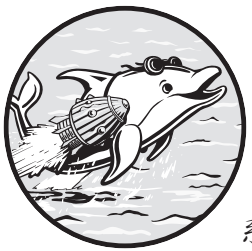
導讀



我的第一份軟體開發工作是從 1980 年代中期開始，這份工作帶領我進入關聯式資料庫管理系統（relational database management system，簡稱 RDBMS）的領域，這種系統是用於儲存和取出資料庫中的資料。其實這個概念在 1970 年代左右就已經存在，源自於電腦科學家 E.F. Codd 在他發表的著名論文中導入的關聯式模型。關聯式（relational）這個術語，是指資料其實是儲存在由資料欄和資料列組成的儲存格裡，否則就稱為資料表（table）。

我剛進入業界工作時，商業資料庫系統還不是那麼普及。事實上，那時我根本不知道是不是有其他人也在用資料庫系統。當年使用的關聯式資料庫管理系統一點也不完美，不僅沒有搭載圖形化介面，當然也沒有支援命令列介面，還會因為不明原因定期當機。那個年代還沒發明全球資訊網（World Wide Web），所以我無法求助任何網站，在別

無選擇的情況下，只能重啟系統，希望出現最好的結果。

不過，這在當年這可是相當酷的概念。我能根據想要儲存的資訊性質，將大量的資料儲存在我建立的資料表。定義資料表的資料欄，將檔案裡的資料載入到資料表裡，然後使用結構化查詢語言查詢資料；結構化查詢語言（Structured Query Language，簡稱 SQL）是用於跟資料庫互動的程式語言，允許使用者在彈指之間新增、更改和刪除多個資料列的資料。利用這項技術，我能管理整個公司的資料！

今日，關聯式資料庫管理系統無所不在，而且謝天謝地的是，比我在 80 年代用的破舊系統還要穩定而且先進，SQL 語言本身的能力也大幅提升。本書焦點會放在 MySQL，這項開放原始碼資料庫技術自 1995 年問世以來，就已經成為全世界最熱門的關聯式資料庫管理系統。

本書內容架構

本書會教讀者如何利用 MySQL 提供的 Community Server（也稱為社群版）來學習使用 MySQL，不僅免費使用而且具有多數人需要的功能。MySQL 也有提供其他付費版，包含搭載其他功能和性能的企業版。MySQL 具有一套健全的功能和工具，所有版本都能在各種作業系統上執行，像是 Linux、Windows、macOS，甚至是雲端環境。

全書內容主軸為探索 MySQL 開發範疇裡最有用的部分，以及分享我過去多來在這個領域裡獲得的見解。內容涵蓋如何撰寫 SQL 陳述式，建立資料表、函式、觸發器和檢視表，以及確保資料的完整性，最後三個章節會透過專題實作，帶讀者了解如何在實務環境中使用 MySQL。

本書章節內容規劃為五大部分：

PART I：起步

第 1 章：安裝 MySQL 與工具

本章說明如何下載 MySQL，並且提供一些安裝訣竅，協助讀者在各個作業系統上安裝 MySQL。為了存取 MySQL 的資料，還要再安裝兩個工具：MySQL Workbench 和 MySQL 命令列客戶端。

