Banco de Dados

Conectividade em BD:

Desenvolvendo aplicações de BD

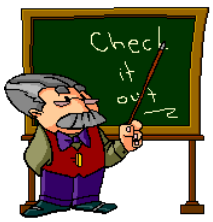
Prof. Rinaldo Lima
rinaldo.ufrpe@gmail.com

UFRPE Universidade Federal Rural de Pernambuco
DEINFO Departamento de Estatística e Informática

19-dez-17

Roteiro

- **Necessidade de acesso ao banco via aplicação;**
- **Abordagens para conexão e uso de BD em aplicações**
- **APIs para acesso ao banco:**
  - ODBC;
  - JDBC.



Conectar...

- **Até agora vínhamos executando comandos SQL diretamente no Banco de Dados:**
  - Usando SQL
- **Mas... Como conectar a aplicação à base de dados?**
  - Como garantir a conexão entre aplicações em diferentes linguagens e bases de dados em SGBDs distintos?

Padronização!

Opções de Conexão

- **Código SQL embutido:**
  - Sintaxe para executar código SQL direto da aplicação;
- **Acesso via API padronizadas:**
  - Em 1989 desenvolvedores de SGBDs criaram o SAG (SQL Access Group):
    - Desenvolver especificações de uma interface para uso comum no acesso aos dados gerenciados pelos diferentes SGBDs.
    - SQL CLI – *SQL Call Level Interface*
    - *JDBC (Java Database Conectivity)*
- **Linguagem nativas de BD PL/SQL (Oracle):**
  - *Evita o problema de "Impendância de Representação de Dados"*

Exemplo de Código SQL Embutido Em C

```
    //Program CLI1:
 0) #include sqlcli.h ;
 1) void printSal() {
 2) SQLHSTMT stmt1 ;
 3) SQLHDBC con1 ;
 4) SQLHENV env1 ;
 5) SQLRETURN ret1, ret2, ret3, ret4 ;
 6) ret1 = SQLAllocHandle(SQL_HANDLE_ENV, SQL_NULL_HANDLE, &env1) ;
 7) if (!ret1) ret2 = SQLAllocHandle(SQL_HANDLE_DBC, env1, &con1) else exit ;
 8) if (!ret2) ret3 = SQLConnect(con1, "dbs", SQL_NTS, "js", SQL_NTS, "xyz",
      SQL_NTS) else exit ;
 9) if (!ret3) ret4 = SQLAllocHandle(SQL_HANDLE_STMT, con1, &stmt1) else exit ;
10) SQLPrepare(stmt1, "select Lname, Salary from EMPLOYEE where Ssn = ?",
      SQL_NTS) ;
11) prompt("Enter a Social Security Number: ", ssn) ;
12) SQLBindParameter(stmt1, 1, SQL_CHAR, &ssn, 9, &fetchlen1) ;
13) ret1 = SQLExecute(stmt1) ;
14) if (!ret1) {
15)   SQLBindCol(stmt1, 1, SQL_CHAR, &lname, 15, &fetchlen1) ;
16)   SQLBindCol(stmt1, 2, SQL_FLOAT, &salary, 4, &fetchlen2) ;
17)   ret2 = SQLFetch(stmt1) ;
18)   if (!ret2) printf(ssn, lname, salary)
19)     else printf("Social Security Number does not exist: ", ssn) ;
20)   }
21) }
```

Figure 13.10
Program segment CLI1, a C program segment with SQL/CLI.

