Trabajo 4
Índices Bitmap vs Índices B

Facultad De Ingeniería, Universidad De Cuenca
BASES DE DATOS 2
Freddy L. Abad L.
{freddy.abadl}@ucuenca.edu.ec

```
SQL> Create  table test_normal (empno number(10), ename varchar2(30), sal number(10));

Table created.
```

```
SQL> Begin
  2   For i in  1..1000000
  3   Loop
  4   Insert into test_normal
  5   values(i, dbms_random.string('U',30), dbms_random.value(1000,7000));
  6   If mod(i, 10000) = 0 then
  7   Commit;
  8   End if;
  9   End loop;
 10   End;
 11   /

PL/SQL procedure successfully completed.
```

```
SQL> Create  table test_random
  2  as
  3  select  /*+ append */ * from test_normal order by dbms_random.random;

Table created.
```

```
SQL> select count(*) "Total Rows" from test_normal;

Total Rows
----------
   1880000
```

```
SQL> select count(distinct empno) "Distinct Values" from test_normal;

Distinct Values
---------------
        1000000
```

```
SQL> select count(*) "Total Rows" from test_random;

Total Rows
----------
   1880000
```

```
SQL> select count(distinct empno) "Distinct Values" from test_random;

Distinct Values
---------------
        1000000
```

Paso 1A (con TEST_NORMAL)

En este paso, crearemos un índice de mapa de bits para la tabla TEST_NORMAL y, a continuación, verificaremos el tamaño del índice, su factor de agrupamiento [*clustering factor*] y el tamaño de la tabla. Luego ejecutaremos algunas consultas con predicados de igualdad y registraremos las E/S de las consultas empleando el índice de mapa de bits.

```
SQL> create bitmap index normal_empno_bmx on test_normal(empno);

Index created.
```

```
SQL> analyze table test_normal compute statistics for table for all indexes for all  indexed columns;

Table analyzed.
```

```
SQL> select substr(segment_name,1,30) segment_name, bytes/1024/1024 "Size in  MB"
  2  from  user_segments
  3  where segment_name in ('TEST_NORMAL','NORMAL_EMPNO_BMX');

SEGMENT_NAME                   Size in  MB
------------------------------ ----------
NORMAL_EMPNO_BMX                       32
TEST_NORMAL                            96
```

```
SQL> select substr(segment_name,1,30) segment_name, bytes/1024/1024 "Size in  MB"
  2  from  user_segments
  3  where segment_name in ('TEST_NORMAL','NORMAL_EMPNO_BMX');

SEGMENT_NAME                   Size in  MB
------------------------------ ----------
NORMAL_EMPNO_BMX                       32
TEST_NORMAL                            96

SQL> select index_name, clustering_factor from user_indexes;

INDEX_NAME                     CLUSTERING_FACTOR
------------------------------ -----------------
NORMAL_EMPNO_BMX                         1000000
SYS_IL0000064060C00005$$
SYS_IL0000064060C00004$$
HELP_TOPIC_SEQ                                10
SYS_C003491
SYS_IL0000012833C00025$$
SYS_C003488
SYS_IL0000012822C00025$$
DEF$_TRANORDER
LOGSTDBY$EDS_TABLES_PKEY                       0
LOGSTDBY$SKIP_IND                              1
```

```
INDEX_NAME                     CLUSTERING_FACTOR
------------------------------ -----------------
LOGSTDBY$SKIP_IDX1                             0
LOGSTDBY$SKIP_IDX2                             0
SYS_IL0000010618C00003$$
SYS_C003294                                    0
SYS_IL0000010606C00009$$
LOGSTDBY$EVENTS_IND                            0
LOGSTDBY$EVENTS_IND_SCN                        0
LOGSTDBY$EVENTS_IND_XID                        0
LOGMNRP_CTAS_PART_MAP_PK                       0
LOGMNRP_CTAS_PART_MAP_I                        0
LOGMNR_I1PARTOBJ$                              0

INDEX_NAME                     CLUSTERING_FACTOR
------------------------------ -----------------
LOGMNR_I1REFCON$                               0
LOGMNR_I1ENC$                                  0
LOGMNR_I1PROPS$                                0
LOGMNR_I1KOPM$                                 0
LOGMNR_I1SUBCOLTYPE$                           0
LOGMNR_I1OPQTYPE$                              0
LOGMNR_I1NTAB$                                 0
LOGMNR_I2NTAB$                                 0
LOGMNR_I1LOGMNR_BUILDLOG                       0
LOGMNR_I1INDCOMPART$                           0
LOGMNR_I1INDSUBPART$                           0
```

```
INDEX_NAME                     CLUSTERING_FACTOR
------------------------------ -----------------
LOGMNR_I1INDPART$                              0
LOGMNR_I2INDPART$                              0
LOGMNR_I1LOBFRAG$                              0
LOGMNR_I1ICOL$                                 0
LOGMNR_I1CCOL$                                 0
LOGMNR_I1CDEF$                                 0
LOGMNR_I1LOB$                                  0
LOGMNR_I1ATTRIBUTE$                            0
LOGMNR_I1COLTYPE$                              0
LOGMNR_I1TYPE$                                 0
LOGMNR_I1TABCOMPART$                           0

INDEX_NAME                     CLUSTERING_FACTOR
------------------------------ -----------------
LOGMNR_I2TABCOMPART$                           0
LOGMNR_I1TABSUBPART$                           0
LOGMNR_I2TABSUBPART$                           0
LOGMNR_I1TABPART$                              0
LOGMNR_I2TABPART$                              0
LOGMNR_I1USER$                                 0
LOGMNR_I1IND$                                  0
LOGMNR_I2IND$                                  0
LOGMNR_I1TS$                                   0
LOGMNR_I1ATTRCOL$                              0
LOGMNR_I1COL$                                  0

INDEX_NAME                     CLUSTERING_FACTOR
------------------------------ -----------------
LOGMNR_I2COL$                                  0
LOGMNR_I3COL$                                  0
LOGMNR_I1TAB$                                  0
LOGMNR_I2TAB$                                  0
LOGMNR_I1OBJ$                                  0
LOGMNR_I2OBJ$                                  0
LOGMNR_I1DICTIONARY$                           0
LOGMNR_I1SEED$                                 0
LOGMNR_I2SEED$                                 0
LOGMNRC_GSBA_PK                                0
LOGMNRC_GSII_PK                                0
```

INDEX_NAME	CLUSTERING_FACTOR
LOGMNRC_GTCS_PK	0
LOGMNRC_I2GTCS	0
LOGMNRC_GTLO_PK	0
LOGMNRC_I2GTLO	0
LOGMNRC_I3GTLO	0
REPCAT$_SITES_NEW_PK	0
REPCAT$_SITES_NEW_FK2_IDX	0
REPCAT$_SITES_NEW_FK1_IDX	0
SYS_C003040	0
REPCAT$_INSTANTIATION_DDL_PK	0
SYS_IL0000007247C00002$$	

INDEX_NAME	CLUSTERING_FACTOR
REPCAT$_EXCEPTIONS_PK	0
SYS_IL0000007240C00003$$	
TEMPLATE$_TARGETS_PK	0
REPCAT$_TEMPLATE_TARGETS_U1	0
SYS_IL0000007229C00003$$	
REPCAT$_RUNTIME_PARMS_PK	0
REPCAT$_SITE_OBJECTS_U1	0
REPCAT$_SITE_OBJECTS_N1	0
REPCAT$_TEMPLATE_SITES_PK	0
REPCAT$_TEMPLATE_SITES_U1	0
REPCAT$_USER_PARM_VALUES_PK	0

INDEX_NAME	CLUSTERING_FACTOR
SYS_IL0000007203C00004$$	
REPCAT$_USER_PARM_VALUES_U1	0
REPCAT$_OBJECT_PARMS_PK	0
REPCAT$_OBJECT_PARMS_N2	0
REPCAT$_TEMPLATE_PARMS_PK	0
SYS_IL0000007188C00004$$	
REPCAT$_TEMPLATE_PARMS_U1	0
REPCAT$_TEMPLATE_OBJECTS_PK	0
SYS_IL0000007175C00006$$	
REPCAT$_TEMPLATE_OBJECTS_U1	0
REPCAT$_TEMPLATE_OBJECTS_N1	0

INDEX_NAME	CLUSTERING_FACTOR
REPCAT$_TEMPLATE_OBJECTS_N2	0
REPCAT$_TEMPLATE_REFGROUPS_PK	0
REPCAT$_TEMPLATE_REFGROUPS_N1	0
REPCAT$_TEMPLATE_REFGROUPS_N2	0
REPCAT$_OBJECT_TYPE_PK	1
REPCAT$_USER_AUTHORIZATIONS_PK	0
REPCAT$_USER_AUTHORIZATIONS_U1	0
REPCAT$_USER_AUTHORIZATIONS_N1	0
REPCAT$_REFRESH_TEMPLATES_PK	0
REPCAT$_REFRESH_TEMPLATES_U1	0
REPCAT$_TEMPLATE_TYPES_PK	1

