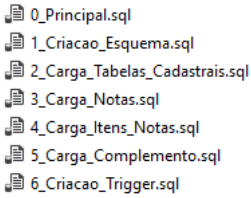
11

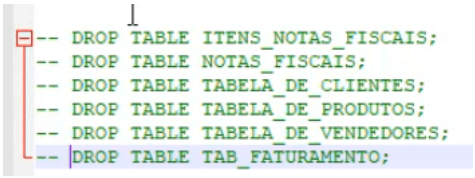
# Consolidando o seu conhecimento

Chegou a hora de você pôr em prática o que foi visto na aula. Para isso, execute os passos listados abaixo.

1) Caso você ainda não tenha feito, faça o download dos arquivos (https://caelum-online-public.s3.amazonaws.com/1603-oracle-db-performance-parte-2/05/arquivos-aula-5.zip) desta aula:




2) Edite o arquivo **1_Criacao_Esquema.sql** e inclua comentários nas primeiras linhas, como mostrado abaixo:




3) Na linha de comando do seu sistema operacional, acesse o **SQL*Plus**, com o usuário **USER13**.

4) Execute os *scripts* baixados. Iniciando pelo primeiro (neste caso, os *scripts* estão localizados no diretório **C:\Arquivos**, então edite e inclua o diretório usado por você):

```
@c:\arquivos\1_Criacao_Esquema.sql;
Commit;
```

5) Siga com o segundo *script*:

```
@c:\arquivos\2_Carga_Tabelas_Cadastrais.sql;
Commit;
```

6) O terceiro:

```
@c:\arquivos\3_Carga_Notas_Fiscais.sql;
Commit;
```

7) O quarto:

```
@c:\arquivos\4_Carga_Itens_Notas.sql;
Commit;
```

8) O quinto:

```
@c:\arquivos\5_Carga_Complemento.sql;
Commit;
```

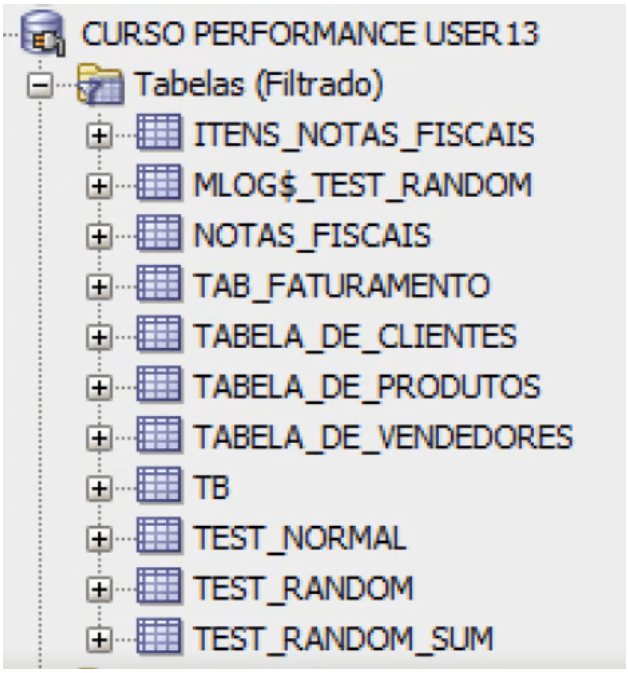
9) E por último, o sexto:

```
@c:\arquivos\6_Criacao_Trigger.sql;
Commit;
```

10) Saia do **SQL*Plus**, digitando:

```
Exit;
```

11) Vá agora para o **SQL Developer**, criando uma área de edição associada ao usuário **USER13**. Você pode ver, ao lado, todas as tabelas criadas até agora:




12) É preciso entender o problema a ser resolvido. Copie o conteúdo do *script* **SQL_Comando,sql**, baixado anteriormente, para a área de edição de comandos SQL criada no passo anterior.