5

Conectividade via Interfaces Padrão

Definições e Conceitos Básicos

UFRPE Universidade Federal Rural de Pernambuco

DEINFO Departamento de Estatística e Informática

19-dez-17

Acessando SGBD via Interfaces

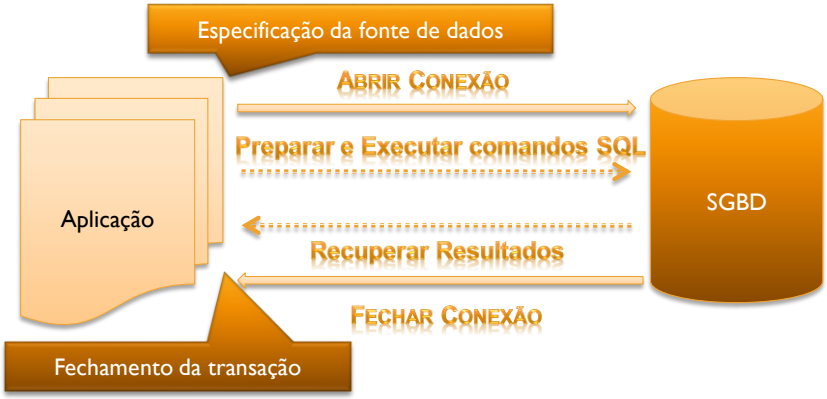
- **Objetivos da SQL CLI:**
  - Simplificar o acesso ao banco de dados emitindo comandos SQL diretamente pelos programas de aplicação;
  - Abolir o pré-processamento dos comandos embutidos;
  - Padronizar mensagens e protocolos:
    - Permitindo interoperabilidade entre diferentes linguagens e diferentes SGBDs

Driver específico para cada SGBD

7

Implementações da SQL - CLI

- **APIs comuns de acordo com SQL – CLI, permitem:**



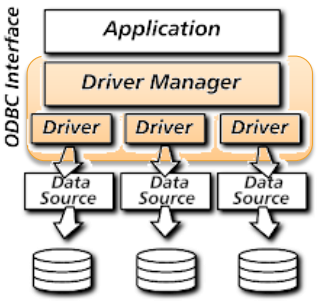
8

Implementações da SQL - CLI

- **Principais APIs:**
  - ODBC e JDBC;
- **Vantagens:**
  - Acesso padronizado a diversos SGBDs;
  - Abstração das particularidades dos SGBDs pelos desenvolvedores:
    - Desenvolvimento mais ágil.

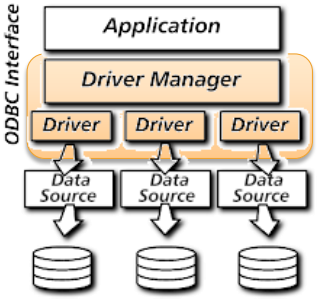
ODBC - Open Database Conectivity

- **Desenvolvido pela Microsoft;**
- **Arquitetura:**
  - ***Driver Manager*** – gerencia os drivers, carregando o driver específico para cada aplicação;
    - Aloca e desaloca recursos;
    - Cria e finaliza conexões com os BDs;
    - Passa as chamadas de função da aplicação para os drivers;
    - Gerencia os diversas conexões da aplicação com o SGBD.



ODBC - Open Database Conectivity

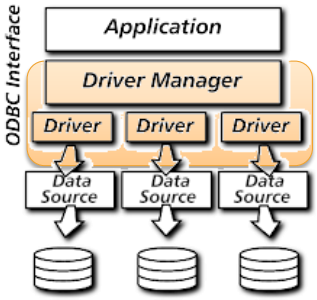
- **Arquitetura: (cont.)**
  - ***Driver*** – processa as chamadas de cada aplicação, repassando para o SGBD adequado e devolvendo o resultado para a aplicação.
    - Específico para cada SGBD:
      - Existem drivers para mais de um SGBD.
    - Gerencia a obtenção dos resultados de comandos SQL (como a manipulação de cursores);
    - Exemplos: Oracle, SQL Server, Jet (Acess), MySQL…



11

ODBC - Open Database Conectivity

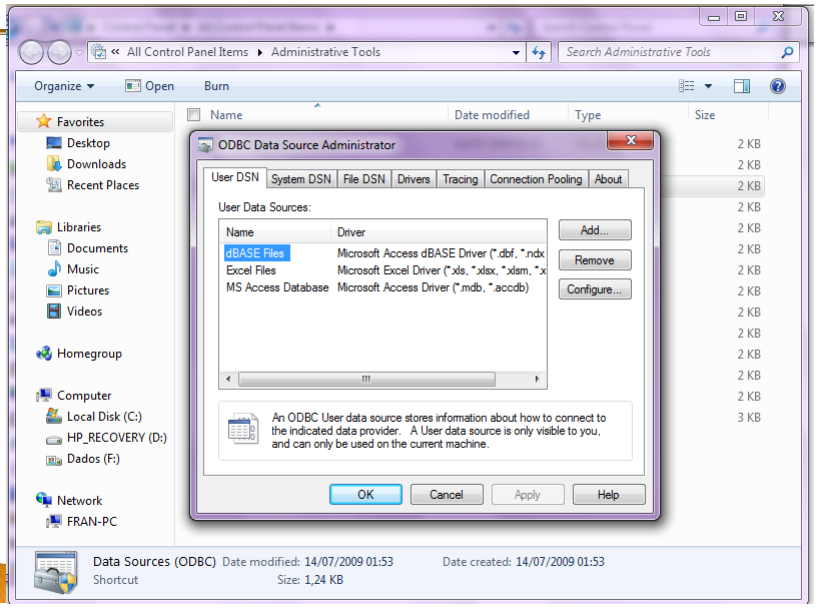
- **Arquitetura: (cont.)**
  - ***Data Source*** – SGBD-alvo.



