UM AMBIENTE PARA DESENVOLVIMENTO DE PROTÓTIPOS DE BANCOS DE DADOS DEDUTIVOS

por

Nina Edelweiss

Antônio Carlos da Rocha Costa

RP nº 97 Novembro 1988

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
Av. Osvaldo Aranha, 99
90.210 - Porto Alegre - RS - BRASIL
Telefone: (0512) 21-84-99
Telex: (051) 2680 - CCUF BR

Correspondência: UFRGS-CPGCC
Caixa Postal 1501
90001 - Porto Alegre - RS - BRASIL

Banco Dados

ÍNDICE

RESUMO

Este trabalho apresenta as principais características de um Ambiente para Desenvolvimento de Protótipos de Bancos de Dados Dedutivos.

São apresentados os módulos que constituem a sua arquitetura, ressaltando os aspectos de construção mais importantes; as instruções necessárias para a utilização do ambiente e um exemplo de sua utilização.

ABSTRACT

This work presents the major features of a Prototype Development Environment for Deductive Data Bases.

There are described the environment architecture emphasizing the major aspects of its construction, the instructions needed to operate the system and an exemple if its use.

1. INTRODUCÃO

1.1 MOTIVAÇÃO

O objetivo deste trabalho é a criação de um ambiente para o desenvolvimento de protótipos de Banco de Dados Dedutivos.

Dentro deste ambiente, um usuário pode testar a implementação de um protótipo de Banco de Dados, a partir de um esquema conceitual que descreva as informações que vão compor a sua base de dados, e suas retrições de integridade. Através da criação e da manipulação deste protótipo, o usuário pode visualizar o Banco de Dados enquanto vai sendo estruturado, além de validar seu esquema conceitual.

A implementação do sistema que gerencia este ambiente foi feita em PROLOG, podendo rodar em computadores *PC-compatíveis*. No exemplo de protótipo, desenvolvido para ilustrar o uso do ambiente, foi utilizado o modelo E-R (Entidade/Relacionamento) [CHE76] de Banco de Dados.

1.2 ORGANIZAÇÃO DO TEXTO

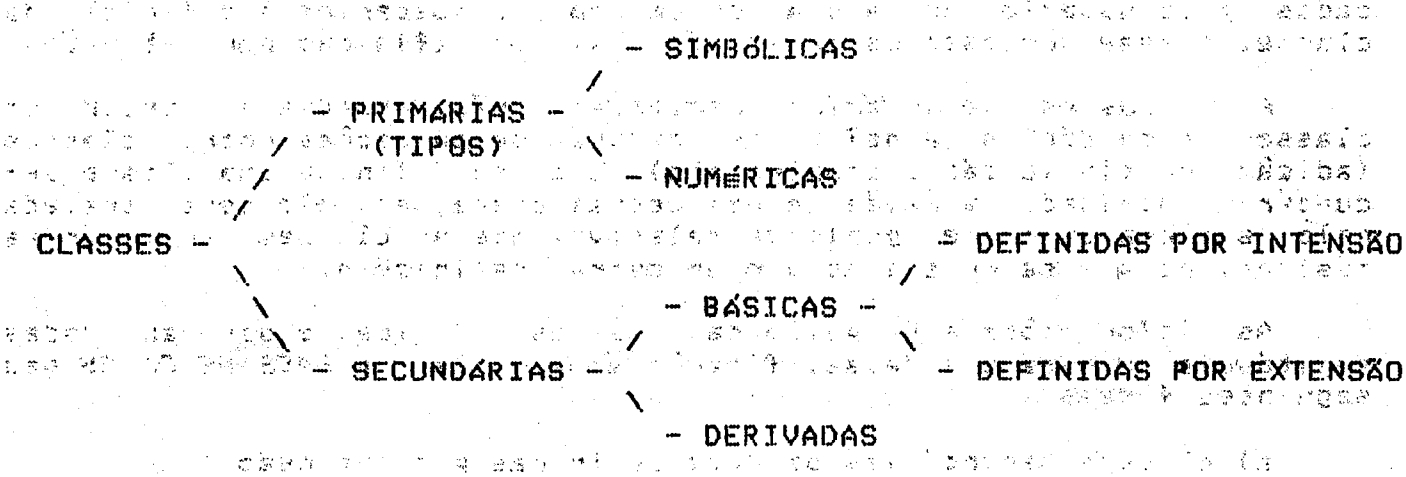
O presente texto está organizado em 6 seções. Na seção 2 é descrito o Modelo de Dados Lógico, utilizado como base para os modelos conceituais dos protótipos a serem desenvolvidos no sistema. Na seção 3 é feita uma descrição dos módulos dos quais o sistema é composto e de seu funcionamento interno. Na seção 4 é detalhado o processo de utilização do sistema por parte de um usuário. A seção 5 apresenta um exemplo de utilização do sistema, enquanto a seção 6 apresenta conclusões e idéias de continuação deste trabalho.

2. MODELO DE DADOS LÓGICO

Como o objetivo era implementar um Banco de Dados Dedutivo, foi buscado um modelo lógico para representar as informações que facilitasse, o mais possível, a implementação desse conceito. Nossa hipótese é a de que um modelo lógico simples é capaz de cumprir esse requisito. Por isso, escolhemos como base do modelo a lógica das ***Classes*** e das ***Relações*** [TAR54].

2.1 CLASSES

O sistema admite o seguinte conjunto de ***classes***:



Existe uma ***classe primária simbólica*** que contém todos os símbolos reconhecidos pelo *PROLOG*. Ela é tratada no sistema como *TIPO SIMBÓLICO*. As ***classes primárias numéricas*** dividem-se em *TIPO INTEIRO* e *TIPO REAL*, e contém todos os números inteiros e reais reconhecidos pelo PROLOG.

As ***classes secundárias*** podem ser de dois tipos diferentes: básicas ou derivadas.

As ***classes secundárias básicas*** serão formadas a partir de elementos retirados diretamente das classes primárias, enquanto que as ***classes secundárias derivadas*** serão resultantes de operações efetuadas entre classes secundárias básicas ou derivadas.

Para definir uma ***classe secundária básica***, o usuário deverá fornecer o *NOME* que ela deverá apresentar, associado ao *TIPO* (classe primária) dos elementos que a vão compor. Desta maneira, ele estará fi-

xando as condições básicas de aceitação de entidades nesta classe.

São três os *TIPOS* que poderão ser associados às classes: *INTEIRO*, *REAL* e *SIMBÓLICO*.

As ***classes secundárias básicas definidas por intensão*** são as de tipo INTEIRO e REAL. Ao associar o nome de uma classe ao *TIPO INTEIRO*, esta irá possuir, como entidades aceitáveis, todos os números inteiros. O mesmo se dará ao ser definida uma classe com *TIPO REAL*, que define uma classe onde as entidades aceitáveis serão todos os números reais. A esse processo de indicação de entidades aceitáveis chamamos definição por intensão. Ao ser definida uma classe de *TIPO SIMBÓLICO*, se estará criando uma ***classe secundária básica definida por extensão***. Para estas classes, a associação de seu nome ao tipo não define as entidades que elas podem aceitar. Estas deverão ser explicitamente indicadas pelo usuário, uma a uma, em um momento posterior à definição da classe. A esse processo de indicação chamamos definição por extensão.

As ***classes secundárias derivadas*** serão formadas a partir de classes secundárias já definidas, através de operações entre classes (adição, multiplicação e complemento). Uma vez definida uma classe secundária derivada através de uma destas operações, ela será tratada pelo sistema em pé de igualdade relativamente às classes secundárias básicas, no que se refere ao uso em outras definições.

As informações a respeito das classes definidas e das entidades aceitáveis em cada uma delas, ficarão armazenadas na *BASE DE DADOS* nas seguintes formas:

a) classes secundárias básicas definidas por intensão :

```
classe(Nome,inteiro).
entidade(Nome,X) :- not var(X),integer(X).

classe(Nome,real).
entidade(Nome,X) :- not var(X),Posit is abs(X),
                    (integer(X),!;float(Posit)).
```

Como pode ser observado acima, nestes dois tipos de classe o sistema armazena, em um fato, o nome da classe (Nome) e o tipo das entidades da mesma. Em se tratando de classes definidas por intensão, o sistema também armazena a regra de formação das entidades aceitáveis em cada uma delas. Isto é feito no mesmo momento em que a classe é definida.

b) classes secundárias básicas definidas por extensão :

```
classe(Nome,simbolico).
entidade(Classe,Entidade).
```

Neste caso, o usuário deverá fornecer as entidades aceitáveis na classe em um momento posterior e independente ao da definição da classe. Cada entidade aceitável será armazenada em um fato, associada ao nome da classe a que pertence.

c) classes secundárias derivadas :

Quando for definida uma classe derivada, o sistema armazenará o nome da classe juntamente com seu tipo (derivada), a operação que vai ser efetuada para obter a nova classe e o(s) nome(s) da(s) classe(s) de origem. Além disto, o sistema também armazenará a regra de formação das entidades válidas nesta classe, de acordo com a operação efetuada.

Por exemplo, no caso de uma classe definida através de uma operação de união de outras duas classes, as informações armazenadas seriam as seguintes:

```
classe(Nome,derivada,uniao,(Classe1,Classe2)).
entidade(Nome,X) :-   entidade(Classe1,X)
                    ; entidade(Classe2,X).
```

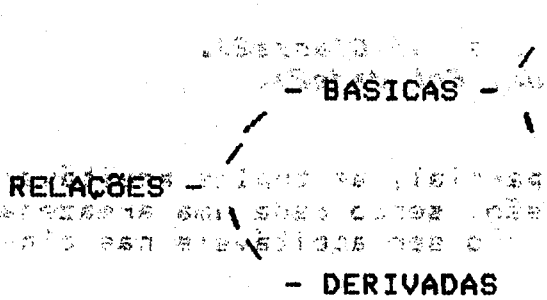
As operações sobre classes implementadas no sistema são:

- união de duas classes ;
- intersecção de duas classes ;
- complemento de uma classe ;
- complemento relativo de uma classe em relação a outra;

2.2 RELAÇÕES

Uma vez definidas as classes e as suas entidades aceitáveis, passa-se à definição de *relações*. A definição de uma relação só poderá ser efetuada se as classes envolvidas em sua definição já estiverem definidas.

O sistema admite o seguinte conjunto de relações :



As *relações básicas* são definidas diretamente entre as classes. Podem ser *totais* e *parciais*.

Uma *relação básica total* entre duas classes define como aceitáveis todas as tuplas que envolvam uma entidade de cada uma das classes, na ordem da definição (produto cartesiano).

Uma *relação básica parcial* entre duas classes aceita todas as *possíveis* tuplas entre entidades das duas classes. Entretanto, o usuário deverá definir por extensão, em um momento posterior e independente, quais as tuplas que são válidas neste tipo de classe. Ao realizar este processo (inserção de tuplas em uma relação parcial), o usuário deverá fornecer os pares de entidades que formarão cada tupla, sendo que estas entidades deverão ser aceitáveis nas classes relacionadas em questão.

Uma *relação derivada* é aquela que é obtida a partir de relações já definidas, através de operações entre relações (união, intersecção, complemento, composição, inversão).

