INTO members(memberId, memberName, memberLocation,  
memberOrder)  
VALUES  
(1002,'小花', '台中市', 5),  
(1003,'Amy', 'USA', 8);
```

新增資料(Insert Data)

練習實作資料表

請根據學習到的資料庫技術，做出以下 產品表格，表格明稱為 phone_products

product_name	brand	price
iPhone 12	Apple	19900
Pixel 5	Google	18900
Galaxy s21	Samsung	19800

新增後使用 `Select * from phone_products` 檢查資料

NULL 和 DEFAULT

NULL 和 DEFAULT

NULL 說明

前面課程用 DESC phone_products 來查看 Table 內欄位資訊時，有個欄位 Null，在此說明：

Field	Type	Null	Key	Default	Extra
product_name	varchar(50)	YES		NULL	
brand	varchar(20)	YES		NULL	
price	int	YES		NULL	

NULL 定義：

不知道值是甚麼。若該欄位的 NULL 設定為 YES，表示可以塞入未知的值。

注意：NULL 不是 0 也不是空字串('')。

新增後：

iPhone 6s	NULL	NULL
NULL	NULL	NULL

NULL 和 DEFAULT NOT NULL 說明與設定

若要避免產生 NULL 資料, 創建表格時需要設定該欄位不可為 NULL。語法為:

NOT NULL

範例:

```
create table phone_products2(  
  product_name varchar(50) not null,  
  brand varchar(20) not null,  
  price int not null  
);
```

Field	Type	Null	Key	Default	Extra
product_name	varchar(50)	NO		NULL	
brand	varchar(20)	NO		NULL	
price	int	NO		NULL	

NULL 和 DEFAULT NOT NULL 說明

加入 NOT NULL 後, 若新增不完整的資料:

```
insert into  
phone_products2(product_name)  
values('iPhone X');
```

資料**不會**新增成功, 而且會顯示以下警告:

	Level	Code	Message
▶	Error	1364	Field 'brand' doesn't have a default value
	Error	1364	Field 'price' doesn't have a default value

NULL 和 DEFAULT

Default 說明與設定

當新增資料時，該欄位沒有塞入 值，就可以使用 Default 設定其預設值。

範例：

```
create table phone_products3(  
  product_name varchar(50) not null default 'unnamed',  
  brand varchar(20) not null default 'not sure',  
  price int not null default 0  
);
```

Field	Type	Null	Key	Default	Extra
product_name	varchar(50)	NO		unnamed	
brand	varchar(20)	NO		not sure	
price	int	NO		0	

NULL 和 DEFAULT

不含 NOT NULL 的 DEFAULT

```
create table phone_products4(  
  product_name varchar(50) default 'unnamed',  
  brand varchar(20) default 'not sure',  
  price int default 0  
);
```

```
insert into phone_products4(product_name, brand)  
values (null, 'Apple'), (null, 'Google');
```

注意：

通常設計表格時，該欄位需要有 DEFAULT 通常都會設計成 NOT NULL，這邊只是說明 DEFAULT 可以用在允許 NULL 的情況，但 **不建議這樣設計**。

初探主鍵 (PRIMARY KEY)

初探主鍵(Primary Key)

在商業邏輯中，極可能發生一件事：不同一間商品，但資料看起來一樣的情況
例如：商品、使用者若有重複的名稱，就無法分辨，缺少識別性。

product_name	brand	price
iPhone 12	Apple	19900
iPhone 12	Apple	19900
iPhone 12	Apple	19900

- 這樣子的資料缺乏識別性 (identity)，也就是無法分清楚誰是誰，這時 id 的重要性就顯現出來 (類似身分證號碼)。
- 在資料表設計中，實務上會將每一個 Table 加上主鍵(Primary Key)，也稱 ID，增加其每一筆資料的識別性。
- 若識別資料不重要，則不需要主鍵，例如：初步取得的大量資訊，龐雜廣告 內容等。

初探主鍵(Primary Key)

所以, 每一筆資料應該都要有 ID, 也就是主鍵:

product_id	product_name	brand	price
1	iPhone 12	Apple	19900
2	iPhone 12	Apple	19900
3	iPhone 12	Apple	19900

- 主鍵(Primary key; PK)特性: 主鍵必須是獨特的(不可重複)、可識別的。
- 因此若是新增已經有的 Primary Key, 則會產生錯誤, 且新增失敗。
- 這種規則又稱「主鍵約束」(Primary Key Constraint)
- 主鍵可以由多個欄位組成 (2個以上)

初探主鍵(Primary Key)

主鍵語法:

```
create table my_products(  
  product_id int not null,  
  product_name varchar(50),  
  price int,  
  primary key (product_id)  
);
```

或

```
create table my_products2(  
  product_id int not null primary key,  
  product_name varchar(50),  
  price int  
);
```

- 主鍵(Primary key; PK)特性: 主鍵必須是獨特的(不可重複)、可識別的。
- 因此若是新增已經有的 Primary Key , 則會產生錯誤, 且新增失敗。

初探主鍵(Primary Key)

自動產生並增加主鍵值(AUTO_INCREMENT)

- 當新增資料時，要知道上一個資料的主鍵值，才能知道下一個資料要給甚麼主鍵值是不方便的，因此，通常資料庫都有自動增加主鍵值的機制，語法為
AUTO_INCREMENT
- 在 MySQL 中 AUTO_INCREMENT 初始值是 1，每次增加 1。
- 一個 Table 內只能有一個欄位使用 AUTO_INCREMENT，這欄位必須是主鍵。
- AUTO_INCREMENT 約束的欄位可以是整數類型(INT, TINYINT, SMALLINT, BIGINT 等)，通常會使用 INT。

初探主鍵(Primary Key)

自動產生並增加主鍵值(AUTO_INCREMENT)

- AUTO_INCREMENT 範例

