Universidade Federal de Paraíba
Centro de Ciências e Teclologia
Coordenação de Pós-Graduação em Informática

Detectando Ambigüidade em Consultas Conceituais

Vicente Manuel Moreira Júnior

Campina Grande
Julho - 1992

Vicente Manuel Moreira Júnior

Detectando Ambigüidade em Consultas Conceituais

Dissertação apresentada ao Curso de MESTRADO EM INFORMÁTICA da Universidade Federal da Paraíba, em cumprimento às exigências para obtenção do Grau de Mestre.

Área de Concentração : Ciência da Computação

Maria de Fátima Q. Vieira Turnell
(Orientadora)

Pedro Sérgio Nicolletti
(Co-Orientador)

Campina Grande
Julho - 1992

M835d Moreira Junior, Vicente Manuel
Detectando ambiguidade em consultas conceituais / Vicente Manuel Moreira Junior. - Campina Grande, 1992.
90 f. : il.

Dissertacao (Mestrado em Informatica) - Universidade Federal da Paraiba, Centro de Ciencias e Tecnologia.

1. Banco de Dados-Consultas Conceituais 2. Sistemas de Computacao 3. Modelo de Dados 4. Ciencia da Computacao 5. Informatica 6. Dissertacao I. Turnell, Maria de Fatima Q., Ph.D. II. Nicolletti, Pedro Sergio, M.Sc. III. Universidade Federal da Paraiba - Campina Grande (PB) IV. Título

CDU 004.65(043)

DETECTANDO AMBIGÜIDADE EM CONSULTAS CONCEITUAIS

VICENTE MANUEL MOREIRA JÚNIOR

DISSERTAÇÃO APROVADA EM 10/07/92

Maria de Fatima Queiroz Vieira Turnell

MARIA DE FÁTIMA Q. V. TURNELL Ph.D.
(ORIENTADORA)

PEDRO SÉRGIO NICOLLETTI M.Sc.
(CO-ORIENTADOR)

RAIMUNDO HAROLDO DO CARMO CATUNDA M.Sc.
(COMPONENTE DA BANCA)

ULRICH SCHIEL Ph.D.
(COMPONENTE DA BANCA)

CAMPINA GRANDE - PB
JULHO - 92

Agradecimentos

Dona Antônia, minha mãe, Isaac, Adenauer, Denise, Deise e Débora, meus irmãos, pelo total apoio.;

A meu pai, em memória;

A meus grandes amigos Fernando (Mandi), Lauro (Satoro), Luiz (Baiano), Sérgio Murilo (Serjão), Jorge Augusto (Boi) e Bezerra por toda ajuda que só eles sabem;

A Fátima Turnell, Pedro Sérgio Nicolletti e Marcus Sampaio, meus orientadores e amigos, por confiarem a mim este trabalho;

A Rita Cavalcanti, Luana e Salete, por todo o carinho e amizade.

E a todos que sabem o quanto me ajudaram, seja de forma efetiva ou por pensamento como Duse Orrico e Paulo Cézar em Salvador ou minhas novas amigas Cristiana Accioly (Kika), Zélia e Márcia.

A todos, muito obrigado.

Vicente.

Sumário

Um dos principais objetivos subjacentes dos sistemas de computação é oferecer uma interação homem-máquina, a mais natural possível. Atender a esta necessidade, do ponto de vista de consultas à banco de dados, é o alvo dessa dissertação, visando detectar e expor ambigüidades ocorridas em consultas conceituais.

A interface utilizada para a realização das consultas é a Interface de Relação Universal [Korth84], que provê independência lógica de dados.

Para a detecção de ambigüidade, é utilizada a Abordagem Objeto Mínimo [Wald90]. A ambigüidade detectada é à nível do esquema do BD.

E utilizado um modelo de dados Entidade-Relacionamento Estendido (ERE), que é uma extensão ao modelo Entidade-Relacionamento (ER) de Chen [Chen76], por permitir especificar melhor a semântica dos dados, facilitando futuramente a interpretação de consultas.

Abstract

One of the main objectives of information systems is to offer a human-computer interaction the most natural possible. To meet this necessity, from the database query viewpoint, the objective of this dissertation is to detect and expose ambiguies in conceptual queries.

The interface used for query construction is the Universal Relation Interface [Korth84], which provides logical data independence.

For ambiguity detection, the Minimum Object Approach [Wald90] is used. Ambiguitiesare detected is at the DB scheme level.

An Extended Entity-Relationship data model (EER) is used, which is an extension of Chen's Entity-Relationship (ER) Data Model [Chen76], by allowing a better specification of the data semantics aiding the query interpretation.

Lista de Figuras

Sumário

Capítulo I

Capítulo II

Capítulo III

Capítulo IV

Capítulo I

1 - Introdução

Motivação

Os sistemas de gerenciamento de banco de dados (**SGBD**) disponíveis comercialmente (em sua maioria) têm como um dos seus objetivos, **isolar usuários e programadores da estrutura física dos dados**. Para isto, existem linguagens como SQL [Camberlin76], QUEL [Stonebraker76] e QBE [Zloof75], que propiciam a independência física de dados, não exigindo que o usuário possua conhecimento sobre a estrutura física em que os dados estão armazenados. Uma meta ainda maior da próxima geração de SGBDs, é a de **isolar o usuários da estrutura lógica dos dados**. As linguagens mencionadas acima apenas propiciam independência física de dados, forçando o usuário a navegar na sua estrutura lógica (relações, entidades, relacionamentos, etc.).

Para ilustrar a noção de independência lógica de dados, será apresentado um pequeno esquema de banco de dados (BD) de controle acadêmico. O modelo utilizado é uma simplificação do Modelo Entidade-Relacionamento de Chen [Chen76]. Temos **Matriculado** como sendo um conjunto de pares ordenados < **E,D**> significando: o estudante **E** está matriculado na disciplina **D.** Temos **Registrado** como sendo um conjunto de pares ordenados <**E,C**> significando: o estudante **E** está registrado no curso **C**. Temos **Oferecido** como sendo um conjunto de pares ordenados <**D** ,**C**> significando: a disciplina **D** é oferecida pelo curso **C**. *Matriculado*, *Registrado* e *Oferecido* são relações binárias na matemática. Este esquema está representado na Figura 1.1, onde os retângulos simbolizam conjuntos de entidades e as retas, conjuntos de relacionamentos.

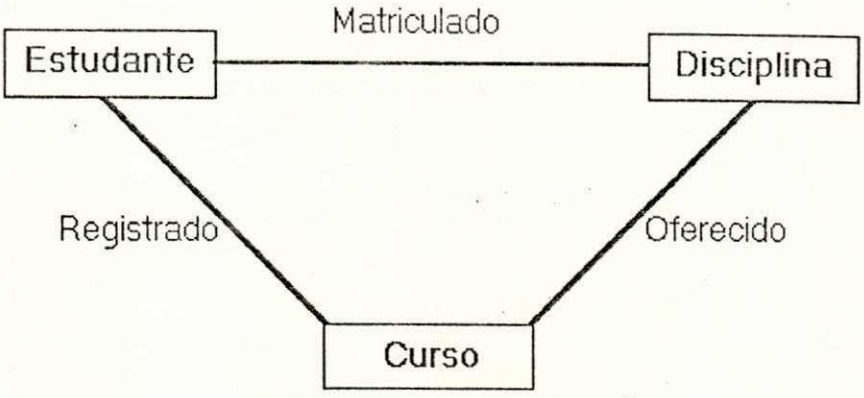


Fig. 1.1 - Esquema de um BD Universidade

Na maioria das linguagens de consulta, o usuário, mesmo leigo, precisaria conhecer o esquema em todos os seus detalhes. Por exemplo, considere a seguinte consulta em uma linguagem fictícia:

```
DISPLAY Estudante.Nome
PATH Matriculado, Oferecido
WHERE Curso.Nome = "Computação"
```

Esta consulta retorna os estudantes que estão matriculados em disciplinas do Curso de Computação. Dizemos que tal linguagem de consulta não oferece independência lógica de dados, necessitando, portanto, que o usuário conheça a estrutura lógica dos dados (Matriculado e Oferecido).

Algumas Propostas Existentes

Uma linha de pesquisa muito interessante baseia-se no conceito de Relação Universal (**RU**) [Maier82a], [Maier82b], [Maier83], [Ullman89], [Korth84], [Vardi88] (Seção 2.1, Hipótese da Relação Universal) que provê independência lógica de dados, apresentando uma visão simples do BD, exigindo que o usuário apenas conheça seus atributos para expressar suas consultas. Chamemos consultas neste nível de **consultas**

conceituais. Por exemplo, a seguinte consulta conceitual, equivale a consulta lógica apresentada anteriormente:

DISPLAY Nome.estudante **WHERE** Curso = "Computação"

Em uma interface de relação universal, o processador de consultas traduz uma consulta conceitual em uma consulta lógica. Deste modo, o processador, e não o usuário, é o responsável por navegar no BD. Algumas vezes, consultas conceituais podem ser traduzidas em mais de uma consulta lógica; neste caso, a consulta considerada **ambígua**, como ilustrado na Figura 1.2.

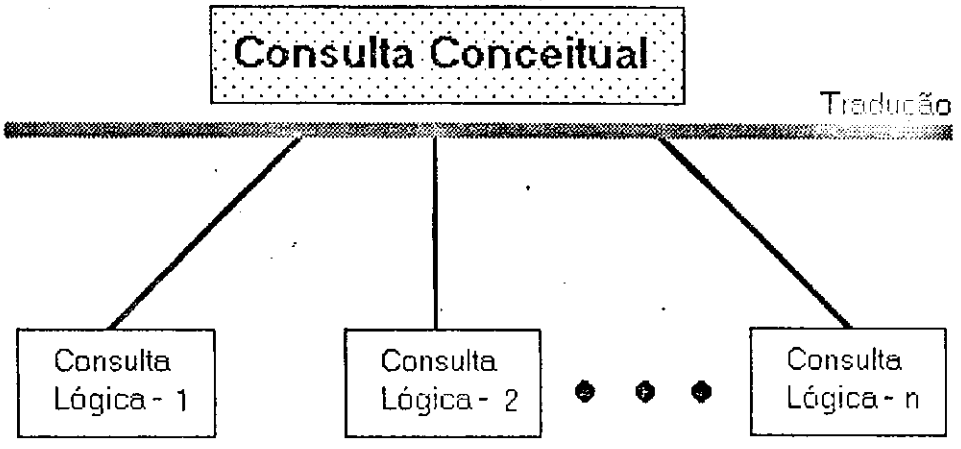


Fig. 1.2 - Consulta Conceitual Ambígua

É importante salientar aqui, que SGBDs experimentais baseados no conceito de RU como **System/U** [Korth84], da Universidade de Stanford - USA, não explanam consultas ambíguas, definindo por conta própria o caminho no esquema do BD para tentar expressar a intenção da consulta do usuário, caso seja detectada ambigüidade, gerando, portanto, um possível problema, pois a opção escolhida pelo processador, pode não ser a pretendida pelo usuário.

Existe também um trabalho intitulado **ERROL** [Marcowitz83]. ERROL é uma linguagem "natural", baseada em uma linguagem de consulta do modelo ER. Nela, o papel que uma entidade desempenha em um relacionamento é identificado por verbos. Esses

verbos são utilizados para retirar ambigüidade das consultas, embora ERROL não amplie o modelo ER para permitir o uso de preposições, o que restringe de forma significativa a "naturalidade" da consulta.

Um outro trabalho muito interessante, chamado **SQL-Fácil** [Bezerra89], utiliza a estratégia da RU para oferecer uma interface amigável para o usuário. SQL-Fácil serviu como base filosófica para a implementação do nosso trabalho, como por exemplo, na intenção de oferecer uma interface para BD's, na qual o usuário esteja realmente independente do Esquema do BD, como é o caso da Interface de RU.

DETECT

Este trabalho trata do problema da geração e exposição de frases simples, que explicam as várias interpretações de uma consulta conceitual, caso tenha sido detectada ambigüidade na mesma. O sistema é intitulado **DETECT** (Sistema Detector de Ambigüidade em Consultas Conceituais) o qual procura oferecer uma interface bastante amigável para o usuário final. Um outro aspecto do trabalho a ser destacado, é o da **implementação de uma estratégia automática para a detecção de ambigüidade**, baseado em [Wald90], utilizando-se da linguagem de RU como linguagem de consulta. Este projeto visa isolar o usuário não só da estrutura lógica dos dados, o que já é oferecido por trabalhos na mesma linha de pesquisa da RU citados anteriormente, como. também tem uma pretensão ainda maior, que é de **livrá-lo da necessidade de conhecer o Esquema do BD**. No ambiente DETECT, o usuário não precisa saber antecipadamente, nem mesmo os nomes dos atributos sobre os quais realizará suas consultas; estes atributos, estarão à sua disposição na interface de menus, que conterá os relacionamentos e entidades acompanhadas de seus respectivos atributos ver (Seção 4.2, Descrição da Interface com o Usuário). Esta facilidade parece decisiva na utilização do DETECT por usuários ocasionais ou não especialistas.

No Capítulo 3 (Modelo de Dados) é introduzido modelo de dados ER devidamente estendido para possibilitar a montagem de frases simples de uma forma mais natural. O acréscimo ao modelo ER diz respeito a adição de verbos e preposições. Os substantivos surgem naturalmente dos conjuntos de entidades e relacionamentos contidos no esquema. A escolha de se adaptar um modelo de dados, deu-se pelo fato de tentarmos desvincular o

trabalho de um modelo ER específico para a interface de RU, o que não ocorreria caso fosse escolhido o modelo ER adaptado especificamente para a interface de RU [Brady85]. Não nos interessa, prioritariamente, o modelo de dados utilizado ou problemas técnicos existentes na RU, e sim resolver o problema da detecção e exposição da ambigüidade que ocorre em algumas consultas conceituais.

A Figura 1.3, mostra um diagrama hierárquico do Sistema DETECT, seguido da descrição de seus módulos.

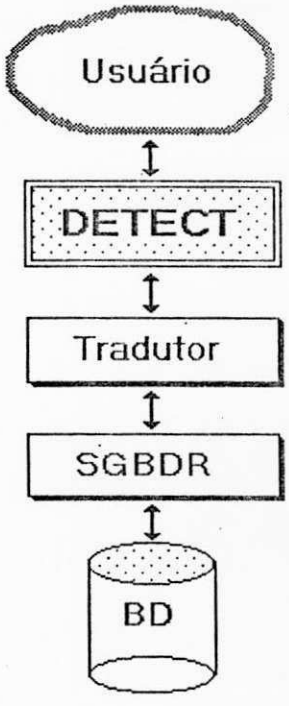


Fig. 1.3 - Hierarquia do Sistema DETECT

O módulo **DETECT**, é um ambiente para o processamento de consultas a BD's relacionais. As consultas são realizadas utilizando-se a Interface de RU. Este módulo é o responsável pela detecção de ambigüidade nas consultas realizadas ao BD. O sistema apresentará possíveis opções de reformulação da consulta, para que a ambigüidade seja retirada. Exemplos de consultas ambíguas serão apresentados na Seção 6.2 (Explanação de Consultas Ambíguas). A consulta realizada pelo usuário só poderá passar ao módulo TRADUTOR, após retirada a ambigüidade da mesma.

O módulo **TRADUTOR** é o responsável por receber as consultas, sem ambigüidade, do módulo DETECT, transformando-as em consultas na linguagem SQL [Chamberlin76]. Uma alternativa para representar o módulo TRADUTOR, seria a utilização do sistema SQL-Fácil [Bezerra89], que efetua a tradução de uma consulta realizada na linguagem de RU para uma consulta em SQL. Alguns problemas inviabilizaram esta conexão, como por exemplo: DETECT utiliza a estratégia **Objeto Mínimo** (Seção 2.4.3) para interpretar as consultas realizadas na linguagem de RU, enquanto que SQL-Fácil, utiliza a estratégia **Objeto Máximo** (Seção 2.4.2); a linguagem de definição de dados e o modelo de dados utilizados por SQL-Fácil, não oferecem um bom suporte para a geração de frases simples, que é um dos principais objetivos do DETECT. Estas e outras incompatibilidades, inviabilizaram a conexão entre os dois sistemas, embora ambos sigam a mesma filosofia da RU.

O módulo intitulado **SGBDR**, no qual pode ser utilizado por exemplo DB2[Date89], INGRES [Stonebraker76] ou Oracle [Oracle84], recebe uma consulta em SQL do módulo TRADUTOR e a efetua no BD, devolvendo o resultado ao módulo TRADUTOR, para ser exposto ao usuário.

Neste capítulo, foi descrito um panorama geral do comportamento das Interfaces de RU e quais as estratégias utilizadas para se resolver o problema da ambigüidade, assim como uma apresentação dos objetivos deste trabalho.

No Capítulo II (Conceitos Básicos) é apresentada uma avaliação mais profunda das Interfaces de RU, acompanhada dos conceitos básicos necessários para que se possa trabalhar com estas interfaces. Consta também um comparativo entre as duas abordagens na qual se apoia a RU (Objeto Máximo e Objeto Mínimo) e uma análise das interfaces de Linguagem Natural.

No Capítulo III (Modelo de Dados) é apresentado primeiramente o Modelo de Dados Entidade-Relacionamento Estendido (ERE) que foi adaptado para melhor capturar a semântica do mundo real e facilitar a formulação das frases simples na explanação de ambigüidades. Em seguida é apresentada a Linguagem de Definição de Esquemas (LDE), utilizada para descrever o esquema do BD projetado no Modelo de Dados ERE. Finalmente é descrita a Linguagem de Consulta do DETECT (LC-DETECT), que é uma adaptação da linguagem de consulta utilizada em SQL-Fácil [Bezerra89].

O Capítulo IV (Arquitetura do Sistema) descreve a estrutura lógica do sistema DETECT, seus principais módulos e funcionalidades, assim como a interface do sistema com o usuário, seguida de uma simulação de utilização do sistema.

No Capítulo V (Estrutura de Dados) é apresentada a estrutura de dados utilizada para dar suporte ao sistema, incluindo as estruturas originadas a partir do modelo de dados ERE e o arquivo de objetos. Este último, é o responsável por armazenar os objetos sobre os quais o usuário pode realizar suas consultas.

O Capítulo VI (Explanação de Ambigüidades) contém algumas das estratégias utilizadas para se realizar a explanação de consultas ambíguas e a geração de frases simples. É apresentada uma estratégia geral para explanar ambigüidades (utilizada pelo DETECT) e um mecanismo para se gerar sentenças (frases mais completas).

O último capítulo se refere às conclusões e contribuições obtidas com o trabalho. Para trabalhos futuros, são feitas algumas recomendações e sugestões relevantes para implementar melhorias no DETECT.

No Apêndice A, é apresentado um esquema de BD de um Controle Acadêmico, seguido de sua descrição em LDE. No Apêndice B, é apresentado um esquema de BD de um Simpósio, seguido de sua descrição em LDE e no Apêndice C, são apresentados com mais detalhes, os tipos de ambigüidades passíveis de ocorrerem nas interfaces de Linguagem Natural.

Capítulo II

2 - Conceitos Básicos

Neste capítulo, serão apresentados os conceitos básicos para que se possa trabalhar com a Hipótese da RU. Serão apresentadas as características da RU, assim como alguns sistemas que as utilizam. Será feito um paralelo entre uma interface de Linguagem Natural e uma Interface de RU. Em seguida, serão apresentados os conceitos de Objeto, as Abordagens Objeto Máximo e Mínimo, assim como uma comparação entre elas.

2.1- Interface de Relação Universal

A independência lógica de dados é um dos objetivos da Hipótese da Relação Universal (**RU**). Já existem sistemas protótipos que implementam o conceito da RU, como por exemplo System/U [Korth84] ou FIDL [Addis82]. No entanto, a Hipótese da RU possui alguns problemas apontados por Kent em [Kent81], que apresenta alguns pontos não muito claros na proposta da RU, como por exemplo, quando questiona a possibilidade da ocorrência de um mesmo nome de atributo definido em duas relações, estarem sob o mesmo domínio; ou se estes atributos são agrupáveis; ou no caso de ocorrer uma junção destas colunas, como seriam renomeadas para eliminar nomes duplicados de colunas? Estas e outras questões foram esclarecidas por Ullman em [Ullman83], no qual são apresentadas algumas premissas não consideradas por Kent, para que se possa utilizar a RU.

Para usuários e programadores, a RU significa o desaparecimento da navegação lógica no BD. Por exemplo, considere as seguintes relações para um BD de uma companhia aérea:

COMPANHIA (Nome-Comp)
AVIÃO (#Num-avião, Tipo-avião, Nome-comp)
VÔO (#Num-vôo, Destino-vôo, #Num-avião)
RESERVA (#Num-Res, #Num-vôo, #Num-Cliente, Poltrona, Nome-Cliente)

Sem a hipótese da RU, para se obter os tipos de aviões nos quais um cliente 'X' possui reserva, necessitaria que o usuário do sistema realizasse os seguintes passos junto ao SGBD:

1 - Selecione as tuplas da relação RESERVA, para o cliente 'X' e projete sobre #Num-vôo. Com isto, se obtém o conjunto de vôos no qual o cliente 'X' possui reserva.

2 - Realize uma junção da relação resultante em (1) com a relação Vôo, e projete-a em #Num-avião. Com isto se obtém o conjunto de aviões associados aos vôos;

3 - Realize a junção dos resultados obtidos em (2) com a relação AVIÃO e projete-a sobre Tipo-avião. Obtendo, assim, o resultado desejado.

Realizando a consulta anterior através da linguagem de consulta do System/U, que utiliza a hipótese da RU, a consulta seria efetuada da seguinte forma:

RETRIEVE (Tipo-avião) **WHERE** Nome-cliente = 'X'

Vê-se, portanto, que o usuário não precisa conhecer os nomes das relações envolvidas no processamento da consulta ou quais as junções necessárias para a realização da mesma. O que ele precisa saber, é apenas sobre quais atributos realizará sua consulta, independentemente destes atributos estarem contidos em uma determinada relação. *O usuário vê o BD como uma única relação.* A relação RESERVA, poderia ser armazenada como descrita anteriormente, ou como duas relações normalizadas RESERVA (# Num-res, # Num-vôo, # Num-Cliente, Poltrona) e CLIENTE (# Num-cliente, Nome-cliente). A RU provê absoluta independência lógica de dados e uma sintaxe bastante simplificada para o processamento de consultas e especificação do BD.

A RU é uma *relação imaginária*, que representa todos os dados de um BD, embora existam sistemas como FIDL [Addis82], que é baseado em uma implementação física da RU. A linguagem de consulta utilizada por uma interface de RU é muito mais simples do que as linguagens de consulta tipicamente relacionais, porque é preciso apenas mencionar os atributos desejados, descartando-se as junções. Assim como nas interfaces para BD's baseadas em Linguagem Natural, na interface de RU, é necessária a tradução da consulta

da linguagem de RU para uma linguagem de consulta adequada para acessar o BD, como por exemplo SQL, para acessar um BD relacional.

O sistema DETECT, não se preocupa com problemas relativos à tradução de uma consulta à nível de RU para uma consulta, por exemplo, em SQL, interessa-se apenas em reconhecer, através da consulta elaborada, se a mesma contém ambigüidade, baseado no esquema do BD. Traduções deste tipo são realizadas por sistemas como por exemplo: SQL-Fácil, System/U, DURST, FIDL, etc. (veja Seção 2.2).

A principal motivação para se utilizar uma interface de RU, é que ela permite que usuários não especializados e com um mínimo de treinamento, efetuem consultas ao BD sem a necessidade de qualquer conhecimento sobre como o BD está estruturado logicamente.

Um exemplo apresentado em SQL-Fácil [Bezerra89], demonstra a facilidade da formulação de uma consulta realizada em uma interface de RU, em relação a uma consulta realizada utilizando-se da linguagem de consulta SQL. Para demonstrar este fato, será apresentada uma consulta utilizando a Hipótese da RU, realizada através da linguagem de consulta projetada para o DETECT ao invés da linguagem de consulta do sistema SQL-Fácil. Esta consulta tomará como base o esquema do BD apresentado no Apêndice B. Considere a seguinte consulta:

Consulta: 'Quais os artigos da conferência A, seção Banco de Dados, que foram julgados por João ?'

