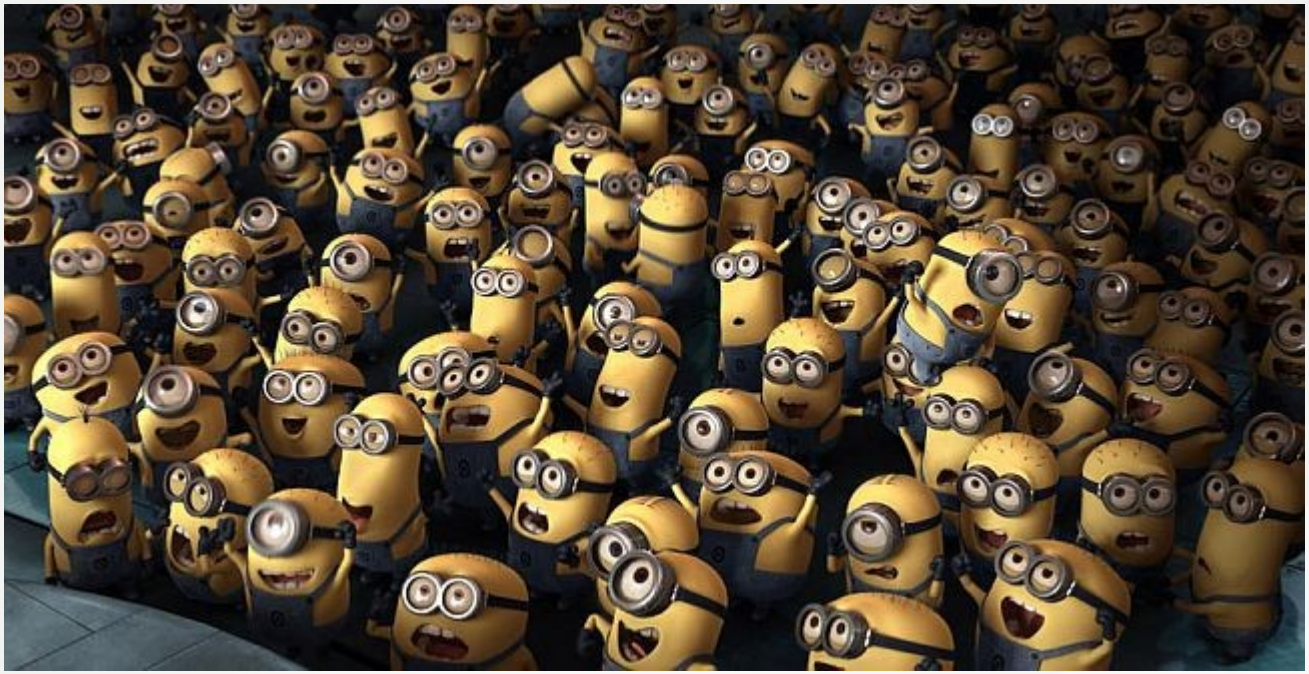
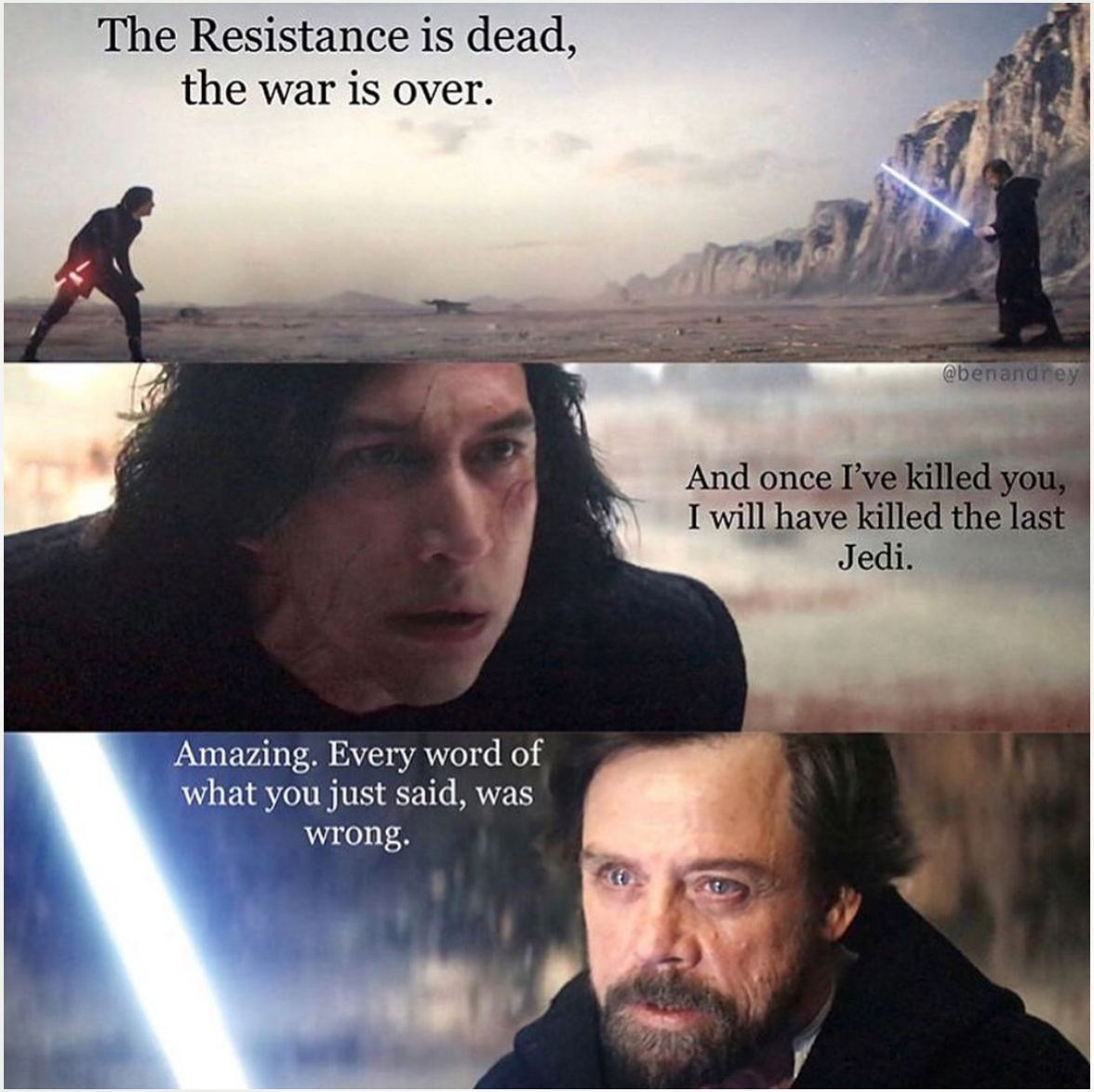
Lendas de Índices

Ricardo Portilho Proni
ricardo@nervinformatica.com.br

Não acredite em tudo o que lê... ou ouve...ou vê...



Não acredite em tudo o que lê... ou ouve...ou vê...



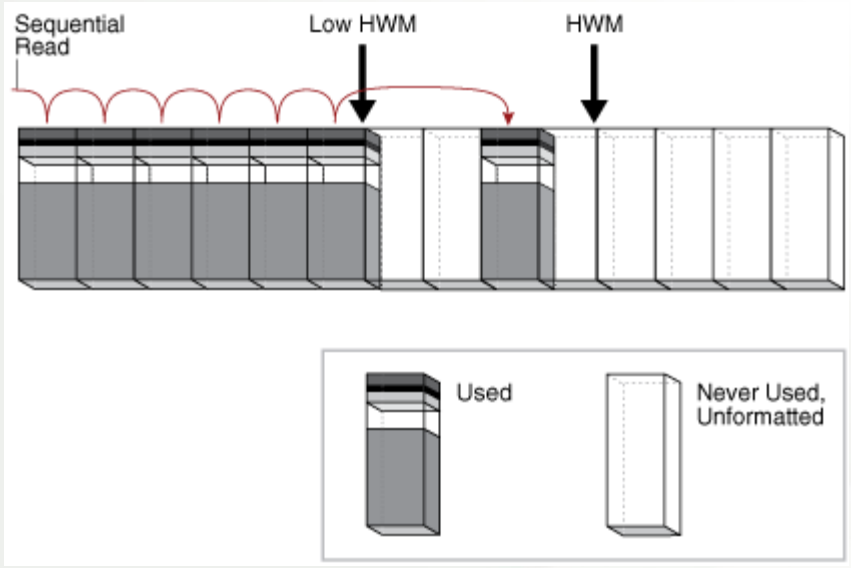
Lenda: Usar Índice é melhor que usar Full Table Scan