```
create table my_products3(  
  product_id int not null auto_increment,  
  ...  
  primary key (product_id)  
);
```

Field	Type	Null	Key	Default	Extra
product_id	int	NO	PRI	<small>NULL</small>	auto_increment
product_name	varchar(50)	YES		<small>NULL</small>	
price	int	YES		<small>NULL</small>	

初探主鍵(Primary Key)

自動產生並增加主鍵值(AUTO_INCREMENT)

當有設定 AUTO_INCREMENT 時, 若新增時還給 id, 則後面的資料會根據前一筆 id + 1, 若想保持 id 整齊性, 在 AUTO_INCREMENT 情況下, 就不建議自己塞入 id 值。

```
insert into my_products3(product_id,product_name, price)
values(1001,'i13', 23000);
```

這時它的下一筆就會從 1002 開始

```
insert into my_products3(product_name, price)
values('i14', 23000);
```

初探主鍵(Primary Key)

增加主鍵值之客製化

若自動增加的主鍵值不想從 1 開始, 則可以使用
AUTO_INCREMENT = 100 設定其初始值, 讓他從 100 開始

```
create table my_products4(  
  product_id int not null auto_increment,  
  product_name varchar(50),  
  price int,  
  primary key (product_id)  
);  
  
alter table my_products4 auto_increment = 100;
```

註：若要自己客製化初始值, 建議新增資料以前就先設定, 保持 id 一致性。

表格實作

請做出一個表格：

- 名稱為 my_food , 必須有主鍵 food_id, 且需要自動產生 id
- 包含一個 food_name 欄位, 填入自己喜歡的食物
- 包含一個 food_location 欄位, 填入該食物的地點(ex. 台北市信義路)
- 包含一個 food_price 欄位, 需是 INT, 填入該食物的價錢
- 至少 3 筆資料

資料操作(CRUD)

資料操作

CRUD簡介

基本的資料操作有四個: C, R, U, D

- **C**reate : 新增資料 (INSERT)
- **R**ead : 查詢資料 (SELECT)
- **U**ppdate : 更新/修改資料
- **D**elete : 刪除資料

說明：

- 這四種是資料操作的基本大面向，每一個面向都可能極為簡單或複雜，視商業邏輯而定。
- 前面課程內容已經說明如何新增資料，本章說明如何做資料的查詢、修改、刪除。

資料操作(CRUD)

預備資料

- 這是一個員工資料表 employee, 共有 7 位。

```
create table employee(  
  emp_id int not null auto_increment,  
  emp_name varchar(50),  
  emp_department varchar(50),  
  emp_age int,  
  primary key (emp_id)  
);
```

```
insert into employee(emp_name,  
  emp_department, emp_age)  
values('Tony','HR', 35),  
('Amy','RD', 30),('Jenny', 'Sells',28),  
('Tom','RD',22),('Betty','HR',25),  
('John','Account',36),('Mary','Sells', 22);
```

資料操作(CRUD)

Read: 查詢語句(SELECT)

```
SELECT * FROM <table_name>  
* 表示全部欄位的全部資料
```

```
SELECT emp_age FROM employee;  
只選擇查看 emp_age 欄位的全部資料
```

```
SELECT emp_name FROM employee;  
只選擇查看 emp_name 欄位的全部資料
```

```
SELECT emp_name, emp_age FROM employee;  
只看兩個欄位
```

```
SELECT emp_id, emp_age, emp_name FROM employee;  
SELECT emp_age, emp_name, emp_id FROM employee;  
select * from employee;  
欄位顯示順序可以調整
```

資料操作(CRUD)

有條件地查詢(WHERE)

```
SELECT * FROM employee WHERE emp_age=22;
```

根據年齡查找

```
select * from employee WHERE emp_name = 'Tom';  
select * from employee WHERE emp_name = 'TOM';
```

根據姓名查找

註 1：注意資料型別，決定是否要有單引號' ')把資料包起來。

註 2：字串英文大小寫都可以找到(case insensitive 無關大小寫)，方便查詢，例如 Email 資料。

註 3：WHERE 子句不只可以在 SELECT 使用，修改、刪除語法也可以搭配 WHERE，後面說明。

資料操作(CRUD)

用 BINARY 搜尋字串則分辨大小寫

```
select * from employee WHERE emp_name = 'Tom';
```

大小寫都接受

```
select * from employee WHERE binary emp_name = 'TOM';
```

會分辨大小寫

資料操作(CRUD)

練習 SELECT, WHERE

1. 寫出右方表格的查詢語法(Query)。
2. 查詢所有員工的 id (只要 emp_id 欄位)。
3. 查詢出 HR 部門的所有員工, 只需要姓名和年齡欄位。

emp_name	emp_department
Tony	HR
Amy	RD
Jenny	Sells
Tom	RD
Betty	HR
John	Account
Mary	Sells

資料操作(CRUD)

別名(Aliases)

有時欄位名稱的長度很長 (不好看), 在寫 SQL 的時候、顯示的時候不方便, 這時就可以用別名來替代。後面章節多表格操作時會更有感覺。

關鍵字: **AS**

