Tópicos avançados em Programação

JDBC - ***Java Data Base Connection***

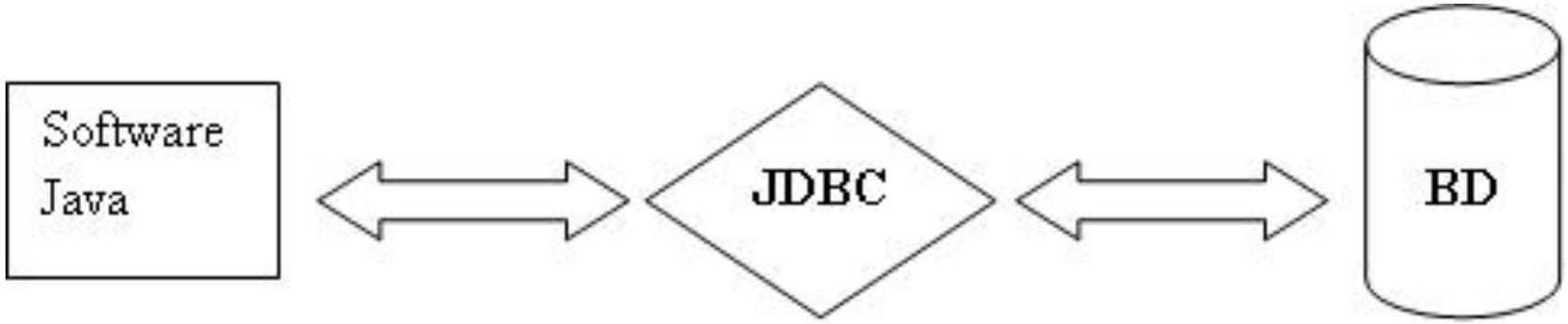
http://dl.dropbox.com/u/3025380/prog2/aula9.pdf

flavio.ceci@unisul.br

Mas como eu vou fazer o Java se comunicar com o SGBD?

JDBC

- O mecanismo existente na linguagem Java que permite tal conexão com bancos de dados corresponde a um pacote de classes chamado de JDBC (Java Data Base Connection).
- Abaixo é mostrada uma figura que tenta ilustrar como o JDBC funciona:



JDBC

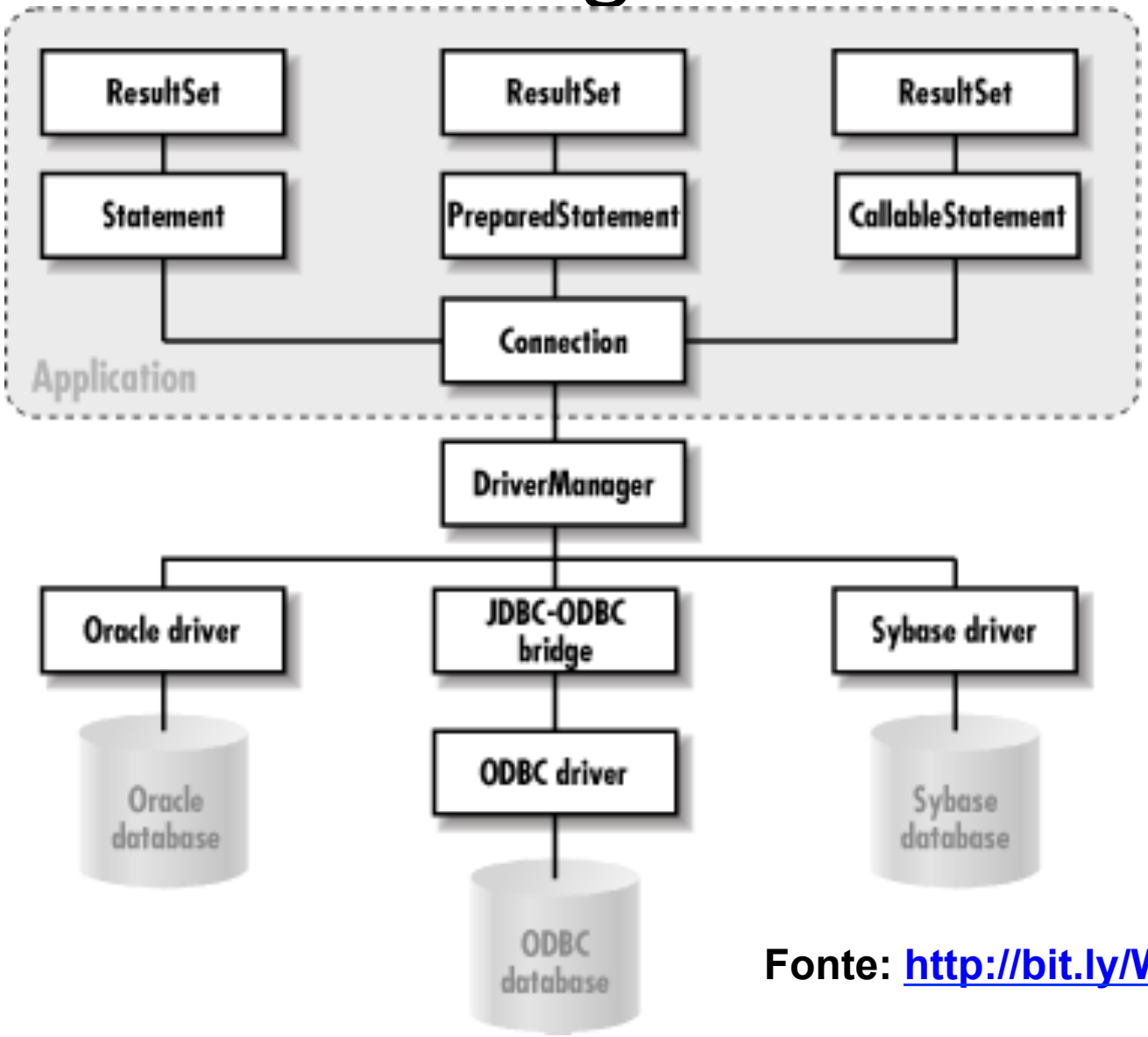
- Os comandos usados para estabelecer a comunicação com os bancos de dados, independente de qual banco seja usado:
  - Postgres;
  - Oracle;
  - Sybase;
  - MySQL;
  - entre outros
- serão sempre da mesma forma.

JDBC

- Tudo o que é necessário é que o software que vocês está desenvolvendo utilize as classes e métodos definidos no pacote JDBC (.jar);

