

資料型別

資料型別

- 定義資料表的欄位、宣告程式中的變數時, 都需要為它們設定一個資料型態, 用意是指定該欄位或變數所存放的資料是整數、字串、貨幣、日期或是其它型別的資料, 以及會用多少空間來儲存資料。

整數

- 用來定義存放整數資料 (如：123、8000) 的欄位或變數，有 bigint、int、smallint、tinyint、bit 五種。其中 bit 雖歸為整數類，但它只能儲存 1、0 及 NULL 這 3 種值。

資料型別	資料範圍	使用的位元數(長度)
bigint	$-2^{63} \sim 2^{63} - 1$ (-9,223,372,036,854,775,808~ 9,223,372,036,854,776,807)	8 bytes
int	$-2^{31} \sim 2^{31} - 1$ (-2,147,483,648 ~ 2,147,483,647)	4 bytes
smallint	$-2^{15} \sim 2^{15} - 1$ (-32,768 ~ 32,767)	2 bytes
tinyint	$0 \sim 2^8 - 1$ (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

- 用來定義帶有小數部份的數值, 如：123.0、8000.56。
- 資料型別有 numeric 與 decimal 兩種;在SQL SERVER中這兩種類型完全一樣。(新版建議使用decimal)
- 使用時, 須指明精確度與小數點位數, 例如 numeric(5,2), 表示總共為 5 位數 (精確度為 5), 其中有 3 位整數及 2 位小數; 若不指定, 則預設為 numeric(18,0)。
- 精確度可指定的範圍為 1 ~ 38, 小數點位數可指定的範圍最少為 0, 最多不可超過精確度。

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

浮點數值

- 數值非常大或非常小時, 可用浮點數的資料型態來取其“近似值”, 例如: 23456646677799 變成 $2.35\text{E}+13$ 。
- 近似浮點數值型別有 float 和 real 兩種。
- 當數值位數超過有效位數的限制時, 則儲存的數值會因四捨五入而產生誤差。

近似浮點數值

資料型別	資料範圍	位元組數
float(n)	-1.79E+308 ~ 1.79E+308，精確度是1~15位數	4或8
real	-3.40E+38 ~ 3.40E+38，精確度是1~7位數	4

- 上列表格中的 E 為科學記號, E+308 代表 10 的 308 次方。
- 使用 float 資料型別, 以科學記號表示數值時, 如：
N E+100, 則 N 最多可表示 15 位數, 若超過 15 位數
則會出現誤差；使用 real 資料型別最多可表示到 7
位數。

日期時間

- 用來儲存日期與時間資料, 如： '1999-04-30 12:20:30' 。
- 2008版後新增date 、 time 、 datetime2 、 datetimeoffset 。

日期時間

資料類型	資料範圍	位元組數
datetime	1753年1月1日 ~ 9999年12月31日，時間精確至毫秒，其值以 .000、.003 或 .007 秒遞增	8
smalldatetime	1900年1月1日 ~ 2079年6月6日，時間可精確至分	4
Date	0001年1月1日 ~ 9999年12月31日	3
time(n)	00:00:00.0000000 ~ 23:59:59.9999999，可精確至100奈秒	3~5
datetime2(n)	0001年1月1日 ~ 9999年12月31日，時間可精確至100奈秒	6~8
datetimeoffset(n)	0001年1月1日 ~ 9999年12月31日，時間可精確至100奈秒（以UTC為單位）	8~10

字串型別

- 用來儲存字串資料,如：‘台北市政府’、‘北市職能學院’等。
- 此類資料型別有四種：char(n)、varchar(n)、varchar(max) 與 text。
- 其中 varchar(n)、varchar(max) 及 text 的實際儲存長度會依資料量而調整,例如 varchar(10) 表示最多可儲存 10 bytes, 但若只填入 5 個字元, 那麼只會佔用 5 bytes。
- 未來可能刪除Text型別。

字串型別

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

- char(n) 與 varchar(n) 最多只能儲存 8000 個字元, 若資料會超過此長度, 請改用 varchar(max) 或 text 型別。
- 使用 char(n) 及 varchar(n) 時須指定字元長度 (n), 例如 char(50)、varchar(50); 若未指定, 則預設為 1。
- varchar(max) 及 text 型別都不必指定長度。

