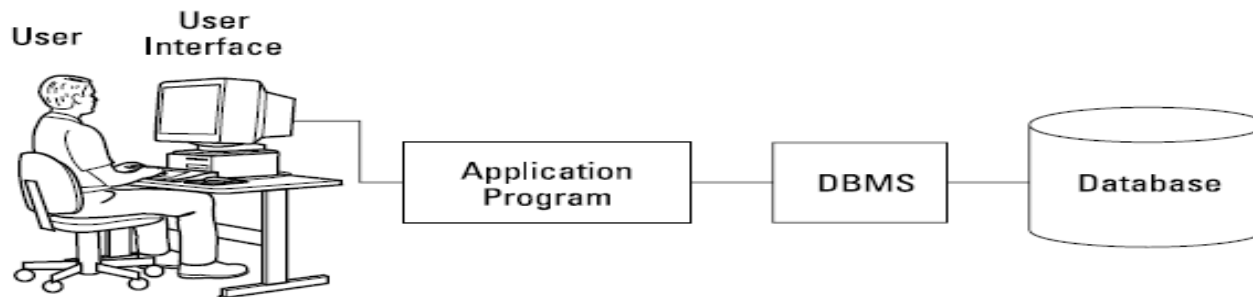
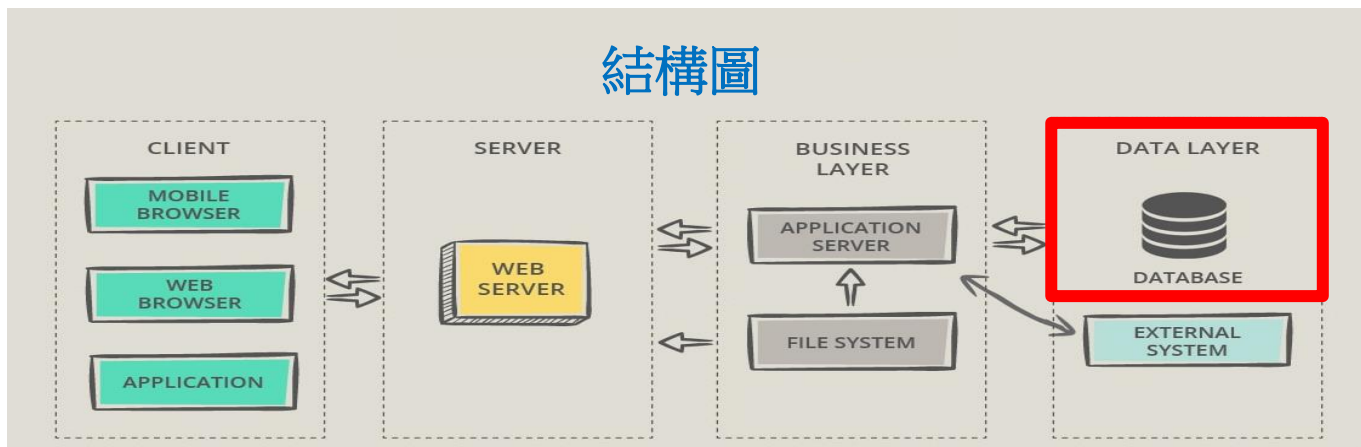




# 資料庫概論

什麼是  
SQL資料庫呢？

## 結構圖



# 所以身為一個資料庫，它就應該.....

1. 存資料超方便超快。
2. 存取的載體要夠穩定，我不要再有資料遺失。
3. 不管今天我們資料量成長到多大，我希望查詢的時間都不要變慢.....
4. 我希望這個資料庫可以簡單、快速的把我要的資料取出來.....
5. 資料庫應該是可以多個人一起用的.....

## 如果你用Excel存資料...那你可能碰到...

1. 當你資料不斷長大，你的Excel可能...[GG](#)
2. Excel最方便的就是「修改資料」超方便。
3. 因為只有你有檔案，所以這個那個都你改。

# SQL為什麼叫SQL

IBM 發展了結構化英文查詢語言(Structured English Query Language) , 並於 1974 年將此語言命名為SEQUEL。

但很不幸地，SEQUEL 已被英國的航空公司註冊，所以 “English” 這個字眼被放棄了。

於是，此新語言被更名為 SQL 或 Structured Query Language。  
(但是，它的發音仍然是 "sequel"。)

Structured Query Language 直接翻譯是結構化查詢語言，但直接翻譯對這個語言的理解毫無幫助，我們試圖用一句話解釋：

SQL 是 Structured Query Language 的縮寫，是一個專門針對關聯式資料庫中所儲存的資料進行查詢、定義、操作與控制的語言。

SQL 在 1970 年代由國際商業機器公司（IBM）創造，剛開發出來時候僅只是為了更有效率地「查詢」儲存於關聯式資料庫中的資料，但是到了現代，除了查詢以外像是資料的建立、更新與刪除，也都能靠著 SQL 來完成。

具體來說，SQL 是由保留字（Keyword）、符號、常數與函數所組合而成的一種語言，按照使用目的可以再細分為資料查詢語言（Data Query Language, DQL）、資料定義語言（Data Definition Language, DDL）、資料操作語言（Data Manipulation Language, DML）、資料控制語言（Data Control Language, DCL）以及交易控制語言（Transaction Control Language, TCL）；本書是初學者友善的導向，內容會以資料查詢語言為主，資料定義語言與資料操作語言為輔。

因此我們可以理解 **SQL** 是一個能夠與關聯式資料庫互動的專用語言，常見的互動有四個：

包含創造（**Create**）、查詢（**Read**）、更新（**Update**）與刪除（**Delete**），這四個動作又在業界與社群被簡稱為 **CRUD**，

聽起來十分抽象，但其實與現代生活形影不離。舉例來說在社群應用程式中的一舉一動，不論是透過滑鼠點擊或者手勢觸控，都會被應用程式轉換為 **CRUD** 的指令：上傳新的動態與貼文，就是創造的體現；瀏覽追蹤對象的動態與貼文，就是查詢的體現；編輯動態與貼文，就是更新的體現；撤掉動態與貼文，就是刪除的體現。



# 什麼是關聯式 資料庫

欲解釋何謂關聯式資料庫，我們將它拆成「關聯式」與「資料庫」分別定義。資料庫是一種特定、經過加工的資料集合，能夠放置在伺服器、個人電腦、手機或者微型電腦之中，資料庫可以透過 **SQL** 與之互動，有效率地進行資料查詢、定義、操作與控制。關聯式則是描述資料庫中的資料集合是以列 ( **Rows** ) 與欄 ( **Columns** ) 所組成的二維表格形式記錄，並且遵守關聯式模型準則設計，這樣的資料庫就被稱為關聯式資料庫。

有時列(row)也有其他別名，像是紀錄 ( **Records** )、觀測值 ( **Observations** )、元組 ( **Tuples** ) 等；  
欄(col)的其他別名則有欄位 ( **Fields** )、變數 ( **Variables** )、屬性 ( **Attributes** ) 等。

<https://db-engines.com/en/ranking>

421 systems in ranking, July 2024

Rank			DBMS	Database Model	Score		
Jul 2024	Jun 2024	Jul 2023			Jul 2024	Jun 2024	Jul 2023
1.	1.	1.	Oracle +	Relational, Multi-model ⓘ	1240.37	-3.72	-15.64
2.	2.	2.	MySQL +	Relational, Multi-model ⓘ	1039.46	-21.89	-110.89
3.	3.	3.	Microsoft SQL Server +	Relational, Multi-model ⓘ	807.65	-13.91	-113.95
4.	4.	4.	PostgreSQL +	Relational, Multi-model ⓘ	638.91	+2.66	+21.08
5.	5.	5.	MongoDB +	Document, Multi-model ⓘ	429.83	+8.75	-5.67
6.	6.	6.	Redis +	Key-value, Multi-model ⓘ	156.77	+0.82	-7.00
7.	↑ 8.	↑ 11.	Snowflake +	Relational	136.53	+6.17	+18.84
8.	↓ 7.	8.	Elasticsearch	Search engine, Multi-model ⓘ	130.82	-2.01	-8.77
9.	9.	↓ 7.	IBM Db2	Relational, Multi-model ⓘ	124.40	-1.50	-15.41
10.	10.	10.	SQLite +	Relational	109.95	-1.46	-20.25

## **SQL 與關聯式資料庫管理系統是重要的**

SQL 與關聯式資料庫管理系統不論是對於資料科學家、軟體工程師都是至關重要的語言與技術，在資料科學應用領域中，關聯式資料庫管理系統是常見的資料來源，對於在大型企業工作的資料科學家來說更是如此，透過 SQL 能夠載入產品、服務與財務的相關數據進行分析探索；

