BABATUNDE AYODELE ORESOTU

UM SISTEMA DE REPRESENTAÇAO E RECUPERAÇAO DE DADOS INCOMPLETOS E INFORMAÇAO TEMPORAL

Dissertação apresentada ao CURSO DE MESTRADO EM SISTEMAS E COMPUTAÇAO da Universidade Federal da Paraiba, em cumprimento à parte das exigências para obtenção de Grau de Mestre.

ULRICH SCHIEL

Orientador

CAMPINA GRANDE

MARÇO - 1988

UM SISTEMA DE REPRESENTAÇÃO E RECUPERAÇÃO DE DADOS INCOMPLETOS E INFORMAÇÃO TEMPORAL

O66s Oresotu, Babatunde Ayodele
Um sistema de representacao e recuperacao de dados incompletos e informacao temporal / Bbatunde Ayodele Oresotu. - Campina Grande, 1988.
88 f. : il.

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

1. Representacao de Dados - 2. Recuperacao de Dados - 3. Informacao Temporal 4. Sistemas de Computacao 5. Dissertacao I. Schiel, Ulrich II. Universidade Federal da Paraiba - Campina Grande (PB) III. Título

CDU 004.22(043)

UM SISTEMA DE REPRESENTAÇÃO E RECUPERAÇÃO
DE DADOS INCOMPLETOS E INFORMAÇÃO TEMPORAL

BABATUNDE AYODELE ORESOTU

DISSERTAÇÃO APROVADA EM 22.03.88

ULRICH SCHIEL
Orientador

ROBERT KALLEY C. DE MENEZES
Componente da Banca

JOSÉ HAMURABI N. DE MEDEIROS
Componente da Banca

CAMPINA GRANDE
MARÇO - 1988

AGRADECIMENTOS

Gostaria de agradecer ao meu orientador Professor Ulrich Schiel por seu maior interesse, incentivo e paciência ao longo da preparação deste trabalho.

Agradeço aos Professores José Luiz Neto, José Albos Rodrigues, por suas valiosas críticas e comentários sobre o manuscrito deste sistema. Também, agradeço ao Professor Edilson Ferneda por ter ajudado, particularmente na parte de implementação deste projeto. Agradeço a João Cordeiro da Fonseca e a Antonio Alvino de Souza por terem-me ajudado na redação deste trabalho.

Finalmente, agradeço aos demais professores e colegas de trabalho, cujos nomes não foram mencionados, que contribuiram e colaboraram para o sucesso deste trabalho.

Aos meus Pais e Irmãos, pelo incentivo e apoio demonstrado.

RESUMO

Este trabalho apresenta o projeto e implementação de um sistema de representação e recuperação de dados incompletos e informação temporal (RADINT).

Inicialmente, apresentamos conceitos básicos e, em seguida, analisamos detalhes de implementação do sistema. Uma estrutura para representar dados incompletos em Banco de Dados Relacional é proposta. Também, investigamos problemas relacionados com o processamento de dados incompletos.

Os principais aspectos considerados no projeto são dados incompletos, valores nulos marcados ou não marcados, fatos condicionais, valores nebulosos (fuzzy). Todos estes assuntos foram incluídos na implementação do sistema.

Enfatizamos aspectos temporais como pontos e intervalos. Também, distinguimos três tipos de dados incompletos tais como: Negativo, Desconhecido e Impreciso. Apresentamos várias formas de representação e recuperação de dado incompleto. Além do mais, o sistema tem capacidade de deduzir e responder consultas de recuperação e booleanas tais como "YES","NO","UNKNOWN" e "POSSIBLE".

ABSTRACT

This work presents the design and implemention of a system about representation and retrieval of incomplete data and temporal information.

Initially, we present the basic concepts and then we analyse in details, the major components of the system. A structure to represent incomplete data in a Relational Database Form is presented. We also investigate the problems related with the processing of incomplete data. The principal aspects considered in this project are incomplete data, marked or unmarked null values, conditional facts and fuzzy values. All these topics were included in the implemented system.

More emphasis were laid on temporal aspects like Points and Intervals. Also, we differenciate three kinds of incomplete data as Negative, Unknown and Imprecise. Several forms of representation and recuperation of incomplete data were presented. Besides all these, the system has the capacity to deduce and answer retrieval and boolean queries such as "Yes", "No", "Unknown" and "Possible".

CONTEÚDO

LISTA DAS ILUSTRAÇOES

1. INTRODUÇÃO

Atualmente, muito pouco está sendo feito esclarecimento dos problemas ligados ao processamento de dados incompletos e informações temporais. Provavelmente, esta seja a razão pela qual os Bancos de Dados (BD) atuais ofereçam pouco ou nenhun apoio sobre este assunto, embora a não disponibilidade de alguma informação necessária à resposta a uma consulta seja um problema comum para muitos sistemas de banco de dados. Evidentemente, a resposta a qualquer consulta feita quando a informação num banco de dados é incompleta, construi-se em aproximação do resultado real. Recentemente, surgiram trabalhos manifestando mais interesse nesta área e buscando uma maneira de melhorar a representação e recuperação de dado incompleto. Um desses, é **[Schiel/85a]**, motivador do desenvolvimento do presente trabalho.

