

# Java JDBC

Aula 05

Prof. Dr. Rodrigo Plotze

# **STRUCTURED QUERY LANGUAGE**

## **SQL**

# Structured Query Language (SQL)

- Um banco de dados é uma coleção organizada de dados.
- Um sistema de gerenciamento de bancos de dados (database management system – DBMS) fornece mecanismos para armazenar, organizar, recuperar e modificar dados para muitos usuários.
- SQL é a linguagem padrão internacional utilizada quase universalmente com bancos de dados relacionais para realizar consultas e para manipular dados.

# Structured Query Language (SQL)

- Sistemas de gerenciamento de banco de dados relacional (SGBD) populares
  - Microsoft SQL Server
  - Oracle
  - Sybase
  - IBM DB2
  - Informix
  - PostgreSQL
  - MySQL

# Structured Query Language (SQL)

- Structured Query Language (SQL)
  - DDL – Data Definition Language
    - Comandos:
      - Create
      - Alter
      - Drop
    - Criação dos objetos
      - Banco de dados, tabelas, índices, etc

# Structured Query Language (SQL)

- Structured Query Language (SQL)
  - DML – Data Manipulation Language
    - Comandos
      - Insert
      - Update
      - Delete
      - Select
    - Manipula os registros dentro de um banco de dados

# Structured Query Language (SQL)

- Structured Query Language (SQL)
  - DCL – Data Control Language
    - Comandos
      - Grant
      - Revoke
    - Controle o acesso dos usuários

# Structured Query Language (SQL)

- Sistema de Gerenciamento de Banco de Dados (SGBD)
  - Sistema computacional para gerenciamento de uma base de dados.
- Banco de Dados Relacional
  - Representação lógica de dados que permite o acesso sem considerar sua estrutura física.
  - Os dados são armazenados em tabelas.



# Structured Query Language (SQL)

- Os programas Java comunicam-se com bancos de dados e manipulam seus dados utilizando a Java Database Connectivity (JDBC™) API.
- Um driver JDBC permite aos aplicativos Java conectar-se a um banco de dados em um DBMS particular e permite a você manipular esse banco de dados utilizando o JDBC API.



## **Observação de engenharia de software 28.1**

*Usar a JDBC API permite que desenvolvedores mudem o DBMS subjacente sem modificar o código Java que acessa o banco de dados.*

# Structured Query Language (SQL)

- Um banco de dados relacional é uma representação lógica de dados que permite que os dados sejam acessados sem considerar sua estrutura física.
- Um banco de dados relacional armazena dados em tabelas.
- As tabelas são compostas de linhas e as linhas são compostas de colunas nas quais os valores são armazenados.
- Chave primária — uma coluna (ou grupo de colunas) com um valor único que não pode ser duplicado em outras linhas.

# Structured Query Language (SQL)

- Tabelas
  - Compostas por linhas e colunas

	Number	Name	Department	Salary	Location
	23603	Jones	413	1100	New Jersey
	24568	Kerwin	413	2000	New Jersey
Linha {	34589	Larson	642	1800	Los Angeles
	35761	Myers	611	1400	Orlando
	47132	Neumann	413	9000	New Jersey
	78321	Stephens	611	8500	Orlando
	Chave primária		Coluna		

# Structured Query Language (SQL)

- Tabelas
  - Exemplo de armazenamento de dados sobre autores.

AuthorID	FirstName	LastName
1	Harvey	Deitel
2	Paul	Deitel
3	Andrew	Goldberg
4	David	Choffnes

# Structured Query Language (SQL)

Palavras-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 de 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.

# Structured Query Language (SQL)

- SQL: *Select*

- Uma consulta de SQL “seleciona” linhas e colunas de uma ou mais tabelas em um banco de dados.
- O formato básico de uma consulta SELECT é
  - SELECT \* FROM nomeDaTabela
- em que o asterisco (\*) indica que todas as colunas da tabela nomeDaTabela devem ser recuperadas.

# Structured Query Language (SQL)

- SQL: ***Select***

- Para recuperar todos os dados na tabela Authors, utilize
  - `SELECT * FROM Authors`
- Para recuperar somente colunas específicas de uma tabela, substitua o asterisco (\*) por uma lista dos nomes de coluna separados por vírgulas
  - `SELECT AuthorID, LastName FROM Authors`

# Structured Query Language (SQL)

- SQL: *Where*

- Na maioria dos casos, apenas linhas que atendem aos critérios de seleção são selecionadas.
- A SQL utiliza a cláusula WHERE opcional em uma consulta para especificar os critérios de seleção para a consulta.
- A forma básica de uma consulta com critérios de seleção é
  - `SELECT nomeDaColuna1, nomeDaColuna2, ... FROM nomeDaTabela WHERE critérios`



# Structured Query Language (SQL)

- SQL: *Where*

- Para selecionar as colunas Title, EditionNumber e Copyright da tabela Titles para a qual a data do Copyright é maior que 2005, use a consulta
  - `SELECT Title, EditionNumber, Copyright  
FROM Titles  
WHERE Copyright > '2005'`
- As strings em SQL são delimitadas por aspas simples (') em vez de aspas duplas (").
- Os critérios da cláusula WHERE podem conter os operadores <, >, <=, >=, =, <> e LIKE.

