Universidade Presbiteriana Mackenzie

Acesso ao SGBD via JDBC

Prof. Fabio Kawaoka Takase

Faculdade de Computação e Informática

Objetivo

- Revisão sobre sistemas web com Java e Banco de Dados (JDBC).
- Apresentar de forma estruturada a API do Java para acesso a banco de dados (JDBC).

Contexto

Torre de Babel

Dezenas de produtos de banco de dados...

- *... open-source.*
- *... fácil utilização.*
- *... alta peformance.*

cada um com sua própria linguagem/protocolo de comunicação...

Tentativas de padronização

SQL (Structured Query Language)

- *tentativa bem sucedida.*
- *simplificou o acesso ao banco de dados.*
- *padrão bem aceito (SQL2 ou SQL/92).*

Filosofia Java

A Alma do Java ...

Escrever uma vez, compilar uma vez e executar em qualquer lugar.

Filosofia Java para Persistência

A Alma *Persistente* do Java ...

Escrever uma vez, compilar uma vez, executar em qualquer lugar e persistir em qualquer SGBD.

JDBC – Oque é?

JDBC = API (Application Programming Interface) única para acesso ao BD.

API: Interface unificada para acesso aos diferentes tipos de SGBD.

Três objetivos básicos em seu projeto:

1. *API que permitisse o acesso ao BD através do SQL.*
2. *API simples.*
3. *Pragmatismo - API com características semelhantes a propostas já bem sucedidas.*

Acesso ao BD através do SQL

- *Tirar proveito de uma padronização bem sucedida.*
- *Diminuir o gap entre o modelo de execução do Java e do SQL (retorno de objetos e lançamento de exceções na ocorrência de erros).*

Simplicidade

- *Esforço constante em manter a API simples.*
- *Para tarefas comuns há a necessidade de conhecer apenas 3 interfaces da API.*
- *Tarefas menos comuns também podem ser realizadas, pois a API fornece outras interfaces para usos menos comuns.*

