Interpretador de consultas para objetos Java

Douglas Matheus de Souza
Prof. Marcel Hugo, Mestre - Orientador

Roteiro

- Introdução
- Objetivos
- Fundamentação teórica
- Desenvolvimento da ferramenta
- Operacionalidade
- Resultados e discussão
- Conclusões
- Versões futuras

Introdução

- Aplicações → Consultas de dados
- Banco de dados → Linguagem de consulta
- Vetores e Collections → laços for e if

```
List<Artista> artistas = new ArrayList<Artista>();
//
for (Artista a : bancoConsulta.getArtistas()) {
    if (a.getNmArtista().equalsIgnoreCase("metallica")) {
        artistas.add(a);
    }
}
```

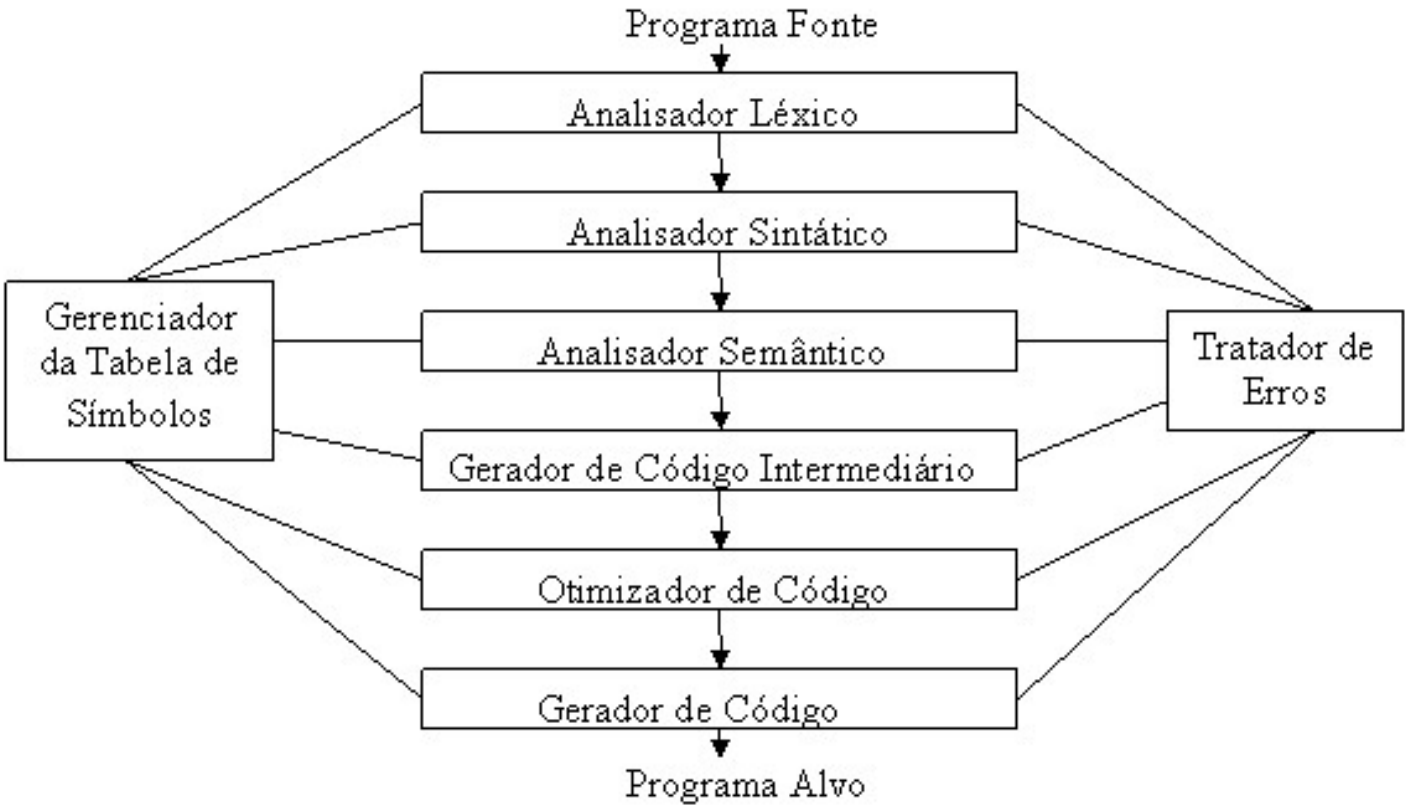
Objetivos

- Disponibilizar uma linguagem de consulta para coleções (Java Collections) e vetores
- Implementar as operações básicas da álgebra relacional
- Disponibilizar o uso de funções de agregação

FUNDAMENTAÇÃO TEÓRICA

Compilador

- Programa que interpreta um outro programa escrito em certa linguagem



Compilador

- Léxico
  - Reconhecimento dos tokens
  - Token → Palavra / Símbolo do texto
  - Lista de tokens
- Sintático
  - Verificação da lista de tokens
- Semântico
  - Verificação de tipos
  - Verificar declaração de variáveis
  - Execução de ações (interpretadores)

Álgebra relacional

- Representação formal de consultas
- Cláusula de consulta → Operadores
  - Seleção / Restrição
  - Projeção
  - Produto cartesiano
  - Junção
  - Agrupamento
  - Ordenação
- Combinação de operadores → Árvore de consulta
  - Folhas → Relações
  - Nós internos → Operadores algébricos

Ferramentas para consulta em coleções

- LambdaJ

```
BancoConsulta bancoConsulta = new BancoConsulta();
//
List<Artista> colecao = filter(new Predicate<Artista>() {
    @Override
    public boolean apply(Artista artista) {
        return artista.getNmArtista().equalsIgnoreCase("Metallica");
    }
}, bancoConsulta.getArtistas());
```

- Coollection

```
BancoConsulta bancoConsulta = new BancoConsulta();
//
List<Artista> colecao = from(bancoConsulta.getArtistas()).
                        where("getNmArtista", eq("Metallica")).
                        all();
```