SQL:
```
SELECT ARTIGOS.Nome-artigo
FROM JULGADORES, ARTIGOS, SELECIONAM, SEÇÖES,
CONFERENCIAS
WHERE CONFERENCIAS.Nome-confer = 'A'
   AND SEÇÖES.Nome-seção = 'Banco de Dados'
   AND JULGADORES.Nome-julgador = 'João'
   AND JULGADORES.Nome-julgador = SELECIONAM.Nome-selec
   AND SELECIONAM.Codigo-selec = ARTIGOS.Codigo-artigo
   AND ARTIGOS.Nome-seção = SEÇÖES.Nome-seção
   AND SEÇÖES.Nome-confer = CONFERENCIAS.Nome-confer
```

DETECT:
```
LISTE Nome-artigo
   ONDE Nome-confer = 'A' E Nome-seção = 'Banco de Dados' E
        Nome-julgador = 'João'
```

As consultas anteriores, tomaram como base o esquema relacional a seguir:

JULGADORES (Nome-julgador, País-Julg)
SELECIONA (Nome-selec, Codigo-selec)
ARTIGOS (Codigo-artigo, Nome-artigo, Nome-seção, Nome-confer)
CONFERÊNCIAS (Nome-Confer, País-confer)
SEÇÕES (Nome-seção, Nome-confer, Local-seção)

o exemplo anterior demonstra claramente a simplicidade de se processar consultas em uma interface de RU, livrando o usuário de navegar na estrutura lógica do BD, que é expressa através das junções e dos nomes das relações.

2.2 - Sistemas que Utilizam a Interface da Relação Universal

Nesta sub-seção será apresentado um breve resumo das principais características de alguns sistemas que utilizam a hipótese da RU para implementar suas interfaces para BD's.

2.2.1- System/U

System/U [Korth84] é um SGBD que utiliza a hipótese da RU como interface para acesso a BD's. Foi projetado na Universidade de Stanford-USA. Este sistema foi implementado na linguagem C, em ambiente UNIX. System/U baseia-se no conceito de atributos, objetos e objetos maximais (Seção 2.4 Objetos). Possui uma linguagem de definição de dados que permite a declaração de atributos, relações, dependências funcionais e objetos, para serem utilizados posteriormente na interpretação das consultas dos usuários. A linguagem de consulta é essencialmente QUEL [Stonebraker76], com uma diferença: não é necessária, na maioria dos casos, a utilização de **variáveis-tupla**[2] [Date86].

A gramática da linguagem de consulta do System/U é da seguinte forma:

RETRIEVE < Lista-de-atributos>
WHERE <Qualificação>

2. Variável-tupla é uma variável cujos únicos valores permitidos são tuplas da própria relação.

Onde, <Lista-de-atributos> são os nomes dos atributos separados por ',' (vírgula) e <Qualificação> são expressões lógicas de seleção sobre os atributos.

O System/U adiciona internamente uma variável tupla para inferir um caminho lógico, caso seja detectada ambigüidade na consulta do usuário, porém esta declaração pode também ser realizada pelo próprio usuário caso este conheça o esquema do BD, o que não nos parece muito interessante. Vale ressaltar, que se o usuário não perceber que sua consulta é ambígua, o processador de consultas do System/U escolherá por conta própria, um dos caminho possíveis para realizar a consulta. A seguir serão apresentados dois exemplos simples de consultas formuladas utilizando-se da linguagem de consulta do System/U, baseado no esquema de BD de um Controle Acadêmico (Apêndice A).

Consulta 1: 'Quais os nomes dos alunos que são do curso de Computação cujo ano de admissão seja posterior a 1990 ?'

```
System/U: RETRIEVE Nome-aluno
          WHERE Nome-curso = 'Computação'
          AND Ano-admissão > 1990
```

Consulta 2: 'Quais os códigos dos professores que estão lecionando a disciplina Eletrônica Digital ?'

```
System/U: RETRIEVE Codigo-prof
          WHERE Nome-disc = 'Eletrônica Digital'
```

Estes exemplos demonstram, mais uma vez, a facilidade de se consultar um BD que utilize uma interface de RU.

2.2.2 - FIDL

O sistema FIDL (Flexible Interrogation and Declaration Language) é um SGBD que utiliza a hipótese de RU. Foi desenvolvido pela International Computer Ltd, na Inglaterra [Addis82]. O FIDL é baseado na implementação física da RU. A RU que é chamada de JNF (Joint Normal Form) é armazenada em um hardware especial, chamado CAFS (Content-Addressable File Store).

O modelo de dados do FIDL é baseado no conceito de *Implication Network*, que é uma descrição gráfica das dependências funcionais entre as relações do BD. Estas implicações servem para se realizar a interpretação das consultas realizadas ao BD.

Funcionam como se fossem os objetos definidos para o System/U. Para se obter informações do BD, junções são formuladas através das dependências funcionais representadas pelas *Implications Network*.

2.2.3 - DURST

DURST (Datenbank mit Universalrelation-Scnittstelle) [Biskup83] é um sistema desenvolvido pela 'University of Dortmund' na Alemanha. É um sistema similar ao FIDL (Seção 2.2.2), porém não inclui uma implementação física da RU.

Este sistema possui uma característica muito interessante, que é a de projetar esquemas de BD sem ambigüidade. Isto é possível através da criação do chamado *atributo característico*, que pode servir como substituto do nome de relações e assim, serem utilizados para remover ambigüidades na interpertação da consulta do usuário. Estes atributos são utilizados para guiar logicamente a consulta no BD.

O problema que vemos neste tipo de estratégia, é que desta forma, o usuário estará cada vez mais preso ao esquema lógico do BD, necessitando conhecê-lo detalhadamente, para que possa realizar suas consultas sem correr o risco de realizar consultas ambíguas.

2.2.4 - AURICAL

AURICAL [Kuck82] é um sistema baseado na hipótese da RU via o modelo CODASYL. É um sistema de RU da 'University of Illinois'. Está implementado na linguagem FORTRAN, utilizando-se de uma interface de um BD CODASYL.

2.3 - Interface de Linguagem Natural

Uma Interface de Linguagem Natural (LN) é uma interface muito interessante para BD's, onde os usuários se expressam em termos mais familiares a si e não nos termos utilizados no esquema do BD. O sistema é quem se responsabiliza pela interpretação da intenção da consulta formulada pelo usuário, traduzindo-a para uma linguagem reconhecida pelo esquema do BD.

Existem alguns trabalhos apresentados em [Crout81], [Grosz83] e [Kaplan83], que servem de interface de LN para BD's, porém, é importante ressaltar que o uso da LN como interface, implica em algumas restrições, como por exemplo, a dependência da aplicação sobre a qual a LN será utilizada, e também, como acontece com alguns sistemas que utilizam uma linguagem que oferece independência lógica de dados, não foi resolvido o problema da geração de frases não interpretáveis pelo sistema, limitando assim, o poder de expressão do usuário.

O uso da LN como interface para BD's é de extrema importância, pois, através dela, podemos oferecer uma independência lógica de dados para o usuário, oferecendo uma linguagem de consulta de fácil assimilação pelo usuário, facilitando, portanto, o processamento de suas consultas.

Boguraev em [Boguraev84], tentou projetar uma LN para servir de interface para BD's que oferecesse portabilidade e facilidade de integração com o usuário, porém, a portabilidade só foi conseguida em testes com BD's não muito complexos, caindo novamente no problema da dependência do domínio da aplicação, embora seja uma contribuição extremamente importante.

A escolha por utilizar a interface de RU e não uma interface de LN, ocorreu pelo fato de que a própria LN permite a formulação de consultas ambíguas, o que seria uma preocupação a mais, e desvirtuaria o objetivo básico deste trabalho, que é o de *verificar ambigüidade à nível do esquema do BD e não da linguagem de consulta.*

A LN aceita uma infinidade de combinações possíveis para se formular uma consulta ao BD, podendo, portanto, causar alguns problemas para o usuário na realização de suas consultas. Este tipo de problema provocaria possivelmente a execução de um passo a mais na realização das mesmas, pois, caso a consulta em LN fosse ambígua, o sistema teria que pedir ao usuário para reformulá-la, retirando a *ambigüidade à nível da linguagem de consulta*, e só após, é que se checaria a *ambigüidade à nível do esquema do BD*. Caso fosse detectada ambigüidade ao nível do esquema do BD, o sistema teria que pedir para o usuário reformular mais uma vez sua consulta, já que outro tipo de ambigüidade foi detectado.

"A ambigüidade é um dos problemas computacionais dos mais difíceis em uma interface de LN, por ser muito complicado identificar as várias interpretações possíveis para uma frase" . [Savadovsky88]. Existem três tipos básicos de ambigüidades, os quais são: Ambigüidade *Léxica*, *Sintática* e *Semântica*, que são apresentados em detalhes no Apêndice C.

A questão da ambigüidade, a nossa falta de habilidade com LN e a existência de outros problemas que não fazem parte dos objetivos deste trabalho, foram alguns dos fatores que levaram-nos a optar pela interface de RU ao invés da interface de LN para ser utilizada como linguagem de consulta do sistema DETECT.

A seguir será apresentada a conceituação necessária para se trabalhar com a Hipótese da RU.

2.4 - Objetos

Entidades e relacionamentos estabelecem cada um, uma associação primitiva entre um conjunto de atributos. Por exemplo:

Estudante = {Codigo-Estudante, Nome-Estudante} e
Registrado = {Nome-Curso, Codigo-Estudante, Ano-Admissão}

A relação Estudante, possui a associação com os atributos {Codigo-estudante e Nome-estudante} e a relação Registrado, uma associação direta com (Nome-curso, Codigo-estudante e Ano-Admissão). Um objeto X é um conjunto de associações primitivas. Por exemplo, os seguintes objetos, entre outros, podem ser definidos para o esquema de BD apresentado na Figura 1.1:

Obj-1 - { Estudante }
Obj-2 - { Curso }
Obj-3 - { Estudante, Registrado }
Obj-4 - { Disciplina, Matriculado }
Obj-5 - { Estudante, Matriculado, Disciplina }

Um conjunto de objetos em um esquema de BD pode ser definido *explicitamente*, por um Administrador de BD (ABD) como em [Aho] ou [Maier82b], que é a forma utilizada neste trabalho, ou implicitamente, utilizando-se de alguma *construção automática* como em [Maier83].

A princípio, apenas associações significativas entre atributos são expressas por objetos. Isto é, um *conjunto arbitrário dos esquemas das relações do BD, não necessariamente representa um objeto*. Há alguma subjetividade na determinação do conjunto de objetos para um dado esquema de BD. Uma diretriz utilizada para se determinar os objetos, é que as ligações sejam construídas corretamente, e que as entidades definidas no objeto tenham uma correspondência lógica entre si (dependência funcional). Por exemplo, em um BD hipotético de um Departamento, um objeto que possuísse as relações {Estudante, Funcionário}, possivelmente não teria sentido, pois, não existe, a princípio, uma associação direta entre as relações Estudante e Funcionário.

Os objetos funcionam como um dispositivo de *visões* [Date86], sobre as quais o usuário realiza suas consultas. Caso o usuário tenha mencionado em sua consulta um atributo que não pertença a alguma das relações contidas no objeto sobre as quais ele pode realizar suas consultas, o sistema retornaria uma mensagem, informando-o da impossibilidade de realizar aquela consulta. O ABD controla quais objetos serão carregados para que um determinado usuário possa consultar o BD. Este mecanismo implementa também a segurança através das visões, pois, o usuário só terá acesso às informações que lhes forem permitidas através dos objetos que o ABD colocou à sua disposição. O usuário poderá solicitar ao ABD, o carregamento de outros objetos para realizar suas consultas, caso tenha permissão para utilizá-los. Por exemplo, um funcionário que não fizer parte da diretoria, não terá acesso aos atributos correspondentes ao lucro da empresa ou suas aplicações no mercado.

A Seção 5.2 (Arquivo de Objetos) mostra a sintaxe dos objetos definidos pelo ABD, juntamente com exemplos de objetos que poderiam definidos para o BD de Controle Acadêmico (Apêndice A).

Na Seção 2.4.1 (Conexão) será mostrado como se realiza a interpretação da consulta formulada pelo usuário em relação aos objetos à sua disposição.

2.4.1 - Conexão

Uma das grandes dificuldades dos sistemas que utilizam a RU é o de definir quais os objetos que abrangem as conexões. **Uma conexão, é o conjunto de atributos que aparece na consulta do usuário**. A conexão é a ponte entre o BD como uma coleção de relações e

a semântica que estas relações representam. Uma conexão é vista pelo usuário como pertencente a uma única relação, mas, de fato, pode pertencer a diversas relações do conjunto de objetos. Vejamos qual a estratégia geralmente utilizada por um processador de consultas, em uma interface de RU, para determinar os objetos da consulta, baseada no esquema da Figura 1.1:

LISTE Nume-estudante **WHERE** Ano-admissão = 1990

A conexão é C = {Nome-estudante, Ano-admissão}. Dizemos que, se um objeto X, contém C, então X é uma **interpretação** possível para a consulta. Considerando os cinco objetos definidos na Seção 2.4, o único objeto que contém os atributos representados em C, é {Estudante, Registrado}, este objeto é também **mínimo**, porque não existe um outro objeto que, por sua vez, também contenha C.

O objetivo do processador de consultas é o de detectar a conexão que o usuário deseja. No nosso processador todos os objetos calculados para a consulta são mínimos (**Abordagem Objeto Mínimo** [Wald90], Seção 2.4.3). Assim, a única interpretação para a consulta em questão, é a de que o usuário deseja todos os nomes dos estudantes cujo ano de admissão seja igual a 1990.

Maier e Ullman em [Maier83], trabalham com a **Abordagem Objeto Máximo** (Seção 2.4.2) utilizando-se de variáveis-tupla, para evitar ambigüidades. Em contrapartida, a interpretação inferida, pode não corresponder a intenção do usuário, caso este não indique o caminho lógico a percorrer, através das variáveis-tupla.

As Seções 2.4.2 e 2.4.3 apresentam mais detalhadamente as abordagens objeto máximo e mínimo respectivamente, e na Seção 2.4.4 é feito um comparativo entre as duas abordagens e uma justificativa da nossa escolha pela Abordagem Objeto Mínimo.

2.4.2 - Abordagem Objeto Máximo

Nesta seção será descrita a Abordagem Objeto Máximo (ObjMax) [Maier83], que é uma estratégia utilizada para se interpretar consultas a BD's realizadas em uma linguagem de RU.

Objeto, no contexto da Abordagem ObjMax, é o menor conjunto de atributos que tem uma significação coletiva [Korth83]. Supõe-se que cada objeto está contido em apenas uma relação.

Objeto Máximo é o conjunto de objetos do qual o sistema se utiliza para navegar na estrutura do BD. O exemplo clássico, a seguir, mostra a noção de objetos e objeto máximo. No exemplo serão descritas as operações executadas por uma Instituição Bancária. Os atributos são:

- Banco
- Conta (Conta do cliente)
- Saldo
- Empréstimo
- Total-empr (Total do empréstimo)
- Cliente
- Endereço-cliente

Cliente relaciona-se através de uma ramificação como sendo possuidor de uma Conta ou Empréstimo. Ambos (Conta e Empréstimo) podem ser compartilhados por alguns Clientes, mas cada conta possui apenas um dono (neste modêlo). O esquema do BD está apresentado na Figura 2.4.2.1. Os objetos são {Conta / Saldo, Conta / Cliente, Conta / Banco, Cliente / Endereço-cliente, Cliente / Empréstimo, Empréstimo / Banco e Empréstimo / Total-empr}.

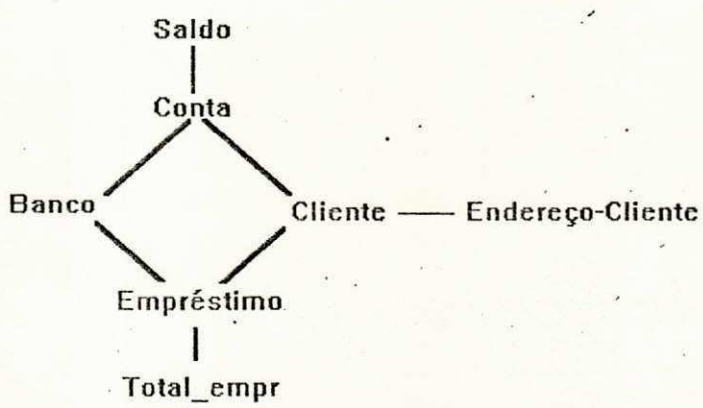


Fig. 2.4.2.1 - Esquema do BD Instituição Bancária

Para ilustrar a Abordagem ObjMax, será descrito como se comporta o System/U (Seção 2.1.1) que utiliza esta abordagem. No System/U, a navegação segue a *dependência funcional*[2] ([Date86]) entre os atributos.

No exemplo da Figura 2.4.2.1, Banco e Saldo são funcionalmente dependentes (FD's) de Conta. Similarmente, Banco e Total-empr são FD's de Empréstimo. Por fim, Endereço-cliente é FD de Cliente e Saldo é FD de Conta.

As dependências mencionadas anteriormente, dão origem a dois objetos máximos, que são apresentados na Figura 2.4.2.2. Não há um relacionamento básico entre, por exemplo, Empréstimo e Conta, porque estes atributos não ocorrem no mesmo objeto máximo.

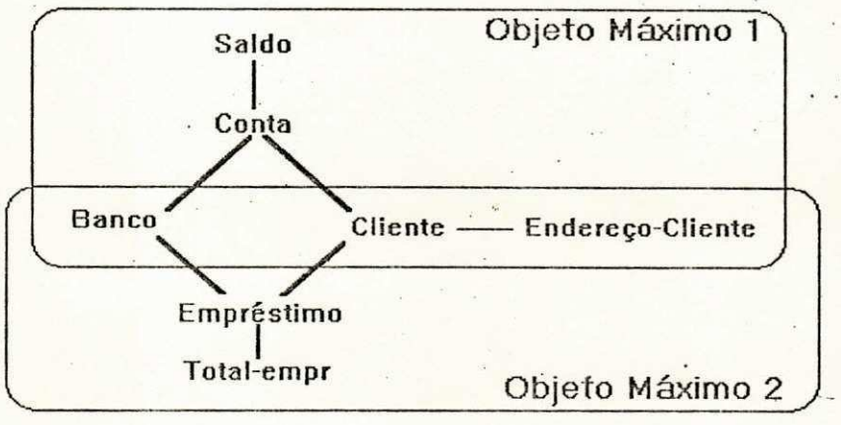


Fig. 2.4.2.2 - Objetos Máximos no BD Instituição Bancária

Objetos máximos são criados para que se possa interpretar a conexão (Seção 2.4.1) da consulta do usuário no System/U. Por exemplo, poderíamos realizar a seguinte consulta no System/U.

---

2. Dada uma relação R, o atributo Y de R é funcionalmente dependente do atributo X de R, se e somente se cada valor de X em R, tiver a ele associado precisamente um valor de Y em R (a qualquer momento).

RETRIEVE (Banco)
WHERE Cliente = 'Pedro'

Primeiramente seria computada a conexão, que no caso é Q = {Banco, Cliente}. Então, seriam pesquisados no conjunto de objetos máximos, quais deles contém a conexão Q. No caso, seríam considerados os dois objetos máximos da Figura 2.4.2.2. Finalmente, o sistema realizaria uma junção dos objetos em cada objeto máximo selecionado, projetaria sobre os atributos mencionados e efetuaria a união dos resultados para serem selecionados aqueles indicados na consulta. Neste caso, teríamos: Todos os bancos em que Pedro possui uma conta ou empréstimo.

Na próxima seção, será apresentada a Abordagem Objeto Mínimo, que é outra forma utilizada para se tentar interpretar o significado da consulta do usuário.

2.4.3 Abordagem Objeto Mínimo

Para dotar o interpretador de consultas da capacidade de identificar quais as possíveis interpretações para uma determinada consulta, existe também uma estratégia denoninada **Abordagem Objeto Mínimo** (ObjMin) [Wald90], que pelo próprio nome já sugere o oposto da abordagem apresentada anteriormente, Abordagem Objeto Máximo (Seção 2.4.2).

Seja X = $\{x_1, x_2, ..., x_n\}$ um objeto. O conjunto de atributos no objeto X, denotado por ATTR(X), é $x_1 \cup x_2 \cup ... \cup x_n$. Ou seja, ATTR(X) é a união dos atributos de todas as classes (entidades ou relacionamentos) contídas no objeto X. Por exemplo: Sejam as seguintes relações do BD apresentado na Figura 1.1:

ESTUDANTE (#Cod-Estud, Nome-Estud, Idade-Estud)
MATRICULA (Nota)
CURSO (#Cod-Curso, Nome-Curso, Depto-Curso)

Seja X um objeto do tipo X = {Estudante, Matricula, Curso}. Então, teremos ATTR(X) = {#Cod-Estud, Nome-Estud, Idade-Estud, Nota, #Cod-Curso, Nome-Curso, Depto-Curso}

Definições:

1 - Um objeto X cobre uma conexão Q, se Q $\subseteq$ ATTR(X).

2 - Um objeto X é um **objeto mínimo** para a conexão Q, se X cobre Q e não existe um objeto Y onde Y cobre Q e Y está contido em X.

Sejam X e Y objetos cobrindo a conexão Q. A Figura 2.4.3 demonstra que, apesar do objeto X cobrir a conexão Q, X não é um objeto mínimo para Q, pois existe um objeto Y contido em X, que também cobre Q, portanto, Y é o objeto mínimo para Q.

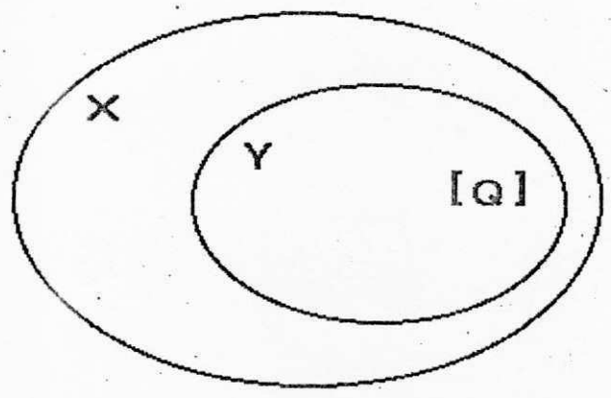


Fig. 2.4.3 - Cobertura da Conexão Q, pelos Objetos X e Y

Exemplo: Considere os seguintes objetos definidos para o BD de Controle Acadêmico (Apêndice A):

Obj-1 = { Estudante }
Obj-2 = { Estudante, Registra }
Obj-3 = { Estudante, Registra, Curso }
Obj-4 = { Curso, Oferece, Disciplina }

Considere a seguinte consulta:

LISTE Nome-estud **ONDE** Curso-estud = 'Computação'

A conexão Q = {Nome-estud, Curso-estud} que são atributos mencionados na consulta do usuário. Com esta conexão, os objetos selecionados são (Obj-1, Obj-2 e Obj-3), já que Q contém os atributos da relação Estudante e esta se encontra em Obj-1, Obj-2 e Obj-3.

Utilizando-se da Abordagem ObjMin e da definição (2), mencionada anteriormente, como Obj-1 está contido em Obj-2, que por sua vez, está contido em Obj-3 e Obj-1 cobre a conexão Q, o objeto mínimo selecionado será Obj-1, pois ele é suficiente para que o processador de consultas interprete a consulta do usuário, ou seja, ATTR(Obj-1) cobre totalmente a conexão Q, e é mínimo.

2.4.4 - Abordagem Objeto Máximo X Abordagem Objeto Mínimo

Através da pesquisa bibliográfica, notou-se duas correntes, no que diz respeito a escolha de qual estratégia utilizar para se interpretar a consulta do usuário. Destacam-se os sistemas System/U e Durst (Seção 2.1.1 e 2.2.3 respectivamente) e SQL- Fácil [Bezerra89], utilizando-se da Abordagem ObjMax e no outro lado em [Wald90] e DETECT [Moreira92], utilizando a Abordagem ObjMin.

