**MYSQL – INDEXAÇÃO**

LUCAS LAMOUNIER GONCALVES DUARTE - 2016012688

KEVIN VIEIRA PEREIRA - 2016015385

RODRIGO APARECIDO SILVA MAIA - 2016013095

Itajubá, 9 de abril de 2018

Sumário

[**QUESTÃO 1: ESTUDAR AS ESTRUTURAS DE ÍNDICES DISPONÍVEIS NO SEU SGBD. DETALHE A SINTAXE DE CRIAÇÃO E REMOÇÃO DE ÍNDICES.** 2](#_Toc511044545)

[**QUESTÃO 2: COMO LISTAR OS ÍNDICES DE UMA TABELA DO BANCO?** 6](#_Toc511044546)

[**QUESTÃO 3: CRIAR UMA VIEW NO BANCO. É POSSÍVEL INDEXAR ESSA VIEW?** 7](#_Toc511044547)

[**QUESTÃO 4: QUAIS FERRAMENTAS DE ANÁLISE DE CONSULTA SEU SGBD OFERECE?** 8](#_Toc511044548)

[**QUESTÃO 5: IMPLEMENTE UMA CONSULTA ‘PESADA’ NO SEU BANCO.** 10](#_Toc511044549)

[a) Verifique o tempo necessário para processar essa consulta. 10](#_Toc511044550)

[b) Agora crie um índice e refaça a consulta. Diminuiu o tempo de consulta? Justifique o índice criado. 10](#_Toc511044551)

[**QUESTÃO 6: AVALIE POR MEIO DE CONSULTAS E DESCREVA A IMPRESSÃO DO GRUPO SOBRE:** 10](#_Toc511044552)

[a) Índice de Hash 10](#_Toc511044553)

[b) Índice composto 11](#_Toc511044554)

[c) Índice em um campo BLOB 11](#_Toc511044555)

[**QUESTÃO 7: QUAIS SÃO AS FORMAS POSSÍVEIS DE INICIAR UMA TRANSAÇÃO NO SGBD?** 11](#_Toc511044556)

[**QUESTÃO 8: PESQUISE SOBRE O CHAMADO SQL INJECTION (O QUE É, COMO PREVINIR).** 11](#_Toc511044557)

[**REFERÊNCIA:** 12](#_Toc511044558)

# 

# **QUESTÃO 1: ESTUDAR AS ESTRUTURAS DE ÍNDICES DISPONÍVEIS NO SEU SGBD. DETALHE A SINTAXE DE CRIAÇÃO E REMOÇÃO DE ÍNDICES.**

As estruturas de índices disponíveis para MySQL podem ser consultadas através do comando **SHOW ENGINES,** no terminal de controle após ter se conectado ao servidor MySQL, a figura abaixo mostra as engines disponíveis.

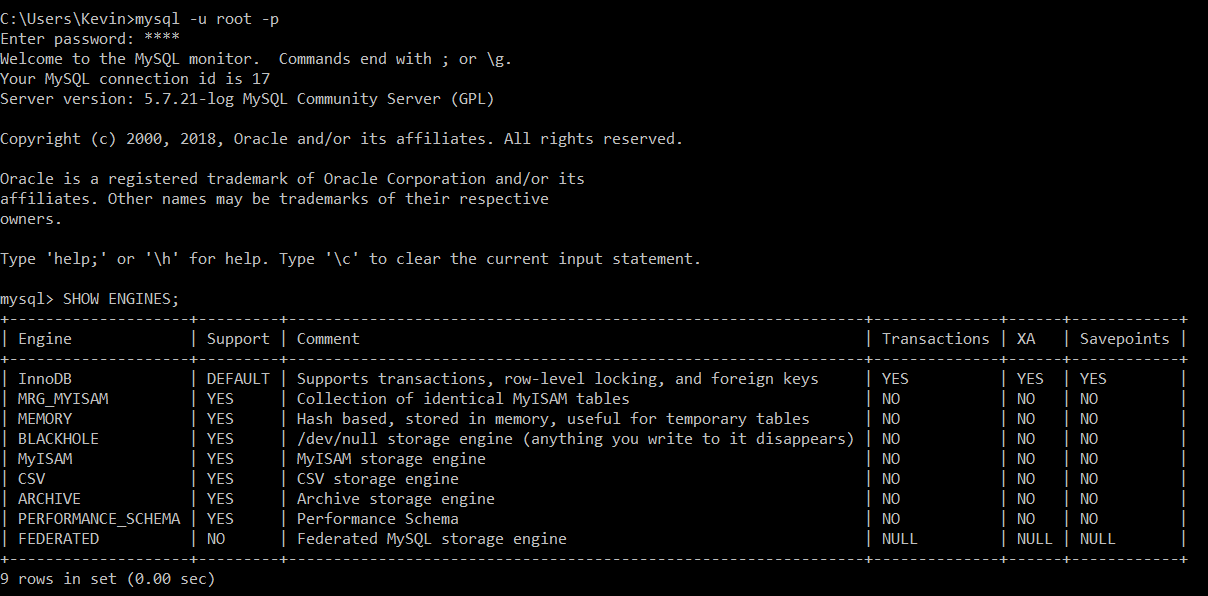


Figura 1 – Mostrando as engines disponíveis

Por padrão a engine de armazenamento usada é a InnoDB, para se utilizar de outra engine, pode-se realizar através de 3 diferentes formas:

1. Pode-se especificar durante a criação de uma tabela a engine usada para aquela tabela, isso ocorre através do comando **CREATE TABLE nomeDaTabela (atributo tipo, ...) ENGINE = NomeDaEngine**;



Figura 2 – Como especificar uma engine através do comando create

1. Pode-se alterar a engine de uma tabela através do comando **ALTER TABLE nomeDaTabela ENGINE = NomeDaEngine;**

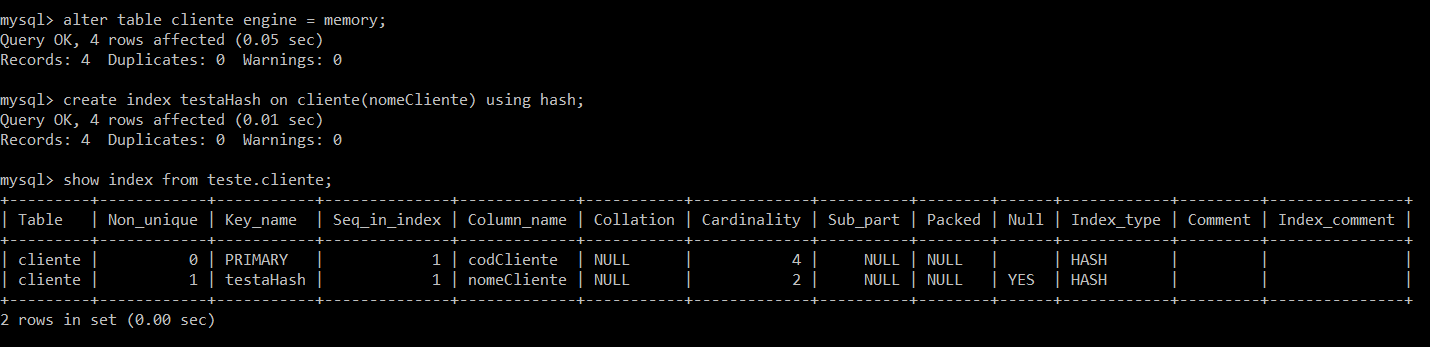


Figura 3 – Mostrando como trocar a engine de uma tabela

1. Também é possível alterar engine padrão do banco, para isso é necessário executar o comando **set default\_storage\_engine = NomeDaEngine;**