DESENVOLVIMENTO DA FERRAMENTA

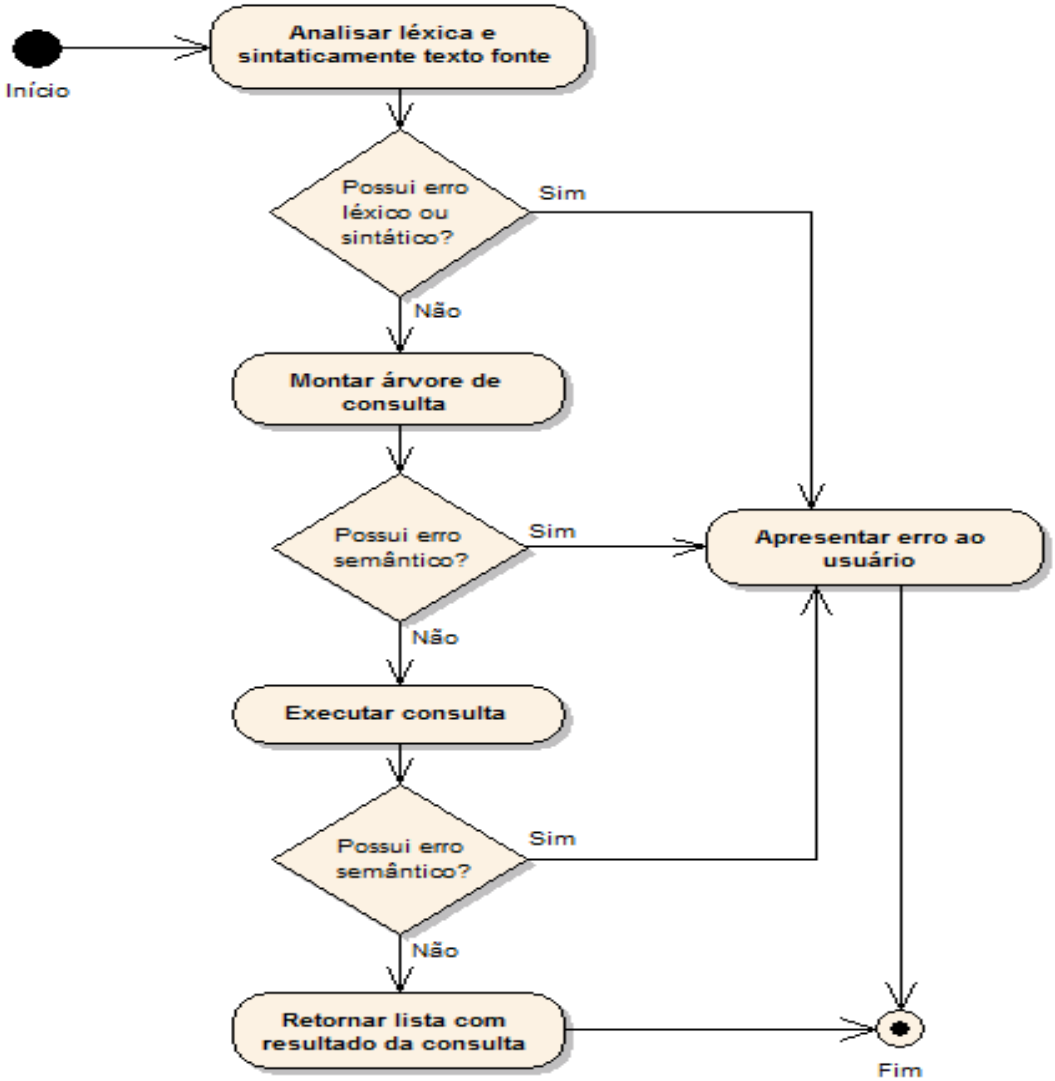
Requisitos da ferramenta

- Permitir a realização de consultas em coleções ou vetores
- Permitir ao desenvolvedor informar uma sequência de comandos de consulta
- Interpretar a sequência de comandos informada pelo usuário
- Exibir mensagens de erros caso comandos inválidos sejam encontrados
- Retornar uma coleção com o resultado da consulta efetuada
- Ter uma linguagem de consulta similar ao SQL
- Ser desenvolvido em Java

Comandos

- select → informar as projeção
- from → informar as relações / coleções
- where → informar as restrições
- group by → informar o agrupamento
- order by → informar a ordenação
- range → informar um intervalo de registros

