CHAP 04 認識 SQL 語言與資料型別

- 4-1 SQL 語言的興起與語法標準
- 4-2 SQL 語言與傳統程式語言的差別
- 4-3 關鍵字、子句與敘述
- 4-4 SQL 語言的功能分類
- 4-5 資料型別
- 4-6 欄位的 NULL 值與 DEFAULT 值
- 4-7 識別名稱 (Identifier)

4-1 SQL 語言的興起與語法標準

- SQL 語言是在 1970 年代晚期，由 IBM 公司在美國加州聖荷西 (San Jose) 的研究單位所發展出來的一套程式語言
- SQL 的語法標準有「業界標準」與「ANSI SQL 標準」之分

符合 ANSI SQL 標準的 SQL 敘述，可以在任何資料庫系統上運行。

4-2 SQL 語言與傳統程式語言的差別

- 用 **SQL** 語言寫成的程式必須應用在資料庫管理系統中，本身並不能獨立執行，而且是 "非程序性" **(non-procedural)** 語言，**MS SQL Server** 使用的叫 **T-SQL (Transact-SQL)**。

```
SELECT *          ←—— 挑選出資料表中的所有欄位（* 表示所有的欄位）
FROM   書籍資料表  ←—— 指定資料表
WHERE  價格 > 400  ←—— 挑選的條件
```

4-3 關鍵字、子句與敘述

- **SQL** 語法的基礎是子句 **(clause)**，子句中會包括一些關鍵字 **(keyword)**，關鍵字是對 **SQL Server** 有特別意義的字。敘述 **(statement)** 則是指一組可產生存取資料庫結果的子句集合。

```
SELECT *                ◄── 這是一個 SELECT 子句, 使用到 SELECT 關鍵字
FROM   書籍資料表       ◄── 這是一個 FROM 子句, 使用到 FROM 關鍵字
WHERE  價格 > 400       ◄── 這是一個 WHERE 子句, 使用到 WHERE 關鍵字
```

```
CREATE DATABASE MyDatabase ◄── 這個子句使用了 CREATE 與 DATABASE 這兩個關鍵字
```

以上是兩個 **SQL** 敘述，每一個敘述最後可以加上一個『 ； 』符號，表示該敘述已經結束。

4-4 SQL 語言的功能分類

- DDL (Data Definition Language)
- DML (Data Manipulation Language)
- DCL (Data Control Language)

DDL
(DATA DEFINITION LANGUAGE)

- 用來定義 (或建立) 資料庫物件，以及修改資料庫物件結構的 **SQL** 敘述，都屬於 **DDL** (資料定義語言)。

```
CREATE DATABASE MyDB          ←—— 建立一個資料庫, 叫做 MyDB

CREATE TABLE MyTable          ←—— 建立一個資料表, 叫做 MyTable
  (
   書籍編號   int,
   書籍名稱   char(30),       ——— 資料表中的欄位定義
   價格       smallmoney
  )
```

DML
(DATA MANIPULATION LANGUAGE)

- 用來做資料處理的敘述則屬於 **DML (資料處理語言)**。

```
INSERT MyTable                          ←—— 在資料表中新增一筆記錄
        (書籍編號, 書籍名稱, 價格)
VALUES
        (101, 'Windows 使用手冊', 500)

SELECT   書籍編號, 書籍名稱, 價格       ←—— 從 MyTable 資料表顯示指定欄位的資料
FROM  MyTable

UPDATE   MyTable                        ←—— 更改記錄的內容
SET      價格 = 499
WHERE    書籍名稱 = 'Windows 使用手冊'

DELETE   FROM MyTable                   ←—— 將符合條件的記錄刪除
WHERE    書籍名稱 = 'Windows 使用手冊'
```

DCL
(DATA CONTROL LANGUAGE)

- **DCL** (資料控制語言) 一般是指專門用來設定資料庫物件 (如資料表、檢視表、預存程序等) 使用權限的敘述。 例如，**GRANT** (允許使用)、**DENY** (拒絕使用)、**REVOKE** (取消權限設定)。

4-5 資料型別

- 整數
- 精確位數
- 近似浮點數值
- 日期時間
- 字串
- **Unicode** 字串
- 二元碼字串
- 貨幣
- 標記
- **XML**
- 空間資料
- 其它