INDEX_NAME	CLUSTERING_FACTOR
REPCAT$_TEMPLATE_STATUS_PK	1
REPCAT$_FLAVOR_OBJECTS_PK	0
REPCAT$_FLAVOR_OBJECTS_FG	0
REPCAT$_FLAVOR_OBJECTS_FK1_IDX	0
REPCAT$_FLAVOR_OBJECTS_FK2_IDX	0
REPCAT$_AUDIT_COLUMN_PK	0
REPCAT$_AUDIT_COLUMN_F1_IDX	0
REPCAT$_AUDIT_COLUMN_F2_IDX	0
REPCAT$_AUDIT_ATTRIBUTE_PK	1
REPCAT$_PARAMETER_COLUMN_PK	0
REPCAT$_PARAMETER_COLUMN_F1_I	0

INDEX_NAME	CLUSTERING_FACTOR
REPCAT$_RESOL_STATS_CTRL_PK	0
REPCAT$_RESOLUTION_STATS_N1	0
REPCAT$_RESOLUTION_PK	0
REPCAT$_RESOLUTION_F3_IDX	0
REPCAT$_RESOLUTION_IDX2	0
REPCAT$_RESOL_METHOD_PK	1
REPCAT$_CONFLICT_PK	0
REPCAT$_GROUPED_COLUMN_PK	0
REPCAT$_GROUPED_COLUMN_F1_IDX	0
REPCAT$_COLUMN_GROUP_PK	0
REPCAT$_PRIORITY_PK	0

INDEX_NAME	CLUSTERING_FACTOR
REPCAT$_PRIORITY_F1_IDX	0
REPCAT$_PRIORITY_GROUP_PK	0
REPCAT$_PRIORITY_GROUP_U1	0
REPCAT$_REPGROUP_PRIVS_UK	0
REPCAT$_REPGROUP_PRIVS_N1	0
REPCAT$_REPGROUP_PRIVS_FK_IDX	0
REPCAT$_DDL_INDEX	0
REPCAT$_DDL	0
REPCAT$_REPCATLOG_PRIMARY	0
REPCAT$_REPCATLOG_GNAME	0
REPCAT$_REPPROP_PRIMARY	0

INDEX_NAME	CLUSTERING_FACTOR
REPCAT$_REPPROP_DBLINK_HOW	0
REPCAT$_REPPROP_KEY_INDEX	0
REPCAT$_REPPROP_PRNT_IDX	0
REPCAT$_REPPROP_PRNT2_IDX	0
REPCAT$_REPGEN_PRIMARY	0
REPCAT$_GENERATED_N1	0
REPCAT$_REPGEN_PRNT_IDX	0
REPCAT$_KEY_COLUMNS_PRIMARY	0
REPCAT$_KEY_COLUMNS_PRNT_IDX	0
REPCAT$_REPCOLUMN_PK	0
REPCAT$_REPCOLUMN_FK_IDX	0

INDEX_NAME	CLUSTERING_FACTOR
REPCAT$_REPOBJECT_PRIMARY	0
REPCAT$_REPOBJECT_GNAME	0
REPCAT$_REPOBJECT_PRNT_IDX	0
I_REPCAT$_SNAPGROUP1	0
REPCAT$_REPSCHEMA_PRIMARY	0
REPCAT$_REPSCHEMA_DEST_IDX	0
REPCAT$_REPSCHEMA_PRNT_IDX	0
REPCAT$_FLAVORS_UNQ1	0
REPCAT$_FLAVORS_FNAME	0
REPCAT$_FLAVORS_GNAME	0
REPCAT$_FLAVORS_FK1_IDX	0

INDEX_NAME	CLUSTERING_FACTOR
REPCAT$_REPCAT_PRIMARY	0
DEF$_PUSHED_TRAN_PRIMARY	0
DEF$_PROPAGATOR_PRIMARY	0
DEF$_LOB_PRIMARY	0
SYS_IL0000006819C00005$$	
SYS_IL0000006819C00004$$	
SYS_IL0000006819C00003$$	
DEF$_LOB_N1	0
DEF$_DEFAULTDEST_PRIMARY	0
DEF$_CALLDEST_PRIMARY	0
DEF$_CALLDEST_N2	0

INDEX_NAME	CLUSTERING_FACTOR
DEF$_DESTINATION_PRIMARY	0
DEF$_ERROR_PRIMARY	0
SYS_IL0000005961C00021$$	
OL$HNT_NUM	
OL$NAME	
OL$SIGNATURE	
UNQ_PAIRS	1
SYS_C001684	1
AQ$_SCHEDULES_PRIMARY	0
AQ$_SCHEDULES_CHECK	0
AQ$_QUEUES_PRIMARY	1

INDEX_NAME	CLUSTERING_FACTOR
SYS_IL0000005598C00012$$	
AQ$_QUEUES_CHECK	1
I1_QUEUES	1
AQ$_QUEUE_TABLES_PRIMARY	1
I1_QUEUE_TABLES	1
MVIEW$_ADV_JOURNAL_PK	0
MVIEW$_ADV_INFO_PK	0
MVIEW$_ADV_PARAMETERS_PK	1
MVIEW$_ADV_OUTPUT_PK	0
MVIEW$_ADV_ELIGIBLE_PK	0
MVIEW$_ADV_CLIQUE_PK	0

```
INDEX_NAME                     CLUSTERING_FACTOR
------------------------------ -----------------
MVIEW$_ADV_GC_PK                               0
MVIEW$_ADV_FJG_PK                              0
MVIEW$_ADV_AJG_PK                              0
MVIEW$_ADV_ROLLUP_PK                           0
MVIEW$_ADV_LEVEL_PK                            0
MVIEW$_ADV_LOG_PK                              0
MVIEW$_ADV_FILTER_PK                           0
MVIEW$_ADV_TEMP_IDX_01                         0
MVIEW$_ADV_PRETTY_IDX_01                       0
MVIEW$_ADV_SQLDEPEND_IDX_01                    0
MVIEW$_ADV_BASETABLE_IDX_01                    0

INDEX_NAME                     CLUSTERING_FACTOR
------------------------------ -----------------
MVIEW$_ADV_WORKLOAD_PK                         0
MVIEW$_ADV_WORKLOAD_IDX_01                     0
LOGMNRT_MDDL$_PK
LOGMNR_SESSION_PK                              0
LOGMNR_SESSION_UK1                             0
LOGMNR_PARAMETER_INDX                          0
LOGMNR_SESSION_ACTION$_PK                      0
LOGMNR_RESTART_CKPT$_PK                        0
SYS_IL0000001087C00012$$
SYS_IL0000001087C00009$$
LOGMNR_RESTART_CKPT_TXINFO$_PK                 0

INDEX_NAME                     CLUSTERING_FACTOR
------------------------------ -----------------
SYS_IL0000001082C00011$$
LOGMNR_AGE_SPILL$_PK                           0
SYS_IL0000001078C00008$$
LOGMNR_SPILL$_PK                               0
SYS_IL0000001074C00010$$
LOGMNR_PROCESSED_LOG$_PK                       0
LOGMNR_LOG$_PK                                 0
LOGMNR_LOG$_FLAGS                              0
LOGMNR_LOG$_FIRST_CHANGE#                      0
LOGMNR_LOG$_RECID                              0
LOGMNRC_DBNAME_UID_MAP_PK                      0
```

```
INDEX_NAME                     CLUSTERING_FACTOR
------------------------------ -----------------
LOGMNR_UID$_PK                                 0
LOGMNR_SESSION_EVOLVE$_PK                      0

233 rows selected.
```

```
SQL>  set autotrace only
Usage: SET AUTOT[RACE] {OFF | ON | TRACE[ONLY]} [EXP[LAIN]] [STAT[ISTICS]]
```

```
SQL> select * from test_normal where empno=&empno;
Enter value for empno: 1000
old   1: select * from test_normal where empno=&empno
new   1: select * from test_normal where empno=1000

     EMPNO ENAME                                SAL
---------- ------------------------------ ----------
      1000 RAEUMEEGGIEXTYZZEIDLCARDBAQREP       6393
      1000 CMPNNEHLQMQAOVIWYODLTMUWJPUJLB       3947
      1000 ODIPGMCLYGYLSONLRZJLTLQLOTJCMF       3065
```

Paso 1B (con TEST_NORMAL)