```
SELECT emp_name AS Names, emp_department AS Department FROM employee;  
SELECT emp_name AS 'Employee names', emp_department as Department FROM employee;
```

Names	Department
Tony	HR
Amy	RD
Jenny	Sells
Tom	RD
Betty	HR
John	Account
Mary	Sells

employer names	Department
Tony	HR
Amy	RD
Jenny	Sells
Tom	RD
Betty	HR
John	Account
Mary	Sells

資料操作(CRUD)

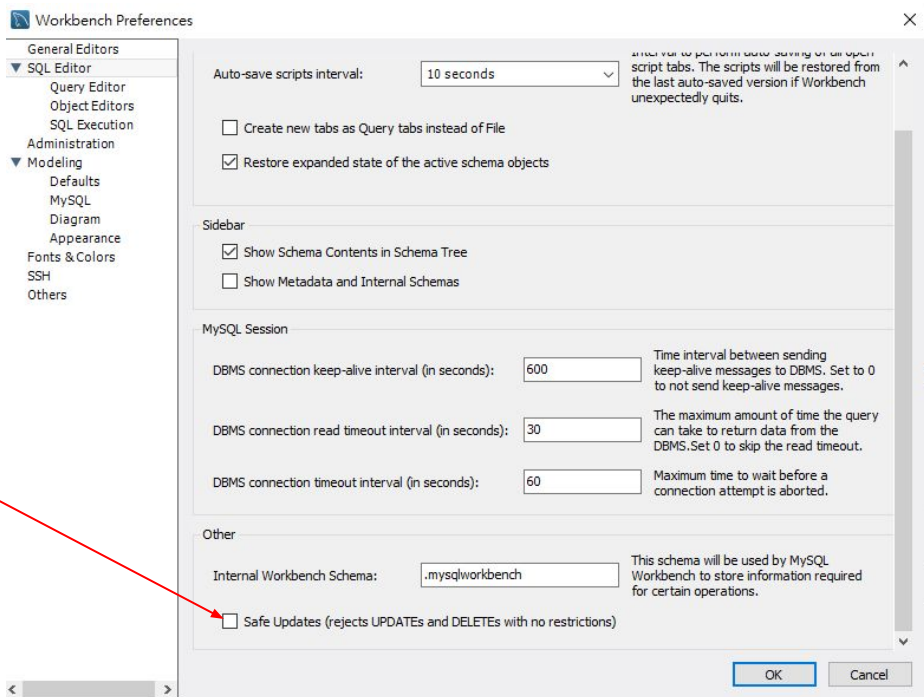
Update 前置設定

因 Workbench 預設只能更新 where 指定有 Key 的欄位, 因此我們先關閉此設定。若您是使用命令提示字元視窗 (小黑窗) 執行 SQL, 則不必做此設定。

左上角 Edit > Preference > SQL Editor

取消勾選 Safe Updates

左上角 Query > Reconnect to Server



資料操作(CRUD)

Update 更新資料

更新資料的關鍵字為：**UPDATE, SET, WHERE**

```
UPDATE employee SET emp_department='Accounting'  
WHERE emp_department='Account';  
會計部門換名子
```

```
update employee set emp_department='Accounting'  
where emp_name='Mary';  
Mary 換到會計部門
```

註：更新是無法復原(undo)的，因此比較好的做法是先找到資料，確認後再進行更新，這個方式我們在稍後的刪除資料一起說明。因刪除也有此特性。

資料操作(CRUD)

Update 更新資料練習實作

1. Amy 生日, 請將她的年齡改為 31。
2. John 想要改名成 Johnny。

資料操作(CRUD)

Delete 刪除資料

刪除資料的關鍵字為 DELETE，通常會搭配 WHERE 使用。
刪除資料前也建議先查詢出來，才可確定是要刪除的資訊。

假設 Jonny 將從目前的分公司升遷到總公司，所以這邊要移除 Jonny。

