

---

# Banco de Dados

---

---

# Sumário

- Introdução;
- Banco de Dados Relacionais;
- SQL;
- MySQL;
- Manipulando Banco de Dados com o JDBC;
- Bibliografia;

# Introdução

- **Banco de Dados** é uma coleção organizada de dados;
- Um sistema de gerenciamento de banco de dados - **SGBD** fornece mecanismos para armazenar, organizar e recuperar dados para muitos usuários;
- Os sistemas mais populares de hoje são os **banco de dados relacionais**;
  - A linguagem SQL é a linguagem padrão internacional utilizada com banco de dados relacionais para realizar consultas e manipular dados;

---

# Introdução

- Alguns sistemas de gerenciamento de banco de dados relacionais populares são:
  - Microsoft SQL Server
    - (<http://www.microsoft.com/brasil/sql/>)
  - Oracle
    - (<http://www.oracle.com/global/br/index.html>)
  - IBM DB2
    - (<http://www.ibm.com/db2>)
  - PostgreSQL
    - (<http://www.postgresql.org.br/>)
  - MySQL
    - (<http://www.mysql.com/>)
- Os exemplos desta aula serão apresentados **utilizando o MySQL;**

---

# Banco de Dados Relacionais

---

# SQL

- Vejamos algumas palavras-chave de consulta de SQL:
  - `SELECT authorID, lastName FROM authors;`
  - `SELECT title FROM titles WHERE copyright > 2002`
  - `SELECT authorID, firstName, lastName FROM authors ORDER BY lastName ASC`
  - `SELECT authorID, firstName, lastName FROM authors ORDER BY lastName DESC`
  - `SELECT authorID, firstName, lastName FROM authors ORDER BY lastName, firstName`
  - `SELECT isbn, title, price FROM titles WHERE title LIKE '%How to Program' ORDER BY title ASC`

# SQL

- Vejamos algumas palavras-chave de inserção e atualização e remoção de SQL:
  - `INSERT INTO authors (firstName, lastName ) VALUES ('Sue', 'Smith')`
  - `UPDATE authors SET lastName = 'Jones' WHERE lastName = 'Smith' AND firstName = 'Sue'`
  - `DELETE FROM authors WHERE lastName = 'Jones' AND firstName = 'Sue'`
  - `DROP DATABASE books;`

# SQL

- Mesclar dados a partir de múltiplas tabelas: INNER JOIN
  - Os projetistas de BD costumam dividir os dados relacionados em tabelas separadas para assegurar que um banco de dados não armazene dados de maneira redundante;
  - Com o INNER JOIN, ou junção interna, de tabelas é possível mesclar linhas de duas tabelas correspondendo valores em colunas que são comuns às tabelas;
  - `SELECT firstName, lastName, isbn FROM authors INNER JOIN authorISBN ON authors.authorID = authorISBN.authorID ORDER BY lastName, firstName`



---

# SQL

- A consulta anterior mescla dados das colunas `firstName` e `lastName` da tabela `authors` com a coluna `isbn` da tabela `authorISBN`, classificando o resultado em ordem crescente por `lastName` e `firstName`;
- É utilizado a sintaxe `nomeDaTabela.nomeDaColuna` na cláusula `ON` para especificar as colunas de cada tabela que devem ser comparadas para unir as tabelas;

---

# MySQL

# Manipulando Banco de Dados com o JDBC

- Os programas Java comunicam-se com banco de dados e manipulam seus dados utilizando a **API do JDBC™**;
- Um **driver JDBC** permite aos aplicativos Java conectar-se a um banco de dados em um SGBD particular e permite aos programadores manipular esse BD utilizando a API do JDBC;
- Para obter informações adicionais sobre o JDBC, visite:
  - [java.sun.com/products/jdbc](http://java.sun.com/products/jdbc)

# Manipulando Banco de Dados com o JDBC

- Como nossos exemplos usarão o MySQL precisamos usar o **driver** (mysql-connector-java-3.0.14-production-bin.jar);
  - [http://www.univasf.edu.br/~leonardo.campos/Arquivos/Disciplinas/POO\\_2007\\_2/mysql-connector-java-3.1.12-bin.jar](http://www.univasf.edu.br/~leonardo.campos/Arquivos/Disciplinas/POO_2007_2/mysql-connector-java-3.1.12-bin.jar)
- O **programa deve carregar** o driver de banco de dados antes de conectar-se ao banco de dados, para tanto, devemos:
  - **Incluir o driver no diretório** jdk1.5.0\_09\jre\lib\ext
  - Incluir o driver no caminho de classe do programa ao executar o programa ou

