前　　言

近年來，隨著系統的資料量逐年增加，平行處理的量也成倍增長，SQL 效能越來越成為 IT 系統設計和開發時首要考慮的問題之一。SQL 效能問題已經逐步發展成為資料庫效能的首要問題，80%的資料庫效能問題都是因 SQL 而導致。面對日益增多的 SQL 效能問題，如何下手以及如何提前審核已經成為越來越多的 IT 從業者必須要考慮的問題。

現在將 8 年專職 SQL 最佳化的經驗和心得與大家一起分享，以揭開 SQL 最佳化的神秘面紗，讓一線工程師在實際開發中不再寢食難安、聞聲色變，最終能夠對 SQL 最佳化技能駕輕就熟。

編寫本書也是對多年學習累積的一個總結，鞭策自己再接再厲。如果能夠給各位讀者在 SQL 最佳化上提供一點幫助，也不枉個中辛苦。

2014 年，作者羅炳森與有教無類（網名）聯合編寫了《Oracle 查詢優化改寫技巧與案例》一書，該書主要側重於 SQL 最佳化改寫技巧。到目前為止，該書仍然是市面上唯一一本專門講解 SQL 改寫技巧的圖書。

因為《Oracle 查詢優化改寫技巧與案例》只專注於 SQL 改寫技巧，並沒有談到 SQL 最佳化的具體思維、方法和步驟，所以可以將本書視為對《Oracle 查詢優化改寫技巧與案例》一書的進一步補充。

本書共 10 章，各章的主要內容如下：

第 1 章詳細介紹了 SQL 最佳化的基礎知識以及初學者切實需要掌握的基本內容，本章可以幫助初學者快速入門。

第 2 章詳細講解統計資訊定義、統計資訊的重要性、統計資訊相關參數設定方案以及統計資訊收集策略。

碁峰 www.gotop.com.tw

第 3 章詳細講解執行計畫、各種執行計畫的使用場景以及執行計畫的閱讀方法，透過制定執行計畫，讀者可以快速找出 SQL 效能瓶頸。

第 4 章詳細講解常見的存取路徑，這是閱讀執行計畫中比較重要的環節，需要掌握各種常見的存取路徑。

第 5 章詳細講解表的各種連接方式、各種表連接方式的等價改寫以及相互轉換，這也是本書的核心章節。

第 6 章介紹單表存取以及索引掃描的成本計算方法，並由此引出 SQL 最佳化的核心思維。

第 7 章講解常見的查詢變換，分別是子查詢非巢狀嵌套、檢視合併和謂詞推入。如果要對複雜的 SQL（包含各種子查詢的 SQL）進行最佳化，讀者就必須掌握查詢變換技巧。

第 8 章講解各種最佳化技巧，其中涵蓋分頁語句最佳化思維、分析函數減少資料表掃描次數、超大表與超大表關聯最佳化方法、dblink 最佳化思考方式，以及大表的 rowid 切片最佳化技巧。掌握這些調校技巧往往能夠事半功倍。

第 9 章分享在 SQL 最佳化實戰中遇到的經典案例，讀者可以在欣賞 SQL 最佳化案例的同時學習羅老師多年專職 SQL 最佳化的經驗，同時學到很多具有實戰意義的最佳化思維以及最佳化方法與技巧。

第 10 章講解全自動 SQL 審核，將有效能問題的 SQL 扼殺在「搖籃」裡，確保系統上線之後，不會因為 SQL 寫法導致效能問題，並且還能找出不符合 SQL 編碼規範但是已經上線的 SQL。

本書對系統面臨效能壓力挑戰的一線工程師、營運與維護工程師、資料庫管理員（DBA）、系統設計與開發人員具有極大的參考價值。

為了滿足不同層次的讀者需求，本書在寫作的內容上儘量由淺入深，前 5 章比較淺顯易懂，適合 SQL 最佳化初學者閱讀。通讀完前 5 章之後，初學者能夠對 SQL 最佳化有一定認識。後 5 章屬於進階和高級內容，適合有一定基礎的人閱讀。通讀完後 5 章的內容之後，無論是初學者或是有一定基礎的讀者都能從中獲益良多。

