

## 如何利用 SQL Optimizer 調校複雜 SQL

很多人在使用 SQL Optimizer 時常遇到複雜的 SQL 要分析很久，或是提供不出有任何建議改寫方法，在 Quest 技術人員提供一些經驗分享，讓讀者也能更善用此工具來處理類似問題。首先，說明為何執行分析 SQL 時會沒有 SQL 建議跑出來，這主要原因是因為在我們的 PC 中可能只有 2GB 的記憶體，而在處理複雜的 SQL 時因等待的時間過久造成分析因此而中斷，所以並非 SQL Optimizer 無法處理複雜問題。以下舉個實例複雜 SQL 供參考：

*The following SQL was created to mimic the original SQL statement for discussion and education only.*

The Case SQL statement

```
SELECT a.apc_cdr,
       a.crfc_no,
       a.crfc_grp_type_cdr,
       c.pdc_cdr,
       e.ccr_grp_cdr,
       long_term,
       CASE long_term WHEN 'Y' THEN d.more_12_wgh_ftr_pc
                WHEN 'N' THEN d.lss_12_wgh_ftr_pc
                ELSE NULL END AS wgh_ftr_pc
FROM tbp108 a,
     (SELECT a.ld_dt,
            a.ld_type_cdr,
            a.apc_cdr,
            a.crfc_no,
            CASE WHEN nbr_month = 12 THEN 'Y'
                 ELSE 'N' END AS long_term
     FROM tbp108 a,
     (SELECT apc_cdr,
            crfc_no,
            NVL(osd_am, 0),
            COUNT(*) AS nbr_month
     FROM tbp108
     WHERE ld_type_cdr = 'M'
            AND ld_dt >= ADD_MONTHS((SELECT MAX(ld_dt)
```

*March 10 M-Power eNew*

本篇文章版權為倍力資訊股份有限公司所有，未經書面同意，嚴禁複製、轉載

```
FROM tbp104
WHERE ld_type_cdr = 'M'
AND vsn_cdr <> '2'),

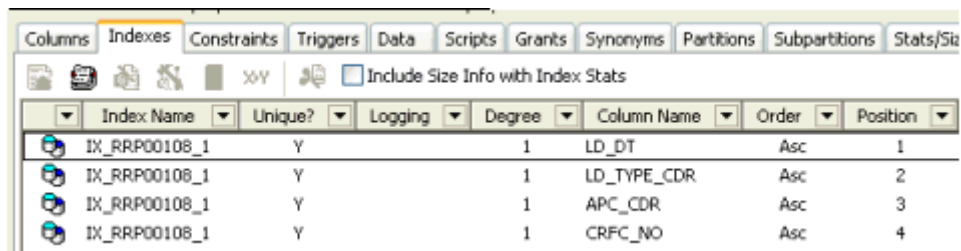
-11)
GROUP BY apc_cdr, crfc_no, NVL(osd_am, 0)
HAVING COUNT(*) = 12) b
WHERE a.ld_dt = (SELECT MAX(ld_dt)
FROM tbp104
WHERE ld_type_cdr = 'M'
AND vsn_cdr <> '2')
AND a.ld_type_cdr = 'M'
AND a.apc_cdr = b.apc_cdr (+)
AND a.crfc_no = b.crfc_no (+)
UNION
SELECT a.ld_dt,
a.ld_type_cdr,
a.apc_cdr,
a.crfc_no,
'N' AS long_term
FROM tbp108 a
WHERE a.ld_dt = to_date('02/10/2008', 'dd/mm/yyyy')
AND a.ld_type_cdr = TRIM('D')
AND a.cmt_ic = '1'
AND a.eur_cmt_am - NVL(a.eur_osd_am, 0) > 0
AND NOT EXISTS (SELECT *
FROM tbp108 p
WHERE ld_type_cdr = 'M'
AND ld_dt = (SELECT MAX(ld_dt)
FROM tbp104
WHERE ld_type_cdr = 'M'
AND vsn_cdr <> '2')
AND p.apc_cdr = a.apc_cdr
AND p.crfc_no = a.crfc_no)) b,
tbp109 c,
tbp072 d,
tbp069 e
```

```

WHERE a.ld_dt = to_date('02/10/2008', 'dd/mm/yyyy')
      AND a.ld_type_cdr = TRIM('D')
      AND a.eur_cmt_am - NVL(a.eur_osd_am, 0) > 0
      AND a.cmt_ic = '1'
      AND a.apc_cdr = b.apc_cdr
      AND a.crfc_no = b.crfc_no
      AND a.ld_dt = c.ld_dt
      AND a.ld_type_cdr = c.ld_type_cdr
      AND a.apc_cdr = c.apc_cdr
      AND a.crfc_no = c.crfc_no
      AND a.ld_dt = d.ld_dt
      AND a.ld_type_cdr = d.ld_type_cdr
      AND a.rvv_cdr = d.rvv_cdr
      AND c.pdc_cdr = d.pdc_cdr
      AND c.spdc_cdr = d.spdc_cdr
      AND (d.dlt_cdr = 0
           OR d.dlt_cdr IS NULL)
      AND a.ld_dt = e.ld_dt
      AND a.ld_type_cdr = e.ld_type_cdr
      AND a.rvv_cdr = e.rvv_cdr
      AND c.pdc_cdr = e.pdc_cdr
      AND c.spdc_cdr = e.spdc_cdr
      AND (e.dlt_cdr = 0
           OR e.dlt_cdr IS NULL)
ORDER BY a.apc_cdr, a.crfc_no, e.ccr_grp_cdr
    
```