Unicode字串型別

- Unicode 字串,不論中英字,都佔2Bytes。
- 資料型別有 nchar(n)、nvarchar(n)、varchar(max) 與 ntext 四種。
- 未指定指定長度 (n),則預設為 1字元。
- nvarchar(max) 及 ntext 型別都不必指定長度。

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

貨幣

- 用來定義貨幣資料，精確度達金融單位的萬分之一。
- 此類有 money 與 smallmoney 兩種資料型別。

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

- 使用習慣上常會將金額以每 3 個位數加上一個逗號隔開, 以便於閱讀。若以這樣的形式輸入金額 (如：123,456) 會被接受, 但資料表不會將逗號表示出來。

二元碼字串(二進位資料型別)

- 此類型別用來記錄二進位的資料,一般都用16進位來表示; 0x開頭即代表16進位。
- 此類的資料型別有 binary(n)、varbinary(n)、varbinary(max) 與 image。
- varbinary(max) 與 image型別可用來儲存 Word 文件、Excel 試算表、以及點陣圖、GIF、JPEG 等影像檔。

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

標記型別

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 0x000000000000000130	8 bytes
uniqueidentifier	16 bytes 的 16 進位值 如 4C047CB3-B007-11D2-9C59-0080C846994D	16 bytes

TIMESTAMP(ROWVERSION)標記型態

- rowversion (timestamp) 是記錄資料更新的時間戳記, 當某筆記錄有變動時, 該筆記錄的 rowversion 欄位便會自動產生新值, 此值會是整個資料庫的唯一值。
- T-SQL **timestamp** 資料型態不同於 SQL-2003 標準中所定義的 **timestamp** 資料型態。SQL-2003 **timestamp** 資料型態相當於 T-SQL **datetime** 資料型態。
- CREATE TABLE 或 ALTER TABLE 陳述式中, 不需指定 **timestamp** 型態的欄位名稱。若使用 **rowversion**, 則必須指定一個欄位名稱。
- 若要傳回資料庫目前的時間戳記值, 可使用 @@DBTS。

Uniqueidentifier標記型態

- 用來儲存全域唯一識別碼(GUID, Global Unique Identifier)的資料。
- Uniqueidentifier欄位可用在各資料庫之間識別記錄的唯一性。
- 資料格式為16Bytes的16進位值，如DB2248A9-F4A6-4FAE-BB8F-8EE4D78DEF96
- 可透過NEWID()來產生GUID的值。

Sql_variant資料型態

- 凡使用sql_variant型態的欄位、參數或變數，均可以儲存不同資料型態的記錄。
- text、ntext、image、timestamp、sql_variant等型態則除外。
- 使用SQL_VARIANT_PROPERTY取得資料記錄的資料型態。

語法如下：

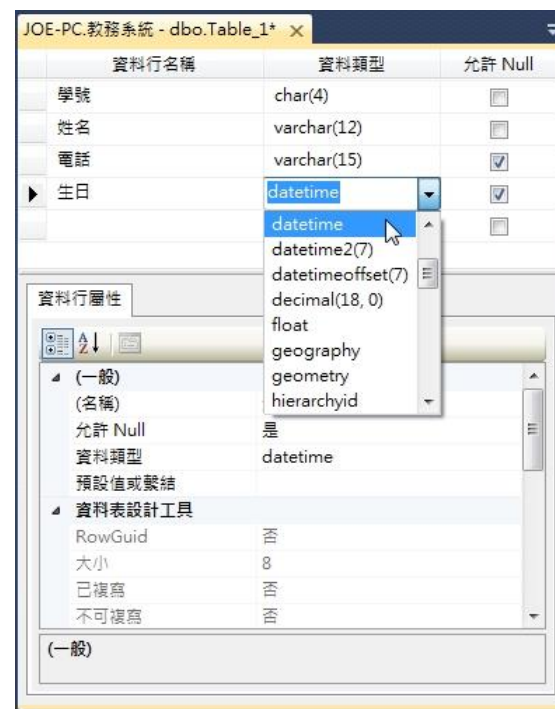
SQL_VARIANT_PROPERTY
(*expression* , *property*)

Property	說明
BaseType	基本資料型別
Precision	精確度
Scale	小數點位數
TotalBytes	保存此sql_variant資料所需的位元數
MaxLength	資料的最大長度

資料表的建立與維護

建立資料表(Create Table)

- 使用SQL Server Management Studio：SSMS 管理工具建立
 - Management Studio提供圖形編輯畫面來建立資料表的定義資料
- 使用DDL語法：Create Table建立