Chegou-se a conclusão, através desta pesquisa, de que a escolha entre essas abordagens, está diretamente ligada à intenção que vem posteriormente a interpretação da consulta. Foi observado que os sistemas que utilizam a Abordagem ObjMax, visam evitar o problema da ambigüidade, necessitando que o próprio usuário perceba o problema e resolva-o, enquanto que os sistemas que utilizam a Abordagem ObjMin, visam livrar o usuário do conhecimento da estrutura lógica dos dados, informando-o se sua consulta realizada gerou ou não ambigüidade (mais de uma interpretação possível para a consulta).

É importante observar, que os sistemas que dizem oferecer independência lógica de dados, na verdade não a oferecem por completo. Isto é observado tanto no System/U como no Durst, que utilizam-se das variáveis-tupla descritas pelo usuário, para navegar na estrutura lógica do BD sem ambigüidade ou no sistema SQL-Fácil, através da cláusula 'USING', na qual o usuário descreve o caminho lógico a ser seguido pelo sistema, através do esquema do BD.

Observou-se também, que os sistemas mencionados acima, que utilizam a Abordagem ObjMax, não visam explanar a ambigüidade detectada para o usuário, enquanto que, os que utilizam a Abordagem ObjMin, o fazem.

O problema aparente, é que na abordagem ObjMax, todas as classes (entidades ou relacionamentos) contidas no objeto, são consideradas para se interpretar a consulta do usuário, o que se fosse convertido para gerar frases, provocaria a formação de frases extensas e possivelmente ambíguas, enquanto que na Abordagem ObjMin, apenas as classes que contém os atributos mencionados na consulta do usuário são consideradas, podendo assim, serem geradas frases simples que expressem claramente as interpretações inferidas pelo processador de consultas. Portanto, a Abordagem ObjMin, apresentou-se mais adequada aos objetivos deste trabalho

No próximo capítulo serão apresentados o Modelo de Dados Entidade-Relacionamento Estendido, a Linguagem de Definição de Esquemas e a Linguagem de Consulta, que foram projetados para tornar possível a implementação de todas as idéias apresentadas neste capítulo.

Capítulo III

3 - Modelo de Dados

Neste capítulo será apresentado um modelo de dados adaptado, que é uma extensão do Modelo Entidade-Relacionamento de Chen [Chen76], assim como a linguagem de definição de esquema, que mapeia as estruturas projetadas no modelo de dados estendido e a linguagem de consulta, projetada para acessar informações do BD.

3.1 - O Modelo de Dados Entidade-Relacionamento Estendido

O Modelo de Dados Entidade-Relacionamento Estendido (**ERE**), adaptado para este trabalho, é muito parecido com o apresentado em [Ceri85], como por exemplo, no uso de auto-relacionamento e generalização hierárquica. O modelo ERE, facilita o mapeamento para a linguagem de definição de esquemas (Seção 3.2) e é mais atraente que o modelo Entidade-Relacionamento Envolvimento [Wald90], projetado com a finalidade de facilitar a geração de frases simples para explanar ambigüidades em consultas realizadas a BD's. O termo **classe** será utilizado para referenciar um conjunto de entidades ou relacionamentos, e o termo **envolvimento** para expressar algum tipo de associação direta entre um par de classes.

Alguns fatores influenciaram na escolha de um modelo de dados no estilo do modelo Entidade-Relacionamento:

- É um modelo simples e de fácil adaptação pelo usuário,
- Oferece independência de dados,
- Mapeia eficientemente a semântica do mundo real,
- É um modelo amplamente utilizado,
- Facilita o mapeamento para a linguagem de definição de esquemas, o que é primordial para este trabalho, pois é através deste mapeamento, que será definida a clareza na explanação de ambigüidades detectadas.

O modelo ERE não suporta o relacionamento ternário, pois, este dificulta a formulação de frases simples, porém, qualquer relacionamento ternário pode ser dismembrado em relacionamentos binários.

As Figuras (3.1a e 3.1b) apresentam os conceitos do modelo ERE e sua representação:

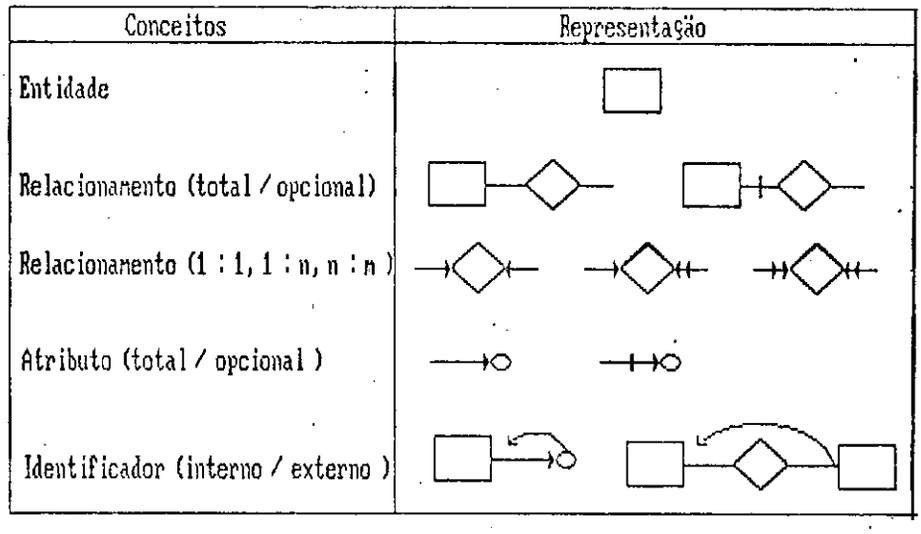


Fig. 3.1a - Representação gráfica do modelo ERE

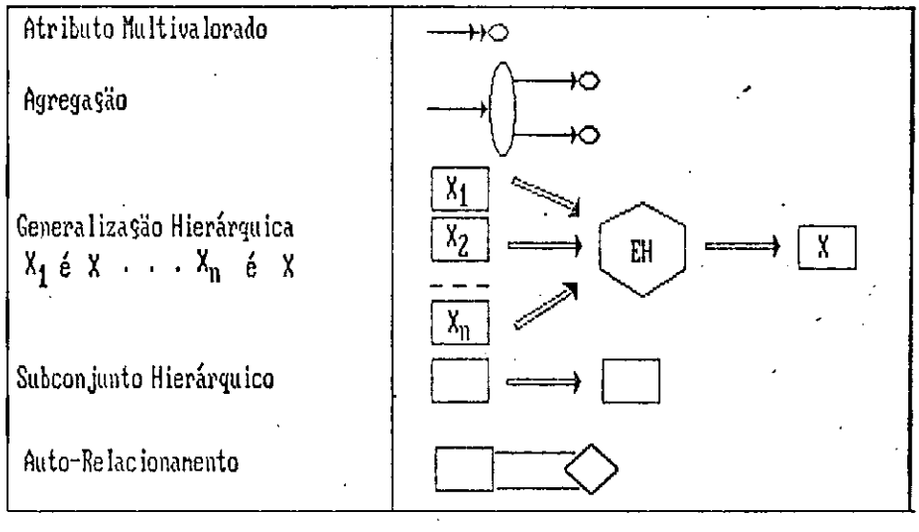


Fig. 3.1b - Representação gráfica do modelo ERE (cont.)

No Apêndice A, é apresentado um esquema de um BD de um Controle Acadêmico, no qual são utilizados quase todos os conceitos expostos nas Figuras (3.1a e 3.1b) e nas próximas seções, serão apresentados vários exemplos de esquemas de BD's que utilizam os conceitos do modelo ERE adaptado para este trabalho.

Todos os esquemas de BD's apresentados a partir desta seção, estarão baseados no modelo ERE.

3.2 - Linguagem de Definição de Esquemas

Um esquema na Linguagem de Definição de Esquemas (**LDE**), criada para este trabalho, consiste de um conjunto de definições de cláusulas. Em cada tipo de cláusula, todas as entidades, seus atributos e relacionamentos são definidos.

Em LDE, ao nome de um relacionamento, é geralmente adicionado um prefixo ou sufixo (preposições), que facilite a formação de frase simples na explanação da ambigüidade. O ABD é uma peça muito importante no sistema, pois, quanto melhor elé descrever o esquema do BD, através da LDE, mais expressivas serão as frases simples geradas na explanação. A LDE permite ao ABD se expressar utilizando-se de preposições, para captar toda a riqueza semântica do modelo de dados. A preposição é separada do verbo através do caráter '_' (sublinha), que posteriormente será retirado, para a formulação das frases simples.

O ABD tem total liberdade para utilizar preposições ou verbos que o auxiliem, propiciando, assim, um maior poder de expressão, como por exemplo: ensinada_pelo, coordenada_por, matriculada_no, etc. A LDE aceita também os quantificadores existenciais (**todo, toda** e **existe**).

A gramática da LDE segue a especificação formalizada de Backus e Naur (BNF) [Aho88]. Cada esquema consiste de um símbolo terminal e de uma expressão em BNF, separados por um sinal de igualdade. O símbolo terminal é um 'metasímbolo' (constante sintática representada por uma palavra). A este formalismo, acrescentamos o caráter '*', para identificação de atributo multivalorado.

Uma expressão em BNF consiste de zero ou mais símbolos terminais, não-terminais e outros metasímbolos sumarizados na especificação a seguir:

Metasímbolo	Significado
< X >	Metaidentificador
|	Alternativamente
[X]	0 ou 1 instância de X
{ X }	0 ou mais instâncias de X
*	Identificação de atributo multivalorado

A seguir é apresentada a gramática da LDE:

Definição do Esquema do BD = { Definição-de-Tipos }

Definição-de-Tipo = **ENTIDADE** Nome-entidade
[**EH** Nome-entidade]
ATRIBUTOS
{<Parte-atributos>}
[**RELACIONAMENTOS**
{<Parte-Relacionamento>}
[**ATRIBUTOS** { <Parte-atributos>}
CHAVE (Nome-atributo, Nome-atributo **)**]]

< Parte-atributo> = Nome-atributo [ÚNICO] < Tipo-básico>

<Parte-relacionamento> = [**OPCIONALMENTE**] [<Quantificador>]
(Nome-Relacionamento) Verbo[{_<Preposição>}]
<Cardinalidade>

<Tipo-básico> = **CADEIA** | **NUMERICO** | **LOGICO** | **AGREGADO**
{ Parte-Atributos } | *Nome-atributo

<Quantificador> = **Todo** | **Toda** | **Existe**

<Preposição> = em | na | pelo | ... | etc.

<Cardinalidade> = 1 | N

Obs: 1 - Palavras reservadas estão em negrito.
2 - A cardinalidade mencionada é da entidade origem para a entidade destino.

Exemplo 1: Considere o seguinte esquema de um BD de um Almoxarifado, Figura 3.2.1, seguido de sua descrição:

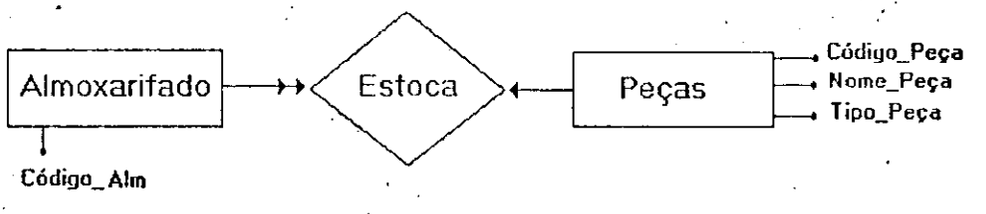


Fig. 3.2.1 - Esquema de um BD de um Almoxarifado

Descrição em LDE do esquema apresentado na Figura 3.2.1:

```
ENTIDADE Almoxarifado
    ATRIBUTOS
        Codigo-alm UNICO NUMERICO
    RELACIONAMENTOS
        ( Estoca ) estoca ( Peças ) N

ENTIDADE Peça
    ATRIBUTOS
        Codigo-peça UNICO NUMERICO
        Nome-peça   CADEIA
        Tipo-peça   CADEIA
    RELACIONAMENTOS
        ( Estoca ) estocadas_no ( Almoxarifado )  N
```

Exemplo 2: Considere o seguinte esquema de um BD de um departamento, Figura 3.2.2:

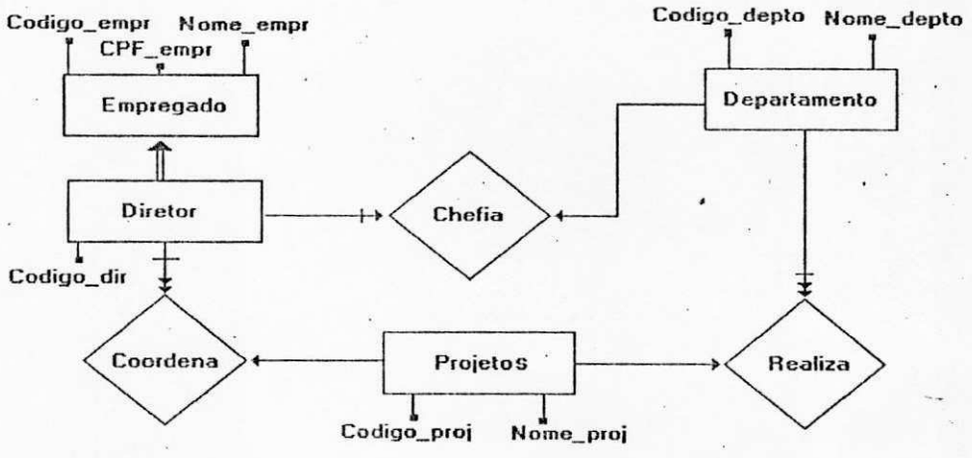


Fig. 3.2.2 - Esquema de um BD de Departamento

Descrição em LDE do esquema de BD apresentado na Figura 3.2.2:

```
ENTIDADE EMPREGADO
    ATRIBUTOS
        Codigo-empr UNICO NUMERICO
        CPF-empr    UNICO CADEIA
        Nome-empr   CADEIA
```

```
ENTIDADE Diretor  EH Empregado
    ATRIBUTOS
        Codigo-dir UNICO NUMERICO
    RELACIONAMENTOS
        OPCIONALMENTE ( Chefia ) chefia ( Departamento ) 1
        OPCIONALMENTE ( Coordena ) coordena ( Projetos ) N
ENTIDADE  Projetos
    ATRIBUTOS
        Codigo-proj UNICO NUMERICO Nome-proj   CADEIA
    RELACIONAMENTOS
        ( Coordena ) coordenado_por ( Diretor ) 1
        ( Realiza ) realizado_pelo  ( Departamento ) 1

ENTIDADE  Departamento
    ATRIBUTOS
        Codigo-depto UNICO NUMERICO
        Nome-depto   CADEIA
    RELACIONAMENTOS
        OPCIONALMENTE ( Realiza ) realiza ( Projetos ) N
        OPCIONALMENTE ( Chefia )  chefiado_por ( Diretor ) 1
```

Um exemplo mais abrangente é apresentado no Apêndice A, com a descrição completa em LDE do BD de um Controle Acadêmico, que utiliza a maioria dos conceitos do modelo ERE e explora bastante, as potencialidades da LDE.

Apesar da LDE poder expressar agregação, este conceito não foi considerado na geração das frases simples, pois, o sistema é um protótipo e sendo assim, foi implementado apenas os conceitos principais do modelo ERE.

3.3 - Linguagem de Consulta do DETECT

A linguagem de consulta do DETECT (**LC-DETECT**) é a mesma utilizada na interface do sistema SQL-Fácil [Bezerra89], com algumas adaptações. A filosofia é a mesma das linguagens de consulta dos sistemas que utilizam uma interface de RU. A descrição da LC-DETECT, segue a especificação formalizada de Backus e Naus (BNF), como apresentada anteriormente (Seção 3.2).

A LC-DETECT é dividida nas três partes apresentadas a seguir:

LISTE < Atributos>
ONDE < Qualificação>
[**USANDO** <Caminho-lógico>]

Onde:

<Atributos> - é o conjunto de atributos da RU que retornarão o resultado da consulta do usuário. Os nomes dos atributos são separados por um ou mais espaços em branco.

<Qualificação> - é a condição lógica realizada sobre os atributos da RU

<Caminho-lógico> - é uma cláusula opcional, utilizada por usuários especializados ou não-eventuais.

É importante salientar, que a cláusula (USANDO <Caminho-lógico>) utilizada em SQL-Fácil, possui um sentido sutilmente diferente do que esta cláusula representa no sistema DETECT. Enquanto que em SQL-Fácil, a cláusula serve para evitar *ambigüidade por todos os tipos de usuários*, no DETECT, ela é utilizada apenas por usuários especializados ou não-eventuais, para consultas rotineiras, evitando que o sistema realize os procedimentos necessários para detectar ambigüidade, já que anteriormente, esta consulta já pode ter sido explanada pelo DETECT e armazenada em disco (Seção 4.2.5.2 Modo Usuário). A cláusula (USANDO <Caminho-lógico>), tem bastante utilidade para o ABD, pois, ele conhece o esquema do BD por inteiro, não necessitando, portanto, que o sistema explane ambigüidade (já que ele conhece todos os caminhos lógicos para realizar suas consultas).

A seguir será apresentada a gramática da LC-DETECT:

LISTE < Lista-de-atributos>
ONDE < Qualificação>
USANDO <Caminho-lógico>

<Lista-de-atributos> = Atributo <Mais-atributo>
< Mais-atributo> = Atributo < Mais-atributo> | &
<Qualificação> = <Expr-booleana>
<Expr-booleana> = <Expr-E> **OU** <Expr-booleana> | <Expr-E>
<Expr-E> = <Expr-relacional> **E** < Expr-E> | < Expr-relacional>
<Expr-relacional> = <Expressão> <Oper-relacional> <Expressão>

<Oper-relacional> = <
|>
|<=
|>=
|=
|<>
<Expressão> = <Termo> <Oper-adição> <Expressão> | <Termo>
< Oper-adição> = + | -
<Termo> = <Fator> <Oper-multi> <Termo> | <Fator>
< Oper-multi> = * | /
<Fator> = atributo | <Constante> | '(' <Expr-booleana>' ')'
< Constante> = Número | Cadeia
< Caminho-lógico> = Relacionamento < Caminho-lógico> | &

Onde:

&: significa vazio,

Atributo: é o nome do atributo definido pelo ABD,

Número: é um valor numérico (Inteiro ou Real),

Cadeia: é uma cadeia de caracteres entre aspas,

Relacionamento : é o nome do relacionamento definido pelo ABD para expressar o caminho lógico.

Obs.: As palavras reservadas da linguagem estão destacadas em negrito.

A seguir serão apresentados exemplos de consultas que podem ser realizadas através da LC-DETECT no BD de Controle Acadêmico (Apêndice A):

Exemplo 1: Liste os nome, código e sala, das disciplinas oferecidas pelo curso de Computação e ensinadas pelo professor José João.

LC-DETECT: **LISTE** Nome-disc Codigo-disc Sala-disc
ONDE (Nome-curso = 'Computação') **E**
(Nome-prof = 'José João')

Exemplo 2: Liste os nomes dos candidatos que desejam bolsa do tipo PICD e são indicados pelo Delegado 003.

LC-DETECT: **LISTE** Nome-estud
ONDE (Tipo-bolsa = 'PICD') **E** (Indica-cand = 003)
USANDO Indica

Comentários: A última consulta possui algumas peculiaridades:

1ª: O usuário deseja os nomes dos candidatos, mas como todo candidato é estudante, o atributo mencionado na consulta para se obter o nome do candidato é 'Nome-estud' ou seja, o nome do estudante, pois candidato herda todos os atributos de estudante.

2ª: Esta consulta possivelmente seria realizada por um usuário especializado, pois, foi utilizada a cláusula '*USANDO Indica*', na qual, 'Indica', é o caminho lógico a ser seguido pelo sistema. Caso isto não fosse feito, o sistema detectaria ambigüidade e perguntaria ao usuário se ele desejaria seguir pelo caminho lógico, no qual o delegado decide bolsa desejada pelo candidato ou aquele onde o delegado indica candidato, caso os objetos selecionados para esta consulta propiciassem esta ambigüidade.

Vários exemplos de consultas válidas, utilizando a LC-DETECT, serão apresentadas ao longo deste trabalho. Na Seção 4.2 (Descrição da Interface com o Usuário) será mostrado como o usuário pode usufruir das facilidades que a interface oferece para que ele possa realizar suas consultas. Como por exemplo, utilizando-se do menu de entidades e atributos, no qual o usuário não precisa saber previamente qual a grafia dos nomes dos atributos definidos pelo ABD, eliminando, portanto, erros léxicos no processamento de suas consultas.

Na linguagem de consulta do DETECT não se utilizou as funções estatísticas, como por exemplo Média(), Soma(), etc, pelo fato desta linguagem se tratar de um protótipo.

No próximo capítulo será apresentada a arquitetura do DETECT, na qual serão descritos os módulos principais do sistema, como por exemplo os analisadores sintáticos que mapeiam os esquemas descritos em LDE, para a estrutura de dados na memória principal e o analisador sintático que identifica os objetos definidos pelo ABD para que o usuário possa realizar suas consultas, assim como a interface com o usuário.

Capítulo IV

4 - Arquitetura do Sistema

Neste capítulo, será apresentada a arquitetura do sistema juntamente com a funcionalidade de cada módulo componente da mesma. Como complemento deste capítulo, será apresentada uma descrição da interface do DETECT com o usuário, seguida de uma simulação de utilização do sistema.

4.1 - Módulos do Sistema

O sistema DETECT está dividido logicamente em 5 (cinco) módulos: *Inicialização*, *Obtenção da Consulta*, *Identificação de Conexões*, *Algoritmo Cruzamento* e *Explanação*. Existe um sexto módulo conectado à arquitetura do sistema, que é intitulado '*Consulta ao BD*', porém, este não faz parte do DETECT, pois já foi implementado pelo sistema SQL-Fácil [Bezerra89], o qual está descrito no Capítulo 1 (Tradutor).

Os módulos da Figura 4 que estão sombreados fazem parte do DETECT. Em seguida , será apresentada uma breve descrição de cada módulo componente do sistema.

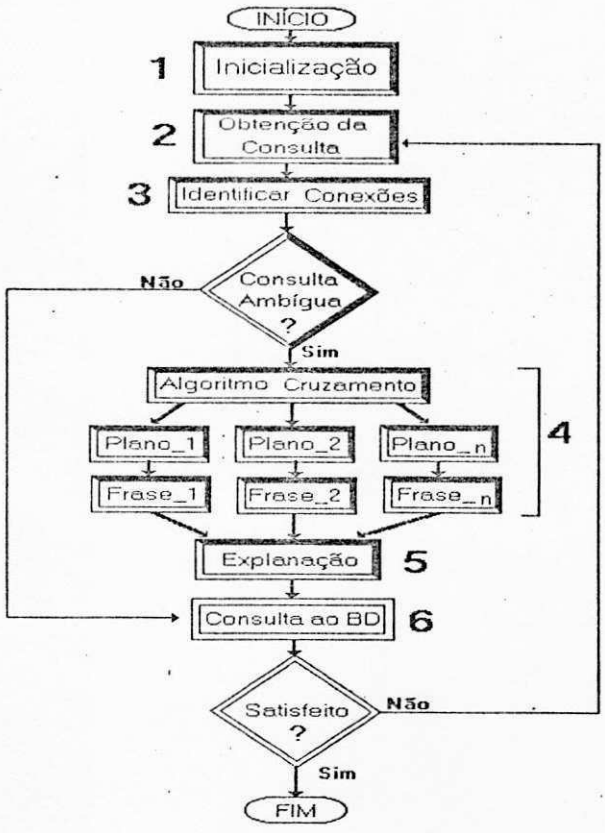


Fig. 4 - Arquitetura do Sistema

4.1.1 - Inicialização

O Módulo de Inicialização do Sistema é sub-dividido em 4 (quatro) partes: *Modo Administrador*, *Carregar BD*, *Carregar Objetos* e *Incializar Estruturas de Dados*. A Figura 4.1, apresenta as subdivisões do Módulo Inicialização:

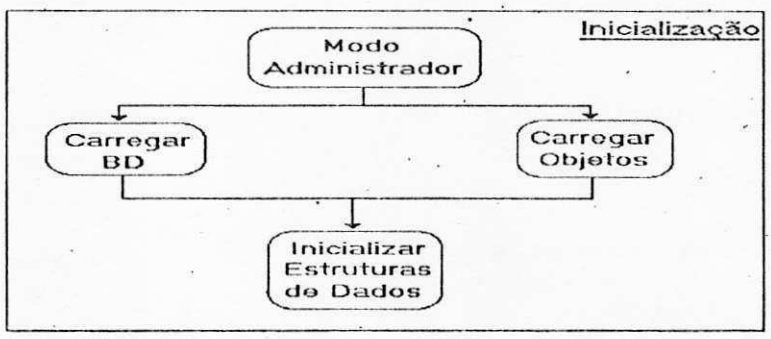


Fig. 4.1 - Módulo de Inicialização

A seguir, será apresentada a descrição funcional de cada sub-módulo do módulo Inicialização.