# Structured Query Language (SQL)

- SQL: ***Order By***

- As linhas no resultado de uma consulta podem ser classificadas em ordem crescente ou decrescente utilizando a cláusula ORDER BY opcional.
- O formato básico de uma consulta com uma cláusula ORDER BY é
  - `SELECT nomeDaColuna1, nomeDaColuna2, ... FROM nomeDaTabela  
ORDER BY coluna ASC`  
`SELECT nomeDaColuna1, nomeDaColuna2, ... FROM nomeDaTabela  
ORDER BY coluna DESC`
- ASC especifica a ordem ascendente (do mais baixo para o mais alto).
- DESC especifica ordem descendente (do mais alto para o mais baixo).
- coluna especifica a coluna em que a classificação é baseada.

# Structured Query Language (SQL)

- SQL: INNER JOIN

- Os projetistas de banco de dados costumam dividir os dados relacionados em tabelas separadas para assegurar que um banco de dados não armazene dados de maneira redundante.
- Frequentemente é necessário mesclar dados de múltiplas tabelas em um único resultado.
- Isso é chamado de junção, ou join, de tabelas.

# Structured Query Language (SQL)

- SQL: INNER JOIN

- Uma INNER JOIN mescla linhas de duas tabelas correspondendo valores em colunas que são comuns às tabelas.
  - `SELECT nomeDaColuna1, nomeDaColuna2, ...`  
`FROM tabela1`  
`INNER JOIN tabela2`  
`ON tabela1.nomeDaColuna = tabela2.nomeDaColuna`
- A cláusula ON especifica as colunas de cada tabela que são comparadas para determinar as linhas que são mescladas.

# Structured Query Language (SQL)

- SQL: INNER JOIN

```
SELECT FirstName, LastName, ISBN  
FROM Authors  
INNER JOIN AuthorISBN  
    ON Authors.AuthorID = AuthorISBN.AuthorID  
ORDER BY LastName, FirstName
```

# Structured Query Language (SQL)

## ■ SQL: INSERT

- A instrução INSERT insere uma linha em uma tabela.
  - `INSERT INTO nomeDaTabela (`  
    `nomeDaColuna1, nomeDaColuna2, ...,`  
    `nomeDaColunaN)`  
        `VALUES ( valor1, valor2, ..., valorN )`
- onde nomeDaTabela é a tabela na qual inserir a linha.
- nomeDaTabela é seguido por uma lista separada por vírgulas de nomes de coluna entre parênteses
- não necessária se a operação INSERT especificar um valor para cada coluna da tabela na ordem correta

# Structured Query Language (SQL)

- SQL: INSERT

```
INSERT INTO Authors (  
    FirstName, LastName )  
    VALUES ( 'Sue', 'Smith' )
```

# Structured Query Language (SQL)

- SQL: UPDATE

- Uma instrução UPDATE modifica os dados em uma tabela.

- UPDATE nomeDaTabela  
          SET nomeDaColuna1 = valor1,  
          nomeDaColuna2 = valor2, ..., nomeDaColunaN =  
          valorN  
          Critérios WHERE

- onde nomeDaTabela é a tabela a atualizar.
    - nomeDaTabela é seguido pela palavra-chave SET e uma lista separada por vírgulas de pares nome/valor de coluna no formato nomeDaColuna = valor.
    - A cláusula WHERE opcional fornece critérios que determinam quais linhas atualizar.



# Structured Query Language (SQL)

- SQL: UPDATE

```
UPDATE Authors  
  SET LastName = 'Jones'  
 WHERE LastName = 'Smith'  
    AND FirstName = 'Sue'
```

# Structured Query Language (SQL)

- SQL: DELETE

- Uma instrução DELETE de SQL remove as linhas de uma tabela.
  - `DELETE FROM nomeDaTabela WHERE critérios`
- onde nomeDaTabela é a tabela a partir da qual excluir.
- A cláusula WHERE opcional especifica os critérios utilizados para determinar quais linhas excluir.
  - Se essa cláusula for omitida, todas as linhas da tabela serão excluídas.

# Structured Query Language (SQL)

- SQL: DELETE

```
DELETE FROM Authors  
    WHERE LastName = 'Jones'  
    AND FirstName = 'Sue'
```

# Atividade Prática

# Atividade Prática

- Criar um novo database denominado ***universidade***.
- Criar três tabelas:
  - aluno: id, código, nome, email
  - disciplina: id, nome
  - aluno\_disciplina: id, aluno\_id, disciplina\_id, ano, semestre
- Demonstrar o uso dos comandos: insert, update, delete e select.

