MAC0439 – Laboratório de Bancos de Dados

Aula 20 – Parte 1

Programação para Bancos de Dados

JDBC

9 de novembro de 2016

Profa. Kelly Rosa Braghetto

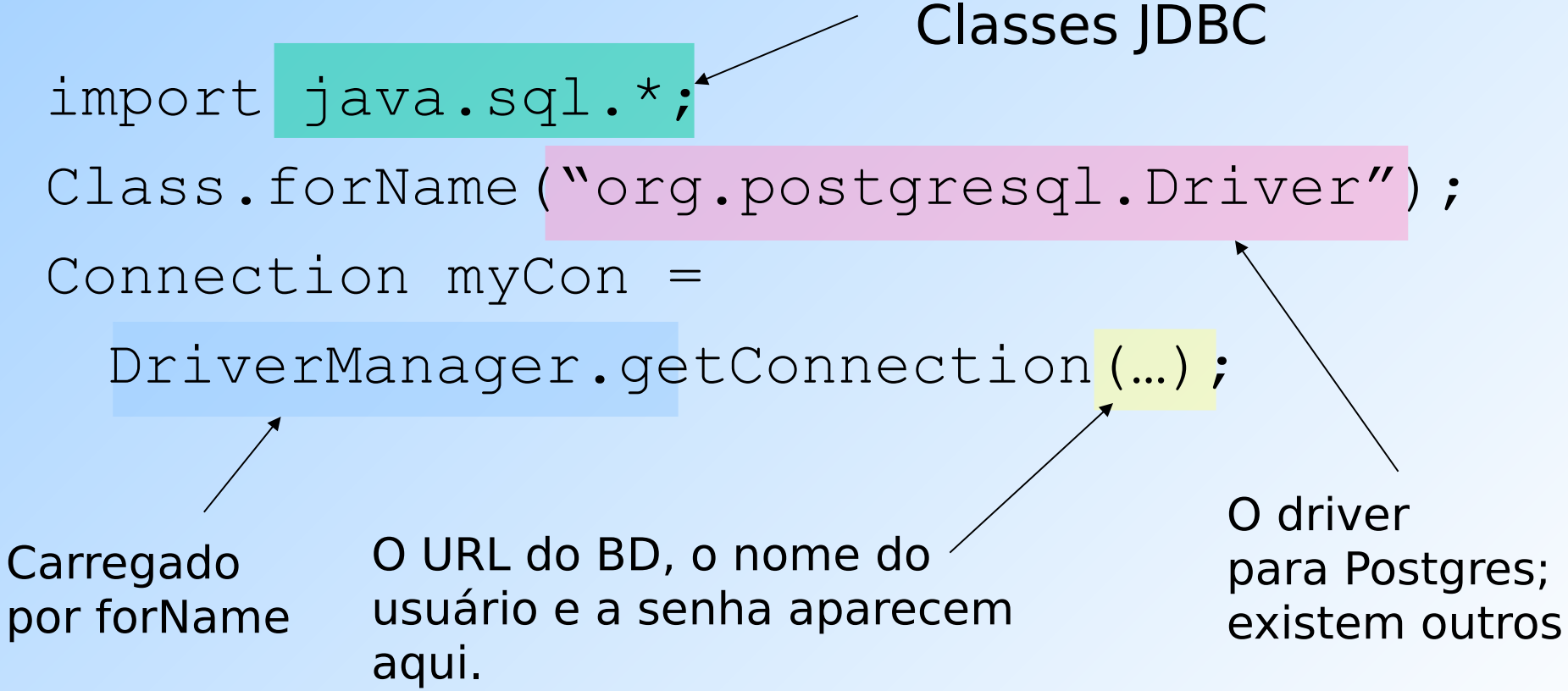
Interfaces entre Hospedeira/SQL via Bibliotecas

- Existem 2 abordagens que integram linguagens de programação e bancos de dados, mas que envolvem o uso de códigos que são específicos para um dado SGBD:
  - *Stored procedures*
  - SQL embutida
- Existe ainda uma 3ª abordagem, em que os recursos de bancos de dados são expostos ao programador de uma forma padronizada (independente de SGBD), por meio de uma biblioteca (API)

