Universidade Federal do Amazonas - Instituto de Computação
Bancos de Dados I – 2017/02
Trabalho Prático 09/10/2017
Data da Entrega: 30/10/2017

As tarefas especificadas abaixo devem ter seus resultados apresentados somente em fichas cujo modelo foi distribuído em sala de aula e no blog. Qualquer outro formato será desconsiderado para fins de avaliação. O trabalho dever ser entregue **impresso** e não eletronicamente.

PARTE I

Esta primeira parte do trabalho tem dois objetivos principais. O primeiro é verificar alguns parâmetros do hardware e do software que vamos utilizar e o segundo é que os alunos preparem e se familiarizem com os ambientes dos sistemas PostgreSQL e SQLite.

Tarefa 1 – Instalação do PostgreSQL

O PostgreSQL é um SGBD objeto-relacional que possui recursos comuns à SGBDs de grande porte. Trata-se de um sistema versátil, robusto e livre. É possível utilizar o PostgreSQL em vários sistemas operacionais, dentre os quais o Windows e Linux, ou em qualquer sistema compatível com as especificações POSIX. Segundo informações do site oficial, o PostgreSQL permite a criação de bancos de dados de tamanho infinito. Cada tabela pode ter até 16 Terabytes sendo que cada tupla pode ter até 1,6 TB e cada atributo 1 GB. O sistema tem recursos como triggers, integridade referencial, entre outros, além de ser compatível com uma série de linguagens, tais como C, C++, PHP, Python, Java, Perl, etc. O uso do PostgreSQL é muito difundido, pois várias empresas perceberam que com ele podem criar BDs complexos sem a necessidade de gastar altos valores na aquisição de licenças. Outra vantagem, é que o PostgreSQL possui uma documentação muito abrangente, o que permite suporte adequado às necessidades.

A primeira tarefa deste trabalho consiste na instalação do PostgreSQL nos computadores do Laboratório. Os fontes e as instruções de instalação estão em: http://www.postgresql.org.

O que entregar: Deve ser apresentada saída do log de instalação.

Tarefa 2 – Instalação do SQLite

SQLite é uma pequena biblioteca C que implementa um banco de dados SQL completo, embutido e sem configurações. Programas que usam a biblioteca SQLite podem ter acesso a banco de dados SQL sem executar um processo de SGBD separado. SQLite não é uma biblioteca de cliente usada para conectar com um grande servidor de banco de dados. A própria biblioteca SQLite é o servidor. A biblioteca SQLite lê e escreve diretamente em arquivos de banco de dados no disco. Suas características incluem: Transações são atômicas, consistentes, isoladas e duráveis (ACID); Configuração-zero - nenhuma instalação ou administração necessária; Implementação da maior parte do padrão SQL92; Um BD completo é armazenado em apenas um arquivo de sistema; Arquivos de banco de dados podem ser livremente compartilhados entre máquinas com diferentes ordens de byte; Suporta bancos de dados de até 2 terabytes de tamanho; Tamanho de strings e BLOBs limitados apenas pela memória disponível; Mais rápido que SGBDs cliente/servidor populares para a maioria das operações comuns; API simples e fácil de usar; Auto-contido: sem dependências externas. Estas características o tornam ideal para desenvolver programas standalone, pequenos e médios sites, etc.

A segunda tarefa deste trabalho consiste na instalação da biblioteca nos computadores do Laboratório. Os fontes e as instruções de instalação estão em: http://www.sqlite.org.

O que entregar: Deve ser apresentada a saída de um pequeno programa em C que testa a execução da biblioteca.

Tarefa 3 – Geração de um BD de testes

A terceira tarefa deste trabalho consiste na criação de banco de dados em cada um dos sistemas e o 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:

http://dsl.cds.iisc.ac.in/projects/PICASSO/picasso_download/doc/Installation/tpch.htm

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

Tarefa 4 – Execução de Consultas

A quarta tarefa deste trabalho consiste na execução de uma série de consultas do Benchmark TPC-H nos sistemas instalados.

Usando as instruções disponíveis nas mesmas URLs acima, execute os passos referentes ao PostgreSQL. Adapte estes passos para o SQLite e execute as mesmas consultas neste sistema.

O que entregar: Esta tarefa será verificada no laboratório.

Tarefa 5 – Identificação do Sistema

Identifique o sistema que será usada para os experimentos incluindo informações sobre o hardware e o Sistema Operacional utilizado. Sobre o hardware, incluir informações como marca, modelo, número de série, tipo de processador, quantidade de memoria RAM, tamanho do disco. Devem ser também apresentadas informações sobre as caches existentes. Sobrehvvcnfefnpbjrpífexbjtepbjfnpbjf pbmbjfmpbmvfa

PARTE II

A segunda parte do trabalho visa verificar as características de organização física dos dados, registros, blocos e arquivos nos nossos dois sistemas alvo. Para tanto, realizaremos várias cargas experimentais de dados variando vários parâmetros de armazenamento e verificando a implicação destas variações no tempo despendido e no espaço ocupado.