- **Desvantagens:**
  - Acesso mais lento;
  - Cuidado com a segurança!
- **Configuração:**
  - Painel de controle >> ferramentas administrativas >> administração de fonte de dados ODBC.

12

ODBC - Open Database Conectivity



13

JDBC - Java Database Connectivity

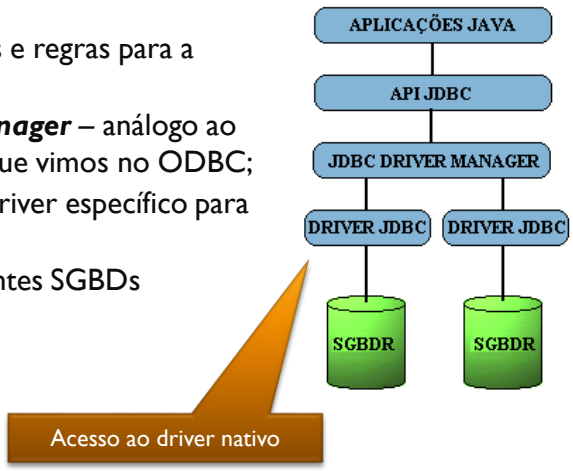
- **Biblioteca desenvolvida pela Sun Microsystems:**
  - Define um conjunto de interfaces de acesso ao BD:
    - Cada driver implementa uma destas interfaces.
- **A API padrão do Java já vem com o driver JDBC-ODBC:**
  - Utiliza o driver ODBC da máquina.
- **Vantagem:**

Java + JDBC = Multiplataforma

14

JDBC - Java Database Connectivity

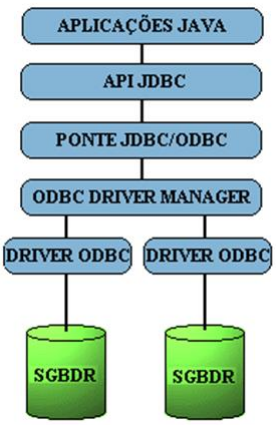
- **Arquitetura:**
  - ***API*** – Protocolos e regras para a Comunicação;
  - ***JDBC driver manager*** – análogo ao driver manager que vimos no ODBC;
  - ***Driver JDBC*** – driver específico para cada SGBD;
  - ***SGBDR*** – diferentes SGBDs relacionais.



15

JDBC – Tipos de Drivers

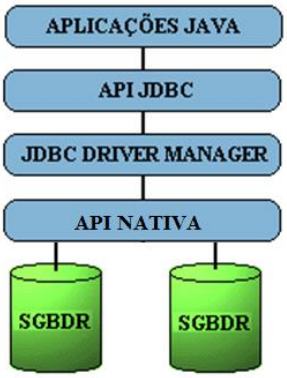
- **Driver Ponte JDBC-ODBC**
  - Usa a implementação nativa:
    - Traduz JDBC em ODBC.
  - Usa ODBC para conectar aplicação Java ao SGBD.



16

JDBC – Tipos de Drivers

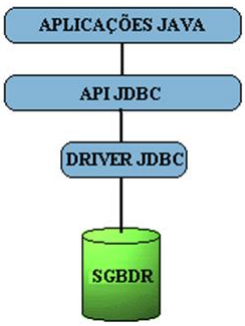
- **Driver API nativa parcialmente Java:**
  - Encapsula a API nativa do SGBD com chamadas JDBC;
  - Utiliza código extra sobre a plataforma, além do JDBC.



