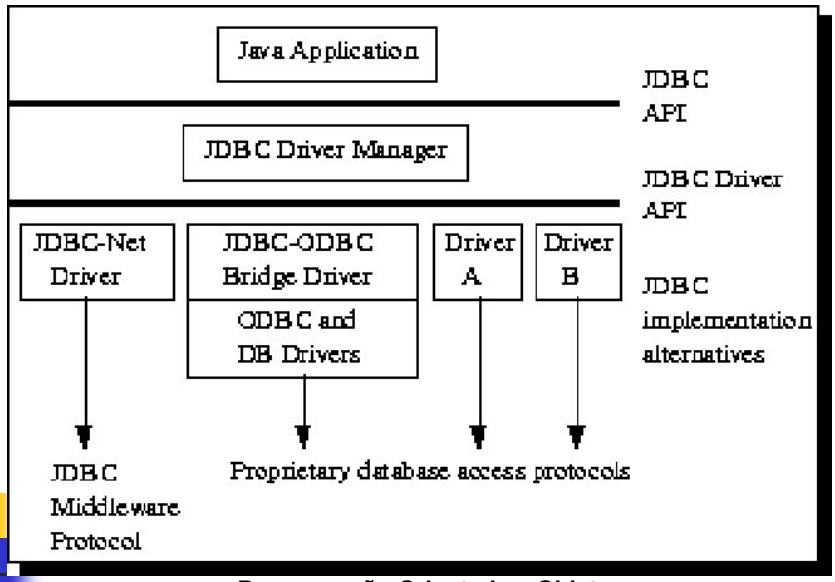
JAVA – JDBC

- JDBC – JAVA DATABASE CONNECTIVITY
  - Permite o acesso a banco de dados
  - Uma das formas de acesso é utilizando o driver JDBC-ODBC que permite a conexão através de um DRIVER OBDC
  - O **ODBC** (Open Database Connectivity) é um padrão para acesso aos banco de dados mais utilizados no mercado: SQLSERVER; ORACLE; MYSQL; POSTGRES; MS ACCESS;
  - COMO FUNCIONA
  - TIPOS DE DRIVERS
  - COMO UTILIZAR

JAVA – JDBC

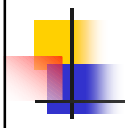
- COMO FUNCIONA



JAVA – JDBC

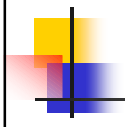
- Os drivers baseados na tecnologia JBDC são divididos em quatro tipos ou categorias.
- Os drivers do tipo 1, podem ser utilizados sempre que não houver um driver específico para um determinado banco de dados

1. *JDBC-ODBC + ODBC driver:* Java acessa o banco através de drivers ODBC. O driver deve ser carregado em cada cliente que realiza acesso ao banco.
2. *Driver Java com API-Nativa:* Neste caso as chamadas JDBC são convertidas diretamente em chamadas para a API dos banco de dados. Neste caso também é necessário que um código binário específico esteja presente no cliente



JDBC – TIPOS DE DRIVERS

3. *Driver Java Puro, JDBC-Net :* Este driver traduz chamadas JDBC em chamadas para um protocolo de Rede/DBMS independente que em seguida é traduzido para o DBMS por um servidor. Este Middleware permite que cliente java "puros" se conectem com diferentes BD
4. *Protocolo Nativo – Driver Java Puro:* Neste caso as chamadas JDBC são convertidas diretamente para o protocolo utilizado pelo DBMS, permitindo uma chamada direta do cliente para o servidor. A maioria destes drivers é proprietário.



JDBC – COMO UTILIZAR

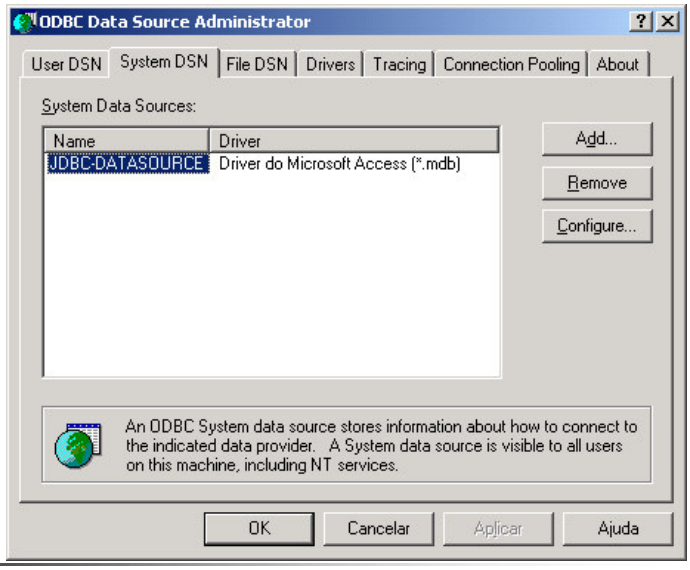
- OS seguintes passos são necessários para utilizar a tecnologia JDBC-ODBC
  1. CRIANDO A CONEXÃO ODBC COMO BD
  2. **CARREGAR O DRIVER**
  3. **CRIAR A CONEXÃO**
  4. **CRIAR COMANDOS SQL (STATEMENTS)**
  5. **PROCESSAR COMANDOS**
  6. **FINALIZAR A CONEXÃO COM O BANCO DE DADOS**
- Os passos de 2 a 6, são executados diretamente no código java.

