Appendix

B

規則物件、預設值物件與使用者定義資料類型物件

本章將介紹資料庫中的規則物件(Rules)、預設值物件(Defaults)、與使用者定義料類型物件(User-Defined Data Types, UDTs)。運用這 3 種物件可以讓我們更方便地設計及維護資料庫。

本章將使用**練習 BB** 資料庫為例說明, 請依關於光碟中的說明, 附加光碟中的資料庫到 SQL Server 中一起操作。

B-1 規則物件(Rule)

資料表中每個欄位都有特定的資料型別，可供輸入時判斷資料的正確性之用，例如 int 資料型別的欄位只能輸入整數數值，不能輸入字母、中文、特殊符號...等。然而光靠資料型別來查核資料的正確性是非常有限的，例如 int 型別的欄位可接受的數值範圍是 -2, 147, 483, 648 ~ 2, 147, 483, 647，假若我們想限定可輸入的資料範圍在 1 ~ 50000 之間的時候該怎麼做呢？又電話號碼的欄位使用 char 資料型別，但要如何限定只能輸入 0~9 的數字而不能輸入其它的字元呢？

其實在第 7 章所介紹的 CHECK 條件約束 (CHECK Constraint)，即可設定欄位值的檢查條件，例如：

```
CREATE TABLE 員工薪資
(
編號 int IDENTITY PRIMARY KEY,
薪資 smallmoney,
CHECK (薪資 > 0 AND 薪資 <= 50000)  ◄── 設定只能輸入 1 ~ 50000 之間的數值
)
```

CHECK 條件約束會儲存在資料表中 (屬於資料表的一部份)，而且可以針對多個欄位做檢查，例如：

```
CREATE TABLE 經銷商
(
 編號 int IDENTITY PRIMARY KEY,
 姓名 char(20) NOT NULL,
 地址 char(50),
 電話 char(13),
 CHECK (地址 IS NOT NULL OR 電話 IS NOT NULL) ◄── 檢查地址及電話欄位至少要填入一項
)
```

規則物件的功能和 CHECK 非常類似，不過**規則**物件是獨立儲存的物件，它必須與資料表的欄位繫結 (Bind) 後，才能發揮查核的作用。**規則**物件與 CHECK 條件約束的優點分別是：

- **規則物件的優點**是，同一個規則物件可以供不同資料表的不同欄位使用，但每個欄位最多只能和一個規則物件繫結。此外，規則物件還可以與**使用者自訂資料型別**物件結合，讓該型別具備資料查核功能 (請參閱 B-3 節)。

- **CHECK 的優點**則是，每個資料表可以有多個 CHECK 條件約束，而且 CHECK 條件約束可以針對所屬資料表中的多個欄位做查核設定。

資料表可以同時使用 CHECK 條件約束與規則物件。但微軟建議我們儘量使用 CHECK 來做查核，這是因為當很多規則物件和很多欄位繫結時，資料庫會變得較為複雜而不易管理及維護，而且規則物件只能針對單一欄位做檢查，不像 CHECK 這麼有彈性。基本上，規則物件只是 SQL Server 2016 為了與以前版本相容而保留的功能，未來版本可能不再提供此功能，所以建議您還是儘可能使用 CHECK 條件約束。

建立規則物件

建立規則物件雖然是要用來限制欄位的輸入範圍或條件，但規則物件並不直接附屬於欄位。我們要先建立規則物件，然後再將之繫結 (Bind) 到某個指定的欄位上，這才能夠使規則物件發揮作用。要提醒您注意的是，若原本欄位有設定**預設值**，則該**預設值**必須在規則物件的限制範圍內，否則會發生錯誤。

利用 CREATE RULE 敘述可建立**規則**物件，其語法如下：

```
CREATE RULE rule_name      ◄── 指定規則物件的名稱
AS condition_expression    ◄── 設定條件內容
```

比照設定 WHERE 子句的方式來設定規則物件的條件即可，但是由於規則物件並不直接附屬於欄位，且可供不同資料表的不同欄位使用，因而規則物件的條件中必須使用一個區域變數 (名稱須以 @ 開頭) 來代表欄位值，此變數的名稱可以任意取名。底下是幾種常用的規則物件類型：

◉ 限定數值的範圍

```
CREATE RULE Price_rule          ◄── 限制數值在 1 ~ 50000 之間
AS @price >= 1 AND @price <= 50000
```