碁峰 www.gotop.com.tw

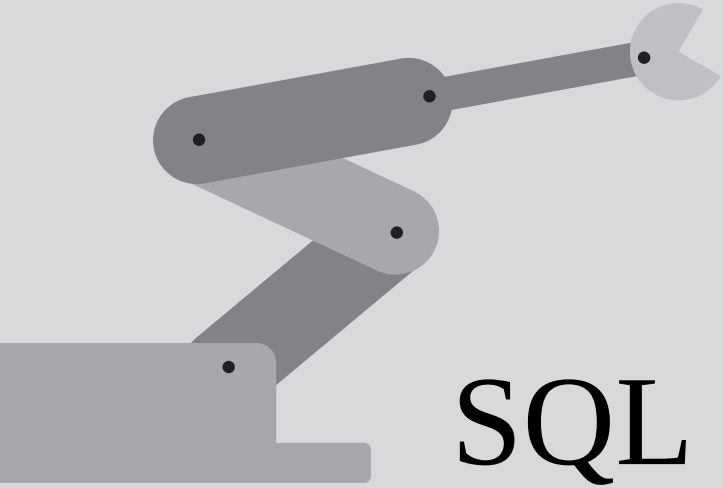
本書專注於 SQL 最佳化技巧，因此書中不會談太多資料庫系統最佳化的內容。

雖然本書是以 Oracle 為基礎來編寫的，但是關聯式資料庫的最佳化方法都殊途同歸，因此無論是 DB2 從業者、SQL SERVER 從業者、MYSQL 從業者，亦或是 PostGre SQL 從業者等，都能從本書中學到所需要的 SQL 最佳化知識。

因筆者能力有限，本書在編寫過程中難免有錯漏之處，懇請讀者批評、指正。聯繫我們的方式如下：692162374@qq.com（QQ 好友數已達上限）或者 327165427@qq.com（新開 QQ 帳號）。

如果有讀者想進一步學習 SQL 最佳化技能或者一些公司或機構需要開展 SQL 最佳化方面的教育訓練，都可以聯絡作者。另外，作者還開設了實體教育訓練課程，可零基礎學習，結業後可以順利就業，歡迎與羅老師聯繫。

碁峰 www.gotop.com.tw



第 1 章

SQL 最佳化必懂概念

碁峰
www.gotop.com.tw

1.1 基數（CARDINALITY）

某個欄唯一鍵（Distinct_Keys）的數量叫作基數。比如性別欄，該欄只有男女之分，所以這一欄基數是 2。主鍵欄的基數等於表的總列數。基數的高低影響欄的資料分佈。

以測試表 test 為例，owner 欄和 object_id 欄的基數分別如下所示。

```
SQL> select count(distinct owner),count(distinct object_id),count(*) from test;

COUNT(DISTINCTOWNER) COUNT(DISTINCTOBJECT_ID)   COUNT(*)
-------------------- ------------------------ ----------
                  29                    72462      72462
```

TEST 表的總列數為 72462，owner 欄的基數為 29，說明 owner 欄裡面有大量重複值，object_id 欄的基數等於總列數，說明 object_id 欄沒有重複值，相當於主鍵。owner 欄的資料分佈如下。

```
SQL> select owner,count(*) from test group by owner order by 2 desc;

OWNER                  COUNT(*)
-------------------- ----------
SYS                       30808
PUBLIC                    27699
SYSMAN                     3491
ORDSYS                     2532
APEX_030200                2406
MDSYS                      1509
XDB                         844
OLAPSYS                     719
SYSTEM                      529
CTXSYS                      366
WMSYS                       316
EXFSYS                      310
SH                          306
ORDDATA                     248
OE                          127
DBSNMP                       57
IX                           55
HR                           34
PM                           27
FLOWS_FILES                  12
OWBSYS_AUDIT                 12
ORDPLUGINS                   10
OUTLN                         9
BI                            8
SI_INFORMTN_SCHEMA            8
ORACLE_OCM                    8
SCOTT                         7
```


碁峰 www.gotop.com.tw

```
APPQOSSYS                      3
OWBSYS                         2
```

owner 欄的資料分佈極不均衡，我們執行如下 SQL。

```
select * from test where owner='SYS';
```

SYS 有 30808 條資料，從 72462 條資料裡面查詢 30808 條資料，也就是說要返回表中 42.5%的資料。

```
SQL> select 30808/72462*100 "Percent" from dual;

   Percent
----------
42.5160774
```

那麼請思考，你認為以上查詢應該使用索引嗎？現在我們換一種查詢語法。

```
select * from test where owner='SCOTT';
```

SCOTT 有 7 條資料，從 72462 條資料裡面查詢 7 條資料，也就是說要返回表中 0.009%的資料。

```
SQL> select 7/72462*100 "Percent" from dual;

   Percent
----------
.009660236
```

