DML語法-SELECT的基本查詢

SELECT查詢指令

- SELECT指令是DML指令中語法最複雜的一個，其基本語法如下所示：

  SELECT 欄位清單
  FROM 資料表來源
  [WHERE 搜尋條件]
  [GROUP BY 欄位清單]
  [HAVING 搜尋條件]
  [ORDER BY 欄位清單]

SELECT查詢指令-說明

- SELECT指令各子句的說明，如下表所示：

子句	說明
SELECT	指定查詢結果包含哪些欄位
FROM	指定查詢的資料來源是哪些資料表
WHERE	過濾查詢結果的條件，可以從資料表來源取得符合條件的查詢結果
GROUP BY	可以將相同欄位值的欄位群組在一起，以便執行群組查詢
HAVING	搭配GROUP BY子句進一步過濾群組查詢的條件
ORDER BY	指定查詢結果的排序欄位

SELECT查詢語法的邏輯順序

(8) SELECT (9) [DISTINCT]

(12) [Top n] 傳回列表 [INTO 新資料名稱]

(1) FROM 資料表

(3) [INNER | LEFT | RIGHT] JOIN 資料表

(2) ON <資料表 JOIN 的條件>

(4) [WHERE <過濾條件>]

(5) [GROUP BY <群組語法>]

(6) [WITH {CUBE | ROLLUP}]

(7) [HAVING <過濾條件>]

(10) [UNION <連結資料集>]

(11) [ORDER BY <排序列表> [ASC | DESC]]

SELECT子句-語法

- SELECT子句是指定查詢結果包含哪些欄位，其語法如下所示：

  SELECT [ALL | DISTINCT]
  [TOP n | PERCENT] [WITH TIES]
  欄位規格 [[AS] 欄位別名] [, 欄位規格 [[AS] 欄位別名]]

- 上述ALL是預設值可以顯示所有記錄的欄位值，DISTINCT只顯示不重複欄位值的記錄，TOP關鍵字可以顯示查詢結果的前幾筆記錄或多少百分比。

SELECT子句-欄位規格

- 欄位規格（Column Specification）指查詢結果的欄位清單，可以使用AS關鍵字指定欄位別名。
- 欄位規格可以是資料表欄位或計算值的運算式。

欄位規格	說明
資料表的欄位（Base Table Column）	即資料表的欄位名稱清單，或使用「*」符號代表所有欄位
計算值欄位（Calculated Value Column）	算術運算子、字串或函數組成的運算式欄位

- 例如: SELECT * FROM NorthwindC.dbo.員工
  SELECT * FROM 員工
  SELECT 員工編號 , 姓名 , 職稱 FROM 員工

欄位別名

- 透過AS子句可指派不同的名稱或別名給結果集資料行、或衍生的資料行,以增加可讀性。
- 範例：

```
SELECT 類別名稱 AS 產品類別,說明 AS 類別描述
FROM dbo.產品類別
```

```
SELECT 類別名稱 產品類別,說明 類別描述
FROM dbo.產品類別
```

計算值欄位

- SELECT子句的欄位規格可以是算術運算子、字串或函數來組成運算式欄位。
- 計算值欄位沒有欄位名稱，可以使用AS來指定欄位的別名。

```
SELECT [單價]*[數量] AS [價格] FROM [訂貨明細]
```

```
SELECT 姓名+' ' +職稱 AS [公司成員] FROM [員工]
```

```
SELECT LEFT(姓名,1) + 職稱 + ' `+ RIGHT(姓名 , len(姓名)-1) AS [公司成員] FROM 員工
```

移除重複的資料列(DISTINCT)

- DISTINCT可將重複的資料列從SELECT陳述式的查詢結果中消除。
- Null值視同彼此重覆。

```
SELECT    DISTINCT    [單價]
FROM      dbo.產品資料
```

使用運算式

- 算術運算：

```
SELECT    [產品],[單價] * [庫存量]  '總金額'
FROM      dbo.產品資料
ORDER BY  2
```

- 日期運算：

```
SELECT GETDATE() '今天'  , GETDATE()+1 '明天'  , GETDATE()-1 '昨天'
```

使用運算式

- 字串運算：

```
SELECT   姓名+’‘+稱呼 [尊稱]
FROM     dbo.員工
```

- 舊版語法：

```
SELECT  尊稱= 姓名+’‘+稱呼
FROM    dbo.員工
```

- 註：運算元需具有相同的資料類型。

組合不同類型的資料

- 運算子左右兩邊運算元資料類型不同時，則需要自行使用CONVERT函數轉換。
- 範例：

```
SELECT  14+姓名  [尊稱]
FROM    dbo.員工
```

錯誤範例

```
SELECT CONVERT(nvarchar,14)+姓名   尊稱
FROM   dbo.員工
```

正確範例

```
SELECT '14'+姓名   [尊稱]
FROM   dbo.員工
```

正確範例

Where 子句

- 透過Where子句的輔助，可以更精準的得到我們要的資料。
- 當資料欄位屬性有下列幾種類型時，請記得前後加上單引號：

  char、nchar、varchar、nvarchar、text、datetime、smalldatetime。
- 使用SELECT子句查詢資料時，應盡量避免使用萬用字元(*) 。

Where 子句

- Where可以搭配不同的比較運算子或關鍵字來限制查詢條件，如右圖表：

運算子	意義
=	等於
>	大於
<	小於
>=	大於或等於
<=	小於或等於
<>	不等於
!=	不等於(非SQL-92標準)
!<	不小於(非SQL-92標準)
!>	不大於(非SQL-92標準)

Where 子句範例

- 使用比較運子來設定查詢的條件。

```
SELECT * FROM dbo.員工 WHERE 職稱= '業務'
```

```
SELECT 員工編號,姓名,職稱 FROM dbo.員工
WHERE year(雇用日期)<=1993
```

- 除了text、ntext、image等資料類型的運算式，其他均可使用比較運算子。

```
SELECT *
FROM dbo.產品類別
WHERE 圖片='0x151C2F0002000000'
```

錯誤範例

LIKE關鍵字

- LIKE關鍵字可以搜尋符合指定模式的字元字串、日期或時間數值。

萬用字元	意義
%	任何具有零或多個字元的字串
_	任何單一字元
[-]	符合括號內「-」字元範圍的任何一個字元，例如：[A-J]
[]	指定範圍中的任何單一字元(例如[a-f]或集合[abcdef])
[^]	不在指定範圍中的任何單一字元(例如[^a-f]或集合[^abcdef])

