IComp/UFAM - Bancos de Dados 1 – 2017/02
Ficha de Resposta do Trabalho Prático 3

Atividade: 1	**Tarefa:** 2	**Data:** 18/10	**Folha:** 1 **de** 1

Aluno: Micael Levi **Matrícula:** 21554923
Aluno: Moisés Gomes **Matrícula:** 21550188

```
import sqlite3

try:

    con = sqlite3.connect(':memory:')
    print('pysqlite version =', sqlite3.version)
    print('SQLite db lib version =', sqlite3.sqlite_version)

    with con:
        cur = con.cursor()
        cur.execute("ATTACH DATABASE 'firstDB.db' AS db")

        cur.executescript("""
            DROP TABLE IF EXISTS db.alunos;
            CREATE TABLE IF NOT EXISTS db.alunos (matricula INT PRIMARY KEY, nome TEXT NOT NULL);
            INSERT INTO db.alunos VALUES(21554923, 'micael');
            INSERT INTO db.alunos VALUES(21550188, 'moisés');
        """)

        cur.execute("SELECT * FROM db.alunos")
        rows = cur.fetchall()
        for row in rows:
            print(row)

        cur.execute("DETACH DATABASE db")
        con.commit()

except sqlite3.Error as e:
    if con: con.rollback()
    print('[ERROR]', e)
finally:
    if con: con.close()
```

IComp/UFAM - Bancos de Dados 1 – 2017/02
Ficha de Resposta do Trabalho Prático 3

Atividade: 1	**Tarefa:** 1	**Data:** 18/10	**Folha:** 1 **de** 1

Aluno: Micael Levi **Matrícula:** 21554923
Aluno: Moisés Gomes **Matrícula:** 21550188

```
2017-10-18 09:10:10 AMT LOG:  database system was shut down at 2017-10-18 09:10:09 AMT
2017-10-18 09:10:10 AMT LOG:  MultiXact member wraparound protections are now enabled
2017-10-18 09:10:10 AMT LOG:  autovacuum launcher started
2017-10-18 09:10:10 AMT LOG:  database system is ready to accept connections
2017-10-18 09:10:11 AMT LOG:  incomplete startup packet
```

IComp/UFAM - Bancos de Dados 1 – 2017/02
Ficha de Resposta do Trabalho Prático 3

Atividade: 1	**Tarefa:** 5	**Data:** 23/10	**Folha:** 1 **de** 1

Aluno: Micael Levi **Matrícula:** 21554923
Aluno: Moisés Gomes **Matrícula:** 21550188

```
SO:                          Linux Mint 17.3 "Rosa" - Cinnamon
Arquitetura:                 x86_64
Modo(s) operacional da CPU: 32-bit, 64-bit
processador:                 Intel(R) Core(TM) i5-5200U CP
RAM:                         8GiB SODIMM DDR3 Síncrono 1600
Byte Order:                  Little Endian
CPU(s):                      4
On-line CPU(s) list:         0-3
Thread(s) per núcleo:        2
Núcleo(s) por soquete:       2
Soquete(s):                  1
Nó(s) de NUMA:               1
ID de fornecedor:            GenuineIntel
Família da CPU:              6
Modelo:                      61
Step:                        4
CPU MHz:                     987.937
BogoMIPS:                    4389.53
Virtualização:               VT-x
cache de L1d:                32K
cache de L1i:                32K
cache de L2:                 256K
cache de L3:                 3072K
NUMA node0 CPU(s):           0-3
```

IComp/UFAM - Bancos de Dados 1 – 2017/02
Ficha de Resposta do Trabalho Prático 3

Atividade: 2	**Tarefa:** 7	**Data:** 25/10	**Folha:** 1 **de** 2

Aluno: Micael Levi **Matrícula:** 21554923
Aluno: Moisés Gomes **Matrícula:** 21550188