As informações a respeito das relações e das tuplas aceitáveis em cada uma delas são armazenadas na *BASE DE DADOS* através dos seguintes fatos:

a) relação básica total :

```
relacao(Nome,total,Classe1,Classe2).
tupla(Relaçao,Entidade1,Entidade2) :-
    entidade(Classe1,Entidade1),entidade(Classe2,Entidade2).
```

Como pode ser observado acima, o nome da relação é armazenado em um fato, juntamente com o seu tipo (total) e com o nome das duas classes que relaciona. No mesmo momento da definição da relação, o sistema armazena a regra que define as tuplas aceitáveis nesta relação, que são todos os possíveis pares de entidades aceitáveis nas duas classes em questão.

b) relação básica parcial :

```
relação(Nome,parcial,Classe1,Classe2).
tupla(Relação,Entidade1,Entidade2).
```

No caso da relação ser parcial, as tuplas aceitáveis serão fornecidas pelo usuário por extensão, sendo cada uma armazenada em um fato. As entidades das tuplas deverão ser aceitáveis nas classes correspondentes.

c) relação derivada :

Quando a relação definida for do tipo derivada, o sistema armazenará, além do nome e do tipo desta relação, qual a operação que foi

utilizada para a sua formação, e o(s) nome(s) da(s) relação(s) que serviu de base. Ainda no momento da definição da relação derivada, o sistema armazenará a regra que define as suas tuplas válidas.

Por exemplo, se a relação derivada for obtida a partir da operação de união de duas relações, as informações armazenadas seriam as seguintes:

```
relação(Nome,derivada,uniao,(Relação1,Relação2)).
tupla(Relação,Entidade1,Entidade2) :-
        tupla(Relação1,Entidade1,Entidade2)
    ; tupla(Relação2,Entidade1,Entidade2).
```

As operações para definição de relações derivadas que estão implementadas no sistema são:

- união de duas relações ;
- intersecção entre duas relações ;
- complemento de uma relação ;
- composição de duas relações ;
- inversa de uma relação.

Nota-se que foi adotado utilizar somente *relações binárias*, relacionando, portanto, somente duas classes. A representação de relações n-árias deve ser feita com o auxílio de *n* relações binárias envolvendo *n+1* classes (as *n* classes originais, mais uma classe de entidades abstratas, representativas de cada tupla da relação n_ária [KOW79]).

2.3 IGUALDADE E ORDEM

Como o sistema gerenciador deste ambiente foi implementado em PROLOG, poderão ser utilizados todos os operadores pré-definidos nesta linguagem para efetuar comparações e operações matemáticas entre entidades definidas nas classes, seja nas de tipo simbólico, como nas numéricas (inteiro e real).

3. DESCRIÇÃO DO SISTEMA

A figura 3.1 mostra o esquema do sistema implementado, com o fluxo de informações entre os diferentes módulos.

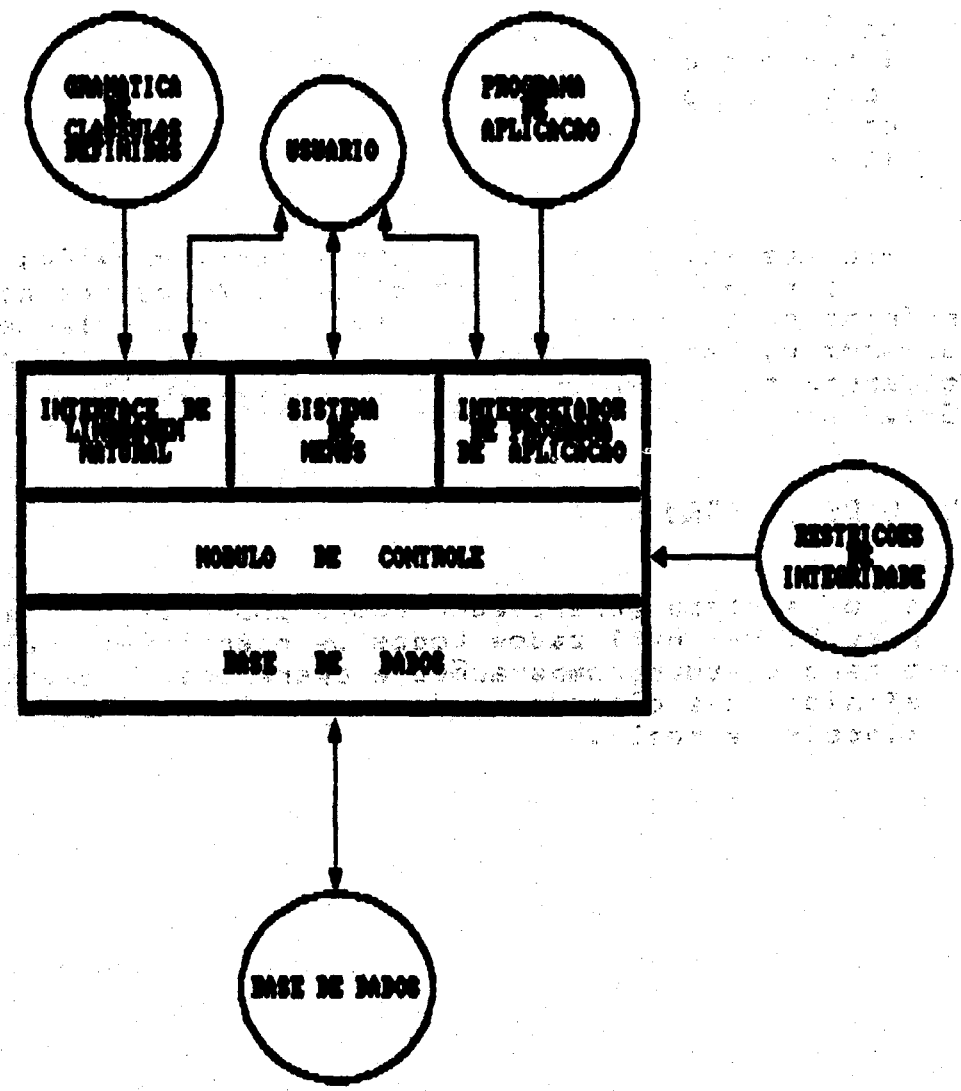


figura 3.1

A comunicação do usuário com o sistema pode ser feita de dois modos. Como objetivo de facilitar a sua utilização, foi criado um ***sistema de menus***. Através destes *menus*, o usuário tem acesso a todos os outros módulos. Outro modo de comunicação é acessando diretamente os módulos do Interface de Linguagem Natural e do Interpretador do Programa de Aplicação.

Foi criado um ambiente que proporciona uma interação do usuário com o Banco de Dados em diferentes níveis, à sua escolha :

- através dos *menus* do próprio sistema;
- através de uma linguagem que se assemelhe à linguagem natural;
- através da execução de programas de aplicação.

Em qualquer destes níveis podem ser incluídas ou retiradas informações da base de dados, sendo providenciadas várias opções para recuperação das mesmas.

Quando o usuário incluir ***restrições de integridade*** na base de dados, estas serão levadas em consideração em cada alteração que for efetuada. Se for solicitada alguma alteração que vá violar estas restrições, ela não será efetivada.

Através dos *menus* do sistema, o usuário pode:

- carregar a base de dados a partir de um arquivo externo;
- carregar as restrições de integridade na base de dados, também a partir de um arquivo externo;
- fazer consultas aos dados armazenados na base de dados;
- fazer alterações na base de dados, incluindo novos dados ou retirando dados lá armazenados, respeitando sempre as suas restrições de integridade;
- solicitar interação através de linguagem próxima à natural;
- solicitar a execução de um programa de aplicação.

A interação do usuário com o sistema através de ***linguagem natural*** é feita a partir de uma gramática de reconhecimento fornecida pelo próprio usuário (Gramática de Cláusulas Definidas) [STEB6]. Ela pode ser solicitada através dos *menus* ou diretamente, como já mencionado antes.

Existe, ainda, a possibilidade de executar, através do sistema, ***programas de aplicação*** escritos pelo usuário em uma linguagem específica. Estes programas podem efetuar alterações nas informações da base de dados, obedecendo sempre às restrições de integridade existentes. Podem, ainda, incluir informações temporárias na base de dados, informações estas que podem ser fatos ou regras de inferência. A inclusão de regras caracteriza o Banco de Dados como ***Dedutivo***. Além das facilidades de alteração vistas, a linguagem em que estes programas devem ser escritos proporciona recursos de avaliação de informações contidas na base de dados, e de entrada e saída de dados. Os programas podem estar gravados em disco, ou serem interpretados, comando a comando, fornecidos a partir do teclado. A execução dos programas de aplicação poderá ser solicitada a partir do sistema de *menus* ou diretamente, através do interface com o seu interpretador.

Todo o conjunto de informações da base de dados estará residente na memória principal - o que o caracteriza como um *protótipo*, não sendo adequado para manipular grandes massas de dados. Entretanto, foi criada a possibilidade de salvar os dados em um arquivo, o que só será feito se o usuário assim o desejar. Com esta finalidade foi criada a opção de carregar dados na memória principal, a partir de um arquivo, possibilitando assim a recuperação destes dados.

3.1 BASE DE DADOS

Todas as informações armazenadas na base de dados estarão na forma dos fatos e das regras mostrados no modelo lógico dos dados (seção 2).

Os programas de aplicação poderão incluir fatos e regras de inferência provisórios nesta base de dados. Estes fatos, cuja representação é diferente daquela vista para o modelo lógico dos dados, e estas regras, serão retirados da base de dados ao final da execução do programa de aplicação.

As restrições de integridade, representadas por regras de inferência apropriadas, quando existentes, serão lidas de um arquivo e passarão a fazer parte da base de dados na memória principal durante a execução do programa.

As informações que fazem parte da base de dados poderão estar armazenadas em disco. No início da execução do sistema, o usuário deverá solicitar que elas sejam carregadas para a memória principal, fornecendo ao sistema o nome do arquivo correspondente. Esta solicitação é feita através de uma das opções do *menu* inicial do sistema. Pode-se colocar na base de dados informações de mais de um arquivo, sendo que na memória principal a base será uma só. No final da execução do sistema, a base de dados poderá ser novamente gravada em disco, a critério do usuário, que deverá fornecer o nome do arquivo onde será feita a gravação.

3.2 GERENCIADOR DO BANCO DE DADOS

é o programa que controla as interações entre o usuário e o Banco de Dados, efetuadas através dos *menus*. É ativado pelo usuário ao acionar o predicado *bd*.

O gerenciador é composto de três módulos:

- módulo de controle
- interface de linguagem natural
- interpretador de programas de aplicação

3.2.1 MÓDULO DE CONTROLE

A carga de dados a partir de arquivos, as alterações e consultas à base de dados, a gravação de dados em arquivo, as explicações do sistema ao usuário, tudo é controlado pelo *módulo de controle*, interagindo com o usuário através de *menus*. Ao final de cada operação, este módulo retorna a um *menu* inicial, onde o usuário escolhe a próxima operação a ser efetuada.

3.2.2 INTERFACE DE LINGUAGEM NATURAL

O *Interface de Linguagem Natural* faz a análise do diálogo entre o usuário e o gerenciador através de *linguagem natural*. Para isto, o usuário deve fornecer ao gerenciador as regras gramaticais da linguagem que vai utilizar, gravadas antecipadamente em um arquivo. Estas regras devem ser escritas na forma de *Gramática de Cláusulas Definidas* (D.C.G.) [STE86].