LIKE關鍵字

- 萬用字元與字元字串需括在單引號中。

LIKE ‘Mc%’	搜尋以字母Mc開頭的所有字串(例如：McBadden)
LIKE ‘%inger’	搜尋以字母inger結尾的所有字串(例如：Ringer)
LIKE ‘%en%’	搜尋字串中任意位置包含字母en的所有字串(例如：Bennet、McBadden)
LIKE ‘_heryl’	搜尋以字母heryl結尾、且長度為六個字母的所有字串(例如：Cheryl、Sheryl)
LIKE ‘[CK]ars[eo]n’	搜尋Carsen、karsen…等字串
LIKE ‘[M-Z]inger’	搜尋以字母inger結尾，並以M到Z之間任何單一字母開頭的所有字串(例如Ringer)
LIKE ‘M[^c]%’	搜尋以字母M開頭，但並未以字母c作為第二個字母的所有字串(例如：MacFeather)

範例:

```
SELECT    *
FROM      dbo.員工
WHERE     姓名 like '陳%'
```

```
SELECT    *
FROM      dbo.員工
WHERE     地址 like '%北平東路%'
```

使用邏輯運算子

- AND：判斷兩個布林運算式都為true時，才傳回true。
- OR：判斷兩個布林運算式，其中任一個為true，傳回true。
- NOT :反轉任何布林運算的值。
- 邏輯運算子優先順序：

  NOT > AND > OR

使用邏輯運算子

- 找出女姓業務主管

```
SELECT   *
FROM     dbo.員工
WHERE    職稱= ‘業務主管’   AND  稱呼= ‘小姐’
```

- 找出牛奶及汽水的產品資料

```
SELECT   *
FROM     產品資料
WHERE    產品= ‘牛奶’   OR  產品= ‘汽水’
```

邏輯運算子-連接多個條件與括號

- WHERE子句可以使用AND和OR來連接多個不同條件。
- 優先順序是位在括號中運算式優先。

```
SELECT   *
FROM     dbo.產品資料
WHERE    庫存量<=安全存量  and  類別編號=1  or  類別編號=3
```

```
SELECT   *
FROM     dbo.產品資料
WHERE    庫存量<=安全存量  and  (類別編號=1 or 類別編號=3)
```

IN關鍵字

- IN關鍵字可以用來判斷指定的值是否符合子查詢或清單中的任何值。
- 例如：列出來自台北、 新竹 、高雄城市的供應商。

```
SELECT  *
FROM    dbo.供應商
WHERE   行政區= ‘台北’ OR 行政區= ‘新竹’ OR 行政區= ‘高雄'
```

```
SELECT  *
FROM    dbo.供應商
WHERE  行政區 IN (‘台北’,’新竹’,’高雄’)
```

IN關鍵字

- IN關鍵字搭配NOT邏輯運算子

```
SELECT  *
FROM     dbo.供應商
WHERE  行政區 NOT IN (‘台北’,‘新竹’,‘高雄’)
```

NULL值

- Null值表示未知的值。
- Null值與空值或零值不同。
- 兩個Null值永遠不會相等。
- 測試Null，需在Where子句中使用IS NULL或IS NOT NULL
- 將某個欄位的資料清成NULL，可使用快速鍵〔Crtl-0〕或只直接鍵入NULL(大寫不加引號)

NULL值

- Null值不得用於主索引鍵或外部索引鍵
- 新增一筆記錄時，若欄位沒有輸入資料、沒有預設值及Default條件約束時，該欄位自動填入NULL值
- 範例語法：

```
SELECT   *
FROM     dbo.員工
WHERE    相片  IS  NOT  NULL
```

ISNULL()函數

- 查詢資料表時，如果有欄位值是NULL值時，可以使用ISNULL()函數來輸出替代值，其語法如下所示：

  ISNULL(檢查運算式, 替代值)

- 上述語法的檢查運算式可以檢查運算式是否為NULL空值，如果是，就以替代值輸出。
- 函數中的兩個參數類型需相同，如果不同，需使用CAST(欄位名稱 AS 類型) 來進行轉換。

```
SELECT    員工編號, 姓名 ,
          ISNULL(國家地區 ,'未輸入') 國家地區
FROM      dbo.員工
```

聚合函數-說明

- 聚合函數（Aggregate Functions）可以進行選取記錄欄位值的筆數、平均、範圍和統計函數，以便提供進一步欄位資料的分析結果。
- SELECT指令敘述擁有聚合函數稱為摘要查詢（Summary Query）。常用的聚合函數說明，如右表所示：

函數	說明
COUNT(運算式)	計算記錄筆數
AVG(運算式)	計算欄位平均值
MAX(運算式)	取得記錄欄位的最大值
MIN(運算式)	取得記錄欄位的最小值
SUM(運算式)	取得記錄欄位的總計

聚合函數

聚合函數	語法定義	使用時之限制
MIN MAX	MIN([ALL|DISTINCT]運算式) MAX([ALL|DISTINCT]運算式)	不可使用於bit資料型態會忽略Null值 字元資料，MAX尋找定序順序最高值，MIN反之。 DISTINCT對MIN、max沒有意義
SUM AVG	SUM([ALL|DISTINCT]運算式) AVG([ALL|DISTINCT]運算式)	只能使用在下列型態： Int、smallint、tinyint、decimal、numeric、float、real、money、smallmoney
COUNT	COUNT([ALL|DISTINCT]運算式)	使用COUNT需注意NULL值的影響

聚合函數

- 計算NorthwindC資料庫[訂貨明細]資料表產品編號51的訂購總數、訂單筆數及最小最大訂購數量。

```
SELECT   SUM(數量) '訂購總數',
         COUNT(訂單號碼) '訂單筆數',
         AVG(數量) '平均數量',
         MIN(數量) '單筆訂購最小值',
         MAX(數量) '單筆訂購最大值'
FROM     dbo.[訂貨明細]
WHERE    產品編號=51
```

Group By子句

- Group By可將多個資料列結合成單一個資料列，搭配使用彙總函數計算分組後的結果。
- Group By可以配合聚合函數進行資料統計
- 同時有Where子句與Group By子句時，Where子句一定要放在Group By子句之前。