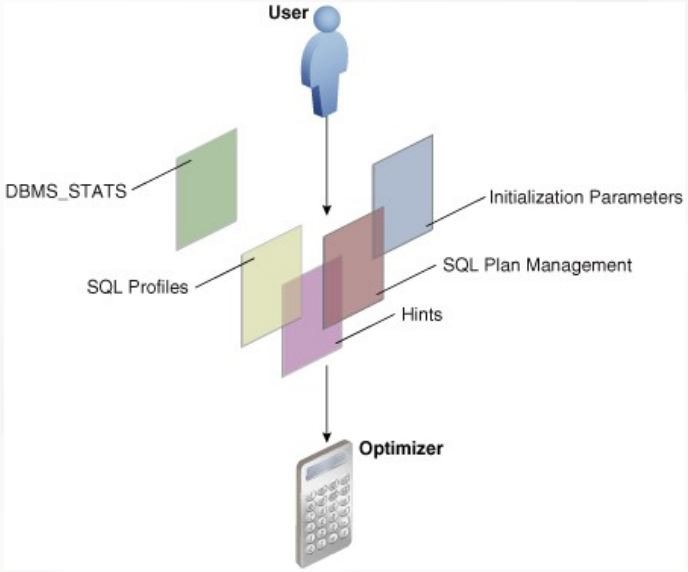
```
$ rlwrap sqlplus SCOTT/TIGER@PROD
SQL> CREATE TABLE T AS SELECT * FROM ALL_OBJECTS;
SQL> INSERT INTO T SELECT * FROM T;
SQL> INSERT INTO T SELECT * FROM T;
SQL> INSERT INTO T SELECT * FROM T;
SQL> INSERT INTO T SELECT * FROM T;
SQL> INSERT INTO T SELECT * FROM T;
SQL> INSERT INTO T SELECT * FROM T;
SQL> INSERT INTO T SELECT * FROM T;
SQL> INSERT INTO T SELECT * FROM T;
SQL> INSERT INTO T SELECT * FROM T;
SQL> COMMIT;

SQL> CREATE INDEX IDX_T_OBJECT_ID ON T(OBJECT_ID);

SQL> SET AUTOTRACE TRACEONLY EXPLAIN
SQL> SELECT OBJECT_NAME FROM T WHERE OBJECT_ID = 1000;
SQL> ALTER INDEX IDX_T_OBJECT_NAME INVISIBLE;
SQL> SELECT OBJECT_ID FROM T WHERE OBJECT_NAME = 'T';
SQL> SELECT /*+ INDEX(T IDX_T_OBJECT_ID)*/ OBJECT_ID FROM T WHERE
OBJECT_NAME = 'T';
SQL> ALTER INDEX IDX_T_OBJECT_NAME VISIBLE;

SQL> SET AUTOTRACE OFF
```

Lenda: SELECT de N% da tabela não vai usar Índice

Full Table Scan depende de:

- Tamanho da Tabela e Índice até o HWM (High Water Mark).
- Percentual de dados que serão acessados;
- Velocidade de leitura de múltiplos blocos x único bloco (System Statistics);
- Distribuição das linhas nos blocos (Clustering Factor);
- Parâmetros de Controle do CBO:
  - OPTIMIZER_MODE (FIRST_ROWS_n / ALL_ROWS)
  - DB_FILE_MULTIBLOCK_READ_COUNT
  - OPTIMIZER_INDEX_CACHING (0 a 100, padrão 0)
  - OPTIMIZER_INDEX_COST_ADJ (1 a 10000, padrão 100)





Lenda: SELECT de + de N% da tabela não vai usar Índice

Crie as duas tabelas abaixo com o usuário SCOTT, e compare as duas.

```
$ rlwrap sqlplus SCOTT/TIGER@PROD

SQL> CREATE TABLE T1 AS
    SELECT TRUNC((ROWNUM-1)/100) ID, RPAD(ROWNUM,100) NAME
    FROM DBA_SOURCE
    WHERE ROWNUM <= 10000;
SQL> CREATE INDEX T1_IDX1 ON T1(ID);

SQL> CREATE TABLE T2 AS
    SELECT MOD(ROWNUM,100) ID, RPAD(ROWNUM,100) NAME
    FROM DBA_SOURCE
    WHERE ROWNUM <= 10000;
SQL> CREATE INDEX T2_IDX1 ON T2(ID);

SQL> SELECT COUNT(*) FROM T1;
SQL> SELECT COUNT(*) FROM T2;
SQL> SELECT MIN(ID) FROM T1;
SQL> SELECT MIN(ID) FROM T2;
SQL> SELECT MAX(ID) FROM T1;
SQL> SELECT MAX(ID) FROM T2;
SQL> SELECT COUNT(*) FROM T1 WHERE ID = 1;
SQL> SELECT COUNT(*) FROM T2 WHERE ID = 1;
```

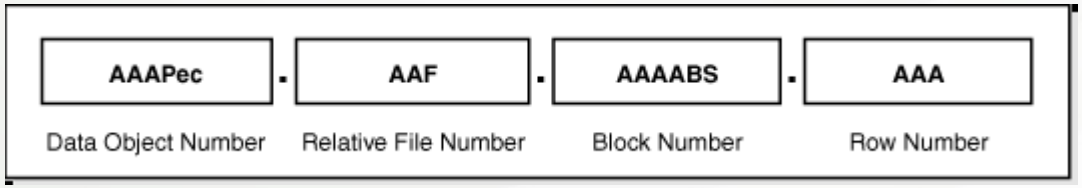
Lenda: SELECT de + de N% da tabela não vai usar Índice

Compare os planos de execução de SQL iguais para as duas tabelas.

```
SQL> SET AUTOTRACE TRACEONLY EXPLAIN
SQL> SELECT ID, NAME FROM T1 WHERE ID = 1;
SQL> SELECT ID, NAME FROM T2 WHERE ID = 1;
SQL> SELECT ID, NAME FROM T1 WHERE ID < 5;
SQL> SELECT ID, NAME FROM T2 WHERE ID < 5;
SQL> SELECT ID, NAME FROM T1 WHERE ID < 10;
SQL> SELECT ID, NAME FROM T2 WHERE ID < 10;
```

Verifique a ordenação física dos dados das tabelas.

```
SQL> SET AUTOTRACE OFF
SQL> SELECT ID, NAME FROM T1;
SQL> SELECT ID, NAME FROM T2;
SQL> SELECT ROWID, ID, NAME FROM T1 ORDER BY 2;
SQL> SELECT ROWID, ID, NAME FROM T2 ORDER BY 2;
```

