

# Sistema para aprendizado de Álgebra Relacional e Linguagem SQL

André Martins dos Santos

Prof. Roosevelt dos Santos Junior, Orientador



# Roteiro da apresentação

- ◆ Introdução
  - ◆ Fundamentação teórica
  - ◆ Especificação
  - ◆ Implementação
  - ◆ Operacionalidade
  - ◆ Resultados e discussão
  - ◆ Conclusões
  - ◆ Extensões
-

# Introdução



# Objetivos do trabalho

Disponibilizar um sistema para aprendizado de álgebra relacional e linguagem SQL (Processos básicos de consulta).

- ◆ Disponibilizar um editor gráfico para construção de árvores de expressões algébricas
  - ◆ Fazer a conversão das árvores de expressões algébricas em linguagem SQL
  - ◆ Executar a linguagem SQL nos bancos de dados relacionais Oracle e MySQL
-

# Álgebra Relacional

- ◆ Linguagem formal de consulta associada ao modelo relacional
  - ◆ Construída utilizando uma coleção de operadores
  - ◆ Expressa através de expressões algébricas ou árvores de expressões
-

# Expressões Algébricas

- Sequência de operações de álgebra relacional

Seja a relação StarsIn (title,year,starName) :

title	year	starName
A Caçada	1996	Laurence Fishburn
A Vida é Bela	1997	Roberto Benigni
A Negociação	1998	Samuel L. Jackson

Expressão:  $\pi_{\text{starName}} (\sigma_{\text{year} = 1996} (\pi_{\text{starName,year}} (\text{StarsIn})))$

Resultado:

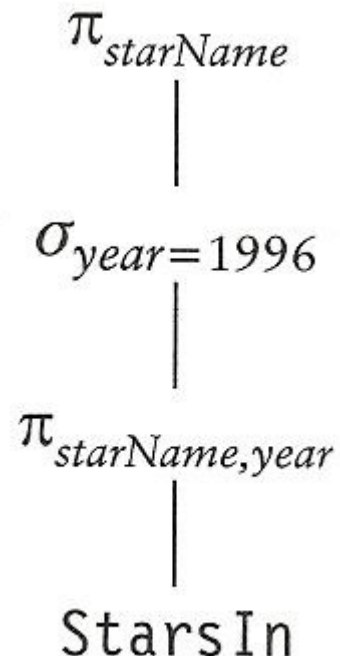
starName
Laurence Fishburn

# Árvores de expressões

- Representação de uma expressão algébrica em formato de árvore onde as folhas são as relações e cada um dos interiores é identificado por um operador

- $\pi_{\text{starName}} (\sigma_{\text{year} = 1996} (\pi_{\text{starName,year}} (\text{StarsIn})))$

- Árvore de expressão:



# Principais requisitos

- ◆ Permitir a construção de árvores de expressão algébricas utilizando *drag-and-drop*