函數	進行的資料統計
AVG()函數	計算各群組的平均
SUM()函數	計算各群組的總和
COUNT()函數	計算各群組的記錄數

Group By子句

- Group By子句之後列出的欄位最好不要含有Null值。
- text、ntext、和 image 資料型別的欄位, 不能作為 GROUP BY 子句中的分組依據。
- GROUP BY 子句中不能使用欄位別名。
- 範例：

```
SELECT  職稱,COUNT(*)
FROM     dbo.員工
GROUP BY 職稱
```

Having子句

- HAVING 子句也可以設定查詢的條件, 但一般會和 GROUP BY 子句搭配使用。
- 查詢中沒有使用 GROUP BY 子句, 則 HAVING 子句的用途和 WHERE 子句的用途相似, 不過 HAVING 子句和 WHERE 子句還是有差別的, 即彙總函數無法在 WHERE 子句中使用, 只能用在 HAVING 子句中。

Having子句

範例：查詢NorthwindC資料庫[員工]資料表相同職稱四個以上的稱謂

```
SELECT   職稱,COUNT(*) '人數'
FROM     dbo.員工
GROUP BY  職稱
HAVING  COUNT(*)>4
```

WITH ROLLUP運算子

- WITH ROLLUP 只會依據 GROUP BY 子句所列的第一個欄位做加總小計運算
- 範例：

```
SELECT    產品編號, 單價 , SUM(數量)  [總數量]
FROM      dbo.[訂貨明細]
WHERE     產品編號 IN (50,51)
Group By  產品編號,單價
WITH ROLLUP
```

WITH CUBE運算子

- WITH CUBE 是對 GROUP BY 子句列出的每個欄位都做加總小計運算
- 範例：

```
SELECT    產品編號, 單價,SUM(數量) [總數量]
FROM      dbo.[訂貨明細]
WHERE     產品編號 IN (50,51)
Group By 產品編號,單價
WITH CUBE
```

GROUPING SETS子句

- SQL Server 2008版擴充GROUP BY子句的功能，新增GROUPING SETS子句
- 使用者自行定義傳回哪些欄位的聚合統計資料

```
SELECT    產品編號, 單價,SUM(數量) [總數量]
FROM      dbo.[訂貨明細]
WHERE     產品編號 IN (50,51)
Group By  Grouping sets((產品編號,單價),產品編號)
```

```
SELECT    產品編號, 單價,SUM(數量) [總數量]
FROM      dbo.[訂貨明細]
WHERE     產品編號 IN (50,51)
Group By  Grouping sets((產品編號,單價),產品編號,())
```

排序資料(ORDER BY)

- 部份情況下，資料的排列會以資料建立的時間為依據存檔。
- 查詢的結果集是以隨機方式排列
- ORDER BY子句項目數沒有限制
- 排序可以遞增（ASC）或遞減（DESC）
- 預設採遞增排序，若排序內容為NULL值，則當作最小值處理。
- 不能使用於ntext、text 、image 、xml型態

排序資料(ORDER BY)

```
SELECT 類別編號 [產品類別代號], 產品 [產品名稱],
       單價 [產品單價]
FROM   dbo.產品資料
ORDER BY 單價 ASC
```

```
SELECT 類別編號 [產品類別代號], 產品 [產品名稱],
       單價 [產品單價]
FROM   dbo.產品資料
ORDER BY [產品單價] ASC
```

```
SELECT 類別編號 [產品類別代號], 產品 [產品名稱],
       單價 [產品單價]
FROM   dbo.產品資料
ORDER BY 3
```

列出前N筆資料

- 透過TOP N關鍵字可以傳回資料表中前N筆資料。
- 若再加上 PERCENT, 即 TOP n PERCENT, 則表示查詢前面 n 百分比的記錄, 此時 n 的值可以從 0 到 100。
- 搭配WITH TIES使用,可將排序資料平手的狀況一併列出，但必需搭配ORDER BY 子句使用。
- 通常會搭配ORDER BY 進行資料排序才有意義。

範例：

- 列出前 5 筆訂貨資料。

```
SELECT    Top 5 *
FROM      dbo.[訂貨主檔]
```

- 列出前10%筆訂貨資料。

```
SELECT    Top 10 Percent  *
FROM      dbo. [訂貨主檔]
```

- 列出訂貨明細數量最高的前 5 筆資料

```
SELECT  TOP (5) with ties *
FROM  dbo.[訂貨明細]
ORDER by 數量 desc
```

OFFSET和FETCH NEXT分頁查詢

- OFFSET子句可以指定位移幾筆記錄後開始傳回查詢結果，其語法如下：

  OFFSET 整數常數或運算式 ROW | ROWS

- 上述語法的位移量可以是整數常數，例如：5或10等，或一個傳回大於0整數值的運算式，最後的ROW或ROWS關鍵字是同義詞，任選一個使用，其目的是為了和ANSI相容。

```
SELECT    訂單號碼,員工編號
FROM      訂貨主檔
ORDER BY  訂單編號 OFFSET  10  ROW
```

OFFSET和FETCH NEXT分頁查詢

- FETCH NEXT寫在OFFSET之後，可以指定傳回位移之後的幾筆記錄，其語法如下所示：

FETCH FIRST | NEXT 整數常數或運算式 ROW | ROWS ONLY

- 上述語法的FIRST和NEXT是同義詞，可以任選一個使用，傳回的筆數是整數常數、運算式或子查詢，ROW或ROWS關鍵字也是同義詞，請任選一個使用。

```
SELECT    訂單號碼,員工編號
FROM      訂貨主檔
ORDER BY  訂單編號  OFFSET  10  ROW  FETCH NEXT  5  ROWS  ONLY
```

註:同一個查詢或子查詢中，TOP與 OFFSET 不能共用

多資料表查詢

多資料表查詢

- 合併查詢（Join Query）：其主要目的是將正規化分割的資料表，還原成使用者習慣閱讀的資訊。因為正規化的目的是避免資料重複，擁有重複資料的資訊反而易於使用者閱讀和了解。
- 集合運算查詢（Set Operation Query）：SQL可以使用集合運算：聯集、交集或差集來執行兩個資料表的集合運算查詢。
- 子查詢（Subquery）：子查詢也屬於是一種多資料表的查詢，子查詢是指在SELECT指令（主查詢）中擁有其他SELECT指令（子查詢），也稱為巢狀查詢（Nested Query）。