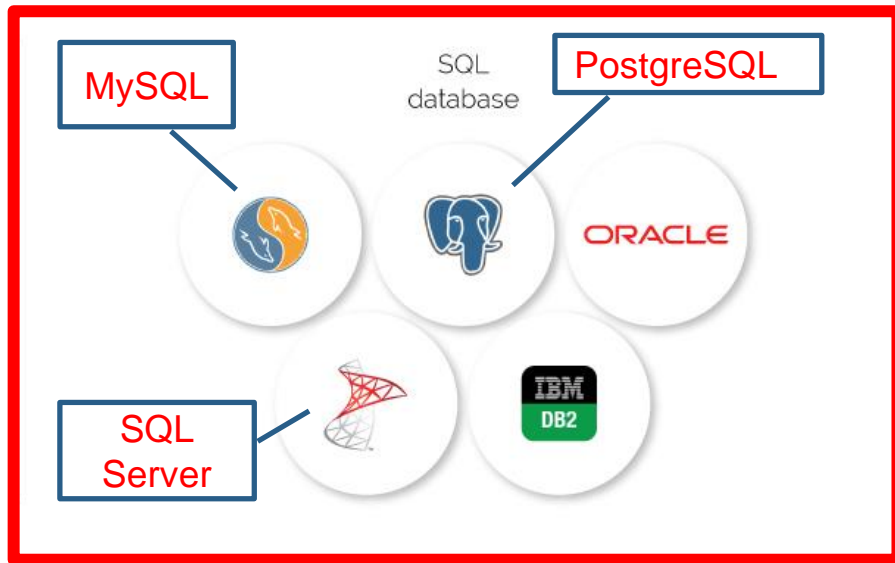
在軟體開發領域中，關聯式資料庫管理系統是網頁應用程式、手機應用程式或桌面應用程式不可或缺的一環，透過 SQL 能夠讓應用程式的使用者與資料進行互動。就算並非從事資料科學、軟體開發的相關領域工作，生活中關聯式資料庫管理系統也是無所不在，小至手機的通話紀錄與通訊錄、大至社群應用程式與購物網站、銀行的存款與交易資訊，能夠讓成千上萬個使用者自動化、大規模地同時運作無虞，背後都有 SQL、關聯式資料庫管理系統與應用程式在支撐。

# 資料庫的兩大陣營

NoSQL  
database



SQL  
database



# NoSQL

NoSQL是Not Only SQL的縮寫，和剛才前面介紹的不同是，NoSQL 資料庫是一種非關聯式資料庫，將資料儲存為類似 JSON 的文件，並對資料進行查詢，這是一個 document 資料庫模型，在這個模型中，資料並不存儲在 Table 中，document 是 key-value 的有序集合。資料庫中的每個document 不需要具有相同的數據結構，下方使用紀錄書籍資訊的JSON格式作為範例，讓你更容易理解。

```
[ { "year" : 2017, "title" : "斜槓青年", "info" :  
  { "release_date" : "2017-09-01", "rating" : 8.2,  
    "genres" : ["生涯規劃", "商業理財"], "plot" : "全球職涯  
    新趨勢，迎接更有價值的多職人生", "actors" : "Susan  
    Kuang" } }, { "year": 2017, "title": "巴菲特寫給股東的信",  
  "info": { "plot": "巴菲特親筆撰述唯一著作", "rating":  
    8.3 } } ]
```

請你幫我整理一下上面的這段碼，幫我看看你發現了甚麼？

# NoSQL

存取NoSQL資料的方式，必須使用資料庫系統提供的 API 才能夠新增、修改、刪除資料，也就是需要透過程式語言呼叫特定的函式，或是使用資料庫系統 提供的指令，遠端連線執行後才能存取資料

NoSQL 資料庫的優點是它們能夠處理大量的結構化或半結構化數據，更容易擴展，並且能夠提供高可用性。使用NoSQL 資料庫的一些好例子包括：事件記錄（例如存儲用戶登錄）、電子商務應用（存儲用戶購物車的數據）和即時的分析。

# SQL與NoSQL 的比較

雖然NoSQL 資料庫與關聯式資料庫相比確實有一些優勢，但也有很多缺點。

數據一致性和數據完整性是NoSQL資料庫的主要問題，另一個缺點是缺乏標準化。例如SQL語法適用於所有關聯式資料庫（MySQL、Oracle、MS SQL等），雖然一些特定的函式，語法上有些差異，但大致來說都是使用SQL的語法操作資料庫，而NoSQL資料庫沒有這樣的標準語言，不同的NoSQL資料庫之間存在很大的差異，當你使用AWS DynamoDB，我們需要匯入AWS的套件，以及使用特定的函式操作，但如果是使用另一個NoSQL資料庫，像是Firebase，那就需要匯入另一個套件和使用另一組函式來操作資料庫，所以學習SQL 程式語言會是CP值更高的一種投資，因為你不用再另外花時間學習多個操作資料庫的方法。

在這個信息驅動的大數據時代，為你開發的系統選擇正確的資料庫，比以往任何時候都重要。雖然關聯式資料庫已經存在了40多年，你可能會被“較新”的NoSQL資料庫技術所吸引，但有一些事情你需要考慮。關係型資料庫提供的數據一致性和數據完整性意味著，如果你的系統是處理金融信息和交易，那麼數據一致性是必須的。如果你希望你的資料庫是安全的，關聯式資料庫提供的安全功能，在這方面有明顯的優勢，而且你只需要學習10~20條簡單的SQL語法，就能夠操作大部分的資料庫數據。

# 資料庫設計的概念





Name: Anna Laurier  
Email: anna.laurier@customer2.net  
Phone: +1 555 2222

Name: John Smith  
Email: john.smith@customer1.com  
Phone: +1 555 1111

資料庫 ( DB ) 是一個電子倉庫，用於存儲數據。DB中的數據是對現實世界中存在的信息的表示，並且它們被嚴格組織管理。我們的電子郵件、社交網絡、銀行帳戶、醫療記錄或學校記錄都存儲在數據庫中。我們可以在數據庫中存儲實際上的任何信息。

接下來我們要舉個例子幫助您形成整個概念。讓我們將聯絡人從舊筆記本（紙質）轉移到文件（電子）中。首先，我們創建一個文本文件，如下所示：

在我們添加了一些聯絡人之後，我們注意到每個聯絡人都重複顯示姓名、電子郵件和電話信息，這不僅浪費時間，還占用了空間。我們將文本文件的格式轉換為以下格式：

```
Name: Anna Laurier  
Email: anna.laurier@customer2.net  
Phone: +1 555 2222
```

```
Name: John Smith  
Email: john.smith@customer1.com  
Phone: +1 555 1111
```

Name	Email	Phone
Anna Laurier	anna.laurier@customer2.net	+1 555 2222
John Smith	john.smith@customer1.com	+1 555 1111

我們的下一步是將數據從文本文件移動到像Microsoft Office Excel或 LibreOffice Calc這樣的電子表格軟件中 - 這是最簡單的數據庫。

電子表格使我們比文本文件更容易處理信息 - 可將每個數據切片插入到自己的單元格中，進行快速搜索，對列添加過濾器，對數據進行排序等等。

Name	Email	Phone
Anna Laurier	anna.laurier@customer2.net	+1 555 2222
John Smith	john.smith@customer1.com	+1 555 1111

當我們開始使用這個試算表（實際上這是我們的第一個數據庫）時，我們為單個聯絡人添加了多個電子郵件和電話。

Name	Email	Phone
Anna Laurier	anna.laurier@customer2.net, al@customer2.net	+1 555 2222, +1 555 2223
John Smith	john.smith@customer1.com, js@customer1.com	+1 555 1111, +1 555 1112, +1 555 1113

哎呀！我們發現了第一個數據問題：我們使用逗號 (,) 來分割數據，但逗號本身也是數據的一部分。

我們通過將數據拆分成單獨的單元格來解決這個問題：

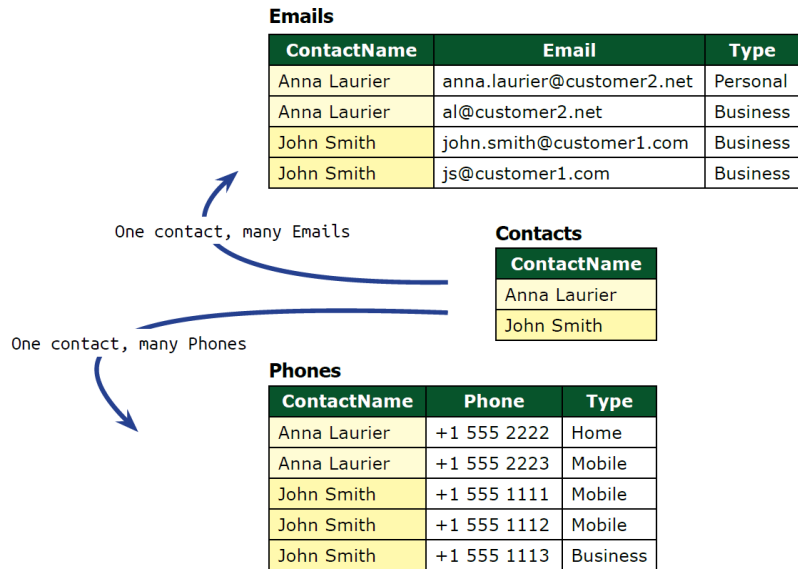
Name	Email 1	Email 2	Phone 1	Phone 2	Phone 3
Anna Laurier	anna.laurier@customer2.net	al@customer2.net	+1 555 2222	+1 555 2223	
John Smith	john.smith@customer1.com	js@customer1.com	+1 555 1111	+1 555 1112	+1 555 1113