DETALHES DE ARMAZENAMENTO FÍSICO DE DADOS **PostgreSQL**	
Base de Dados	Os arquivos de dados usados por um cluster de banco de dados são armazenados juntos dentro do diretório de dados do cluster, chamado de PGDATA.
Tabelas	São armazenadas em um arquivo separado. Para relações comuns, esses arquivos são nomeados após a tabela ou o número do arquivo de indice. Quando uma tabela excede 1GB ela é dividida em seguimentos.
Linhas da Tabela	Como as tabelas são armazenadas como uma matriz de páginas de tamanho fixo (8kb), todas as tabelas são de tamanho equivalente, logo as linhas podem ser armazenadas em qualquer página.
Número máximo de colunas	Entre 250 e 1600 dependendo dos tipos de coluna. Colunas não podem abranger páginas.
Atributo de tamanho grande	Usam a técnica de armazenamento de atributo de tamanho grande (TOAST), que separa os dados das colunas grandes em "peças" menores e as armazena em uma tabela TOAST.
Mapa de espaço livre	Cada tabela possui um; armazena informações sobre a quantidade de espaço livre na relação. É armazenado em um arquivo com o "número do arquivo filenode mais sufixo.fsm"
Mapa de visibilidade	Cada tabela possui um; Serve para acompanhar as páginas que contêm as tuplas que são conhecidas por serem visíveis para todas as transações ativas. É armazenado ao lado do arquivo de tabela em um arquivo separado, nomeado com o "número do arquivo filenode mais _vm".
Índices	Também são armazenados como arquivos no mesmo diretório que as tabelas.
Catálogo	Contém as tabelas do sistema e todos os tipos, funções e operadores de dados incorporados.

Referência: http://rachbelaid.com/introduction-to-postgres-physical-storage/

IComp/UFAM - Bancos de Dados 1 – 2017/02
Ficha de Resposta do Trabalho Prático 3

Atividade: 2	**Tarefa:** 7	**Data:** 25/10	**Folha:** 2 **de** 2

Aluno: Micael Levi **Matrícula:** 21554923
Aluno: Moisés Gomes **Matrícula:** 21550188

SQLite
(Alguns detalhes, não todos)

Arquivo de Banco de Dados	Contido em um único arquivo de dados armazenados no disco, chamado de "arquivo de dados principal". Consiste em uma ou mais páginas.
Páginas	O tamanho varia entre 512 e 65536. Possuem mesmo tamanho, que é definido por um inteiro de 2 bytes localizado em um deslocamento de 16 bytes desde o início do arquivo de banco de dados
Cabeçalho de banco de Dados	Compreendido nos primeiros 100 bytes do arquivo de banco de dados. É dividido em vários campos
Página Lock-Byte	Única página do arquivo de banco de dados que contém os bytes em offset entre 107374824 e 107374335. Arquivos de banco de dados menores que 107374824 bytes não possuem página lock-byte, os maiores que 107374335 possuem apenas uma única página lock-byte.
Freelist	Armazena as páginas não utilizadas. São organizadas como uma lista encadeada.
Páginas da árvore B	É uma página interna ou uma página de folhas, que contém chaves e, no caso de uma tabela b-tree, cada chave tem dados associados. Possuem uma cabeçalho de 8 bytes para páginas de folhas e 12 para páginas internas.
Páginas de sobrecarga de carga útil de células	Armazenam o transbordamento de carga útil de uma célula b-tree; Formam um lista encadeada, em que os primeiros 4 bytes guardam o número da próxima página na cadeia ou zero para a página final. O quinto byte é usado para manter o conteúdo do transbordamento.
Mapa de ponteiro ou páginas de Prtmap	Páginas extras para tornar a operação dos modos auto_vacuum e incremental_vacuum mais eficiente.
Rollback Journal	Arquivo associado a cada banco de dados que contém informações usadas para restaurar o arquivo de banco de dados para seu estado inicial.

Referência: https://www.sqlite.org/fileformat.html

Análise

Enquanto o PostgreSQL armazena os arquivos de dados em arquivos localizados em diretórios diferentes - o que nos dá a ideia de algo mais organizado -, o SQLite armazena todos os dados em um único arquivo de banco de dados que contém um cabeçalho com um endereçamento para saber em que byte começa cada dado especifico.

