TRABALHO DE BANCO DE DADOS II

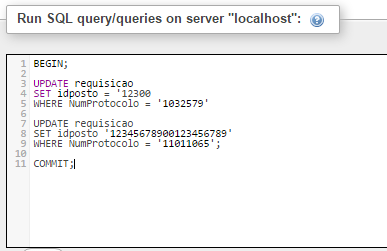
OTIMIZAÇÃO DE CONSULTAS

Docente da Disciplina: Maria Madalena

|  |  |
| --- | --- |
| RA: | DISCENTES: |
|  | Carlos Henrique |
|  | Gabriel Belline |
|  | Leonardo Neuman |
|  | Thiago Rodrigo |

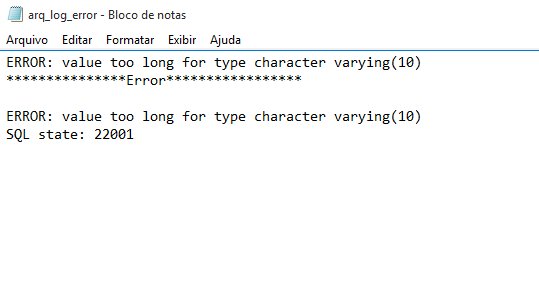
Maringá, 2015

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.

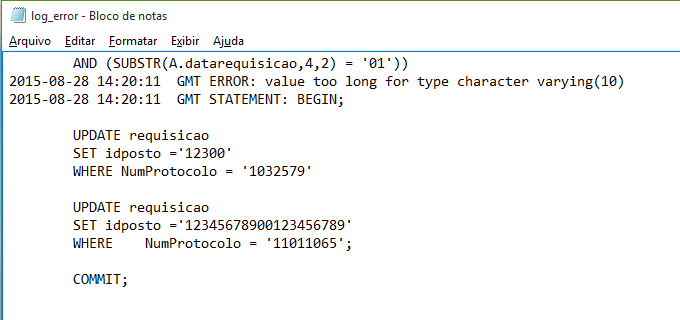


Nessa transação vamos atualizar o valor da coluna “idposto” da tabela “requisição” para dois cadastros: (1) Cadastro que possui o número de protocolo igual a 1032579; (2) Cadastro que possui o número de protocolo igual a 11011065. No caso do primeiro valor é um caso valido, pois, a coluna idposto é definida 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.

Ao tentar executar a transação no SGDB, o seguinte erro aparece na tela:



No log do SGBD podemos ver a falha na transação:



Isso ocorre, pois, caso haja um erro o SGDB executa um comando ROLLBACK.

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.

Para a elaboração deste trabalho utilizamos o MySQL, que é um banco de dados *open source* muito conhecido no mercado, por ser rápido, simples e funcional. Criamos três tabelas no banco de dados, utilizando os arquivos fornecidos pela professora de uma base de dados sobre Plano de saúde (PACIENTE, REQUISIÇÃO, RESULTADOEXAME). Como podemos visualizar na figura abaixo:



Figura 1: Tabelas no Banco de Dados

Em seguida, foi feita a conexão do banco de dados com o Netbeans, que é uma IDE utilizada para codificação na linguagem JAVA, ilustrado na figura abaixo:

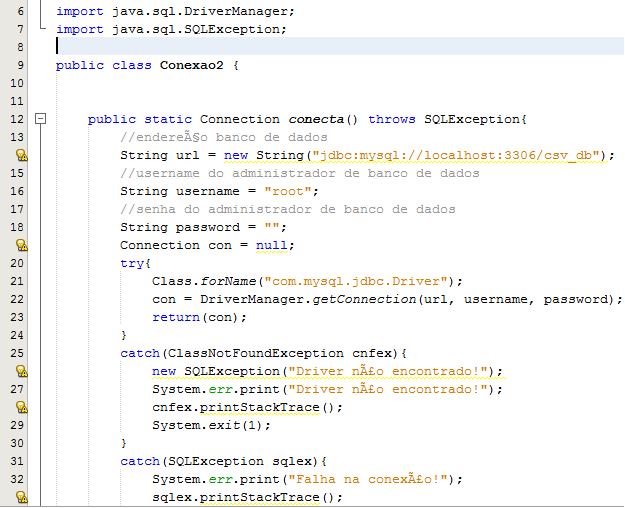


Figura 2: Conexão JAVA com banco de dados

Com a conexão estabelecida, realizamos a implementação de duas transações simples que acessam as mesmas tabelas do banco de dados. Essas foram implementadas na linguagem java, utilizando threads. Foi feito um *select* na tabela paciente, recuperando o NumPaciente, CodGrupo, sexo, cep, cidade e bairro de todos os campos cujo bairro = centro.

