TABLE ACCESS 패턴을 이용한 SQL 튜닝

㈜엑셈 컨설팅본부/DB컨설팅팀 오 경렬

본 문서는 필자가 사용하는 TABLE ACCESS 패턴을 확인하는 스크립트에 대한 내용입니다.
필자가 SQL 튜닝을 진행하면서 애용하는 스크립트입니다.

본 스크립트를 사용하는 경우는 크게 2 가지가 있습니다.

1. 인덱스 생성 가이드를 하는 경우
2. 테이블 단위로 튜닝을 진행하는 경우

1. 인덱스 생성 가이드를 하는경우

SQL 튜닝을 하다 보면 인덱스 생성 가이드 해야 하는 경우가 있습니다. UNIQUE 한 성격의 칼럼 이라면 별다른 고민 없이 생성 권고가 가능하겠지만 복잡한 업무 패턴을 가지고 있거나 다른 쿼리 플랜에 영향을 줄 수 있는 경우는 상당히 신중하게 작업을 진행해야 합니다.
인덱스를 생성하려는 테이블을 참조하는 모든 쿼리를 확인하면 좋겠지만 우리에게 주어진 시간은 한정적입니다. 컨설팅의 질 만큼 양도 중요하기 때문에 짧은 시간에 최대한 많은 정보를 확인할 필요가 있습니다. 필자는 엑셀 메크로를 활용해 가독성을 높입니다. 스크립트는 가독성을 높이면서 필요한 정보를 최대한 많이 보여 주도록 작성했습니다.

▶ 스크립트

스크립트는 다음과 같은 기능을 갖습니다.

1. Access Predicate / Filter Predicate 확인
2. HASH JOIN 에 사용된 칼럼 표시
3. 일량/수행횟수/모듈명 등 쿼리 수행 이력 확인
4. 리터럴 쿼리를 하나로 합치고 리터럴 쿼리의 갯수를 확인
5. 리터럴 쿼리의 경우 가장 많은 비효율을 차지하는 대표 쿼리의 SQL_ID 표시

▶ 엑셀 메크로 활용

수행 결과 -> HTML -> 엑셀

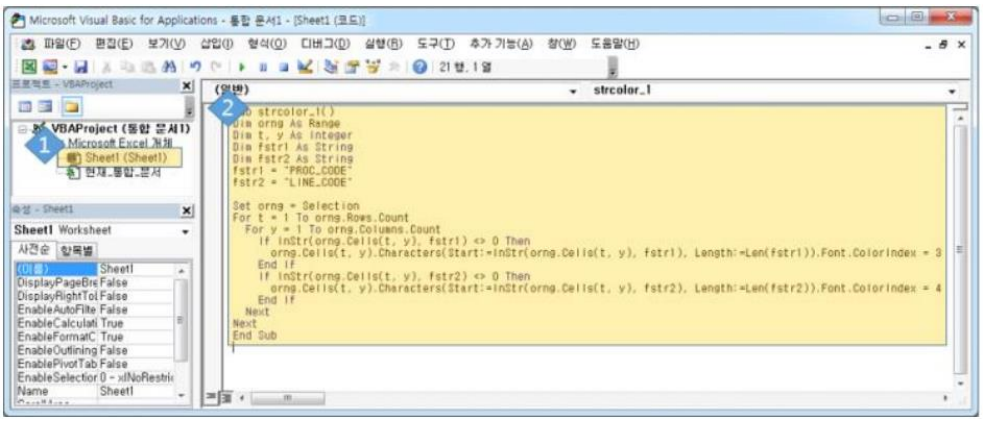
가독성을 위해 엑셀 메크로를 활용해 특정 문자에 색칠 하는데

이때 옮기는 HTML 을 이용해야 (LITE PLUS 에서는) SQL_TEXT 가 한 샐안에 들어갑니다.

엑셀 메크로는 반 자동화 하여 사용합니다.

▶ 엑셀 메크로 적용

Step by Step 으로 적겠습니다. 엑셀에 붙여넣기 한 후에 ALT+F11 Click



<1> 더블 클릭

<2> 더블 클릭 후 창이 뜨면 아래 소스를 붙여 넣습니다.