IComp/UFAM - Bancos de Dados 1 – 2017/02
Ficha de Resposta do Trabalho Prático 3

Atividade: 2	**Tarefa:** 8	**Data:** 28/10	**Folha:** 1 **de** 2

Aluno: Micael Levi **Matrícula:** 21554923
Aluno: Moisés Gomes **Matrícula:** 21550188

Em relação aos limites:

Sistema	Tamanho máximo em nomes	Caracteres permitidos em nomes	Tamanho máximo de um arquivo	Tamanho máximo do volume
Ext2	255 bytes	qualquer, exceto NULL e '/'	16 GiB a 2 TiB	2 TiB a 32 TiB
Ext3	255 bytes	qualquer, exceto NULL e '/'	16 GiB a 2 TiB	2 TiB a 32 TiB
ReiserFS	4032 bytes	qualquer, exceto NULL e '/'	4 GiB a 8 TiB	16 TiB
XFS	255 bytes	qualquer, exceto NULL ('\0')	8 EiB	8 EiB

Fonte: https://en.wikipedia.org/wiki/Comparison_of_file_systems

Vantagens e desvantagens gerais de cada sistema:

Ext2

- Provêm um sistema que respeita a semântica UNIX
- Tal influência pode ser vista na utilização de grupos de blocos (conjunto de setores de 512 bytes)
- A menor unidade de alocação é o bloco; pode ter tamanho de 1024, 2048 ou 4096 bytes (4 KiB)
- Na escrita em um arquivo, tenta-se alocar blocos de dados no mesmo grupo que contém o inodes, reduzindo o movimento da(s) cabeça(s) de leitura-escrita
- Os metadados do sistema de arquivos estão em locais fixos conhecidos, permitindo que este sistema seja recuperado em corrupção de dados significante

Ext3

- Acrescenta recursos ao Ext2, como o journaling (registro de transações para recuperação do sistema)
- Desempenho inferior ao ReiserFS e XFS
- Vantagem de permitir que seja feita a atualização direta a partir de um sistema com Ext2 sem realizar backup e restaurar os dados
- Proporciona um menor consumo de processamento
- Não há uma ferramenta online de desfragmentação

IComp/UFAM - Bancos de Dados 1 – 2017/02
Ficha de Resposta do Trabalho Prático 3

Atividade: 2	**Tarefa:** 8	**Data:** 28/10	**Folha:** 2 **de** 2

Aluno: Micael Levi **Matrícula:** 21554923
Aluno: Moisés Gomes **Matrícula:** 21550188

ReiserFS

- Primeiro sistema com suporte a journaling
- Recupera a consistência do sistema de arquivos em pouco tempo e com menor perda de pastas ou partições
- Usa árvores balanceadas para tornar a busca e outras operações mais eficiente
- Os dados de arquivos pequenos podem ser armazenados próximo aos metadados, agilizando na recuperação de ambos
- Melhor desempenho ao abrir vários arquivos pequenos
- Um bloco pode ser formatado ou não-formatado e o tamanho suportado é de 4 KiB
- Alto consumo de CPU (de 7% a 99%)

XFS

- Tem suporte a journaling
- Aloca extensões em vez de blocos
- Usa alocação dinâmica de inodes

IComp/UFAM - Bancos de Dados 1 – 2017/02
Ficha de Resposta do Trabalho Prático 3

Atividade: 3	**Tarefa:** 11	**Data:** 20/10	**Folha:** 1 **de** 1

Aluno: Micael Levi **Matrícula:** 21554923
Aluno: Moisés Gomes **Matrícula:** 21550188

```
postgres=# SELECT DISTINCT v FROM t;
 v
---
 6
 8
 1
 2
 3
 4
 5
 9
 0
 7
(10 rows)

postgres=# SELECT COUNT(*) FROM t;
 count
--------
 100000
(1 row)
```

IComp/UFAM - Bancos de Dados 1 – 2017/02
Ficha de Resposta do Trabalho Prático 3

Atividade: 3	**Tarefa:** 12	**Data:** 23/10	**Folha:** 1 **de** 1

Aluno: Micael Levi **Matrícula:** 21554923
Aluno: Moisés Gomes **Matrícula:** 21550188

```
postgres=# SELECT relname, relpages, reltuples FROM pg_class
WHERE relname='t';
 relname | relpages | reltuples
---------+----------+-----------
 t       |      443 |    100000
(1 row)
```