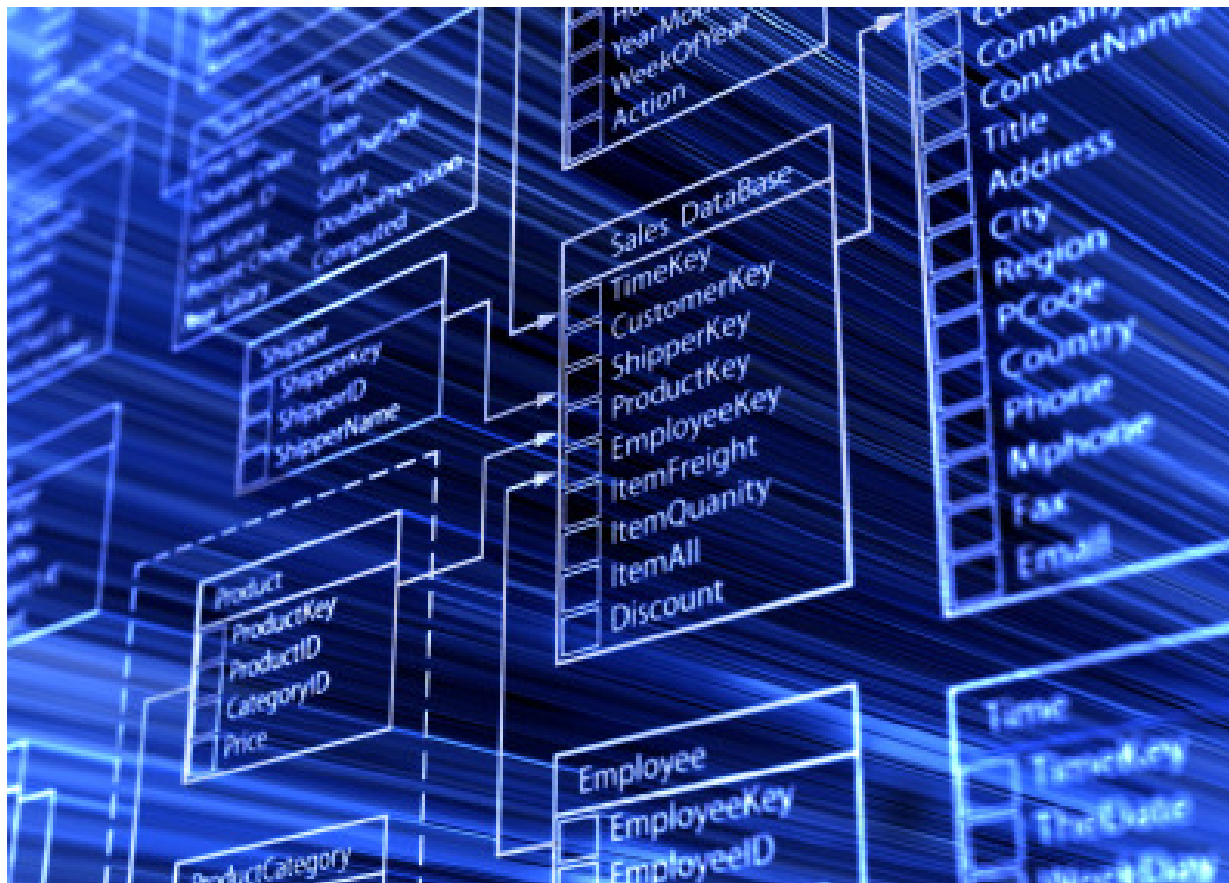
- ◆ Suportar os operadores: União, Diferença, Projeção, Seleção, Junção Theta, Produto Cartesiano, Agrupamento/Agregação e Reordenação

$\cup$     $-$     $\pi$     $\sigma$     $\bowtie_c$     $\times$     $\gamma$     $\tau$