Ahora quitaremos el índice de mapa de bits y crearemos un índice de árbol B asociado a la columna EMPNO. Como antes, verificaremos el tamaño del índice y el factor de agrupamiento, y ejecutaremos algunas consultas para el mismo conjunto de valores a fin de comparar las E/S.

```
SQL> drop index NORMAL_EMPNO_BMX;

Index dropped.
```

```
SQL> create index normal_empno_idx on test_normal(empno);

Index created.
```

```
SQL> analyze table test_normal compute statistics for table for all indexes for all  indexed columns;

Table analyzed.
```

```
SQL> analyze table test_normal compute statistics for table for all indexes for all  indexed columns;

Table analyzed.
```

```
SQL> set autot trace
```

```
SQL> select * from test_normal where empno=&empno;
Enter value for empno: 1000
old   1: select * from test_normal where empno=&empno
new   1: select * from test_normal where empno=1000
```

```
Execution Plan
----------------------------------------------------------
Plan hash value: 1781697849

-------------------------------------------------------------------------------
---------------

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

-------------------------------------------------------------------------------
---------------

|   0 | SELECT STATEMENT            |                 |     2 |    68 |     5
 (0)| 00:00:01 |

|   1 |  TABLE ACCESS BY INDEX ROWID| TEST_NORMAL     |     2 |    68 |     5
 (0)| 00:00:01 |

|*  2 |   INDEX RANGE SCAN          | NORMAL_EMPNO_IDX |     2 |       |     3
 (0)| 00:00:01 |

-------------------------------------------------------------------------------
---------------
```

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

   2 - access("EMPNO"=1000)


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
          7  consistent gets
          0  physical reads
          0  redo size
        848  bytes sent via SQL*Net to client
        519  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          3  rows processed
```

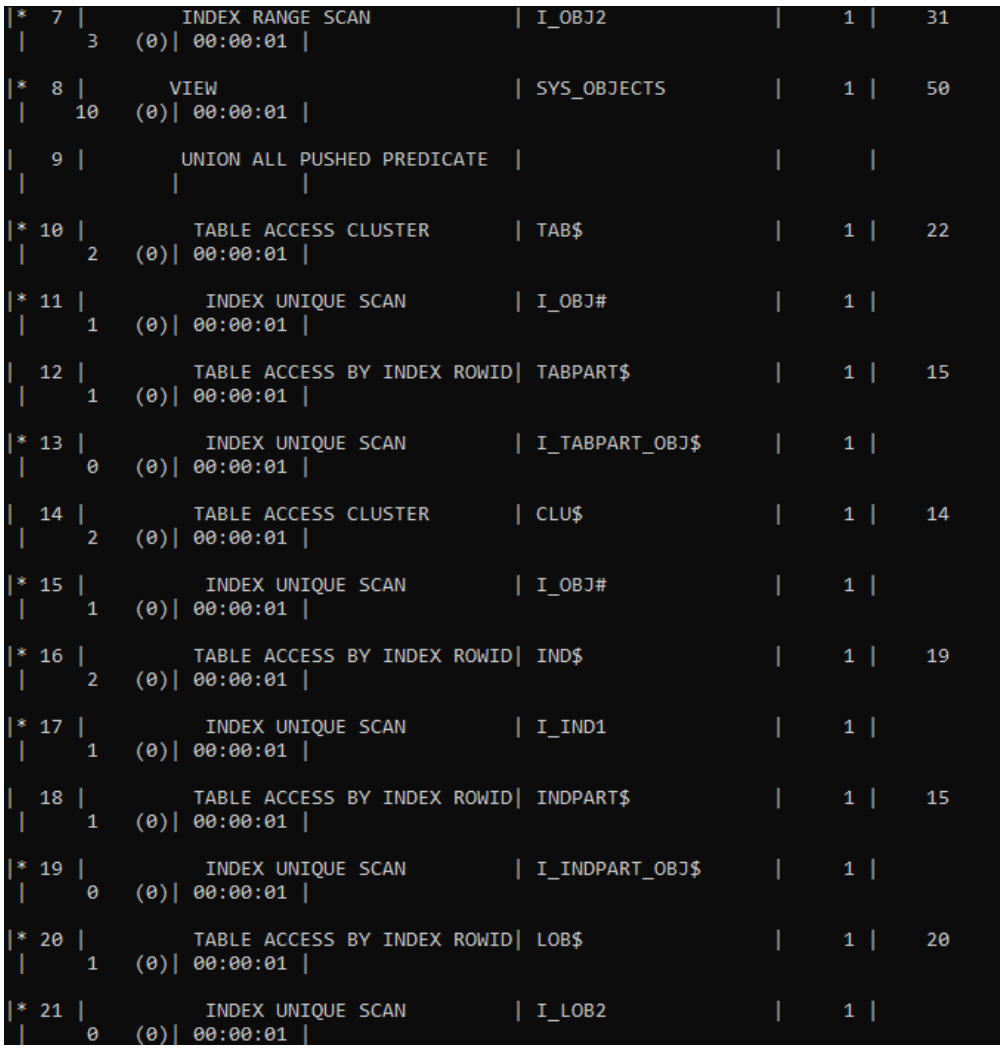
Paso 2A (con TEST_RANDOM)

Ahora realizaremos el mismo experimento con TEST_RANDOM

```
SQL> create bitmap index random_empno_bmx on test_random(empno);

Index created.
```

```
SQL> analyze table test_random compute statistics for table for all indexes for all  indexed columns;

Table analyzed.
```

```
SQL> select substr(segment_name,1,30) segment_name, bytes/1024/1024 "Size in  MB"
  2  from  user_segments
  3  where segment_name in  ('TEST_RANDOM','RANDOM_EMPNO_BMX');


Execution Plan
----------------------------------------------------------
Plan hash value: 2297303168

-------------------------------------------------------------------------------
--------------------------

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

-------------------------------------------------------------------------------
--------------------------