請思考，返回表中 0.009%的資料應不應該走索引？

如果你還不懂索引，沒關係，後面的章節我們會詳細介紹。如果你回答不了上面的問題，我們先提醒一下。當查詢結果是返回表中 5% 以內的資料時，應該走索引；當查詢結果返回的是超過表中 5% 的資料時，應該走全資料表掃描。

當然了，返回表中 5% 以內的資料走索引，返回超過 5% 的資料就使用全資料表掃描，這個結論太絕對了，因為你還沒掌握後面章節的知識，這裡暫且記住 5% 這個界限就行。我們之所以在這裡講 5%，是怕一些初學者不知道上面問題的答案而糾結。

現在有如下查詢語法。

```
select * from test where owner=:B1;
```


碁峰 www.gotop.com.tw

語法中，「:B1」是綁定變數，可以傳入任意值，該查詢可能走索引也可能走全資料表掃描。

現在得到一個結論：如果某個欄基數很低，該欄資料分佈就會非常不均衡，因為該欄資料分佈不均衡，會導致 SQL 查詢可能走索引，也可能走全資料表掃描。在做 SQL 最佳化的時候，如果懷疑列資料分佈不均衡，我們可以使用「select 欄, count(*) from 表 group by 欄 order by 2 desc」來查看欄的資料分佈。

如果 SQL 語法是單表存取，那麼可能走索引、可能走全資料表掃描，也可能走實體化檢視（materialized view）掃描。在不考慮有實體化檢視的情況下，單表存取要麼走索引，要麼走全資料表掃描。現在，回憶一下走索引的條件：返回表中 5% 以內的資料走索引，超過 5% 的時候走全資料表掃描。相信大家讀到這裡，已經搞懂了單表存取的最佳化方法。

我們來看如下查詢。

```
select * from test where object_id=:B1;
```

不管 object_id 傳入任何值，都應該走索引。

我們再思考如下查詢語法。

```
select * from test where object_name=:B1;
```

不管給 object_name 傳入任何值，請問該查詢應該走索引嗎？

請你去查看 object_name 的資料分佈。寫到這裡，其實有點想把本節名稱改為「資料分佈」。大家在以後的工作中一定要注意列的資料分佈！

碁峰 www.gotop.com.tw

1.2 選擇性（SELECTIVITY）

基數與總列數的比值再乘以 100% 就是某個欄位的選擇性。

在進行 SQL 最佳化的時候，單獨看欄位的基數是沒有意義的，基數必須對比總列數才有實際意義，正是因為這個原因，我們才引出了選擇性這個概念。

下面我們查看 test 表各個欄位的基數與選擇性，為了查看選擇性，必須先收集統計資訊。關於統計資訊，我們在第 2 章會詳細介紹。下面的腳本用於收集 test 表的統計資訊。

```
SQL> BEGIN
  2    DBMS_STATS.GATHER_TABLE_STATS(ownname          => 'SCOTT',
  3                                  tabname          => 'TEST',
  4                                  estimate_percent => 100,
  5                                  method_opt => 'for all columns size 1',
  6                                  no_invalidate    => FALSE,
  7                                  degree           => 1,
  8                                  cascade          => TRUE);
  9  END;
 10  /

PL/SQL procedure successfully completed.
```

下面的腳本用於查看 test 表中每個欄位的基數與選擇性。

```
SQL> select a.column_name,
  2         b.num_rows,
  3         a.num_distinct Cardinality,
  4         round(a.num_distinct / b.num_rows * 100, 2) selectivity,
  5         a.histogram,
  6         a.num_buckets
  7    from dba_tab_col_statistics a, dba_tables b
  8   where a.owner = b.owner
  9     and a.table_name = b.table_name
 10     and a.owner = 'SCOTT'
 11     and a.table_name = 'TEST';

COLUMN_NAME       NUM_ROWS CARDINALITY SELECTIVITY HISTOGRAM NUM_BUCKETS
--------------- ---------- ----------- ----------- --------- -----------
OWNER                72462          29         .04 NONE                1
OBJECT_NAME          72462       44236       61.05 NONE                1
SUBOBJECT_NAME       72462         106         .15 NONE                1
OBJECT_ID            72462       72462         100 NONE                1
DATA_OBJECT_ID       72462        7608        10.5 NONE                1
OBJECT_TYPE          72462          44         .06 NONE                1
CREATED              72462        1366        1.89 NONE                1
LAST_DDL_TIME        72462        1412        1.95 NONE                1
TIMESTAMP            72462        1480        2.04 NONE                1
```