定義資料表的欄位屬性

屬性	說明
允許 Null	欄位值是否可以是 NULL 空值
長度	欄位資料的長度，以位元組為單位，因為有些資料類型是固定長度，所以不一定可以設定此屬性
資料類型	欄位儲存資料的資料類型
預設值或繫結	指定欄位的預設值，當新增記錄時，如果沒有輸入資料，就是填入此預設值
整數位數	指定資料類型 decimal 和 numeric 欄位的整數位數
小數位數	指定資料類型 decimal 和 numeric 欄位的小數位數
RowGuid	指定資料類型 uniqueidentifier 欄位是否自動產生全域唯一識別碼，即在【預設值或繫結】屬性使用 NEWID()函數產生識別碼
計算資料行規格/公式	指定計算欄位的運算式

定義資料表的欄位屬性

屬性	說明
定序	指定欄位的定序，預設為資料庫的定序設定，只有 char、varchar、text、nchar、nvarchar 和 ntext 資料類型可以更改
描述	欄位說明文字
識別規格/(為識別)	指定欄位值是否自動編號
識別規格/識別值種子	指定自動編號的起始值，預設為 1
識別規格/識別值增量	指定自動編號的遞增值，預設為 1

使用DDL語法：Create Table建立

CREATETABLE

[databasae_name.[schema]. | schema.] table_name ← 設定資料表名稱

({ <column_definition> ← 定義欄位屬性與條件約束


| column_name AS computed_column_expression ← 定義計算欄位

| <table_constraint> } ← 設定資料表條件約束

[, ...n])

[ON { filegroup | "default" }]

[TEXTIMAGE_ON {filegroup | "default"}]