4.1.1.1 - Modo Administrador

Neste ambiente, o usuário administrador do BD, ABD, tem permissão para executar algumas tarefas especiais, como por exemplo: *Carregar o arquivo de objetos* ou o *arquivo contendo o esquema do banco de dados.* O Modo Administrador está descrito mais detalhadamente na seção 4.2.5.3.

4.1.1.2 - Carregar BD

A opção Carregar BD, está contida no menu do Modo Administrador, apresentada na Seção 4.2.5.3 (Menu Modo Usuário) com o título de '*Banco de Dados*'.

Carregar BD, significa coloçar à disposição do usuário um determinado esquema de BD, para que ele possa realizar suas consultas. Antes do BD estar à disposição do usuário, o sistema realiza uma análise *léxica, sintática* e *semântica*, apresentadas a seguir, no arquivo que descreve o esquema do BD em questão. A descrição da definição do esquema do BD deve estar de acordo com a especificação definida na Seção 3.2 (Linguagem de Definição de Esquemas) caso isso não ocorra, o sistema acusará um erro na descrição do esquema do BD e não o carregará.

Análise Léxica, Sintática e Semântica

O *Analisador léxico*, se encarrega de verificar se a descrição do esquema do BD está de acordo com a estrutura léxica permitida. Por exemplo, verifica se todos os caracteres especiais utilizados na descrição são permitidos pela linguagem, assim como, se os identificadores e números, estão conforme o formato especificado, etc.

O *Analisador Sintático*, utiliza o analisador léxico toda vez que necessite de um símbolo da cadeia de entrada e tem a finalidade de verificar se as frases descritas no programa-fonte, estão nos termos das construções permitidas pela linguagem. O tipo de análise sintática utilizada foi a '*Descendente Recursiva*' [Aho88], pois este tipo de análise é

eficiente e indicada para linguagens que possuem gramáticas pequenas (com poucas regras), que é o caso da LDE, LC-DETECT e a linguagem de Definição de Objetos.

O Analisador Semântico, verifica se as sentenças descritas possuem um significado semântico correto, ou seja, um significado associado a cada construção da linguagem, as quais serão utilizadas na fase de Síntese do compilador.

Os analisadores léxico, sintático e semântico, seguem as especificações definidas em [Aho88].

4.1.1.3 - Carregar Objetos

Esta opção, assim como a opção Carregar BD, está contida no menu do Modo Administrador (Seção 4.2.5.3) com o título de '*Objetos*'. Os objetos contidos no arquivo de objetos, servirão para que se possa identificar, junto aos atributos mencionados na consulta do usuário, se a mesma é ambígua. A Seção 5.2 (Arquivo de Objetos) descreve mais algumas funções dos objetos. Os analisadores léxico e sintático para o arquivo de objetos, funcionam de acordo com o procedimento apresentado na Seção 4.1.1.2, com a diferença de que o analisador sintático segue a gramática dos Objetos, definida na Seção 5.2.

4.1.1.4 - Inicializar Estruturas de Dados

As estruturas de dados são inicializadas à medida em que está sendo compilado o arquivo de objetos ou o arquivo do esquema do BD. As estruturas de dados preenchidas estão expostas detalhadamente no Capítulo 5 (Estrutura de Dados).

4.1.2 - Obtenção da Consulta

A obtenção da consulta é realizada no módulo usuário ou administrador, através do editor de consultas.

A Seção 3.3, apresenta a linguagem de consulta do DETECT, LC-DETECT, na qual é mostrada a sintaxe das consultas aceitas pelo sistema, e nas seções 4.2.5.2 e 4.2.5.3, são apresentadas a interface do editor de consultas para o *modo usuário* e *modo administrador*, respectivamente.

Os analisadores léxicos e sintáticos da LC-DETECT, seguem o mesmo comportamento dos analisadores apresentados anteriormente, com a diferença de atuarem de acordo com a gramática da LC- DETECT.

4.1.3 - Identificar Conexões

A identificação de conexões é um procedimento efetuado pelo sistema, para identificar os atributos mencionados na consulta do usuário. A conexão, servirá para verificar se a consulta é ambígua ou não. A Seção 2.4.1 (Conexão) descreve sucintamente o mecanismo utilizado pelo sistema para analisar as conexões geradas pela consulta do usuário.

4.1.4 - Algoritmo Cruzamento

A Algoritmo Cruzamento é o responsável pela geração dos *planos*, que são os caminhos lógicos seguidos pelo sistema devido à detecção de ambigüidade. Estes planos serão transformados em frases simples no momento da explanação da ambigüidade para o usuário.

A Seção 6.4 (Planejando Cruzamento) apresenta de forma mais detalhada o funcionamento deste algoritmo.

4.1.5 - Explanação

O módulo Esplanação é o responsável pela apresentação das frases simples geradas pelo algoritmo cruzamento, Seção 6.4. A explanação de consultas ambíguas é descrita em detalhes na Seção 6.2 (Explanação de Consultas Ambíguas).

4.2 - Descrição da Interface com o Usuário

Introdução

Tornar cada vez mais fácil a interação entre o usuário e o computador, tem sido uma das principais preocupações dos diversos produtos na área de infomática lançados ultimamente. Cresce o número de usuários mais esclarecidos, com maior poder de contestação. Este fato, junto à grande concorrência entre os produtos, faz com que a preocupação na produção de software volte-se para o real objetivo, que é o de satisfazer plenamente o usuário final e não apenas a obtenção de lucro financeiro.

Na descrição a seguir, constam informações sobre o projeto de interface do Sistema Detector de Ambigüidade em Consultas Conceituais a Banco de Dados (**DETECT**).

Descrição Funcional da Aplicação

Para dar maior clareza sobre as escolhas dos estilos de interfaces escolhidas, será esclarecido, de forma breve qual a finalidade e comportamento do sistema.

A ferramenta é um sistema detector de ambigüidade em consultas formuladas a um BD.

O sistema é subdividido em duas partes, as quais são: **Modo Usuário** e **Modo Administrador.**

No **Modo Usuário**, será permitido ao mesmo, realizar consultas ao BD. Caso esta consulta seja ambígua (mais de uma interpretação possível) o usuário terá que escolher entre as várias alternativas, apresentadas em forma de menu, a que reflita melhor sua real intenção de consulta. Este poderá também recuperar o texto de consultas previamente armazenadas, assim como, armazenar novas consultas (não o resultado).

No **Modo Administrador**, será permitido ao mesmo, realizar todas as funções permitidas no Modo Usuário e algumas funções especiais, como por exemplo: alterar uma descrição do BD, inicializar um BD para consultas, definir visões para os usuários, etc.

Características dos Usuários

Um usuário ocasional desta aplicação, não precisa ser necessariamente especializado na área para poder utilizar o sistema, bastando que obtenha um conhecimento prévio da finalidade do sistema e do seu comportamento geral. Para suportar esta naturalidade de interação com a aplicação, foram utilizadas técnicas de interface, as mais variadas, como por exemplo:

- Ajuda 'ON-LINE', na qual o usuário obtém informações imediatas sobre o que ele pode fazer a partir do contexto atual, no qual foi solicitada a ajuda,
- Utilização de menus e de teclas de funções,
- Dois níveis de acesso para atender à usuários com diferentes níveis de conhecimento, etc.

4.2.1 - Descrição da Interface

Inicialização do Sistema

Para dar início ao trabalho no sistema, devem ser realizados os seguintes procedimentos: Ativar o sistema através da linha de comando de um sistema operacional, como por exemplo no 'DOS' teríamos: 'C> DETECT <CR>', onde 'C>' indica a unidade de disco atual; 'DETECT' é o nome do sistema a ser ativado e '<CR>' é a indicação do acionamento da tecla 'RETURN' ou 'ENTER', pelo usuário. A partir daí, o usuário teria à sua disposição o sistema ativado, para que possa realizar suas consultas no usuário (Modo Usuário) ou administrador do BD (Modo Administrador).

O mecanismo de navegação através do sistema funciona por meio das teclas de funções (F1 ... Fn), que são apresentadas no rodapé de cada tela. A partir da tela de abertura, que expõe uma breve explicação da aplicabilidade do sistema, o usuário poderá dar início ao seu trabalho. Através desta tela inicial o usuário deve identificar-se como sendo usuário convencional (tecla F2) ou usuário administrador (tecla F3). A tela inicial do sistema será apresentada em um tópico mais adiante (Simulando uma Utilização do Sistema), que expõe as possíveis operações realizáveis no sistema.

Estilo de Diálogo

O estilo de diálogo escolhido foi o de seleção por menus, auxiliados pelas teclas de funções. Esta escolha foi, a princípio, a que mais se identificou com a aplicação e as características dos usuários. Foi levado em consideração na escolha da utilização de menu, o fato de que o número de opções de cada menu não é muito grande, facilitando assim, a localização por parte dos usuários. A aplicação não suporta o recurso de "Mouse" devido a simplicidade da mesma. Os ítens do menu podem ser acionados pela letra em destaque contida em cada opção do menu, o que facilita uma busca lógica da opção desejada. Esta facilidade não é oferecida para todos os menus, apenas quando possível.

Caso o usuário deseje alguma informação sobre o menu atualmente exposto, este deve recorrer à opção de ajuda contida no próprio menu, que exporá descritivamente a função de cada opção exposta.

A navegação através no menus é feita com o uso das teclas de direcionamento do teclado (PgUp, PgDn, Home, End e das setas de direção) com as quais o usuário poderá locomover-se através do menu.

Para que o usuário possa saber em qual opção o cursor está localizado, foi utilizado o recurso de vídeo reverso, que coloca em destaque a opção atual de escolha e, a efetivação da mesma é feita através da tecla "ENTER" ou "RETURN". Através deste mecanismo, o usuário não poderá efetuar uma escolha que não esteja à sua disposição, evitando assim, o tratamento de erros para este nível, embora isto não elimine a possibilidade do usuário escolher uma opção errada através do menu ou escolher uma letra que não esteja em destaque no menu.

4.2.2 - Apresentação da Informação

O "Layout" de telas é dividido em cinco partes bem definidas com posições pré-fixadas, as quais são: Área de consulta, Área de mensagens de erro ou advertência, Área de entrada e saída, Área de menus, Área de teclas de funções e Área de explanação.

A escolha de áreas específicas de trabalho deu-se pelo fato desta estratégia facilitar a familiarização mais rápida com o sistema por parte do usuário, pois, este tem uma prévia

noção do que está acontecendo no vídeo. As áreas de mensagens de erro, entrada e saída e de menus, compartilham uma única área, que se localiza abaixo da área de consulta. A

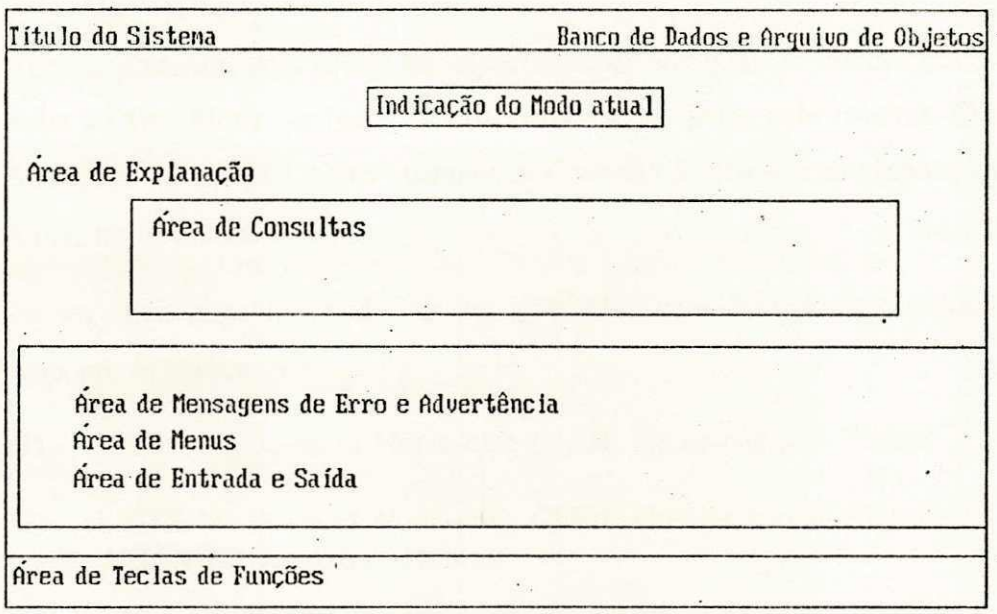


Fig. 4.2.1 - Apresentação da Informação

Figura 4.2.1, apresenta o "Layout" das áreas pré-definidas:

As áreas pré-definidas apresentam informações específicas como apresentadas abaixo:

Área de Consulta

Nesta área, o usuário poderá realizar suas consultas ao BD. Esta área é gerenciada por um módulo editor de texto, que permite ao usuário inserir novas palavras em uma frase já formulada, eliminar um palavra da frase e algumas funções básicas de um editor de texto.

No módulo de consulta, o usuário pode trabalhar em dois níveis de especialização:

(1) - Quando o usuário desconhece a estrutura lógica do armazenamento das informações e

(2) - Consulta realizada por um usuário já familiarizado com o sistema ou um usuário especializado.

Para cada nível há uma forma de consulta:

(1) - **LISTE** Atributos **ONDE** Qualificação

(2) - **LISTE** Atributos **ONDE** Qualificação
USANDO Caminho-lógico.

Vemos, portanto, dois níveis de especialização nos quais o usuário poderá realizar sua consulta ao BD. Onde *Atributos* são os atributos desejados pelo usuário, *Qualificação* são as restrições de seleção sobre atributos e *Caminho-lógico* é o caminho a ser seguido pelo sistema, no esquema do BD.

As seguintes consultas poderíam ser realizadas, considerando-se o esquema do BD apresentado no Apêndice A:

(1) - **LISTE** Nome-estud Nome-disc ONDE Nome-curso = 'Física'

(2) - **LISTE** Nome-estud Nome-disc **ONDE** Nome-curso = 'Física'
USANDO Registra Oferece

A consulta número (2), demonstra que o usuário conhece o esquema do BD, que foi expresso através dos relacionamentos (Registra e Oferece). Evitando assim o problema da ambigüidade.

Área de Mensagens de Erro ou Advertência

Nesta área o usuário terá informações sobre qualquer problema na tarefa que mandou executar. As mensagens de erro são acompanhadas por um breve sinal sonoro, enquanto que mensagens de advertência são apresentadas sem o acompanhamento de sons. As mensagens podem ocorrer por exemplo, pela detecção da existência de um arquivo sobre o qual o usuário mandou gravar uma consulta ou indicou a leitura de um arquivo não existente no disco; podem ocorrer, também, pela detecção de um erro léxico ou sintático no momento da leitura da descrição de um esquema de um BD ou do arquivo de objetos, por ordem do administrador do BD, etc.

Área de Entrada e Saída

É responsável pela obtenção de nomes de arquivos sobre os quais o usuário ou administrador deseje trabalhar. Através desta área, o usário poderá por exemplo, gravar o texto da consulta sendo processada atualmente ou recuperar o texto de uma consulta

anteriormente armazenada; assim como o administrador poderá carregar arquivos contendo as visões (Objetos) que delimitarão o acesso dos usuários ao BD.

Área de Menus

Esta área está reservada para a apresentação de menus que serão expostos ao usuário ou administrador. A disposição dos menus e seu processamento foram mencionados no tópico Estilo de Diálogo da Seção 4.2.1.

Área de Teclas de Funções

É a área reservada para uma espécie de menu, que figurará na maioria das telas do sistema e que é acessado pelas teclas de funções (F1... Fn). Esta área muda de acordo com o contexto, por exemplo, a tecla F1, apesar de ser utilizada todo o tempo para acionar uma tela de ajuda ao usuário, esta apresenta conteúdos diferentes caso seja acionada em contextos diferentes. Por exemplo, caso o usuário pressione a tecla F1 na primeira tela do sistema, este obtém informações sobre as teclas de funções que estão no rodapé da tela corrente, enquanto que, se isto for feito no momento em que o usuário estiver trabalhando na área de consulta, é apresentada uma tela contendo informações sobre o editor de consultas e qual a sintaxe de uma consulta aceita pelo sistema.

Área de Explanação

Nesta área o usuário obtém informações sobre uma ambigüidade detectada no processamento de sua consulta. Esta explanação é apresentada sob a forma de menu, no qual o usuário retifica sua consulta de acordo com as opções expostas pelo sistema ou formula outra consulta. A Figura 4.2.2, demonstra como poderiam ser expostas as frases geradas pelo sistema e no final desta dissertação são apresentadas uma outras propostas de exposição das frases, que seriam expressas através de figuras pré-definidas associadas às entidades do BD ou através da Liguagem Natural.

A Figura 4.2.2, apresenta uma tela de uma explanação:

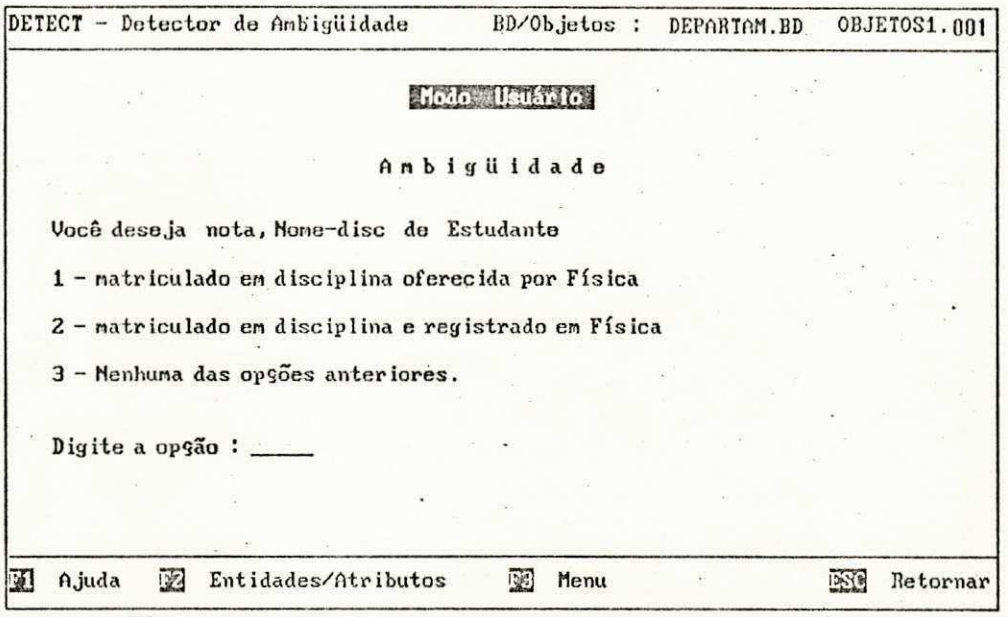


Fig. 4.2.2 - Explanação de uma Consulta Ambígua

4.2.3 - Métodos de Ajuda

O sistema oferece acesso aos seus métodos de ajuda das seguintes formas:

- Através das teclas de funções localizadas no rodapé de cada tela,
- Na apresentação de mensagens de erro ou advertência que foram apresentadas anteriormente,
- Ajuda "on-line", onde são descritas orientações no uso da interface e oferece também, uma breve ajuda "off-line", que será apresentada posteriormente no tópico (Simulando uma utilização do sistema).

4.2.4 - Ambiente de Operação

O sistema trabalha basicamente em um ambiente PC-XT e compatíveis, uma unidade de disco rígido, um monitor de vídeo, preferencialmente colorido, pois o sistema utiliza cores padronizadas, teclado modelo XT/AT e 640K bytes de memória RAM.

Na sua versão atual, o sistema possui aproximadamente 4000 (quatro mil) linhas de código fonte, programadas na linguagem de programação Pascal versão 6.0, porém, não foi utilizado o recurso de Orientação a Objetos.

Seria interessante, em versões futuras, utilizar a memória extendida dos computadores para que se possa carregar esquemas de BD's grandes.

4.2.5 - Simulando uma Utilização do Sistema

Após executado o procedimento visto no tópico Modo de Inicialização do Sistema, a tela que aparecerá no vídeo está exposta na Figura 4.2.3:

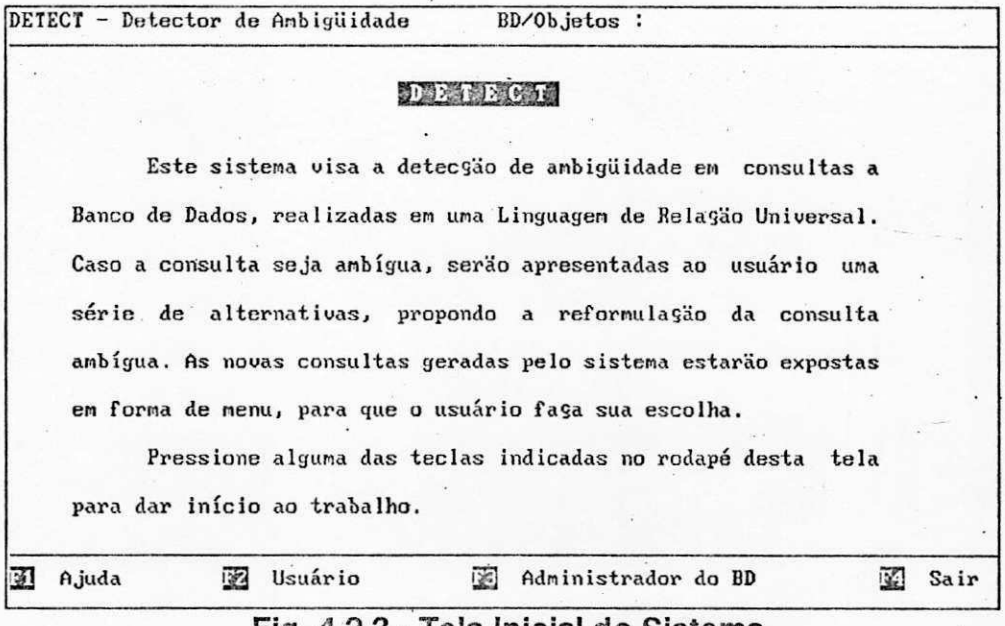


Fig. 4.2.3 - Tela Inicial do Sistema

4.2.5.1 - Ajuda (Tela Inicial)

Caso seja pressionado a tecla F1 neste nível do sistema, será apresentada uma tela de ajuda, a qual contém breves explicações sobre as escolhas possíveis, apresentadas no rodapé da tela inicial do sistema. A tela de ajuda será apresentada conforme a Figura 4.2.4:

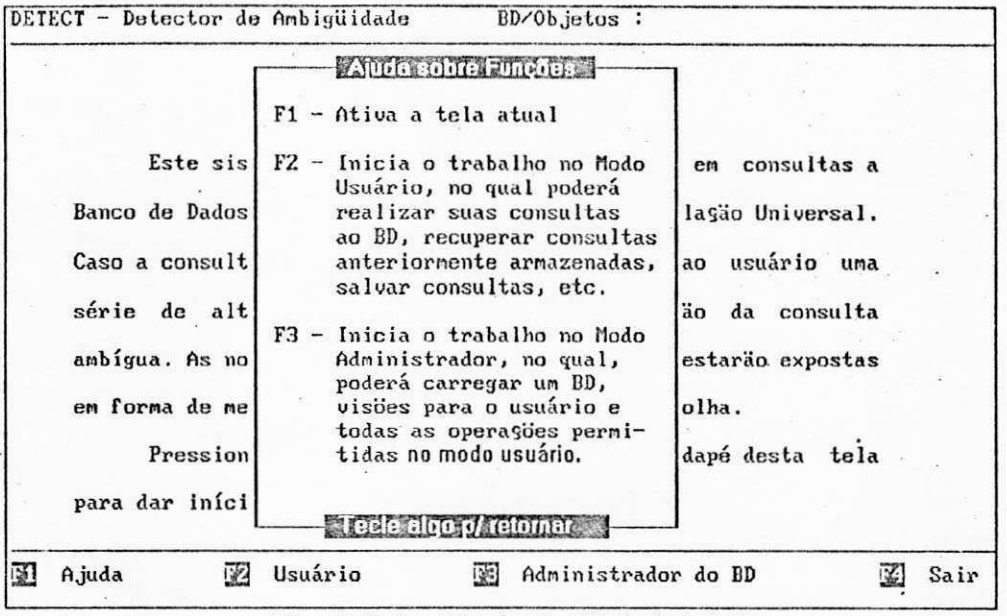


Fig. 4.2.4 - Ajuda da Tela Inicial

4.2.5.2 - Modo Usuário

Neste nível, o usuário terá acesso à sua área de trabalho, na qual poderá realizar uma consulta ao BD e todas as operações a ele permitidas. A princípio, será apresentada uma tela contendo uma área de entrada de consultas, na qual o usuário poderá trabalhar e alterar consultas utilizando-se dos comandos normais de um editor convencional. Por exemplo, poderá trabalhar no modo de inserção para adicionar novos atributos à consulta, usar a tecla 'Backspace' para excluir atributos não mais desejados, etc. São oferecidas algumas funções de apoio como (Ajuda, Entidades/Atributos e Menu) que serão descritas a seguir. A Figura 4.2.5, apresenta a tela principal do Modo Usuário.