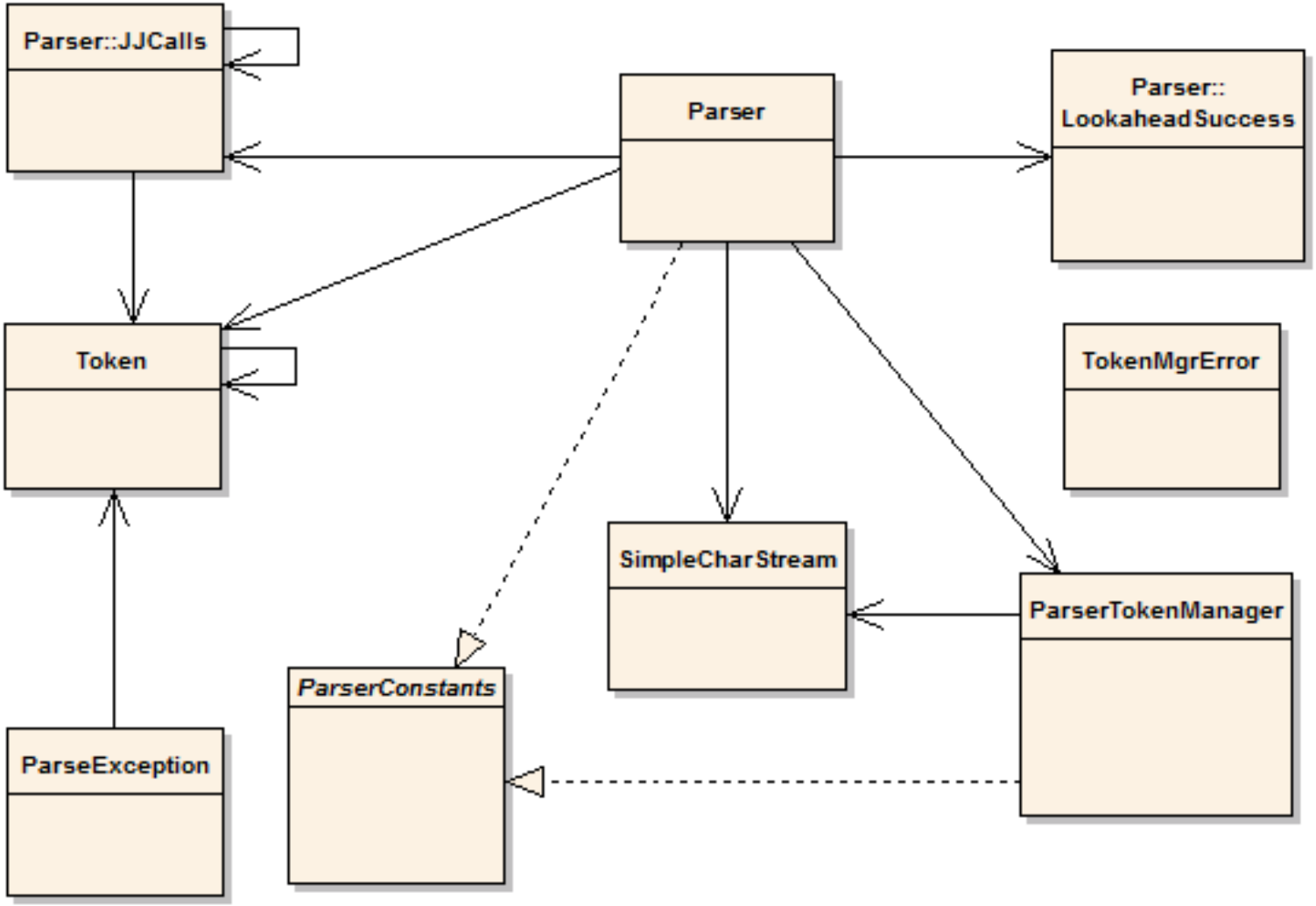
Interpretador - Diagrama de atividades



Análises léxica e sintática

- JavaCC
  - Definição dos tokens
  - Regras gramaticais
  - Notação EBNF
  - Especificação com código puramente Java
- Principais classes
  - ParserTokenManager → Reconhecimento dos tokens
  - Parser → Analisador sintático
  - TokenMgrError → Erros léxicos
  - ParseException → Erros sintáticos
- Objetivo
  - Gerar listas de operadores da álgebra relacional