```
SELECT * FROM employee WHERE emp_name='johnny';
```

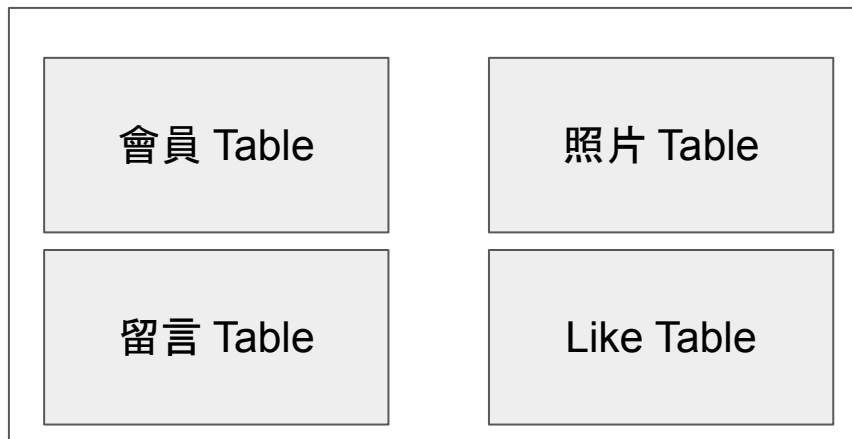
```
DELETE FROM employee where emp_name='johnny';
```

Table 之間的關係

Table 之間的關係

本節說明

社群網站 Database



- 一般來說，實務上的表格不會像前面課程那樣單純只用一個表格。
- 例如一個店商系統，會有的表格有：會員、商品、購物車、訂單。
- 例如一個社群網站(e.g. Instagram)，會有會員、照片、留言、喜歡等表格。

Table 之間的關係

設計 Table 的方法

- Google 大法(SQL schema xxx system)
- 若都沒人做過, 可以用**功能**為單位去思考。
- 不同邏輯不放在同個表格為原則(正規化)
- 用誰擁有甚麼去設計
- 初期變動表格之間的關係是合理的, 但建議還是完全設計好再開始寫SQL 敘述。

社群網站 Database

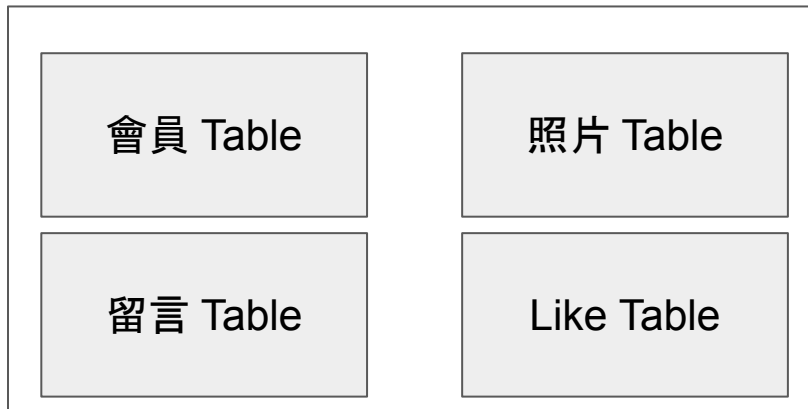


Table 之間的關係

設計 Table 的方法(觀察 IG 使用者頁面資料)

使用者資訊



追蹤人數, 被追蹤人數

使用者有許多照片

註：我們先假設一個貼文只有一張照片

Table 之間的關係

設計 Table 的方法(觀察 IG 照片資料)



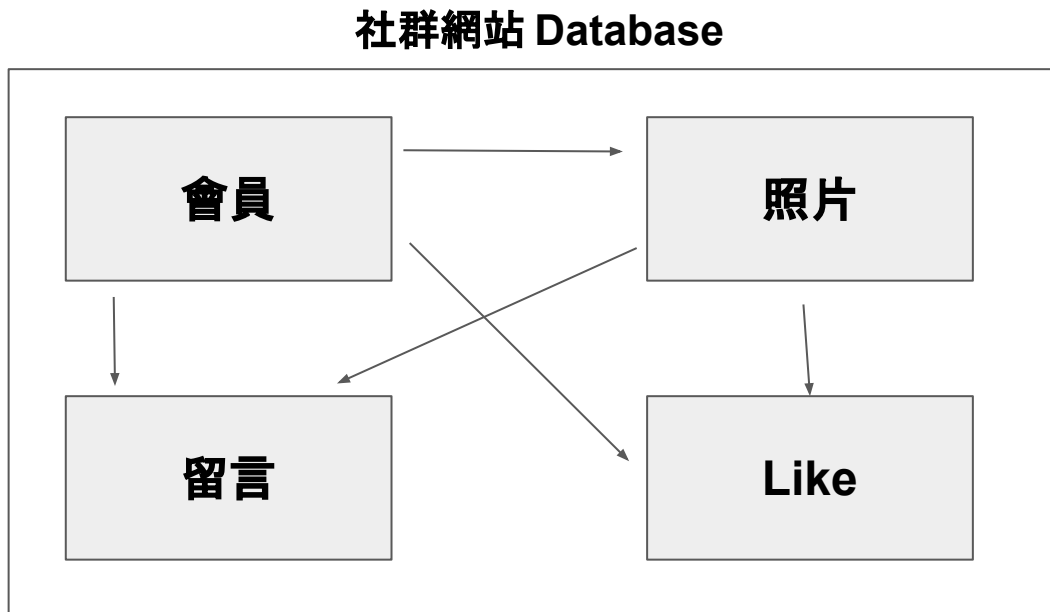
使用者可以對照片按讚,
這個讚也屬於這個照片

一個照片有許多留言

使用者可以在某照片留言,
留言也屬於這一個照片

Table 之間的關係

社群網站 Table 關聯關係



註：這邊只要先了解關係大概是如何，後面會詳細 說明有哪些關係定義。

Table 之間的關係

主要有四種關聯關係

1. 一對多 (One-to-Many)
2. 多對一 (Many-to-One)
3. 一對一 (One-to-One)
4. 多對多 (Many-to-Many)

Table 之間的關係

一對多(One-to-Many) 關聯



使用者有許多照片

註：我們先假設一個貼文只有一張照片

Table 之間的關係

一對多(One-to-Many) 關聯

以使用者的角度來看照片會是一對多關係

也會使用以下句子描述：
一位使用者有多個照片
A user **has many** photo

