

Java DataBase Connectivity (JDBC)

Vinicius Rosalen

Parte IV

Mapeamento Objeto Relacional

➡ De forma geral, ao se realizar uma transposição de um modelo orientado a objetos para um modelo relacional, algumas regras devem ser seguidas....

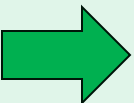
➡ 1. Todas as tabelas (ou relações) devem ter uma chave primária.

➡ Em um sistema orientado a objetos, cada objeto é único.

➡ Essa unicidade é garantida através da introdução de um “**identificador de objetos**” (**OID – Object Identifier**).

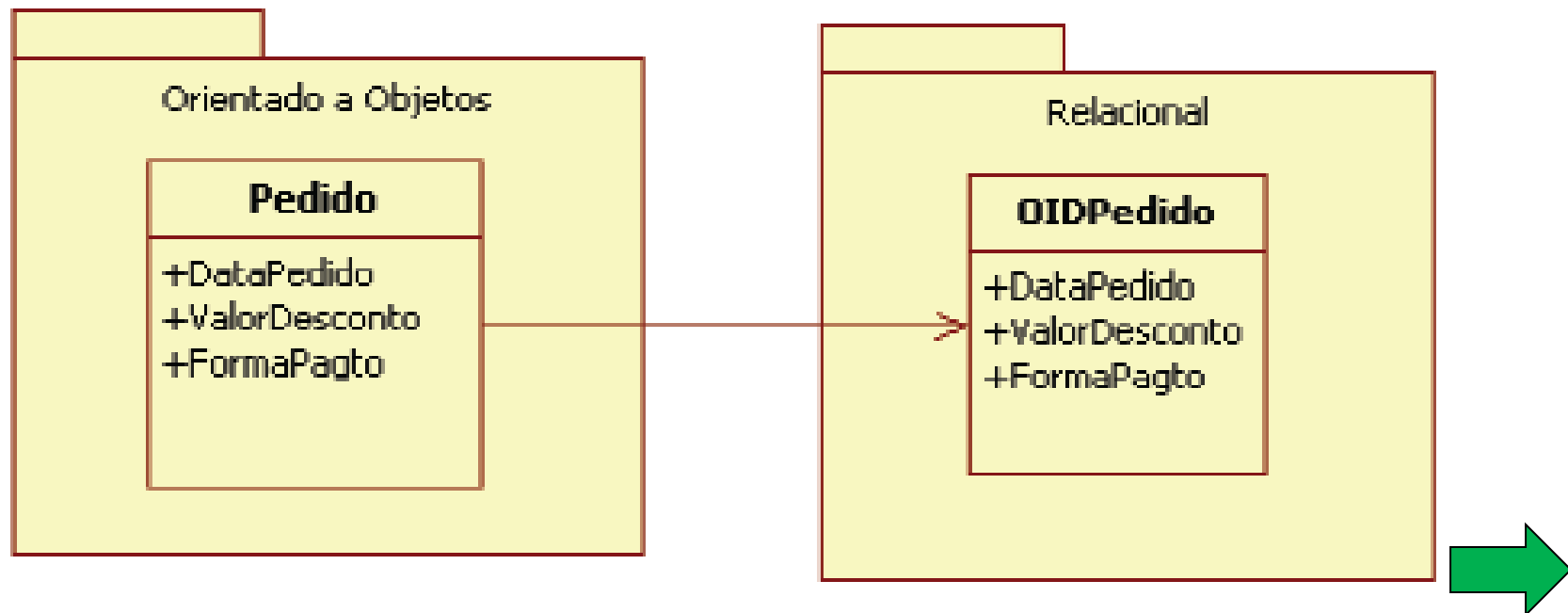
- Esse OID normalmente é gerado pelo sistema, não podendo ser alterado pelo usuário.

➡ Espera-se que seu valor não depende dos valores dos atributos do objeto.



Mapeamento Objeto Relacional

- ➔ 1. Todas as tabelas (ou relações) devem ter uma chave primária.



Mapeamento Objeto Relacional

➡ De forma geral, ao se realizar uma transposição de um modelo orientado a objetos para um modelo relacional, algumas regras devem ser seguidas....

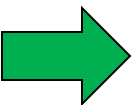
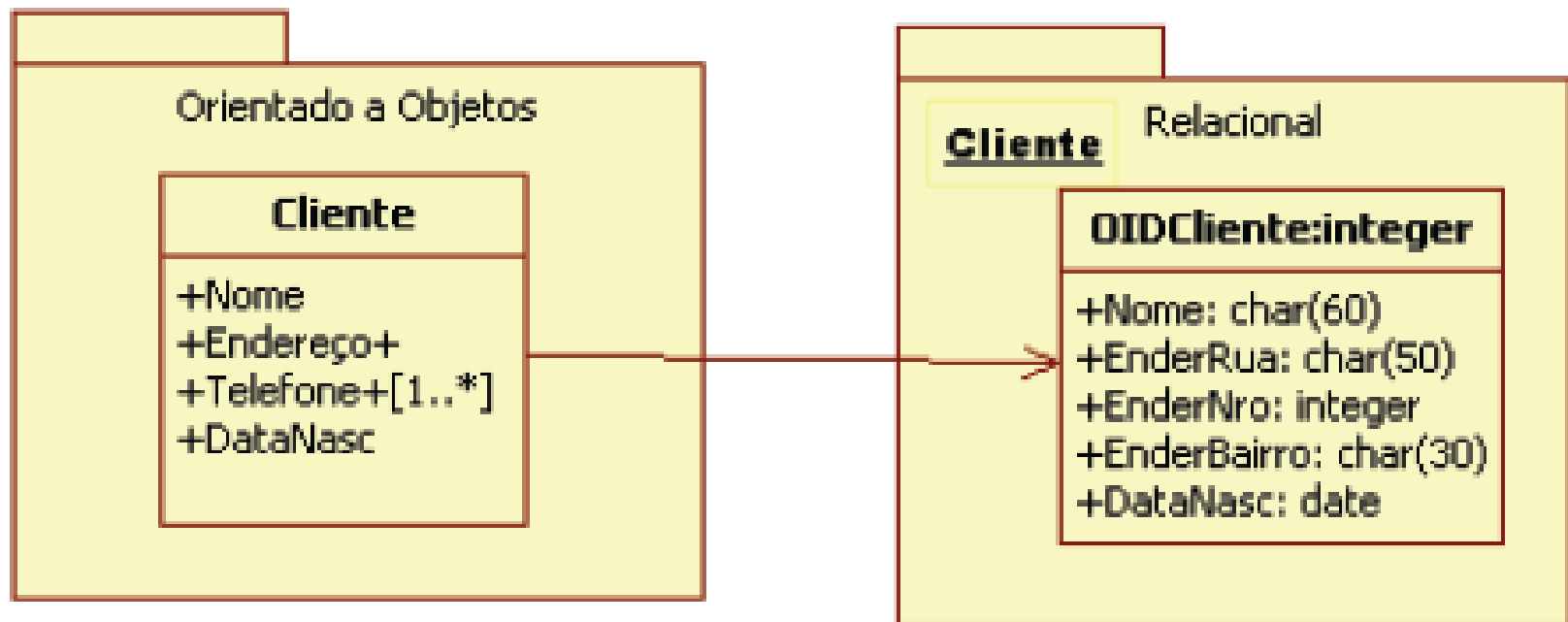
➡ 2. Mapeamento de atributos.

- ➡ Os atributos simples são mapeados para colunas.
- ➡ Os atributos compostos podem ser mapeados em várias colunas.
- ➡ Já os atributos multivalorados devem ser mapeados em tabelas
 - Onde a chave primária é composta pela chave primária da tabela que representa a classe que contém o atributo multivalorado e pela chave primária que representa o atributo multivalorado.



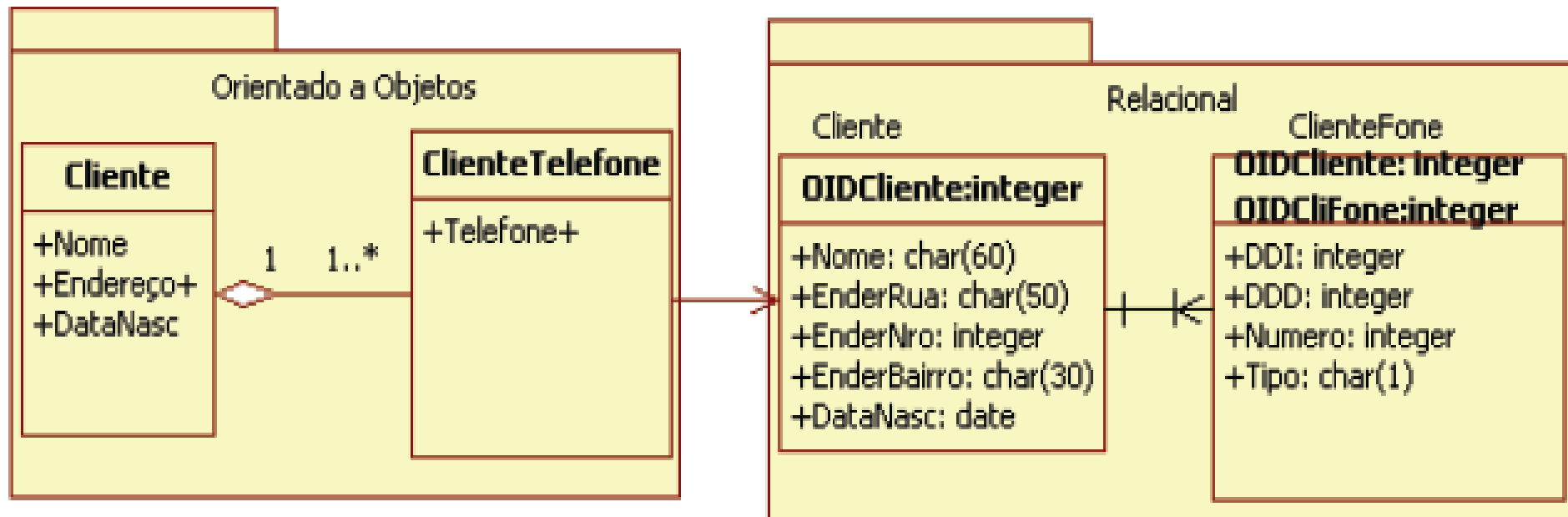
Mapeamento Objeto Relacional

➔ 2. Mapeamento de atributos.



Mapeamento Objeto Relacional

➔ 2. Mapeamento de atributos.

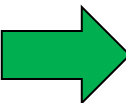


Mapeamento Objeto Relacional

➡ De forma geral, ao se realizar uma transposição de um modelo orientado a objetos para um modelo relacional, algumas regras devem ser seguidas....

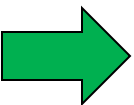
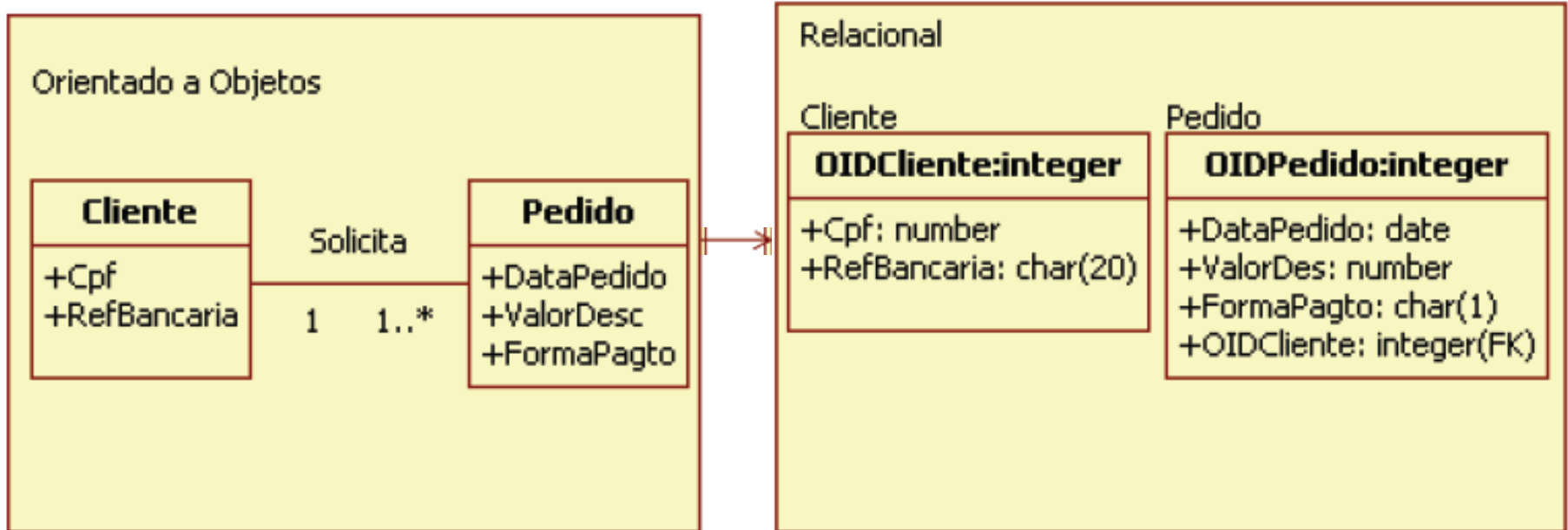
➡ 3. Associações Um-para-Muitos:

➡ Neste caso, a tabela cujos registros podem ser endereçados diversas vezes (lado Muitos do relacionamento) é a que herda a referência da tabela cuja correspondência é unitária.