第 2 章：建立資料庫和資料表

本章說明如何定義與建立資料庫和資料表，對資料表加入條件約束，強制規定資料表能接受的資料，以及了解索引如何加快資料檢索的速度。

PART II：從 MySQL 資料庫選取資料

第 3 章：SQL 入門簡介

本章內容涵蓋如何對資料庫的資料表進行查詢，以選取想要顯示的資訊；排序查詢結果、對 SQL 程式碼加入註解以及處理空值。

第 4 章：MySQL 資料型態

本章討論的資料型態是用來定義資料表的資料欄，以及了解如何定義資料欄來保存字串、整數、日期等等資料。

第 5 章：合併資料庫的資料表

本章摘要幾個不同的做法，說明如何一次從兩個資料表選取資料，內容涵蓋合併查詢的主要類型，以及如何為資料欄和資料表建立別名。

第 6 章：對多個資料表執行複雜的合併查詢

本章會示範如何針對多個資料表進行合併查詢，以及如何運用暫存資料表、通用資料表運算式、衍生資料表和子查詢。

第 7 章：比較不同的查詢值

本章會帶讀者實作一遍，如何比較不同的 SQL 查詢值。例如，了解有哪些方法可以檢查某個值是否等於、大於另一個值，或是落在某個範圍值內。

第 8 章：呼叫 MySQL 內建函式

本章會解釋何謂函式、如何呼叫函式，以及介紹哪些函式最好用；學習用來處理數學、日期和字串的各項函式，以及運用彙總函式來處理一組值。

第 9 章：插入、更新和刪除資料

本章會說明如何對資料表新增、修改和移除資料。

PART III：資料庫物件

第 10 章：建立檢視表

本章會探索資料庫檢視表，這是根據我們建立的查詢來產生虛擬資料表。

第 11 章：自訂函式與程序

本章會介紹如何撰寫可重複使用的預存常用程序。

第 12 章：建立觸發器

本章會解釋如何撰寫資料庫觸發器，當資料發生異動時就會自動執行觸發器。

第 13 章：建立事件

本章會說明如何設定功能，然後根據已經定義好的排程執行。

PART IV：進階主題

第 14 章：實用的技巧與訣竅

本章會討論如何避免一些常見的問題、支援現有系統以及將檔案裡的資料載入到資料表裡。

第 15 章：從其他程式語言呼叫 MySQL

本章會探索如何從 PHP、Python 和 Java 程式呼叫 MySQL。

Part V：專題

第 16 章：建立天氣資料庫

本章會示範如何建立一套系統，利用 cron 和 Bash 這些技術，讓卡車運輸公司的資料庫載入天氣資料。

第 17 章：利用觸發器追蹤投票者資料異動

本章會指導讀者完成一套流程：建立選舉資料庫、使用資料庫觸發器來避免資料錯誤以及追蹤使用者對資料的異動情況。

第 18 章：利用檢視表保護薪資資料

本章會示範如何利用檢視表對特定使用者公開或隱藏敏感資料。

每一章都有準備練習題讓讀者「動手試試看」，幫助讀者精通文中解釋的觀念。

本書為誰而寫

所有對 MySQL 有興趣的讀者，不論是剛接觸 MySQL 和資料庫的初學者，還是想要再進修的開發人員，甚至是想從其他資料庫系統轉換到 MySQL 的資深軟體開發人員，都非常適合閱讀本書。

全書內容會聚焦在 MySQL 開發而非資料庫管理，從事 MySQL 資料庫管理人員（database administrator，簡稱 DBA）的讀者可能需要視個人需求參考其他書籍。雖然我偶而會提到一些我有興趣的 DBA 主題（像是對資料表設定權限），但整體內容不會深入探討伺服器建置、資料庫儲存容量、備份、復原或其他跟 DBA 相關的主要議題。

本書是為 MySQL 的初學者而設計，讀者若想在自己的 MySQL 環境下試做練習題，請參照第 1 章的內容，會帶領讀者從下載到安裝 MySQL。

比較 MySQL 和其他資料庫系統使用的 SQL

使用 MySQL 的重要關鍵之一是學習 SQL 語言。利用 SQL 語言，我們能儲存、修改和刪除資料庫裡的資料，建立和移除資料表，以及查詢資料等等。

目前採用 SQL 語言的關聯式資料庫管理系統，除了 MySQL，還包括 Oracle、Microsoft SQL Server 和 PostgreSQL。理論上，這些資料庫系統應該是使用經過標準化的 SQL 語言，也就是根據美國國家標準協會（American National Standards Institute，簡稱 ANSI）規範的 ANSI SQL。然而，實務上，這些資料庫系統之間還是存在著一些差異。

每一家資料庫系統都有搭配自己的 SQL 擴展，例如，Oracle 針對程序性語言提供的 SQL 擴展語言，稱為 Procedural Language/SQL（簡稱 PL/SQL）；Microsoft SQL Server 是搭配 Transact-SQL（簡稱 T-SQL），PostgreSQL 則是搭配 Procedural Language/PostgreSQL（簡稱 PL/pgSQL）；MySQL 提供的 SQL 擴展沒有很炫的名字，就只是叫 MySQL 預存程式語言（stored program language）。這些 SQL 擴展都各自使用不同的語法。

