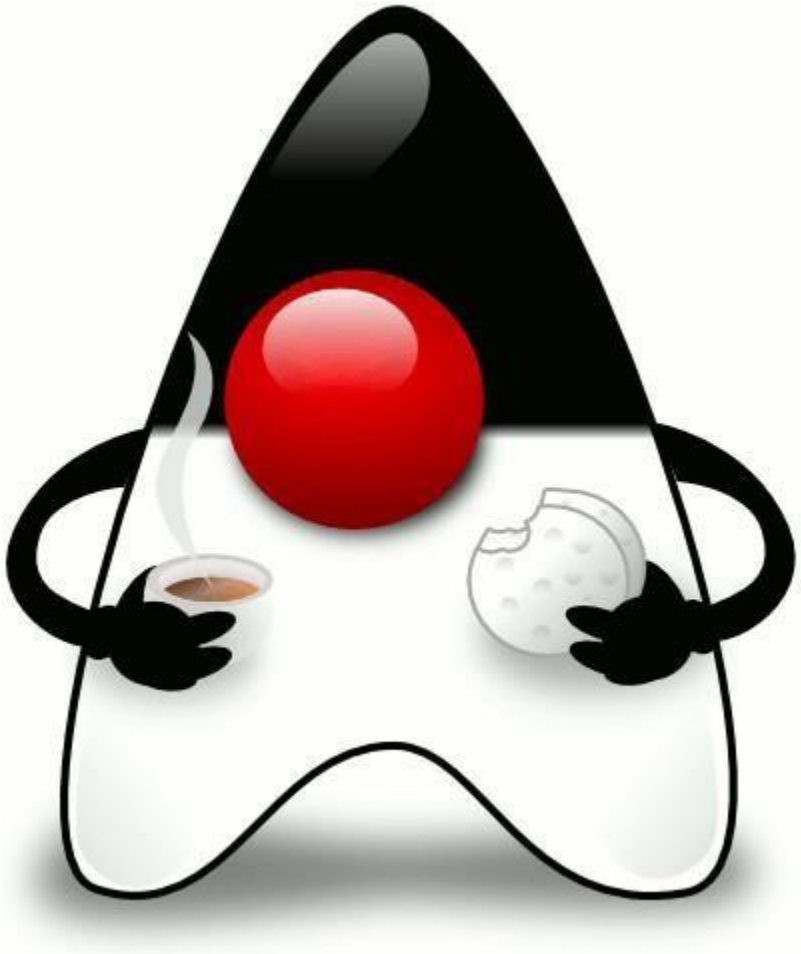
UNIP
UNIVERSIDADE PAULISTA

Disciplina: Aplicação de Linguagem de Programação Orientada a Objetos (ALPOO)

UNIP
UNIVERSIDADE PAULISTA

Introdução a Java Database Conectivity (JDBC)



Introdução a JDBC

Os programas Java comunicam-se com BD e manipulam seus dados utilizando a API JDBC (Java DataBase Connection).

Um driver JDBC permite aos aplicativos Java conectar-se a um BD em um SGDBMS particular e permite aos programadores manipular esse banco de dados utilizando a API JDBC.

O JDBC e quase sempre utilizado com um BD relacional, mas pode ser utilizado com qualquer origem de dados baseada em tabela.

UNIP
UNIVERSIDADE PAULISTA

Introdução a JDBC

Um Banco de dados e uma colecao de dados inter-relacionados, representando informacoes sobre um dominio especifico.

Basicamente e uma tabela composta de varias linhas divididas em colunas que sao identificadas por campos, e cada linha representa um registro do banco de dados.

Um Sistema de Gerenciamento de Bancos de Dados (SGDBMS – DataBase Management System) fornece mecanismos para armazenar, organizar, recuperar e modificar dados para muitos usuarios.

