Java e Bancos de Dados

Profa Andréa Schwertner Charão

DELC/CT/UFSM

Bancos de dados

- Tipos: relacionais, orientados a objetos, etc.
- Bancos de dados **relacionais** são os mais usados
  - Oracle, DB2, SQL Server, MySQL, PostreSQL, HSQLDB, JavaDB, SQLite, etc.
- Linguagem SQL: usada para manipular dados em BDs relacionais (criação, inserção, consulta, alteração, etc.)

Bancos de dados relacionais

- Compostos por tabelas
- Cada tabela possui alguns campos (atributos), cada campo de um tipo
- As linhas de uma tabela são chamadas de registros

exemplo: tabela simplificada de ACGs (Atividades Complementares de Graduação)c

ID	DESCR	CH
1	Participação na ERAD 2010	30
2	Pesquisa no LSC	60
3	Participação na ERRC 2009	30

Linguagem SQL

- Ver tutorial em http://www.w3schools.com/sql
- Exemplo:

```
CREATE TABLE acg
   (id INTEGER,
    descr VARCHAR(255),
    ch INTEGER,
    PRIMARY KEY(id))

INSERT INTO acg VALUES(1, 'Participação na ERAD 2010', 30)
INSERT INTO acg VALUES(2, 'Pesquisa no LSC', 60)
INSERT INTO acg VALUES(3, 'Participação na ERRC 2009', 30)

SELECT * FROM acg
SELECT ch FROM acg
SELECT SUM(ch) FROM acg
```

