StatsPack スナップショットとは

StatsPack スナップショットは、stats$sql_summary に代表されるいくつかの stats$で始まるテーブルに実行計画と実行統計について記録した情報の集まりである

stats$snapshot： StatsPack スナップショット取得日時
stats$sql_summary： 実行 SQL 文の個別実行統計記録
stats$sqltext： 実行 SQL 文のテキスト
stats$sql_plan_usage : PLAN_HASH_VALUE を求めるための変換テーブル
stats$sql_plan： 実行 SQL 文の実行計画

取得されているデータについての注意点）

・CPU 使用時間が短い or ディスク I/O が小さいなどの負荷が小さい SQL 文に対しては、stats$sql_summary の中に記録されない

・記録されているデータは累積値として保存されているので、使用する場合は終了時点のスナップ ID の統計値から開始時点のスナップ ID の統計値を引き算して使用することになる

・記録されているデータは、データベースの稼働が連続の場合のみ有効な値として意味を持つ
２つのスナップショットの間で、**シャットダウンが行われている場合は、データが意味を持たない**

・シャットダウンが行われた場合は、各項目のデータは、メモリの余力状態でクリアされたり、されなかったりしているので、データベース起動後の１回目のスナップショットの単独での値は、意味を持っていない
~~また、SQL 文に対する SID や Plan_hash_value が変更されてしまう~~

StatsPack レポートとは、下記の stats$sql_summary を中心とする stats$の名前で始まるテーブルの情報を見やすくまとめた出力結果のサマリー帳票である

stats$sql_summary のデータは累積値として記録されているので、プログラミングして Select した結果を加工しても、得られる内容はレポートの内容とほぼ同等程度にしかならないと考えられる

StatsPack 詳細レポートについても同様で、stats$の名前で始まるテーブルの情報から、特定の１つの SQL ステートメントについて「実行計画」と「実行統計」の情報を詳細に出力した帳票である

StatsPack のスナップショットにおいて、実行計画を記録しているテーブルは、stats$sql_plan である。

StatsPack のスナップショットにおいて記録された **SQL 文が稼働していた期間（時刻）**は、syswait スクリプトのログから調査すれば、判明する

Statspack 詳細レポートを使った「SQL 実行計画」と「実行統計情報」の入手

OLD_HASH_VALUE 値が判明していれば、それに対応する SQL 文の Statspack 詳細レポートを作成すれば、「SQL 実行計画」と「実行統計情報」が詳細に判明する

手順１.

OLD_HASH_VALUE 値を入手する

注意） CPU 使用時間が短い or ディスク I/O が小さいなどの負荷が小さい SQL 文に対しては、stats$sql_summary の中に記録されない

【Statspack レポートからの場合】

CPU 実行 or Disk I/O 時間 or 物理ブロック読込み数において、負荷の高かった SQL 文は、レポートの中でそのリソース使用状況の概要が出力されている。

その中にこの SQL 文にたいする OLD_HASH_VALUE 値も記述されている

【SQL テキスト文からの場合】

注意） 実行した SQL 文は、stats$sqltext テーブルでは 64 文字ずつにレコード分割され、PIECE 列の連番によって管理されている
よって、検索する文字列が２つのレコードに分割されている場合があるので、分割されていない範囲の文字列を Where 条件に指定すること

```
select    SNAP_ID ,    SST.OLD_HASH_VALUE ,    SST.PIECE,
          SSU.HASH_VALUE ,  SSU.SQL_ID ,   SST.SQL_TEXT
  from  stats$sql_summary  SSU ,  stats$sqltext  SST
  Where  SSU.OLD_HASH_VALUE  =  SST.OLD_HASH_VALUE
    and  SST.OLD_HASH_VALUE  in
  (  select  OLD_HASH_VALUE
       from  stats$sqltext  SST2
      Where  SST2.SQL_TEXT  like  '%＜sql 文＞%'
  )
  Order by  SNAP_ID ,  SST.OLD_HASH_VALUE ,  SST.PIECE ;
```

【SQL_ID 値からの場合】

```
select  SNAP_ID , OLD_HASH_VALUE , HASH_VALUE ,  SQL_ID ,
        TEXT_SUBSET
  from  stats$sql_summary
  WHERE   SQL_ID  =  '＜SQL_ID 値＞'  ;
```

【HASH_VALUE 値からの場合】

```
select   SNAP_ID , OLD_HASH_VALUE , HASH_VALUE , TEXT_SUBSET
         from   stats$sql_summary
         WHERE    HASH_VALUE   = ＜sql_hash_value 値＞ ;
```

手順２.