UNIP
UNIVERSIDADE PAULISTA

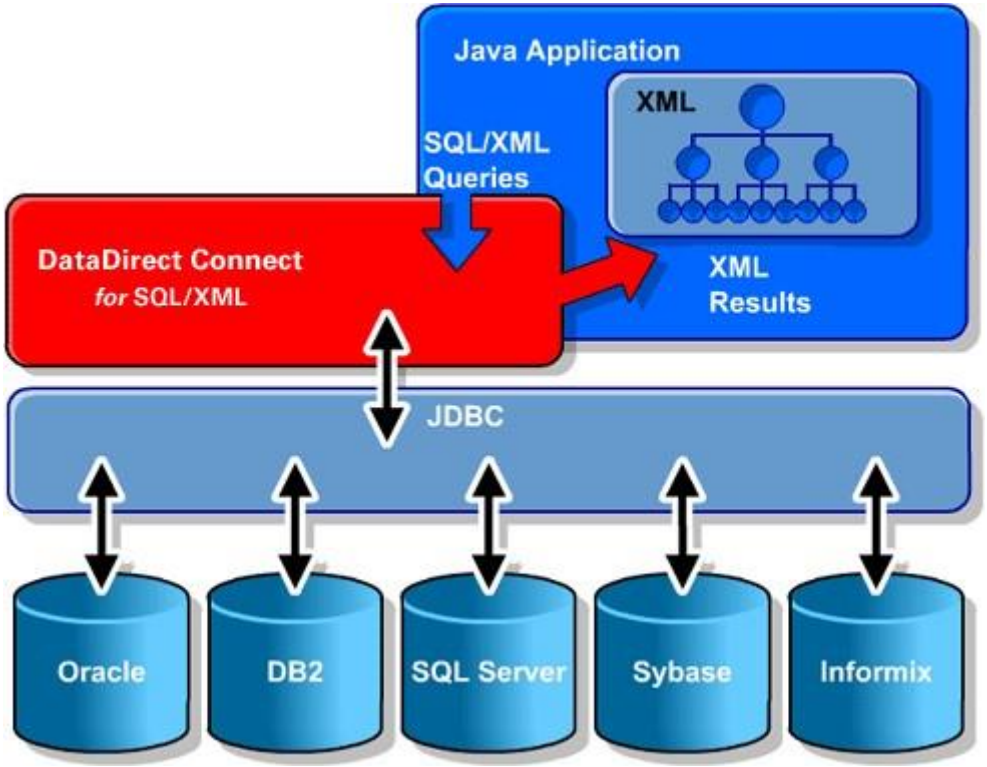
Introdução a JDBC

Para manipular informações em um SGDBMS é utilizada uma linguagem padrão para BD conhecida como Structured Query Language (SQL).

A SQL e um padrão internacional utilizada em BD relacionais para realizar consultas, isto e, solicitar informacões que satisfazem os critérios dados, e manipular estes dados.

Alguns SGDBRMS populares são MS SQL Server, Oracle, Sybase, IBM DB2, Informix, PostgreSQL e MySQL.

Introdução a JDBC



Linguagem SQL

A linguagem SQL for originalmente desenvolvida na IBM em um projeto de SGBDR, como uma linguagem de consulta a BD denominada SEQUEL, sigla para Structured English Query Language (Linguagem de Consulta Estruturada em Inglês).

A IBM também implementou SQL em seus sistemas de BD DB2 e SQL/DS.

Entre os anos 80 e 90, os produtos com SQL se multiplicaram e hoje SQL é largamente implementada e aceita como o padrão de fato da indústria para linguagem de acesso a BD, desde sistemas desktop a Mainframes

Em 1986 o **American National Standard Institute (ANSI)**, publicou um padrao SQL. A ultima especificacao publica pelo ANSI foi em 1992. Hoje a linguagem SQL e conhecida como SQL ANSI-92.

Banco de Dados Relacionais

Um BD relacional é uma representação lógica de dados que permite o acesso aos dados sem considerar sua estrutura física.

Em um BD são armazenados as tabelas com as informações.

As tabelas são compostas de linhas (registros) e as linhas são compostas de colunas (campos) nas quais os valores são armazenados.

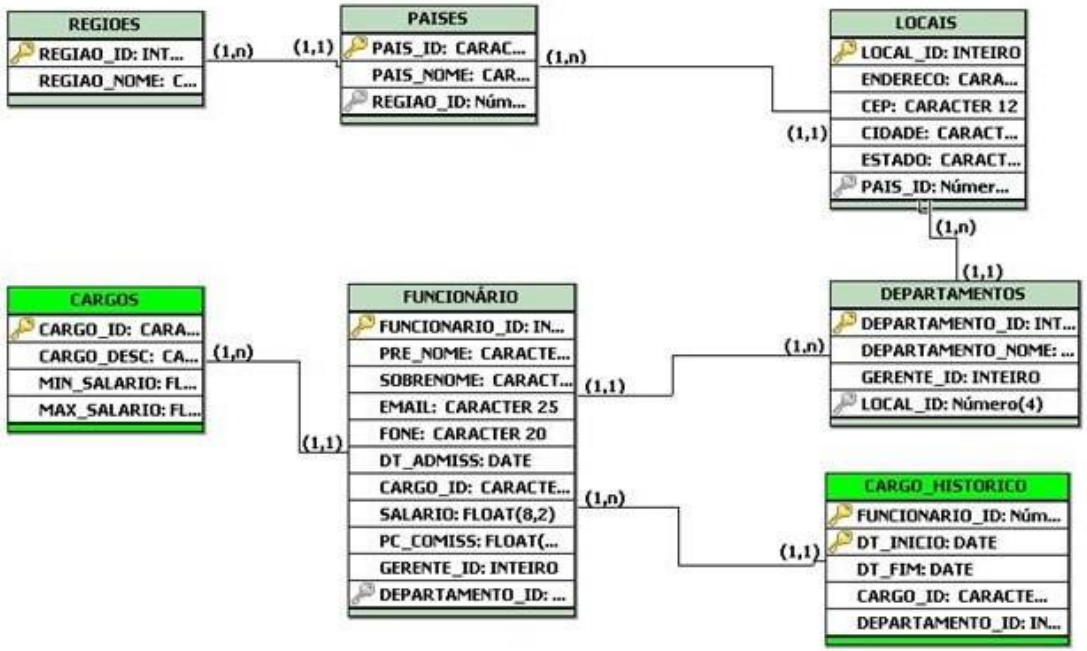
Banco de Dados Relacionais

Numero	Nome	Departamento	Salario	Localizacao
23603	João	413	1100	São Paulo
24568	Carlos	413	2000	São Paulo
34589	Lailson	642	1800	Campinas
35761	Marcos	611	1500	Americana
47132	Nilmar	413	9000	São Paulo
78321	Salomão	611	8500	Americana

Tabela Empregados com seis linhas (registros) e 5 colunas (campos)

Diagrama entidade relacionamento

ER (entity relationship), também conhecido como Entidade-Relacionamento. Esse diagrama mostra as tabelas no BD e os relacionamentos entre elas.



Componentes da linguagem SQL e operadores

Linguagem de definicao de dados (Data Definition Language - DDL)

Inclui a sintaxe mais utilizada para definir, alterar e eliminar as tabelas do banco de dados.

