

Chapter 1

學習 SQL 的 事前準備

資料庫就是「資料的基地」，要在電腦建立資料庫，就要使用 SQL 這種程式語言。

讓我們先安裝「MySQL」這套免費的 DBMS，接著建立練習用的資料庫，再新增練習用的資料，完成練習 SQL 的事前準備。

Lesson

1-1

「資料庫」到底是什麼？

了解資料庫的基礎

資料庫就是記錄資料的檔案，這個檔案可供多位使用者共用，能讓多名使用者使用這個檔案的程式就是 DBMS。使用者可利用 SQL 語言撰寫 SQL 陳述式，再將這個陳述式傳遞給 DBMS，藉此使用資料庫。



在電腦使用的語言不是只有程式設計語言吧！

對啊，本書主題的「SQL」是一種將命令傳遞給 DBMS 的語言。



所謂的「使用資料庫」，到底都會做哪些事情呢？

會建立、搜尋與更新資料庫，本書要介紹的就是執行這些處理的 SQL 陳述式。

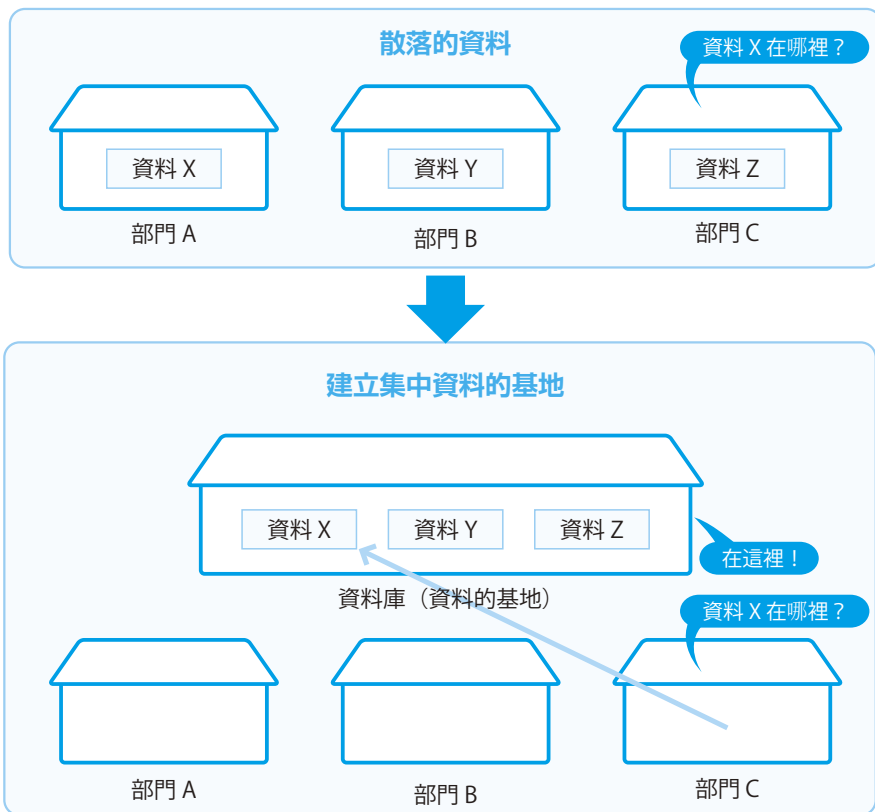


1-1-1 資料庫就是資料的基地

「資料庫 (data base)」就是「資料 (data)」的「基地 (base)」，據說這個字眼誕生於 1950 年代的美國國防部。如果資料分散在各地，那麼要執行某些動作時，就必須花很多時間收集資料。

為了節省這些時間，就要先收集資料，並且隨時更新資料的內容，以便在需要的時候可以立刻取出使用，就是所謂的資料的基地，也就是資料庫（圖 1-1-1）。

圖 1-1-1 將散落的資料集中在一處的位置就是資料庫



這個時代的資料庫就像是收集書面資料的建築物。就現在而言，儲存檔案資料的電腦已被當成資料庫使用。

這台電腦會向多位使用者提供資料，所以又被稱為「**伺服器** (server = 提供者)」，也因為是提供資料庫功能的伺服器，所以又稱為「**資料伺服器**」，使用伺服器的使用者電腦則稱為**用戶端** (client = 使用者)。

重要!