# Atividade Prática

Query Editor    Query History

```
1  -- CRIAÇÃO de um NOVO banco de dados
2  CREATE DATABASE "UNIVERSIDADE"
3      ENCODING "UTF-8"
4      CONNECTION LIMIT -1;
5
```

Database Explorer

- > LBD
- > UNIVE
- > postgr
- > Login/Gro
- > Tablespac

Query Editor    Query History

```
1  -- CRIAÇÃO de uma TABELA no database UNIVERSIDADE
2  create table tb_aluno(
3
4      id
5      codigo
6      nome
7      email
8
9      CONSTRAINT pk_aluno_id_aluno PRIMARY KEY(id)
10 );|
```

Nome da TABELA


Tipos de dados e propriedades

Definição da CHAVE PRIMÁRIA

Nomes dos CAMPOS  
OU  
Nomes das COLUNAS

Data Output    Explain    Messages    Notifications

```
1  -- CRIAÇÃO de uma TABELA no database UNIVERSIDADE
2  create table tb_aluno(
3
4      id          SERIAL,
5      codigo      INTEGER,
6      nome        VARCHAR(60) NOT NULL,
7      email       VARCHAR(45),
8
9      CONSTRAINT pk_aluno_id_aluno PRIMARY KEY(id)
10 );
```



**Os valores da coluna "id" serão gerados automaticamente por uma sequência de números inteiros.**



```




1  select * from tb_aluno;
2
3  insert into tb_aluno
4      (codigo,nome,email) values
5      (123456,'João da Silva','joao@joao.com');
6
7  insert into tb_aluno
8      (codigo,nome,email) values
9      (456113,'Ana Maria','ana@maria.com');
10

```

	id [PK] integer	codigo integer	nome character varying (60)	email character varying (45)
1	1	123456	João da Silva	joao@joao.com
2	2	456113	Ana Maria	ana@maria.com

```
1  -- CRIAÇÃO da tabela de DISCIPLINAS
2  create table tb_disciplina(
3
4      id          SERIAL,
5      nome        VARCHAR(80),
6      CONSTRAINT pk_disciplina_id PRIMARY KEY(id)
7  );
```

```
1 insert into tb_disciplina
2     (nome) values ('Laboratório de Programação I');
3
4 insert into tb_disciplina
5     (nome) values ('Laboratório de Programação II');
6
7 insert into tb_disciplina
8     (nome) values ('Laboratório de Programação III');
9
10 insert into tb_disciplina
11     (nome) values ('Laboratório de Programação IV');
12
13 select * from tb_disciplina;
```

	 id [PK] integer 	nome  character varying (80)	
1	1	Laboratório de Programação I	
2	2	Laboratório de Programação...	
3	3	Laboratório de Programação...	
4	4	Laboratório de Programação...	

Query Editor

Query History

1 `select * from tb_aluno WHERE id = 2;`

Data Output

Explain

Messages

Notifications

	id [PK] integer	codigo integer	nome character varying (60)	email character varying (45)
1	2	456113	Ana Maria	ana@maria.com

```
1 select * from tb_aluno WHERE nome = 'João da Silva';
2 |
3 select * from tb_disciplina WHERE nome LIKE 'Laboratório%';
```

O operador de % é um coringa, pois permite recuperar dados avaliando o conteúdo da coluna indicada.

	id [PK] integer	nome character varying (80)
1	1	Laboratório de Programação I
2	2	Laboratório de Programação II
3	3	Laboratório de Programação III
4	4	Laboratório de Programação IV

```

5 insert into tb_disciplina(nome) values ('Estrutura de Dados');
6 select * from tb_disciplina WHERE nome LIKE '%de%';
7




```

Data Output Explain Messages Notifications

	id [PK] integer	nome character varying (80)
1	1	Laboratório de Programação I
2	2	Laboratório de Programação II
3	3	Laboratório de Programação III
4	4	Laboratório de Programação IV
5	6	Estrutura de Dados

**Recuperar todas as disciplinas  
que tenha a palavra 'de' em qualquer  
parte do nome**

```
1  -- Ordenação ascendente (do menor para o maior)
2  select * from tb_disciplina ORDER BY nome;
3
4  -- Ordenação decendente (do maior para o menor)
5  select * from tb_disciplina ORDER BY id DESC;
6
```

	 id [PK] integer 	nome  character varying (80)	
1	6	Estrutura de Dados	
2	5	Computação Gráfica	
3	4	Laboratório de Programação IV	
4	3	Laboratório de Programação III	
5	2	Laboratório de Programação II	
6	1	Laboratório de Programação I	



```

1  -- ATUALIZAR dados da tabela
2  update tb_disciplina
3      set nome='Sistemas Distribuídos'
4      where id = 6;
5
6  select * from tb_disciplina;
7


```

Data Output

Explain

Messages

Notifications

	id [PK] integer		nome character varying (80)
1		1	Laboratório de Programação I
2		2	Laboratório de Programação II
3		3	Laboratório de Programação III
4		4	Laboratório de Programação IV
5		5	Computação Gráfica
6		6	Sistemas Distribuídos



```
1  -- ATUALIZAR dados da tabela
2  update tb_aluno
3      set nome = 'José Antônio', email='jose@jose.com'
4      where id = 2;
5
6  select * from tb_aluno;
7
```

Data Output Explain Messages Notifications





	id [PK] integer	codigo integer	nome character varying (60)	email character varying (45)
1	1	123456	João da Silva	joao@joao.com
2	2	456113	José Antônio	jose@jose.com

```

1  -- APAGAR dados da TABELA
2  delete from tb_aluno
3      where id = 2;
4
5  select * from tb_aluno;
6