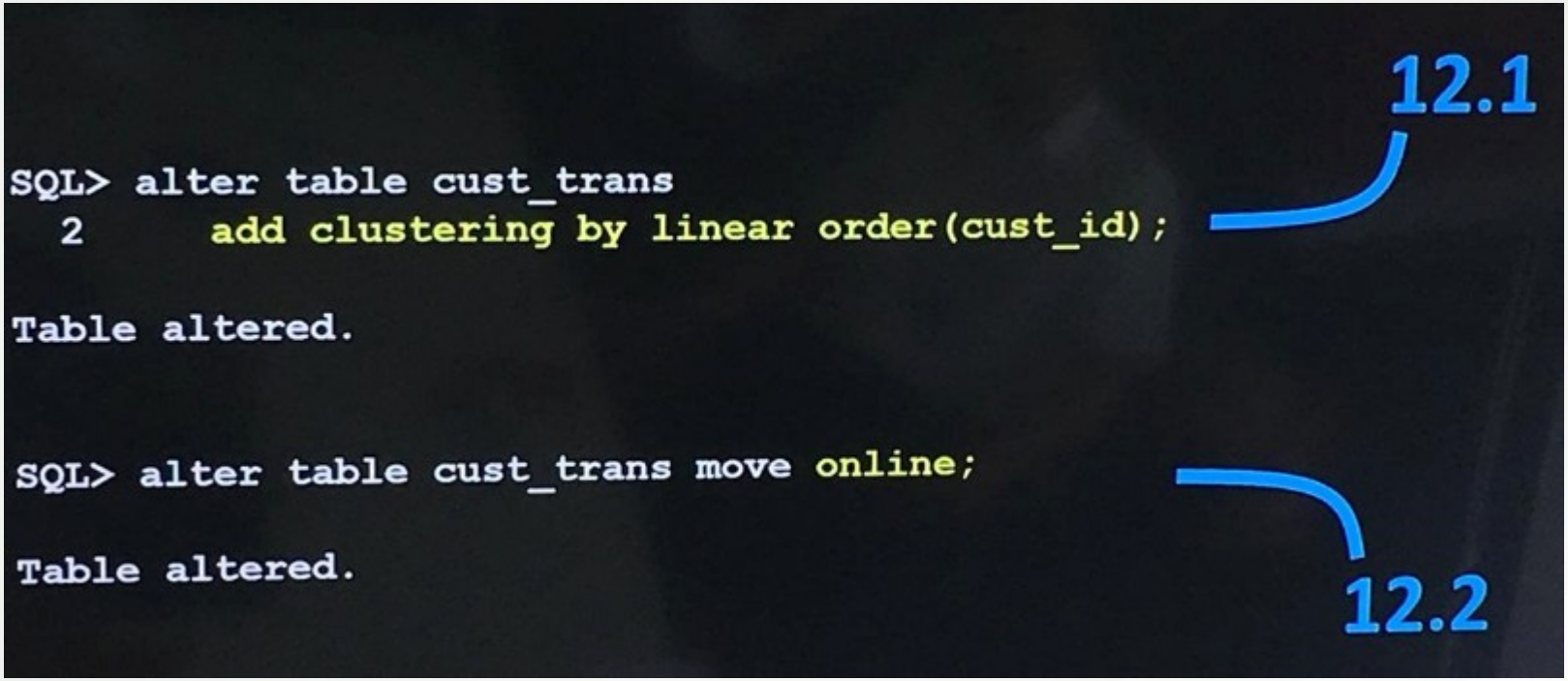


Lenda: SELECT de + de N% da tabela não vai usar Índice

Compare as estatísticas das duas tabelas.

```
SQL> SET AUTOTRACE OFF
SQL> COL TABLE_NAME FORMAT A20
SQL> COL INDEX_NAME FORMAT A30

SQL> SELECT
       T.TABLE_NAME,
       I.INDEX_NAME,
       I.CLUSTERING_FACTOR,
       T.BLOCKS,
       T.NUM_ROWS
   FROM DBA_TABLES T, DBA_INDEXES I
   WHERE   T.TABLE_NAME = I.TABLE_NAME AND
           T.OWNER = 'SCOTT'
           ORDER BY T.TABLE_NAME, I.INDEX_NAME;
```

Lenda: SELECT de + de N% da tabela não vai usar Índice



Lenda: Use Índices BITMAP em coluna de Estado / Sexo

```
$ rlwrap sqlplus SCOTT/TIGER@PROD
SQL> SELECT COUNT(*) FROM T;

  COUNT(*)
----------
  32997376

Elapsed: 00:00:02.15

SQL> SELECT COUNT(DISTINCT(OBJECT_TYPE)) FROM T;

COUNT(DISTINCT(OBJECT_TYPE))
----------------------------
                          27

Elapsed: 00:01:37.70

SQL> SELECT COUNT(DISTINCT(OBJECT_NAME)) FROM T;

COUNT(DISTINCT(OBJECT_NAME))
----------------------------
                       54129

Elapsed: 00:01:06.35
```

Lenda: Use Índices BITMAP em coluna de Estado / Sexo

```
SQL> SELECT COUNT(OBJECT_TYPE) FROM T;
Elapsed: 00:01:11.67

SQL> CREATE INDEX IDX_T_OBJECT_TYPE ON T(OBJECT_TYPE);
Elapsed: 00:10:21.97

SQL> SELECT COUNT(OBJECT_TYPE) FROM T;
Elapsed: 00:00:01.00

SQL> DROP INDEX IDX_T_OBJECT_TYPE;
SQL> CREATE BITMAP INDEX IDX_T_OBJECT_TYPE_BITMAP ON T(OBJECT_TYPE);
Elapsed: 00:01:14.65

SQL> SELECT COUNT(OBJECT_TYPE) FROM T;
Elapsed: 00:00:00.55

SQL> CREATE INDEX IDX_T_OBJECT_NAME ON T(OBJECT_NAME);
Elapsed: 00:55:18.06

SQL> SELECT COUNT(OBJECT_NAME) FROM T;
Elapsed: 00:00:00.03

SQL> DROP INDEX IDX_T_OBJECT_NAME;
SQL> CREATE BITMAP INDEX IDX_T_OBJECT_NAME_BITMAP ON T(OBJECT_NAME);
Elapsed: 00:09:48.71

SQL> SELECT COUNT(OBJECT_NAME) FROM T;
Elapsed: 00:00:00.03
```

(Porque não é) Lenda: Use Índices BITMAP em BI

1ª Sessão:

```
$ rlwrap sqlplus SCOTT/TIGER@PROD
SQL> CREATE TABLE T3 (C1 NUMBER);
SQL> CREATE BITMAP INDEX
IDX_T3_BITMAP ON T3(C1);

SQL> INSERT INTO T3 VALUES (1);

SQL> COMMIT;

SQL> INSERT INTO T3 VALUES (1);

SQL> COMMIT;

SQL> INSERT INTO T3 VALUES (1);

SQL> INSERT INTO T3 VALUES (10);
```

2ª Sessão:

```
$ rlwrap sqlplus SCOTT/TIGER@PROD

SQL> INSERT INTO T3 VALUES (10);

SQL> COMMIT;

SQL> INSERT INTO T3 VALUES (1);

COMMIT;

SQL> INSERT INTO T3 VALUES (10);

SQL> INSERT INTO T3 VALUES (1);
```

Lenda: SELECT COUNT(*) não usa Índice

Execute logon com o usuário SCOTT, e verifique qual é seu arquivo de TRACE:

```
$ rlwrap sqlplus SCOTT/TIGER@PROD
SQL> COLUMN TRACEFILE FORMAT A100
SQL> SELECT P.TRACEFILE FROM V$SESSION S, V$PROCESS P
    WHERE S.PADDR = P.ADDR AND S.USERNAME = 'SCOTT';
```

Coloque sua sessão em TRACE 10053, e execute os comandos abaixo.

```
SQL> ALTER SESSION SET EVENTS '10053 TRACE NAME CONTEXT FOREVER, LEVEL 1';
SQL> SELECT COUNT(EMPNO) FROM EMP;
SQL> SELECT COUNT(1) FROM EMP;
SQL> SELECT COUNT(2) FROM EMP;
SQL> SELECT COUNT(*) FROM EMP;
SQL> SELECT COUNT(ROWID) FROM EMP;
SQL> SELECT COUNT(MGR) FROM EMP;
```

Edite o seu arquivo de TRACE.

```
$ vi /u01/app/oracle/rdbms/ORCL/orcl/trace/ORCL_ora_12345.trc
```

Lenda: SELECT COUNT(*) não usa Índice

```
CNT:   Considering count(col) to count(*) on query block SEL$1 (#0)
**************************
Count(col) to Count(*) (CNT)
**************************
CNT:     Converting COUNT(EMPNO) to COUNT(*).
CNT:     COUNT() to COUNT(*) done.
```

```
Final query after transformations:******* UNPARSED QUERY IS *******
SELECT COUNT(*) "COUNT(EMPNO)" FROM "SCOTT"."EMP" "EMP"
kkoqbc: optimizing query block SEL$1 (#0)
```

```
Final query after transformations:******* UNPARSED QUERY IS *******
SELECT COUNT(*) "COUNT(1)" FROM "SCOTT"."EMP" "EMP"
kkoqbc: optimizing query block SEL$1 (#0)
```

```
Final query after transformations:******* UNPARSED QUERY IS *******
SELECT COUNT(*) "COUNT(2)" FROM "SCOTT"."EMP" "EMP"
kkoqbc: optimizing query block SEL$1 (#0)
```

```
Final query after transformations:******* UNPARSED QUERY IS *******
SELECT COUNT(*) "COUNT(*)" FROM "SCOTT"."EMP" "EMP"
kkoqbc: optimizing query block SEL$1 (#0)
```