443 páginas com blocos foram criadas

IComp/UFAM - Bancos de Dados 1 – 2017/02
Ficha de Resposta do Trabalho Prático 3

Atividade: 3	**Tarefa:** 12	**Data:** 23/10	**Folha:** 1 **de** 1

Aluno: Micael Levi **Matrícula:** 21554923
Aluno: Moisés Gomes **Matrícula:** 21550188


```
postgres=# SELECT * FROM pg_stats WHERE tablename='t';
-[ RECORD 1 ]----------
+----------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------
schemaname             | public
tablename              | t
attname                | k
inherited              | f
null_frac              | 0
avg_width              | 4
n_distinct             | -1
most_common_vals       |
most_common_freqs      |
histogram_bounds       |
{2,1108,2200,3225,4165,5225,6213,7246,8171,9142,10245,11247,12227,13247,14287,15277,16297,17258,18287,19248,20233,21269,22323,23310
,24350,25338,26314,27346,28406,29397,30306,31299,32221,33217,34216,35317,36284,37194,38126,39130,40094,41153,42154,43132,44082,4504
5,46057,47071,48066,49031,49974,50922,51935,52974,54001,54995,56000,57021,57980,59019,59963,60943,61913,62932,64000,65051,66081,670
87,68054,69178,70141,71198,72251,73233,74180,75152,76128,77171,78069,79095,80124,81071,82048,82992,84044,85065,85977,87018,88109,89
118,90074,91113,92161,93174,94152,95054,96031,97015,98032,99015,99998}
correlation            | 1
most_common_elems      |
most_common_elem_freqs |
elem_count_histogram   |
-[ RECORD 2 ]----------
+----------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------
schemaname             | public
tablename              | t
attname                | v
inherited              | f
null_frac              | 0
avg_width              | 4
n_distinct             | 10
most_common_vals       | {8,7,5,4,6,0,1,2,3,9}
most_common_freqs      | {0.103767,0.1018,0.101667,0.101333,0.1007,0.0985,0.0984333,0.0981333,0.0978333,0.0978333}
histogram_bounds       |
correlation            | 0.101509
most_common_elems      |
most_common_elem_freqs |
elem_count_histogram   |
```

2 blocos/registros foram efetivamente usados na consulta

IComp/UFAM - Bancos de Dados 1 – 2017/02
Ficha de Resposta do Trabalho Prático 3

Atividade: 3	**Tarefa:** 14	**Data:** 25/10	**Folha:** 1 **de** 1

Aluno: Micael Levi **Matrícula:** 21554923
Aluno: Moisés Gomes **Matrícula:** 21550188

```
\timing
CREATE INDEX v_idx ON t (v);
SELECT count(*) FROM t WHERE v=1;
REINDEX INDEX v_idx;
```

quantidade de tuplas	tempo gasto para realizar uma consulta para um valor (ms)	tempo gasto para re-criar um índice para o atributo 'v' (ms)
10.000	2,967	156,934
10.000.000	366,185	19448,764

3	15	27/10	1 1

Micael Levi 21554923
Moisés Gomes 21550188

Consulta realizada nos testes:

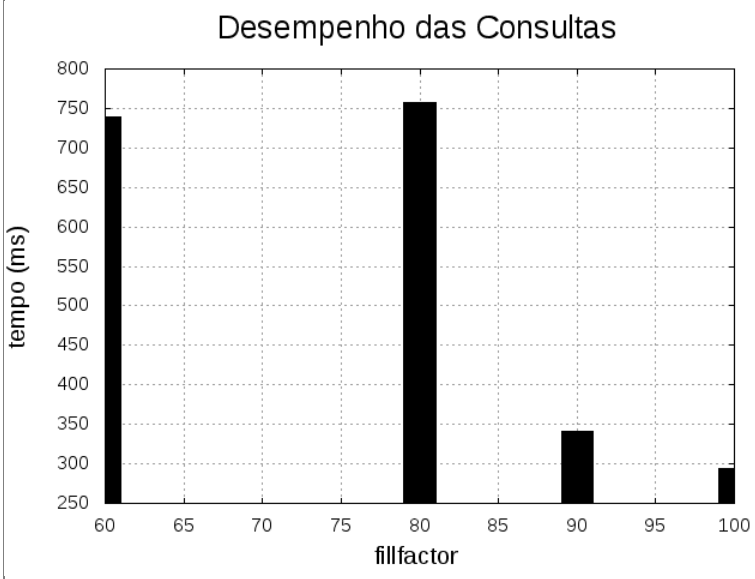
```
SELECT count(*) FROM t WHERE v=1;
```