### Table Sizes

Table Name	Size	Number of Rows
TBP108	2.36G	1,777,110

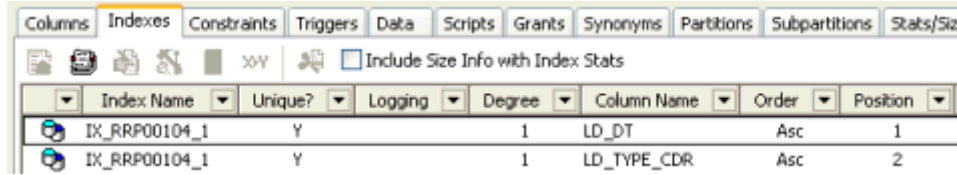


Index Name	Unique?	Logging	Degree	Column Name	Order	Position
IX_RRP00108_1	Y		1	LD_DT	Asc	1
IX_RRP00108_1	Y		1	LD_TYPE_CDR	Asc	2
IX_RRP00108_1	Y		1	APC_CDR	Asc	3
IX_RRP00108_1	Y		1	CRFC_NO	Asc	4

*March 10 M-Power eNew*

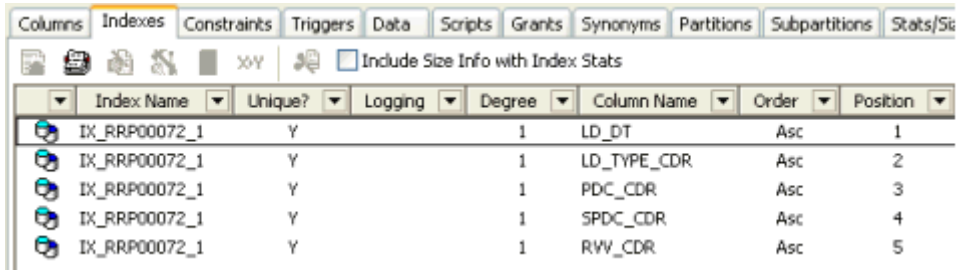
本篇文章版權為倍力資訊股份有限公司所有，未經書面同意，嚴禁複製、轉載

Table Name	Size	Number of Rows
TBP104	230 Mb	31



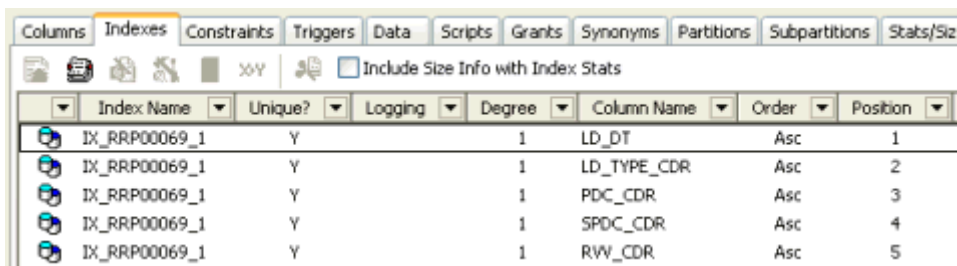
Index Name	Unique?	Logging	Degree	Column Name	Order	Position
IX_RRP00104_1	Y		1	LD_DT	Asc	1
IX_RRP00104_1	Y		1	LD_TYPE_CDR	Asc	2