資料型別可分為系統內建資料型別與使用者自訂資料型別。每一家資料庫管理系統的資料型別名稱會有差異，本課程皆以 **T-SQL** 為準。

整數

資料型別	資料範圍	使用的位元數(長度)
bigint	-2^{63} ~ 2^{63} -1 (-9,223,372,036,854,775,808~ 9,223,372,036,854,776,807)	8 bytes
int	-2^{31} ~ 2^{31} -1 (-2,147,483,648 ~ 2,147,483,647)	4 bytes
smallint	-2^{15} ~ 2^{15} -1 (-32,768 ~ 32,767)	2 bytes
tinyint	0 ~ 2^{8} -1(0 ~ 255)	1 byte
bit	0、1、NULL	實際使用 1 bit, 但會佔用 1 byte。若資料表中有數個 bit 欄位, 則會共用 1 byte。例如, 若有 1 ~ 8 個 bit 欄位便佔 1 byte, 9 ~16 個 bit 欄位便佔 2 bytes, 以此類推

精確位數

資料型別	資料範圍	使用的位元數(長度)
numeric	$-10^{38} + 1 \sim 10^{38} - 1$	視精確度 (即全部有效位數) 而定 1 ~ 9 位數使用 5 bytes 10 ~ 19 位數使用 9 bytes 20 ~ 28 位數使用 13 bytes 29 ~ 38 位數使用 17 bytes
decimal	$-10^{38} + 1 \sim 10^{38} - 1$	與 numeric 相同

這兩種資料型別使用上完全相同，留著是為了保留相容性。
numeric (5, 2)：共有 **5** 位數，其中 **3** 位整數及 **2** 位小數。

近似浮點數值

資料型別	資料範圍	使用的位元數(長度)
float	-1.79E ＋ 308 ~ 1.79E ＋ 308 最多可表示 15 位數	8 bytes
real	-3.40E ＋ 38 ~ 3.40E ＋ 38 最多可表示 7 位數	4 bytes

日期時間

資料型別	資料範圍	使用的位元數(長度)
datetime	1753/1/1 ~ 9999/12/31, 時間可精確到 "3.33 毫秒" (即 3.33/1000 秒), 輸入格式為 yyyy-mm-dd hh:mm:ss, 如：2006-1-23 15:43:26.799	8 bytes 前 4 個 bytes 儲存日期 後 4 個 bytes 儲存時間
datetime2	0001-01-01 00:00:00.0000000 ~ 9999-12-31 23:59:59.9999999, 時間可精確到 "100 奈秒" (即 100×10^{-9} 秒), 輸入格式為 yyyy-mm-dd hh:mm:ss[.nnnnnnn], 如 2008-10-25 17:28:19.53	6 ~ 8 個 bytes
smalldatetime	1900/1/1 ~ 2079/6/6, 時間可精確到 "分", 輸入格式為 yyyy-mm-dd hh:mm 如：2006- 1-23 15:43	4 bytes 前 2 個 bytes 儲存日期 後 2 個 bytes 儲存時間
date	0001:01:01 ~ 9999:12:31, 時間可精確到 "天", 輸入格式為 yyyy-mm-dd, 如 2008-11-07	3 個 bytes
time	00:00:00.0000000 ~ 23:59:59.9999999, 時間可精確到 "100 奈秒" (即 100×10^{-9} 秒), 輸入格式為 hh:mm:ss[.nnnnnnn], 如 22:15:33.681	3~5 個 bytes
datetimeoffset	0001-01-01 00:00:00.0000000 ~ 9999-12-31 23:59:59.9999999 ±14:00, 時間可精確到 "100 奈秒" (即 100×10^{-9} 秒), 輸入格式為 yyyy-mm-dd hh:mm:ss[.nnnnnnn] {+|-}hh:mm, 如 2008-11-07 22:15:33 +08:00	8~10 個 bytes

字串

資料型別	資料範圍	使用的位元數(長度)
char(n)	1 ~ 8000 個字元	1 個字元 1 byte, 為固定長度, 未填滿資料的部份會自動補上空白字元
varchar(n)	1 ~ 8000 個字元	1 個字元 1 byte, 儲存多少字元即佔多少空間
varchar(max)	1 ~ 2^{31} -1 個字元	為變動長度, 輸入多少字元即佔用多少空間, 最大可達 2GB
text	1 ~ 2^{31} -1 個字元	變動長度, 輸入多少字元即佔用多少空間, 最大可達 2GB

微軟建議在 SQL Server 中使用 varchar(max)，而不使用 text。舊版 SQL Server (2005) 讀取 varchar(max) 型別資料，會被辨識為 text。

