

INSTITUTO MAUÁ DE TECNOLOGIA




Linguagens I

Java + Banco de Dados MySQL
JavaFX ListView

Profº. Tiago Sanches da Silva
Prof. Murilo Zanini de Carvalho

Retomando da Aula Anterior

- Persistência de dados consiste em armazenar um conjunto de dados para uso posterior por um conjunto de aplicações.
 - Idealmente, ao persistir um conjunto de dados, esses devem possuir grande disponibilidade, garantia de integridade e possibilidade de acesso por múltiplas instâncias.
- 

Retomando da Aula Anterior

- Uma das formas de manipular esses dados é utilizando bando de dados.



Retirado de (<https://i.ytimg.com/vi/etReM7odebE/maxresdefault.jpg>), em 02/09/2018

Retirado de (https://zhangliting.github.io/images/main_databases.jpg), em 02/09/2018

Persistência através de sockets

É possível conectar-se com qualquer base de dados através da abertura de um socket TCP com o servidor que o hospeda, por exemplo um Oracle ou MySQL e nos comunicarmos com ele através de seu protocolo proprietário.

Porém conhecer o protocolo proprietário complexo em profundidade é difícil, e trabalhar com ele é muito trabalhoso.

Conectar-se a um banco de dados com Java é feito de maneira elegante. Para **evitar** que cada banco tenha a sua própria API e conjunto de classes e métodos, temos um único conjunto de **interfaces** muito bem definidas que devem ser implementadas.

Esse conjunto de interfaces fica dentro do pacote **java.sql** e nos referiremos a ela como **JDBC**.

Java DataBase Connectivity



Conexão em Java

Interfaces java.sql:

<https://docs.oracle.com/javase/8/docs/api/java/sql/package-summary.html>

JDBC API:

<https://docs.oracle.com/javase/8/docs/technotes/guides/jdbc/>



O que é o JDBC?

Pode-se dizer que é uma **API** (Interface de Programação de Aplicativos) que **reúne conjuntos de classes e interfaces** escritas na linguagem Java na qual possibilita se conectar através de um **driver específico** do banco de dados desejado.

Com esse driver pode-se executar instruções SQL de qualquer tipo de banco de dados relacional.

Para fazer a comunicação entre a **aplicação** e o Banco de Dados é necessário possuir um **driver para a conexão desejada**. Geralmente, as empresas de Banco de Dados oferecem o driver de conexão que seguem a especificação **JDBC**.

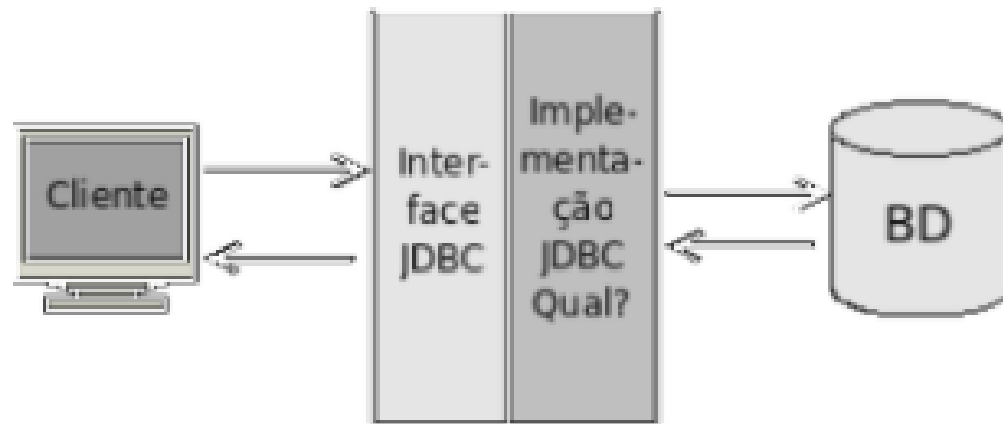
MySQL: <https://dev.mysql.com/downloads/connector/j/>



Driver?

Caso queiramos trabalhar com o **MySQL**, precisamos de classes concretas que implementem essas interfaces do pacote **java.sql**.