Table Name	Size	Number of Rows
TBP072	230 Mb	3,497



Index Name	Unique?	Logging	Degree	Column Name	Order	Position
IX_RRP00072_1	Y		1	LD_DT	Asc	1
IX_RRP00072_1	Y		1	LD_TYPE_CDR	Asc	2
IX_RRP00072_1	Y		1	PDC_CDR	Asc	3
IX_RRP00072_1	Y		1	SPDC_CDR	Asc	4
IX_RRP00072_1	Y		1	RVW_CDR	Asc	5

Table Name	Size	Number of Rows
TBP069	230 Mb	5,180

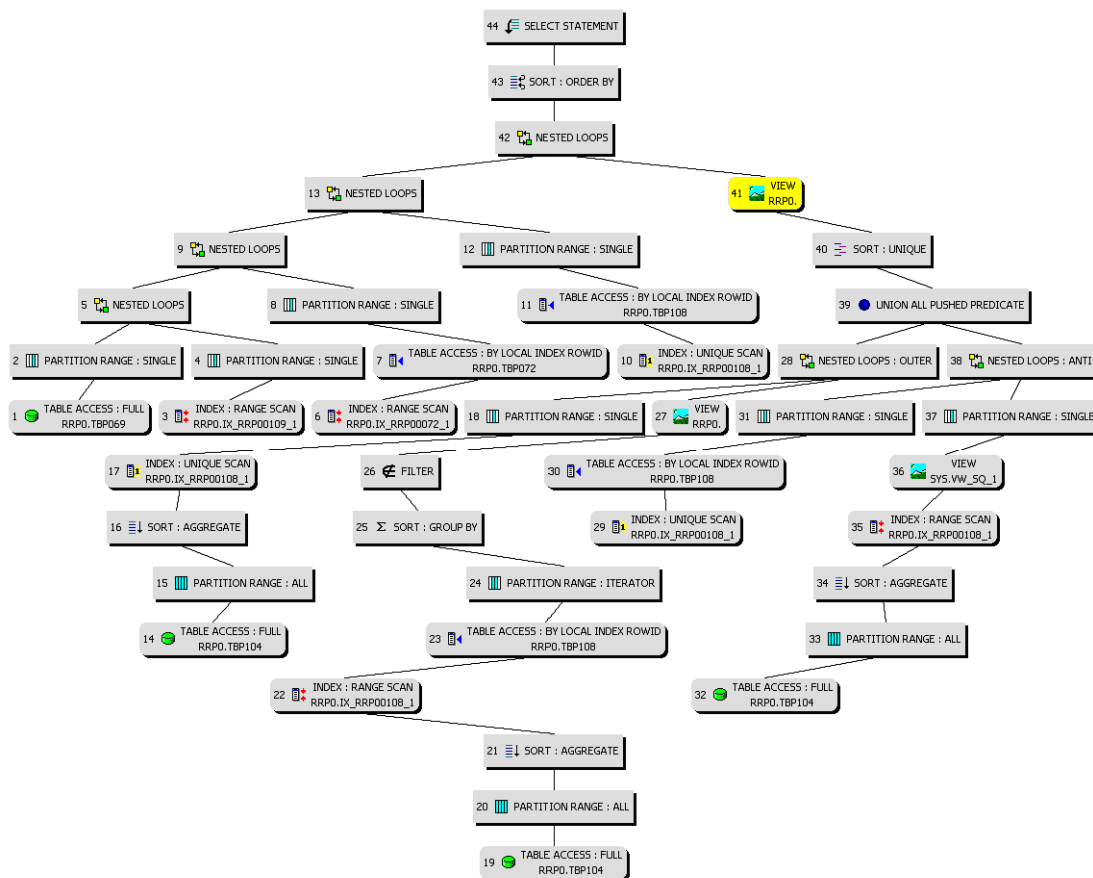


Index Name	Unique?	Logging	Degree	Column Name	Order	Position
IX_RRP00069_1	Y		1	LD_DT	Asc	1
IX_RRP00069_1	Y		1	LD_TYPE_CDR	Asc	2
IX_RRP00069_1	Y		1	PDC_CDR	Asc	3
IX_RRP00069_1	Y		1	SPDC_CDR	Asc	4
IX_RRP00069_1	Y		1	RVW_CDR	Asc	5