13) Melhore a performance de dois comandos de consulta e de um conjunto de comandos de inclusão. Execute estes comandos e apure o custo das consultas e o tempo da inclusão. As consultas e comandos de inclusão são mostrados abaixo:

```
SELECT TO_CHAR(NOTAS_FISCAIS.DATA_VENDA,'YYYY-MM') AS MES_ANO, SUM(ITENS_NOTAS_FISCAIS.QUANTIDAI
FROM NOTAS_FISCAIS INNER JOIN ITENS_NOTAS_FISCAIS
ON NOTAS_FISCAIS.NUMERO = ITENS_NOTAS_FISCAIS.NUMERO
GROUP BY TO_CHAR(NOTAS_FISCAIS.DATA_VENDA,'YYYY-MM')
ORDER BY TO_CHAR(NOTAS_FISCAIS.DATA_VENDA,'YYYY-MM');

SELECT TABELA_DE_CLIENTES.ESTADO, NOTAS_FISCAIS.TIPO_VENDA, TO_CHAR(NOTAS_FISCAIS.DATA_VENDA,'Y
COUNT(NOTAS_FISCAIS.NUMERO) AS NUMERO_VENDAS FROM NOTAS_FISCAIS
INNER JOIN TABELA_DE_CLIENTES ON NOTAS_FISCAIS.CPF = TABELA_DE_CLIENTES.CPF
GROUP BY TABELA_DE_CLIENTES.ESTADO, NOTAS_FISCAIS.TIPO_VENDA, TO_CHAR(NOTAS_FISCAIS.DATA_VENDA,
```

```
ORDER BY TABELA_DE_CLIENTES.ESTADO, NOTAS_FISCAIS.TIPO_VENDA, TO_CHAR(NOTAS_FISCAIS.DATA_VENDA,

INSERT INTO NOTAS_FISCAIS VALUES ('50534475787','00236',TO_DATE('2018-04-01','YYYY-MM-DD'),1000
INSERT INTO ITENS_NOTAS_FISCAIS VALUES (10000001, '1013793', 63, 24.01);
INSERT INTO ITENS_NOTAS_FISCAIS VALUES (10000001, '1101035', 26, 9.0105);
```

14) Apure o tempo de execução de cada consulta e dos comandos de inclusão, anotando-os em uma planilha:

| | CONSULTA 1 | CONSULTA 2 | INCLUSÃO |
|---|---|---|---|
| STATUS INICIAL | 22,36 | 3,26 | 41,43 |

15) Primeiramente, revise a memória do ambiente. Para isso, crie uma área de edição de comandos SQL, no **SQL*Plus**, associada ao usuário **SYS**. Digite e execute:

```
SELECT MEMORY_SIZE, MEMORY_SIZE_FACTOR, ESTD_DB_TIME
FROM V$MEMORY_TARGET_ADVICE WHERE MEMORY_SIZE_FACTOR > 1;
```

| | MEMORY_SIZE | MEMORY_SIZE_FACTOR | ESTD_DB_TIME |
|---|---|---|---|
| 1 | 1000 | 1,25 | 284 |
| 2 | 1200 | 1,5 | 284 |
| 3 | 1400 | 1,75 | 270 |
| 4 | 1600 | 2 | 270 |

O resultado acima mostra que a memória pode ser melhorada. Claro que este resultado vai depender de cada máquina.

16) Você pode observar os valores para os parâmetros alvo configurados no momento. Digite e execute:

```
SHOW PARAMETERS TARGET;
```

```
fast_start_mttr_target          integer     0
memory_max_target               big integer 1000M
memory_target                   big integer 800M
parallel_servers_target         integer     128
```

17) Melhore estes parâmetros. Digite e execute:

```
ALTER SYSTEM SET MEMORY_MAX_TARGET = 1800M SCOPE=SPFILE;
ALTER SYSTEM SET MEMORY_TARGET = 1400M SCOPE=SPFILE;
```

18) Para estes novos parâmetros valerem, acesse o **SQL*Plus** via linha de comando, e se conecte como:

```
sqlplus / as sysdba
```

19) Pare e inicialize o ambiente. Lembre-se de fechar todas as sessões abertas no **SQL Developer**:

```
shutdown;
startup;
```

20) Volte ao **SQL Developer** e entre como **USER13**.