# Exemplo 01

- **Link:** [http://www.univasf.edu.br/~leonardo.campos/Arquivos/Disciplinas/POO\\_2007\\_2/Codigos\\_Java/BD/Exemplo01/](http://www.univasf.edu.br/~leonardo.campos/Arquivos/Disciplinas/POO_2007_2/Codigos_Java/BD/Exemplo01/)

```
1 import java.sql.Connection;
2 import java.sql.Statement;
3 import java.sql.DriverManager;
4 import java.sql.ResultSet;
5 import java.sql.ResultSetMetaData;
6 import java.sql.SQLException;
7
8 public class DisplayAuthors
9 {
10     // nome do driver JDBC e URL do banco de dados
11     static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
12     static final String DATABASE_URL = "jdbc:mysql://localhost/books";
13
14     // carrega o aplicativo
15     public static void main( String args[] )
16     {
17         Connection connection = null; // gerencia a conexão
18         Statement statement = null; // instrução de consulta
19
20         // conecta-se ao banco de dados books e o consulta
```

# Exemplo 01

```
21     try
22     {
23         Class.forName( JDBC_DRIVER ); // carrega classe de driver do banco de dados
24
25         // estabelece conexão com o banco de dados
26         connection =
27             DriverManager.getConnection( DATABASE_URL, "root", "admin" );
28
29         // cria Statement para consultar banco de dados
30         statement = connection.createStatement();
31
32         // consulta o banco de dados
33         ResultSet resultSet = statement.executeQuery(
34             "SELECT authorID, firstName, lastName FROM authors" );
35
36         // processa resultados da consulta
37         ResultSetMetaData metaData = resultSet.getMetaData();
38         int numberOfColumns = metaData.getColumnCount();
39         System.out.println( "Authors Table of Books Database:" );
40     }
```

# Exemplo 01

```
41     for ( int i = 1; i <= numberOfColumns; i++ )
42         System.out.printf( "%-8s\t", metaData.getColumnName( i ) );
43     System.out.println();
44
45     while (resultSet.next())
46     {
47         for ( int i = 1; i <= numberOfColumns; i++ )
48             System.out.printf( "%-8s\t", resultSet.getObject( i ) );
49         System.out.println();
50     } // fim do while
51 } // fim do try
52 catch (SQLException sqlException)
53 {
54     sqlException.printStackTrace();
55     System.exit( 1 );
56 } // fim do catch
57 catch (ClassNotFoundException classNotFound)
58 {
59     classNotFound.printStackTrace();
60     System.exit( 1 );
61 } // fim do catch
```

# Exemplo 01

```
62     finally // assegura que a instrução e conexão são fechadas adequadamente
63     {
64         try
65         {
66             statement.close();
67             connection.close();
68         } // fim do try
69         catch ( Exception exception )
70         {
71             exception.printStackTrace();
72             System.exit( 1 );
73         } // fim do catch
74     } // fim do finally
75 } // fim de main
76 } // fim da classe DisplayAuthors
77
```



---

# Exemplo 01

- Considerações:
  - O método `static forName` da classe `Class` carrega a classe para o driver de banco de dados;
  - O objeto `connection` implementa a interface `Connection` para gerenciar a conexão entre o programa Java e o banco de dados;
  - Os objetos `connection` permitem aos programas criar instruções SQL que acessem banco de dados;
  - O programa inicializa `connection` com o resultado de uma chamada para o método `static getConnection` da classe `DriverManager`, que tenta conectar-se ao banco de dados especificado por sua URL

# Exemplo 01

## ■ Considerações:

- ❑ O método `getConnection` aceita três argumentos - uma `string` que especifica a URL de banco de dados, uma `string` que especifica o nome do usuário e uma `string` que especifica a senha;
- ❑ O método `createStatement` obtém um objeto que implementa a interface `statement`. O programa utiliza o objeto `Statement` para submeter SQL para o banco de dados;
- ❑ O método `executeQuery` do objeto `statement` submete uma consulta que seleciona todas as informações de autor da tabela `authors`;

# Exemplo 01

## ■ Considerações:

- ❑ O resultado das consultas retorna um objeto que implementa a interface `ResultSet`. São os métodos `ResultSet` que permitem manipular o resultado da consulta;
- ❑ Para obter informações sobre nomes de coluna e tipos de `ResultSet` utilizamos `metadados`; São eles que descrevem o conteúdo do `ResultSet`;
- ❑ O laço `for` utiliza o método `getColumnCount` para recuperar o número de colunas no `ResultSet` para serem exibidas em seguida;

