SQL 結構化查詢語言

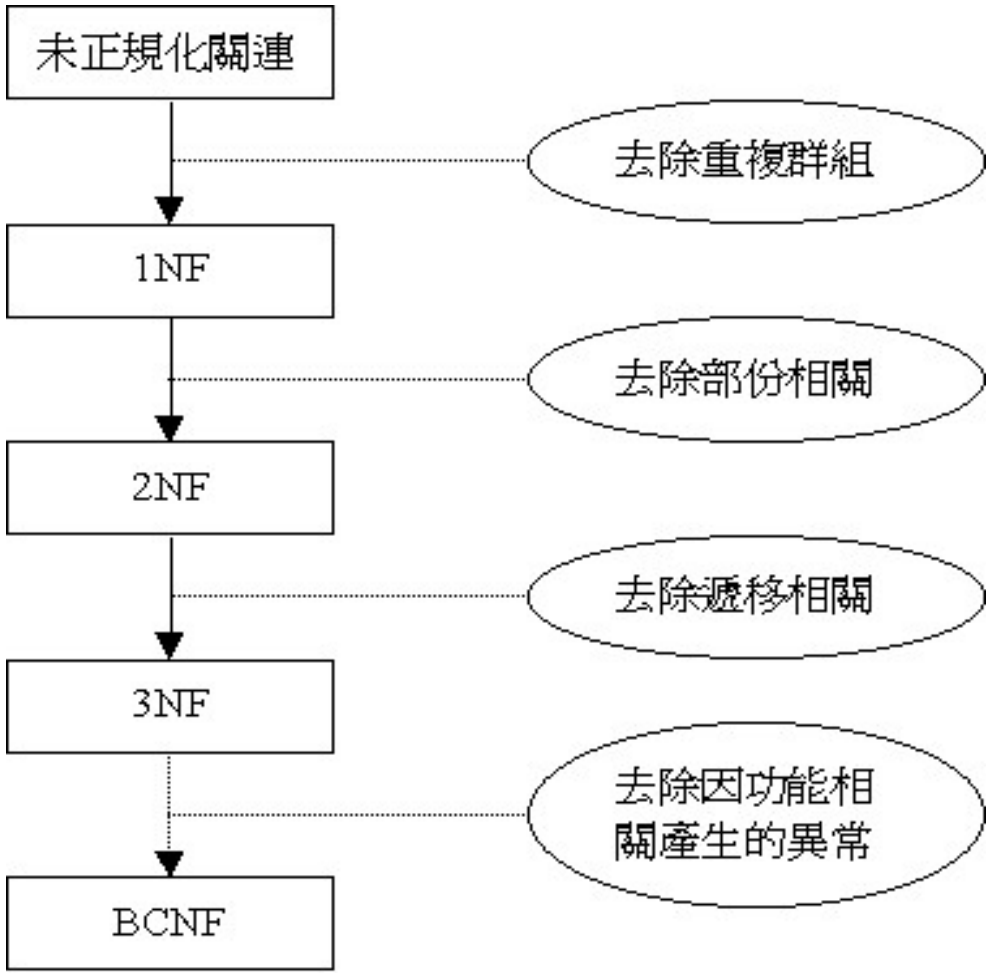
第三堂 主要函數的使用、基本操作 (2)

聯成電腦

張明泰 mtchang.tw@gmail.com

正規化

- 將表格細分成多個表格，直到每個 Table 只描述一種事實為止，其間的過程就稱為正規化。



實例探討

- 假設我們將要設計一個成績單郵寄列印系統，需要學號、地址、郵遞區號、學科代碼與各科成績等資料，而初步搜集到的原始資料如下表所示：

Stu_no	City	ZIP	Subject_no, Score
75312	台中市	400	(S5302, 89), (S5345, 90), (S8005, 78), (S3581, 80), (M1201, 65), (M5251, 95)
75524	高雄市	800	(S5302, 88)
75302	高雄縣	830	(S5302, 98), (S5345, 90), (S3581, 84), (M5251, 85)