合併查詢基本原理

- JOIN 是將多個資料表的記錄橫向連接起來, 然後雙方資料行的值相等為條件，來過濾不需要的記錄。

編號	名稱
1	Windows 使用手冊
2	Linux 架站實務
3	SQL 指令寶典

企劃書籍

編號	價錢
1	320
3	380

企劃書籍預定價

編號	名稱	編號	價錢
1	Windows 使用手冊	1	320
1	Windows 使用手冊	3	380
2	Linux 架站實務	1	320
2	Linux 架站實務	3	380
3	SQL 指令寶典	1	320
3	SQL 指令寶典	3	380

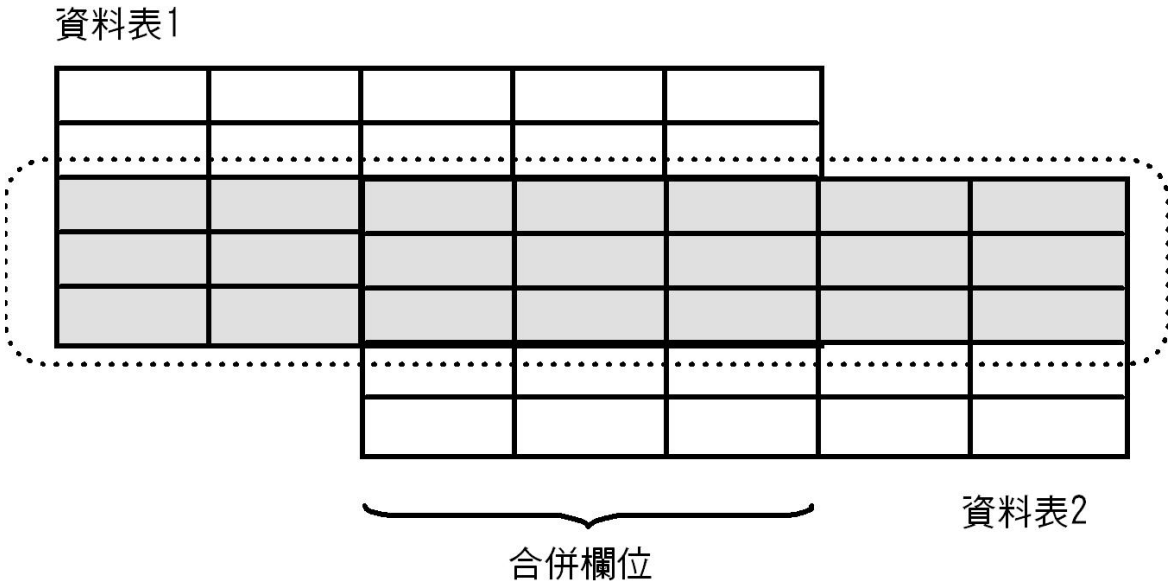
企劃書籍資料表的每一筆記錄, 都會和企劃書籍預定價資料表的每一筆記錄連接爲一筆新記錄, 而產生了 3*2=6 筆的記錄

JOIN 的類型

- [INNER] JOIN：只顯示符合條件的資料列, 此為預設的 JOIN 方式, 因此 INNER 參數可以省略。
- LEFT [OUTER] JOIN：顯示符合條件的資料列, 以及左邊資料表中不符合條件的資料列 (此時右邊資料列會以 NULL 來顯示)。
- RIGHT [OUTER] JOIN：顯示符合條件的資料列, 以及右邊資料表中不符合條件的資料列 (此時左邊資料列會以 NULL 來顯示)。
- FULL [OUTER] JOIN：顯示符合條件的資料列, 以及左邊和右邊資料表中不符合條件的資料列 (此時缺乏資料的資料列會以 NULL 來顯示)。
- CROSS JOIN：此類型會直接將一個資料表的每一筆資料列和另一個資料表的每一筆資料列搭配成新的資料列, 不需要用 ON 來設定條件。