21) Avalie os dois comandos de SQL e os comandos de `INSERT`. Compare os tempos. No caso do vídeo, foi obtido:

| | CONSULTA 1 | CONSULTA 2 | INCLUSÃO |
|---|---|---|---|
| STATUS INICIAL | 22,36 | 3,26 | 41,43 |
| APÓS MEMÓRIA | 15,32 | 2,93 | 27,25 |

22) Uma ação importante seria a de atualizar as estatísticas. Isso pode melhorar a performance. Para isso digite:

```
EXEC DBMS_STATS.GATHER_SCHEMA_STATS('SYSTEM');
```

23) Confira novamente as consultas e as inclusões:

| | CONSULTA 1 | CONSULTA 2 | INCLUSÃO |
|---|---|---|---|
| STATUS INICIAL | 22,36 | 3,26 | 41,43 |
| APÓS MEMÓRIA | 15,32 | 2,93 | 27,25 |
| APÓS ATUALIZAÇÃO ESTATÍSTICAS | 16,62 | 2,9 | 28,73 |

24) Execute o **Tuning Advisor** para os dois comandos SQL e veja se existem dicas para serem atualizadas. O primeiro SQL é o seguinte:

```
SELECT TO_CHAR(NOTAS_FISCAIS.DATA_VENDA,'YYYY-MM') AS MES_ANO,
    SUM(ITENS_NOTAS_FISCAIS.QUANTIDADE * ITENS_NOTAS_FISCAIS.PRECO) AS TOTAL_VENDA
FROM NOTAS_FISCAIS INNER JOIN ITENS_NOTAS_FISCAIS
ON NOTAS_FISCAIS.NUMERO = ITENS_NOTAS_FISCAIS.NUMERO
GROUP BY TO_CHAR(NOTAS_FISCAIS.DATA_VENDA,'YYYY-MM')
ORDER BY TO_CHAR(NOTAS_FISCAIS.DATA_VENDA,'YYYY-MM');
```

Ele possui, no seu plano de execução, um custo de **74319**.

25) Crie uma nova área de edição de comandos SQL e execute o **Tuning Advisor** para o primeiro comando SQL:

```
VARIABLE vtask VARCHAR2(1000);

EXECUTE :vtask:=DBMS_SQLTUNE.CREATE_TUNING_TASK(
    task_name => 'TAREFA_SQL_01',
    sql_text => 'SELECT TO_CHAR(NOTAS_FISCAIS.DATA_VENDA,''YYYY-MM'') AS MES_ANO, SUM(ITENS_NOTA
);

EXECUTE DBMS_SQLTUNE.EXECUTE_TUNING_TASK('TAREFA_SQL_01');

SELECT DBMS_SQLTUNE.REPORT_TUNING_TASK('TAREFA_SQL_01') FROM DUAL;
```

26) Foi sugerido usar um outro plano de execução. Para isso, deve ser executado o comando marcado no relatório emitido pelo **Advisor**:

```
Recommendation (estimated benefit: 73.41%)
------------------------------------------
- Considere a hipótese de aceitar o perfil de SQL recomendado para usar a
  execução em paralelo nesta instrução.
  execute dbms_sqltune.accept_sql_profile(task_name => 'TAREFA_SQL_01',
          task_owner => 'USER13', replace => TRUE, profile_type =>
          DBMS_SQLTUNE.PX_PROFILE);
```

27) Execute:

```
execute dbms_sqltune.accept_sql_profile(
    task_name => 'TAREFA_SQL_01',
    task_owner => 'USER13',
    replace => TRUE,
    profile_type => DBMS_SQLTUNE.PX_PROFILE
);
```

28) Execute novamente a primeira consulta e veja o seu plano de execução. O custo cai para **19759**. A estatística fica semelhante a:

| | CONSULTA 1 | CONSULTA 2 | INCLUSÃO |
|---|---|---|---|
| STATUS INICIAL | 22,36 | 3,26 | 41,43 |
| APÓS MEMÓRIA | 15,32 | 2,93 | 27,25 |
| APÓS ATUALIZAÇÃO ESTATÍSTICAS | 16,62 | 2,9 | 28,73 |
| APÓS TUNNING ADVISOR | 7,39 | | |