Criação dos esquemas (estrutura das tabelas, definição de visões...), exclusão, criação de índices, modificação nos Esquemas.

Lista das principais Sentenças:
CREATE (criar);
ALTER (alterar); e
DROP (apagar).

Componentes da linguagem SQL e operadores

Linguagem de consulta de dados (Data Query Language - DQL)

Contem os componentes da linguagem e conceitos para a consulta e recuperação da informação armazenados em tabelas.

Sentença:

SELECT (listar linhas)

Componentes da linguagem SQL e operadores

Linguagem de Manutenção de dados (Data Manipulation Language - DML)

Contem os componentes da linguagem e conceitos para a manipulação da informação armazenados em tabelas.
Inserção, exclusão e alteração;

Lista das principais Sentenças:
INSERT (inserir dados),
DELETE (apagar dados); e
UPDATE (atualizar dados).

Componentes da linguagem SQL e operadores

Operadores

Operadores logicos: AND, OR

Operadores de negacao:NOT

Operadores aritmeticos: *, /, -, +

Operadores de comparacao:

= (igualdade),
!= (diferenca),
<,>
=<
>=

Componentes da linguagem SQL e operadores

Palavra chave de SQL	Descrição
SELECT	Recupera dados de uma ou mais tabelas.
FROM	Tabelas envolvidas na consulta. Requeridas em cada SELECT.
WHERE	Critérios de seleção que determinam as linhas a ser recuperadas, excluídas ou atualizadas. Opcional em uma consulta ou uma instrução SQL.
GROUP BY	Critérios para agrupar linhas. Opcional em uma consulta SELECT.
ORDER BY	Critérios para ordenar linhas. Opcional em uma consulta SELECT.
INNER JOIN	Mescla linhas de múltiplas tabelas.
INSERT	Insere linhas em uma tabela especificada.
UPDATE	Atualiza linhas em uma tabela especificada.
DELETE	Exclui linhas de uma tabela especificada.

Palavras chave de uma consulta SQL

Acessando dados com a aplicação em Java

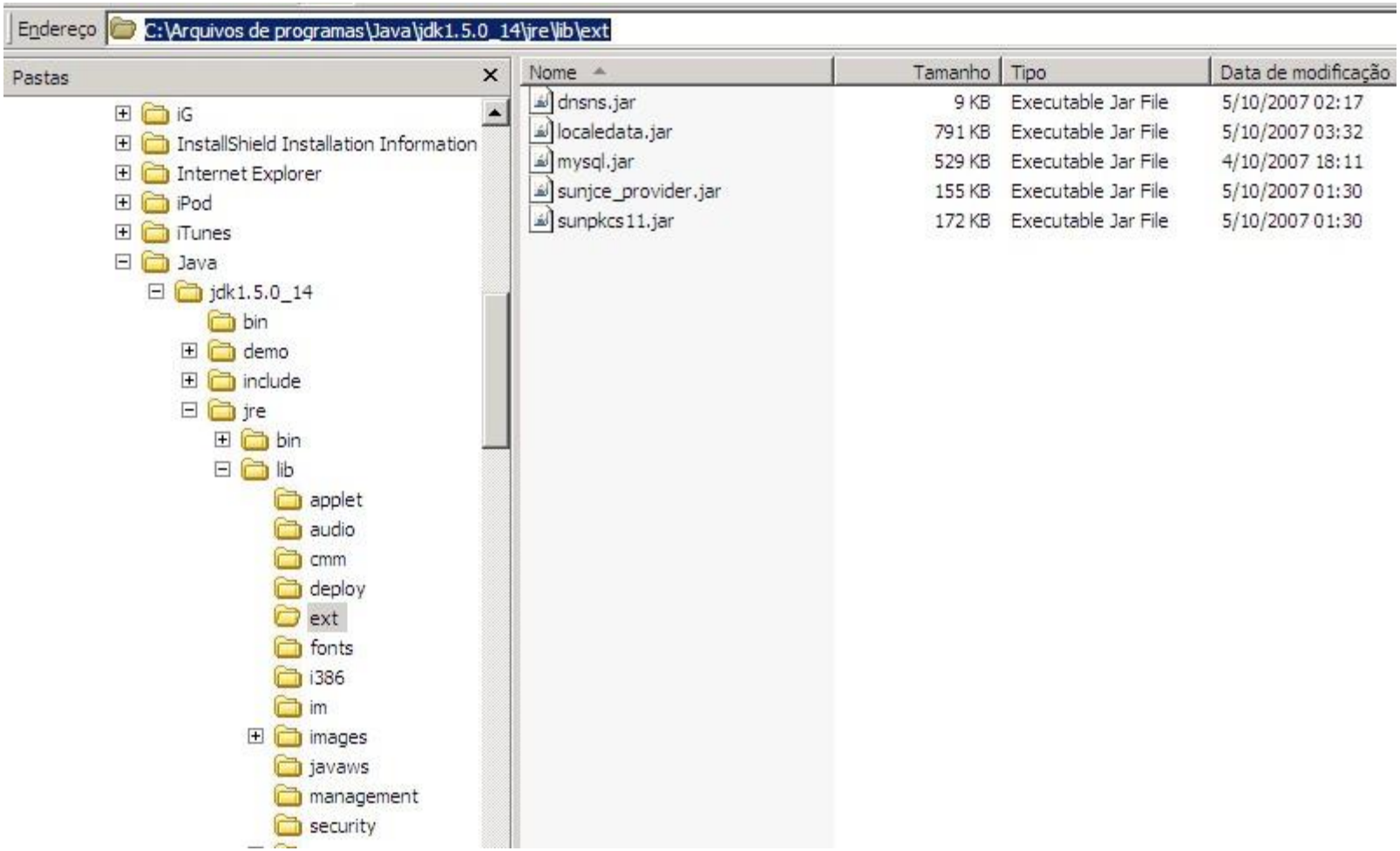
Para acessar os dados armazenados em uma banco de dados MySql devem ser realizados os seguintes passos:

1. Baixe o arquivo **mysql-connector-java-5.0.8.tar.gz.**

2. Descompacte o arquivo e coloque o arquivo **mysql-connectorjava-5.0.8-bin.jar** no diretorio **C:\Arquivos de programas\Java\jdk1.5.0_14\jre\lib\ext** e renomeie para **mysql.jar.**

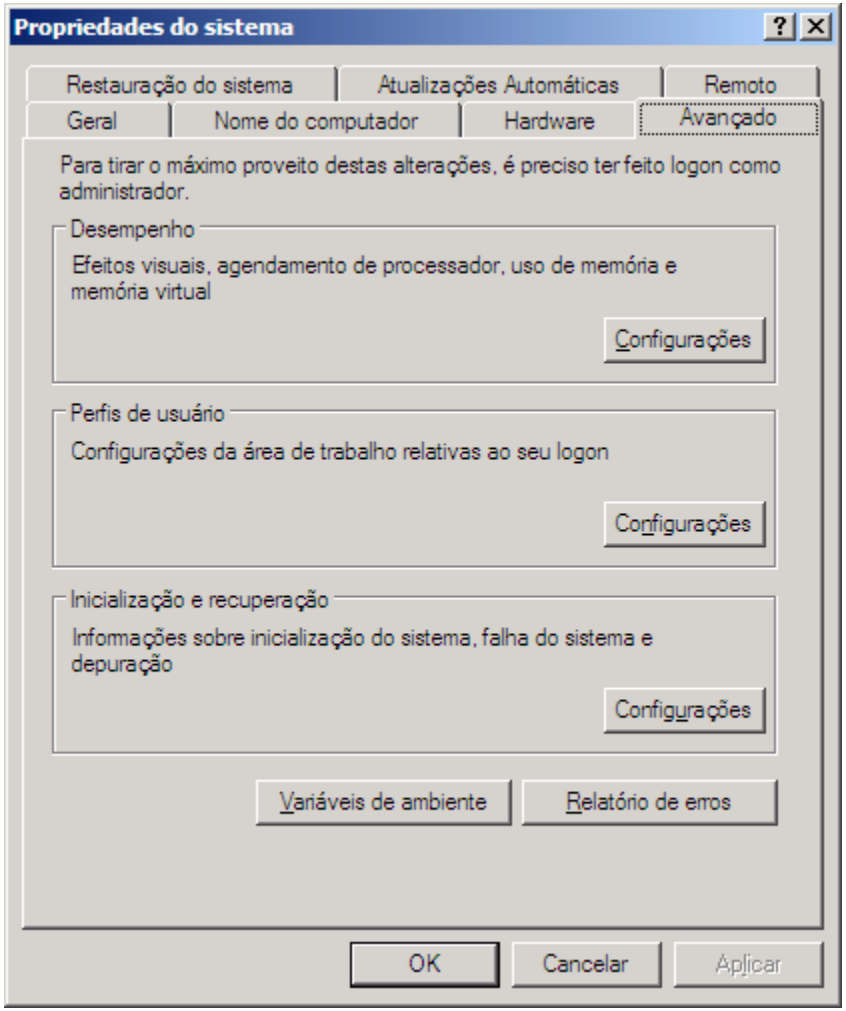
3. Crie no CLASSPATH um alias para esse arquivo como mostra o exemplo a seguir.

