Universidade Federal do Ceará
Campus de Quixadá

Construção de Sistemas de Gerência de Banco de Dados

1º Lista de CSGBD

Aluna: Bárbara Stéphanie Neves
Professora: Lívia Almada

Março
2019

Universidade Federal do Ceará
Campus de Quixadá

Construção de Sistemas de Gerência de Banco de Dados

Lista Prática e Teórica

Relatório da 1ª Lista da disciplina de Construção de Sistemas de Gerência de Banco de Dados em que a 1º questão se trata da prática feita em sala de aula e as questões restantes são teóricas.

Aluna: Bárbara Stéphanie Neves

Matrículas: 388713

Professora: Lívia Almada

Curso: Ciência da Computação

Março
2019

Conteúdo

1 Respostas das Questões da Lista **1**

1.1 Prática 1

1.2 Porque você optaria usar um sistema de banco de dados ao invés de mantê-los em sistemas de arquivos? 5

1.3 Quando não é adequado o uso de sistemas de banco de dados? . . 6

1.4 Descreva a arquitetura de um SGBD. 6

1.5 Qual o papel do gerenciador de buffer? 7

1.6 Qual a estrutura básica dos discos magnéticos? 8

1.7 Considere um disco com o tamanho de setor igual a 512 bytes, 200 trilhas por superfície, 50 setores por trilha, 5 pratos de disco com dois lados. 8

1.8 Explique como é calculado o tempo de acesso a um bloco do disco. 9

1.9 Explique quais são propriedades de um frame. 10

1.10 Explique o que o gerenciador de buffer deve fazer para processar uma requisição de leitura por uma página. O que acontece se uma página requisitada está no buffer pool? 10

1.11 Quando um gerenciador de buffer escreve a página no disco? . . . 10

1.12 Quais são os algoritmos de paginação? Explique cada um deles, apresentando suas vantagens e desvantagens. 11

1 Respostas das Questões da Lista

1.1 Prática

(a) Crie o banco de dados ‘condominio’;

(b) Utilize o arquivo ‘2.DDL-condominio.sql’ para definir as tabelas;

(c) Utilize o arquivo ‘3.DML-condominio.sql’ para inserção de tuplas;

(d) Utilize o PSQL Console conectado ao banco de dados ‘condominio’;

(e) Descubra para cada tabela do banco de dados (acesse pg_class):

1. o nome da tabela (coluna: relname),
2. o tamanho em número de páginas (coluna: relpages),
3. o número de tuplas (coluna: reltuples),
4. se há ou não presença de índice (coluna: relhasindex);

O quadro abaixo contém todos os valores encontrados para cada tabela presente no banco de dados. A consulta feita para conseguir esses valores foi a seguinte:

SELECT relname, relpages, reltuples, relhasindex
FROM pg_class
WHERE relname = ‘nome_da_tabela’;

Relname	Relpages	Reltuples	Relhaindex
l01_morador	1	1	True
l02_unidade	0	0	True
l03_veiculo	0	0	True
l04_familiar	0	0	True
l05_prestador	0	0	True
l06_requisita	0	0	True
l07_administracao	0	0	True
l08_despesa	0	0	True
l09_funcionario	0	0	True
l10_permanente	0	0	True
l11_terceirizado	0	0	True
l12_contrata	0	0	True

(f) Você percebeu algo estranho com o número de tuplas e páginas reportadas como resposta do item anterior em relação ao que você inseriu no item c? Como forçar o SGBD a reavaliar as informações de metadados desatualizados? Execute o comando de correção.

Percebi que em todas as tabelas, com exceção da l01_morador, o número de páginas e tuplas estão zerados, o que não faz sentido, porque o **item (c)** pede para que várias informações sejam inseridas no BD e para todas as 12 tabelas pertencentes a ele.

Para forçar o SGBD a reavaliar as informações de metadados desatualizados, podemos utilizar dois comandos:

- O **comando VACUUM** recuperará o espaço usado pelos dados que foram atualizados. Como estou usando o PostgreSQL, as tuplas de valor-chave atualizadas não são removidas das tabelas quando as linhas são alteradas, portanto, o comando VACUUM deve ser executado ocasionalmente para fazer isso.
- Já o **comando ANALYZE** reúne estatísticas para que o planejador de consulta possa criar os caminhos de execução mais eficientes. De acordo com a documentação do PostgreSQL, as estatísticas precisas me ajudarão a escolher o plano de consulta mais apropriado e, assim, melhorar a velocidade do processamento de consultas.