---

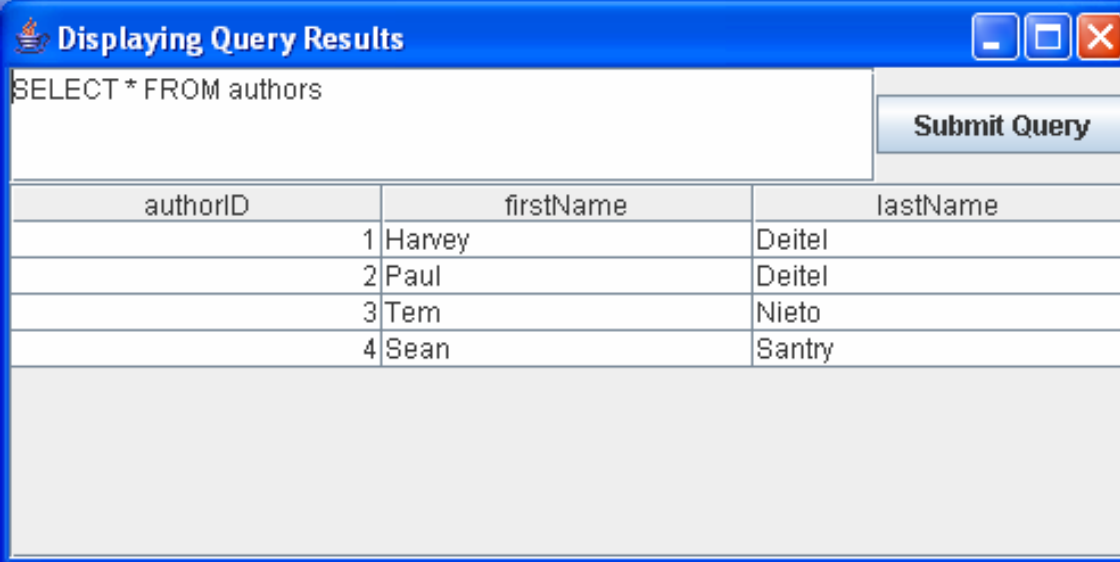
# Exemplo 01

- **Considerações:**

- Antes de processar o `ResultSet`, o programa posiciona o cursor do `ResultSet` na primeira linha do `ResultSet` com o método `next`;
- O método `next` retorna um valor boolean `true` se ele for capaz de se posicionar na próxima linha; caso contrário, o método retorna `false`;
- O bloco `finally` fecha `statement` e o banco de dados `Connection`;

# Exemplo 2

- **Link:** [http://www.univasf.edu.br/~leonardo.campos/Arquivos/Disciplinas/POO\\_2007\\_2/Codigos\\_Java/BD/Exemplo02/](http://www.univasf.edu.br/~leonardo.campos/Arquivos/Disciplinas/POO_2007_2/Codigos_Java/BD/Exemplo02/)



The screenshot shows a web application window titled "Displaying Query Results". The window contains a text input field with the SQL query "SELECT \* FROM authors". To the right of the input field is a "Submit Query" button. Below the input field is a table with three columns: "authorID", "firstName", and "lastName". The table contains four rows of data.

authorID	firstName	lastName
1	Harvey	Deitel
2	Paul	Deitel
3	Terri	Nieto
4	Sean	Santry

# Exemplo 02

- Considerações:
  - O exemplo permite ao usuário inserir qualquer consulta no programa e exibe o resultado de uma consulta em uma JTable, utilizando o objeto `TableModel` para fornecer os dados de `ResultSet` para a `JTable`;
  - A classe `ResultSetTableModel` estende a classe `AbstractTableModel`, que implementa a interface `TableModel`. Essa classe sobreescreve os métodos:
    - `getColumnClass`,
    - `getColumnCount`,
    - `getColumnName`,
    - `getRowCount` e
    - `getValueAt`

# Exemplo 02

## ■ Considerações:

- As implementações-padrão dos métodos `TableModel` `isCellEditable` e `setValueAt` não são sobrescritas, porque esse exemplo não suporta editar células da `JTable`;
- O método `getColumnns` retorna um objeto `Class` que representa a superclasse de todos os objetos em uma coluna particular;
  - É utilizado `column + 1` pois o número de linhas e colunas da `JTable` são contados a partir de 0;
- O método `getColumnCount` retorna o número de colunas no `ResultSet` adjacente do modelo;

# Exemplo 02

## ■ Considerações:

- ❑ O método `getRowCount` retorna o número de linhas no `ResultSet` adjacente do modelo;
- ❑ O método `getValueAt` retorna o object em uma linha e coluna subjacente do modelo.
- ❑ O método `absolute` posiciona o cursos `ResultSet` em uma linha específica;
- ❑ O método `setQuery` executa a consulta que ele recebe como um argumento para obter um novo `ResultSet`;



---

# Bibliografia

- Deitel, H. M. & Deitel, P. J. *Java: como programar*, Editora Bookman. 6ª ed. São Paulo: 2005.