Tarefa 7 – Analisar e descrever os detalhes de armazenamento físico de dados no PostgreSQL e no SQLite. Construir uma tabela comparativa das principais características de cada um dos dois sistemas. Utilize, se necessário, diagramas, gráficos, etc.

O que entregar: Relatório com o resultado da análise e descrição

Tarefa 8 – Analisar e descrever os detalhes dos seguintes sistemas de arquivo disponíveis no Linux: Ext2, Ext3, ReiserFS e XFS. Construir uma tabela comparativa das principais características de cada um dos dois sistemas. Utilize, se necessário, diagramas, gráficos, etc.

O que entregar: Relatório com o resultado da análise e descrição

Tarefa 9 – Re-executar a carga de dados no PostgreSQL e SQLlite para vários tipos de arquivos e configurações distintas.

Os arquivos utilizados devem ter as seguintes características:

- 1 arquivo de registros longos (>100 Kb) e com poucos registros (10 mil)
- 1 arquivo de registros curtos (<4 Kb) e com muitos registros (1 milhão)
- 1 arquivo de registros longos (>100 Kb) e com muitos registros (1 milhão)
- 1 arquivo de registros variáveis (ex. muitos NULLS) e com muitos registros (1 milhão)

Para a geração destes arquivos, modificar os scripts configurados na **Tarefa 3**

Para cada arquivo, devem ser usados os seguintes sistemas de arquivos: Ext2, Ext3 e XFS. Para cada carga, medir o tempo necessário para geração dos arquivos e espaço ocupado no disco.

O que entregar: Tabelas comparativas para cada SGBD, arquivo e sistema de arquivos usados em termos do tempo de execução e do espaço ocupado no disco.

Tarefa 10 – Analisar as tabelas a Tarefa 9 explicando os valores de tempo e espaço obtidos.

O que entregar: Relatório com o resultado da análise

PARTE III

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 as tabelas “movies”, “actors”, “casting”, e sobre uma tabela criada com dados aleatórios. Esta tarefa deverá ser executada somente com o PostegreSQL.

Tarefa 11 – Preparação de Tabela Exemplo

Criar uma tabela com uma chave simples e alguns dados de exemplo. Cada valor de chave é um número incremental e esta 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);.
```

Tarefa 12 – Páginas criadas

Verifique quantas páginas com blocos foram criadas para a tabela da Tarefa 11.

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

Tarefa 13 – 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
```

Tarefa 14 – Índice

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

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

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

Qual o tempo é gasto para realizar uma consulta para um valor específico?
Qual o tempo é gasto para re-criar um índice para o atributo ‘v’?

Tarefa 15 - Fill factor

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

Crie novos índices usando fillfactor=60,80,90 e 100. Analise o desempenho de suas consultas usando as mesmas condições da Tarefa 14

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

Tarefa 16 – Usando índice com múltiplas colunas

Use as tabelas de movies, actor e casting e realize a criação de índice que utilizem 2, 3 e 4 colunas:

Use a tabela com 1000 tuplas e relate o desempenho de suas consultas.
Use a tabela com > 100.000 tuplas e relate o desempenho de suas consultas.

Tarefa 17 - Utilize índices com ordem DESC

Repita os testes das Tarefas 12,13 e 14 usando índices descendentes. Avalie e registre na sua ficha do Lab.

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

Parte IV. 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, que permite visualizar todas as etapas envolvidas no processamento de uma consulta.

Tarefa 18. Preparação e Verificação do Ambiente

a) Execute o script movie.sql para criar as tabelas e índices e carregar os dados necessários às próximas atividades
b) Verifique no catalogo do banco de dados os seguintes meta-dados 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 19. Consultas por intervalo e índices secundários

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

Tarefa 20. 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 sobre as duas consultas acima
b) Existe alguma diferença entre os planos de consultas? Qual das duas é mais eficiente? Explique?

Tarefa 21. 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 sobre as duas consultas acima
b) Existe alguma diferença entre os planos de consultas? Qual das duas é mais eficiente? Explique?

Tarefa 22. 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 pra todas elas? Justifique.

Tarefa 23. 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?

PARTE V

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 C++ usando o SGBD PostgreSQL.

Considerando o descrito acima, execute as seguintes tarefas:

Tarefa 24. Implemente as versões a e b do processo de reserva.

O que entregar: Código fonte em C++ documentado das duas versões.

Tarefa 25. 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 efetivem suas reservas. Um gráfico diferente deve ser apresentado para cada par de versão da reserva e nível de isolamento.

Tarefa 26. Apresente uma tabela com o número máximo, mínimo e médio de vezes que um cliente deve que tentar reservar um assento até conseguir, ou seja, o número de vezes que uma reserva teve que ser refeita. A tabela de considera as variações de k, versão de reserva e nível de isolamento.

Tarefa 27. Apresente uma análise dos resultados obtidos em cada de versão de reserva e tipo de isolamento, explicando as diferenças entre resultados.