Para a tabela com 10.000 tuplas:

fillfactor	tempo gasto para realizar uma consulta para um valor (ms)	tempo gasto para re-criar um índice para o atributo 'v' (ms)
60	3,585	157,827
80	2,926	157,776
90	2,925	159,235
100	2,734	160,604

Para a tabela com 10.000.000 tuplas:

fillfactor	tempo gasto para realizar uma consulta para um valor (ms)	tempo gasto para re-criar um índice para o atributo 'v' (ms)
60	739,597	17679,004
80	757,435	17323,549
90	340,481	17772,883
100	294,379	20873,345



Dos dados coletados (expostos acima) é possível concluir que, quanto menos espaço livre é deixado (aumento do *fillfactor*) uma consulta simples sobre um atributo indexado é realizada em menos tempo, independente do número de tuplas que serão retornadas pela consulta. Pode-se observer a que alterando o fator de preenchimento de 60 para 100 obtemos mais de 100% de redução no tempo de execução da consulta. O inverso acontece com o tempo gasto na reconstrução do índice, i.e., quanto maior o *fillfactor*, mais alto será o tempo de execução.

IComp/UFAM - Bancos de Dados 1 – 2017/02
Ficha de Resposta do Trabalho Prático 3

Atividade: 3	**Tarefa:** 16	**Data:** 28/10	**Folha:** 1 **de** 1

Aluno: Micael Levi **Matrícula:** 21554923
Aluno: Moisés Gomes **Matrícula:** 21550188

As querys utilizadas na criação dos índices foram:

- `CREATE INDEX tt ON actor (id, name);`
- `CREATE INDEX yy ON casting (movieid, actorid, ord);`
- `CREATE INDEX uu ON movie (title, yr, score, votes);`

As consultas realizadas foram:

- `SELECT COUNT(*) FROM actor WHERE id=1000 AND name='x';`
- `SELECT COUNT(*) FROM casting WHERE movieid=1000 AND actorid=1000 AND ord=1;`
- `SELECT COUNT(*) FROM movie WHERE title='x' AND yr=1000 AND score=8.8 AND votes=1000;`

Tabela	Tempo de execução da consulta (ms)	Número de tuplas	Número de colunas no índice
actor	0,043	1000	2
casting	0,036	3808	3
movie	0,040	1000	4

IComp/UFAM - Bancos de Dados 1 – 2017/02
Ficha de Resposta do Trabalho Prático 3

Atividade: 3	**Tarefa:** 17	**Data:** 27/10	**Folha:** 1 **de** 1

Aluno: Micael Levi **Matrícula:** 21554923
Aluno: Moisés Gomes **Matrícula:** 21550188

Tarefa 12 refeita de acordo com o que se pede:

```
postgres=# SELECT relname, relpages, reltuples FROM pg_class WHERE relname='t';
 relname | relpages | reltuples
---------+----------+-----------
 t       |    44248 |    1e+07
(1 row)
```

