

PUCRIO

Data: 29/09/1994

Professor: Seibel

Alunos: João Garcia(1912657), Wellington Bezerra(1413383)

Trabalho

Script para criação de índices

```
CREATE BITMAP INDEX TIME_FUT_INDEX_CIDADE ON TIME_FUT (CIDADE);
```

```
CREATE INDEX TIME_FUT_INDEX_NOME ON TIME_FUT (NOME);
```

```
CREATE BITMAP INDEX TIME_FUT_INDEX_TITULOS ON TIME_FUT (NUM_TIT);
```

```
CREATE INDEX TIME_FUT_INDEX_TORCIDA ON TIME_FUT  
(TORCIDA_ESTIMADA);
```

```
ALTER TABLE TIME_FUT ADD CONSTRAINT TIME_FUT_PK PRIMARY KEY (  
COD ) ENABLE;
```

1)

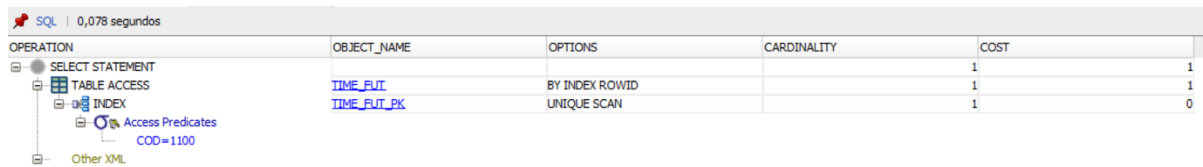
```
SELECT *  
FROM time_fut  
WHERE cidade in ('Rio de Janeiro', 'Belo Horizonte') and num_tit <= 10  
AND torcida_estimada < 39500000
```

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				2
TABLE ACCESS	TIME_FUT	BY INDEX ROWID BATCHED	77	2
Filter Predicates				
AND				
OR				
CIDADE='Belo Horizonte'				
CIDADE='Rio de Janeiro'				
TORCIDA_ESTIMADA < 39500000				
BITMAP CONVERSION		TO ROWIDS		
BITMAP INDEX	TIME_FUT_INDEX_TITULOS	RANGE SCAN		
Access Predicates				
NUM_TIT <= 10				
Filter Predicates				
NUM_TIT <= 10				

Como podemos ver, a consulta acima não é boa, pois ela varre toda a tabela em busca dos atributos desejados dado as comparações com grandeza(\leq , $<$). Esse fato pode ser verificado ao analisarmos o custo da operação que é 2.

2)

```
SELECT *  
FROM time_fut  
WHERE cod = 1100;
```



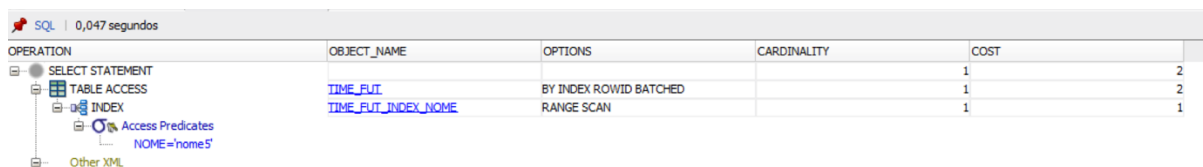
SQL | 0,078 segundos

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				1
TABLE ACCESS	TIME_FUT	BY INDEX ROWID	1	1
INDEX	TIME_FUT_PK	UNIQUE SCAN	1	0
Access Predicates				
		COD=1100		
Other XML				

Como podemos ver acima, o custo dessa consulta é baixo, pois é uma método de Seleção Simples, utiliza uma Busca Binária, já que o arquivo está ordenado pelo código e envolve uma comparação de igualdade em um atributo chave.