碁峰 www.gotop.com.tw

```
STATUS               72462          1          0 NONE                1
TEMPORARY            72462          2          0 NONE                1
GENERATED            72462          2          0 NONE                1
SECONDARY            72462          2          0 NONE                1
NAMESPACE            72462         21        .03 NONE                1
EDITION_NAME         72462          0          0 NONE                0

15 rows selected.
```

請思考：什麼樣的欄位必須建立索引呢？

有人說基數高的欄位，有人說在 where 條件中的欄位。這些答案並不完美。基數高究竟是多高？沒有和總列數對比，始終不知道有多高。比如某個欄位的基數有幾萬列，但是總列數有幾十億列，那麼這個欄位的基數還高嗎？這就是要引出選擇性的根本原因。

當一個欄位選擇性大於 20%，說明該欄的資料分佈就比較均衡了。測試表 test 中 object_name、object_id 的選擇性均大於 20%，其中 object_name 欄的選擇性為 61.05%。現在我們查看該欄資料分佈（為了方便展示，只輸出前 10 列資料的分佈情況）。

```
SQL> select *
  2    from (select object_name, count(*)
  3            from test
  4           group by object_name
  5           order by 2 desc)
  6   where rownum <= 10;

OBJECT_NAME          COUNT(*)
------------------ ----------
COSTS                      30
SALES                      30
SALES_CHANNEL_BIX          29
COSTS_TIME_BIX             29
COSTS_PROD_BIX             29
SALES_TIME_BIX             29
SALES_PROMO_BIX            29
SALES_PROD_BIX             29
SALES_CUST_BIX             29
DBMS_REPCAT_AUTH            5

10 rows selected.
```

由上面的查詢結果我們可知，object_name 欄的資料分佈非常均衡。我們查詢以下 SQL。

碁峰 www.gotop.com.tw

```
select * from test where object_name=:B1;
```

不管 object_name 傳入任何值，最多返回 30 列資料。

什麼樣的列必須要新建索引呢？當一個列出現在 where 條件中，該列沒有新建索引並且選擇性大於 20%，那麼該欄就必須新建索引，從而提升 SQL 查詢效能。當然了，如果表只有幾百條資料，那我們就不用新建索引了。

下面拋出 SQL 最佳化核心思維第一個觀點：只有大表才會產生效能問題。

也許有人會說：「我有個表很小，只有幾百條，但是該表經常進行 DML，會產生熱點塊，也會出現效能問題。」對此我們並不想過多地討論此問題，這屬於應用程式設計問題，不屬於 SQL 最佳化的範疇。

下面我們將透過實驗為大家分享本書第一個全自動最佳化腳本。

抓出必須新建索引的欄位（請讀者對該腳本適當修改，以便用於生產環境）。

首先，該欄必須出現在 where 條件中，怎麼抓出表中的哪個欄位出現在 where 條件中呢？有兩種方法，一種是可以透過 V$SQL_PLAN 抓取，另一種是透過下面的腳本抓取。

先執行下面的儲存過程，更新資料庫監控資訊。

```
begin
  dbms_stats.flush_database_monitoring_info;
end;
```

執行完上面的命令之後，再執行下面的查詢語法就可以查詢出哪個表的哪個欄位出現在 where 條件中。

```
select r.name owner,
       o.name table_name,
       c.name column_name,
       equality_preds, ---等值過濾
       equijoin_preds, ---等值 JOIN 比如 where a.id=b.id
       nonequijoin_preds, ----不等 JOIN
       range_preds, ----範圍過濾次數 > >= < <= between and
       like_preds, ----LIKE 過濾
       null_preds, ----NULL 過濾
       timestamp
  from sys.col_usage$ u, sys.obj$ o, sys.col$ c, sys.user$ r
 where o.obj# = u.obj#
   and c.obj# = u.obj#
   and c.col# = u.intcol#
```


碁峰 www.gotop.com.tw

```
  and r.name = 'SCOTT'
  and o.name = 'TEST';
```

下面是實驗步驟。

我們首先執行一個查詢語法，讓 owner 與 object_id 欄出現在 where 條件中。

```
SQL> select object_id, owner, object_type
  2    from test
  3   where owner = 'SYS'
  4     and object_id < 100
  5     and rownum <= 10;

 OBJECT_ID OWNER                OBJECT_TYPE
---------- -------------------- -----------
        20 SYS                  TABLE
        46 SYS                  INDEX
        28 SYS                  TABLE
        15 SYS                  TABLE
        29 SYS                  CLUSTER
         3 SYS                  INDEX
        25 SYS                  TABLE
        41 SYS                  INDEX
        54 SYS                  INDEX
        40 SYS                  INDEX

10 rows selected.
```