目前的資料庫都是向多位使用者提供資料的伺服器。

COLUMN


不要忘記加上分號

剛開始學習 SQL 時，很容易忘記在最後加上「;(分號)」。如果「按下 `Enter` 鍵，SQL 陳述式不執行的話」，請先冷靜地想想，是不是忘了加上分號，如果真的忘記加上分號，只需要輸入分號再按下 `Enter` 鍵，SQL 陳述式就會自動執行了。

5 試著輸入「\c」取消輸入

如果不小心輸入錯誤的命令，例如將「`SELECT 1 + 2;`」輸入成「`SLECT 1 +2;`」少輸入了一個「E」的命令就按下 `Enter` 鍵的話，會因為無法剖析「`SLECT`」這個命令而顯示「You have an error in your SQL syntax」這個「SQL 語法有誤」的錯誤訊息（圖 1-2-34）。

圖 1-2-34 執行了錯誤的 SQL 陳述式，就會顯示錯誤訊息

```
mysql> SLECT 1 + 2;   
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 'SLECT 1 + 2' atline 1  
mysql>
```

如果知道顯示錯誤訊息的原因，請輸入正確的 SQL 陳述式再重新執行一次，如果在按下 `Enter` 鍵之前就發現輸入的 SQL 陳述式有錯，也可以利用 `Backspace` 鍵刪除文字再重新輸入。

不過，有時候會想取消整個輸入的命令，此時只需要輸入「\c」再按下 `Enter` 鍵。這裡的「\」可讓 MySQL 命令列用戶端知道接下來的內容是具有特殊意義的文字，而不是 SQL 陳述式。

「c」是「cancel」的首字，代表取消輸入的 SQL 陳述式。輸入「\c」可阻止 MySQL 命令列用戶端將 SQL 陳述式傳遞給 MySQL 資料庫伺服器，此時命令提示字元就會轉換成等待命令輸入的「mysql>」。

圖 1-4-9 在 Department 表格新增資料

```
mysql> INSERT INTO Department VALUES ('D001', '總務部', 8); 
Query OK, 1 row affected (0.20 sec)

mysql> INSERT INTO Department VALUES ('D002', '人才開發部', 8); 
Query OK, 1 row affected (0.11 sec)

mysql> INSERT INTO Department VALUES ('D003', '系統開發部', 12); 
Query OK, 1 row affected (0.10 sec)
```

COLUMN

不要忘記輸入「結尾用」的單引號！

字串類型與日期類型的資料都需要以單引號括住，但一定要記得輸入最後「結尾用」的單引號。例如剛剛在 Department 表新增資料的 SQL 陳述句以單引號括住「系統開發部」這個字串時，若是將「'系統開發部'」設定成「'系統開發部」這種少了「結尾單引號」的內容，就會得到下列的結果（圖 1-4-10）。

圖 1-4-10 忘記輸入結尾的單引號就執行 SQL 陳述句的結果

```
mysql> INSERT INTO Department
-> VALUES ('D003', '系統開發部', 12); 
'>
```

SQL 陳述句會無法執行，也只會顯示「'>」這種命令提示字元，這代表，「'系統開發部,12」；的部分」因為少了結尾的單引號而被剖析成字串資料，此時用於結尾的分號也被剖析成字串的一部分，而不是命令的結尾，所以這個陳述句才無法執行。「'>」命令提示字元代表正在輸入字串的意思。此時只有輸入一個單引號結尾（命令提示字元會轉換成「->」），再輸入「\c」取消剛剛的輸入。

忘記輸入結尾的單引號算是常見的失誤，如果還沒養成輸入的習慣，有可能會覺得「明明輸入了分號，怎麼 SQL 陳述句不執行」。從命令提示字元轉換成「'>」這點可提醒我們忘記輸入結尾的單引號。