# Principais requisitos

- Listar as relações de *schemas* do banco o qual esta conectado



# Principais requisitos

- Transformar as árvores em comandos SQL



# Principais requisitos

- Executar os comandos SQL nos bancos de dados Oracle XE e MySQL

**ORACLE**<sup>®</sup>  
DATABASE  
EXPRESS EDITION

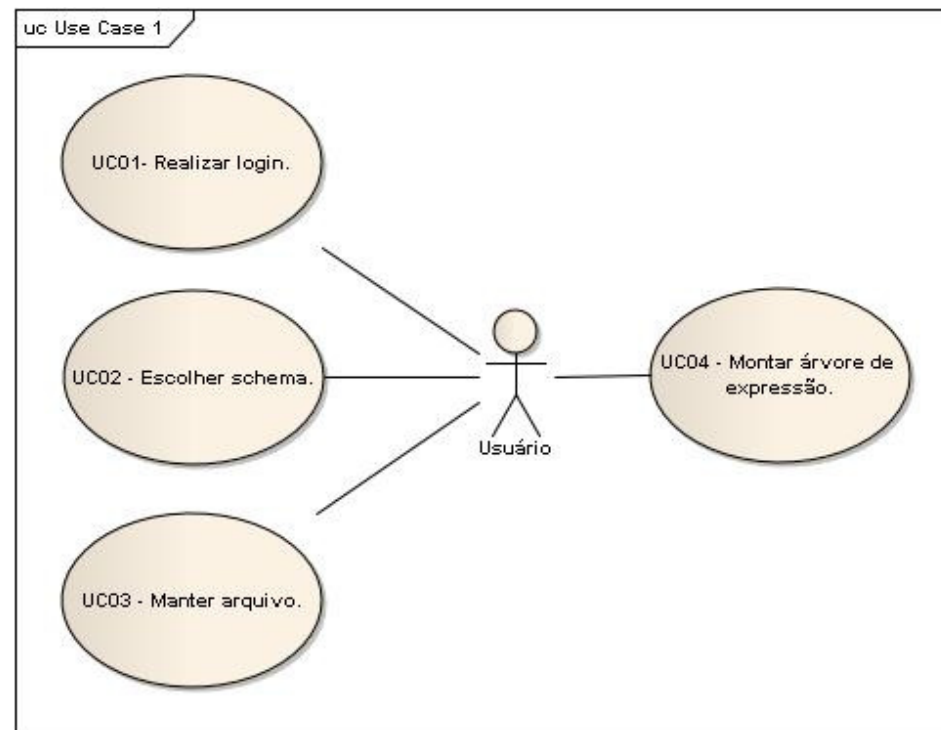


# Especificação da ferramenta



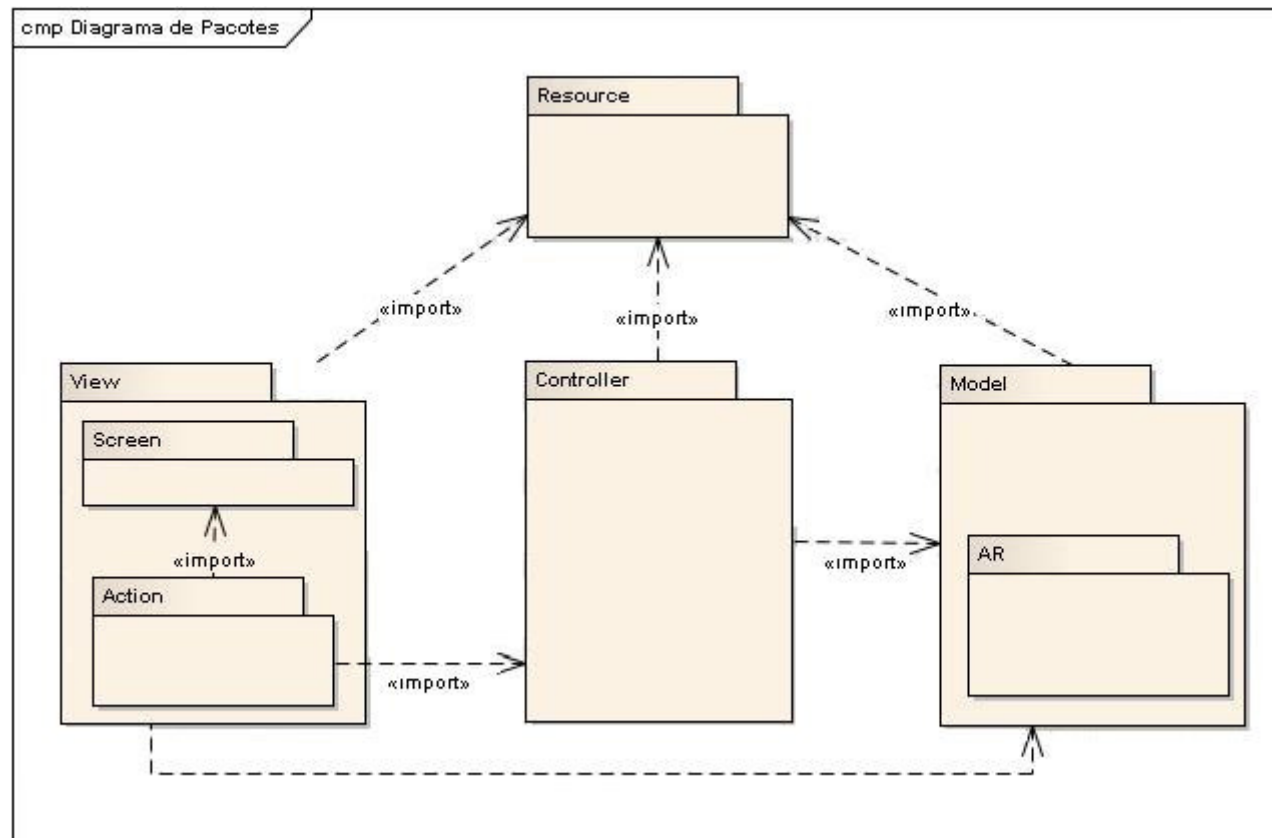
# Especificação da ferramenta

## ◆ Diagrama de Caso de Uso



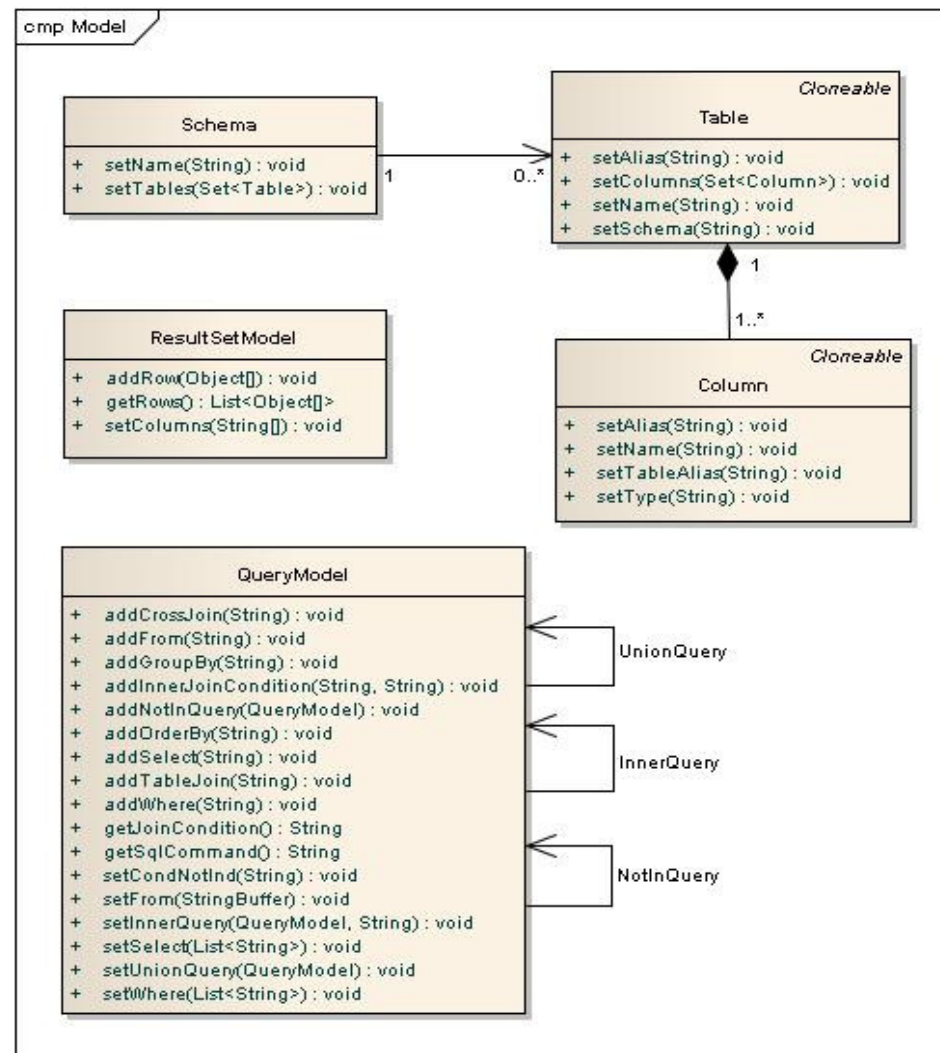
# Especificação da ferramenta

## ◆ Diagramas de pacotes



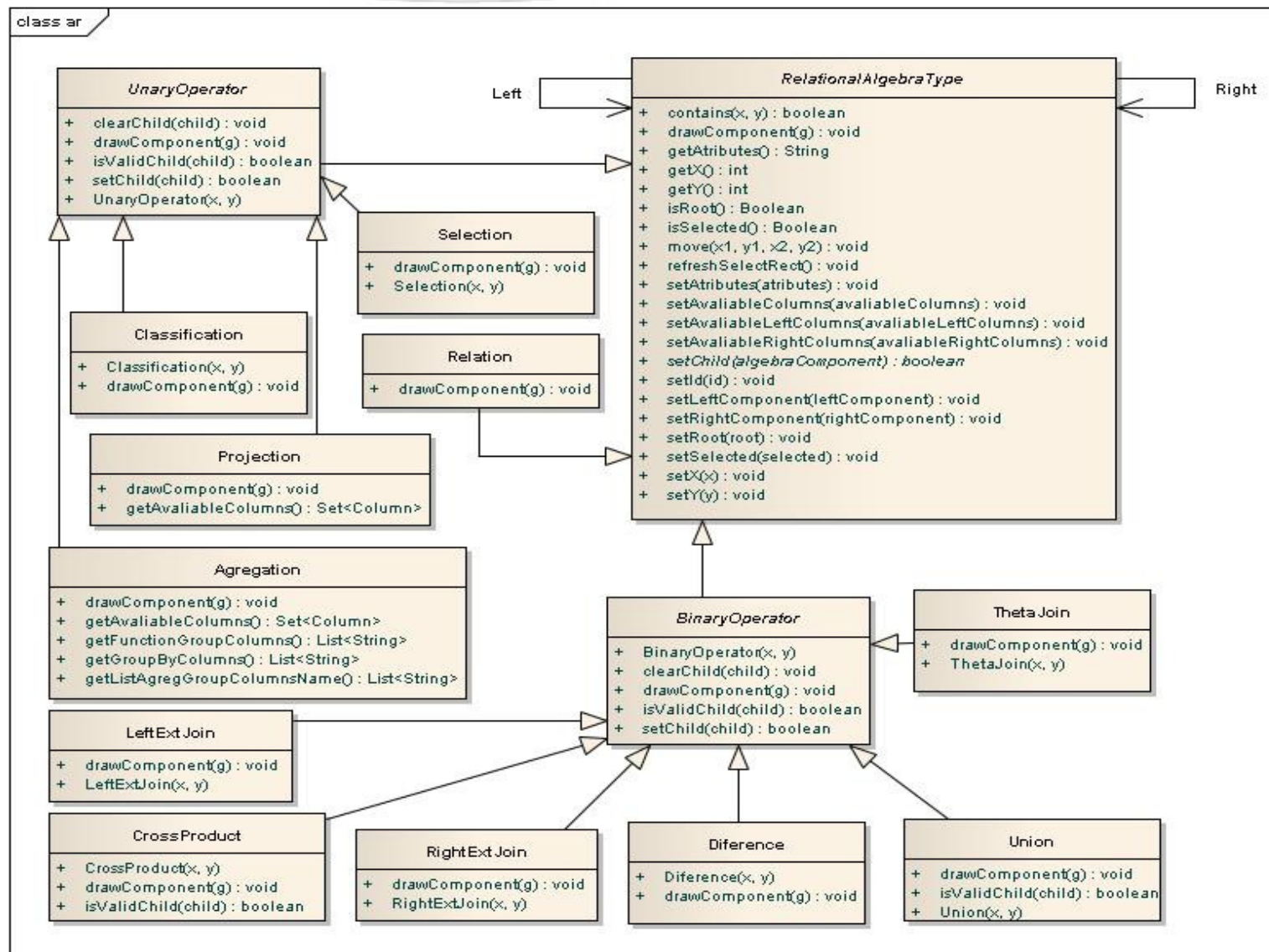