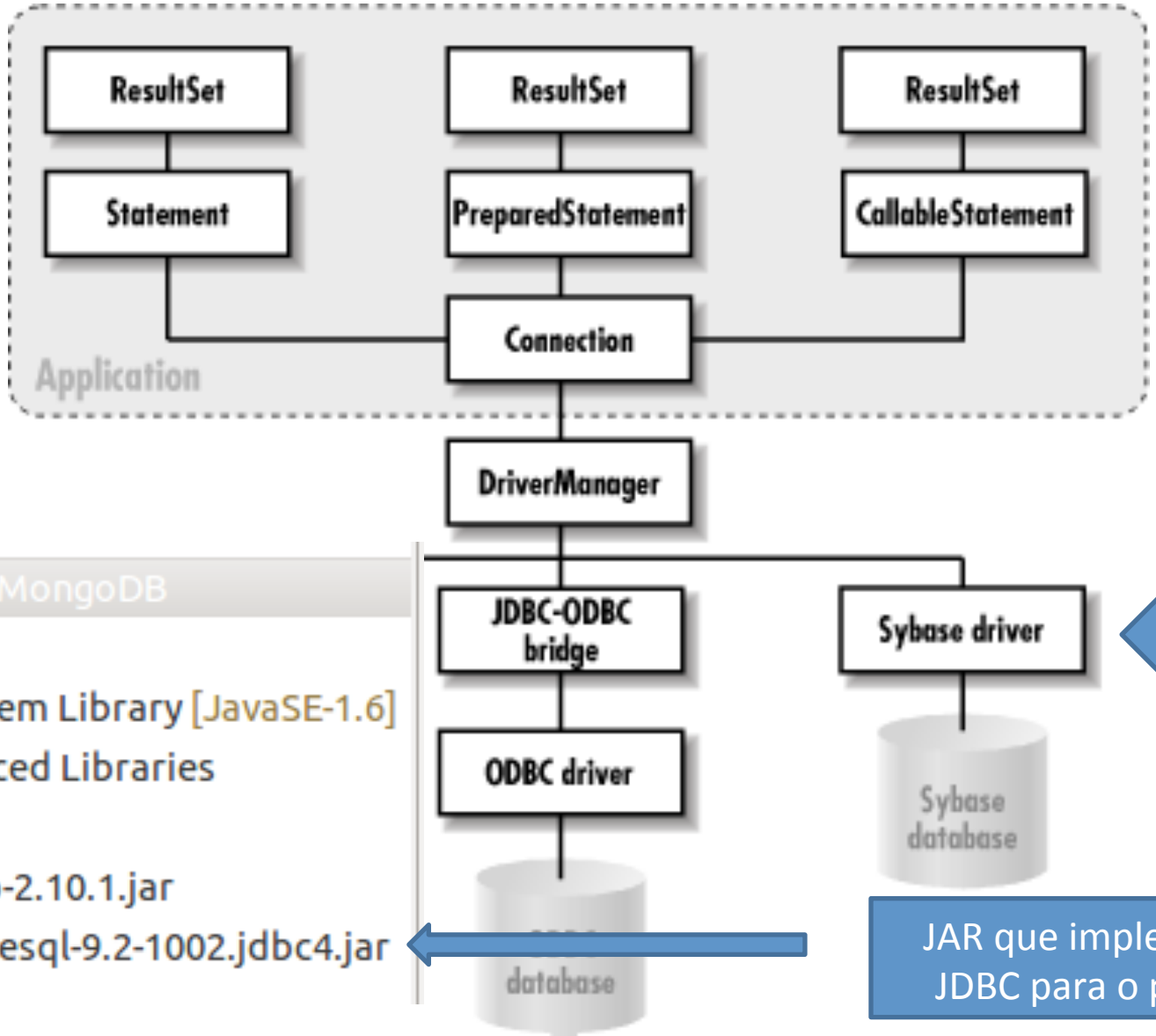
- Uma grande vantagem na utilização deste pacote, é conseguir uma independência entre o aplicativo Java e a plataforma de banco de dados utilizada.