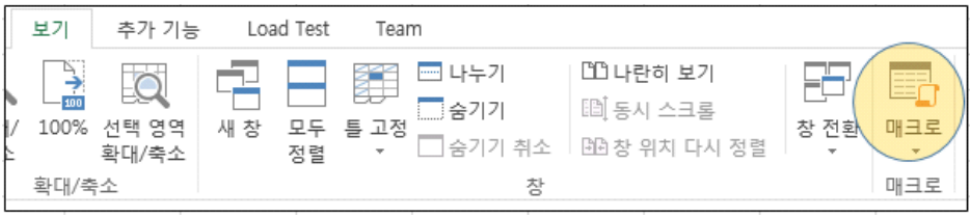
fstr -> 하이라이트 할 문자

Font.ColorIndex -> 하이라이트 색

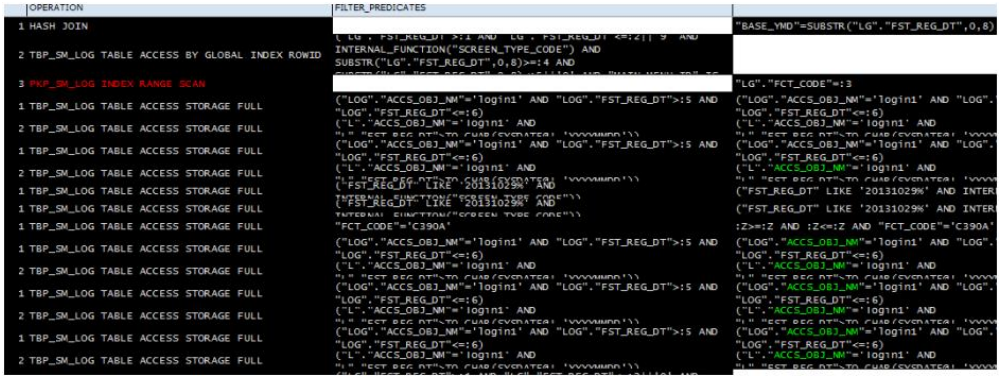
```
Sub strcolor_1()
Dim orng As Range
Dim t, y As Integer
Dim fstr1 As String
Dim fstr2 As String
```

```
fstr1 = "PROC_CODE"
fstr2 = "LINE_CODE"
Set orng = Selection
For t = 1 To orng.Rows.Count
  For y = 1 To orng.Columns.Count
    If InStr(orng.Cells(t, y), fstr1) <> 0 Then
      orng.Cells(t, y).Characters(Start:=InStr(orng.Cells(t, y), fstr1),
      Length:=Len(fstr1)).Font.ColorIndex = 3
    End If
    If InStr(orng.Cells(t, y), fstr2) <> 0 Then
      orng.Cells(t, y).Characters(Start:=InStr(orng.Cells(t, y), fstr2),
      Length:=Len(fstr2)).Font.ColorIndex = 4
    End If
  Next
Next
End Sub
```

<3> 셀 선택 후 매크로 적용



<4> 결과



2. 테이블 단위로 튜닝을 진행하는 경우

튜닝 포인트를 발췌하는 것도 컨설턴트가 갖춰야 할 중요한 역량입니다. 테이블 크기 / 테이블 Description 등을 확인 한 뒤 BIG 테이블 위주로 TABLE ACCESS 패턴을 확인하면서 비효율이 발생하는 쿼리를 확인하는 방법은 CASE BY CASE 지만 매우 강력한 무기로 작용할 수 있습니다.