Lenda: SELECT COUNT(*) não usa Índice

```
CNT:   Considering count(col) to count(*) on query block SEL$1 (#0)
**************************
Count(col) to Count(*) (CNT)
**************************
CNT:     COUNT() to COUNT(*) not done.
```

```
Final query after transformations:******* UNPARSED QUERY IS *******
SELECT COUNT("EMP".ROWID) "COUNT(ROWID)" FROM "SCOTT"."EMP" "EMP"
kkoqbc: optimizing query block SEL$1 (#0)
```

```
Final query after transformations:******* UNPARSED QUERY IS *******
SELECT COUNT("EMP"."MGR") "COUNT(MGR)" FROM "SCOTT"."EMP" "EMP"
kkoqbc: optimizing query block SEL$1 (#0)
```

Lenda: Tabelas pequenas não utilizam Índice

```
SQL> CREATE TABLE T4 (C1 NUMBER);

SQL> INSERT INTO T4 VALUES (1);
SQL> INSERT INTO T4 VALUES (2);
SQL> INSERT INTO T4 VALUES (3);
SQL> INSERT INTO T4 VALUES (4);
SQL> INSERT INTO T4 VALUES (5);
SQL> INSERT INTO T4 VALUES (6);
SQL> INSERT INTO T4 VALUES (7);
SQL> INSERT INTO T4 VALUES (8);
SQL> INSERT INTO T4 VALUES (9);
SQL> INSERT INTO T4 VALUES (10);
SQL> COMMIT;

SQL> SET AUTOTRACE TRACEONLY
SQL> SELECT C1 FROM T4 WHERE C1 = 1;

SQL> CREATE INDEX IDX_T4 ON T4(C1);
SQL> SELECT C1 FROM T4 WHERE C1 = 1;

SQL> SET AUTOTRACE OFF
```

Lenda: Cláusulas de negação não utilizam Índice

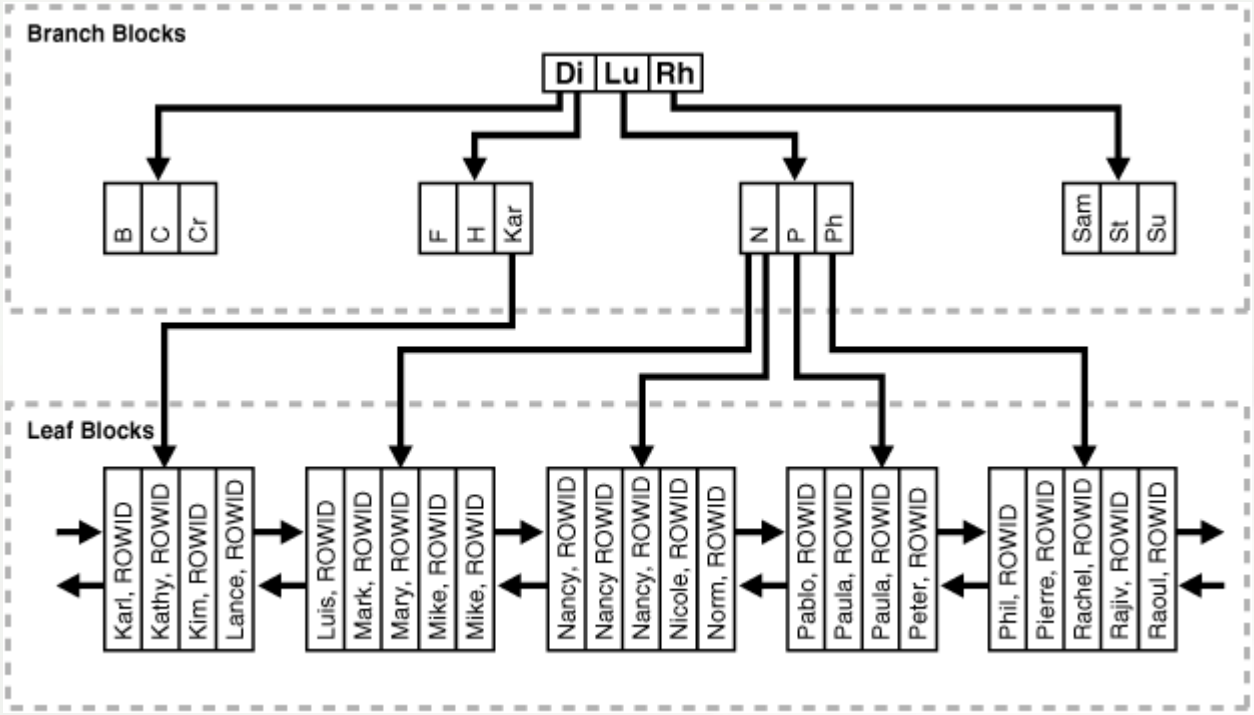
```
SQL> CREATE TABLE T5 AS SELECT * FROM ALL_OBJECTS;
SQL> CREATE INDEX IDX_T5 ON T5(OBJECT_TYPE);

SQL> SET AUTOTRACE TRACEONLY EXPLAIN
SQL> SELECT COUNT(OBJECT_TYPE) FROM T5;
SQL> SELECT COUNT(OBJECT_TYPE) FROM T5 WHERE OBJECT_TYPE = 'TABLE';
SQL> SELECT COUNT(OBJECT_TYPE) FROM T5 WHERE OBJECT_TYPE != 'TABLE';
SQL: SELECT COUNT(OBJECT_TYPE) FROM T5 WHERE OBJECT_TYPE NOT IN ('TABLE',
'INDEX');

SQL> SET AUTOTRACE OFF
```

Lenda: Busca por NULL não utiliza Índice

```
SQL> UPDATE T5 SET OBJECT_TYPE = NULL WHERE OBJECT_TYPE = 'SYNONYM';
SQL> COMMIT;

SQL> SET AUTOTRACE TRACEONLY EXPLAIN
SQL> SELECT COUNT(OBJECT_TYPE) FROM T5 WHERE OBJECT_TYPE IS NULL;
SQL> SELECT OBJECT_ID FROM T5 WHERE OBJECT_TYPE IS NULL;

SQL> DROP INDEX IDX_T5;
SQL> CREATE BITMAP INDEX IDX_T5 ON T5(OBJECT_TYPE);
SQL> SELECT COUNT(OBJECT_TYPE) FROM T5 WHERE OBJECT_TYPE IS NULL;
SQL> SELECT OBJECT_ID FROM T5 WHERE OBJECT_TYPE IS NULL;

SQL> SET AUTOTRACE OFF
```

(mais ou menos) Lenda: Índices em TABLESPACEs de 32k

- B-tree = Árvore Balanceada
- Root Block / Branch Blocks / Leaf Blocks
- Height / BEVEL (quando o Height / BLEVEL aumenta?)



Lenda: TABLESPACE de Índices em NOLOGGING

DML:

- Direct-path `INSERT` (serial or parallel) resulting either from an `INSERT` or a `MERGE` statement. `NOLOGGING` is not applicable to any `UPDATE` operations resulting from the `MERGE` statement.
- Direct Loader (SQL*Loader)

DDL:

- `CREATE TABLE ... AS SELECT` (In `NOLOGGING` mode, the creation of the table will be logged, but direct-path inserts will not be logged.)

  `CREATE TABLE ... LOB_storage_clause ... LOB_parameters ... CACHE | NOCACHE | CACHE READS`
- `ALTER TABLE ... LOB_storage_clause ... LOB_parameters ... CACHE | NOCACHE | CACHE READS` (to specify logging of newly created LOB columns)
- `ALTER TABLE ... modify_LOB_storage_clause ... modify_LOB_parameters ... CACHE | NOCACHE | CACHE READS` (to change logging of existing LOB columns)
- `ALTER TABLE ... MOVE`
- `ALTER TABLE ...` (all partition operations that involve data movement)
  - `ALTER TABLE ... ADD PARTITION` (hash partition only)
  - `ALTER TABLE ... MERGE PARTITIONS`
  - `ALTER TABLE ... SPLIT PARTITION`

(+ ou -) Lenda: Desfragmente seus Índices regularmente.

Tipo de Fragmentação

- Blocos logicamente contíguos espalhados fisicamente.
- Espaço livre na TABLESPACE / DATAFILEs.
- Espaço livre da TABELA / Espaço livre no ÍNDICE.
- Row Chaining / Migrated Rows.
- EXTENTs.