O conceito de tempo é decisivo para qualquer BD, embora, infelizmente, seja tratado apenas implicitamente nos modelos existentes. Muitos trabalhos nesta área não cuidavam da representação de tempo possivelmente por causa da complexidade de fazê-lo. A maioria dos BDs convencionais são estáticos, baseados em modelo hierárquico, rede, relacional ou entidade-relacionamento. Esses bancos de dados convencionais representam um estado do mundo real em um único momento [Snodgrass/84]. Neste caso, as mudanças dos dados no mundo real que acontecem em decorrência do tempo são esquecidas e tratadas como informação velha e obsoleta. Essas mudanças são vistas como modificações de informações as quais, por esse motivo, devem ser excluídas do

banco de dados [Snodgrass/84]. Muitas aplicações de BD modernos devem manter informações não apenas do presente, mas também do passado e futuro, porque o conceito de tempo é indispensável. O BD que modela esse tipo de aplicação não pode ser baseado no conceito de um único estado porque informações atualmente consideradas obsoletas poderão ser referidas no futuro.

O sistema aqui apresentado aceita consultas do usuário, processa essa consulta, utilizando o conjunto de informação incompleta e temporal armazenada no banco de dados. Incluímos noções de incertezas e mecanismo para manipulá-las. Também, o conceito de dado incompleto é analisado neste trabalho, no contexto de informação com dimensão de tempo.

Este sistema pode ser usado para muitas aplicações envolvendo ou não a dimensão de tempo, preciso ou impreciso. Exemplos de tais aplicações incluem os sistemas de controle de inventário, previsão de economia, diagnóstico de dados de pacientes em clínica, em banco, registros médicos, controle acadêmico e outros. Planejamento é uma aplicação onde este sistema é muito útil, porque na previsão da economia, por exemplo, os valores de muitos parâmetros não são conhecidos.

Colocamos algumas metas para serem atingidas. São as seguintes:

1) O sistema deve ser generalizado para uma variedade de aplicações. Neste caso, pode envolver aplicações com ou sem tempo, precisas ou imprecisas.
2) A linguagem de consultas deve ser simples.

3) O sistema deve refletir mais as informações do mundo real.

4) Deve garantir que as respostas não sejam contraditórias com a realidade.

5) O tempo de resposta deve ser aceitável.

Neste trabalho, utilizamos a linguagem PROLOG para a implementação deste sistema, por considerá-la mais apropriada, no sentido de oferecer facilidade na implementação do analisador de consultas, além de ser uma linguagem mais próxima à natureza do trabalho no que tange a dedução de fatos no BD.

1.1 OBJETIVO DA PESQUISA

O objetivo principal deste trabalho é desenvolver uma ferramenta capaz de manipular dados incompletos (parciais) e informações temporais e responder consultas baseadas neles. Isto é, o sistema aceita consultas do usuário, processa essas consultas e as respostas, utilizando conjuntos de dados incompletos armazenados no banco de dados. Os aspectos considerados no projeto são valores nulos, marcados ou não marcados, fatos condicionais, valores nebulosos. Os aspectos temporais são pontos, intervalos e operadores temporais, precisos e imprecisos. Os problemas causados pelo dado incompleto são imensos, porque para representar informação, não basta apenas colocar ou representar essa lacuna com valor nulo. É necessário saber diferenciar e mostrar formas de manipulá-las.

Outro objetivo é atingir melhor o modelo do mundo real uma vez que o nosso conhecimento sobre as coisas sempre é parcial e impreciso. Então é necessário pesquisar mais sobre sistemas de informação e de banco de dados. Desta forma, estamos representando e modelando melhor o mundo real.

Na área de banco de dados, haverá de constituir-se em uma ferramenta muito útil na representação de valores nulos, dados parciais e informações temporais. Esses elementos mencionados são fatos que acontecem no dia a dia.

1.2 PESQUISA DOS TRABALHOS REALIZADOS

Este trabalho de Dados Incompletos e Informações Temporais foi influenciado por alguns trabalhos feitos anteriormente. Embora, esses trabalhos não tenham tratado destes assuntos em conjunto, foram tratados separadamente. Esta seção examina brevemente alguns destes trabalhos. Maiores detalhes podem ser encontrados ao longo deste trabalho.

O trabalho de Allen [Allen/83], identifica os relacionamentos entre objetos temporais. Também faz algumas aproximações diferentes sobre o problema de informação parcial e o uso de intervalo de referência. Lipski publicou uma teoria para processamento de dados incompletos, mas não principalmente interessado com modelo de BD relacional. Um trabalho publicado por Mariane Wilkins [Wilkins/86a] tinha concentrado mais na atualização de dados incompletos.

Em relação ao tempo, vários autores como Allen [Allen/83], Clifford e Warren [Clifford e Warren/83], escreveram sobre os problemas de incluir tempo no BD. Outros pesquisadores tinham usado outras maneiras de representar tempo.