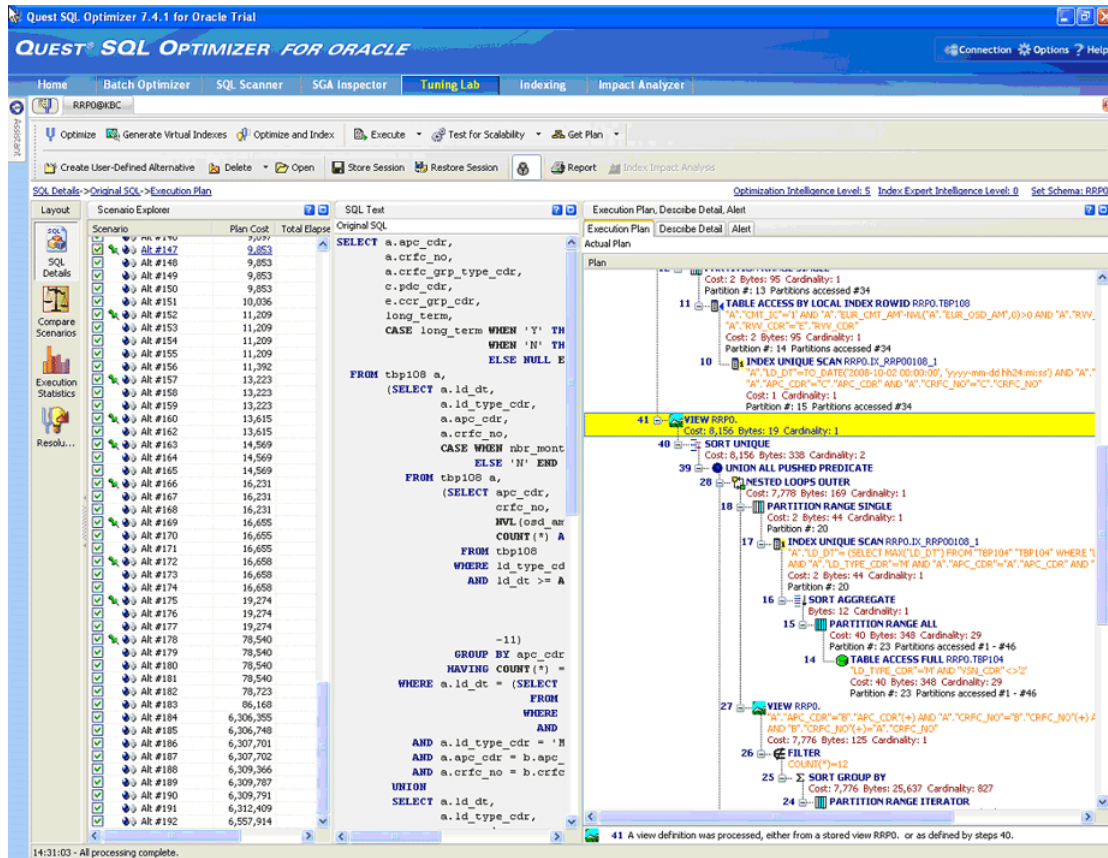
Table Name	Size	Number of Rows
TBP109	525 Mb	3,753,720

Index Name	Unique?	Logging	Degree	Column Name	Order	Position
IX_RRP00109_1	Y		1	LD_DT	Asc	1
IX_RRP00109_1	Y		1	LD_TYPE_CDR	Asc	2
IX_RRP00109_1	Y		1	PDC_CDR	Asc	3
IX_RRP00109_1	Y		1	SPDC_CDR	Asc	4
IX_RRP00109_1	Y		1	APC_CDR	Asc	5
IX_RRP00109_1	Y		1	CRFC_NO	Asc	6

