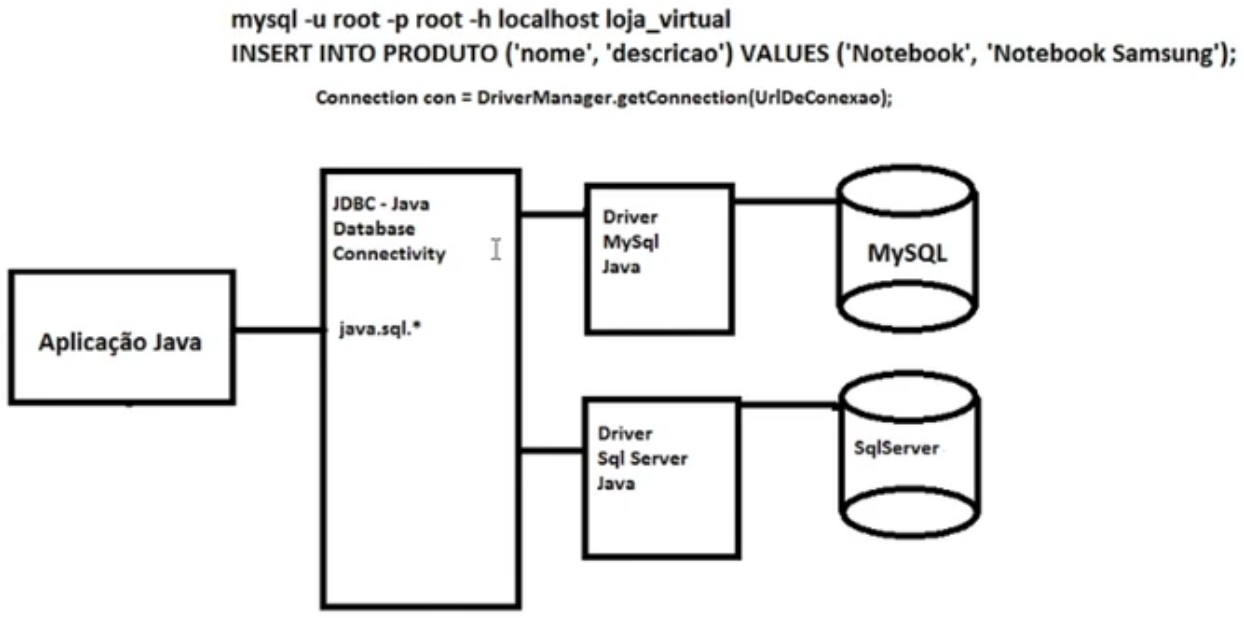
03

Usando um banco de dados

Transcrição

No desenho, no começo do vídeo, é mostrado o comando necessário para inserir um produto: `INSERT INTO PRODUTO ('nome', 'descricao') VALUES ('Notebook', 'Notebook Samsung');` Esse comando não irá funcionar caso seja executado, pois não há aspas simples (') nos atributos da tabela. O correto seria: `INSERT INTO PRODUTO (nome, descricao) VALUES ('Notebook', 'Notebook Samsung');`

[00:00] Fala, aluno, tudo bom? Na última aula, nós aprendemos como configurar o nosso banco de dados. Uma vez configurado, nós conseguimos recuperar a conexão, nós conseguimos nos conectar no banco de dados e realizar alguns comandos. No nosso caso, nós inserimos um produto na nossa tabela Produto.



[00:22] A nossa motivação agora é que nós consigamos fazer essa mesma coisa, recuperar uma conexão, ou inserir um produto, ou listar um produto, mas agora a partir da nossa aplicação Java. A grande questão aqui é que a nossa aplicação, ela não consegue conversar com o nosso banco de dados de forma nativa.

[00:46] Aqui nós temos uma aplicação Java rodando Java, e aqui, do outro lado, nós temos um banco de dados que roda um protocolo conhecido só por ele. Então o Java não tem como chegar no banco de dados e simplesmente fazer uma conexão com o MySQL, por exemplo. Então, como nós vamos conseguir fazer essa conexão?

[01:09] Para facilitar a vida do desenvolvedor, os desenvolvedores, a equipe do MySQL, ela criou para nós uma biblioteca Java. Uma biblioteca Java aqui, entre a aplicação e o banco de dados. A biblioteca vai conhecer todo esse lado do MySQL e vai expor para a nossa aplicação, então ela vai expor de uma forma que a nossa aplicação conheça tudo o que eu preciso para me comunicar com o MySQL.

[01:52] Então essa biblioteca vai ser o nosso famoso JAR, o JAR que nós já conhecemos da nossa linguagem Java. Então esse JAR, ele é desenvolvido pela equipe de desenvolvimento do MySQL. Quando a minha aplicação chega no JAR, eu tenho alguns conjuntos no JAR de classes, de interfaces, que dado determinado comando, vai conseguir chegar no meu banco de dados.