Como detectar:

```
$ rlwrap sqlplus / AS SYSDBA
SQL> ALTER SESSION SET CONTAINER = PROD;
SQL> EXEC DBMS_STATS.GATHER_SCHEMA_STATS('SCOTT');
SQL> @OracleBaseAdvisor.sql TABLESPACE USERS NULL
SQL> @OracleBaseAdvisor.sql TABLE SCOTT T5
SQL> @OracleBaseAdvisor.sql INDEX SCOTT IDX_T5
```

Como corrigir:

```
SQL> ALTER TABLE SCOTT.T5 MOVE;
SQL> ALTER INDEX SCOTT.IDX_T5 REBUILD;
SQL> EXEC DBMS_STATS.GATHER_SCHEMA_STATS('SCOTT');
SQL> @OracleBaseAdvisor.sql TABLE SCOTT T5
SQL> @OracleBaseAdvisor.sql INDEX SCOTT IDX_T5
```

Script: https://oracle-base.com/dba/script?category=10g&file=segment_advisor.sql

Lenda: O Oracle só tem 2 Índices: BTREE e BITMAP

- B-tree
- Bitmap
- IOT (Index-Organized Table)
- Bitmap Join
- Function-Based
- Bitmap Function-Based
- Text Index (CONTEXT, CTXCAT ou CTXRULE)
- R-Tree Index (Spatial Data)

Variações

- Partitioned Indexes
- Partial Indexes
- Ascending
- Descending
- Reverse Key
- Compressed
- Invisible Indexes
- Virtual Indexes

IOT – Index Organized Table

```
$ rlwrap sqlplus SCOTT/TIGER@PROD

SQL> CREATE TABLE SMALL (ID NUMBER PRIMARY KEY, NAME VARCHAR2(10));
SQL> INSERT INTO SMALL SELECT ROWNUM, 'BOWIE' FROM DUAL CONNECT BY LEVEL <=100;
SQL> COMMIT;
SQL> SET AUTOTRACE TRACEONLY
SQL> SELECT * FROM SMALL WHERE ID = 42;
SQL> SELECT * FROM SMALL WHERE ID = 42;
SQL> SELECT * FROM SMALL WHERE ID = 42;
SQL> SET AUTOTRACE OFF

SQL> DROP TABLE SMALL;
SQL> CREATE TABLE SMALL (ID NUMBER PRIMARY KEY, NAME VARCHAR2(10)) ORGANIZATION
INDEX;
SQL> INSERT INTO SMALL SELECT ROWNUM, 'BOWIE' FROM DUAL CONNECT BY LEVEL <=100;
SQL> COMMIT;
SQL> SET AUTOTRACE TRACEONLY
SQL> SELECT * FROM SMALL WHERE ID = 42;
SQL> SELECT * FROM SMALL WHERE ID = 42;
SQL> SELECT * FROM SMALL WHERE ID = 42;
SQL> SET AUTOTRACE OFF
```

Bitmap Join Index

```
$ rlwrap sqlplus SCOTT/TIGER@PROD

SQL> CREATE TABLE BIG_DWH_TABLE (ID NUMBER PRIMARY KEY, ALBUM_ID NUMBER, ARTIST_ID
NUMBER, COUNTRY_ID NUMBER, FORMAT_ID NUMBER, RELEASE_DATE DATE, TOTAL_SALES NUMBER);
SQL> CREATE SEQUENCE DWH_SEQ;

SQL> CREATE OR REPLACE PROCEDURE POP_BIG_DWH_TABLE AS
    V_ID          NUMBER;
    V_ARTIST_ID   NUMBER;
    BEGIN
        FOR V_ALBUM_ID IN 1..10000 LOOP
        V_ARTIST_ID:= CEIL(DBMS_RANDOM.VALUE(0,100));
            FOR V_COUNTRY_ID IN 1..100 LOOP
            SELECT DWH_SEQ.NEXTVAL INTO V_ID FROM DUAL;
            INSERT INTO BIG_DWH_TABLE VALUES
            (V_ID, V_ALBUM_ID, V_ARTIST_ID, V_COUNTRY_ID,
            CEIL(DBMS_RANDOM.VALUE(0,4)),
            TRUNC(SYSDATE-MOD(V_ID,CEIL(DBMS_RANDOM.VALUE(0,1000)))),
            CEIL(DBMS_RANDOM.VALUE(0,500000)));
            END LOOP;
        END LOOP;
COMMIT;
END;
/
```

Bitmap Join Index

```
SQL> EXEC POP_BIG_DWH_TABLE;
SQL> CREATE BITMAP INDEX BIG_DWH_TABLE_ALBUM_ID_I ON BIG_DWH_TABLE(ALBUM_ID);

SQL> CREATE TABLE ALBUMS (ALBUM_ID NUMBER, ALBUM_DETAILS VARCHAR2(30));
SQL> INSERT INTO ALBUMS SELECT ROWNUM, SUBSTR(OBJECT_NAME,1,30) FROM DBA_OBJECTS
WHERE ROWNUM <= 10000;
SQL> COMMIT;
SQL> ALTER TABLE ALBUMS ADD PRIMARY KEY(ALBUM_ID);
SQL> CREATE INDEX ALBUMS_DETAILS_I ON ALBUMS(ALBUM_DETAILS);



SQL> SET AUTOTRACE TRACEONLY EXPLAIN
SQL> SELECT B.ID, B.ALBUM_ID, B.FORMAT_ID FROM BIG_DWH_TABLE B, ALBUMS A WHERE
B.ALBUM_ID = A.ALBUM_ID AND A.ALBUM_DETAILS = 'TAB$';

SQL> DROP INDEX ALBUMS_DETAILS_I;
SQL> CREATE BITMAP INDEX BIG_DWH_ALBUM_DETAILS_I ON BIG_DWH_TABLE(A.ALBUM_DETAILS)
FROM BIG_DWH_TABLE B, ALBUMS A WHERE B.ALBUM_ID = A.ALBUM_ID;

SQL> SELECT B.ID, B.ALBUM_ID, B.FORMAT_ID FROM BIG_DWH_TABLE B, ALBUMS A WHERE
B.ALBUM_ID = A.ALBUM_ID AND A.ALBUM_DETAILS = 'TAB$';

SQL> SET AUTOTRACE OFF
```

Function-Based Index

```
$ rlwrap sqlplus SCOTT/TIGER@PROD

SQL> CREATE TABLE USER_DATA (ID NUMBER(10) NOT NULL, FIRST_NAME
VARCHAR2(40) NOT NULL, LAST_NAME VARCHAR2(40) NOT NULL, GENDER
VARCHAR2(1), DOB DATE);

SQL> BEGIN
 FOR CUR_REC IN 1 .. 2000 LOOP
 IF MOD(CUR_REC, 2) = 0 THEN
 INSERT INTO USER_DATA
 VALUES (CUR_REC, 'John' || CUR_REC, 'Doe', 'M', SYSDATE);
 ELSE
 INSERT INTO USER_DATA
 VALUES (CUR_REC, 'Jayne' || CUR_REC, 'Doe', 'F', SYSDATE);
 END IF;
 COMMIT;
 END LOOP;
END;
/
```

Function-Based Index

```
SQL> CREATE INDEX FIRST_NAME_IDX ON USER_DATA (FIRST_NAME);
SQL> SET AUTOTRACE ON
SQL> SELECT * FROM USER_DATA WHERE UPPER(FIRST_NAME) = 'JOHN2';
SQL> SELECT * FROM USER_DATA WHERE UPPER(FIRST_NAME) = 'JOHN2';
SQL> SELECT * FROM USER_DATA WHERE UPPER(FIRST_NAME) = 'JOHN2';
SQL> SET AUTOTRACE OFF

SQL> DROP INDEX FIRST_NAME_IDX;
SQL> CREATE INDEX FIRST_NAME_IDX ON USER_DATA (UPPER(FIRST_NAME));
SQL> SET AUTOTRACE ON
SQL> SELECT * FROM USER_DATA WHERE UPPER(FIRST_NAME) = 'JOHN2';
SQL> SELECT * FROM USER_DATA WHERE UPPER(FIRST_NAME) = 'JOHN2';
SQL> SELECT * FROM USER_DATA WHERE UPPER(FIRST_NAME) = 'JOHN2';
SQL> SET AUTOTRACE OFF
```

