



## MANUAL DO INICIANTE

# PL SQL TURNING INDEX

BY JAILTON DPAULA



## Sumário

<b>REGRAS BÁSICAS .....</b>	<b>3</b>
<b>SELETIVIDADE ou DENSIDADE .....</b>	<b>3</b>
<b>PLANO DE EXECUÇÃO .....</b>	<b>3</b>
<b>INDEX UNIQUE SCAN.....</b>	<b>3</b>
<b>INDEX RANGER SCAN .....</b>	<b>3</b>
<b>INDEX SKIP SCAN .....</b>	<b>3</b>
<b>INDEX FULL SCAN .....</b>	<b>3</b>
<b>INDEX FAST FULL SCAN.....</b>	<b>3</b>
<b>INDEX.....</b>	<b>4</b>
<b>ÍNDICE INVISÍVEL .....</b>	<b>4</b>
<b>ÍNDICE FUNCTION BASE INDEX .....</b>	<b>4</b>
<b>ÍNDICE COMPACTADO.....</b>	<b>5</b>
<b>ÍNDICE COMPOSTO .....</b>	<b>5</b>
<b>ÍNDICE VIRTUAL .....</b>	<b>6</b>
<b>ÍNDICE DE CHAVE REVERSA .....</b>	<b>6</b>
<b>INFLUÊNCIA DO CUSTO NA ESCOLHA DO ÍNDICE .....</b>	<b>7</b>



## REGRAS BÁSICAS

- Definir *PRIMARY KEY* para cada tabela criada.
- Colunas de valores únicos definir a restrição *UNIQUE*.
- Sempre que uma coluna referenciar um *PRIMARY KEY* de outra tabela criar as *FOREIGN KEY*.
- Se a coluna for muito utilizada na clausura *WHERE* e não tenha nenhum tipo de *INDEX* criar um *INDEX*.
- Criar um *INDEX* composto quando a regra acima utilizar mais de uma coluna.

## SELETIVIDADE ou DENSIDADE

A seletividade tem uma escala de **0.0** até **1.0**, quanto menor melhor o número melhor.



Tabelas para consultas:

**ALL\_TAB\_COLUMNS**  
**DBA\_TAB\_COLUMNS**  
**DBA\_TAB\_COLUMNS**  
**USER\_TAB\_COL\_STATISTICS**

## PLANO DE EXECUÇÃO

### INDEX UNIQUE SCAN

- É utilizado em constraints *PRIMARY KEY* e *UNIQUE*.
- Operador utilizado na consulta é "**=**".

### INDEX RANGER SCAN

- Operador utilizado na consulta é "**=**" e o index é *NONUNIQUE*.
- Operador utilizado é "**>**", "**<**", "**>=**", "**<=**" ou **BETWEEN**.

### INDEX SKIP SCAN

- Quando um índice é composto e na clausula *WHERE* a primeira coluna é ignorado.

### INDEX FULL SCAN

- Todas colunas na clausula *ORDER BY* esteja presente em um índice.
- Todas colunas na clausula *GROUP BY* esteja presente em um índice.

### INDEX FAST FULL SCAN

- Toda coluna está presente no *SELECT* e *GROUP BY* ou *ORDER BY*.



Pode ocorrer de mesmo com essas regras otimizado escolha *TABLE ACCESS FULL* pois a seletividade do index pode ser muito alta.

## INDEX

### ÍNDICE INVISÍVEL

- O índice invisível é um índice que no plano de execução ele é ignorado pelo otimizador, porém sua estrutura é mantida e continua sendo atualizada.
- As restrições contidas nos índices invisíveis como *UNIQUE* continua valendo.
- As coletas automáticas de estatísticas não são coletadas nos índices invisíveis.
- Os índices invisíveis são recomendados para testar novos índices ou validar a exclusão de índices já existentes.
- Não utilizar índice invisível em *PRIMARY KEY*, *FOREIGN KEY* ou *UNIQUE*.



É possível utilizar índice invisível pelo otimizador em uma sessão específica:

```
ALTER SESSION SET OPTIMIZER_USE_INVISIBLE_INDEXES=TRUE;
```



```
CREATE INDEX nome índice ON tabela (coluna) INVISIBLE;
```

```
ALTER INDEX nome índice INVISIBLE;
```

```
ALTER INDEX nome índice VISIBLE;
```



Tabelas para consulta coluna *VISIBILITY*:

```
DBA_INDEXES
```

```
ALL_INDEXES
```

```
USER_INDEXES
```

### ÍNDICE FUNCTION BASE INDEX

- A coluna que recebe um índice FBI cria uma coluna virtual com valor do resultado. Essas colunas não geram estatística automática tendo que ser feito manual através do pacote *DBMS\_STATS.GATHER\_TABLE\_STATS*.
- Para criar um índice FBI não pode utilizar função agregada como *AVG*, *SUM*, *MAX* e etc.
- Exemplo de funções padrões permitidos utilizar *UPPER*, *LOWER*, *DECODE*, *NVL* entre outras.
- Uma coluna que usa índice quando submetido a uma função na cláusula *WHERE* não é utilizada o índice dela, para esses casos o índice FBI é utilizado.



```
CREATE INDEX nome índice ON tabela (função (coluna));
```