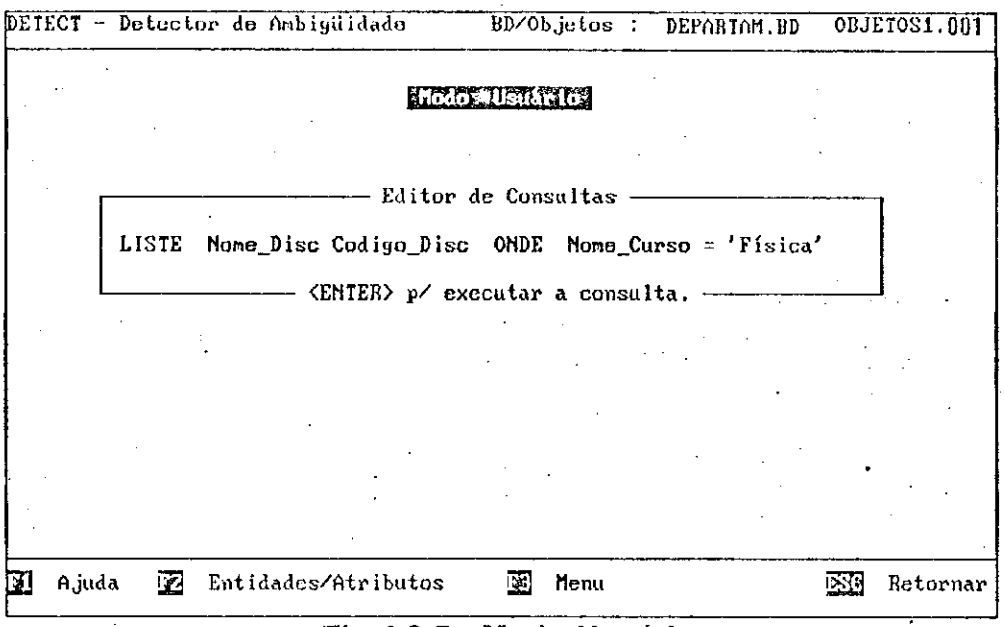


Fig 4.2.5 - Modo Usuário

Ajuda (Modo Usuário)

Através desta opção, o usuário terá acesso a informações sobre o modo de utilização do editor de consultas e, sobre qual a sintaxe em que as consultas devem ser elaboradas. O formato da tela está apresentado na Figura 4.2.6:

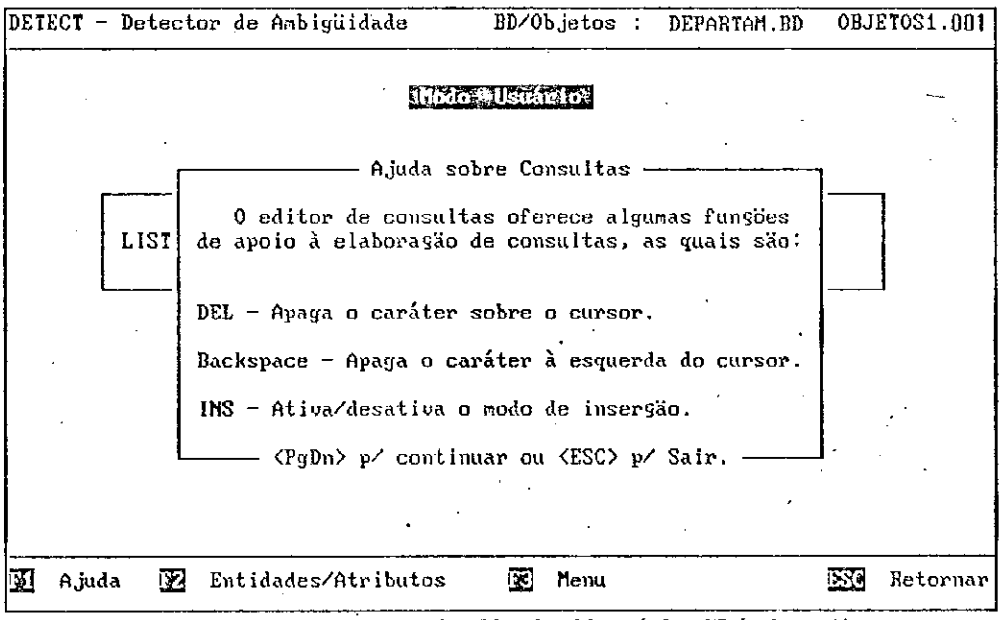


Fig. 4.2.6 - Ajuda do Modo Usuário (Página 1)

Caso o usuário pressione a tecla (PgDn), o sistema mostrará o complemento da tela de ajuda à consultas, como é indicado no rodapé da tela de ajuda na figura anterior. A Figura 4.2.7, apresenta este complemento.

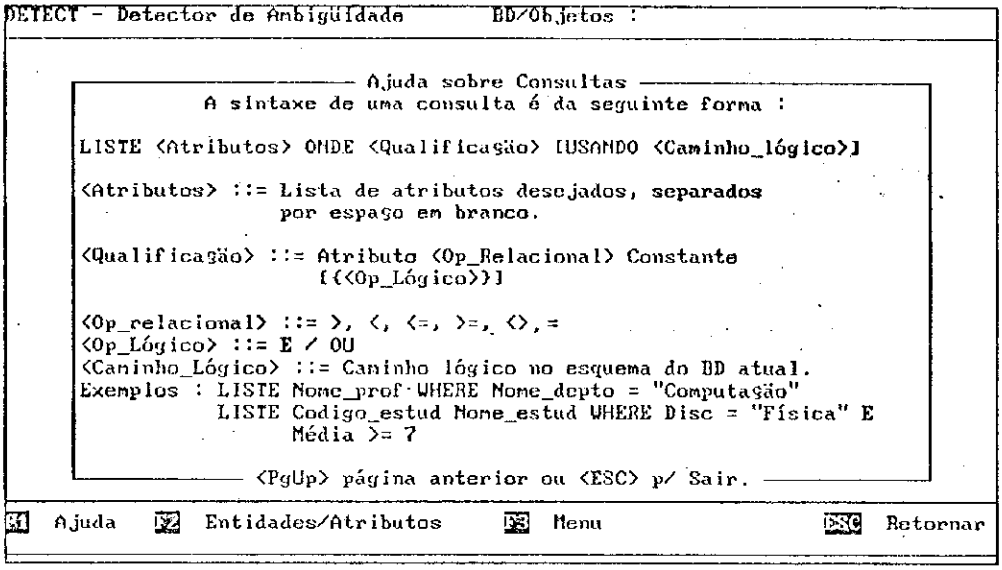


Fig. 4.2.7 - Ajuda do Modo Usuário (Página 2)

Menu (Modo Usuário)

Com esta escolha, o usuário terá à sua disposição um menu contendo funções como: salvar consultas, recuperar consultas, imprimir consultas, ir para o sistema operacional e ajuda sobre o menu em andamento. A tela é apresentada conforme a Figura 4.2.8:

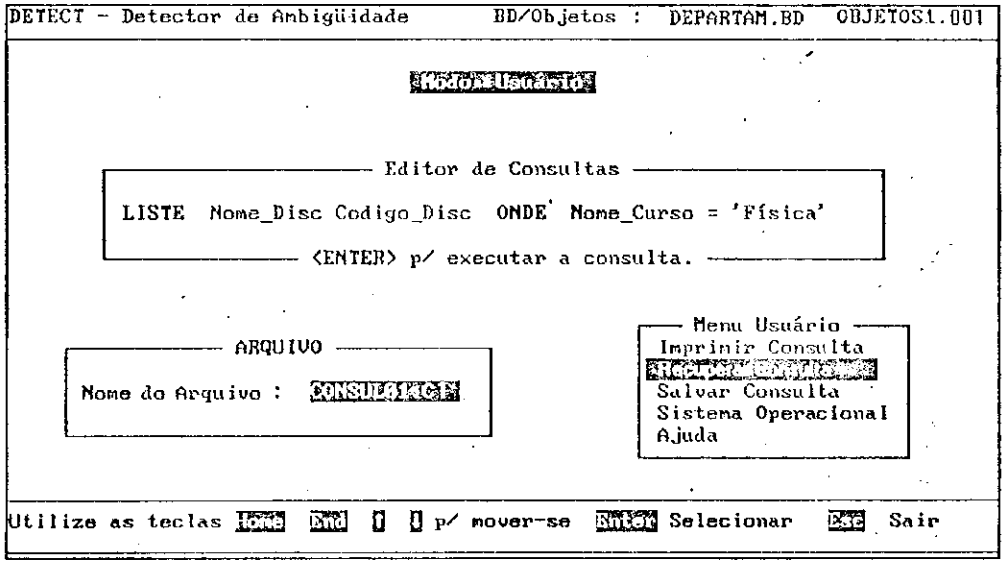


Fig. 4.2.8 - Menu Usuário

Ajuda (Menu do Modo Usuário)

Através desta opção o usuário terá acesso a informações detalhadas sobre as opções que estão presentes no menu. A tela de ajuda é apresentada conforme a Figura 4.2.9:

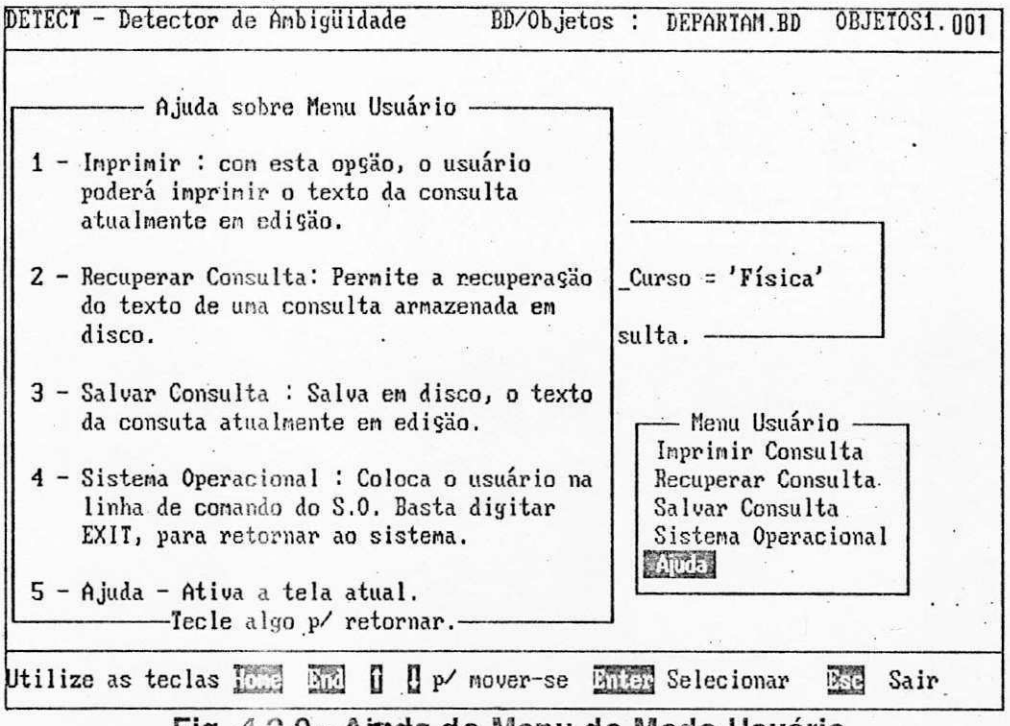


Fig. 4.2.9 - Ajuda do Menu do Modo Usuário

4.2.5.3 - Modo Administrador

Esta opção define a área de trabalho do administrador do sistema. Nesta seção, o administrador detém total controle do sistema, pois este poderá trabalhar com todas as funções permitidas aos usuários e também trabalhar com as funções reservadas aos administradores, como por exemplo: definir as visões dos usuários (objetos) ou carregar um BD. A tela inicial de trabalho do administrador, é basicamente do mesmo tipo da tela apresentada aos usuários, como mostrado na Figura 4.2.10:

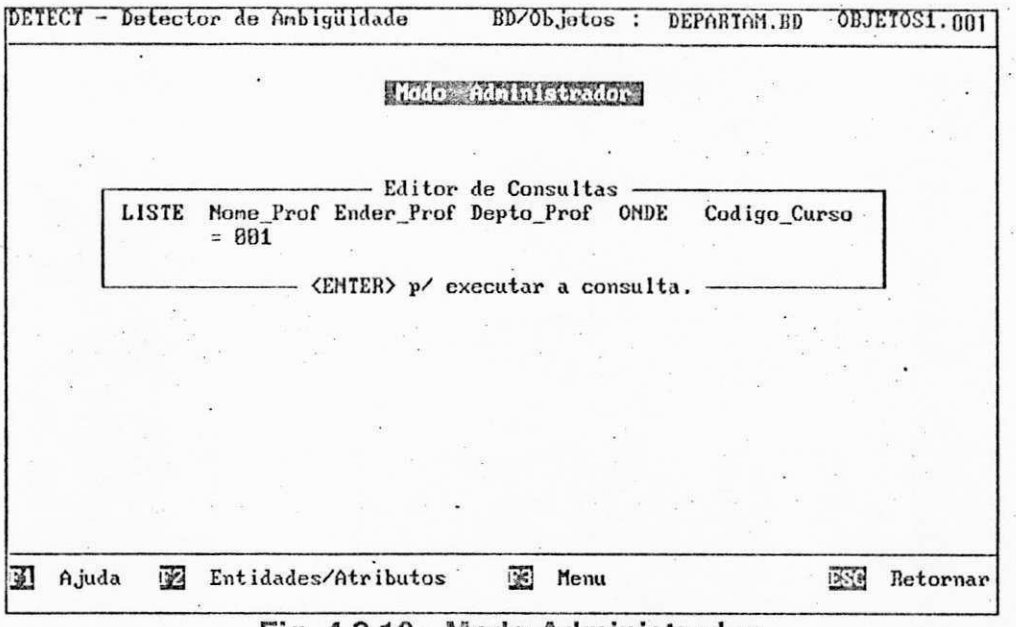


Fig. 4.2.10 - Modo Administrador

Ajuda (Modo Administrador)

No modo administrador, caso seja pedido ajuda através da tecla F1 (Ajuda), a tela apresentada será a mesma descrita no tópico (Menu Ajuda Modo Usuário) Seção 4.2.5.3, que expõe carcterísticas sobre o editor e consultas.

Menu (Modo Administrador)

Com esta opção, o administrador terá à sua disposição, duas funções especiais em relação aos usuários comuns, as quais são: carregar objetos e banco de dados. A opção Objetos, é de extrema responsabilidade do administrador do sistema, pois é ela que permite manter a segurança de acesso ao BD por parte dos usuários. A opção banco de dados, é a que definirá qual BD estará ativo para que o usuário realize suas consultas.

A tela de menu do Modo Administrador é bastante parecida com a tela apresentada na Seção 4.2.5.3 (Menu Modo Usuário), a diferença está na adição das opções *Objetos* e Banco de Dados, o mesmo ocorre com a tela apresentada quando se deseja informações sobre as opções do menu através da opção (Ajuda) no menu do Modo Usuário. A Figura 4.2.11, apresenta o Menu do Modo Administrador:

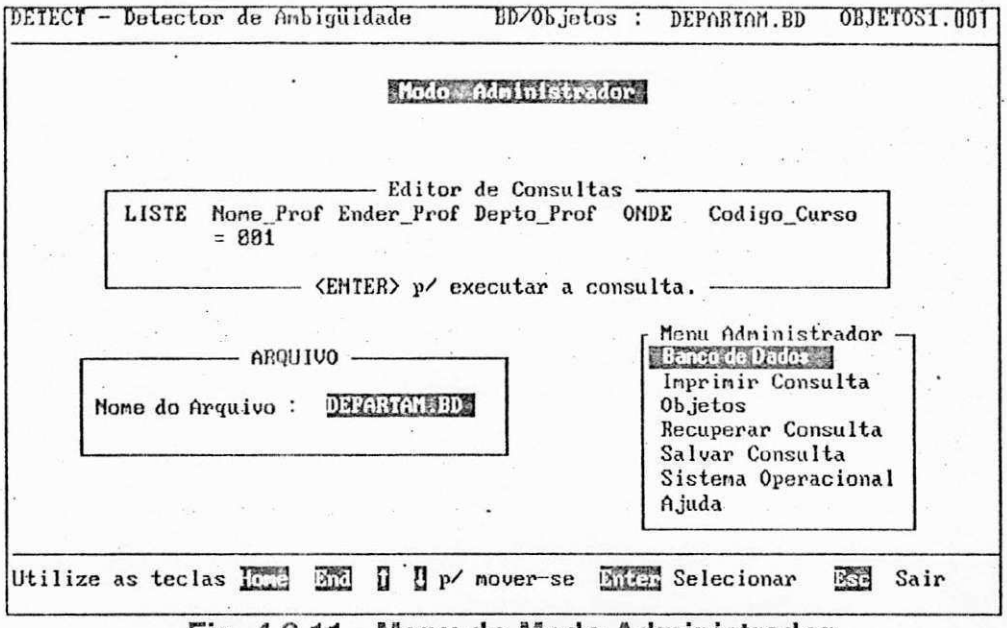


Fig. 4.2.11 - Menu do Modo Administrador

No modo de consulta, tanto no Modo Administrador, quanto no Modo Usuário, existe a opção de ajuda de seleção dos atributos a partir da tecla F2 (Entidades/Atributos). Este recurso facilita a consulta ao BD, principalmente à usuários eventuais do sistema, que não se adaptam à grafia dos nomes dos atributos definidos no Esquema do BD. Através deste menu de Entidades/Atributos, o usuário seleciona os atributos desejados sem a necessidade de digitá-los, bastando apenas que os selecione no Menu de Atributos, para

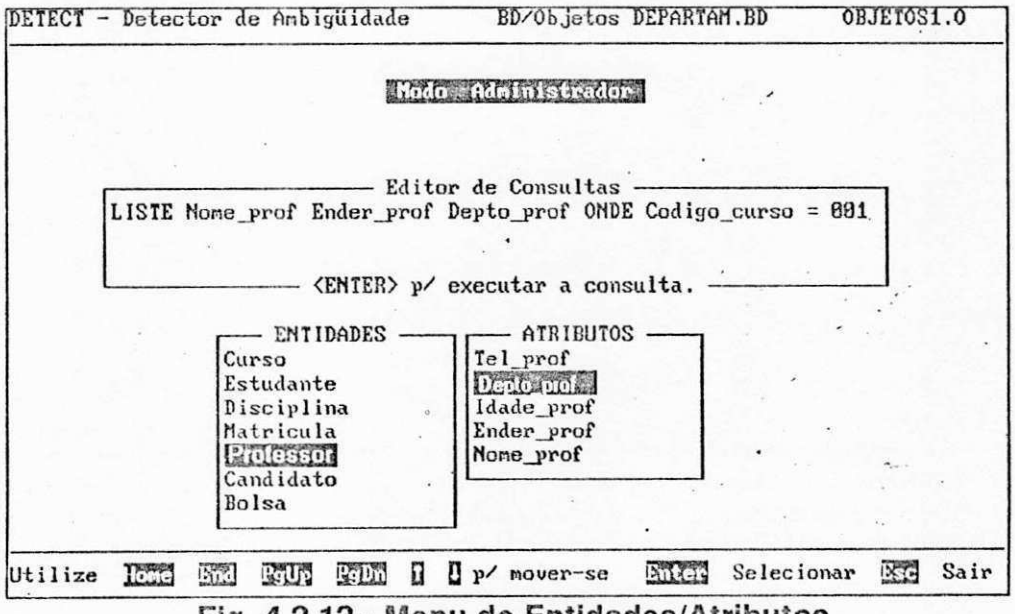


Fig. 4.2.12 - Menu de Entidades/Atributos

que estes sejam adicionados à consulta que está sendo elaborada. A Figura 4.2.12, apresenta um exemplo de utilização do Menu Entidade/Atributos.

No Modo Administrador, quando ocorre um erro de compilação do arquivo do Esquema do BD ou o arquivo de Objetos, o ABD pode utilizar um editor de programas de seu conhecimento para reparar o erro e submetê-lo à nova compilação, porém, para que ele possa consertar os erros sem sair do sistema, pode ser utilizado um editor de programas residente, como por exemplo o 'SideKick'. A Figura 4.2.13, mostra uma utilização do 'SideKick', para reparar um erro de compilação ocorrido no arquivo de Objetos, que no caso do exemplo, é a falta do caráter '{' (abre colchetes) na 1ª linha do arquivo de objeto. Este conserto é feito sem precisar sair do ambiente do DETECT.

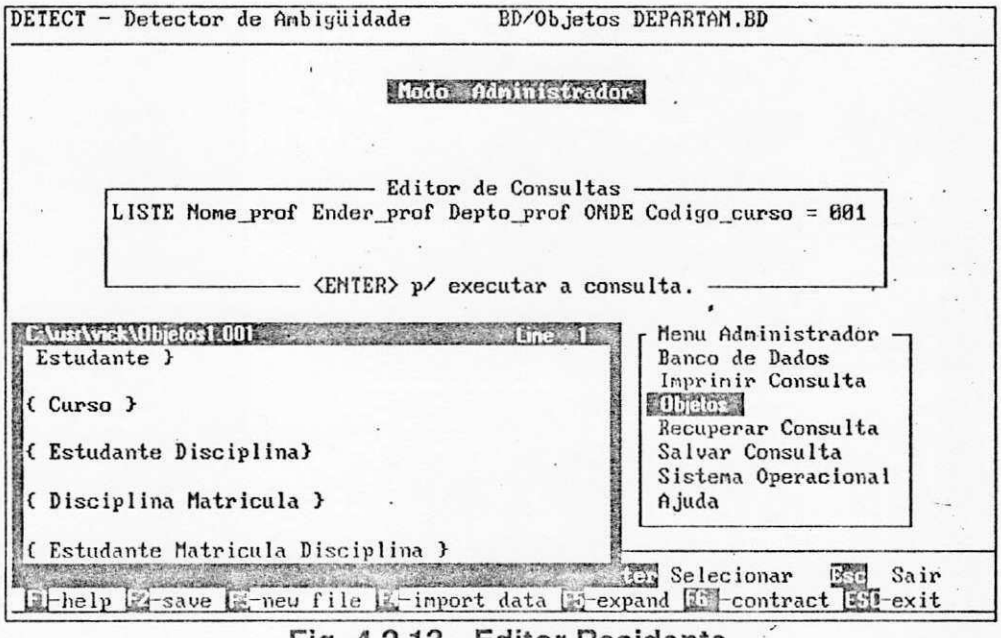


Fig. 4.2.13 - Editor Residente

O ambiente da interface do DETECT, no diz respeito aos menus e solicitação dos nomes de arquivos de trabalho, foi projetado a partir da utilização de um Software de projeto de menus e entrada de dados (MEW e SG4) respectivamente, que fazem parte do projeto da dissertação de mestrado intitulada SISAL (Sistema de Autoria de Lições) [Alves89].