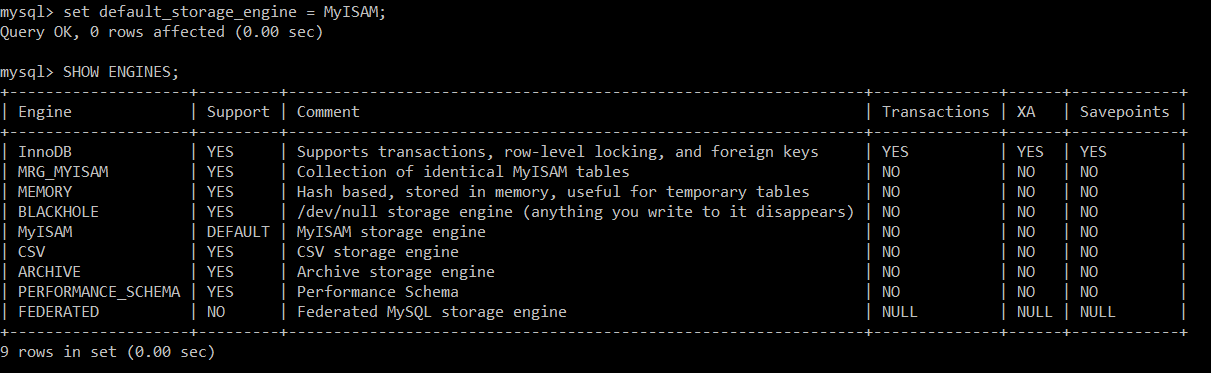


Figura 4 –Trocando a engine padrão do banco

Abaixo será explicado de forma simples cada engine:

* A engine InnoDB permite a realização de uma transação segura, pois implementa o protocolo ACID e permite a utilização de chaves estrangeiras.
* A MyISAM é caracterizada por possuir um nível de proteção de tabela (table lock), que limita a performance para leitura e escrita apenas.
* Memory engine, armazena todos os dados na memória RAM, permitindo acesso rápido, melhor utilizada para tabelas temporárias;
* Engine CSV utiliza de tabelas que não são indexadas, então normalmente se utiliza da InnoDB para manipulação e depois caso seja necessário importar ou exportar arquivos do tipo CSV utilizará desta engine.
* A Archive engine é caracterizada por ser compacta e intencionada para armazenar e retirar grandes quantidades de informações como as geradas por auditorias de segurança.
* A engine Blackhole não armazena dados, então comandos como insert, update e delete não tem efeitos. Sendo utilizada essencialmente quando se precisa passar comandos DML para uma base de dados escravo sem armazenar os comandos na base de dados principal;
* Federated engine permite a possibilidade de se conectar diversas bases de dados MySQL, fazendo com que elas ajam como uma só, boa para a distribuição de dados;

A engine que será analisada mais a fundo será a InnoDB por ser a engine padrão do MySQL. Ela utiliza de índices baseados na árvore B, e para os tipos espaciais usa arvore R que são especificas para dados multidimensionais, o padrão de tamanho de um índice é de 16KB.

Todas as tabelas utilizam do índice Clustering, que usa a primary key como o índice. Caso não se tenha declarado uma primary key, mas se existir um campo unique, que possua nenhum valor nulo este passará a ser o novo índice, ou, se houver uma coluna unique com valores nulos, será criado um índice chamado GEN\_CLUST\_INDEX que será o índice cluestering, sendo auto incrementado.

A implementação do índice secundário utiliza da primary key, e das colunas especificadas, para um melhor desempenho é recomendado que a primary key seja curta. O comando de criação de um índice secundário para InnoDB é **CREATE Prefixo INDEX nomeDoIndice ON nomeDaTabela(nomeDasColunas) COMMENT;** o prefixo pode ser utilizado para demonstrar que o índice é composto de valores únicos, textuais ou até do tipo espacial. Também é possível colocar um comentário no final do comando.

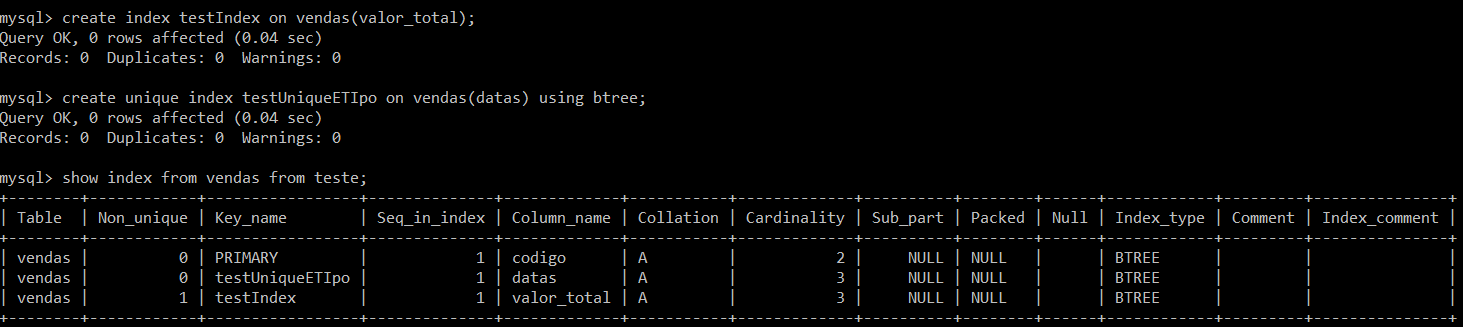


Figura 5 – Criando um índice, através do create index

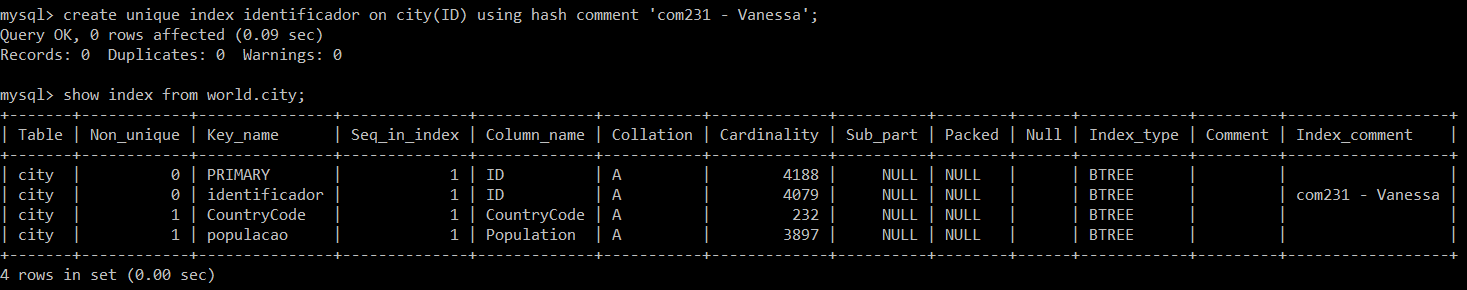


Figura 6 – Criando um índice, com o prefixo unique, e mostrando a tentativa de usa hash

Na imagem acima foi demonstrado a utilização do prefixo e do comment, e que a engine InnoDB até permite a utilização do comando using hash, mas sem efeito pois será utilizado a árvore B, caso se deseje utilizar a estrutura hash é necessário mudar a engine para uma que aceite hash, como é demonstrado na imagem abaixo.