Em seguida, nas threads escolhemos um determinado NumPaciente e fizemos uma alteração do bairro = centro para “Jd. Paraiso” e na segunda thread é feito a alteração desse campo para ”Jd. Paraiso 2”. A partir disso analisando os resultados obtidos chegamos à conclusão de que o banco de dados utilizado faz o uso de bloqueio compartilhado para leitura e exclusivo para escrita, conforme observamos nas figuras abaixo.

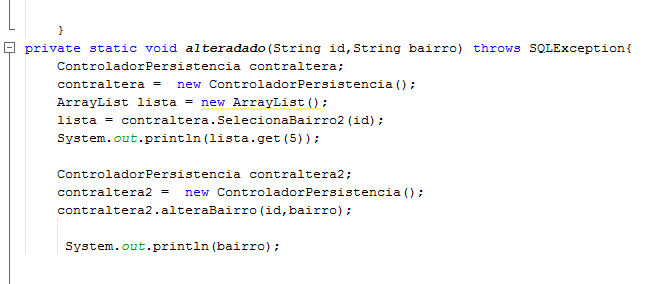


Figura 3: Altera dados das tabelas

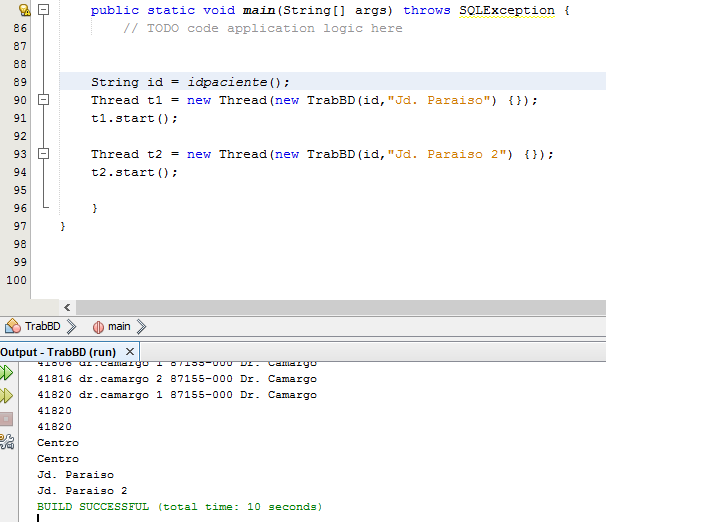


Figura 4: Threads

O banco está realizando a leitura compartilhada e bloqueio exclusivo, pois, as funções t1() e t2() fazem leitura simultaneamente.

Sendo assim, foram feitas duas consultas no banco de dados como requisitado pela professora, sendo a primeira sem otimização e a segunda com otimização de consulta, pudemos verificar que houve uma diminuição no tempo de resposta da consulta otimizada em relação à consulta não otimizada, à conclusão pode ser visualizada nas imagens a seguir:

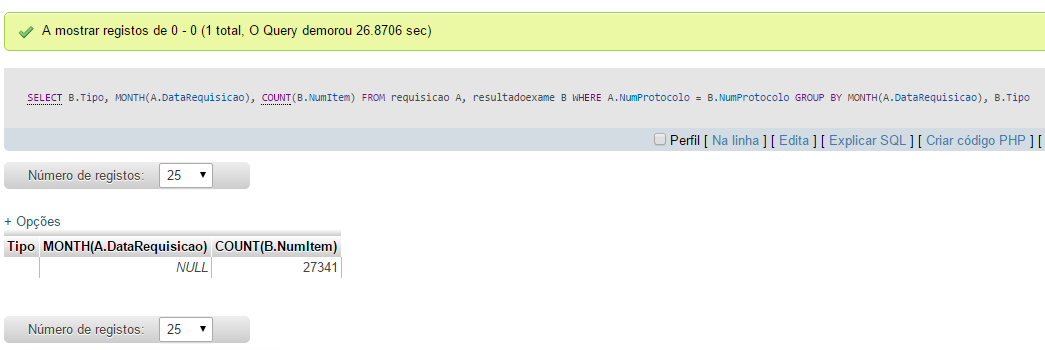


Figura 5: Consulta sem otimização, SQL Query: SELECT b.Tipo, MONTH(a.DataRequisicao), COUNT(b.Tipo) FROM requisicao a, (SELECT Tipo, NumProtocolo FROM resultadoexame) as b WHERE b.NumProtocolo = a.NumProtocolo GROUP BY MONTH(a.DataRequisicao), b.Tipo;

