Banco de Dados II: Otimização de Consultas

|  |  |
| --- | --- |
| **Aluno** | **R.A.** |
| Carlos H. Paisca | 62030 |
| Gabriel Papa Belini | 67020 |
| Leonardo Neumann | 70132 |
| Thiago Bucalon | 68962 |

**Introdução**

Bancos de dados são coleções organizadas de dados que se relacionam de forma a criar algum sentido (informação) e dar mais eficiência durante uma pesquisa ou estudo. Se tornaram peças vitais em organizações após a evolução dos sistemas de informação. Existem diversas formas de se modelar um banco de dados; entre as mais populares temos o *modelo plano (tabela)*, *modelo em rede (grafo)*, *modelo hierárquico (árvore)*, *orientado a objetos (classe)* e *modelo relacional (relações tabulares)*.

Os sistemas de gerenciamento de banco de dados (*SGBD*) são programas de computador responsáveis por fazer o gerenciamento de bancos de dados. Seu principal objetivo é retirar da aplicação cliente a responsabilidade de gerenciar o acesso, a manipulação e a organização dos dados. Entre os sistemas *SGBD*, o mais utilizado, desde 1980, é o modelo relacional.

Em definição simplificada, o modelo relacional baseia-se em dois conceitos: conceito de entidade e relação - Uma entidade é um elemento caracterizado pelos dados que são recolhidos na sua identificação vulgarmente designado por tabela. Na construção da tabela identificam-se os dados da entidade. A atribuição de valores a uma entidade constrói um registro da tabela. A relação determina o modo como cada registro de cada tabela se associa a registros de outras tabelas. Os *SGBD*s relacionais normalmente trabalham com a *Structured Query Language* (*SQL*), que é uma linguagem de consulta onde é possível especificar e trabalhar com as relações diretamente na consulta.

Neste trabalho realizaremos o estudo proposto nas especificações da disciplina utilizando o *SGBD* *open source* MySQL, que é um banco de dados relacional relativamente simples e rápido. Além disso, será utilizado a interface de gerenciamento PHPMyAdmin.

**Desenvolvimento**

Importamos três tabelas no banco de dados MySQL/InnoDB, utilizando os arquivos fornecidos na especificação do trabalho, referentes a uma base de dados de planos de saúde (PACIENTE, REQUISIÇÃO, RESULTADOEXAME).

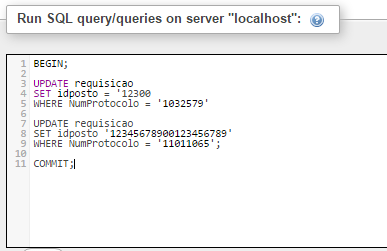


Figura 1: Tabelas no Banco de Dados

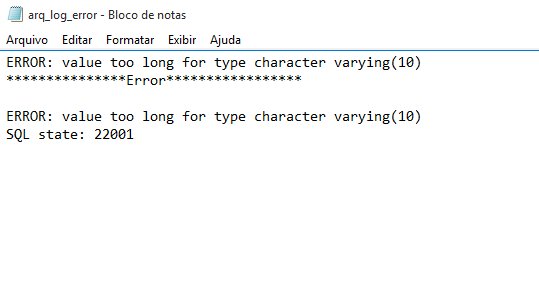
O MySQL/InnoDB apresenta controle automático de acesso concorrente utilizando travas de acesso exclusivo e compartilhado, portanto, nenhuma implementação extra será necessária. Além disso, o SGBD trabalha com atualização imediata, mas utilizando uma sintaxe especial ‘DELAYED’ é possível realizar a atualização atrasada.