後來，我們搜索一個聯繫人，我們感到困惑："Phone 1"列是移動電話還是商務電話？同樣的問題也適用於"Email"列。  
我們添加了額外的欄位來定義電話和郵件的類型：

Name	Email 1	Email 1 Type	Email 2	Email 2 Type	Phone 1	Phone 1 Type	Phone 2	Phone 2 Type	Phone 3	Phone 3 Type
Anna Laurier	anna.laurier@customer2.net	Personal	al@customer2.net	Business	+1 555 2222	Home	+1 555 2223	Mobile		
John Smith	john.smith@customer1.com	Business	js@customer1.com	Business	+1 555 1111	Mobile	+1 555 1112	Mobile	+1 555 1113	Business

使用試算表一年後，我們注意到一個人有50個電話，而我們添加了100列（每個"電話"和"電話類型"一對），但是...如果我們將這些屬性（郵件和電話）拆分為單獨的表格，並且一個聯繫人的一個電話是一行呢？

接下來的步驟是將我們的聯繫人數據庫從存儲所有數據的單一表格轉換為多個相關的表格：



我們剛剛建立了我們的第一個一對多關係 - 一個聯繫人可以擁有多個電話和郵件地址。

幾個月後，我們決定為電話和郵件類型添加描述。

我們發現，將電子郵件和電話類型存儲在單獨的表中並將它們鏈接到存儲電子郵件和電話的表格中將更容易管理。

我們還決定為我們的聯繫人添加圖片。由於我們決定每個聯繫人只顯示一張圖片，所以我們在"聯繫人"表中添加了"Picture"欄位。

另一個重大的變更是將長列標題"ContactName"、"EmailType"和"PhoneType"替換為短數字，用於鏈接表格。

EmailTypes

EmailTypeID	EmailTypeName	EmailTypeDescription
1	Personal	Send jokes, pictures and any other personal data.
2	Business	Related to business conversations. Send contacts and invoices.

Emails

ContactID	Email	EmailTypeID
1	anna.laurier@customer2.net	1
1	al@customer2.net	2
2	john.smith@customer1.com	2
2	js@customer1.com	2

Contacts

ContactID	ContactName	Picture
1	Anna Laurier	1-1.png
2	John Smith	2-3.png

Link on ContactID

Phones



ContactID	Phone	PhoneTypeID
1	+1 555 2222	1
1	+1 555 2223	2
2	+1 555 1111	2
2	+1 555 1112	2
2	+1 555 1113	3

PhoneTypes

PhoneTypeID	PhoneTypeName	PhoneTypeDescription
1	Home	Stationary at home. Used by all family members.
2	Mobile	Available all the time.
3	Business	In business hours only. Located in the office.



在我們的示例中的最後一個步驟  
是將數據轉換為像這樣的"易讀"  
報告：

Anna Laurier	
	<b>Emails</b>
	anna.laurier@customer2.net (Personal)
	al@customer2.net (Business)
	<b>Phones</b>
	+1 555 2222 (Home)
	+1 555 2223 (Mobile)
John Smith	
	<b>Emails</b>
	john.smith@customer1.com (Business)
	js@customer1.com (Business)
	<b>Phones</b>
	+1 555 1111 (Mobile)
	+1 555 1112 (Mobile)
	+1 555 1113 (Business)

# 何謂Table

當您將信息存儲在文件櫃中時，您不只是將它扔進抽屜裡。相反，您會在文件櫃內創建文件，然後將相關數據存儲在特定文件中。

在數據庫世界中，該文件稱為表格(**Table**)。表格是一個結構化的文件，可以存儲特定類型的數據。一個表格可能包含客戶列表、產品目錄或任何其他信息列表。

關鍵在於存儲在表格中的數據是同一種類型的數據。您永遠不會在同一個數據庫表格中存儲客戶列表和訂單列表。這樣做將使後續的檢索和訪問變得困難。相反，您會創建兩個表格，每個表格對應一個列表。

數據庫中的每個表格都有一個標識它的名稱。該名稱始終是唯一的，這意味著該數據庫中的其他表格不能具有相同的名稱。

注意：表格名稱 使表格名稱唯一的實際上是數個因素的組合，包括數據庫名稱和表格名稱。這意味著雖然您不能在同一數據庫中兩次使用相同的表格名稱，但您絕對可以在不同的數據庫中重複使用表格名稱。

表格具有定義數據存儲方式的特性和屬性。這些特性包括有關可以存儲哪些數據、數據如何分割、如何命名各個信息片段等信息。描述一個表格的這一組信息稱為模式 (**schema**)，模式用於描述數據庫內特定的表格以及整個數據庫（以及其中表格之間的關係，如果有的話）。

# Columns

表格由列(Columns)組成。每個列包含表格中的特定信息片段。

理解這一點的最佳方法是將數據庫表格想像成網格，有點像電子表格。網格中的每一列都包含特定的信息片段。例如，在客戶表格中，一列包含客戶編號，另一列包含客戶名稱，地址、城市、州和郵政編碼都存儲在它們自己的列中。

提示：拆分數據 正確將數據拆分為多個列非常重要。例如，城市、州和郵政編碼應始終是分開的列。通過這樣拆分，可以按特定列（例如，查找特定州或城市的所有客戶）對數據進行排序或過濾。如果城市和州合併為一列，將極難按州進行排序或過濾。

# Datatype

數據庫中的每個列都有相關的數據類型。數據類型定義了列可以包含的數據類型。例如，如果列要包含一個數字（也許是訂單中的商品數量），那麼數據類型將是數值型的。如果列要包含日期、文本、備註、貨幣金額等，則將使用適當的數據類型來指定這些內容。

新術語：數據類型(Datatype) 允許的數據類型。每個表格列都有相關的數據類型，限制（或允許）該列中的具體數據。

數據類型限制了可以存儲在列中的數據類型（例如，防止將字母字符輸入到數值字段中）。數據類型還有助於正確排序數據，並在優化磁盤使用方面發揮重要作用。因此，在創建表格時必須特別注意選擇正確的數據類型。

# Row

行 表格中的數據存儲在行中；每個保存的記錄都存儲在自己的行中。再次想像表格為一個類似電子表格風格的網格，網格中的垂直列是表格的列，水平行是表格的行。

例如，一個客戶表格可能每行存儲一個客戶。表格中的行數是其中的記錄數。

# Primary Keys

主鍵 表格中的每一行都應該具有某些列（或一組列），這些列能夠唯一標識它。包含客戶信息的表格可能會使用客戶編號列，而包含訂單的表格可能會使用訂單ID。員工列表表格可能會使用員工ID或員工社會安全號碼列。

新術語：主鍵

列（或一組列），其值唯一標識表格中的每一行。能夠唯一標識表格中每一行的這一系列（或一組列）稱為主鍵。(A column (or set of columns) whose values uniquely identify every row in a table.)

主鍵用於參照特定行。如果沒有主鍵，更新或刪除表格中的特定行將變得極為困難，因為沒有保證安全的方法來引用要受影響的行。

提示：總是定義主鍵 儘管主鍵實際上並不是必需的，但大多數數據庫設計師確保他們創建的每個表格都具有主鍵，以便未來的數據操作能夠進行且易於管理。

# Primary Keys

## 如何設計主鍵

任何表格中的列都可以設置為主鍵，只要滿足以下條件：

1. 沒有兩行可以具有相同的主鍵值。
2. 每一行都必須具有主鍵值（主鍵列不允許 NULL 值）。

提示：主鍵規則 這裡列出的規則是由 SQL Server 自身強制執行的。通常，主鍵是在表格中的單個列上定義的。但這不是必需的，多個列可以一起用作主鍵。當使用多個列時，之前列出的規則必須適用於構成主鍵的所有列，所有列的值在一起必須是唯一的（各個列不需要具有唯一值）。

提示：主鍵最佳實踐 除了 SQL Server 強制執行的規則之外，還應遵守一些普遍接受的最佳實踐：

- 不要在主鍵列中更新值。
- 不要在主鍵列中重複使用值。
- 不要在主鍵列中使用可能會更改的值。（例如，當您使用名稱作為主鍵來識別供應商時，當供應商合併並更改其名稱時，您必須更改主鍵。）

啟動SQLiteStudio





[Home](#) [News](#) [Gallery](#) [Download](#) [GitHub](#)

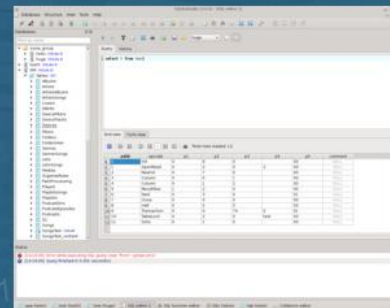
# SQLiteStudio

Create, edit, browse SQLite databases.

