Universidade Federal de Itajubá

Atividade recuperação final

COM231 – Banco de Dados II

Aluno: Ygor Salles Aniceto Carvalho 2017014382

Professora: Vanessa Cristina Oliveira de Souza

Dezembro

2020

Primeira Questão (2 pontos)
Um arquivo sequencial armazena 20.000 registros de ALUNO de tamanho fixo. Cada registro tem os seguintes campos: Nome (30 bytes), Cpf (9 bytes), Endereço (40 bytes), Telefone (10 bytes), Data_nascimento (8 bytes), Sexo (1 byte), Dep_princ (4 bytes), Dep_sec (4 bytes), Tipo_aluno (4 bytes) e Titulo_academico (3 bytes). Um byte adicional é usado como um marcador de exclusão. A chave primária da tabela aluno é CPF. O arquivo é armazenado num disco, cujo bloco tem 512 bytes.

a) Se um ponteiro ocupa 4 bytes, quantos bytes são necessários para o índice primário?

- ponteiro = 4 bytes
- 1 registro = 113 bytes + 1 do marcador de exclusão = 114 bytes
- reg/bloco = 512 bytes / 114 bytes = 4 regBloco
- 1 entrada de índice = 4 bytes do ponteiro + 9 bytes da PK
- FB = *qtd de bytes por bloco*/ 1 *entrada de índice* => 512 *bytes* / 13 *bytes* = 39 reg/bloco
- bloco de dados = *total de registros* / *reg.Bloco* = 20.000 / 4 = 5.000 blocos
- x = *bloco de dados* / *FB* = 5.000 *blocos* / 39 *reg.Bloco*
- 129 blocos
- 1 bloco = 512 bytes, possuindo no arquivo 129 blocos, logo:
- 512 * 129 => São necessário 65.048 bytes para o índice primário

b) Qual o tamanho de um índice secundário criado sobre o atributo Sexo, cujo ponteiro ocupa 8 bytes?

- ponteiro = 8 bytes
- 1 registro = 113 bytes + 1 do marcador de exclusão = 114 bytes
- reg/bloco = 512 bytes / 114 bytes = 4 regBloco
- 1 entrada de índice = 8 bytes do ponteiro + 1 byte do atrib. Sexo + 9 bytes da PK
- FB = *qtd de bytes por bloco*/ 1 *entrada de índice* => 512 *bytes* / 18 *bytes* = 28 reg/bloco
- bloco de dados = *total de registros* / *reg.Bloco* = 20.000 / 4 = 5.000 blocos
- x = *bloco de dados* / *FB* = 5.000 *blocos* / 28 *reg.Bloco*
- 179 blocos

c) Sobre esses índices acima mencionados, escreva em SQL e justifique uma consulta que usaria o índice e outra que não usaria. ATENÇÃO : A consulta precisa utilizar o atributo indexado.

EXPLAIN ANALYZE SELECT * FROM aluno
Para consultar todos os registros da tabela sem filtro utiliza-se somente o Seq Scan sem a necessidade de criar um índice.

CREATE INDEX aluno_nascimento ON aluno USING btree (data_nascimento);
EXPLAIN ANALYZE SELECT * FROM aluno WHERE data_nascimento <='2002/01/01'
Para uma consulta que fosse necessário buscar alunos maiores de idade por exemplo, deve-se utilizar índice btree com Bitmap Index Scan, pois o hash não seria possível nessa consulta já que o hash não consegue realizar consultas com intervalos, apenas com

igualdade. E como contém 20.000 registros ou seja uma quantidade alta e necessita de ordenação para a data de nascimento, esse seria o melhor.

CREATE INDEX aluno_sexo ON aluno U(sexo)
EXPLAIN ANALYZE SELECT nome, sexo FROM aluno WHERE sexo = ‘M’
Para uma consulta que retorne todos os alunos do sexo masculino utilizaria o índice hash, Bitmap Heap Scan, pois eu preciso apenas realizar uma consulta de igualdade para isso. Visto que o índice de hash ocupa menos espaço no disco que um índice btree.

Segunda Questão (2 pontos)
a) O que é mapeamento objeto-relacional?

Mapeamento objeto-relacional é uma técnica utilizada para reduzir a impedância ou seja a incompatibilidade entre o modelo relacional de tabelas com o modelo orientado a objetos.

É a persistência automatizada e transparente de objetos em uma aplicação Orientada a Objetos para as tabelas de um banco de dados relacional, utilizando metadados que descrevem o mapeamento entre os objetos e banco de dados.

É onde o programador tenta não trabalhar com modelo relacional dentro da aplicação. As tabelas do banco de dados são representadas através de classes e os registros de cada tabela são representados como instâncias das classes correspondentes.

b) Por que os bancos relacionais não atendem os requisitos necessários para aplicações big data?

Primeiro ponto é que o tipo de dado utilizado numa aplicação big data na maioria das vezes é desestruturado, ao contrário do banco relacional onde o dado é totalmente estruturado. Exemplo: um atributo do tipo inteiro no modelo relacional não pode receber dado de outro tipo, diferentemente de uma aplicação big data onde dependendo da regra o campo pode receber qualquer tipo de dado, seja int, string, imagem, vídeo, etc.

Outro ponto é que uma aplicação big data trabalha com 4 Vs (volume, velocidade, variedade e veracidade). Como o banco de dados relacional trabalha com tabelas e cada tabela representa um arquivo no storage, para aumentar a capacidade de armazenamento é necessário crescer de forma vertical ou seja estendo uma máquina, adicionando novos componentes na máquina onde fica hospedado o banco. Porém chega um momento que não é mais possível estender a máquina, seria necessário adquirir um novo hardware e conforme fosse crescendo chega um momento que não há hardware que suporte a quantidade de dados. É possível sim crescer o banco relacional de forma horizontal (armazenando o banco em servidores distintos) porém a complexidade é muito grande e as buscas de dados nessa arquitetura para o relacional torna muito lenta se comparado ao modelo NoSQL.

