Protótipo de um Gerenciador de Base de Dados para um Ambiente de Programação Lógica

Acadêmico: Aoron Beyer

Orientador: Prof. Roberto Heinzle

Roteiro

- Introdução
- Sistemas Especialistas
- Linguagens de Programação
- Programação Lógica
- Prolog
- Gerenciador da Base de Dados Prolog
- Protótipo
- Conclusões

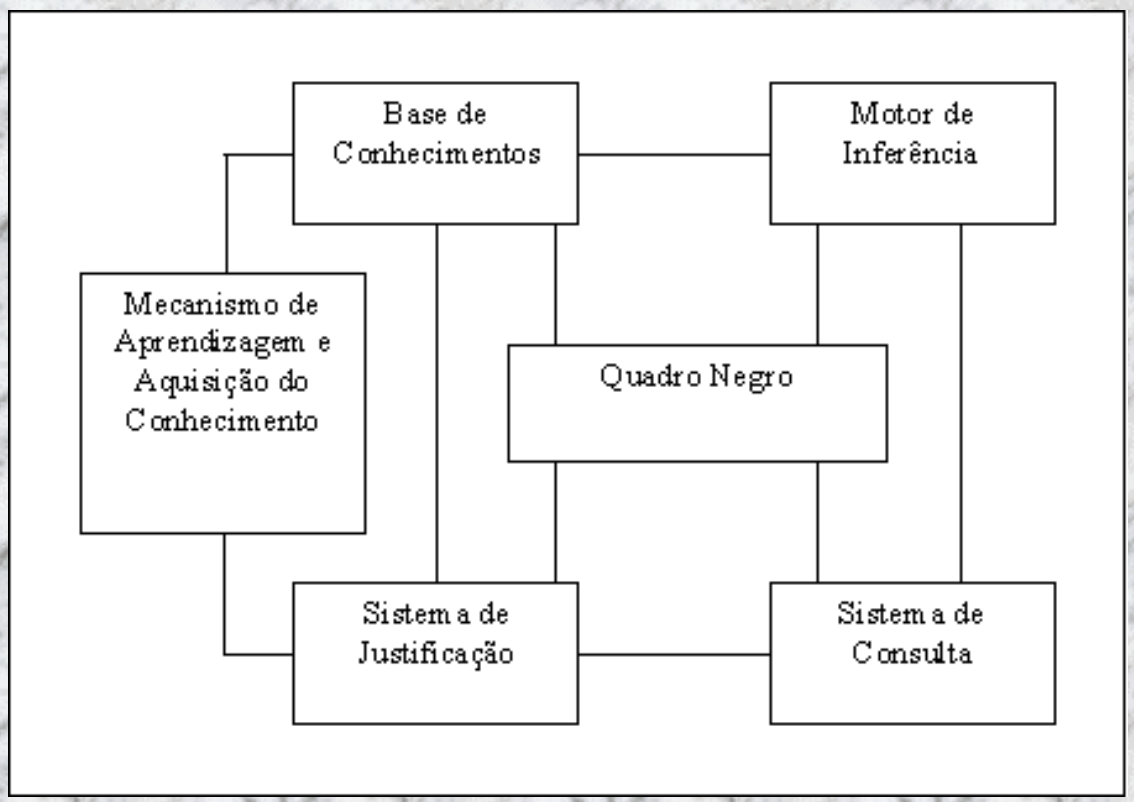
Introdução

- Motivação
- Objetivos
- Problema

Sistemas Especialistas

- Simulam o homem na solução de problemas
- Processa conhecimentos ao invés de dados
- Usa heurísticas

Componentes de um Sistema Especialista



Ferramentas para Construção de Sistemas Especialistas

- Shells
- Linguagens
  - Lisp
  - Prolog

Linguagens de Programação

- Linguagens Procedurais Imperativas
- Linguagens Declarativas
  - Programação Funcional
  - Programação Lógica

Regras

```
progenitor(maria, josé).    progenitor(joão, josé).
progenitor(joão, ana).      progenitor(josé, júlia).
progenitor(josé, íris).     progenitor(íris, jorge).
masculino(joão).
masculino(jorge).
feminino(maria).
feminino(ana).

Para todo X e Y
       Y é filho de X se
       X é progenitor de Y

filho(Y, X) <- progenitor(

Para todo X e Y
       X é mãe de Y se
       X é progenitor de Y
       X é feminino.

mae(X,Y) <- progenitor(X,Y), feminino(X).
```

Variáveis

```
?- comprar(mario,X).
X=bola

?- comprar(X,carro).
X=jorge ->;
X=claudio

?- comprar(X,Y).
X= mario
Y=bola ->;

X=pedro
Y = computador ->.
yes
```