44284 páginas com blocos foram criadas

Tarefa 13 refeita de acordo com o que se pede:

```
-[ RECORD 1 ]----------
+------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------
schemaname             | public
tablename              | t
attname                | k
inherited              | f
null_frac              | 0
avg_width              | 4
n_distinct             | -1
most_common_vals       |
most_common_freqs      |
histogram_bounds       |
{182,110551,211880,303876,387320,484897,591927,702629,789915,885853,984262,1080896,1176484,1275515,1368756,1455099,1559080,1667473,1768210,18
78795,1978318,2066257,2155661,2258906,2361893,2459810,2575369,2667884,2766953,2866257,2961187,3062700,3173017,3266715,3358478,3456497,3568467
,3668514,3758794,3844804,3944929,4040108,4134493,4230934,4328137,4439530,4557691,4662497,4753318,4860526,4967657,5088775,5196992,5302290,5404
259,5495202,5594029,5690386,5795225,5894025,5985304,6084598,6174208,6281931,6383357,6470063,6565486,6670944,6784729,6889349,6997241,7100593,7
191342,7293890,7392952,7497782,7587366,7682495,7796132,7903325,7992969,8092434,8200228,8291058,8402228,8489977,8589113,8685962,8779371,887713
1,8987149,9095152,9194705,9287129,9394033,9492732,9596000,9691983,9788451,9897995,9999996}
correlation            | 1
most_common_elems      |
most_common_elem_freqs |
elem_count_histogram   |
-[ RECORD 2 ]----------
+------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------
schemaname             | public
tablename              | t
attname                | v
inherited              | f
null_frac              | 0
avg_width              | 4
n_distinct             | 10
most_common_vals       | {4,6,0,7,9,5,3,1,2,8}
most_common_freqs      | {0.100933,0.100633,0.100467,0.100333,0.100333,0.100233,0.0997,0.0994667,0.0993333,0.0985667}
histogram_bounds       |
correlation            | 0.0967634
most_common_elems      |
most_common_elem_freqs |
elem_count_histogram   |
```

Tarefa 14 refeita de acordo com o que se pede:

```
\timing
CREATE INDEX v_idxd ON t (v DESC NULLS FIRST);
SELECT count(*) FROM t WHERE v=1;
REINDEX INDEX v_idxd;
```

quantidade de tuplas	tempo gasto para realizar uma consulta para um valor (ms)	tempo gasto para re-criar um índice para o atributo 'v' (ms)
10.000	8,235	168,334
10.000.000	1001,090	19935,746

Ati%idade& 4	Tare'a& 19	Data& 27/10	Fo!ha& 1 de 1

A!(no& Micael Levi Matr)c(!a& 21554923
A!(no& Moisés Gomes Matr)c(!a& 21550188

*a+

```
postgres=# EXPLAIN ANALYZE SELECT * FROM movie WHERE votes>40000;
                                    QUERY PLAN
-------------------------------------------------------------------------------
Index Scan using movie_votes on movie  (cost=0.28..8.42 rows=8 width=30)
      (actual time=0.003..0.004 rows=4 loops=1)
   Index Cond: (votes > 40000)
Planning time: 0.142 ms
Execution time: 0.018 ms
(4 rows)
```

Nesta consulta o índice ‘movie_votes’ foi utilizado e 4 tuplas foram retornadas.

* +

```
postgres=# EXPLAIN ANALYZE SELECT * FROM movie WHERE votes>=1000;
                                  QUERY PLAN
-------------------------------------------------------------------------------
Seq Scan on movie  (cost=0.00..38.05 rows=1515 width=30)
                   (actual time=0.015..0.429 rows=1518 loops=1)
   Filter: (votes >= 1000)
   Rows Removed by Filter: 326
Planning time: 0.087 ms
Execution time: 0.525 ms
(5 rows)
```

Nesta consulta foi realizada uma varredura sequencial e 1518 tuplas foram retornadas (mais de 82% do total).

*c+ Uma consulta que (possivelmente) recuperará uma quantia considerada pequena (pelas estatísticas do SGBS) influencia na escolha de busca no índice criado pois provavelmente os dados que serão retornados estarão lá. O oposto acontece quando o sistema percebe que a quantidade de registros que serão retornados é provavelmente superior ao que o índice armazena.