第一正規化

- 將表格中的變動項目展開來，其結果即為一級正規化。
- 1. 必須為 row-column 的二維式 table
- 2. table 的每一筆資料（ row ）只描述一件事情
- 3. 每一欄位只含有單一事物的特性（欄位的唯一性）
- 4. 每一筆 row 的欄位內只允許存放單一值
- 5. 每個欄位名稱必須是獨一無二的
- 6. 沒有任何兩筆資料是相同的
- 7. row 或欄位的先後順序是無關緊要的
- 結論：消除重覆性資料

1NF 後 - 表格 A

- 最重要的是能滿足
- 「**每個欄位只能含有一個值**」

Stu_no	City	ZIP	Subject_no	Score
75312	台中市	400	S5302	89
75312	台中市	400	S5345	90
75312	台中市	400	S8005	78
75312	台中市	400	S3581	80
75312	台中市	400	M1201	65
75312	台中市	400	M5251	95
75524	高雄市	800	S5302	88
75302	高雄縣	830	S5302	98
75302	高雄縣	830	S5345	90
75302	高雄縣	830	S3581	84
75302	高雄縣	830	M5251	85

1NF 後的狀況

- 在同一學生只能選修同科目一次的條件下
- Stu_no 」加上「 Subject_no 」可以做為 A 的主鍵（ Primary key ）
- 有三項「功能相依」關係是錯誤的（如紅線所示），City 與 ZIP 的值與 Subject_no 無關。

Stu_no	City	ZIP	Subject_no	Score
75312	台中市	400	S5302	89
75312	台中市	400	S5345	90
75312	台中市	400	S8005	78
75312	台中市	400	S3581	80
75312	台中市	400	M1201	65
75312	台中市	400	M5251	95
75524	高雄市	800	S5302	88
75302	高雄縣	830	S5302	98
75302	高雄縣	830	S5345	90
75302	高雄縣	830	S3581	84
75302	高雄縣	830	M5251	85

1NF 後的問題

- 無法單獨新增一筆學生資料。因為 Subject_no 是 Primary key 之一，不能為空值（Null）；因此，一個未修習任何課程學生的資料，將無法寫入 A。
- 無法單獨刪除一筆成績資料。如果我們打算刪除（75524, S5302）這筆資料的話，該生的地址資料也將一併消失。
- 需要同步異動的資料太多。假如 75312 這個學生搬家了，那麼我們得異動其中的 6 筆紀錄。
- 因此得繼續進行 2NF

第二正規化

- 移去部分相關性 (Partial dependency) 得到二級正規化。
- 第二正規化的表格必須合下面條件：
- 移去部分相關性
- →　結論：消除功能相依（ Functional Dependency ）
- 所謂功能相依是指表格和表格之間的相互關係，若某個表格中有兩個欄位 A 及 B ，當 A 欄位值可推導出 B 欄位值，稱功能相依性。
- 即若一關連 R ，其屬性 Y 功能相關於屬性 X ，記作 R.X→R.Y ；若且唯若 R 中有二個 X 值相同時，其 Y 值亦相同。

2NF 後 - 表格 B1,B2

- **非主鍵的欄位都要對主鍵有「完全地功能性相依（Fully Functional Depen-dency）」關係**

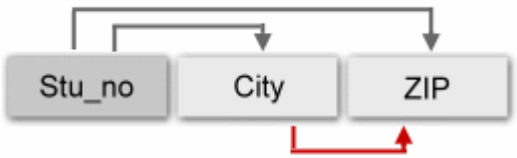
《B1》

Stu_no	City	ZIP
75312	台中市	400
75524	高雄市	800
75302	高雄縣	830

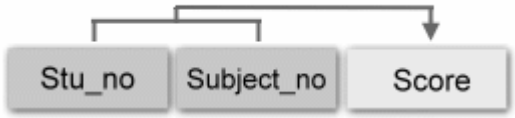
《B2》

Stu_no	Subject_no	Score
75312	S5302	89
75312	S5345	90
75312	S8005	78
75312	S3581	80
75312	M1201	65
75312	M5251	95
75524	S5302	88
75302	S5302	98
75302	S5345	90
75302	S3581	84
75302	M5251	85

2NF 後的狀況



B1



B2

- 「無法單獨新增一筆學生資料」與「無法單獨刪除一筆成績資料」的問題都解決了。
- 在一個表格中，如果某一欄位值可決定其他欄位值；而這些欄位中又存在某一欄位可以決定剩餘欄位的值，稱為「遞移相依性（Transitive Dependency）」。