Através de um diálogo, um usuário pode não só recuperar informações contidas na base de dados , mas também, alterar o conteúdo desta. *Pode incluir novas informações ou remover informações lá armazenadas.*

Na seção de utilização do sistema (item 4.5) está descrita a maneira como estas gramáticas devem ser escritas. Um exemplo de gramática é mostrado no exemplo da seção 5.

3.2.3 INTERPRETADOR DE PROGRAMAS DE APLICAÇÃO

O *Interpretador de Programas de Aplicação* executa as instruções a partir de um arquivo onde o programa está gravado, ou lendo-as a partir do teclado. O usuário deverá escrever este programa em uma linguagem especial, cujos comandos estão descritos na seção de utilização do sistema, no item 4.6.

Através dos programas de aplicação também poderão ser feitas inclusões e retirada de informações da base de dados, usando os predicados citados no item 4.6.

Novas restrições de integridade também poderão ser incluidas na base de dados, faltando ainda implementar a inclusão destas no arquivo externo que contém todas as retrições de integridade da base de dados.

3.3 RESTRIÇÕES DE INTEGRIDADE

O usuário poderá incluir *restrições de integridade* no Banco de Dados. Para a manipulação destas, foi adotada a idéia apresentada por [CAS87]. As restrições devem ser traduzidas em regras de inferência

que, quando satisfeitas, violem a integridade da base de dados.

Todas estas regras fornecidas devem ter a seguinte forma:

viola :- (condição de violação).

Ao ser solicitada alguma alteração de informações na base de dados (inclusão ou retirada de dados), o gerenciador efetua esta alteração. Em seguida, testa se houve violação de integridade ao ser feita a alteração e, caso isso tenha ocorrido, desfaz a alteração, dando uma mensagem ao usuário. O gerenciador testa a ocorrência de violação de integridade através do predicado *viola*.

4. UTILIZAÇÃO DO SISTEMA

Antes de ativar o sistema, o usuário deverá se certificar de que os arquivos que serão utilizados estejam gravados e disponíveis.

Os seguintes arquivos poderão ser utilizados pelo sistema:

- da base de dados;
- de restrições de integridade, quando houver;
- do programa de aplicação, quando for feita uma aplicação;
- da gramática de cláusulas definidas, quando houver interação em linguagem natural.

Como o sistema está implementado em PROLOG, a primeira atitude do usuário deverá ser a de acionar o interpretador PROLOG.

Já sob o interpretador PROLOG, o usuário deverá carregar o programa gerenciador do banco de dados. Isto é feito fornecendo ao interpretador o nome do arquivo onde este programa está gravado.

Feito isto, o sistema é colocado em execução ao ser acionado o predicado:

bd.

Aparecerá o *menu inicial*, com as seguintes opções:

1 : carga do B.D. a partir de um arquivo
2 : carga das restrições de integridade
3 : alterações no B.D.
4 : consultas aos arquivos ou ao B.D.
5 : manipulação do B.D. em linguagem natural
6 : programa de aplicação
f : FIM

O usuário será solicitado a escolher uma das opções oferecidas e a ação correspondente será executada. A escolha da ação será feita digitando o número que a precede no *menu*, seguido de um ponto.

Após o término da execução da ação, o *menu inicial* retornará ao vídeo. Isto se repetirá até que a opção solicitada seja *'FIM'*, quando então o programa será encerrado.

Vamos, a seguir, dar algumas explicações a respeito de cada uma das opções do *menu inicial*.

4.1 CARGA DO B.D. A PARTIR DE UM ARQUIVO

Ao ser acionada esta opção, o sistema perguntará ao usuário o nome do arquivo que deve ser carregado na memória. Este nome deverá ser fornecido entre apóstrofes, completo e seguido de um ponto. Por exemplo:

'arquivo.dad'.

Como esta opção pode ser acionada mais de uma vez, o usuário tem a possibilidade de unir mais de um arquivo em uma só base de dados.

O sistema cuidará para que não haja fatos duplicados na base de dados.

4.2 CARGA DAS RESTRIÇÕES DE INTEGRIDADE

Através desta opção, serão incluídas na base de dados as restrições de integridade que estiverem gravadas em um arquivo. Para tanto, o sistema solicita o nome do arquivo que deverá ser carregado. Este nome deverá ser apresentado da mesma maneira descrita em 4.1.

As restrições de integridade deverão ser escritas na forma de regras de inferência que, quando satisfeitas, violem a integridade da base de dados. Todas as regras devem ser da forma:

viola :- (condição de violação).

Por exemplo, uma regra que diz que o pai deve ser mais velho que o filho deverá ser apresentada da seguinte forma:

```
viola :- tupla(pai,NomePai,NomeFilho) ,
         tupla(idade,NomePai,IdadePai),
         tupla(idade,NomeFilho,IdadeFilho),
         IdadePai =< IdadeFilho.
```

4.3 ALTERAÇÕES NO BANCO DE DADOS

As alterações que podem ser efetuadas são de dois tipos básicos:

- inclusão de novas informações
- remoção de informações

O usuário é solicitado a informar qual a alteração que será efetuada. As opções de alteração são apresentadas através do seguinte *menu*:

```
1 : criar classes
2 : inserir entidades
3 : remover entidades
4 : remover classe
5 : criar relações
6 : inserir tuplas
7 : remover tuplas
8 : remover relações
```

A cada alteração efetuada, o sistema efetuará testes para verificar se não houve violação da integridade do banco de dados. Caso a

alteração venha a violar esta integridade, ela será desfeita e o usuário será informado.

4.3.1 CRIAÇÃO DE CLASSES

A criação de novas classes é feita fornecendo, para cada classe, seu nome e informando qual o tipo dos elementos que a vão compor. No sistema foram definidos os seguintes tipos:

inteiro
real
simbolico
derivada

Quando a classe definida for dos tipos *inteiro*, *real* e *simbolico*, a criação da classe é feita diretamente. Quando, entretanto a classe for do tipo *derivada*, o sistema listará quais as operações disponíveis, do seguinte modo:

Qual a operação inter-classes a ser utilizada ?
1 - uniao
2 - interseccao
3 - complemento
4 - complemento relativo

O usuário escolherá, então, qual a operação que deverá ser efetuada. A seguir, o sistema solicitará os nomes das classes sobre as quais a operação deverá ser efetuada. Estas classes já deverão estar definidas no sistema.

Não será possível definir uma classe com um nome que já corresponda a outra classe definida antes.

Para terminar o processo de criação de classes, o usuário deverá fornecer, como nome de uma classe, a palavra *fim*. O sistema não aceitará classe com este nome.

4.3.2 INSERÇÃO DE ENTIDADES EM UMA CLASSE

Através desta opção, um conjunto de entidades passará a fazer parte de uma classe. A inserção de entidades somente é aceita em classes secundárias básicas definidas por extenção (do tipo *simbólico*). O sistema não permite inserção de entidades em classes definidas com tipo *inteiro*, *real* e *derivada*, informando isto ao usuário quando necessário.

O usuário deverá fornecer o nome da classe na qual deseja inserir entidades, devendo esta classe ter sido definida previamente. Em seguida, deverá fornecer cada uma das entidades. Para indicar que terminou o conjunto de entidades, o usuário deve digitar a palavra *fim* (conseqüentemente, esta palavra não poderá nunca ser uma entidade).

4.3.3 REMOÇÃO DE UMA ENTIDADE DE UMA CLASSE

Esta ação será efetivada após o usuário fornecer o nome da classe em que ela está definida, e o nome da entidade que deve ser removida. Somente poderão ser removidas entidades definidas por extenção, em classes do tipo *simbólico*.

Ao remover a entidade, o sistema também removerá todas as tuplas definidas com esta entidade, informando ao usuário a cada remoção efetuada.

4.3.4 REMOÇÃO DE UMA CLASSE

Para remover uma classe da base de dados, basta o usuário fornecer o seu nome.

Se a classe removida for do tipo *inteiro*, *real* ou *derivada*, a regra de formação das suas entidades válidas também é removida.

Ao efetuar a remoção da classe, o sistema também removerá todas as entidades nela definidas, todas as relações que a envolvam, todas as relações derivadas que envolvam as relações removidas e todas as tuplas definidas nestas relações. Também serão removidas as regras de formação das tuplas válidas nas relações derivadas removidas.

Todas estas remoções serão informadas ao usuário, uma a uma e na ordem em que vão ocorrendo.

As condições de integridade da base de dados serão testadas a cada remoção realizada. Em caso de haver alguma violação, o sistema informará qual a remoção que causou a violação e restaurará todas as informações à base de dados.

4.3.5 CRIAÇÃO DE RELAÇÕES

Este processo permite a definição de um conjunto de relações, sendo terminado quando for digitado *fim* como nome de relação. Observe-se, portanto, que o sistema não aceitará o nome *fim* como nome de relação.

Para definir uma relação, o usuário deverá fornecer seu nome e o seu tipo. Os tipos de relações implementados são os seguintes:

total - relação básica que envolve todas as entidades de duas classes.
parcial - relação básica com tuplas definidas por extensão.
derivada - relação derivada de outras relações.

Caso a relação definida seja de tipo *total* ou *parcial*, o sistema somente solicitará ao usuário o nome das duas classes que serão rela-

cionadas. Estas classes já deverão estar definidas no sistema, e não precisam ser distintas. No caso da classe ser *total*, o sistema se encarregará de definir a regra de formação das tuplas válidas nesta classe, além da definição da classe propriamente dita.

Quando a relação definida for de tipo *derivada*, o sistema perguntará qual a operação que deverá ser efetuada. Isto será feito através do seguinte texto:

Qual a operação que vai definir a classe derivada ?
1 - uniao de duas relações
2 - interseccao entre duas relações
3 - complemento de uma relação
4 - composição de duas relações
5 - inversa de uma relação

Feita a escolha da operação, deverão ser fornecidos os nomes das relações envolvidas, que deverão estar definidas no sistema. Além da definição da relação, o sistema armazenará a regra de formação das tuplas válidas na nova relação, de acordo com a operação efetuada. nome das duas classes que serão relacionados.

Não é possível definir duas relações com o mesmo nome.

4.3.6 INSERÇÃO DE TUPLAS

Através desta opção, um conjunto de tuplas será definido em uma relação do tipo *parcial*. Os outros tipos de relações não admitem inserção de tuplas.

Inicialmente, o usuário fornecerá o nome da relação, que deverá ter sido definida anteriormente. Em seguida, deverá fornecer as tuplas, separando cada par de entidades por uma vírgula. O sistema se encarregará de verificar se cada uma das entidades é válida na classe correspondente

Não são aceitas tuplas iguais em uma mesma relação.

Para indicar que terminou o conjunto de tuplas a serem definidas, o usuário deverá digitar a palavra *fim*.

4.3.7 REMOÇÃO DE UMA TUPLA

Esta opção permite remover, da base de dados, uma tupla definida em uma relação.

O usuário deverá, inicialmente, fornecer o nome desta relação. Em seguida, deverá fornecer o par de entidades que constitui a tupla, separando-as por uma vírgula.

4.3.8 REMOÇÃO DE UMA RELAÇÃO

Uma relação pode ser removida da base de dados através desta opção. Para tanto, o usuário deverá fornecer somente o nome da relação.