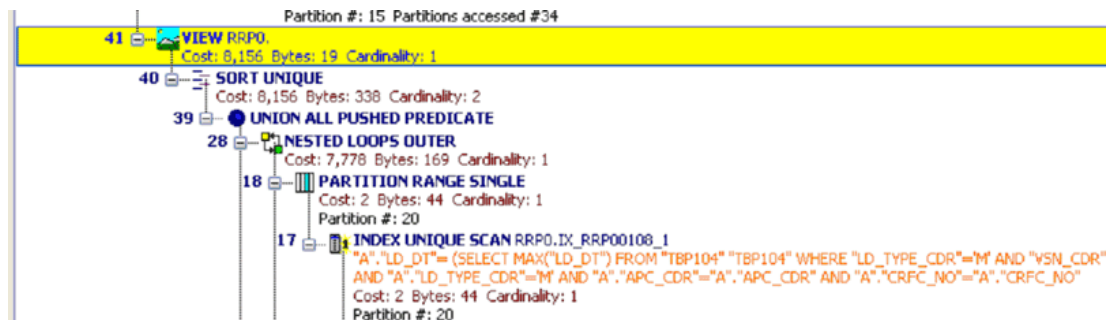
在這個 Case 中 Quest 技術人員只有三天時間必須完成 SQL 效能調校，因此他花了 20 分鐘並中斷執行原本可能要跑 8 個小時才能完成的 SQL，產生如下的 Query Plan



他使用的 PC 也是 2GB memory，並將 Intelligence Level 設為 5，執行 10 分鐘後產生 192 組建議，然而如果每一組分析下來可能需要花 64 天(192\*8hrs)，這實際上是不可行的，所以重點來了，利用上乘的速讀功力加上棋聖張栩思慮縝密，來分析這 192 組必定可完成不可能任務，可惜我們都來不及栽培了。這位 Quest 老兄用了一個技巧“user-defined time”來過濾過多的組數，設定 10 分鐘，只要執行超過 10 分鐘即換下一組分析，這招倒是值得學習。



總共花了 32 小時執行完成，但很不幸的花了一天還無法很快找到效能好的 SQL 建議，只好中斷執行改採一一瀏覽 Query Plan(只能說是本能反應)，突然間看到一個 View 是 nested-loop 另一 Table return Cardinality=1，如下圖所示。



View 基本上是一個暫存的 Table，是很多個 Query 組成，假使處理這個 View 只執行一次，再去 Join 其他 Table 就沒有問題，但是這個 View 的 Cardinality=1，所以 Optimizer 會選擇用 nested-loop JOIN 方式，而偏偏這個 View 又相當複雜，所以 Optimizer 不易估算成本。在原始 SQL 中有標紅色及綠色部份，針對這部份查詢約有 66,614 筆資料，加上這個 Nested-Loop 可想而知反應的效能一定很低，如何解決呢？

要解決這個問題，首先先找出那些建議的 SQL 中針對這個 View 有採用 Hash Join 或是 Sort Merge Join，在 Alt #161 有找到並執行 12 分鐘，之前設定是控制在 10 分鐘內的所以這筆被忽略掉，光是這 12 分鐘就比原有的 SQL 快了 40 倍之多。

March 10 M-Power eNew

本篇文章版權為倍力資訊股份有限公司所有，未經書面同意，嚴禁複製、轉載

The screenshot shows the 'Actual Plan' for a query. The plan starts with a 'HASH JOIN' (step 40) which is a 'NESTED LOOPS' join. It involves several 'HASH JOIN' steps (0, 5, 11) and 'PARTITION RANGE SINGLE' steps (2, 4, 7, 11). The plan also includes 'TABLE ACCESS FULL' and 'INDEX FAST FULL SCAN' operations. The cost of the entire plan is 13,614 bytes and 224 cardinality.

所以在原 SQL 中紅色部份是最耗資源的，改善了這部份就大勢抵定，接下來在原 SQL 中綠色部份還可以透過分析建議出 5 組 SQL，原 SQL 綠色部份 sub query 大約執行 9 分多，而在 Alt #5 只有執行 17 秒，這部份又是可以再縮短 12 分鐘的執行，以下是執行內容。

The screenshot shows the 'Compare' window with the following data:

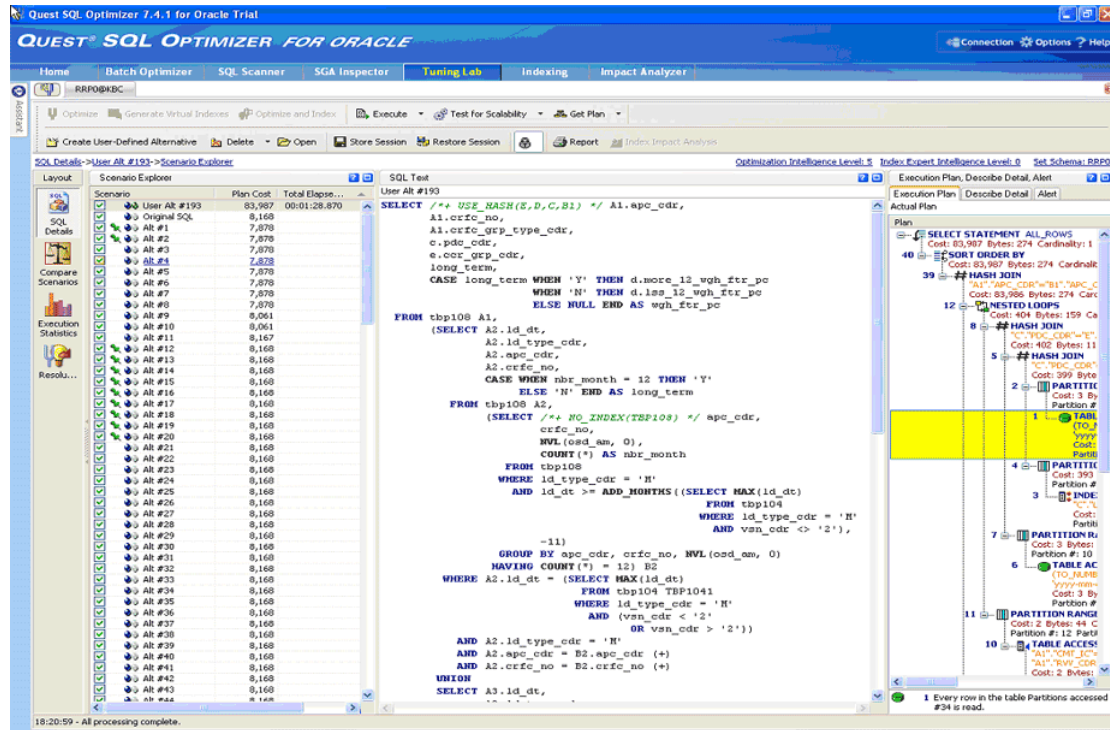
Scenario	Results Comparison	Plan Cost	Optimize: Total Elapsed Time	Total Elapsed ...	Total CPU	First Row Elapsed	Last Row Elapsed	Physical Reads	Logical Reads	Sc
Original SQL	Identical	78,148	7,776	00:00:17.700	0.00	00:00:16.700	00:00:01.000	0	0	
Alt #1		7,776	7,776	00:00:04.660	0.00	00:00:03.910	00:00:00.750	0	0	
Alt #2		7,797	7,797							
Alt #3		7,797	7,797							
Alt #4	Query Canceled, Ex...	8,012								

The 'Actual Plan' for Alt #5 shows a 'SELECT STATEMENT ALL\_ROWS' with a 'HASH GROUP BY' operation. The cost is 7,776 bytes and 25,637 cardinality. The plan includes a 'PARTITION RANGE ITERATOR', 'TABLE ACCESS FULL', 'SORT AGGREGATE', and 'PARTITION RANGE ALL' steps.

March 10 M-Power eNew

本篇文章版權為倍力資訊股份有限公司所有，未經書面同意，嚴禁複製、轉載

各位讀者一定會發現 Alt #5 是不用 index 的 Hint，結果還比使用 index 還快。整個調校即將完成，分別調整二組 Sub Query，將原本執行 8 小時的 SQL，最後執行約 1 分 28 秒，也在 3 天時限內完成了複雜的 SQL 調校。



從以上的調校過程中，我們可以學習到幾點：

1. 一個複雜的 SQL 也是由幾個部份 SQL 所組成的，只要針對可疑的部份著手應該就好處理。
2. SQL Optimizer 有時無法一次就建議你最好的 SQL 寫法，主要原因是時間的關係，我們沒有那麼多的時間可以等待每一種可能的寫法去執行分析，所以，有時要學會化整為零來分析，這樣會比較有效率。
3. Oracle Optimizer 會依據資料量，資料分布，資料重複性等統計資訊分析，但未必會建議出理想的 Query Plan，有時是統計資訊的更新問題，有時是查詢的需求特性，Optimizer 無法一一考量。
4. 凡是次查詢，Join 部份都是我們要注意的地方。
5. 這位 Quest DBA 專家也是靠工具幫助他進行分析，否則要自己想 SQL 及分析可能要花更時間，學習的效果也會打折的。

【參考來源】

Quest 官方技術文件。