Statspack の詳細レポートの出力方法

```
@?/rdbms/admin/sprepsql.sql
```

← Statspack 詳細レポート作成用 SQL スクリプトの実行

Specify the Begin and End Snapshot Ids

begin_snap に値を入力してください： 52

end_snap に値を入力してください： 54

hash_value に値を入力してください： ＜old_hash_value の値＞

→ Statspack 詳細レポートでの条件入力は、**old_hash_value** 値を指定する

report_name に値を入力してください： ＜作成レポート名＞

→ 作成する詳細レポートのファイル名を入力する

これにより、old_hash_value の値が対応する SQL 文の Statspack 詳細レポートが作成される

Statspack 詳細レポート（特定 SQL 文に対する詳細情報）の内容

対象 SQL　：実行時に指定した OLD_HASH_VALUE SQL 値に対する SQL 文が対象

リスト名　：実行時に指定したファイル名が、Statspack 詳細レポートの名前になる

実　行　計　画

実 行 統 計 情 報

stats$snapshot テーブルの構成

StatsPack スナップショットを採取した取得日時

列　　名	データ型	説　　明
SNAP_ID	NUMBER	スナップショット番号
SNAP_TIME	DATE	スナップショットの取得日時
STARTUP_TIME	DATE	スナップショット取得時の Oracle インスタンスの OPEN 開始日時
SNAP_LEVEL	NUMBER	スナップショットのレベル

stats$sql_summary テーブルの構成

StatsPack スナップショットでは、stats$sql_summary テーブルに、実行 SQL 文の個別の実行統計の情報が記録される

注意）　CPU 使用時間が短い or ディスク I/O が小さいなどの負荷が小さい SQL 文に対しては、stats$sqltext の中に記録されない

列　　名	データ型	説　　明
SNAP_ID	NUMBER	スナップショット・番号
DBID	NUMBER	データベース・ID
INSTANCE_NUMBER	NUMBER	Oracle インスタンスに割り当てられた番号
TEXT_SUBSET	VARCHAR2(31)	SQL 文の最初の 31 文字
OLD_HASH_VALUE	NUMBER	SQL 文を識別する オールド・ハッシュ・バリュー値
SQL_TEXT	VARCHAR2 (100)	この列には、SQL 文の**値はセットされない** SQL 文が必要な場合には、 **stats$sqltext** シノニムから SQL 文を抜き出して使用する必要がある ※ 64 文字分割されているので**要注意**
SQL_ID	VARCHAR2(13)	SQL 文を識別する SQL_ID 値
SHARABLE_MEM	NUMBER	共有メモリ使用量
SORTS	NUMBER	ソート実行の確保量 メモリ領域＋ディスク領域
MODULE	VARCHAR2(64)	
LOADED_VERSIONS	NUMBER	

FETCHES	NUMBER	フェッチ回数
EXECUTIONS	NUMBER	実行回数
PX_SERVERS_EXECUTIONS	NUMBER	
END_OF_FETCH_COUNT	NUMBER	
LOADS	NUMBER	ロード回数
INVALIDATIONS	NUMBER	
PARSE_CALLS	NUMBER	解析が行われた回数
DISK_READS	NUMBER	ディスクアクセス量
DIRECT_WRITES	NUMBER	バッファを使わずに、直接ディスクへ書き込んだデータ量
BUFFER_GETS	NUMBER	バッファからのデータ読み込み量
APPLICATION_WAIT_TIME	NUMBER	
CONCURRENCY_WAIT_TIME	NUMBER	
CLUSTER_WAIT_TIME	NUMBER	
USER_IO_WAIT_TIME	NUMBER	ユーザーI/O での待機時間
PLSQL_EXEC_TIME	NUMBER	PL/SQL 全体での実行時間
JAVA_EXEC_TIME	NUMBER	
ROWS_PROCESSED	NUMBER	
COMMAND_TYPE	NUMBER	実行されたコマンドの種類を示す値
ADDRESS	RAW(8)	SQL 文を識別するアドレス値
HASH_VALUE	NUMBER	SQL 文を識別するハッシュ・バリュー値
VERSION_COUNT	NUMBER	
CPU_TIME	NUMBER	CPU 実行時間
ELAPSED_TIME	NUMBER	CPU 実行時間 ＋ 待機時間
AVG_HARD_PARSE_TIME	NUMBER	
OUTLINE_SID	NUMBER	
OUTLINE_CATEGORY	VARCHAR2(64)	
CHILD_LATCH	NUMBER	
SQL_PROFILE	VARCHAR2(64)	
PROGRAM_ID	NUMBER	
PROGRAM_LINE#	NUMBER	