UNICODE 字串

資料型別	資料範圍	使用的位元數(長度)
nchar(n)	1 ~ 4000 個字元	1 個字元 2 byte, 為固定長度, 未填滿資料的部份會自動補上空白字元
nvarchar(n)	1 ~ 4000 個字元	1 個字元 2 byte, 儲存多少字元即佔多少空間
nvarchar(max)	1 ~ 2^{30} -1 個字元	1 個字元 2 byte, 為變動長度, 輸入多少字元即佔用多少空間, 最大可達 2GB
ntext	1 ~ 2^{30} -1 個字元	1 個字元 2 byte, 為變動長度, 輸入多少字元即佔用多少空間, 最大可達 2GB

二元碼字串

資料型別	資料範圍	使用的位元數(長度)
binary(n)	1 ~ 8000 個 bytes	固定長度, 輸入的資料若未達 8000 個 bytes, 不足的部份則自動補上 0x00
varbinary(n)	1 ~ 8000 個 bytes	變動長度, 輸入多少資料即佔用多少空間
varbinary(max)	1 ~ 2^{31} -1 個 bytes	為變動長度, 輸入多少資料即佔用多少空間, 最大可達 2GB
image	1 ~ 2^{31} -1 個 bytes	變動長度, 輸入多大資料即佔用多少空間, 最大可達 2GB

可以用來儲存 **Word、Excel、BMP、GIF、JPEG**

貨幣

資料型別	資料範圍	使用的位元數(長度)
money	-2^{63} ~ $2^{63}-1$ (-922,337,203,685,477.5808~922,337,203,685,477.5807) 可精確到小數第 4 位	8 bytes
smallmoney	-2^{31} ~ $2^{31}-1$ (-214,748.3648 ~ 214,748.3647) 可精確到小數第 4 位	4 bytes

標記

資料型別	資料範圍	使用的位元數(長度)
timestamp	8 bytes 的 16 進位值 如 0x0000000000000130	8 bytes
uniqueidentifier	16 bytes 的 16 進位值 如 4C047CB3-B007-11D2-9C59-0080C846994D	16 bytes

timestamp：紀錄資料更新的時間戳記，紀錄更新時，此值也會隨之更新，整個資料庫中的唯一值。

uniqueidentifier：是全域唯一識別碼，可以用來識別每一筆紀錄的唯一性。當紀錄中的此欄位值產生後，就不會自己改變。

```
CREATE TABLE 員工記錄
(
   記錄編號 int IDENTITY(1, 1) NOT NULL,
      異動日期 datetime,
      員工編號 int,
   薪資 smallmoney,
      備註 sql_variant,          ← 這是 D-3 節將介紹的 sql_variant 資料型別
   tmstmp timestamp,             ← tmstmp 欄位的資料型別為 timestamp
      guid uniqueidentifier      ← 這是 D-2 節將介紹的 uniqueidentifier 資料型別
)
```

```
INSERT 員工記錄 (異動日期, 員工編號, 薪資)
VALUES ( GETDATE(), 5, 30000 )

SELECT * FROM 員工記錄
```

	記錄編號	異動日期	員工編號	薪資	備註	tmstmp	guid
1	6	2016-12-18	5	30000.00	NULL	0x00000000000002EE1	NULL

自動產生 tmstmp 欄位的值

```
UPDATE 員工記錄
SET 異動日期 = GETDATE()
WHERE 員工編號 = 5

SELECT * FROM 員工記錄
```

	記錄編號	異動日期	員工編號	薪資	備註	tmstmp	guid
1	6	2016-12-18	5	30000.00	NULL	0x00000000000002EE2	NULL

因為記錄被修改, tmstmp 欄位的值也自動改變了

Step1 新增一筆記錄，暫不指定 guid 欄位的值：

```
INSERT 員工記錄 (異動日期, 員工編號, 薪資)
VALUES ( GETDATE(), 6, 32000 )

SELECT * FROM 員工記錄
```

	記錄編號	異動日期	員工編號	薪資	備註	tmstmp	guid
1	6	2016-12-18	5	30000.00	NULL	0x0000000000002EE2	NULL
2	7	2016-12-18	6	32000.00	NULL	0x0000000000002EE3	NULL

Step2 利用 NEWID() 函數為剛才那筆記錄的 guid 欄位產生 GUID 值：

```
UPDATE 員工記錄
SET guid = NEWID()        ← 使用 NEWID() 函數為 guid 欄位設定新值
WHERE 員工編號 = 6

SELECT * FROM 員工記錄
```

	記錄編號	異動日期	員工編號	薪資	備註	tmstmp	guid
1	6	2016-12-18	5	30000.00	NULL	0x0000000000002EE2	NULL
2	7	2016-12-18	6	32000.00	NULL	0x0000000000002EE4	887056BE-20B0-4EE8-9EEB-E550320CDBBF