Interpretador - Análises léxica e sintática
Diagrama de classes

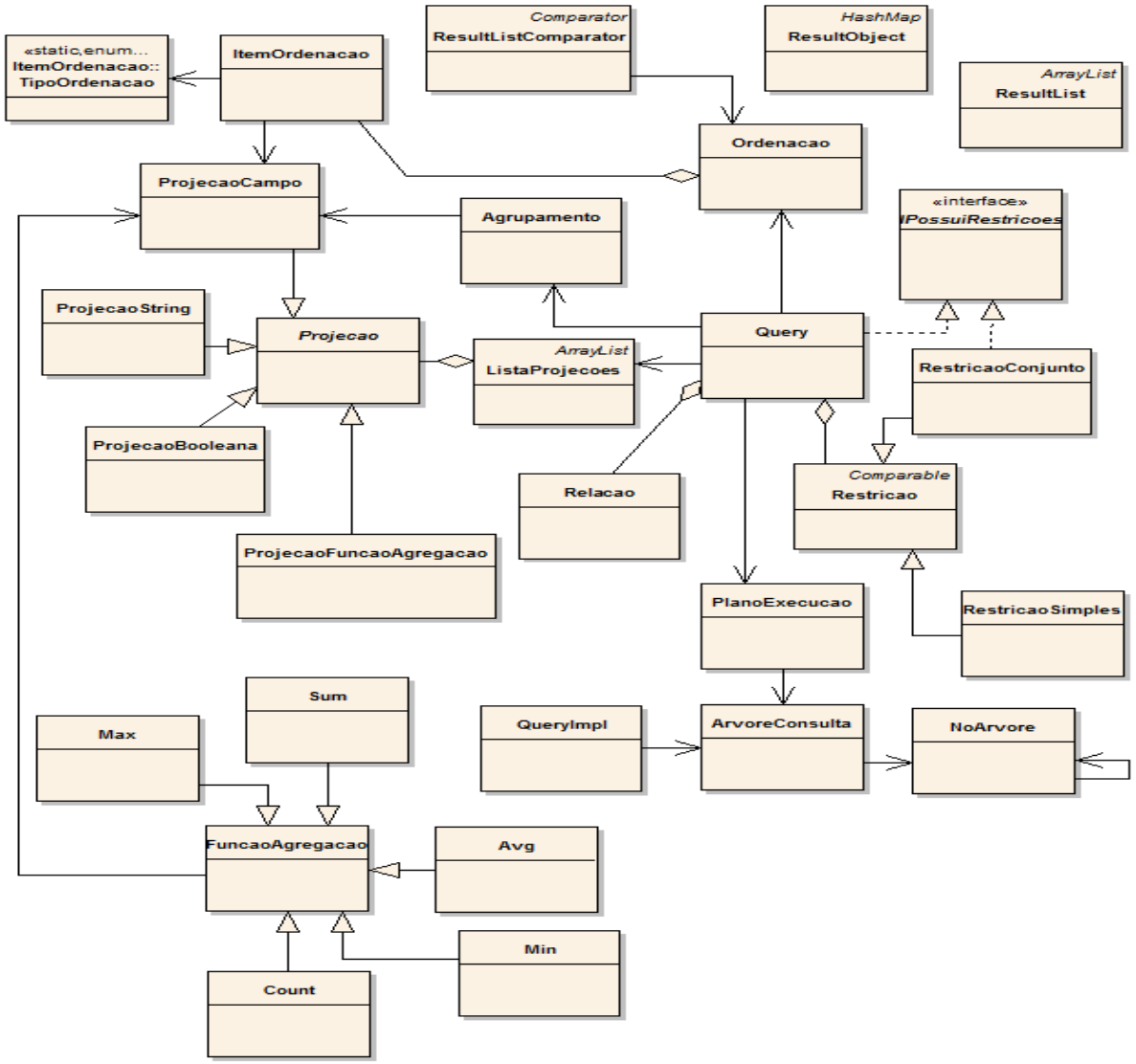


Análise semântica

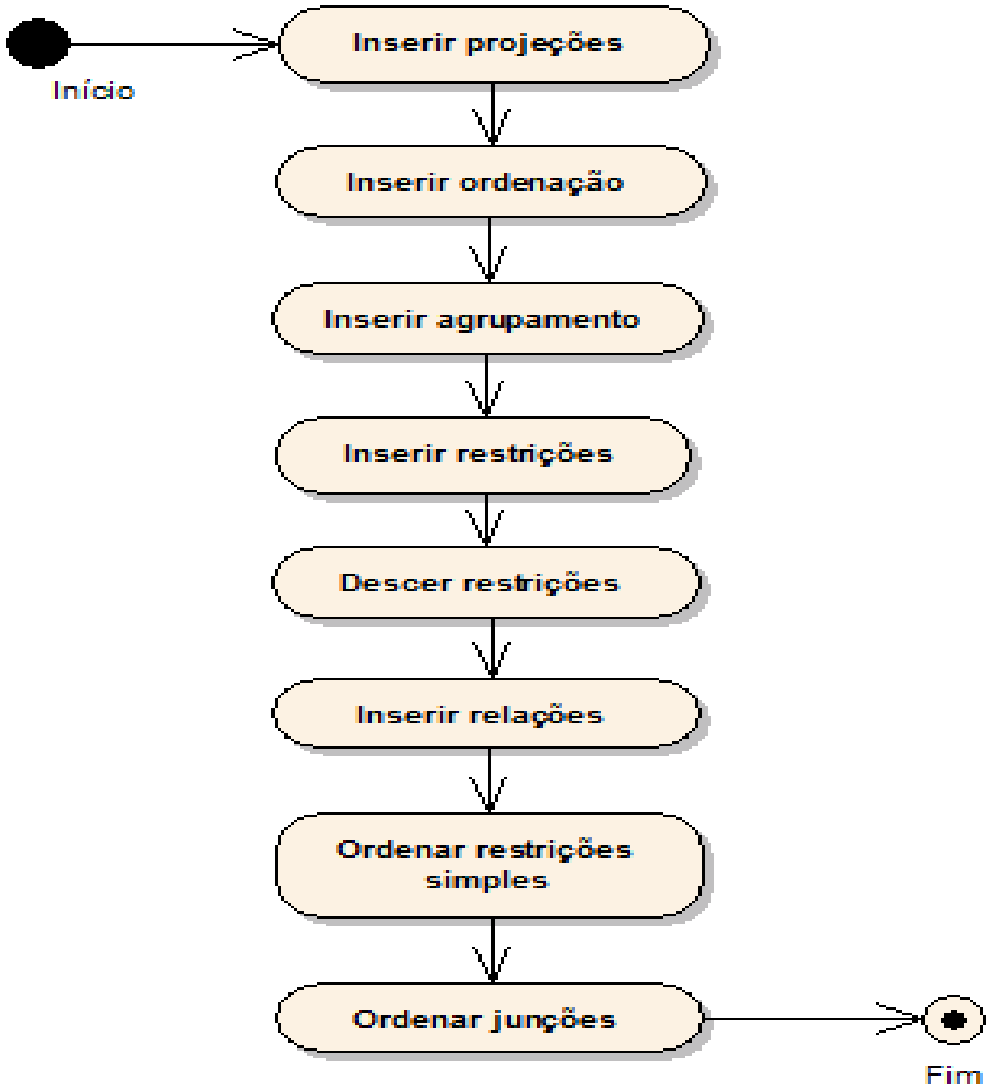
- Duas etapas principais
  - Plano de execução
    - Etapa muito importante
    - Montar / Otimizar árvore de consulta
  - Execução da consulta

- Verificações semânticas
  - Existência das relações declaradas
  - Existência dos atributos nas relações
  - Compatibilidade de tipos