其次更新資料庫監控資訊。

```
SQL> begin
  2    dbms_stats.flush_database_monitoring_info;
  3  end;
  4  /

PL/SQL procedure successfully completed.
```

然後我們查看 test 表有哪些欄位出現在 where 條件中。

```
SQL> select r.name owner, o.name table_name, c.name column_name
  2    from sys.col_usage$ u, sys.obj$ o, sys.col$ c, sys.user$ r
  3   where o.obj# = u.obj#
  4     and c.obj# = u.obj#
  5     and c.col# = u.intcol#
  6     and r.name = 'SCOTT'
  7     and o.name = 'TEST';

OWNER      TABLE_NAME COLUMN_NAME
---------- ---------- ------------------------------
SCOTT      TEST       OWNER
SCOTT      TEST       OBJECT_ID
```


碁峰 www.gotop.com.tw

接下來我們查詢出選擇性大於等於 20%的欄位。

```
SQL> select a.owner,
  2         a.table_name,
  3         a.column_name,
  4         round(a.num_distinct / b.num_rows * 100, 2) selectivity
  5    from dba_tab_col_statistics a, dba_tables b
  6   where a.owner = b.owner
  7     and a.table_name = b.table_name
  8     and a.owner = 'SCOTT'
  9     and a.table_name = 'TEST'
 10     and a.num_distinct / b.num_rows >= 0.2;

OWNER      TABLE_NAME COLUMN_NAME   SELECTIVITY
---------- ---------- ------------- -----------
SCOTT      TEST       OBJECT_NAME         61.05
SCOTT      TEST       OBJECT_ID             100
```

最後，確保這些欄位沒有新建索引。

```
SQL> select table_owner, table_name, column_name, index_name
  2    from dba_ind_columns
  3   where table_owner = 'SCOTT'
  4     and table_name = 'TEST';
未選定行
```

把上面的腳本組合起來，我們就可以得到全自動的最佳化腳本了。

```
SQL> select owner,
  2         column_name,
  3         num_rows,
  4         Cardinality,
  5         selectivity,
  6         'Need index' as notice
  7    from (select b.owner,
  8                 a.column_name,
  9                 b.num_rows,
 10                 a.num_distinct Cardinality,
 11                 round(a.num_distinct / b.num_rows * 100, 2) selectivity
 12            from dba_tab_col_statistics a, dba_tables b
 13           where a.owner = b.owner
 14             and a.table_name = b.table_name
 15             and a.owner = 'SCOTT'
 16             and a.table_name = 'TEST')
 17   where selectivity >= 20
 18     and column_name not in (select column_name
 19                               from dba_ind_columns
 20                              where table_owner = 'SCOTT'
 21                                and table_name = 'TEST')
 22     and column_name in
 23         (select c.name
 24            from sys.col_usage$ u, sys.obj$ o, sys.col$ c, sys.user$ r
 25           where o.obj# = u.obj#
 26             and c.obj# = u.obj#
```


碁峰 www.gotop.com.tw

```
 27             and c.col# = u.intcol#
 28             and r.name = 'SCOTT'
 29             and o.name = 'TEST');

OWNER      COLUMN_NAME     NUM_ROWS CARDINALITY SELECTIVITY NOTICE
---------- ------------- ---------- ----------- ----------- ----------
SCOTT      OBJECT_ID          72462       72462         100 Need index
```

1.3 直方圖（HISTOGRAM）

前面提到，當某個欄基數很低，該列資料分佈就會不均衡。資料分佈不均衡會導致在查詢該欄的時候，要麼走全資料表掃描，要麼走索引掃描，這個時候很容易走錯執行計畫。

如果沒有對基數低的欄位收集直方圖統計資訊，基於成本的最佳化器（CBO）會認為該欄資料分佈是均衡的。

下面我們還是以測試表 test 為例，用實驗講解直方圖。

首先我們對測試表 test 收集統計資訊，在收集統計資訊的時候，不收集列的直方圖，語法 `for all columns size 1` 表示對所有欄位都不收集直方圖。

```
SQL> BEGIN
  2    DBMS_STATS.GATHER_TABLE_STATS(ownname          => 'SCOTT',
  3                                  tabname          => 'TEST',
  4                                  estimate_percent => 100,
  5                                  method_opt      => 'for all columns size 1',
  6                                  no_invalidate    => FALSE,
  7                                  degree           => 1,
  8                                  cascade          => TRUE);
  9  END;
 10  /

PL/SQL procedure successfully completed.
```