EXACT_MATCHING_SIGNATURE	NUMBER	
FORCE_MATCHING_SIGNATURE	NUMBER	
LAST_ACTIVE_TIME	DATE	

stats$sql|text テーブルの構成

StatsPack スナップショットにおいて、実行した SQL 文のテキストが記録されている

列　　名	データ型	説　　明
OLD_HASH_VALUE	NUMBER	SQL 文を識別する オールド・ハッシュ・バリュー値
TEXT_SUBSET	VARCHAR2(31)	SQL コマンドの最初の 31 文字文字 連番が異なっても、同一
PIECE	NUMBER	SQL_TEXT が 64 文字ずつ分割されてレコード化されているので、その連番 連番 0,1,2,3・・・
SQL_ID	VARCHAR2(13)	SQL 文を識別する SQL_ID 値
SQL_TEXT	VARCHAR2(64)	実行した SQL 文 ただし、64 バイトずつにレコード分割され、構成メンバーは、PIECE 列の連番によって管理される PL/SQL プロシージャを実行した場合には、プロシージャの全実行コードが入る
ADDRESS	RAW(8)	SQL 文を識別するアドレス値
COMMAND_TYPE	NUMBER	実行されたコマンドの種類を示す値
LAST_SNAP_ID	NUMBER	

stats$sql_plan_usage テーブルの構成

PLAN_HASH_VALUE を求めるための変換テーブル

列　　名	データ型	説　　明
SNAP_ID		スナップショット番号
PLAN_HASH_VALUE		実行計画プランハッシュ値
SQL_ID		SQL_ID 値
HASH_VALUE		SQL に対するハッシュ・バリュー値
OLD_HASH_VALUE		旧ハッシュ・バリュー値

stats$sql_plan テーブルの構成

StatsPack スナップショットにおいて、実行計画を記録しているテーブル

列　　名	データ型	説　　明
PLAN_HASH_VALUE		実行計画プランハッシュ値
SNAP_ID		スナップショット番号
ID		実行計画表の表示段
DEPTH		操作を表するときの左空白文字の数
OPERATION		操作
OPTIONS		
OBJECT_NODE		
OBJECT#		
OBJECT_OWNER		
OBJECT_NAME		使用オブジェクトの名前
OBJECT_ALIAS		
OBJECT_TYPE		
OPTIMIZER		
PARENT_ID		
COST		
BYTES		
CPU_COST		
IO_COST		
ACCESS_PREDICATES		
FILTER_PREDICATES		
TIME		

実行例）

```
---  SQL 文から OLD_HASH__VALUE の求め方  ---
select  sql_text ,  old_hash_value  from  stats$sqltext
          where  sql_text  like  '%＜検索 SQL 文字列＞%';


---  PLAN_HASH_VALUE の求め方  ---
select  snap_id ,  plan_hash_value ,  old_hash_value ,  hash_value  , sql_id ,
          text_subset
     from  stats$sql_plan_usage
     where  old_hash_value  =  ＜OLD_HASH__VALUE 値＞ ;


---  SQL 文から PLAN_HASH__VALUE の求め方  ---
select  SP.snap_id ,  SP.plan_hash_value ,  SP.old_hash_value , SP.hash_value  ,
          SP.sql_id ,  SP.text_subset
     from  stats$sql_plan_usage  SP ,  stats$sqltext  ST
       where  ST.sql_text  like  '%＜検索 SQL 文字列＞%'
          and  ST.old_hash_value  =  SP.old_hash_value ;


---  記録されているスナップショットが、採取された日時の求め方  ---
select  snap_id ,
          to_char( snap_time ,  'YYYY-MM-DD HH24:MI:SS' )  snap_time ,
          to_char( startup_time ,  'YYYY-MM-DD HH24:MI:SS' )  startup_time
     from  stats$snapshot ;
```

--- SQL 文から stats&sql_summary の情報取得 ---

※ ただし、記録されているデータは累積値として保存されているので、使用する場合は終了時点のスナップ ID の統計値から開始時点のスナップ ID の統計値を引き算して使用することになる

２つのスナップショットの間でシャットダウンが行われている場合は、この２つのスナップショットに対する stats&sql_summary データは意味を持たない

終了時点のスナップショットだけに SQL_ID がある（開始時点のスナップショットには SQL_ID レコードが無い）レコードは、新規に発生した SQL 文であり、実行回数等のデータの差分値は、０からと考える