《B1》

Stu_no	City	ZIP
75312	台中市	400
75524	高雄市	800
75302	高雄縣	830

《B2》

Stu_no	Subject_no	Score
75312	S5302	89
75312	S5345	90
75312	S8005	78
75312	S3581	80
75312	M1201	65
75312	M5251	95
75524	S5302	88
75302	S5302	98
75302	S5345	90
75302	S3581	84
75302	M5251	85

2NF 後的問題

- 在 **B1** 之中便有「遞移相依性」關係存在： B1.Stu_no -> B1.City 且 B1.City -> B1.ZIP 。在這樣的架構下，將產生下列問題：

- 無法單獨新增一筆縣市資料。因為 Stu_no 是 Primary key ，不能為空值（ Null ）；因此，若無任何學生居住的某個縣市，其郵遞區號資料將無法被事先建立。
- 無法單獨刪除一筆學生資料。如果我們打算刪除 75524 這筆資料的話，該生所在的高雄市郵遞區號資料也將一併消失。
- 仍有需要同步異動的資料。假如台中市的郵遞區號修改了，且住在該地區的學生又不只一位時，那麼我們又得異動多筆紀錄了。
- 因此，還得繼續進行 3NF 。

第三正規化

- 再來消除關連間之遞移相關 (Transitive dependency) 最後得到三級正規化。
- 第三正規化的表格必須合下面條件：
- 消除遞移相依（ Transitive Dependency ）
- 所謂遞移相依是指在一個表格中，如果某一欄位值可決定其他欄位值，但這些欄位中又存在某一欄位可以決定剩餘欄位值，稱遞移相依性。若有上述情況存在，如果在刪除資料時，可能會造成其他資料損毀。
- 一個 FD 若 R.A→R.B 且 R.B→R.C 則， R.A→R.C 成立，此種相關性稱為遞移相關。

3NF 後 - 表格 c1,c2,b2

- 「無法單獨新增一筆縣市資料」與「無法單獨刪除一筆學生資料」的問題都解決了，需要同步異動大量資料的情況似乎也不復存在了。

《C1》

Stu_no	City
75312	台中市
75524	高雄市
75302	高雄縣

《C2》

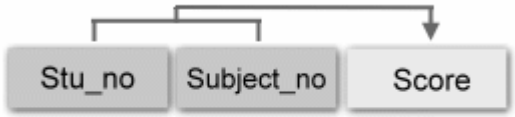
City	ZIP
台中市	400
高雄市	800
高雄縣	830

《B2》

Stu_no	Subject_no	Score
75312	S5302	89
75312	S5345	90
75312	S8005	78
75312	S3581	80
75312	M1201	65
75312	M5251	95
75524	S5302	88
75302	S5302	98
75302	S5345	90
75302	S3581	84
75302	M5251	85

3NF 後的狀況

- 一般表格進行至
  第三正規化時，
  多半沒有什麼狀況了



C2

B2

《C1》

Stu_no	City
75312	台中市
75524	高雄市
75302	高雄縣

《C2》

City	ZIP
台中市	400
高雄市	800
高雄縣	830

《B2》

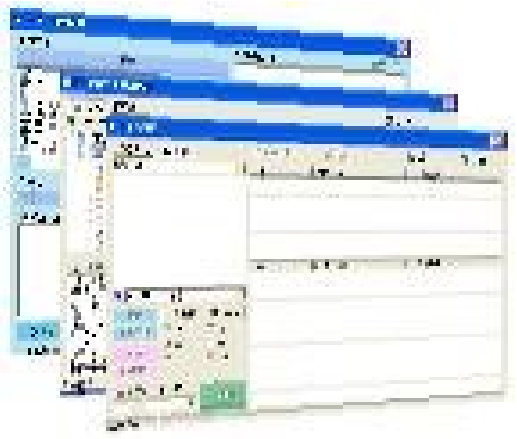
Stu_no	Subject_no	Score
75312	S5302	89
75312	S5345	90
75312	S8005	78
75312	S3581	80
75312	M1201	65
75312	M5251	95
75524	S5302	88
75302	S5302	98
75302	S5345	90
75302	S3581	84
75302	M5251	85