Já o modelo NoSQL oferece total suporte para aplicações desse tipo, sendo desestruturado (aceitando dados de qualquer tipo) e possuindo uma forma totalmente adequada para arquiteturas horizontais, onde os dados podem crescer de forma exponencial não tendo problemas de complexidade e eficiência ao hospedar em diversos servidores.

Terceira Questão (2 pontos)
Dado o escalonamento abaixo:

Tempo	T1	T2
1	READ_LOCK(Y)	
2	READ_ITEM(Y)	
3	UNLOCK(Y)	
4		READ_LOCK(X)
5		READ_ITEM(X)
6		UNLOCK(X)
7		WRITE_LOCK(Y)
8		READ_ITEM(Y)
9		Y:= X+Y
10		WRITE_ITEM(Y)
11		UNLOCK(Y)
12	WRITE_LOCK(X)	
13	READ_ITEM(X)	
14	X:= X+Y	
15	WRITE_ITEM(X)	
16	UNLOCK(X)	

a) É serializável?
Não é serializável pois transação 2 não obedece ao bloqueio de duas fases, e todas devem obedecer ao bloqueio de duas fases para ser serializável.

b) As transações obedecem ao bloqueio de duas fases? Por quê?
A transação 2 não obedece o bloqueio de duas fases pois ela bloqueia o X para leitura, faz a leitura e desbloqueia, porém depois bloqueou novamente um item que é o Y. Ela deveria bloquear o X, fazer a leitura dele, bloquear o Y para escrita e só assim poderia desbloquear o X.

c) Se X inicia com valor 20 e Y com valor 30, qual o resultado da execução das transações?
Resultado:
x = 70
y = 50

d) Se ocorrer uma falha no sistema no tempo 12:
i. Como estará o log de recuperação?
Se ocorrer falha no sistema no tempo 12 ocorrerá o rollback em cascata onde a T1 vai sofrer rollback e a T2 também por estarem manipulando o mesmo item de dados no caso o X. x=20 e y=30

ii. Considerando as formas de recuperação baseadas em log vistos, o que o sistema deve fazer quando for reinicializado? Justifique.
Caso ocorra falha no tempo 2, ao reiniciar o sistema deve fazer rollback na transação T1 e T2 por estarem manipulando o mesmo item de dados no caso o X

Quarta Questão (2 pontos):
Considere a estrutura do documento abaixo, armazenado em um banco de dados MongoDB, de uma coleção chamada 'álbum'. Crie as consultas que responda às seguintes consultas:

```
{
  type: "Audio Album",
  title: "A Love Supreme",
  description: "by John Coltrane",

  shipping: {
    weight: 6,
    dimensions: {
      width: 10,
      height: 10,
      depth: 1
    },
  },

  pricing: {
    list: 1200,
    retail: 1100,
    savings: 100,
    pct_savings: 8
  },

  details: {
    title: "A Love Supreme [Original Recording Reissued]",
    artist: "John Coltrane",
    genre: [ "Jazz", "General" ],
    tracks: [
      "A Love Supreme Part I: Acknowledgement",
      "A Love Supreme Part II - Resolution",
      "A Love Supreme, Part III: Pursuance",
      "A Love Supreme, Part IV-Psalm"
    ],
  },
}
```

a) Retornar os álbuns por gênero (genre) e ordenados pelo nome do artista.
db.album.find({}, {"details.genre": true, "_id": false}).sort({"details.artist": 1})

b) Retornar os álbuns cujo preço no varejo (retail) é menor que 100.
db.album.find({"pricing.retail":{$lt : 100}}).pretty()

c) Retornar a quantidade de álbuns por artista.
db.album.aggregate([{$group : {_id : "$details.artist", count: {$sum : 1}}}]).pretty()

Quinta Questão (2 pontos)
Imagine um banco de dados usado no Sistema Controle de Escola. Crie as seguintes permissões (utilize SQL), indicando como a implementação desses privilégios pode ser feita.

Usuários	Permissões	Tabelas
Coordenação	Inserir, alterar, apagar	Professor e Curso
Coordenação	Consultar	Aluno
Gerente do Financeiro	Consultar o valor total da folha de pagamento	Professor, Funcionários
Professor	Consultar matricula e nome dos seus alunos	Aluno

```
CREATE ROLE coordenacao;
GRANT INSERT, UPDATE, DELETE ON TABLE  public.professor, public.curso TO coordenacao;
GRANT SELECT ON TABLE public.aluno TO coordenacao;
GRANT USAGE ON SCHEMA public TO coordenacao;


--Necessário criar uma view para exibir o total de salario do professor e total do funcionario
--Fiz um select de union pois professor e funcionário ao meu ver não se relacionam
CREATE VIEW folha_pagamento AS (
	SELECT SUM(salario) FROM professor
	UNION
	SELECT SUM(salario) FROM funcionario
);
CREATE ROLE gerente_financeiro;
GRANT SELECT ON public.folha_pagamento TO gerente_financeiro;
GRANT USAGE ON SCHEMA public TO gerente_financeiro;


--Necessário criar uma view para exibir somente os atributos nome e matricula do aluno
CREATE VIEW consulta_aluno AS (SELECT matricula, nome FROM aluno);
CREATE ROLE professor;
GRANT SELECT  ON public.consulta_aluno TO professor;
GRANT USAGE ON SCHEMA public TO professor;
```