Download



Donate



Releases

Tags

Find a release

Apr 7, 2023

 pawelsalawa

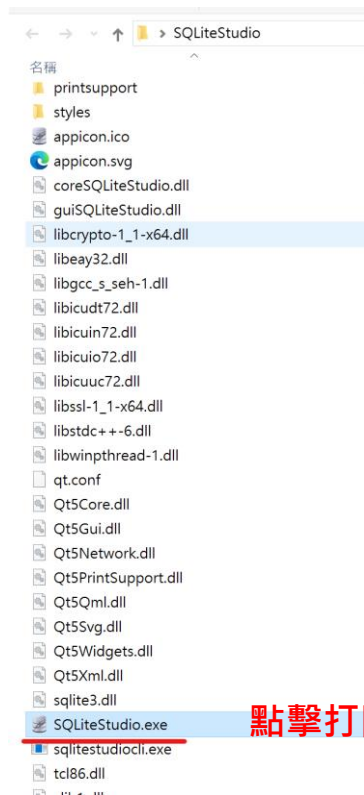
3.4.4

c65ea68

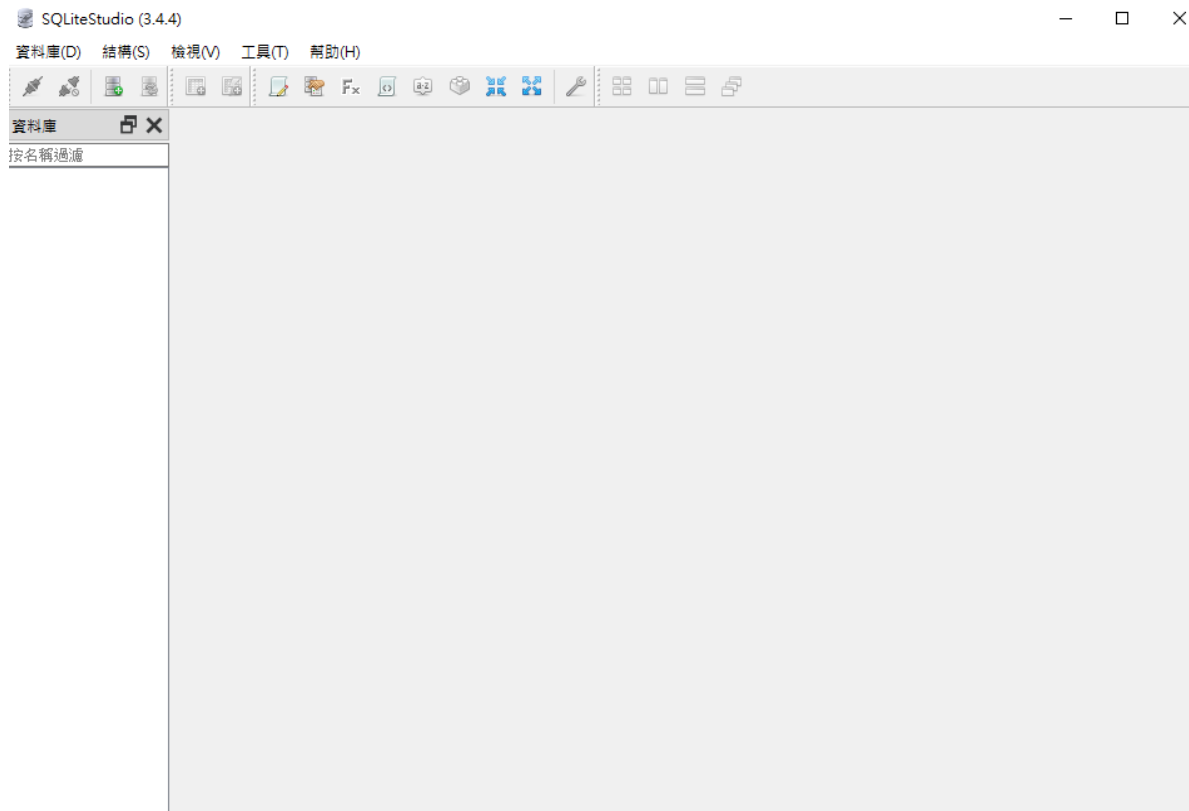
Compare

## 3.4.4 release Latest

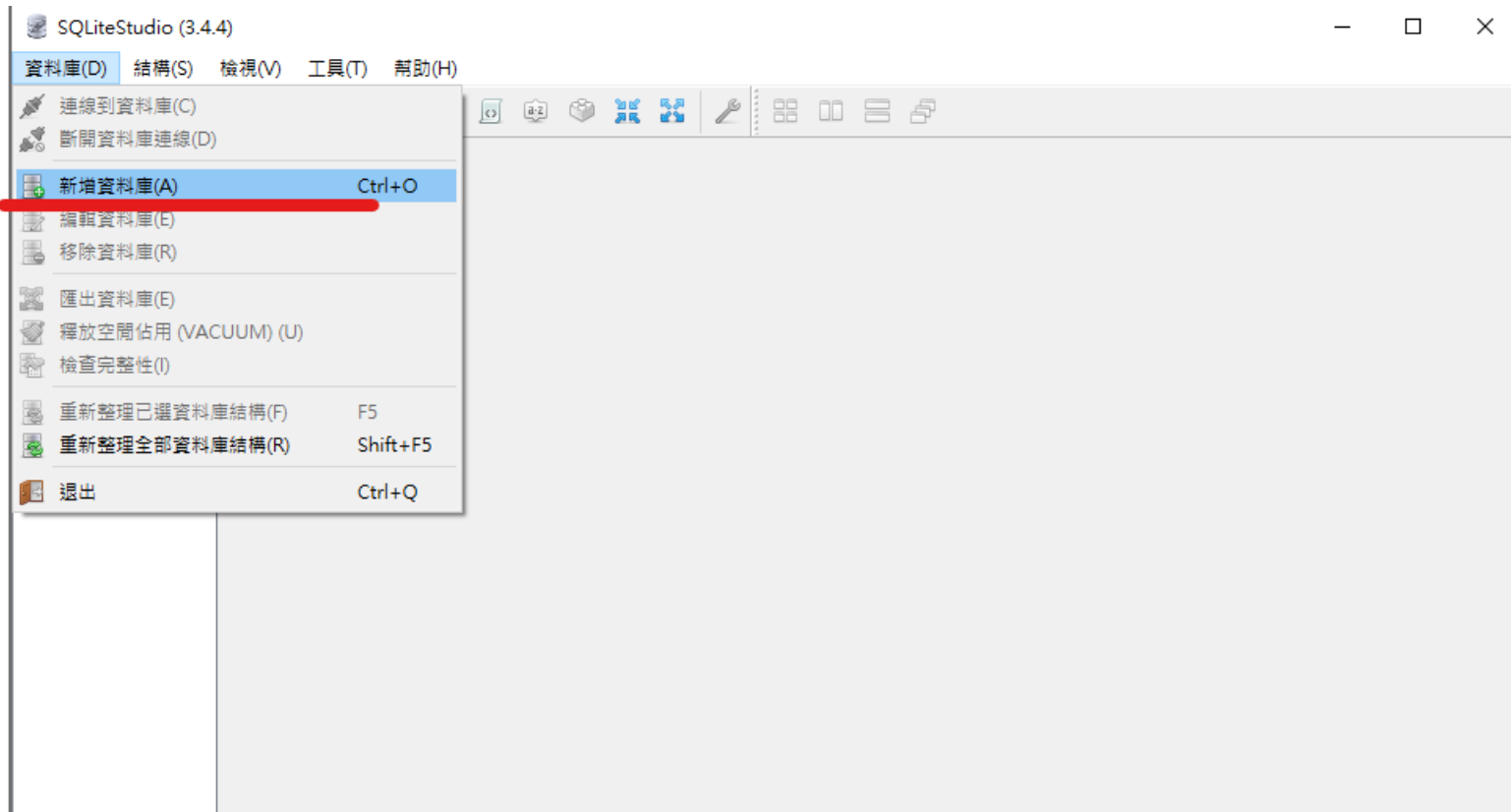
Platform	Package type	File name	SHA256 checksum
Windows x64	Portable	<a href="#">sqlitestudio_x64-3.4.4.zip</a>	1836c8c1f32d879098fcbda4d11391f91407b4b9fa8a3a799b61ec5b582b6425
Windows x64	Installer	<a href="#">SQLiteStudio-3.4.4-windows-x64-installer.exe</a>	4428f83914717fa581b4bd1eca35573739aa8ebee14e58b0392dec41b631a72
Windows i386	Portable	<a href="#">sqlitestudio_i386-3.4.4.zip</a>	1cc779d6eb41952131eba63e9d221454daf9e582e864ec64f32175bd2f3156d1
Windows i386	Installer	<a href="#">SQLiteStudio-3.4.4-windows-installer.exe</a>	39aa488e76fc030a0e34fe797ff7e4ce856a1be31d8881d61dae72b6f3e36da0
MacOS X x64	Portable	<a href="#">sqlitestudio-3.4.4.dmg</a>	bd1bf5cd0e442b867ef9417e6c849d7b9f4d38f4305804c4b9d58d905092d8ef
MacOS X x64	Installer	<a href="#">SQLiteStudio-3.4.4-osx_installer.dmg</a>	2d3fe97c332c7b314fad180d8129c0eb6a49ada66f594f3a8e244403bdafa850