A questão básica em relação ao tratamento de valores ausentes ("missing values"), foi inicialmente levantada por Codd [Codd/79], onde fez extensões nos operadores de álgebra relacional para manipular valores ausentes. Grant criticou este trabalho e distinguiu o problema de valores nulos. O trabalho inicial de Zadeh sobre conjuntos "fuzzy" também é estudado. Vários artigos escritos por Zadeh tratou outros problemas relativos, particularmente com conjuntos "fuzzy".

Como podemos ver, é crescente o interesse nesta área, embora nenhum trabalho conhecido abranja, ao mesmo tempo, o conceito de dados (dados incompletos e informações temporais). A linguagem de consulta temporal, TQUEL de Richard Snodgrass [Snodgrass/84] modificou a linguagem QUEL (Query Language) para permitir uma representação de tempo em BD. TQUEL é uma hiper conjunto de QUEL, linguagem de consultas ao sistema Ingres de gerenciamento de banco de dados relacional. Nenhuma facilidade foi oferecida para valores nulos e predicados de tempo. Pelo nosso conhecimento, a única pesquisa que abrange ambas as áreas é o trabalho de **Schiel [Schiel/85a]**.

1.3 SUMARIO DOS CAPITULOS

Capitulo 1: Apresentamos uma breve introdução ao trabalho como um todo e incluimos pesquisa dos trabalhos relacionados.

Capitulo 2: Aqui apresentamos os conceitos básicos sobre objetos espaciais e temporais. Também, valores nulos são identificados e estudados.

Capitulo 3: Inicialmente, descrevemos várias formas de representar dados parciais e informação temporal. O resto do capitulo trata dos problemas e solução na atualização de informações parciais.

Capitulo 4: Dedicado exclusivamente para descrever a recuperação de informação. Mencionamos os tipos de consultas e lógica quatrivalente no final do capitulo.

Capitulo 5: Discutimos em detalhe a estrutura do sistema implementado na linguagem PROLOG de acordo com os conceitos descritos nos capitulos anteriores.

Capitulo 6: Finalmente, apresentamos as conclusões do trabalho e discussões sobre futuros trabalhos e alguns assuntos relacionados.

2. CONCEITOS FUNDAMENTAIS

Uma maneira fácil, conveniente e natural de representar dados é numa organização tabular. Esta organização de banco de dados relacional é baseada no formalismo da teoria de conjuntos e relações. Este conceito foi introduzido por Codd [Codd/1970]. Toda relação é representada por uma tabela de duas dimensões. As colunas de cada tabela representam atributos de relação. Uma linha corresponde a uma tupla, e cada tupla representa um conjunto de valores de atributos. O modelo relacional é simples e estabelece uma independência de dados, permitindo o usuário a pensar mais em termos de conjuntos e relações do que em termos de arranjos e ponteiros, embora exista restrição por não ter capacidade de armazenar dados implicitos. Uma das propriedades fundamentais do modelo relacional de dados é que todas as entradas na tabela são elementos atômicos e não há duplicidade de tuplas.

A estrutura de uma relação é chamado INTENÇÃO e o conjunto de todas as tuplas armazenadas no banco de dados é chamada EXTENSÃO, segundo Codd [Codd/1979]. A extensão do BD reflete o estado corrente de conhecimento sobre o mundo e está sujeito a atualizações frequentes. Intenção e Extensão do BD podem ser ligadas de duas maneiras diferentes: Suposição do mundo fechado (Closed World Assumption) e suposição do mundo aberto (Open World Assumption). A primeira supõe que se um fato não pode ser derivado dentro das informações armazenadas no banco de dados, sua negação pode ser assumida. A segunda supõe que um fato é considerado falso, se e somente se, a negação for logicamente

derivada [Keller & Wilkins/85]. Este trabalho é baseado na segunda suposição porque permite resultados Verdadeiro (True), Falso (false) e desconhecido (Unknown). O resultado desconhecido ocorre quando nem o fato, nem sua negação, podem ser obtidos do banco de dados.

Diferenciamos dois conceitos principais na descrição de sistema de informação [Schiel/85b], são os objetos de espaço e os de tempo. Depois discutiremos como essas idéias são incorporadas no sistema.

2.1. OBJETOS ESPACIAIS

Os objetos básicos de espaço são ENTIDADES, RELACIONAMENTOS e EVENTOS. **Entidades** são objetos do mundo real que são consideradas como elementos de interesse no processo de modelagem. Exemplos de entidades são: uma casa, um livro, uma bola etc. **Relacionamento** é a ligação entre duas ou mais entidades onde cada entidade representa um certo papel. "Gostar" é um relacionamento entre João e Maria representado como: GOSTA (JOAO, MARIA). **Evento** é uma mudança ou uma ocorrência no mundo real. Exemplo de um evento é: Pedro trabalha para IBM desde 1980.

2.2. OBJETOS TEMPORAIS

Objetos de tempo, consistindo de pontos e intervalos, podem ser representados com valores absolutos de tempo ou predicados sobre esses valores de tempo. Também, é muito importante entender como eventos são relacionados entre si. O ponto principal nessa seção é diferenciar um ponto e um intervalo de tempo.