```

**Data Output**   Explain   Messages   Notifications

	id [PK] integer 	codigo integer 	nome character varying (60) 	email character varying (45) 
1	1	123456	João da Silva	joao@joao.com

```
1  create table tb_aluno_disciplina(  
2  
3      id                serial,  
4      aluno_id         int,  
5      disciplina_id    int,  
6      fg_ativo         int,  
7  
8      constraint pk_aluno_disciplina_id PRIMARY KEY (id),  
9      constraint fk_aluno_id FOREIGN KEY(aluno_id) REFERENCES tb_aluno(id),  
10     constraint fk_disciplina_id FOREIGN KEY(disciplina_id) REFERENCES tb_disciplina(id)  
11  
12 );
```

```
select ad.id, ad.aluno_id, ad.disciplina_id, d.nome, ad.fg_ativo
from tb_aluno_disciplina as ad
inner join tb_disciplina as d
on ad.disciplina_id = d.id;
```

```
select ad.id, ad.aluno_id, a.nome, ad.disciplina_id, d.nome, ad.fg_ativo
from tb_aluno_disciplina as ad
inner join tb_aluno as a
on ad.aluno_id = a.id
inner join tb_disciplina as d
on ad.disciplina_id = d.id;
```

# **JAVA DATABASE CONNECTIVITY**

## **JDBC**

# Java Database Connectivity (JDBC)

- JDBC API

- É o padrão para a conectividade entre o banco de dados e a linguagem de programação Java.
- Permite a conexão com uma vasta gama de bases de dados.
- Utiliza chamadas com a linguagem SQL para acesso ao banco de dados.
- Independente de plataforma e sistema operacional.
  - *Write Once, Run Anywhere*

# Java Database Connectivity (JDBC)

- JDBC API
  - Pacotes Principais
    - java.sql    □ pacote principal
    - javax.sql   □ extensões (RowSet)
  - Documentação
    - [oracle.com/technetwork/java/javase/jdbc/index.html](http://oracle.com/technetwork/java/javase/jdbc/index.html)

# Java Database Connectivity (JDBC)

- JDBC API
  - Permite a realização de três tarefas essenciais:
    - Estabelecer uma conexão com o banco de dados
    - Enviar instruções SQL
    - Processar os resultados



# Java Database Connectivity (JDBC)

- JDBC Arquitetura

- *Direct-to-Database Pure Java Driver*

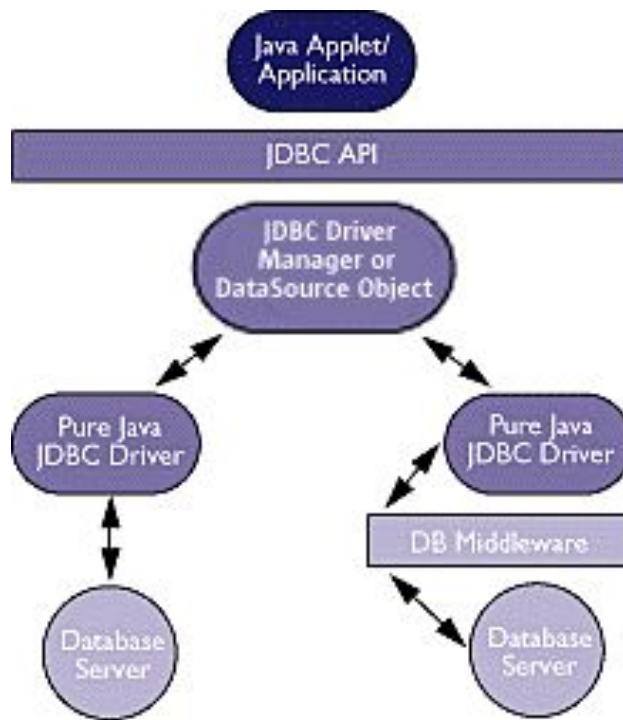
- Converte as chamadas JDBC diretamente para o protocolo de rede utilizado pelo SGBD

- *Pure Java Driver for Database Middleware*

- Traduz as chamadas JDBC para o protocolo utilizado pelo middleware do fabricante do SGBD, que então traduz para o protocolo de rede utilizado pelo SGBD