◉ 限定字串的範圍、格式

```
CREATE RULE charset_rule
AS @charset LIKE 'F0[1-9][1-9]-[A-E]_'
```

'F0[1-9][1-9]-[A-E]_' 只允許輸入如 F0nn-xx 這種型式的資料，其中[1-9]表示數字 1~9；[A-E] 表示只允許 A~E 的字母；而最後的'_'，則表示可以是任何字母，也可以不填入。例如：F055-EW 或 F035-A 皆符合此規則物件的條件要求。

◉ 限定只能使用指定的值

```
CREATE RULE Gender_rule
AS @gender IN ( '男' , '女' )  ◄── 只允許輸入這 2 種值
```

請勿將 NULL 設定為限定條件

依筆者的經驗，若在限定能使用的值中，加上限定使用 NULL 值，將會導致此規則失去作用，因此請勿將 NULL 設定為限定條件，而要改用資料欄位的屬性來設定：

◉ **限定時間的範圍**

```
CREATE RULE payday_rule
AS @payday >= getdate()          ◄── 限定日期必須不晚於記錄建立的時間
```

再次提醒您，在規則物件的條件內容中，不能包含任何欄位名稱或物件名稱在內。

繫結規則物件與欄位

建立規則物件後，接下來則要將它與資料表的欄位繫結，使其發揮作用。但要請您注意，欲繫結的欄位，其資料型別必須與規則物件的資料型別相容，而且規則物件不能與 text、ntext、image、varchar(max)、nvarchar(max)、varbinary(max)、xml、或 timestamp 型別的欄位繫結。

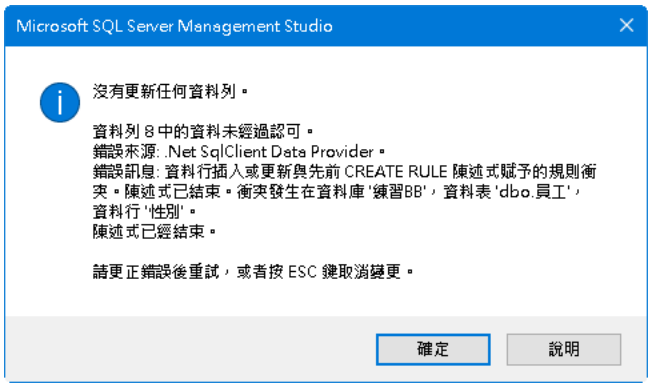
我們可以使用 SQL Server 內建的預存程序 sp_bindrule 來繫結規則物件與欄位，其語法如下：

```
EXEC sp_bindrule rule_name, 'table_name.column_name'
```

例如：

```
EXEC sp_bindrule Gender_rule, '員工.性別'   ◄── 將前面 B-4 頁建立的 Gender_rule
                                              與員工資料表的性別欄位繫結
```

欄位與規則結合之後，就受該規則的限制，例如我們在**員工**資料表的**性別**欄位輸入 '男'、'女' 之外的值，便會出現錯誤訊息：



用新的規則取代舊的

如果想要使用一個新的規則物件取代先前為欄位繫結的舊規則物件，只要直接用 sp_bindrule 預存程序去結合新的規則物件，SQL Server 就會自動用新的規則取代舊的。

解除規則與欄位的繫結關係

若要解除規則物件與欄位的繫結關係，可利用 sp_unbindrule 預存程序，其語法如下：

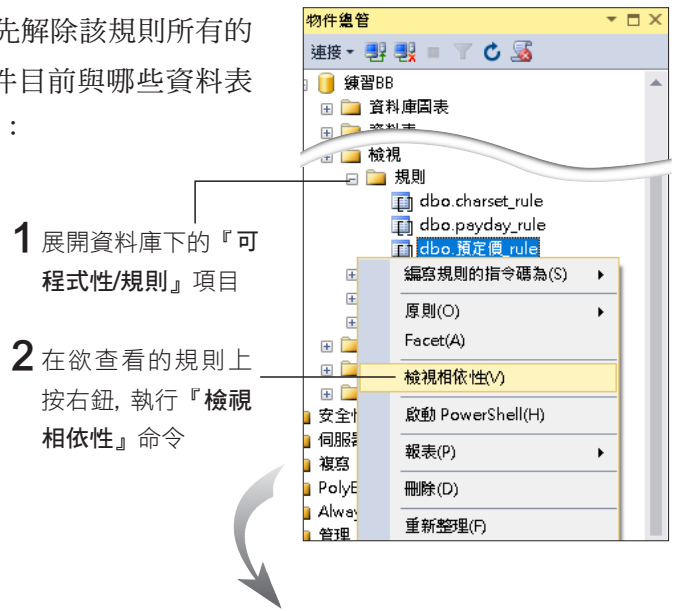
```
EXEC sp_unbindrule 'table_name.column_name' ◄── 只需指定欲解除的欄位名稱即可
```