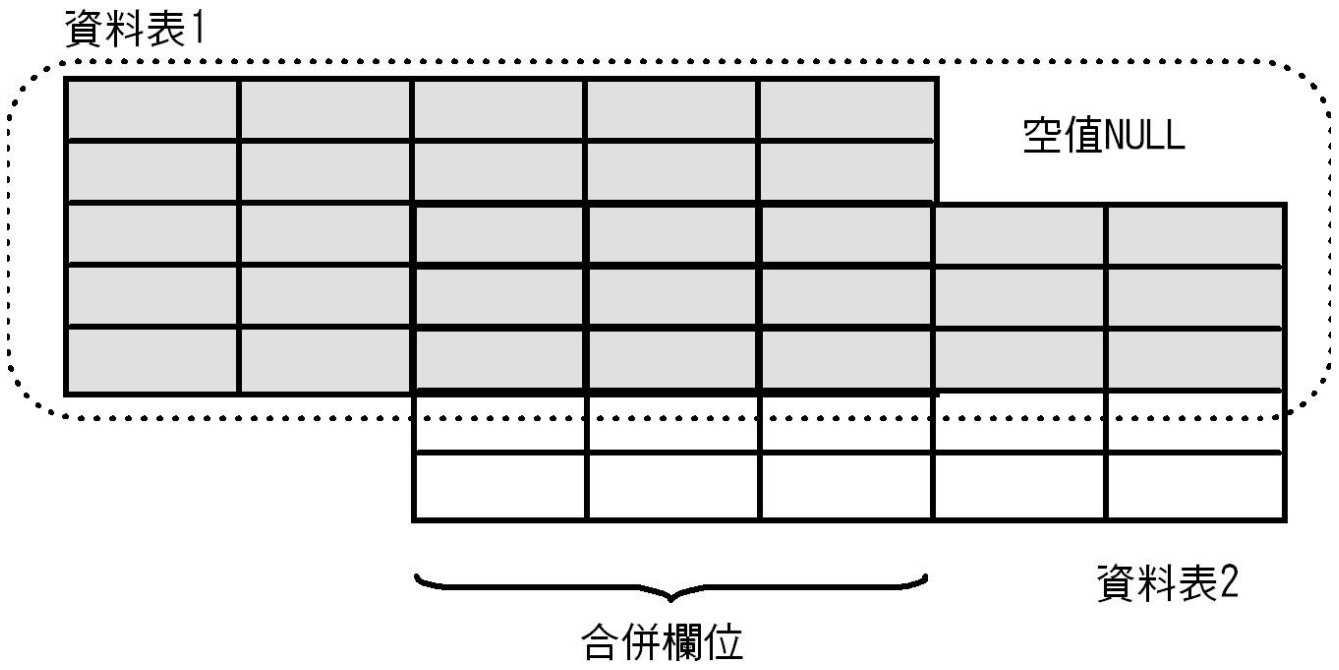
INNER JOIN查詢

- 內部合併查詢只取回多個資料表符合合併條件的記錄資料，即都存在合併欄位的記錄資料，如下圖所示：



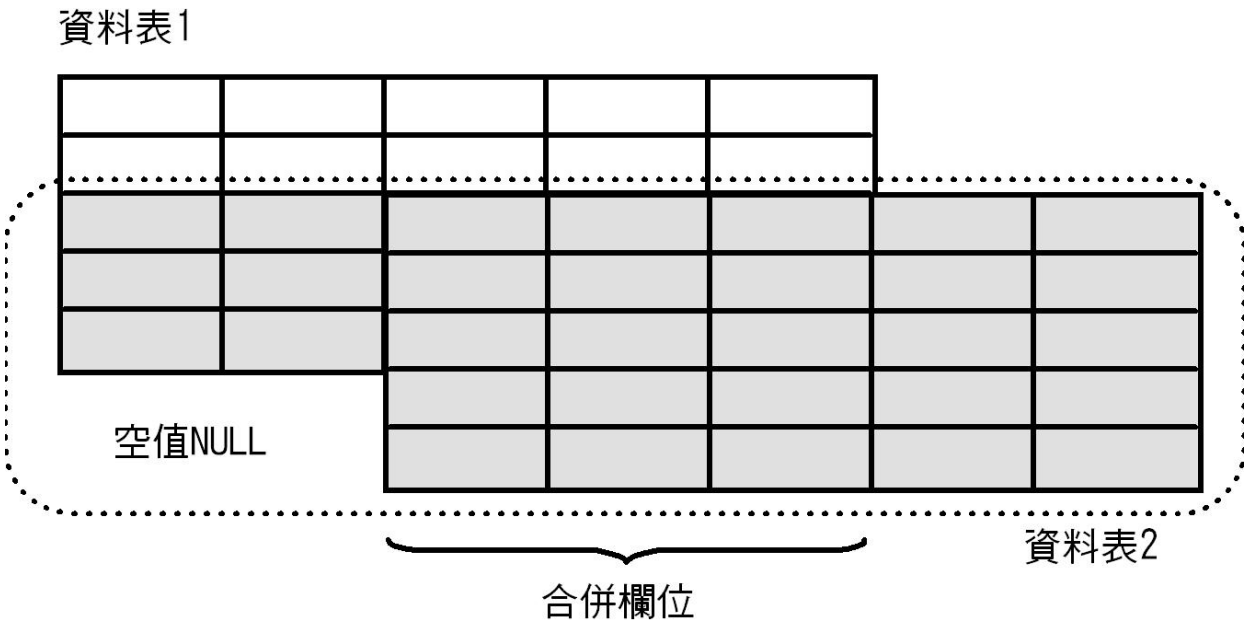
LEFT JOIN查詢

- 左外部合併查詢（LEFT JOIN）：取回左邊資料表內的所有記錄，如下圖所示：



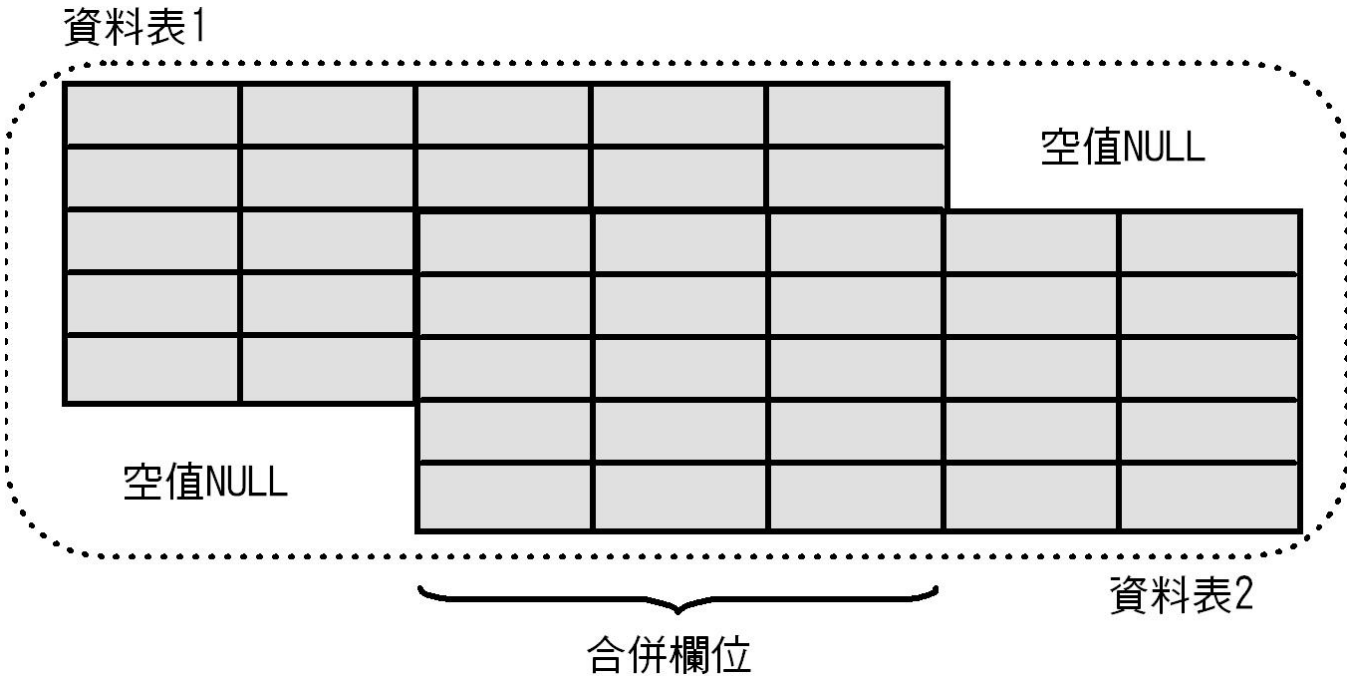
RIGHT JOIN查詢

- 右外部合併查詢（RIGHT JOIN）：取回右邊資料表內的所有記錄，如下圖所示：



完全外部合併（FULL JOIN）

- 完全外部合併（FULL JOIN）：取回左、右邊資料表內的所有記錄，如下圖所示：



CROSS JOIN查詢

- 交叉合併查詢就是關聯式代數的卡笛生乘積運算（Cartesian Product），其查詢結果的記錄數是兩個資料表記錄數的乘積。
- 交叉合併查詢是將一個資料表的每一筆記錄都和合併資料表的記錄合併成一筆新記錄，換句話說，如果兩個資料表的記錄數分別是5和4筆記錄，執行交叉合併查詢後的記錄數就是5 X 4 = 20筆記錄。