No próximo capítulo, será apresentada a estrutura de dados utilizada para dar suporte ao sistema, incluindo as estruturas originadas a partir do Modelo de Dados ERE e o arquivo de objetos, que é o responsável por armazenar os objetos sobre os quais o usuário pode realizar suas consultas.

Capítulo V

5 - Estrutura de Dados

A estratégia utilizada para a escolha da Estrutura de Dados foi a de seguir uma das máximas da programação: **Quanto mais simples, mais eficiente**. Utilizou-se desta máxima, para implementar as estruturas de dados básicas do Sistema DETECT. A Figura 5.1, mostra os módulos mais importantes da estrutura de dados, seguidos de suas descrições.

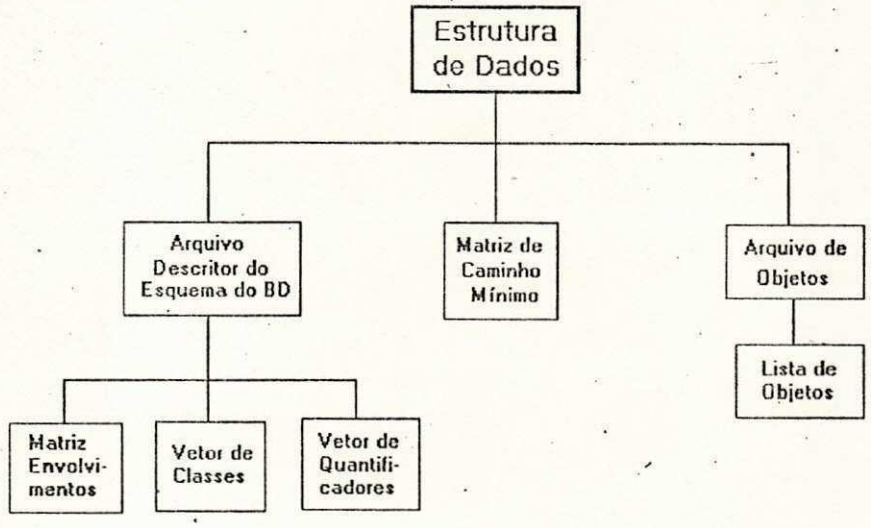


Fig. 5.1 - Módulos da Estrutura de Dados

5.1 - Arquivo Descritor do Esquema do BD

Um Arquivo Descritor, contém toda a descrição semântica do BD. É nele que se encontram as informações necessárias para se inicializar a Matriz de Envolvimentos, o Vetor de Classes, o Vetor de Quantificadores e a Matriz de Caminho Mínimo. O conteúdo deste arquivo deve ser descrito de acordo com a especificação da Linguagem de Definição

de Esquemas (LDE) apresentada na Seção 3.2. O ABD, tem a total responsabilidade sobre as informações contidas nos arquivos descritores. É através destes, que o ABD definirá qual o BD ativo para o usuário trabalhar. Exemplos de um Arquivos Descritores de Esquemas de BD, encontram-se no Apêndice A e Apêndice B, os quais, contêm a descrição de um BD de Controle Acadêmico e de um esquema de um BD de u Simpósio, respectivamente..

5.1.1 - Matriz Envolvimentos

A escolha do nome **Matriz Envolvimentos**, deve-se ao fato de que esta matriz armazena informações sobre as ligações entre as classes (entidades e relacionamentos) do BD, indicando os caminhos possíveis para a geração das frases simples.

A opção escolhida para a estrutura de dados da Matriz Envolvimentos, foi a de implementar uma *matriz esparsa* [Horowitz87]. A escolha de *matriz*, deu-se pelo fato de que sua própria estrutura de tabela, já mapeia naturalmente os envolvimentos entre as classes. A escolha de matriz *esparsa*, foi porque a maioria das células estaria vazia. A Figura 5.2, mostra a estrutura de dados da Matriz Envolvimentos com dimensões (1 a N) onde N, é o número de classes contidas no arquivo descritor do esquema do BD.

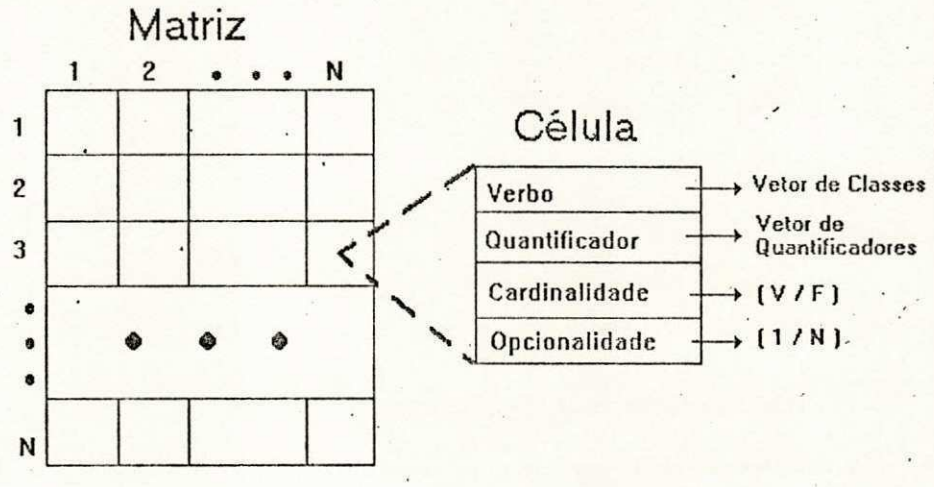


Fig. 5.2 - Matriz de Envolvimentos

Seja **X** um índice de linha ou coluna da Matriz Envolvimentos. O nome da classe relativo a **X**, está armazenado no campo nome do vetor de classe [X] (Seção 5.1.2).

No campo *Verbo*, é guardado o verbo que existe na ligação entre as classes; no campo Opcionalidade é registrado se o envolvimento de uma classe **X** para uma classe **Y**, é opcional; no campo *Cardinalidade* é guardada a cardinalidade do relacionamento (**1** ou **N**) que poderá ser utilizada na explanação das frases simples e no campo *Quantificador* é guardado o quantificador definido para o relacionamento (**todo**, **toda** e **existe**). Como exemplo, observemos o esquema de BD apresentado na Figura 5.3, seguido de descrição em LDE:

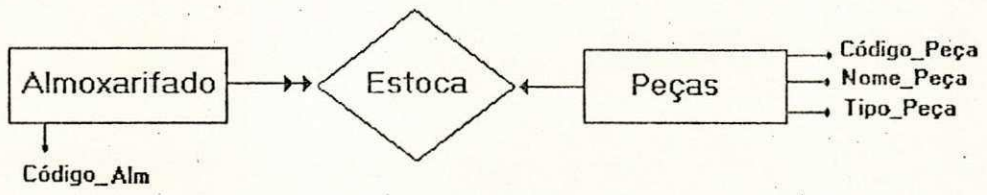


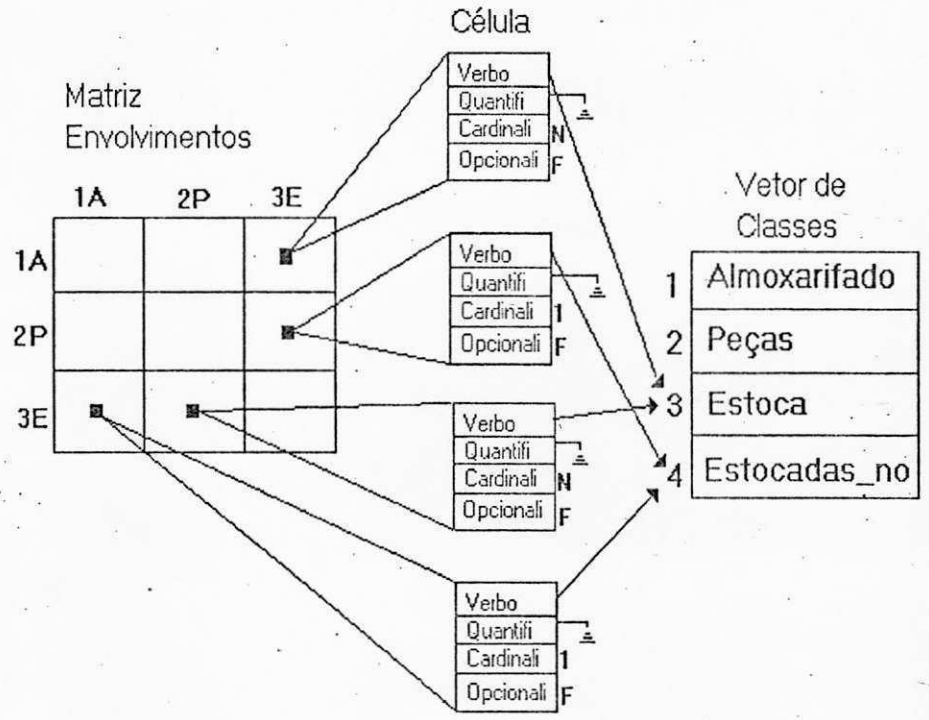
Fig. 5.3 - Esquema de um BD Almoxarifado-Peças

Descrição do esquema da Figura 5.3, em LDE:

```
ENTIDADE Almoxarifado
    ATRIBUTOS
        Codigo-Alm UNICO NUMERICO
    RELACIONAMENTOS
        (Estoca) estoca (Peças) N

ENTIDADE Peças
    ATRIBUTOS
        Codigo-peça UNICO NUMERICO
        Nome-peça CADEIA
        Tipo-peça CADEIA
    RELACIONAMENTOS
        (Estoca) estocadas_no (Almoxarifado) 1
```

Seja *A*(Almoxarifado), *P*(Peças), *E*(Estoca) e *X* --> *Y* (de *X* para *Y*). A Figura 5.4, mostra como a Matriz Envolvimentos estaria preenchida para representar um envolvimento direcionado do tipo: *A* --> *E*, *E* --> *P*, *P* --> *E* e *E* --> *A*, mostrado na Figura 5.3.

Obs.: Sejam *X* e *Y* classes de um esquema de BD. SE X=Y, então Envolvimento[X,Y] é vazio. Esta observação, especifica que não é considerada a existlia de envolvimento de uma classe consigo mesma, resultando, assim, em uma célula vazia na Matriz Envolvimentos.



Obs.: (V / F) = Verdade / Falso, (1 / N) = Cardinalidade, (≙) = Vazio

Fig. 5.4 - Matriz de Envolvimentos Preenchida

Nota-se que as células da Matriz Envolvimentos que estão vazias, demonstram a falta de envolvimento direto entra as classes. Por exemplo, entre [Almoxarifado,Peças] e [Estoca,Estoca].

5.1.2 - Vetor de Classes

O **Vetor de Classes** é a estrutura de dados que contém os nomes de todas as entidades, verbos e atributos relativos a cada entidade, especificados no arquivo que

contém a descrição do esquema do BD. Todas as demais estruturas de dados possuem apontadores para este vetor, evitando a duplicidade de dados. O Vetor de Classes junto com a Matriz Envolvimentos, concentram basicamente toda a informação necessária para o funcionamento do Sistema. A Figura 5.5, apresenta a estrutura do Vetor de Classes.

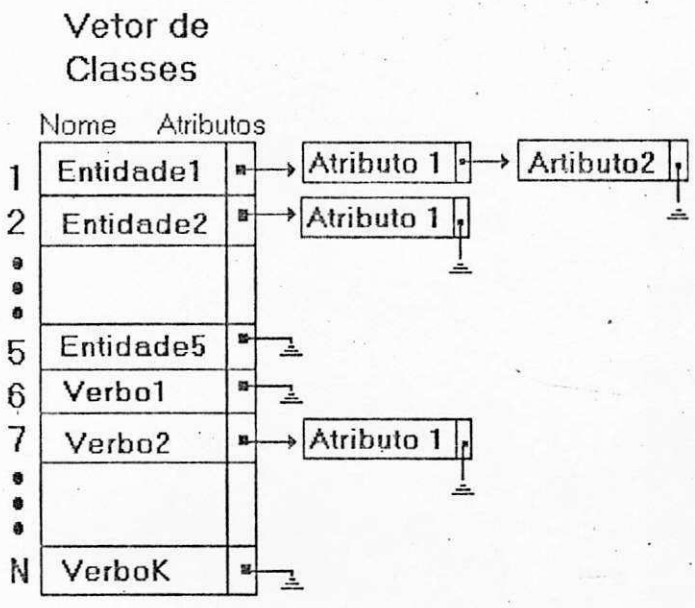


Fig. 5.5 - Vetor de Classes

Observa-se que a lista de atributos é uma *Estrutura Dinâmica* [Horowitz87]. Esta escolha, baseou-se no fato de o número de atributos ser variável para cada objeto da classe. A Figura 5.6, mostra o Vetor de Classes, caso fosse inicializado com o arquivo contendo a descrição em LDE do esquema apresentado na Figura 5.3.

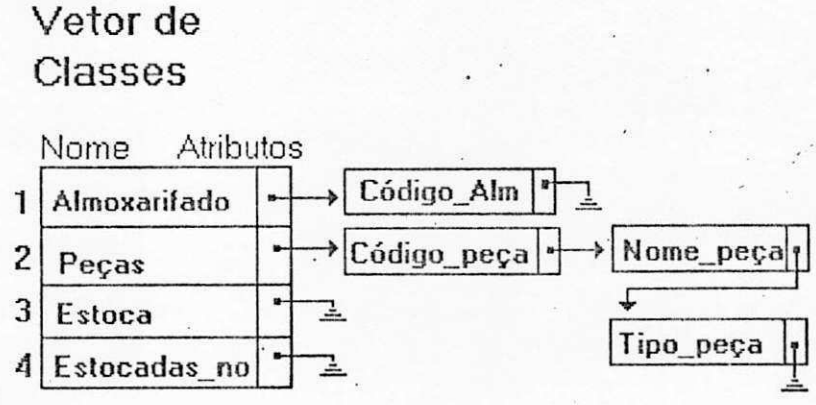
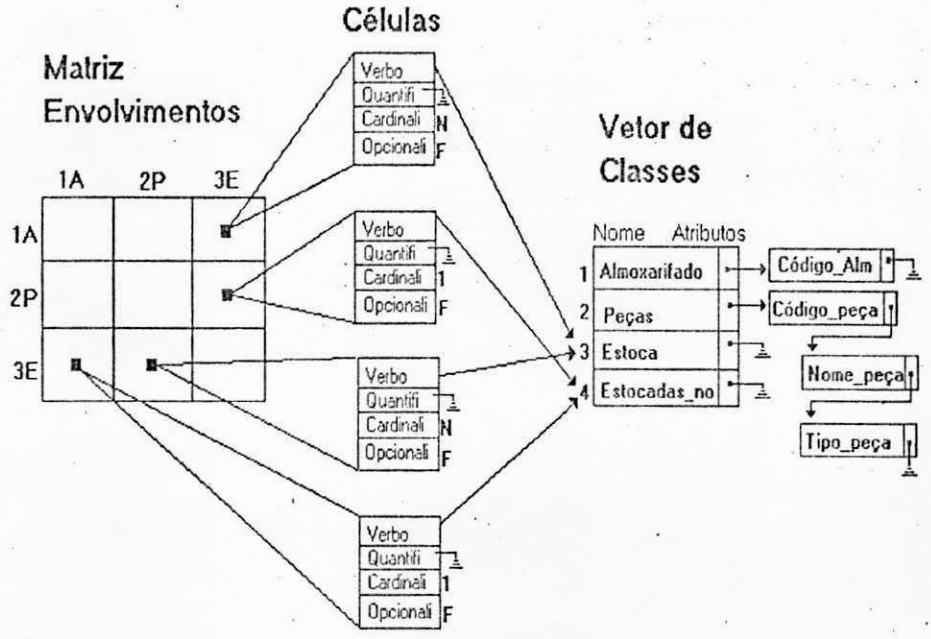


Fig. 5.6 - Vetor de Classes Preenchido

A Figura 5.7, mostra como as estruturas de dados mencionadas nesta Seção estriam interligadas, caso se estivesse trabalhando com o esquema apresentado na Figura 5.3 (BD Almoxarifado-Peças).



Obs.: (V / F) = Verdade / Falso, (1 / N) = Cardinalidade, (⊥) = Vazio.
(A) = Almoxarifado, (P) = Peças e (E) = Estoca

Fig. 5.7 - Interligação das Estruturas de Dados

5.1.3 - Vetor de Quantificadores

A estrutura de dados que armazena os quantificadores é a mesma utilizada para as palavras reservadas da LDE, visto que os quantificadores são também palavras reservadas e estão definidos na própria LDE.

Os quantificadores existenciais (**todo, toda** e **existe**) servem para dar maior riqueza semântica à LDE e para uma possível utilização na explanação das frases simples geradas pelo Sistema. Estes quantificadores são referenciados pelo campo *Quantificador* das células da Matriz Envolvimento, apresentado na Figura 5.2. Poderiam ser utilizados quantificadores para gerar frases simples, como as apresentadas a seguir:

- Todo professor está lotado em um departamento,
- Existe professor que não ensina disciplina,
- Toda disciplina possui professor,
- etc.

5.2 - Arquivo de Objetos

O **Arquivo de Objetos**, armazena os objetos definidos pelo ABD, que serão utilizados para ativar a visão temporária que o usuário terá do BD e sobre a qual poderá realizar suas consultas. Caso o usuário deseje acessar informações não permitidas pela visão atual, o ABD se encarregará de ativar uma outra visão desejada pelo usuário, caso este tenha permissão de utilizá-la. O Arquivo de Objetos, segue a sintaxe definida pelo diagrama da Figura 5.8.

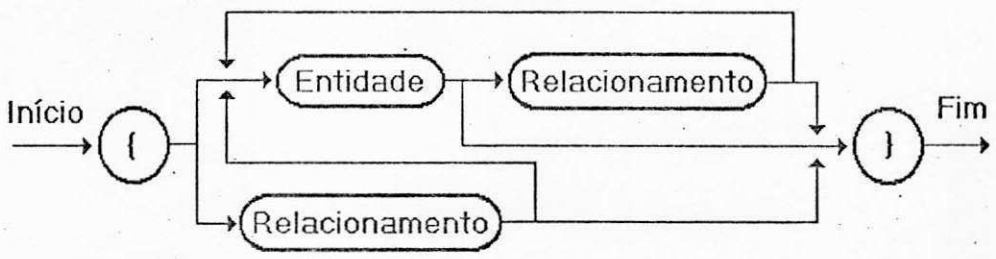


Fig. 5.8 - Sintaxe de um Objeto

Exemplos de objetos que poderiam ser definidos para o BD de Controle Acadêmico (Apêndice A):

{ Estudante }
{ Curso }
{ Estudante Matricula Disciplina Oferece Curso }
{ Candidato Indica Delegado }
{ Professor }

As entidades e/ou relacionamentos definidos no Arquivo Objetos, devem estar com a mesma grafia daqueles definidos no Arquivo Descritor do Esquema do BD (Seção 5.1). Caso isto não aconteça, será acusado um erro, em tempo de compilação, no Arquivo de Objetos. O procedimento para a inicialização deste arquivo, está definido na Seção 4.2.5.2 (Modo Usuário).

5.2.1 - Lista de Objetos

A estrutura de dados escolhida para armazenar os objetos definidos pelo ABD, foi a de uma *Lista Encadeada* [Horowitz87]. Escolheu-se uma estrutura de dados dinâmica, dado que não se tem o controle do número de objetos definidos e de quantos elementos compõem cada objeto. A Figura 5.9, apresenta a Lista de Objetos.

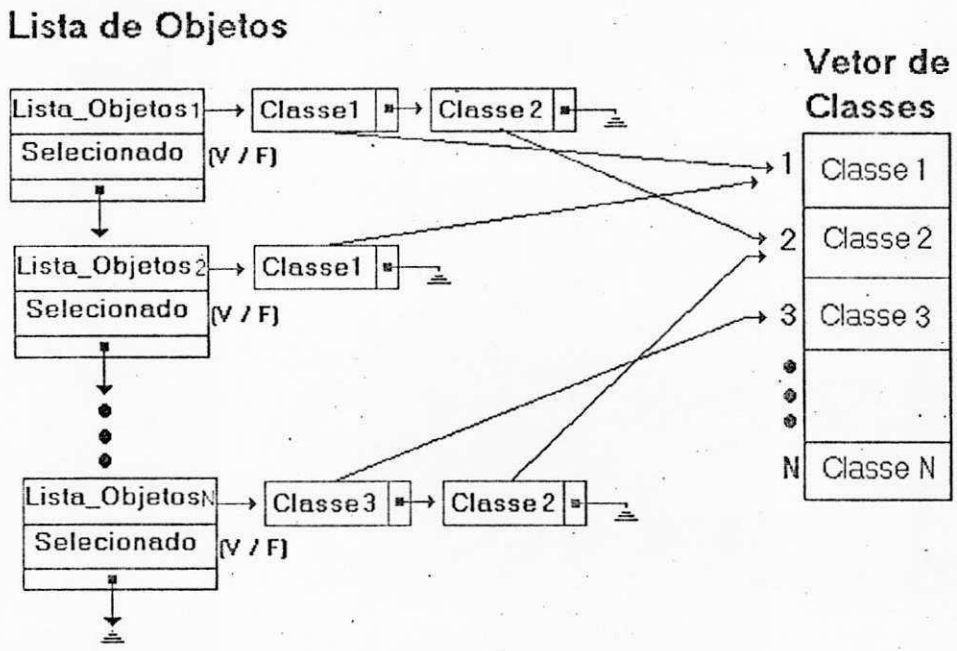


Fig. 5.9 - Lista de Objetos

O campo *Selecionado*, serve para identificar os objetos que serão utilizados para a formação das frases simples para o usuário. Caso mais de um objeto tenha sido selecionado como (Verdade), significa que houve ambigüidade na consulta realizada ao BD. O campo *Lista-Objetos*, indica quais as classes de cada objeto definido. Cada classe aponta para o Vetor de Classes (Seção 5.1.2) evitando assim, a duplicação de informação.

A Figura 5.10, mostra como a estrutura de dados da *Lista-Objetos*, estaria preenchida, caso fossem lidos os objetos definidos na Seção 5.2.