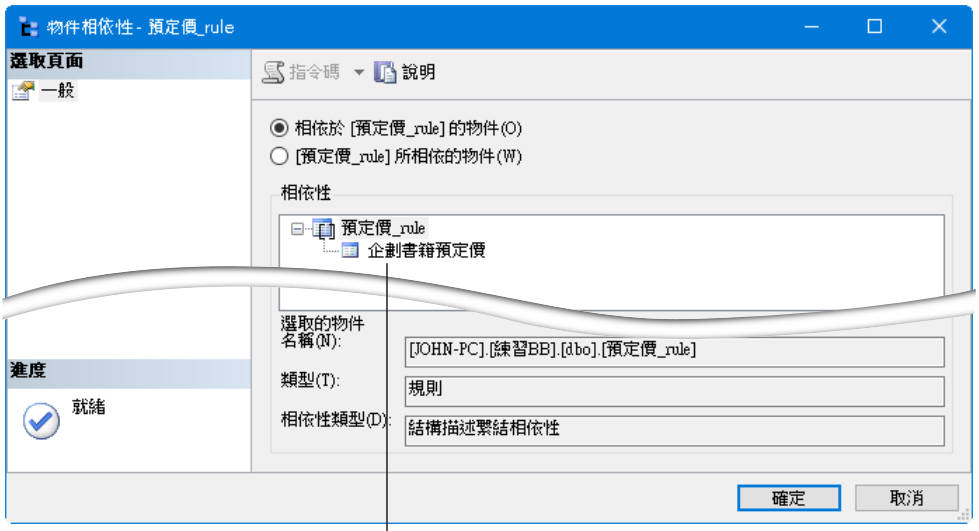
例如我們想解除先前建立的**員工**資料表**性別**欄位與 **Gender_rule** 的繫結關係，執行如下敘述即可：

```
EXEC sp_unbindrule '員工.性別'
```

刪除規則物件

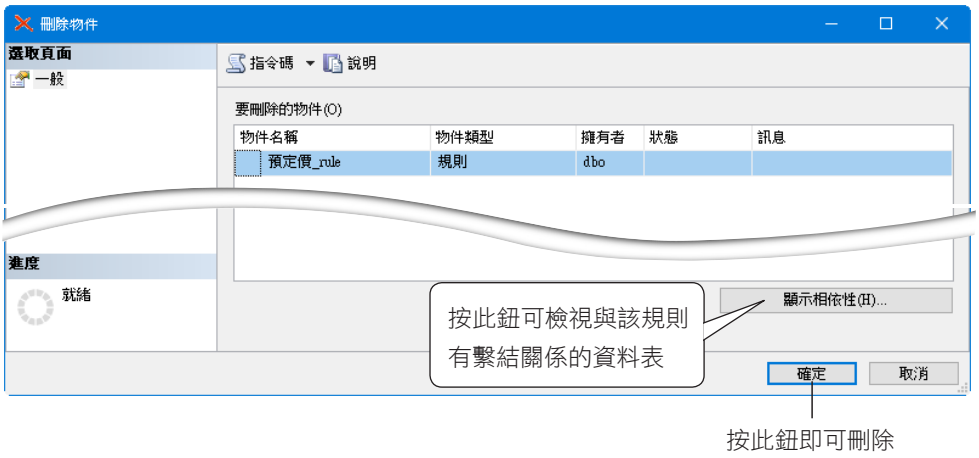
要刪除規則之前，必須先解除該規則所有的繫結關係。欲查看規則物件目前與哪些資料表有繫結關係，操作方法如下：