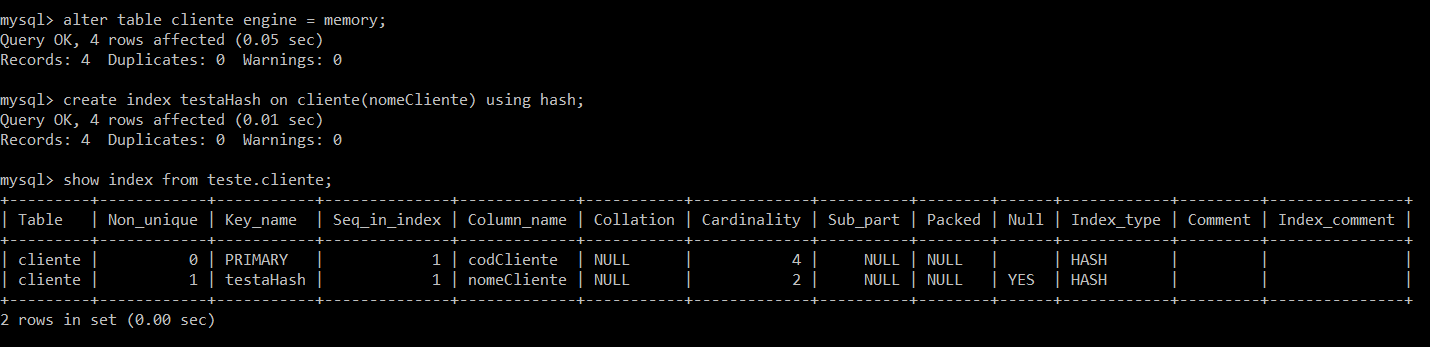


Figura 7 – Criando um índice com a utilização de hash

A inserção de um índice também pode se dar pelo comando **ALTER TABLE nomeDaTabela ADD INDEX nomeDoIndice (NomeDaColuna);**

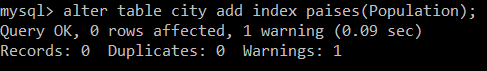


Figura 8 – Criando um índice, através do alter table

O método para se remover um índice secundário é **DROP INDEX nomeDoIndice ON nomeDaTabela;**

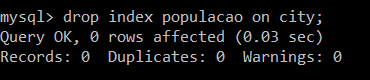


Figura 9 – Removendo um índice, pelo comando drop index

Também pode-se remover pelo comando **ALTER TABLE nomeDaTabela DROP INDEX nomeDoIndice;**

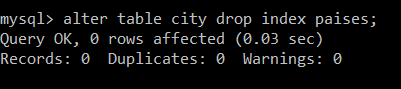


Figura 10 – Removendo um índice, pelo comando alter table

# **QUESTÃO 2: COMO LISTAR OS ÍNDICES DE UMA TABELA DO BANCO?**

Para se listar os índices de uma tabela em específico usa-se os comandos **SHOW INDEX FROM nomeDaTabela FROM nomeDaBaseDeDados;** ou através de **SHOW INDEX FROM nomeDaBaseDeDados.nomeDaTabela;**

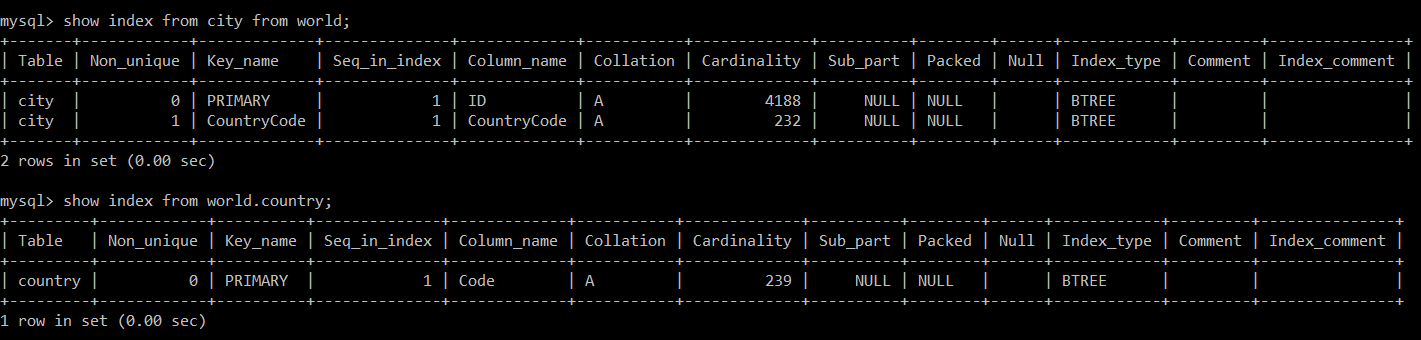


Figura 11 – Mostrando os índices de uma tabela em específico

Como retorno obtém-se os seguintes itens:

* O campo table mostra a tabela que se foi consultada;
* Non\_unique se estiver 0 significa que o índice não pode ter duplicatas, já se for preenchido como 1 pode;
* Key\_Name é o nome que o índice assume, caso seja uma primary key terá como nome primary;
* Column\_name é o nome da coluna usada pelo índice;
* Collation é a respeito de como a coluna está ordenada, se estiver como A significa que está ordenada ou se estiver com Null significa que está desordenada;
* Cardinality é uma estimativa de quantos índices com valores uniques existem, é baseado nas estatísticas e quanto maior esse índice, melhores a chances de o MySQL utiliza-lo;
* Sub\_part caso o índice possua um prefixo será mostrado, senão será preenchido como Null;
* Packed indica como o índice está guardado, se não estiver estará preenchida como Null;
* Null se a coluna possuir valores nulos será mostrado nessa coluna, se não continuará como vazio;
* Index\_type tipo de estrutura usada no índice, podendo ser Btree, Rtree, Hash ou Fulltext;
* Comment preenchida pelo próprio banco em situações especificas, como exemplo se o índice for desabilitado estará nessa coluna disabled;
* Index\_comment comentário escrito no momento da criação do índice.

