

建立高執行效能的資料模型-經驗分享

高執行效能的設計上，資料模型設計要注意哪些事

一個資料庫應用系統的執行效能好壞，我們可以從高層次的角度，把系統區分為關聯式資料庫檔案設計、應用程式開發、系統執行環境三個部份。在這三個部份之中，任何一個地方規劃不當都將會是系統執行效能的瓶頸，以下是三個部份包含的內容架構：

■ 關聯式資料庫檔案設計

- 正規化/非正規化取捨上-正規化過度，跑報表需 Join 很多的檔案
- 是否需要統計檔-產生報表的資料要從統計檔抓?或者從明細檔抓?
- 檔案是否要做切割-交易資料是一個檔或者需要多個檔?或者要做 Table Partition?
- 是否需要分歷史檔-歷史檔要怎麼分?

■ 應用程式開發

- 接受使用者不適當的需求
- 太複雜的 UI Pattern
- 該寫在 Server Site 的程式(Store Procedure),寫在 Client Site(網路頻寬越差,速度效能差異越大)
- 效率不彰的 SQL Command

■ 系統執行環境

- 作業系統以及硬體相關環境的架構與配置
 - ✓ OS、 Raid System、 Disk I/O、 CPU、 Network Card、 Ram 、網路架構
- 資料庫安裝
 - ✓ 實體資料庫檔案規劃，避免 I/O 競爭(例:Oracle:Redo log file 、 Archieve file 與 Table 、 Index 分在不同磁碟)
- 資料庫空間及物件配置
 - ✓ 資料庫及物件的成長配置方式
 - ✓ Memory 配置，避免 Memory Page in/out

接下來可能有人會問，以整體來看，上述哪個部份是影響系統執行效能最重的?我個人認為是資料庫的 ER/Model 的檔案架構規劃。因為所有應用系統都是根據資料庫關連式的檔案架構，透過程式的撰寫來存取資料庫裡頭的資料，檔案架構的重要性如同大樓的地基一般；檔案沒架構好，系統效能差，只要檔案架構一翻，很多程式都要跟著改，那樣修改程式的成本是非常非常可觀的，而這樣的致命錯誤，確無法透過 DBA 深厚的資料庫調校功力來達成使用者期望的系統執行效能。