確定了繫結的資料表後，就可以先用前面介紹的 sp_unbindrule 預存程序來解除繫結關係，接著再如下刪除規則物件。

刪除規則的方法有 2 種：一是在**物件總管**窗格中展開資料庫的**可程式性/規則**項目，直接選取欲刪除的規則物件後，按 Delete 鍵 (或按右鈕執行『**刪除**』命令、或執行『**編輯/刪除**』命令)，接著便會出現如下的交談窗：



另一種方法是利用 DROP RULE 敘述來刪除規則物件，語法如下：

```
DROP RULE rule_name [ , ...n]
```

例如：

```
DROP RULE Price_rule, Geneder_rule        ◄── 可同時指定多個規則物件一併刪除
```

B-2 預設值物件(Default)

預設值(Default) 物件可用來指定欄位的預設值。雖然在設計資料表的結構時，已有一個**預設值**屬性可以直接指定欄位的預設值，但這個值只有該欄位可以使用。此處要談的**預設值**物件則是給預設值一個名稱 (例如用 rate 來表示 0.9)，然後將之儲存成物件，任何資料表欄位都可以與 rate 物件結合來設定欄位的預設值。

其實**預設值**物件的性質和**規則**物件很像，都是獨立儲存的物件，並且必須先與欄位繫結 (Bind) 之後才有作用。**預設值**物件也可以與**使用者自訂資料類型**物件繫結，作為使用該自訂類型欄位的預設值 (詳情可參閱 B-3 節)。至於資料表欄位的**預設值**屬性則類似 CHECK，都是直接儲存於資料表中，不能重複使用。

此外，有件事要在此說明，**預設值**物件固然有其優點，但它其實和**規則**物件一樣，都是 SQL Server 2016 為了和以前版本相容所保留的功能。所以若要設定欄位的預設值，還是儘可能使用資料表的**預設值**屬性。

建立預設值物件

我們可以用 CREATE DEFAULT 敘述來建立預設值物件。CREATE DEFUALT 敘述的語法和之前建立規則物件的 CREATE RULE 敘述很像，就連設定的方式也差不多：

```
CREATE DEFAULT default_name      ◄── 設定預設值物件的名稱
AS constant_expression           ◄── 指定預設值物件的內容，如一個數值或字串
```

例如：

```
CREATE DEFAULT 性別_df
AS '男'
```

或是

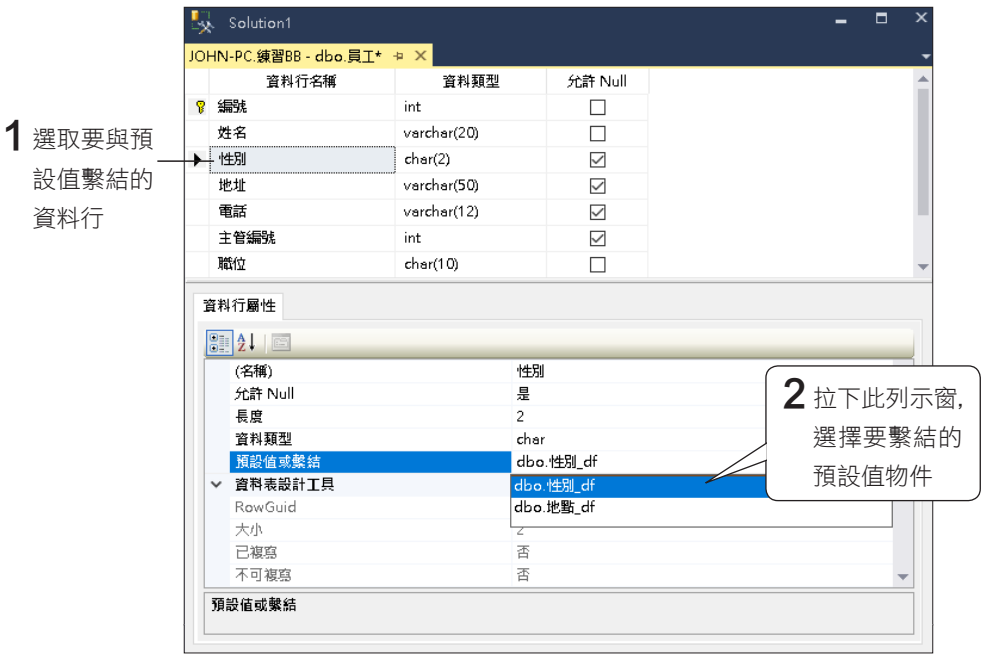
```
CREATE DEFAULT 地點_df
AS '台灣'
```

繫結預設值物件與資料表欄位

如上所述建立好預設值物件後，接著便可以如下操作將預設值物件繫結到資料表欄位上。

使用 SQL Server Management Studio 管理工具繫結

若要使用 SQL Server Management Studio 管理工具進行繫結，請先在**物件總管**窗格中選取欲處理的資料表，如**員工**資料表，再按滑鼠右鈕執行『**設計**』命令：