# Java Database Connectivity (JDBC)

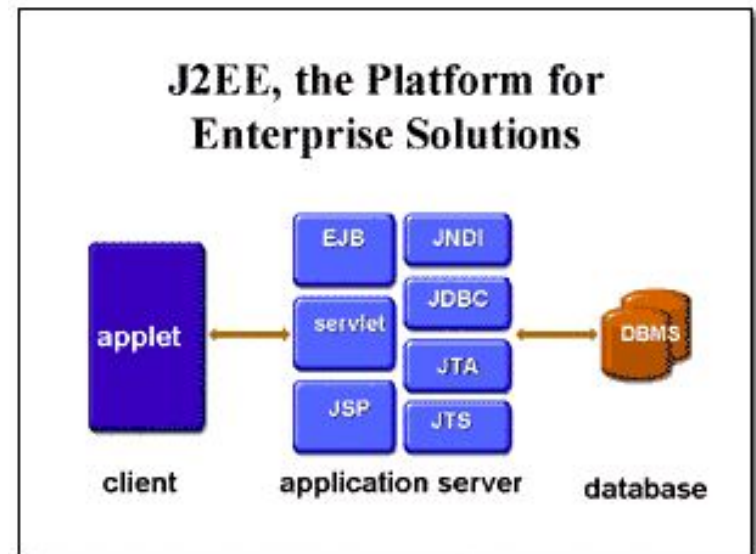
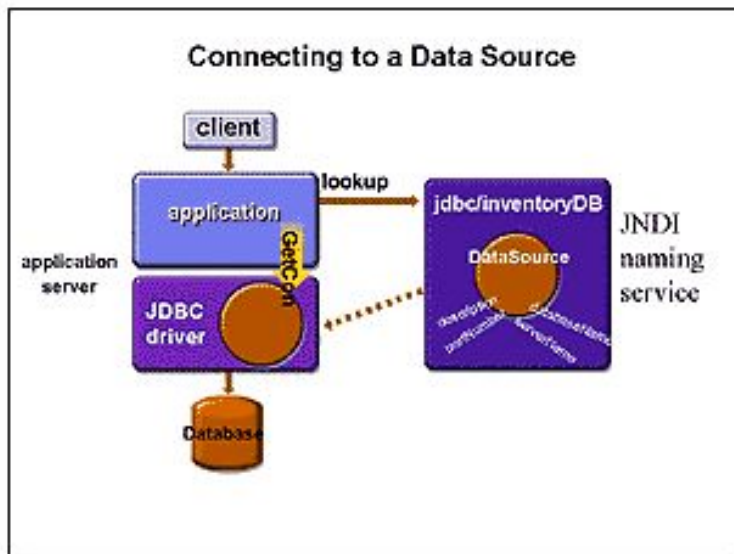
## ■ JDBC Arquitetura



# Java Database Connectivity (JDBC)

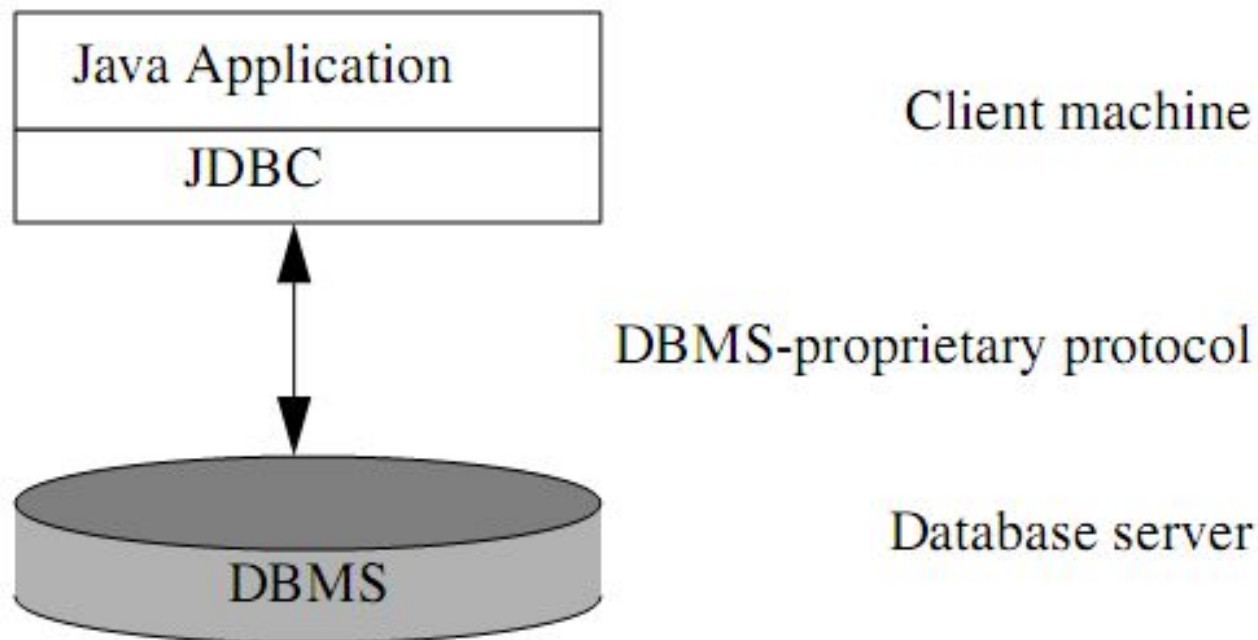
- JDBC Arquitetura

- Conexões com o SGBD são identificadas por meio de uma URL.
  - *string de conexão*