3)

```
SELECT *  
FROM time_fut  
WHERE nome = 'nome5';
```



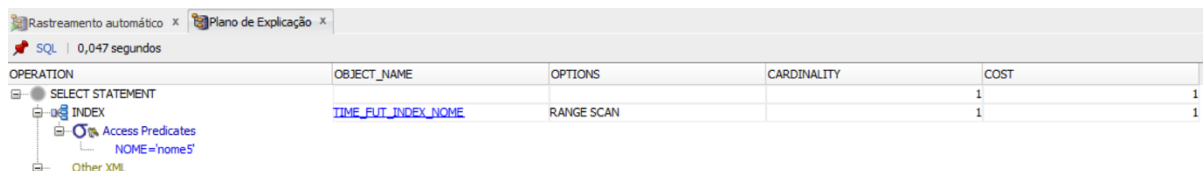
SQL | 0,047 segundos

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				2
TABLE ACCESS	TIME_FUT	BY INDEX ROWID BATCHED	1	2
INDEX	TIME_FUT_INDEX_NOME	RANGE SCAN	1	1
Access Predicates				
		NOME='nome5'		
Other XML				

Dado o fato de que nome já é uma PK de time_fut existe um índice hashing para ela. Ao criarmos um índice B+ precisamos fazer mais operações para a realização dessa busca por que ambos os índices devem ser consultados.

4)

```
SELECT nome  
FROM time_fut  
WHERE nome = 'nome5';
```



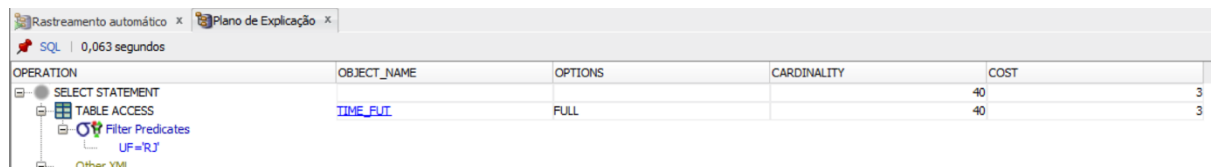
Rastreamento automático x Plano de Explicação x
SQL | 0,047 segundos

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				1
INDEX	TIME_FUT_INDEX_NOME	RANGE SCAN	1	1
Access Predicates				
		NOME='nome5'		
Other XML				

Ao contrário do caso a cima, como não é necessário obter o resto do conteúdo da tupla basta realizar um dos índices por tanto o custo da operação é apenas 1.

5)

```
SELECT *  
FROM time_fut  
WHERE uf = 'RJ';
```

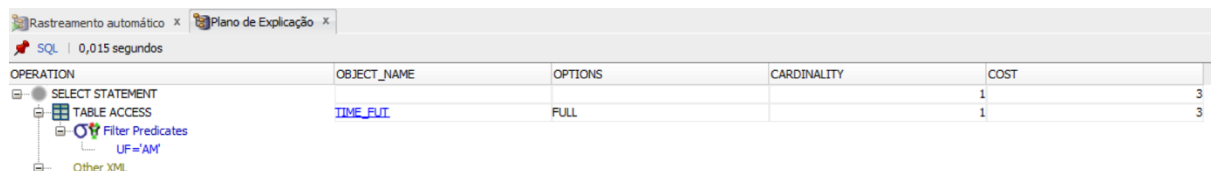


OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				3
TABLE ACCESS	TIME_FUT	FULL	40	40
Filter Predicates				
UF='RJ'				

O custo dessa consulta é muito alto pela necessidade de varrer a tabela completa dado que não existe um índice ou chave auxiliando o processo.

6)

```
SELECT cod  
FROM time_fut  
WHERE uf = 'AM';
```

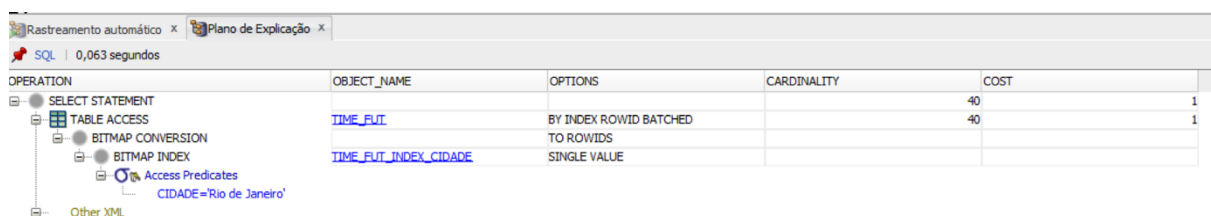


OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				3
TABLE ACCESS	TIME_FUT	FULL	1	1
Filter Predicates				
UF='AM'				