2.2.1 INTERVALOS.

Um intervalo consiste em um conjunto conexo de pontos de tempo. Pode-se descrever um intervalo com um par ordenado de um ponto de inicio e um de fim, onde o ponto de inicio é menor que o ponto de fim. Os pontos extremos de um intervalo I são chamados INICIO(I) e FIM(I). Esses pontos extremos de tempo, também podem ser considerados como intervalos. Se Inicio(I) é um intervalo, então I começa no **Inicio(Inicio(I))** e se Fim(I) é um intervalo, I termina no **Fim(Fim(I))** **[Schiel/85a]**. Neste projeto, o tamanho minimo de intervalo de tempo reconhecido é de um dia. Para reconhecer outros pequenos intervalos como horas, minutos e segundos, a mesma forma de aplicação poderia ser usada.

2.2.2 PONTOS.

Podemos definir ponto como um instante ou um momento no tempo. Esta menor unidade de tempo considerada, é chamada granularidade de tempo. Vê-se que esta menor unidade também é um intervalo de tempo e, com isso, os pontos absolutos não aparecem na prática. Um "ponto" passa a ser um intervalo pequeno, isto é, um intervalo sem subintervalos. O conjunto de todos os pontos de tempo é um conjunto totalmente ordenado, chamado EIXO DO TEMPO.

TIPOS DE INTERVALOS

A tabela abaixo mostra os tipos de intervalos.

TIPO	NOTAÇÃO
Intervalo aberto	( )
Intervalo fechado	[ ]
Intervalo aberto-fechado	( ]
Intervalo fechado-aberto	[ )

TABELA 2.1: TIPOS DE INTERVALOS

2.3. DADO INCOMPLETO

Dado incompleto é a falta de conhecimento sobre algo. Há ocasiões em que sabemos tudo ou nada sobre alguma informação. Também, há casos em que temos informação parcial e imprecisa sobre uma determinada informação. A falta de alguma parte dos dados é um problema muito comum a vários sistemas de banco de dados, como já foi mencionado no capitulo 1. Num banco de dados de uma empresa, por exemplo, a falta de informação sobre o endereço de um determinado empregado não deve impedir o sistema de incluir informações do empregado. Entretanto, incluindo tal empregado no BD implica que um valor nulo apropriado deve ser armazenado no lugar do endereço do empregado para representar melhor o dominio ausente.

Normalmente, com **dado completo**, uma consulta resulta numa resposta definida, verdadeira ou falsa no mundo que está sendo modelado. Uma consulta direcionada para um conjunto de dados completos do mundo real, produz resultado exato para a consulta. Numa informação parcial, as coisas se diferem, os resultados de uma consulta podem ser classificados de três maneiras: **verdadeiro**, **falso** e **possivel**. Então, podemos resumir que as respostas verdadeiras e falsas são resultados definidos e a terceira resposta como resultado indefinido.

Existem muitos problemas que aparecem com dados incompletos, tanto na parte de representação como na atualização de dados. Detalhes sobre isso vão ser apresentados nos próximos capitulos.

2.3.1. POR QUE TEMOS DADOS INCOMPLETOS?

Existem muitas situações, nas quais o BD não pode fornecer uma resposta precisa a algumas perguntas. Isto acontece porque há casos em que não conhecemos tudo ou nada sobre uma determinada informação. O não conhecimento ou a não disponibilidade de uma parte de dado é chamado dado parcial ou incompleto. Fontes de imprecisão, como foi colocado por Keller e Mariane [Keller e Mariane Wilkins/85], variam e incluem os seguintes casos:

i) Em muitas aplicações, o conhecimento, passa por uma continua evolução. No estágio inicial do uso de um novo BD, é muito dificil ter em mãos todas as informações necessárias. A informação adquirida pode ser imprecisa ou de natureza indefinida.

ii) Alguns atributos podem não ser aplicáveis a uma determinada tupla indicando que a estrutura do modelo de BD não corresponde exatamente a estrutura do mundo real.

iii) Alguma informação pode não estar disponivel porque é onerosa e de dificil obtenção.

iv) Podemos não querer armazenar uma determinada informação para assegurar a privacidade ou segurança da mesma.

v) Num BD distribuido, a responsabilidade de captar informação pode ser descentralizada. A visão do usuário pode omitir informação armazenada no BD.

vi) Outros casos, são exclusões de dados obsoletos e ocorrência de alguns erros.

2.3.2. TIPOS DE DADOS INCOMPLETOS

Distinguimos três tipos de dados incompletos que são: **dados negativos, dados imprecisos, e dados desconhecidos.**

2.3.2.1 Dado negativo (Negative data)

É uma informação negativa ou seja, uma informação sobre algum fato negado. Um dado negativo não é exatamente um dado incompleto, mas para facilitar o trabalho, foi classificado assim. Por exemplo, relações como:

Not(tem_salário(João)) --› significa que João não recebe salário.

Not(empregado (João, 1980, now)) --› Significa que João é desempregado desde 1980 até agora.

Todos os fatos que não são armazenados no BD são considerados como informação desconhecida. Por isso é que não adotamos suposição do mundo fechado.

2.3.2.2 Dados imprecisos (Imprecise)

São dados sobre os quais temos alguma informação mas não temos certeza sobre o valor exato ou a precisão. Por exemplo, se a informação sobre salário de João é conhecida mas não sabemos o valor exato, então isso pode ser considerado como informação imprecisa. Por exemplo:

Tem_salário(João, 10.000 ou 15.000) --› Significa que João ganha entre 10.000 e 15.000.