17

JDBC – Tipos de Drivers

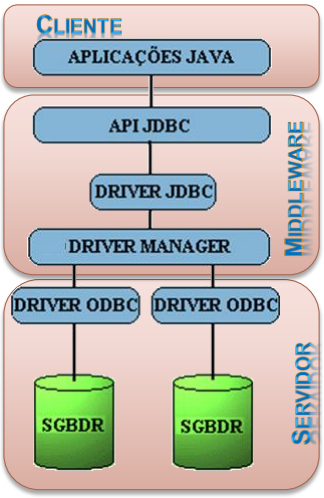
- **Driver Java Puro**
  - Driver totalmente implementado em Java;
  - Protocolo específico do SGBD;
  - Independente de plataforma.
  - ☺ **melhor opção!**



18

JDBC – Tipos de Drivers

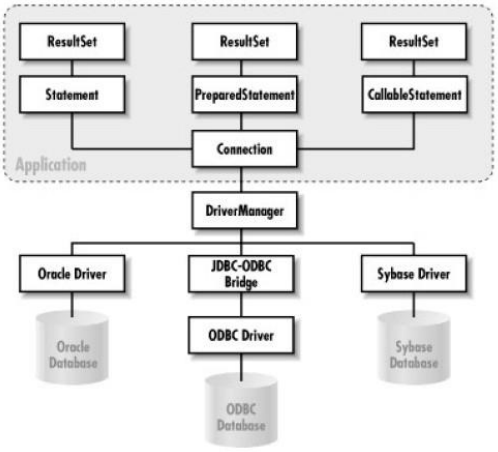
- **Driver *Middleware***
  - Driver totalmente implementado em Java;
  - Utiliza um *middleware* para a conexão com o banco de dados;
  - Independente de SGBD e de plataforma.



19

JDBC - API

- **Principais classes:**
  - DriverManager;
  - Connection;
  - Statement;
  - PreparedStatement;
  - ResultSet.

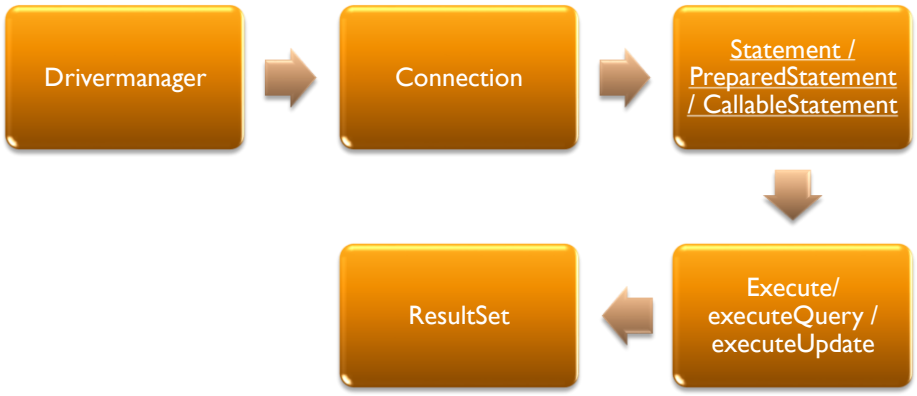


- http://java.sun.com/javase/6/docs/api/java/sql/package-summary.html
- http://docs.oracle.com/javase/tutorial/jdbc/

20

Utilizando API

Detalhes da API em:
http://docs.oracle.com/javase/tutorial/jdbc/basics/processingsqlstatements.html



Configurando Eclipse EE

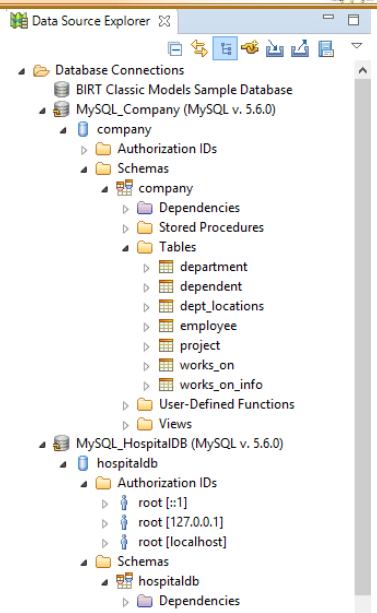
Versão Java EE do Eclipse para aplicações de BD