▶ SCRIPT

```
WITH w_sql_list AS (SELECT DISTINCT sql_id ,
                           inst_id
                      FROM gv$sql_plan p
                     WHERE 1=1
                       AND object_name IN (SELECT UPPER( :tname )
                                             FROM dual
                                            UNION ALL
SELECT index_name
                                             FROM dba_indexes
                                            WHERE table_name = UPPER( :tname ) ) ) ,
       w_sql_face AS (SELECT /*+ leading(sp) use_hash(sa) */
```

```
   sa.inst_id ,
        sa.sql_id ,
        sa.version_count ,
        sa.executions ,
        sa.first_load_time ,
        sa.disk_reads ,
        sa.direct_writes ,
        sa.buffer_gets ,
        sa.rows_processed ,
        sa.optimizer_cost ,
        sa.parsing_schema_name ,
        sa.hash_value ,
        sa.plan_hash_value ,
        sa.module ,
        sa.cpu_time ,
        sa.elapsed_time ,
        sa.last_load_time ,
        RANK( ) over( PARTITION BY sa.inst_id , sa.plan_hash_value
          ORDER BY buffer_gets DESC ) rk
   FROM gv$sqlarea sa ,
        w_sql_list sp
  WHERE 1=1
    AND sa.sql_id = sp.sql_id
    AND sa.inst_id = sp.inst_id
    AND MODULE NOT LIKE 'TOAD%'
    AND MODULE NOT LIKE 'MAX%'
    AND MODULE NOT LIKE 'Toad%'
    AND MODULE NOT LIKE 'Lite%'
    AND MODULE NOT LIKE 'LINESTOP%'
    AND MODULE NOT LIKE 'Golden%'
    AND MODULE NOT LIKE 'Golden%'
    AND MODULE NOT LIKE 'Orange%'
    AND parsing_schema_name LIKE 'GMES20%' ) ,
w_sql_pt AS (SELECT /*+ use_hash(p x) */
   p.inst_id ,
        p.sql_id ,
        p.plan_hash_value ,
        p.object_name ||' ' ||p.operation ||' ' ||p.options AS operation ,
        p.object_type ,
        p.id ,
        p.parent_id ,
```

```
                p.depth ,
                p.access_predicates ,
                p.filter_predicates ,
                p.child_number
           FROM gv$sql_plan p ,
                (SELECT /*+ no_merge use_hash(p f) */
                   p.inst_id ,
                        p.plan_hash_value ,
                        p.operation ,
                        p.options ,
                        p.object_name ,
                        p.object_type ,
                        p.id ,
                        p.parent_id ,
                        p.depth ,
                        p.access_predicates ,
                        p.filter_predicates ,
                        p.sql_id
                    FROM gv$sql_plan p ,
                        w_sql_face f
                   WHERE 1=1
                     AND f.rk = 1
                     AND f.sql_id = p.sql_id
                     AND object_name IN (SELECT UPPER( :tname )
                            FROM dual
                           UNION ALL
SELECT index_name
                            FROM dba_indexes
                           WHERE table_name = UPPER( :tname ) ) ) x
           WHERE x.inst_id = p.inst_id
             AND x.parent_id = p.id
             AND p.sql_id = x.sql_id
             AND p.plan_hash_value = x.plan_hash_value
             AND p.operation = 'HASH JOIN'
           UNION ALL
SELECT /*+ use_hash(p f)*/
            p.inst_id ,
                p.sql_id ,
                p.plan_hash_value ,
                p.object_name ||' ' ||p.operation ||' ' ||p.options AS operation ,
                p.object_type ,
```

```
                p.id ,
                p.parent_id ,
                p.depth ,
                p.access_predicates ,
                p.filter_predicates ,
                p.child_number
           FROM gv$sql_plan p ,
                w_sql_face f
          WHERE 1=1
            AND f.rk = 1
            AND f.sql_id = p.sql_id
            AND object_name IN (SELECT :tname
                  FROM dual
                 UNION ALL
SELECT index_name
                  FROM dba_indexes
                 WHERE table_name = :tname ) )
SELECT decode( seq# , 1 , decode( inst_cnt , 2 , 3 , 1 , inst_id ) ) inst_id ,
       decode( seq# , 1 , MODULE ) MODULE ,
       decode( seq# , 1 , (SELECT KGLFNOBJ
                  FROM sys.XM$KGLCURSOR_CHILD x
                 WHERE x.kglobt03 = a.sql_id
                   AND ROWNUM <= 1 ) ) text ,
       decode( seq# , 1 , sql_id ) sql_id ,
       decode( seq# , 1 , liter_cnt ) liter_cnt ,
       decode( seq# , 1 , plan_hash_value ) plan_hash_value ,
       decode( seq# , 1 , parsing_schema_name ) parsing_schema_name ,