Tem_salário(João, ‹ 20.000) --› Significa que o salário de João é menor que 20.000.

Tem-salário(Pedro, Alto)---› Significa que o salário de Pedro é alto.

2.3.2.3. DADOS DESCONHECIDOS (VALORES NULOS).

Vários autores fizeram muitas tentativas no tratamento de valores nulos num BD. Por exemplo, Lipski define informação que possivelmente pode ser extraída de BD na presença de valores nulos [Lipski/79]. Imielinski e Lipski [Imielinski e Lipski/84], definiram sistemas capazes de manipular valores nulos quando o subconjunto de operadores relacionais é usado. Também o trabalho de Codd sobre este assunto pode ser considerado como um modelo teórico [Codd/79]. Valores nulos já foram estudados na [Ansi/x3/SPARC/1975] e foram classificados em 13 tipos mas, somente os de maior interesse serão mencionados aqui:

1) Valor não marcado (Unknown Data)

   Este pode ser considerado como valor ainda não conhecido mas definitivamente não nulo. Pelo nome, podemos entender facilmente o significado disso. O dado não-conhecido é denotado por asterisco (*).

   Por exemplo:

   Tem_salário(João,*) --› significa que João recebe algum salário mas não temos informação sobre o valor ou, por questões de segurança, não desejamos armazenar esse salário.

2) Valor marcado (marked value).

   É um valor que representa domínio do mesmo atributo, mas, para diferenciar estes atributos é necessário associar um número a cada atributo. Os valores marcados são assumidos iguais se tiverem a mesma marca.

Por exemplo:

Tem_salário(João,*1) e Tem_salário(Maria,*1)--›

significa que João e Maria recebem mesmo salário mas não temos informação sobre o valor.

2.4 INFORMAÇAO TEMPORAL

A necessidade de representar tempo é muito importante. Nos últimos 5 anos, aumentou o interesse na área de BD temporais. Muitos outros pesquisadores tinham discutido o tema "tempo no BD", introduzindo conceitos como eventos e processos [Allen/84]. As atividades desses pesquisadores podem ser classificadas em três etapas.

1) A formação da semântica do tempo no nivel conceitual.
2) O desenvolvimento do modelo para BD.
3) O projeto da linguagem de consulta temporal.

Um bom sistema de informação deve possibilitar a manipulação de dados obsoletos. Essas informações não devem ser simplesmente deletadas porque elimina a possibilidade de acessar informação do passado. Então, há uma necessidade imensa de representar informação temporal. Consideremos uma consulta tal que:

Qual estudante que terminou o curso de computação no ano passado?

Tendo o evento e o tempo em que o evento ocorreu no BD, ficaria fácil de responder esta pergunta ou até deduzir se realmente ocorreu o evento.

2.4.1 LØGICA TEMPORAL

O papel de tempo é um componente integrado de dados e está ganhando mais atenção. A lógica temporal pode ser usada para representar informação e para resolver problemas. Como as informações mudam, consequentemente, representação dessas informações precisam ser mudadas.

O conceito de BD temporal é introduzido formalmente. Esta aproximação é baseada na lógica temporal de Sernadas [Sernadas/80], que utiliza predicados de tempo para referir ocorrência de eventos no BD.

OPERADORES LØGICA DE TEMPO

Para melhor representação de tempo neste sistema, os operadores lógicos de tempo são utilizados. Os conjuntos de operadores lógicos de tempo que foram considerados neste trabalho são mostrados abaixo com seus significados:

before(I) --› Sempre antes Inicio(I).

bef(I) --› Alguma vez antes Inicio(I).

after(I) --› Sempre após Fim(I).

aft(I) --› Alguma vez após Fim(I).

during(I) --› Sempre durante I.

dur(I) --› Alguma vez durante I.

Now --› Agora

2.5 CONJUNTO "FUZZY"

Esta seção, trata de informação nebulosa ou fuzzy no processamento de consultas do sistema de banco de dados. Os mecanismos para manipulação são fornecidos mais adiante. Aqui, mencionamos somente a consulta que utiliza predicados fuzzy ou imprecisos.

Uma consulta fuzzy segundo Tahani [Tahani/76], é consulta que utiliza palavras "fuzzy". Exemplos de palavras fuzzy são muito utilizados no dia a dia. Maiores detalhes sobre conjunto "fuzzy" podem ser vistos no apêndice B deste trabalho. Exemplos de expressões fuzzy são:

a) Salário é **alto**;

b) Ano de conta do empregado é **recente**;

c) Salário é **muito maior que** 20.000,00;

d) Quantidade de peças em estoque é **pequena**;

e) etc.