|   0 | SELECT STATEMENT              |                   |     7 |  1113
 |    37   (3)| 00:00:01 |

|   1 |  VIEW                         | SYS_USER_SEGS     |     7 |  1113
 |    37   (3)| 00:00:01 |

|   2 |   UNION-ALL                   |                   |       |
 |            |          |

|   3 |    NESTED LOOPS               |                   |     1 |   119
 |    15   (0)| 00:00:01 |

|   4 |     NESTED LOOPS              |                   |     1 |   112
 |    14   (0)| 00:00:01 |

|   5 |      NESTED LOOPS             |                   |     1 |    81
 |    13   (0)| 00:00:01 |

|   6 |       INLIST ITERATOR         |                   |       |
 |            |          |
```



```
|*  7 |        INDEX RANGE SCAN           | I_OBJ2           |     1 |    31
 |     3   (0)| 00:00:01 |

|*  8 |       VIEW                        | SYS_OBJECTS      |     1 |    50
 |    10   (0)| 00:00:01 |

|   9 |        UNION ALL PUSHED PREDICATE |                  |       |
 |           |          |

|* 10 |         TABLE ACCESS CLUSTER      | TAB$             |     1 |    22
 |     2   (0)| 00:00:01 |

|* 11 |          INDEX UNIQUE SCAN        | I_OBJ#           |     1 |
 |     1   (0)| 00:00:01 |

|  12 |         TABLE ACCESS BY INDEX ROWID| TABPART$         |     1 |    15
 |     1   (0)| 00:00:01 |

|* 13 |          INDEX UNIQUE SCAN        | I_TABPART_OBJ$   |     1 |
 |     0   (0)| 00:00:01 |

|  14 |         TABLE ACCESS CLUSTER      | CLU$             |     1 |    14
 |     2   (0)| 00:00:01 |

|* 15 |          INDEX UNIQUE SCAN        | I_OBJ#           |     1 |
 |     1   (0)| 00:00:01 |

|* 16 |         TABLE ACCESS BY INDEX ROWID| IND$             |     1 |    19
 |     2   (0)| 00:00:01 |

|* 17 |          INDEX UNIQUE SCAN        | I_IND1           |     1 |
 |     1   (0)| 00:00:01 |

|  18 |         TABLE ACCESS BY INDEX ROWID| INDPART$         |     1 |    15
 |     1   (0)| 00:00:01 |

|* 19 |          INDEX UNIQUE SCAN        | I_INDPART_OBJ$   |     1 |
 |     0   (0)| 00:00:01 |

|* 20 |         TABLE ACCESS BY INDEX ROWID| LOB$             |     1 |    20
 |     1   (0)| 00:00:01 |

|* 21 |          INDEX UNIQUE SCAN        | I_LOB2           |     1 |
 |     0   (0)| 00:00:01 |

|  22 |         TABLE ACCESS BY INDEX ROWID| TABSUBPART$      |     1 |    52
 |     0   (0)| 00:00:01 |

|* 23 |          INDEX UNIQUE SCAN        | I_TABSUBPART$_OBJ$ |     1 |
 |     0   (0)| 00:00:01 |

|  24 |         TABLE ACCESS BY INDEX ROWID| INDSUBPART$      |     1 |    52
 |     0   (0)| 00:00:01 |

|* 25 |          INDEX UNIQUE SCAN        | I_INDSUBPART_OBJ$ |     1 |
 |     0   (0)| 00:00:01 |

|  26 |         TABLE ACCESS BY INDEX ROWID| LOBFRAG$         |     1 |    17
 |     1   (0)| 00:00:01 |

|* 27 |          INDEX UNIQUE SCAN        | I_LOBFRAG$_FRAGOBJ$ |     1 |
 |     0   (0)| 00:00:01 |

|* 28 |       TABLE ACCESS CLUSTER        | SEG$             |     1 |    31
 |     1   (0)| 00:00:01 |

|* 29 |        INDEX UNIQUE SCAN          | I_FILE#_BLOCK#   |     1 |
 |     0   (0)| 00:00:01 |

|  30 |      TABLE ACCESS CLUSTER         | TS$              |     1 |     7
 |     1   (0)| 00:00:01 |

|* 31 |       INDEX UNIQUE SCAN           | I_TS#            |     1 |
 |     0   (0)| 00:00:01 |

|  32 |     NESTED LOOPS                  |                  |     2 |   148
 |     6   (0)| 00:00:01 |

|  33 |      NESTED LOOPS                 |                  |     2 |   134
 |     4   (0)| 00:00:01 |

|* 34 |       TABLE ACCESS FULL           | UNDO$            |     2 |    66
 |     2   (0)| 00:00:01 |
```

```
|* 35 |      TABLE ACCESS CLUSTER          | SEG$               |     1 |    34
 |     1   (0)| 00:00:01 |

|* 36 |       INDEX UNIQUE SCAN            | I_FILE#_BLOCK#     |     1 |
 |     0   (0)| 00:00:01 |

|  37 |     TABLE ACCESS CLUSTER           | TS$                |     1 |     7
 |     1   (0)| 00:00:01 |

|* 38 |      INDEX UNIQUE SCAN             | I_TS#              |     1 |
 |     0   (0)| 00:00:01 |

|* 39 |    HASH JOIN                       |                    |     4 |   200
 |    16   (7)| 00:00:01 |

|  40 |     NESTED LOOPS                   |                    |     4 |   172
 |    12   (0)| 00:00:01 |

|  41 |      TABLE ACCESS FULL             | FILE$              |     4 |    36
 |     2   (0)| 00:00:01 |

|* 42 |      TABLE ACCESS CLUSTER          | SEG$               |     1 |    34
 |     3   (0)| 00:00:01 |

|* 43 |       INDEX RANGE SCAN             | I_FILE#_BLOCK#     |     1 |
 |     2   (0)| 00:00:01 |

|  44 |     TABLE ACCESS FULL              | TS$                |     5 |    35
 |     3   (0)| 00:00:01 |

---------------------------------------------------------------------------
---------------------------
```

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

   7 - access("O"."OWNER#"=USERENV('SCHEMAID') AND ("O"."NAME"='RANDOM_EMPNO_BMX
' OR

              "O"."NAME"='TEST_RANDOM'))
   8 - filter("O"."TYPE#"="SO"."OBJECT_TYPE_ID")
  10 - filter(BITAND("T"."PROPERTY",1024)=0)
  11 - access("T"."OBJ#"="O"."OBJ#")
  13 - access("TP"."OBJ#"="O"."OBJ#")
  15 - access("C"."OBJ#"="O"."OBJ#")
  16 - filter("I"."TYPE#"=1 OR "I"."TYPE#"=2 OR "I"."TYPE#"=3 OR "I"."TYPE#"=4 O
R "I"."TYPE#"=6

              OR "I"."TYPE#"=7 OR "I"."TYPE#"=8 OR "I"."TYPE#"=9)
  17 - access("I"."OBJ#"="O"."OBJ#")
  19 - access("IP"."OBJ#"="O"."OBJ#")
  20 - filter(BITAND("L"."PROPERTY",64)=0 OR BITAND("L"."PROPERTY",128)=128)
  21 - access("L"."LOBJ#"="O"."OBJ#")
  23 - access("TSP"."OBJ#"="O"."OBJ#")
  25 - access("ISP"."OBJ#"="O"."OBJ#")
  27 - access("LF"."FRAGOBJ#"="O"."OBJ#")
  28 - filter("S"."TYPE#"="SO"."SEGMENT_TYPE_ID")
  29 - access("S"."TS#"="SO"."TS_NUMBER" AND "S"."FILE#"="SO"."HEADER_FILE" AND
              "S"."BLOCK#"="SO"."HEADER_BLOCK")
  31 - access("S"."TS#"="TS"."TS#")
  34 - filter(("UN"."NAME"='RANDOM_EMPNO_BMX' OR "UN"."NAME"='TEST_RANDOM') AND
"UN"."STATUS$"<>1)

  35 - filter(("S"."TYPE#"=1 OR "S"."TYPE#"=10) AND "S"."USER#"=USERENV('SCHEMAI
D'))

  36 - access("S"."TS#"="UN"."TS#" AND "S"."FILE#"="UN"."FILE#" AND "S"."BLOCK#"
="UN"."BLOCK#")

  38 - access("S"."TS#"="TS"."TS#")
  39 - access("S"."TS#"="TS"."TS#")
  42 - filter("S"."TYPE#"<>1 AND "S"."TYPE#"<>5 AND "S"."TYPE#"<>6 AND "S"."TYPE
#"<>8 AND

              "S"."TYPE#"<>10 AND "S"."USER#"=USERENV('SCHEMAID'))
  43 - access("S"."TS#"="F"."TS#" AND "S"."FILE#"="F"."RELFILE#")
       filter(TO_CHAR("F"."FILE#")||'.'||TO_CHAR("S"."BLOCK#")='TEST_RANDOM' OR
              TO_CHAR("F"."FILE#")||'.'||TO_CHAR("S"."BLOCK#")='RANDOM_EMPNO_BMX
')



Statistics
----------------------------------------------------------
         22  recursive calls
          0  db block gets
         95  consistent gets
         41  physical reads
          0  redo size
        704  bytes sent via SQL*Net to client
        519  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          2  rows processed
```

```
SQL> select * from test_random where empno=&empno;
Enter value for empno: 1000
old   1: select * from test_random where empno=&empno
new   1: select * from test_random where empno=1000


Execution Plan
----------------------------------------------------------
Plan hash value: 3133668422

-------------------------------------------------------------------------------
----------------

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

-------------------------------------------------------------------------------
----------------

|   0 | SELECT STATEMENT             |                 |     2 |    68 |     3
  (0)| 00:00:01 |

|   1 |  TABLE ACCESS BY INDEX ROWID | TEST_RANDOM     |     2 |    68 |     3
  (0)| 00:00:01 |

|   2 |   BITMAP CONVERSION TO ROWIDS|                 |       |       |
     |          |

|*  3 |    BITMAP INDEX SINGLE VALUE | RANDOM_EMPNO_BMX |       |       |
     |          |

-------------------------------------------------------------------------------
----------------
```

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

   3 - access("EMPNO"=1000)


Statistics
----------------------------------------------------------
         44  recursive calls
          0  db block gets
         11  consistent gets
          0  physical reads
          0  redo size
        848  bytes sent via SQL*Net to client
        519  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          3  rows processed
```

Paso 2B (con TEST_RANDOM)

Ahora, como en el paso 1B, quitaremos el índice de mapa de bits y crearemos un índice de árbol B asociado a la columna EMPNO.

```
SQL> drop index RANDOM_EMPNO_BMX;

Index dropped.
```

```
SQL> create index random_empno_idx on test_random(empno);

Index created.
```

```
SQL> select substr(segment_name,1,30) segment_name, bytes/1024/1024 "Size in  MB"
  2  from  user_segments
  3  where segment_name in ('TEST_RANDOM','RANDOM_EMPNO_IDX');


Execution Plan
----------------------------------------------------------
Plan hash value: 2297303168

-------------------------------------------------------------------------------
-------------------------

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

-------------------------------------------------------------------------------
-------------------------
```