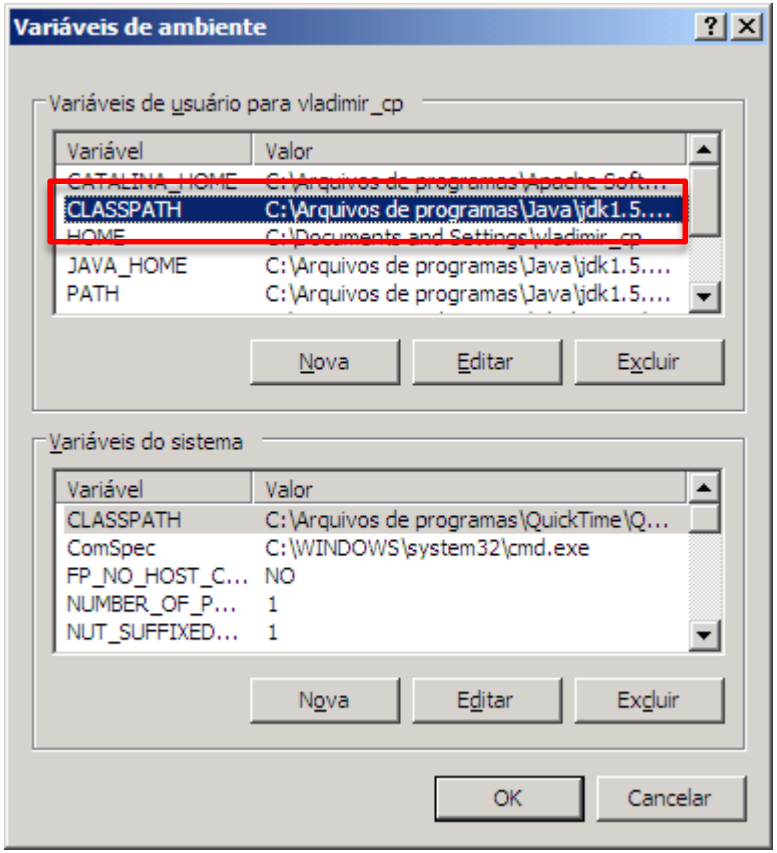
Acessando dados com a aplicação em Java



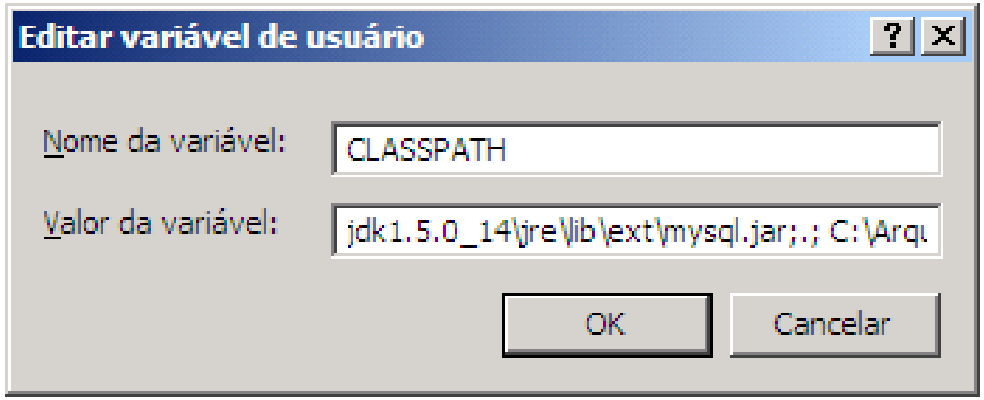
UNIP
UNIVERSIDADE PAULISTA

Acessando dados com a aplicação em Java