JOIN語法

T-SQL提供兩種不同的JOIN語法，

- --ANSI SQL：1992提供的語法

  FROM 資料表1 <連結型態> 資料表2 ON <連結條件>

- --ANSI SQL：1989提供的語法(舊式語法)

  FROM 資料表1 , 資料表2 WHERE <連結條件>

JOIN語法 - SQL-92標準

- SQL-92標準可以指定多種連結類型 ，如內部連結(INNER JOIN) 、外部連結(OUTER JOIN) 、交叉連結(CROSS JOIN)，並使用ON關鍵字指定『連結條件』 。『連結條件』會使用『比較運算子』比較連結資料表內資料行的值，並將結果串聯起來。

JOIN語法 - SQL-89標準

- SQL-89標準的連結語法是將要連結的資料表，透過逗號分隔的方式接續在FROM子句後，『連結條件』則指定於WHERE子句中。

INNER JOIN(內部連結)

- 範例：請列出NorthwindC資料庫中每一筆訂單的負責業務人員。

```
SELECT   dbo.訂貨主檔.訂單號碼  '訂單編號',
         dbo.員工.姓名    '負責業務人員'
FROM     dbo.訂貨主檔 INNER JOIN  dbo.員工
   ON    dbo.訂貨主檔. 員工編號=dbo.員工.員工編號
```

```
SELECT   dbo.訂貨主檔.訂單號碼   '訂單編號',
         dbo.員工.姓名    '負責業務人員'
FROM     dbo.訂貨主檔,dbo.員工
WHERE    dbo.訂貨主檔.員工編號=dbo.員工.員工編號
```

INNER JOIN

- 為了讓程式撰寫方便，通常會在資料表部份使用別名，以便閱讀。

```
SELECT    O.訂單號碼  '訂單編號',
            E.姓名    '負責業務人員'
FROM      dbo.訂貨主檔  O  INNER JOIN  dbo.員工  E
   ON     O.員工編號=E.員工編號
```

```
SELECT   O.訂單號碼   '訂單編號',
           E.姓名    '負責業務人員'
FROM     dbo.訂貨主檔  O , dbo.員工  E
WHERE    O.員工編號=E.員工編號
```

OUTER JOIN(外部連結)

- 用於兩個資料表不一定有相同資料時。
- OUTER JOIN包括：
  - LEFT OUTER JOIN
  - RIGHT OUTER JOIN
  - FULL OUTER JOIN

OUTER JOIN(外部連結)

- Left OUTER JOIN

```
SELECT    E.員工編號,O.員工編號, O.訂單號碼
FROM      dbo.訂貨主檔 O LEFT JOIN dbo.員工 E
   ON     O.員工編號 = E.員工編號
ORDER BY  E.員工編號
```

- Right OUTER JOIN

```
SELECT     E.員工編號,O.員工編號,O.訂單號碼
FROM       dbo.訂貨主檔  O  Right  JOIN  dbo.員工  E
   ON      O.員工編號=E.員工編號
ORDER BY  E.員工編號
```

OUTER JOIN(外部連結)

- Full OUTER JOIN

```
SELECT    E.員工編號,O.員工編號,O.訂單號碼
FROM      dbo.訂貨主檔  O  FULL  JOIN  dbo.員工  E
   ON     O.員工編號=E.員工編號
ORDER BY E.員工編號
```

OUTER JOIN(外部連結)

- 舊式外部連結語法

  星號在左邊Left Outer Join，在右邊Right Outer Join

```
WHERE  table1.ColumnNmae *=table2. ColumnName
```

- 使用前需先執行下列指令：

```
EXEC sp_dbcmptlevel 資料庫名稱, 80 ;
```

- 取消使用則執行下列指令：

```
EXEC sp_dbcmptlevel 資料庫名稱, 90 ;
```

OUTER JOIN(外部連結)

範例如下：

```
EXEC  sp_dbcmptlevel  NorthwindC, 80 ;

SELECT      E.員工編號,O.員工編號,O.訂單號碼
FROM        dbo.訂貨主檔  O ,dbo.員工   E
WHERE       O.員工編號=*E.員工編號  AND
            O.訂單號碼<=10255
ORDER BY  E.員工編號
```

CROSS JOIN(交叉連結)

- CROSS JOIN是兩個資料表在JOIN過程中，所有可能產生的資料列全部列出來
- 不需使用ON來設定條件
- 例如：A資料表有20筆資料，B資料表有30筆資料，當A CROSS JOIN B，結果等於20*30=600筆資料。
- 範例語法：

```
SELECT   *
FROM    dbo.訂貨主檔   CROSS JOIN   dbo.員工
```