A informação é naturalmente imprecisa ou nebulosa na representação de planejamento econômico, decisões, caracteristicas pessoais ou fisicas etc. A maioria dos problemas de recuperação de informação que ocorre na vida real são de natureza imprecisa. Na área de estatistica econômica por exemplo, valores numéricos de desempregos são disponiveis como 3,8%, 6,4%, 11,3%. A representação desses valores pode permitir que eles sejam representados como baixo, moderado e muito_alto respectivamente. Outro exemplo de uma consulta que consiste de predicado preciso como "idade igual 20" e predicado fuzzy como "idade igual jovem". Podemos notar que esses predicados linguisticos como **jovem** representa a mesma coisa que o valor

numérico. A diferença é que a primeira forma é menos informativa e menos precisa do que a segunda. Os elementos fuzzy podem ser colocados num sistema de recuperação na maneira do usuário e podem empregar alguns termos de fuzzy tais, como: **velho**, **jovem**, **recente**, **rico**, **bom**.

Um sistema de recuperação é baseado na teoria de conjunto fuzzy e variáveis linguísticas [Zadeh/65]. Termos "Fuzzy" (valores linguísticos) são definidos como o conjunto fuzzy no universo de discurso os quais contém valores numéricos de domínio da relação no sistema do BD. Os domínios para BD relacional "fuzzy" são de números discretos obtidos de um conjunto finito ou infinito. Os valores de domínio de uma determinada tupla pode ser de números incluindo os valores nulos.

NOME	DEPT	IDADE	SALARIO
CARLOS	DSC	Jovem	20.000
JOAO	DSC	40	Alto
PAULO	DEE	Velho	Muito_alto

TABELA 2.2.1 : RELAÇAO DE PROFESSORES COM FUZZY TERMOS

A tabela acima mostra a relação de professores na qual algumas palavras linguísticas (Termos Fuzzy) são usadas na construção da relação. Por exemplo, os termos **jovem** e **muito alto** respectivamente, representam valores dos domínios dos atributos **idade** e **salário**.

3. REPRESENTAÇÃO DE DADOS INCOMPLETOS E INFORMAÇÃO TEMPORAL NO SISTEMA RADINT

Um dos problemas mais cruciais na área de banco de dados é o de como representar a informação do mundo real. Aqui, vamos trabalhar no contexto de modelo relacional. Vimos no capitulo 2, as vantagens deste modelo. Então, vamos assumir que o modelo relacional é capaz de representar uma boa parte das informações do mundo real. Informações temporais são representadas com atributos adicionais e informação incompleta é representada com valores nulos, valores "fuzzy" e um atributo condicional.

3.1 REPRESENTAÇÃO DE DADOS INCOMPLETOS

Existem várias formas através das quais informações incompletas podem ser representadas. Neste trabalho são abordadas três formas de representá-las: Valores nulos, Atribuição condicional e valores indiretos. Foi observado no trabalho de Keller e Wilkins [Keller & Wilkins/85] que a representação permite mais de um dominio para alguns atributos do BD. Por exemplo:

NOME	ENDERECO	TELEFONE
Susan	{Apto7,Apto12}	555-0123
Pat	Apto7	555-9876
Sandy	Apto17	Unknown

TABELA 3.1: RELACAO CADASTRO-TELEFONICO.

Na tabela 3.1, Susan pode estar morando no apartamento 7 ou 12. Esta situação viola o conceito do modelo relacional que determina: "cada valor de atributo para cada tupla deve ser valor atômico". Em [Keller & Wilkins/85] são considerados vários mundos possiveis cada um com um BD relacional.

3.1.1 VALORES NULOS

Valor nulo já foi mencionado na seção 2.3.2.3. Na nossa representação, foi incluido um atributo-chave chamado "ID" para identificação das entidades e relacionamentos. Valores nulos são representados com valores especiais onde NOT é utilizado para valor negativo e asteriscos (*) para valores desconhecidos. Também, para valores marcados, podemos associar números para distinguir esses valores, por exemplo *1, *2, *3. Quando o número associado é o mesmo, então podemos dizer que os valores marcados são iguais.

ID	NOME	DEPT	IDADE	SALARIO
4	JOAO	*1	38	*
4	ANTONIO	*1	VELHO	30.000

TABELA 2.2.2 RELAÇAO DE PROFESSORES COM VALORES NULOS

Por exemplo, da tabela 2.2.2, pode ser extraida a informação segundo a qual João e Antônio trabalham no mesmo departamento.

3.1.2 ATRIBUIÇAO CONDICIONAL

Vários autores como Lipski [Lipski/79], utilizaram BD relacional com atributos de condição (status) para determinar estado de cada tupla. Também, Imielinski [Imielinski/85] introduziu tabela-v (V-table) e tabela de condição (conditional table) para representar dados incompletos.

Para nossa representação, mais um atributo é usado para representar os fatos negativos. Este atributo representa a situação (condição) de cada tupla. Também incluimos situações como OR, NOT e DADOS CONDICIONAIS. Observamos que, para algumas tuplas com a condição OR, da mesma identificação, as tuplas diferem só num atributo.

ID	NOME	DEPT	CONDICAO
5	MARIA	DEE	OR
5	MARIA	DSC	OR
6	JOSE	DEE	NOT

TABELA 2.2.3 RELAÇAO DE PROFESSORES COM ATRIBUTOS DE CONDIÇAO

Observe, a tabela acima representa:

5) Maria é lotada no departamento DEE ou DSC.

6) João não é lotado no departamento DEE.