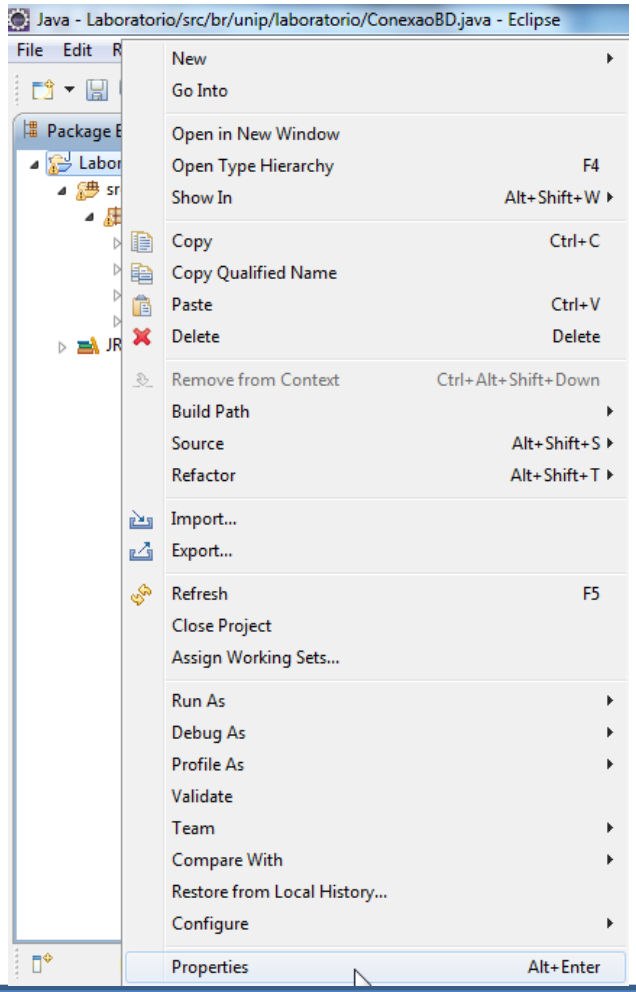




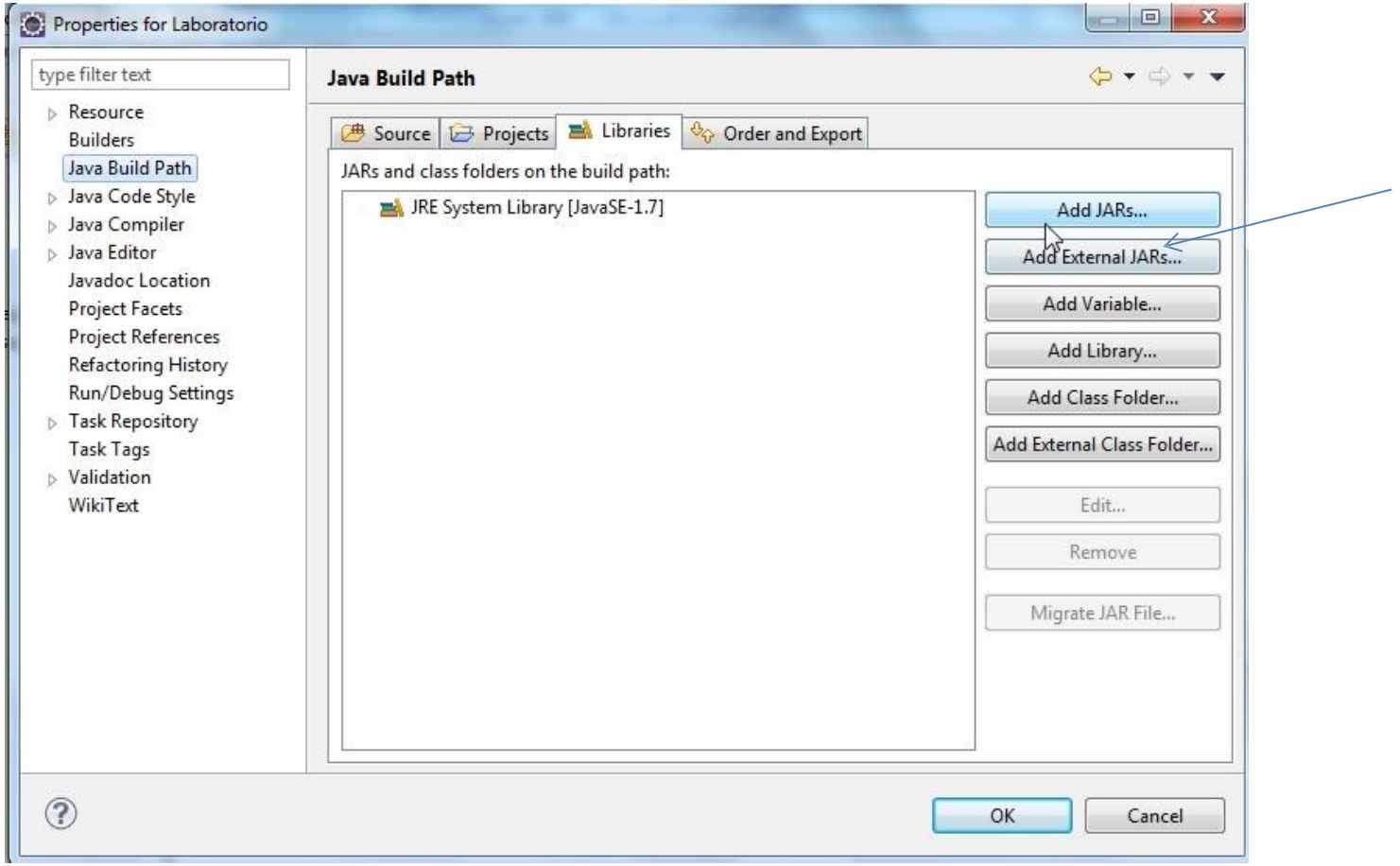
Acessando dados com a aplicação em Java



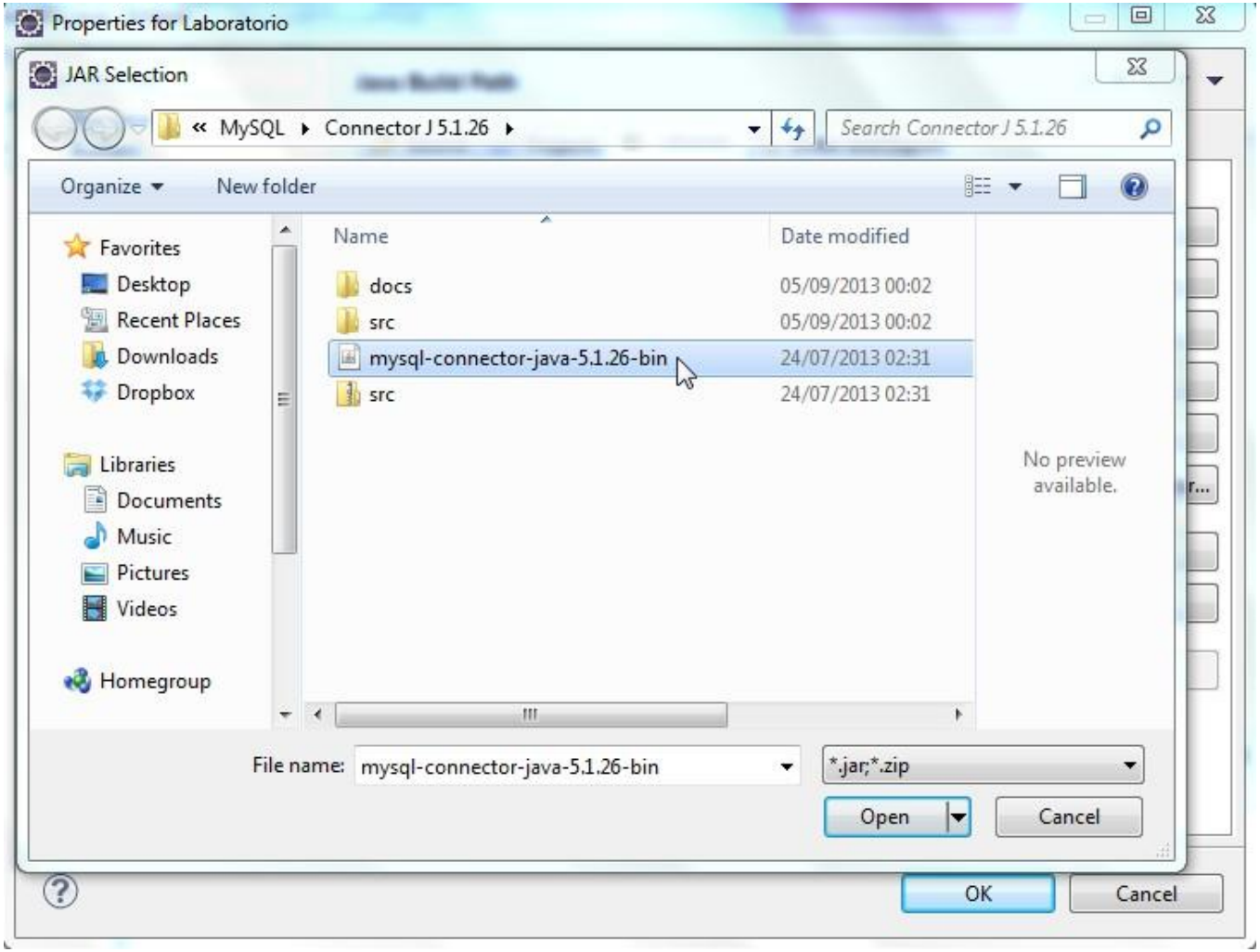
Acessando dados com a aplicação em Java – Através do eclipse