O mesmo exemplo a cima, mas esse prova que o anterior teve de varrer a tabela inteira. Isso se da por que ambos tiveram o mesmo custo tratando no caso anterior de um valor existente na tabela e esse um inexistente.

7)

```
SELECT *  
FROM time_fut  
WHERE cidade = 'Rio de Janeiro';
```

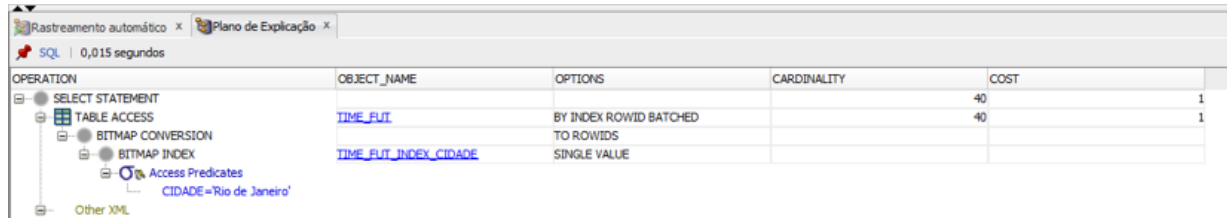


OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				1
TABLE ACCESS	TIME_FUT	BY INDEX ROWID BATCHED	40	40
BITMAP CONVERSION TO ROWIDS				
BITMAP INDEX	TIME_FUT_INDEX_CIDADE	SINGLE VALUE		
Access Predicates				
CIDADE='Rio de Janeiro'				

Ao contrario dos anteriores este exemplo tem um índice no campo "cidade", por tanto o custo da operação é muito inferior.

8)

```
SELECT cod
FROM time_fut
WHERE cidade = 'Rio de Janeiro';
```

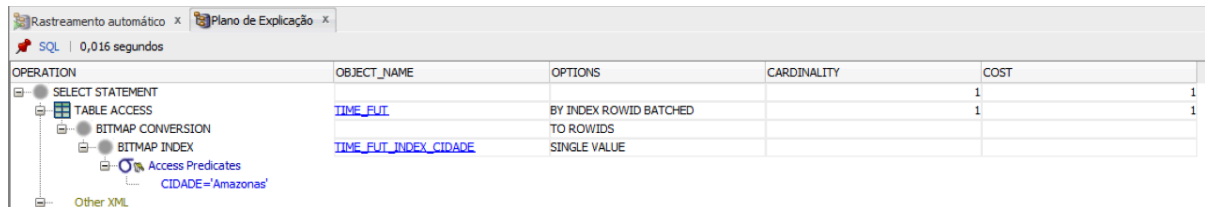


OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				1
TABLE ACCESS	TIME_FUT	BY INDEX ROWID BATCHED	40	40
BITMAP CONVERSION		TO ROWIDS		
BITMAP INDEX	TIME_FUT_INDEX_CIDADE	SINGLE VALUE		

Como o parâmetro de busca nesse caso tem apenas um índice não existe diferença ao buscar por * ou uma única coluna específica, diferente dos números 3 e 4.

9)

```
SELECT cod
FROM time_fut
WHERE cidade = 'Amazonas';
```



OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				1
TABLE ACCESS	TIME_FUT	BY INDEX ROWID BATCHED	1	1
BITMAP CONVERSION		TO ROWIDS		
BITMAP INDEX	TIME_FUT_INDEX_CIDADE	SINGLE VALUE		

Por não existir 'Amazonas' na tabela a cardinalidade do atributo é muito menor e isso pode ser visto nas estatísticas e com essa informação o otimizador consegue estruturar melhor a consulta.