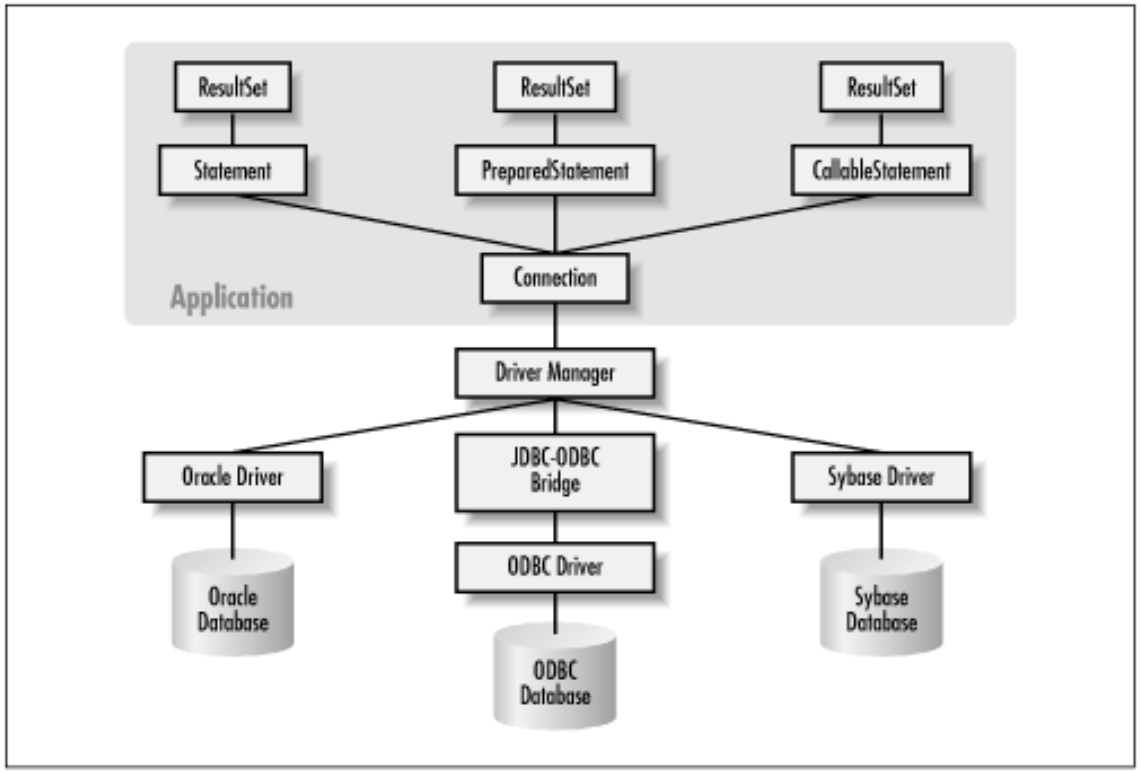
Acesso a BD em Java

- Contexto:
  - durante a execução, objetos em memória
  - BD garante persistência dos objetos
- Várias opções:
  - JDBC (Java DataBase Connectivity)
  - JPA (Java Persistence API)
  - Hibernate (mapeamento objeto-relacional)
  - etc.

JDBC (Java DataBase Connectivity)

- API comum para acesso a qualquer gerenciador de banco de dados relacional
- Driver específico (.jar) para cada gerenciador de BD
- Principal pacote: java.sql
- Classes para:
  - Estabelecer conexão com o banco
  - Criar comandos
  - Submeter comandos ao banco
  - Processar resultados dos comandos
  - Fechar conexão

JDBC (Java DataBase Connectivity)



Fonte: Java Servlet Programming. OŔeilly.

HSQLDB (HyperSQL)

- Gerenciador de BD feito em Java
- Disponível em: http://hsqldb.org
- Tem bom desempenho
- Tem interface gráfica para gerenciamento
- Funciona em vários modos:
  - servidor
  - in-process (standalone)
  - memory-only
- Usaremos modo standalone (ver http://hsqldb.org/doc/guide/ch01.html)

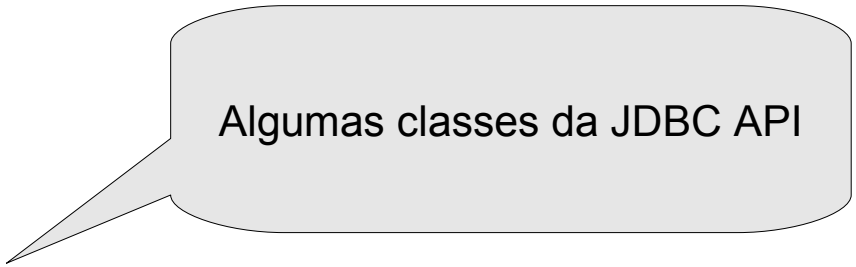
HSQLDB (HyperSQL)

```
java -cp ./hsqldb.jar
org.hsqldb.util.DatabaseManagerSwing
-url jdbc:hsqldb:file:acgdb
```

ExemploJDBC.java

```java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.SQLException;
import java.sql.Statement;

public class ExemploJDBC {

   // Constantes para conexao com HSQLSB (modo standalone)
   static final String JDBC_DRIVER = "org.hsqldb.jdbcDriver";
   static final String DB_NAME = "acgdb";
   static final String DB_URL = "jdbc:hsqldb:file:" + DB_NAME;
   static final String DB_USER = "sa";
   static final String DB_PASSWD = "";

   public static void main(String[] args) throws ClassNotFoundException, SQLException {
      // Carrega o driver JDBC do HSQLDB.
      // O driver hsqldb.jar deve estar no classpath.
      Class.forName(JDBC_DRIVER);

      // Conecta o programa com o banco de dados.
      // Os arquivos do banco sao criados se nao existirem.
      Connection c = DriverManager.getConnection(DB_URL,
                                                 DB_USER,
                                                 DB_PASSWD);
```