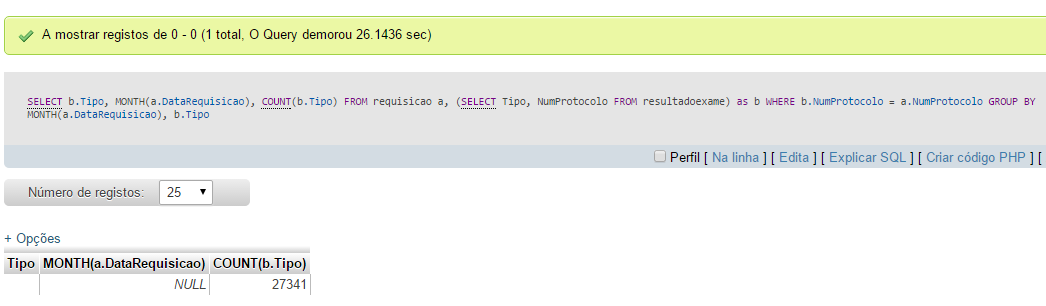


Figura 6: Consulta com otimização, SQLQuery: SELECT b.Tipo, MONTH(a.DataRequisicao), COUNT(b.Tipo) FROM requisicao a, (SELECT Tipo, NumProtocolo FROM resultadoexame) as b WHERE b.NumProtocolo = a.NumProtocolo GROUP BY MONTH(a.DataRequisicao), b.Tipo;

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 erros presentes no arquivo “resultadoexame.txt” que não foram tratados por nós, porém mesmo assim podemos visualizar a otimização feita entre as duas consultadas analisando o tempo de resposta sendo 26,8707 sem otimização e 26,1436 com otimização.

Para o exercicio 2-b realizamos duas operações de consulta na base de dados, seguindo a mesma linha de raciocínio do exercício anterior, as consulta sem otimização e com otimização otimização pode ser visualizadas respectivamente nas figuras abaixo.

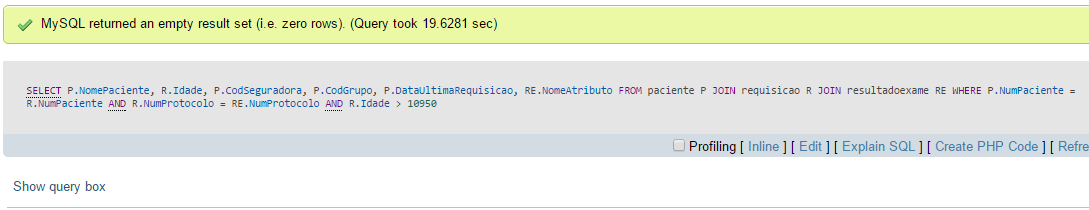


Figura 7: Consulta sem otimização, SQL Query: SELECT P.NomePaciente ,R.Idade, P.CodSeguradora, P.CodGrupo, P.DataUltimaRequisicao, RE.NomeAtributo FROM paciente P JOIN requisicao R JOIN resultadoexame RE WHERE P.NumPaciente = R.NumPaciente AND R.NumProtocolo = RE.NumProtocolo AND R.Idade > 10950.