3.1.3 VALORES INDIRETOS

Podemos representar valores incompletos com predicados ou funções em vez de valores. Esses predicados especificam informação precisa ou imprecisa sobre o dominio desejado. Por exemplo:

ID	NOME	DEPT	IDADE	SALARIO	CONDICAO
7	CARLOS	DEE	30	14.000	
8	MARIA	IRMAO(CARLOS)	40	15.000	

TABELA 2.2.4 RELAÇAO DOS PROFESSORES COM VALORES INDIRETOS

8) MARIA, com 40 anos, recebe 15 mil cruzados e é lotada no mesmo departamento que o irmão de CARLOS.

3.2. REPRESENTAÇAO DE INFORMAÇAO TEMPORAL

Existem pelo menos duas possibilidades para o desenvolvimento de um modelo para BD que envolve tempo:

1) Estender as semânticas do modelo relacional para incorporar tempo diretamente.
2) Incluir tempo como o dominio adicional.

Para converter um BD estático em BD dinâmico, o dominio temporal é acrescentado a cada relação. O valor deste dominio é especificado quando aquela tupla for válida. Neste trabalho, adotamos o segundo método, vamos ver primeiro os operadores temporais e logo em seguida ver como esses operadores podem ser representados.

3.2.1. OPERADORES TEMPORAIS

Existem vários operadores de tempo que são usados para o mesmo objetivo. O trabalho de Sernadas [Sernadas/80] define modelo temporal em que operadores temporais foram apresentados. Allen [Allen/84] classificou os trabalhos de tempo em três áreas:

1) Ordenação de pontos temporais.
2) Idéia de aspectos de relação onde foram utilizados os conceitos das partes de eventos tais como: inicio, meio e fim.
3) Conceito de duração como componentes de tempo ou distância entre tempos. Estes são representados por algumas palavras adverbiais como: de_para, durante e outros.

Dentro das três áreas acima, o conceito de ordenação é o mais desenvolvido onde os predicados como before(I), after(I), during(I) são usados.

Neste trabalho, são permitidos os seguintes operadores de tempo abaixo, que por sua vez são classificados:

1) **Constantes:**

a) Constantes de tempo fixo.

são números como 1987, 9/1981, 7/9/1981

b) Constante de tempo variável.

tempos como "now".

2) **Operadores de ordenação**

before(I), after(I)	valendo o intervalo todo.
bef(I), aft(I)	valendo alguns pontos quaisquer dentro do intervalo.

3) **Operador de duração**

during(I), dur(I).

3.2.2 REPRESENTAÇAO DE TEMPO.

A nossa representação de tempo concentra no tempo de ocorrência de fatos. Isto é, representar o tempo de vida de um determinado fato, sabendo-se que todos os fatos acontecem durante determinado intervalo de tempo. O método adotado aqui é o adicionamento de atributos de tempo. Cada relação contém um domínio adicional temporal especificando o intervalo durante o qual aquela tupla foi válida.

ID	NOME	DEPT	INICIO	FIM	CONDICAO
9	MIGUEL	DEE	1970	NOW	
10	JOANA	DSC	aft(1982)	1987	

TABELA 2.2.5 RELAÇAO DOS PROFESSORES COM PREDICADOS TEMPORAIS

Observe, a tabela acima representa:

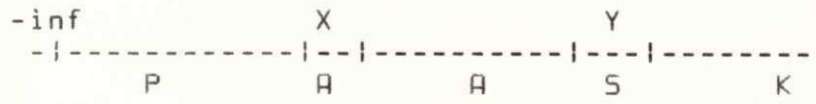
9) Miguel é lotado no departamento DEE desde 1970 até agora.

10) Joana é lotada no departamento DEE desde alguma data após 1982 até 1987.

3.2.3. TABELA DE PREDICADOS DE TEMPO

Para melhor avaliação de tempo, é necessário ter um mecanismo para definição dos intervalos dos predicados de tempo, já que existem muitas possibilidades na combinação dos predicados aplicados aos pontos extremos Inicio(I) e Fim(I) de um intervalo I. Sejam X e Y os pontos extremos de um intervalo, o intervalo formado pode ser fechado [X,Y], aberto (X,Y) ou semi-aberto (X,Y] ou [X,Y). Devido aos predicados de tempo como bef(X) e after(Y), devemos considerar uma linha infinita de tempo (-inf,+inf).

Então, cada intervalo I = ‹p(X), q(y)› em que p(X) e q(Y) são possiveis predicados aplicados aos pontos X,Y, determina diversos valores-verdade para toda a reta do tempo ‹-inf,+inf›. Por exemplo, tomando bef(X) e dur(Y) como pontos extremos (Inicio(I) e Fim(I)), graficamente podemos gerar os seguintes intervalos:



onde existem intervalos P, A, S e K. Para uma avaliação de uma consulta e o BD com os seguintes intervalos, as respostas devem ser possiveis nos intervalos P e S, yes e unknown nos intervalos de A e K. Observe que o intervalo K existe em alguns predicados de tempo.