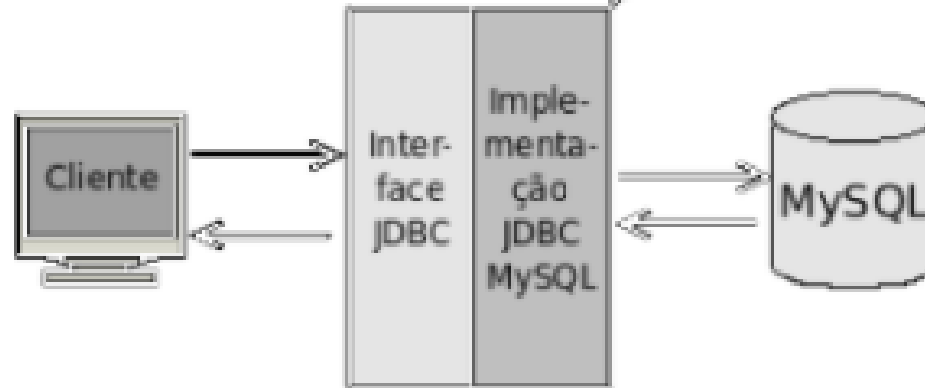
Esse conjunto de classes concretas é quem fará a ponte entre o código cliente que usa a **API JDBC** e o **banco de dados**. São essas classes que sabem se **comunicar através do protocolo proprietário** do banco de dados. Esse conjunto de classes recebe o nome de **driver**.



Driver?

Todos os principais bancos de dados do mercado possuem **drivers JDBC** para que você possa utilizá-los com Java.

```
DriverManager.getConnection("jdbc:mysql://localhost/teste");
```



Atenção

Importe do java.sql

Existe um ponto de atenção na importação das classes ou interfaces relacionadas ao pacote a ser usado no momento do desenvolvimento.

A correta a importação do pacote referente à classe Connection pertencente ao pacote **java.sql**.

Esse é um fator a ser observado com cautela, pois isso é considerado um dos erros mais comuns justamente pelo fato do desenvolvedor pensar muitas vezes em usar o pacote **com.mysql.jdbc** sendo que está utilizando o **driver JDBC** do banco **MySQL**.

Pacote java.sql

Pacote java.sql

Esse pacote oferece a biblioteca Java o acesso e processamento de dados em uma banco de dados. As classes e interfaces mais importantes são:

Classe	Interface
DriverManager	Driver
	Connection
	Statement
	ResultSet
	PreparedStatement

DriverManager

Para abrir uma conexão com um banco de dados, precisamos utilizar sempre um driver. A classe **DriverManager** é a responsável por se comunicar com todos os drivers que você deixou disponível.

Para isso, invocamos o método estático **getConnection** com uma **String** que indica a qual banco desejamos nos conectar.

Essa **String** - chamada de **String de conexão JDBC** - que utilizaremos para acessar o **MySQL** tem sempre a seguinte forma:

```
jdbc:mysql://ip/nome_do_database
```

DriverManager - Exemplo

jdbc:mysql://ip/nome_do_database

```
DriverManager.getConnection("jdbc:mysql://localhost/alunosteste", "root", "XXXXXXXX");
```

Ao tentar executar essa linha sem ter carregado o pacote com o driver correto receberemos uma exception.

```
java.sql.SQLException: No suitable driver found for
```

A conexão não pôde ser aberta por que?

Carregar o pacote do driver para o projeto

- O que precisamos fazer é adicionar o driver do **MySQL** ao **classpath**, o arquivo **.jar** contendo a implementação **JDBC** do **MySQL** (**mysql connector**) precisa ser colocado em um lugar visível pelo seu projeto ou adicionado à variável de ambiente **CLASSPATH**.
- O arquivo jar pode ser encontrado em:
<https://dev.mysql.com/downloads/connector/j/8.0.html>

Generally Available (GA) Releases


Connector/J 8.0.12

Select Operating System:

Platform Independent ▾

Looking for previous GA versions?

Platform Independent (Architecture Independent), Compressed TAR Archive (mysql-connector-java-8.0.12.tar.gz)	8.0.12	4.8M	Download
MD5: 368c686aec816ca7d2ad9c98ac0e739f Signature			
Platform Independent (Architecture Independent), ZIP Archive (mysql-connector-java-8.0.12.zip)	8.0.12	5.5M	Download
MD5: d7f2ef57f603d245dc7b86db2941ccd6 Signature			

 We suggest that you use the [MD5 checksums](#) and [GnuPG signatures](#) to verify the integrity of the packages you download.

Adaptado de
(<https://dev.mysql.com/downloads/connector/j/8.0.html>), em
02/09/2018

Mas eu vi em algum tutorial...

E o `Class.forName()`?

Até a versão 3 do JDBC, antes de chamar o `DriverManager.getConnection()` era necessário registrar o driver JDBC que iria ser utilizado através do método `Class.forName("com.mysql.jdbc.Driver")`, no caso do MySQL, que carregava essa classe, e essa se comunicava com o `DriverManager`.