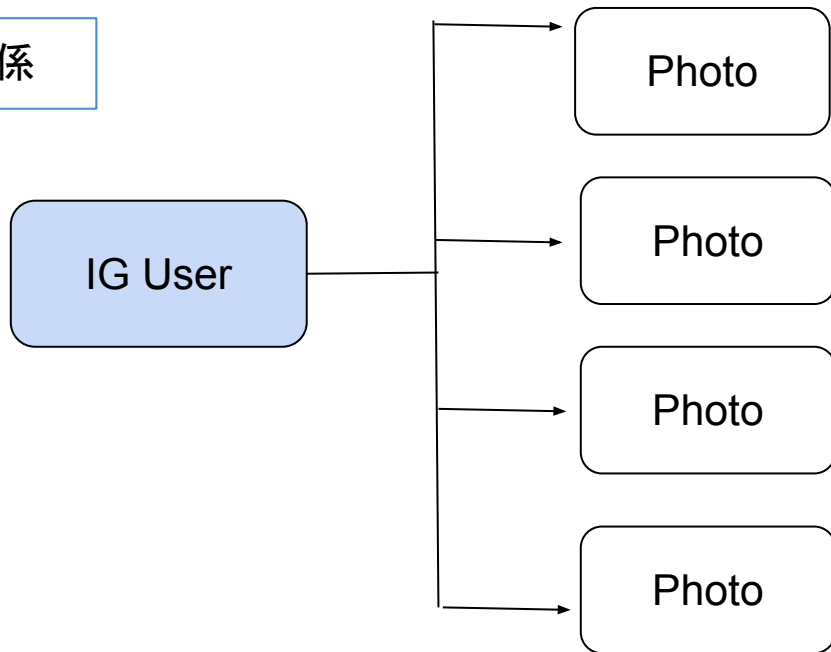
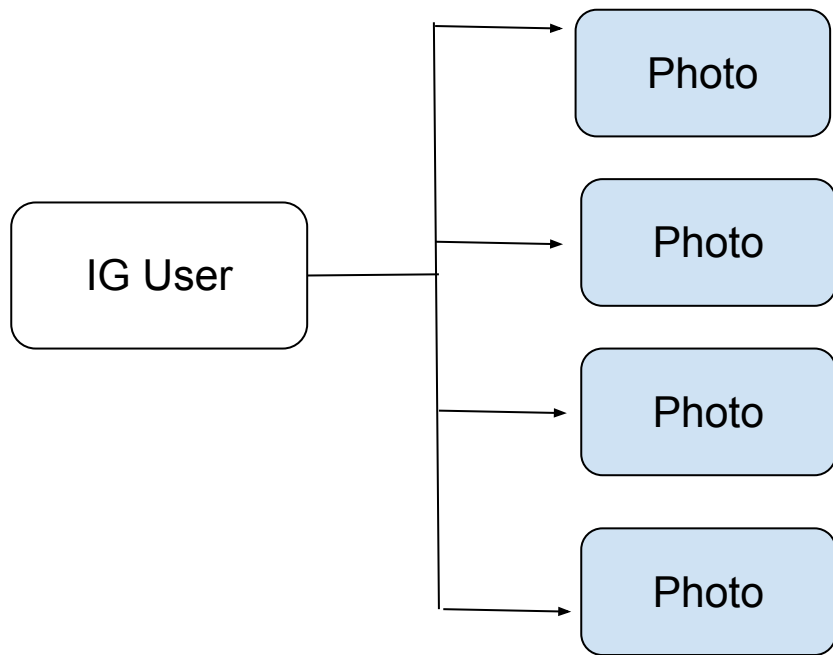


Table 之間的關係

多對一(Many-to-One) 關聯



若以照片的角度來看使用者，
則是多對一。
多張照片有同一個使用者且
一張照片只屬於一個使用者
A photo **has one** user.

說明：所以「一對多」與「多對一」是描述同
樣結構的資料，只是觀看角度的不同

Table 之間的關係

一對多與多對一 關聯

使用者可以有很多讚(like)
，一個讚只會屬於一個使用者



一張照片有多個留言，一個留言只會屬於一張照片

Table 之間的關係

照片對留言：一對多(One-to-Many) 關聯

以照片的角度來看留言會是一對多關係

也會使用以下句子描述：
一張照片有多個留言
A photo **has many** comments.

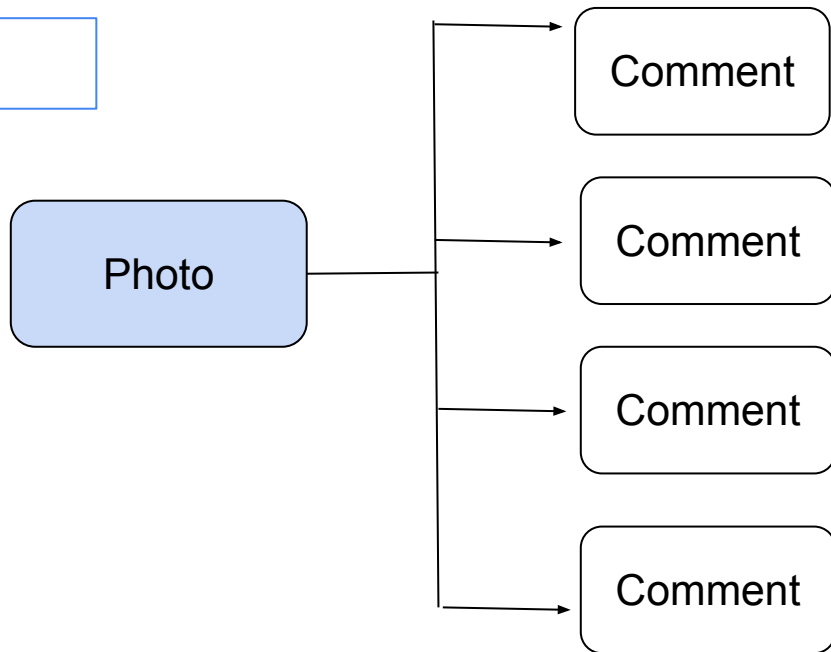
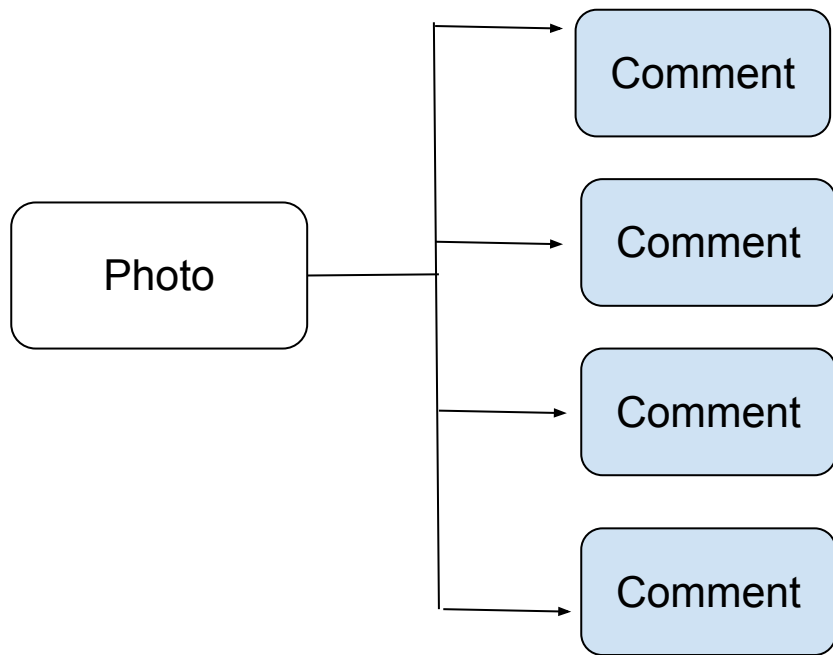


Table 之間的關係

留言對照片：多對一(Many-to-One) 關聯



若以留言的角度來看照片，
則是多對一。
多個留言屬於同一張照片且
一個留言只屬於一張照片
A comment **has one** photo.

Table 之間的關係

一對多、多對一 其他例子

- 一間公司有多位員工, 一位員工只屬於一間公司
- 一個出版社有多本書, 一本書只會屬於一個出版社
- 一個球隊有多位球員, 一位球員只會屬於一個球隊
- 一個廠牌有多種筆電, 一台筆電只屬於一個廠牌
- 一個班級有多位學生, 一位學生只屬於一個班級
- 一個貼文有多個讚, 一個讚只會屬於一個貼文

Table 之間的關係

一對一關係(One-to-One)

- 一對一關係是指兩個表格之間，一邊的一筆資料，只會屬於另一邊的一筆資料。
- 方向反過來，一邊也會只有另一邊的一筆資料。

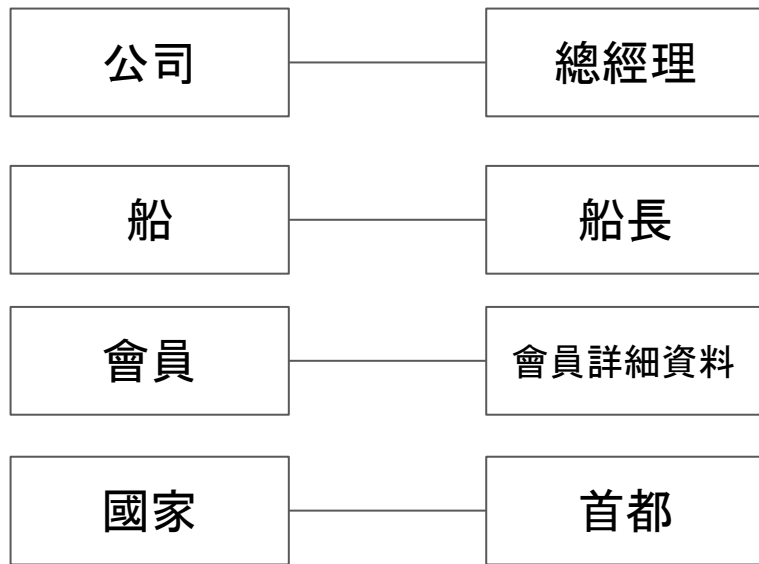


Table 之間的關係

多對多關係(Many-to-Many)

- 多對多關係是指這邊表個的多筆資料，在對面表格也有多筆資料。
- 因文字說明比較抽象，我們直接看例子：



外來鍵

Foreign Key

外來鍵 (Foreign Key)

基本規則

- 外來鍵簡稱外鍵, 有時以FK 簡稱, 須自己定義
- 又稱外來鍵約束(Foreign Key Constraint)