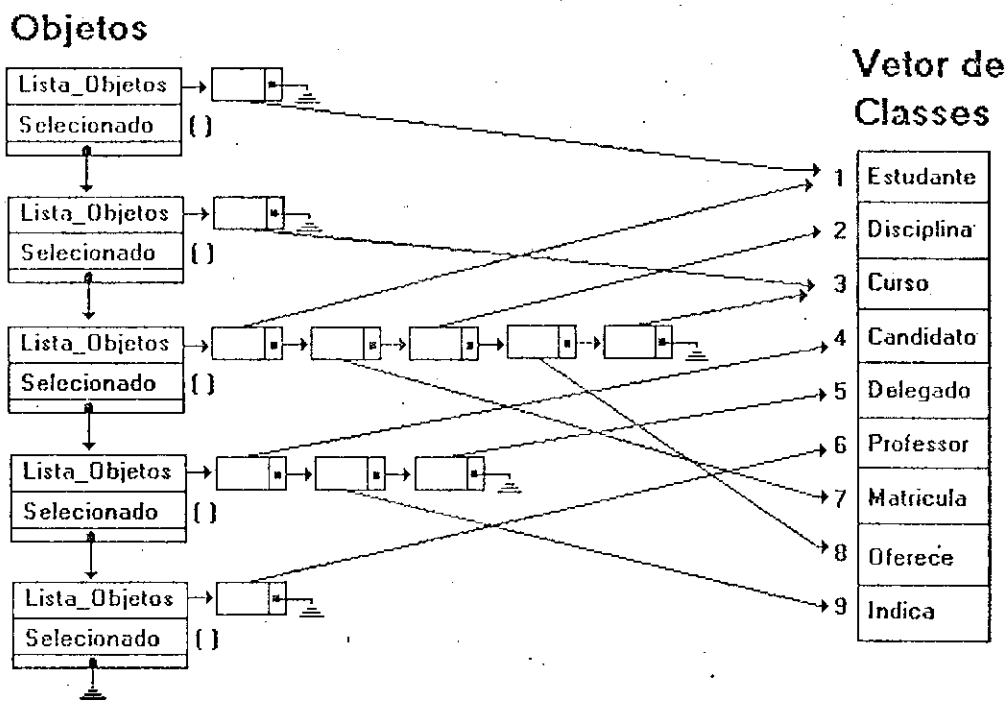


Fig. 5.10 - Lista Objetos Preenchida

As estruturas de dados temporárias, como por exemplo, a Matriz de Caminho Mínimo, serão descritas à medida que forem sendo mencionadas, para facilitar o entendimento.

No próximo capítulo, serão apresentadas algumas estratégias utilizadas para se realizar a explanação de consultas ambíguas, juntamente com o algoritmo Cruzamento que utiliza as estruturas de dados mencionadas neste capítulo.

Capítulo VI

6 - Explanação de Ambigüidades

Neste capítulo, serão apresentadas algumas estratégias utilizadas para se explanar interpretações, explanação de consultas ambíguas, geração de frases simples, planejar caminhamento no esquema do BD e, por fim, a geração de frases mais completas que são as sentenças.

6.1 - Explanando Interpretações

Explanar interpretações significa expor ao usuário, quais as interpretações que o processador de consultas assumiu como verdadeiras para a consulta realizada. Caso haja mais de uma interpretação para uma dada consulta, o usuário deve ser notificado de sua existência e, assim, decidir se alguma das interpretações expostas pelo processador expressa a sua verdadeira intenção.

No DETECT, assumiu-se que é de função do processador de consultas, além de detectar ambigüidade nas consultas realizadas, expor ao usuário as interpretações possíveis para a consulta.

A seguir, serão apresentadas três, entre as possíveis, estratégias para se realizar explanação de consultas ambíguas.

Estratégia 1: Listar o conjunto de classes envolvidas na interpretação, ou apresentar um subdiagrama do modelo de dados. Esta estratégia foi utilizada em SQL-Fácil [Bezerra89].

Característica: O usuário precisa conhecer o modelo de dados, para que possa compreender o que está sendo exposto.

Estratégia 2: Utilizar a álgebra ou cálculo relacional para realizar a explanação.

Característica: Válidas apenas para usuários especializados.

Estratégia 3: Expor a interpretação em linguagem 'quase natural' através de menu analítico.

Característica: É uma interface bem mais amigável, à qual o usuário se adapta mais rapidamente.

O DETECT, baseia-se na *Estratégia3*, e é sobre ela que o trabalho está centrado.

6.2 - Explanação de Consultas Ambíguas

A explanação de consultas ambíguas é baseada nos objetos selecionados pelo processador de consultas. No DETECT, o processador de consultas trabalha com os objetos definidos pelo ABD e o Algoritmo Cruzamento (ver Seção 6.4) para formulação das frases simples.

A seguir serão apresentados alguns exemplos de explanações baseadas nos objetos selecionados.

Considere o esquema ERE e sua descrição em LDE, do BD de Controle Acadêmico (Apêndice A). Seja a seguinte consulta:

Consulta 1:

LISTE Nota Nome-disc **ONDE** Nome-curso = 'Medicina'

A conexão (ver Seção 2.4.1) é C = {Nota, Nome-disc, Nome-curso}. Considerando vários objetos definidos pelo ABD, os seguintes objetos poderiam ser selecionados:

Obj-1 = { matricula, Disciplina, oferece, Curso } e
Obj-2 = { matricula, Estudante, registra, Curso }

Considerando os dois objetos selecionados acima, os verbos e as preposições definidas na descrição do esquema do BD, a seguinte explanação poderia ser realizada:

1ª - Estudante matriculado na disciplina oferecida pelo curso,

2ª - Estudante matriculado na disciplina e registrado no curso.

Considere o mesmo esquema ERE e sua descrição em LDE utilizados no exemplo anterior. Seja a seguinte consulta:

Consulta 2:

LISTE Nome-estud **ONDE** (Tipo-bolsa = 'CNPQ') **E**
(Indica-cand = 003)

A conexão é C = {Nome-estud, Tipo-bolsa, Indica-cand}. Considerando alguns objetos definidos, os seguintes poderiam ser selecionados:

Obj-1 = { Delegado, decide, Bolsa, deseja, Candidato }
Obj-2 = { Delegado, indica, Candidato }

Considerando os dois objetos selecionados acima, os verbos e preposições definidas na descrição do esquema do BD, poderia ser gerada a seguinte explanação:

1ª - Delegado decide bolsa desejada pelo candidato,

2ª - Delegado indica candidato.

Provavelmente o usuário optaria pela 2ª explanação, pois, o caminho lógico desejado, parece ser o caminho no qual o 'Candidato possui bolsa indicada por um delegado' e não decidida pelo delegado. Isto é fácil observar, pelo fato de que o usuário mencionou em sua consulta o atributo 'Indica-cand', o que leva ao caminho lógico no qual 'Delegado indica bolsa', porém, não é tarefa do processador do DETECT inferir sobre a consulta realizada.

Um usuário mais experiente, usaria a cláusula '*USANDO Caminho-lógico*' (ver Seção 3.3) na qual seria mencionado o caminho lógico 'USANDO Indica', evitando assim, a explanação da ambigüidade pelo sistema.

Considere o esquema de BD de um Simpósio apresentado no Apêndice B, e sua descrição em LDE. Seja a seguinte consulta:

Consulta 3:

LISTE Nome-artigo
ONDE (Nome-confer = 'Energia Nuclear') **E**
(Nome-seção = 'Física Nuclear') **E**
(Nome-julgador = 'Pedro da Silva')

A conexão é C = {Nome-artigo, Nome-confer, Nome-seção, Nome-julgador}. Considerando vários objetos definidos pelo ABD, os seguintes objetos poderiam ser selecionados:

Obj-1 ={Julgadores, seleciona, Artigos, envia, Conferências, divide, Seções}
Obj-2 = {Julgadores, seleciona, Artigos, agrupa, Seções, divide, Conferências}

Considere os objetos selecionados anteriormente e os verbos e preposições definidos na descrição do esquema do BD. A seguinte explanação poderia ser realizada:

1ª - Julgadores selecionam artigos *enviados* para as conferências divididas em seções,

2ª - Julgadores selecionam artigos *agrupados* por seções divididas em conferências.

Na 1ª frase, tem-se os nomes dos artigos selecionados por 'Pedro da Silva' e enviados para a conferência 'Energia Nuclear' cuja seção é 'Física Nuclear' e na 2ª frase, tem-se os nomes dos artigos selecionados por 'Pedro da Silva' e agrupados pela seção 'Física Nuclear' da conferência 'Energia Nuclear'. A diferença é bastante sutil, porém, poderia provocar alguns problemas para o usuário, caso não hovesse uma notificação pelo sistema da ocorrência desta ambigüidade. A diferença básica entre 'Enviados' e 'Agrupados', é que os artigos enviados, são todos os artigos submetidos ao simpósio, enquanto que os agrupados, já são os artigos selecionados para publicação.

A seguir será apresentado um exemplo bem claro, que ilustra a importância da explanação de ambigüidades detectadas pelo sistema.

Considere o seguinte esquema de um BD de uma Loja de Departamentos, apresentado na Figura 6.2.

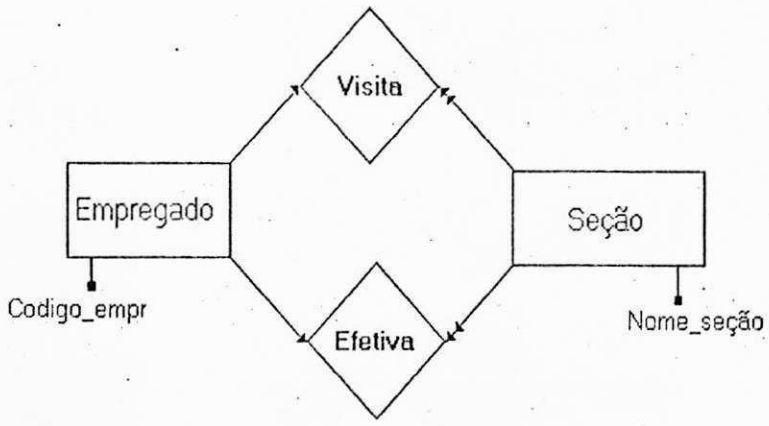


Fig. 6.2 - Esquema de BD Loja de Departamentos

Considere que o usuário realize a seguinte operação no BD em uma linguagem fictícia:

Demita todos os empregados da seção de vendas.

Em uma loja de departamentos, normalmente após as festas de fim de ano, os funcionários extras são dispensados. A falta de uma explanação pelo sistema da existência, no modelo de dados, de empregados 'Visitantes' e 'Efetivos', poderia causar um grande transtorno para a empresa, pois, caso o sistema escolhesse o caminho lógico dos funcionários *efetivos*, todo um processo de demissão seria ativado para o grupo errado dos empregados.

Na próxima seção, será apresentada a estratégia utilizada pelo DETECT para gerar frases simples através do esquema do BD descrito em LDE.

6.3 - Gerando Frases Simples

A geração de frases simples inteligíveis, está diretamente ligada ao bom senso com que o ABD define os objetos, verbos e preposições. Nesta seção será apresentada qual a estratégia utilizada pelo DETECT para gerar as frases simples na explanação de ambigüidades.

Inicialmente, a estratégia projetada para gerar as frases simples pelo DETECT, englobava várias estruturas gramaticais da Língua Portuguesa, como artigos, numerais, etc. Pensou-se na possibilidade de gerar frases como as apresentadas a seguir:

- O pesquisador participa de projetos,
- Os pesquisadores participam de projetos,
- A pesquisadora participa de projetos,
- As pesquisadoras participam de projetos.

O problema observado, reside justamente na riqueza semêntica da Língua Portuguesa. Nota-se nas frases apresentadas anteriormente, que é necessário mudar o artigo definido, para acompanhar a variação do agente da frase e vice-versa. Este tipo de problema não existiria, por exemplo, se fosse utilizada a Língua Inglesa, pois o artigo '*The*' (o, a, os, as) seria comum a todas as frases, embora houvesse uma variação do agente e eventualmente do verbo. A inflexibilidade do artigo já seria de grande ajuda para gerar frases mais próximas da linguagem coloquial.

A seguir será apresentado como o DETECT formula frases simples através do esquema do BD descrito em LDE (ver seção 3.2).

Lembrando que o termo **envolvimento** significa, neste trabalho, uma espécie de associação direta entre um par de classes, considere os seguintes casos:

1º Caso: **Envolvimento Entidade-Relacionamento**

Seja o esquema em LDE:

```
ENTIDADE Entidade-1
    ATRIBUTOS
        Atributo-1
    RELACIONAMENTOS
        OPCIONALMENTE (Relacionamento-1) Verbo-1 (Entidade-2) 1/N

ENTIDADE Entidade-2
    ATRIBUTOS
        Atributo-1
    RELACIONAMENTOS
        (Relacionamento-1) Verbo-2 (Entidade-1) 1/N
```

As seguintes frases poderiam ser geradas:

1ª - Opcionalmente Entidade-1 Verbo-1 Entidade-2

2ª - Entidade-2 Verbo-2 Entidade-1

Exemplo: Considere as seguintes descrições em LDE do esquema do BD apresentado no Apêndice A.

```
ENTIDADE Professor
    ATRIBUTOS
        Nome-prof CADEIA
    RELACIONAMENTOS
        OPCIONALMENTE (Ensína) Ensina (Disciplina) N

ENTIDADE Disciplina
    ATRIBUTOS
        Nome-disc CADEIA
    RELACIONAMENTOS
        (Ensina) Ensinada_pelo (Professor) N
```

As seguintes frases simples poderiam ser geradas:

1ª - Professor opcionalmente ensina disciplina,

2ª - Disciplina ensinada pelo professor.

2º Caso: **Envolvimento Entidade-Entidade**
(Especialização/Generalização)

Seja o seguinte esquema em LDE:

```
ENTIDADE Entidade-1 EH Entidade-2
    ATRIBUTOS
        Atributo-1 CADEIA
    RELACIONAMENTOS
        (Relacionamento) Verbo (Entidade-3)

ENTIDADE Entidade-2
    ATRIBUTOS
        Atributo-1 CADEIA
    RELACIONAMENTOS
        (Relacionamento) Verbo (Entidade-3)
```

As seguintes frases poderiam ser geradas:

1ª- Entidade-1 é entidade-2,

2ª - Entidade-2 pode ser entidade-1.

Exemplo: Considere as seguintes declarações em LDE do esquema do BD apresentado no Apêndice A:

```
ENTIDADE Estudante
    ATRIBUTOS
        Atributo-1 CADEIA
    RELACIONAMENTOS
        (Matricula) matriculado-na (Disciplina) N

ENTIDADE Candidato EH Estudante
    ATRIBUTOS
        Atributo-1 CADEIA
    RELACIONAMENTOS
        (indica) Indicado_pelo (Delegado)
```

As seguintes frases simples poderiam ser geradas:

1ª - Candidato é estudante

2ª - Estudante pode ser candidato.

Na próxima seção, será apresentada uma estratégia automática, que é a utilizada pelo DETECT, para explanar ambigüidades.

6.4 - Planejando Cruzamento

Uma estratégia geral para explanar uma interpretação de uma consulta será apresentada nesta seção. Esta estratégia resolve as seguintes questões:

a) Em que sentido, se há uma variedade, irá um envolvimento ser percorrido para gerar uma frase simples ?

b) Em que ordem devem ser apresentadas as frases ?

c) Como podem ser combinadas as frases simples para que formem sentenças ?

Matriz Envolvimentos Simplificada

Para efeito de ilustração e para facilitar o acompanhamento do Algoritmo Cruzamento, considere uma simplificação da Matriz Envolvimentos apresentada na Seção 5.1.1. Todas as informações sobre envolvimentos estão apresentadas na matriz **ENVOLVE** e no vetor **VERBOS**. A preposição acompanha o verbo como forma de sufixo ou prefixo separado pelo caráter "_" (sublinha). A cada envolvimento está associado um índice. Seja J um índice de um envolvimento **direcionado da classe S para a classe D**. Então, ENVOLVE[S,D] = J e ENVOLVE[D,S] = -J. Caso não haja envolvimento entre as classes, então ENVOLVE[S,D] = ENVOLVE[D,S] = 0. O verbo correspondente ao envolvimento J é armazenado em VERBO[J]. A Figura 6.4.1 e Figura 6.4.2 mostram os conteúdos da matriz envolvimentos e do vetor de verbos, respectivamente, para o esquema ERE apresentado no Apêndice A.

	1 Estu	2 Prof	3 Disc	4 Curs	5 Bols	6 Dele	7 Cand	8 Matr	9 Ofer	10 Regi	11 Indi	12 Dese	13 Deci	14 Ensi
1 Estudante							-8	-3		-1				
2 Professor						-8								7
3 Disciplina								3	-2					-7
4 Curso									2	1				
5 Bolsa												-5	-6	
6 Delegado		8									4		6	
7 Candidato	8										-4	5		
8 Matricula	3		-3											
9 Oferece			2	-2										
10 Registra	1			-1										
11 Indica						-4	4							
12 Deseja					5		-5							
13 Decide					6	-6								
14 Ensina		-7	7											

Fig. 6.4.1 - Matriz Envolvimento

Indice	Verbo	Indice	Verbo
1	registra	-1	registrado_no
2	oferece	-2	oferecida_pelo
3	matricula	-3	matriculado_na
4	indica	-4	indicado_pelo
5	deseja	-5	desejada_pelo
6	decide	-6	decidida_pelo
7	ensina	-7	ensinada_pelo
8	é	-8	pode ser

Fig. 6.4.2 - Vetor de Verbos

Algoritmo Cruzamento

Com o objetivo de gerar uma explanação coerente da interpretação, o plano é: *cruzar os envolvimentos no objeto ao longo do conjunto de caminhos.*

Uma classe é dita **mencionada**, quando um dos atributos utilizados é mencionado na consulta. A meta é: *visitar a classe destino de um objeto aproximadamente na mesma ordem que eles são mencionados na consulta.* Serão visitados todos os nodos no caminho, porém, dando-se prioridade para o caminho mais curto. Daí, o próximo caminho no plano será o menor de uma classe visitada para a próxima classe destino não visitada. Os envolvimentos restantes do objeto serão adicionados ao plano, sem no entanto, assumir uma ordem particular.

Dada uma interpretação, um algoritmo denominado **Cruzamento**, gera um plano que especifica a ordem e a orientação na qual os envolvimentos serão percorridos. A interpretação é representada pela lista **INTERPRETATION** na qual cada elemento é um índice para a classe no objeto. O plano criado pelo algoritmo é representado por uma lista **PLAN**, que é composta por dois elementos sublistas das classes, e cada sublista representa arestas no subgrafo do esquema ERE.

O algoritmo Cruzamento, usa a matriz **ENVOLVE**, descrita anteriormente, para construir as matrizes intermediárias **SHORT** e **FIRST**. SHORT[X,Y] contém o número de arestas no caminho mais curto entre a classe X e Y. Esta matriz é utilizada para determinar o início do próximo caminho selecionado. O destino é sempre a primeira classe destino

não visitada. FIRST[X,Y] contém a primeira classe sobre um caminho mais curto da classe X para a classe Y. Esta matriz é usada para determinar a classe subsequente no caminho. É necessário estabelecer uma estratégia para identificar quais as classes que foram visitadas na construção do plano.

Os parâmetros do Algoritmo Cruzamento são os seguintes:

Parâmetros de Entrada

- *INTERPRETATION* : é uma interpretação possível para a consulta do usuário, que será explanada.
- *TARGET* : são as classes que contém atributos mencionados na consulta do usuário.

Parâmetro de Saída

- *PLAN* : é o plano gerado pelo Algoritmo Cruzamento, no qual está expresso o caminho percorrido no esquema do BD, para realizar a interpretação.

O algoritmo será apresentado na página seguinte:

```
Algoritmo Cruzamento (INTERPRETATION,TARGET,PLAN)

/* Inicialização da procedure caminho mais curto */
For i in INTERPRETATION
    For j in INTERPRETATION
        If INVOLVE[i,j] = 0 then /* sem envolvimento */
            SHORT[i,j] := infinity  /* maior tamanho possível */
        else {  /* classes são adjascentes }
            SHORT[i,j] = 1
            FIRST[i,j] = j }
/* Algoritmo Marshall's p/ computar caminho mais curto */
For k in INTERPRETATION
    For i in INTERPRETATION
        For k in INTERPRETATION
            distance := SHORT[i,k] + SHORT[k,j]
            If distance < SHORT[i,j] then
                { SHORT[i,j] := distance
                FIRST[i,k] := FIRST[I,K] }
/* Inicialize variáveis PLAN */
PLAN := empty
Mark HEAD(INTERPRETATION) as visited
For classe in TAIL(INTERPRETATION)
    Mark class as UnVisited
/* Encontre caminho mais curto da classe visitada p/ a classe destino não visitada. */
For target in TARGET
    If target is not visited then
        closet := HEAD(INTERPRETATION)
    /* Inicialmente assume o primeiro destino como fechado */
    For class in TAIL(INTERPRETATION)
        If (class is visited) and ( SHORT[class,target] < SHORT[closest,target] ) THEN
            closest := class
    /* Adicione caminho p/ plano corrente */
    While closest not equal target {
        /* Próxima classe no caminho */
        class := FIRST[closest,target]
        Mark class as visited
        append(closest,class) to PLAN
        /* Marca envolvimento como processado*/
        SHORT[closest,class] := 0
        closest := class
    }
{ Checa restante dos envolvientos  por causa de ciclos }
LIST := TARGET
While LIST is not empty{
    i := HEAD(LIST)
    For j in TAIL(LIST)
        If SHORT[i,j] = 1 then /* Envolvimento nao marcado */
            append(i,j) to PLAN
    LIST := TAIL(LIST)
}
FIM Cruzamento
```

Como exemplo, considere a seguinte consulta baseada no esquema apresentado no Apêndice A:

LISTE Nome-Curso Nome-Disc Nome-Prof **ONDE** Nota-Candidato > 7.5

Usando a abordagem Objeto Mínimo, existem duas interpretações:

1ª - Interpretação: {Curso, registra, Estudante, Candidato, matricula, Disciplina, Ensina, Professor}

2ª - Interpretação: {Curso, oferece, Disciplina, ensina, Professor, matricula, Estudante, Candidato}

Então, os seguintes planos são produzidos pelo algoritmo:

1º- Plano: ((Curso,registra), (registra,Estudante), (Estudante, Candidato), (Estudante,matricula),(matricula,Disciplina), (Disciplina,ensina), (ensina,Professor))

2º - Plano = ((Curso, oferece), (oferece, Disciplina), (Disciplina, ensina),(ensina,Professor), (Disciplina, matricula), (matricula, Estudante),(Estudante, Candidato))

A partir destes planos e tomando como base a matriz ENVOLV e o vetor VERBOS, apresentados anteriormente, a frase será montada. Para tanto, utiliza-se das coordenadas geradas pelo algoritmo através da lista PLAN. Por exemplo, a cada par de classes do 1º plano apresentado anteriormente, estão associados os seguintes verbos:

```
ENVOLV(Curso,registra) = 1          ->  Verbo[1] = registra,
ENVOLV(Estudante,matricula) = -3    ->  Verbo[-3] = matriculado_na,
ENVOLV(Disciplina,ensina) = -7      ->  Verbo[-7] = ensinada_pelo.
```

E assim por diante.

Estes planos gerariam as seguintes frases:

1ª - Curso registra candidato matriculado na disciplina ensinada pelo Professor,

2ª - Curso oferece disciplina ensinada pelo professor e disciplina matricula candidato.

Considere a repetição da *Consulta 3*, apresentada na Seção 6.2 baseada no Apêndice B, para verificar como se apresentam os planos gerados pelo algoritmo cruzamento:

Consulta:

LISTE Nome-artigo
ONDE (Nome-confer = 'Energia Nuclear') **E**
(Nome-seção = 'Física Nuclear') **E**
(Nome-julgador = 'Pedro da Silva')

Usando a Abordagem Objeto Mínimo, existem duas interpretações:

1ª Interpretação: {Julgadores, seleciona, Artigos, envia, Conferências, divide, Seções}

2ª Interpretação: {Julgadores, seleciona, Artigos, agrupa, Seções, divide, Conferências}

Considere que para os objetos selecionados anteriormente, os seguintes planos são produzidos:

1º Plano: ((Julgadores,seleciona), (seleciona,Artigos) (Artigos,envia), (envia,Conferências), (Conferências,divide), (divide,Seções))

2º Plano: ((Julgadores,seleciona), (seleciona,Artigos), (Artigos,agrupa), (agrupa,Seções), (Seções,divide), (divide,Conferências))

As frases geradas seriam as seguintes:

1ª - Julgadores selecionam artigos enviados para as conferências divididas em seções,

2ª - Julgadores selecionam artigos agrupados por seções divididas em conferências.

6.5 - Gerando Sentenças

Esta seção descreve como frases simples podem ser combinadas para formarem sentenças.

Suponhamos K frases simples que têm o mesmo sujeito (uma classe S). Estas frases podem ser combinadas da seguinte forma:

S resto da frase **1** e ... e resto da frase **K.**

Considere, por exemplo, o seguinte par de frases:

Estudante registrado no curso.

