UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Introdução à conectividade Java com Banco de Dados usando JDBC

Programação Desktop

Prof. Fabrício M. Lopes
fabricio@utfpr.edu.br

Organização das informações do banco de dados

- Um banco de dados relacional armazena informações em tabelas.
- Logo, cada linha representa um registro e cada coluna representa um campo desse registro. A figura abaixo exibe uma tabela típica.

Product

Product_Code	Description	Price
116-064	Toaster	24.95
257-535	Hair dryer	29.95
643-119	Car vacuum	19.99

Tipos de dados SQL x Java

Table 1 Some Standard SQL Types and Their Corresponding Java Types

SQL Data Type	Java Data Type
`INTEGER or INT`	`int`
`REAL`	`float`
`DOUBLE`	`double`
`DECIMAL(`*m*`, `*n*`)`	Fixed-point decimal numbers with *m* total digits and *n* digits after the decimal point; similar to `BigDecimal`
`BOOLEAN`	`boolean`
`VARCHAR(`*n*`)`	Variable-length `String` of length up to *n*
`CHARACTER(`*n*`) or CHAR(`*n*`)`	Fixed-length `String` of length *n*

SQL

- A maioria dos bancos de dados relacionais segue o padrão SQL.
- Por exemplo, aqui está o comando SQL para criar uma tabela chamada Produto:

```
CREATE TABLE Product(
    Product_Code CHAR(7),
    Description VARCHAR(40),
    Price FLOAT
)
```

SQL

- Ao contrário do Java, o SQL não é sensível a maiúsculas e minúsculas.
  - No entanto é utilizado maiúsculo para as palavras-chave SQL e letras maiúsculas mistas para nomes de tabelas e colunas.
- Para inserir linhas na tabela, use o comando INSERIR INTO.
  - Exemplo:

```
INSERT INTO product
   VALUES ('257-535', 'Hair dryer', 29.95)
```

SQL

- Para Selecionar linhas na tabela, use o comando SELECT.
  - Exemplo:

```
SELECT * from product
```

SQL

- Para Atualizar linhas na tabela, use o comando UPDATE.
  - Exemplo:

```
UPDATE Product
   SET description = 'Shampoo'
   WHERE product_code = '252-531'
```

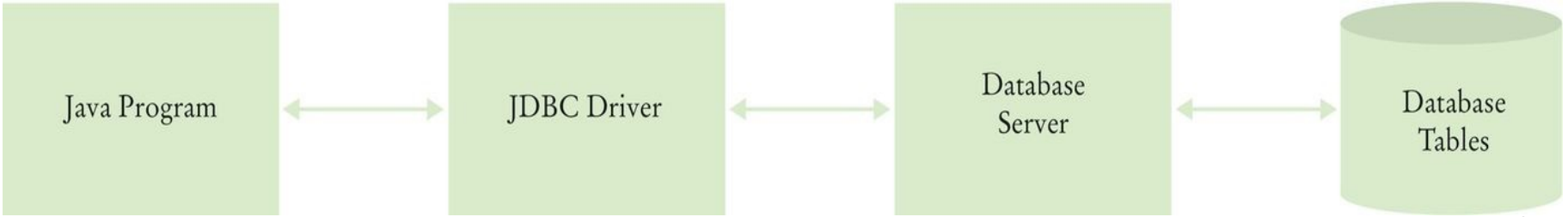
SQL

- Para excluir linhas na tabela, use o comando DELETE.
  - Exemplo:

```
DELETE FROM product
   WHERE product_code = '111-111'
```

JDBC - Java Database Connectivity

- java.sql.*;
- Provê acesso a banco de dados a partir de um programa Java, utilizando drivers.
- Bancos de dados diferentes requerem drivers diferentes.
- Quando seu programa Java envia comandos SQL, o driver os encaminha para o banco de dados e permite que seu programa receba os resultados.



Programação com banco de dados em Java

- O objeto **Connection** provê acesso ao banco de dados a partir da programação Java.
- Primeiro, o driver precisa ser carregado.
- A partir do Java 6 o carregamento do driver é automático, caso contrário é necessário utilizar:
  - ***Class.forName(driver)***; // carrega driver

Programação com banco de dados em Java

- A classe **DriverManager** implementa o método getConnection(url, username, password) para a criação do objeto de conexão. Exemplo:
  - String url = ***jdbc:postgresql://localhost:5432/postgres***;
  - String username = "***postgres***";
  - String password = "***postgres***";
  - ***Connection conn = DriverManager.getConnection(url, username, password)***;
  - ***connection.setAutoCommit(false);***

Programação com banco de dados em Java

- Uma vez que a conexão está definida, o objeto connection é usado para criar objetos **Statement**.
- Os objetos **Statement** são necessários para executar comandos SQL. Exemplo de criação:
  - **Statement stat = conn.createStatement();**
- Objetos **Statement** criados usando esse construtor serão, por padrão, do tipo **TYPE_FORWARD_ONLY** e terão um nível de concorrência **CONCUR_READ_ONLY**.