正規化重點

- 正規化只是建立資料表的原則，而非鐵律。
- 切莫因為過度正規化，反而導致資料存取的效率下降。
- 有時在優先考量執行效率的前提下，我們還必須做適當的反正規化（Denormalize）。

補充 - 中文編碼的轉換

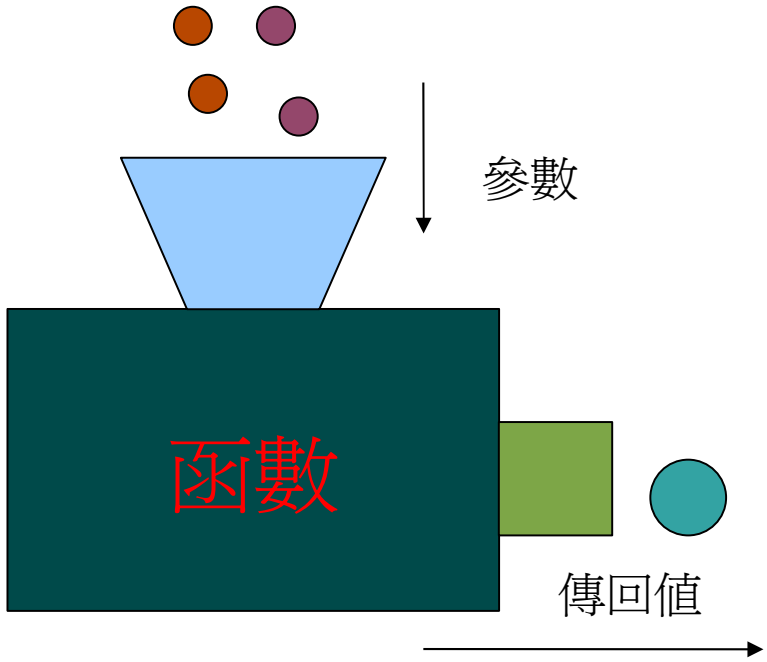
- **ConvertZ v8.02**
- http://alf-li.pcdiscuss.com/
- NotePAD++
- http://notepad-plus.sourceforge.net/



休息一下

主要函數的使用

- 函數：在 SQL 裡面函數是只一堆處理的集合。



需要參數的函數種類

- 單一行函數

  針對各列去執行處理，以行為單位傳回結果。有多少列就傳回多少結果。

- 群集函數

  多列當成一個群集，最後傳回一個結果。

- 數值函數

  指的是數學上的處理函數，含有數值的列也可以當成引數來指定。

CEILING,CEIL

- 此函數會傳回一個不小於引數的整數
- 範例： `SELECT CEILING(數量/3) AS AVG FROM sales;`
- 在 oracle 或 postgreSQL 則是使用 CEIL 函數。

FLOOR

- Floor 會傳回不大於引數的最大整數值。
- 範例： SELECT FLOOR(數量 /3) AS AVG FROM sales;
- 在 SQL-server MySQL PGSQL Oracle 都可以使用

RAND

- RAND 會在 0 到 1.0 之間取一個亂數。
- 範例： SELECT RAND(6);
- 在 SQL-server 及 MySQL 中可使用。 Access 中可以使用 RND 函數

其它的數值函數

- 函數名稱 功能 注意事項
- ABS(m) *1 傳回 m 的絕對值
- ROUND(m[,x])*1*2 把 m 在小數點以下 x 為作四捨五入
- POWER(m,n) *1 傳回 m 的 n 次方
- SQRT(m) *1 傳回 m 的平方根
- MOD(m,n)*1 傳回 m 除以 n 的餘數
- SIN(m) *1
- COS(m) *1
- TAN(m) *1
- EXP(m) *1 傳回 m 的指數值
- LOG([m,]n)*1 傳回以 m 為底的 n 的自然對數
- SIGN(m) *1 傳回 m 的正負符號
- 註：*1:m,n 為數值或是別名 *2:x= 位數

