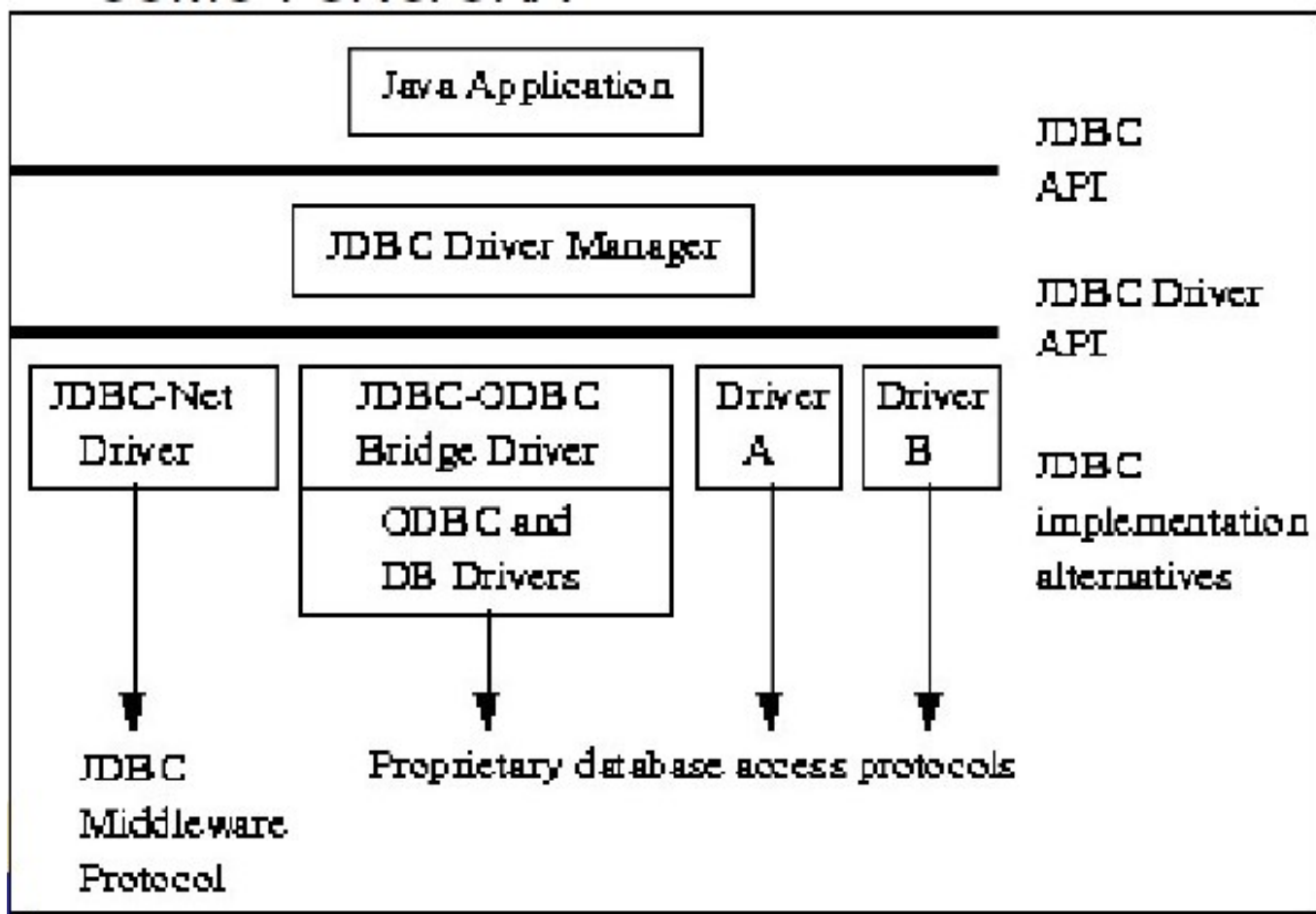


Java JDBC

Como funciona

- Existem várias formas de conexão de uma aplicação Java com o banco de dados



Java JDBC – Tipos de Drivers

- ❑ Os drivers baseados na tecnologia JDBC 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

Java 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.

Java - JDBC

Como Utilizar

- Os seguintes passos são necessários se conectar uma aplicação Java com um banco de dados
 1. Criar a conexão ODBC com o BD (apenas no caso JDBC-ODBC)
 2. Carregar o Driver
 3. Criar a Conexão
 4. Criar os comandos SQL
 5. Processar os comandos
 6. Finalizar a conexão com o banco de dados
- Os passos de 2 a 6, são executados diretamente no código java.

Java JDBC

Carregar Driver

- Para uso do JDBC inicialmente é necessário é carregar o driver
- Isto é feito da seguinte forma:
 - JDBC-ODBC
 - `Class.forName("sun.jdbc.odbc.JdbcOdbcDriver") ;`
 - Driver Tipo 4 – Postgres
 - `Class.forName("org.postgresql.Driver") ;`
- O código acima deve estar dentro de um bloco try/catch.
- A chamada cria uma instância do driver e registra o mesmo juntamente como DriverManager
- Para a utilização do JDBC é necessário o pacote:
 - `import java.sql.* ;`

Java JDBC

Criar 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);`
- ❑ A conexão retornada pelo método já está aberta e pronta para ser utilizada.