# Especificação da ferramenta

## Diagramas de classes Model



# Especificação da ferramenta

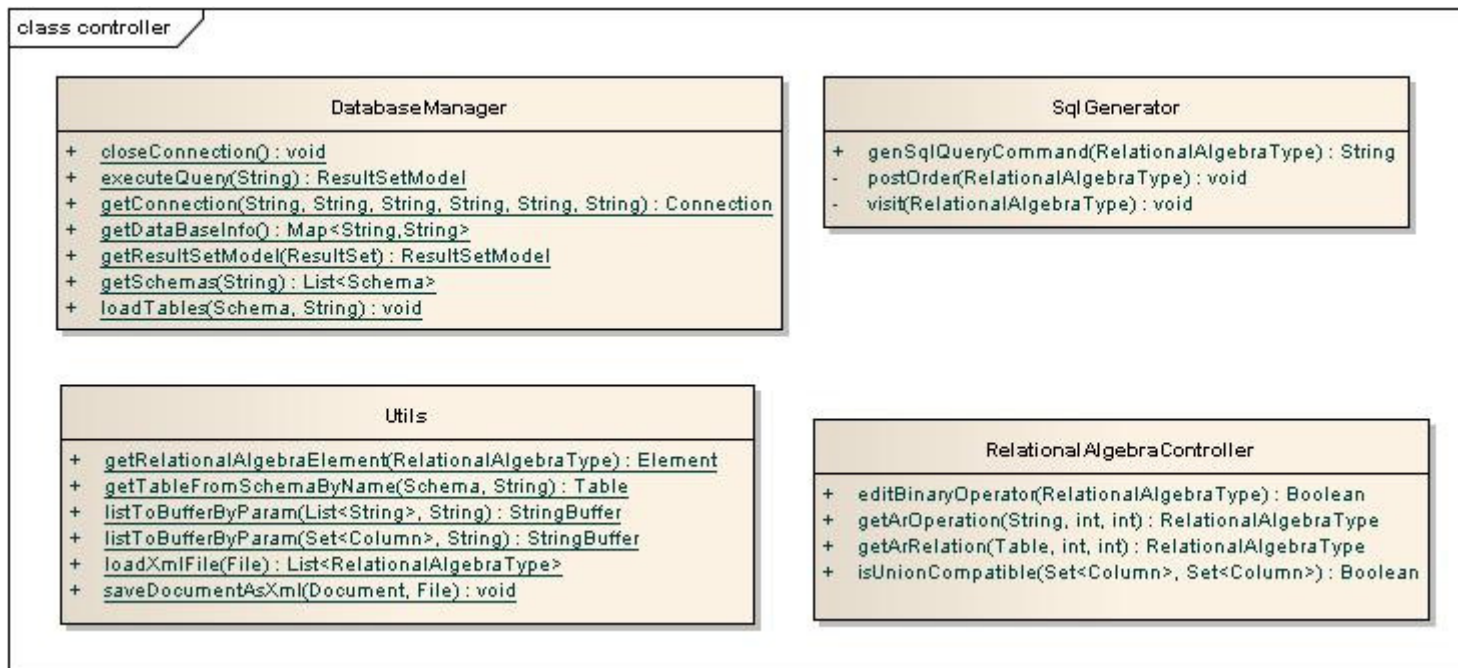
## Diagramas de classes AR





# Especificação da ferramenta

## ◆ Diagramas de classes Controller



# Implementação da ferramenta

- o Técnicas e ferramentas utilizadas
  - ✓ Java versão 6
  - ✓ Java Database Connectivity (***JDBC***)
  - ✓ Eclipse



# Implementação da ferramenta

## ◆ Classe DatabaseManager

### Leitura dos schemas

```
89 public static List<Schema> getSchemas(String dataBase) throws Exception{
90     List<Schema> listSchema = new ArrayList<Schema>();
