Universidade Federal do Amazonas
Instituto de Computação
Bancos de Dados I – 2025/02
Professor: Altigran Soares da Silva <alti@icomp.ufam.edu.br>

Trabalho Prático 3: 17, 18 e 19 de novembro de 2025

As tarefas especificadas abaixo devem ter seus resultados apresentados em um Jupyter Notebook sendo que cada tarefa deve ser apresentada em uma célula.

PARTE I

Esta primeira parte do trabalho tem dois objetivos principais. O primeiro é verificar alguns parâmetros do hardware e do software utilizado e o segundo é que os estudantes preparem e se familiarizar com o ambiente do PostgreSQL

Tarefa 1 – Identificação do Sistema

Identifique o sistema que será usado para os experimentos, incluindo informações sobre o hardware e o Sistema Operacional utilizado. Sobre o tipo de processador, quantidade de memória RAM, tamanho do disco. Devem ser também apresentadas informações sobre as caches existentes. Sobre o Sistema Operacional, que devem ser Linux, incluir informações sobre qual a distribuição usada, versão do sistema, versão do Kernel, etc.

O que entregar: As informações pedidas devem ser apresentadas no jupyter notebook

Tarefa 2 - Verificação de parâmetros de armazenamento

a) Verifique no disco que será usado para os experimentos no laboratório os seguintes parâmetros: Nr. de superfícies, cilindros, setores por trilha, velocidade de rotação, latência rotacional; tempos de seek médio, máximo e mínimo; tempo para a próxima trilha; e taxa de transferência.

b) Utilizando o comando “stat” do Linux, verifique os parâmetros dos parâmetros de S.O. que serão utilizados para o disco.

c) Verifique o tamanho de bloco utilizado e mostre como alterar o tamanho dos blocos

O que entregar: Os resultados de cada verificação devem ser preenchidas no jupyter notebook

Tarefa 3 – Geração de um BD para testes

Essa tarefa consiste na criação de banco de dados e povoamento destas tabelas com dados sintéticos. Para a definição do esquema das tabelas e os dados a serem carregados usaremos a especificação e os utilitários fornecidos pelos Benchmark TPC-H (http://www.tpc.org/tpch/).

Para geração do BD no PostgreSQL, siga as instruções disponíveis em:

https://github.com/foliveirafilho/tpch-pgsql

O que entregar: Devem ser apresentadas as saídas da execução dos scripts de geração.

Tarefa 3 – Execução de Consultas

A segunda tarefa deste trabalho consiste em executar e analisar um conjunto de consultas analíticas do benchmark TPC-H sobre o banco de dados gerado na Tarefa 1, observando o comportamento do PostgreSQL em diferentes tipos de operações (filtros, junções, agregações e ordenações). O objetivo é compreender como o otimizador escolhe planos de execução e medir o tempo de resposta de consultas típicas de um sistema de apoio à decisão.

O TPC-H define 22 consultas padronizadas (Q1 a Q22), mas para este trabalho será utilizado um subconjunto representativo composto por 8 consultas que abrangem diferentes padrões de acesso.

Consultas a executar: Q1, Q3, Q5, Q6, Q7, Q9, Q10 e Q12. As consultas originais podem ser obtidos no repositório oficial do benchmark. Essas consultas devem ser executadas no mesmo banco TPC-H criado na Tarefa 1, utilizando o PostgreSQL.

Para cada consulta:

- Execute o comando SQL completo no PostgreSQL.
- Mostre as 10 primeiras linhas do resultado (ou um trecho representativo).
- Execute o comando EXPLAIN ANALYZE antes da consulta para exibir o plano de execução e o tempo real medido O comando EXPLAIN ANALYZE mostra as etapas internas do plano escolhido pelo otimizador, o número de linhas processadas e o tempo de cada operação.
- Registre o tempo total de execução e comente brevemente o tipo de acesso observado (por exemplo, varredura sequencial, uso de índice, junções em hash, etc.).

O que entregar: Devem ser apresentados os resultados acima

PARTE II

O objetivo desta parte do trabalho é analisar o comportamento dos índices das tabelas do SGBD através do exame e análise das tabelas de estatísticas para consultas SQL sobre uma tabela criada com dados aleatórios.

Tarefa 5 – Preparação da Tabela Aleatória

Criar uma tabela com uma chave simples e alguns dados de exemplo. Cada valor de chave é um número incremental e está associado a com valores que variam de 0 até 10:

```
DROP TABLE IF EXISTS t;
CREATE TABLE t (k serial PRIMARY KEY, v integer);

INSERT INTO t(v)
SELECT trunc(random() * 10)  FROM generate_series(1,100000);
```

O que entregar: Imprimir os valores das 10 primeiras tuplas da tabela, ordenando por k.