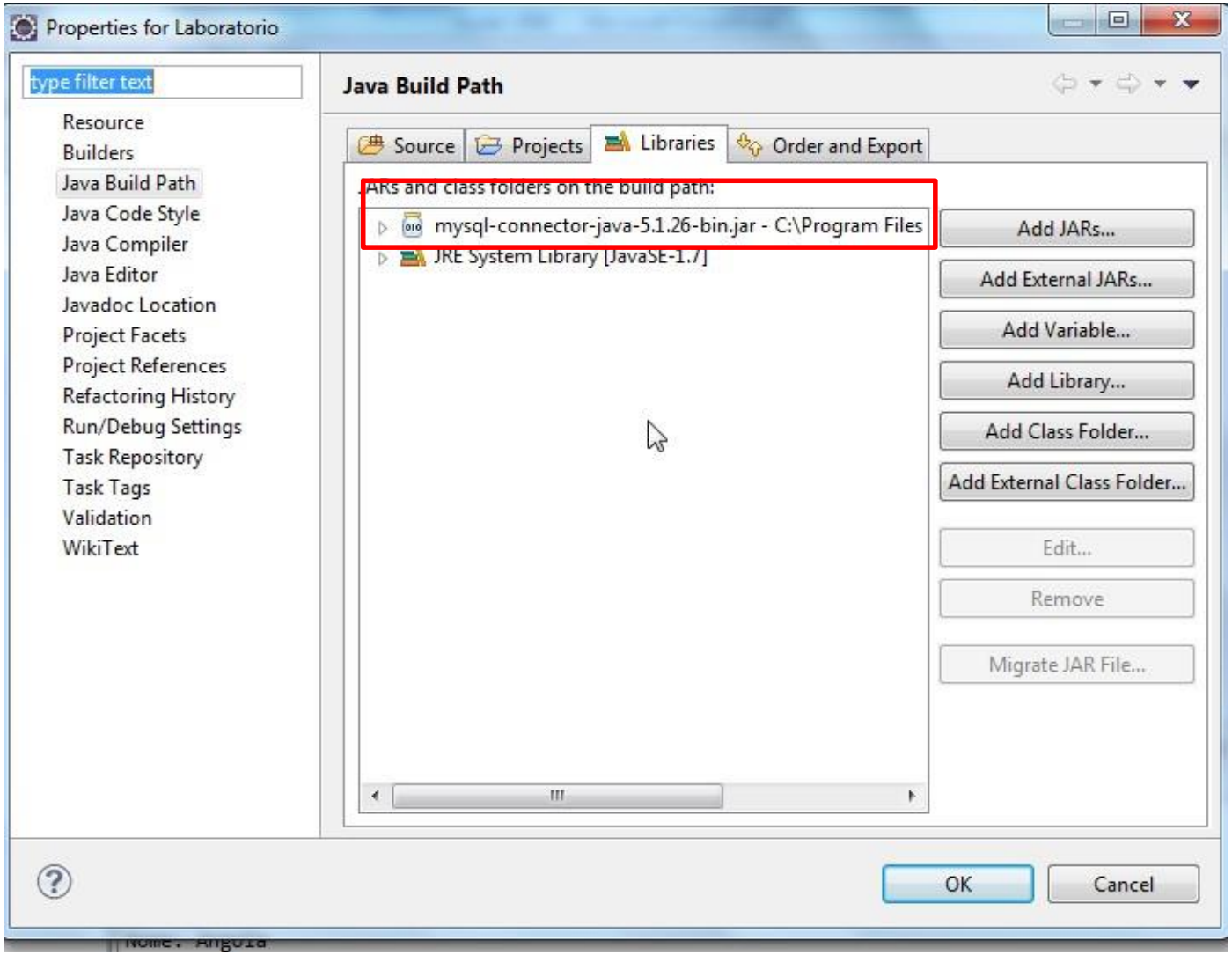
Acessando dados com a aplicação em Java – Através do eclipse



Acessando dados com a aplicação em Java



Acessando dados com a aplicação em Java



Exemplo de código para acessar o banco

```
public static void main(String[] args) {
    try {
        // Especifica o driver JDBC usado para a conexao
        // Nesse exemplo eh MySQL
        Class.forName("com.mysql.jdbc.Driver");

        // URL de conexao com o banco
        String url  = "jdbc:mysql://localhost:3306/world";

        // Usuario de acesso ao banco
        String login = "root";

        // Senha de acesso ao banco
        String senha = "password";

        Connection conn = DriverManager.getConnection(url, login, senha);

        Statement stat = conn.createStatement();

        ResultSet rs = stat.executeQuery("select * from country");

        while(rs.next()) {
            String code = rs.getString("code");
            String nome = rs.getString("name");

            System.out.println("Cod: " + code + "Nome: " + nome);
        }

    } catch (ClassNotFoundException e) {
        System.out.println("Driver jdbc nao encontrado");
    } catch (SQLException e) {
        System.out.println("Problema na execucao da instrucao no banco de dados");
    }
```