LEN,LENGTH

- LEN 或 LENGTH 會傳回字串的文字數量。
- 範例： SELECT LEN('中文');
- 此範例傳回值為 6 是因為 mysql 中文字採 unicode 編碼以 byte 計算 (一個 unicode 的中文字佔用 3bytes) 。如果使用 access 或 SQL server 回傳值為 2 。
- 引數中若是字串或字串行的列名比需使用「'」單引號刮起來。
- 在 SQL server 及 access 使用 len ， oracle 、 mysql 及 postgreSQL 使用 length

substring,substr

- 從字串只取出指定的部分來回傳
- 範例： SELECT substring(' 故事書 ',2,2);
- 練習：請從 students 表格中取出全部學生的姓名，的「姓」
- Mssql,pgsql 使用 substring
- Oracle 中使用 substr
- Access 同樣的函數是 MID

upper,lower

- Upper 是把字串寫成大寫的函數， lower 是把字串寫成小寫的函數。
- 範例： select LOWER('CompanyName');
- select upper('CompanyName');

日期函數

- **取得目前的日期**
- **Mysql 使用 Getdate 函數 ,access 使用 date,oracle 和 mysql 使用 sysdate,ppgsql 使用 now**
- **範例：** `SELECT GETDATE()`
- 思考？如何取得日、月、年的資料？

MSSQL 及 ACCESS

- 範例： **SELECT** day(getdate()); 取得日期的天
- 範例： select month(getdate()); 取得日期的月
- 範例： select year(getdate()); 取得日期的年

合成函數 -avg,sum,count

語法： avg(欄位); 限數值型

用途：針對列中所含的值，求出平均值， null 除外。

語法： sum(欄位); 限數值型

用途：針對列中的值，求出總和， null 除外。

語法： count(欄位);

用途：可以指定 * 或是欄位名，以欄位名的話，會以 null 以外的行為對象作計算，以 * 則是會以含有 null 直的所有列為對象。

合成函數

- 範例： **SELECT COUNT(** 公假 **) FROM** records;
- 範例： **SELECT SUM(** 公假 **) FROM** records;
- 範例： **SELECT AVG(** 公假 **) FROM** records;
- 練習 1 ：計算出在 records 裡面班級座號的列數，需扣除重複的班級座號。以傳回植的方式顯示。
- 練習 2 ：使用 records 及合成函數，計算出每個班級座號學生所請的每一種假的總數。

MAX,MIN

- MAX 求出欄位中的最大值， MIN 函數會求出欄位中的最小值。
- 在 mssql,access,oracle,mysql,pgsql 都可使用 。
- 範例： **SELECT MAX(公假) FROM** records ;