Exemplos de APIs de Integração a BDs

- C + SQL/CLI (*Call Level Interface*) – que é parte do padrão SQL
- Java + JDBC (Java Database Connectivity)
- PHP + PEAR MDB2

Criando uma Conexão em JDBC



Comandos

- JDBC disponibiliza duas classes de comandos:
  1. *Statement* = um objeto que pode aceitar uma *string* que é um comando SQL e executá-lo.
  2. *PreparedStatement* = um objeto que possui um comando SQL associado a ele, pronto para ser executado.

Criação de Comandos

- A classe de conexão possui métodos para a criação de *Statements* e *PreparedStatements*.

```
Statement stat1 = myCon.createStatement();
PreparedStatement stat2 =
    myCon.prepareStatement(
        ”SELECT nome_refri, preco FROM Vendas ” +
        ”WHERE nome_lanch = ? ”
    );
```

Parâmetro da consulta, a ser definido na chamada ao *execute*

Execução de Comandos SQL

- O JDBC diferencia consultas e modificações, que ele chama de "updates."
- Tanto *Statement* quanto *PreparedStatement* possuem os métodos executeQuery e executeUpdate.
  - Para *Statements*: um só argumento – a consulta ou modificação a ser realizada.
  - Para *PreparedStatements*: número de argumentos reflete os argumentos ("?") existentes na *string* usada em sua criação.

Exemplo: Modificação

- stat1 é um *Statement*.
- Podemos usá-lo para inserir uma tupla, como em:

```
stat1.executeUpdate(
  ”INSERT INTO Vendas ” +
  ”VALUES(’Sujinhos’,’Dolli’,3.00)”
);
```

Exemplo: Consulta

- stat2 é um *PreparedStatement* com a consulta ”SELECT nome_refri, preco FROM Vendas WHERE nome_lanch = ?”.
- **executeQuery** devolve um objeto da classe **ResultSet** (sobre a qual veremos mais detalhes depois)
- A consulta:

```
stat2.setString(1, “Sujinhos”);
ResultSet menu = stat2.executeQuery();
```

O Uso do ResultSet

- Um objeto do tipo **ResultSet** é algo parecido com um cursor.
- O método next() avança o “cursor” para a próxima tupla.
  - Na primeira vez em que o next() é aplicado, ele obtém a primeira tupla.
  - Se não há mais tuplas, next() devolve o valor false.

Acessando os Atributos de uma Tupla

- Quando um ResultSet se refere a uma tupla, podemos obter os atributos dessa tupla executando determinados métodos do ResultSet.
- O método get*X*(*i*), where *X* é um tipo e *i* é o número do atributo, devolve o valor do atributo.
  - O valor precisa ser do tipo *X*.

Exemplo: Acessando Atributos

- menu = ResultSet da consulta “SELECT nome_refri, preco FROM Vendas WHERE nome_lanch = ’Sujinhos’ ”.
- Para obter o refri e o preço em cada tupla:

```
while ( menu.next() ) {
  oRefri = menu.getString(1);
  oPreco = menu.getFloat(2);
  /* algum processamento envolvendo oRefri
     e oPreco */
}
```

Um parênteses: Injeção de SQL

- Consultas SQL frequentemente são construídas por programas.
- Essas consultas podem envolver dados fornecidos como entrada por usuários.
- Um código “descuidado” pode permitir que consultas maliciosas sejam construídas e executadas.