Mapeamento Objeto Relacional

➔ 3. Associações Um-para-Muitos.



Mapeamento Objeto Relacional

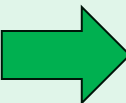
➡ De forma geral, ao se realizar uma transposição de um modelo orientado a objetos para um modelo relacional, algumas regras devem ser seguidas....

➡ 4. Associações Um-para-Um:

➡ Nesse tipo de associação é possível optar por 1. gerar uma única tabela ou então 2. gerar duas tabelas (uma para cada classe).


➡ Na primeira opção, os atributos da classe agregada devem ser colocados na mesma tabela da classe agregadora.


- Vantagem: performance é melhor pois é preciso acessar uma única tabela.
- Desvantagem: aumenta a quantidade de páginas que são recuperadas em cada acesso à tabela.



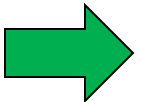
Mapeamento Objeto Relacional

4. Associações Um-para-Um:

 Nesse tipo de associação é possível optar por 1. gerar uma única tabela ou então 2. gerar duas tabelas (uma para cada classe).

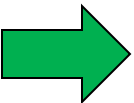
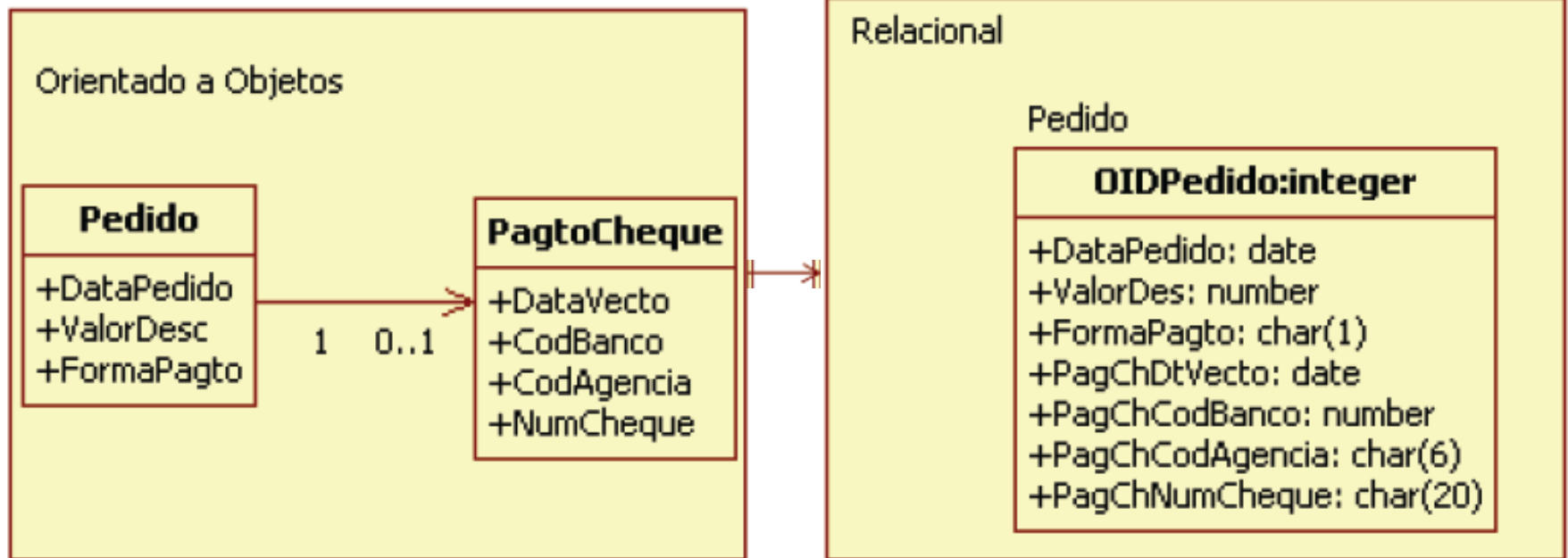
 Na segunda opção, uma delas deve herdar como um atributo normal (chave estrangeira) a chave primária da outra tabela.

- **Vantagem:** facilita a manutenção das tabelas e torna a estrutura do banco de dados mais flexível.
- **Desvantagem:** consultas necessitam de uma operação de junção (join) ou pelo menos dois acessos ao banco de dados e é mais trabalhoso de se implementar no sistema.



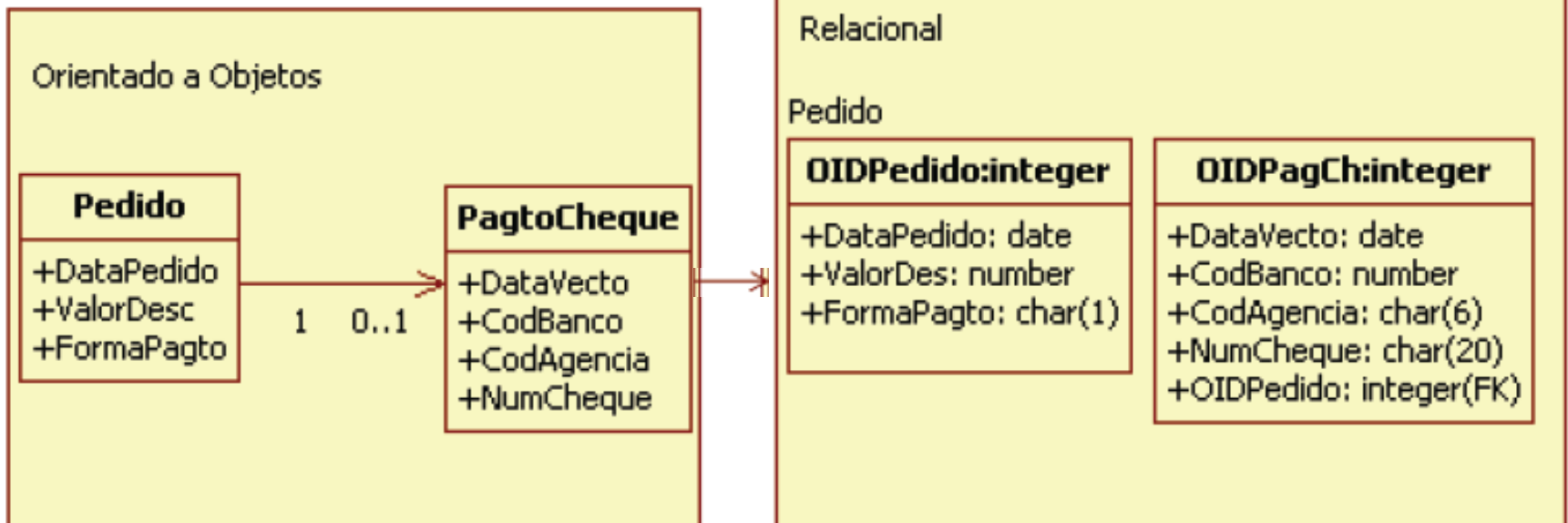
Mapeamento Objeto Relacional

➔ 4. Associações Um-para-Um.



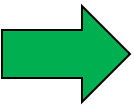
Mapeamento Objeto Relacional

➔ 4. Associações Um-para-Um.



Mapeamento 1x1 e 1xN

- Blz,
 - Chega de conversa, vamos trabalhar...
- Objetivo
 - Montar uma modelagem Livro1x1Autor e Livro1xNAutor com WorkBench e MySQL
 - Fazer o código OO e implementar o Salvar e Consultar com DAO



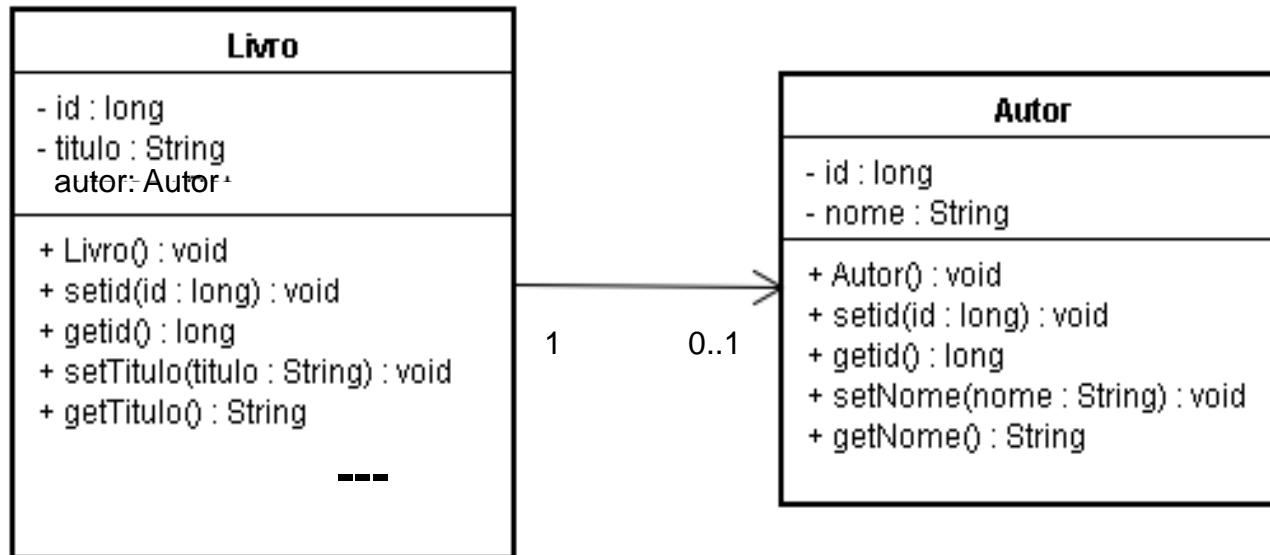
Mapeamento 1x1

➔ Considere a relação livro-autor:

- Um livro pode ser escrito por um autor;
- Um autor, por sua vez, pode escrever um livro

