Lição 5

Conexão com Banco de Dados: SQL e JDBC

Objetivos

Ao final desta lição, o estudante será capaz de:

- Construir um objeto da classe Connection utilizando a classe *DriverManager* ou *DataSource*
- Construir um objeto da classe *Statement* usando o método createStatement() disponível na classe Connection
- Executar consultas SQL usando um objeto da classe *Statement* e recuperar os resultados
- Encerrar os objetos do banco de dados

Bancos de Dados Relacionais

- Mídia de armazenamento escolhida por muitas aplicações baseadas na WEB que requerem conteúdo dinâmico
- Sintaxe básica necessária para recuperar e manipular dados armazenados, de fácil aprendizado
- Possui suporte muito difundido pela indústria
- Armazena registros como conjuntos relacionados

Bancos de Dados Relacionais

userid	name	address	contactnum
14627895	Duke	California	0924562788
65248987	Kimi	Finland	8687243217

Bancos de Dados Relacionais: Tabelas

- Tabelas de bancos de dados são projetadas com restrições lógicas para preservar a consistência de seus dados
  - Assinalação de tipos de dados
  - Singularidade
  - Outros

Comandos SQL

- Operações em bancos de dados relacionais são realizadas por meio do uso de SQL
- Há diversos tipos de comandos de SQL
- Entre estes:
  - Recuperação de dados
  - Manipulação de dados

Recuperação de Dados

- Focado na leitura de dados a partir de uma ou mais tabelas no banco de dados
  - Pode ser aberta para recuperar TODOS os conjuntos de dados
  - Pode ser parametrizada com valores conhecidos
- Somente um comando SQL abrange este tipo: **SELECT**

Comando SELECT

- Usado para examinar o banco de dados

```
SELECT column(s) FROM tablename WHERE condition(s)
```

Cláusula FROM

- Define em uma declaração SELECT as tabelas a partir das quais os dados serão reunidos
  - Se o dados vem de uma só tabela
  - Se o dados vem de mais de uma tabela

União de Tabelas

- Tabelas podem ser separadas por meio de vírgulas
- Modo mais simples entretanto, o desempenho classifica os dados mais lentamente.
- Executa um produto cartesiano nas tabelas
- Exemplo:
    - Dadas duas tabelas, **users** e **userdownloads**, a união é executada por

```
... FROM users, userdownloads ...
```

União de Tabelas

- Usando uma das várias palavras-chave
  - JOIN
  - LEFT JOIN
  - RIGHT JOIN
  - INNER JOIN
- Exemplos:

```
FROM tabela1 JOIN tabela2
FROM tabela1 LEFT JOIN tabela2
FROM tabela1 RIGHT JOIN tabela2
FROM tabela1 INNER JOIN tabela2
```

Exemplo de UNIÃO

User

userid	name	address	contactnum
14627895	Duke	San Francisco	0924562788
65248987	Kimi	Finland	8687243217
84321874	Dante	San Jose	6365498428

UserDownloads

userid	downloaditem	downloaddate
14627895	Courseware Notes	Dec. 19, 2005
36542036	Exercises	Feb. 11, 2006
84321874	Slides	March 13, 2006

userid	name	address	contactnum	userid	downloaditem	downloaddate
14627895	Duke	San Francisco	0924562788	14627895	Courseware Notes	Dec. 19, 2005
14627895	Duke	San Francisco	0924562788	36542036	Exercises	Feb. 11, 2006
14627895	Duke	San Francisco	0924562788	84321874	Slides	March 13, 2006
65248987	Kimi	Finland	8687243217	14627895	Courseware Notes	Dec. 19, 2005
65248987	Kimi	Finland	8687243217	36542036	Exercises	Feb. 11, 2006
65248987	Kimi	Finland	8687243217	84321874	Slides	March 13, 2006
84321874	Dante	San Jose	6365498428	14627895	Courseware Notes	Dec. 19, 2005
84321874	Dante	San Jose	6365498428	36542036	Exercises	Feb. 11, 2006
84321874	Dante	San Jose	6365498428	84321874	Slides	March 13, 2006

Exemplo de UNIÃO

- LEFT JOIN, com a condição:

`User.userid = UserDownloads.userid`

User			
userid	**name**	**address**	**contactnum**
14627895	Duke	San Francisco	0924562788
65248987	Kimi	Finland	8687243217
84321874	Dante	San Jose	6365498428

UserDownloads		
userid	**downloaditem**	**downloaddate**
14627895	Courseware Notes	Dec. 19, 2005
36542036	Exercises	Feb. 11, 2006
84321874	Slides	March 13, 2006

userid	name	address	contactnum	userid	downloaditem	downloaddate
14627895	Duke	San Francisco	0924562788	14627895	Courseware Notes	Dec. 19, 2005
65248987	Kimi	Finland	8687243217			
84321874	Dante	San Jose	6365498428	84321874	Slides	March 13, 2006

Exemplo de UNIÃO

- RIGHT JOIN, com a condição:

```
User.userid = UserDownloads.userid
```

User			
userid	**name**	**address**	**contactnum**
14627895	Duke	San Francisco	0924562788
65248987	Kimi	Finland	8687243217
84321874	Dante	San Jose	6365498428

UserDownloads		
userid	**downloaditem**	**downloaddate**
14627895	Courseware Notes	Dec. 19, 2005
36542036	Exercises	Feb. 11, 2006
84321874	Slides	March 13, 2006

userid	name	address	contactnum	userid	downloaditem	downloaddate
14627895	Duke	San Francisco	0924562788	14627895	Courseware Notes	Dec. 19, 2005
				36542036	Exercises	Feb. 11, 2006
84321874	Dante	San Jose	6365498428	84321874	Slides	March 13, 2006

Exemplo de UNIÃO

- INNER JOIN, com a condição:

`User.userid = UserDownloads.userid`

User			
userid	**name**	**address**	**contactnum**
14627895	Duke	San Francisco	0924562788
65248987	Kimi	Finland	8687243217
84321874	Dante	San Jose	6365498428

UserDownloads		
userid	**downloaditem**	**downloaddate**
14627895	Courseware Notes	Dec. 19, 2005
36542036	Exercises	Feb. 11, 2006
84321874	Slides	March 13, 2006

userid	name	address	contactnum	userid	downloaditem	downloaddate
14627895	Duke	San Francisco	0924562788	14627895	Courseware Notes	Dec. 19, 2005
84321874	Dante	San Jose	6365498428	84321874	Slides	March 13, 2006

Exemplo de UNIÃO

- Na maioria dos casos, uma INNER JOIN rende os resultados mais relevantes para operações de união
- A casos onde as entradas de uma tabela deveriam aparecer não importando o quê, a utilização de LEFT JOIN ou RIGHT JOIN é mais apropriada
- Evite usar união delimitada por vírgulas
- É mais simples e conveniente de escrever

Cláusula WHERE

- Especifica uma condição que deve ser casada pelas entradas na tabela selecionada para que elas sejam parte do resultado
  - Operadores lógicos podem ser usados
    - = (igualdade)
    - <= (menor ou igual) , < (menor)
    - >= (maior ou igual), > (maior)
  - like : Executa uma comparação entre dois operandos
    - % - se aplica a uma *String*
    - _ - se aplica a um caractere

Exemplos de Declarações SELECT

- Recupera todos os dados disponíveis na tabela users:

```
SELECT * from users;
```

- Retorna o campo *address* na tabela users com o nome Smith

```
SELECT address from users where name ='Smith';
```

- Retorna todos os campos da tabela *users* com o nome iniciado por 'S'

```
SELECT * from users where name like 'S%';
```

- SQL não diferencia letras maiúsculas de minúsculas de comandos
- Considera valores na qual ela executa comparações

```
SELECT * from users where name ='sMith';
```

Manipulação de Dados

- Usado para modificar o estado de dados dentro de tabelas
- Comandos SQL podem ser:
  - Comando INSERT
  - Comando UPDATE
  - Comando DELETE

Comando INSERT

- Sintaxe:

```
INSERT INTO table-name VALUES(value1, value2, ...)
```

- Qualquer chamada INSERT deve seguir as regras de integridade definidas na tabela