這幾家資料庫系統會各自開發出這些擴展，是因為 SQL 屬於非程序性語言（non-procedural language），意思是說 SQL 雖然非常適合從資料庫取出和儲存資料，但是它的設計不是針對程序性程式語言（像是 Java 或 Python），所以無法使用 `if...then` 邏輯或是 `while` 迴圈。為了增加這方面的功能性，各家資料庫搭配 SQL 擴展來支援程序性語言。

因此，雖然讀者從本書學到的 SQL 知識，大多都可以轉移到其他資料庫系統，但如果想在 MySQL 以外的資料庫系統上執行查詢，可能還是需要某些其他語法。

4

MYSQL 資料型態

本章會介紹 MySQL 能使用的所有資料型態。我們已經在前幾章看過 int 和 varchar，分別是用來處理整數和字元資料，但 MySQL 其實還有其他資料型態可用於儲存日期、時間，甚至是二進位資料。本章還會帶讀者探索，如何為資料欄選擇最佳的資料型態以及每一種資料型態的優缺點。

建立資料表的時候，我們需要根據各個資料欄要儲存的資料種類，為每一個資料欄定義資料型態。例如，假設某個資料欄是用於儲存名字，我們就不會用只接受數字的資料型態。此外，還要考慮資料欄必須容納的資料值範圍，如果資料欄需要儲存的內容是像 3.1415 這樣的值，則使用的資料型態就應該允許小數點之後存在四個位數的小數值。最後一點是，當我們在處理資料欄需要儲存的值，如果有超過一種以上資料型態可以使用，應該選擇儲存量最小的那一個型態。

此處假設我們想要建立一個名稱為 solar_eclipse 的資料表，用於儲存日蝕相關資料，包括發生日蝕的日期和時間，以及日蝕的種類和食分，原始資料請參見表 4-1。

表 4-1：日蝕觀測資料

發生日期	最長食甚時間	日蝕種類	食分
2022-04-30	20:42:36	Partial	0.640
2022-10-25	11:01:20	Partial	0.862
2023-04-20	04:17:56	Hybrid	1.013

為了在 MySQL 資料庫儲存這份資料，我們需要建立有四個資料欄的資料表：

```
create table solar_eclipse
(
    eclipse_date                date,
    time_of_greatest_eclipse    time,
    eclipse_type                varchar(10),
    magnitude                   decimal(4,3)
);
```

這個資料表的四個資料欄都各自定義了不同的資料型態。由於資料欄 eclipse_date 是儲存日期，所以使用的資料型態是 date。資料型態 time 是專門設計來儲存時間資料，所以會應用在資料欄 time_of_greatest_eclipse。

資料欄 eclipse_type 使用的資料型態是 varchar，因為我們需要儲存長度不定的字元資料，不過，我們不希望這些字元值太長，所以定義為 varchar(10)，設定最大字元數為 10。

資料欄 magnitude 使用的資料型態則是 decimal，指定儲存值共四位數，小數點後是三位數。

除了這幾個資料型態，接下來我們還要深入了解其他資料型態的用法，探討每個資料型態的適當使用時機。

資料型態：用於處理字串

字串（string）是由一組字元組成，包含字母、數字、空白字元（例如，空格和 tab 字元）和符號（像是標點符號）。對於只有數字的值，應該使用數值型資料型態，而非字串型資料型態。以「I love MySQL

8.0!」這個值為例，應該使用字串型資料型態，但像「8.0」這個值則要使用數值型資料型態。

本節接著就來探討 MySQL 提供的字串型資料型態。

char

資料型態 char 是用於儲存長度固定的字串，也就是用在確實知道字串具有多少字元數的情況。以下方程式碼為例，假設現在有一個資料表 country_code 要定義一個資料欄，用於儲存三個字母的國碼（例如，USA、GBR 和 JPN），可以使用 char(3) 來定義資料型態：

```
create table country_code
(
    country_code    char(3)
);
```

將資料欄的資料型態定義為 char，要在小括號內指定字串長度。如果省略小括號，資料型態 char 的預設字串長度是 1 個字元，不過，在只需要 1 個字元的情況，以 char(1) 指定字串長度，還是比只用 char 來得更清楚。

資料值的字串長度不能超過小括號內定義的長度，讀者如果嘗試在資料欄 country_code 插入 JAPAN，MySQL 會拒絕這個值，因為定義資料欄的時候已經指定最多只能儲存 3 個字元。不過，MySQL 允許插入少於三個字元的字串，像是 JP；處理方式只是在 JP 結尾加上一個空白，然後將這個值儲存在資料欄。