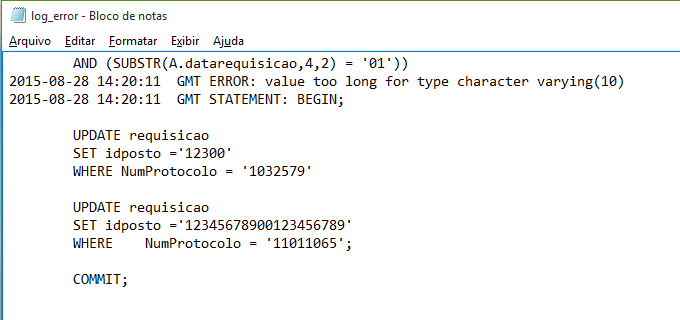
**Recuperação de Falhas**

Primeiramente vamos forçar a ocorrência de uma falha das transações e verificar se o SGBD relata alguma recuperação de falhas, em seguida verificaremos o log do sistema e analisaremos o que foi antes e depois de ter ocorrido a falha. Além disso, ver se o SGBD faz atualização imediata ou adiada, através do log do sistema ou da documentação do SGBD.



Nessa transação vamos atualizar o valor da coluna “idposto” da tabela “requisição” para dois cadastros: (1) Cadastro que possui o número de protocolo igual a 1032579; (2) Cadastro que possui o número de protocolo igual a 11011065. No caso do primeiro valor é um caso valido, pois, a coluna idposto é definida na tabela com o tipo “varchar(10)”, no entanto para o segundo update o valor passado não é valido, pois excede o tamanho da string que é igual a 10. O seguinte erro é apresentado ao executar a transação no SGBD:

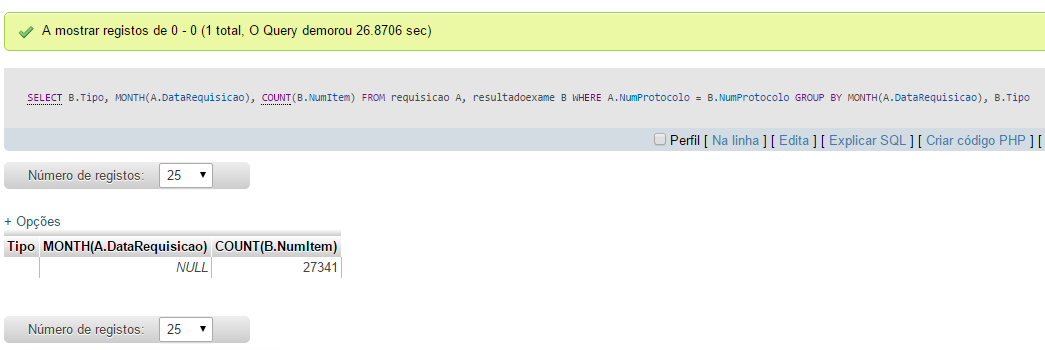




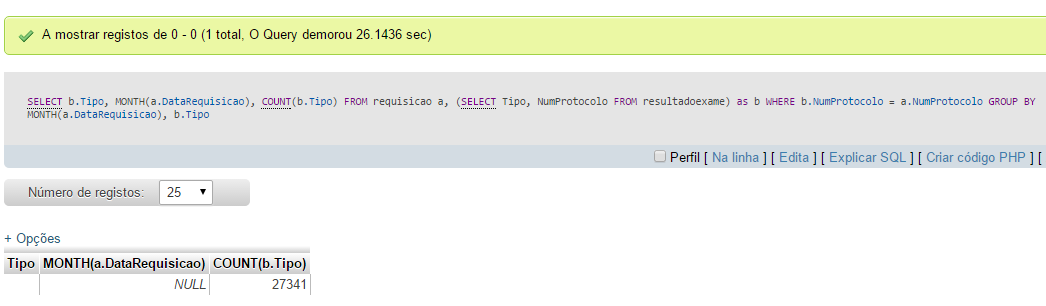
Ao consultar o banco de dados, estes eram no exatamente iguais ao momento em que a consulta foi solicitada, portanto, as mudanças ocorridas depois disso não foram registradas. Isso ocorre pelo fato do MySQL tratar a execução de qualquer comando SQL como sendo executado dentro de uma transação.