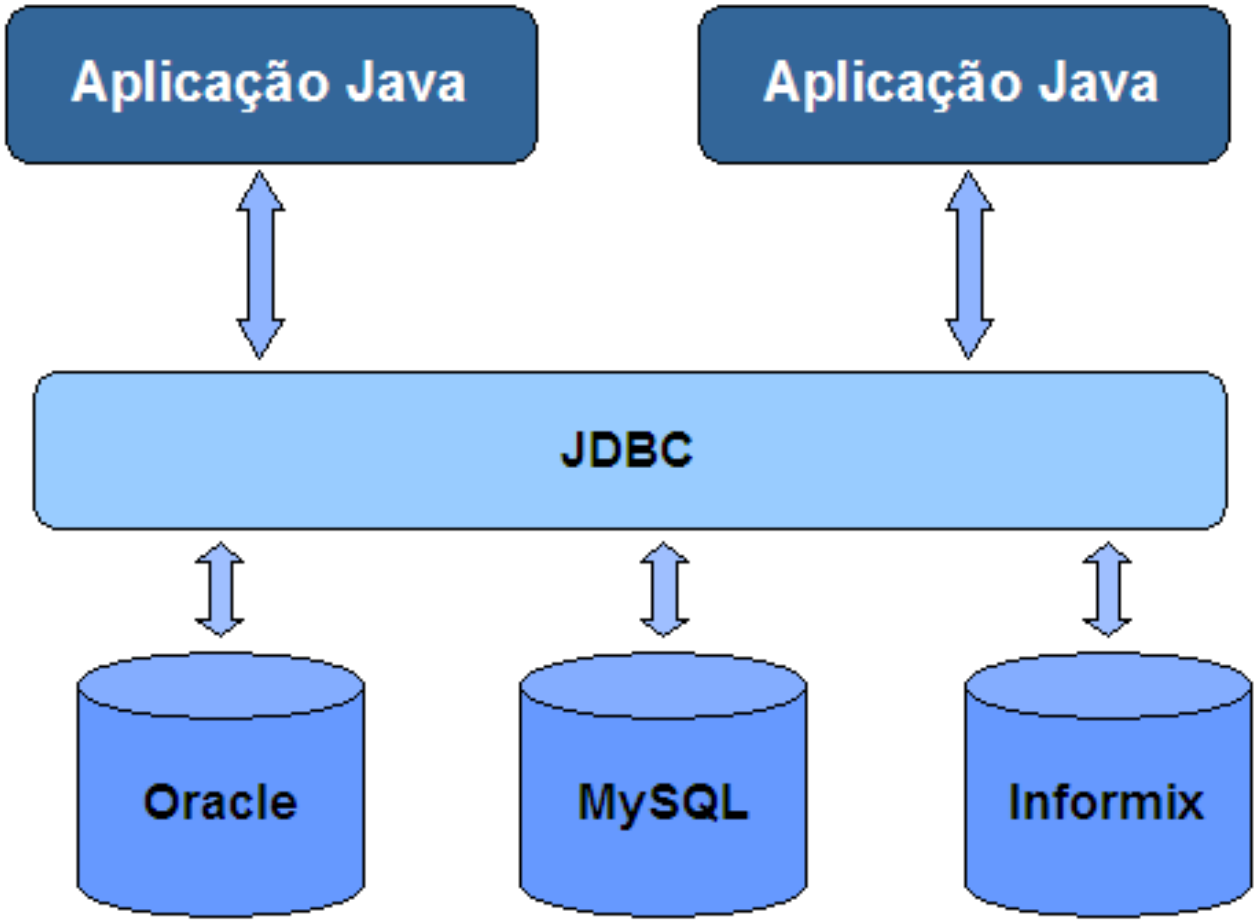
Pragmatismo

- *Tentativas de criar uma API universal de acesso ao BD não era idéia nova (motivadas pelo desconforto do uso de APIs proprietárias)*
- *ODBC (Open DataBase Connectivity) iniciativa bem sucedida de padronização no ambiente Windows.*
- *SQL CLI (Call Level Interface) da X/Open - API de programação de BD bem sucedida.*