以函數處理結果為條件，取出特定資料

- 在取出符合條件時，可用 where 子句，但在 where 裡面不能使用合成函數。所以如果向用的話必須使用 group by 和 having 結合使用。

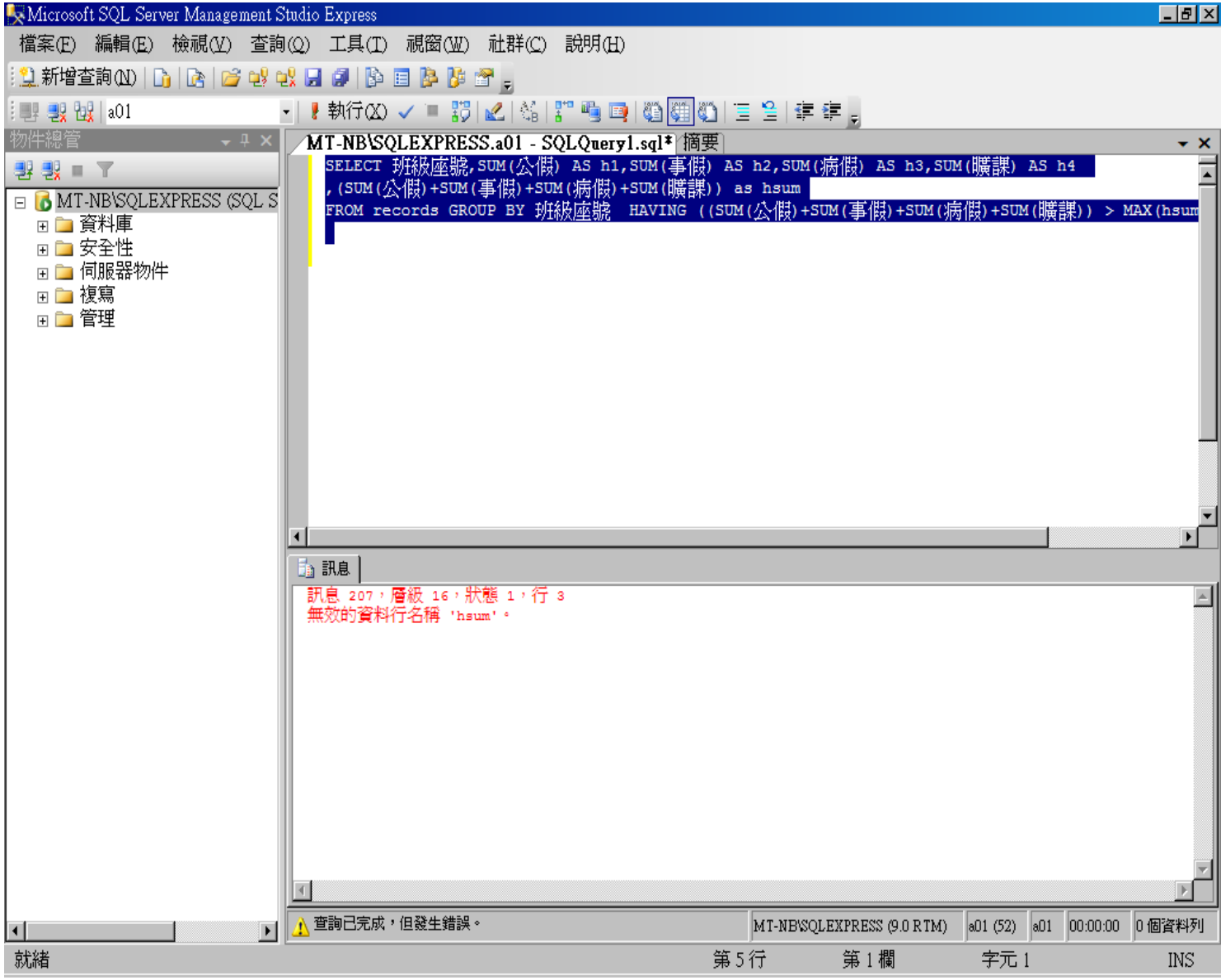
- 範例： SELECT 班級座號 ,SUM(公假) AS h1,SUM(事假) AS h2,SUM(病假) AS h3,SUM(曠課) AS h4
- ,(SUM(公假)+SUM(事假)+SUM(病假)+SUM(曠課)) as hsum
- FROM records GROUP BY 班級座號 HAVING ((SUM(公假) +SUM(事假)+SUM(病假)+SUM(曠課)) > 10);
- 說明：找出 records 中，累積總和大於十節的學生。

CAST 函數

- 用途：資料型別轉換
- 範例： cast(0.245 as varchar)
- 範例： cast('2000-02-02' as datetime)
- 練習：使用型別轉換函數，將 studetns 表格的資料，依' 班級座號' 由小到大排序。