A partir do JDBC 4, que está presente no Java 6, esse passo não é mais necessário. Mas lembre-se: caso você utilize JDBC em um projeto com Java 5 ou anterior, será preciso fazer o registro do Driver JDBC, carregando a sua classe, que vai se registrar no `DriverManager`.

Interface Connection

Representa uma conexão ao banco de dados. Nessa interface são apresentados os métodos mais utilizados.

Caso o **DriverManager** consiga realizar a conexão com o banco de dados ele retorna uma instancia de um objeto **Connection**.

Com ele você conseguirá executar **queries**.

Editando o Banco de Dados

- Para editar os bancos no MySQL, é possível fazer essa tarefa apenas com a interface de linha de comando (CLI) ou utilizando o Workbench.



MySQL Shell

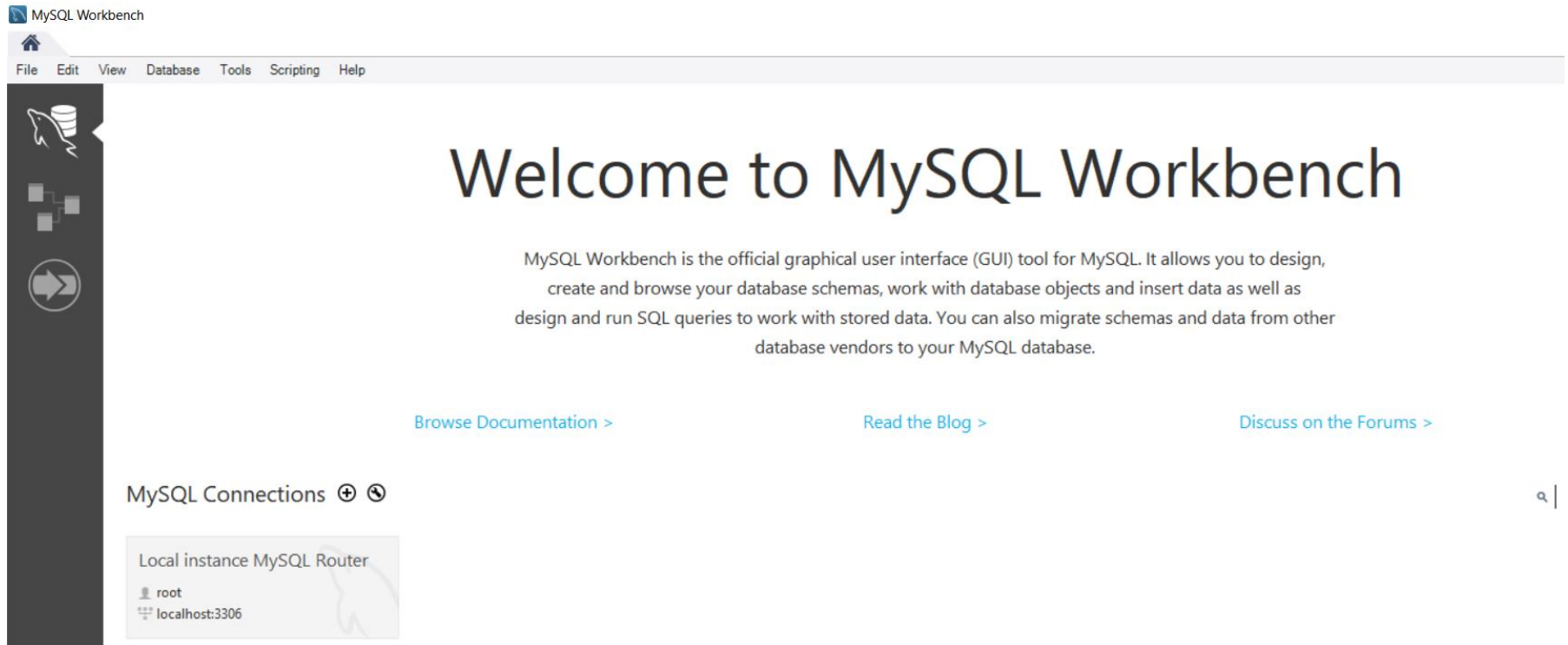
Aplicativo da área de trabalho



MySQL Workbench 6.3 CE

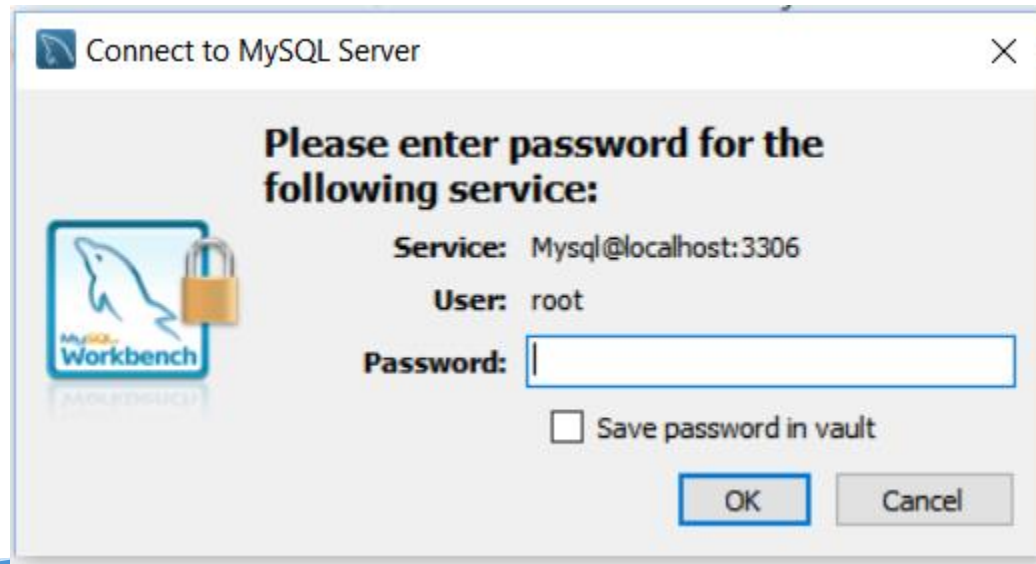
Aplicativo da área de trabalho

- Interface padrão do Workbench

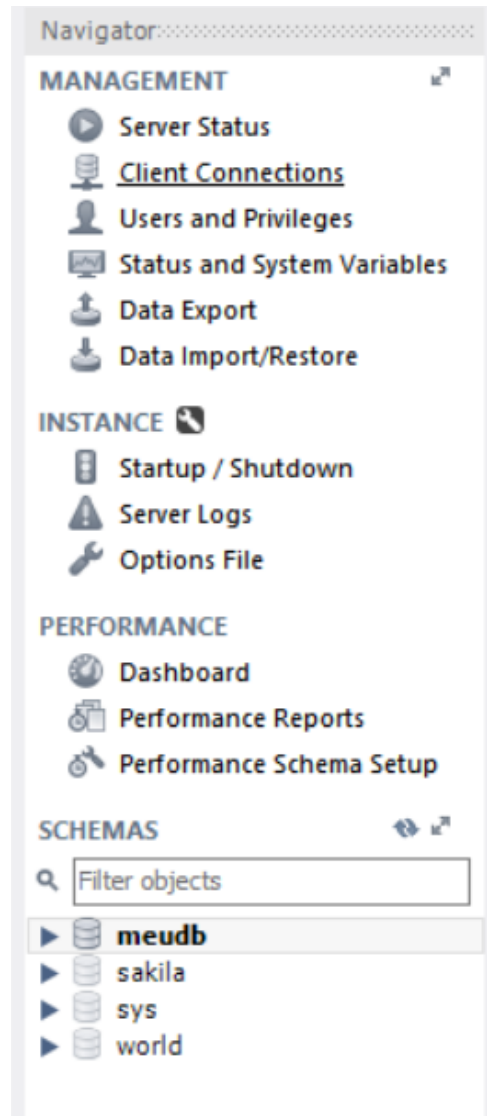


Editando o Banco de Dados

- Por padrão, o usuário root precisa ter uma senha para garantir que ninguém faça alterações nas bases de dados sem permissão. Senha padrão na sala: 431620416