下述幾點是我個人認為在關連式檔案架構規劃上，會影響系統效能的重大考量：

Embarcadero DatabaseGear 跨多資料庫的資料庫管理工具權威

(一)正規化/非正規化取捨上-正規化過度，跑報表需 Join 很多的檔案

正規化的目的是希望多個資料檔案間能夠保有一致性，避免檔案重複存放，以免同樣意義的內容欄位，因改 A 檔案而忘了改 B 檔案，造成資料整體的不一致。

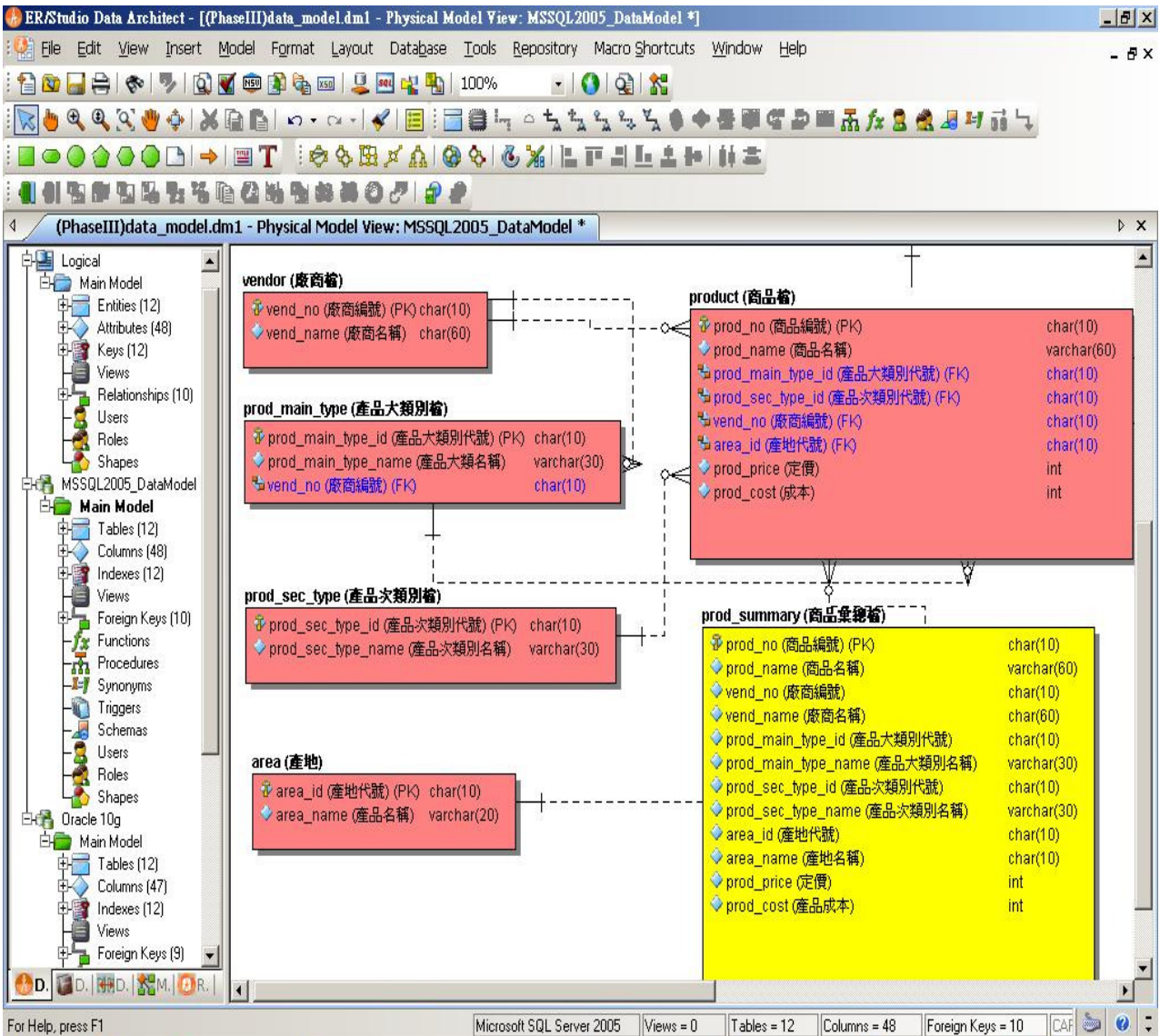
但往往在一個複雜的關連式的檔案架構與報表格式上的需求，因為資料量太大，在加上正規化作的夠徹底，導致在跑報表時，因為資料庫要 Join 太多的 Table，再加上錯綜複雜條件，而導致資料庫抓取資料的時間過久。

有些人在設計檔案架構的時候，在考量執行效能時，會把它某些檔案設計成非正規化，為了能夠減少一些檔案的 Join，來提昇系統執行效能。

所以在兼顧系統效能的情形下，在設計檔案架構時，正規化/非正規化的取捨的平衡點，就變得很重要了。

在此我提供一個案例做參考，即可兼顧正規化又能兼顧系統效能，提供給各位做參考。

[表 A] 商品相關檔案架構



ER/STUDIO 資料建模工具權威

- 官方網址:

<http://www.embarcadero.com/products/er-studio>

- 先前研討會簡報資料:

http://www.sinter.com.tw/codegear/seminars/seminar_doc/20100115ERSTUDIO.pdf

上述[表 A]是商品相關檔案架構，其中商品彙總檔的功能主要是針對大部分商品相關報表常用的欄位，匯集成一個檔案，當在跑報表的時候，可以省去很多檔案的 Join，以提升執行效能。

這個商品彙總檔案的產生時機，可以在執行系統日結時，由日結程式產生一份最新的商品彙總資料。可能有人會問到，那平常上班時間，若有人透過維護程式去更動到商品相關資料的時候，商品彙總檔要如何更新?這個部份可以在維護程式要更動商品模組資料的時候，也同時更新商品彙總檔的資料，畢竟這樣的資料更動的負擔不是很大，但受益的是幾十隻甚至上百隻的報表程式。

(二)是否需要統計檔-產生報表的資料要從統計檔抓?或者從明細檔抓?

有沒有需要設計統計檔這件事的考量，跟我們平常在做系統的時候，對於使用者報表上的需求，有沒有問到報表的使用週期以及報表彙總的日期單位有關係，使用者跑報表的週期是如何(日、週、月、季、年)?看資料彙總的日期單位是如何(日、週、月、季、年)?以及資料量是多少?最後您可以做一張統計表，針對每種業務報表需求做統計(日報表、週報表、月報表、季報表、年報表各有多少張?)彙集這些資訊，接下來就可以評估是否要統計檔這件事。

設計統計檔的負擔是每次只要有新的明細資料進來，就要花時間去更新統計檔；不過如果報表的需求量很大，當有設計統計檔的情形下，更新統計檔一次，但後續跑報表的速度就會很快，要抓的資訊不用再從明細檔抓完再做加總。若在沒有統計檔的情況下，所有彙總資訊都只能從明細檔抓，再加上報表的需求以及資料量都大的情形下，系統效能會非常得吃不消，這部分就不是 DBA 可以解決的部分了。

(三)檔案是否要做切割-交易資料是一個檔或者需要多個檔?或者要做 Table Partition?