UFRPE Universidade Federal Rural de Pernambuco

DEINFO Departamento de Estatística e Informática

Integrando Eclipse com Banco de Dados

- **Eclipse Data Tools Platform**
  - http://www.eclipse.org/datatools/
  - Já incluído na versão Java EE (Enterprise Edition)
  - **Eclipse Mars Edition**
- **Integração completa com a base de dados**
- **SQL Query builder**

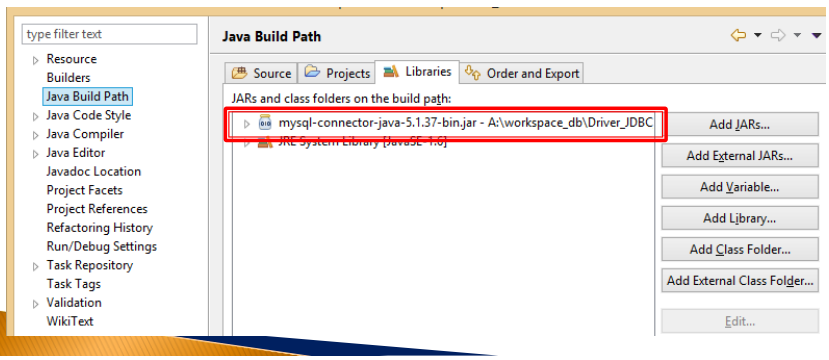


23

Passo-a-passo (1/5)

1. **Instale o Eclipse - Enterprise Edition (Java EE) - Mars**
   - Esta versão já inclui Eclipse Data Tool Platform (DTP)
   - Baixe o **driver de conexão** com o BD e configure uma conexão com o BC no Eclipse

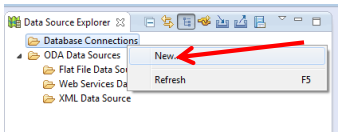
     Menu Window → Preferences → Data Management → Connectivity → Driver definitions
   - Este mesmo arquivo de driver deve ser adicionado ao **classpath** do projeto



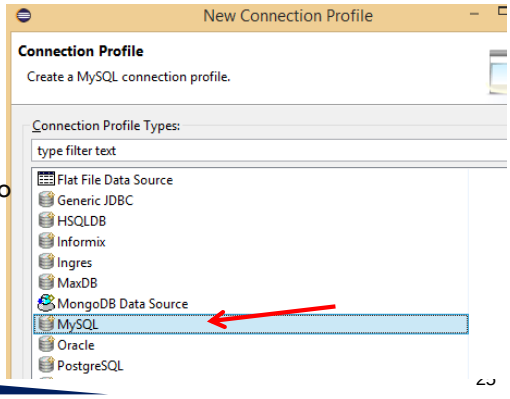
24

Passo-a-passo (2/5)

2. **Estabeleça uma nova conexão com o BD usando a perspetiva "Database development"**
   - No Eclipse:
     Menu Window → Open perspective → Other → Database development

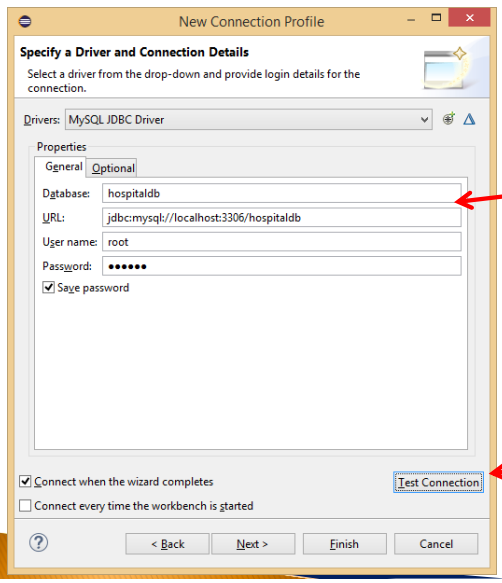


   - Selecione o tipo de SGBD na lista e dê nome a sua conexão

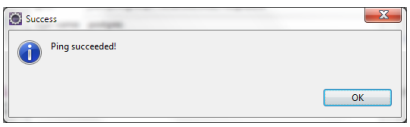


Passo-a-passo (3/5)