No caso, para esse problema, fiz a execução do seguinte comando para todas as tabelas do banco:

VACUUM(FULL VERBOSE ANALYZE) nome_da_tabela;

Utilizei acima também o **comando VERBOSE** para que mostrasse a saída de progresso detalhada.

(g) Agora reporte novamente os valores questionados na letra e;

Relname	Relpages	Reltuples	Relhaindex
l01_morador	1	9	True
l02_unidade	1	8	True
l03_veiculo	1	14	True
l04_familiar	1	18	True
l05_prestador	1	10	True
l06_requisita	1	30	True
l07_administracao	1	12	True
l08_despesa	1	38	True
l09_funcionario	1	12	True
l10_permanente	1	6	True
l11_terceirizado	1	6	True
l12_contrata	1	16	True

(h) Utilize o comando: select pg_stat_reset(). Este é responsável por resetar algumas informações estatísticas para zero.

(i) Escolha uma tabela do seu banco de dados (acesse: pg_stat_all_tables) reporte:

1. o nome da tabela(coluna: relname),
2. o tamanho em quantidade de leituras sequenciais já realizadas (coluna: seq_scan),
3. o número de tuplas inseridas (coluna: n_tup_ins),
4. número de tuplas modificadas (coluna: n_tup_upd),
5. número de tuplas deletadas (coluna: n_tup_del);

Tabela escolhida: **l08_despesa**.

Relname	Seq_scan	N_tup_ins	N_tup_upd	N_tup_del
l08_despesa	0	0	0	0

(j) Realize um comando de inserção, remoção, consulta e modificação na tabela que você escolheu no item anterior. Se você percebeu algum erro ocorrido, não o esqueça!! Execute novamente o que foi questionado no item anterior e reporte.

Comandos

- de inserção: **INSERT INTO** L08_DESPESA (DNRO, DTIPO, DFUNCAOADMIN, DANOADMIN)
  VALUES (‘41’, ‘gas’, ‘sindico’, ‘2011’);
- de remoção: **DELETE** FROM L08_DESPESA
  WHERE DNRO = ‘40’;
- de consulta: **SELECT** L08.DFUNCAOADMIN
  FROM L08_DESPESA L08
  WHERE L08.DTIPO = ‘agua’;
- de modificação: **UPDATE** L08_DESPESA
  SET DANOADMIN = ‘2011’
  WHERE DNRO = ‘39’;

Relname	Seq_scan	N_tup_ins	N_tup_upd	N_tup_del
l08_despesa	6	1	1	1

(k) A nível de banco de dados (acesse: pg_stat_database) e reporte para o banco de dados condominio (datname = ‘condominio’):

1. o nome do banco número de blocos lidos (coluna: blks_read),
2. número de encontrado no buffer cache (coluna: blks_hit),
3. número de reporte para o banco de dados de dados (coluna: datname),
4. vezes em que um bloco foi consultas canceladas devido a conflitos nesse banco de dados (coluna: conflicts),
5. número de deadlocks (coluna: deadlocks),
6. número de transações comitadas (coluna: xact_commit)
7. e número de transações desfeitas (coluna: xact_rollback);

Comando feito:

SELECT blks_read, blks_hit, datname, conflicts, deadlocks, xact_commit, xact_rollback
FROM pg_stat_database
WHERE datname = ‘condominio’;

Blks_read	Blks_hit	Datname	Conflicts	Deadlocks	Xact_commit	Xact_rollback
8	2953	condominio	0	0	1601	3

(l) Ocorreu algum rollback no banco de dados? Além disso, como o SGBD pode se valer dos metadados destacados no item anterior para tomar avaliar sua politica de substituição de páginas e controle de concorrência?

Sim, ocorreram 3 rollbacks.

Sobre a segunda pergunta: não sei.

1.2 Porque você optaria usar um sistema de banco de dados ao invés de mantê-los em sistemas de arquivos?

Porque sistemas de arquivos possuem algumas das desvantagens descritas abaixo.