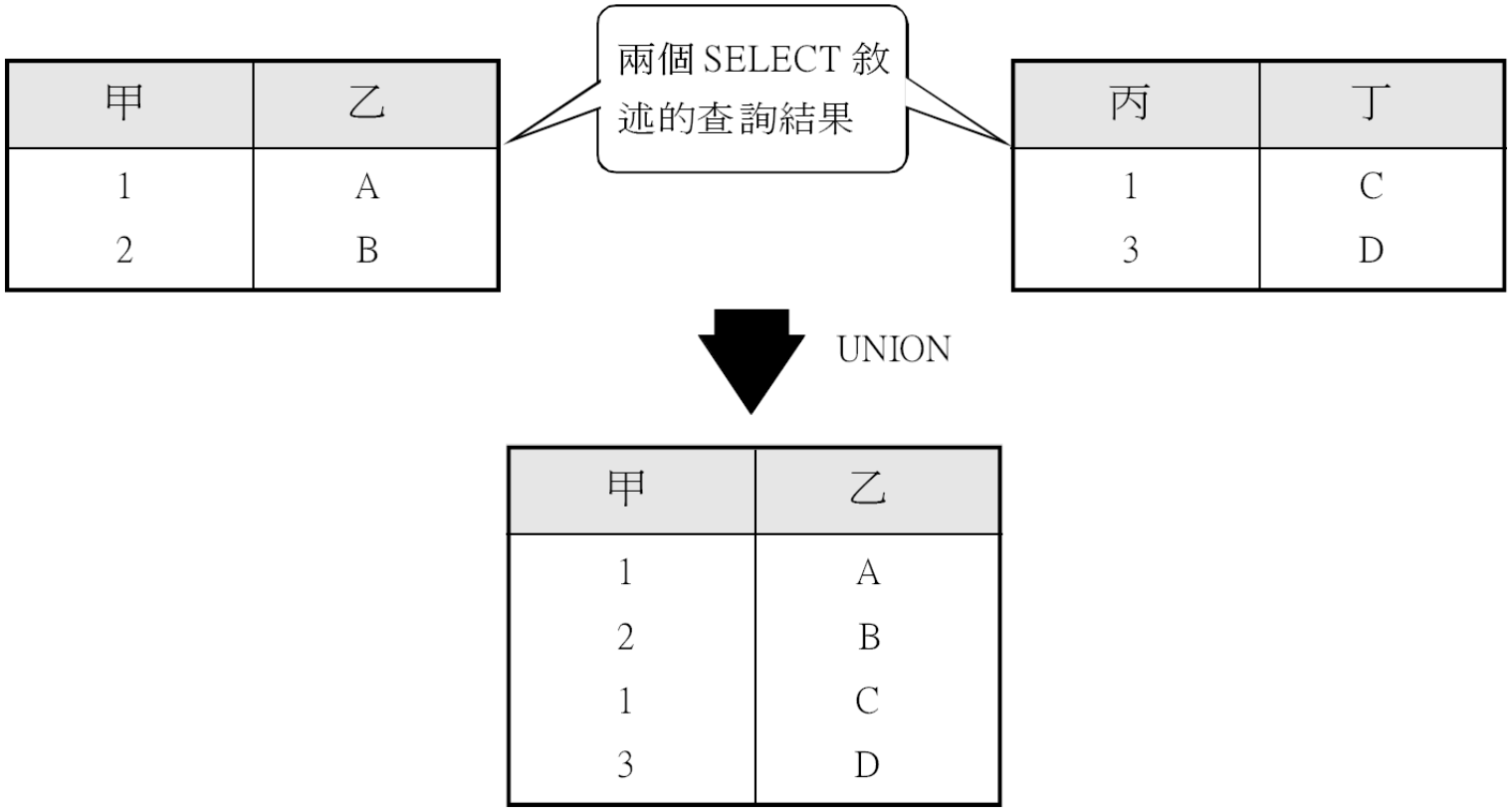
SELF JOIN

- SELF JOIN-自己JOIN自己
- 使用SELF JOIN必需使用別名
- 範例：請列出NorthwindC資料庫，每一位員工的主管。

```
SELECT E1.員工編號  [員工編號],
          E1.姓名  [員工姓名],
          E2.員工編號   [主管編號],
          E2.姓名   [主管姓名]
FROM   dbo.員工  E1   LEFT  JOIN  dbo.員工   E2
         ON  E1.主管=E2.員工編號
```

UNION聯集查詢

- UNION 可將多個 SELECT 敘述的查詢結果合併成一組。



UNION聯集查詢

- UNION 是將多個查詢結果做 “上下垂直” 合併, 所以欄位數不會增加。
- JOIN 是將資料表的欄位做左右水平合併, 所以通常欄位數會增多。
- UNION合併條件：
  - 欲合併的查詢結果, 其欄位數必須相同。
  - 欲合併的查詢結果, 其對應的欄位 (如上圖的甲欄和丙欄、乙欄和丁欄) 一定要具備 "相容" 的資料型別, 即資料型別可以不同, 但兩者必須能夠互相轉換。

UNION聯集查詢

- 合併後的結果：
  - 合併結果的欄位名稱會以第一個查詢結果的欄位名稱命名, 其他查詢結果的欄位名稱則會被忽略掉。
  - 合併時, 若對應的欄位具備不同的資料型別, 則 SQL Server 會進行相容性的型別轉換, 轉換的原則是以“可容納較多資料的型別為主”。就拿上圖來說, 假設乙欄是 CHAR(10) 型別, 丁欄是 CHAR(20) 型別, 則合併後的乙欄便會是 CHAR (20) 型別。
- 注意：並不是每種資料型別都可以互相轉換。如果無法自動轉換, 除非人為強制介入轉換, 否則便會顯示錯誤訊息, 無法完成合併。

UNION聯集查詢

- UNION語法如下：

```
select_statement UNION [ALL] select_statement
                    [ UNION [ALL] select_statement ] [ ... n]
[ ORDER BY ... ]
```

```
SELECT 城市, 行政區
FROM   dbo.員工
UNION
SELECT  城市, 行政區
FROM   dbo.客戶
```

UNION聯集查詢

- 語法說明：
  - ALL：如果設定 ALL 參數, 最後的合併結果會將重複的記錄都顯示出來。如果不設定 ALL 參數, 則在合併結果中, 重複的記錄將只會顯示一筆。
  - 小括弧 ( )：合併的順序原則上是 "由左至右", 但可利用小括弧 ( )來改變合併的優先順序。例如：select1 UNION (select2 UNION select3), 則會先合併 select2、select3, 得到的結果再和 select1 合併。

UNION聯集查詢

- 語法說明：
  - select_statement：和一般的SELECT語法一樣。
  - GROUP BY 和 HAVING 子句只能用在個別的 select_statement 中, 不可用於整個 UNION 敘述的最後。
  - ORDER BY 則只能用在整個敘述的最後, 針對最後的合併結果做排序或計算, 不能用在個別的 select_statement 中。
  - 只有第一個 select_statement 可以設定 INTO 子句。

UNION聯集查詢

- 判讀兩個以上的資料來源是否重覆，需要額外的計算比較，尤其當欄位很多時，更是會佔用很多的效能來運作。所以若確定參與UNION的兩個資料來源不會有重覆資料，或不在乎重複紀錄時，使用UNION ALL會比UNION來得有效率。