```
-------------------------------------------------------------------------------
-------------------------

|   0 | SELECT STATEMENT                   |                    |     7 |  1113
 |    37   (3)| 00:00:01 |

|   1 |  VIEW                              | SYS_USER_SEGS      |     7 |  1113
 |    37   (3)| 00:00:01 |

|   2 |   UNION-ALL                        |                    |       |
 |            |          |

|   3 |    NESTED LOOPS                    |                    |     1 |   119
 |    15   (0)| 00:00:01 |

|   4 |     NESTED LOOPS                   |                    |     1 |   112
 |    14   (0)| 00:00:01 |

|   5 |      NESTED LOOPS                  |                    |     1 |    81
 |    13   (0)| 00:00:01 |

|   6 |       INLIST ITERATOR              |                    |       |
 |            |          |

|*  7 |        INDEX RANGE SCAN            | I_OBJ2             |     1 |    31
 |     3   (0)| 00:00:01 |

|*  8 |       VIEW                         | SYS_OBJECTS        |     1 |    50
 |    10   (0)| 00:00:01 |

|   9 |        UNION ALL PUSHED PREDICATE  |                    |       |
 |            |          |

|* 10 |         TABLE ACCESS CLUSTER       | TAB$               |     1 |    22
 |     2   (0)| 00:00:01 |

|* 11 |          INDEX UNIQUE SCAN         | I_OBJ#             |     1 |
 |     1   (0)| 00:00:01 |

|  12 |         TABLE ACCESS BY INDEX ROWID| TABPART$           |     1 |    15
 |     1   (0)| 00:00:01 |

|* 13 |          INDEX UNIQUE SCAN         | I_TABPART_OBJ$     |     1 |
 |     0   (0)| 00:00:01 |
```

```
|  14 |         TABLE ACCESS CLUSTER       | CLU$                |     1 |    14
 |     2   (0)| 00:00:01 |

|* 15 |          INDEX UNIQUE SCAN         | I_OBJ#              |     1 |
 |     1   (0)| 00:00:01 |

|* 16 |         TABLE ACCESS BY INDEX ROWID| IND$                |     1 |    19
 |     2   (0)| 00:00:01 |

|* 17 |          INDEX UNIQUE SCAN         | I_IND1              |     1 |
 |     1   (0)| 00:00:01 |

|  18 |         TABLE ACCESS BY INDEX ROWID| INDPART$            |     1 |    15
 |     1   (0)| 00:00:01 |

|* 19 |          INDEX UNIQUE SCAN         | I_INDPART_OBJ$      |     1 |
 |     0   (0)| 00:00:01 |

|* 20 |         TABLE ACCESS BY INDEX ROWID| LOB$                |     1 |    20
 |     1   (0)| 00:00:01 |

|* 21 |          INDEX UNIQUE SCAN         | I_LOB2              |     1 |
 |     0   (0)| 00:00:01 |

|  22 |         TABLE ACCESS BY INDEX ROWID| TABSUBPART$         |     1 |    52
 |     0   (0)| 00:00:01 |

|* 23 |          INDEX UNIQUE SCAN         | I_TABSUBPART$_OBJ$  |     1 |
 |     0   (0)| 00:00:01 |

|  24 |         TABLE ACCESS BY INDEX ROWID| INDSUBPART$         |     1 |    52
 |     0   (0)| 00:00:01 |

|* 25 |          INDEX UNIQUE SCAN         | I_INDSUBPART_OBJ$   |     1 |
 |     0   (0)| 00:00:01 |

|  26 |         TABLE ACCESS BY INDEX ROWID| LOBFRAG$            |     1 |    17
 |     1   (0)| 00:00:01 |

|* 27 |          INDEX UNIQUE SCAN         | I_LOBFRAG$_FRAGOBJ$ |     1 |
 |     0   (0)| 00:00:01 |
```

```
|* 28 |       TABLE ACCESS CLUSTER         | SEG$                |     1 |    31
 |     1   (0)| 00:00:01 |

|* 29 |        INDEX UNIQUE SCAN           | I_FILE#_BLOCK#      |     1 |
 |     0   (0)| 00:00:01 |

|  30 |       TABLE ACCESS CLUSTER         | TS$                 |     1 |     7
 |     1   (0)| 00:00:01 |

|* 31 |        INDEX UNIQUE SCAN           | I_TS#               |     1 |
 |     0   (0)| 00:00:01 |

|  32 |      NESTED LOOPS                  |                     |     2 |   148
 |     6   (0)| 00:00:01 |

|  33 |       NESTED LOOPS                 |                     |     2 |   134
 |     4   (0)| 00:00:01 |

|* 34 |        TABLE ACCESS FULL           | UNDO$               |     2 |    66
 |     2   (0)| 00:00:01 |

|* 35 |        TABLE ACCESS CLUSTER        | SEG$                |     1 |    34
 |     1   (0)| 00:00:01 |

|* 36 |         INDEX UNIQUE SCAN          | I_FILE#_BLOCK#      |     1 |
 |     0   (0)| 00:00:01 |

|  37 |       TABLE ACCESS CLUSTER         | TS$                 |     1 |     7
 |     1   (0)| 00:00:01 |

|* 38 |        INDEX UNIQUE SCAN           | I_TS#               |     1 |
 |     0   (0)| 00:00:01 |

|* 39 |      HASH JOIN                     |                     |     4 |   200
 |    16   (7)| 00:00:01 |

|  40 |       NESTED LOOPS                 |                     |     4 |   172
 |    12   (0)| 00:00:01 |

|  41 |        TABLE ACCESS FULL           | FILE$               |     4 |    36
 |     2   (0)| 00:00:01 |
```

```
|* 42 |      TABLE ACCESS CLUSTER          | SEG$                |     1 |    34
 |     3   (0)| 00:00:01 |

|* 43 |       INDEX RANGE SCAN             | I_FILE#_BLOCK#      |     1 |
 |     2   (0)| 00:00:01 |

|  44 |     TABLE ACCESS FULL              | TS$                 |     5 |    35
 |     3   (0)| 00:00:01 |

--------------------------------------------------------------------------------
-------------------------


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

   7 - access("O"."OWNER#"=USERENV('SCHEMAID') AND ("O"."NAME"='RANDOM_EMPNO_IDX
' OR

              "O"."NAME"='TEST_RANDOM'))
   8 - filter("O"."TYPE#"="SO"."OBJECT_TYPE_ID")
  10 - filter(BITAND("T"."PROPERTY",1024)=0)
  11 - access("T"."OBJ#"="O"."OBJ#")
  13 - access("TP"."OBJ#"="O"."OBJ#")
  15 - access("C"."OBJ#"="O"."OBJ#")
  16 - filter("I"."TYPE#"=1 OR "I"."TYPE#"=2 OR "I"."TYPE#"=3 OR "I"."TYPE#"=4 O
R "I"."TYPE#"=6

             OR "I"."TYPE#"=7 OR "I"."TYPE#"=8 OR "I"."TYPE#"=9)
  17 - access("I"."OBJ#"="O"."OBJ#")
  19 - access("IP"."OBJ#"="O"."OBJ#")
  20 - filter(BITAND("L"."PROPERTY",64)=0 OR BITAND("L"."PROPERTY",128)=128)
  21 - access("L"."LOBJ#"="O"."OBJ#")
  23 - access("TSP"."OBJ#"="O"."OBJ#")
  25 - access("ISP"."OBJ#"="O"."OBJ#")
  27 - access("LF"."FRAGOBJ#"="O"."OBJ#")
  28 - filter("S"."TYPE#"="SO"."SEGMENT_TYPE_ID")
  29 - access("S"."TS#"="SO"."TS_NUMBER" AND "S"."FILE#"="SO"."HEADER_FILE" AND
              "S"."BLOCK#"="SO"."HEADER_BLOCK")
  31 - access("S"."TS#"="TS"."TS#")
  34 - filter(("UN"."NAME"='RANDOM_EMPNO_IDX' OR "UN"."NAME"='TEST_RANDOM') AND
"UN"."STATUS$"<>1)

  35 - filter(("S"."TYPE#"=1 OR "S"."TYPE#"=10) AND "S"."USER#"=USERENV('SCHEMAI
D'))

  36 - access("S"."TS#"="UN"."TS#" AND "S"."FILE#"="UN"."FILE#" AND "S"."BLOCK#"
="UN"."BLOCK#")

  38 - access("S"."TS#"="TS"."TS#")
  39 - access("S"."TS#"="TS"."TS#")
  42 - filter("S"."TYPE#"<>1 AND "S"."TYPE#"<>5 AND "S"."TYPE#"<>6 AND "S"."TYPE
#"<>8 AND

             "S"."TYPE#"<>10 AND "S"."USER#"=USERENV('SCHEMAID'))
  43 - access("S"."TS#"="F"."TS#" AND "S"."FILE#"="F"."RELFILE#")
       filter(TO_CHAR("F"."FILE#")||'.'||TO_CHAR("S"."BLOCK#")='TEST_RANDOM' OR
              TO_CHAR("F"."FILE#")||'.'||TO_CHAR("S"."BLOCK#")='RANDOM_EMPNO_IDX
')