Tarefa 6 – Páginas criadas

Verifique quantas páginas com blocos foram criadas para a tabela da Tarefa 5.

Commando: **SELECT relname, relpages, reltuples FROM pg_class WHERE relname='t';**

O que entregar: Imprimir o resultado do comando SQL.

Tarefa 7 – Blocos

Verifique quantos blocos foram efetivamente usados numa consulta

Comando:
SELECT pg_sleep(1);
\pset x on
SELECT * FROM pg_stats WHERE relname='t';
SELECT pg_stat_reset();
\pset x off

Observação: Em algumas versões do PostgreSQL, o atributo é chamado de **tablename** em vez de **relname**.

O que entregar: Imprimir o resultado do comando SQL.

Tarefa 8 – Índice

Crie um índice para o atributo ‘v’ e realize consultas e criação de índice

Qual o tempo gasto para realizar uma consulta para um valor (tendo a tabela 100000 tuplas)?
Qual o tempo gasto para recriar um índice para o atributo ‘v’?

Remova a tabela ‘t’ e crie novamente com 1.0000.000 de tuplas

Qual o tempo gasto para realizar uma consulta para um valor específico?
Qual o tempo gasto para recriar um índice para o atributo ‘v’?

O que entregar: Relatório com o resultado das perguntas

Tarefa 9 - Fill factor

Quando se cria um novo índice, nem toda entrada no bloco do índice é usada. Um espaço livre é deixado, conforme o parâmetro **fillfactor**.

Crie novos índices usando fillfactor=60,80,90 e 100. Analise o desempenho de suas consultas usando as mesmas condições da Tarefa 12

ALTER TABLE foo SET (fillfactor = 50);
VACUUM FULL foo;

O que entregar: Relatório com o resultado das perguntas

Tarefa 10 - Utilize índices com ordem DESC

Repita os testes das Tarefas 8 e 9 usando índices descendentes. Avalie e registre o resultado

Comando: **CREATE INDEX i ON t(v DESC NULLS FIRST);**

O que entregar: Relatório com o resultado da avaliação e uma análise dos resultados.

Parte III. O objetivo desta parte do trabalho é estudar o comportamento dos otimizadores de consulta dos SGBDs através do exame e análise dos planos de execução para consultas SQL sobre tabelas que serão fornecidos. Será bastante utilizado o comando EXPLAIN ANALYZE, que permite visualizar todas as etapas envolvidas no processamento de uma consulta. Usaremos para isso a tabela "movies".

Tarefa 11. Preparação e Verificação do Ambiente

a) Execute o script movie.sql em movies para criar as tabelas e índices e carregar os dados necessários às próximas atividades
b) Verifique no catálogo do banco de dados os seguintes metadados sobre os índices associados às tabelas e apresente-os no relatório: Nome do índice, nome da tabela, altura, número máximo de chaves por bloco, número médio de chaves por bloco, número de blocos folha, número de médio de blocos folha por chave, número médio de blocos de dados por chave, número de linhas e número de chaves distintas.

O que entregar: Relatório com os resultados da verificação

Tarefa 12. Consultas por intervalo e índices secundários

a) Escreva uma consulta em SQL sobre o atributo VOTES da tabela MOVIE que recupera um número pequeno de tuplas (<10 tuplas); Execute o comando EXPLAIN ANALYZE sobre esta consulta e apresente os resultados.
b) Escreva uma consulta em SQL sobre o atributo VOTES da tabela MOVIE que recupera um número grande de tuplas (>80% das tuplas). Execute o comando EXPLAIN ANALYZE sobre esta consulta e apresente os resultados.
c) Explique porque o índice sobre VOTES não é sempre usado nas consultas sobre este atributo.

O que entregar: Relatório com as respostas das questões.

Tarefa 13. Comparações de operadores de agregação.

Considere as seguintes consultas em SQL, sobre o atributo VOTES, as quais são equivalentes:

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

a) Apresente o resultado do comando EXPLAIN ANALYZE sobre as duas consultas acima
b) Existe alguma diferença entre os planos de consultas? Qual das duas é mais eficiente? Explique?

O que entregar: Relatório com as respostas das questões.