 指定存放資料表資料
的檔案群組

CREATE TABLE [訂貨明細](

[訂單號碼] int NOT NULL,

[產品編號] int NOT NULL,

[原價] money NOT NULL,

[數量] smallint NOT NULL,

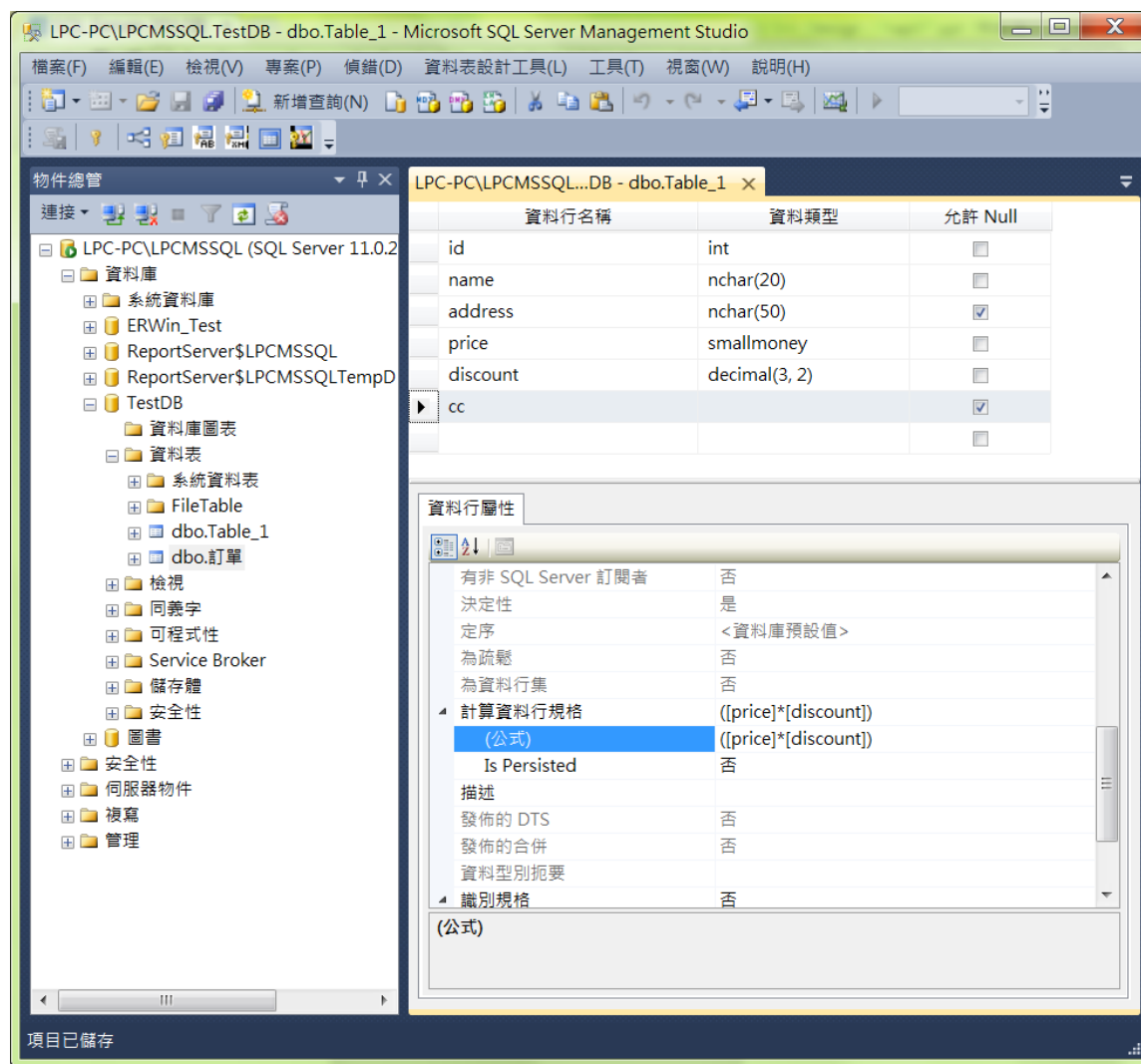
[折扣] decimal(3,2) NOT NULL

)

建立計算欄位-說明

- 計算欄位值是使用同一筆記錄的其他欄位運算而來。
- 欄位並沒有真正儲存資料，是一種虛擬欄位
- 因為沒有存入值，所以不能指定DEFAULT、NOT NULL、NULL等欄位屬性和條件約束。
- 如果計算欄位值是唯一值且不會更動，則可指定成PRIMARY KEY和UNIQUE欄位，不過很少會如此設定。

建立計算欄位



建立計算欄位

- 語法建立時，須配合使用 AS 關鍵字

```
CREATE TABLE [dbo].[訂貨明細]( [訂單號碼] int NOT NULL,  
[產品編號] int NOT NULL,  
[單價] money NOT NULL,  
[數量] smallint NOT NULL,  
[折扣] decimal(2,2) NOT NULL,  
[小計] AS [單價]*[數量]*(1-[折扣])  
) ON [PRIMARY]
```

條件約束(Constraint)

- 目的：減少輸入錯誤資料的機率、維護資料的完整性。

條件約束	欄位層級	資料表層級
NOT NULL	指定欄位不可是空值	N/A
PRIMARY KEY	指定單一欄位的主鍵	指定一到多欄位集合的主鍵
UNIQUE	指定單一欄位值是唯一值	指定一到多欄位集合的值是唯一值
CHECK	指定單一欄位值的範圍	指定多欄位值的範圍
FOREIGN KEY / REFERENCES	指定單一欄位的外來鍵，即建立關聯性	指定一到多欄位集合的外來鍵，即建立關聯性

條件約束(Constraint)

- **Primary key** :
 - 每一資料表只能有一個主鍵。
 - 主鍵不可為NULL，且必需是唯一不可重覆的。
 - IMAGE和TEXT欄位不能做主鍵
 - 主鍵具有索引功能，可用來加快查詢速度。

```
[CONSTRAINT constraint_name]
```

```
PRIMARY KEY (欄位名稱1[,欄位名稱2[,...欄位名稱16]])
```

```
[ON 檔案群組名稱 | DEFAULT]
```

條件約束(Constraint)

- **NULL、NOT NULL**：當資料表欄位一定要輸入資料時，欄位可限制為 NOT NULL。若允許不輸入資料，欄位則可設定為 NULL (通常資料表欄位預設為允許 NULL)。
- **DEFAULT**：當欄位設定 DEFAULT 條件約束時，該欄位若未輸入資料，將自動以預設值填入欄位中。(DEFAULT值可以是常數、系統函數或NULL值)

[CONSTRAINT constraint_name] DEFAULT 內定值

條件約束(Constraint)