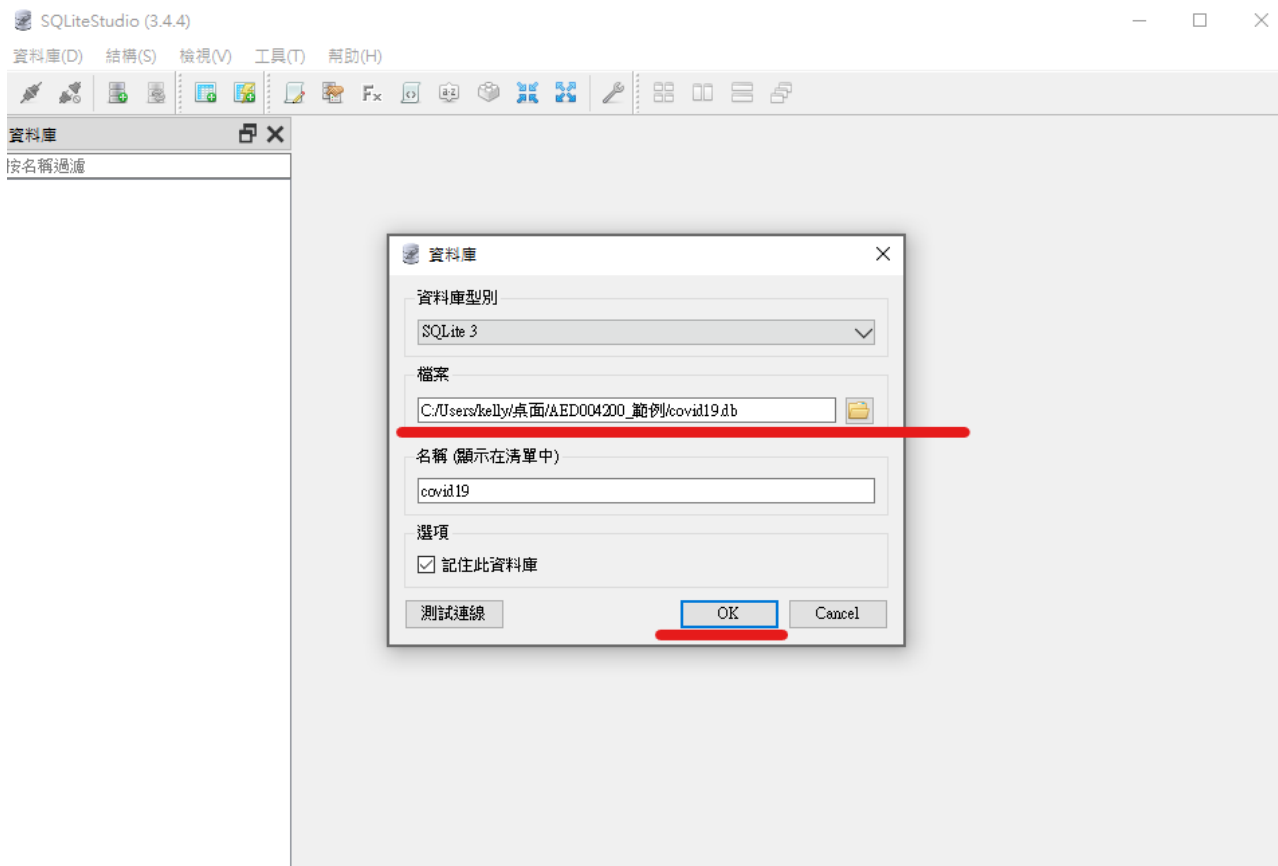


點擊打開

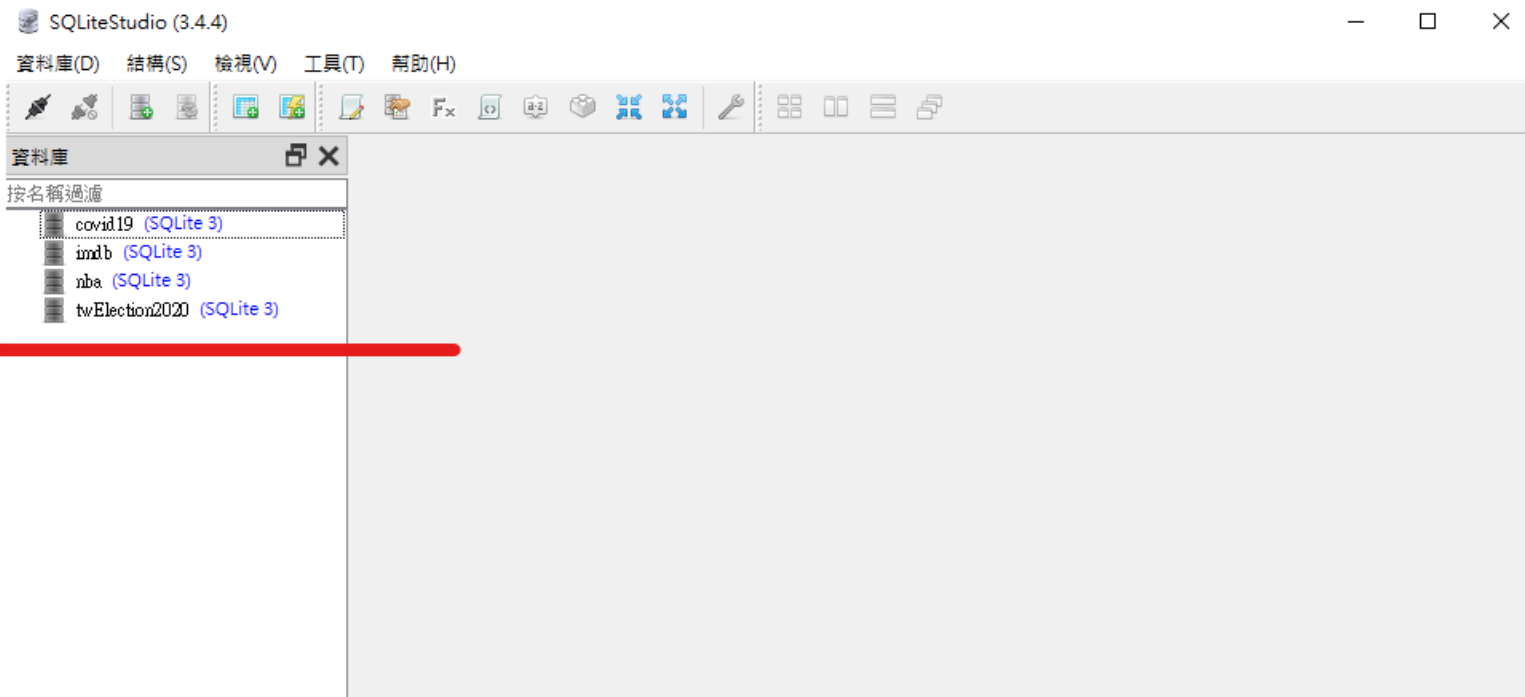


# 添加資料庫

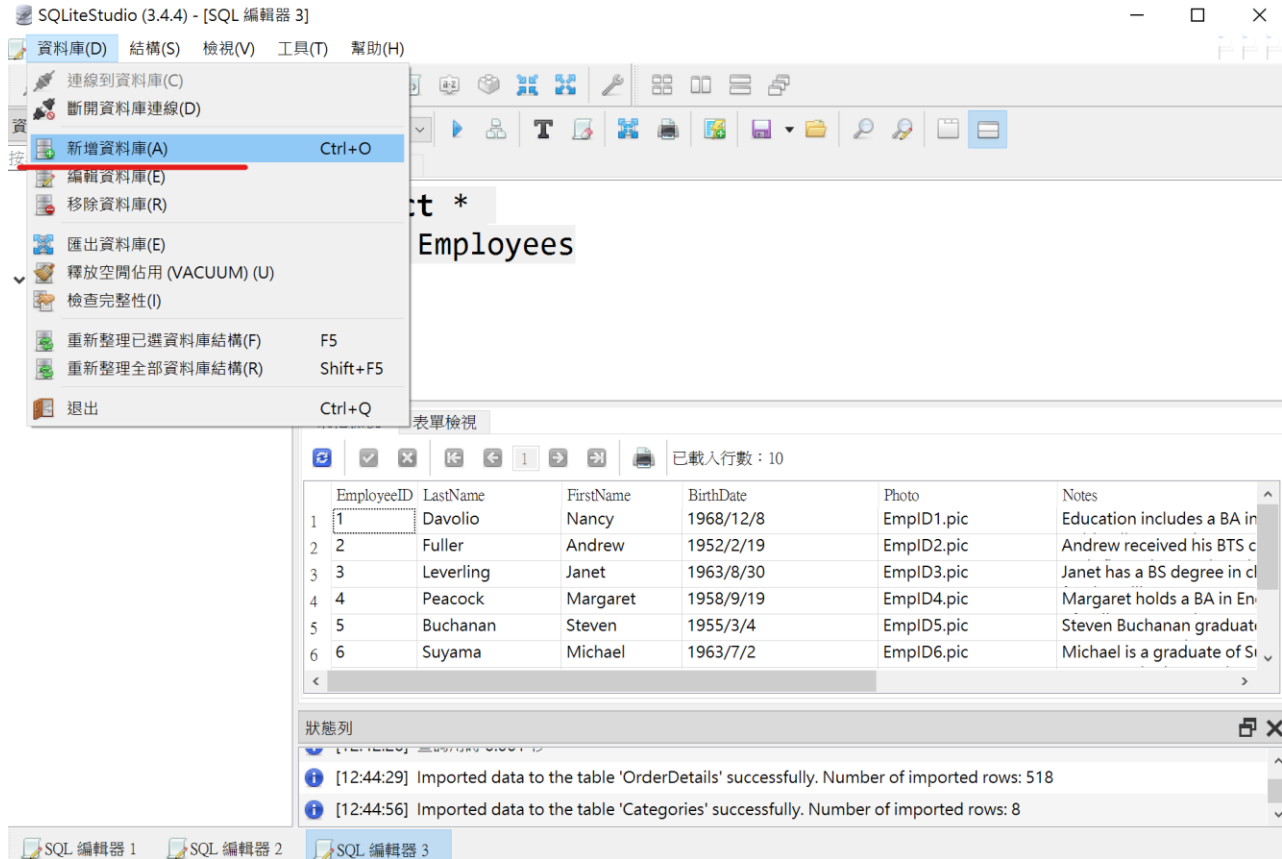




把四個資料庫添加上來

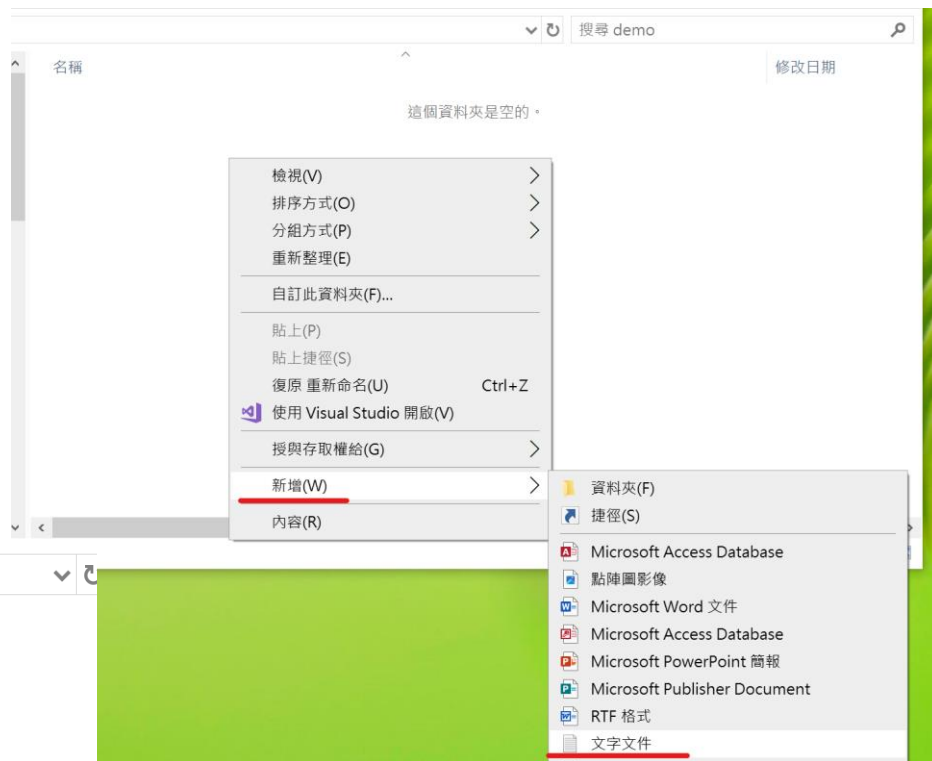


建自己的一個資料庫