# Java Database Connectivity (JDBC)

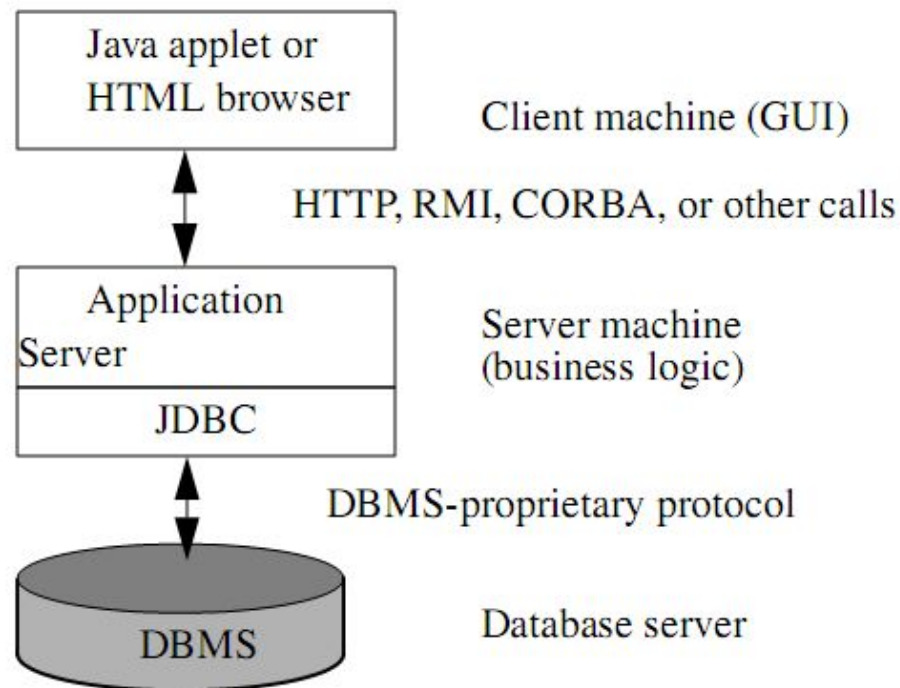
- JDBC Arquitetura
  - Suporte ao modelo de desenvolvimento 2-Camadas, 3-Camadas ou N-Camadas



# Java Database Connectivity (JDBC)

- JDBC Arquitetura

- Suporte ao modelo de desenvolvimento 2-Camadas, 3-Camadas ou N-Camadas



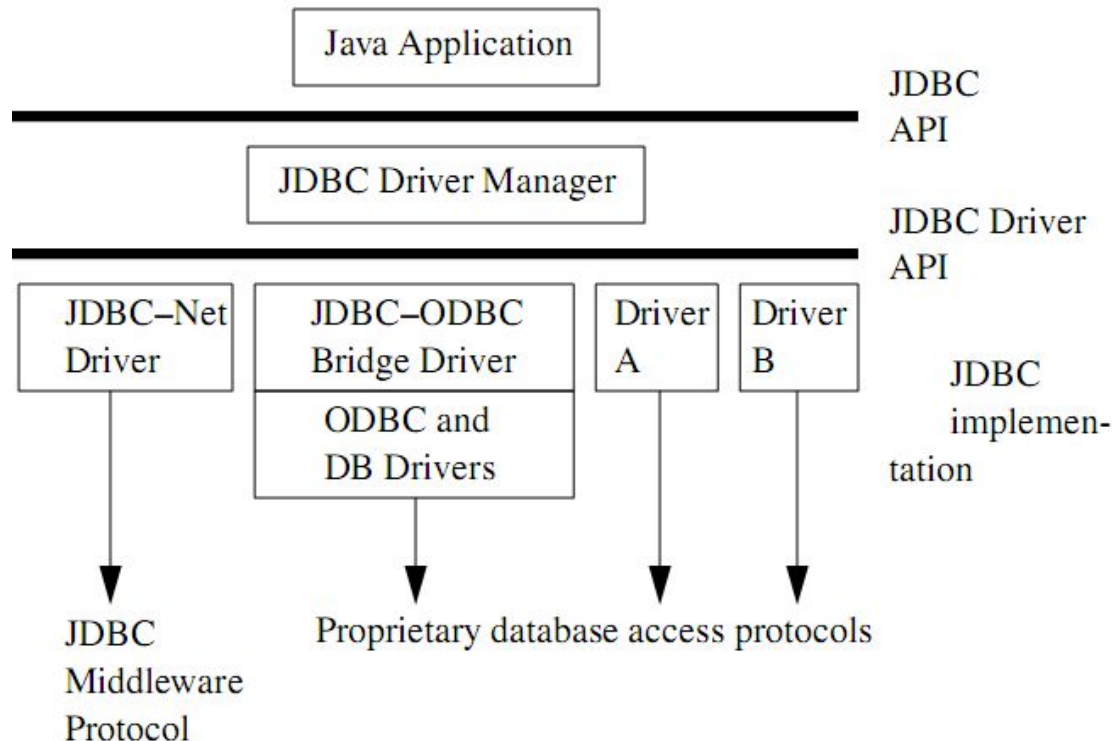
# Java Database Connectivity (JDBC)

- JDBC Mapeamento
  - JDBC Type e Java Type

JDBC Type	Java Type
CHAR	String
VARCHAR	String
LONGVARCHAR	String
NUMERIC	java.math.BigDecimal
DECIMAL	java.math.BigDecimal
BIT	boolean
BOOLEAN	boolean
TINYINT	byte
SMALLINT	short
INTEGER	int
BIGINT	long
REAL	float
FLOAT	double
DOUBLE	double
BINARY	byte[]
VARBINARY	byte[]
LONGVARBINARY	byte[]
DATE	java.sql.Date
TIME	java.sql.Time
TIMESTAMP	java.sql.Timestamp
CLOB	Clob
BLOB	Blob
ARRAY	Array
DISTINCT	mapping of underlying type
STRUCT	Struct
REF	Ref
DATALINK	java.net.URL
JAVA_OBJECT	underlying Java class

# Java Database Connectivity (JDBC)

- JDBC Driver Manager
  - Realiza a conexão das aplicações Java a partir de um driver JDBC