91     DatabaseMetaData dbmdt = (DatabaseMetaData) connection.getMetaData();
92     ResultSet rsSchemas = null;
93     String schemaColumn = "";
94
95     if(dataBase.equals(Constants.ORACLE_DB)){
96         rsSchemas = dbmdt.getSchemas();
97         schemaColumn = "TABLE_SCHEM";
98     }else if (dataBase.equals(Constants.MYSQL_DB)){
99         rsSchemas = dbmdt.getCatalogs();
100        schemaColumn = "TABLE_CAT";
101    }
102
103    String tableCat;
104    Schema schema = null;
105
106    while(rsSchemas.next()){
107        tableCat = rsSchemas.getString(schemaColumn);
108        schema = new Schema(tableCat);
109        listSchema.add(schema);
110    }
111
112    return listSchema;
113 }
```

# Implementação da ferramenta

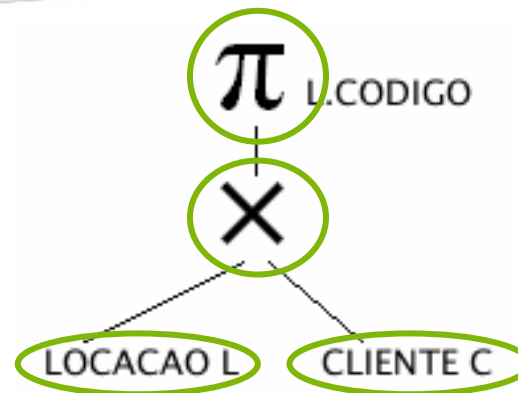
Geração da linguagem SQL

- ◆ Classe QueryModel
- ◆ Classe SqlGenerator

```
27 public String genSqlQueryCommand(RelationalAlgebraComponent root) {  
28     postOrder(root);  
29     return sqlCommand;  
30 }  
31  
32 private void postOrder(RelationalAlgebraComponent n) {  
33     if (n != null) {  
34         postOrder(n.getLeftComponent());  
35         postOrder(n.getRightComponent());  
36         visit(n);  
37     }  
38 }
```