# **QUESTÃO 3: CRIAR UMA VIEW NO BANCO. É POSSÍVEL INDEXAR ESSA VIEW?**

Criação da view CidadesPopulosas na base de dados world, para cidades que possuam mais de um milhão de habitantes.

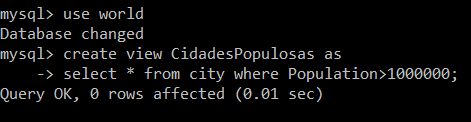


Figura 12 – Criando view

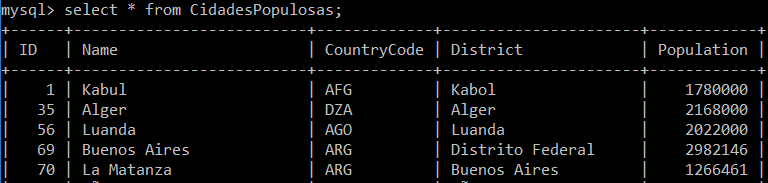
**

Figura 13 – Select na view, mostrando que foi criada com sucesso

O MySQL não permite indexar views, como pode ser observado na imagem abaixo, ele dará erro afirmando que a view não é uma tabela. As views podem utilizar de índices para um melhor desempenho, mas em si elas não podem ter nenhum índice associado.

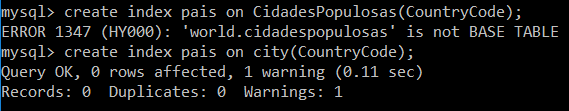
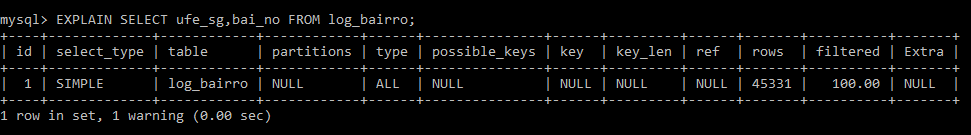


Figura 14 – Tentando adicionar um índice a view

# **QUESTÃO 4: QUAIS FERRAMENTAS DE ANÁLISE DE CONSULTA SEU SGBD OFERECE?**

Uma das ferramentas disponibilizadas pelo MySQL para análise de consultas é o comando EXPLAIN, que retorna uma tabela com as seguintes colunas:

* Select\_type – Retorna operação que será executada, podendo ser Simple, union, insert, update, delete, sub-query e etc.
* Table – Tabela onde está sendo executado o comando.
* Partitions – As partições da tabela.
* Type – Tipo de dados que podem ser retornados.
* Possible\_keys – Retorna uma possível chave.
* Key – Retorna uma chave primária caso a seleção seja por ela.
* Key\_len – Tamanho da chave.
* Ref – Chave estrangeira caso ela exista.
* Rows - Numero de linha que podem ser afetadas.
* Filtered – Porcentagem de registros filtrados sob o total de registros.
* Extra - Informações extra.



Além do comando Explain o MySQL disponibiliza o *MySQL Perfomance Schema* que monitora os eventos do servidor em um nível baixo e armazena as informações no banco performance\_schema, esse monitoramento é continuo, discreto e garante pouca sobrecarga adicional.

As informações dessa ferramenta permanecem em tabelas armazenadas na memória e não são persistidas, dessa forma ao iniciar o banco essas tabelas são repovoadas e ao encerrar o servidor elas são descartadas.

Para inicializar o armazenamento das informações de performance como cronometro é necessário habilitar através do comando de ‘UPDATE’ alguns campos em algumas tabelas que armazenam as configurações do serviço MySQL Perfomance Schema.

1. Defina como ativa a instrução e a instrumentação de estágio:

**UPDATE performance\_schema.setup\_instruments SET ENABLED = 'YES', TIMED = 'YES' WHERE NAME LIKE '%statement/%';**

**UPDATE performance\_schema.setup\_instruments SET ENABLED = 'YES', TIMED = 'YES' WHERE NAME LIKE '%stage/%';**

1. Em seguida defina como habilitado os campos **events\_statements\_\*** e **events\_stages\_** da tabela **setup\_consumers: UPDATE performance\_schema.setup\_consumers SET ENABLED = 'YES' WHERE NAME LIKE '%events\_statements\_%';**

**UPDATE performance\_schema.setup\_consumers SET ENABLED = 'YES' WHERE NAME LIKE '%events\_stages\_%';**

Obs.: Em alguns servidores pode ter essas configurações definidas por padrão.

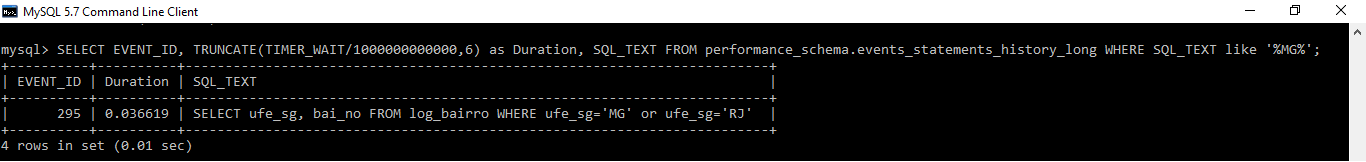
Depois de habilitado o serviço de performance você pode consultar a tabela **events\_statements\_history\_long** para encontrar os dados sobre a consulta efetuada como a consulta: **SELECT <Campos que deseja selecionar> FROM performance\_schema.events\_statements\_history\_long WHERE SQL\_TEXT like '%<Algum dado que identifique a consulta alvo, pode ser a sintaxe de filtro ‘where’ que a consulta alvo realizou e etc.%';**

# **QUESTÃO 5: IMPLEMENTE UMA CONSULTA ‘PESADA’ NO SEU BANCO.**

Nesse exercício foi utilizado o banco Logradouro, disponibilizado em aula, mas com nome COM231 na base de dados, como banco de teste, pois as suas tabelas são mais populosas que o banco World disponível por padrão do MySQL

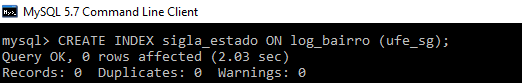
## Verifique o tempo necessário para processar essa consulta.

Foi efetuada no banco a consulta “**SELECT ufe\_sg, bai\_no FROM log\_bairro WHERE ufe\_sg='MG' or ufe\_sg='RJ';”** que se concluiu em 0.036619(ms).