Após remover a relação, o sistema também removerá todas as tuplas que foram definidas nesta relação, listando-as para conhecimento do usuário. Removerá, ainda, todas as relações derivadas que foram definidas a partir desta relação ou de alguma relação derivada removida neste processo. Todas estas remoções são listadas pelo sistema, na ordem de sua ocorrência.

A cada remoção ocorrida, são avaliadas as condições de integridade da base de dados. Caso ocorra alguma violação destas condições, o sistema informa qual a remoção que causou a violação e retorna todas as informações à base de dados.

4.4 CONSULTAS AOS ARQUIVOS E AO BANCO DE DADOS

Através desta opção poderão ser efetuadas consultas aos arquivos definidos pelo usuário e às informações contidas na base de dados.

4.4.1 CONSULTA AOS ARQUIVOS

Os arquivos utilizados pelo sistema, que foram definidos pelo usuário, são aqueles que contém:

- as restrições de integridade
- o programa de aplicação
- a gramática de cláusulas definidas

A consulta será efetuada fornecendo ao sistema o nome do arquivo, da forma apresentada em 4.2. Em seguida, o sistema listará o conteúdo do arquivo solicitado.

4.4.2 CONSULTA AO BANCO DE DADOS

As opções de consulta são apresentadas ao usuário no *menu* abaixo:

1 : consulta às classes definidos
2 : consulta às relações definidas
3 : consulta às entidades de uma classe
4 : consulta às tuplas de uma relação
5 : consulta ao tipo de uma classe
6 : consulta a classes e tipo de uma relação
7 : comparação entre duas classes definidas
8 : comparação entre duas relações definidas

Após indicar qual a opção de consulta que quer efetuar, a ação correspondente será efetuada pelo sistema.

A opção 1 lista todas as ***classes definidas*** na base de dados. Além do nome de cada classe, a listagem apresenta seu tipo e, quando este for *derivada*, a operação inter-classes efetuada e as classes que lhe deram origem.

A opção 2 lista todas as ***relações definidas*** na base de dados. Também nesta opção, além do nome de cada relação, é listado seu tipo e, no caso da relação ser derivada, a operação realizada e as relações utilizadas em sua definição.

Para obter a consulta da opção 3 - ***entidades de uma classe*** - o usuário deverá fornecer somente o nome da classe. Caso o classe seja do tipo *inteiro* ou *real*, o sitema informará que as entidades definidas na classe são, respectivamente, todos os numeros inteiros ou todos os reais. No caso das classes *parcial* e derivadas por união, intersecção e complemento relativo, o sistema listará todas as entidades nela definidas diretamente ou através da operação de formação da classe. Para as classes derivadas por complemento, o sistema apresentará as entidades que não são válidas na classe.

As ***tuplas de uma relação*** serão listadas através da opção 4, após o usuário ter fornecido o nome da relação. Se esta relação for do tipo *total*, o sistema informará que as tuplas válidas são todas as possíveis combinações entre entidades das duas classes relacionadas, informando ainda quais estas duas classes.

Para auxiliar na definição de entidades e de tuplas, foi incluída a opção de consultar qual o ***tipo de uma classe***. Isto pode ser feito através da opção 5, fornecendo ao sistema o nome da classe.

A opção 6 permite consultar quais as ***classes relacionadas*** e qual o ***tipo de uma relação***, fornecendo o seu nome.

Através da opção 7 podem ser feitas ***comparações entre duas classes definidas*** na base de dados. Inicialmente, o usuário deverá escolher qual a comparação que deverá ser efetuada, através das opções oferecidas no seguinte *menu* :

```
Comparacões que podem ser efetuadas :
        1 : se uma classe esta contida na outra
        2 : se duas classes se interseptam
        3 : se duas classes são disjuntas
        4 : se duas classes são idênticas
```

Após ter sido feita a opção, o usuário fornecerá o nome das duas classes a serem comparadas. O sistema fará a comparação pedida, fornecendo de imediato a resposta.

A última opção permite ***comparações entre duas relações definidas*** na base de dados. As opções de comparação são oferecidas no *menu* abaixo:

Comparações que podem ser efetuadas :

1 : se uma relação está contida em outra
2 : identidade de entidades de duas relações
3 : diversidade entre as entidades de duas relações
4 : se uma relação é a inversa de outra.

O sistema efetuará a comparação solicitada pelo usuário.

4.5. MANIPULAÇÃO DO BANCO DE DADOS EM LINGUAGEM NATURAL

O sistema proporciona, ainda, a possibilidade de se interagir com a base de dados, com os propósitos de incluir informações e de fazer consultas, através de uma linguagem mais próxima à linguagem natural. Para que isto seja possível, o usuário deverá fornecer, ao sistema, as regras da gramática de reconhecimento desta linguagem. Estas deverão ser apresentadas na forma de uma *Gramática de Cláusulas Definidas* (D.C.G.) [STE86]. O sistema buscará a gramática em um arquivo externo, cujo nome será fornecido pelo usuário, na forma apresentada em 4.1.

Uma vez que o sistema tenha carregado na memória a gramática de reconhecimento, será iniciado o diálogo. O usuário fornecerá as perguntas ou as informações, uma a uma, ao lado do sinal '>' colocado no vídeo pelo sistema. O sistema fará o reconhecimento da sentença fornecida baseado na gramática, dando as respostas adequadas. Este diálogo só será terminado quando o usuário digitar a palavra *fim*.

A gramática fornecida pelo usuário deve ter, sempre, uma regra cuja cabeça é a palavra *regra*. O reconhecimento de qualquer sentença sempre será iniciado através desta regra. Por exemplo:

```
regra --> sentenca ; consulta .
```

A gramática de cláusulas definidas permite que se definam ações semânticas associadas às regras. Toda vez que uma regra for reconhecida, a ação correspondente será executada. Estas ações deverão ser escritas ao final da regra correspondente, entre os símbolos '{' e '}'.

Nas ações semânticas poderão ser utilizadas alguns predicados do programa gerenciador, destinadas a incluir novas informações na base de dados. Estas serão necessárias, provavelmente, para a definição de linguagem e passarão a integrar permanentemente a base de dados. Também poderão ser utilizados predicados do programa gerenciador com a finalidade de retirar informações da base de dados. Os predicados são os seguintes:

```
defineClasse(Nome,Tipo)
defineEntidade(Classe,Entidade)
defineRelacao(Nome,Classe1,Classe2)
defineTupla(Relacao,Entidade1,Entidade2)
retiraClasse(Nome)
retiraEntidade(Classe,Entidade)
retiraRelacao(Nome)
retiraTupla(Relacao,(Entidade1,Entidade2))
```

Também poderão ser utilizadas outros predicados do programa gerenciador, que auxiliam na busca de informações na base de dados. Considerando o par de entidades relacionadas, estes predicados buscam todas as ocorrências de uma determinada entidade na segunda posição, a partir de condições impostas à primeira entidade, juntando-as em uma lista.

Estes predicados são as seguintes:

f(Lista,ListaResultante)
fef(Lista1,Lista2,ListaResultante)
fouf(Lista1,Lista2,ListaResultante)

onde

<Lista> ::= <Relação> <Segunda_Entidade>

Cada uma destas regras devolve, na *ListaResultante*, todas as ocorrências das segundas entidades das tuplas definidas para esta Relação, que apresentem a primeira entidade fornecida.

Além disso, a regra *fef* procura os elementos que satisfaçam, simultaneamente, as condições impostas nas duas listas (E lógico) e *fouf*, aqueles que satisfaçam uma das duas (OU lógico).

Na seção 5 é apresentado um exemplo de gramática.

4.6 PROGRAMAS DE APLICAÇÃO

O sistema executa programas de aplicação escritos pelo usuário em uma linguagem especial, que será explicada a seguir. O programa poderá estar gravado em disco ou ser fornecido a partir do teclado.

Ao solicitar que um programa de aplicação seja executado a partir de um arquivo, o usuário deverá fornecer ao sistema o nome deste arquivo, apresentando da maneira vista em 4.1. Isto fará com que o programa leia um registro do arquivo de cada vez, liste o conteúdo lido no registro, passando logo a executar a ação identificada. Quando a execução for a partir do teclado, será efetuada comando a comando.

Para auxiliar na depuração destes programas, foi incluida a possibilidade de monitorar a entrada e saída dos procedimentos executados.

Através dos programas de aplicação, o usuário poderá alterar as informações da base de dados, incluindo novos dados. Conforme a necessidade, estes dados poderão passar a fazer parte efetiva da base de dados, ou poderão ser temporários, existindo sòmente durante o tempo de execução do programa de aplicação. No caso de incluir dados permanentes, a ação só será executada após a verificação de todos os testes de consistência do banco de dados.

Também poderão ser feitas retiradas definitivas de informações da base de dados, além de se ter acesso às opções de recuperação de informações oferecidas pelo sistema através dos *menus*.

Além de informações, o usuário poderá incuir regras de inferência na base de dados. Estas regras poderão ser avaliadas pelo programa de aplicação sempre que necessário e existirão somente durante o tempo de execução deste programa.

A linguagem que deverá ser usada ao escrever os programas de aplicação é a seguinte:

incluatupla (Relação) ((Entidade1) , (Entidade2)).

Define uma tupla que será incluída definitivamente na base de dados.

incluaclasse (Nome) ((Tipo)).

Define uma nova classe, associando seu nome ao tipo de dados de seus elementos. Esta classe fará parte permanentemente de base dados.

incluaentidade (Classe) ((Nome)).

Define uma entidade permanente em uma classe.

incluarelacao (Nome) ((Tipo) , (Classe1) , (Classe2)).

Define uma relação entre duas classes, também permanentemente incuída na base de dados.

removaclasse (Nome).

Remove a classe especificada da base de dados, removendo também todas as informações baseadas nela.

removaentidade (Classe) ((Entidade)).

Remove a entidade da classe especificada. Remove, também, todas as tuplas definidas com esta entidade.

removarelacao (Nome).

Remove esta relação, todas as suas tuplas e todas as relações derivadas a partir dela.

removatupla (Relação) ((Entidade1) , (Entidade2)).

Remove esta tupla desta relação.

listeclasses.

Lista todas as classes armazenadas na base de dados, informando seu *TIPO*.

listerelacoes.

Lista todas as relações armazenadas na base de dados, informando seu *TIPO* e as classes que relaciona.

listeentidades (Classe).

Lista todas as entidades definidas nesta classe.

listetuplas (Relação).

Lista todas as tuplas definidas para esta relação.

compareclasses.

Faz com que seja ativado o processo de comparação entre classes.

comparerelacoes.

Faz com que seja ativado o processo de comparação entre relações.

fato (Fato)

Define um fato que será temporariamente incluído na base da dados. Não será feito nenhum teste de consistência sôbre este fato. Após a execução do programa de aplicação, este fato é retirado da base de dados pelo próprio sistema.

se (Condição) entao (Conclusão)

Define uma regra que será temporariamente incluída na base de dados, na forma:

(Conclusão) :- (Condição) .

Após a execução do programa de aplicação, esta regra é retirada da base de dados pelo próprio sistema.

execute (conjunto_de_itens)

Faz com que o *(conjunto_de_itens)* seja executado pelo interpretador.
No *(conjunto_de_itens)* poderão ser utilizados os comandos:

escreva (elemento)
leia (elemento)
novalinha
limpatela
avalie (conjunto_de_fatos)

Quando mais de um comando for utilizado, deverão ser empregados os conectivos lógicos *'e'* e *'ou'*

ligaMonitor
desligaMonitor

Comandos utilizados para ativar e desativar a monitoração da execução do programa de aplicação.

fim

Comando que encerra a execução do programa de aplicação.