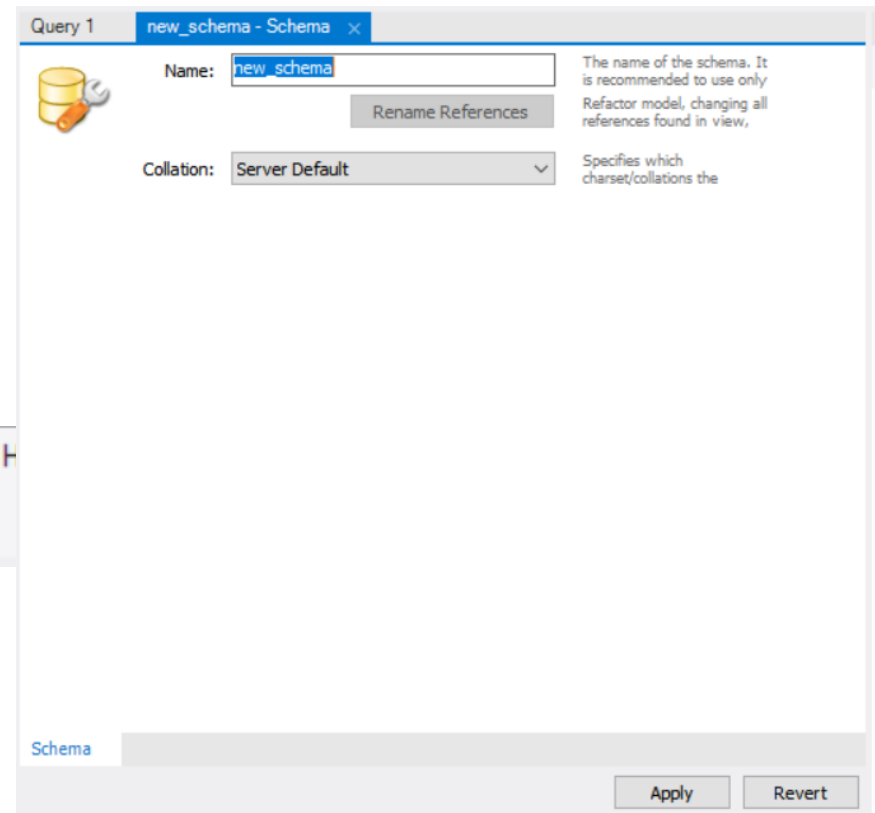
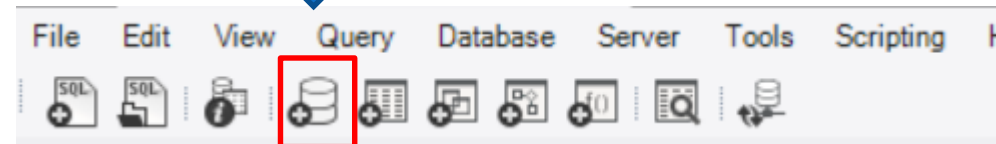
Editando o Banco de Dados



- Do menu “Navigator”, cada Schema representa um novo banco com suas próprias tabelas de dados.