2-2-2 以 NULL / 非 NULL 的條件取得資料

代表欄位值為「空白」的 **NULL** 無法以「=」或「<>」進行比較，必須以「**IS NULL**（是 **NULL**）」與「**IS NOT NULL**（非 **NULL**）」進行比較。

SQL 的語法

從表格取得欄位名稱為 NULL 的資料

```
SELECT...FROM 表格名稱 WHERE 欄位名稱 IS NULL;
```

SQL 的語法

從表格取得欄位名稱不為 NULL 的資料

```
SELECT...FROM 表格名稱 WHERE 欄位名稱 IS NOT NULL;
```

範例 2-2-2

以 `dep_id` 欄位為 **NULL** 的條件，從 `Employee` 表格取得 `name` 與 `dep_id` 欄位的資料。

```
SELECT name, dep_id FROM Employee
WHERE dep_id IS NULL;
```

圖 2-2-3 範例 2-2-2 的執行結果

```
mysql> SELECT name, dep_id FROM Employee WHERE dep_id IS NULL;
+-----+-----+
| name   | dep_id |
+-----+-----+
| 山田太郎 | NULL   |
+-----+-----+
1 row in set (0.00 sec)
```

有誰沒有分配到部門？

原來是這個人！

3-1-3 從指定的起始位置依照指定的筆數取得資料

利用 **ORDER BY** 排序之後，有時會發現不一定需要所有的資料，例如只想知道「薪資前三名是誰」或「只想知道第四名與第五名是誰」，就是這類情況。

此時可使用「**LIMIT**」限制 **SELECT** 命令取得的資料筆數。**LIMIT** 的意思就是「限制」。

SQL 的語法

從起始位置開始，取得指定筆數的資料（只寫出 **LIMIT** 陳述句）

LIMIT 起始位置，列數

要請大家注意的是，取得資料之後，資料的開頭是從第 0 列開始，所以「若只想取得前三名的資料」，必須指定為「**LIMIT 0,3**」。這行命令的意思為「從開頭（第 0 列）取得三筆資料」的意思，若想「取得第 4 列與第 5 列資料」，可指定為「**LIMIT 3,2**」。如此一來，就是「從第 3 列取得 2 筆資料」的意思。由於開頭是第 0 列，所以實際上是「從第 4 列取得 2 筆資料」，因此結果就是「取得第 4 列與第 5 列」。

之所以將開頭設定為第 0 列，是因為「**LIMIT 起始位置，列數；**」的「起始位置」並非列編號，而是「距離資料開頭有幾筆」的意思，開頭的資料與開頭的距離是 0，所以開頭才會是第 0 列。雖然這有點難懂，不過目前就是這樣，請大家先記下來囉（圖 3-1-6）。

圖 3-1-6 「LIMIT 起始位置，列數；」的語法將開頭定為第 0 列

| | |
|-------|------------|
| 第 0 列 | 取得的第 1 筆資料 |
| 第 1 列 | 取得的第 2 筆資料 |
| 第 2 列 | 取得的第 3 筆資料 |
| 第 3 列 | 取得的第 4 筆資料 |
| 第 4 列 | 取得的第 5 筆資料 |
| 第 5 列 | 取得的第 6 筆資料 |
| 第 6 列 | 取得的第 7 筆資料 |

前三名的資料就是從第 0 列起算的三筆資料，所以要指定為 **LIMIT 0,3**

第四與第五名的資料就是從第 3 列起算的 2 筆資料，所以要指定為 **LIMIT 3,2**

BY dep_id」。用於群組化的欄位稱為「彙總鍵」，而這裡的「彙總」就是「群組化」的意思。

SQL 的語法以欄位名稱為彙總鍵，群組化資料（只寫出 **GROUP BY** 陳述句）**GROUP BY** 欄位名稱

使用 **GROUP BY** 群組化資料之後，能於 **SELECT** 命令後面指定的只有彙總函數或是接在 **GROUP BY** 後面的欄位名稱（彙總鍵）。其實仔細一想就不難明白，因為以 **GROUP BY** gender 將性別建立為群組，**SELECT** 的後面若是員工姓名的 name 欄位或 birthday 欄位這類與群組無關的資料，不是很奇怪的事嗎？