[02:27] E essa biblioteca aqui, ela tem um nome, ela é na verdade um driver. Então para ficar mais bonito o desenho, vamos colocar o nome que é dado para essa biblioteca, esse JAR. Então ela é um driver. Então tenho aqui o meu driver MySQL Java. Agora, com esse driver, se eu quiser me comunicar, por exemplo, com outro banco de dados, que eu vou pegar, por exemplo, um SQL Server.

[03:10] Então agora eu não tenho mais um MySQL, eu tenho um SQL Server. Eu vou precisar só pegar um driver SQL, que a nossa aplicação vai conseguir se comunicar com o banco de dados SQL Server. Então seria algo semelhante a

um driver SQL Server do Java. Uma vez que tenho esse novo driver, agora a minha aplicação passa a conhecer o protocolo que é utilizado no SQL Server. Então ligar aqui para o nosso desenho ficar bonito.

[04:04] Só que nesse segundo banco, já começamos a perceber um problema. Porque, o que acontece? No primeiro driver eu tenho um conjunto de classes e interfaces do MySQL. No segundo driver, eu tenho um conjunto de classe e interfaces do SQL Server. Então dificilmente nós vamos ter chamadas iguais, vamos ter classes iguais.

[04:28] Então, para representar o que eu estou querendo falar, vamos supor que eu quero pegar uma conexão do MySQL e na classe `MySqlConnector`, por exemplo, eu tenho um `MySqlConnector.getConnection();`, que vai receber alguns parâmetros, como `(usuário, )`, que nós passamos para o nosso banco de dados, `MySqlConnector.getConnection(usuario, senha, db, servidor);` e etc.

[05:05] Muito dificilmente vamos ter algo igual do lado do SQL Server. Então, por exemplo, vamos supor que o pessoal do SQL Server criou aqui uma classe `SqlServerConnectionProvider` e tem um método `SqlServerConnectionProvider.connect()`, que também pode receber um `SqlServerConnectionProvider.connect(usuário, senha);`, enfim, pode receber os atributos que precisam de conexão com o SQL Server.

[05:52] Então aqui nós já vemos um problema: se eu precisar mudar do MySQL para o SQL Server, eu vou ter que, na hora de pegar a conexão, vou ter que alterar a chamada desse método para eu poder utilizar um outro tipo de banco. Aqui nós estamos trabalhando com duas opções, mas ainda temos bancos como PostgreSQL e qualquer um que quiséssemos nos conectar dessa maneira, seria bem trabalhoso.

[06:28] Então foi aí que o Java nos facilitou com uma camada de abstração, que vai ficar antes desses drivers. Então aqui, para ajeitar o nosso desenho, vamo mudar a forma como estávamos pensando aqui. E eu tenho aqui, após a

aplicação Java, essa abstração, que vai ficar entre os drivers e vai ficar entre a minha aplicação.

[07:14] Essa abstração, ela é chamada de JDBC, ou melhor dizendo, Java Database - calma, ele não pulou linha, ficou ruim de enxergar. Então vai ser Java Database Connectivity. Agora, esse JDBC também vai ter uma abstração, ou seja, essas bibliotecas, os drivers aqui, deverão implementar os métodos que eu tenho no meu JDBC. Agora a minha aplicação só precisa conhecer esse JDBC.

[08:00] Então com a minha aplicação conhecendo JDBC, o JDBC dado algum comando, ele vai saber para qual aplicação ou para qual banco de dados que eu quero me conectar. Esse JDBC nada mais é do que o nosso pacote java.sql. Então tudo o que tem dentro de java.sql, é o nosso JDBC, essa nossa camada, essa nossa abstração da conexão com o banco de dados.

[08:43] E para conseguirmos pegar uma conexão com qualquer banco de dados que quisermos nos comunicar, eu vou ter no JDBC uma interface chamada "Connection com" e eu tenho no java.sql um `Connection con = DriverManager();` , que ele vai pegar uma conexão. Como ele vai saber qual é a conexão que é para recuperar? Dentro dos parênteses, eu vou ter uma `Connection con = DriverManager(UrlDeConexao);` .

[09:22] Com essa URL, eu vou passar qual é o tipo do banco de dados que eu quero me comunicar, eu vou falar qual é o meu usuário, qual é a minha senha, vou falar onde que está esse banco de dados e qual é a minha Database. Uma vez que eu tenho aqui um *connection*, eu tenho a minha conexão recuperada, consigo agora, da minha aplicação, realizar qualquer comando que eu quiser.

[09:55] Então nesse desenho, nós já conseguimos ver que dessa maneira, pouco importa qual é o tipo de banco que eu vou ter do outro lado. Com a minha camada de abstração, com as minhas interfaces e com esse conjunto de classes que o JDBC traz para nós, essa conexão, para nós, fica bem mais simples, fica

bem mais fácil de se conectar a um banco de dados. Então é isso, aluno. Espero que tenham gostado e até a próxima aula.