Visão geral

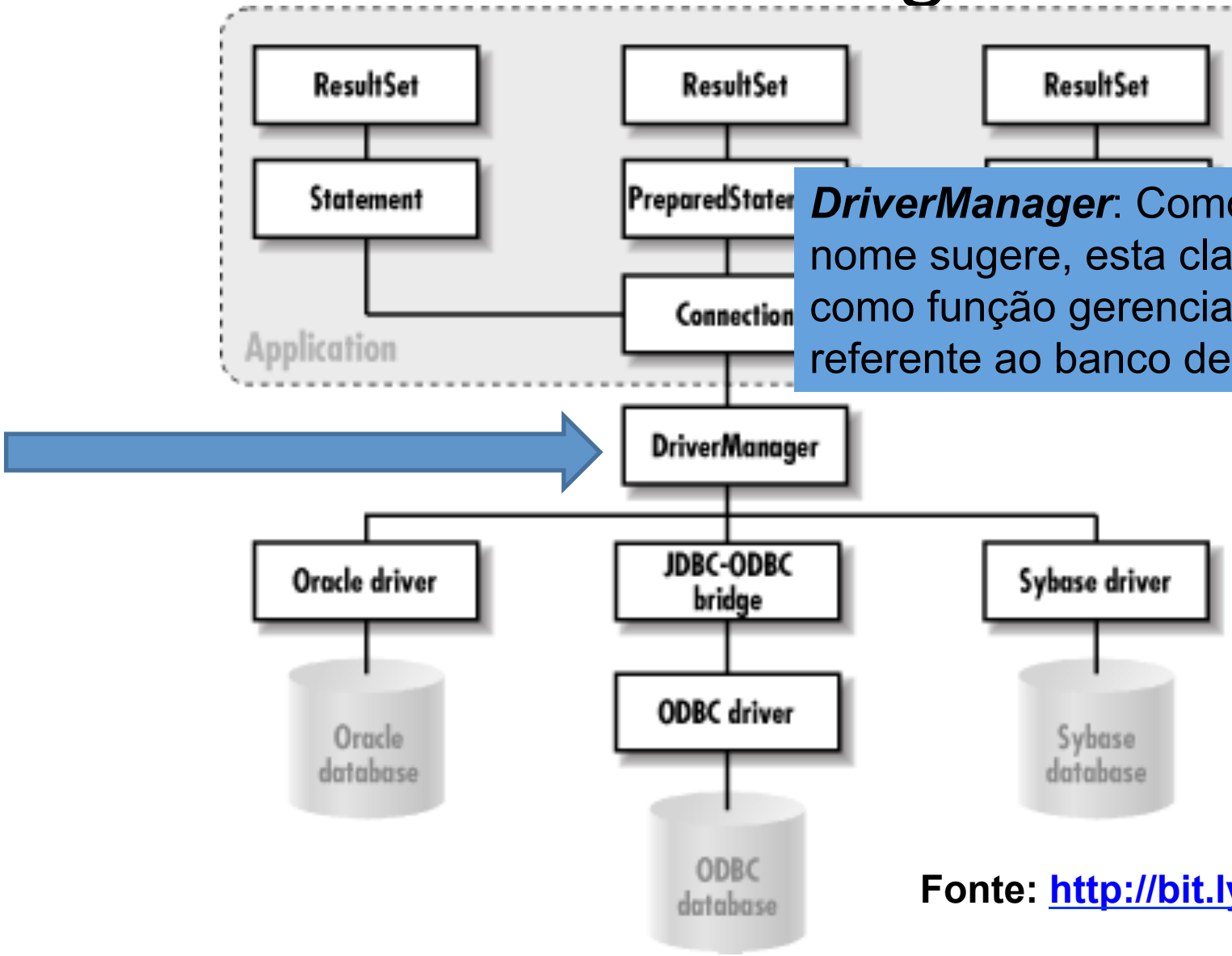


Fonte: http://bit.ly/WS8RFo

JDBC


ResultSet
ResultSet
ResultSet
Statement
PreparedStatement
CallableStatement
Connection
Application
DriverManager
MigrationMongoDB
src
JRE System Library [JavaSE-1.6]
Referenced Libraries
lib
mongo-2.10.1.jar
postgresql-9.2-1002.jdbc4.jar
JDBC-ODBC bridge
ODBC driver
database
Sybase driver
Sybase database
JAR que implementa o JDBC para o postgres

DriverManager



DriverManager: Como o próprio nome sugere, esta classe tem como função gerenciar o **driver** referente ao banco desejado.