- Dependendo da aplicação, é quase certo que ela não terá memória suficiente para armazenar todos os dados, ou seja, será necessário um dispositivo de armazenamento, como disco ou fita, além de carregar partes relevantes dos dados na memória principal conforme necessário;
- Pode ser preciso programar algum método de identificação de todos os itens de dados, porque se o sistema computacional da aplicação tiver 32 bits de endereçamento, por exemplo, não tem como referir diretamente a mais do que aproximadamente 4 GB de dados;
- Também pode ser necessário escrever programas especiais para responder a cada pergunta que um usuário pode desejar fazer sobre os dados. Esses programas provavelmente serão complexos em razão do grande volume de dados a ser pesquisado;
- Também é útil proteger os dados de alterações inconsistentes realizadas por usuários diferentes acessando os dados de forma concorrente. Se os aplicativos devem tratar dos detalhes desse acesso concorrente, isto aumenta significativamente a sua complexidade;
- É preciso assegurar que os dados sejam restaurados a um estado consistente se o sistema falhar, enquanto as alterações estão sendo realizadas;
- E, os sistemas operacionais provêm apenas um mecanismo de senha para segurança. Isso não é suficientemente flexível para reforçar as políticas de segurança nas quais usuários diferentes têm permissão de acessar subconjuntos diferentes de dados.

Já um SGBD é um pacote de software projetado para executar mais facilmente as tarefas mencionadas anteriormente. Armazenando-se dados no SGBD em vez de em uma coleção de arquivos do sistema operacional, é possível utilizar os recursos do SGBD para gerenciar os dados de uma forma robusta e eficiente. À medida que cresce o volume de dados e o número de usuários, o suporte de um SGBD torna-se indispensável.

1.3 Quando não é adequado o uso de sistemas de banco de dados?

Um SGBD é um software complexo, otimizado para alguns tipos de processamento, como responder a perguntas complexas ou tratar várias requisições concorrentes, e o seu desempenho pode não ser adequado para determinados aplicativos especializados.

Exemplos:

- aplicativos que possuem restrições rígidas de tempo real;
- operações críticas bem definidas para as quais um código customizado eficiente pode ser escrito;
- e, aplicativos que precisam manipular os dados de maneiras não suportadas pela linguagem de consulta, como os BDs relacionais, já que não suportam a análise flexível de dados textuais.

Resumindo: se desempenho especializado ou solicitações de manipulação de dados são essenciais num aplicativo, pode-se optar por não utilizar um SGBD, especialmente se os benefícios de um SGBD (consulta flexível, segurança, acesso concorrente e recuperação de falha) não forem exigidos.

1.4 Descreva a arquitetura de um SGBD.

- O SGBD aceita comandos SQL gerados de uma variedade de interfaces de usuário, como formulários Web, front-ends de aplicativos, ou uma interface SQL, em que produz planos de avaliação de consulta, executa estes planos no banco de dados, e retorna as respostas.
- Quando um usuário emite uma consulta, ela que foi **sintaticamente analisada** (pelo **Analisador Sintático**) é apresentada a um **otimizador de consulta** (**Otimizador**), que usa a informação sobre como o dado é apresentado para produzir um plano de execução (**Executor de Plano**) eficiente para processar a consulta. Esse plano de execução é um projeto para processar

uma consulta, normalmente representado como uma árvore de operadores relacionais.

- O código que implementa os **operadores relacionais** (**Avaliador de Operador**) situa-se acima da camada de arquivos e métodos de acesso. Essa camada suporta o conceito de um **arquivo**, que, em um SGBD, é uma coleção de páginas ou de registros. Além de controlar as páginas de um arquivo, essa camada organiza as informações dentro de uma página.

- O código da camada de arquivos e métodos de acesso (**Arquivos e Métodos de Acesso**) situa-se acima do **Gerenciador de Buffer**, que carrega as páginas do disco para memória principal conforme necessário em resposta às solicitações de leitura.

- A camada mais inferior do SGBD trata do gerenciamento do espaço no disco, onde os dados são armazenados. As camadas superiores alocam, liberam, leem e escrevem páginas através de rotinas fornecidas por essa camada, chamada **Gerenciador de Espaço em Disco**.

- O SGBD suporta a concorrência e a recuperação de falha planejando cuidadosamente as solicitações de usuário e mantendo um log de todas as alterações realizadas no BD. Os componentes de um SGBD associados com o controle de concorrência e recuperação incluem o **Gerenciador de Transações**, que assegura que as transações solicitem e liberem bloqueios de acordo com um protocolo adequado de bloqueio e planeja a execução das transações.

- O **Gerenciador de Bloqueio**, controla as requisições por bloqueio e concede o direito de bloqueio nos objetos de banco de dados quando eles se tornam disponíveis.

- O **Gerenciador de Recuperação** é responsável por manter um log e restaurar o sistema a um estado consistente após a concorrência de uma falha.