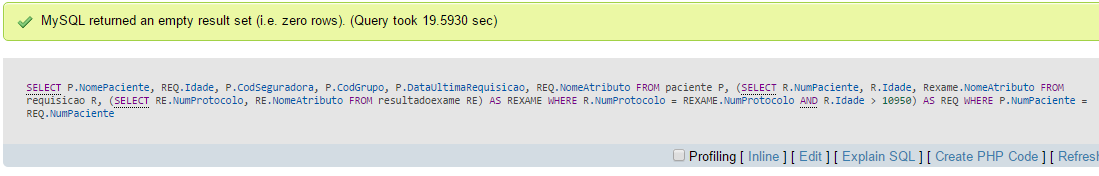


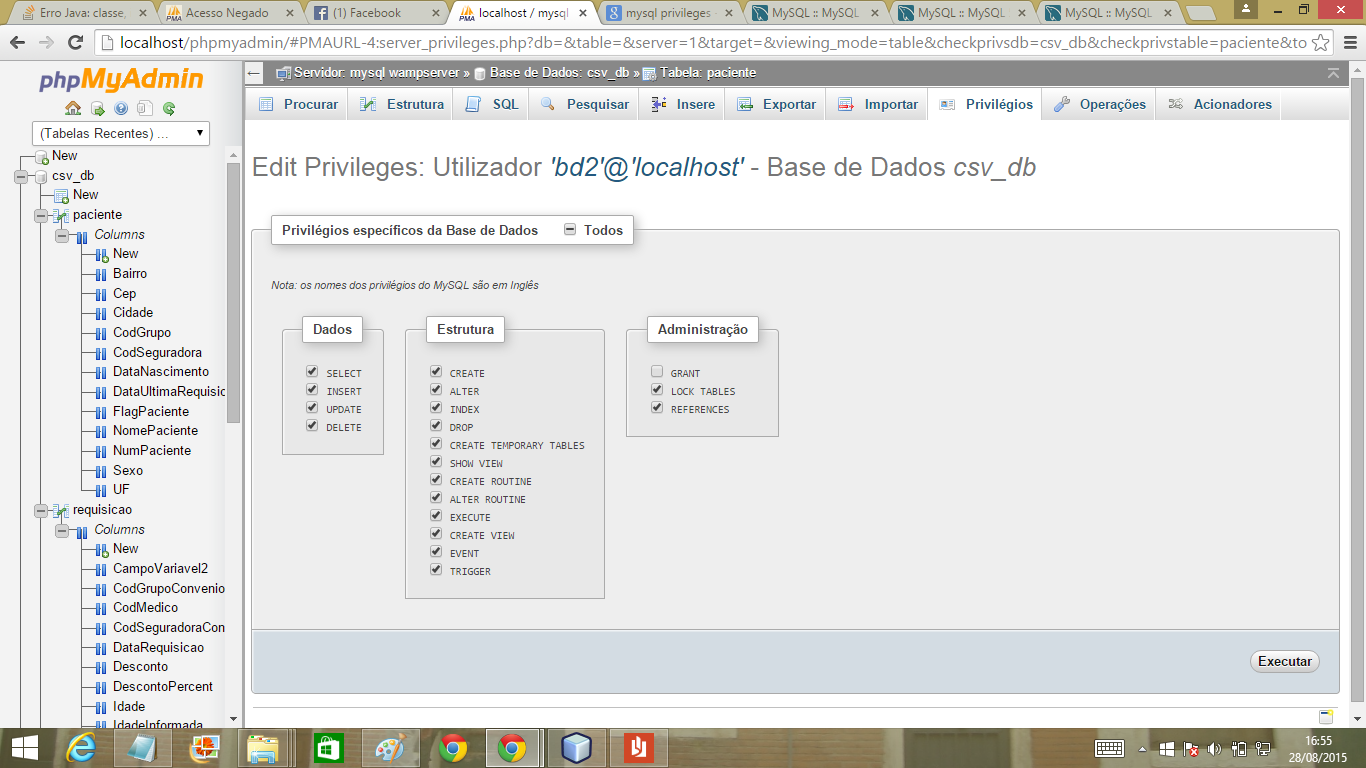
Figura 8: Consulta com otimização, SQL Query: SELECT P.NomePaciente, REQ.Idade, P.CodSeguradora, P.CodGrupo, P.DataUltimaRequisicao, REQ.NomeAtributo FROM paciente P,(SELECT R.NumPaciente, R.Idade, R.exame.NomeAtributo FROM requisicao R, (SELECT RE.NumProtocolo, RE.NomeAtributo FROM resultadoexame RE)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

A criação de novos usuários no PHPMyAdmin é feita no próprio console da aplicação por meio dos seguintes comandos:

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

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

O primeiro deles cria um usuário com nome bd2 em localhost com uma senha “1234”, já a segunda linha permite que tal usuário acesse as tabelas do banco “csv\_db” com seu host bd2@localhost, além disso o parâmetro ‘GRANT ALL’ permite que o usuário tenha acesso total dentro do banco de dados “csv\_db”. O controle de acesso a tabelas e autorizações de operações podem ser modificadas na própria interface do PHPMyAdmin, como ilustra a figura abaixo:



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