資料型態 char 可以定義的最大長度是 255 個字元。如果定義資料欄時，試圖將資料型態指定為 char(256)，就會得到錯誤訊息，因為超出 char 可以指定的範圍。

varchar

varchar 是我們之前就已經看過的資料型態，用於儲存可變長度的字串或是指定最大字元數的字串。當我們必須儲存字串卻又不確定字串究竟多長的時候，這個資料型態就能派上用場。舉例說明，假設我們要建立一個資料表 interesting_people，並且定義一個資料欄 interesting_name，用來儲存各種名字，這個資料欄必須能滿足短的名字（像是 Jet Li），也要能容納長的名字（像是 Hubert Blaine Wolfe schlegelsteinhausenbergerdorff）：

```
create table interesting_people
(
    interesting_name    varchar(100)
);
```

上方程式碼在小括號裡，定義資料欄 interesting_name 的字串長度，限制為 100 個字元，預期資料庫內不會出現任何人的名字會超過 100 字元。

varchar 可以接受的最大字元數，取決於 MySQL 的配置，讀者可以請資料庫管理人員（database administrator，簡稱 DBA）協助設定；或是使用快速的訣竅來決定最大字元數，就是在撰寫陳述式 create table 的時候，為建立的資料欄定義資料型態 varchar，然後指定長得離譜的最大字元值（如下所示）：

```
create table test_varchar_size
(
    huge_column varchar(999999999)
);
```

這個陳述式 create table 會執行失敗，出現像以下的錯誤訊息：

```
Error Code: 1074. Column length too big for column 'huge_column'
(max = 16383);
use BLOB or TEXT instead
```

從上方的錯誤訊息得知，無法建立資料表是因為 varchar 定義的字元數太大，在這個環境下，varchar 可以接受的最大字元數是 16,383 或是指定為 varchar(16383)。

資料型態 varchar 主要用於儲存長度較短的字串，在儲存資料超過 5,000 字元的情況，本書會建議改用資料型態 text（隨後就會介紹這個資料型態）。

enum

資料型態 enum 是 enumeration（列舉）的簡寫，作用是讓我們建立清單，列出允許資料欄接受的字串值。以下舉例說明如何建立資料表 student，具有資料欄 student_class，而且這個資料欄只接受以下其中一個值——Freshman、Sophomore、Junior 或 Senior：

```
create table student
    (
    student_id     int,
    student_class  enum('Freshman','Sophomore','Junior','Senior')
    );
```

如果我們想要加入資料欄的值不是上方清單接受的值，MySQL 就會拒絕加入。即使是清單可以接受的值，資料欄 student_class 也只能加入其中一個值，意思是說學生不能同時兼具大一（freshman）和大二（sophomore）的身分。

set

資料型態 set 跟 enum 類似，但 set 允許我們能一次選擇多個值。下方陳述式 create table 是建立資料表 interpreter，具有資料欄 language_spoken，目的是定義一份語言清單：

```
create table interpreter
    (
    interpreter_id     int,
    language_spoken    set('English','German','French','Spanish')
    );
```

因為某個人可能會說這些語言裡的一個或多個，所以此處使用資料型態 set 就能允許我們將 set 清單裡的任何一個或所有語言，加到資料欄 language_spoken。然而，如果試圖將清單以外的任何值加到資料欄，MySQL 就會拒絕加入。

tinytext、text、mediumtext 和 longtext

MySQL 包含四種文字型資料型態，用於儲存可變長度的字串：

tinytext	最長儲存 255 個字元
text	最長儲存 65,535 字元，約為 64KB
mediumtext	最長儲存 16,777,215 字元，約為 16MB
longtext	最長儲存 4,294,967,295 字元，約為 4GB

下方陳述式 create table 是建立資料表 book，具有四個資料欄。最後三個資料欄：author_bio、book_proposal 和 entire_book 都分別用了不同大小的文字型資料型態：

```
create table book
    (
    book_id             int,
    author_bio          tinytext,
    book_proposal       text,
    entire_book         mediumtext
    );
```

資料欄 author_bio 是使用資料型態 tinytext，因為預期所有作者簡介的長度都不會超過 255 字元，這也是強迫使用者確定個人簡介一定要少於 255 字元。資料欄 book_proposal 選擇使用資料型態 text，是因為不希望任何一本書的提案內容超過 64KB。最後是資料欄 entire_book，選擇使用資料型態 mediumtext，每一本書的內容大小限制在 16MB 以下。