開始以前のスナップショットのデータに対象の SQL_ID があっても、一度メモリがクリアになった時点で、カウント０からのリセットと考える

（実行されなかった SQL に対しても、ライブラリキャッシュにあれば stats&sql_summary データは、保存されている
この根拠は、EXECUTIONS（実行回数）が開始時点 SNAP_ID と終了時点 SNAP_ID で、同じ値の SQL_ID レコードの存在から分かる）

```
VARIABLE  start_snapid  NUMBER  ;
VARIABLE  end_snapid  NUMBER  ;
EXECUTE  :start_snapid  :=  &start_snapid
```

ここで、入力操作を一旦区切ること

```
EXECUTE  :end_snapid  :=  &end_snapid
```

ここで、入力操作を一旦区切ること

```
PRINT  'start_snapid :  '  ||  start_snapid  ||  '  end_snapid :  '  ||  end_snapid
SELECT  *  FROM  (
select  SU2.snap_id  "End_snap"  , SU1.snap_id  "Start_snap" ,  SU1.sql_id ,
        SU2.executions  -  SU1.executions  "Executions" ,
        SU2.elapsed_time  -  SU1.elapsed_time  "Elapsed_time" ,
        SU2.cpu_time  -  SU1.cpu_time  "Cpu_time" ,
        SU2.elapsed_time  -  SU2.cpu_time  -  SU1.elapsed_time  +  SU1.cpu_time
                "Wait_time" ,
        SU2.buffer_gets  -  SU1.buffer_gets  "Buffer_gets" ,
        SU2.disk_reads  -  SU1.disk_reads  "Disk_reads" ,
        SU2.direct_writes  -  SU1.direct_writes  "Direct_writes" ,
        SU2.user_io_wait_time  -  SU1.user_io_wait_time  "User_io_wait_time" ,
        SU2.application_wait_time  -  SU1.application_wait_time
                "Application_wait_time" ,
        SU2.concurrency_wait_time  -  SU1.concurrency_wait_time
                "Concurrency_wait_time"
```

```
    from   stats$sql_summary   SU1 ,   stats$sql_summary   SU2
      where   SU1.snap_id   =   :START_SNAPID
        and   SU2.snap_id   =   :END_SNAPID
        and   SU2.sql_id   =   SU1.sql_id
UNION
select   SU3.snap_id   "End_snap"   ,   -1   "Start_snap" ,   SU3.sql_id ,
        SU3.executions   "Executions" ,
        SU3.elapsed_time   "Elapsed_time" ,
        SU3.cpu_time   "Cpu_time" ,
        SU3.elapsed_time   -   SU3.cpu_time   "Wait_time" ,
        SU3.buffer_gets   "Buffer_gets" ,
        SU3.disk_reads   "Disk_reads" ,
        SU3.direct_writes   "Direct_writes" ,
        SU3.user_io_wait_time   "User_io_wait_time" ,
        SU3.application_wait_time   "Application_wait_time" ,
        SU3.concurrency_wait_time   "Concurrency_wait_time"
    from   stats$sql_summary   SU3
        where   SU3.snap_id   =   :END_SNAPID
           and   SU3.sql_id   IN   (
        select   SU4.sql_id
            from   stats$sql_summary   SU4
              where   SU4.snap_id   =   :END_SNAPID
        MINUS
        select   SU5.sql_id
            from   stats$sql_summary   SU5
              where   SU5.snap_id   =   :START_SNAPID )
      order by   "Cpu_time"   Desc
    )   WHERE   rownum   <=   20 ;
```

SQL 文を条件に加える場合の Where 付加条件

```
      and   ST.sql_text   like   '%＜検索条件文字列＞%'
```