Na seção 5 é apresentado um programa de aplicação, escrito de acordo com esta linguagem.

5. EXEMPLO DE UTILIZAÇÃO DO SISTEMA

Esta seção apresenta um exemplo, bastante rudimentar, de utilização do sistema. A intenção não é de explorar e aprofundar os conceitos do modelo escolhido, mas, tão somente, de apresentar as potencialidades do sistema ao gerar um protótipo.

O exemplo escolhido foi um banco de dados para controle acadêmico, muito simplificado [FUR85]. Inicialmente a base de dados irá armazenar informações a respeito de nomes, médias e turmas de alunos, e de nomes, disciplinas e de turmas de professores. Mais tarde, através do programa de aplicação, as informações armazenadas serão ampliadas, incluindo nomes e salários de funcionários, além de salários dos professores. Serão, ainda, feitas algumas consultas para demonstrar as comparações que podem ser feitas entre classes.

5.1 MODELO CONCEITUAL

Em um Banco de Dados encontram-se dois tipos de informações: descrição de ***entidades*** e de ***relacionamentos***. Uma ***entidade*** é descrita por seus ***atributos***, sendo que estes podem representar tão somente uma propriedade, como podem, também, ser objetos que poderiam ser vistos como entidades. Um ***relacionamento*** representa uma determinada associação entre entidades, sendo o número de entidades associadas representado pelo ***grau*** do relacionamento. Os relacionamentos também podem apresentar atributos [FUR85].

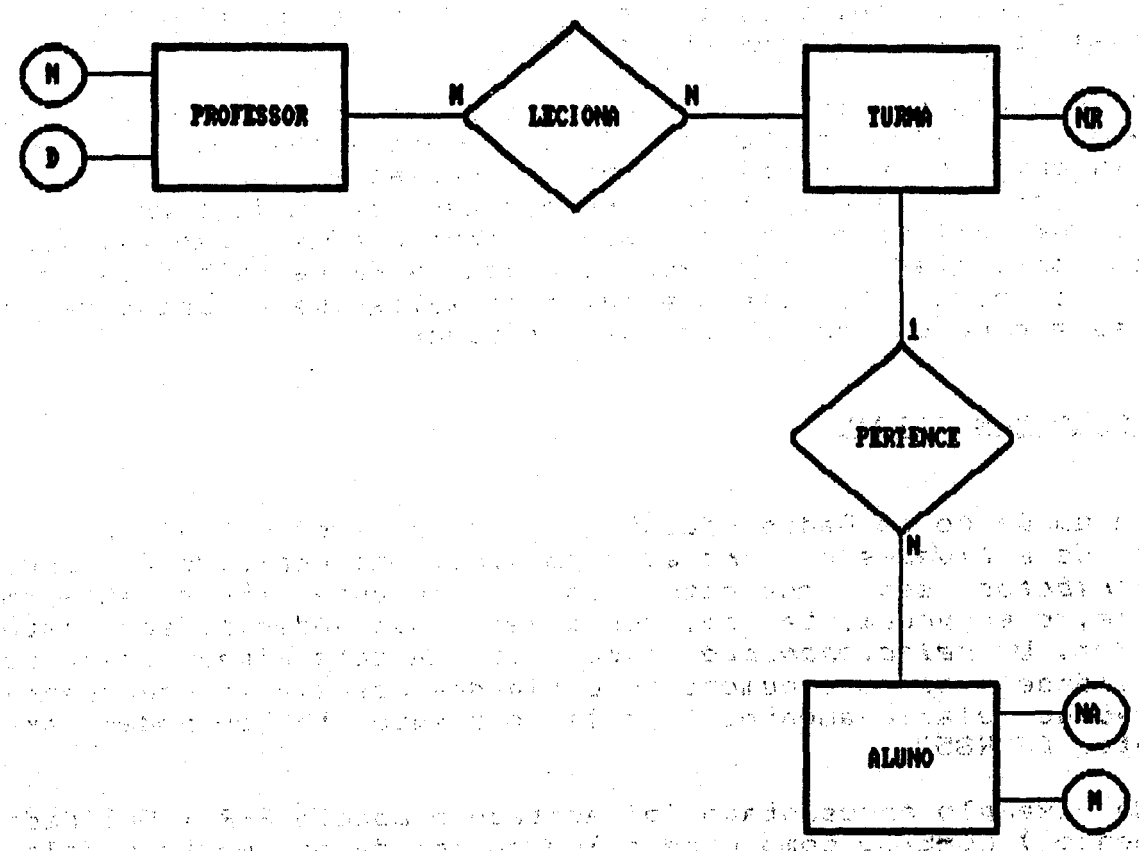
No exemplo apresentado foi adotado o modelo E-R (Entidade-Relacionamento) [CHE76] como modelo lógico dos dados. Neste modelo, a informação é descrita em termos de ***entidades***, ***relacionamentos*** e ***valores*** (que são os atributos). As entidades são definidas em ***domínios*** de diferentes espécies ou tipos, o mesmo acontecendo com os relacionamentos e os valores, podendo, ainda, as entidades e os relacionamentos apresentar atributos.

De acordo com o modelo lógico dos dados usado no protótipo, os domínios de atributos (valores) serão representados através das ***classes***, sendo que o restante das informações deverá ser representado por ***relações*** entre estas classes.

O modelo lógico do exemplo inicial está representado pelo diagrama E-R da figura 5.1. No diagrama da figura 5.2 está representada a extensão que será feita neste modelo através do programa de aplicação, quando novas informações serão armazenadas.

5.2 REPRESENTAÇÃO DO MODELO NO PROTÓTIPO

No protótipo a ser implementado, as informações deverão ser armazenadas em ***classes***, ***entidades***, ***relações*** e ***tuplas***. Inicialmente serão necessárias 5 classes para definir os domínios de atributos.



ENTIDADES:

professor: professores da instituição de ensino
Atributos: N - nome do professor
D - disciplina que leciona

turma: turma da instituição de ensino
Atributos: NR - número da turma

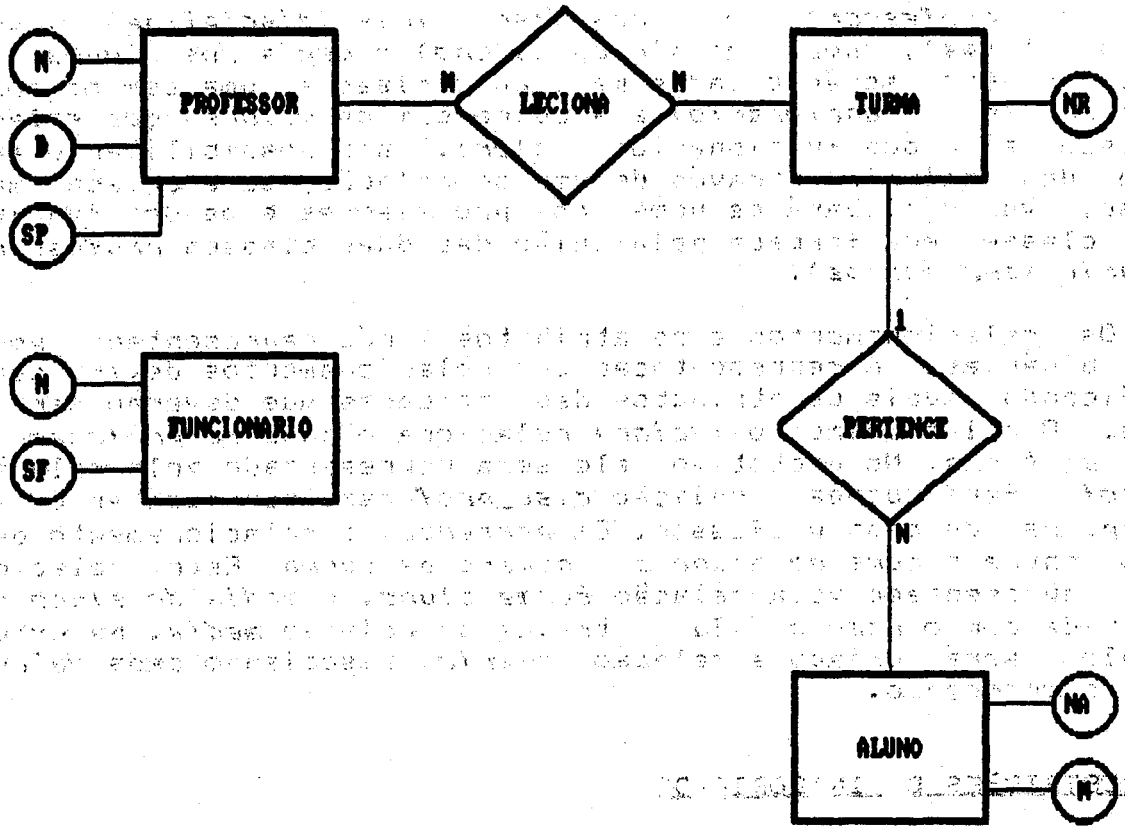
aluno: alunos da instituição de ensino
Atributos: NA - nome do aluno
M - média do aluno

RELACIONAMENTOS:

leciona: professor leciona em turma
Grau: M:N (cada professor pode lecionar em várias turmas e cada turma pode ter vários professores).
Atributos: -

pertence: aluno pertence a turma
Grau: N:1 (cada aluno só pode pertencer a uma turma e cada turma possui vários alunos).
Atributos: -

Figura 5.1



ENTIDADES:

professor: professores da instituição de ensino
Atributos: N - nome do professor
D - disciplina que leciona
SP - salário do professor

turma: turma da instituição de ensino
Atributos: NR - número da turma

aluno: alunos da instituição de ensino
Atributos: NA - nome do aluno
M - média do aluno

funcionario: funcionários, sem englobar os professores
Atributos: SF - salário do funcionário
N - nome do funcionário

RELACIONAMENTOS:

leciona: professor leciona em turma
Grau: M:N (cada professor pode lecionar em várias turmas e cada turma pode ter vários professores).
Atributos: -

pertence: aluno pertence a turma
Grau: N:1 (cada aluno só pode pertencer a uma turma e cada turma possui vários alunos).
Atributos: -

figura 5.2

As classes que serão definidas são as seguintes: *nomes dos professores* (***professor***), *nomes das disciplinas* (***disciplina***), *número das turmas* (***turma***), *nomes dos alunos* (***aluno***) e *média dos alunos* (***media***). Posteriormente serão criadas mais duas classes, uma com os *nomes dos funcionários* (***funcionario***) e a outra com os *valores dos salários* dos professores e dos funcionários (***valor***). Para possibilitar o armazenamento dos salários através de uma só relação, será criada mais uma classe, que englobará os nomes dos professores e os dos funcionários. Esta classe será formada pela união das duas classes *professor* e *funcionario* (***empregados***).

Os relacionamentos e os atributos serão representados por relações binárias. A representação dos relacionamentos deverá ser feita verificando quais os atributos das entidades que deverão ser relacionados. O relacionamento *leciona* relaciona o *nome do professor* e o *número da turma*. No protótipo, ele será representado pela relação ***turma_prof***. Será usada a relação ***disc_prof*** para representar o atributo *disciplina* de cada professor. Em *pertence*, o relacionamento deve ser feito entre o *nome do aluno* e o *número da turma*. Este relacionamento será representado pela relação ***turma_aluno***. A *média do aluno* será relacionada com o *nome do aluno* através da relação ***media***. Na extensão do exemplo, será criada a relação ***salario***, associando cada *valor* com o nome do *empregado*.