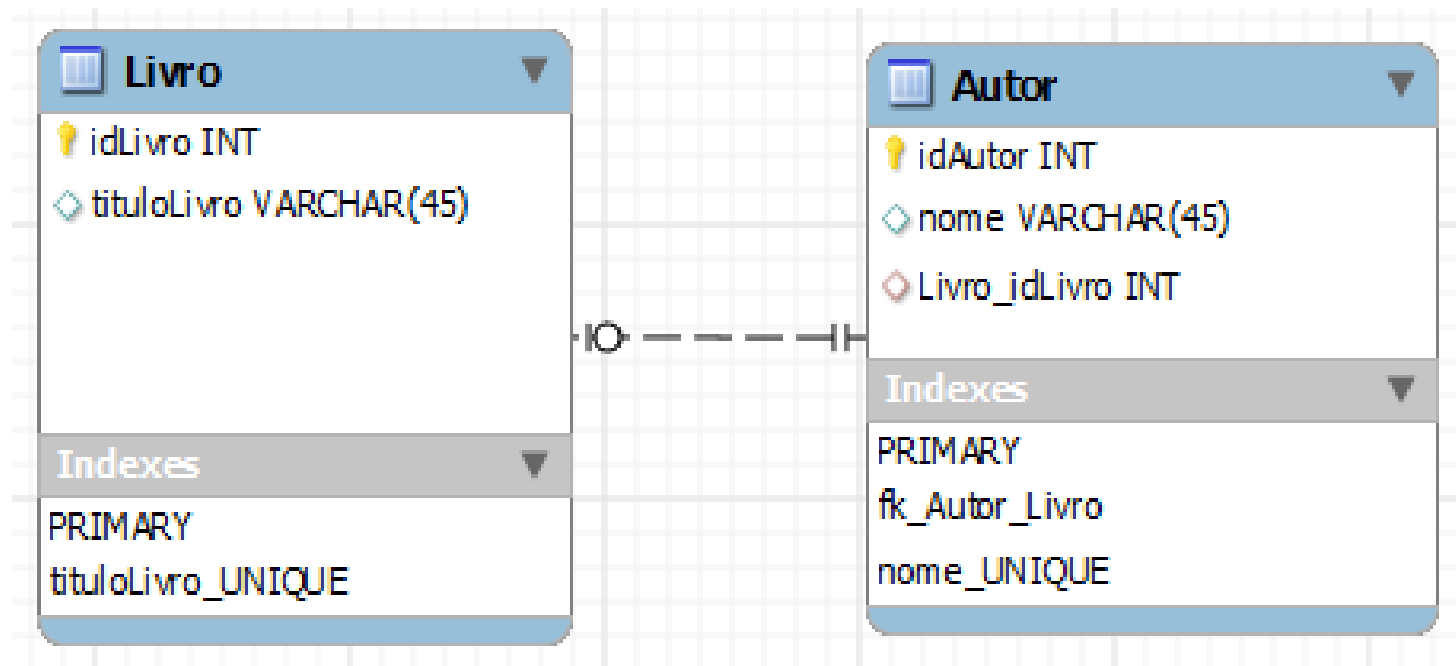
➔ Na implementação de 1-para-1 em OO...

- Dependendo da modelagem, basta definir um atributo do tipo em questão na outra classe



Mapeamento 1x1

➡ Diagrama Relacional criado utilizando o MySQL Workbench



➡ A questão agora é: como fazer o mapeamento objeto-relacional dessas associações no código??

MySQL Workbench



Data Modeling

Create and manage models, forward & reverse engineer, compare and synchronize schemas, report.



Open Existing EER Model

Or select a model to open or click here to browse...



Create New EER Model

Create a new EER Model from scratch.

Description Editor

No Selection

Model Overview

Add Diagram

Physical Schemata

mydb
MySQL Schema

Tables (0 items)

Add Table

Views (0 items)

Add View

Routines (0 items)

Add Routine

Routine Groups (0 items)

Add Group

Schema Privileges

SQL Scripts

Model Notes

User Types List

Name	Definition	Flags
BOOL	TINYINT(1)	
BOOLE...	TINYINT(1)	
FIXED	DECIMAL(10 ...	
FLOAT4	FLOAT	
FLOAT8	DOUBLE	
INT1	TINYINT(4)	
INT2	SMALLINT(6)	
INT3	MEDIUMINT...	
INT4	INT(11)	

Model Overview

Add Diagram

Physical Schemata

mysqlLivroAutorNxN
MySQL Schema

Tables (0 items)

Add Table

Views (0 items)

Add View

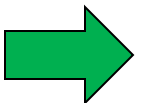
mysqllivroautor1x1 x

Name: mysqllivroautor1x1

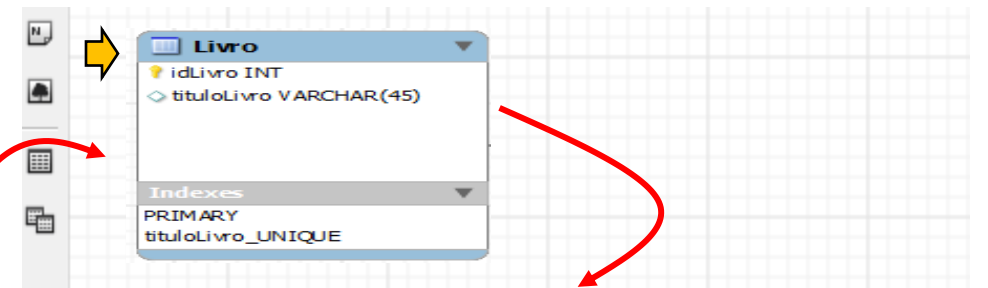
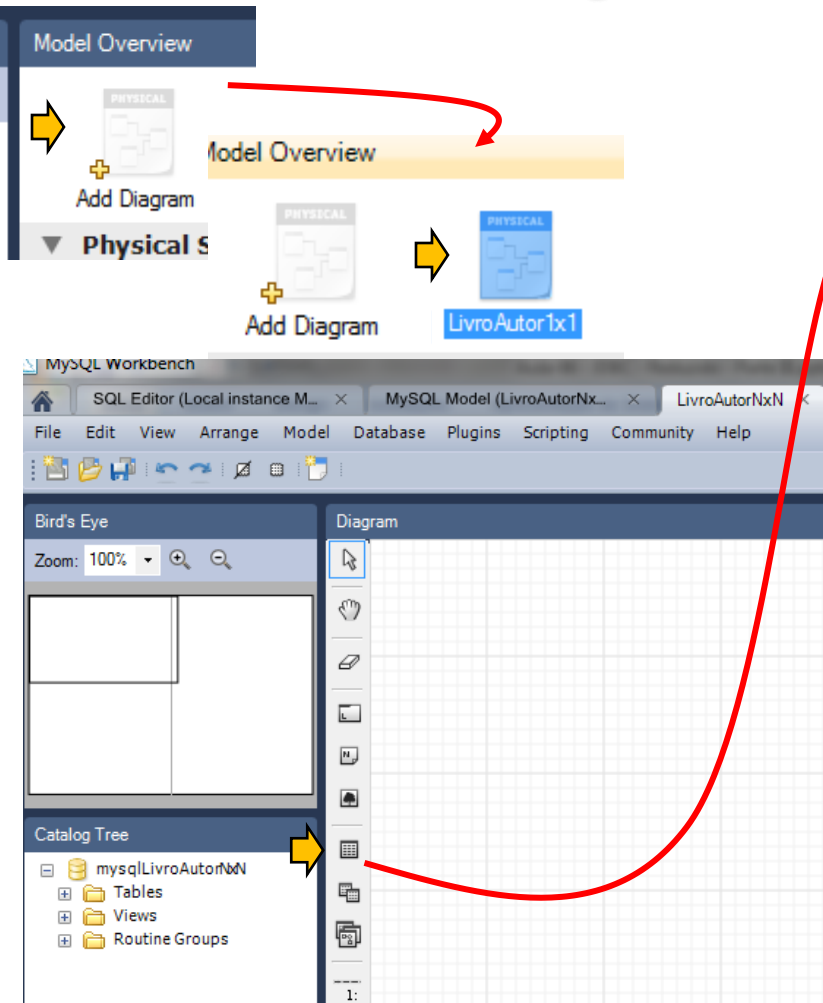
Collation: utf8 - utf8_general_ci

Comments:

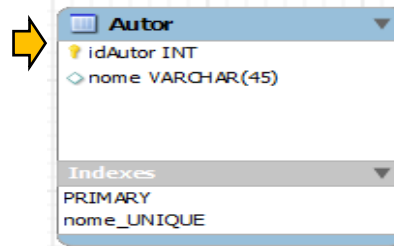
- Cut 'mysqlLivroAutorNxN'
- Copy 'mysqlLivroAutorNxN'
- Paste
- Edit Schema...
- Edit in New Window...
- Copy SQL to Clipboard
- Delete 'mysqlLivroAutorNxN'



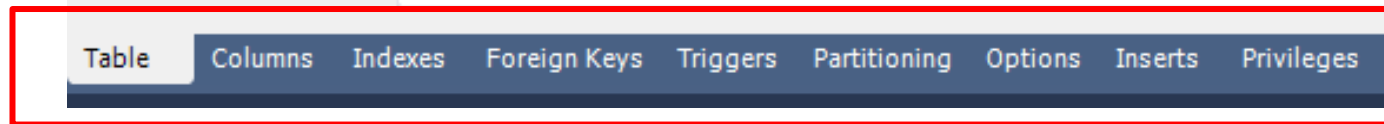
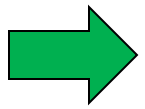
MySQL Workbench



Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI
idLivro	BIGINT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
tituloLivro	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>



Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI
idAutor	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
nome	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>



MySQL Workbench

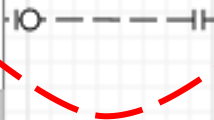


Livro

- idLivro INT
- tituloLivro VARCHAR(45)

Indexes

- PRIMARY
- tituloLivro_UNIQUE



Autor

- idAutor INT
- nome VARCHAR(45)
- Livro_idLivro INT

Indexes

- PRIMARY
- fk_Autor_Livro
- nome_UNIQUE

Livro

Column Name	Datatype	PK	NN
idLivro	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
tituloLivro	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>

Table Columns Indexes Foreign Keys Triggers Partitions

Autor

Column Name	Datatype	PK	NN	UQ
idAutor	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
nome	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Livro_idLivro	INT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Table Columns Indexes Foreign Keys Triggers

Autor

Index Name	Type
PRIMARY	PRIMARY
fk_Autor_Livro	INDEX
nome_UNIQUE	UNIQUE

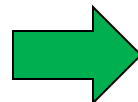
Table Columns Indexes Foreign Keys Triggers

Foreign Key Name: fk_Autor_Livro
Referenced Table: `mysqllivroautor1x1`.`Livro`

Foreign Key Options

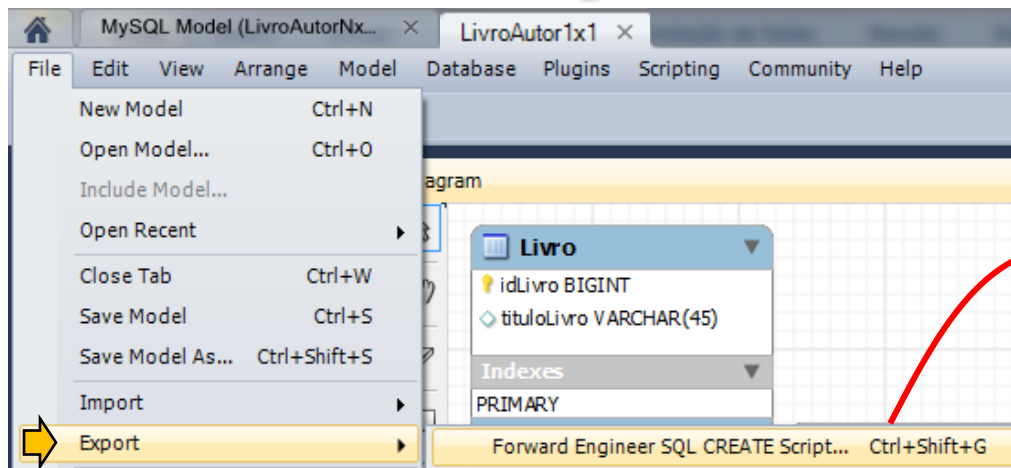
- On Update: CASCADE
- On Delete: CASCADE
- ☐ Skip in SQL generation
- Foreign Key Comment:

Table Columns Indexes Foreign Keys Triggers

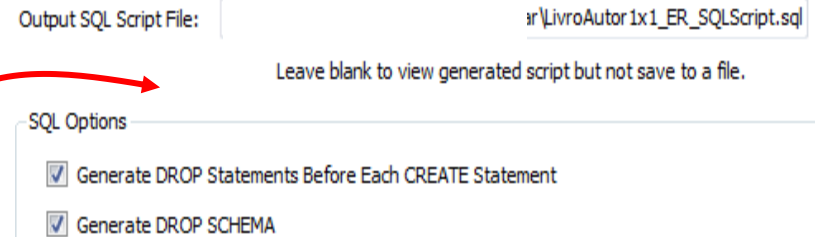


Exportando o script .sql

MySQL Workbench

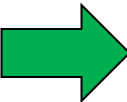
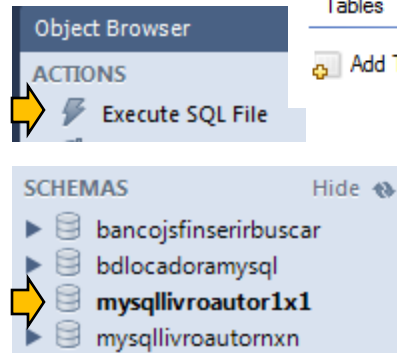
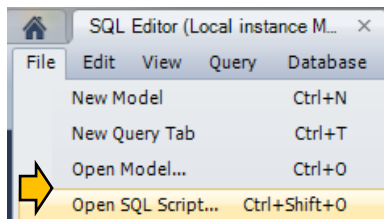
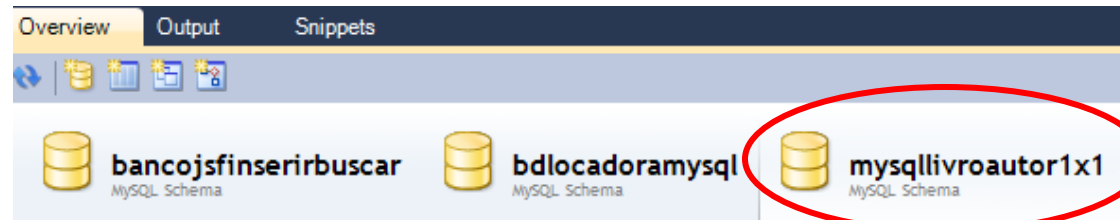
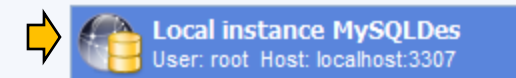


SQL Export Options



“Instalando” as tabelas no banco

Open Connection to Start Querying
Or click a DB connection to open the SQL Editor



MySQL Workbench

➡ Curiosidade: SQL Gerado...

```
-- Table `mysqllivroautor1x1`.`Livro`
```

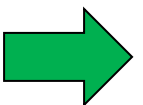
```
DROP TABLE IF EXISTS `mysqllivroautor1x1`.`Livro` ;
```

```
CREATE TABLE IF NOT EXISTS `mysqllivroautor1x1`.`Livro` (  
  `idLivro` INT NOT NULL AUTO_INCREMENT ,  
  `tituloLivro` VARCHAR(45) NULL ,  
  PRIMARY KEY (`idLivro`) ,  
  UNIQUE INDEX `tituloLivro_UNIQUE` (`tituloLivro` ASC) )  
ENGINE = InnoDB;
```

```
-- Table `mysqllivroautor1x1`.`Autor`
```

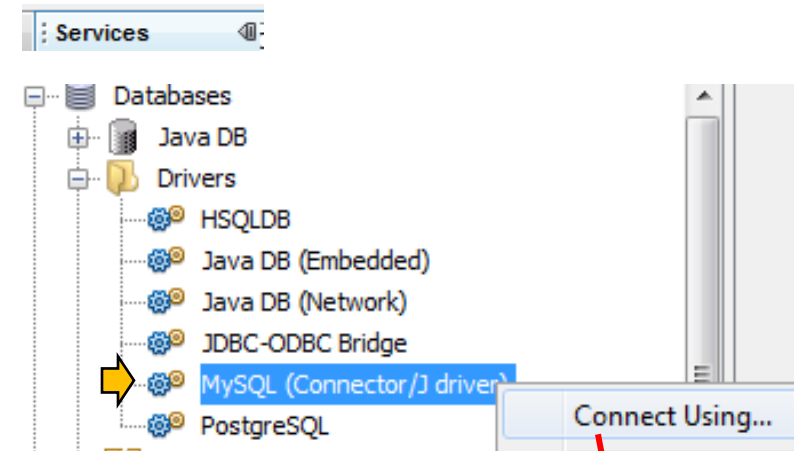
```
DROP TABLE IF EXISTS `mysqllivroautor1x1`.`Autor` ;
```

```
CREATE TABLE IF NOT EXISTS `mysqllivroautor1x1`.`Autor` (  
  `idAutor` INT NOT NULL AUTO_INCREMENT ,  
  `nome` VARCHAR(45) NULL ,  
  `Livro_idLivro` INT NULL ,  
  PRIMARY KEY (`idAutor`) ,  
  INDEX `fk_Autor_Livro` (`Livro_idLivro` ASC) ,  
  UNIQUE INDEX `nome_UNIQUE` (`nome` ASC) ,  
  CONSTRAINT `fk_Autor_Livro`  
    FOREIGN KEY (`Livro_idLivro`)  
    REFERENCES `mysqllivroautor1x1`.`Livro` (`idLivro` )  
    ON DELETE CASCADE  
    ON UPDATE CASCADE)  
ENGINE = InnoDB;
```



JDBC e NetBeans

➔ Testando conexão com o banco



New Database Connection

Basic setting | **Advanced**

Data Input Mode: ☒ Field Entry ☐ Direct URL Entry

Driver Name: MySQL (Connector/J driver)

Host: localhost

Port: 3307

Database: mysqllivroautor1x1

User Name: root

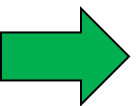
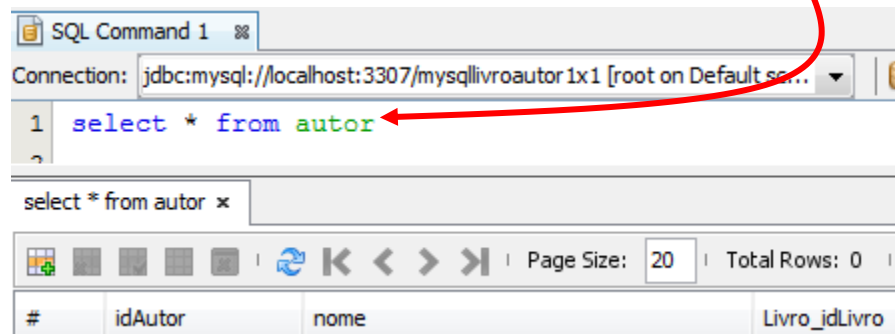
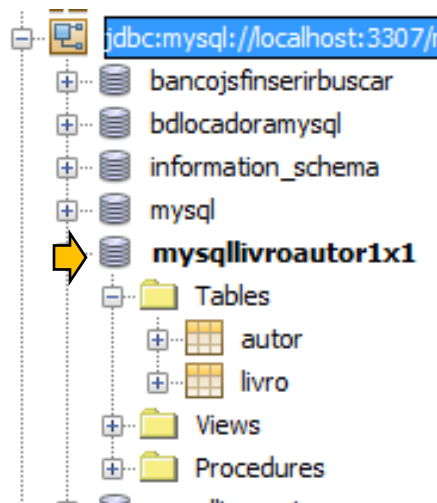
Password:

Display Name (Optional):

☐ Remember password (see help for information on security risks)

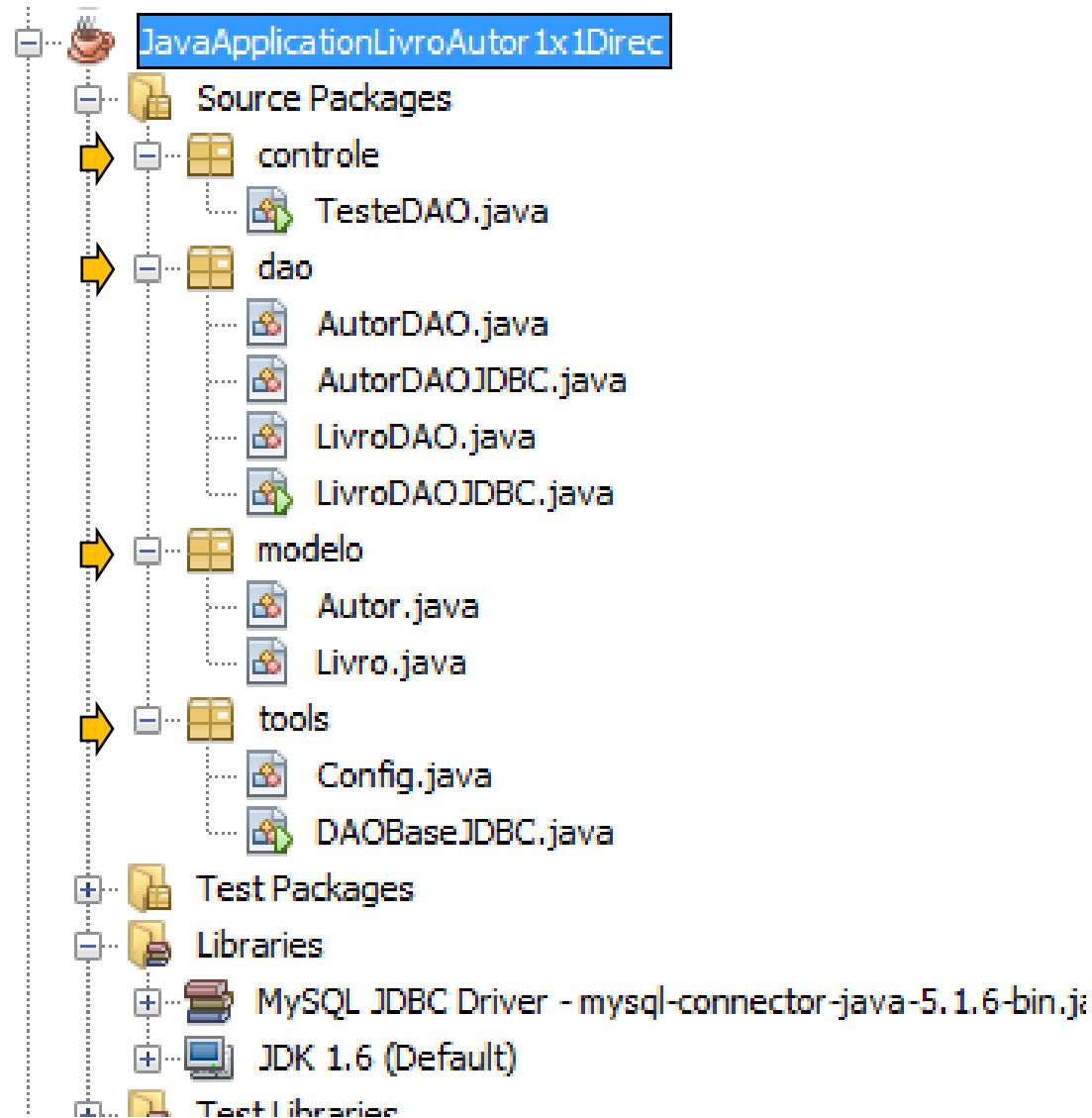
Additional Props:

☒ Show JDBC URL jdbc:mysql://localhost:3307/mysqllivroautor1x1



JDBC e NetBeans

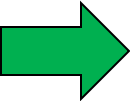
➡ Montando o projeto... no final teremos a seguinte estrutura



Mapeamento código OO x Relacional

➡ Primeiro vamos criar uma classe que será responsável pela conexão com o banco

```
public class DAOBaseJDBC {  
    protected static Connection conn;  
  
    static {  
        try {  
            // Carrega o driver do MySQL e conecta.  
            Class.forName(Config.NOME_DRIVER);  
            conn = DriverManager.getConnection(Config.BD_URL, Config.BD_LOGIN, Config.BD_SENHA);  
        } catch (ClassNotFoundException e) {  
            System.out.println("FATAL: driver não encontrado.");  
            System.exit(1);  
        } catch (SQLException e) {  
            System.out.println("Erro SQL: " + e.getMessage());  
            System.exit(1);  
        }  
    }  
}  
  
public interface Config {  
    public static final String NOME_DRIVER = "com.mysql.jdbc.Driver";  
    public static final String BD_URL = "jdbc:mysql://localhost:3307/mysqllivroautor1x1";  
    public static final String BD_LOGIN = "root";  
    public static final String BD_SENHA = "123456";  
}
```