Statistics
----------------------------------------------------------
         22  recursive calls
          0  db block gets
         95  consistent gets
          0  physical reads
          0  redo size
        704  bytes sent via SQL*Net to client
        519  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          2  rows processed
```

Paso 3A (con TEST_NORMAL)

En este paso, crearemos el índice de mapa de bits (de manera similar a lo que hicimos en el paso 1A). Conocemos el tamaño del índice y el factor de agrupamiento del índice, que es igual a la cantidad de las filas de la tabla. Ahora ejecutaremos algunas consultas con predicados de rango.

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

   1 - filter("EMPNO"<=2300 AND "EMPNO">=1)


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
      12144  consistent gets
          3  physical reads
          0  redo size
     361732  bytes sent via SQL*Net to client
       5568  bytes received via SQL*Net from client
        461  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
       6900  rows processed
```

```
SQL> select * from test_normal where empno between &range1 and &range2;
Enter value for range1: 1
Enter value for range2: 2300
old   1: select * from test_normal where empno between &range1 and &range2
new   1: select * from test_normal where empno between 1 and 2300

6900 rows selected.


Execution Plan
----------------------------------------------------------
Plan hash value: 512490529

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

--------------------------------------------------------------------------------
-
|   0 | SELECT STATEMENT  |             |  6897 |   229K|  3195   (2)| 00:00:39
|
|*  1 |  TABLE ACCESS FULL| TEST_NORMAL |  6897 |   229K|  3195   (2)| 00:00:39
|

--------------------------------------------------------------------------------
-


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

   1 - filter("EMPNO"<=2300 AND "EMPNO">=1)
```

Paso 3B (con TEST_NORMAL)

En este paso, ejecutaremos las consultas en la tabla TEST_NORMAL con un índice de árbol B asociado a esa tabla.

```
SQL> select * from test_normal where empno between &range1 and &range2;
Enter value for range1: 1
Enter value for range2: 2300
old   1: select * from test_normal where empno between &range1 and &range2
new   1: select * from test_normal where empno between 1 and 2300

6900 rows selected.


Execution Plan
----------------------------------------------------------
Plan hash value: 512490529

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

--------------------------------------------------------------------------------
-
|   0 | SELECT STATEMENT  |             |  6897 |   229K|  3195   (2)| 00:00:39
|
|*  1 |  TABLE ACCESS FULL| TEST_NORMAL |  6897 |   229K|  3195   (2)| 00:00:39
|

--------------------------------------------------------------------------------
-
```

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

   1 - filter("EMPNO"<=2300 AND "EMPNO">=1)


Statistics
----------------------------------------------------------
          0  recursive calls
          0  db block gets
      12144  consistent gets
          0  physical reads
          0  redo size
     361732  bytes sent via SQL*Net to client
       5568  bytes received via SQL*Net from client
        461  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
       6900  rows processed
```

Paso 4A (con TEST_RANDOM)

En este paso, ejecutaremos las consultas con predicados de rango en la tabla TEST_RANDOM con el índice de mapa de bits y verificaremos la cantidad de lecturas coherentes y lecturas físicas. En este ejemplo se verá el impacto del factor de agrupamiento.

```
SQL> select  * from test_random where empno between &range1 and &range2;
Enter value for range1: 1
Enter value for range2: 2300
old   1: select  * from test_random where empno between &range1 and &range2
new   1: select  * from test_random where empno between 1 and 2300

6900 rows selected.


Execution Plan
----------------------------------------------------------
Plan hash value: 2650160170

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

--------------------------------------------------------------------------------
-
|   0 | SELECT STATEMENT  |             |  6897 |   229K|  3197   (2)| 00:00:39
|
|*  1 |  TABLE ACCESS FULL| TEST_RANDOM |  6897 |   229K|  3197   (2)| 00:00:39
|

--------------------------------------------------------------------------------
-


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

   1 - filter("EMPNO"<=2300 AND "EMPNO">=1)
```

```
Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
      12149  consistent gets
          0  physical reads
          0  redo size
     361732  bytes sent via SQL*Net to client
       5568  bytes received via SQL*Net from client
        461  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
       6900  rows processed
```

Paso 4B (con TEST_RANDOM)

En este paso, ejecutaremos las consultas con predicados de rango en la tabla TEST_RANDOM con un índice de árbol B asociado a esa tabla. Recuérdese que el factor de agrupamiento de este índice se aproximaba mucho a la cantidad de filas de la tabla (por lo que el índice resultaba ineficiente). A continuación, se incluye el procedimiento aplicado por el optimizador:

```
SQL> select * from test_random where empno between &range1 and &range2;
Enter value for range1: 1
Enter value for range2: 2300
old   1: select * from test_random where empno between &range1 and &range2
new   1: select * from test_random where empno between 1 and 2300

6900 rows selected.


Execution Plan
----------------------------------------------------------
Plan hash value: 2650160170

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

--------------------------------------------------------------------------------
-
|   0 | SELECT STATEMENT  |             |  6897 |   229K|  3197   (2)| 00:00:39
|
|*  1 |  TABLE ACCESS FULL| TEST_RANDOM |  6897 |   229K|  3197   (2)| 00:00:39
|

--------------------------------------------------------------------------------
-


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

   1 - filter("EMPNO"<=2300 AND "EMPNO">=1)
```

```
Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
      12149  consistent gets
          0  physical reads
          0  redo size
     361732  bytes sent via SQL*Net to client
       5568  bytes received via SQL*Net from client
        461  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
       6900  rows processed
```

Paso 5A (con TEST_NORMAL)

Crear un índice de mapa de bits asociado a la columna SAL [salario] de la tabla TEST_NORMAL. Esa columna tiene cardinalidad normal.

```
SQL> create bitmap index normal_sal_bmx on test_normal(sal);

Index created.
```

```
SQL> analyze table test_normal compute statistics for table for all indexes for all  indexed columns;

Table analyzed.
```

```
SQL> select  substr(segment_name,1,30) segment_name, bytes/1024/1024 "Size in MB"
  2  from user_segments
  3  where segment_name in  ('TEST_NORMAL','NORMAL_SAL_BMX');


Execution Plan
----------------------------------------------------------
Plan hash value: 2297303168

-------------------------------------------------------------------------------
--------------------------

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

-------------------------------------------------------------------------------
--------------------------

|   0 | SELECT STATEMENT                   |                    |     7 |  1113
 |    37   (3)| 00:00:01 |

|   1 |  VIEW                              | SYS_USER_SEGS      |     7 |  1113
 |    37   (3)| 00:00:01 |

|   2 |   UNION-ALL                        |                    |       |
 |            |          |

|   3 |    NESTED LOOPS                    |                    |     1 |   119
 |    15   (0)| 00:00:01 |

|   4 |     NESTED LOOPS                   |                    |     1 |   112
 |    14   (0)| 00:00:01 |

|   5 |      NESTED LOOPS                  |                    |     1 |    81
 |    13   (0)| 00:00:01 |

|   6 |       INLIST ITERATOR              |                    |       |
 |            |          |
```

```
SQL> set autot trace
```

```
SQL> select * from test_normal where sal=&sal;
Enter value for sal: 1869
old   1: select * from test_normal where sal=&sal
new   1: select * from test_normal where sal=1869

315 rows selected.


Execution Plan
----------------------------------------------------------
Plan hash value: 257953309

--------------------------------------------------------------------------------
--------------

| Id  | Operation                    | Name           | Rows  | Bytes | Cost (%C
PU)| Time     |

--------------------------------------------------------------------------------
--------------