   - Configure sua conexão

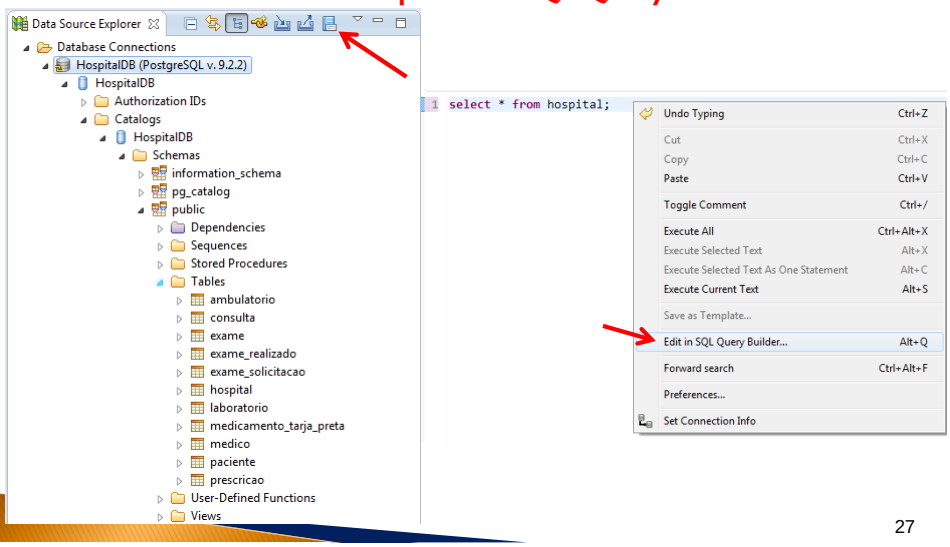


A base de dados já precisa ter sido criada



Passo-a-passo (4/5)

4. **Visualizando tabelas**
   - Usando o "**Data Source Explorer**" e "**SQL Query Builder**"

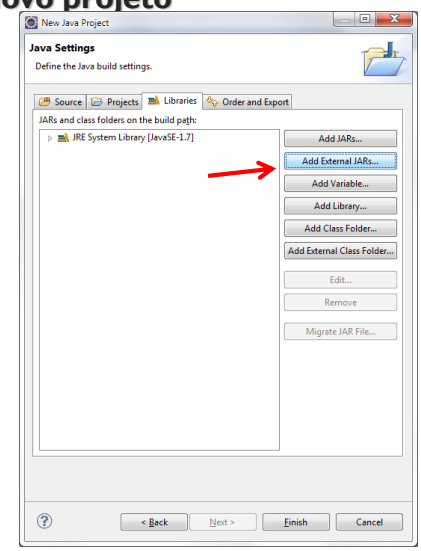


27

Passo-a-passo (5/5)

5. **Criando e configurando seu novo projeto**

- Crie um novo projeto Java comum
- Escolha um nome e pressione "Next"
- Clique na aba "Libraries" e depois em "**Add External JAR**".
- Aponte para o caminho onde se encontra o seu Driver (arquivo.jar)



28

Usando código Java com JDBC

Exemplos de código

UFRPE Universidade Federal Rural de Pernambuco | DEINFO Departamento de Estatística e Informática

19-dez-17

Em resumo

- **Para qualquer operação com o BD é necessária uma conexão.**
- **Procedimento Geral em JDBC (Java):**
  1. Carregar o Driver
  2. Abrir conexão a partir de um Gerenciador de Driver (*DriverManager*)
  3. Executar o *Statement*
  4. Processar o resultado (*ResultSet*), se houver
  5. Fechar a conexão
- **Um *Statement* pode ser de 3 tipos**
  - **Simples (*Statement*):** executa uma operação de busca (método `executeQuery`) ou atualizações do tipo INSERT, UPDATE ou DELETE (método `execute` ou `executeUpdate`)
  - **PreparedStatement:** pré-compila as consultas antes de executá-las.
    - **Mais indicado para comandos que serão executados várias vezes**
  - **CallableStatement:** permite a execução de *stored procedures*.