XML

資料型別	資料範圍	使用的位元數(長度)
xml	符合 xml 格式的任何資料	最大可儲存 2GB

國內訂單

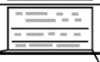
下單日期：2016/2/10

項目編號	書籍名稱	出版公司	價格	數量
1	Linux 實務應用	旗旗出版公司	620	150
2	BIOS 玩家實戰	旗旗出版公司	299	100
3	Windows 系統秘笈	旗旗出版公司	490	80

應付總價：162100

送貨地址：台北市忠孝東路九段 10 號

聯絡人：吳明士

廠商編號	廠商名稱	地址	電話	聯絡人	訂單
001	十全書店	台北市	0212345678	陳圓圓	
002	Bear Education	Boston	(617)555-0701	Bear Gray	

國際訂單

下單日期：2016/2/10　　出口報單號碼　AC20251355

項目編號	書籍名稱	出版公司	價格	數量
1	XOOPS 架站王	旗旗出版公司	620	150
2	Linux 指令詳解辭典	旗旗出版公司	299	100
	小計			122,900
	出口關稅			1,570
	海運費用			2,580
	應付總價：			127,050

送貨地址：Bear Education, INC
Rights and Contracts Department, 45 Arlington Street,
Suite 310 Boston, MA 02116

聯絡人：Bear Gray

空間資料

- 空間資料共有兩種資料型別：**geometry** 與 **geography**。**geometry** 可以儲存平面資料，支援開放式地理空間協會 **(Open Geospatial Consortium**，簡稱 **OGC)** 的 **SQL** 簡單特徵規格 **1.1.0** 版；**geography** 則可以儲存橢圓體資料，格式較為複雜，本課程不多加說明。

其它

資料型別	資料範圍	使用的位元數(長度)
sql_variant	可存放各種資料型別的資料，除了 text、ntext、image、timestamp、sql_variant 以外	視儲存的資料型別而訂，最大長度為 8016 個位元 (bit)
cursor	查詢結果的資料集	
table	表格型式的資料	
hierarchyid	樹狀目錄階層中的位置	

如果一個欄位可能存放不同型別的資料，即可將此欄位設為 **sql_variant** 型別。

4-6 欄位的 NULL 值與 DEFAULT 值

- NULL 值
- DEFAULT 值

NULL 值

- **NULL** 表示 "不知 道"、"不確定" 或 "暫時沒有資料" 的意思。

```
CREATE  TABLE  客戶資料表 ◄── 建立客戶資料表, 下方( ) 內在設定資料表中的欄位
(
  客戶編號      int          NOT NULL,    ┐ 這兩個欄位都設為 NOT NULL,
  客戶名稱      char(30)     NOT NULL,    ┘ 表示一定要輸入資料
  地址          char(60),                 ┐
  聯絡電話      char(12),                 │ 這些沒有特別指定 NOT NULL
  傳真號碼      char(12),                 │ 的欄位, 則可以不要輸入資料
  電子郵件      char(30)                  ┘
)
```

DEFAULT 值

- **Step 1**

```
CREATE  TABLE  訂單
(
  訂單編號    int           NOT NULL,
  接單日期    datetime      DEFAULT getdate(),
  訂購金額    smallmoney    DEFAULT 350,
  聯絡人      char(10)      DEFAULT 'Nobody'
)
```

用 getdate() 函數取得系統的日期時間，作為接單日期欄位的預設值

← 設定預設的金額

← 設定預設的聯絡人

DEFAULT 值

- **Step 2**

```
INSERT  訂單                          ←—— 只輸入訂單編號, 其它 3 個欄位將採用預設值
(訂單編號)
VALUES
(1)

INSERT  訂單                          ←—— 輸入 3 個欄位的值, 接單日期則採用預設值
(訂單編號，訂購金額，聯絡人)
VALUES
(2，760，'吳明士')

INSERT  訂單                          ←—— 輸入 4 個欄位的值, 不採用預設值
(訂單編號，接單日期，訂購金額，聯絡人)
VALUES
(3，'2016-1-7'，1335，'卜之道' )
```

DEFAULT 值

- **Step 3**

```
SELECT *
FROM 訂單
```

結果 訊息

	訂單編號	接單日期	訂購金額	聯絡人
1	1	2016-11-02 15:37:50.040	350.00	Nobody
2	2	2016-11-02 15:37:50.043	760.00	吳明士
3	3	2016-11-07 00:00:00.000	1335.00	卜之道

從這 3 個欄位就可以比對出使用預設值的差別了

4-7 識別名稱 (IDENTIFIER)