|   0 | SELECT STATEMENT             |                |   313 |  7512 |    68
(0)| 00:00:01 |

|   1 |  TABLE ACCESS BY INDEX ROWID | TEST_NORMAL    |   313 |  7512 |    68
(0)| 00:00:01 |

|   2 |   BITMAP CONVERSION TO ROWIDS|                |       |       |
   |          |

|*  3 |    BITMAP INDEX SINGLE VALUE | NORMAL_SAL_BMX |       |       |
   |          |

--------------------------------------------------------------------------------
--------------
```

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

   3 - access("SAL"=1869)


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
        315  consistent gets
          0  physical reads
          0  redo size
      18613  bytes sent via SQL*Net to client
        739  bytes received via SQL*Net from client
         22  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
        315  rows processed
```

```
SQL> select * from test_normal where sal between &sal1 and &sal2;
Enter value for sal1: 1500
Enter value for sal2: 2000
old   1: select * from test_normal where sal between &sal1 and &sal2
new   1: select * from test_normal where sal between 1500 and 2000

156804 rows selected.


Execution Plan
----------------------------------------------------------
Plan hash value: 512490529

-------------------------------------------------------------------------------
-

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

-------------------------------------------------------------------------------
-

|   0 | SELECT STATEMENT  |             |   156K|  3664K|  3202   (2)| 00:00:39
|

|*  1 |  TABLE ACCESS FULL| TEST_NORMAL |   156K|  3664K|  3202   (2)| 00:00:39
|

-------------------------------------------------------------------------------
-


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

   1 - filter("SAL"<=2000 AND "SAL">=1500)
```

```
Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
      22067  consistent gets
          0  physical reads
          0  redo size
    8368002  bytes sent via SQL*Net to client
     115502  bytes received via SQL*Net from client
      10455  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
     156804  rows processed
```

```
SQL> analyze table test_normal compute statistics for table for all indexes for all  indexed columns;

Table analyzed.
```

```
SQL> select substr(segment_name,1,30) segment_name, bytes/1024/1024 "Size in  MB"
  2  from  user_segments
  3  where segment_name in ('TEST_NORMAL','NORMAL_SAL_IDX');


Execution Plan
----------------------------------------------------------
Plan hash value: 2297303168

-------------------------------------------------------------------------------
--------------------------

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

-------------------------------------------------------------------------------
--------------------------

|   0 | SELECT STATEMENT                   |                     |     7 |  1113
 |    37   (3)| 00:00:01 |

|   1 |  VIEW                              | SYS_USER_SEGS       |     7 |  1113
 |    37   (3)| 00:00:01 |

|   2 |   UNION-ALL                        |                     |       |
 |            |          |

|   3 |    NESTED LOOPS                    |                     |     1 |   119
 |    15   (0)| 00:00:01 |

|   4 |     NESTED LOOPS                   |                     |     1 |   112
 |    14   (0)| 00:00:01 |

|   5 |      NESTED LOOPS                  |                     |     1 |    81
 |    13   (0)| 00:00:01 |

|   6 |       INLIST ITERATOR              |                     |       |
 |            |          |

|*  7 |        INDEX RANGE SCAN            | I_OBJ2              |     1 |    31
 |     3   (0)| 00:00:01 |
```

```
|*  8 |       VIEW                          | SYS_OBJECTS         |     1 |    50
 |    10   (0)| 00:00:01 |

|   9 |        UNION ALL PUSHED PREDICATE   |                     |       |
 |            |          |

|* 10 |         TABLE ACCESS CLUSTER        | TAB$                |     1 |    22
 |     2   (0)| 00:00:01 |

|* 11 |          INDEX UNIQUE SCAN          | I_OBJ#              |     1 |
 |     1   (0)| 00:00:01 |

|  12 |         TABLE ACCESS BY INDEX ROWID| TABPART$            |     1 |    15
 |     1   (0)| 00:00:01 |

|* 13 |          INDEX UNIQUE SCAN          | I_TABPART_OBJ$      |     1 |
 |     0   (0)| 00:00:01 |

|  14 |         TABLE ACCESS CLUSTER        | CLU$                |     1 |    14
 |     2   (0)| 00:00:01 |

|* 15 |          INDEX UNIQUE SCAN          | I_OBJ#              |     1 |
 |     1   (0)| 00:00:01 |

|* 16 |         TABLE ACCESS BY INDEX ROWID| IND$                |     1 |    19
 |     2   (0)| 00:00:01 |

|* 17 |          INDEX UNIQUE SCAN          | I_IND1              |     1 |
 |     1   (0)| 00:00:01 |

|  18 |         TABLE ACCESS BY INDEX ROWID| INDPART$            |     1 |    15
 |     1   (0)| 00:00:01 |

|* 19 |          INDEX UNIQUE SCAN          | I_INDPART_OBJ$      |     1 |
 |     0   (0)| 00:00:01 |

|* 20 |         TABLE ACCESS BY INDEX ROWID| LOB$                |     1 |    20
 |     1   (0)| 00:00:01 |

|* 21 |          INDEX UNIQUE SCAN          | I_LOB2              |     1 |
 |     0   (0)| 00:00:01 |
```

```
|  22 |         TABLE ACCESS BY INDEX ROWID| TABSUBPART$         |     1 |    52
 |     0   (0)| 00:00:01 |

|* 23 |          INDEX UNIQUE SCAN         | I_TABSUBPART$_OBJ$  |     1 |
 |     0   (0)| 00:00:01 |

|  24 |         TABLE ACCESS BY INDEX ROWID| INDSUBPART$         |     1 |    52
 |     0   (0)| 00:00:01 |

|* 25 |          INDEX UNIQUE SCAN         | I_INDSUBPART_OBJ$   |     1 |
 |     0   (0)| 00:00:01 |

|  26 |         TABLE ACCESS BY INDEX ROWID| LOBFRAG$            |     1 |    17
 |     1   (0)| 00:00:01 |

|* 27 |          INDEX UNIQUE SCAN         | I_LOBFRAG$_FRAGOBJ$ |     1 |
 |     0   (0)| 00:00:01 |

|* 28 |       TABLE ACCESS CLUSTER         | SEG$                |     1 |    31
 |     1   (0)| 00:00:01 |

|* 29 |        INDEX UNIQUE SCAN           | I_FILE#_BLOCK#      |     1 |
 |     0   (0)| 00:00:01 |

|  30 |      TABLE ACCESS CLUSTER          | TS$                 |     1 |     7
 |     1   (0)| 00:00:01 |

|* 31 |       INDEX UNIQUE SCAN            | I_TS#               |     1 |
 |     0   (0)| 00:00:01 |

|  32 |     NESTED LOOPS                   |                     |     2 |   148
 |     6   (0)| 00:00:01 |

|  33 |      NESTED LOOPS                  |                     |     2 |   134
 |     4   (0)| 00:00:01 |

|* 34 |       TABLE ACCESS FULL            | UNDO$               |     2 |    66
 |     2   (0)| 00:00:01 |

|* 35 |       TABLE ACCESS CLUSTER         | SEG$                |     1 |    34
 |     1   (0)| 00:00:01 |
```

```
|* 36 |       INDEX UNIQUE SCAN           | I_FILE#_BLOCK#      |     1 |
 |     0   (0)| 00:00:01 |

|  37 |     TABLE ACCESS CLUSTER          | TS$                 |     1 |     7
 |     1   (0)| 00:00:01 |

|* 38 |      INDEX UNIQUE SCAN            | I_TS#               |     1 |
 |     0   (0)| 00:00:01 |

|* 39 |    HASH JOIN                      |                     |     4 |   200
 |    16   (7)| 00:00:01 |

|  40 |     NESTED LOOPS                  |                     |     4 |   172
 |    12   (0)| 00:00:01 |

|  41 |      TABLE ACCESS FULL            | FILE$               |     4 |    36
 |     2   (0)| 00:00:01 |

|* 42 |      TABLE ACCESS CLUSTER         | SEG$                |     1 |    34
 |     3   (0)| 00:00:01 |

|* 43 |       INDEX RANGE SCAN            | I_FILE#_BLOCK#      |     1 |
 |     2   (0)| 00:00:01 |

|  44 |     TABLE ACCESS FULL             | TS$                 |     5 |    35
 |     3   (0)| 00:00:01 |

--------------------------------------------------------------------------------
-------------------------


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

   7 - access("O"."OWNER#"=USERENV('SCHEMAID') AND ("O"."NAME"='NORMAL_SAL_IDX'
OR

              "O"."NAME"='TEST_NORMAL'))
   8 - filter("O"."TYPE#"="SO"."OBJECT_TYPE_ID")
  10 - filter(BITAND("T"."PROPERTY",1024)=0)
  11 - access("T"."OBJ#"="O"."OBJ#")
  13 - access("TP"."OBJ#"="O"."OBJ#")
```

```
  16 - filter("I"."TYPE#"=1 OR "I"."TYPE#"=2 OR "I"."TYPE#"=3 OR "I"."TYPE#"=4 O
R "I"."TYPE#"=6

              OR "I"."TYPE#"=7 OR "I"."TYPE#"=8 OR "I"."TYPE#"=9)
  17 - access("I"."OBJ#"="O"."OBJ#")
  19 - access("IP"."OBJ#"="O"."OBJ#")
  20 - filter(BITAND("L"."PROPERTY",64)=0 OR BITAND("L"."PROPERTY",128)=128)
  21 - access("L"."LOBJ#"="O"."OBJ#")
  23 - access("TSP"."OBJ#"="O"."OBJ#")
  25 - access("ISP"."OBJ#"="O"."OBJ#")
  27 - access("LF"."FRAGOBJ#"="O"."OBJ#")
  28 - filter("S"."TYPE#"="SO"."SEGMENT_TYPE_ID")
  29 - access("S"."TS#"="SO"."TS_NUMBER" AND "S"."FILE#"="SO"."HEADER_FILE" AND
              "S"."BLOCK#"="SO"."HEADER_BLOCK")
  31 - access("S"."TS#"="TS"."TS#")
  34 - filter(("UN"."NAME"='NORMAL_SAL_IDX' OR "UN"."NAME"='TEST_NORMAL') AND "U
N"."STATUS$"<>1)

  35 - filter(("S"."TYPE#"=1 OR "S"."TYPE#"=10) AND "S"."USER#"=USERENV('SCHEMAI
D'))

  36 - access("S"."TS#"="UN"."TS#" AND "S"."FILE#"="UN"."FILE#" AND "S"."BLOCK#"
="UN"."BLOCK#")

  38 - access("S"."TS#"="TS"."TS#")
  39 - access("S"."TS#"="TS"."TS#")
  42 - filter("S"."TYPE#"<>1 AND "S"."TYPE#"<>5 AND "S"."TYPE#"<>6 AND "S"."TYPE
#"<>8 AND

              "S"."TYPE#"<>10 AND "S"."USER#"=USERENV('SCHEMAID'))
  43 - access("S"."TS#"="F"."TS#" AND "S"."FILE#"="F"."RELFILE#")
       filter(TO_CHAR("F"."FILE#")||'.'||TO_CHAR("S"."BLOCK#")='TEST_NORMAL' OR
              TO_CHAR("F"."FILE#")||'.'||TO_CHAR("S"."BLOCK#")='NORMAL_SAL_IDX')


Statistics
----------------------------------------------------------
         22  recursive calls
          0  db block gets
         79  consistent gets
          0  physical reads
          0  redo size
        633  bytes sent via SQL*Net to client
        519  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          1  rows processed
```

```
SQL> select index_name, clustering_factor from user_indexes;

235 rows selected.


Execution Plan
----------------------------------------------------------
Plan hash value: 4210828137

-------------------------------------------------------------------------------
--------------

| Id  | Operation                      | Name    | Rows  | Bytes | Cost (%CP
U)| Time     |

-------------------------------------------------------------------------------
--------------

|   0 | SELECT STATEMENT               |         |  1356 |   131K|   288   (
2)| 00:00:04 |

|*  1 |  HASH JOIN RIGHT OUTER         |         |  1356 |   131K|   288   (
2)| 00:00:04 |

|   2 |   INDEX FULL SCAN              | I_USER2 |    33 |    99 |     1   (
0)| 00:00:01 |

|*  3 |   HASH JOIN OUTER              |         |  1356 |   127K|   287   (
2)| 00:00:04 |

|*  4 |    HASH JOIN                   |         |  1356 |   116K|   276   (
2)| 00:00:04 |

|   5 |     INDEX FULL SCAN            | I_USER2 |    33 |    99 |     1   (
0)| 00:00:01 |

|*  6 |     HASH JOIN                  |         |  1356 |   112K|   275   (
2)| 00:00:04 |

|*  7 |      HASH JOIN RIGHT OUTER     |         |  1356 |   101K|   264   (
1)| 00:00:04 |

|   8 |       TABLE ACCESS FULL        | SEG$    |  2517 | 27687 |    22   (
0)| 00:00:01 |
```

```
|*  9 |       HASH JOIN RIGHT OUTER        |         |  1356 | 89496 |   241   (
1)| 00:00:03 |

|  10 |        TABLE ACCESS FULL           | TS$     |     5 |    15 |     3   (
0)| 00:00:01 |

|  11 |        NESTED LOOPS                |         |       |       |
  |          |

|  12 |         NESTED LOOPS               |         |  1356 | 85428 |   238   (
1)| 00:00:03 |

|* 13 |          TABLE ACCESS FULL         | OBJ$    |  1831 | 54930 |    44   (
3)| 00:00:01 |

|* 14 |          INDEX UNIQUE SCAN         | I_IND1  |     1 |       |     0   (
0)| 00:00:01 |

|* 15 |         TABLE ACCESS BY INDEX ROWID| IND$    |     1 |    33 |     1   (
0)| 00:00:01 |

|  16 |      INDEX FAST FULL SCAN          | I_OBJ1  | 12815 |   100K|    10   (
0)| 00:00:01 |

|  17 |    INDEX FAST FULL SCAN            | I_OBJ1  | 12815 |   100K|    10   (
0)| 00:00:01 |

---------------------------------------------------------------------------
--------------


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

   1 - access("ITO"."OWNER#"="ITU"."USER#"(+))
   3 - access("I"."INDMETHOD#"="ITO"."OBJ#"(+))
   4 - access("IO"."OWNER#"="IU"."USER#")
   6 - access("I"."BO#"="IO"."OBJ#")
   7 - access("I"."FILE#"="S"."FILE#"(+) AND "I"."BLOCK#"="S"."BLOCK#"(+) AND
              "I"."TS#"="S"."TS#"(+))
   9 - access("I"."TS#"="TS"."TS#"(+))
  13 - filter("O"."OWNER#"=USERENV('SCHEMAID') AND BITAND("O"."FLAGS",128)=0)
  14 - access("O"."OBJ#"="I"."OBJ#")
  15 - filter(BITAND("I"."FLAGS",4096)=0 AND ("I"."TYPE#"=1 OR "I"."TYPE#"=2 OR
  14 - access("O"."OBJ#"="I"."OBJ#")
  15 - filter(BITAND("I"."FLAGS",4096)=0 AND ("I"."TYPE#"=1 OR "I"."TYPE#"=2 OR
              "I"."TYPE#"=3 OR "I"."TYPE#"=4 OR "I"."TYPE#"=6 OR "I"."TYPE#"=7 O
R "I"."TYPE#"=8 OR

              "I"."TYPE#"=9))


Statistics
----------------------------------------------------------
          8  recursive calls
          0  db block gets
       1614  consistent gets
        872  physical reads
          0  redo size
       9814  bytes sent via SQL*Net to client
        684  bytes received via SQL*Net from client
         17  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
        235  rows processed
```

```
SQL> set autot trace
```

```
SQL>  select * from test_normal where sal=&sal;
Enter value for sal: 1869
old   1:  select * from test_normal where sal=&sal
new   1:  select * from test_normal where sal=1869

315 rows selected.


Execution Plan
----------------------------------------------------------
Plan hash value: 257953309

--------------------------------------------------------------------------------
--------------

| Id  | Operation                    | Name           | Rows  | Bytes | Cost (%C
PU)| Time     |

--------------------------------------------------------------------------------
--------------

|   0 | SELECT STATEMENT             |                |   313 |  7512 |    68
(0)| 00:00:01 |

|   1 |  TABLE ACCESS BY INDEX ROWID | TEST_NORMAL    |   313 |  7512 |    68
(0)| 00:00:01 |

|   2 |   BITMAP CONVERSION TO ROWIDS|                |       |       |
   |          |

|*  3 |    BITMAP INDEX SINGLE VALUE | NORMAL_SAL_BMX |       |       |
   |          |

--------------------------------------------------------------------------------
--------------


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

   3 - access("SAL"=1869)


Statistics
----------------------------------------------------------
          1  recursive calls
          0  db block gets
        315  consistent gets
          0  physical reads
          0  redo size
      18613  bytes sent via SQL*Net to client
        739  bytes received via SQL*Net from client
         22  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
        315  rows processed
```

```
SQL> select * from test_normal where sal between &sal1 and &sal2;
Enter value for sal1: 1500
Enter value for sal2: 2000
old   1: select * from test_normal where sal between &sal1 and &sal2
new   1: select * from test_normal where sal between 1500 and 2000

156804 rows selected.


Execution Plan
----------------------------------------------------------
Plan hash value: 512490529

-------------------------------------------------------------------------------
-

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

-------------------------------------------------------------------------------
-

|   0 | SELECT STATEMENT  |             |   156K|  3664K|  3202   (2)| 00:00:39
|

|*  1 |  TABLE ACCESS FULL| TEST_NORMAL |   156K|  3664K|  3202   (2)| 00:00:39
|

-------------------------------------------------------------------------------
-


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

   1 - filter("SAL"<=2000 AND "SAL">=1500)


Statistics
----------------------------------------------------------
         1  recursive calls
         0  db block gets
     22067  consistent gets
         0  physical reads
         0  [illegible]
   8368002  bytes sent via SQL*Net to client
    115502  bytes received via SQL*Net from client
     10455  SQL*Net roundtrips to/from client
         0  sorts (memory)
         0  sorts (disk)
    156804  rows processed
```