29) Execute o **Tuning Advisor** para a segunda consulta, em uma outra sessão de comandos SQL, associada ao usuário **USER13**:

```
VARIABLE vtask VARCHAR2(1000);

EXECUTE :vtask:=DBMS_SQLTUNE.CREATE_TUNING_TASK(
    task_name => 'TAREFA_SQL_02',
    sql_text => 'SELECT TABELA_DE_CLIENTES.ESTADO, NOTAS_FISCAIS.TIPO_VENDA, TO_CHAR(NOTAS_FISC
);

EXECUTE DBMS_SQLTUNE.EXECUTE_TUNING_TASK('TAREFA_SQL_02');

SELECT DBMS_SQLTUNE.REPORT_TUNING_TASK('TAREFA_SQL_02') FROM DUAL;
```

30) Também será achado um plano de execução melhor. Digite então os comandos sugeridos pelo **Advisor**:

```
execute dbms_sqltune.accept_sql_profile(
    task_name => 'TAREFA_SQL_02',
    task_owner => 'USER13',
    replace => TRUE,
    profile_type => DBMS_SQLTUNE.PX_PROFILE
);
```

31) Executando a consulta, você terá a nova estatística de ganho de performance:

| | CONSULTA 1 | CONSULTA 2 | INCLUSÃO |
|---|---|---|---|
| STATUS INICIAL | 22,36 | 3,26 | 41,43 |
| APÓS MEMÓRIA | 15,32 | 2,93 | 27,25 |
| APÓS ATUALIZAÇÃO ESTATÍSTICAS | 16,62 | 2,9 | 28,73 |
| APÓS TUNNING ADVISOR | 7,39 | 2,62 | 28,73 |

32) Aplique o **Access Advisor** para as duas consultas. Em uma outra sessão de comandos SQL, digite:

```
EXECUTE DBMS_ADVISOR.QUICK_TUNE(
    DBMS_ADVISOR.SQLACCESS_ADVISOR,
    'TAREFA_SQL_01_B',
    'SELECT TO_CHAR(NOTAS_FISCAIS.DATA_VENDA,''YYYY-MM'') AS MES_ANO, SUM(ITENS_NOTAS_FISCAIS.QU
);

EXECUTE DBMS_ADVISOR.CREATE_FILE(
    DBMS_ADVISOR.GET_TASK_SCRIPT('TAREFA_SQL_01_B'),
    'TUNE_SCRIPTS',
    'TAREFA_SQL_01_B.SQL'
);
```

33) No arquivo externo criado pelo segundo comando acima, há a sugestão de criar uma ***Materialized View*** e executar a atualização das estatísticas. Proceda com isso:

```
CREATE MATERIALIZED VIEW "USER13"."TAREFA_SQL_01_B"
REFRESH FORCE WITH ROWID
ENABLE QUERY REWRITE
AS
SELECT TO_CHAR(NOTAS_FISCAIS.DATA_VENDA,'YYYY-MM') AS MES_ANO,
    SUM(ITENS_NOTAS_FISCAIS.QUANTIDADE * ITENS_NOTAS_FISCAIS.PRECO) AS TOTAL_VENDA
FROM NOTAS_FISCAIS
INNER JOIN ITENS_NOTAS_FISCAIS ON NOTAS_FISCAIS.NUMERO = ITENS_NOTAS_FISCAIS.NUMERO
GROUP BY TO_CHAR(NOTAS_FISCAIS.DATA_VENDA,'YYYY-MM');
begin
    dbms_stats.gather_table_stats('"USER13"','"TAREFA_SQL_01_B"',NULL,dbms_stats.auto_sample_siz
end;
```

34) Revisando a estatística para a primeira consulta, você terá:

| | CONSULTA 1 | CONSULTA 2 | INCLUSÃO |
|---|---|---|---|
| STATUS INICIAL | 22,36 | 3,26 | 41,43 |
| APÓS MEMÓRIA | 15,32 | 2,93 | 27,25 |
| APÓS ATUALIZAÇÃO ESTATÍSTICAS | 16,62 | 2,9 | 28,73 |
| APÓS TUNNING ADVISOR | 7,39 | 2,62 | 28,73 |
| APÓS ACCESS ADVISOR | 0,02 | | |