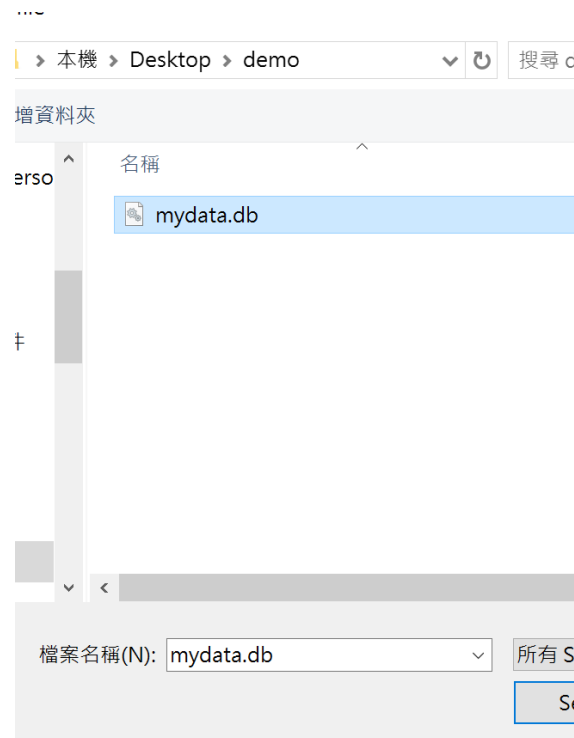
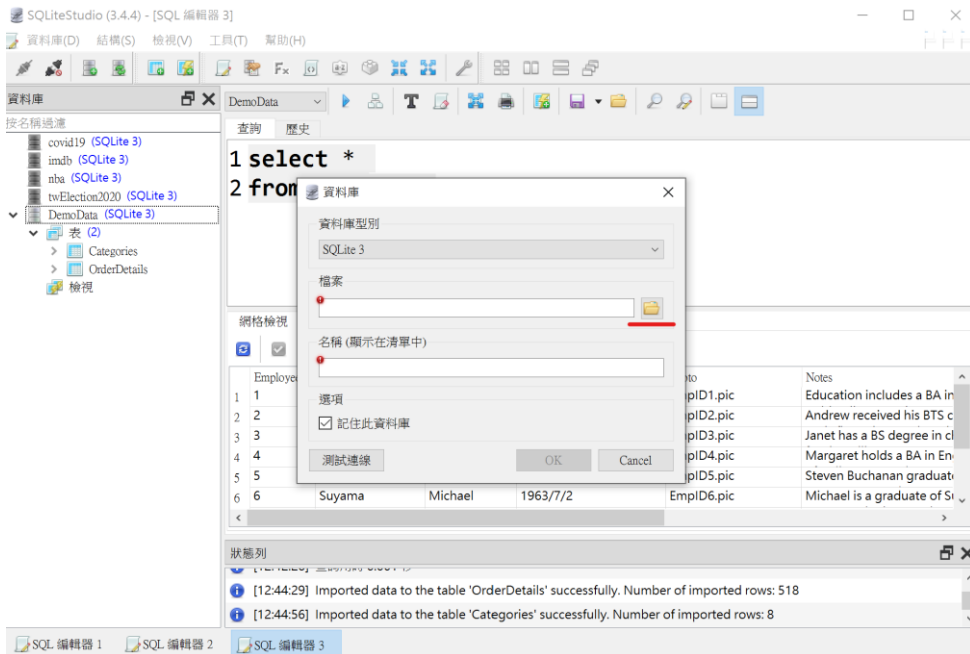




找個地方先新增一個檔案，副檔名就是.db  
這樣我們就給我們的資料庫一個儲存的地方了



## 選擇剛剛建立的db檔案



資料庫建立完成

資料庫

資料庫型別

SQLite 3

檔案

C:/Users/kelly/桌面/demo/mydata.db

名稱 (顯示在清單中)