UNIP
UNIVERSIDADE PAULISTA

Resultado da execução

```
<terminated> ConexaoBD [Java Application] C:\Program Files\Java\jre7\bin\javaw
Cod: TGO Nome: Togo
Cod: THA Nome: Thailand
Cod: TJK Nome: Tajikistan
Cod: TKL Nome: Tokelau
Cod: TKM Nome: Turkmenistan
Cod: TMP Nome: East Timor
Cod: TON Nome: Tonga
Cod: TTO Nome: Trinidad and Tobago
Cod: TUN Nome: Tunisia
Cod: TUR Nome: Turkey
Cod: TUV Nome: Tuvalu
Cod: TWN Nome: Taiwan
Cod: TZA Nome: Tanzania
Cod: UGA Nome: Uganda
Cod: UKR Nome: Ukraine
Cod: UMI Nome: United States Minor Outlying Islands
Cod: URY Nome: Uruguay
Cod: USA Nome: United States
Cod: UZB Nome: Uzbekistan
Cod: VAT Nome: Holy See (Vatican City State)
Cod: VCT Nome: Saint Vincent and the Grenadines
Cod: VEN Nome: Venezuela
Cod: VGB Nome: Virgin Islands, British
Cod: VIR Nome: Virgin Islands, U.S.
Cod: VNM Nome: Vietnam
Cod: VUT Nome: Vanuatu
Cod: WLF Nome: Wallis and Futuna
Cod: WSM Nome: Samoa
Cod: YEM Nome: Yemen
Cod: YUG Nome: Yugoslavia
Cod: ZAF Nome: South Africa
Cod: ZMB Nome: Zambia
Cod: ZWE Nome: Zimbabwe
```

Pacote java.sql

Fornece a API para acesso e processamento de dados;

Principais classes e interfaces são:

DriverManager: responsável por criar uma conexão com o BD;

Connection: classe responsável por manter uma conexão aberta com o BD;

Statement: gerencia e executa instruções SQL;

PreparedStatement: gerencia e executa instruções SQL, permitindo também a passagem de parâmetros em uma instrução;

ResultSet: responsável por receber os dados obtidos em uma pesquisa ao banco.

Pacote java.sql

Para abrir uma conexão com um banco de dados e preciso realizar duas tarefas distintas:

- verificar a existência do driver de conexão e solicitar a abertura da Conexão.
- A verificação da existência do driver pode ser feita utilizando-se a seguinte instrução:

```
try {
    // Especifica o driver JDBC usado para a conexao
    // Nesse exemplo eh MySQL
    Class.forName("com.mysql.jdbc.Driver");

} catch (ClassNotFoundException e) {
    System.out.println("Driver jdbc nao encontrado");
}
```

Pacote java.sql

Connection

representa a conexão com o banco de dados;

proporcionar informações sobre as tabelas do BD por meio de transações;

Os métodos desta interface frequentemente utilizados são:

***commit()*:** executa todas as alterações feitas com o banco de dados pela atual transação.

***rollback()*:** desfaz qualquer alteração feita com o banco de dados pela atual transação.

***close()*:** libera o recurso que estava sendo utilizado pelo objeto.

Pacote java.sql

A String "com.mysql.jdbc.Driver" representa o nome de classe do driver MySQL JDBC a ser utilizada.

Para carregá-la, basta invocar o método estático forName() da classe java.lang.Class.

A próxima tarefa é solicitar a abertura da conexão utilizando a classe java.sql.DriverManager da interface java.sql.Connection.

A DriverManager representa o servico básico de gerenciamento de um conjunto de drivers JDBC.

Ela contem um método estático, chamado getConnection(), responsável por estabelecer uma conexão com um banco de dados.

Pacote java.sql

Depois de ser utilizada, a conexão deve ser finalizada.

Para terminar uma conexão representada por uma instancia de classe Connection, basta invocar seu método close().

```
        } catch (ClassNotFoundException e) {
            System.out.println("Driver jdbc nao encontrado");
        } catch (SQLException e) {
            System.out.println("Problema na execucao da instrucao no banco de dados");
        } finally {
            if(conn != null) {
                conn.close();
            }
        }
```

Pacote java.sql

Para enviar uma instrução SQL ao banco de dados (BD), a aplicação Java precisa criar uma instância da interface Statement.

A função desta interface é enviar instruções SQL ao banco e captar o retorno produzido.

Para instanciar a interface Statement, é preciso ter uma conexão aberta com o BD.

Uma instancia da interface Statement e criada utilizando-se o metodo estatico createStatement() da interface Connection.

Toda instancia da interface Statement é vinculada, no momento de sua criação, a uma conexão ativa e, portanto, as instruções SQL a serem enviadas por ela já tem um BD específico como destino.

Pacote java.sql

Para executar uma instrução SQL no banco de dados deve ser utilizado os métodos da classe Statement:

executeQuery: para executar somente a instrução SELECT.

executeUpdate: para executar as instruções DELETE, INSERT e UPDATE.

Pacote java.sql

A interface ResultSet representa uma estrutura de dados bidimensional (matriz) resultante de uma consulta a um banco.

Uma instância dessa interface mantem um cursor apontando para uma linha dessa estrutura de dados, que inicialmente esta posicionada antes do primeiro registro.

O método next() move o cursor para a próxima linha retornando um valor booleano (false) quando é invocado e já se encontra na ultima linha.

Em conjunto com o laco while possibilita a varredura dos dados contidos nessa estrutura. O método get<tipo>(<campo>) permite recuperar o valor do campo para o registro atual.