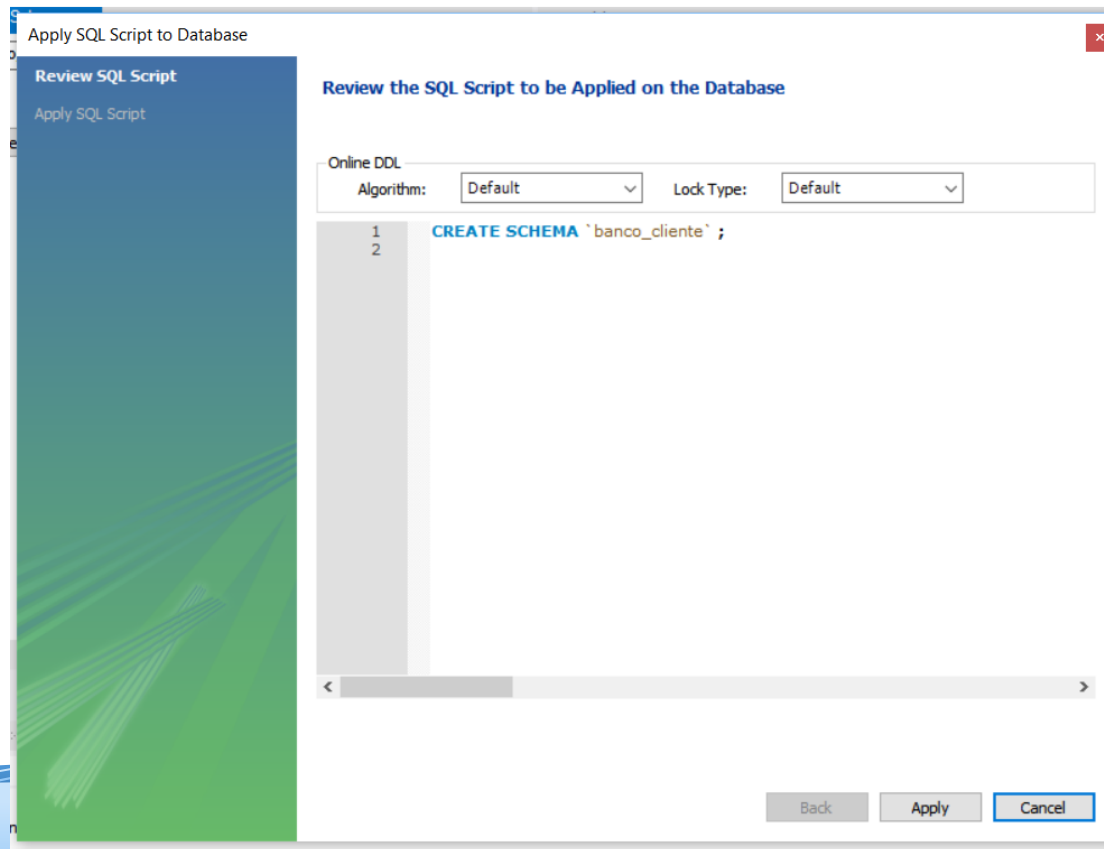
Editando o Banco de Dados

- Crie um novo schema com o nome meudb.



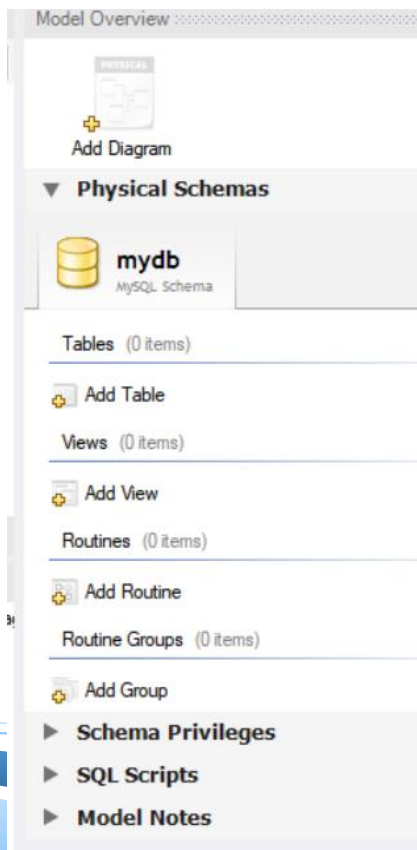
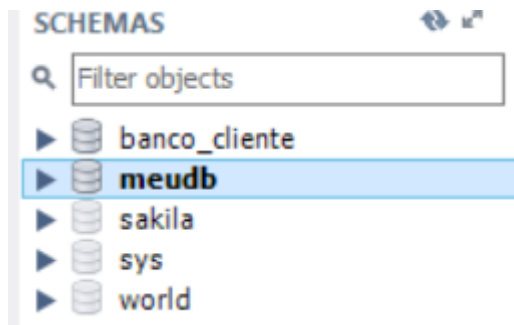
Editando o Banco de Dados

- O que o Workbench vai fazer é permitir editar os comandos enviados para o CLI de forma gráfica.



Editando o Banco de Dados

- Com o Schema criado ativo, vamos editar as tabelas de dados dentro do banco. File-> New Model. Depois “Add Diagram”.

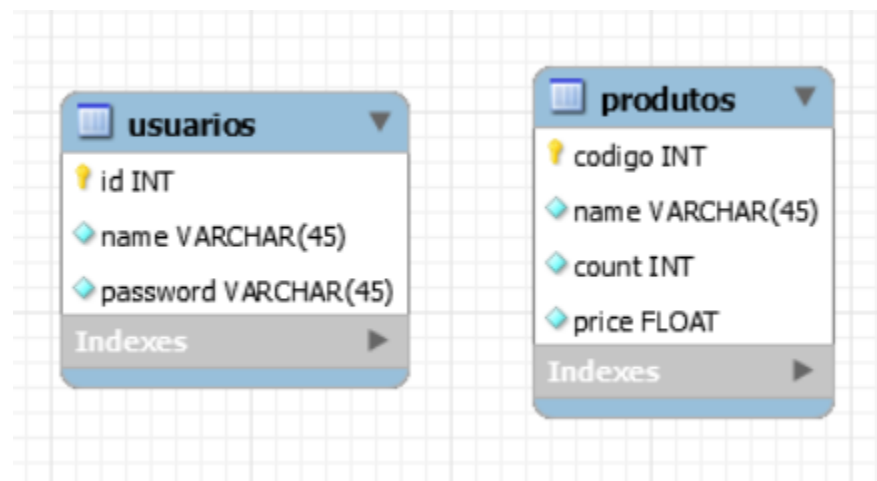


Editando o Banco de Dados

- Com o diagrama de classes é possível definir a forma como as tabelas estão modeladas dentro do banco de dados.
- Lembre-se, um bom modelo é aquele que descreve melhor as características do serviço/produto modelado.
- Um modelo não precisa ser composto por uma única tabela gigante, mas pode ser dividido em diversas tabelas menores relacionadas.