mydata

選項

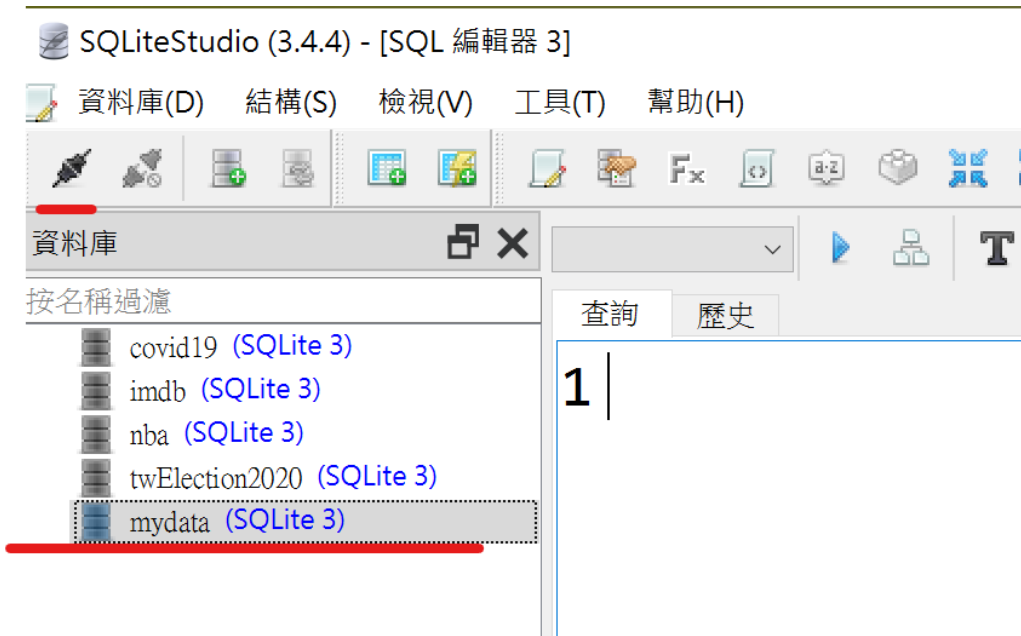
☒ 記住此資料庫

測試連線

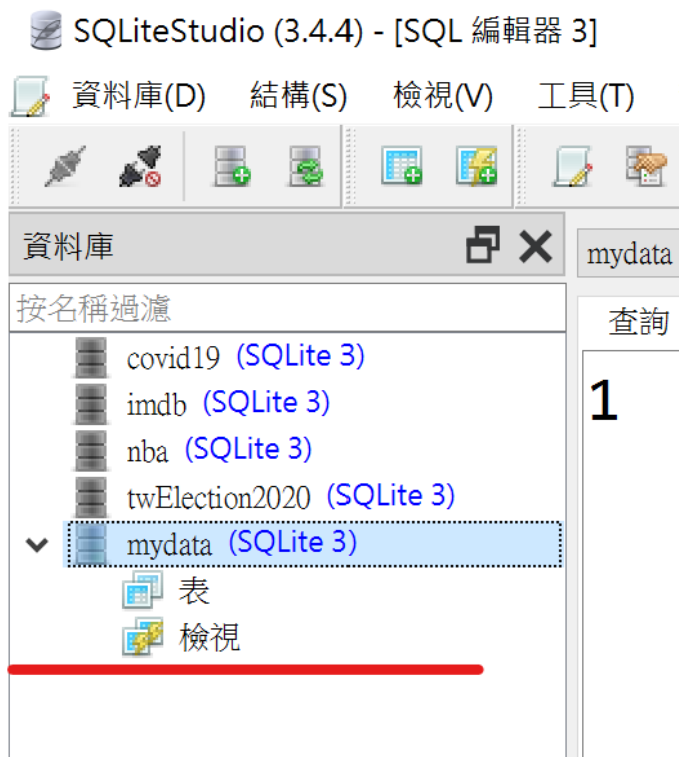
OK

Cancel

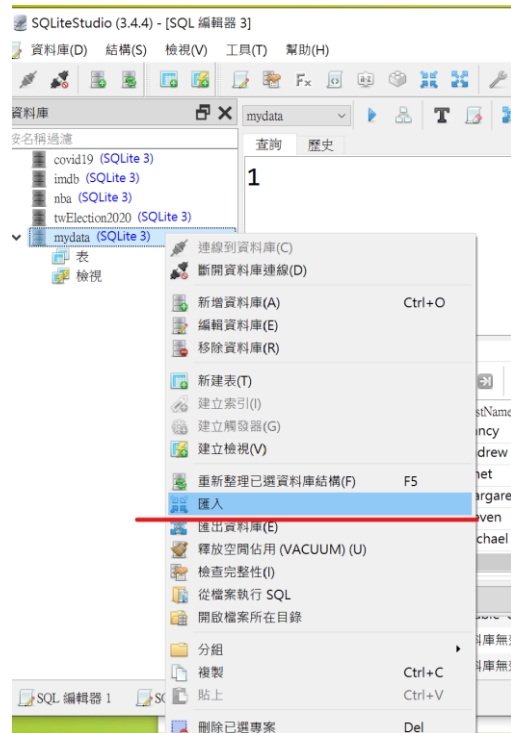
先連上這個資料庫



這樣代表連接上了



開始匯入資料

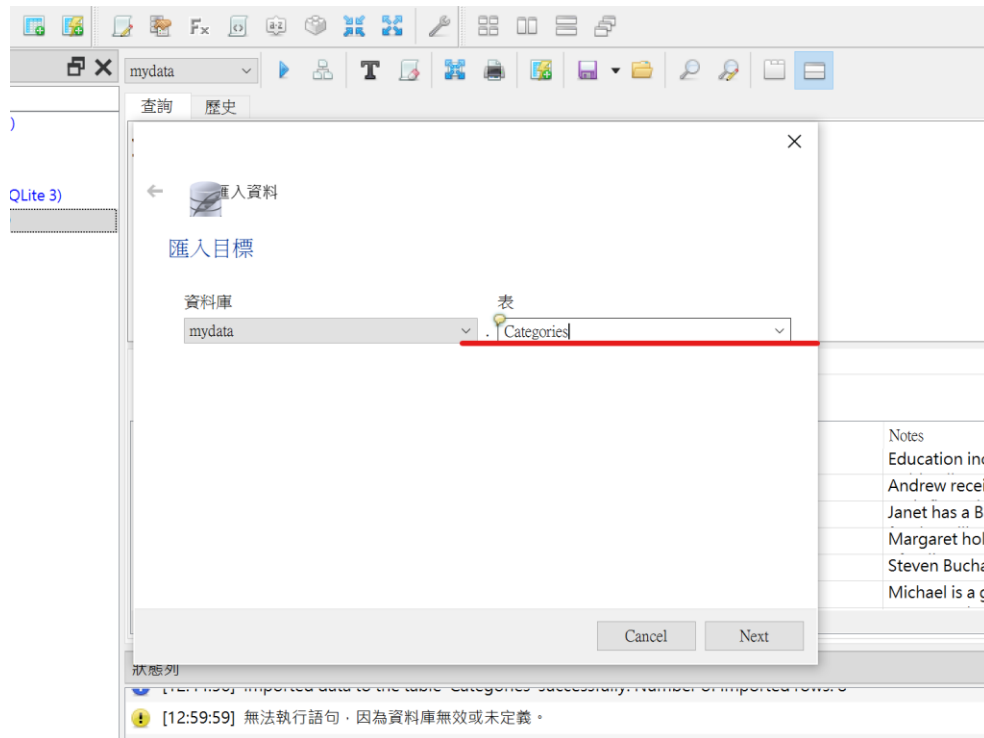


填一下table的名字

> CSV

名稱

Categories.csv  
Customers.csv  
Employees.csv  
OrderDetails.csv  
Orders.csv  
Products.csv  
Shippers.csv  
Suppliers.csv



## 倒進資料

查詢 歷史

← 匯入資料

匯入資料來源

資料來源型別

CSV

選項

輸入檔案: C:/Users/kelly/桌面/7.SQLite用法/CSV/Categories.csv

文字編碼: System

☐ 忽略錯誤

資料來源選項

☒ 第一行表示 CSV 列名

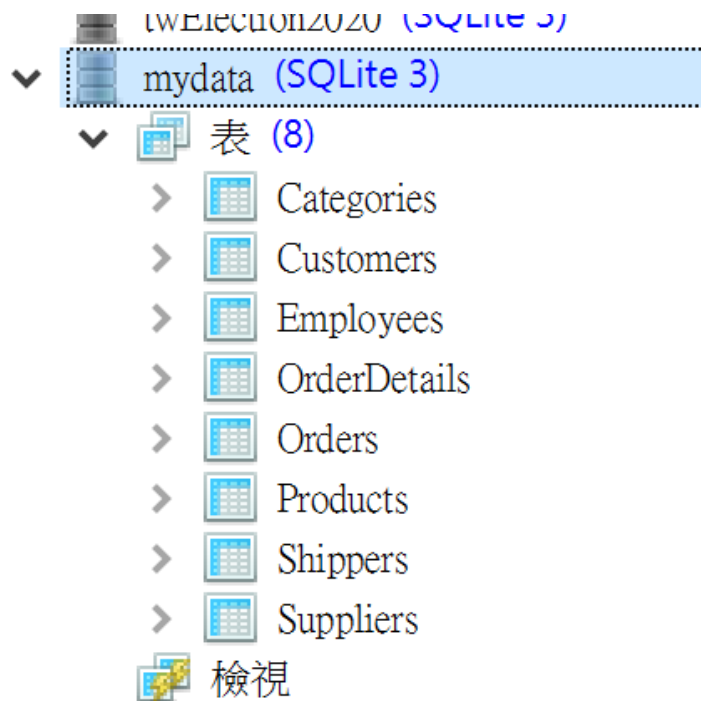
Column separator: ,(逗號)

Cancel Finish

狀態列

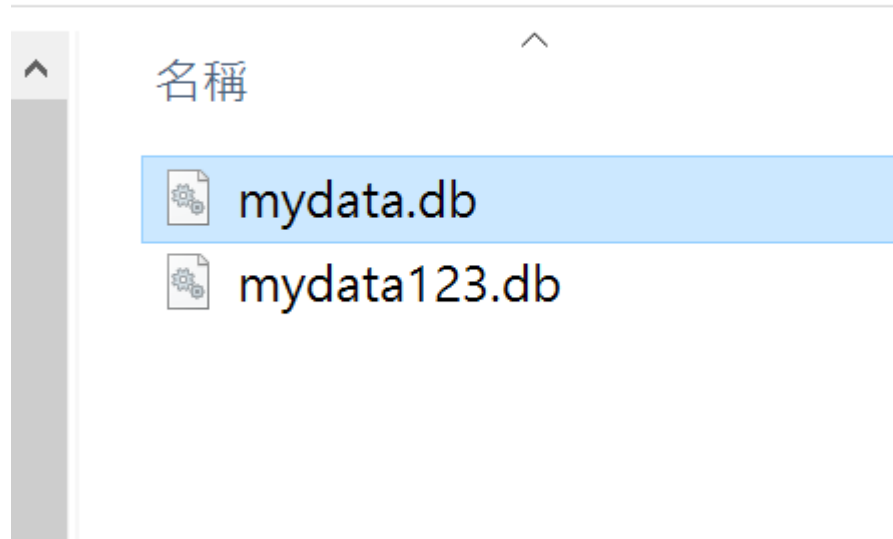
[1:15:00] Imported data to the table 'Categories' successfully. Number of imported rows: 0