1.5 Qual o papel do gerenciador de buffer?

Um Gerenciador de Buffer praticamente recupera objetos em disco e carrega-os na memória principal em forma de páginas. Existem semelhanças óbvias entre memória virtual em SOs e o Gerenciador de Buffer em SGBDs, tudo porque, em ambos os casos, o objetivo é fornecer acesso a mais dados do que cabem na memória principal, e a ideia básica é trazer páginas do disco para a memória principal conforme necessário, substituindo páginas que não sejam mais acessadas.

1.6 Qual a estrutura básica dos discos magnéticos?

- Os dados são armazenados no disco em unidades chamadas **blocos de disco**.
- Os blocos são organizados em anéis concêntricos chamados **trilhas**, em um ou mais **pratos**.
- O conjunto de todas as trilhas com o mesmo diâmetro é chamado de **cilindro**, porque o espaço ocupado por essas trilhas tem o formato de um cilindro; um cilindro contém uma trilha por superfície de prato.
- Cada trilha é dividida em arcos, chamados **setores**, cujo tamanho é uma característica do disco e não pode ser alterada.
- Um array de **cabeçotes de disco**, um por superfície gravada, é movido como uma unidade.
- Um **controlador de disco** realiza a interface de uma unidade de disco com o computador. Ele implementa comandos para ler ou gravar um setor movendo o braço e transferindo dados para e das superfícies do disco.
- Um **checksum** é calculado quando os dados são gravados em um setor e armazenados com o setor.
- Enquanto o acesso direto a qualquer local desejado na memória principal leva aproximadamente o mesmo tempo, determinar o **tempo de acesso** a um local do disco é mais complicado. O tempo de acesso a um bloco de disco possui diversos componentes:
  - **tempo de busca** é o tempo gasto para mover os cabeçotes dos discos para a trilha na qual um bloco desejado está localizado;
  - **latência rotacional** é o tempo de espera para que o bloco desejado gire sob o cabeçote do disco (é o tempo necessário para meia rotação em média e é geralmente menor do que o tempo de busca);
  - e, **tempo de transferência** é o tempo para realmente ler ou gravar os dados no bloco assim que o cabeçote esteja posicionado, ou seja, o tempo para o disco girar sobre o bloco.

1.7 Considere um disco com o tamanho de setor igual a 512 bytes, 200 trilhas por superfície, 50 setores por trilha, 5 pratos de disco com dois lados.

(a) Qual a capacidade da trilha em bytes?
A capacidade da trilha em bytes é dada pelo (tamanho do setor x número de

setores por trilha) =
512 *x* 50 bytes =
25.600 bytes.

(b) Qual a capacidade de cada superfície?
A capacidade de cada superfície é equivalente ao (tamanho do setor *x* número de setores por trilha *x* número de trilhas) =
512 *x* 50 *x* 200 bytes =
5.120.000

(c) Qual a capacidade do disco?
A capacidade do disco é igual ao (tamanho do setor *x* número de setores por trilha *x* número de trilhas *x* número de faces) =
512 *x* 50 *x* 200 *x* 2 bytes =
10.240.000

(d) Quantos cilindros o disco possui?
Cilindro é o conjunto de trilhas pertencentes a diferentes superfícies e discos de uma mesma unidade, mas que apresentam a mesma posição relativa. No disco considerado na questão, cada cilindro possui 10 trilhas (cinco pratos *x* duas faces). No entanto, a quantidade de cilindros do disco é igual ao número de trilhas existentes em cada superfície. Logo, o disco possui 200 cilindros.

(e) Dê exemplo de tamanhos de blocos válidos. 256 é um tamanho válido? 2.048? 51.200?
Como pelo enunciado os setores do disco possuem 512 bytes, temos como exemplos de tamanhos de blocos válidos: 512, 1.024, 2.048 ou 4.096 bytes por bloco. Como o setor é a menor unidade de informação que pode ser escrita no disco e o setor do disco tem 512 bytes, então não é possível existir bloco menor que 512 bytes. Logo, não pode existir um bloco de 256 bytes para o disco da questão. Como pelo enunciado os setores possuem 512 bytes, podemos ter blocos de 2.048 bytes ou mesmo de 51.200, uma vez que 2.048 e 51.200 são múltiplos de 512. Nesse caso, um bloco de 2.048 bytes seria formado por 4 setores e um bloco de 51.200 bytes seria formado por 100 setores.