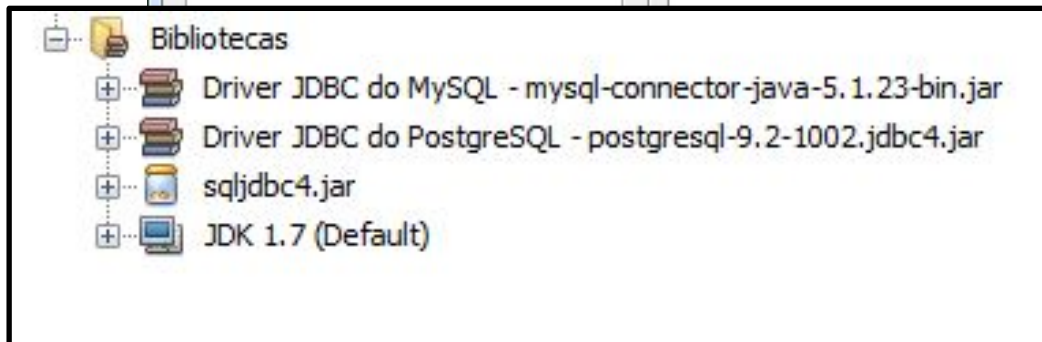
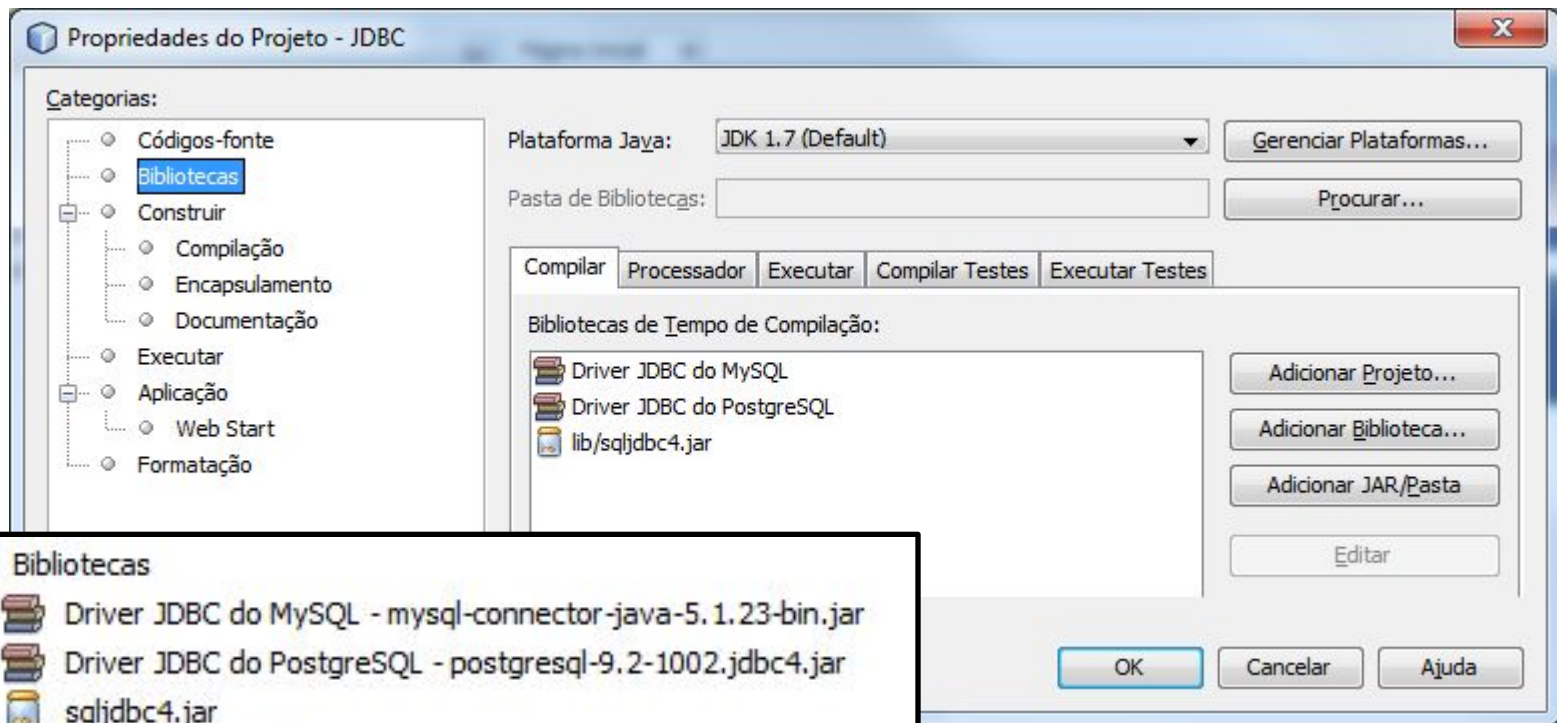
# Java Database Connectivity (JDBC)