- 識別名稱的表示法
- 特殊的識別名稱

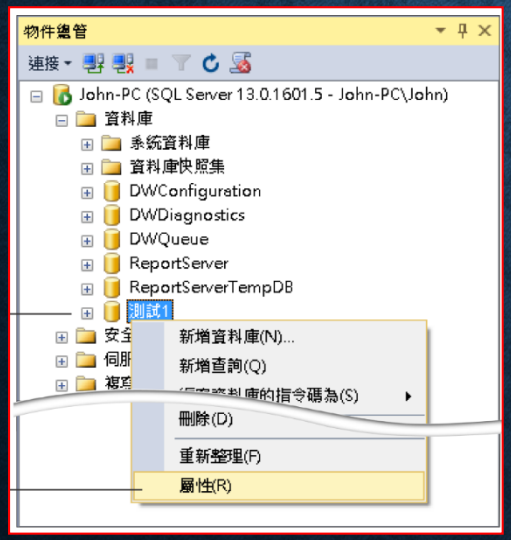
識別名稱的表示法

類別	說　　明
英文字	A-Z 或 a-z, 在 SQL 中是不用區分大小寫的
數字	0 -9, 但數字不得做為識別名稱的第一個字元
特殊字元	_、#、@、$, 但 $ 不得做為識別名稱的第一個字元
特殊語系的合法文字	例如中文字也可做為識別名稱的合法字元

```
SELECT 電話,                  ◄── 可用中文
       [First Name],          ◄── 中間有空白, 要加〔〕
       [1ABC],                ◄── 以數字開頭, 要加〔〕
       [^%*\!(+=)|&]          ◄── 特殊字元, 要加〔〕
FROM   [select]               ◄── select 是關鍵字, 要加〔〕
WHERE  [WHERE] = '台北'       ◄── WHERE 是關鍵字, 要加〔〕
```

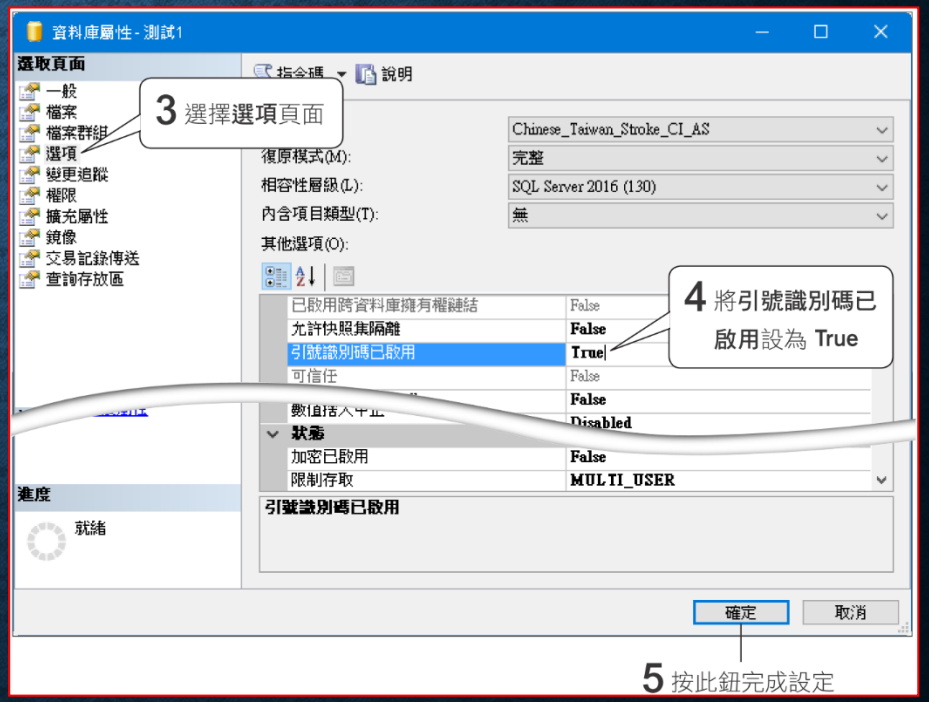
識別名稱的表示法

- 用 " " 代替 [ ]



1 在**物件總管**窗格中選取要設定的資料庫

2 在資料庫上按滑鼠右鈕執行『**屬性**』命令



不建議改變 [ ] 此習慣的用法。

特殊的識別名稱

開頭字元	例如	意義
@	@var	區域變數名稱必須以 @ 開頭
@@	@@ERROR	全域的系統變數是以 @@ 開頭
#	#table	區域的暫存資料表 (或預存程序)
##	##gtable	全域的暫存資料表 (或預存程序)