       decode( seq# , 1 , executions ) exec_t ,
       decode( seq# , 1 , face_exec ) fa_exec ,
       decode( seq# , 1 , buffer_gets ) bg ,
       decode( seq# , 1 , face_bgets ) fa_bg ,
       decode( seq# , 1 , ROUND(( buffer_gets ) /decode(( executions ) , 0 , 1 ,
( executions ) ) , 0 ) ) bg_1 ,
       decode( seq# , 1 , ROUND(( disk_reads ) /decode(( executions ) , 0 , 1 ,
( executions ) ) , 2 ) ) dr_1 ,
       decode( seq# , 1 , ROUND(( rows_processed ) /decode(( executions ) , 0 , 1 ,
( executions ) ) , 0 ) ) row_1 ,
       decode( seq# , 1 , ROUND((( elapsed_time ) /decode(( executions ) , 0 , 1 ,
( executions ) ) ) /1000000 , 3 ) ) elapst_1 ,
       decode( seq# , 1 , ROUND((( cpu_time ) /decode(( executions ) , 0 , 1 ,
( executions ) ) ) /1000000 , 3 ) ) cput_1 ,
```

```
      id ,
      parent_id ,
      seq# ,
      operation ,
      filter_predicates ,
      access_predicates
  FROM (SELECT a.* ,
               RANK( ) over (
                 ORDER BY buffer_gets DESC, sql_id, plan_hash_value) bg_seq#
          FROM (SELECT /*+ leading(p) use_hash(p f s) */
                  ROW_NUMBER( ) over( PARTITION BY x.sql_id, x.plan_hash_value
                         ORDER BY x.id ) seq# ,
                       x.*
                  FROM (SELECT p.id ,
                               p.parent_id ,
                               p.sql_id ,
                               p.plan_hash_value ,
                               SUM( s.liter_cnt ) liter_cnt ,
                               MAX( p.operation ) operation ,
                               MAX( p.filter_predicates ) filter_predicates ,
                               MAX( p.access_predicates ) access_predicates ,
                               MAX( f.module ) MODULE ,
                               MAX( f.parsing_schema_name ) parsing_schema_name ,
                               MAX( s.executions ) executions ,
                               MAX( f.executions ) face_exec ,
                               MAX( s.buffer_gets ) buffer_gets ,
                               MAX( f.buffer_gets ) face_bgets ,
                               MAX( s.disk_reads ) disk_reads ,
                               MAX( s.rows_processed ) rows_processed ,
                               MAX( s.elapsed_time ) elapsed_time ,
                               MAX( s.cpu_time ) cpu_time ,
                               MAX( f.inst_id ) inst_id ,
                               MAX( DISTINCT f.inst_id ) inst_cnt
                          FROM w_sql_pt p ,
                               w_sql_face f ,
                               ( --같은 Plan_Hash_value 를 같는 쿼리의 일량을 모두 합침
SELECT inst_id ,
                                        COUNT( DISTINCT sql_id ) liter_cnt ,
                                        MAX( version_count ) version_count ,
                                        SUM( executions ) executions ,
                                        MIN( first_load_time ) first_load_time ,
```

```
                                    SUM( disk_reads ) disk_reads ,
                                    SUM( direct_writes ) direct_writes ,
                                    SUM( buffer_gets ) buffer_gets ,
                                    SUM( rows_processed ) rows_processed ,
                                    AVG( optimizer_cost ) optimizer_cost ,
                                    MAX( parsing_schema_name ) parsing_schema_name ,
                                    MAX( hash_value ) hash_value ,
                                    plan_hash_value ,
                                    MAX( MODULE ) MODULE ,
                                    SUM( cpu_time ) cpu_time ,
                                    SUM( elapsed_time ) elapsed_time ,
                                    MAX( last_load_time ) last_load_time
                               FROM w_sql_face
                              WHERE 1=1
                              GROUP BY inst_id ,
                                    plan_hash_value ) s
                        WHERE f.rk = 1
                          AND f.plan_hash_value = s.plan_hash_value
                          AND f.inst_id = s.inst_id
                          AND f.sql_id = p.sql_id
                          AND f.inst_id = p.inst_id
                        GROUP BY p.id ,
                              p.sql_id ,
                              p.parent_id ,
                              p.plan_hash_value ) x ) a ) a
ORDER BY bg_seq# ,
      seq# ;
```