5.3 RESTRIÇÕES DE INTEGRIDADE

As restrições de integridade necessárias devem ser gravadas em um arquivo separado. Para o banco de dados escolhido, existem as seguintes retrições que devem ser observadas:

- cada aluno só pode pertencer a uma turma, o que leva à seguinte regra de restrição:

```
viola :- tupla(turma_aluno,Turma1,NomeAluno),
         tupla(turma_aluno,Turma2,NomeAluno),
         Turma1 \= Turma2 .
```

- cada professor só deverá dar aulas de uma disciplina, que corresponde à restrição:

```
viola :- tupla(disc_prof,Disciplina1,NomeProf),
         tupla(disc_prof,Disciplina2,NomeProf),
         Disciplina1 \= Disciplina2 .
```

- as turmas são numeradas a partir de 1 (um), o que leva às seguintes duas regras:

```
viola :- tupla(turma_aluno,Turma,Aluno),
         Turma < 1 .

viola :- tupla(turma_prof,Turma,NomeProf),
         Turma < 1 .
```

- as médias deverão ter valores entre 0 e 10, que se traduz nas regras:

```
viola :- tupla(media,Valor,NomeAluno),
         Valor < 0 .

viola :- tupla(media,Valor,NomeAluno),
         Valor > 10 .
```

5.4 INTRODUÇÃO DE INFORMAÇÕES NA BASE DE DADOS ATRAVÉS DOS MENUS

As informações do modelo inicial foram introduzidas na base de dados através dos *menus* oferecidos pelo sistema.

Conhecendo o modelo lógico dos dados, deve-se definir inicialmente as classes, depois as entidades, as relações e, então, introduzir as informações desejadas através das tuplas.

Classes básicas definidas:

- professor
- aluno
- disciplina
- turma
- media

As três primeiras classes devem ser do tipo *simbólico*. Os números de turmas são inteiros e os valores das medias, reais.

Feita esta definição, na base de dados encontram-sa as seguintes informações:

```
classe(professor,simbolico).
classe(aluno,simbolico).
classe(disciplina,simbolico).
classe(turma,inteiro).
entidade(turma,X):-integer(X).
classe(media,real).
entidade(media,X):-Posit is abs(X),
                   (integer(X),!;float(Posit)).
```

Uma vez definidas as classes, nelas foram inseridas as *entidade* válidas, lembrando que isto só é feito nas classes de tipo simbólico.

Exemplo de algumas entidades definidas e de como se apresentam n base de dados:

```
entidade(aluno,alice).
entidade(professor,joao).
entidade(professor,jose).
entidade(disciplina,matematica).
```

As relações necessárias à representação do exemplo, que envolvem somente classes já existentes, são as seguintes:

```
relacao(disc_prof,parcial,disciplina,professor).
relacao(turma_aluno,parcial,turma,aluno).
relacao(turma_prof,parcial,turma,professor).
relacao(media,parcial,media,aluno).
```

Agora poderão ser definidas as *tuplas* válidas nas relações parciais, como por exemplo:

```
tupla(turma_prof,10,joao).
tupla(turma_prof,20,jose).
tupla(disc_prof,matematica,jose).
```

É bom lembrar que, a cada inclusão de dados que é feita na base de dados, são testadas as restrições de integridade. Assim, não seria possível definir, por exemplo, um aluno para uma turma de número negativo, ou, um professor associado a mais de uma disciplina.

5.5 MANIPULAÇÃO DO SISTEMA ATRAVÉS DE LINGUAGEM NATURAL

Foi definida uma G.C.D. que controla um diálogo com os seguintes verbos:

- ***da_aula_de*** : verbo que relaciona um professor com sua disciplina;

- ***aluno_da*** ou ***tem_aula_na*** : relaciona o nome de um aluno com a turma a que pertence;

- ***da_aula_na*** : relaciona o nome de um professor com uma turma.

Através desta gramática poderão ser fornecidos dados (que serão introduzidos na base de dados), através de frases colocadas entre colchetes e separando as palavras por vírgulas, como, por exemplo:

```
) [alice,tem_aula_na,10].
```

Esta frase define a seguinte tupla:

```
tupla(turma_aluno,10,alice).
```

```
) [joao,da_aula_de,portugues].
```

Esta frase define a tupla:

```
tupla(disc_prof,portugues,joao).
```

```
) [joao,da_aula_na,20].
```

Esta frase define a tupla:

```
tupla(turma_prof,20,joao).
```

Além da definição de dados, a gramática reconhecerá perguntas, respondendo a elas após consultar a base de dados. As perguntas poderão ter as seguintes formas:

```
> [quem,da_aula_na,20].
```

A esta pergunta o sistema responderia o nome de todos os professores que dão aula na turma 20:

```
[joao,jose].
```

```
> [quem,tem_aula_na,10].
```

O sistema forneceria o nome dos alunos da turma 10:

```
[alice].
```

```
> [quem,da_aula_de,historia].
```

O sistema forneceria o nome de todos os professores desta disciplina. Caso não tenha sido fornecido nenhum, responderia:

```
ninguem.
```

```
> [quem,da_aula_de,matematica,e,da_aula_na,20].
```

O sistema forneceria o nome de todos os professores que satisfizessem, simultaneamente, as duas condições colocadas, ou seja, lecionassem matemática, na sala 20:

```
[jose].
```

A gramática utilizada neste exemplo está listada no Anexo 2 (arquivo ***bd.ln***). No Anexo 1 é apresentado um diálogo utilizando esta linguagem.

5.6 PROGRAMA DE APLICAÇÃO

Foi escrito um programa de aplicação que tem, por finalidade, ampliar o conjunto de informações do banco de dados. Este programa foi gravado no arquivo ***bd.prg***, que está listado no Anexo 2. Através deste programa foram incluídos na base de dados os funcionários (seus nomes) da instituição de ensino e os salários dos professores e destes funcionários.

Algumas das informações que o programa de aplicação gravará na base de dados são as seguintes:

```
classe(funcionario,simbolico).
classe(valor,real).
entidade(valor,X):- Posit is abs(X),
        (integer(X),!;float(Posit)).
entidade(funcionario,pedro).
entidade(funcionário,paulo).
classe(empregados,derivada,uniao,(professor,funcionario)).
entidade(empregados,X):-
        entidade(professor,X);entidade(funcionario,X).
relacao(salario,parcial,valor,empregado).
tupla(salario,50000,joao).
tupla(salario,10000,pedro).
```

Também é possível, através do programa de aplicação, definir informações provisórias, isto é, que serão incluídas na base de dados, mas retiradas ao final da execução do programa de aplicação. Na inclusão destas informações não são testadas as restrições de integridade. Para definir uma média que não deverá ser incluída permanentemente na base de dados, o comando a utilizar seria o seguinte:

```
fato media(7,ana).
```

Como sobre estas definições provisórias não é feito nenhum teste de consistência, poderia ser definido o fato:

```
fato media(200,joaquim).
```

Neste fato, o nome não foi definido como entidade válida na classe dos nomes de alunos, e a média estaria fora dos limites colocados.

Além da inclusão de novos valores na base de dados, procurou-se, através deste mesmo programa de aplicação, verificar se os alunos estão ou não aprovados. Isto foi feito incluindo na base de dados, a seguinte ***regra de inferência***:

```
se media(Aluno,Valor) e Valor >= 7 entao aprovado(Aluno).
```

Esta regra procuraria o *Valor* da *media* do *Aluno*, se este valor for superior ou igual a *7* então o predicado *aprovado* seria verdadeiro, caso contrário seria falso. Assim, o programa de aplicação poderá avaliar o predicado *aprovado*, que a base de dados lhe fornecerá a resposta. Por exemplo, o comando:

```
execute avalie aprovado(alice) e escreva aprovado.
```

fará com que seja avaliado se *alice* está aprovada, através da regra de inferência colocada na base de dados. Se isto for verdadeiro, o programa escreverá *aprovado*, caso contrário, indicará *falso*.

5.7 CONSULTAS E REMOÇÃO DE DADOS

Para exemplificar a utilização das diversas opções de consultas oferecidas pelo sistema e, também, para dar uma imagem das informações

que estavam sendo armazenadas, foram feitas diversas consultas à base de dados. Inicialmente foi feita uma consulta através da linguagem de programa de aplicação, sendo executado diretamente através do teclado (quais as classes armazenadas na base de dados). Posteriormente, utilizando a interação através do sistema de *menus*, foram feitas mais algumas consultas, inclusive comparações entre classes armazenadas.

No final do exemplo foi retirada a classe ***professor*** da base de dados, com a única finalidade de demonstrar a utilização desta opção oferecida pelo sistema. É importante observar que a retirada desta classe provoca a retirada de todas as entidades definidas nela, de relações definidas a partir dela, de tuplas definidas nestas relações, das classes derivadas a partir dela, ou seja, provoca a retirada de todas as informações nas quais ela teve alguma contribuição.

As listagens deste exemplo estão no Anexo 1, e os arquivos utilizados (da G.C.D., do programa de aplicação, das retrições de integridade e da base de dados resultante antes da retirada da classe *professor*) são apresentados no Anexo 2.

6. CONCLUSÕES E PROPOSTAS DE CONTINUAÇÃO DESTE TRABALHO

Os objetivos buscados no início da implementação deste ambiente foram plenamente alcançados, mostrando-se êle muito dinâmico, rápido em execução e de muito boa visualização.

O sistema apresentado deverá ser ainda expandido, com a inclusão de mais alguns aspectos. Pretende-se implementar ainda:

6.1 COMPARAÇÕES ENVOLVENDO CLASSES DERIVADAS

No estágio atual, o sistema só efetua comparações entre classes de tipos *simbólico*, *inteiro* e *real*. Como as entidades válidas nas classes *inteiro* e *real* são de número infinito (todos os números inteiros e todos os reais), seu tratamento nestas comparações é diferenciado. As classes derivadas poderão envolver, em diferentes níveis, estas classes, o que deverá ser considerado no momento das comparações.

6.2 DECLARAÇÃO DE RELAÇÕES EM UMA CLASSE

Não foi ainda implementada a declaração de relações definidas em uma só classe, e suas propriedades, a saber:

reflexividade
simetria
transitividade
conectividade

6.3 EXTENSÃO DA LINGUAGEM DOS PROGRAMAS DE APLICAÇÃO

Seria interessante estender a linguagem dos programas de aplicação para permitir, através deles, a consulta às comparações entre classes e entre relações. Poderiam, então, ser formuladas perguntas quanto ao conteúdo da base de dados, em uma linguagem apoiada em lógica, usando, entre outros, os quantificadores.

6.4 AMPLIAÇÃO DO MODELO LÓGICO DOS DADOS

Está prevista, ainda, a extensão do Modelo Lógico dos Dados, através da inclusão neste da parte dedutiva, ou seja, de regras de inferência. No estágio atual da implementação, esta parte é incluída na base de dados através de programas de aplicação, ficando armazenada somente durante o tempo de execução destes programas. Quando for incluída diretamente no Modelo Lógico dos Dados, ficará definitivamente armazenada na base de dados.