例如交易檔是否要切成多個檔?主要是取決於日常產生交易的筆數。假定原始銷售的交易資料如果一個月有一百萬筆資料量，一年就會有一千兩百萬筆資料，到第五年就會有六千萬筆資料。若銷售檔案只架構成一個，您會發現系統隨著上線時間越長，系統執行效能就會越差。

Embarcadero DatabaseGear 跨多資料庫的資料庫管理工具權威

這時候有兩種方式可以解決，第一種是把交易檔依一定的日期單位來切割檔案，假定是要切到月檔，讓交易檔依照月份存放，避免我們在跑報表時要去參考太大的檔案，由其是報表上的需求 Join 很多 Table。不過成本就是程式會比較不好寫，由其考慮使用者要看的資料範圍有跨月的情形下，你就無法使用單一的 SQL 語法來抓取你要的資料，這個時候你可以要搭配 Store Procedure、Cursor、暫存性 Table 以及動態 SQL 的技術來完成。

另外，如果您用的是較新版本的 Oracle 或者是 MSSQL，您可以使用 Table Partition 的功能，讓您擁有一個 Table Name，讓資料庫用邏輯的方式幫您切割月檔，讓您可以擁有開月檔的效能，讓您在存取資料時擁有存取一個 Table 的方便性，可以省去複雜程式的撰寫工作。

(四)是否需要分歷史檔-歷史檔要怎麼分?

這部份也是在需求訪談的時候，可以了解一下使用者通常看資料的範圍大約是多久以及資料量是多少。假定使用者大部份看資料的範圍是兩年，可評估把超過兩年的資料歸在歷史檔，那您就可以寫一個過檔程式，負責把超過兩年的資料歸到歷史檔去。

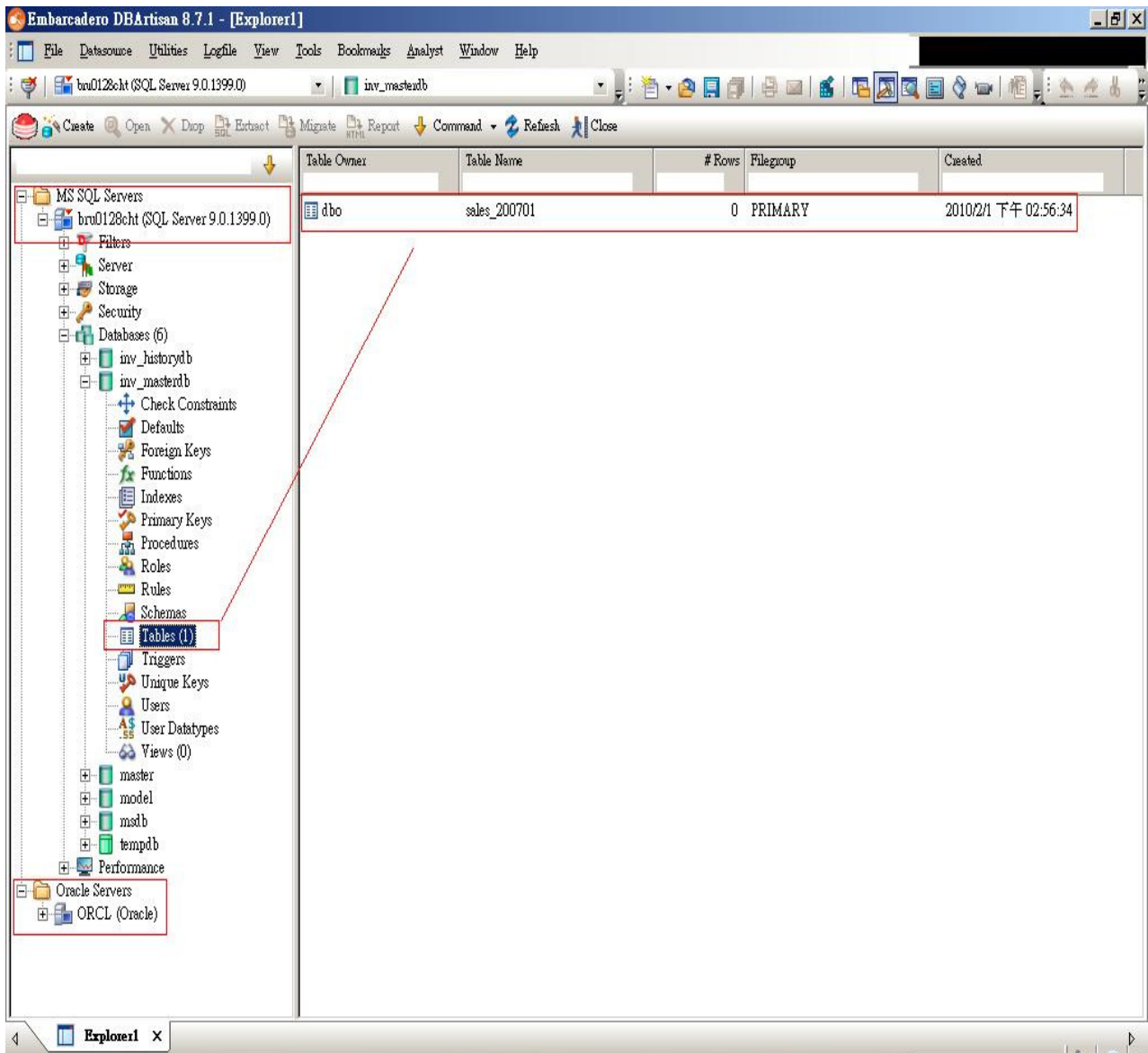
現在在硬碟空間很便宜的情形下，有時候不見得要把過久的歷史資料刪除掉，只要把最近兩年會用的資料庫與歷史檔的資料用不同的資料庫來區隔，歷史資料庫也可以用年來切資料庫，這樣好處是每次備份資料庫時，不用整個資料庫都備份，每天只要備份現行資料庫即可(兩年內會用到的資料庫)，每個月才備份全部的資料庫。若要不改程式的情況下，程式如何可以同時讀取現行資料庫的檔案以及歷史資料庫的檔案呢?以下以 MSSQL 為例：