- **UNIQUE**：某欄位不允許出現重複值, 也就是每個欄位值必須唯一, 則可為該欄位設定 UNIQUE 條件約束。UNIQUE 的欄位允許輸入 NULL 值, 但為保持唯一性, 最多只能出現一個 NULL 值。

```
[CONSTRAINT constraint_name]
    UNIQUE [CLUSTERED | NONCLUSTERED]
    [(欄位名稱1 [,欄位名稱2 [...,欄位名稱16]])]
    ON 檔案群組名稱
```

UNIQUE 與 Primary key 的差異

- UNIQUE 欄位允許輸入 NULL 值, 但 Primary key 欄位則不允許。
- 一個資料表中可以定義多個 UNIQUE 條件約束, 但只能定義一個 Primary key 條件約束。

條件約束(Constraint)

- **CHECK**：CHECK 條件約束可用來設定輸入值檢查條件，當資料新增或異動時,輸入的欄位值必須符合該檢查條件。例如使用CHECK檢查某欄位的輸入輸入值是否介於 1 到 100 之間的整數。

```
[CONSTRAINT constraint_name]  
CHECK [NOT FOR REPLICATION] (條件運算式)
```

註：一個欄位可以設定多個 CHECK 條件約束,而一個 CHECK 條件約束也可以針對多個欄位做檢查。

條件約束(Constraint)

- **Foreign key**：限制欄位的值必須是來自於其所參考到的資料表。當在 Foreign key 的欄位中輸入 NULL 值或該值不在其所參考資料表中時，則該筆輸入會被拒絕，如此可避免造成資料關聯的不完整。

```
[CONSTRAINT constraint_name]
```

```
FOREIGN KEY [(欄位名稱1 [,欄位名稱2 [...,欄位名稱16]])]
```

```
REFERENCES 參考資料表
```

```
[(參考欄位1 [,參考欄位2 [...,參考欄位16]])]
```


維護資料完整性的對策

完整性種類	可用的對策
實體完整性 (Entity Integrity) 維持每筆記錄的唯一性	Primary key UNIQUE IDENTITY (識別屬性)
區域完整性 (Domain Integrity) 維持欄位資料的正確性	DEFAULT (預設值或繫結屬性) Foreign key CHECK NOT NULL
參考完整性 (Referential Integrity) 資料表間關聯的完整性	Foreign key CHECK
使用者定義的完整性 (User-defined) 我們自訂的資料完整性	所有的條件約束 預存程序 (Stored procedures) 觸發程序 (Triggers)

- 查詢資料表設定的條件約束

```
EXEC sp_helpconstraint table_name
```

ALTER TABLE敘述

ALTER TABLE table

← 指定欲進行修改的資料表名稱

[ALTER COLUMN column_name

{ new_data_type [(precision [, scale])]

[COLLATE collation_name]

[NULL | NOT NULL]

| { ADD | DROP } ROWGUIDCOL }]

修改欄位屬性

| ADD

{ [<column_definition>]

| column_name AS computed_column_expression } [...n]

新增欄位或新增計算欄位

| [WITH CHECK | WITH NOCHECK] ADD

{ < table_constraint > } [...n]

新增資料表條件約束

| DROP

{ [CONSTRAINT] constraint_name

| COLUMN column } [...n]

刪除條件約束或刪除欄位

| { CHECK | NOCHECK } CONSTRAINT

{ ALL | constraint_name [...n] }

啟動或關閉條件約束

修改欄位屬性

- 修改欄位屬性的語法如下：

```
ALTER TABLE table
  [ ALTER COLUMN column_name
    { new_data_type [ ( precision [, scale] ) ]
    [ COLLATE collation_name ]
    [ NULL | NOT NULL ]
    | {ADD | DROP } ROWGUIDCOL } ]
```

- **column_name**：指定欲修改屬性的欄位名稱。
- **new_data_type (precision [, scale])**：為欄位指定新的資料型別，若是 decimal、numeric 型別，則還需指定**精確度** (precision) 和**小數點位數** (scale)。

修改欄位屬性

- **ADD | DROP ROWGUIDCOL**：為欄位加上 (ADD ROWGUIDCOL) 或移除 (DROP ROWGUIDCOL) **RowGuid** 屬性。注意, 只有 **uniqueidentifier** 型別的欄位可以設定為 **RowGuid** 屬性；同時, 一個資料表中也只能有一個具備 **RowGuid** 屬性的欄位。
- 假設**客戶**資料表的**地址**欄位, 其屬性原為 **nvarchar(50)** 型別、NOT NULL, 現變更為 **nvarchar(100)**、NOT NULL：

```
ALTER TABLE 客戶
```

```
    ALTER COLUMN 地址 nvarchar(100) NOT NULL
```