ExemploJDBC.java




```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.SQLException;
import java.sql.Statement;

public class ExemploJDBC {

   // Constantes para conexao com HSQLSB (modo standalone)
   static final String JDBC_DRIVER = "org.hsqldb.jdbcDriver";
   static final String DB_NAME = "acgdb";
   static final String DB_URL = "jdbc:hsqldb:file:" + DB_NAME;
   static final String DB_USER = "sa";
   static final String DB_PASSWD = "";

   public static void main(String[] args) throws ClassNotFoundException, SQLException {
      // Carrega o driver JDBC do HSQLDB.
      // O driver hsqldb.jar deve estar no classpath.
      Class.forName(JDBC_DRIVER);

      // Conecta o programa com o banco de dados.
      // Os arquivos do banco sao criados se nao existirem.
      Connection c = DriverManager.getConnection(DB_URL,
                                                 DB_USER,
                                                 DB_PASSWD);
```

ExemploJDBC.java

```java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.SQLException;
import java.sql.Statement;

public class ExemploJDBC {

   // Constantes para conexao com HSQLSB (modo standalone)
   static final String JDBC_DRIVER = "org.hsqldb.jdbcDriver";
   static final String DB_NAME = "acgdb";
   static final String DB_URL = "jdbc:hsqldb:file:" + DB_NAME;
   static final String DB_USER = "sa";
   static final String DB_PASSWD = "";

   public static void main(String[] args) throws ClassNotFoundException, SQLException {
      // Carrega o driver JDBC do HSQLDB.
      // O driver hsqldb.jar deve estar no classpath.
      Class.forName(JDBC_DRIVER);

      // Conecta o programa com o banco de dados.
      // Os arquivos do banco sao criados se nao existirem.
      Connection c = DriverManager.getConnection(DB_URL,
                                                 DB_USER,
                                                 DB_PASSWD);
```

Constantes específicas do banco

Carregar driver JDBC

```java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.SQLException;
import java.sql.Statement;

public class ExemploJDBC {

   // Constantes para conexao com HSQLSB (modo standalone)
   static final String JDBC_DRIVER = "org.hsqldb.jdbcDriver";
   static final String DB_NAME = "acgdb";
   static final String DB_URL = "jdbc:hsqldb:file:" + DB_NAME;
   static final String DB_USER = "sa";
   static final String DB_PASSWD = "";

   public static void main(String[] args) throws ClassNotFoundException, SQLException {
      // Carrega o driver JDBC do HSQLDB.
      // O driver hsqldb.jar deve estar no classpath.
      Class.forName(JDBC_DRIVER);

      // Conecta o programa com o banco de dados.
      // Os arquivos do banco sao criados se nao existirem.
      Connection c = DriverManager.getConnection(DB_URL,
                                                 DB_USER,
                                                 DB_PASSWD);
```