JDBC – CRIANDO A CONEXÃO COM O BD

- Para utilizar os drivers do tipo 1, inicialmente é necessário carregar o driver na máquina do cliente, onde o aplicativo java será utilizado.
- Existem scripts para a instalação de drivers ODBC. No windows a instalação de um driver basicamente necessita de modificações no registro e a utilização de Dlls específicas para cada banco de dados.
- No windows através do painel de controle – "Fontes de dados OBDC" é possível criar novas conexões para os mais diversos banco de dados.

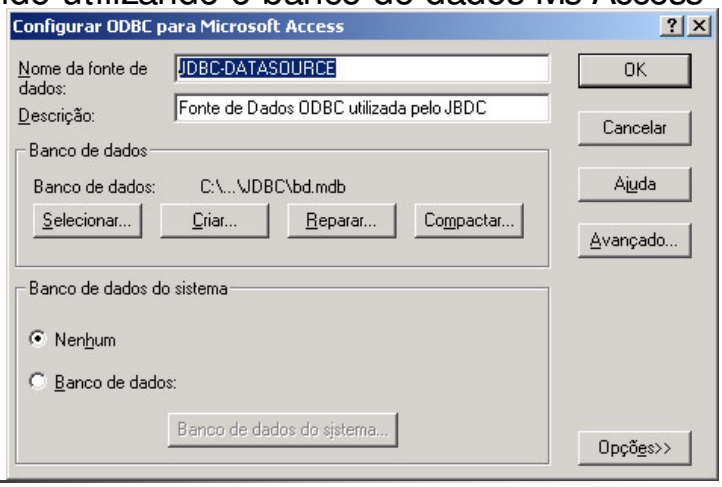
JDBC – CRIANDO A CONEXÃO COM O BD

- Abaixo é mostrado como criar uma fonte de dados ODBC



JDBC – CRIANDO A CONEXÃO COM O BD

- É necessário informar o nome da fonte de dados e a localização do banco de dados. Neste caso está sendo utilizando o banco de dados Ms Access



JDBC – CRIANDO A CONEXÃO COM O BD

- Ao clicar no botão "Avançado..." (mostrado na tela anterior) é possível configurar o Nome de Logon e senha utilizada pelo banco de dados

JDBC – CARREGANDO O DRIVER

- O primeiro passo para trabalhar com o JBDC é carregar o driver
- Inicialmente o driver deve ser carregado. No caso do JDBC-ODBC isto é feito da seguinte forma:

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
```

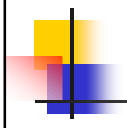
- O código acima deve estar dentro de um bloco try/catch.
- A chamada acima cria uma instância do driver e registra o mesmo juntamente como DriverManager
- Para a utilização do JBDC é necessário o pacote:

```
import java.sql.*;
```

JDBC – CRIANDO A CONEXÃO

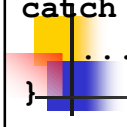
- Em seguida deve ser criada a conexão com o banco de dados. Para isto é utilizado o método getConnection da classe DriverManager.

  connection = DriverManager.getConnection(sBdName, sUserName, sPassword);
- Quando o driver é carregado uma instância da classe DriverManager é criada.
- A conexão retornada pelo método já está aberta e pronta para ser utilizada.



JDBC – Exemplos: Passo 2 e Passo 3

```
...
final String sBdName = "jdbc:odbc:Jdbc-
   Datasource";
protected Connection connection;
final String sUserName = "anonymous";
final String sPassword = "guest";
//Cria a conexão com o banco de dados
try{
  Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
  connection = DriverManager.getConnection
  (sBdName, sUserName, sPassword);
}
catch (ClassNotFoundException clex){
  ...
}
```



JDBC – CRIANDO COMANDOS SQL

- Um comando SQL consiste em um objeto da classe Statement.
- O primero passo para executar um comando é criar um objeto da classe Statement. Isto deve ser feito a partir da conexão criada anteriormente:

  statement = connection.createStatement();
- Para executar um SELECT deve ser utilizado o método executeQuery(), conforme mostrado abaixo:

```
public ResultSet executeQuery(String sql)
throws SQLException
```

- Para realizar uma modificação (INSERT; DELETE; UPDATE) no banco deve ser utilizado o método

```
public int executeUpdate(String sql)
throws SQLException
```


JDBC – Exemplo: Passo 4