不能變更屬性的欄位

- 已具備 RowGuid 屬性的欄位。
- 計算欄位或計算欄位用到的欄位。
- 用於 PRIMARY KEY 或 FOREIGN KEY 條件約束中的欄位。
- 用於 CHECK 或 UNIQUE 條件約束中的欄位, 但可變更這些欄位的長度 (前提是該欄位為可變動長度的型別)。
- 用於 DEFAULT 中的欄位, 但若沒有變更此欄位的資料型別, 則可更改欄位的長度、精確度及小數點位數。

變更型別的注意事項

- 新資料型別必須與原資料型別相容, 亦即能夠進行**隱含式轉換**。
- 不能變更為 timestamp 型別。
- 若變更屬性的欄位原就具備**識別規格**中的屬性, 新資料型別也必須要能夠支援該屬性。

更改欄位或資料表名稱

- 使用ALTER TABLE是無法變更欄位或資料表名稱，必需使用 sp_rename 預存程序。
- sp_rename 預存程序可用來變更資料庫中所有非系統建立的物件名稱。
- 語法如下：

```
sp_rename 'object_name', 'new_name' [, 'object_type']
```

↑
原物件名稱

↑
新物件名稱

↑
物件型態

更改欄位或資料表名稱

- Object_type說明：

object_type	說 明
COLUMN	變更欄位名稱
DATABASE	變更資料庫名稱
INDEX	變更使用者自訂的索引名稱
OBJECT	變更條件約束 (CHECK、FOREIGN KEY、PRIMARY KEY、UNIQUE)、資料表、檢視表、預存程序、觸發程序的名稱
USERDATATYPE	變更使用者自訂資料型別物件的名稱

- 範例：

```
EXEC SP_RENAME 'FK__訂貨明細__產品編號__2F10007B',  
               'FK_訂貨明細_產品編號_1','OBJECT'
```

```
EXEC SP_RENAME '供應商','廠商'
```


新增欄位 / 計算欄位

```
ALTER TABLE table
```

```
ADD
```

```
{ [ <column_definition> ]
```

← 新增一般欄位

```
| column_name AS computed_column_expression }
```

← 新增計算欄位

```
[,...n ]
```

```
< column_definition > ::= { column_name data_type }
```

```
[ [ DEFAULT constant_expression ] [ WITH VALUES ]
```

```
| [ IDENTITY [ (seed, increment) [ NOT FOR REPLICATION ] ] ] ]
```

```
[ ROWGUIDCOL ]
```

```
[ COLLATE collation_name ]
```

```
[ < column_constraint > ] [ ...n ]
```

新增資料表條件約束

```
ALTER TABLE table
    [ WITH CHECK | WITH NOCHECK ] ADD
        { < table_constraint > } [...n ]

< table_constraint > ::= [ CONSTRAINT constraint_name ]
    { [ { PRIMARY KEY | UNIQUE }
        [ CLUSTERED | NONCLUSTERED ]
        { ( column [ ,...n ] ) }
        [ WITH FILLFACTOR = fillfactor ]
        [ ON { filegroup | "default" } ] ]
    | FOREIGN KEY
        [ ( column [ ,...n ] ) ]
        REFERENCES ref_table [ ( ref_column [ ,...n ] ) ]
        [ ON DELETE { CASCADE | NO ACTION } ]
        [ ON UPDATE { CASCADE | NO ACTION } ]
        [ NOT FOR REPLICATION ]
    | DEFAULT constant_expression
        [ FOR column ] [ WITH VALUES ]
    | CHECK [ NOT FOR REPLICATION ]
        ( logical_expression ) }
```

新增資料表條件約束

- WITH CHECK 表示要套用新增的條件約束去檢查舊有記錄，WITH NOCHECK 表示不檢查舊有記錄。若省略則預設為 WITH CHECK。
- 範例如下：

```
ALTER TABLE [客戶] WITH CHECK ADD  
    CONSTRAINT CK_客戶電話  
    CHECK (電話 IS NOT NULL)
```

刪除條件約束 / 欄位

- 語法：

```
ALTER TABLE table  
  DROP
```

```
    { [ CONSTRAINT ] constraint_name
```