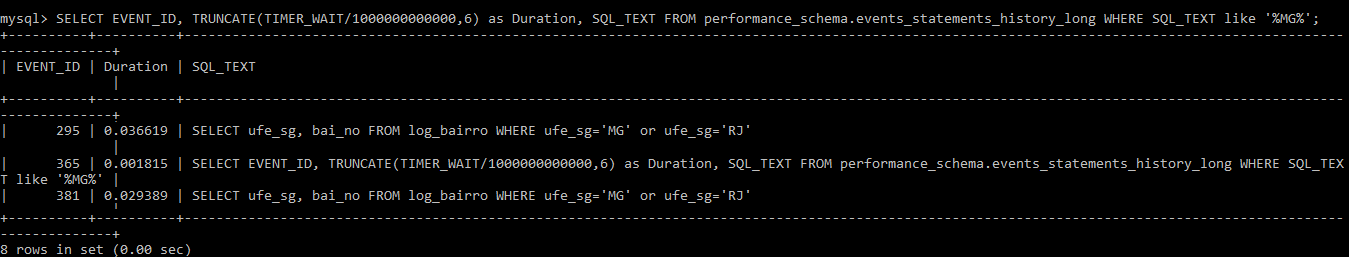
 Figura 16 – Cronômetro da consulta.

## Agora crie um índice e refaça a consulta. Diminuiu o tempo de consulta? Justifique o índice criado.

Foi criado um index para coluna “**ufe\_sg”** visando que a consulta efetua um filtro nessa coluna, dessa forma o index otimizará com sucesso as consultas que buscam bairros por estados federativos.

 Figura 17 – Criando index para coluna “ufe\_sg”.

# Ao efetuar a consulta novamente foi possível observar uma diminuição no tempo de consulta, que passou de 0.036619(ms) para 0.29389(ms).

 Figura 18 – Cronômetro das consultas com uso de index e sem o uso do mesmo.

# **QUESTÃO 6: AVALIE POR MEIO DE CONSULTAS E DESCREVA A IMPRESSÃO DO GRUPO SOBRE:**

## Índice de Hash

Um índice hash foi criado para uma tabela com 4079 linhas com nomes e informações adicionais de cidades. O índice hash foi gerado em cima dos nomes das cidades, a consulta foi realizada com busca pela cidade de Columbia.



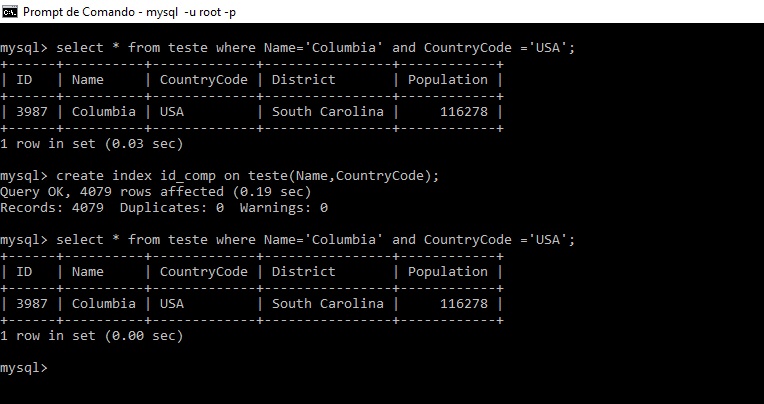
Na imagem acima primeiro realizamos a verificação da engine da tabela, já que segundo o manual do mysql a engine InnoDB não implementava um índice do tipo hash. A tabela a princípio era InnoDB então alteramos sua engine para MyISAM. Após a alteração foi efetuada uma consulta buscando informações da cidade de Columbia, e o tempo obtido foi de 0.06 segundos para esta consulta sem o índice hash.

Em seguida criamos um índice do tipo hash em cima do campo Name, e realizamos novamente a mesma consulta, e o tempo agora foi de 0.00 segundo.

Neste caso o índice hash reduziu o tempo de consulta de maneira significativa.

## Índice composto

Um índice composto foi criado para uma tabela com 4079 linhas com nomes e informações adicionais de cidades. O índice composto referia ao nome da cidade (coluna Name) e ao código do pais a qual ela pertence (coluna CountryCode).

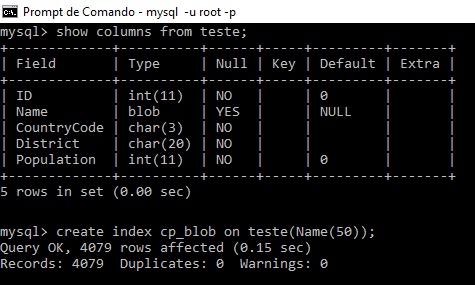


Na imagem acima realizamos uma consulta na tabela buscando pela cidade Columbia que possuía o CountryCode igual a ‘USA’. Sem o índice esta consulta levou 0.03 segundos. Após a consulta criamos um índice composto em cima dos campos Name e CountryCode, e realizamos novamente o mesmo select, obtendo o resultado em um tempo de 0.00 segundos.

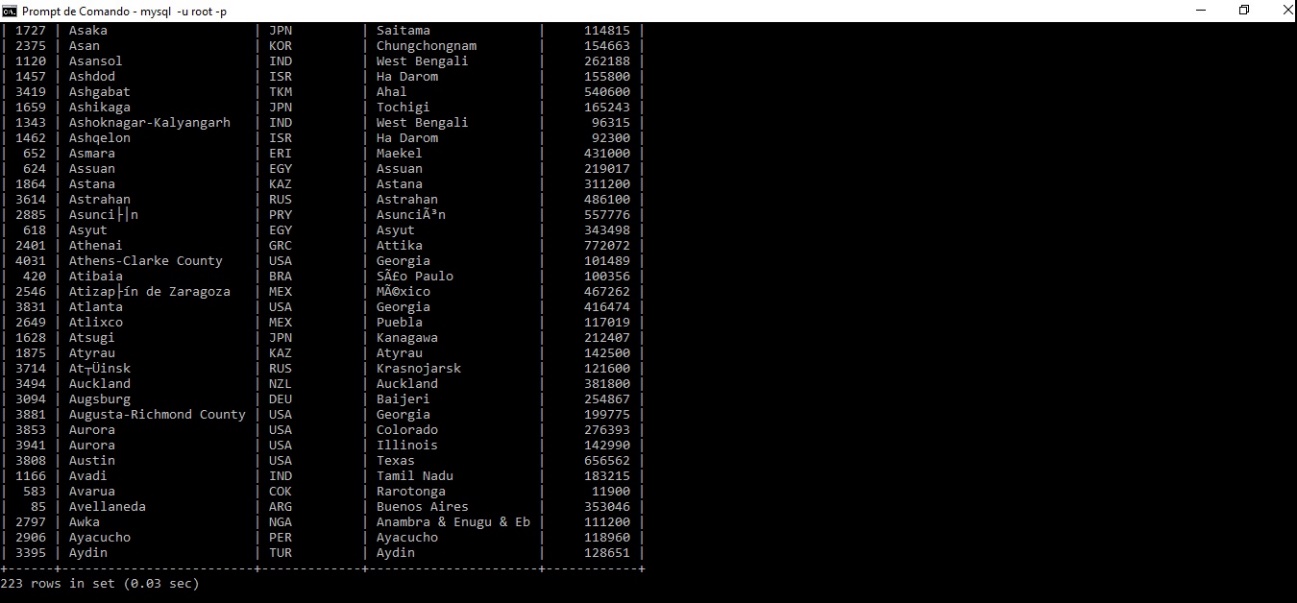
Notamos novamente uma redução no tempo de busca.