IComp/UFAM - Bancos de Dados 1 – 2017/02
Ficha de Resposta do Trabalho Prático 3

Atividade: 4	**Tarefa:** 20	**Data:** 27/10	**Folha:** 1 **de** 1

Aluno: Micael Levi **Matrícula:** 21554923
Aluno: Moisés Gomes **Matrícula:** 21550188

a)

```
EXPLAIN SELECT title FROM movie WHERE votes>=(SELECT MAX(votes) FROM movie);

                            QUERY PLAN
-------------------------------------------------------------------------------
 Index Scan using movie_votes on movie  (cost=0.62..35.38 rows=615 width=16)
   Index Cond: (votes >= $1)
   InitPlan 2 (returns $1)
     ->  Result  (cost=0.33..0.34 rows=1 width=0)
           InitPlan 1 (returns $0)
             ->  Limit  (cost=0.28..0.33 rows=1 width=4)
                   ->  Index Only Scan Backward using movie_votes on movie
movie_1  (cost=0.28..94.55 rows=1844 width=4)
                         Index Cond: (votes IS NOT NULL)
(8 rows)
Time: 0,613 ms




EXPLAIN SELECT title FROM movie WHERE votes>=ALL(SELECT votes FROM movie);

                                  QUERY PLAN
-------------------------------------------------------------------------------
 Seq Scan on movie  (cost=0.00..43620.99 rows=922 width=16)
   Filter: (SubPlan 1)
   SubPlan 1
     ->  Materialize  (cost=0.00..42.66 rows=1844 width=4)
           ->  Seq Scan on movie movie_1  (cost=0.00..33.44 rows=1844
width=4)
(5 rows)
Time: 0,623 ms
```

b)
Sim. Cada uma trata a consulta de um jeito diferente. Usando “\timing” foi possível perceber que a primeira consulta teve um tempo de execução menor. Em relação ao custo, a primeira teve 0,28..94,55 enquanto que a segunda teve um custo de 0,00..42,66. Então pode-se inferir que a primeira consulta é mais eficiente em relação a tempo de execução. No entanto, em relação ao custo a segunda consulta é mais eficiente.

IComp/UFAM - Bancos de Dados 1 – 2017/02
Ficha de Resposta do Tra a!ho "r#tico $

Ati%idade& 4	Tare'a& 21	Data& 29/10	Fo!ha& 1 de 1

A!(no& Micael Levi Matr)c(!a& 21554923
A!(no& Moisés Gomes Matr)c(!a& 21550188

*a+

1.

```
EXPLAIN ANALYZE
SELECT title
FROM movie
WHERE votes > (SELECT votes FROM movie WHERE title = ©Star Wars©);
                                  QUERY PLAN
-------------------------------------------------------------------------------
 Index Scan using movie_votes on movie  (cost=8.57..43.34 rows=615 width=16)
                                        (actual time=11.499..11.499 rows=0 loops=1)
   Index Cond: (votes > $0)
   InitPlan 1 (returns $0)
     ->  Index Scan using movie_title on movie movie_1  (cost=0.28..8.29 rows=1 width=4)
                                                        (actual time=11.494..11.494 rows=1 loops=1)
           Index Cond: ((title)::text = 'Star Wars'::text)
 Planning time: 0.092 ms
 Execution time: 11.518 ms
(7 rows)
```

2.

```
EXPLAIN ANALYZE
SELECT m1.title
FROM movie m1, movie m2
WHERE m1.votes > m2.votes AND m2.title = ©Star Wars©;
                                  QUERY PLAN
-------------------------------------------------------------------------------
 Nested Loop  (cost=0.56..49.49 rows=615 width=16)
              (actual time=0.022..0.022 rows=0 loops=1)
   ->  Index Scan using movie_title on movie m2  (cost=0.28..8.29 rows=1 width=4)
                                                 (actual time=0.015..0.015 rows=1 loops=1)
         Index Cond: ((title)::text = 'Star Wars'::text)
   ->  Index Scan using movie_votes on movie m1  (cost=0.28..35.04 rows=615 width=20)
                                                 (actual time=0.003..0.003 rows=0 loops=1)
         Index Cond: (votes > m2.votes)
 Planning time: 0.106 ms
 Execution time: 0.043 ms
(7 rows)
```

* + Sim, a primeira possui o menor tempo de planejamento. Porém, o seu tempo de execução é muito superior ao da segunda consulta. Um dos fatores para isso ocorrer está no fato de que na segunda há o uso de dois índices, enquanto na primeira o mesmo não ocorre.