用預存程序 sp_bindefault 繫結

我們也可以利用 SQL Server 內建的 sp_bindefault 預存程序來繫結**預設值**物件與欄位。sp_bindefualt 的語法如下：

```
EXEC sp_bindefualt default_name, 'table_name.column_name'
```

例如：

```
EXEC sp_bindefault 性別_df , '員工.性別'      ←── 將先前建立的性別_df 和員工
                                                  資料表的性別欄位繫結
```

解除預設值物件與欄位的繫結關係

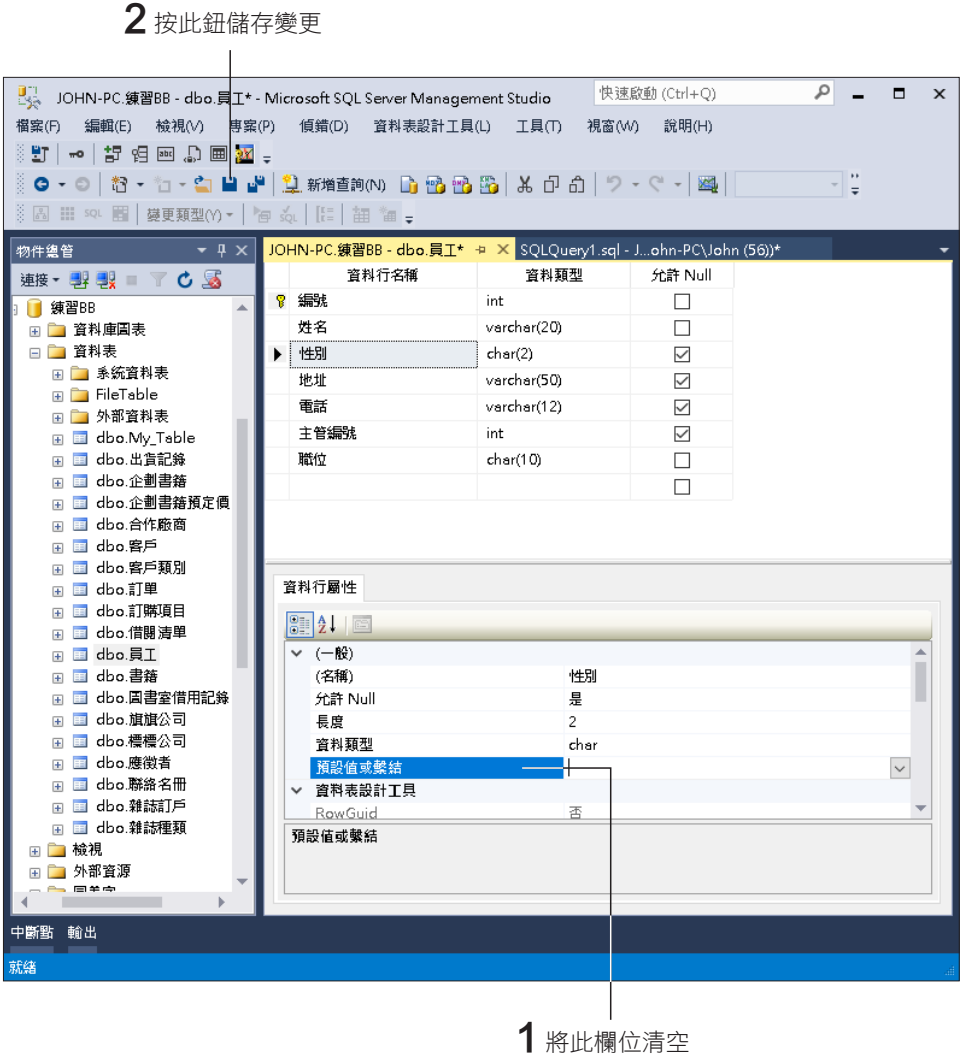
若要解除**預設值**物件與欄位的繫結關係，可利用 sp_unbindefault 預存程序，其語法如下：

```
sp_unbindefault 'table_name.column_name'
```