← 刪除條件約束, 指定欲刪除的條件約束名稱即可

```
    | COLUMN column } [...n ]
```

← 刪除欄位, 指定欲刪除的欄位名稱即可

- 刪除條件約束

```
ALTER TABLE 訂貨明細
```

```
  DROP CONSTRAINT PK_訂貨明細
```

- 刪除欄位

```
ALTER TABLE 訂貨明細
```

```
  DROP COLUMN 訂單號碼, 產品編號
```

暫時關閉條件約束

- 只能用在Check與Foreign Key
- 語法說明:

```
ALTER TABLE table  
    { CHECK | NOCHECK } CONSTRAINT  
        { ALL | constraint_name [...n] }
```

- **CHECK | NOCHECK**：CHECK 表示啟動條件約束，NOCHECK 表示關閉條件約束。
- **ALL**：指資料表中所有的 FOREIGN KEY 與 CHECK 條件約束。

暫時關閉條件約束

- 範例:

建立CONSTRAINT:

```
ALTER TABLE [訂貨明細] WITH CHECK ADD  
    FOREIGN KEY ([產品編號])  
    REFERENCES [產品資料] ([產品編號])  
    NOT FOR REPLICATION
```

停用CONSTRAINT:

```
ALTER TABLE [訂貨明細] NOCHECK  
CONSTRAINT FK__訂貨明細__產品編號__2E1BDC42
```

啟用CONSTRAINT:

```
ALTER TABLE [訂貨明細] CHECK  
CONSTRAINT FK__訂貨明細__產品編號__2E1BDC42
```

刪除資料表

- 刪除資料表使用DROP TABLE，語法：

```
DROP TABLE table_name
```

- 範例：

```
USE dbTest2  
DROP TABLE tb1
```
- 刪除的資料表是 FOREIGN KEY 所參照的資料表，需先刪除該資料表的關聯，否則無法使用DROP TABLE 敘述。

資料増刪修

INSERT語法

- 語法結構：

```
INSERT [INTO] table_name [column_list] VALUES  
({default | NULL | 運算式}[,...n]) | 衍伸資料表 |  
執行預存程序或動態字串
```

INTO：增加整個敘述的可讀性,可省略。

- **table_name**：要新增記錄的資料表名稱。
- **column_list**：列出預備要輸入值的欄位名稱,欄位名稱之間請用逗號相隔。此處若沒有指定任何欄位,則表示資料表中的所有欄位。
- **data_values**：列出要填入欄位中的值,值與值之間須用逗號隔開。

UPDATE 敘述

- UPDATE 敘述可以更新一筆或多筆記錄, 語法如下：

```
UPDATE table_name
```

```
SET { column_name = { expression | DEFAULT | NULL} } [...n]
```

```
[ WHERE search_condition ]
```

- **SET** 子句用來設定欄位的新值。
- **column_name**：指定更新的欄位名稱。
- **expression**：指定新的內容值, 內容可以是常數、運算式、變數 ... 等。
- **DEFAULT**：可將欄位值重新設為預設值。
- **NULL**：將欄位值重新設為 NULL。

UPDATE 敘述

- 範例：

```
USE NorthwindC
UPDATE dbo.訂貨明細 SET 單價=20
WHERE 訂單號碼=10590 AND 產品編號=2
```

```
USE NorthwindC
UPDATE dbo.訂貨明細 SET 單價=20 ,折扣=1
WHERE 訂單號碼=10590 AND 產品編號=2
```

DELETE 敘述

- DELETE刪除資料，可同時刪除一筆或多筆資料。
- DELETE [FROM] {資料表名稱 | 檢視名稱}
[WHERE 條件]

DELETE FROM 產品說明 WHERE 產品編號=1

DELETE 產品說明 WHERE 產品類別='調味品'

DELETE 產品說明

TRUNCATE敘述

- TRUNCATE TABLE 用來刪除資料表記錄，刪除過程取消配置用來儲存資料表資料的資料頁，以移除資料，交易記錄只會記錄頁面的取消配置；不同於DELETE敘述會將逐筆刪除的資料記錄到交易記錄檔中。
- TRUNCATE TABLE 只會移除資料表中的所有資料，至於資料表欄位結構、條件約束、索引等全都保留，不會移除。
- 語法：
`TRUNCATE TABLE table_name`
- 範例：
`TRUNCATE TABLE 產品說明`