字串格式

字串值必須以單引號或雙引號括起來，以下查詢是使用單引號包圍在字串 Town Supply 前後：

```
select  *
from    store
where   store_name = 'Town Supply';
```

下方這個查詢則是使用雙引號：

```
select  *
from    store
where   store_name = "Town Supply";
```

這兩種查詢格式會回傳相同的結果，但如果跟含有特殊字元的字串比較，像是撇號（'）、引號或 tab 字元，情況就會變得更妙。例如，「Town Supply」這個字串使用單引號可以正常運作，但如果換成字串「Bill's Supply」使用單引號，

```
select  *
from    store
where   store_name = 'Bill's Supply';
```

就會產生以下錯誤訊息：

```
Error Code: 1064. 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 's Supply'' at line 1
```

MySQL 感到困惑的地方是因為字串開頭和結尾的單引號，跟 Bill's 的撇號（'）是相同的字元，所以 MySQL 搞不清楚此處的撇號究竟是要當成字串結尾，還是字串的一部分。

要解決這個問題，可以改用雙引號將字串括起來，取代原本使用的單引號：

```
select  *
from    store
where   store_name = "Bill's Supply";
```

MySQL 現在知道撇號是字串的一部分。

另外一種修正錯誤的做法是以單引號將字串括起來，然後以跳脫字元的方式處理撇號：

```
select  *
from    store
where   store_name = 'Bill\'s Supply';
```

上方程式碼中的反斜線字元（\）是跳脫字元，會建立跳脫序列（escape sequence），告訴 MySQL 下一個字元屬於字串的一部分。其他能使用的跳脫序列如下：

\"　　雙引號

\n　　換行

\r　　return 字元

\t　　tab 字元

\\　　反斜線

使用跳脫序列就可以將特殊字元加入到字串裡，像以下程式碼這樣，使用雙引號將暱稱 Kitty 括起來：

```
select  *
from    accountant
where   accountant_name = "Kathy \"Kitty\" McGillicuddy";
```

在這個例子裡，我們也可以用單引號將字串包起來，這樣就不必用跳脫字元來處理雙引號：

```
select  *
from    accountant
where   accountant_name = 'Kathy "Kitty" McGillicuddy';
```

不管用哪種方法，兩者回傳的結果都是 Kathy "Kitty" McGillicuddy。

資料型態：用於處理二進位

針對人類無法閱讀的格式，MySQL 有提供資料型態來儲存二進位資料（binary data），或是以位元組為單位的原始資料。

tinyblob, blob, mediumblob, and longblob

大型二進位物件（binary large object，簡稱 BLOB）屬於可變長度的字串，以位元組為單位。BLOB 物件可用於儲存二進位資料，例如，圖像、PDF 檔和影片。BLOB 資料型態和文字型資料型態的大小一樣，只不過 tinytext 最大可儲存 255 字元，tinyblob 則是最大儲存 255 位元組。

tinyblob	最大儲存 255 位元組
blob	最大儲存 65,535 位元組，約為 64KB
mediumblob	最大儲存 16,777,215 位元組，約為 16MB
longblob	最大儲存 4,294,967,295 位元組，約為 4GB

binary

資料型態 binary 是用於儲存長度固定的二進位資料，類似資料型態 char，只不過不是用於儲存字元字串，而是二進位資料的字串。在小括號內指定字串的位元組大小，如下所示：

```
create table encryption
    (
    key_id          int,
    encryption_key  binary(50)
    );
```

在上方程式碼中，資料表 encryption 的資料欄 encryption_key 是將字串的位元組大小設定為最大 50 位元組。

varbinary

資料型態 varbinary 是用於儲存可變長度的二進位資料，在小括號內指定字串的位元組大小的最大值：

```
create table signature
    (
    signature_id    int,
    signature       varbinary(400)
    );
```

此處的程式碼是為相同名稱的資料表建立資料欄 signature，設定最大值為 400 位元組。

bit

bit 的作用是儲存位元值，算是比較少用的資料型態，可以指定想要儲存多少位元，最大可儲存 64 位元。例如，bit(15) 的定義是允許我們最多儲存 15 位元。