35) Executando o **Access Advisor** para a segunda consulta:

```
EXECUTE DBMS_ADVISOR.QUICK_TUNE(
    DBMS_ADVISOR.SQLACCESS_ADVISOR,
    'TAREFA_SQL_02_B',
    'SELECT TABELA_DE_CLIENTES.ESTADO, NOTAS_FISCAIS.TIPO_VENDA, TO_CHAR(NOTAS_FISCAIS.DATA_VENI
);

EXECUTE DBMS_ADVISOR.CREATE_FILE(DBMS_ADVISOR.GET_TASK_SCRIPT('TAREFA_SQL_02_B'),'TUNE_SCRIPTS',
```

36) E depois executando as dicas apresentadas pelo arquivo gerado:

```
CREATE MATERIALIZED VIEW "USER13"."TAREFA_SQL_02_B"
REFRESH FORCE WITH ROWID
```

```
ENABLE QUERY REWRITE
AS
SELECT TABELA_DE_CLIENTES.ESTADO, NOTAS_FISCAIS.TIPO_VENDA,
    TO_CHAR(NOTAS_FISCAIS.DATA_VENDA, 'YYYY-MM') AS MES_ANO,
    COUNT(NOTAS_FISCAIS.NUMERO) AS NUMERO_VENDAS
FROM NOTAS_FISCAIS
INNER JOIN TABELA_DE_CLIENTES ON NOTAS_FISCAIS.CPF = TABELA_DE_CLIENTES.CPF
GROUP BY TABELA_DE_CLIENTES.ESTADO, NOTAS_FISCAIS.TIPO_VENDA,
    TO_CHAR(NOTAS_FISCAIS.DATA_VENDA,'YYYY-MM');

begin
    dbms_stats.gather_table_stats('"USER13"','"TAREFA_SQL_02_B"',NULL,dbms_stats.auto_sample_si
end;
```

37) Você terá as novas estatísticas:

| | CONSULTA 1 | CONSULTA 2 | INCLUSÃO |
|---|---|---|---|
| STATUS INICIAL | 22,36 | 3,26 | 41,43 |
| APÓS MEMÓRIA | 15,32 | 2,93 | 27,25 |
| APÓS ATUALIZAÇÃO ESTATÍSTICAS | 16,62 | 2,9 | 28,73 |
| APÓS TUNNING ADVISOR | 7,39 | 2,62 | 28,73 |
| APÓS ACCESS ADVISOR | 0,02 | 0,017 | 28,73 |

38) A solução sugerida foi criar uma *Materialized View*. Mas nem sempre a solução será criar a visão. Crie mais uma área de edição, associada ao usuário **USER13** e execute o seguinte comando de seleção:

```
SELECT TIPO_VENDA, COUNT(*) AS NUM_NOTAS FROM NOTAS_FISCAIS
WHERE TIPO_VENDA = 'Atacado'
GROUP BY TIPO_VENDA;
```

Anote o tempo da execução desta consulta e o custo pelo plano de execução:

| CONSULTA 3 | |
|---|---|
| 1,684 | 21091 |

39) Rode o **Access Advisor** sobre ela:

```
EXECUTE DBMS_ADVISOR.QUICK_TUNE(
    DBMS_ADVISOR.SQLACCESS_ADVISOR,
    'TAREFA_SQL_03_B',
    'SELECT TIPO_VENDA, COUNT(*) AS NUM_NOTAS FROM NOTAS_FISCAIS WHERE TIPO_VENDA = ''Atacado''
);

EXECUTE DBMS_ADVISOR.CREATE_FILE(
    DBMS_ADVISOR.GET_TASK_SCRIPT('TAREFA_SQL_03_B'),
    'TUNE_SCRIPTS',
    'TAREFA_SQL_03_B.SQL'
);
```

40) A sugestão foi de criar um índice **BITMAP**. Execute a sugestão do **Access Advisor**:

```
CREATE BITMAP INDEX "USER13"."NOTAS_FISCAIS_IDX_TIPO_VENDA"
ON "USER13"."NOTAS_FISCAIS"
("TIPO_VENDA")
COMPUTE STATISTICS;
```