- 外鍵的作用是確保資料的一致性、完整性
- 一個表可以有一個或多個外鍵
- 外來鍵可以是本表主鍵
- 外來鍵可以不是本表的主鍵, 但這時一定要是其他表的主鍵
- 外鍵可以是 NULL , 但若不是 NULL 則必須要是另外一個表的一行資料(row)中主鍵的值

外來鍵 (Foreign Key) 示意圖

Database

Photos		
id INT	url VARCHAR(200)	user_id INT
1	https:...	3
2	https:...	3
3	https:...	1

主鍵

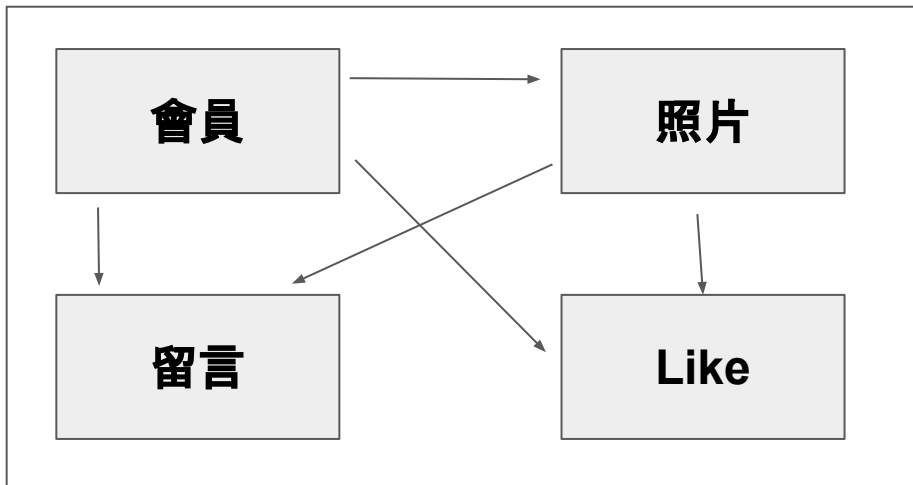
外來鍵

users		
id INT	u_name VARCHAR(200)	email VARCHAR(50)
1	阿明	gmail...
2	Tom	hotmail...
3	Sara	outlook...
4	小美	xxx@outl ook...

主鍵

外來鍵 (Foreign Key) 示意圖

社群網站 Database



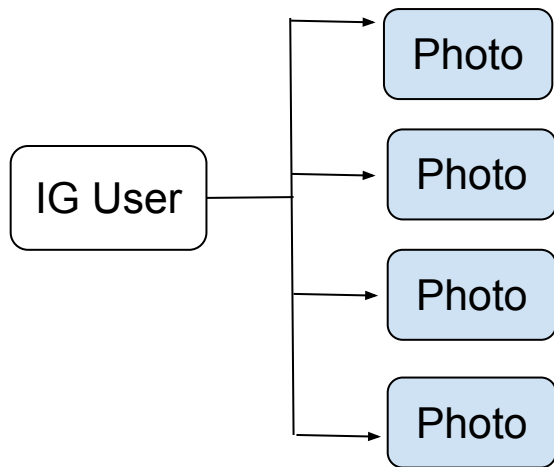
留言屬於一張照片：留言資料表要有一個外來鍵的欄位指向照片 id 欄位，因為每一個留言都要屬於一張照片

留言屬於一個會員：留言資料表要有一個外來鍵欄位指向會員 id 欄位，因為每一個留言都屬於一個會員

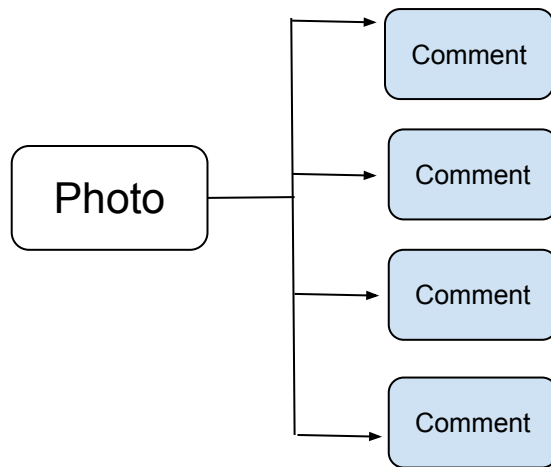
外來鍵 (Foreign Key)

外來鍵的位置

外來鍵欄位的位置要在多方表格



要有一個外來鍵欄位
指向 IGuser



要有一個外來鍵欄位
指向 Photo

外來鍵 (Foreign Key) 留言(comments)的外鍵

comments			
id INT	text VARCHAR(200)	user_id INT	photo_id INT
1	good	3	2
2	酷唷	3	3
3	太神啦	1	2
4	很可以	2	1

外鍵

外鍵

users		
id INT	u_name VARCHAR(200)	email VARCHAR(50)
1	阿明	xxx@gm ail...
2	Tom	xx@hotm ail...
3	Sara	xxx@outl ook...
4	小美	xxx@outl ook...

Photos		
id INT	url VARCHAR(200)	user_id INT
1	https:xxx	3
2	https:xxx	3
3	https:xx	1

外來鍵 (Foreign Key)

建立外鍵