Análise semântica
Diagrama de classes

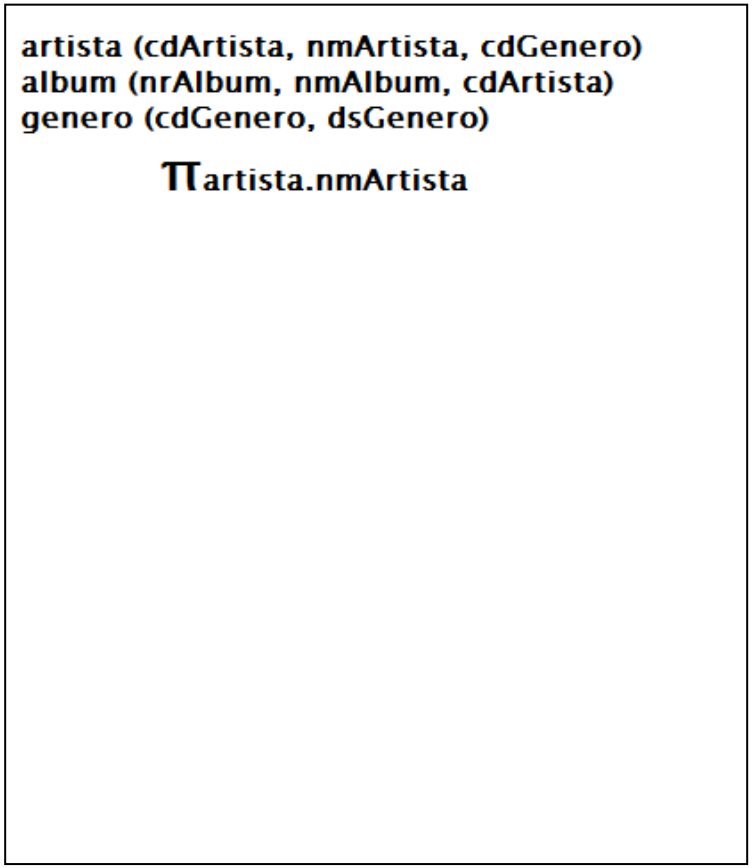


Plano de execução
Diagrama de atividades



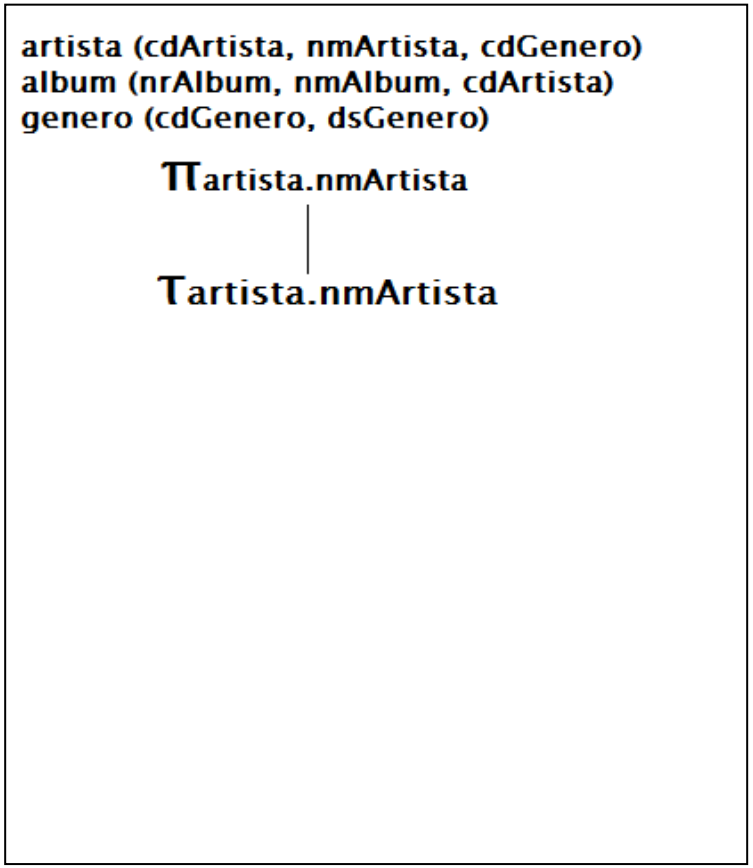
Plano de execução

- Inserir projeções
  - Executa por último
  - Precisa da coleção final



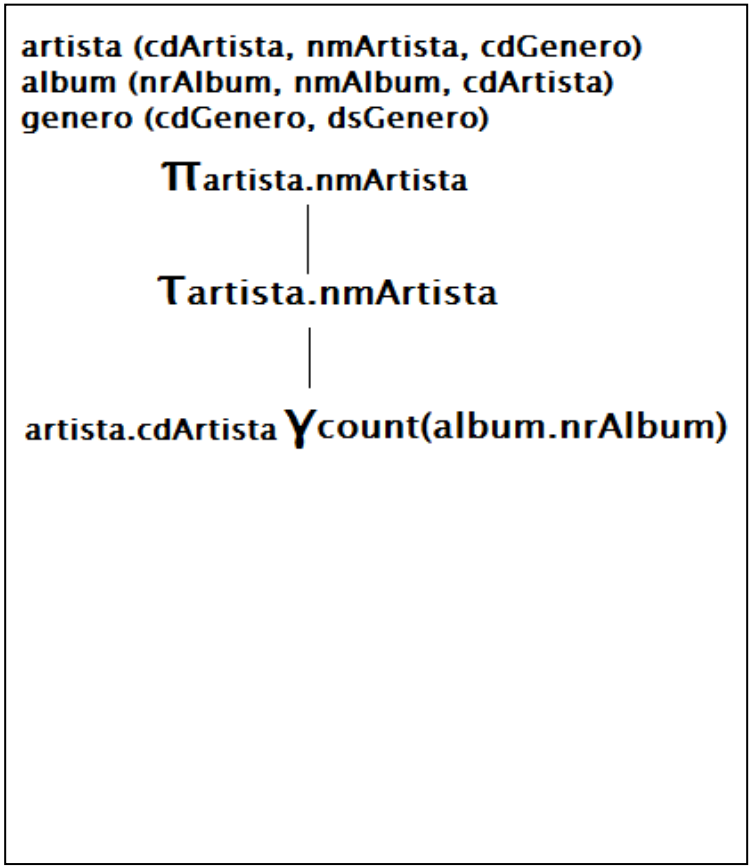
Plano de execução

- Inserir projeções
- Inserir ordenação
  - Antes do agrupamento
  - Agrupamento diminui coleção



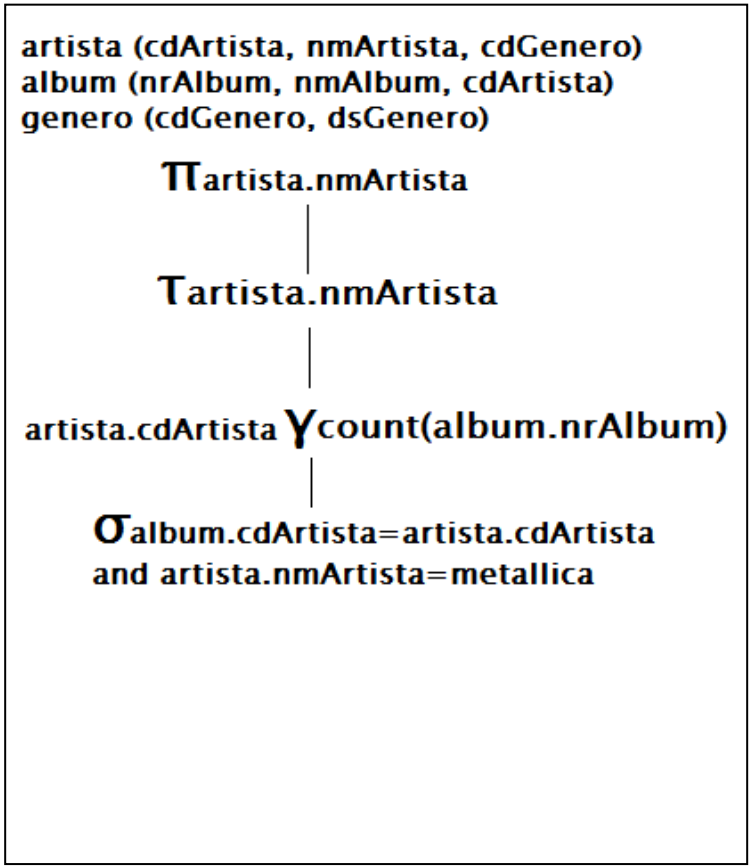
Plano de execução

- Inserir projeções
- Inserir ordenação
- Inserir agrupamento
  - Calcula funções de agregação



Plano de execução

- Inserir projeções
- Inserir ordenação
- Inserir agrupamento
- Inserir restrições
  - Restrições do **where**
  - Ordem que foram informadas