Programação com banco de dados em Java

- Objeto **Statement** podem ser criados usando a especificação do tipo e concorrência:
  - **Statement stat = conn.createStatement(int tipo, int concorrencia)**
- Parâmetros possíveis para:
- **Tipo**
  - **ResultSet.TYPE_FORWARD_ONLY, ResultSet.TYPE_SCROLLL_INSENSITIVE, ou ResultSet.TYPE_SCROLLL_SENSITIVE**
- **Concorrência**
  - **ResultSet.CONCUR_READ_ONLY ou ResultSet.CONCUR_UPDATABLE**

Programação com banco de dados em Java

- Um objeto **Statement**[1] possui três métodos principais:
  - **boolean execute(String sql)**
  - **int executeUpdate(String sql)**
  - **ResultSet executeQuery(String sql)**

[1] https://docs.oracle.com/javase/7/docs/api/java/sql/Statement.html

Programação com banco de dados em Java

- O método **execute(String sql)** é indicado para executar uma instrução SQL “Data Definition Language” (DDL), que pode ser uma instrução CREATE, ALTER, ou DROP.
- O retorno do método é um booleano que indica se a execução da instrução SQL definida no parâmetro do método foi executada com sucesso.

Programação com banco de dados em Java

- O método **executeUpdate(String sql)** executa uma instrução SQL “Data Manipulation Language” (DML) , que pode ser uma instrução INSERT, UPDATE, ou DELETE.
- O retorno do método é um número inteiro que indica a quantidade de linhas (registros) alterados no banco de dados pela execução da instrução SQL definida no parâmetro do método.

Programação com banco de dados em Java

- O método **executeQuery(String sql)** executa uma instrução SQL “Data Query Language” (DQL), que pode ser uma instrução SELECT.
- O retorno do método é um objeto **ResultSet.**

Programação com banco de dados em Java

- O objeto **ResultSet** consiste de uma tabela de dados representando um conjunto de resultados de uma declaração que consulta o banco de dados.
- Um objeto **ResultSet** mantém um cursor apontando para sua linha atual de dados. Inicialmente, o cursor é posicionado antes da primeira linha.
- O objeto possui métodos para a navegação entre os dados:
  - first(), next(), previous() e last().

Programação com banco de dados em Java

- Para recuperar os dados de um objeto ResultSet, existem vários métodos que retornam o valor da coluna de acordo com o seu valor: número, string, data, etc.
- Logo, para cada tipo de dado, há dois métodos que podem ser usados. O primeiro recebe um parãmetro inteiro que indica a posição da coluna. O segundo recebe um parãmetro String que indica o nome da coluna. Por exemplo, uma campo da tabela chamado “product_code” que ocupa a primeira coluna poderia ser acessado como:
  - String productCode = resultset.getString(1);
  - ou
  - String productCode = resultset.getString("product_code");

Programação com banco de dados em Java

- Logo, se for realizada uma consulta SELECT *, é uma boa idéia usar um nome de coluna ao invés de um índice de coluna, oq eu torna o código mais fácil de ler, e não é necessário atualizar o código quando o layout da coluna muda.
- Para obter um número inteiro use o método getInt. Para obter um número real use o método getDouble e assim por diante, exemplo:
  - int quantity = resultset.getInt("quantity");
  - double price = resultset.getDouble("price");

Tratamento de Exceções

- **SQLException**
  - Exceção padrão para possíveis erros de acesso ao banco de dados.
- **ClassNotFoundException**
  - Tratamento de possível erro na identificação e carregamento do driver de acesso ao banco de dados.

Apresentação de Exemplos

- Classe AcessaBD.java

Referências

- DEITEL, P.J. Java - Como Programar. Porto Alegre: Bookman, 2001.
- NIEMEYER, Patrick. Aprendendo java 2 SDK. Rio de Janeiro: Campus, 2000.
- MORGAN, Michael. Java 2 para Programadores Profissionais. Rio de Janeiro: Ciência Moderna, 2000.
- HORSTMANN, Cay, S. e CORNELL, Gary. Core Java 2. São Paulo: Makron Books, 2001 v.1. e v.2.