Mapeamento código OO x Relacional

➔ Agora vamos construir as classes de entidades do domínio:
Livro e Autor

Autor.java

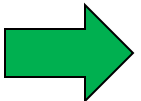
```
public class Autor {  
    private Long id;  
    private String nome;
```

```
    public Autor() {...}  
    public Autor(Long id) {...}  
    public Long getId() {...}  
    public void setId(long id) {...}  
    public String getNome() {...}  
    public void setNome(String nome) {...}  
    @Override  
    public String toString() {...}  
}
```

Livro.java

```
public class Livro {  
    private Long id;  
    private String titulo;  
    Autor autor;
```

```
    public Livro() {...}  
    public Autor getAutor() {...}  
    public void setAutor(Autor autor) {...}  
    public Long getId() {...}  
    public void setId(Long id) {...}  
    public String getTitulo() {...}  
    public void setTitulo(String titulo) {...}  
  
    @Override  
    public String toString() {...}  
}
```



Mapeamento código OO x Relacional

➡ Agora vamos construir o **AutorDAO**, responsável pela persistência dos objetos do tipo **Autor**

AutorDAO.java

AutorDAOJDBC.java

```
public class AutorDAOJDBC extends DAOBaseJDBC implements AutorDAO {
```

```
    public boolean salvar(Autor autor) {
```

```
        PreparedStatement stmt;
```

```
        try {
```

```
            if (autor.getId() == null) {
```

```
                System.out.println(" === salvando..... === ");
```

```
                stmt =
```

```
                conn.prepareStatement("INSERT INTO Autor (nome) VALUES (?)");
```

```
            } else {
```

```
                System.out.println(" === atualizando..... === ");
```

```
                stmt =
```

```
                conn.prepareStatement("UPDATE Autor SET nome = ? WHERE idAutor = ?");
```

```
                stmt.setLong(2, autor.getId());
```

```
            }
```

```
            stmt.setString(1, autor.getNome());
```

```
            stmt.executeUpdate() stmt.close();
```

```
        } catch (SQLException e) {
```

```
            System.out.println("Erro SQL: " + e.getMessage());
```

```
            System.out.println("Autor não gravado pois já esta na base de dados.....: " + autor);
```

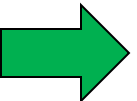
```
            return false;
```

```
        }
```

```
        return true;
```

```
    }
```

```
public interface AutorDAO {  
    public boolean salvar(Autor artista);  
    public Autor consultar(String nome);  
}
```



Mapeamento código OO x Relacional

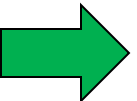
➡ Agora vamos construir o **AutorDAO**, responsável pela persistência dos objetos do tipo **Autor**

AutorDAOJDBC.java

```
public Autor consultar(String nome) {  
    Autor autorLido = null;  
    try {  
        PreparedStatement stmt =  
            conn.prepareStatement(  
                "SELECT idAutor, nome FROM Autor WHERE nome = ?");  
        stmt.setString(1, nome);  
        ResultSet rset = stmt.executeQuery();  
        if (rset.next()) {  
            autorLido = new Autor();  
            autorLido.setId(new Long(rset.getLong("idAutor")));  
            autorLido.setNome(rset.getString("nome")); stmt.close();  
        }  
        else  
            return null;  
    } catch (SQLException e) {  
        System.out.println("Falha na consulta: " + e.getMessage());  
        return null;  
    }  
    return autorLido;  
}
```

AutorDAO.java

```
public interface AutorDAO {  
    public boolean salvar(Autor artista);  
    public Autor consultar(String nome);  
}
```

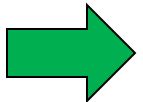


Mapeamento código OO x Relacional

➡ Agora vamos construir o LivroDAO, responsável pela persistência dos objetos do tipo Livro

```
public class LivroDAOJDBC extends DAOBaseJDBC implements LivroDAO {  
  
    public boolean salvar(Livro livro) {  
        PreparedStatement stmt = null;  
        try {  
            conn.setAutoCommit(false);  
            if (livro.getId() == null) {  
                stmt = conn.prepareStatement("INSERT INTO Livro (tituloLivro) VALUES (?)");  
            } else {  
                stmt = conn.prepareStatement("UPDATE Livro SET tituloLivro = ? WHERE idLivro = ?");  
                stmt.setLong(2, livro.getId());  
            }  
            stmt.setString(1, livro.getTitulo());  
            stmt.executeUpdate(); stmt.close();  
            this.gravarAutor(livro);  
            conn.commit(); conn.setAutoCommit(true);  
        } catch (SQLException e) {  
            try {  
                conn.rollback(); conn.setAutoCommit(true);  
            } catch (SQLException ex) {System.out.println("Erro no rollback...");}  
            return false;  
        }  
        return true;  
    }  
}
```

```
public interface LivroDAO {  
    public boolean salvar(Livro livro);  
    public Livro consultar(String titulo);  
}
```



Mapeamento código OO x Relacional

➔ Agora vamos construir o LivroDAO, responsável pela persistência dos objetos do tipo Livro

LivroDAOJDBC.java

LivroDAO.java

```
public interface LivroDAO {  
    public boolean salvar(Livro livro);  
    public Livro consultar(String titulo);  
}
```

```
// pré: o Autor associado ao livro já possuiu o id recuperado...  
// papel do controle neste exemplo  
private void gravarAutor(Livro livro) {
```

```
String sql = "UPDATE Autor SET Livro_idLivro = ? WHERE idAutor = ?";
```

```
PreparedStatement stmt;
```

```
try {
```

```
    stmt = conn.prepareStatement(sql);
```

```
    stmt.setInt(1, this.lerIdLivro());
```

```
    Autor autor = livro.getAutor();
```

```
    stmt.setLong(2, autor.getId());
```

```
    stmt.executeUpdate(); stmt.close();
```

```
} catch (SQLException e) {
```

```
    System.out.println("Autor não atual.
```

```
}
```

```
private int lerIdLivro() {
```

```
String sql = "SELECT MAX(idLivro) FROM livro";
```

```
PreparedStatement stmt = null;
```

```
int idLivro = 0;
```

```
try {
```

```
    stmt = conn.prepareStatement(sql);
```

```
    ResultSet rs = stmt.executeQuery();
```

```
    rs.next();
```

```
    idLivro = rs.getInt(1); stmt.close();
```

```
} catch (SQLException e) {
```

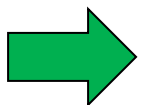
```
    e.printStackTrace();
```

```
    System.out.println("Não foi possível recuperar o MAX idLivro");
```

```
}
```

```
return idLivro;
```

```
}
```



Mapeamento código OO x Relacional

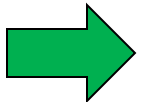
➡ Agora vamos construir o LivroDAO, responsável pela persistência dos objetos do tipo Livro

LivroDAO.java

```
public interface LivroDAO {  
    public boolean salvar(Livro livro);  
    public Livro consultar(String titulo);  
}
```

```
public Livro consultar(String titulo) {  
    Livro livroLido = null;  
    try {  
        PreparedStatement stmt =  
            conn.prepareStatement(  
                "SELECT idLivro, tituloLivro FROM Livro WHERE tituloLivro = ?");  
        stmt.setString(1, titulo);  
  
        ResultSet rset = stmt.executeQuery();  
        if (rset.next()) {  
            livroLido = new Livro();  
            livroLido.setId(new Long(rset.getLong("idLivro")));  
            livroLido.setTitulo(rset.getString("tituloLivro"));  
  
            Autor autor = this.lerAutor(livroLido.getId());  
            livroLido.setAutor(autor); stmt.close();  
        }  
        else  
            return null;  
    } catch (SQLException e) {  
        System.out.println("Falha na consulta: " + e.getMessage());  
        return null;  
    }  
    return livroLido;  
}
```

LivroDAOJDBC.java



Mapeamento código OO x Relacional

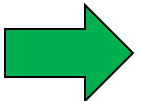
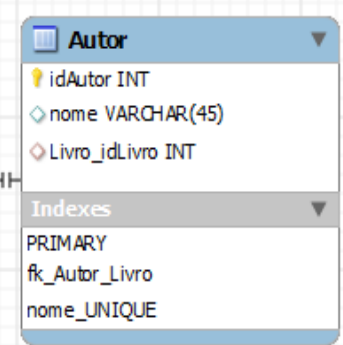
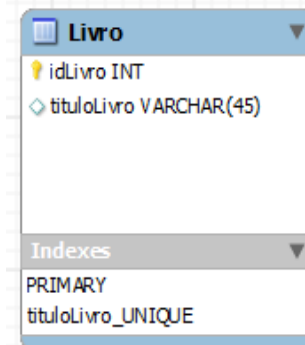
➔ Agora vamos construir o LivroDAO, responsável pela persistência dos objetos do tipo Livro

LivroDAO.java

```
public interface LivroDAO {  
    public boolean salvar(Livro livro);  
    public Livro consultar(String titulo);  
}
```

LivroDAOJDBC.java

```
private Autor lerAutor(long idLivro) {  
    String sql = "SELECT Autor.idAutor, Autor.nome "  
        + "FROM Livro, Autor "  
        + "WHERE Livro.idLivro = ? AND "  
        + "Livro.idLivro = Autor.Livro_idLivro";  
    PreparedStatement stmt = null;  
    Autor autorLido = null;  
    try {  
        stmt = conn.prepareStatement(sql);  
        stmt.setLong(1, idLivro);  
        ResultSet rset = stmt.executeQuery();  
  
        if (rset.next()) {  
            autorLido = new Autor();  
            autorLido.setId(rset.getInt("idAutor"));  
            autorLido.setNome(rset.getString("nome"));  
        } else {  
            return null;  
        }  
    } catch (SQLException e) {  
        System.out.println("Falha na consulta para recuperar o Autor do Livro: ");  
        return null;  
    }  
    return autorLido;  
}
```



Mapeamento código OO x Relacional

➡ Pra fechar, podemos construir uma classe simulando as chamadas do controle à estrutura construída...

```
public class TesteDAO {
```

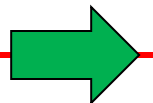
TesteDAO.java

```
public static void main(String args[]) {
```

```
LivroDAOJDBC livroDAO = new LivroDAOJDBC();  
AutorDAOJDBC autorDAO = new AutorDAOJDBC();
```

```
System.out.println("=====");  
System.out.println("Gravando Autores.....");  
System.out.println("=====");  
Autor autor01 = new Autor();  
autor01.setNome("Vinicius");  
autorDAO.salvar(autor01);  
  
Autor autor02 = new Autor();  
autor02.setNome("Rosalen");  
autorDAO.salvar(autor02);
```

```
System.out.println("=====");  
System.out.println("Gravando livros.....");  
System.out.println("=====");  
Livro livro01 = new Livro();  
livro01.setTitulo("Java para vida");  
autor01.setId((autorDAO.consultar("Vinicius")).getId());  
livro01.setAutor(autor01);  
livroDAO.salvar(livro01);  
  
Livro livro02 = new Livro();  
livro02.setTitulo("Opa, agora vai!");  
autor02.setId((autorDAO.consultar("Rosalen")).getId());  
livro02.setAutor(autor02);  
livroDAO.salvar(livro02);
```



Mapeamento código OO x Relacional

➔ Pra fechar, podemos construir uma classe simulando as chamadas do controle à estrutura construída...