IComp/UFAM - Bancos de Dados 1 – 2017/02
Ficha de Resposta do Tra a!ho "r#tico $

Ati%idade& 4	Tare'a& 22	Data& 27/10	Fo!ha& 1 de 1

A!(no& Micael Levi Matr)c(!a& 21554923
A!(no& Moisés Gomes Matr)c(!a& 21550188

*a+

1. EXPLAIN ANALYZE SELECT title FROM movie WHERE title LIKE ©I%©;

```
                                QUERY PLAN
-------------------------------------------------------------------------------
Seq Scan on movie  (cost=0.00..38.05 rows=18 width=16)
                   (actual time=0.012..0.271 rows=25 loops=1)
   Filter: ((title)::text ~~ 'I%'::text)
   Rows Removed by Filter: 1819
 Planning time: 0.064 ms
 Execution time: 0.288 ms
(5 rows)
```

2. EXPLAIN ANALYZE SELECT title FROM movie WHERE substr(title, 1, 1) = ©I©;

```
                                QUERY PLAN
-------------------------------------------------------------------------------
 Seq Scan on movie  (cost=0.00..42.66 rows=9 width=16)
                    (actual time=0.014..0.438 rows=25 loops=1)
   Filter: (substr((title)::text, 1, 1) = 'I'::text)
   Rows Removed by Filter: 1819
 Planning time: 0.051 ms
 Execution time: 0.452 ms
(5 rows)
```

3. EXPLAIN ANALYZE SELECT title FROM movie WHERE title LIKE ©%A©;

```
                                QUERY PLAN
-------------------------------------------------------------------------------
 Seq Scan on movie  (cost=0.00..38.05 rows=18 width=16)
                    (actual time=0.013..0.312 rows=30 loops=1)
   Filter: ((title)::text ~~ '%A'::text)
   Rows Removed by Filter: 1814
 Planning time: 0.073 ms
 Execution time: 0.327 ms
(5 rows)
```

* + A primeira possui menor tempo de execução pois a comparação se dá no primeiro caractere/byte da cadeia de caracteres e por retonar uma quantidade inferior de tuplas em relação à última consulta (apesar do custo de planejamento ser superior ao da segunda consulta).

*c+ Como pode ser visto na saída do EXPLAIN de todas as consultas, uma varredura sequencial foi realizada sobre a tabela ‘movie’, ou seja, em nenhuma delas o índice foi utilizado. Isso acontece por que esse tipo de consulta (comparação com atributos texto) são imprevisíveis, o índice só será efetivo após as estatísticas apontarem uma alta quantidade de consultas que retornam conjuntos repetidos de tuplas.

IComp/UFAM - Bancos de Dados 1 – 2017/02
Ficha de Resposta do Tra a!ho "r#tico $

Ati%idade& 4	Tare'a& 23	Data& 29/10	Fo!ha& 1 de 1

A!(no& Micael Levi Matr)c(!a& 21554923
A!(no& Moisés Gomes Matr)c(!a& 2155018

*a+

1. EXPLAIN ANALYZE SELECT title FROM movie WHERE votes < 1000;

```
                                QUERY PLAN
-------------------------------------------------------------------------------
 Index Scan using movie_votes on movie  (cost=0.28..20.04 rows=329 width=16)
                                   (actual time=0.018..0.180 rows=326 loops=1)
   Index Cond: (votes < 1000)
 Planning time: 0.083 ms
 Execution time: 0.226 ms
(4 rows)
```

2. EXPLAIN ANALYZE SELECT title FROM movie WHERE votes > 40000;

```
                                QUERY PLAN
-------------------------------------------------------------------------------
 Index Scan using movie_votes on movie  (cost=0.28..8.42 rows=8 width=16)
                                  (actual time=0.008..0.010 rows=4 loops=1)
   Index Cond: (votes > 40000)
 Planning time: 0.132 ms
 Execution time: 0.033 ms
(4 rows)
```

* + Na primeira query, o número de tuplas foi 326, um valor quase 82 vezes maior que o selecionado pela segunda query. Como a segunda consulta é muito menos seletiva, o seu tempo de planejamento é maior. Assim o seu tempo de execução é inferior – apesar de ambos os planos utilizarem a mesma estratégia.