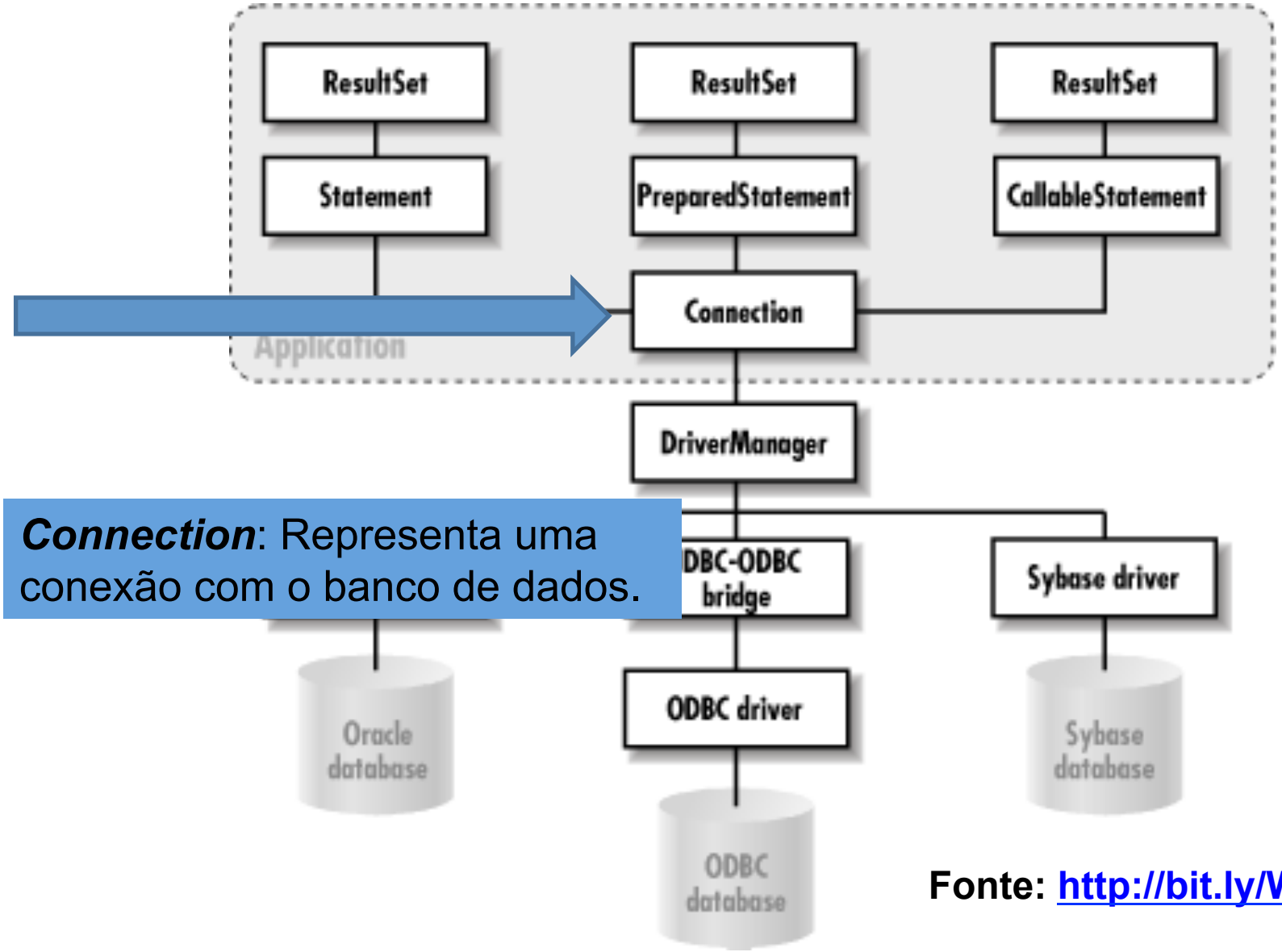
Fonte: http://bit.ly/WS8RFo

DriverManager

- Principais métodos:

static Connection	**getConnection**(String url) Attempts to establish a connection to the given database URL.
static Connection	**getConnection**(String url, Properties info) Attempts to establish a connection to the given database URL.
static Connection	**getConnection**(String url, String user, String password) Attempts to establish a connection to the given database URL.

Connection



Connection: Representa uma conexão com o banco de dados.