TesteDAO.java

```
Autor autorDaPesquisa = null;
System.out.println("=====");
System.out.println("Recuperando autores...");
System.out.println("=====");

autorDaPesquisa = autorDAO.consultar("Vinicius");
System.out.println(autorDaPesquisa);

autorDaPesquisa = autorDAO.consultar("Rosalen");
System.out.println(autorDaPesquisa);
```

```
Livro livroDaPesquisa = null;
System.out.println("=====");
System.out.println("Recuperando Livros...");
System.out.println("=====");

livroDaPesquisa = livroDAO.consultar("Java para vida");
System.out.println(livroDaPesquisa);

livroDaPesquisa = livroDAO.consultar("Opa, agora vai!");
System.out.println(livroDaPesquisa);
```

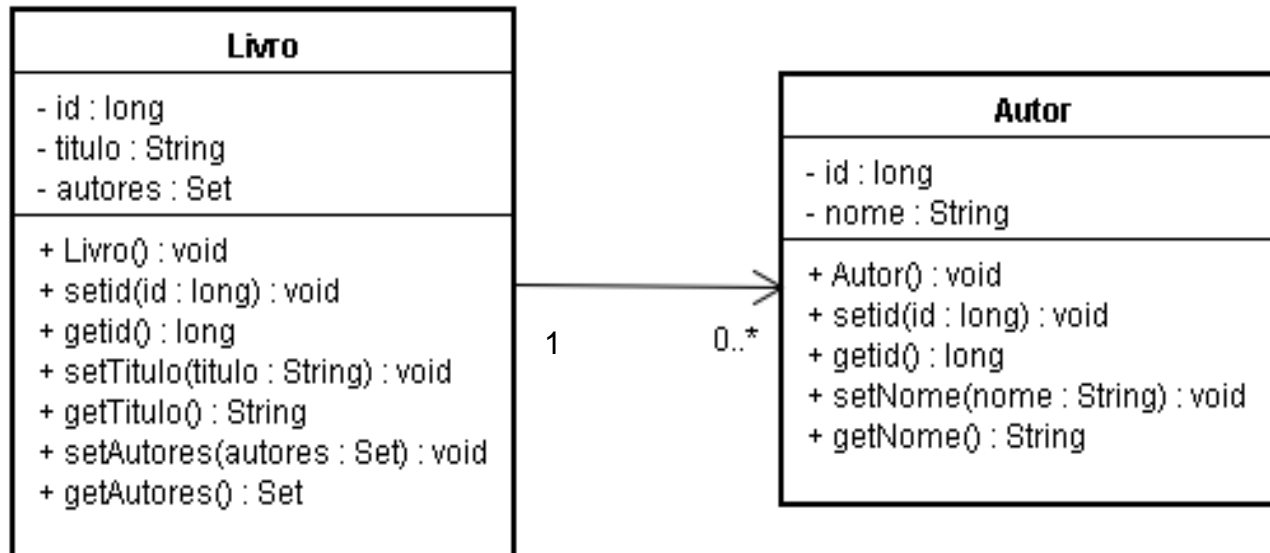
Mapeamento 1xN

➔ Considere a relação livro-autor:

- Um livro pode ser escrito por vários autor;
- Um autor, por sua vez, pode escrever um livro

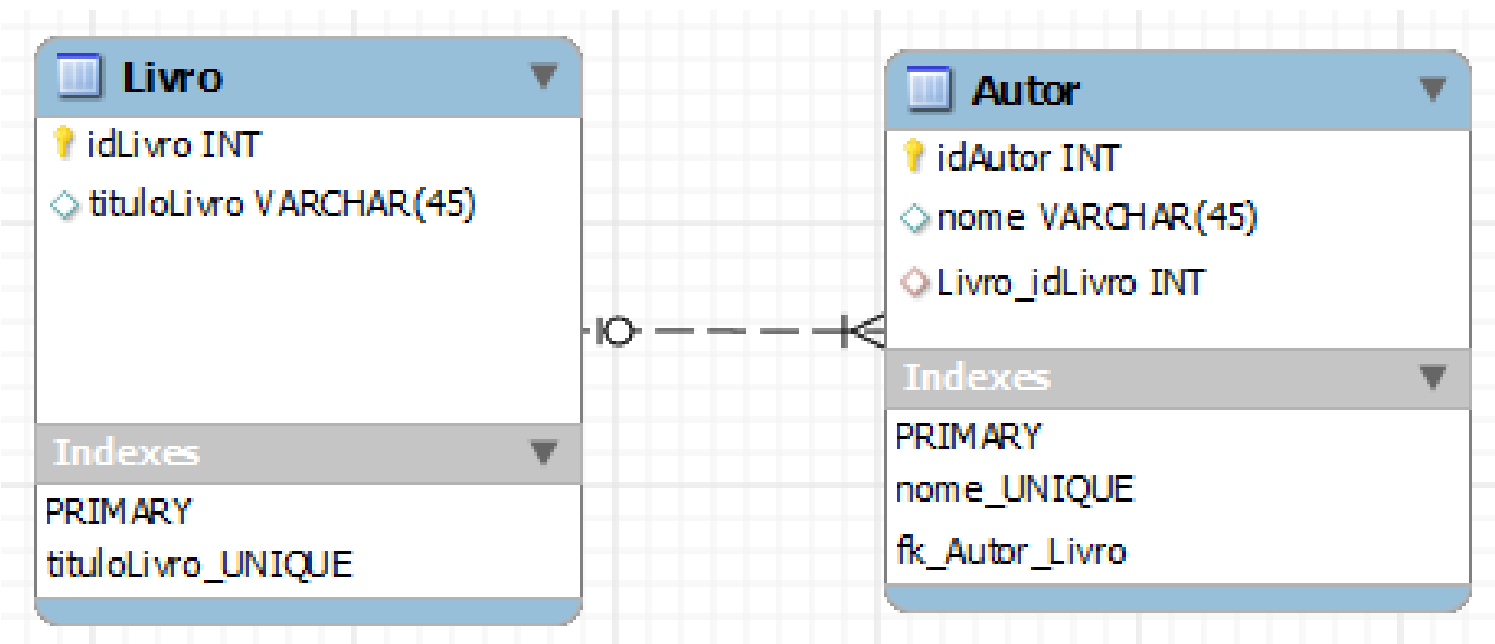
➔ Na implementação de 1-para-N em OO...

- Dependendo da modelagem, basta definir uma coleção como atributo do tipo em questão na outra classe



Mapeamento 1xN

➔ Diagrama Relacional criado utilizando o MySQL Workbench



➔ A questão agora é: como fazer o mapeamento objeto-relacional dessas associações no código??

MySQL Workbench



Livro Autor 1xN



Data Modeling

Create and manage models, forward & reverse engineer, compare and synchronize schemas, report.



Open Existing EER Model

Or select a model to open or click here to browse...



Create New EER Model

Create a new EER Model from scratch.

Description Editor

No Selection

Model Overview

Add Diagram

Physical Schemata

mydb
MySQL Schema

Tables (0 items)

Add Table

Views (0 items)

Add View

Routines (0 items)

Add Routine

Routine Groups (0 items)

Add Group

Schema Privileges

SQL Scripts

Model Notes

User Types List

Name	Definition	Flags
BOOL	TINYINT(1)	
BOOLE...	TINYINT(1)	
FIXED	DECIMAL(10 ...	
FLOAT4	FLOAT	
FLOAT8	DOUBLE	
INT1	TINYINT(4)	
INT2	SMALLINT(6)	
INT3	MEDIUMINT...	
INT4	INT(11)	

Model Overview

Add Diagram

Physical Schemata

mysqlLivroAutorNxN
MySQL Schema

Tables (0 items)

Add Table

Views (0 items)

Add View

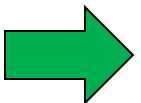
mysqlLivroAutorNxN

Name: mysqllivroautor 1xN

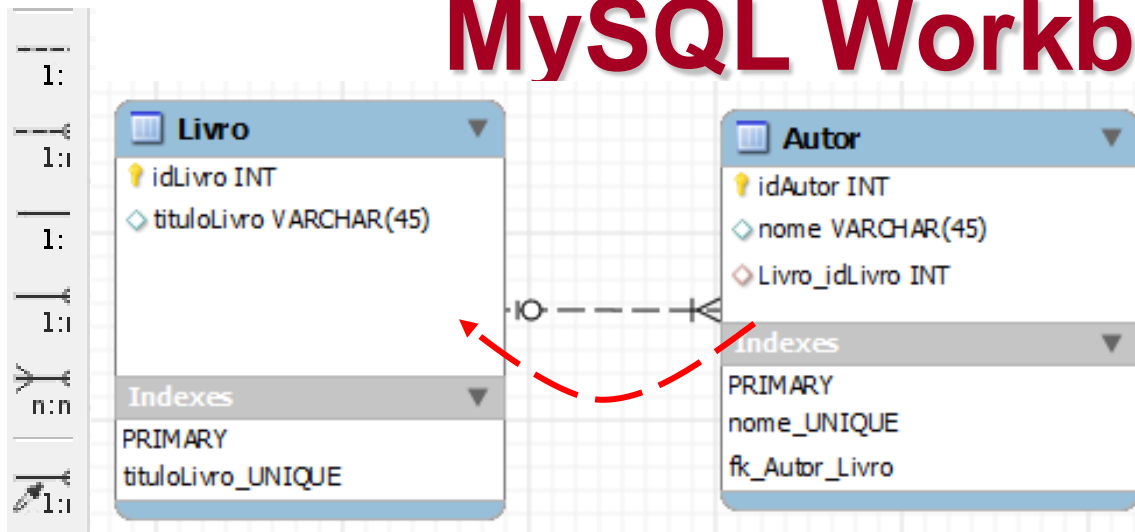
Collation: utf8 - utf8_general_ci

Comments:

- Cut 'mysqlLivroAutorNxN'
- Copy 'mysqlLivroAutorNxN'
- Paste
- Edit Schema...
- Edit in New Window...
- Copy SQL to Clipboard
- Delete 'mysqlLivroAutorNxN'



MySQL Workbench



Livro

Column Name	Datatype	PK	NN	UQ
idLivro	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
tituloLivro	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Table Columns Indexes Foreign Keys Triggers Partitions

Autor

Column Name	Datatype	PK	NN	UQ
idAutor	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
nome	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Livro_idLivro	INT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Table Columns Indexes Foreign Keys Triggers

Autor

Index Name	Type
PRIMARY	PRIMARY
fk_Autor_Livro	INDEX
nome_UNIQUE	UNIQUE

Table Columns Indexes Foreign Keys Triggers

Foreign Key Name

fk_Autor_Livro

Referenced Table

mysqllivroautor1x1, Livro

Foreign Key Options

On Update: CASCADE

On Delete: CASCADE

☐ Skip in SQL generation

Foreign Key Comment

Table Columns Indexes Foreign Keys Triggers

MySQL Workbench

➡ Curiosidade: SQL Gerado...

```
-- Table `mysqllivroautor1xN`.`Livro`
```

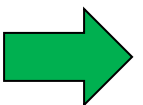
```
DROP TABLE IF EXISTS `mysqllivroautor1xN`.`Livro` ;
```

```
CREATE TABLE IF NOT EXISTS `mysqllivroautor1xN`.`Livro` (  
  `idLivro` INT NOT NULL AUTO_INCREMENT,  
  `tituloLivro` VARCHAR(45) NULL,  
  PRIMARY KEY (`idLivro`),  
  UNIQUE INDEX `tituloLivro_UNIQUE` (`tituloLivro` ASC),  
  ENGINE = InnoDB;
```

```
-- Table `mysqllivroautor1xN`.`Autor`
```