Ex：假設有兩個資料庫，現行資料庫叫做 inv_masterdb，歷史資料庫叫做 inv_historydb，假定報表程式要去讀取 inv_masterdb 底下的 sales_200701 table 的資料，後來執行了過檔式後，sales_200701 table 的資料被移到 inv_historydb。這樣前後的變動，要如何讓前端的程式，在更動程式的情形下，一樣可以執行，甚至可以說，前端程式根本不知道，sales_200701 table 已經換位置了，這是怎麼做到的呢?

DBARTISAN 跨多廠資料庫管理工具權威

- 官方網址：
<http://www.embarcadero.com/products/dbartisan>
- 先前研討會簡報資料：
http://www.sinter.com.tw/codegear/seminars/seminar_doc/embt_dbartisan.pdf

[表 B] 執行過檔前



Embarcadero DatabaseGear 跨多資料庫的資料庫管理工具權威

[表 C] 執行過檔後

The screenshot shows the Embarcadero DBArtisan 8.7.1 interface. On the left, the Explorer pane shows a tree view of databases, including 'inv_historydb' and 'inv_masterdb'. The 'inv_masterdb' database is expanded, showing a table named 'sales_200701' in the 'dbo' schema. The main window displays the DDL Editor for 'View dbo.sales_200701'. The SQL script in the editor is as follows:

```
1 USE inv_masterdb
2 go
3 IF OBJECT_ID(N'dbo.sales_200701') IS NOT NULL
4 BEGIN
5     DROP VIEW dbo.sales_200701
6     IF OBJECT_ID(N'dbo.sales_200701') IS NOT NULL
7         PRINT N'<<< FAILED DROPPING VIEW dbo.sales_200701 >>>'
8     ELSE
9         PRINT N'<<< DROPPED VIEW dbo.sales_200701 >>>'
10 END
11 go
12 create view sales_200701
13 as
14 select * from inv_historydb.dbo.sales_200701
15 go
16 IF OBJECT_ID(N'dbo.sales_200701') IS NOT NULL
17     PRINT N'<<< CREATED VIEW dbo.sales_200701 >>>'
18 ELSE
19     PRINT N'<<< FAILED CREATING VIEW dbo.sales_200701 >>>'
20 go
21
```

秘訣只是在我們在下 SQL 命令的時候，我們的程式一直都是指向 inv_masterdb，而我們利用 select from 後面接的可以是 table name 或者是 view name 的特性，利用過檔程式把，sales_200701 的 table 從 inv_masterdb 搬到 inv_historydb 後，然再把原 inv_masterdb 的 sales_200701 table drop 掉，並且在 inv_masterdb 建立一個名稱是 sales_200701 的 view，而 view 的內容指向 inv_historydb 的 sales_200701 table。這樣報表程式就不會因為過檔程式更動了 sales_200701，而需要去更改報表程式，請參考上面的[表 B]與[表 C]。