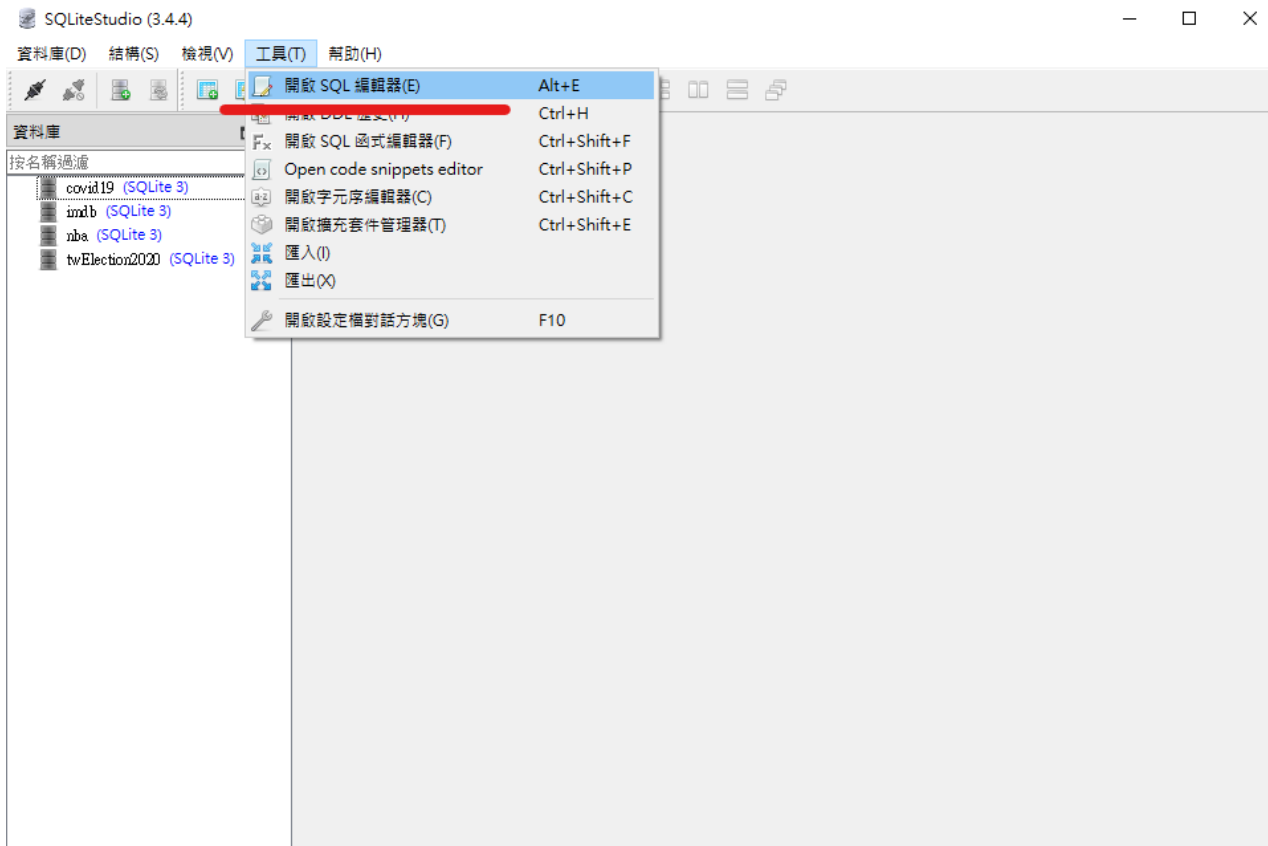


網格檢視			表
			 
	EmployeeID	L	
1	1	C	
2	2	F	

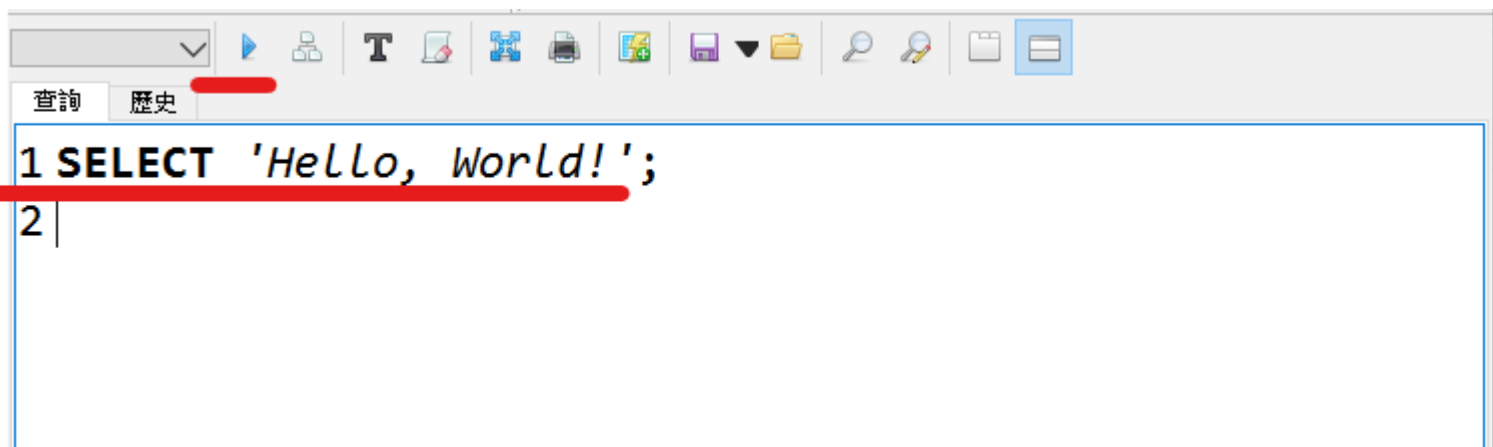
請你把db再複製一個  
恢復到studio裡面



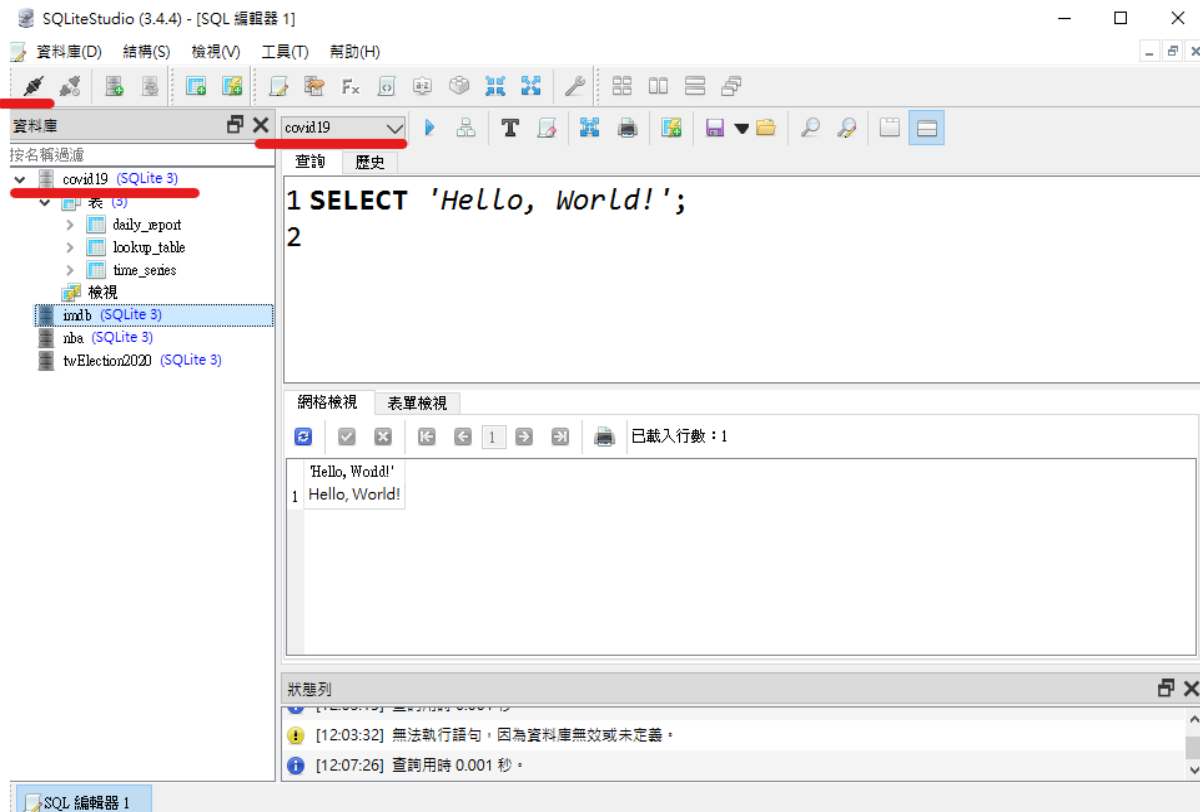
啟動SQL編輯器



打打碼試試看



前面結果不能跑，那是因為sql語句要配上資料庫才可以運作



進入ch2