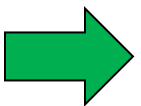
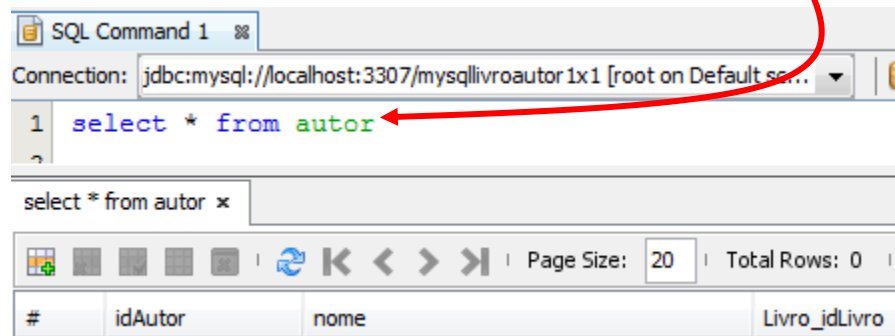
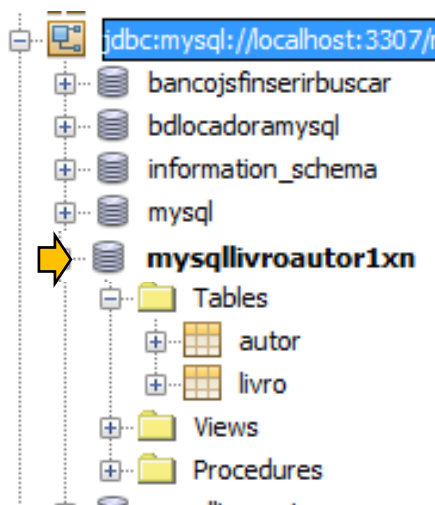
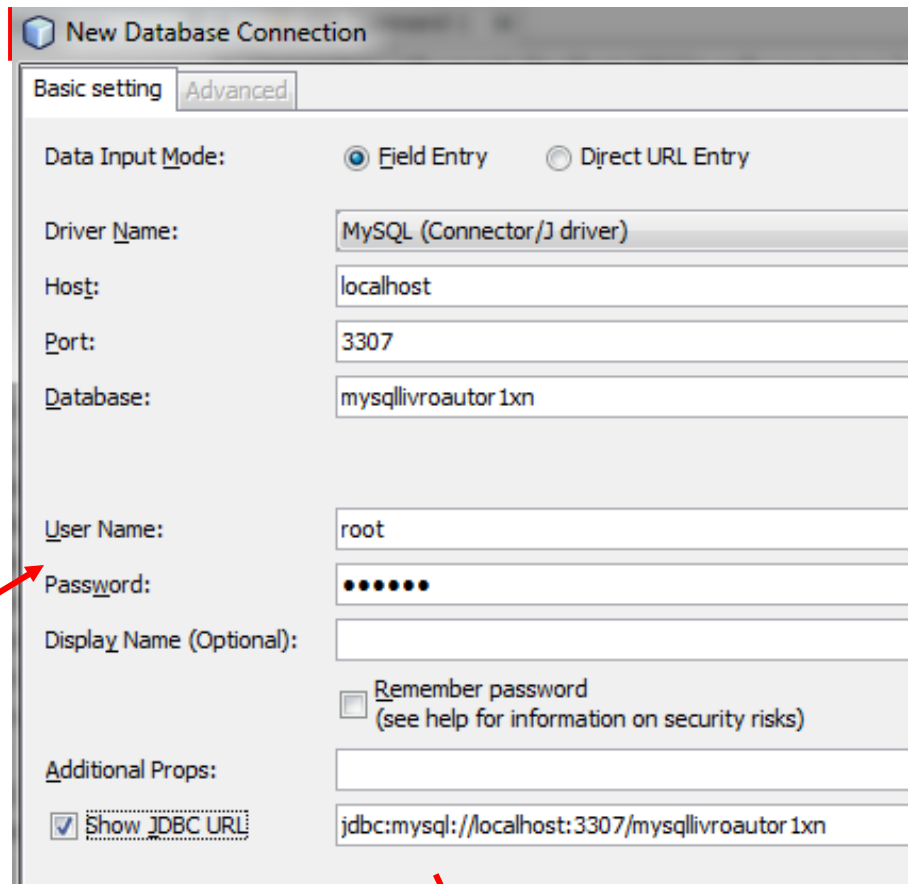
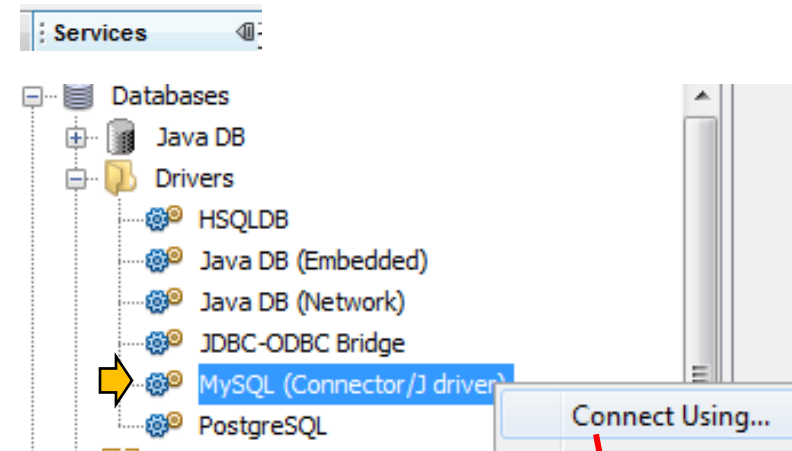
```
DROP TABLE IF EXISTS `mysqllivroautor1xN`.`Autor` ;
```

```
CREATE TABLE IF NOT EXISTS `mysqllivroautor1xN`.`Autor` (  
  `idAutor` INT NOT NULL AUTO_INCREMENT,  
  `nome` VARCHAR(45) NULL,  
  `Livro_idLivro` INT NULL,  
  PRIMARY KEY (`idAutor`),  
  UNIQUE INDEX `nome_UNIQUE` (`nome` ASC),  
  INDEX `fk_Autor_Livro` (`Livro_idLivro` ASC),  
  CONSTRAINT `fk_Autor_Livro`  
    FOREIGN KEY (`Livro_idLivro`)  
    REFERENCES `mysqllivroautor1xN`.`Livro` (`idLivro`)  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION)  
  ENGINE = InnoDB;
```



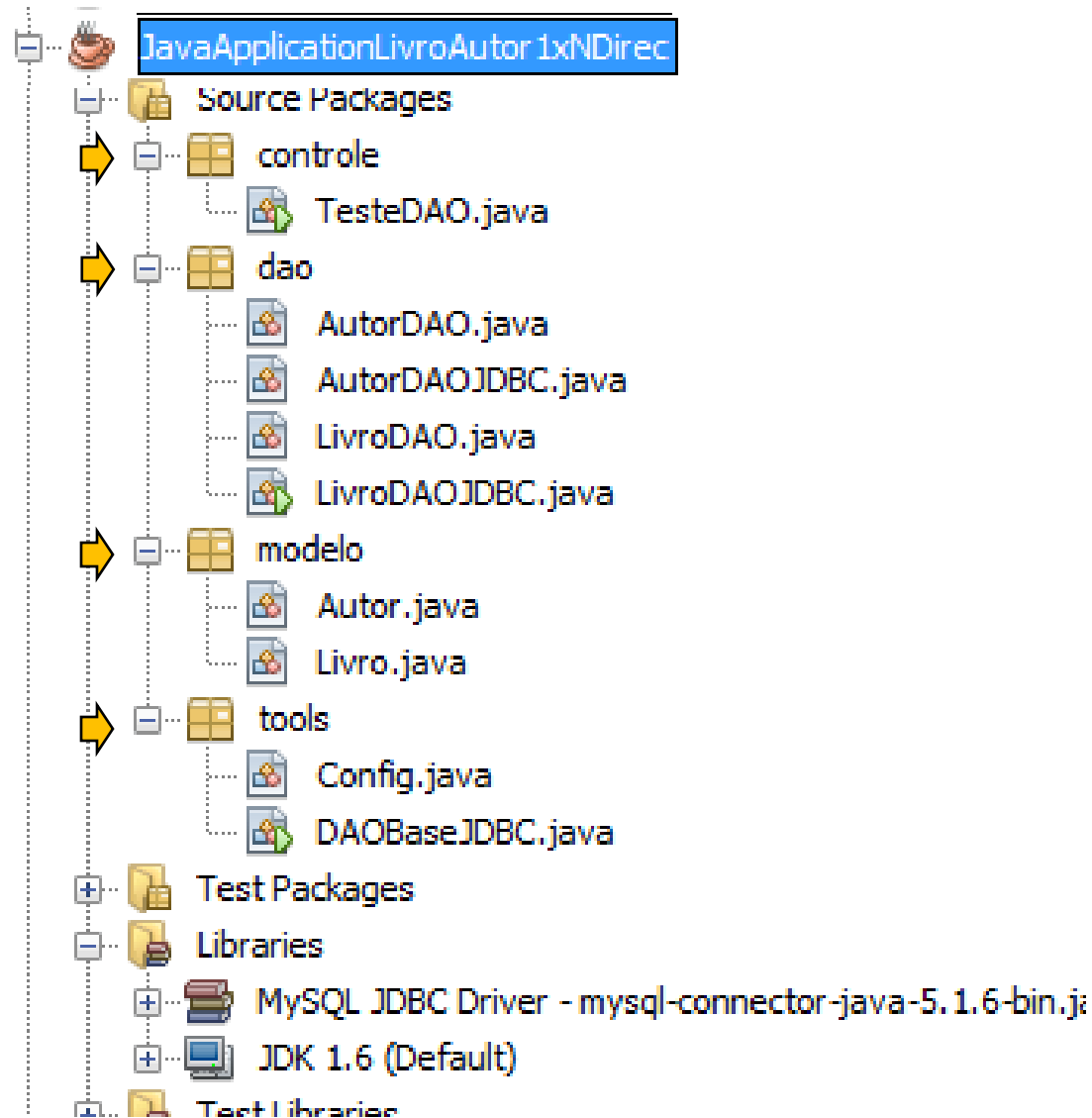
JDBC e NetBeans

➔ Testando conexão com o



JDBC e NetBeans

➡ Montando o projeto... no final teremos a seguinte estrutura



Mapeamento código OO x Relacional

➡ Primeiro vamos criar uma classe que será responsável pela conexão com o banco

```
public class DAOBaseJDBC {
```

DAOBaseJDBC.java

```
protected static Connection conn;
```

```
static {
```

```
try {
```

```
// Carrega o driver do MySQL e conecta.
```

```
Class.forName(Config.NOME_DRIVER);
```

```
conn = DriverManager.getConnection(Config.BD_URL, Config.BD_LOGIN, Config.BD_SENHA);
```

```
} catch (ClassNotFoundException e) {
```

```
System.out.println("FATAL: driver não encontrado.");
```

```
System.exit(1);
```

```
} catch (SQLException e) {
```

```
System.out.println("Erro SQL: " + e.getMessage());
```

```
System.exit(1);
```

```
}
```

```
}
```

```
public static void closeConn() {
```

```
try {
```

```
if (conn != null)
```

```
conn.close();
```

```
} catch (SQLException ex) {
```

```
Logger.getLogger(DAOBaseJI
```

```
}
```

```
}
```

Config.java

```
public interface Config {
```

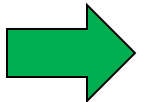
```
public static final String NOME_DRIVER = "com.mysql.jdbc.Driver";
```

```
public static final String BD_URL = "jdbc:mysql://localhost:3307/mysql livro autor1xn";
```

```
public static final String BD_LOGIN = "root";
```

```
public static final String BD_SENHA = "123456";
```

```
}
```



Mapeamento código OO x Relacional

➔ Agora vamos construir as classes de entidades do domínio:
Livro e Autor

Autor.java

```
public class Autor {  
    private Long id;  
    private String nome;  
  
    public Autor() {...}  
    public Autor(Long id) {...}  
    public Long getId() {...}  
    public void setId(long id) {...}  
    public String getNome() {...}  
    public void setNome(String nome) {...}  
    @Override  
    public String toString() {...}  
    @Override  
    public boolean equals(Object obj) {...}  
    @Override  
    public int hashCode() {...}  
}
```

Livro.java

```
public class Livro {  
    private Long id;  
    private String titulo;  
    Set<Autor> autores;  
  
    public Livro() {  
        this.autores = new HashSet<Autor>();  
    }  
    //Inserir métodos para adicionar e remover objetos na coleção  
    public void adicionarAutor(Autor autor) {  
        this.getAutores().add(autor);  
    }  
    public void removerConta(Autor autor) {  
        this.getAutores().remove(autor);  
    }  
  
    public Set<Autor> getAutores() {...}  
    public void setAutores(Set<Autor> autores) {...}  
    public Long getId() {...}  
    public void setId(Long id) {...}  
    public String getTitulo() {...}  
    public void setTitulo(String titulo) {...}  
    @Override  
    public String toString() {...}  
}
```

Mapeamento código OO x Relacional

➡ Agora vamos construir o **AutorDAO**, responsável pela persistência dos objetos do tipo **Autor**

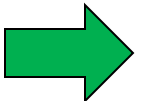
```
AutorDAOJDBC.java
public class AutorDAOJDBC extends DAOBaseJDBC implements AutorDAO {

    public boolean salvar(Autor autor) {
        PreparedStatement stmt;
        try {
            if (autor.getId() == null) {
                System.out.println("=== salvando..... === ");
                stmt =
                    conn.prepareStatement("INSERT INTO Autor (nome) VALUES (?)");
            } else {
                System.out.println("=== atualizando..... === ");
                stmt =
                    conn.prepareStatement("UPDATE Autor SET nome = ? WHERE idAutor = ?");
                stmt.setLong(2, autor.getId());
            }

            stmt.setString(1, autor.getNome());
            stmt.executeUpdate(); stmt.close();
        } catch (SQLException e) {
            System.out.println("Erro SQL: " + e.getMessage());
            System.out.println("Autor não gravado pois já esta na base de dados.....: " + autor);
            return false;
        }
        return true;
    }
}
```