1.8 Explique como é calculado o tempo de acesso a um bloco do disco.

O tempo de acesso, ou seja, o tempo para ler ou gravar em um bloco, varia dependendo da localização dos dados e é calculado utilizando o (tempo de busca

+ o atraso/latência rotacional + tempo de transferência).

Tempo de busca, atraso/latência rotacional e tempo de transferência foram explicados na **Questão 1.6** sobre a estrutura de discos magnéticos.

1.9 Explique quais são propriedades de um frame.

Frames (quadros) são locais que podem guardar uma página, geralmente páginas da memória principal, que em geral fica em disco ou em outro meio de armazenamento secundário.

1.10 Explique o que o gerenciador de buffer deve fazer para processar uma requisição de leitura por uma página. O que acontece se uma página requisitada está no buffer pool?

O Gerenciador de Buffer decide qual página existente na memória principal deve ser substituída para criar espaço para uma nova página. Para isso, ele utiliza uma política usada para decidir qual página substituir: a **política de substituição**.

O **buffer pool (ou pool de buffers)** se trata de uma coleção de páginas particionadas da memória principal disponível. Essa páginas da memória principal presentes no buffer pool são chamadas de **frames (quadros)**. Além do próprio buffer pool, o Gerenciador de Buffer mantém algumas informações de contagem e duas variáveis para cada frame no pool: ***pin_count*** e ***dirty***.

Quando uma página é solicitada, e ela está no buffer pool, o Gerenciador de Buffer faz o seguinte:

- Primeiro, quando qualquer página é solicitada, ele verifica o buffer pool para ver se algum frame contém a página, e, caso afirmativo, incrementa o *pin_count* deste frame.
- Depois, retorna o endereço (de memória principal) do frame contendo a página solicitada para o solicitante.

1.11 Quando um gerenciador de buffer escreve a página no disco?

O Gerenciador de Buffer escreve a página no disco quando ela não está no buffer pool. Portanto, primeiro ele escolhe um frame para substituição, usando a política de substituição, e incrementa seu *pin_count*. Se o *dirty* do frame de substituição estiver ligado, ele grava a página que ele contém, ou seja, a cópia em

disco da página é sobrescrita com o conteúdo do frame. Depois, ele lê a página solicitada para o frame de substituição, e retorna o endereço do frame que contém a página para o solicitante.

1.12 Quais são os algoritmos de paginação? Explique cada um deles, apresentando suas vantagens e desvantagens.

Quando não há mais lugar disponível no buffer, um bloco deve ser removido do buffer antes que um novo bloco possa ser copiado para ele. O objetivo principal de uma política (ou estratégia) de substituição de páginas (blocos) em um buffer é minimizar os acessos a disco.

- LRU: a política de substituição mais conhecida se trata do algoritmo **Lest Recently Used (LRU)**. Ela pode ser implementada no Gerenciador de Buffer usando-se uma fila de ponteiros para frames com *pin_count* 0. Um frame é adicionado ao final da fila quando se torna candidato a substituição e a página escolhida para substituição é a do frame do início da fila. Ou seja, a página a ser carregada é alocada na região de buffer ocupada pela página acessada mais remotamente (ou seja, há mais tempo).

- Clock2: uma variante da LRU, chamada substituição por **relógio (Two Round Clock)**, possui comportamento semelhante, mas com menos sobrecarga. A ideia é escolher uma página para substituição usando uma variável *corrente* que recebe valores de 1 a N, onde N é o número de frames de buffer, em ordem circular.

- Vantagens do LRU e Clock2: o esquema LRU e Clock2, em banco de dados, é mais elaborado que em sistemas operacionais, pois o SGBD é capaz de determinar antecipadamente quais blocos serão necessários por meio da verificação de cada um dos passos necessários para executar a operação solicitada pelo usuário.

- Desvantagens do LRU e Relógio: as políticas dos dois não são sempre as melhores estratégias para um SGBD, especialmente se muitas solicitações de usuários requererem varreduras sequenciais de dados, ou seja, quando vários usuários requisitam uma leitura sequencial (scan) de um conjunto de dados (uma tabela, por exemplo).

- FIFO: O **First-In First-Out** faz com que a página a ser carregada seja alocada na região de buffer ocupada pela página carregada menos recentemente no buffer.

- MRU: O **Most Recently Used** faz com que a página a ser carregada seja alocada na região de buffer ocupada pela página acessada mais recentemente.