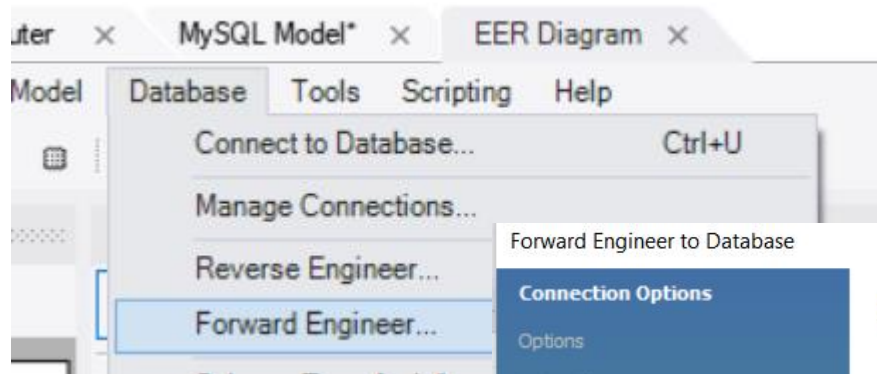
Editando o Banco de Dados

- Crie duas tabelas no banco, uma para representar os usuários do sistema e outra para representar os produtos.



Editando o Banco de Dados

- Para aplicar o diagrama no banco, selecionar a ferramenta “Forward Engineer”.



Forward Engineer to Database

Connection Options

Options

Select Objects

Review SQL Script

Commit Progress

Set Parameters for Connecting to a DBMS

Stored Connection: Local instance MySQL Router Select from saved connection settings

Connection Method: Standard (TCP/IP) Method to use to connect to the RDBMS

Parameters SSL Advanced

Hostname: localhost Port: 3306 Name or IP address of the server host - and TCP/IP port.

Username: root Name of the user to connect with.

Password: Store in Vault ... Clear The user's password. Will be requested later not set.

Default Schema: The schema to use as default schema. Leave blank to select it later.

Editando o Banco de Dados

Set Options for Database to be Created

Tables

- ☐ Skip creation of FOREIGN KEYS
- ☐ Skip creation of FK Indexes as well
- ☐ Generate separate CREATE INDEX statements
- ☐ Generate INSERT statements for tables
- ☐ Disable FK checks for INSERTs

Other Objects

- ☐ Don't create view placeholder tables
- ☐ Do not create users. Only create privileges (GRANTS)

Code Generation

- ☐ DROP objects before each CREATE object
- ☐ Generate DROP SCHEMA
- ☐ Omit schema qualifier in object names
- ☐ Generate USE statements
- ☐ Add SHOW WARNINGS after every DDL statement
- ☒ Include model attached scripts

Back

Next

Cancel

Select Objects to Forward Engineer

To exclude objects of a specific type from the SQL Export, disable the corresponding checkbox. Press Show Filter and add objects or patterns to the ignore list to exclude them from the export.



☒ Export MySQL Table Objects

2 Total Objects, 2 Selected

Show Filter



☐ Export MySQL View Objects

0 Total Objects, 0 Selected

Show Filter



☐ Export MySQL Routine Objects

0 Total Objects, 0 Selected

Show Filter

☐ Export MySQL Trigger Objects

0 Total Objects, 0 Selected

Show Filter

☐ Export User Objects

0 Total Objects, 0 Selected

Show Filter

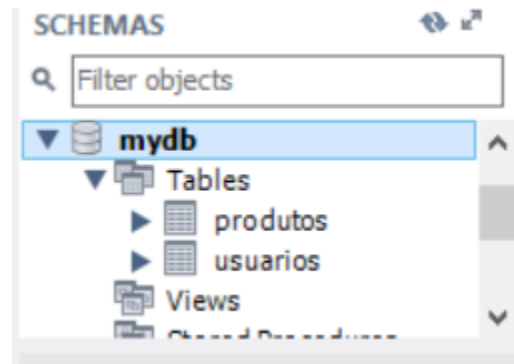
Back

Next

Cancel

Editando o Banco de Dados

- Dentro do Navigator, as tabelas já estaram criadas no schema selecionado.



Editando o Banco de Dados

- Adicionar alguns usuários na tabela de usuários e verificar se a adição ocorreu sem problemas.

```
1 • INSERT INTO usuarios VALUES (null, 'Goku', '1234');  
2 • INSERT INTO usuarios VALUES (null, 'Vegeta', '8001');  
3 • INSERT INTO usuarios VALUES (null, 'root', 'root');
```

Editando o Banco de Dados

- Repetir o procedimento para a tabela de produtos e verificar se a adição ocorreu sem problemas.