AutorDAO.java

```
public interface AutorDAO {
    public boolean salvar(Autor artista);
    public Autor consultar(String nome);
}
```



Mapeamento código OO x Relacional

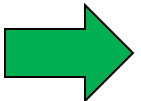
➡ Agora vamos construir o **AutorDAO**, responsável pela persistência dos objetos do tipo **Autor**

AutorDAOJDBC.java

```
public Autor consultar(String nome) {  
    Autor autorLido = null;  
    try {  
        PreparedStatement stmt =  
            conn.prepareStatement(  
                "SELECT idAutor, nome FROM Autor WHERE nome = ?");  
        stmt.setString(1, nome);  
        ResultSet rset = stmt.executeQuery();  
        if (rset.next()) {  
            autorLido = new Autor();  
            autorLido.setId(new Long(rset.getLong("idAutor")));  
            autorLido.setNome(rset.getString("nome")); stmt.close();  
        }  
        else  
            return null;  
    } catch (SQLException e) {  
        System.out.println("Falha na consulta: " + e.getMessage());  
        return null;  
    }  
    return autorLido;  
}
```

AutorDAO.java

```
public interface AutorDAO {  
    public boolean salvar(Autor artista);  
    public Autor consultar(String nome);  
}
```



Mapeamento código OO x Relacional

➡ Agora vamos construir o LivroDAO, responsável pela persistência dos objetos do tipo Livro

```
public class LivroDAOJDBC extends DAOBaseJDBC implements LivroDAO {
```

```
    public boolean salvar(Livro livro) {  
        PreparedStatement stmt = null;  
        try {
```

```
            conn.setAutoCommit(false);
```

```
            if (livro.getId() == null) {
```

```
                stmt = conn.prepareStatement("INSERT INTO Livro (tituloLivro) VALUES (?)");
```

```
            } else {
```

```
                stmt = conn.prepareStatement("UPDATE Livro SET tituloLivro = ? WHERE idLivro = ?");
```

```
                stmt.setLong(2, livro.getId());
```

```
            }
```

```
            stmt.setString(1, livro.getTitulo());
```

```
            stmt.executeUpdate(); stmt.close();
```

```
            this.gravarAutores(livro);
```

```
            conn.commit(); conn.setAutoCommit(true);
```

```
        } catch (SQLException e) {
```

```
            System.out.println("Erro SQL: " + e.getMessage());
```

```
            try {
```

```
                conn.rollback(); conn.setAutoCommit(true);
```

```
            } catch (SQLException ex) { System.out.println("Erro no rollback...");}
```

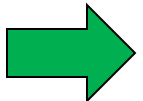
```
            return false;
```

```
        }
```

```
        return true;
```

```
    }
```

```
public interface LivroDAO {  
    public boolean salvar(Livro livro);  
    public Livro consultar(String titulo);  
}
```



Mapeamento código OO x Relacional

➡ Agora vamos construir o LivroDAO, responsável pela persistência dos objetos do tipo Livro

LivroDAOJDBC.java

```
// pré: o Autor associado ao livro já possuiu o id recuperado...
// papel do controle neste exemplo
private void gravarAutores(Livro livro) {
    String sql = "UPDATE Autor SET Livro_idLivro = ? WHERE idAutor = ?";
    PreparedStatement stmt;
    try {
        stmt = conn.prepareStatement(sql);

        stmt.setInt(1, this.lerIdLivro());

        Set<Autor> autores = livro.getAutores();
        for (Autor autor : autores) {
            stmt.setLong(2, autor.getId());
            stmt.executeUpdate();
        }

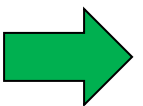
        stmt.close();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

LivroDAO.java

```
public interface LivroDAO {
    public boolean salvar(Livro livro);
    public Livro consultar(String titulo);
}
```

```
private int lerIdLivro() {
    String sql = "SELECT MAX(idLivro) FROM livro";
    PreparedStatement stmt = null;
    int idLivro = 0;
    try {
        stmt = conn.prepareStatement(sql);
        ResultSet rs = stmt.executeQuery();
        rs.next();
        idLivro = rs.getInt(1); stmt.close();
    } catch (SQLException e) {
        e.printStackTrace();
        System.out.println("Não foi possível recuperar o MAX idLivro");
    }

    return idLivro;
}
```



Mapeamento código OO x Relacional

➡ Agora vamos construir o LivroDAO, responsável pela persistência dos objetos do tipo Livro

```
public Livro consultar(String titulo) {  
    Livro livroLido = null;  
    try {
```

LivroDAOJDBC.java

```
        PreparedStatement stmt =  
            conn.prepareStatement(  
                "SELECT idLivro, tituloLivro FROM Livro WHERE tituloLivro = ?");  
        stmt.setString(1, titulo);
```

```
        ResultSet rset = stmt.executeQuery();  
        if (rset.next()) {  
            livroLido = new Livro();  
            livroLido.setId(new Long(rset.getLong("idLivro")));  
            livroLido.setTitulo(rset.getString("tituloLivro"));
```

```
            Set<Autor> autores = this.lerAutores(livroLido.getId());  
            livroLido.setAutores(autores);
```

```
        }    stmt.close();
```

```
        else
```

```
            return null;
```

```
    } catch (SQLException e) {
```

```
        System.out.println("Falha na consulta: " + e.getMessage());
```

```
        return null;
```

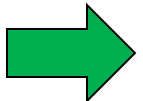
```
    }
```

```
    return livroLido;
```

```
}
```

LivroDAO.java

```
public interface LivroDAO {  
    public boolean salvar(Livro livro);  
    public Livro consultar(String titulo);  
}
```



Mapeamento código OO x Relacional

➔ Agora vamos construir o LivroDAO, responsável pela persistência dos objetos do tipo Livro

```
private Set<Autor> lerAutores(long idLivro) {
```

```
    String sql = "SELECT Autor.idAutor, Autor.nome "  
        + "FROM Livro, Autor "  
        + "WHERE Livro.idLivro = ? AND "  
        + "Livro.idLivro = Autor.Livro_idLivro";
```

```
    PreparedStatement stmt = null;
```

```
    Set<Autor> autores = null;
```

```
    try {
```

```
        stmt = conn.prepareStatement(sql);  
        stmt.setLong(1, idLivro);  
        ResultSet rset = stmt.executeQuery();
```

```
        autores = new HashSet<Autor>();
```

```
        while (rset.next()) {  
            Autor autorLido = new Autor();  
            autorLido.setId(rset.getInt("idAutor"));  
            autorLido.setNome(rset.getString("nome"));  
            autores.add(authorLido);  
        } stmt.close();
```

```
    } catch (SQLException e) {
```

```
        System.out.println("Falha na consulta para recuperar "  
            + e.printStackTrace();  
        return null;
```

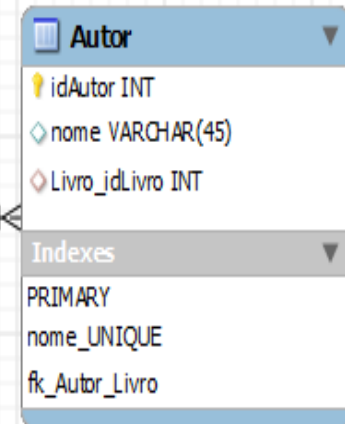
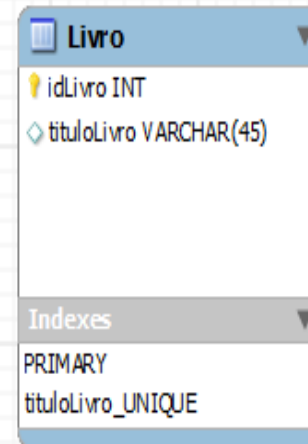
```
    }
```

```
    return autores;
```

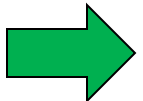
```
}
```

LivroDAO.java

```
public interface LivroDAO {  
    public boolean salvar(Livro livro);  
    public Livro consultar(String titulo);  
}
```





	Livro_idLivro	Autor_idAutor
▶	1	1
	1	2
	2	2



Mapeamento código OO x Relacional

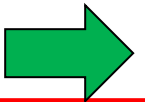
➔ Pra fechar, podemos construir uma classe simulando as chamadas do controle à estrutura construída...

`public class TesteDAO {`  TesteDAO.java 

```
public static void main(String args[]) {  
    LivroDAOJDBC livroDAO = new LivroDAOJDBC();  
    AutorDAOJDBC autorDAO = new AutorDAOJDBC();
```

```
    System.out.println("=====");  
    System.out.println("Gravando Autores.....");  
    System.out.println("=====");  
    Autor autor01 = new Autor();  
    autor01.setNome("Vinicius");  
    autorDAO.salvar(autor01);  
  
    Autor autor02 = new Autor();  
    autor02.setNome("Rosalen");  
    autorDAO.salvar(autor02);  
  
    Autor autor03 = new Autor();  
    autor03.setNome("Vinicios");  
    autorDAO.salvar(autor03);  
  
    Autor autor04 = new Autor();  
    autor04.setNome("Victor");  
    autorDAO.salvar(autor04);
```

```
    System.out.println("=====");  
    System.out.println("Gravando livros.....");  
    System.out.println("=====");  
    Livro livro01 = new Livro();  
    livro01.setTitulo("Java para vida");  
    autor01.setId((autorDAO.consultar("Vinicius")).getId());  
    autor03.setId((autorDAO.consultar("Vinicios")).getId());  
    livro01.adicionarAutor(autor01);  
    livro01.adicionarAutor(autor03);  
    livroDAO.salvar(livro01);  
  
    Livro livro02 = new Livro();  
    livro02.setTitulo("Opa, agora vai!");  
    autor02.setId((autorDAO.consultar("Rosalen")).getId());  
    autor04.setId((autorDAO.consultar("Victor")).getId());  
    livro02.adicionarAutor(autor02);  
    livro02.adicionarAutor(autor04);  
    livroDAO.salvar(livro02);
```



Mapeamento código OO x Relacional

➡ Pra fechar, podemos construir uma classe simulando as chamadas do controle à estrutura construída...

TesteDAO.java

```
Autor autorDaPesquisa = null;
System.out.println("=====");
System.out.println("Recuperando autores...");
System.out.println("=====");

autorDaPesquisa = autorDAO.consultar("Vinicius");
System.out.println(autorDaPesquisa);

autorDaPesquisa = autorDAO.consultar("Rosalen");
System.out.println(autorDaPesquisa);

autorDaPesquisa = autorDAO.consultar("Vinicios");
System.out.println(autorDaPesquisa);

autorDaPesquisa = autorDAO.consultar("Victor");
System.out.println(autorDaPesquisa);
```

```
Livro livroDaPesquisa = null;
System.out.println("=====");
System.out.println("Recuperando Livros...");
System.out.println("=====");

livroDaPesquisa = livroDAO.consultar("Java para vida");
System.out.println(livroDaPesquisa);

livroDaPesquisa = livroDAO.consultar("Opa, agora vai!");
System.out.println(livroDaPesquisa);
```

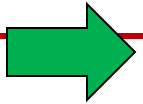
Exercícios

- Blz... Agora é hora de exercitar um pouco mais.....
- Tente resolver os seguintes problemas...
 - Usar o NetBeans...
 - Em dupla



Exercícios

- Implemente as seguintes funcionalidades para cada caso (1x1 e 1xN)...
 - Alterar e Excluir Autor
 - Alterar e Excluir Livro



Exercícios

- Altere o modelo e código para incluir mais atributos em cada entidade
 - Por exemplo, Autor com campos Sexo e Telefone e Livro com ISBN e Número de páginas
- Implemente outros métodos de Busca para Livro e Autor para cada caso (1x1 e 1xN)...
 - Use a criatividade...

Exercícios

- Faça as regras 1xN de JavaDiscos no MySQL
 - Artista1xNCD e Gravadora1xNCD