Histogram 為 none 表示沒有收集直方圖。

```
SQL> select a.column_name,
  2         b.num_rows,
  3         a.num_distinct Cardinality,
  4         round(a.num_distinct / b.num_rows * 100, 2) selectivity,
  5         a.histogram,
  6         a.num_buckets
  7    from dba_tab_col_statistics a, dba_tables b
```


碁峰 www.gotop.com.tw

```
  8   where a.owner = b.owner
  9     and a.table_name = b.table_name
 10     and a.owner = 'SCOTT'
 11     and a.table_name = 'TEST';

COLUMN_NAME       NUM_ROWS CARDINALITY SELECTIVITY HISTOGRAM NUM_BUCKETS
--------------- ---------- ----------- ----------- --------- -----------
OWNER                72462          29         .04 NONE                1
OBJECT_NAME          72462       44236       61.05 NONE                1
SUBOBJECT_NAME       72462         106         .15 NONE                1
OBJECT_ID            72462       72462         100 NONE                1
DATA_OBJECT_ID       72462        7608        10.5 NONE                1
OBJECT_TYPE          72462          44         .06 NONE                1
CREATED              72462        1366        1.89 NONE                1
LAST_DDL_TIME        72462        1412        1.95 NONE                1
TIMESTAMP            72462        1480        2.04 NONE                1
STATUS               72462           1           0 NONE                1
TEMPORARY            72462           2           0 NONE                1
GENERATED            72462           2           0 NONE                1
SECONDARY            72462           2           0 NONE                1
NAMESPACE            72462          21         .03 NONE                1
EDITION_NAME         72462           0           0 NONE                0

15 rows selected.
```

owner 欄基數很低，現在我們對 owner 欄進行查詢。

```
SQL> set autot trace
SQL> select * from test where owner='SCOTT';

7 rows selected.

Execution Plan
----------------------------------------------------------
Plan hash value: 1357081020

---------------------------------------------------------------------------
| Id  | Operation         | Name | Rows  | Bytes | Cost (%CPU)| Time     |
---------------------------------------------------------------------------
|  0  | SELECT STATEMENT  |      |  2499 |   236K|   289   (1)| 00:00:04 |
|* 1  |  TABLE ACCESS FULL| TEST |  2499 |   236K|   289   (1)| 00:00:04 |
---------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter("OWNER"='SCOTT')
```

請注意看粗體字部分，查詢 owner＝'SCOTT'返回了 7 條資料，但是 CBO 在計算 **Rows** 的時候認為 owner='SCOTT'返回 **2499** 條資料，Rows 估算得不是特別準確。從 72462 條資料裡面查詢出 7 條資料，應該走索引，所以現在我們對 owner 欄新建索引。

碁峰 www.gotop.com.tw

```
SQL> create index idx_owner on test(owner);

Index created.
```

我們再來查詢一下。

```
SQL> select * from test where owner='SCOTT';

7 rows selected.

Execution Plan
----------------------------------------------------------
Plan hash value: 3932013684

-----------------------------------------------------------------------------------------
| Id |Operation                    |Name      | Rows  | Bytes | Cost(%CPU)| Time     |
-----------------------------------------------------------------------------------------
|  0 | SELECT STATEMENT            |          |  2499 |  236K |   73   (0)| 00:00:01 |
|  1 | TABLE ACCESS BY INDEX ROWID |TEST      |  2499 |  236K |   73   (0)| 00:00:01 |
|* 2 | INDEX RANGE SCAN            |IDX_OWNER|  2499 |       |    6   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("OWNER"='SCOTT')
```

現在我們查詢 owner='SYS'。

```
SQL> select * from test where owner='SYS';

30808 rows selected.

Execution Plan
----------------------------------------------------------
Plan hash value: 3932013684

-----------------------------------------------------------------------------------------
| Id |Operation                    | Name      | Rows  | Bytes | Cost(%CPU)| Time     |
-----------------------------------------------------------------------------------------
|  0 | SELECT STATEMENT            |           |  2499 |   236K|   73   (0)| 00:00:01 |
|  1 | TABLE ACCESS BY INDEX ROWID| TEST      |  2499 |   236K|   73   (0)| 00:00:01 |
|* 2 | INDEX RANGE SCAN            | IDX_OWNER|  2499 |       |    6   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("OWNER"='SYS')
```

注意粗字體部分，查詢 owner='SYS'返回了 30808 條資料。從 72462 條資料裡面返回 30808 條資料能走索引嗎？很明顯應該走全資料表掃描。也就是說該執行計畫是錯誤的。