org.hsqldb.jdbcDriver é uma classe em hsqldb.jar

Estabelecer conexão

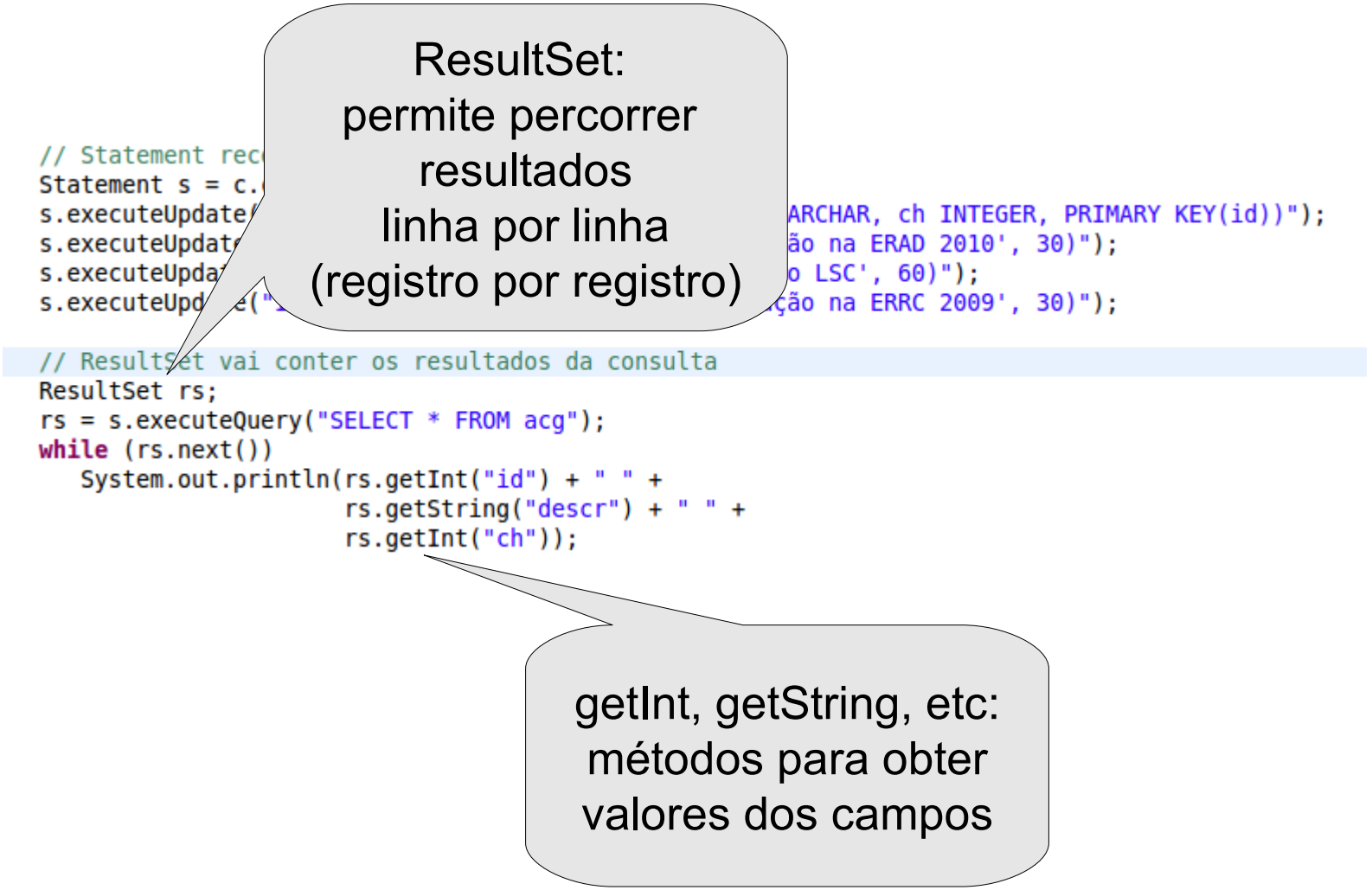
```java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.SQLException;
import java.sql.Statement;

public class ExemploJDBC {

   // Constantes para conexao com HSQLSB (modo standalone)
   static final String JDBC_DRIVER = "org.hsqldb.jdbcDriver";
   static final String DB_NAME = "acgdb";
   static final String DB_URL = "jdbc:hsqldb:file:" + DB_NAME;
   static final String DB_USER = "sa";
   static final String DB_PASSWD = "";

   public static void main(String[] args) throws ClassNotFoundException, SQLException {
      // Carrega o driver JDBC do HSQLDB.
      // O driver hsqldb.jar deve estar no classpath.
      Class.forName(JDBC_DRIVER);

      // Conecta o programa com o banco de dados.
      // Os arquivos do banco sao criados se nao existirem.
      Connection c = DriverManager.getConnection(DB_URL,
                                                 DB_USER,
                                                 DB_PASSWD);
```

Criar e submeter comandos SQL

```
// Statement recebe comandos SQL e os executa
Statement s = c.createStatement();
s.executeUpdate("CREATE TABLE acg (id INTEGER                    KEY(id))");
s.executeUpdate("INSERT INTO acg VALUES(1,
s.executeUpdate("INSERT INTO acg VALUES(2
s.executeUpdate("INSERT INTO acg VALUES

// ResultSet vai conter os resultados da consulta
ResultSet rs;
rs = s.executeQuery("SELECT * FROM acg");
while (rs.next())
   System.out.println(rs.getInt("id") + " " +
                      rs.getString("descr") + " " +
                      rs.getInt("ch"));
```