ANEXO 1

LISTAGEM DAS TELAS DA EXECUÇÃO DO EXEMPLO APRESENTADO NO ITEM 5

```
                GERENCIADOR DE BANCO DE DADOS DEDUTIVO
                ======================================

                1 : carga do B.D. a partir de um arquivo
                2 : carga das restricoes de integridade
                3 : alteracoes no B.D.
                4 : consultas aos arquivos ou ao B.D.
                5 : manipulacao do B.D. em linguagem natural
                6 : programa de aplicacao
                f : FIM

Qual a sua opcao ?   2.
```

CARGA DE RESTRICOES DE INTEGRIDADE

As restricoes de integridade ja estao gravadas em um arquivo separado ? (S/N)
s.

Entao forneca o nome deste arquivo : 'bd.int'.

---) Arquivo carregado.

DIGITE QUALQUER TECLA PARA CONTINUAR

ALTERACAO DE CONTEUDO DO BANCO DE DADOS

1 : criar classes
2 : inserir entidades
3 : remover entidades
4 : remover classe
5 : criar relacoes
6 : inserir tuplas
7 : remover tuplas
8 : remover relacoes

Qual a sua opcao ? 1.

CRIACAO DE CLASSES

Forneca o nome de cada classe que quer definir,
seguido do tipo permitido para suas entidades.

Tipos permitidos : inteiro
real
simbolico
derivada

Para terminar, digite fim como nome de classe.

```
Classe : professor.
Tipo   : simbolico.
--) A classe foi incluida no B.D.

Classe : aluno.
Tipo   : simbolico.
--) A classe foi incluida no B.D.

Classe : disciplina.
Tipo   : simbolico.
--) A classe foi incluida no B.D.

Classe : turma.
Tipo   : inteiro.
--) A classe foi incluida no B.D.

Classe : media.
Tipo   : real.
--) A classe foi incluida no B.D.

Classe : fim.
```

----) Fim do processo de criacao de classes.

DIGITE QUALQUER TECLA PARA CONTINUAR

```
          INSERCAO DE ENTIDADES
          =====================

Forneca a classe na qual deseja inserir entidades: professor.
Forneca as entidades.

Para terminar : fim

jose.
--) Registrado.
joao.
--) Registrado.
joana.
--) Registrado.
julia.
--) Registrado.
fim.
         ---)   Termino da insercao de entidades.
                ========================================

                DIGITE QUALQUER TECLA PARA CONTINUAR


          INSERCAO DE ENTIDADES
          =====================

Forneca a classe na qual deseja inserir entidades: aluno.
Forneca as entidades.

Para terminar : fim

alice.
--) Registrado.
ana.
--) Registrado.
alexandre.
--) Registrado.
alberto.
--) Registrado.
fim.
         ---)   Termino da insercao de entidades.
                ========================================

                DIGITE QUALQUER TECLA PARA CONTINUAR
```

INSERCAO DE ENTIDADES
=====================

Forneca a classe na qual deseja inserir entidades: disciplina.
Forneca as entidades.

Para terminar : fim

portugues.
--) Registrado.
matematica.
--) Registrado.
historia.
--) Registrado.
geografia.
--) Registrado.
fim.
---) Termino da insercao de entidades.
=====================================

DIGITE QUALQUER TECLA PARA CONTINUAR

INSERCAO DE ENTIDADES
=====================

Forneca a classe na qual deseja inserir entidades: turma.
--) O tipo desta classe (inteiro / real)
nao permite insercao de entidades.

DIGITE QUALQUER TECLA PARA CONTINUAR

```
                    DEFINICAO DE RELACOES
                    =====================

Forneca o nome e o tipo de cada relacao, seguidos dos nomes
das duas classes que devera relacionar, separados por virgula.

Tipos de relacoes:

  total     : relacao basica que envolve todas as entidades das duas classes
  parcial   : relacao basica, com tuplas definidas por extensao
  derivada  : relacao derivada de outras relacoes

Para terminar, forneca fim como nome de relacao.
                        ===


Nome da relacao : disc_prof.
Tipo da relacao : parcial.
Nome das classes a serem relacionados : disciplina,professor.
--) A relacao foi definida.

Nome da relacao : turma_prof.
Tipo da relacao : parcial.
Nome das classes a serem relacionados : turma,professor.
--) A relacao foi definida.

Nome da relacao : turma_aluno.
Tipo da relacao : parcial.
Nome das classes a serem relacionados : turma,aluno.
--) A relacao foi definida.

Nome da relacao : media.
Tipo da relacao : parcial.
Nome das classes a serem relacionados : media,aluno.
--) A relacao foi definida.

Nome da relacao : fim.

             ---) Termino da definicao de relacoes.
                  ================================

                 DIGITE QUALQUER TECLA PARA CONTINUAR
```

INSERCAO DE TUPLAS

Forneca o nome da relacao na qual vai inserir tuplas : disc_prof.

Entre com as tuplas, separadas por virgula.
Classes relacionadas : disciplina e professor

Para terminar : fim

matematica,jose.
--) Registrado.
portugues,joana.
--) Registrado.
historia,joao.
--) Registrado.
geografia,julia.
--) Registrado.
fim.

---) Termino da insercao de tuplas.

DIGITE QUALQUER TECLA PARA CONTINUAR

INSERCAO DE TUPLAS

Forneca o nome da relacao na qual vai inserir tuplas : turma_prof.

Entre com as tuplas, separadas por virgula.

Classes relacionadas : turma e professor

Para terminar : fim

10,joao.
--) Registrado.
20,joao.
--) Registrado.
10,jose.
--) Registrado.
30,julia.
--) Registrado.
30,joana.
--) Registrado.
fim.

---) Termino da insercao de tuplas.

DIGITE QUALQUER TECLA PARA CONTINUAR

INSERCAO DE TUPLAS

Forneca o nome da relacao na qual vai inserir tuplas : turma_aluno.

Entre com as tuplas, separadas por virgula.

Classes relacionadas : turma e aluno

Para terminar : fim

10,ana.
--> Registrado.
20,alice.
--> Registrado.
10,alexandre.
--> Registrado.
30,alberto.
--> Registrado.
20,ana.
--> A tupla nao foi incluida no B.D. pois violaria
    suas restricoes de integridade.
fim.

---> Termino da insercao de tuplas.

DIGITE QUALQUER TECLA PARA CONTINUAR

INSERCAO DE TUPLAS

Forneca o nome da relacao na qual vai inserir tuplas : media.

Entre com as tuplas, separadas por virgula.

Classes relacionadas : media e aluno

Para terminar : fim

5,alice.
--> Registrado.
8.5,alexandre.
--> Registrado.
fim.

---> Termino da insercao de tuplas.

DIGITE QUALQUER TECLA PARA CONTINUAR

MANIPULACAO DE BANCO DE DADOS ATRAVES DE LINGUAGEM NATURAL

```
Voce gravou as regras da GCD em um arquivo ? (S/N) s.
Forneca o nome deste arquivo : 'bd.ln'.

        ---) Arquivo carregado.

             DIGITE QUALQUER TECLA PARA CONTINUAR


->[jose,da_aula_na,30].
--) Registrado.

->[joao,da_aula_de,matematica].
--) A tupla nao foi incluida no B.D. pois violaria
    suas restricoes de integridade.

->[quem,da_aula_de,matematica].
[jose]

->[quem,tem_aula_na,10].
[alexandre,ana]

->[quem,aluno_da,50].
ninguem.

->[quem,da_aula_de,matematica,e,da_aula_na,30].
[jose]

->[quem,da_aula_de,matematica,ou,da_aula_de,portugues].
[joana,jose]

->fim
```

```
                    PROGRAMA DE APLICACAO
                    =====================


        1 : programa gravado em arquivo externo
        2 : entrada de comandos pelo teclado

Qual a sua opcao ? 1.


 EXECUCAO DE PROGRAMA DE APLICACAO A PARTIR DE ARQUIVO
 ====================================================

Seu programa ja esta gravado em um arquivo ? (S/N)   s.

Qual o nome do arquivo onde o programa esta gravado ?  'bd,prg'.

File not found

?-
) incluaclasse funcionario(simbolico)
--) A classe foi incluida no B.D.
) incluaclasse valor(real)
--) A classe foi incluida no B.D.
) incluaclasse empregados(derivada)
Qual a operacao inter-classes a ser utilizada ?
                1 - uniao
                2 - interseccao
                3 - complemento
                4 - complemento relativo
De a sua opcao : 1.
Forneca o nome das duas classes basicas cuja uniao sera efetuada :
        1' classe : professor.
        2' classe : funcionario.
--) A classe foi incluida no B.D.
) incluaentidade funcionario(pedro)
--) Registrado.
) incluaentidade funcionario(paulo)
--) Registrado.
) incluarelacao salario(parcial,valor,empregados)
--) A relacao foi definida.
) incluatupla salario(10000,pedro)
--) Registrado.
) incluatupla salario(10000,paulo)
--) Registrado.
) incluatupla salario(50000.0,jose)
--) Registrado.
) incluatupla salario(50000.0,joao)
```

```
--) Registrado.
) incluatupla salario(60000.0,joana)
--) Registrado.
) incluatupla salario(100000.0,julia)
--) Registrado.
) se media(_0BB5,_08C5) e _0BB5 )= 7 entao aprovado(_08C5)
--) Armazenado.
) fato media(7,ana)
--) Armazenado.
) execute avalie aprovado(alexandre) e escreva aprovado
aprovado
) execute avalie aprovado(ana) e escreva aprovado
aprovado
) execute avalie aprovado(alice) e escreva aprovado
--) falso.
) execute leia _0BB5 e escreva _0BB5 e escreva   e avalie media(_08C5,_0BB5) e escrevo me
ana.
ana 7
) ligaMonitor
--) Monitor ligado.
) fato media(10,alberto)
--) Armazenado.
) execute avalie media(_0BB5,alberto) e escreva _0BB5
call(avalie media(_0BB5,alberto) e escreva _0BB5)
call(media(_0BB5,alberto))
exit(media(10,alberto))
10exit(avalie media(10,alberto) e escreva 10)

) desligaMonitor
--) Monitor desligado.
) incluatupla media(200,alberto)
--) A tupla nao foi incluida no B.D. pois violaria
    suas restricoes de integridade.
) fato media(200,joaquim)
--) Armazenado.
) fim

                DIGITE QUALQUER TECLA PARA CONTINUAR
```

PROGRAMA DE APLICACAO

1 : programa gravado em arquivo externo
2 : entrada de comandos pelo teclado

Qual a sua opcao ? 2.

) listeclasses.

CLASSES DEFINIDAS NO B.D. :

NOME : valor TIPO : real
NOME : funcionario TIPO : simbolico
NOME : media TIPO : real
NOME : turma TIPO : inteiro
NOME : disciplina TIPO : simbolico
NOME : aluno TIPO : simbolico
NOME : professor TIPO : simbolico
NOME : empregados TIPO : derivada OPERACAO : uniao,professor , funcionario

---) Final da pesquisa.