碁峰 www.gotop.com.tw

為什麼查詢 owner='SYS'的執行計畫會用錯呢？因為 owner 這個欄基數很低，只有 29，而表的總列數是 72462。**前文著重強調過，當欄位沒有收集直方圖統計資訊的時候，**CBO **會認為該欄資料分佈是均衡的**。正是因為 CBO 認為 owner 欄資料分佈是均衡的，不管 owner 等於任何值，CBO 估算的 Rows 永遠都是 **2499**。而這 **2499** 是怎麼來的呢？答案如下。

```
SQL> select round(72462/29) from dual;

round(72462/29)
---------------
           2499
```

現在大家也知道了，**執行計畫裡面的 Rows 是假的**。執行計畫中的 Rows 是根據統計資訊以及一些數學公式計算出來的。很多 DBA 到現在還不知道執行計畫中 Rows 是假的這個真相，真是令人遺憾。

在做 SQL 最佳化的時候，經常需要做的工作就是幫助 CBO 計算出比較準確的 Rows。注意：我們說的是比較準確的 Rows。CBO 是無法得到精確的 Rows 的，因為對表收集統計資訊的時候，統計資訊一般都不會按照 100%的標準採樣收集，即使按照 100%的標準採樣收集了表的統計資訊，表中的資料也隨時在發生變更。另外計算 Rows 的數學公式目前也是有缺陷的，CBO 永遠不可能計算得到精確的 Rows。

如果 CBO 每次都能計算得到精確的 Rows，那麼相信我們這個時候只需要關心業務邏輯、表設計、SQL 寫法以及如何建立索引，**再也不用擔心 SQL 會走錯執行計畫了**。

Oracle12c 的新功能 SQL Plan Directives 在一定程度上解決了 Rows 估算不準而引發的 SQL 效能問題。關於 SQL Plan Directives，本書不做過多討論。

為了讓 CBO 選擇正確的執行計畫，我們需要對 owner 欄收集直方圖資訊，從而告知 CBO 該欄資料分佈不均衡，讓 CBO 在計算 Rows 的時候參考直方圖統計。現在我們對 owner 欄收集直方圖。

```
SQL> BEGIN
  2   DBMS_STATS.GATHER_TABLE_STATS(ownname         => 'SCOTT',
  3                                 tabname         => 'TEST',
  4                                 estimate_percent => 100,
  5                                 method_opt      => 'for columns owner size skewonly',
  6                                 no_invalidate   => FALSE,
  7                                 degree          => 1,
```


碁峰 www.gotop.com.tw

```
  8                                cascade          => TRUE);
  9  END;
 10  /

PL/SQL procedure successfully completed.
```

查看一下 owner 欄的直方圖資訊。

```
SQL> select a.column_name,
  2         b.num_rows,
  3         a.num_distinct Cardinality,
  4         round(a.num_distinct / b.num_rows * 100, 2) selectivity,
  5         a.histogram,
  6         a.num_buckets
  7    from dba_tab_col_statistics a, dba_tables b
  8   where a.owner = b.owner
  9     and a.table_name = b.table_name
 10     and a.owner = 'SCOTT'
 11     and a.table_name = 'TEST';

COLUMN_NAME       NUM_ROWS CARDINALITY SELECTIVITY HISTOGRAM  NUM_BUCKETS
--------------- ---------- ----------- ----------- ---------- -----------
OWNER                72462          29         .04 FREQUENCY           29
OBJECT_NAME          72462       44236       61.05 NONE                 1
SUBOBJECT_NAME       72462         106         .15 NONE                 1
OBJECT_ID            72462       72462         100 NONE                 1
DATA_OBJECT_ID       72462        7608        10.5 NONE                 1
OBJECT_TYPE          72462          44         .06 NONE                 1
CREATED              72462        1366        1.89 NONE                 1
LAST_DDL_TIME        72462        1412        1.95 NONE                 1
TIMESTAMP            72462        1480        2.04 NONE                 1
STATUS               72462           1           0 NONE                 1
TEMPORARY            72462           2           0 NONE                 1
GENERATED            72462           2           0 NONE                 1
SECONDARY            72462           2           0 NONE                 1
NAMESPACE            72462          21         .03 NONE                 1
EDITION_NAME         72462           0           0 NONE                 0

15 rows selected.
```

現在我們再來查詢上面的 SQL，看執行計畫是否還會走錯並且驗證 Rows 是否還會算錯。

```
SQL> select * from test where owner='SCOTT';

7 rows selected.

Execution Plan
----------------------------------------------------------
Plan hash value: 3932013684

-------------------------------------------------------------------------------
| Id  |Operation                   | Name     | Rows | Bytes | Cost (%CPU)| Time     |
```