Tarefa 14. Consultas com Junção e Seleção

Considere as duas consultas equivalentes em SQL a seguir, as quais retornam os filmes com mais votos que “Star Wars”

- SELECT title FROM movie WHERE votes > (SELECT votes FROM movie WHERE title = 'Star Wars');
- SELECT m1.title FROM movie m1, movie m2 WHERE m1.votes > m2.votes AND m2.title = 'Star Wars';

a) Apresente o resultado do comando EXPLAIN ANALYZE sobre as duas consultas acima
b) Existe alguma diferença entre os planos de consultas? Qual das duas é mais eficiente? Explique?

O que entregar: Relatório com as respostas das questões.

Tarefa 15. Casamento de Strings e Índices

Considere as seguintes consultas SQL sobre o atributo TITLE usando o operador LIKE.

- SELECT title FROM movie WHERE title LIKE 'I%';
- SELECT title FROM movie WHERE substr(title, 1, 1) = 'I';
- SELECT title FROM movie WHERE title LIKE '%A';

a) Apresente o resultado do comando explain sobre as três consultas acima
b) Qual das três apresenta o menor custo? Porque?
c) O índice sobre TITLE foi usado para todas elas? Justifique.

O que entregar: Relatório com as respostas das questões.

Tarefa 16. Verificação da hipótese de distribuição uniforme na estimativa de seletividade

Considere as seguintes consultas sobre o atributo TITLE da tabela MOVIE

- SELECT title FROM movie WHERE votes < 1000;
- SELECT title FROM movie WHERE votes > 40000

a) Apresente o resultado do comando explain sobre as duas consultas acima. Explique o resultado.
b) Compare o número de tuplas selecionadas por cada consulta. Qual das duas tem a menor seletividade?

O que entregar: Relatório com as respostas das questões.

PARTE IV

O objetivo desta parte do trabalho é experimentar estratégias para utilização de transações e níveis de isolamento em SGBDs relacionais. As tarefas envolvem uma simulação de um sistema de reservas de passagem áreas.

Considere a seguinte tabela que registra os assentos reservados em um vôo:

`Assentos(num_voo,disp)`

onde **`num_voo`** um número inteiro de 1 a 200 e **`disp`** é um atributo booleano cujo valor é **`true`** se o assento estiver vago e **`false`** caso contrário. O valor inicial é **`true`**.

A reserva de um assento é feita em três passos:

Passo 1. O sistema recupera a lista dos assentos disponíveis.
Passo 2. O cliente escolhe o assento. Esse passo deve ser simulado pela escolha aleatória de um dos assentos disponíveis, levando para isso um tempo de escolha de 1 segundo.
Passo 3. O sistema registra a reserva do assento escolhido, atualizando o valor de **`disp`** para **`false`**.

Cada assento é reservado individualmente. Duas versões diferentes do processo de reserva devem ser implementadas.

Versão a. A reserva é implementada como uma única transação que inclui os três passos acima.
Versão b. A reserva inclui uma transação para o Passo 1 e outra para o Passo 3. O Passo 2 não faz parte das transações, mas deve ser executado.

Agentes de viagens são responsáveis por realizar as reservas de 200 clientes no total. A atividade de um agente de viagens é simulada por uma *thread*.

Experimentos devem ser realizados simulando a atuação de *k* agentes de viagem trabalhando simultaneamente, onde k = 1,2,4,6,8 e 10. Cada agente/*thread* faz uma reserva de cada vez. As *threads* devem ser reiniciadas até que todos os 200 clientes tenham seus assentos reservados.

Dois conjuntos de experimentos devem ser feitos usando dois níveis de isolamento: *"read committed"* e *"serializable"*. Nos dois casos, o sistema deve ser configurado para realizar bloqueios a nível de tupla (linha).

As implementações devem ser feitas em Python3.8+ usando o SGBD PostgreSQL.

Considerando o descrito acima, execute as seguintes tarefas:

Tarefa 17. Implemente as versões a e b do processo de reserva.

O que entregar: Código fonte Python3.8+ documentado das duas versões.

Tarefa 18. Apresente gráficos de linha onde, para cada valor de *k* (número de agentes) no eixo x, temos no eixo y o tempo necessário para que todos os clientes efetuem suas reservas. Um gráfico diferente deve ser apresentado para cada par de versões da reserva e nível de isolamento.

Tarefa 19. Apresente uma tabela com o número máximo, mínimo e médio de vezes que um cliente teve que tentar reservar um assento até conseguir, ou seja, o número de vezes que uma reserva teve que ser refeita. A tabela considera as variações de k, versão de reserva e nível de isolamento.

Tarefa 20. Apresente uma análise dos resultados obtidos em cada versão de reserva e tipo de isolamento, explicando as diferenças entre resultados.