Gerência de Base de Dados em Programação Lógica

- Aspectos da gerência de bases de dados
- Bases de dados em Prolog
  - Inclusão, exclusão, alteração de termos
  - Controle da ordem em que os termos aparecem na base de dados
  - Livre navegação através dos termos
  - Base de dados dinâmica

Características das Bases de Dados em Prolog

- Chave
- Número de referência
- Cláusulas separadas de dados
- Dados armazenados na memória.
- Estruturas de dados
  - Listas Lineares
  - Árvores de pesquisa
  - Pesquisa Hash

Árvores

Salvando

```
recordb(+Tree_Name,+Sort_Key,+Termo)
```

Rest

Apa

Out

```
what_btrees (-BTree)
```

```
replace(+Ref,+Termo)
```

Tabelas Hash

Salvando

```
recordh(+Table_Name,+Sort_Key,+Termo)
```

Restaurando

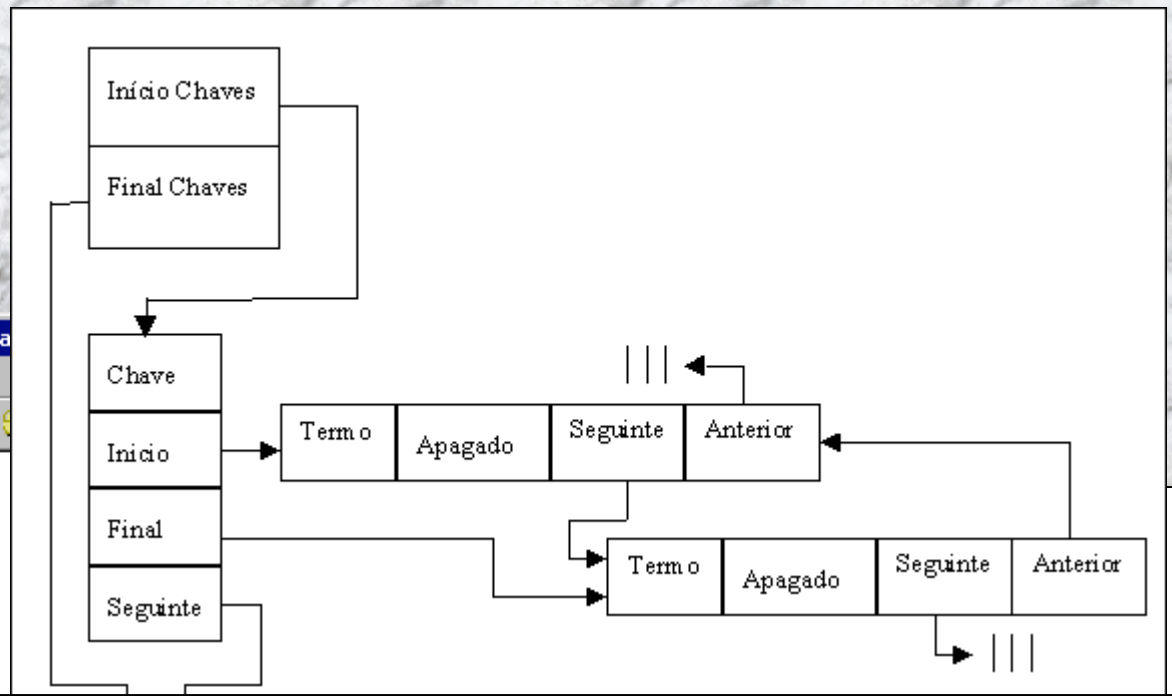
```
retrieveb(+Table_Name,?Sort_Key,?Termo)
```

Apagando

```
removeh(+Table_Name,+Sort_Key,+Termo)
removeallb(+Table_Name)
```

Outros

```
defineh (+Table_Name,+HashBuckets)
```



Inclusão	*Exclusão*	*Alteração*	*Consulta*	*Diversos*	*Base*
recorda	erase	replace	recorded	key	save
recordz	eraseall			instance	restore
record_after	expunge			nref	
	hard_erase			pref	

Considerações Finais

- Trabalhos Correlatos
  - Interpretador para um Ambiente de Programação Lógica
- Extensões
  - Melhorar o protótipo implementando árvores de pesquisa e tabelas hash
  - Implementar os predicados de manipulação de cláusulas
  - Usar técnicas de *parsing* no protótipo
- Limitação
  - Poucos estudos e literatura na área de gerência da base de dados para programação lógica

Conclusões

- Estudo teórico sobre programação lógica e gerência de base de dados para este ambiente
- Desenvolvimento do protótipo
- Metodologia mostrou-se satisfatória
- Ferramenta utilizada (Delphi) eficaz
- Uma ferramenta de gerência de base de dados para programação lógica pode ser muito eficiente