# Implementação da ferramenta

Algoritmo de geração da linguagem SQL

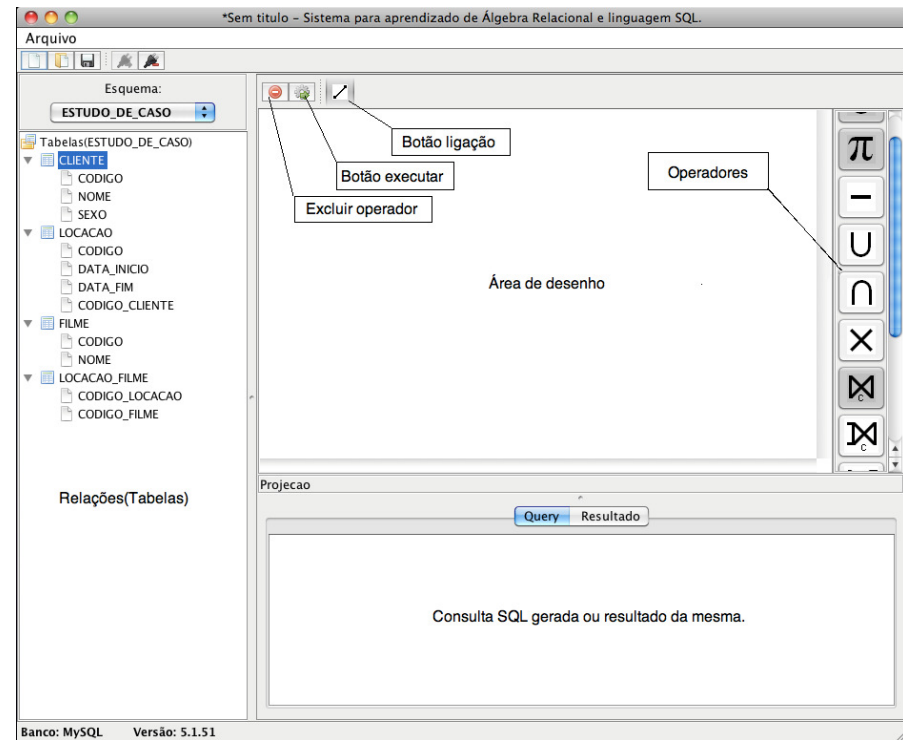
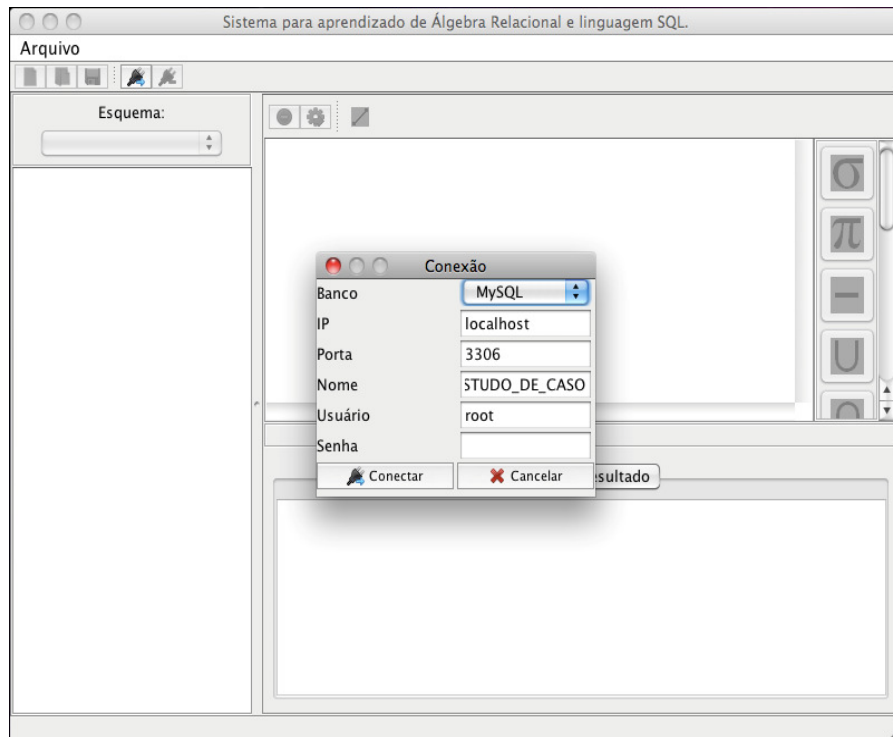