UNION聯集查詢

- 使用UNION加入臨時資料
- 範例如下：

```
SELECT   城市, 行政區, 區域號碼
FROM     dbo.員工
UNION
SELECT   城市, 行政區, 郵遞區號
FROM     dbo.客戶
UNION
SELECT   ‘臺灣’ ,’台北’ ,’882’
```

INTERSECT交集查詢

- INTERSECT交集查詢指令可以從兩個資料表取出同時存在的記錄資料。

```
SELECT   城市, 行政區+'區' 行政區
FROM     dbo.員工
INTERSECT
SELECT   城市, 行政區
FROM     dbo.客戶
```

EXCEPT差集查詢

- EXCEPT差集查詢指令可以取出存在其中一個資料表，而不存在另一個資料表的記錄資料

```
SELECT   城市, 行政區+'區' 行政區
FROM     dbo.員工
EXCEPT
SELECT   城市, 行政區
FROM     dbo.客戶
```

Subquery子查詢

- 指包含在主要查詢中的另一個 SELECT 查詢。通常利用子查詢先挑出部份資料, 做為主要查詢的資料來源或選取條件。
- 範例：使用Subquery計算每一筆訂單訂購的產品總數量。

```
SELECT  訂單號碼,(SELECT SUM(數量)  FROM dbo.訂貨明細
                  WHERE  dbo.訂貨主檔.訂單號碼=訂單號碼 )  總數量
FROM  dbo.訂貨主檔
```

```
SELECT  O.訂單號碼,D.數量
FROM  dbo.訂貨主檔 O JOIN
        (SELECT 訂單號碼,SUM(數量)  數量
         FROM dbo.訂貨明細
         GROUP BY 訂單號碼 ) D  ON  o.訂單號碼=d.訂單號碼
```

獨立子查詢與關聯子查詢

- 獨立(Independent)子查詢：
  - 指可以脫離主查詢, 單獨執行的子查詢。

```
SELECT * FROM dbo.客戶
WHERE 城市  IN (SELECT DISTINCT 城市  FROM dbo.員工)
```

- 關聯(Corelated)子查詢
  - 指無法單獨存在的子查詢

```
SELECT  *
FROM    dbo.訂貨主檔
WHERE  (SELECT SUM(數量) FROM dbo.訂貨明細
          WHERE  dbo.訂貨主檔.訂單號碼=訂單號碼)>100
```

獨立子查詢與關聯子查詢

- 請注意, 雖然內層查詢可以參考到外層查詢的資料表, 但反之卻不行, 也就是外層查詢不可以使用內層查詢的資料表。

Subquery子查詢

- Subquery(子查詢)的語法限制：
  - 整個子查詢敘述需使用小括弧 ( ) 括住。
  - 子查詢中不能使用 INTO 子句。
  - 若子查詢中有用到 "SELECT TOP n…", 才可設定 ORDER BY 子句來排序。

Subquery子查詢

- Subquery的傳回結果 (即查詢結果) 分成 單一值、單欄的多筆資料、不限定欄數的多筆資料等三種。
- 處理Subquery傳回結果的方法也分為 3種：
  - 方法 1：直接取值－直接使用子查詢的傳回值, 例如用 =、>、< 做比較, 或進行加減乘除等運算。
  - 方法 2：比對清單－使用 **IN** 、**ALL**、或 **ANY (SOME)** 運算子判斷某個值是否存在於傳回清單中, 其比對結果為 True 或 False。
  - 方法 3：測試存在－使用 **EXISTS** 運算子判斷是否有傳回資料, 其測試結果亦為 True 或 False。
- 使用時機：

傳回結果	一欄	多欄
一筆記錄	方法 1 、 2 、 3	方法 3
多筆記錄	方法 2 、 3	方法 3

直接取值的Subquery子查詢

- Subquery只傳回單一值時
- 可使用於任何允許運算式出現的地方,
- INSERT 、 UPDATE、DELETE 等敘述中都可使用這種子查詢。

```
SELECT  訂單號碼,
        總數量=(SELECT SUM(數量)  FROM dbo.訂貨明細
                WHERE  dbo.訂貨主檔.訂單號碼=訂單號碼)
FROM   dbo.訂貨主檔
```

```
SELECT  *
FROM    dbo.訂貨主檔
WHERE  (SELECT SUM(數量) FROM dbo.訂貨明細
          WHERE  dbo.訂貨主檔.訂單號碼=訂單號碼)>100
```

比對清單的Subquery子查詢

- 子查詢會傳回單欄的一或多筆記錄 (型式像一份單欄的表格) 。
- 可以使用 IN、ALL或 ANY(SOME) 運算子來做比對, 其中 ALL 和 ANY(SOME) 必須與比較運算子 (如 >、<=、=) 一起使用。
- 此類子查詢可用於邏輯運算式中, 包括 SELECT、INSERT、UPDATE、DELETE 等敘述中的WHERE 或 HAVING 子句。

比對清單的Subquery子查詢

- IN：IN 運算子可用來判斷給定的值是否在指定的子查詢中。

```
SELECT  *
FROM  dbo.客戶
WHERE  城市  IN (SELECT DISTINCT 城市 FROM dbo.員工)
```

- ALL：ALL 運算子表示在查詢中的結果必須滿足子查詢中的所有結果。

```
select  *  from dbo.訂貨明細
where 單價<=All(select DISTINCT 單價 FROM dbo.產品資料)
```

比對清單的Subquery子查詢

- ANY、SOME：ANY 運算子表示查詢結果只要滿足子查詢中任一個值即可
- SOME 是 SQL-92 標準的用法, 意思與 ANY 相同。

```
select * from dbo.訂貨明細
where 單價<=Any(select DISTINCT 單價 FROM dbo.產品資料)
```

測試存在的Subquery子查詢

- 使用 EXISTS 來測試子查詢是否有傳回任何結果, 如果有結果就會傳回 TRUE, 沒有結果則傳回 FALSE。
- 這類的子查詢就不限定傳回值是單一值、單欄或多欄了, 只要有任何結果傳回即為 TRUE (即使傳回 NULL 值也算), 否則為 FALSE 。

```
SELECT *
FROM   dbo.客戶
WHERE  Exists (SELECT *
                    FROM dbo.訂貨主檔
                    WHERE   dbo.客戶.客戶編號=客戶編號)
```

使用**Exists**測試存在時, **Subquery**的**select** 敘述不用指定欄位名稱,只要使用星號『*』即可。