Fonte: http://bit.ly/WS8RFo

Exemplos da criação de conexão

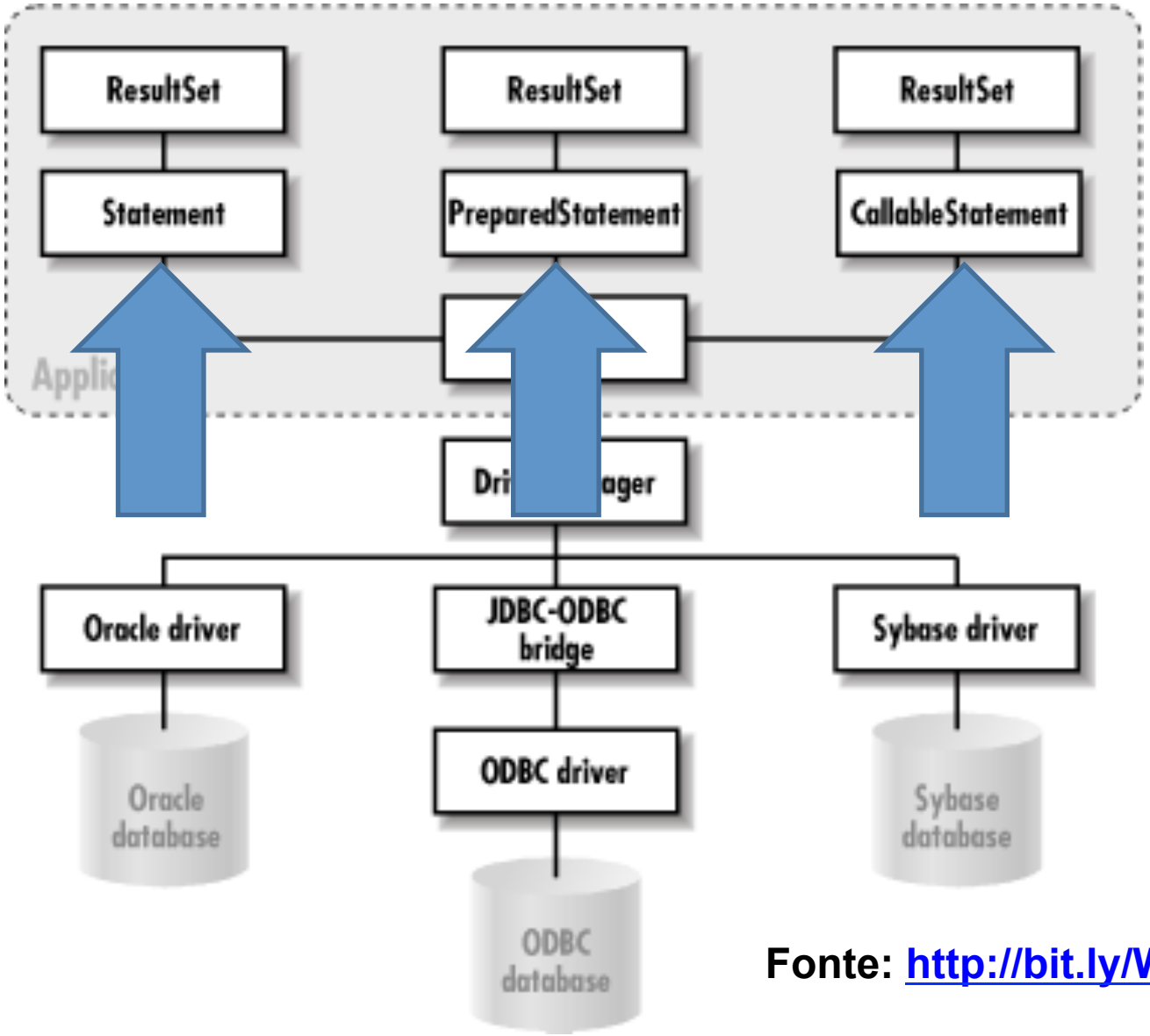
```java
import java.sql.Connection;
import java.sql.DriverManager;

public class DatabaseService {

    public static Connection getConnPostgres() {
        Connection conn = null;

        try {
            Class.forName("org.postgresql.Driver");
            conn = DriverManager.getConnection("jdbc:postgresql://192.168.0.247:5438/database",
                    "user", "password");
        } catch (Exception e) {
            System.err.println(e);
        }

        return conn;
    }
}
```

Exemplo do uso da conexão

```
public void metodoBD() {
    Connection conn = null;

    try {
        conn = DatabaseService.getConnPostgres();

        //TODO fazer a operação desejada

    } catch (Exception e) {
        System.err.println(e);

    } finally {
        if(conn != null) {
            try {
                conn.close();
            } catch (Exception e) {}
        }
    }
}
```

O método abriu recurso? Então depois do seu uso deve-se fechá-lo

Statement's



Fonte: http://bit.ly/WS8RFo

Statement's

- Mas quando usar cada uma das interfaces?
  - **Statement**: Útil quando você estiver usando instruções SQL estáticas em tempo de execução. Esta interface não aceita parâmetros.
  - **PreparedStatement**: Esta interface aceita a inserção de parâmetros em tempo de execução.
  - **CallableStatement**: Deve-se utiliza esta interface quando existe a necessidade de executar uma StoredProcedure.