Function-Based Index

```
$ rlwrap sqlplus HR/HR@PROD

SQL> SET AUTOTRACE ON
SQL> SELECT EMPLOYEE_ID, LAST_NAME, FIRST_NAME, 12 * SALARY * COMMISSION_PCT AS
"ANNUAL SAL" FROM EMPLOYEES WHERE (12 * SALARY * COMMISSION_PCT) < 30000 ORDER BY
"ANNUAL SAL" DESC;
SQL> SELECT EMPLOYEE_ID, LAST_NAME, FIRST_NAME, 12 * SALARY * COMMISSION_PCT AS
"ANNUAL SAL" FROM EMPLOYEES WHERE (12 * SALARY * COMMISSION_PCT) < 30000 ORDER BY
"ANNUAL SAL" DESC;
SQL> SELECT EMPLOYEE_ID, LAST_NAME, FIRST_NAME, 12 * SALARY * COMMISSION_PCT AS
"ANNUAL SAL" FROM EMPLOYEES WHERE (12 * SALARY * COMMISSION_PCT) < 30000 ORDER BY
"ANNUAL SAL" DESC;

SQL> CREATE INDEX EMP_TOTAL_SAL_IDX ON EMPLOYEES (12 * SALARY * COMMISSION_PCT,
SALARY, COMMISSION_PCT);
SQL> SELECT EMPLOYEE_ID, LAST_NAME, FIRST_NAME, 12 * SALARY * COMMISSION_PCT AS
"ANNUAL SAL" FROM EMPLOYEES WHERE (12 * SALARY * COMMISSION_PCT) < 30000 ORDER BY
"ANNUAL SAL" DESC;
SQL> SELECT EMPLOYEE_ID, LAST_NAME, FIRST_NAME, 12 * SALARY * COMMISSION_PCT AS
"ANNUAL SAL" FROM EMPLOYEES WHERE (12 * SALARY * COMMISSION_PCT) < 30000 ORDER BY
"ANNUAL SAL" DESC;
SQL> SELECT EMPLOYEE_ID, LAST_NAME, FIRST_NAME, 12 * SALARY * COMMISSION_PCT AS
"ANNUAL SAL" FROM EMPLOYEES WHERE (12 * SALARY * COMMISSION_PCT) < 30000 ORDER BY
"ANNUAL SAL" DESC;
```

Function-Based Index

```
$ rlwrap sqlplus SCOTT/TIGER@PROD

SQL> CREATE TABLE TX AS SELECT * FROM ALL_OBJECTS;
SQL> CREATE INDEX IDX_T_CREATED ON TX(CREATED);

SQL> SET AUTOTRACE TRACEONLY EXPLAIN
SQL> ALTER SESSION SET NLS_DATE_FORMAT = 'DDMMYYYY';
SQL> SELECT OBJECT_ID FROM TX WHERE CREATED = '18092018';
SQL> SELECT OBJECT_ID FROM TX WHERE TO_CHAR(CREATED,'DDMMYYYY') = '18092018';
SQL> SELECT OBJECT_ID FROM TX WHERE TRUNC(CREATED,'DDMMYYYY') = '18092018';

SQL> CREATE INDEX IDX_T_CREATED_TO_CHAR ON TX(TO_CHAR(CREATED,'DDMMYYYY'));
SQL> SELECT OBJECT_ID FROM TX WHERE TO_CHAR(CREATED,'DDMMYYYY') = '18092018';

SQL> CREATE INDEX IDX_T_CREATED_TRUNC ON TX(TRUNC(CREATED));
SQL> SELECT OBJECT_ID FROM TX WHERE TRUNC(CREATED) = '18092018';
```

Bitmap Function-Based Index

```
$ rlwrap sqlplus SCOTT/TIGER@PROD

SQL> DROP INDEX FIRST_NAME_IDX;
SQL> CREATE BITMAP INDEX FIRST_NAME_IDX ON USER_DATA (UPPER(FIRST_NAME));
SQL> SET AUTOTRACE ON
SQL> SELECT * FROM USER_DATA WHERE UPPER(FIRST_NAME) = 'JOHN2';
SQL> SELECT * FROM USER_DATA WHERE UPPER(FIRST_NAME) = 'JOHN2';
SQL> SELECT * FROM USER_DATA WHERE UPPER(FIRST_NAME) = 'JOHN2';
SQL> SET AUTOTRACE OFF
```

Text Index

CONTEXT	Use this index to build a text retrieval application when your text consists of large, coherent documents in, for example, MS Word, HTML, or plain text. You can customize the index in a variety of ways. This index type requires CTX_DDL.SYNC_INDEX after insert, update, and delete operations to the base table.
CTXCAT	Use this index for better mixed query performance of small documents and text fragments. To improve mixed query performance, include other columns in the base table, such as item names, prices, and descriptions. This index type is transactional. It automatically updates itself after inserts, updates, or deletes to the base table. CTX_DDL.SYNC_INDEX is not necessary.
CTXRULE	Use this index to build a document classification or routing application. Create this index on a table of queries, where the queries define the classification or routing criteria..

```
SQL> CREATE INDEX MY_DOCS_DOC_IDX ON MY_DOCS(DOC) INDEXTYPE IS CTXSYS.CONTEXT;
SQL> SELECT SCORE(1) SCORE, ID, NAME FROM MY_DOCS WHERE CONTAINS(DOC, 'SQL Server',
1) > 0;
```

Reverse Key

```
$ rlwrap sqlplus SCOTT/TIGER@PROD

SQL> CREATE TABLE T8 (C1 NUMBER);
SQL> CREATE INDEX IDX_T8 ON T8(C1);
SQL> CREATE SEQUENCE S1 START WITH 1 INCREMENT BY 1 CACHE 1000;

$ sqlplus SCOTT/TIGER@PROD @ReverseKeyIndex.sql

$ sqlplus -s SCOTT/TIGER@PROD @ReverseKeyIndex.sql &
$ sqlplus -s SCOTT/TIGER@PROD @ReverseKeyIndex.sql &
$ sqlplus -s SCOTT/TIGER@PROD @ReverseKeyIndex.sql &
$ sqlplus -s SCOTT/TIGER@PROD @ReverseKeyIndex.sql &
$ sqlplus -s SCOTT/TIGER@PROD @ReverseKeyIndex.sql &
$ sqlplus -s SCOTT/TIGER@PROD @ReverseKeyIndex.sql &
$ sqlplus -s SCOTT/TIGER@PROD @ReverseKeyIndex.sql &
$ sqlplus -s SCOTT/TIGER@PROD @ReverseKeyIndex.sql &
$ sqlplus -s SCOTT/TIGER@PROD @ReverseKeyIndex.sql &
$ sqlplus -s SCOTT/TIGER@PROD @ReverseKeyIndex.sql &
```

Reverse Key

```
SQL> TRUNCATE TABLE T8;
SQL> DROP INDEX IDX_T8;
SQL> CREATE INDEX IDX_T8 ON T8(C1) REVERSE;
SQL> DROP SEQUENCE S1;
SQL> CREATE SEQUENCE S1 START WITH 1 INCREMENT BY 1 CACHE 1000;

$ sqlplus SCOTT/TIGER@PROD @ReverseKeyIndex.sql

$ sqlplus -s SCOTT/TIGER@PROD @ReverseKeyIndex.sql &
$ sqlplus -s SCOTT/TIGER@PROD @ReverseKeyIndex.sql &
$ sqlplus -s SCOTT/TIGER@PROD @ReverseKeyIndex.sql &
$ sqlplus -s SCOTT/TIGER@PROD @ReverseKeyIndex.sql &
$ sqlplus -s SCOTT/TIGER@PROD @ReverseKeyIndex.sql &
$ sqlplus -s SCOTT/TIGER@PROD @ReverseKeyIndex.sql &
$ sqlplus -s SCOTT/TIGER@PROD @ReverseKeyIndex.sql &
$ sqlplus -s SCOTT/TIGER@PROD @ReverseKeyIndex.sql &
$ sqlplus -s SCOTT/TIGER@PROD @ReverseKeyIndex.sql &
$ sqlplus -s SCOTT/TIGER@PROD @ReverseKeyIndex.sql &
```