例如要解除先前設定的**員工**資料表**性別**欄位與**性別_df** 預設值物件的繫結，則請執行以下敘述：

```
EXEC sp_unbindefault '員工.性別'
```

若要使用 SQL Server Management Studio 提供的管理工具解除繫結，請同樣在**物件總管**窗格中選取欲處理的資料表，然後按滑鼠右鈕執行『**設計**』命令，將**預設值或繫結**欄位清空即可：



刪除預設值物件

建立的**預設值**物件若已經用不到了，可將之刪除；不過在刪除之前，必須先將該物件與欄位的繫結關係全數解除才行。各位可比照**規則**物件的方式來檢視**預設值**物件目前的依存關係，至於解除繫結的方法，剛才已經介紹過了。

而刪除**預設值**物件，若是使用 SQL Server Management Studio 管理工具進行，其方法和刪除**規則**物件一模一樣，我們就不再重覆說明了。另外，也可以用 DROP DEFAULT 敘述來刪除**預設值**物件，其語法如下：

```
DROP DEFAULT default_name [, ...n]

DROP DEFAULT 地點_df
```

B-3 使用者定義資料類型(UDTs) 物件

我們知道，在設計資料表的結構時，每個欄位都要指定一個適當的資料型別，SQL Server 也內建了許多資料型別供我們使用 (內建資料型別在第 4 章就介紹過了)。但是在建立資料表的經驗比較多了以後，其實可以發現，有些資料型別與長度是常常重複使用的，例如通常會將**地址**欄位設定為 varchar(40)，或是將**薪資**欄位設為 numeric(8, 2) ...等。

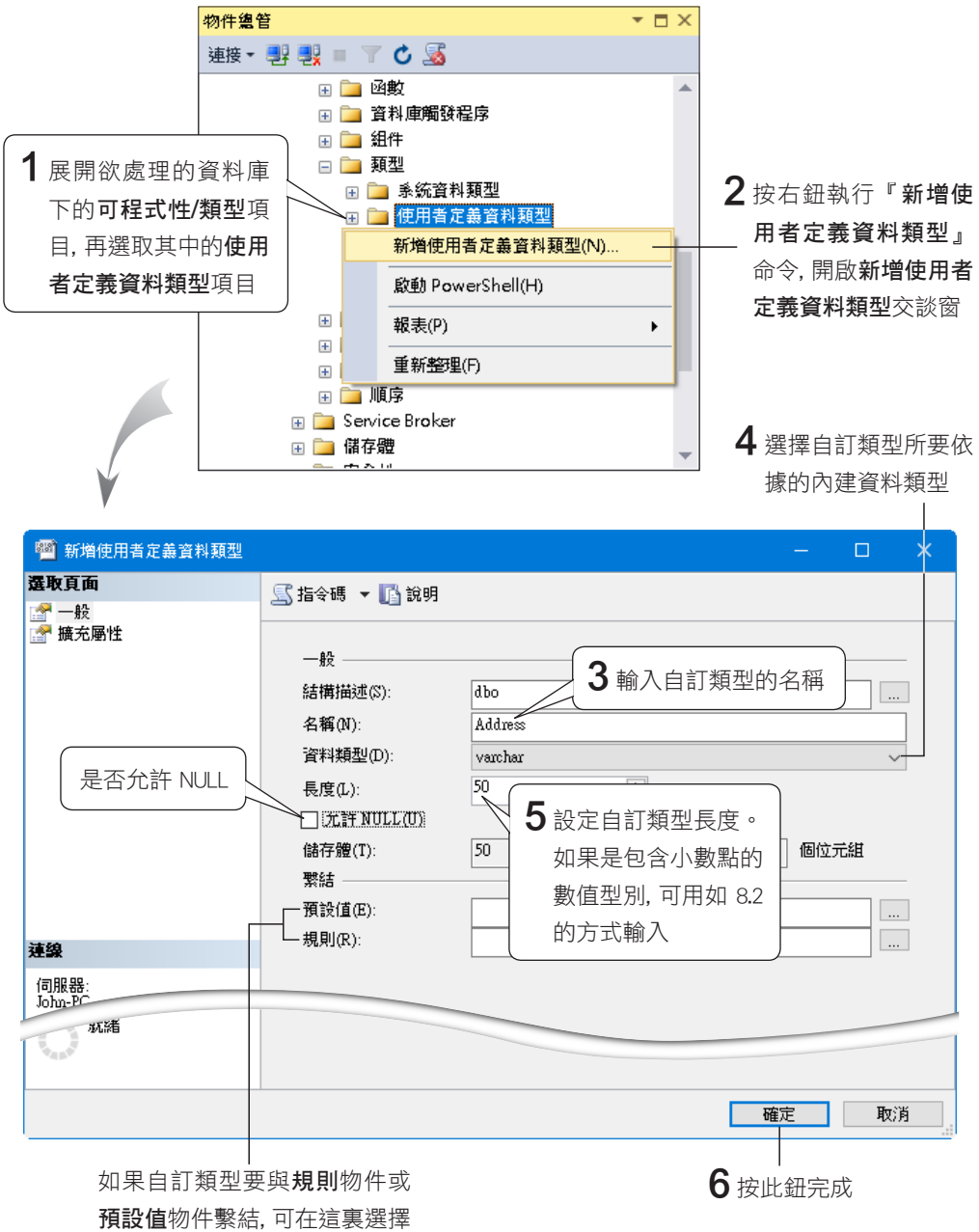
除此之外，相同的欄位也有可能出現在不同的資料表中，例如**員工**資料表有**地址**欄位，**客戶**資料表也有**地址**欄位，若一時不查，將**員工.地址**欄位設為 varchar(40)，而將**客戶.地址**欄位設為 char(30)，這不是自找麻煩嗎？為了避免這樣的情形，我們可以乾脆自己定義一個"address" 的資料型別，如此每次要設定**地址**欄位時，就選擇這個資料型別，這樣就不容易出錯了。