- Estabelecendo uma Conexão via JDBC
  - Driver do SGBD
    - PostgreSQL
      - [jdbc.postgresql.org](http://jdbc.postgresql.org)
    - MySQL
      - [dev.mysql.com/downloads/connector/j/](http://dev.mysql.com/downloads/connector/j/)
    - Microsoft SQL Server
      - [msdn.microsoft.com/pt-br/sqlserver/aa937724.aspx](http://msdn.microsoft.com/pt-br/sqlserver/aa937724.aspx)



# Java Database Connectivity (JDBC)

- Estabelecendo uma Conexão via JDBC
  - Adicionar Driver ao Projeto



# Java Database Connectivity (JDBC)

- Estabelecendo uma Conexão via JDBC
  - Definir a URL de Conexão com o SGBD

MySQL	<code>jdbc:mysql://hostname:portNumber/databaseName</code>
ORACLE	<code>jdbc:oracle:thin:@hostname:portNumber:databaseName</code>
DB2	<code>jdbc:db2:hostname:portNumber/databaseName</code>
Java DB/Apache Derby	<code>jdbc:derby:databaseName (embedded)</code> <code>jdbc:derby://hostname:portNumber/databaseName (network)</code>
Microsoft SQL Server	<code>jdbc:sqlserver://hostname:portNumber;databaseName=databaseName</code>
Sybase	<code>jdbc:sybase:Tds:hostname:portNumber/databaseName</code>
PostgreSQL	<code>jdbc:postgresql://host:5432/databaseName</code>

# Java Database Connectivity (JDBC)

- Estabelecendo uma Conexão via JDBC
  - Classe de Conexão Genérica

```
public class Conexao {  
  
    public static Connection Conectar(String DRIVER, String URL,  
        String USUARIO, String SENHA) {  
        try {  
            Class.forName(DRIVER);  
            return DriverManager.getConnection(URL, USUARIO, SENHA);  
        } catch (ClassNotFoundException | SQLException e) {  
            System.out.println("ERRO:" + e);  
            return null;  
        }  
    }  
  
    public static void Desconectar(Connection con) {  
        try {  
            con.close();  
        } catch (Exception e) {  
            System.out.println("ERRO: " + e.getMessage());  
        }  
    }  
}
```

# Java Database Connectivity (JDBC)

- Estabelecendo uma Conexão via JDBC
  - Teste de Conexão

```
public class TesteConexao {  
  
    public static void main(String[] args) {  
  
        Connection con = ConexaoPostgreSQL.Conectar();  
  
        if (con != null){  
            System.out.println("Conexao realizada com sucesso!");  
        }else{  
            System.out.println("Impossível conectar");  
        }  
    }  
}
```

# Java Database Connectivity (JDBC)

- Statements

- Objeto utilizado para enviar instruções ao SGBD

- Métodos

- executeQuery

- Utilizado para SELECT

- executeUpdate

- Utilizado para INSERT, UPDATE e DELETE

- Exemplo

```
Connection con = DriverManager.getConnection(url, "sunny", "");  
Statement stmt = con.createStatement();  
ResultSet rs = stmt.executeQuery("SELECT a, b, c FROM Table2");
```

# Java Database Connectivity (JDBC)

- PreparedStatement
  - Tipo especial de Statement
  - Contém instruções SQL previamente compiladas.
  - Utilizado em instruções que serão executadas muitas vezes
    - Reduz tempo de execução
    - SQL pré-compilado é enviado ao SGBD
  - Parâmetros (IN-Parameters)
    - Definidos por meio do símbolo ?

# Java Database Connectivity (JDBC)

- **ResultSet**

- Objeto que contém o resultado da execução de instruções SQL.
- Contém um conjunto de linhas resultantes de uma consulta no SGBD
  - As linhas são indexadas a partir do número 1.
- Para retornar as informações são utilizados os métodos:
  - `next()`      □ Avança uma linha
  - `getInt()`      □ `getInt(1)` ou `getInt("codigo");`
  - `getString()`
  - `getDouble()`

# Atividades Práticas



Escreva uma aplicação para armazenar resultados de jogos de futebol. Crie um banco de dados denominado **campeonato**, a uma tabela nomeada **tb\_resultados** com o seguinte código SQL:

```
CREATE TABLE tb_resultados(  
    id SERIAL PRIMARY KEY,  
    time1 VARCHAR(100),  
    time2 VARCHAR(100),  
    resultado1 INT,  
    resultado2 INT  
);
```



The form is titled "Inserir Resultado". It contains two dropdown menus labeled "Time 1" and "Time 2", both showing "Time 1" and "Time 2" respectively. Below the dropdowns, there are two input fields for goals, containing the numbers "1" and "0", separated by a minus sign "-". At the bottom of the form is a blue button labeled "Inserir".

Utilize uma arquitetura MVC para criação da aplicação e implemente uma operação capaz de inserir os resultados das partidas.

**Objetivo:** Permitir que o usuário visualize os resultados das partidas registradas no sistema. Esta funcionalidade permite ao usuário consultar os resultados de partidas entre dois times, com possibilidade de aplicar filtros como nome dos times.

### Campos de Filtro

- Time 1: Nome parcial ou completo do primeiro time.
- Time 2: Nome parcial ou completo do segundo time.

### Ações:

- O usuário insere critérios de busca nos campos de filtro.
- O sistema consulta o banco de dados e retorna a lista de partidas que atendem aos critérios informados.
- Os resultados são exibidos em um JTable

FIM