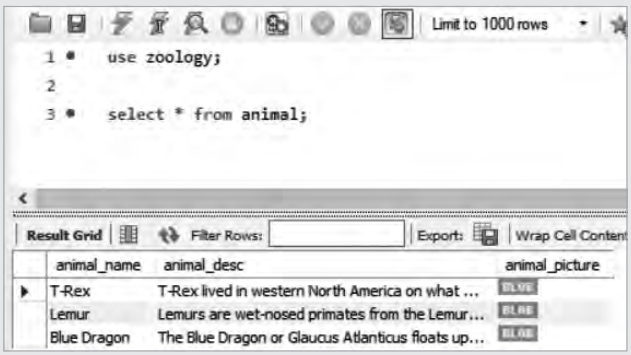
字串：字元 vs. 位元組

字元字串（character string），通常只會稱為字串（string），是一組人類可以閱讀的字元；另一方面，位元組字串（byte string）是由位元組組成的字串，是人類無法閱讀的字串。

以下程式碼是建立資料表 animal，其中資料欄 animal_desc 的資料型態定義為 tinytext，資料欄 animal_picture 的資料型態則定義為 mediumblob：

```
create table animal
    (
    animal_name     varchar(20),
    animal_desc     tinytext,
    animal_picture  mediumblob
    );
```

此處是我們利用 MySQL Workbench 查詢資料表所得到的結果：



資料欄 animal_desc 的值是人類可以閱讀的內容，但 MySQL Workbench 顯示 animal_picture 的內容為 BLOB，因為這個資料欄的值是設定為位元組字串格式。

資料型態：用於處理數值

MySQL 有提供資料型態來儲存不同大小的數字，至於要使用哪種數值型態，還是要取決於我們想要儲存的數字是否含有小數點。

tinyint、smallint、mediumint、int 和 bigint

整數（integer）是指完全沒有分數或小數部分的數字，整數值可以為正數、負數或零。MySQL 包含以下這些整數資料型態：

tinyint 允許儲存的整數值範圍：–128 到 127 或最大儲存 1 位元組

smallint 允許儲存的整數值範圍：–32,768 到 32,767 或最大儲存 2 位元組

mediumint 允許儲存的整數值範圍：–8,388,608 到 8,388,607 或最大儲存 3 位元組

int 允許儲存的整數值範圍：–2,147,483,648 到 2,147,483,647 或最大儲存 4 位元組

bigint 允許儲存的整數值範圍：–9,223,372,036,854,775,808 到 9,223,372,036,854,775,807 或最大儲存 8 位元組

如何知道我們的資料適用於哪一種整數資料型態？請看以下範例清單 4-1 建立的資料表 planet_stat。

範例清單 4-1：根據各行星的統計資料建立資料表

```
create table planet_stat
(
    planet            varchar(20),
    miles_from_earth  bigint,
    diameter_km       mediumint
);
```

上方程式碼建立的資料表包含跟行星有關的統計資料，其中 varchar(20) 是用於儲存星球的名稱，資料型態 bigint 是用於儲存行星距離地球多少英里，mediumint 則是儲存行星的直徑（以公里為單位）。

仔細看看以下結果，會發現 Neptune（海王星）距離地球 2,703,959,966 英里。在這個範例中，資料型態 bigint 是最適合資料欄的選擇，因為 int 允許的範圍對這個值來說不夠大。

```
planet   miles_from_earth  diameter_km
-------  ----------------  -----------
Mars              48678219         6792
Jupiter          390674712       142984
Saturn           792248279       120536
Uranus          1692662533        51118
Neptune         2703959966        49528
```

在考慮到 int 儲存量要 4 位元組，bigint 則需要 8 位元組的情況下，如果 int 對資料欄來說就已經夠大，那麼使用超出需求的 bigint，意味著會占用更多的硬碟空間。在資料量少的資料表裡，若使用 smallint 和 mediumint 就已經夠用的情況，使用 int 不會引起任何問題，但如果資料表擁有 2000 萬筆資料列，就值得花時間為資料欄設定正確的資料型態大小，畢竟額外占用的位元組累積起來會相當可觀。