Comando UPDATE

- Sintaxe:

```
UPDATE table-name set column-value(s) WHERE
condition(s)
```

- Quaisquer atualizações devem ser conforme as regras de integridade no banco de dados

Comando DELETE

- Sintaxe:

```
DELETE FROM table-name WHERE condition(s)
```

- Uma lista de condições, separadas por vírgulas, pode ser especificada
- Se nenhuma condição for dada, o comando elimina todos os registros na tabela especificada

JDBC

- Java Database Connectivity (Conectividade a Bancos de Dados Java)
- Biblioteca padrão que fornece acesso a bancos de dados por meio de Java
- Desenvolvedores podem acessar bancos de dados não importando quem é o distribuidor
  - Distribuidores fornecem a implementação para interfaces abstratas definidas na API
  - Fornece o mesmo conjunto de funcionalidades para o desenvolvedor

JDBC

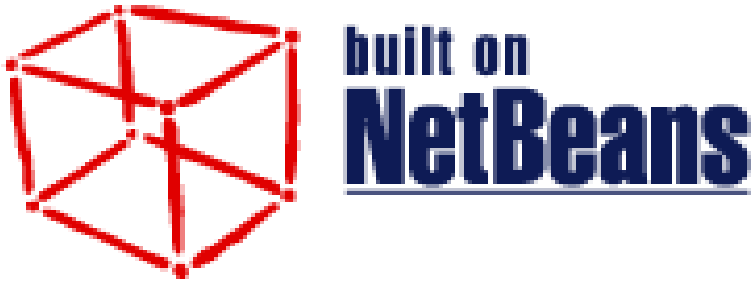
- **java.sql.Connection**: Representa uma conexão com um banco de dados
- **java.sql.DriverManager**: Gerencia drivers JDBC usados pela aplicação
- **javax.sql.DataSource**: Abrange os detalhes de como obter uma conexão para o banco de dados
- **java.sql.Statement**: Fornece métodos para o desenvolvedor executar instruções SQL
- **java.sql.ResultSet:** Representa o resultado de uma instrução SQL de Pesquisa

java.sql.DriverManager

- Permite que um desenvolvedor recupere um objeto *Connection* que pode ser usado para executar instruções em bancos de dados.
  - O *Driver* JDBC deve estar registrado com o DriverManager
  - Usar o método getConnection em DriverManager

java.sql.DriverManager

- Passaremos agora para o NetBeans

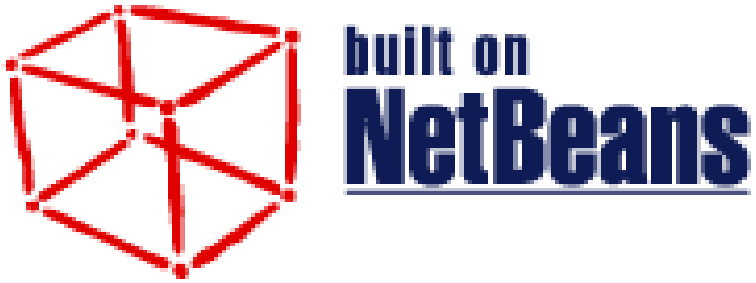


javax.sql.DataSource

- Uma interface definida na API JDBC desde a versão 2 de sua especificação
- E o modo recomendado para um desenvolvedor obter um objeto *Connection*
- Recuperar o objeto *Connection*
  - Chamar o método *getConnection*() em uma instância de *DataSource*
- Obter uma instância de *DataSource*

Configurando DataSource no Sun Application Server

- Passaremos agora para o NetBeans



Recuperando DataSource

- O contexto JNDI abstrai os detalhes de conexão com o diretório
- O nome usado para pesquisar o recurso deve ser o mesmo usado para configurar *DataSource*
- Uma vez que se tenha uma instância *DataSource* válida, obter uma conexão é:

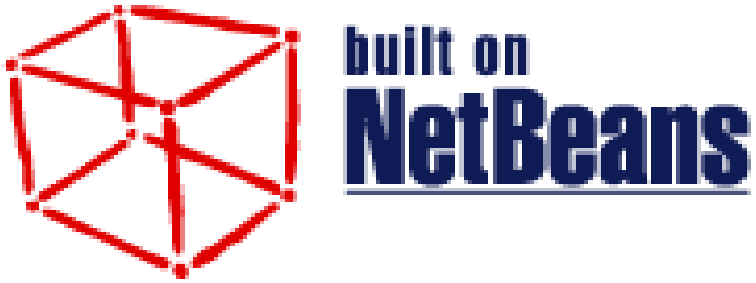
```
Connection conn = ds.getConnection();
```

java.sql.Connection / java.sql.Statement

- Objetos da classe *java.sql.Connection* representam conexões atuais para o banco de dados
- Uma vez de posse deste objeto, criamos um objeto da classe *Statement*, que usamos para executar consultas SQL
  - ***ExecuteQuery***: Tomada em comandos SELECT, retornando o resultado de operações como um objeto ResultSet
  - ***ExecuteUpdate***: Tomada em comandos INSERT, UPDATE ou DELETE, retornando o número de colunas afetadas como um tipo **int**

Executando uma Conexão

- Passaremos agora para o NetBeans

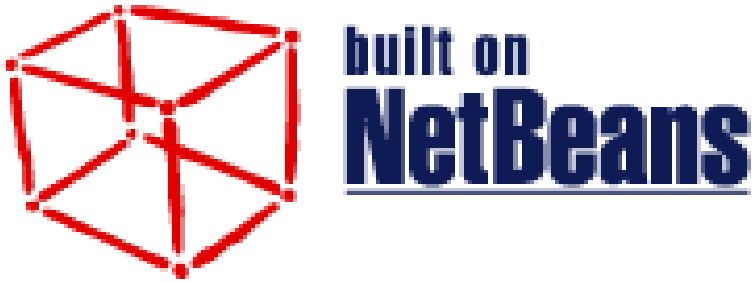


java.sql.ResultSet

- Resultados de uma consulta no banco de dados
- Um objeto *ResultSet* pode ser visualizado como uma tabela
- A informação é recuperada uma coluna por vez
- O objeto *ResultSet* mantém a coluna corrente
- Para percorrer as linhas da tabela em *ResultSet*, usamos o método *next*()

Executando uma Consulta

- Passaremos agora para o NetBeans



Liberando Recursos do Sistema

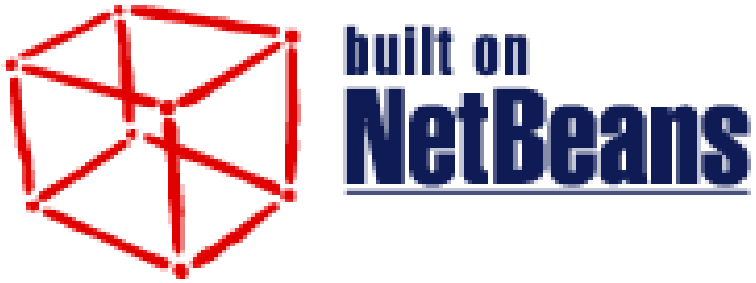
- Este é um passo muito importante que frequentemente é negligenciado após ter sido completada
- Deve ser feita explicitamente e é uma responsabilidade do programador
- Sem executar tal liberação, os recursos tomados pela operação não podem ser usadas no futuro
- Para aplicações muito grandes, isto rapidamente resulta na perda de conexões disponíveis

Liberando Recursos do Sistema

- Executada chamando o método *close*() disponíveis em cada objeto das classes *Connection*, *Statement*, e *ResultSet*
  - Existem uma ordem específica envolvida
  - O método *close* está definido para lançar uma *SQLException*
- Erros comuns dos desenvolvedores: colocar simplesmente os métodos dentro do corpo do programa
- Somente recorrer a condições de bem sucedidas
- O código deve ser colocado dentro de uma cláusula **finally**

Liberando Recursos do Sistema

- Passaremos agora para o NetBeans



Sumário

- Banco de Dados Relacionais
- Comandos SQL
- Recuperação de Dados
- União de Tabelas
- Manipulação de Dados
- Conexão com o Banco de Dados
- Liberando Recursos do Sistema

Parceiros

- **Os seguintes parceiros tornaram JEDI™ possível em Língua Portuguesa:**