## Índice em um campo BLOB

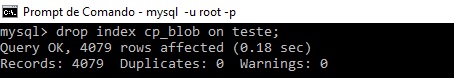
Um índice comum foi criado para um campo do tipo blob em uma tabela com 4079 linhas com nomes e informações adicionais de cidades.



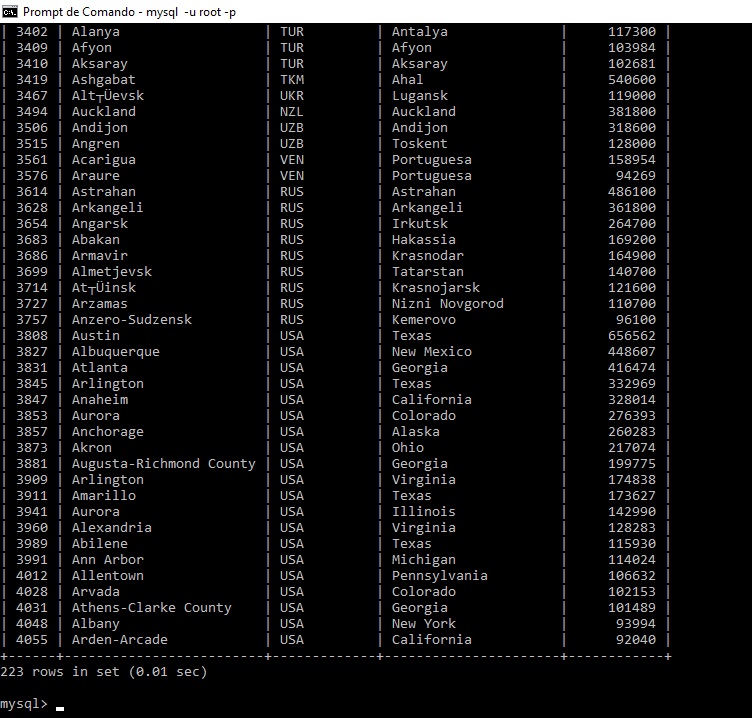
Após a criação do índice foi realizada uma consulta por todas as cidades que começavam com a letra A e o resultado foi obtido em 0.03 segundos conforme a imagem a seguir.



Em seguida o índice foi excluído.



E a consulta foi realizada novamente, e o tempo obtido foi de 0.01 segundos.



Neste caso o índice em cima do campo blob piorou a consulta.

# **QUESTÃO 7: QUAIS SÃO AS FORMAS POSSÍVEIS DE INICIAR UMA TRANSAÇÃO NO SGBD?**

Uma transação pode ser necessária quando um conjunto de instruções SQL devem ser considerados como uma única unidade de trabalho. Ou seja, se toda aquela transição ocorrer você pode persistir a mudança no banco, caso uma delas falhe, você deve refazer todos os passos já realizados e recomeçar o conjunto de instruções.

Os possíveis comando utilizados em mysql para iniciar uma transação são:

* BEGIN (...) END
* BEGIN WORK
* START TRANSACTION

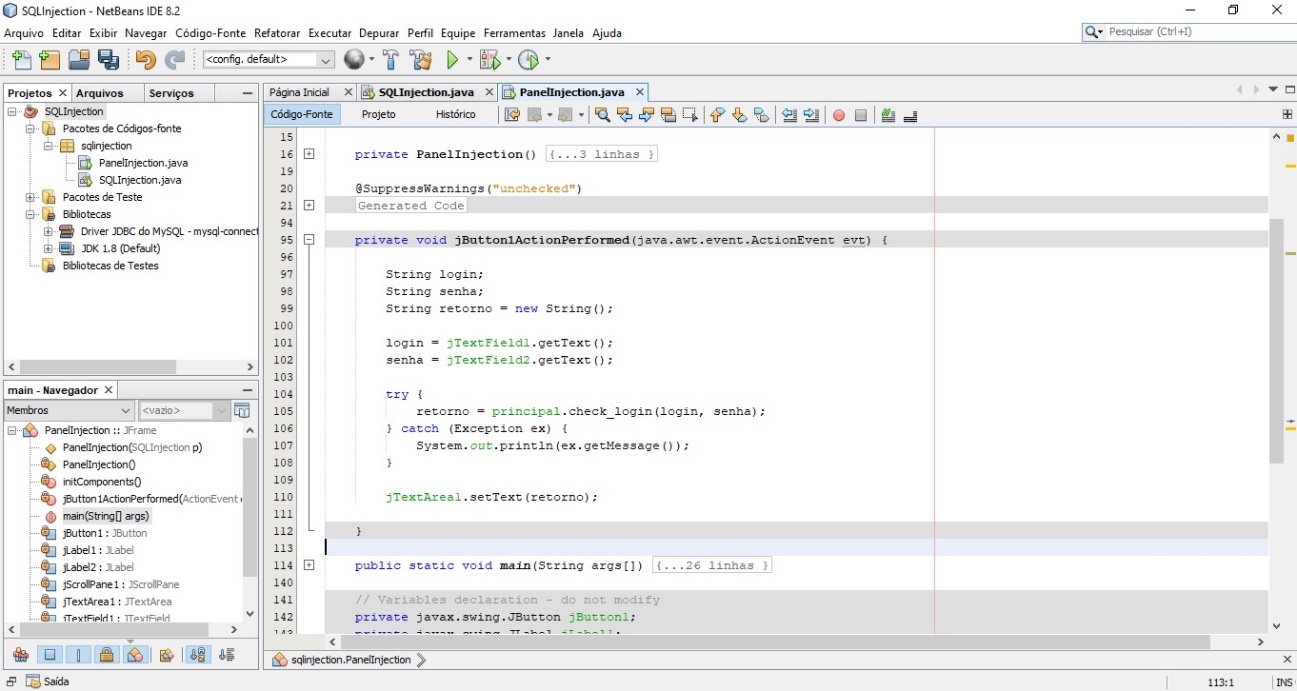
# **QUESTÃO 8: PESQUISE SOBRE O CHAMADO SQL INJECTION (O QUE É, COMO PREVINIR).**

O chamado SQL Injection é um tipo de ataque que se baseia na manipulação do código SQL. Quando uma aplicação recebe um valor digitado pelo usuário e concatena isso a uma string com código SQL para uma consulta em um banco de dados, a maneira que esta string é concatenada pode gerar uma abertura para que partes de código SQL sejam enviadas para a variável que coleta os dados digitados pelo usuário e enviem para manipular a consulta pré-determinada no momento em que a concatenação é feita.