將性別建立為群組之後，只能於群組使用彙總函數或是取得群組化之後的欄位名稱。換言之，只能取得與群組有關的資料。「群組化之後，能接在 **SELECT** 命令之後的只有彙總函數或是接在 **GROUP BY** 的欄位名稱」這項 SQL 規則請先了解意思，再記在腦袋裡。

重要！

「群組化之後，能接在 **SELECT** 命令之後的只有彙總函數或是接在 **GROUP BY** 的欄位名稱。

範例 3-4-1

以 dep_id 欄位群組化 Employee 表格的資料，再取得計算每個群組的 dep_id 欄位以及計算 salary 欄位的平均值。

```
SELECT dep_id, AVG(salary) FROM Employee  
GROUP BY dep_id;
```


圖 3-4-1 範例 3-4-1 的執行結果

```
mysql> SELECT dep_id, AVG(salary) FROM Employee
-> GROUP BY dep_id;
```

| dep_id | AVG(salary) |
|--------|-------------|
| NULL | NULL |
| D001 | 300000.0000 |
| D002 | 250000.0000 |
| D003 | 383333.3333 |

4 rows in set (0.09 sec)

計算每個部門的平均薪資！

即使記錄的 dep_id 為 NULL，仍可自成一個群組！

練習題 3-4-1

以 gender 欄位群組化 Employee 表格，再取得各群組的 gender 欄位以及計算 salary 欄位的最大值。

圖 3-4-2 練習題 3-4-1 的執行結果

```
mysql> [ ? ]
```

| gender | MAX(salary) |
|--------|-------------|
| 男 | 400000 |
| 女 | 450000 |

2 rows in set (0.00 sec)

計算各性別的最高薪資！

喔～結果是這樣啊！

3-4-2 於群組化的資料設定條件

利用 **GROUP BY** 陳述句群組化資料之後，可指定條件取得符合條件的資料，而此時用於指定條件的不是 **WHERE** 陳述句，必須改以 **HAVING** 陳述句指定。若使用 **WHERE** 陳述句，就會出現一個 SQL 陳述式同時有兩個 **WHERE** 陳述句，兩邊都無法正常發揮效果的問題。

因此 SQL 才以 **HAVING** 陳述句取代此時的 **WHERE** 陳述句。**WHERE** 陳述句可指定列資料的篩選條件，而 **HAVING** 陳述句則可指定群組資料的篩選條件。由於使用的關鍵字不同，所以更能明確區分兩者的用途。

FemaleEmployee 視圖則只是替 SELECT 命令的 SQL 陳述式命名所產生的東西，儘管這個 SELECT 命令能取得資料，視圖也只能眺望（取得資料）具有資料實體的表格（圖 4-1-4）。

圖 4-1-4 表格與視圖的差異

Employee 表格
(儲存資料的表格)

| emp_id | name | gender | birthday | salary | dep_id |
|--------|------|--------|------------|--------|--------|
| E00001 | 山田太郎 | 男 | 1995-01-01 | NULL | NULL |
| E00002 | 佐藤次郎 | 男 | 1990-05-03 | 250000 | D001 |
| E00003 | 鈴木花子 | 女 | 1990-02-11 | 250000 | D002 |
| E00004 | 田中三郎 | 男 | 1975-11-03 | 400000 | D003 |
| E00005 | 高橋良子 | 女 | 1985-11-23 | 300000 | D003 |
| E00006 | 鈴木良枝 | 女 | 1970-05-03 | 450000 | D003 |
| E00007 | 佐藤健次 | 男 | 1980-05-05 | 350000 | D001 |

實體



SELECT FROM Employee WHERE gender = '男';