Compressed Index

```
$ rlwrap sqlplus SCOTT/TIGER@PROD

SQL> CREATE TABLE T9 AS SELECT * FROM ALL_OBJECTS;
SQL> CREATE INDEX IDX_T9 ON T9(OBJECT_ID);
SQL> SET AUTOTRACE ON
SQL> SELECT COUNT(OBJECT_ID) FROM T9;
SQL> SELECT COUNT(OBJECT_ID) FROM T9;
SQL> SELECT COUNT(OBJECT_ID) FROM T9;

SQL> DROP INDEX IDX_T9;
SQL> CREATE INDEX IDX_T9 ON T9(OBJECT_ID) COMPRESS;
SQL> SELECT COUNT(OBJECT_ID) FROM T9;
SQL> SELECT COUNT(OBJECT_ID) FROM T9;
SQL> SELECT COUNT(OBJECT_ID) FROM T9;

SQL> DROP INDEX IDX_T9;
SQL> CREATE INDEX IDX_T9 ON T9(OBJECT_ID) COMPRESS ADVANCED LOW;
SQL> SELECT COUNT(OBJECT_ID) FROM T9;
SQL> SELECT COUNT(OBJECT_ID) FROM T9;
SQL> SELECT COUNT(OBJECT_ID) FROM T9;

SQL> DROP INDEX IDX_T9;
SQL> CREATE INDEX IDX_T9 ON T9(OBJECT_ID) COMPRESS ADVANCED HIGH;
SQL> SELECT COUNT(OBJECT_ID) FROM T9;
SQL> SELECT COUNT(OBJECT_ID) FROM T9;
SQL> SELECT COUNT(OBJECT_ID) FROM T9;
```

Compressed Index

```
SQL> DROP INDEX IDX_T9;
SQL> CREATE INDEX IDX_T9 ON T9(OWNER, OBJECT_NAME);
SQL> SELECT COUNT(OBJECT_NAME) FROM T9 WHERE OWNER = 'SYS';
SQL> SELECT COUNT(OBJECT_NAME) FROM T9 WHERE OWNER = 'SYS';
SQL> SELECT COUNT(OBJECT_NAME) FROM T9 WHERE OWNER = 'SYS';

SQL> DROP INDEX IDX_T9;
SQL> CREATE INDEX IDX_T9 ON T9(OWNER, OBJECT_NAME) COMPRESS;
SQL> SELECT COUNT(OBJECT_NAME) FROM T9 WHERE OWNER = 'SYS';
SQL> SELECT COUNT(OBJECT_NAME) FROM T9 WHERE OWNER = 'SYS';
SQL> SELECT COUNT(OBJECT_NAME) FROM T9 WHERE OWNER = 'SYS';

SQL> DROP INDEX IDX_T9;
SQL> CREATE INDEX IDX_T9 ON T9(OWNER, OBJECT_NAME) COMPRESS 1;
SQL> SELECT COUNT(OBJECT_NAME) FROM T9 WHERE OWNER = 'SYS';
SQL> SELECT COUNT(OBJECT_NAME) FROM T9 WHERE OWNER = 'SYS';
SQL> SELECT COUNT(OBJECT_NAME) FROM T9 WHERE OWNER = 'SYS';

SQL> DROP INDEX IDX_T9;
SQL> CREATE INDEX IDX_T9 ON T9(OWNER, OBJECT_NAME) COMPRESS ADVANCED LOW;
SQL> SELECT COUNT(OBJECT_NAME) FROM T9 WHERE OWNER = 'SYS';
SQL> SELECT COUNT(OBJECT_NAME) FROM T9 WHERE OWNER = 'SYS';
SQL> SELECT COUNT(OBJECT_NAME) FROM T9 WHERE OWNER = 'SYS';

SQL> DROP INDEX IDX_T9;
SQL> CREATE INDEX IDX_T9 ON T9(OWNER, OBJECT_NAME) COMPRESS ADVANCED HIGH;
SQL> SELECT COUNT(OBJECT_NAME) FROM T9 WHERE OWNER = 'SYS';
SQL> SELECT COUNT(OBJECT_NAME) FROM T9 WHERE OWNER = 'SYS';
SQL> SELECT COUNT(OBJECT_NAME) FROM T9 WHERE OWNER = 'SYS';
```

Invisible Index

```
$ rlwrap sqlplus SCOTT/TIGER@PROD

SQL> CREATE TABLE T10 AS SELECT * FROM ALL_OBJECTS;
SQL> CREATE INDEX IDX_T10 ON T10(OWNER);

SQL> SET AUTOTRACE TRACEONLY
SQL> SELECT COUNT(OBJECT_NAME) FROM T10 WHERE OWNER = 'SYS';

SQL> ALTER INDEX IDX_T10 INVISIBLE;
SQL> SELECT COUNT(OBJECT_NAME) FROM T10 WHERE OWNER = 'SYS';

SQL> ALTER SESSION SET OPTIMIZER_USE_INVISIBLE_INDEXES=TRUE;
SQL> SELECT COUNT(OBJECT_NAME) FROM T10 WHERE OWNER = 'SYS';

SQL> ALTER INDEX IDX_T10 VISIBLE;
SQL> SELECT COUNT(OBJECT_NAME) FROM T10 WHERE OWNER = 'SYS';

SQL> SET AUTOTRACE OFF
```

Virtual Index

```
SQL> CREATE INDEX IDX_T_OWNER_OBJECT_NAME ON T(OWNER, OBJECT_NAME);
Control+C

SQL> CREATE INDEX IDX_T_OWNER_OBJECT_NAME ON T(OWNER, OBJECT_NAME)
NOSEGMENT;

SQL> SET AUTOTRACE TRACEONLY EXPLAIN
SQL> SELECT COUNT(OBJECT_NAME) FROM T WHERE OWNER = 'SYS';

SQL> ALTER SESSION SET "_USE_NOSEGMENT_INDEXES"=TRUE;
SQL> SELECT COUNT(OBJECT_NAME) FROM T WHERE OWNER = 'SYS';

SQL> SET AUTOTRACE OFF
```

Lenda: um índice impacta prejudica o desempenho em N%

```
$ rlwrap sqlplus SCOTT/TIGER@PROD
SQL> CREATE TABLE T13 (C1 NUMBER);

$ time perl OTI.pl 10000
$ time perl OTI.pl 10000
$ time perl OTI.pl 10000

SQL> CREATE INDEX IDX_BTREE_T13 ON T13(C1);
$ time perl OTI.pl 10000
$ time perl OTI.pl 10000
$ time perl OTI.pl 10000

SQL> DROP INDEX IDX_BTREE_T13;
SQL> CREATE BITMAP INDEX IDX_BITMAP_T13 ON T13(C1);
$ time perl OTI.pl 10000
$ time perl OTI.pl 10000
$ time perl OTI.pl 10000
```

Lenda: um índice impacta prejudica o desempenho em N%

```
$ rlwrap sqlplus SCOTT/TIGER@PROD

SQL> CREATE TABLE T14 AS SELECT * FROM ALL_OBJECTS;
SQL> INSERT INTO T14 SELECT * FROM T14;
SQL> INSERT INTO T14 SELECT * FROM T14;
SQL> INSERT INTO T14 SELECT * FROM T14;
SQL> INSERT INTO T14 SELECT * FROM T14;
SQL> COMMIT;

SQL> CREATE TABLE T15 AS SELECT * FROM T14 WHERE 1=0;

SQL> INSERT INTO T15 SELECT * FROM T14;
SQL> TRUNCATE TABLE T15;
SQL> INSERT INTO T15 SELECT * FROM T14;
SQL> TRUNCATE TABLE T15;
SQL> INSERT INTO T15 SELECT * FROM T14;
SQL> TRUNCATE TABLE T15;
```

Lenda: um índice impacta prejudica o desempenho em N%

Verifique o tempo de sua duplicação, mas com índices.

```
SQL> CREATE INDEX T1_IDX_01 ON T15(OWNER);
SQL> INSERT INTO T15 SELECT * FROM T14;
SQL> TRUNCATE TABLE T15;
SQL> INSERT INTO T15 SELECT * FROM T14;
SQL> TRUNCATE TABLE T15;
SQL> INSERT INTO T15 SELECT * FROM T14;
SQL> TRUNCATE TABLE T15;

SQL> CREATE INDEX T15_IDX_02 ON T15(OBJECT_NAME);
SQL> INSERT INTO T15 SELECT * FROM T14;
SQL> TRUNCATE TABLE T15;
SQL> INSERT INTO T15 SELECT * FROM T14;
SQL> TRUNCATE TABLE T15;
SQL> INSERT INTO T15 SELECT * FROM T14;
SQL> TRUNCATE TABLE T15;
```