30

Carregando o Driver

```
try {
      Class.forName(“com.mysql.jdbc.Driver”)
    } catch (Exception e) {
      System.out.println("Problemas carregando o
        Driver do MySQL");
    }
```

31

Obtendo uma Conexão

```
/**
   * Método para retornar uma nova conexão com o banco de dados
   *
   */
  public static Connection getConexao() throws SQLException {
    Connection retorno = null;
    /*
     * Formato:
     * - Parâmetro 1: URLConexão:@endereço:porta
     * - Parâmetro 2: usuário
     * - Parâmetro 3: senha
     */
    retorno = DriverManager.getConnection(
        "jdbc:mysql://localhost:3306/hospitaldb", "root", “pwd“)

    return retorno;
  }
```

32

Exemplo de uso - Statement

```
Connection connection = getConexao();
Statement simpleStatement = connection.createStatement();
simpleStatement.execute(
  "insert into hospital " +
  "values ('778896.96', 'Hospital Geral', '6997744',
           'Rua da lama, 10, Recife, PE')"
 );
```


Exemplo de uso – Statement Query

```
try {
  Connection connection = getConexao();
  Statement s = connection.createStatement();
  ResultSet rs = s.executeQuery("select * from hospital");
  System.out.printf("%15s %25s %15s %40s\n",
      "CNPJ", "Hospital", "Telefone", "Endereço");
  while (rs.next()) {
    String cnpj = rs.getString("cnpj");
    String nome = rs.getString("nome");
    String telefone = rs.getString("telefone");
    String endereco = rs.getString("endereco");
    System.out.printf("%15s %25s %15s %40s\n",
      cnpj, nome, telefone, endereco);
  }
  s.close();
  connection.close();
} catch (Exception e) {
  e.printStackTrace();
}
```

Exemplo de uso - PreparedStatement

```
Connection connection = getConexao();
String query = "update hospital set nome = ? where cnpj = ?";
PreparedStatement ps = connection.prepareStatement(query);
ps.setString(1, "SUS_INSS");
ps.setString(2, "111666.22");
ps.executeUpdate();
ps.close();
```


Exemplo de uso – CallableStatement (1)

- **Função no MySQL (SCHEMA HospitalDB)**

```
CREATE FUNCTION `isRemedioForte`(remedio_id INT)
RETURNS int(11)

BEGIN  DECLARE res_id INT default 0;

SELECT id INTO res_id
 FROM hospitaldb.medicamento_tarja_preta
  WHERE  id = remedio_id;

RETURN res_id;

END
```

Exemplo de uso – CallableStatement (2)

```
// Executando função preexistente no BD
// Função: isRemedioForte(INT)
// Resultado: retorna o ID se o remédio é controlado, senão retorna 0
try {

  Connection connection = getConexao();
  CallableStatement cs = connection.prepareCall("{? = call isRemedioForte(?)}");

  cs.registerOutParameter(1, Types.INTEGER);  // registra o tipo de retorno da função
  cs.setInt(2,10);  // seta o parametro da função antes de chamá-la
  cs.execute();  // execute em vez de executeQuery()

  int output =  cs.getInt(1);
  if (output != 0)
     System.out.printf("O remédio com ID: " + output + " é um remédio controlado!");
  else
     System.out.printf("O remédio solicitado não foi encontrado!");

} catch (SQLException e) {
    // trata-se possíveis erros aqui
```

37

Atenção!!

- **Lembre-se de fechar:**
  - Statements;
  - PreparedStatements;
  - Connections;
- **Lembre-se de garantir a integridade das transações, por conexão:**
  - Commit;
  - Rollback.

38

Bibliografia

- Navathe, Shamkant B. e Elmasri, Ramez E. Sistemas de Banco de Dados. Pearson Brasil, 2005.
- Abraham, Silberschatz, Korth, Surdarshan. Sistema de Banco de Dados. Makron Books, 2004.
- Elmasri, Navathe. Sistema de Banco de Dados. Pearson Brasil, 2005.
- Heuser. Projeto de Banco de Dados. 4 ed. Instituto de Informática da UFRGS, 1998.

Exercício (1/2)

- Baixem o driver jdbc para o postgreSQL:
  - http://jdbc.postgresql.org/download.html
- Configurem o Eclipse criando uma conexão com a sua base de dados PostgreSQL
- Criem um novo projeto Java e configurem o driver como biblioteca do seu projeto

Exercício (2/2)

- **No Eclipse:**
  - Criem uma Main que acesse o banco de dados e:
    - Imprima os nomes dos funcionários/clientes/usuários do seu projeto;
    - Cadastre um funcionário/cliente/usuário no seu projeto.