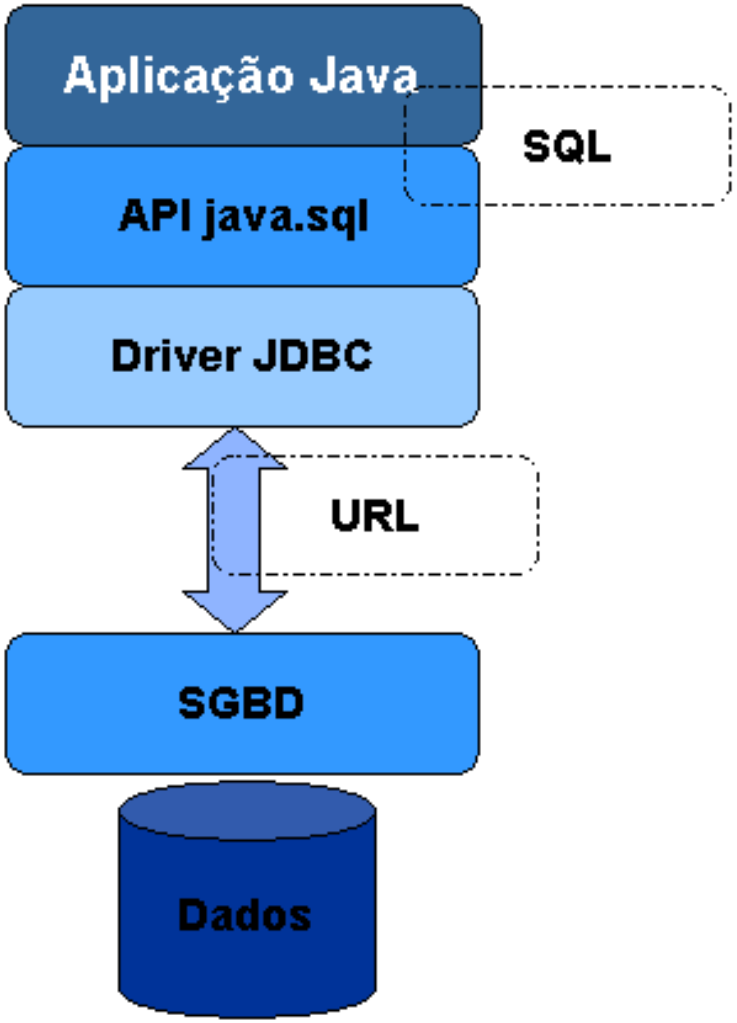
Pragmatismo

- *A API da Sun foi especificada e suas primeiras implementações foram geradas sobre APIs existentes.*
- *Primeira iniciativa foi trabalhar com a Intersolv no mapeamento de chamadas JDBC para chamadas ODBC, dando assim as aplicações Java acesso a todos os SGBDs com suporte ODBC.*
- *Com a consolidação do Java, o interesse em suportar a API JDBC passou a ser dos fornecedores dos SGBDs.*

JDBC - Estrutura Básica



JDBC - arquitetura



Tipos de Drivers JDBC

- *JDBC-ODBC bridge*
- *JDBC-API client (nativa do SGBD)*
- *Broker*
- *Java puro*

JDBC-ODBC bridge

- *Utiliza ODBC para conectar-se ao banco de dados, mapeando métodos JDBC em chamadas as funções do ODBC.*
- *Faz uso da diversidade de drivers ODBC (cascateamento torna acesso mais lento)*
- *O driver ODBC deve estar instalado e configurado na máquina.*

JDBC-API client (nativa do SGBD)

- *Chamadas realizadas diretamente as APIs do BD*
- *Requer software extra instalado na máquina cliente*

Broker

- *No cliente a chamada JDBC é traduzida para um protocolo de rede independente de banco de dados.*
- *No servidor, o protocolo de rede é traduzido para o protocolo do banco de dados.*
- *Através de um mesmo driver conectar a diferentes bancos de dados.*

Java puro

- *Solução 100% Java. (Independência de plataforma)*
- *Converte as chamadas JDBC diretamente para o protocolo (específico) do banco de dados*
- *Implementação dos fornecedores de SGDB.*

Conectando-se ao BD

1. *Localizar o pacote que contém o driver JDBC e incluí-lo no classpath de sua aplicação.*
2. *Registrar o driver JDBC junto ao DriverManager.*
3. *Construir uma URL de conexão.*
4. *Solicitar uma conexão ao DriverManager.*

Inclusão de pacote na execução

```
java -cp .;./postgresql-8.1-404.jdbc3.jar PostSqlApp

java -cp .;./ojdbc6.jar OracleApp

java -cp .;./ifxjdbc.jar InformixApp
```

Registro de driver

```
Class.forName("org.postgresql.Driver").newInstance();

Class.forName("oracle.jdbc.driver.OracleDriver").
   newInstance();

Class.forName("com.informix.jdbc.IfxDriver").
   newInstance();
```

URL de conexão

```
"jdbc:postgresql:[<//host>[:<5432>/]]<db>"

"jdbc:oracle:thin:@<server>[:<1521>]:<db>"

"jdbc:derby:<//host>[:<1527>/]<db>"


"jdbc:informix-sqli://<host>:<port>/<db>:INFO
  RMIXSERVER=<server>"
```

Solicitação de conexão

```
Connection c =
  DriverManager.getConnection (
  url , <user>, <passwd>) ;
```

Exemplo

```
import java.sql.*;

String DRIVER="com.informix.jdbc.IfxDriver";

String
   URL="jdbc:informix-sqli://<host>:<port>/<db>:INFORMIXSERVE
   R=<server>";

String USER="usuario";
String PASS="senha";

Connection connection = null;
Class.forName(DRIVER).newInstance();
connection = DriverManager.getConnection (URL,USER, PASS);
connection . close ( ) ;
```

Execução de Comando

```
Statement stm = connection.createStatement();

String SQL = "SELECT * FROM Alunos ";

ResultSet rs = stm.executeQuery(SQL);

SQL = "INSERT INTO Alunos VALUES ('Joao' ,8)";

stm.executeUpdate(SQL);
```

Resultado de uma consulta

```
ResultSet rs = stm.executeQuery(SQL);

while(rs.next())
{
      String nome = rs.getString("nome");
      int nota = rs.getInt("nota");
}
```

Obrigado

Prof. Fabio Kawaoka Takase
fabio.takase@mackenzie.br