Paso 6 (agregar una columna GENDER)

Antes de realizar la prueba con una columna de baja cardinalidad, agregaremos una columna GENDER a la tabla y la actualizaremos para cargarle los valores *M*, *F* y *null*.

```
SQL> alter table test_normal add GENDER varchar2(1);

Table altered.
```

```
SQL> declare
  2  mivariable number(1);
  3  begin
  4  for i in 1..1000000 loop
  5  select trunc(dbms_random.value(0,3)) into mivariable from dual;
  6  if mivariable = 1 then
  7  update test_normal set gender='H'where empno=1;
  8  elsif mivariable=0 then update test_normal set gender = 'M' where empno=1;
  9  ELSIF mivariable=2 then update test_normal set gender=NULL where empno=1;
 10  end if;
 11  end loop;
 12  end;
 13  /
```

```
SQL> declare
  2  mivariable number(1);
  3  begin
  4  for i in 1..1000000 loop
  5  select trunc(dbms_random.value(0,3)) into mivariable from dual;
  6  if mivariable = 1 then
  7  update test_normal set gender='H'where empno=1;
  8  elsif mivariable=0 then update test_normal set gender = 'M' where empno=1;
  9  ELSIF mivariable=2 then update test_normal set gender=NULL where empno=1;
 10  end if;
 11  end loop;
 12  end;
 13  /

PL/SQL procedure successfully completed.
```