```
create table table_name(  
  id int not null primary key auto_increment,  
  ...  
  本表欄位 int,  
  FOREIGN KEY (本表欄位) REFERENCES 要對應的表(要對應的欄位)  
);
```

```
create table photos(  
  id int not null primary key auto_increment,  
  url varchar(200),  
  user_id int,  
  FOREIGN KEY (user_id) REFERENCES users(id)  
);
```

外來鍵 (Foreign Key)

外鍵欄位 MUL 記號

- 使用 DESC photos; 觀察表格

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
url	varchar(200)	YES		NULL	
user_id	int	YES	MUL	NULL	

說明：MUL 是 Multiple 的意思，指的是這個 key 允許出現重複的值。

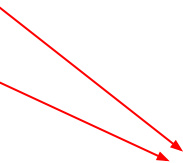
註：key 總共有三種 (PRI, UNI, MUL)，目前已經學會 PRI, MUL，UNI 指的是指定該欄位的值不能重複，但不是主鍵，使用 unique 修飾欄位。
這三種 key 都內建 index，搜尋速度較快。速度上 PRI > UNI > MUL

外來鍵 (Foreign Key)

外鍵與資料一致性(情況1)

外鍵可以保持資料的一致性，因為可以避免意外輸入不存在的資料。

Photos		
id INT	url VARCHAR(200)	user_id INT(fk)
1	https:...	3
2	https:...	3
3	https:...	1
4	https:...	2



users		
id INT	u_name VARCHAR(200)	email VARCHAR(50)
1	阿明	gmail...
2	Tom	hotmail...
3	Sara	outlook...
4	小美	xxx@outlook...

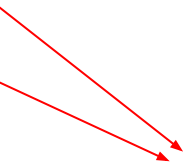
情況1. 外鍵資料存在 > 可以新增資料

外來鍵 (Foreign Key)

外鍵與資料一致性(情況2)

外鍵可以保持資料的一致性，因為可以避免意外輸入不存在的資料。

Photos		
id INT	url VARCHAR(200)	user_id INT(fk)
1	https:...	3
2	https:...	3
3	https:...	1
4	https:...	8888



users		
id INT	u_name VARCHAR(200)	email VARCHAR(50)
1	阿明	gmail...
2	Tom	hotmail...
3	Sara	outlook...
4	小美	xxx@outlook...

情況2. 外鍵資料 **不存在** > 因外來鍵的約束，新增資料時就會出現錯誤，避免塞入不存在的資料

外來鍵 (Foreign Key)

外鍵與資料一致性(情況3)

外鍵可以保持資料的一致性，因為可以避免意外輸入不存在的資料。

Photos		
id INT	url VARCHAR(200)	user_id INT(fk)
1	https:...	3
2	https:...	3
3	https:...	1
4	https:...	null

users		
id INT	u_name VARCHAR(200)	email VARCHAR(50)
1	阿明	gmail...
2	Tom	hotmail...
3	Sara	outlook...
4	小美	xxx@outlook...

情況3. 不想將照片指定使用者，單純想新增照片 > 塞入 null > 可以

外來鍵 (Foreign Key)

表格關聯時，刪除的四種約束模式

1. RESTRICT 模式:
預設的模式，刪除大表時，若小表有對應資料，則會跳出錯誤訊息，且無法刪除。
2. CASCADE 模式:
又稱級聯模式，刪除大表時，若小表有對應資料，則兩邊都刪除相關資料。
3. SET NULL 模式:
刪除大表時，若小表有對應資料，則將小表對應資料的外鍵欄位設定為NULL，小表資料不會被刪除，大表會成功刪除。
4. NO ACTION 模式:
與 RESTRICT 模式相同，只是與其他 RDBMS 命名相同。

外來鍵 (Foreign Key)

測試四種模式前置作業(先複製到筆記本)