Lenda: um índice impacta prejudica o desempenho em N%

Verifique o tempo de sua duplicação, mas um índice composto.

```
SQL> DROP INDEX T15_IDX_01;
SQL> DROP INDEX T15_IDX_02;
SQL> CREATE INDEX T15_IDX_03 ON T15(OWNER,OBJECT_NAME);
SQL> INSERT INTO T15 SELECT * FROM T14;
SQL> TRUNCATE TABLE T15;
SQL> INSERT INTO T15 SELECT * FROM T14;
SQL> TRUNCATE TABLE T15;
SQL> INSERT INTO T15 SELECT * FROM T14;
SQL> TRUNCATE TABLE T15;
```

Lenda: um índice impacta prejudica o desempenho em N%

Verifique o uso dos índices.

```
SQL> CONN SCOTT/TIGER@PROD
SQL> DROP INDEX T15_IDX_03;
SQL> INSERT INTO T15 SELECT * FROM T14;
SQL> CREATE INDEX T15_IDX_01 ON T15(OWNER);
SQL> CREATE INDEX T15_IDX_02 ON T15(OBJECT_NAME);
SQL> ALTER INDEX T15_IDX_01 MONITORING USAGE;
SQL> ALTER INDEX T15_IDX_02 MONITORING USAGE;

SQL> COL INDEX_NAME FORMAT A40
SQL> SELECT INDEX_NAME, MONITORING, USED FROM V$OBJECT_USAGE;
SQL> SELECT * FROM T15 WHERE OWNER = 'SCOTT';
SQL> SELECT COUNT(*) FROM T15 WHERE OWNER = 'SYS';
SQL> SELECT COUNT(*) FROM T15 WHERE OWNER = 'SYSTEM';
SQL> SELECT INDEX_NAME, MONITORING, USED, START_MONITORING
FROM V$OBJECT_USAGE;
```

Problemas a se considerar ao remover índices:

- Não está utilizando o índice, mas deveria utilizar;
- Após o DROP, não é utilizado outro índice;
- Uso da seletividade em índices compostos, mesmo sem utilizar a coluna;
- FKs (Enqueue TM).

Plano de Execução: O que procurar?

- Ponto de aumento de Cost ou Rows.
- Diferença entre A-Rows e E-Rows.
- Nested Loops com grande quantidade de Starts.
- FTS / FIS em objetos com filtros.
- Desperdício:

```
-------------------------------------------------------------------------------
| Id  | Operation                    | Name  | Starts | E-Rows | A-Rows |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |       |      1 |        |      0 |
|   1 |  MERGE JOIN                  |       |      1 |     10 |      0 |
|*  2 |   TABLE ACCESS BY INDEX ROWID | T1    |      1 |     10 |     10 |
|   3 |    INDEX FULL SCAN           | T1_PK |      1 |  10000 |  10000 |
|*  4 |   SORT JOIN                  |       |     10 |     10 |      0 |
|*  5 |    TABLE ACCESS BY INDEX ROWID| T2    |      1 |     10 |     10 |
|   6 |     INDEX FULL SCAN          | T2_PK |      1 |  10000 |  10000 |
-------------------------------------------------------------------------------
```

Plano de Execução: O que procurar (Estatísticas)?

SQL: [+]

ID	Exec Ord	Operation	Go To	More	Cost[2]	Estim Card	LAST Starts	LAST Output Rows	LAST Over/Under Estimate[1]	Work Area
0	19	SELECT STATEMENT			2217	4	1	18	5x under	
1	18	SORT GROUP BY		[+]	2217	4	1	18	5x under	[+]
2	17	. GENERATE CUBE		[+]	2217	4	1	32	8x under	
3	16	.. SORT GROUP BY		[+]	2217	4	1	4	1x	[+]
4	15	... HASH JOIN		[+]	2216	1067	1	1014	1x	[+]
5	1	.... TABLE ACCESS FULL COUNTRIES	[+]	[+]	2	2	1	2	1x	
6	14	.... HASH JOIN		[+]	2214	11736	1	21964	2x under	[+]
7	12	....+ ~~NESTED LOOPS~~		[+]	2214	11736	1	1	*** 11736x over	
8	10	....+. ~~NESTED LOOPS~~		[+]	1694		1	1		
9	8	....+.. ~~STATISTICS COLLECTOR~~		[+]	1694		1	21964		
10	7	....+... HASH JOIN		[+]	1694	11736	1	21964	2x under	[+]
11	5	....+.... MERGE JOIN CARTESIAN		[+]	4	28	1	28	1x	
12	2	....+....+ TABLE ACCESS FULL TIMES	[+]	[+]	2	14	1	14	1x	
13	4	....+....+ BUFFER SORT		[+]	2	2	14	28	1x	[+]
14	3	....+....+. TABLE ACCESS FULL CHANNELS	[+]	[+]	0	2	1	2	1x	
15	6	....+.... TABLE ACCESS FULL SALES	[+]	[+]	1593	14728759	1	14728759	1x	
16	9	....+.. ~~INDEX UNIQUE SCAN CUSTOMERS_PK~~	[+]	[+]			0	0		
17	11	....+. ~~TABLE ACCESS BY INDEX ROWID CUSTOMERS~~	[+]	[+]	510	1	0	0		
18	13	....+ TABLE ACCESS FULL CUSTOMERS	[+]	[+]	510	1626064	1	1626064	1x	

Performance statistics are only available when parameter "statistics_level" was set to "ALL" at hard-parse time, or SQL contains "gather_plan_statistics" hint.
Last Starts and Last Output Rows were captured by SQL Plan Monitor. SQL Exec Start is "2016-07-25 16:52:04". SQL Exec ID is "16777216". Status is "DONE (ALL ROWS)". Key is "433791697552".
*(1) If estim_card * starts < output_rows then under-estimate. If estim_card * starts > output_rows then over-estimate. Color highlights when exceeding * 10x, ** 100x and *** 1000x over/under-estimates.*
(2) Largest contributors for cumulative-statistics columns are shown in red.

```
Oracle 12c - CAT 17:18:38 up: 1.3d,    1 ins,    2 sn,    1 us, 1.4G mt,  4.6% db
ID %CPU LOAD %DCU   AAS  ASC  ASI  ASW  AST IOPS %FR   PGA UTPS UCPS SSRT  %DBT
 1   10    0    3     4    1    0    0    1   20   8 373M    0    6   1m   100

EVENT (C)                                TOT WAITS   TIME(s)  AVG_MS  PCT    WAIT_CLASS
DB CPU                                                   746              58
Datapump dump file I/O                        5704       279    49.0   22       User I/O
db file sequential read                      46252       167     3.6   13       User I/O
direct path write                              714        58    82.6    5       User I/O
direct path read temp                          367        35    97.1    3       User I/O

ID   SID     SPID USR PROG S   PGA SQLID/BLOCKER OPN   E/T STA STE EVENT/*LA  W/T

Enter sql_id: dyrgn2atcaukd

PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------
SQL_ID: dyrgn2atcaukd, child number 0
-------------------------------------------------
SELECT T1.ID, T2.NAME FROM T1, T2 WHERE T1.ID = T2.ID

Plan hash value: 2691992084

-------------------------------------------------------------------------------
|   Id | Operation                      | Name       | Rows | Cost | Stale |
-------------------------------------------------------------------------------
|    0 | SELECT STATEMENT               |            |      |   68 |       |
|    1 |  HASH JOIN                     |            | 1.0M |   68 |       |
|    2 |   INDEX FAST FULL SCAN         | T1_IDX1    |  10k |    7 | YES   |
|    3 |   TABLE ACCESS FULL            | T2         |  10k |   43 | YES   |
-------------------------------------------------------------------------------
```

- SQLT (MOS 215187.1)
- oratop (MOS 1500864.1)

Isenção de responsabilidade

- Não acredite em tudo o que lê. Questione tudo.
- Por algo estar escrito, não significa que é verdade.
- O que é verdade aqui, pode não ser verdade lá.
- O que era verdade ontem, pode não ser verdade hoje.
- O que é verdade hoje, pode não ser verdade amanhã.
- Se os fatos não se adequam à teoria, modifique a teoria.
- Questione, e só acredite em fatos: teste.
- Também tente provar que você está errado.
- Implemente a solução no menor escopo possível.
- Quando você mudar algo, pode acontecer uma de três coisas.

Perguntas?

Este PDF está em http://nervinformatica.com.br/L.pdf

Ricardo Portilho Proni
ricardo@nervinformatica.com.br
Blog: http://nervinformatica.com.br/blog/
LinkedIn: https://www.linkedin.com/in/ricardoportilhoproni/
Twitter: https://twitter.com/rportilhoproni