```
...
//Cria um comando SQL
Statement statement;
//Cria um ResultSet
ResultSet resultset;
try{
 statement = connection.createStatement();
 resultset = statement.executeQuery(sSqlCommand);
 //displayResultSet(resultset);
 statement.close();
}
catch (SQLException sqlex){
...
}
```

JDBC – PROCESSANDO COMANDOS

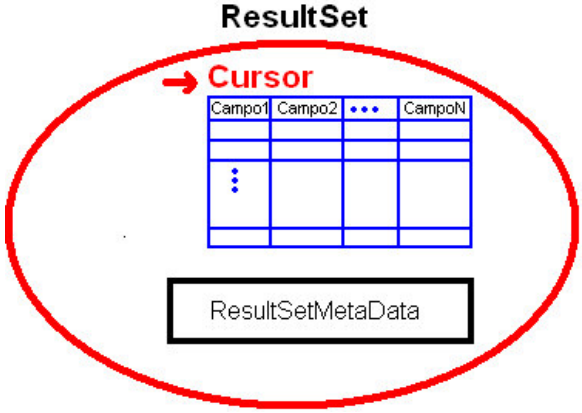
- Comandos o tipo "SELECT" retornam um objeto da classe ResultSet.
- Este objeto representa uma tabela e mantém um cursor apontando para uma linha de dados, além de informações sobre os campos da tabela.
- Inicialmente o cursor está posicionado antes da primeira linha. Para obter o primeiro registro é necessário utilizar o método –

```
public boolean next() throws
SQLException
```

- Para obter informações sobre a estrutura da tabela (nome; tipo e número de campos) deve ser utilizado um objeto do tipo ResultSetMetaData.

JDBC – PROCESSANDO COMANDOS

- A figura abaixo mostra um esquema do ResultSet
- Para obter todos os dados de um ResultSet é necessário percorrer todas as linhas, obtendo a informação de todas as colunas daquela linha



JDBC – PROCESSANDO COMANDOS

- Métodos para manipulação do cursor

  `public boolean next() throws SQLException` – Move próxima linha

  `public boolean previous() throws SQLException` – Move para a linha anterior

  `public boolean isLast() throws SQLException` – Verifica se a linha é a ultima

- Para obter o ResultSetMedaData o seguinte método do ResultSet deve ser utilizado:

```
public ResultSetMetaData getMetaData()
throws SQLException
```


JDBC – PROCESSANDO COMANDOS

- A partir do ResulSetMetaData é possível obter o número, o tipo e o nome das colunas da tabela:

  `public int getColumnCount() throws SQLException` – Recupera o número de colunas do ResultSet

  `public int getColumnType(int column) throws SQLException` – Recupera o tipo de dado contido na coluna.

  `public String getColumnName(int column) throws SQLException` – Recupera o nome da Coluna

- Os tipos das colunas são representados por um objeto da classe Types.

JDBC – Exemplo: Passo 5

```
...
//coloca o cursor no primeiro registro
boolean moreRecords;
//Como inicialmente o cursor esta antes da
//primeira linha, então deve ser movido,
//inicialmente para a primeira
moreRecords = rs.next();
if (!moreRecords){
  JOptionPane.showMessageDialog(null,"Fim dos
   registros");
  return;
}
//Vetor que irá conter campos(colunas)e as linhas
Vector colunas = new Vector();
Vector linhas = new Vector();
```


JDBC – Exemplo: Passo 5

```
//continua...
Vector linha;
try{
  //obtem o nome dos campos da tabela
  ResultSetMetaData rsmd = rs.getMetaData();
  for (int i = 1; i <= rsmd.getColumnCount();
   ++i){
    colunas.addElement(rsmd.getColumnName(i));
  }
  //obtem os dados de cada campo
  do {
    linha = getRowData(rs, rsmd);
    linhas.addElement(linha);
  } while (rs.next());
```

JDBC – Exemplo: Passo 5

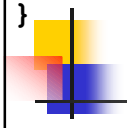
```
  //continua...
  //Adiciona tabela ao JFrame
  table = new JTable(linhas,colunas);
  JScrollPane scrollTable = new
  JScrollPane(table);
  getContentPane().add(scrollTable,
       BorderLayout.CENTER);
  //Reposiona os componentes no container
  validate();
}
catch (SQLException sqlex){
    JOptionPane.showMessageDialog(null,
    sqlex.getMessage());
}
```


JDBC – Exemplo: Passo 5

```
public Vector getRowData(ResultSet rs,
    ResultSetMetaData rsmd) throws SQLException
{
  Vector linhaAtual = new Vector();
  for (int i = 1; i <= rsmd.getColumnCount();
    ++i){
     switch (rsmd.getColumnType(i)){
       case Types.VARCHAR:
         linhaAtual.addElement(rs.getString(i));
         break;
     case Types.INTEGER:
         linhaAtual.addElement(newLong(
         rs.getLong(i)));
         break;
     //continua...
```

JDBC – Exemplo: Passo 5

```
    case Types.DOUBLE:
      linhaAtual.addElement(newDouble(
      rs.getDouble()));
      break;
    default:
      JOptionPane.showMessageDialog(null,"Tipo
      não suportado"+ rsmd.getColumnTypeName(i));
      break;
    }
  }
  return linhaAtual;
}
```



JDBC – FINALIZANDO A CONEXÃO

- Após o processamento a conexão com o banco de dados, que foi criada inicialmente deve ser fechada.
- Para isto é utilizado o seguinte método:

```
public void close() throws SQLException
```