```
create table photos(  
  id int not null primary key auto_increment,  
  url varchar(200),  
  user_id int,  
  FOREIGN KEY (user_id) REFERENCES users(id)  
);
```

```
INSERT INTO photos (url, user_id)  
VALUES  
  ('http://one.jpg', 4),  
  ('http://two.jpg', 1),  
  ('http://25.jpg', 1),  
  ('http://36.jpg', 1),  
  ('http://754.jpg', 2),  
  ('http://35.jpg', 3),  
  ('http://256.jpg', 4);
```

外來鍵 (Foreign Key)

測試 RESTRICT 模式

1. RESTRICT 模式:

預設的模式，刪除大表時，若小表有對應資料，則會跳出錯誤訊息，且無法刪除。

```
delete from users where id =1;
```


外來鍵 (Foreign Key)

測試 CASCADE 模式

2. CASCADE 模式:

又稱級聯模式，刪除大表時，若小表有對應資料，則兩邊都刪除相關資料。

```
create table photos(  
  id int not null primary key auto_increment,  
  url varchar(200),  
  user_id int,  
  FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE  
);
```

外來鍵 (Foreign Key)

測試 SET NULL 模式

3. SET NULL 模式:

刪除大表時，若小表有對應資料，則將小表對應資料的外鍵欄位設定為 NULL，小表資料不會被刪除，大表會成功刪除。

```
create table photos(  
  id int not null primary key auto_increment,  
  url varchar(200),  
  user_id int,  
  FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE SET NULL  
);
```

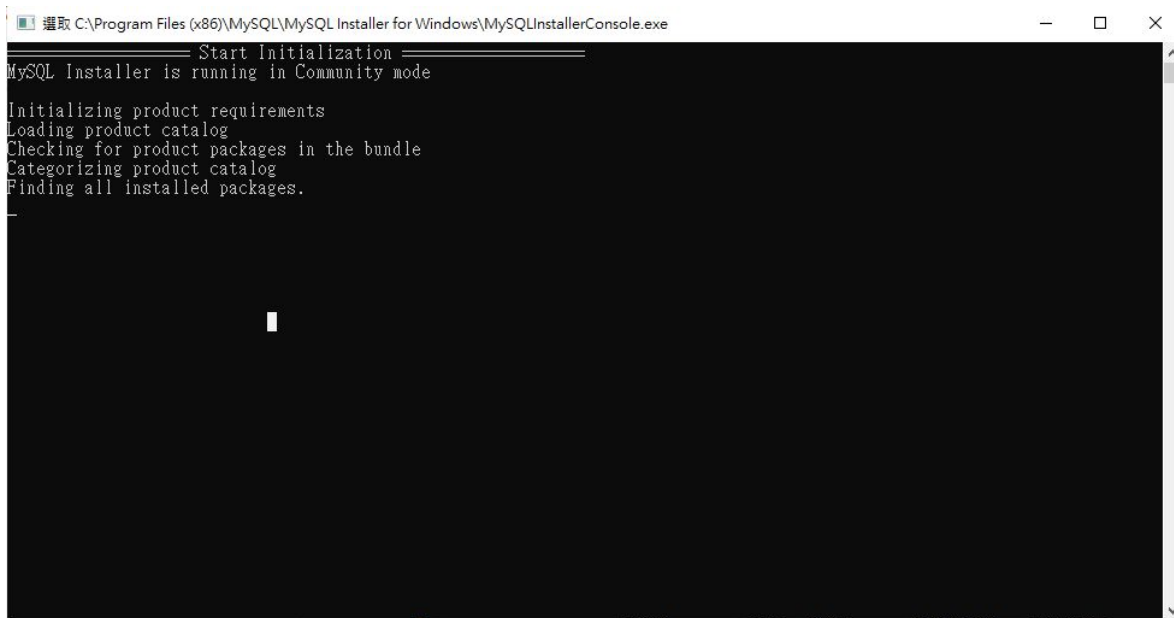
這次想刪除 USER4 但是想保留 USER4 的照片。

附錄

附錄1. 跳出視窗處理

若是安裝完會一直跳出右方視窗，這個是 MySQL 在做更新(預設是每日 0:00)，若不希望專注時跳出，可以先暫時停用。請根據以下步驟關掉排程

控制台 > 系統及安全性 > 系統管理工具 > 排程工作 > 找到 MySQL 底下的 Installer > 中間 ManifestUpdate 按右鍵 > 停用



註：不用專心時記得啟動它，方便更新

附錄2. 開啟與停止 MySQL Server

Windows 開始 > 服務 > 找到 MySQL80
> 按右鍵選擇 > 停止或啟動