此處提一個可以有效利用儲存空間的技巧，就是將整數資料型態定義為 unsigned。整數資料型態的預設用法是允許我們儲存包含負數和正數的整數。讀者如果不需要用到任何負數，使用 unsigned 能避免使用負值，進而增加正數數字的數量。例如，資料型態 tinyint 提供的一般範圍是落在 –128 和 127 之間，但如果指定 unsigned，範圍值就會變成 0 到 255。

若將 smallint 定義為 unsigned，範圍值就會變成 0 到 65,535。經過指定後，資料型態 mediumint 的範圍值會落在 0 到 16,777,215，資料型態 mediumint 的範圍值則會落在 0 到 4,294,967,295。

在範例清單 4-1 中，資料欄 miles_from_earth 的資料型態是定義為 bigint，但如果利用 unsigned 這項技巧來提高範圍值的上限，改用資料型態 int 就足以容納這些資料值。由於所有行星跟地球之間的距離不會小於零，表示這個資料欄永遠不需要儲存負數，所以能放心使用 unsigned：

```
create table planet_stat
(
    planet            varchar(20),
    miles_from_earth  int unsigned, -- 此處現在改用int unsigned，而非bigint
    diameter_km       mediumint
);
```

因此，將資料欄定義為 unsigned 之後，就可以用更小的資料型態 int 來節省儲存空間。

布林值

布林值只有兩種狀態：true（真）或 false（假）、開啟或關閉、1 或 0。從技術層面來說，MySQL 沒有提供可以儲存布林值的資料型態，實際上是儲存為 tinyint(1)，建立資料欄儲存布林值時，會代換成 bool；將資料欄定義為 bool 時，背後的運作原理是建立資料欄，然後指定為 tinyint(1)。

下方建立的資料表 food，擁有兩個布林型態的資料欄：organic_flag 和 gluten_free_flag，告訴我們某一個食物是有機或無麩質：

```
create table food
(
    food              varchar(30),
    organic_flag      bool,
    gluten_free_flag  bool
);
```

若資料欄含有布林值，實務上常見的做法會將結尾詞 _flag 加到資料欄名稱後面，例如，organic_flag；因為這個值會設定為 true 或 false，所以能分別與提升或降低的旗標進行比較。

使用命令：describe 或 desc，可以檢視資料表結構。圖 4-1 是在 MySQL Workbench，顯示命令 desc food 的執行結果。

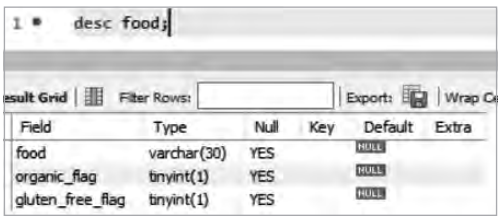


圖 4-1：在 MySQL Workbench 呈現資料表 food

仔細觀察可以發現，建立資料欄 organic_flag 和 gluten_free_flag 的時候，雖然是將資料型態代換成 bool，但實際上用來建立資料欄的資料型態是 tinyint(1)。

資料型態：用於處理小數

針對含有小數點的數字，MySQL 提供的資料型態有：decimal、float 和 double；其中 decimal 是用於儲存精確值，float 和 double 則是用於儲存近似值。基於這點，如果 decimal、float 或 double 都同樣適合我們準備要儲存的值，本書會建議採用資料型態 decimal。

decimal

資料型態 decimal 允許我們定義精確位數（precision）和小數位數（scale）；*精確位數*（precision）是指可以儲存的總位數，*小數位數*（scale）則是指小數點後可以儲存的位數。資料型態 decimal 常用於表示小數點後帶有二位數的貨幣值。

舉個例子，假設我們定義資料欄 price 為 decimal(5,2)，意思是說我們可以儲存的值會介於 –999.99 到 999.99 之間；其中精確位數為 5，表示總共可以儲存五個位數，小數位數為 2，表示小數點後可以儲存兩位數。

以下指定方式代表的意義跟資料型態 decimal 一樣：numeric(5,2)、dec(5,2) 和 fixed(5,2)，而且，所有表示方法的效果也都一樣，都會建立資料型態 decimal(5,2)。

float

資料型態 float 是用於儲存數值資料，帶有浮點小數。不像資料型態 decimal 已經定義小數位數，浮點數的小數點不一定會在相同的位置，也就是說小數點會在數字之間浮動。因此，資料型態 float 能表示這些數字：1.234、12.34 或 123.4。

double

資料型態 double 是 double precision 的簡稱，其所儲存的數字也能帶有未定義的小數位數（意即小數點會存在於數字間的某處）。資料型態 double 類似 float，差別在於 double 可以儲存更準確的數字。MySQL 儲存資料型態 float 時會使用 4 位元組，儲存 double 則要用掉 8 位元組。針對數字較多的浮點數，請使用資料型態 double。