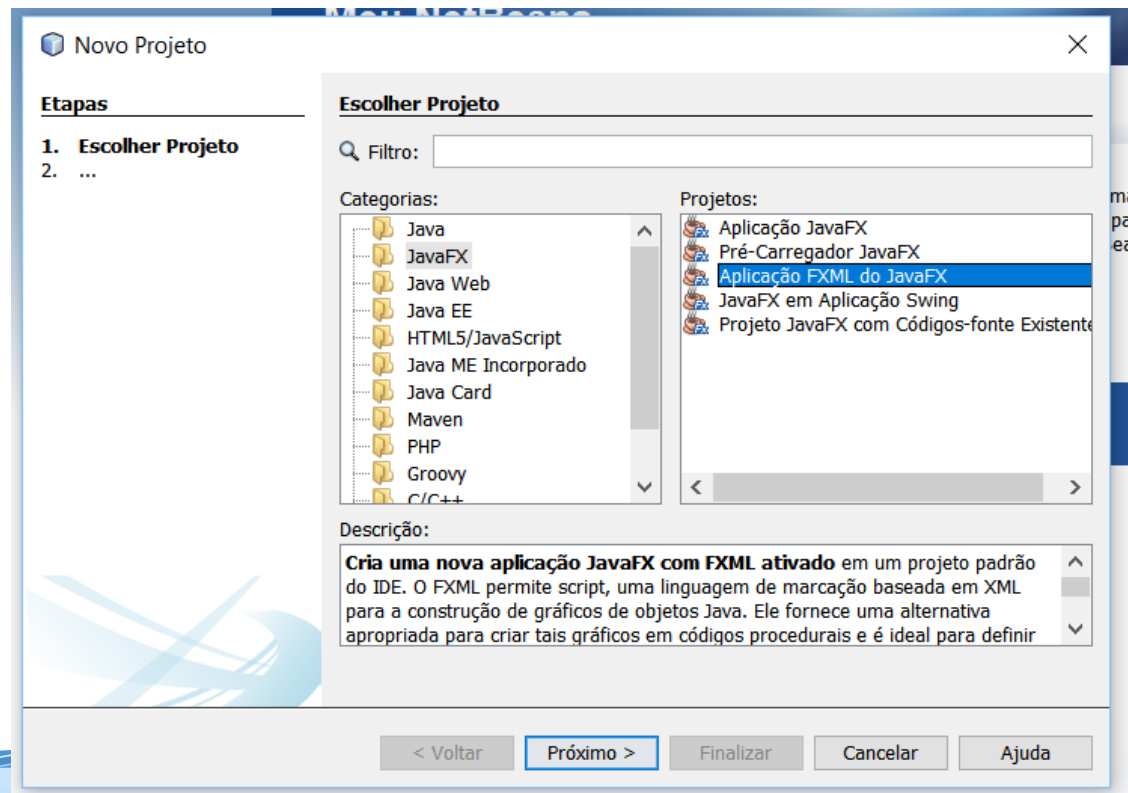
```
1 • INSERT INTO produtos VALUES (0, 'Semente dos Deuses', 10, 199.90);  
2 • INSERT INTO produtos VALUES (1, 'Esfera do Dragão 1 Estrela', 1, 2500);  
3 • INSERT INTO produtos VALUES (2, 'Esfera do Dragão 2 Estrela', 1, 2500);  
4 • INSERT INTO produtos VALUES (3, 'Esfera do Dragão 3 Estrela', 1, 2500);  
5 • INSERT INTO produtos VALUES (4, 'Esfera do Dragão 4 Estrela', 1, 2500);
```


Editando o Banco de Dados


- Qual foi a diferença do processo de adição dos elementos na tabela usuários e na tabela produtos?
- Por que esse procedimento foi diferente?

Interface para o usuário

- Criar um novo projeto no Netbeans, com uma interface gráfica para o usuário, logo um projeto JavaFX



Interface para o usuário - Desafio

- Nossa aplicação será composta de 2 telas, uma de login e a segunda de uso do sistema.
 - Na tela de uso do sistema, o usuário precisa ser capaz de cadastrar novos produtos, realizar a retirada de produtos e verificar o status do sistema.
 - Para a tela de uso do sistema, utilizar um TabView.
- 

Referências

- DevMedia (Thiago Vinícius)
- Oracle
- Caelum