Exemplo: Injeção de SQL

- Relação Cartões(nome, senha, num_cartão).
- Interface Web: obtém nome e senha do usuário, armazena-as nas strings *n* and *s*, constrói a consulta, executa-a e mostra o número da conta.

```
consulta = "SELECT num_cartão FROM
  Cartões WHERE login = '" + n +
 "' AND senha = " + s
```

Usuário (que não é o Bill Gates) Digita:

Comentário em PostgreSQL

Login: `gates' --`

Senha: `qualquer uma`

O número do cartão é: 123.456.789.012

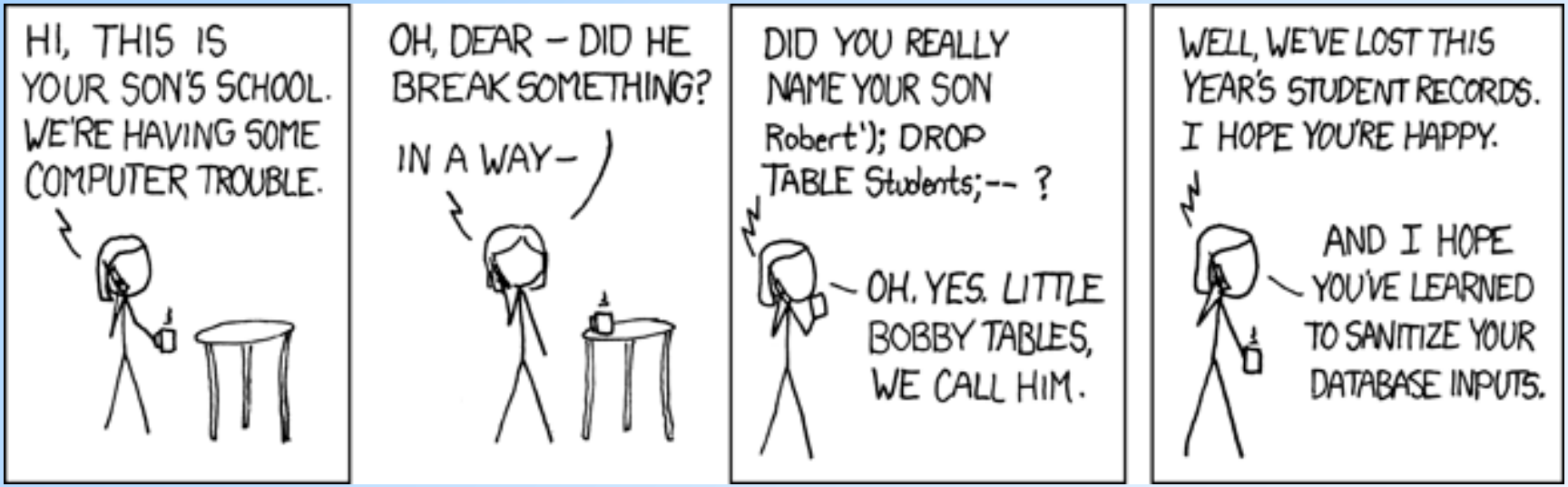
A Consulta Executada

```
SELECT num_cartão FROM Cartões
WHERE login = 'gates'  --' AND
senha = 'qualquer uma'
```

Tudo tratado como comentário!

Exemplo 2: Injeção de SQL

Exemplo 3: Injeção de SQL



Fonte: xkcd (http://xkcd.com/327/)

Bobby Tables: A guide to preventing SQL injection

http://bobby-tables.com/

Injeção de SQL

- O uso do *preparedStatement* da JDBC evita problemas desse tipo
- Num *preparedStatement*, os valores do tipo string atribuídos a parâmetros da consulta são “tratados” antes de serem passados ao BD
- Veja um exemplo detalhado sobre isso em Java + JDBC em

  http://www.journaldev.com/2489/jdbc-statement-vs-preparedstatement-sql-injection-example

Referências Bibliográficas

- Slides Prof. Jeffrey Ullman, da Stanford University
- Referência “oficial”

  http://docs.oracle.com/javase/tutorial/jdbc/
- Capítulo 6 do livro “Database Management Systems” (3rd edition), Ramakrishnan e Gehrke
- Capítulo 13 do livro “Sistemas de Banco de Dados” (6ª edição), Elmasri e Navathe
- SQL Call Level Interface (CLI)

  http://pubs.opengroup.org/onlinepubs/009654899/toc.pdf