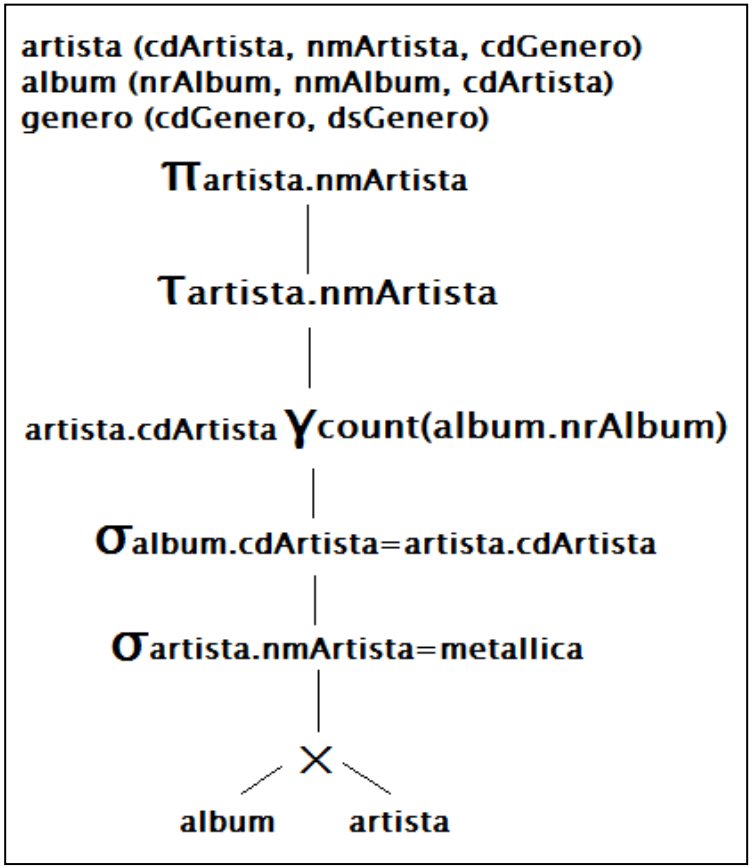


Plano de execução

- Inserir projeções
- Inserir ordenação
- Inserir agrupamento
- Inserir restrições
- Descer restrições simples
  - Posicionar mais abaixo
  - Executar primeiro
  - Diminuir coleção para junção

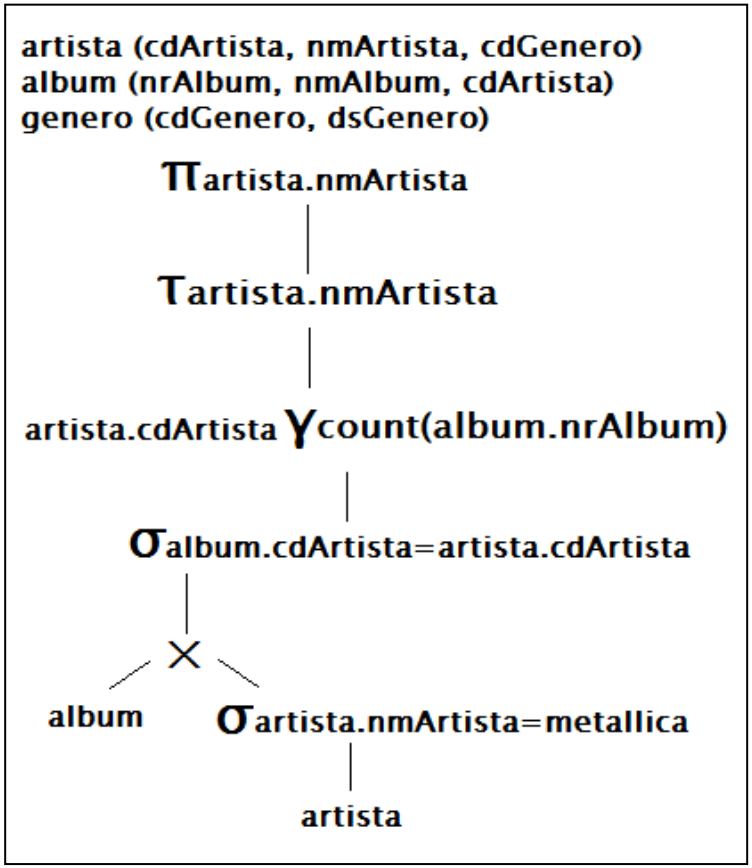
Plano de execução

- Inserir projeções
- Inserir ordenação
- Inserir agrupamento
- Inserir restrições
- Descer restrições simples
- Inserir relações
  - Folhas
  - Formar produto cartesiano



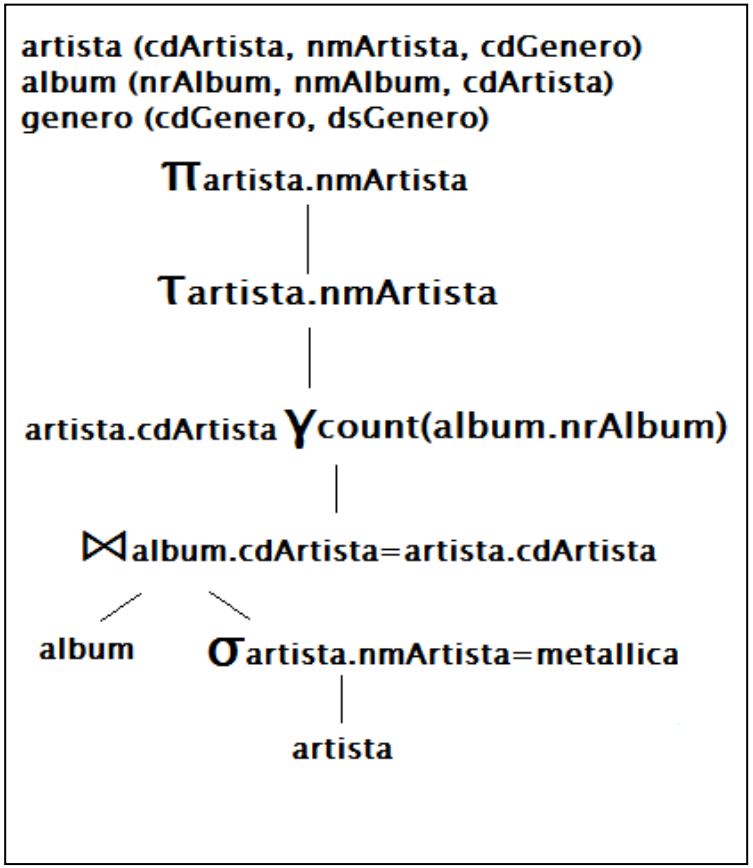
Plano de execução

- Inserir projeções
- Inserir ordenação
- Inserir agrupamento
- Inserir restrições
- Descer restrições simples
- Inserir relações
- Ordenar restrições simples
  - Ligar com suas coleções
  - Diminuir produto cartesiano