DIGITE QUALQUER TECLA PARA CONTINUAR

```
                    CONSULTA AO BANCO DE DADOS
                    ==========================

          1 : consulta as classes definidas
          2 : consulta as relacoes definidas
          3 : consulta as entidades de uma classe
          4 : consulta as tuplas de uma relacao
          5 : consulta ao tipo de uma classe
          6 : consulta a classes e tipo de uma relacao
          7 : comparacao entre duas classes definidas
          8 : comparacao entre duas relacoes definidas

Qual a sua opcao ? 2.


          RELACOES DEFINIDAS NO B.D. :
          ============================

NOME DA RELACAO    : salario
TIPO DA RELACAO    : parcial
RELACIONA CLASSES  : valor e empregados

NOME DA RELACAO    : media
TIPO DA RELACAO    : parcial
RELACIONA CLASSES  : media e aluno

NOME DA RELACAO    : turma_aluno
TIPO DA RELACAO    : parcial
RELACIONA CLASSES  : turma e aluno

NOME DA RELACAO    : turma_prof
TIPO DA RELACAO    : parcial
RELACIONA CLASSES  : turma e professor

NOME DA RELACAO    : disc_prof
TIPO DA RELACAO    : parcial
RELACIONA CLASSES  : disciplina e professor


                  . ---)  Final da Pesquisa.

              DIGITE QUALQUER TECLA PARA CONTINUAR
```

COMPARACAO ENTRE DUAS CLASSES DEFINIDAS NO BANCO DE DADOS

Comparacoes que podem ser efetuadas :

1 : se uma classe esta contida na outra
2 : se duas classes se interseptam
3 : se duas classes sao disjuntas
4 : se duas classes sao identicas

Qual a sua opcao ? 1.

COMPARACAO DE CONTENCAO ENTRE DUAS CLASSES

Este processo respondera se a classe A esta contida na classe B.

Entre com o nome das duas classes, na ordem acima :

Classe A : professor.
Classe B : aluno.

--> NAO , a classe A nao esta contida na classe B.

DIGITE QUALQUER TECLA PARA CONTINUAR

COMPARACAO DE DISJUNCAO ENTRE DUAS CLASSES

Este processo verifica se duas classes sao disjuntas.

Forneca o nome das duas classes

Primeira classe : professor.
Segunda classe : aluno.

--> SIM , as duas classes acima sao disjuntas.

DIGITE QUALQUER TECLA PARA CONTINUAR

REMOCAO DE CLASSE
==========

```
Forneca o nome da classe que deve ser removida : professor.

--) A classe foi removida do B.D.

--) Tambem foi removida : entidade(professor,julia).
--) Tambem foi removida : entidade(professor,joana).
--) Tambem foi removida : entidade(professor,joao).
--) Tambem foi removida : entidade(professor,jose).
--) Tambem foi removida : classe(empregados,derivada , uniao , professor , funcionario
--) Tambem foi removida : regra de formacao da classe.
--) Tambem foi removida : relacao(salario,parcial,valor,empregados).
--) Tambem foi removida : tupla(salario,100000.0,julia).
--) Tambem foi removida : tupla(salario,60000.0,joana).
--) Tambem foi removida : tupla(salario,50000.0,joao).
--) Tambem foi removida : tupla(salario,50000.0,jose).
--) Tambem foi removida : tupla(salario,10000,paulo).
--) Tambem foi removida : tupla(salario,10000,pedro).
--) Tambem foi removida : relacao(turma_prof,parcial,turma,professor).
--) Tambem foi removida : tupla(turma_prof,30,jose).
--) Tambem foi removida : tupla(turma_prof,30,joana).
--) Tambem foi removida : tupla(turma_prof,30,julia).
--) Tambem foi removida : tupla(turma_prof,10,jose).
--) Tambem foi removida : tupla(turma_prof,20,joao).
--) Tambem foi removida : tupla(turma_prof,10,joao).
--) Tambem foi removida : relacao(disc_prof,parcial,disciplina,professor).
--) Tambem foi removida : tupla(disc_prof,geografia,julia).
--) Tambem foi removida : tupla(disc_prof,historia,joao).
--) Tambem foi removida : tupla(disc_prof,portugues,joana).
--) Tambem foi removida : tupla(disc_prof,matematica,jose).
```

DIGITE QUALQUER TECLA PARA CONTINUAR

ANEXO 2

LISTAGEM DOS ARQUIVOS UTILIZADOS NO EXEMPLO DO ITEM 5

```
                                                    copy bd.int prn

viola :- tupla(turma_aluno,Turma1,NomeAluno),
         tupla(turma_aluno,Turma2,NomeAluno),
         Turma1 \= Turma2.
viola :- tupla(disc_prof,Disciplina1,NomeProf),
         tupla(disc_prof,Disciplina2,NomeProf),
         Disciplina1 \= Disciplina2.
viola :- tupla(turma_aluno,Turma,Aluno),
         Turma < 1.
viola :- tupla(turma_prof,Turma,NomeProf),
         Turma < 1.
viola :- tupla(media,Valor,NomeAluno),
         Valor < 0.
viola :- tupla(media,Valor,NomeAluno),
         Valor > 10.
end_of_file.          1 File(s) copied

A>
```

```
  copy bd.ln prn


regra -->
    [quem],pergunta ;dados .
dados -->
    nome_prof(N),[da_aula_de],disciplina(D),
    { defineTupla(disc_prof,D,N) }.
dados -->
    nome_prof(N),[da_aula_na],turma(T),
    { defineTupla(turma_prof,T,N) }.
dados -->
    nome_aluno(N),( [aluno_da] ; [tem_aula_na] ),turma(T),
    { defineTupla(turma_aluno,T,N) }.
pergunta -->
    frase(P1),[e],frase(P2),
    { fef(P1,P2,L),write(L),nl ; write('ninguem.'),nl }.
pergunta -->
    frase(P1),[ou],frase(P2),
    { fouf(P1,P2,L),write(L),nl ; write('ninguem.'),nl }.
pergunta -->
    frase(X),
    {  f(X,L),write(L),nl ; write('ninguem.'),nl }.
frase([disc_prof,D]) -->
    [da_aula_de],disciplina(D).
frase([turma_prof,T]) -->
    [da_aula_na],turma(T).
frase([turma_aluno,T]) -->
    ( [aluno_da] ; [tem_aula_na] ),turma(T).
nome_prof(N) -->
    [N],
    { entidade(professor,N),!
    ; write('Esperada uma entidade definida no dominio professor.'),
      nl,!,fail  }.
nome_aluno(N) -->
    [N],
    { entidade(aluno,N),!
    ; write('Esperada uma entidade definida no dominio aluno.'),
      nl,!,fail  }.
disciplina(D) -->
    [D],
    { entidade(disciplina,D),!
    ; write('Esperada uma entidade definida no dominio disciplina.'),
      nl,!,fail  }.
turma(T) -->
    [T],
    { integer(T),!
    ; write('A turma deve ter um numero inteiro.'),
      nl,!,fail  }.
end_of_file.        1 File(s) copied

A>
```

```
copy bd.prg prn


incluaclasse funcionario(simbolico).
incluaclasse valor(real).
incluaclasse empregados(derivada).
incluaentidade funcionario(pedro).
incluaentidade funcionario(paulo).
incluarelacao salario(parcial,valor,empregados).
incluatupla salario(10000,pedro).
incluatupla salario(10000,paulo).
incluatupla salario(50000,jose).
incluatupla salario(50000,joao).
incluatupla salario(60000,joana).
incluatupla salario(100000,julia).
se media(Nota,Aluno) e Nota)=7 entao aprovado(Aluno).
fato media(7,ana).
execute avalie aprovado(alexandre) e escreva aprovado.
execute avalie aprovado(ana) e escreva aprovado.
execute avalie aprovado(alice)  e escreva aprovado.
execute leia Nome e escreva Nome e escreva  ' '  e
        avalie media(X,Nome) e escreva X.
ligaMonitor.
fato media(10,alberto).
execute avalie media(N,alberto) e escreva N.
desligaMonitor.
incluatupla media(200,alberto).
fato media(200,joaquim).
fim.
end_of_file.

        1 File(s) copied

A)
```

```
C:\> copy exemplo.hd prn

classe(valor,real).
classe(funcionario,simbolico).
classe(media,real).
classe(turma,inteiro).
classe(disciplina,simbolico).
classe(aluno,simbolico).
classe(professor,simbolico).
classe(empregados,derivada,uniao,(professor , funcionario)).
entidade(funcionario,paulo).
entidade(funcionario,pedro).
entidade(valor,_2E59) :- not var(_2E59) , _3189 is abs(_2E59) , (integer(_2E59) ,
entidade(disciplina,geografia).
entidade(disciplina,historia).
entidade(disciplina,matematica).
entidade(disciplina,portugues).
entidade(aluno,alberto).
entidade(aluno,alexandre).
entidade(aluno,ana).
entidade(aluno,alice).
entidade(professor,julia).
entidade(professor,joana).
entidade(professor,joao).
entidade(professor,jose).
entidade(media,_2E59) :- not var(_2E59) , _31ED is abs(_2E59) , (integer(_2E59) ,
entidade(turma,_2E59) :- not var(_2E59) , integer(_2E59).
entidade(empregados,_2E59) :- entidade(professor,_2E59) ; entidade(funcionario,_2E59).
relacao(salario,parcial,valor,empregados).
relacao(media,parcial,media,aluno).
relacao(turma_aluno,parcial,turma,aluno).
relacao(turma_prof,parcial,turma,professor).
relacao(disc_prof,parcial,disciplina,professor).
tupla(salario,100000.0,julia).
tupla(salario,60000.0,joana).
tupla(salario,50000.0,joao).
tupla(salario,50000.0,jose).
tupla(salario,10000,paulo).
tupla(salario,10000,pedro).
tupla(turma_prof,30,jose).
tupla(media,8.5,alexandre).
tupla(media,5,alice).
tupla(turma_aluno,30,alberto).
tupla(turma_aluno,10,alexandre).
tupla(turma_aluno,20,alice).
tupla(turma_aluno,10,ana).
tupla(turma_prof,30,joana).
tupla(turma_prof,30,julia).
tupla(turma_prof,10,jose).
tupla(turma_prof,20,joao).
tupla(turma_prof,10,joao).
tupla(disc_prof,geografia,julia).
tupla(disc_prof,historia,joao).
tupla(disc_prof,portugues,joana).
tupla(disc_prof,matematica,jose).
end_of_file        1 File(s) copied
```

BIBLIOGRAFIA

[ARI86] Arity Corporation. The Arity/Prolog Programming Language. Concord 1986.

[CAS87] CASTILHO,J.M.V. & LEMOS,A.S. A Construção de Protótipos Básicos de Sistemas de Bancos de Dados : Uma Proposta. Porto Alegre, CPGCC / UFRGS, 1987. RP nº78.

[CHE76] CHEN,P.P. The Entity-Relationship Model - Towards a Unified View of Data. ACM Transactions on Database Systems, New York,1(1):páginas 9 a 36, Mar 1976.

[DAT85] DATE,C.J. Introdução a Sistemas de Bancos de Dados. Rio de Janeiro, Campus, 1985.

[FUR85] FURTADO,A.L. & SANTOS,C.S. Organização de Bancos de Dados. Rio de Janeiro, Campus, 1985.

[STE86] STERLING,L. & SHAPIRO,E. The Art of Prolog - Advanced Programming Techniques. Cambridge, The MIT Press, 1986.

[KOW79] KOWALSKI,R. Logic for Problem Solving. New York, North-Holland, 1979.

[TAR54] TARSKI,A. Introduction to Logic and to the Methodology of Deductive Sciences. Oxford, Oxford University Press, 1954.