Java JDBC

Criar os comandos SQL

- ❑ Um comando SQL consiste em um objeto da classe Statement.
- ❑ Através da conexão obtida anteriormente é possível criar um objeto que representa um comando SQL
 - `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`

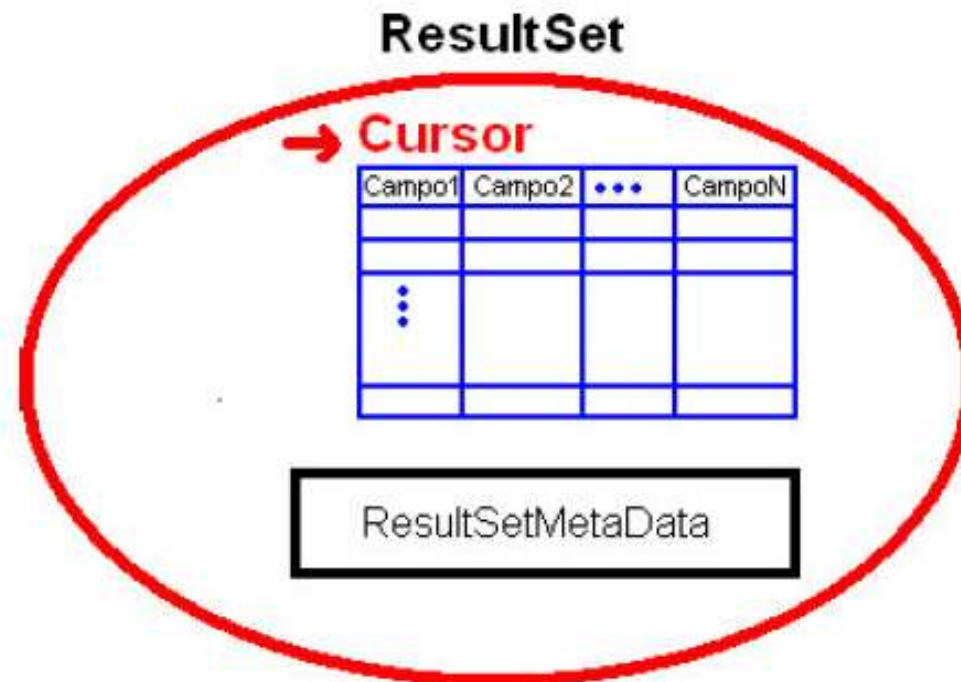
Processar os Comandos

- ❑ 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.

Java - JDBC

Processar os 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



Processar os Comandos

- Métodos para manipulação do cursor
 - Move para a próxima linha
 - `public boolean next() throws SQLException`
 - Move para a linha anterior
 - `public boolean previous() throws SQLException`
 - Verifica se a linha é a última
 - `public boolean isLast() throws SQLException`
- Para obter o `ResultSetMetaData` o seguinte método do `ResultSet` deve ser utilizado:
 - `public ResultSetMetaData getMetaData() throws SQLException`

Processar os Comandos

- ❑ A partir do `ResultSetMetaData` é possível obter o número, o tipo e o nome das colunas da tabela:
- ❑ Recupera o número de colunas do `ResultSet`
 - `public int getColumnCount() throws SQLException`
- ❑ Recupera o tipo de dado contido na coluna
 - `public int getColumnType(int column) throws SQLException`
- ❑ Recupera o nome da Coluna
 - `public String getColumnName(int column) throws SQLException`
- ❑ Os tipos das colunas são representados por um objeto da classe `Types`.

Java - JDBC

Finalizar 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`

Consultas com Parâmetros

- ❑ O usual é que as consultas possuem parâmetros
- ❑ Neste caso deve ser utilizada a classe PreparedStatement
- ❑ No valor de cada parâmetro é colocado um caractere ?
- ❑ A atribuição de valores aos parâmetros é feita através dos métodos setXXX(indiceParametro,valor)
 - indiceParametro equivale a posição do parâmetro da esquerda para a direita da consulta
 - Os métodos setXXX são chamados de acordo com o tipo de dados do parâmetro por exemplo:
 - ❑ void setDouble(int parameterIndex, double x)
 - ❑ void setInt(int parameterIndex, int x)
 - ❑ void setDate(int parameterIndex, Date x)
 - ❑ void setString(int parameterIndex, String value)

Java – JDBC

Consultas com Parâmetros

■ Exemplos

```
String sqlCmd = "UPDATE EMPLOYEES SET SALARY = ? WHERE ID = ?"
```

```
PreparedStatement pstmt = con.prepareStatement(sqlCmd);
```

```
pstmt.setBigDecimal(1, 153833.00)
```

```
pstmt.setInt(2, 110592)
```

```
pstmt.executeUpdate();
```

```
String sqlCmd = "select * from company.deleteEmployee(?)"
```

```
PreparedStatement pstmt = con.prepareStatement(sqlCmd);
```

```
pstmt.setInt(1, 110592)
```

```
pstmt.executeQuery();
```