碁峰 www.gotop.com.tw

```
-----------------------------------------------------------------------------------
|  0 | SELECT STATEMENT           |          |    7 |   679 |     2   (0)| 00:00:01 |
|  1 | TABLE ACCESS BY INDEX ROWID| TEST     |    7 |   679 |     2   (0)| 00:00:01 |
|* 2 | INDEX RANGE SCAN           | IDX_OWNER|    7 |       |     1   (0)| 00:00:01 |
-----------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("OWNER"='SCOTT')

SQL> select * from test where owner='SYS';

30808 rows selected.

Execution Plan
----------------------------------------------------------
Plan hash value: 1357081020

---------------------------------------------------------------------------
| Id  | Operation         | Name | Rows  | Bytes | Cost (%CPU)| Time     |
---------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |      | 30808 |  2918K|   290   (1)| 00:00:04 |
|*  1 |  TABLE ACCESS FULL| TEST | 30808 |  2918K|   290   (1)| 00:00:04 |
---------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter("OWNER"='SYS')
```

對 owner 欄收集完直方圖之後，CBO 估算的 Rows 就基本準確了，一旦 Rows 估算對了，那麼執行計畫也就不會出錯了。

大家是不是很好奇，為什麼收集完直方圖之後，Rows 計算得那麼精確，收集直方圖究竟完成了什麼操作呢？對 owner 欄收集直方圖其實就相當於執行了以下 SQL。

```
select owner,count(*) from test group by owner;
```

直方圖資訊就是以上 SQL 的查詢結果，這些查詢結果會保存在資料字典中。這樣當我們查詢 owner 為任意值的時候，CBO 總會算出正確的 Rows，因為直方圖已經知道每個值有多少列資料。

如果 SQL 使用了綁定變數，綁定變數的欄位收集了直方圖，那麼該 SQL 就會引起綁定變數窺探。綁定變數窺探是一個老生常談的問題，這裡不多做討論。Oracle11g 引入了自我調整游標共用（Adaptive Cursor Sharing）功能，基本上解

碁峰 www.gotop.com.tw

決了綁定變數窺探問題，但是自我調整游標共用也會引起一些新問題，對此也不做過多討論。

當我們遇到一個 SQL 有綁定變數怎麼辦？其實很簡單，只需要執行以下語法。

```
select 欄, count(*) from test group by 欄 order by 2 desc;
```

如果列資料分佈均衡，基本上 SQL 不會出現問題；如果欄位資料分佈不均衡，我們需要對欄位收集直方圖統計。

關於直方圖，其實還有非常多的話題，比如直方圖的種類、直方圖的桶數等，本書在此不做過多討論。在我們看來，**讀者只需知道直方圖是用來幫助 CBO 在對基數很低、資料分佈不均衡的欄位進行 Rows 估算的時候，可以得到更精確的 Rows 就夠了**。

什麼樣的欄位需要收集直方圖呢？當欄位出現在 where 條件中，欄位的選擇性小於 1%並且該欄沒有收集過直方圖，這樣的欄位就應該收集直方圖。注意：千萬不能對沒有出現在 where 條件中的欄位收集直方圖。對沒有出現在 where 條件中的欄位收集直方圖完全是做白功，浪費資料庫的資源。

下面我們為大家分享本書第二個全自動化最佳化腳本。

抓出必須新建直方圖的欄位（大家可以對該腳本進行適當修改，以便用於生產環境）。

```
SQL> select a.owner,
  2         a.table_name,
  3         a.column_name,
  4         b.num_rows,
  5         a.num_distinct,
  6         trunc(num_distinct / num_rows * 100,2) selectivity,
  7         'Need Gather Histogram' notice
  8    from dba_tab_col_statistics a, dba_tables b
  9   where a.owner = 'SCOTT'
 10     and a.table_name = 'TEST'
 11     and a.owner = b.owner
 12     and a.table_name = b.table_name
 13     and num_distinct / num_rows<0.01
 14      and (a.owner, a.table_name, a.column_name) in
 15         (select r.name owner, o.name table_name, c.name column_name
 16            from sys.col_usage$ u, sys.obj$ o, sys.col$ c, sys.user$ r
 17           where o.obj# = u.obj#
 18             and c.obj# = u.obj#
 19             and c.col# = u.intcol#
 20             and r.name = 'SCOTT'
```


碁峰 www.gotop.com.tw