Statement

```
public void removerClientes() {
    Connection conn = null;
    Statement st = null;

    try {
        conn = DatabaseService.getConnPostgres();
        st = conn.createStatement();
        st.execute("DELETE FROM CLIENTE");

    } catch (Exception e) {
        System.err.println(e);
    } finally {
        try {
            if(st != null) {
                st.close();
            }
            if(conn != null) {
                conn.close();
            }
        }catch (Exception e) {}
    }
}
```

Statement


```
public void removeClienteParametro(int idCliente) {
    Connection conn = null;
    Statement st = null;

    try {
        conn = DatabaseService.getConnPostgres();
        st = conn.createStatement();
        st.execute("DELETE FROM CLIENTE WHERE id_cliente = "+idCliente);

    } catch (Exception e) {
        System.err.println(e);
    } finally {
        try {
            if(st != null) {
                st.close();
            }
            if(conn != null) {
                conn.close();
            }
        }catch (Exception e) {}
    }
}
```

Permite a injeção de SQL

Statement – Injeção de SQL

```
public void removeClienteParametro(String nomeCliente) {
    Connection conn = null;
    Statement st = null;

    try {
        conn = DatabaseService.getConnPostgres();
        st = conn.createStatement();
        st.execute("DELETE FROM CLIENTE WHERE nome_cliente = "+nomeCliente);

    } catch (Exception e) {
        System.err.println(e);
    } finally {
        try {
            if(st != null) {
                st.close();
            }
            if(conn != null) {
                conn.close();
            }
        }catch (Exception e) {}
    }
}
```

Exemplo:

nomeCliente = “ 'nome' OR id_cliente > 0”

PreparedStatement

```
public void removeClienteParametroPS(int idCliente) {
    Connection conn = null;
    PreparedStatement ps = null;

    try {
        conn = DatabaseService.getConnPostgres();
        ps = conn.prepareStatement("DELETE FROM CLIENTE WHERE id_cliente = ?");
        ps.setInt(1, idCliente);

        ps.execute();
    } catch (Exception e) {
        System.err.println(e);
    } finally {
        try {
            if(ps != null) {
                ps.close();
            }
            if(conn != null) {
                conn.close();
            }
        }catch (Exception e) {}
    }
}
```

PreparedStatement

- Os dados passados como parâmetro são validade e tratados antes da submissão da instrução para o banco de dados;