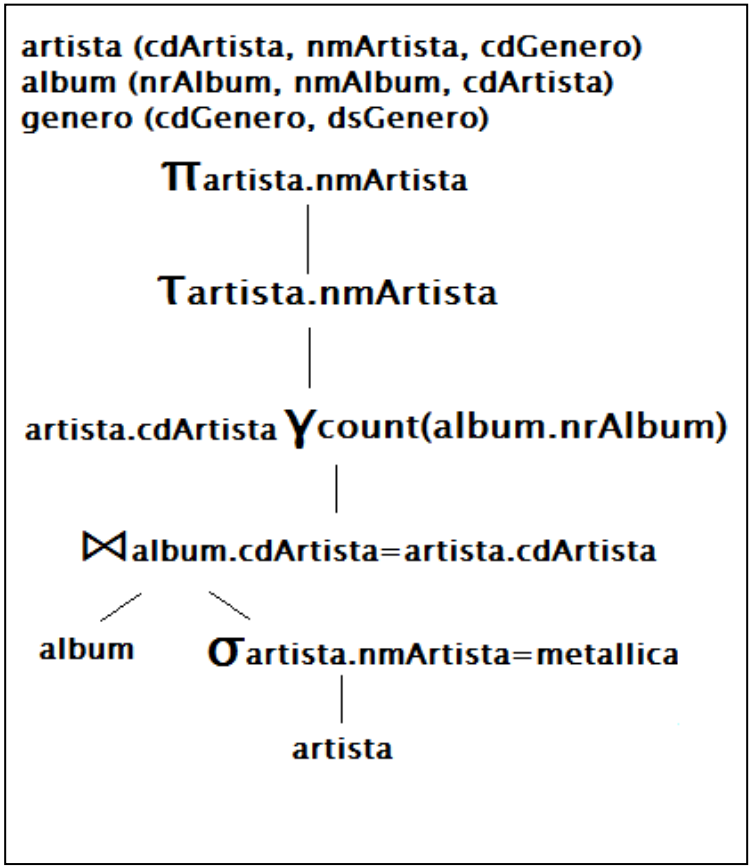
Plano de execução

- Inserir projeções
- Inserir ordenação
- Inserir agrupamento
- Inserir restrições
- Descer restrições simples
- Inserir relações
- Ordenar restrições simples
- Ordenar junções
  - Eliminar produto cartesiano



Plano de execução

```
select      artista.nmArtista,
            count(album.nrAlbum)
from        album,
            artista
where       album.cdArtista = artista.cdArtista and
            artista.nmArtista = 'metallica'
group by    artista.cdArtista
order by    artista.nmArtista
```



Plano de execução
Casos especiais

- Restrições com operador de disjunção OR
  - Cada operador OR → Subárvore na raiz
  - Resolução → Unir das subárvores da raíz

- Restrições entre parênteses
  - Cada restrição → Nova árvore de consulta
  - Resolução → Resolver árvore de consulta

Execução da consulta
Algoritmos dos operadores

- Projeção
  - Linear
- Ordenação
  - Comparator
- Agrupamento
  - Baseado em hash
- Restrições
  - Busca linear
  - Junção de loop aninhado
  - Junção hash

Execução da consulta Projeção

- Algoritmo linear
- Para cada registro → Projetar atributos

```
R → {}
L → lista de projeções
para cada objeto X na coleção R1 faça
    P → {}
    para cada atributo A de projeção em L faça
        P = P ∪ {A}
    fim para
    R = R ∪ {P}
fim para
```

Execução da consulta Agrupamento

- Baseado em hash
- Agrupamento por igualdade de hashes
  - Registros de hash igual são agrupados

```
R → {}
T → tabela hash

para cada objeto X na coleção R1 faça
   h → hash do agrupamento
   se não existe h em T então
      inserir h em T
      para cada função de agregação F faça
         inserir cópia de F como atributo de X
      fim para
   fim se
   para cada função de agregação F faça
      v → valor do atributo agregado na função
      Fo → cópia de F no objeto com hash de agrupamento h
      atualizar resultado final de Fo com v
   fim para
fim para
```

Execução da consulta
Restrição simples

- Busca linear

- Para cada registro → Verifica se atende condição

```
R → {}
para cada objeto X na coleção R1 faça
    se X obedece condição então
        R = R ∪ {X}
fim para
```

Execução da consulta
Junção de loop aninhado

- Laços for aninhados
- Força bruta
- Custo alto → Produto cartesiano → R1 X R2

```
R → {}
para cada objeto X na coleção R1 faça
   para cada objeto Y na coleção R2 faça
      se X e Y satisfazem condição de junção então
         J → junção de X com Y
         R = R ∪ {J}
      fim se
   fim para
fim para
```

Execução da consulta
Junção hash

- Cálculo do hash dos atributos de junção
- Junção por igualdade de hashes
  - Registros com mesmo hash fazem junção