Pilha

$M(\pi L \times C)$
$M(L \times C)$
$M(C)$
$M(L)$

- ✓ Criado modelo  $M(L)$  para  $LOCACAO L$ 
  - ✓ Criado modelo  $M(C)$  para  $CLIENTE C$
  - ✓ Desempilhado os modelos  $M(C)$  e  $M(L)$ , aplicado o Produto ao  $M(L)$  e atribuído os atributos de  $M(C)$ .
  - ✓ Desempilhado o modelo  $M(L \times C)$  e aplicado a Projeção.

# Operacionalidade



# Operacionalidade

The screenshot shows a software interface for learning relational algebra and SQL. The main window displays a query tree diagram. The root node is a projection operator ( $\pi$ ) on the attribute  $C.NOME$ . This is connected to a join operator ( $\bowtie$ ) with the condition  $L2.CODIGO\_CLIENTE = C.CODIGO$ . The join operator connects to a projection operator ( $\pi$ ) on the attribute  $CLIENTE\ C$  and another join operator ( $\bowtie$ ) with the condition  $L.CODIGO\_LOCACAO = L2.CODIGO$ . This second join operator connects to a subtraction operator ( $-$ ). The subtraction operator has two inputs: a projection operator ( $\pi$ ) on the attribute  $L.CODIGO\_LOCACAO$  and another projection operator ( $\pi$ ) on the attribute  $L1.CODIGO\_LOCACAO$ . The first projection operator is connected to a selection operator ( $\sigma$ ) with the condition  $L.CODIGO\_FILME = 5$ , which is connected to the table  $LOCACAO\_FILME\ L$ . The second projection operator is connected to a selection operator ( $\sigma$ ) with the condition  $L1.CODIGO\_FILME = 4$ , which is connected to the table  $LOCACAO\_FILME\ L1$ . The result pane shows a table with one row: 

NOME
José

. The status bar indicates "1 linha(s) selecionada(s)".

The screenshot shows the same software interface, but the result pane now displays the SQL query corresponding to the query tree. The query is: 

```
SELECT C.NOME
FROM LOCACAO_FILME L
INNER JOIN LOCACAO L2 ON L.CODIGO_LOCACAO = L2.CODIGO
INNER JOIN CLIENTE C ON L2.CODIGO_CLIENTE = C.CODIGO
WHERE L.CODIGO_FILME = 5 AND (L.CODIGO_LOCACAO) NOT IN (SELECT
L1.CODIGO_LOCACAO
FROM LOCACAO_FILME L1
WHERE L1.CODIGO_FILME = 4)
```

# Resultados e discussão

- ◆ Boa funcionalidade e de fácil uso
  - ◆ Permite a construção de árvores complexas
  - ◆ Álgebra sobre sacolas
  - ◆ Geração dos comandos SQL utilizando template
-



# Resultados e discussão

- ◆ Comparativo entre os trabalhos correlatos

	<b>IDFQL</b>	<b>EnsinAr</b>	<b>Ferramenta</b>
<b>Árvores de expressões algébricas</b>	<b>Sim</b>	<b>Não</b>	<b>Sim</b>
<b>Expressões algébricas</b>	<b>Não</b>	<b>Sim</b>	<b>Não</b>
<b>Manter arquivos</b>	<b>Sim</b>	<b>Não</b>	<b>Sim</b>
<b>Acesso a tabelas de diversos <i>schemas</i></b>	<b>Não</b>	<b>Não</b>	<b>Sim</b>
<b>Álgebra Estendida</b>	<b>Não</b>	<b>Não</b>	<b>Sim</b>

# Conclusões

- ◆ Ferramenta de apoio ao ensino de álgebra relacional
  - ◆ Objetivos em geral foram atendidos
  - ◆ Limitações
-

# Extensões

- ◆ Disponibilizar os operadores rebatizar, atribuição, junção natural e intersecção
  - ◆ Disponibilizar suporte a expressões de álgebra relacional
  - ◆ Incluir uma interface que realize a conversão de comandos SQL em árvores de expressões
  - ◆ Aprimorar a entrada de dados para os operadores, realizar validações para os nomes de colunas informados e condições de junção e seleção
-

Apresentação prática!