建立使用者定義資料類型

使用者定義資料類型 (User-defined Data Types；或稱為 UDTs，也可簡稱為**自訂型別**) 實際上還是使用 SQL Server 內建的資料型別 (不包括 timestamp) 來變化，並不是讓我們自己變魔術，定義出內建資料型別以外的新資料型別。了解這層意義後，現在就來看看如何建立**使用者定義資料型類型**！

使用 SQL Server Management Studio 建立自訂型別

使用 SQL Server Management Studio 建立**使用者定義資料類型**的步驟如下：



用 CREATETYPE 敘述建立

我們還可以使用 SQL Server 的 CREATE TYPE 敘述來建立使用者定義資料類型，其語法如下：

```
CREATE TYPE type_name
FROM system_data_type
[ NULL | NOT NULL ]
```

- **type_name：**為欲建立的**使用者定義資料類型**的名稱。
- **system_data_type：**設定自訂型別所要依據的內建資料型別。
- **NULL | NOT NULL：**設定此自訂型別是否允許 NULL，允許時為 NULL，不允許則為 NOT NULL。若未指定，則預設為 NULL。

底下示範兩個自訂型別的範例：

```
CREATE TYPE Phone          ←— 定義 Phone 自訂資料型別
FROM char(12)
NO TNULL

CREATE TYPE PayDay         ←— 定義 PayDay 自訂資料型別
FROM date
NULL
```