**Otimização de Consultas**

Sendo assim, foram feitas duas consultas no banco de dados como requisitado, sendo a primeira utilizada como modelo e a segunda com otimização. Podemos verificar que houve uma diminuição no tempo de resposta da consulta otimizada em relação à consulta inicial, o que pode ser concluído nas imagens a seguir:



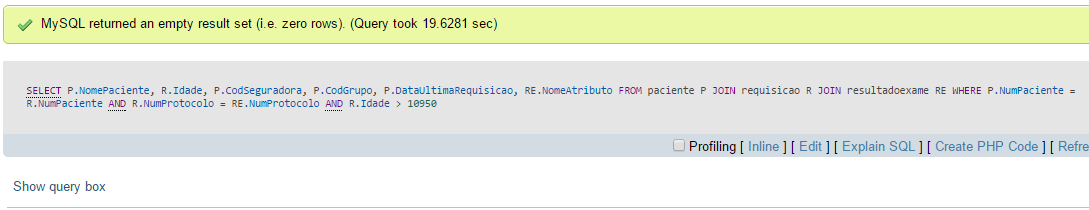
*Figura 5: Consulta sem otimização, SQL Query: SELECT b.Tipo, MONTH(a.DataRequisicao), COUNT(b.Tipo) FROM requisicao a, (SELECT Tipo, NumProtocolo FROM resultadoexame) as b WHERE b.NumProtocolo = a.NumProtocolo GROUP BY MONTH(a.DataRequisicao), b.Tipo;*



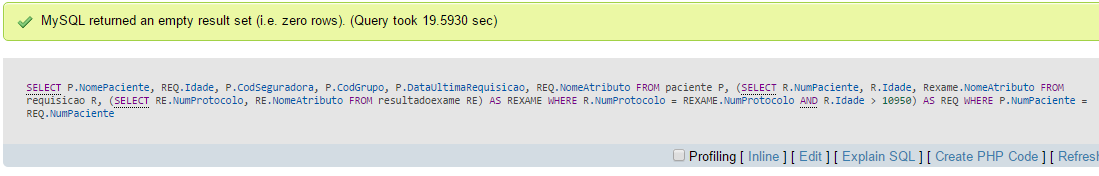
*Figura 6: Consulta com otimização, SQLQuery: SELECT b.Tipo, MONTH(a.DataRequisicao), COUNT(b.Tipo) FROM requisicao a, (SELECT Tipo, NumProtocolo FROM resultadoexame) as b WHERE b.NumProtocolo = a.NumProtocolo GROUP BY MONTH(a.DataRequisicao), b.Tipo;*

Podemos perceber pelas imagens que as consultas não retornaram nenhum registro, isso ocorreu devido ao fato que não conseguimos importar todos os registros da tabela resultadoexame devido a alguns problemas presentes no arquivo de importação, porém mesmo assim podemos visualizar a otimização feita entre as duas consultas analisando o tempo de resposta sendo 26,8707 sem otimização e 26,1436 com otimização.

Realizamos em seguida duas operações de consulta na base de dados, seguindo a mesma linha de raciocínio do exercício anterior, visualizadas respectivamente nas figuras abaixo:



*Figura 7: Consulta sem otimização, SQL Query: SELECT P.NomePaciente ,R.Idade, P.CodSeguradora, P.CodGrupo, P.DataUltimaRequisicao, RE.NomeAtributo FROM paciente P JOIN requisicao R JOIN resultadoexame RE WHERE P.NumPaciente = R.NumPaciente AND R.NumProtocolo = RE.NumProtocolo AND R.Idade > 10950.*



*Figura 8: Consulta com otimização, SQL Query: SELECT P.NomePaciente, REQ.Idade, P.CodSeguradora, P.CodGrupo, P.DataUltimaRequisicao, REQ.NomeAtributo FROM paciente P,(SELECT R.NumPaciente, R.Idade, R.exame.NomeAtributo FROM requisicao R, (SELECT RE.NumProtocolo, RE.NomeAtributo FROM resultadoexame RE) AS REXAME WHERE R.NumProtocolo = REXAME.NumProtocolo AND R.Idade > 10950) AS REQ WHERE P.NumPaciente = REQ.NumPaciente.*