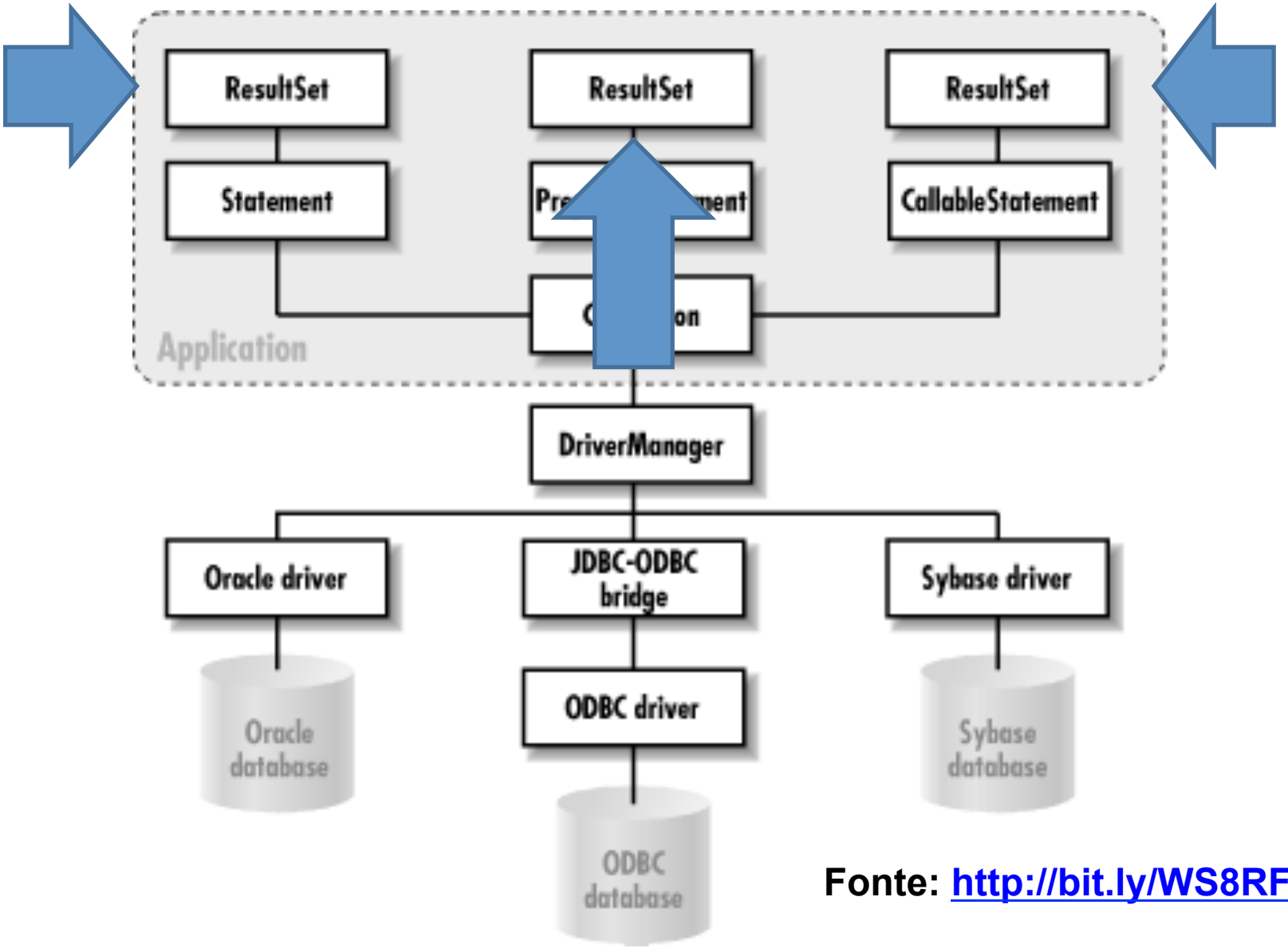
- Converte o tipo do dado para o mais próximo do SGBD específico;

- Não permite a injeção de SQL.

CallableStatement

```
public void executarStoredProcedure() {
    Connection conn = null;
    CallableStatement cs = null;

    try {
        conn = DatabaseService.getConnPostgres();
        conn.setAutoCommit(false);

        cs = conn.prepareCall("{ ? = call upper( ? ) }");
        cs.registerOutParameter(1, Types.VARCHAR);
        cs.setString(2, "lowercase to uppercase");
        cs.execute();

        String upperCased = cs.getString(1);
        System.out.println(upperCased);

    } catch (Exception e) {
        System.err.println(e);
    } finally {
        try {
            if(cs != null) {
                cs.close();
            }
            if(conn != null) {
                conn.close();
            }
        }catch (Exception e) {}
    }
}
```

ResultSet



Fonte: http://bit.ly/WS8RFo

ResultSet

```
public void listarCliente(String estado) {
    Connection conn = null;
    PreparedStatement ps = null;
    ResultSet rs = null;

    try {
        conn = DatabaseService.getConnPostgres();
        ps = conn.prepareStatement("SELECT id_cliente, nome_cliente FROM CLIENTE WHERE estado = ?");
        ps.setString(1, estado);

        rs = ps.executeQuery();
        while(rs.next()) {
            int idCliente = rs.getInt("id_cliente");
            String nomeCliente = rs.getString(2);

            System.out.println("id: "+idCliente+" - nome: "+nomeCliente);
        }
    } catch (Exception e) {
        System.err.println(e);
    } finally {
        try {
            if(rs != null) {
                rs.close();
            }
            if(ps != null) {
                ps.close();
            }
            if(conn != null) {
                conn.close();
            }
        }catch (Exception e) {}
    }
}
```

Continuando o Trabalho Integrador...

Trabalho Integrador

- Deve-se construir um modelo de dados (tabelas de um banco de dados) para suportar os requisitos do sistema da biblioteca universitária;

  http://dl.dropbox.com/u/3025380/prog2/aula6.pdf

- Após a criação do modelo é necessário construir uma camada para persistir e recuperar as informação a partir de um banco de dados.