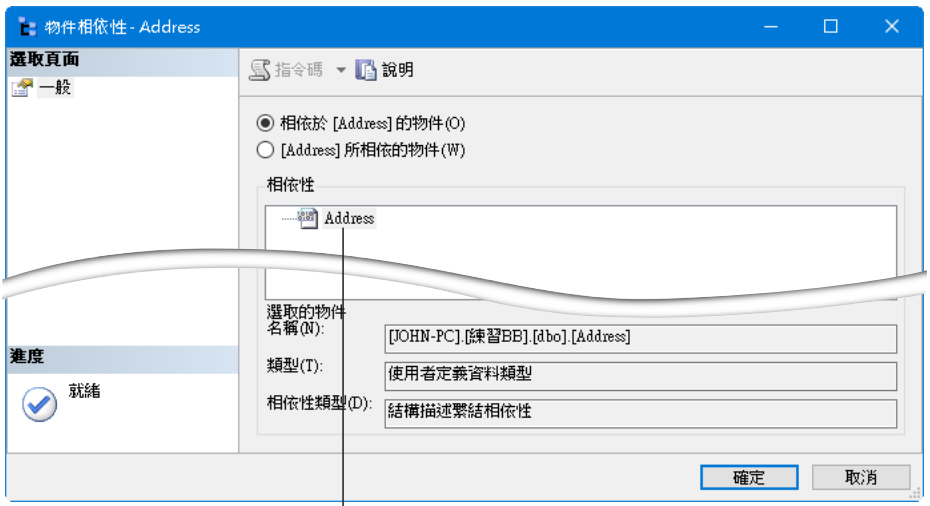
使用自訂型別

建立**使用者定義資料類型**後，以後在定義資料表的欄位時就可直接選擇：

使用者自訂資料類型也會出現在資料型別列示窗中

查看使用者定義資料類型的使用情況

若要查看使用者定義資料類型的使用情況, 可在物件總管窗格內如下查看：



刪除使用者定義資料類型

假若建立的自訂型別已經無用武之地，理所當然可以將之刪除。但是在刪除之前，必須先確定沒有任何欄位使用到這個預備刪除的自訂型別，否則是刪不掉的。

使用 SQL Server Management Studio 管理工具刪除自訂型別的方法和刪除**規則**物件、**預設值**物件一樣，我們不再贅述。另外，也可用 SQL Server 的 DROP TYPE 敘述來刪除**使用者定義資料類型**，其語法如下：

```
DROP TYPE type_name
```

例如：

```
DROP TYPE Phone        ◄── 刪除先前建立的 Phone 資料類型
DROP TYPE PayDay       ◄── 刪除先前建立的 PayDay 資料類型
```

建立『使用者定義資料表類型』

在第 13-14 節中曾介紹過 table 型別，而 table 型別也可以用來建立自訂型別，稱為『**使用者定義資料表類型**』。例如底下建立 MyBook 自訂型別來使用：

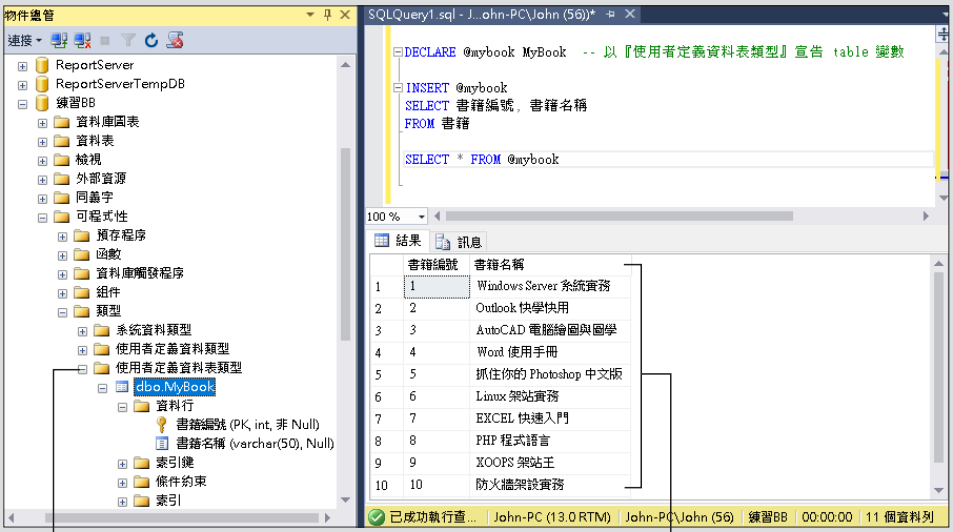
```
CREATE TYPE MyBook AS TABLE     ◄── table 型別必須使用 AS 來定義, 而不是 FROM
  (書籍編號 int PRIMARY KEY,
   書籍名稱 varchar(50))

GO

DECLARE @mybook MyBook          ◄── 以『使用者定義資料表類型』宣告 table 變數

INSERT @mybook
SELECT 書籍編號, 書籍名稱
FROM 書籍

SELECT * FROM @mybook
```



在**使用者定義資料表類型**上按右鈕執行『**重新整理**』命令, 即可看到新建立的 MyBook 型別

執行的結果