***Criação de Usuários e Privilégios***

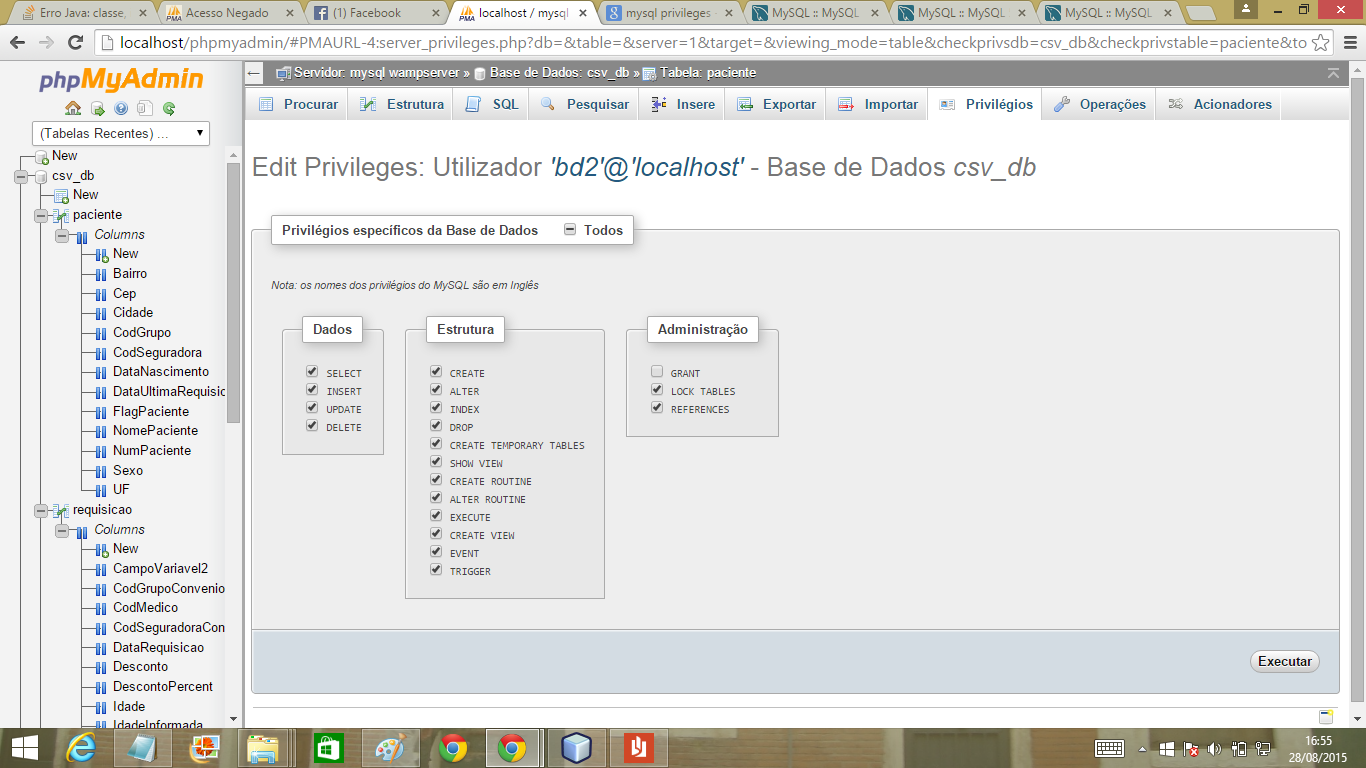
A criação de novos usuários no MySQL é feita da seguinte forma:

***CREATE USER 'bd2'@'localhost' IDENTIFIED BY '1234';***

O comando cria um usuário com nome bd2 em localhost com uma senha “1234”. Para garantir privilégios ao novo usuário à tabela utilizada pelo trabalho, utilizamos o seguinte comando:

***GRANT ALL ON csv\_db.\* TO 'bd2'@'localhost';***

Este comando permite que tal usuário tenha acesso total dentro do banco de dados “csv\_db”. Além disso, o controle de acesso a tabelas e autorizações de operações pode ser realizado na própria interface do PHPMyAdmin, como ilustra a figura abaixo:



***Figura 9: Edição de privilégios através da interface do PHPMyAdmin.***

**Conclusão**

Concluímos que as otimizações e gerenciamento realizados pelo MySQL favorecem a criação de bancos de dados complexos, o que garante ao usuário a vasta possibilidade de abstração dos dados, além de garantir ao mesmo a consistência e controle de concorrência, o que permite que este SGBD seja utilizado em aplicações que dependem de tal nível de confiança.