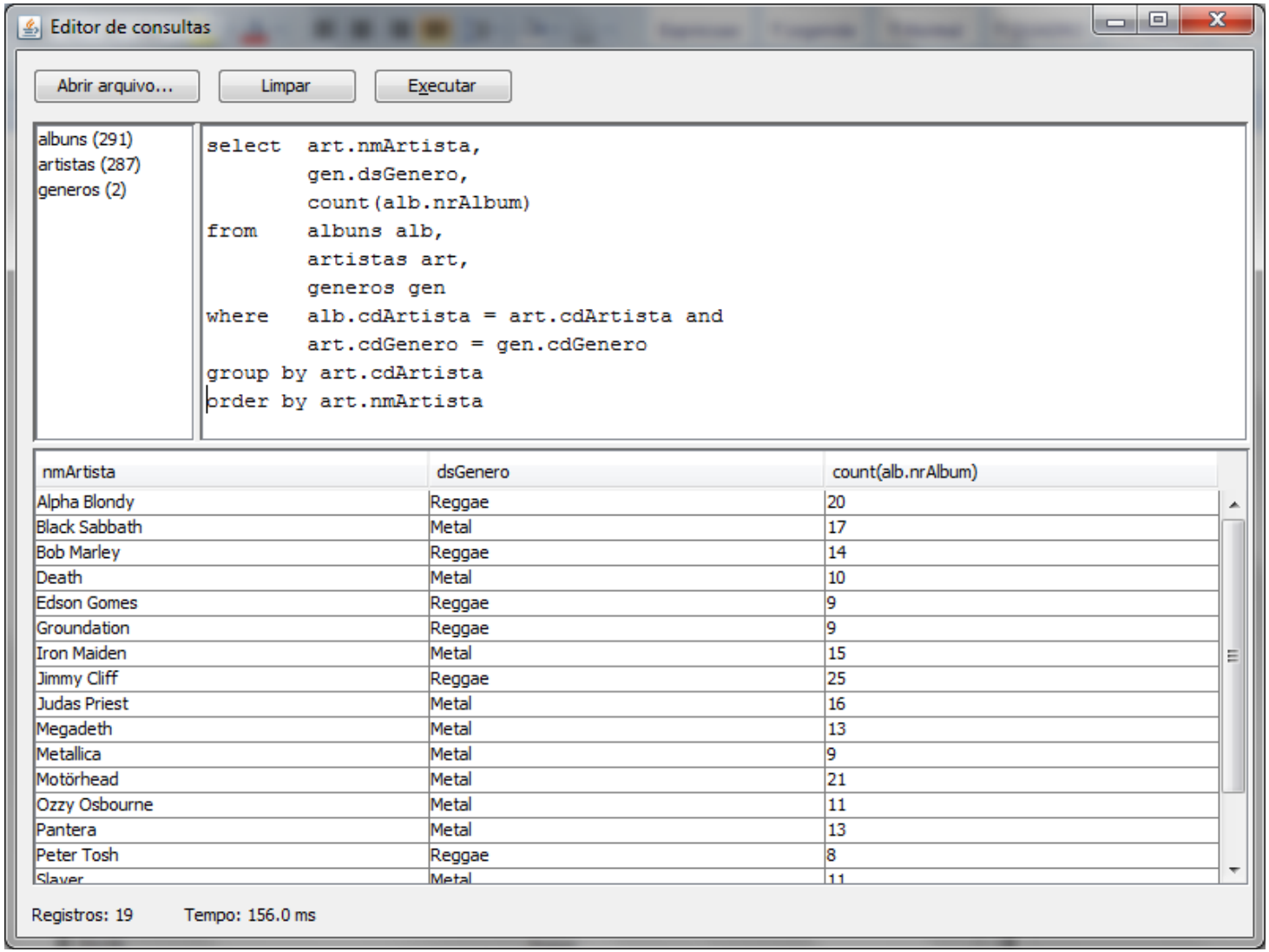
```
R → {}
T → tabela hash

para cada objeto X na coleção R1 faça
   h → hash do atributo de junção em X
   inserir h em T
fim para

para cada objeto Y na coleção  R2 faça
   h → hash do atributo de junção em Y
   O → objeto em T cujo hash é h
   se O não é nulo
      J → junção de O com Y
      R = R U {J}
   fim se
fim para
```

Tratamento de erros

- Erros verificados
  - Erros léxicos e sintáticos → Classes do JavaCC
  - Erros semânticos
    - Não existência de uma coleção
    - Não implementação de **Collection** ou **Object[]**
    - Redeclaração de uma relação
    - Falta de nome das relações nos atributos
    - Não declaração do GROUP BY caso existam funções de agregação

OPERACIONALIDADE

Operacionalidade



nmArtista	dsGenero	count(alb.nrAlbum)
Alpha Blondy	Reggae	20
Black Sabbath	Metal	17
Bob Marley	Reggae	14
Death	Metal	10
Edson Gomes	Reggae	9
Groundation	Reggae	9
Iron Maiden	Metal	15
Jimmy Cliff	Reggae	25
Judas Priest	Metal	16
Megadeth	Metal	13
Metallica	Metal	9
Motörhead	Metal	21
Ozzy Osbourne	Metal	11
Pantera	Metal	13
Peter Tosh	Reggae	8
Slayer	Metal	11

RESULTADOS E DISCUSSÃO

Consulta com laço for X interpretador

- Consulta utilizando laço **for** → Estática

```
List<Artista> artistas = new ArrayList<Artista>();
//
for (Artista a : bancoConsulta.getArtistas()) {
    if (a.getNmArtista().equalsIgnoreCase("metallica")) {
        artistas.add(a);
    }
}
```

- Interpretador → Consulta String → Dinâmica

```
StringBuilder sb = new StringBuilder();
sb.append(" select  nmArtista ");
sb.append(" from    artistas ");
sb.append(" where   nmArtista = 'metallica' ");
//
Query query = new Query(bancoConsulta);
Collection<ResultObject> colecao = query.getResultCollection(sb.toString());
```

Comparativo entre ferramentas

	Query	Tempo (ms)
Interpretador	from artistas where nmArtista = 'metallica'	31
Coollection	from(artistas).where("getNmArtista", eq("Metallica")).all();	24

LambdaJ	filter(new Predicate<Artista>() {
	@Override
	public boolean apply(Artista a) {
	return a.getNmArtista().equalsIgnoreCase("Metallica");
	}
	}, artistas);
		26

LINQ	from art in artistas where art.nmArtista == "metallica" select art	2

Ferramenta X Recursos disponíveis

	Filtragem por atributos	Junção/ Produto cartesiano	Agrupamento/ Agregação	Subconsultas	Ordenação
Interpretador	X	X	X		X
Coollection	X			X	X
LambdaJ	X		X	X	X
LINQ	X	X	X	X	X

CONCLUSÕES

Conclusões

- Recuperar dados em coleções usando linguagem de consulta
- Consulta com String → Mais dinâmicas
- Mais fácil para fazer junções e agrupamentos

VERSÕES FUTURAS

Versões futuras

- Otimizar as projeções
- Dar suporte ao uso de expressões aritméticas com parênteses e nomes de atributos
- Implementar a cláusula having
- Permitir aninhar n atributos nas cláusulas da consulta (ex: pessoa.filho.idade)
- Disponibilizar funções matemáticas
- Disponibilizar funções para tratamento de *Strings*
- Permitir subconsultas

DEMONSTRAÇÃO DA FERRAMENTA