41) Colete os tempos e o custo do plano de execução:

| | CONSULTA 1 | CONSULTA 2 | INCLUSÃO | CONSULTA 3 | |
|---|---|---|---|---|---|
| STATUS INICIAL | 22,36 | 3,26 | 41,43 | 1,684 | 21091 |
| APÓS MEMÓRIA | 15,32 | 2,93 | 27,25 | | |
| APÓS ATUALIZAÇÃO ESTATÍSTICAS | 16,62 | 2,9 | 28,73 | | |
| APÓS TUNNING ADVISOR | 7,39 | 2,62 | 28,73 | | |
| APÓS ACCESS ADVISOR | 0,02 | 0,017 | 28,73 | 0,024 | 127 |

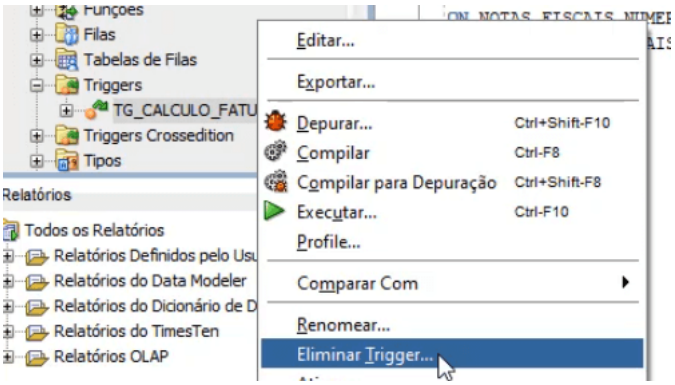
42) Para resolver o problema da inclusão dos dados, não é necessário executar nenhum **Advisor**. Isso porque existe uma *Trigger* que cria uma tabela reduzida dos totais de faturamento e a geração desta tabela, a cada comando de inclusão, alteração ou exclusão, está prejudicando a execução deste comando. Você pode substituir a *Trigger* pela *Materialized View*. Logo, execute o comando abaixo, para criar uma *Materialized View* baseada na mesma consulta da *Trigger*:

```
CREATE MATERIALIZED VIEW USER13.VIEW_TAB_FATURAMENTO
REFRESH FORCE WITH ROWID
ENABLE QUERY REWRITE
AS
SELECT NOTAS_FISCAIS.DATA_VENDA,
    SUM(ITENS_NOTAS_FISCAIS.QUANTIDADE * ITENS_NOTAS_FISCAIS.PRECO) AS TOTAL_VENDA
FROM NOTAS_FISCAIS INNER JOIN ITENS_NOTAS_FISCAIS
ON NOTAS_FISCAIS.NUMERO = ITENS_NOTAS_FISCAIS.NUMERO
GROUP BY NOTAS_FISCAIS.DATA_VENDA;
```

43) Atualize as estatísticas do SQL:

```
begin
    dbms_stats.gather_table_stats(
        '"USER13"',
        '"VIEW_TAB_FATURAMENTO"',
        NULL,
        dbms_stats.auto_sample_size
    );
end;
```

44) Elimine a *Trigger*, clicando com o botão direito do mouse sobre o seu nome, na árvore à esquerda, no **SQL Developer**:



45) Execute os comandos de `INSERT`, editando o número da nota fiscal:

```
INSERT INTO NOTAS_FISCAIS VALUES ('50534475787','00236',TO_DATE('2018-04-01','YYYY-MM-DD'),1000(
INSERT INTO ITENS_NOTAS_FISCAIS VALUES (10000004, '1013793', 63, 24.01);
INSERT INTO ITENS_NOTAS_FISCAIS VALUES (10000004, '1101035', 26, 9.0105);
```

46) Colete os tempos:

| | CONSULTA 1 | CONSULTA 2 | INCLUSÃO |
|---|---|---|---|
| STATUS INICIAL | 22,36 | 3,26 | 41,43 |
| APÓS MEMÓRIA | 15,32 | 2,93 | 27,25 |
| APÓS ATUALIZAÇÃO ESTATÍSTICAS | 16,62 | 2,9 | 28,73 |
| APÓS TUNNING ADVISOR | 7,39 | 2,62 | 28,73 |
| APÓS ACCESS ADVISOR | 0,02 | 0,017 | 28,73 |
| APÓS A VIEW | 0,02 | 0,017 | 0,017 |