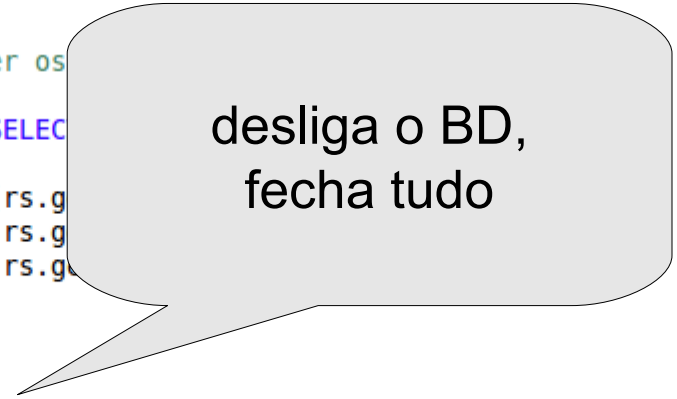
executeQuery:
comandos de consulta
(SELECT)

Processar resultados



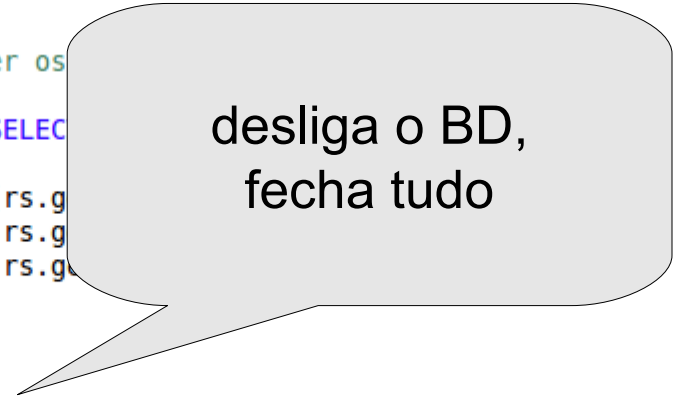
Fechar conexão

```
// ResultSet vai conter os
ResultSet rs;
rs = s.executeQuery("SELEC
while (rs.next())
   System.out.println(rs.g
                      rs.g
                      rs.g

s.execute("SHUTDOWN");
s.close();
c.close();
```



Fechar conexão

```
// ResultSet vai conter os
ResultSet rs;
rs = s.executeQuery("SELEC
while (rs.next())
   System.out.println(rs.g
                      rs.g
                      rs.g

s.execute("SHUTDOWN");
s.close();
c.close();
```



Problemas de ExemploJDBC.java

- Execução de comandos SQL pode gerar falhas de segurança (*SQL injection*)
- Comandos SQL não são reutilizados
- Não usa orientação a objetos para manipular ACGs
- Mistura código de acesso a dados com código de interface com o usuário (percorre cada registro e mostra na saída padrão)

Classe PreparedStatement

- Permite reutilizar comandos SQL
- É mais eficiente e seguro
  - consultas pré-compiladas
  - evita SQL injection

```
PreparedStatement p = c.prepareStatement("INSERT INTO acg VALUES (?, ?, ?)");
p.setInt(1, 1);
p.setString(2, "Participação na ERAD 2010");
p.setInt(3, 30);
p.execute();

//s.executeUpdate("INSERT INTO acg VALUES(1, 'Participação na ERAD 2010', 30)");
//s.executeUpdate("INSERT INTO acg VALUES(2, 'Pesquisa no LSC', 60)");
//s.executeUpdate("INSERT INTO acg VALUES(3, 'Participação na ERRC 2009', 30)");
```