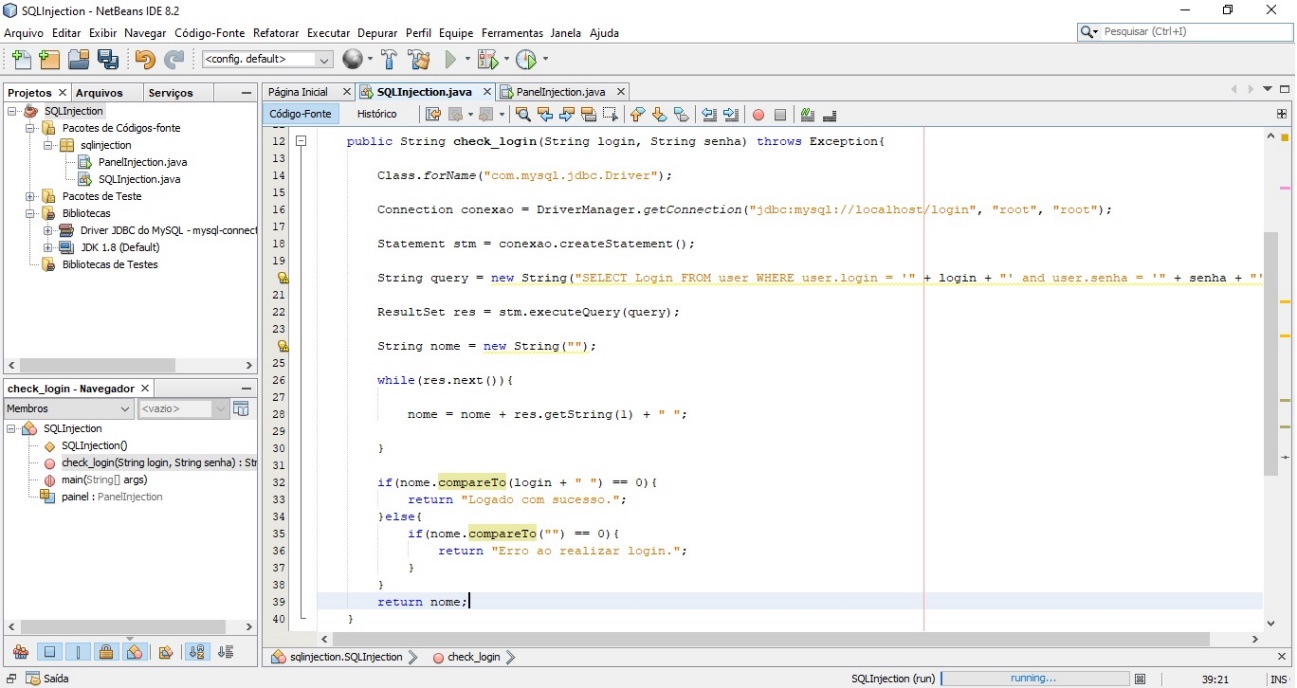
Para evitar isso podemos realizar na aplicação uma limpeza da variável recebida, ou seja, a variável que guarda os dados inseridos pelo usuário deve apenas conter letras e números por exemplo. Para que o usuário insira um SQL Injection ele deverá inserir no campo parte de um código SQL que necessariamente possui caracteres especiais, quando realizamos um filtro/limpeza nesta variável impossibilitamos que código SQL seja enviadas para a consulta no banco.

# **QUESTÃO DESAFIO: Implemente uma aplicação e simule um SQL Injection.**

A seguir uma aplicação em java foi implementada para simular um SQL Injection. Não foi postado o código todo por questões de espaço porém as funções de acesso ao banco de dados e o que ocorre após clica no botão OK na tela de login estão mostradas a baixo e explicadas.

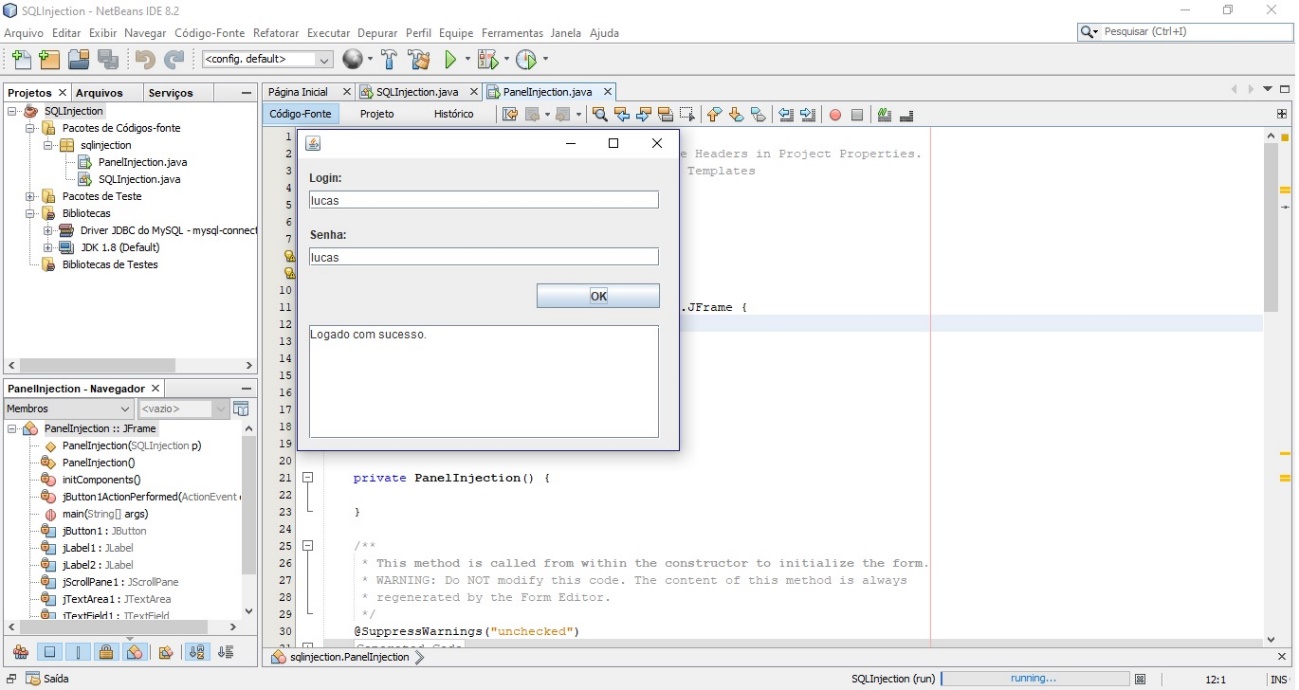


Primeiro temos a parte do código que recebe a ação de clique no botão OK. Quando o usuário clica no botão o código coleta os dados do campo login e senha em variáveis separadas, uma de login outra de senha para efetuar a buscar no banco de dados e permitir ou não o acesso do usuário.