補充教材 - 使用 mssql 參考手冊

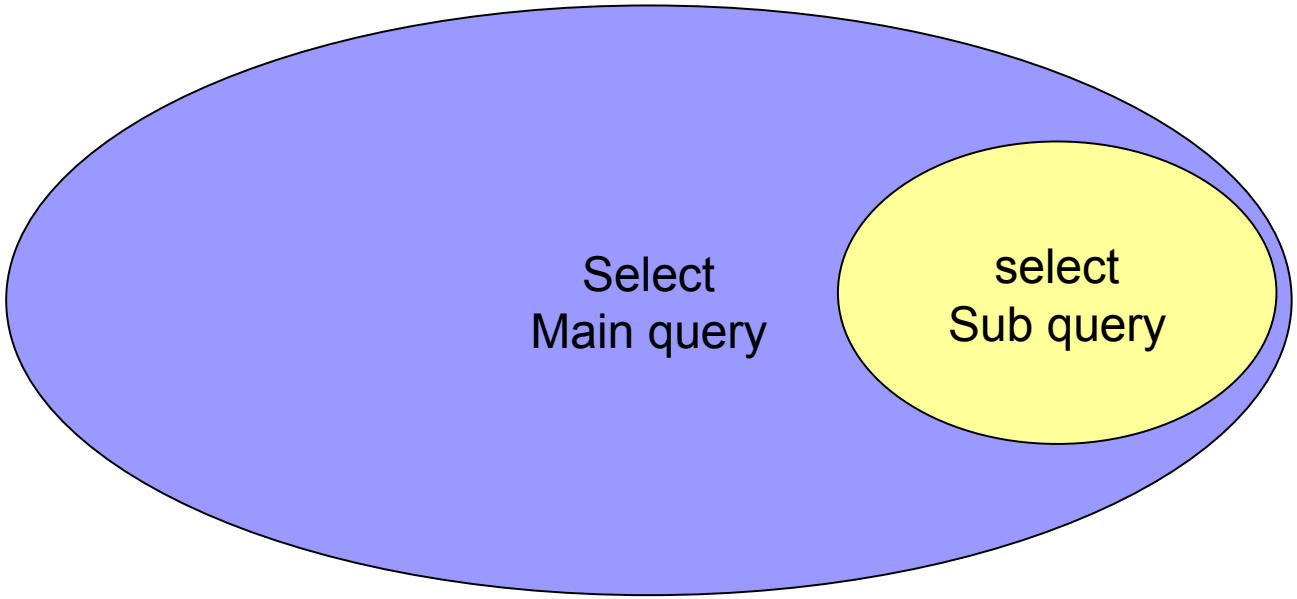
- 使用 mssql 手冊查詢



休息一下

什麼是子查詢？

- 在 select 中在放一個 select 稱之為子查詢 sub query 。
- 在外側的就稱為主查詢 main query 。



Insert 用法

- 範例： **INSERT INTO** records **VALUES** ('10126',900226,5,5,5,5);
- 說明：省略欄名進行登錄
- 範例： **insert into records (班級座號 , 年月日) values('10126',900226);**
- 說明：只在特定欄位做值的登錄
- **CHECK: select** * **from records where 班級座號=10126**

登錄 select 敘述的結果

- **範例：** insert into test(學號 , 姓名 , 住址) select 學號 , 姓名 , 住址 from students;
- **說明：將 select 出來的學號，姓名，住址等資料，插入新的測試表格中。**
- **事前：** CREATE TABLE test(學號 int , 姓名 VARCHAR(10) , 住址 VARCHAR(40));
- 先建立一個測試用的資料表

更新值

- **範例：** **UPDATE test SET 姓名 =' 張小明 ' WHERE 學號 =911001;**
- 說明：更新學號為 911001 的姓名欄位內容改為張小明。
- 範例： **UPDATE test SET 姓名 =' 張小明 ', 住址 ='mtchang.tw@gmail.com' WHERE 學號 =911001;**
- 說明：更新多值使用「，」來作區分。

把符合條件的資料刪除

- **範例：** **DELETE FROM** test ;
- 說明：把 test 表格資料，通通砍光光。
- 範例： delete **from** records **WHERE 班級座號** ='10126' ;
- 說明：把符合條件的資料刪除

在 select 中使用子查詢

- **範例：** **SELECT * FROM** product **WHERE 單價 >=(SELECT AVG(單價) FROM** product);
- 先找出主查詢的範圍，再以子查詢結果過濾。
- 說明：找出在 product 中，高於平均單價的產品。

在 having 中使用子查詢

- **範例：** **SELECT 班級座號 ,SUM(公假) AS s1 FROM records GROUP BY 班級座號 HAVING SUM(公假) > (SELECT AVG(公假) FROM records);**
- 因為有用到合成函數及 group by ，所以需用 having ，後面的子查詢查出的公假平均值是每列的平均值，並非每個學號的平均值。

在 from 中使用子查詢

- **範例：** **SELECT AVG(ss) FROM (SELECT SUM(公假) AS ss FROM** records **GROUP BY** 班級座號 **) AS** s1;
- 說明：計算每個學生的請公假累積數量平均值。
- 注意：在 sql server 裡，如不在行內視界部分以 as 來加上名稱的話，就會有錯誤。
- 練習：找出請公假數量大於全體總和平均值的學生班級座號及請公假的數量。

子查詢的應用

- 子查詢可以應用在 insert,update,delete
- 練習 1 ：把 student 資料庫中，姓王的同學資料匯入到 test 資料庫中。

參考資料

- http://dev.mysql.com/doc/
- MSSQL 參考手冊