--- PLAN_HASH__VALUE から SQL 実行計画の求め方 ---

```
set heading off ver off

select '--------------------------------------------------------------------------------' from dual
union all
select '| Operation                                | PHV/Object Name        |    Rows | Bytes|
Cost |'   as "Optimizer Plan:" from dual
union all
select '--------------------------------------------------------------------------------' from dual
union all
select * from (
        select
        rpad('|'||substr(lpad(' ',1*(depth-1))||operation||
                decode(options, null,'','('||options||')'), 1, 32), 33, ' ')||'|'||
        rpad(decode(id, 0, '----- '||to_char(plan_hash_value)||' -----'
                        , substr(decode(substr(object_name, 1, 7), 'SYS_LE_', null,
object_name)
                                ||' ',1, 20)), 21, ' ')||'|'||
        lpad(decode(cardinality,null,'    ',
                decode(sign(cardinality-1000), -1, cardinality||' ',
                decode(sign(cardinality-1000000), -1, trunc(cardinality/1000)||'K',
                decode(sign(cardinality-1000000000), -1,
trunc(cardinality/1000000)||'M',
                        trunc(cardinality/1000000000)||'G')))), 7, ' ') || '|' ||
        lpad(decode(bytes,null,' ',
                decode(sign(bytes-1024), -1, bytes||' ',
                decode(sign(bytes-1048576), -1, trunc(bytes/1024)||'K',
                decode(sign(bytes-1073741824), -1, trunc(bytes/1048576)||'M',
                        trunc(bytes/1073741824)||'G')))), 6, ' ') || '|' ||
        lpad(decode(cost,null,' ',    decode(sign(cost-10000000), -1, cost||' ',
                decode(sign(cost-1000000000), -1, trunc(cost/1000000)||'M',
                        trunc(cost/1000000000)||'G'))), 8, ' ') || '|' as "Explain plan"
          from   stats$sql_plan
         where plan_hash_value in (3981161016 )
          order by   plan_hash_value,   id
)
union all
select '--------------------------------------------------------------------------------' from dual   ;
```

stats$sqltext テーブルの SQL 文の検索

stats$sqltext の SQL_TEXT には、PIECE 列の連番でレコード分割された 64 バイト分の SQL 文が分割して入れられている

この分割された SQL_TEXT 列を中身で検索する場合、検索対象文字列が PIECE 列の異なった**複数のレコードまたがって**いることを想定して、検索処理を行わなければならない

下記に示した SQL 文では、検索対象の文字列を PIECE 列の連番 0,1,2,3 までを対象にして検索を行わせている

実使用時には、連番対応を調整して実行すること

```
VARIABLE  Search_SQL_TEXT  VARCHAR2(2000);
EXECUTE  :Search_SQL_TEXT  :=  '＜検索対象文字列＞'
--  検索例）
EXECUTE  :Search_SQL_TEXT  :=  'B1 WHERE MARKER_TIMESTAMP'

EXECUTE  :Search_SQL_TEXT  :=  '%' || :Search_SQL_TEXT || '%'

select  S0.old_hash_value ,  S0.piece  ||  S0.sql_text  ||  S1.sql_text  ||
        S2.sql_text  ||  S3.sql_text
    from  stats$sqltext  S0 ,  stats$sqltext  S1 ,  stats$sqltext  S2 ,
          stats$sqltext  S3
   where  S0.piece  =  0  and  S1.piece(+)  = 1  and  S2.piece(+) =  2
     and  S3.piece(+) =  3
     and  S0.old_hash_value  =  S1.old_hash_value (+)
     and  S0.old_hash_value  =  S2.old_hash_value (+)
     and  S0.old_hash_value  =  S3.old_hash_value (+)
     and  S0.sql_text  ||  S1.sql_text  ||  S2.sql_text  ||  S3.sql_text
     like  :Search_SQL_TEXT  ;
```

完全 SQL 文の出力 PL/SQL ソース

```
SET  SERVEROUTPUT  ON;

DECLARE
  total_sqltext  VARCHAR2(2000) ;
  search_old_hash_value  CONSTANT  number  :=  3665763022 ;
  CURSOR  CA  IS
      SELECT  *  from  stats$sqltext
        WHERE  old_hash_value  =  search_old_hash_value
        ORDER  BY  piece  ;
  -- データ格納用変数の定義
  stats_rec  CA%ROWTYPE  ;

BEGIN
  total_sqltext  :=  '' ;
  OPEN  CA  ;

  -- 結果の抽出（１レコードの取出し）
  FETCH  CA  INTO  STATS_REC  ;

  -- 終了判定
  LOOP
    IF  CA%NOTFOUND  THEN
      EXIT  ;
    END  IF  ;
    /*  SELECT レコードに対する処理  */
    total_sqltext  :=  total_sqltext  ||  stats_rec.sql_text  ;
    FETCH  CA  INTO  stats_rec  ;
  END  LOOP  ;
      DBMS_OUTPUT.PUT_LINE(  'stats_ old_hash_value  :  ' || stats_rec.
            old_hash_value ) ;
      DBMS_OUTPUT.PUT_LINE(  'total_sqltext  :  ' || total_sqltext ) ;

  CLOSE  CA ;

END  ;
/
```