MaleEmployee 視圖
(從 Employee 表格取得資料的表格)

| emp_id | name | gender | birthday | salary | dep_id |
|--------|------|--------|------------|--------|--------|
| E00001 | 山田太郎 | 男 | 1995-01-01 | NULL | NULL |
| E00002 | 佐藤次郎 | 男 | 1990-05-03 | 250000 | D001 |
| E00004 | 田中三郎 | 男 | 1975-11-03 | 400000 | D003 |
| E00007 | 佐藤健次 | 男 | 1980-05-05 | 350000 | D001 |

視圖 (眺望)

4-1-3 利用視圖精簡 SQL 陳述式

視圖可精簡冗長的 SQL 陳述式。例如以 gender 欄位為「男」、salary 欄位大於等於 30 萬元為條件，從 Employee 表格取得 name 欄位與 salary 欄位的 SQL 陳述式可寫成下列的內容。這個範例沒有使用視圖，之後會與使用了視圖的 SQL 陳述式比較。

範例 4-1-2

以 gender 欄位為「男」以及 salary 欄位為 30 萬元以上的條件，從 Employee 表格取得 name 欄位與 salary 欄位。

```
SELECT name, salary FROM Employee
WHERE gender = '男' AND salary >= 300000;
```

若使用 MaleEmployee 視圖可將 SQL 陳述式簡精成下列的內容，而且結果相同。
MaleEmployee 視圖只有男性的資訊，所以不需要多寫「gender = '男'」這個條件。

範例 4-1-3

根據 salary 欄位大於等於 30 萬元的條件，從 MaleEmployee 視圖取得 name 欄位與 salary 欄位

```
SELECT name, salary FROM MaleEmployee
WHERE salary >= 300000;
```

不管是否使用視圖，SQL 陳述式的執行結果都是一樣的（圖 4-1-5）。

圖 4-1-5 範例 4-1-2 與範例 4-1-3 的執行結果

```
mysql> SELECT name, salary FROM Employee
-> WHERE gender = '男' AND salary >= 300000;
```

```
+-----+-----+
| name   | salary |
+-----+-----+
| 田中三郎 | 400000 |
| 佐藤健次 | 350000 |
+-----+-----+
```

2 rows in set (0.12 sec)

不使用視圖取得資料！

得到這個結果！

```
mysql> SELECT name, salary FROM MaleEmployee
-> WHERE salary >= 300000;
```

```
+-----+-----+
| name   | salary |
+-----+-----+
| 田中三郎 | 400000 |
| 佐藤健次 | 350000 |
+-----+-----+
```

2 rows in set (0.04 sec)

使用視圖取得資料！

得到這個結果！

使用視圖的 SQL 陳式會在剖析視圖的陳述句時，執行視圖的內容（**SELECT** 陳述式），再取得表格格式的結果。

在剛剛的「**SELECT** name, salary **FROM** MaleEmployee **WHERE** salary >= 300000;」的 SQL 陳述句之中，MaleEmployee 視圖寫在 **FROM** 陳述句之後，所以當 **FROM** 陳述句被剖析時，視圖內容的「**SELECT * FROM** Employee **WHERE** gender = '男';」會執行，作為結果傳回的表格則於 **FROM** 後面指定（圖 4-1-6）。

圖 4-1-6 使用視圖的 SQL 陳述式的剖析順序



練習題 4-1-2

以 birthday 欄位小於「1980-01-01」的條件，從 FemaleEmployee 視圖取得 name 欄位與 birthday 欄位。

圖 4-1-7 練習題 4-1-2 的執行結果

```
mysql> [ ? ]
+-----+-----+
| name   | birthday |
+-----+-----+
| 鈴木良枝 | 1970-05-03 |
+-----+-----+
1 row in set (0.02 sec)
```

女性員工之中，有誰是資深員工？

這位女性員工！

4-1-4 相同的視圖可重複使用

或許大家會覺得使用剛剛的 MaleEmployee 視圖也只能省略「gender = '男'」這個條件，沒有想像中方便，但其實視圖的另一項優點在於可循環利用。舉例來說，