Estudante matriculado na disciplina.

Teríamos: Estudante registrado no curso e matriculado na disciplina

O outro caso ocorre quando o último substantivo de uma frase é o sujeito da próxima. Neste caso o sujeito é omitido na composição da sentença.

Exemplo: - Estudante matriculado na disciplina
- Disciplina oferecida pelo curso.

Teríamos: Estudante matriculado na disciplina oferecida pelo Curso.

A geração de sentenças não foi implementada no DETECT, pois, em alguns casos, podem provocar a construção de frases muito extensas, dificultando a percepção por parte do usuário e dificultando a explanação de ambigüidades.

Conclusão

Na informática tem-se constatado que não existe solução cem por cento perfeita. Por exemplo, quando se consegue um avanço em hardware, em contrapartida, o software não o explora na íntegra. Quando se oferece um sistema com bastante recursos, possivelmente haverá problemas de desempenho. Este tipo de problema existe em todas as ramificações da informática e, como não poderia deixar de ser, ocorre com as interfaces de RU.

Atrás das grandes vantagens apresentadas ao longo deste trabalho, no que diz respeito à RU, existe um grande problema, que é o da ocorrência de ambigüidade nas consultas realizadas.

Vê-se, portanto, que se paga um preço, por se ter livrado o usuário da responsabilidade da navegação lógica no esquema do BD.

Justamente para resolver um dos problemas gerados ao se adotar uma interface deste tipo, para o usuário, é que este trabalho foi idealizado.

Uma de suas grandes contribuições, foi minimizar os efeitos colaterais causados pela ocorrência de ambigüidades em consultas realizadas em uma interface de RU. Embora este não seja um problema muito freqüente, ele pode ser bastante prejudicial ao usuário quando ocorrer.

No DETECT, o usuário toma conhecimento da ocorrência de ambigüidade em sua consulta, através das frases simples geradas pelo sistema. Tal estratégia, vem sendo adotada apenas pela minoria dos trabalhos existentes na mesma linha de pesquisa.

Dentre as contribuições apresentadas por este trabalho, destacam-se:

- A implementação de uma interface extremamante simples para se realizar consultas a BD's, por usuários não especializados,
- Uma linguagem de definição de esquemas, que facilita o mapeamento do esquema do BD, por parte do ABD, assim como uma adaptação do modelo de

dados Entidade-Relacionamento, que facilita a sua utilização por um número muito maior de usuários, por ser um modelo de dados bastante difundido,

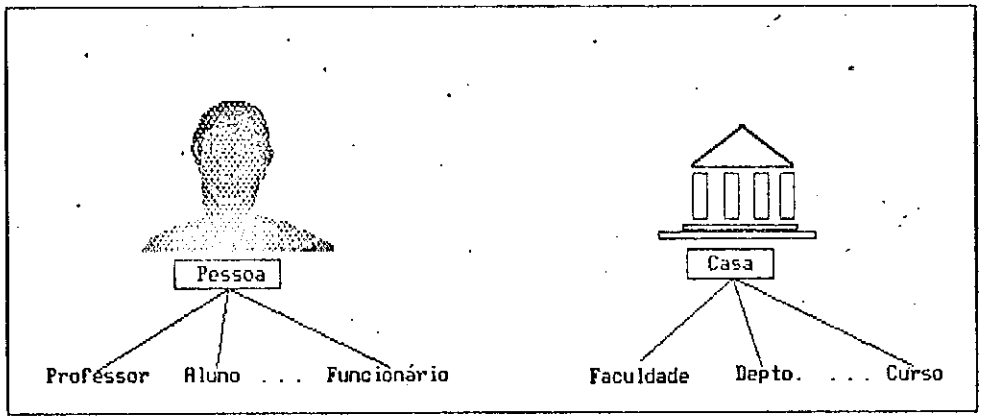
- Uma estratégia geral para a detecção de ambigüidade, que pode ser utilizada em qualquer interface, na qual o usuário realize consultas à nível de atributos, como por exemplo o que acontece, de certa forma, com as interfaces de linguagem natural,
- E por fim, obteve-se um certo grau de continuidade, de um trabalho já realizado na Dissertação de Mestrado intitulada SQL-Fácil [Bezerra89], implementando-se várias observações feitas como propostas para trabalhos futuros.

Trabalhos Futuros

Uma proposta interessante para ser implementada, seria a de se utilizar as interfaces de linguagem natural para realizar a explanação de ambigüidade. Esta alternativa seria viável, pois o Algoritmo Cruzamento, gera caminhos não ambíguos, evitando assim, o problema da geração de frases ambíguas em linguagem natural.

Uma outra proposta para a apresentação das frases geradas pelo sistema, que é um dos pontos importantes do trabalho, seria a de se utilizar 'ícones', que estariam associados a classes específicas de objetos.

A Figura a seguir, demonstra uma proposta de definição de classes de 'ícones', que seriam utilizados na explanação da frases geradas pelo sistema :



Relação de Herança entre Subclasses de Objetos

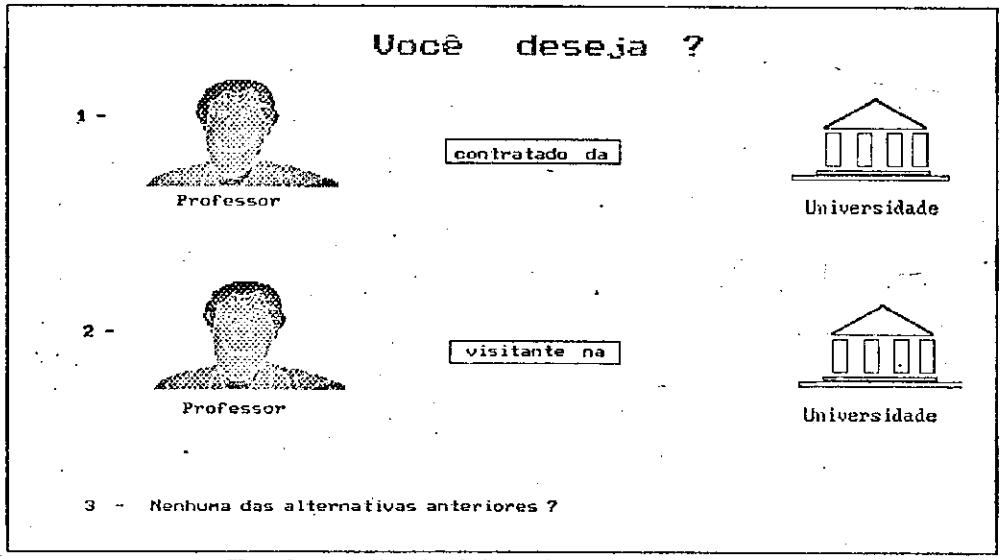
Baseado nesta definição prévia das classes de objetos, o sistema poderia montar uma explanação gráfica das frases geradas.

A seguir será apresentado um exemplo de um mapeamento de frases explanadas graficamente.

Considere a seguinte consulta baseada em um esquema hipotético de BD:

LISTE Nome-func **ONDE** Função-func = 'Professor'

A seguinte explanação poderia ser realzada caso fosse detectada ambigüidade na mesma:



Explanação Gráfica de uma Consulta

Uma explanação desta forma, poderia ser uma boa opção para a interface do DETECT, pois, esta consegue expressar a semântica da base de dados de forma mais agradável para o usuário final.

No DETECT, conseguiu-se um avanço *efetivo* do ponto de vista da independência lógica de dados, que já é natural das interfaces de RU e o que também pode se chamar de *'Transparência de Esquemas'*, pois, o usuário encontra à sua disposição, de forma bastante estruturada e de fácil compreensão, os nomes dos atributos sobre os quais realizará suas consultas, o que facilita extremamente a realização das mesmas.

Bibliografia

[Addis82] Addis, T.R. *A Relation-Based Language Interpreter for a Content-Addressable File Store.* ACM Trans. on Database Systems, July 1982, pp. 125-163.

[Aho] Aho, A.V., and Kernighan, B.W. "*q*", A universal relations system in use at Bell Laboratories. Cited in [Maier 82b]. E- mail comunications.

[Aho88] Aho, A.V., Ullman, J.D. and Ravi Sethi. *Compilers: Principies, Techiniques and Tools.* Addison-Wesley Publishing Company, 1988.

[Alves89] Alves, Claudete Mary de Souza. *SISAL - Sistema de Autoria de Lições.* Dissertação de Mestrado. COPIN - DSC / UFPB - Campina Grande - PB, Nov. 1989.

[Bezerra89] Bezerra, Ed. Pôrto. *SQL-Fácil Uma interface amigável para acesso a banco de dados relacional.* Dissertação de Mestrado. COPIN - DSC / UFPB - Campina Grande - PB, Nov. 1989.

[Biskup83] Biskup, J. & Bruggeman, *H.H. Universal Relations Views: A Pragmatic Approach*. Proc. Ninth Int'l Conf. Very large Databases, William Kaufmann, Los Altos, Calif., 1983, pp. 172-185.

[Boguraev84] Bogureav, B. K. & Sparck, K. Jones. *A Natural Language Front End to Databases with Evaluative Feedback.* Computer Laboratory, University of Cambridge, England, 1984.

[Brady85] Brady, L. I. *A Universal Relation Assumption Based on Entities and Relationships*.Database and Graphics, Mathmatics and Physics, Macquarie University, North Ryde, Australia 2113. IEEE, 1985, 208-215.

[Ceri85] Ceri, Stefano. *Methodology and tools for database design.* Elsevier Science Publishers. B. V. 1985.

[Chamberlin76] Chamberlin, D.D., Astrahm, M.M., Eswarkan,K.P., Lorie,R.A., Mehl,J.W., Reisner,P., and Wade,B.W. *SEQUEL 2: A unified approach to data definition, manipulation, and control.* IBM J. Res. Dev. 20, (Nov, 1976), 560-575.

[Chen76] Chen, P.P., *The entity-relationship model - toward a unified view of data.* ACM TDS 1,1 (March 1976), 9-36.

[Crout81] Crout, J. N. *Evaluation of Natural Language Interfaces to Database Systems: A Panel Discussion.* Proceedings of the 19th Annual Meeting of the Association for Computational Linguistics, Stanford, CA, pp. 29-42, 1981.

[Date86] Date, C.J. *Introdução a Sistemas de Banco de Dados.* Editora Campus, 1986.

[Date90] Date, C.J. & Colin J. White. *A guide to DB2.* Third Edition. Addison Wesley, 1990.

[Grosz83] Grosz, B. TEAM: *A Transportable Natural-Language Interface System.* Proceedings of Conference on Applied Natural Language Processing, Santa Monica, CA, pp. 39-45, 1983.

[Horowitz87] Horowitz, Ellis & Sahni, Sartaj. Fundamentals of Data Structures in Pascal. Computer Science Press, 1987.

[kaplan83] Kaplan, S. J. *Cooperative Responses from a Portable Natural Language Database Query System.* In "Computational Models of Discourse". Eds. M. Brady and R Berwick) pp. 167- 208. The MIT Press, Cambridge, Mass, 1983.

[Kent81] Kent, W. *Consequences of Assuming a Universal Relation.* ACM Trans. on Database Systems, 6, 4, Dec. 1981, pp. 539-556.

[Korth84] Korth, H.F., Kupper, G.M., Feigenebaun,J., Van Gelder,A., and Ullman,J.D. *System/U: A database system based on the universal relation.* ACM TDS 9,4. Sept. 1984, 331-347.

[Kuck82] Kuck, S.M & Sagiv, Y. *A Universal-Relation Database System Implemented via the Network Model.* Proc First Symp. Princ. Database Systems, ACM, New York, 1982, pp. 147-157.

[Maier82a] Maier, D., Rozenshtein, D., Salveter, S., Stein, J., and Warren, D.S. *Toward logical data independence: A relation query language without relations.* Proc. 1982 ACM SIGMOD Conf. (Orlando, FLA.), 51-60.

[Maier82b] Maier, D., and Warren, D.S. *Specifying connections for a universal relation.* Proc. 1982 ACM SIGMOD Conf. (Orlando, Fla.), 1-7.

[Maier83] Maier, D., and Ullman, J.D. *Maximal objects and the semantics of universal relation databases.* ACM Trans. Database Systems, 8, 1. March 1983, 1-14.

[**Marcowitz83**] Marcowitz, V.M., and Raz, Y. *ERROL: An entity- relationship, role oriented, query language.* In "Entity Relationship Approach to Engineering", Davis,C.G., Joadia,S., Ng,P.A., and Yeh,R.T., Eds., North-Holland, Amsterdam, pp. 83,329. 1983.

[**Moreira92**] Moreira Jr, V.M., Turnell, M.F.Q.V., Sampaio, M.C. e Nicolletti, P.S. *Detectando Ambigüidade em Consultas Formais.* 7º Simpósio Brasileiro de BD. Porto Alegre - RG, pp. 207 - 220, Maio 1992.

[**Oracle84**] Manuais de Referência. Oracle Corporation. Menlo Park, Calif. 1984.

[**Stonebraker76**] Stonebraker, M., Wong, E., Kreps, P., and Held, G. *The design and implementation of INGRES.* ACM TDS 1,3 Sept/1976,189-222.

[**Ullman83**] Ullman, J.D. *On Kent's. Consequences of Assuming a Universal Relation.* ACM Trans. on Database Systems, 8, 4, Dec. 1983, pp. 637-643.

[**Ullman89**] Ullman, J. D. *Principles of database and knowledge - base systems* - Vol. II, Computer Science Press, Rokville, Maryland, 1989.

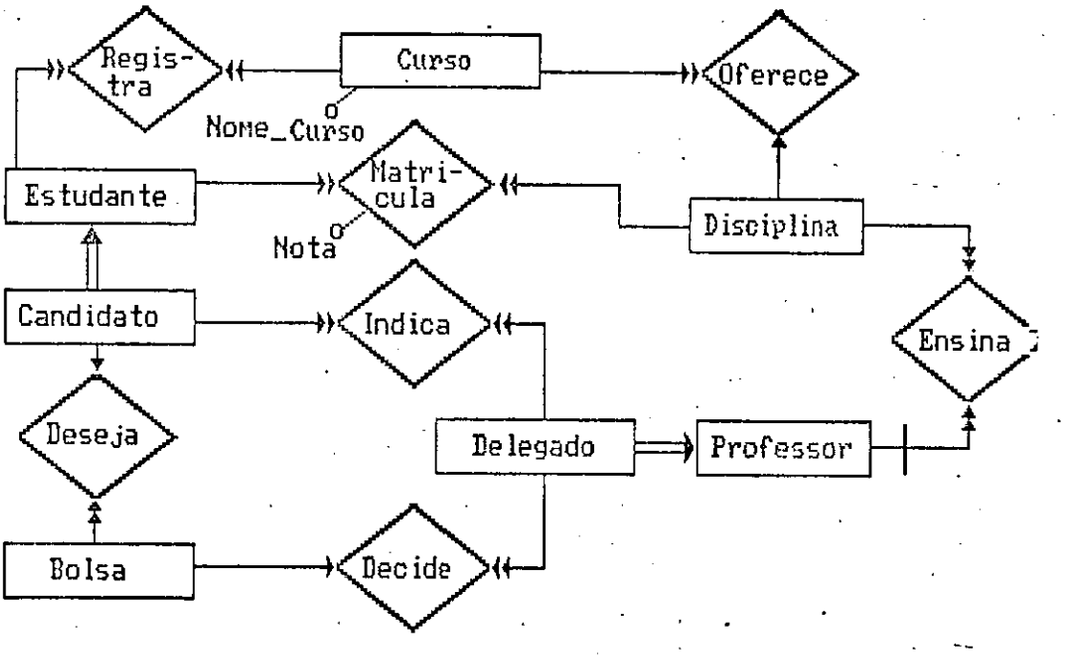
[**Vardi88**] Vardi, Moshe Y. *The Universal' Relation data model for logical idependence*. IBM Almaden Research Center. IEEE, 1988. 80-85.

[**Wald90**] Wald, J.A., Sorenson, P.G. *Explaning ambiguity in a formal query language.* ACM TDS, 15,2, pp. 126-161, June 1990.

[**Zloof75**] Zloof, M.M. *Query by example.* In Proceedings National Computer Conference (Arlignton, Va., May 1975). AFIPS Press, Anaheim, California, 1975, 431-437.

Apêndice A

Esquema de BD de um Controle Acadêmico



Esquema de um BD de um Controle Acadêmico

Definição do esquema Controle Acadêmico em LDE

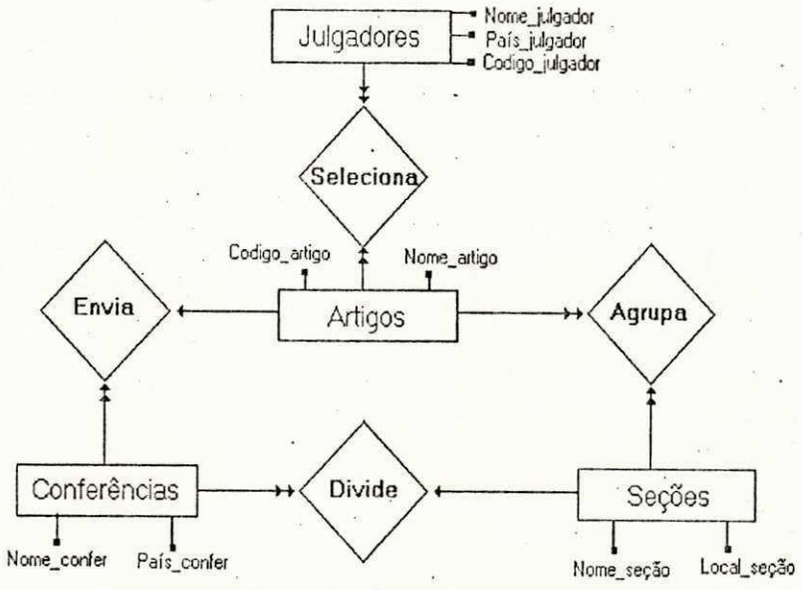
```
ENTIDADE Curso
    ATRIBUTOS
        Codigo-Curso UNICO NUMERICO
        Nome-Curso UNICO CADEIA
        Depto-Curso CADEIA
    RELACIONAMENTOS
        (Registra) registra ( Estudante ) N
        (Oferece) oferece ( Disciplina ) N
```

```
ENTIDADE Estudante
    ATRIBUTOS
        Codigo-estud UNICO NUMERICO
        Curso-estud UNICO CADEIA
        Nome-estud  CADEIA
        Idade-estud  NUMERICO
        Ender-estud CADEIA
        Tel-estud  CADEIA
    RELACIONAMENTOS
        (Matricula) matriculado_na (Disciplina) N
        (Registra)  registrado_no  ( Curso ) N

ENTIDADE Professor
    ATRIBUTOS
        Codigo-prof UNICO NUMERICO
        Nome-prof UNICO CADEIA
        Ender-prof CADEIA
        Idade-prof NUMERICO
        Depto-prof CADEIA
        Tel-prof  CADEIA
    RELACIONAMENTOS
        OPCIONALMENTE (Ensina) ensina (Disciplina) N

ENTIDADE Disciplina
    ATRIBUTOS
        Codigo-Disc UNICO NUMERICO
        Nome-Disc  CADEIA
        Sala-Disc  NUMERICO
    RELACIONAMENTOS
        (Matricula) matricula (Estudante ) N
        ATRIBUTOS
            Nota-Matricula UNICO NUMERICO
        CHAVE
            ( Cod-Estudante  Cod-Disciplina)
        (Ensina) ensinada_pelo (Professor) N
        (Oferece) oferecida_pelo ( Curso ) N

ENTIDADE Candidato  EH  Estudante
    ATRIBUTOS
        Codigo-Cand UNICO NUMERICO
    RELACIONAMENTOS
        (Deseja) deseja (Bolsa) 1
        (Indica)  indicado_pelo (Delegado) N
```

```
ENTIDADE Bolsa
    ATRIBUTOS
        Tipo-Bolsa UNICO CADEIA
        Prazo-Bolsa  NUMERICO
        Entidade-Bolsa  CADEIA
    RELACIONAMENTOS
        (Deseja) desejada_pelo (Candidato) N
        (Decide) decidida_pelo (Delegado) 1

ENTIDADE Delegado EH Professor
    ATRIBUTOS
        Codigo-Del UNICO NUMERICO
    RELACIONAMENTOS
        (Indica) indica (Candidato) N
        (Decide) decide (Bolsa) N
```

Apêndice B

Esquema de um BD de um Simpósio



Esquema de um BD de um Simpósio

Definição do esquema Simpósio em LDE

```
ENTIDADE Julgadores
    ATRIBUTOS
        Nome-julgador CADEIA
        País-julgadorCADEIA
        Codigo-julgador UNICO NUMERICO
    RELACIONAMENTOS
        ( Seleciona ) selecionam ( Artigos ) N

ENTIDADE Artigos
    ATRIBUTOS
        Nome-artigo CADEIA
        Codigo-artigo UNICO CADEIA
    RELACIONAMENTOS
        (Envia) enviados_para (Conferências) 1
        (Agrupa) agrupados_pelas (Seções) 1
```

ENTIDADE Conferências
ATRIBUTOS
Nome-confer CADEIA
País-conferCADEIA
RELACIONAMENTOS
(Envia) recebem (artigos) N
(Divide) divididas_em (Seções) N

ENTIDADE Seções
ATRIBUTOS
Nome-seção CADEIA
Local-seção CADEIA
RELACIONAMENTOS
(Agrupa) agrupa (Artigos) N
(Divide) divididas_em (Conderências) 1

Apêndice C

Ambigüidade em Linguagem Natural

Ambigüidade Léxica

A ambigüidade léxica ocorre quando a mesma palavra, ou expressão básica, denota entidades diferentes no mundo exterior. Por exemplo:

Uma **ação** da Vale do Rio Doce

Uma **ação** contra os devedores

Na primeira frase, **ação** se refere a um título comercial e na segunda, a alguma atividade judicial.

Ao se construir uma interface de LN, às vezes não é possível evitar a ambigüidade léxica. Nestes casos, teremos que ter um critério para decidir qual interpretação é a correta. Normalmente a Ambigüidade Léxica é resolvida pelo contexto da frase. Numa interface simples em que as frases são, com algumas exceções, autônomas (tais como comandos e perguntas diretas), pode ser necessário que o sistema peça um esclarecimento ao usuário.

Ambigüidade Sintática

Neste caso podemos ter mais de uma árvore sintática para derivar a mesma frase. A ambigüidade sintática pode acontecer em frases com mais de um verbo, na qual não está claro qual é o verbo principal:

Apague *todos* os arquivos *usando* **DEL**

Na frase anterior podem ocorrer duas interpretações:

1 - É para apagar qualquer arquivo que esteja utilizando **DEL** (DEL pode ser qualquer coisa) ou

2 - É para usar **DEL** para apagar os arquivos ?

Também pode ser gerada ambigüidade por referências pronomiais em períodos compostos, como em:

Encontre o objeto **X** do banco de dados **Z** e remova-o

A dúvida está no fato de que o usuário deseja remover o Objeto **X** ou o banco de dados **Z** ?

Existem outras possibilidades. No caso da ambigüidade sintática, ela é consequência direta das regras da gramática, e é previsível. Neste caso não é possível resolver a ambigüidade sem consultar o usuário. Em alguns casos, a análise semântica poderia resolver ambigüidades sintáticas. Na frase 'Apague todos os arquivos usando DEL', a análise semântica poderia talvez concluir que arquivos não utilizam coisas do mundo em consideração, e então sobraria a interpretação de que DEL deve ser utilizado para apagar os arquivos. Neste caso o usuário não seria consultado.

Ambigüidade Semântica

A Ambigüidade Semântica, em geral, ocorre quando temos mais de um significado para uma frase. Ocorre acompanhada de ambigüidade sintática, quando as diversas árvores sintáticas produzem análises semânticas válidas, como em:

Pedro *viu* Maria *passeando*

Na frase anterior, o que se quer dizer, é que Pedro estava passeando e viu Maria, ou Pedro viu Maria que estava passeando ?