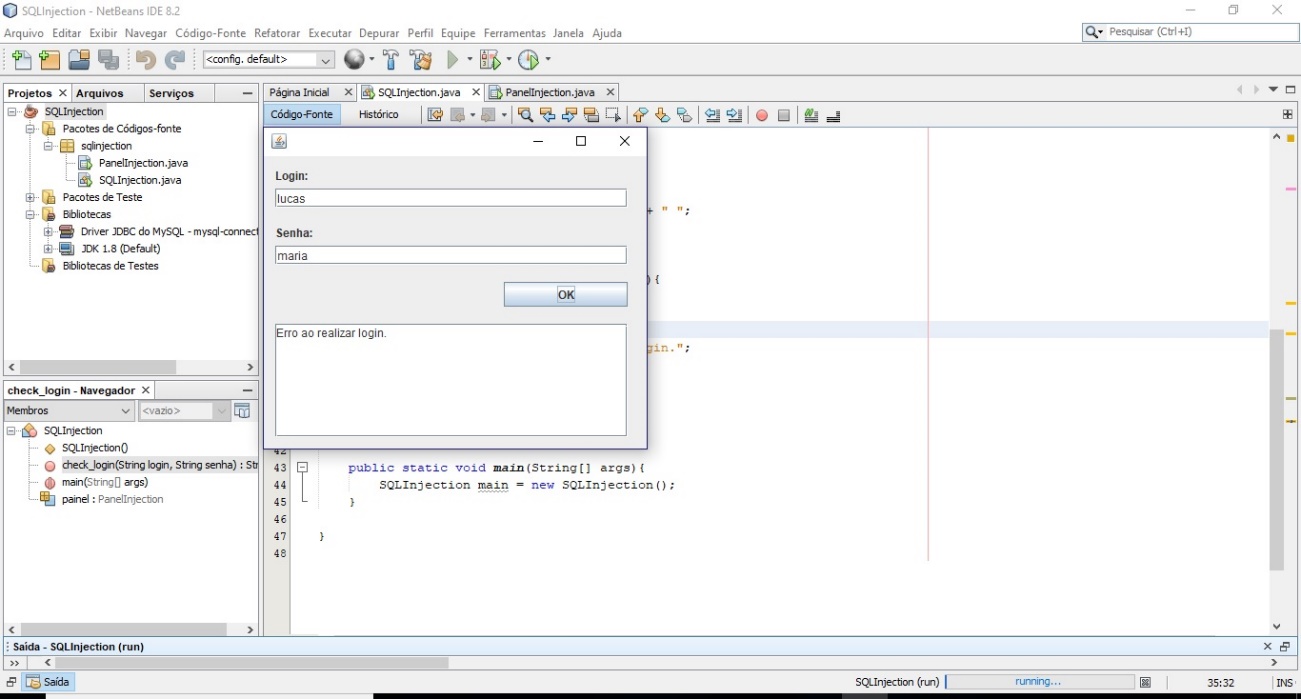


Após clicado no botão OK a aplicação realiza uma conexão ao banco de dados e realiza uma busca pelo usuário informado para verificar se a senha está correta e se o usuário poderá logar na aplicação. Note que, a abertura para o SQL Injection se dá nesta parte do código, onde as variáveis recebidas da tela são inseridas a uma string de código SQL sem nenhum tipo de tratamento.

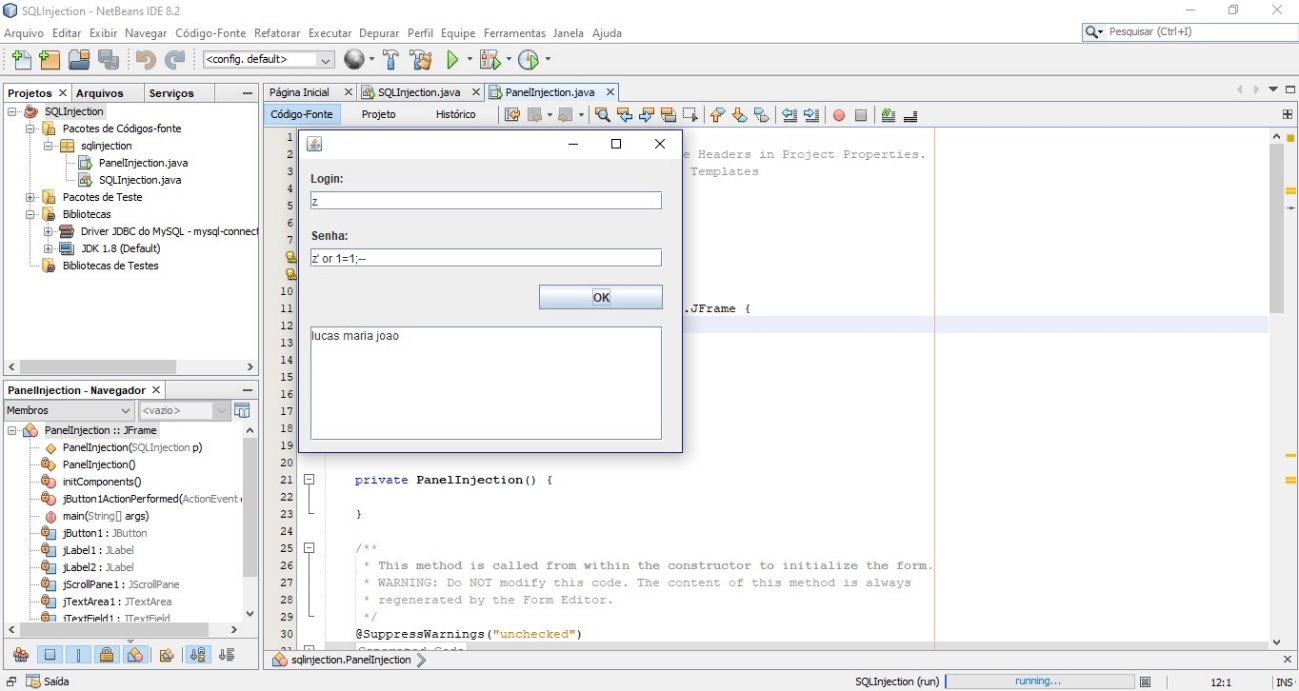
Quando o usuário digita um login e senha válido o sistema informa que o login foi realizado com sucesso como na imagem a seguir.



Quando a senha é invalida ou o usuário invalido o sistema informa como na imagem a seguir.



Agora um exemplo de aplicação de um código SQL no campo onde o usuário deveria informar a senha.



Note que ao invés do sistema informar que aquela senha e/ou usuário são inválidos quando concatenada a string do select na aplicação o campo senha altera o código SQL. Assim a string após a coleta dos campos fica:

SELECT Login FROM user WHERE user.login = ‘z’ and user.senha = ‘z’ or 1=1;-- ’;

Percebam que o – no fim do campo login fez com que o código SQL feito pelo desenvolvedor agora é um comentário (marcado em azul no SQL acima), e agora por conta do 1=1 ao invés de só retornar o nome do usuário caso o mesmo informe dados válidos de login e senha, agora o sistema lista todos os usuários cadastrados no sistema, uma informação que não deveria ser acessada por um usuário que tenta se logar no sistema.

# **REFERÊNCIA:**

ORACLE: **MySQL 5.7 Reference Manual**, 2018. Disponível em: < https://dev.mysql.com/doc/refman/5.7/en/storage-engines.html >. Acesso em 07/04/2018.