資料型態：用於處理日期和時間

MySQL 針對日期和時間提供的資料型態有：date、time、datetime、timestamp 和 year。

date

資料型態 date 儲存日期的格式為 YYYY-MM-DD（分別為年、月和日）。

time

資料型態 time 儲存時間的格式為 hh:mm:ss（分別表示小時、分鐘和秒）。

datetime

資料型態 datetime 是將日期和時間儲存在同一個值裡，儲存格式為 YYYY-MM-DD hh:mm:ss。

timestamp

資料型態 timestamp 也是以相同的格式將日期和時間儲存在同一個值裡：YYYY-MM-DD hh:mm:ss，只不過 timestamp 是儲存當前的日期和時間，datetime 則是專為儲存其他日期和時間值而設計。

timestamp 能接受的範圍值比較小，必須是介於 1970 年到 2038 年之間的日期；資料型態 datetime 能接受的日期範圍較大，可以從 1000 年到 9999 年。唯有當我們想要用戳記標明目前的日期和時間值，才應該使用 timestamp，例如，儲存資料列更新的日期和時間。

year

資料型態 year 儲存年份的格式為 YYYY。

資料型態：用於處理 JSON 格式

JavaScript 物件表示法（JavaScript Object Notation，簡稱 JSON）是時下熱門的格式，用於在電腦之間傳送資料。MySQL 提供資料型態 json，讓我們能在資料庫裡儲存和取出全部的 JSON 文件。MySQL 會先檢查 JSON 文件內是否包含有效的 JSON 格式，若格式有效才允許 JSON 文件儲存到資料欄 json。

簡單的 JSON 文件如下所示：

```
{
   "department":"Marketing",
   "city":"Detroit",
   "managers":[
      {
         "name":"Tom McBride",
         "age":29
      },
      {
         "name":"Jill Hatfield",
         "age":25
      }
   ]
}
```

JSON 文件含有成對的鍵值和資料值，在上方範例中，鍵值是 department，資料值則是 Marketing。這些鍵值和資料值並不是對應資料表的資料列和資料欄，而是將整個 JSON 文件儲存在指定為資料型態 json 的資料欄裡，日後再使用 MySQL 的查詢指令，從 JSON 文件中取出屬性。

資料型態：用於處理空間資料

MySQL 有提供資料型態來表示地理位置資料或稱為 *geodata*（地理資料）。這種類型的資料能協助我們回答這些問題，像是「我在哪個城市？」或「距離我的位置五英里內，有多少家中式餐廳？」

geometry	用於儲存任何地理類型的位置值，包括 point（點）、linestring（線）和 polygon（多邊形）。
point	用於表示位置，具有特定緯度和經度，例如，個人目前的位置。
linestring	用於表示點和各點之間的曲線，例如，高速公路的位置。
polygon	用於表示邊界，例如，圍繞國家或城市的區域界線。
multipoint	用於儲存一組沒有順序的 point 資料型態的集合。

multilinestring	用於儲存一組 linestring 資料型態的集合。
emultipolygon	用於儲存一組 polygon 資料型態的集合。
geometrycollection	用於儲存一組 geometry 資料型態的集合。

動手試試看

4-1. 建立資料庫 rapper，並且撰寫陳述式 create table 來建立資料表 album，資料表 album 具有五個資料欄：

- 資料欄 rapper_id 定義為資料型態 smallint，使用 unsigned。
- 資料欄 album_name 應該定義為可變長度字串，最大可保存 100 字元。
- explicit_lyrics_flag 會儲存布林值。
- 資料欄 album_revenue 會儲存貨幣金額，具有精確位數 12，小數位數為 2。
- 資料欄 album_content 要使用資料型態 longblob。

重點回顧與小結

讀者在本章已經探索可以使用的 MySQL 資料型態，並且了解各種資料型態的使用時機。下一章的學習主軸有：了解 MySQL 提供的不同類型的合併查詢方法，利用這些方法從多個資料表取出資料，再將查詢取得的資料顯示在單一資料表。