Tabelas:

```
USER_TAB_COL_STATISTICS
```

```
USER_IND_COLUMNS
```

```
USER_IND_EXPRESSIONS
```



As funções utilizadas no índice FBI são funções *DETERMINISTIC*.

```
CREATE FUNCTION nome função (variável NUMBER) RETURN NUMBER DETERMINISTIC IS
```



Colunas que precisa criar índice e ela contém valor nulo é recomendado utilizar a função abaixo:

```
CREATE INDEX nome índice ON tabela (NVL(coluna, 'NULL') );
```



## ÍNDICE COMPACTADO

- Quanto menor a seletividade, maior a compactação.
- Só pode ser usado em índice BTREE.
- VANTAGEM: reduz espaço em disco.
- DESVANTAGEM: maior probabilidade de contenção, quando ocorre diversos acessos simultâneo.
- Melhora as operações:
  - INDEX RANGER SCAN
  - INDEX FULL SCAN
  - INDEX FAST FULL SCAN



```
CREATE INDEX nome índice ON tabela (coluna1, coluna2, coluna3) COMPRESS 2;
```

COMPRESS determina o número de coluna que será comprimida.

Determinando colunas para compactação:

O comando VALIDATE INDEX pode ser utilizado para determinar a melhor maneira de criar um índice compactado.



```
VALIDATE INDEX nome index;
```

```
SELECT OPT_CMPR_COUNT, OPT_CMPR_PCTSAVE FROM INDEX_STATS;
```

OPT\_CMPR\_COUNT

Quantidade de coluna que deve ser compactada.

OPT\_CMPR\_PCTSAVE

ganho em porcentagem no tamanho do índice.



Consultar tamanho do índice

```
SELECT ROUND (BYTES/1024) AS KB FROM USER_SERGMENTS WHERE SEGMENT = 'nome índice';
```

## ÍNDICE COMPOSTO

- Índices composto são aqueles que contém duas ou mais colunas da tabela.
- Em uma query para utilização do índice composto todas colunas que faz parte do índice devem estar na clausula WHERE ou pelo menos duas delas.
- Índices composto por sua estrutura reduz o acesso de buffers.

### Regras:

Cria índice composto quando existir mais de uma coluna da mesma tabela que estejam sendo utilizada na clausula *WHERE* e não contém índice nelas.

Ordenar da coluna com menor seletividade para maior. Essa regra ganha vantagem na criação do índice compactado.



Os índices compostos são utilizados também para criar índice para colunas que contém valores nulos, assim quando utilizado o *IS NULL* na clausula *WHERE* a quebra do índice não ocorre, no momento da criação do índice criar com segunda coluna com valor fixo.

```
CREATE INDEX nome índice ON tabela (coluna, 1,);
```

## ÍNDICE VIRTUAL

- Os índices virtuais não criam estrutura física, apenas dicionário de dados.
- Recomendado para validar índices que pretende criar.
- Não utiliza espaço em disco.



**CREATE INDEX** nome índice **ON** tabela (coluna) **NOSEGMENT**;



Tabela:

**USER\_IND\_COLUMNS**

**ALL\_IND\_COLUMNS**

**DBA\_IND\_COLUMNS**



Para o otimizador avaliar um índice virtual em uma sessão utilizar o comando:

**ALTER SESSION SET " \_USE\_NOSEGMENT\_INDEXES"=TRUE;**

## ÍNDICE DE CHAVE REVERSA

- O índice de chave reversa foi criado para resolver um problema de contenção em base rack. quando há várias sessões fazendo *inserts* ao mesmo tempo utilizando o mesmo índice pode acontecer uma contenção, pois eles tentam atualizar o mesmo bloco ao mesmo tempo.
- O índice com chave reversa altera a ordem da chave fazendo a mesma ser gravada em blocos diferentes.



CHAVES -> RESULTADO

281 -> 182

282 -> 282

283 -> 383

No exemplo as chaves mudaram sua ordem assim as mesmas não serão gravadas no mesmo bloco.



Não utiliza operação INDEX RANGE SCAN.

Operadores com m ">", "<", ">=", "<=", "LIKE" e "BETWEEN".



Operadores a ser utilizado para usar o índice "=" e "IN".



**CREATE INDEX** nome índice **ON** tabela (coluna) **REVERSE**;

**ALTER INDEX** nome índice **REBUILD REVERSE**;

**ALTER INDEX** nome índice **NOREVERSE**;



Coluna INDEX\_TYPE nas tabelas:

**USER\_INDEXES**

**ALL\_INDEXES**

**DBA\_INDEXES**



## INFLUÊNCIA DO CUSTO NA ESCOLHA DO ÍNDICE

- Seletividades com valor perto de 1 pode fazer com que o otimizador ignore o índice da coluna.
- Valor de cluter ruim também pode ser fazer com que o otimizador ignore o índice.
- O valor de cluster ruim é aquele que se aproxima da quantidade de linhas da tabela.
- Utilização dos operadores “<>”, “!=”, “**NOT IN**” faz com que o otimizador ignore o índice.
- Quando utilizar o operador “**LIKE**”, da preferência a utilizar o operador coringa “**%**” apenas no final, “**LIKE ‘MARI%’**”.
- A utilização do IS NULL pode fazer com que o índice não seja utilizado



Coluna CLUSTERINC\_FACTOR nas tabelas a seguir para saber a clusterização:

**USER\_INDEXES**

**ALL\_INDEXES**

**DBA\_INDEXES**