OBSERVAÇÕES

A --› Sempre

S --› Alguma vez

P --› Possivel

K --› Desconhecido (*)

N --› Não

NOTAÇÕES

- inf = Menor extremo (Lower bound)
+ inf = Maior extremo (upper bound)
[ou] = Intervalo fechado
(ou) = Intervalo aberto

PONTOS EXTREMOS	constant Y	bef(Y)	aft(Y)	dur(Y)
constant X	A[X,Y]	A[X,X] S(X,Y)	A[X,Y+1] P(Y+1,+inf)	A[X,Y] S[Y,Y]
bef(X)	S(-inf,X) A[X,Y]	S(-inf,X) P[X,Y)	P(-inf,X-1) A[X-1,Y+1] P(Y,+inf)	P(-inf,X-1) A[X-1,Y) S[Y,Y]
aft(X)	P(X,Y) A[Y,Y]	S(X,Y)	S(X,+inf)	P(X,Y) S[Y,Y]
dur(X)	S[X,X] A(X,Y]	S[X,X] P(X,Y)	S[X,X]A(X,Y+1] P(Y+1,+inf)	S[X,X] A(X,Y) S(Y,Y)
before(X)	A(-inf,Y]	S(-inf,X) P[X,Y)	A(-inf,Y+1] P(Y+1,+inf)	A(-inf,Y) S[Y,Y]
after(X)	A(X,Y]	A(X,X+1] P(X+1,Y)	A(X,Y+1] P(Y+1,+inf)	P(X,Y) S[Y,Y]
during(X)	A[X,Y]	A(X,X) P(X,Y)	A(X,Y+1] P(Y+1+inf)	A[X,Y] S[Y,Y]
*	P(-inf,Y) A[Y,Y]	S(-inf,Y)	P(-inf,Y] S(Y,+inf)	P(-inf,Y) S[Y,Y]

PONTOS EXTREMOS	before(Y)	after(Y)	during(Y)	*
constante X	A[X,Y]	A[X,+inf)	A[X,Y]	A[X,X] P(X,+inf)
bef(X)	S(-inf,X) A[X,Y]	P(-inf,X-1) A[X-1,+inf)	S(-inf,X) A[X,Y]	S(-inf,X) P[X,+inf)
aft(X)	P(X,Y-1) A[Y-1,Y-1]	S(X,Y) A[Y,+inf)	P(X,Y) A[Y,Y)	S(X,+inf)
dur(X)	S[X,X] A(X,Y]	S[X,X] A(X,+inf)	S[X,X] A(X,Y]	S[X,X] P(X,+inf)
before(X)	A(-inf,Y]	A(-inf,inf)	A(-inf,Y]	A(-inf,X) P[X,+inf)
after(X)	A(X,Y)	A(X,+inf)	A(X,Y]	A[X+1,X+1] P(X+1,+inf)
during(X)	A[X,Y)	A(X,+inf)	A(X,Y]	A[X,X] P(X,+inf)
*	P(-inf,Y-1) A[Y-1,Y-1]	P(-inf,Y) A[Y,+inf)	P(-inf,Y] A[Y,Y]	S(-inf,+inf)

TABELA 3.3: TABELA DE PREDICADOS DE TEMPO

4. RECUPERAÇAO NO SISTEMA RADINT

A recuperação é uma forma de retirar informação no BD numa maneira mais fácil e sem desperdicio de tempo. A linguagem de definição e manipulação dos dados é uma interface básica entre o sistema e o usuário. É através dela que o usuário faz suas consultas e atualiza os dados, portanto, é um papel muito importante na implementação de um sistema de banco de dados.

4.1. CONSULTAS (QUERIES)

O usuário se comunica com o sistema por uma consulta. Quando uma consulta é feita, ela é processada e o resultado é fornecido, mas não é mostrado como o valor foi computado. Neste trabalho, procuramos simplificar a linguagem de consulta para facilitar a implementação do sistema, e, adotamos a forma tipica de uma consulta chamada a forma lógica canônica tais como:

R(a1, a2, ...,an).

R1(a1,a2, ...,an) and R2(a1,a2, ... an),

onde R é o nome da relação e os a_i s são geralmente valores de atributos e eles são colocados entre parênteses. Exemplos de consultas são mostrados abaixo:

Trabalha (Paulo,?, Professor, _ , <20.000).

Isto é: Paulo trabalhou como professor, com salário menor que 20.000 mil cruzados, não importa o departamento.

Empregado(João, Computação,_1) and trabalhou_como(João,_1, chefão).

Isto é: João formou-se em computação e trabalhou como chefão em alguma "Empresa". Observamos que o simbolo _1 representa um

domínio do mesmo atributo "Empresa", nas duas relações.

Os símbolos usados são:

1) O símbolo interrogação (?) --› para recuperar valor de um determinado atributo ou seja, o atributo desejado.

2) O símbolo sublinha (underscore) --› para representar um atributo anónimo (é um símbolo cujo valor corresponde a qualquer valor do atributo no BD).

OBS.: Quando o símbolo sublinha é acompanhado de um número, é para diferenciar o atributo anônimo; o número demonstra o atributo ao qual ele é ligado na outra relação.

3) O símbolo asterisco (*) representa valor desconhecido.

4) A palavra "AND" representa conjunção.

5) A palavra "OR" representa disjunção.

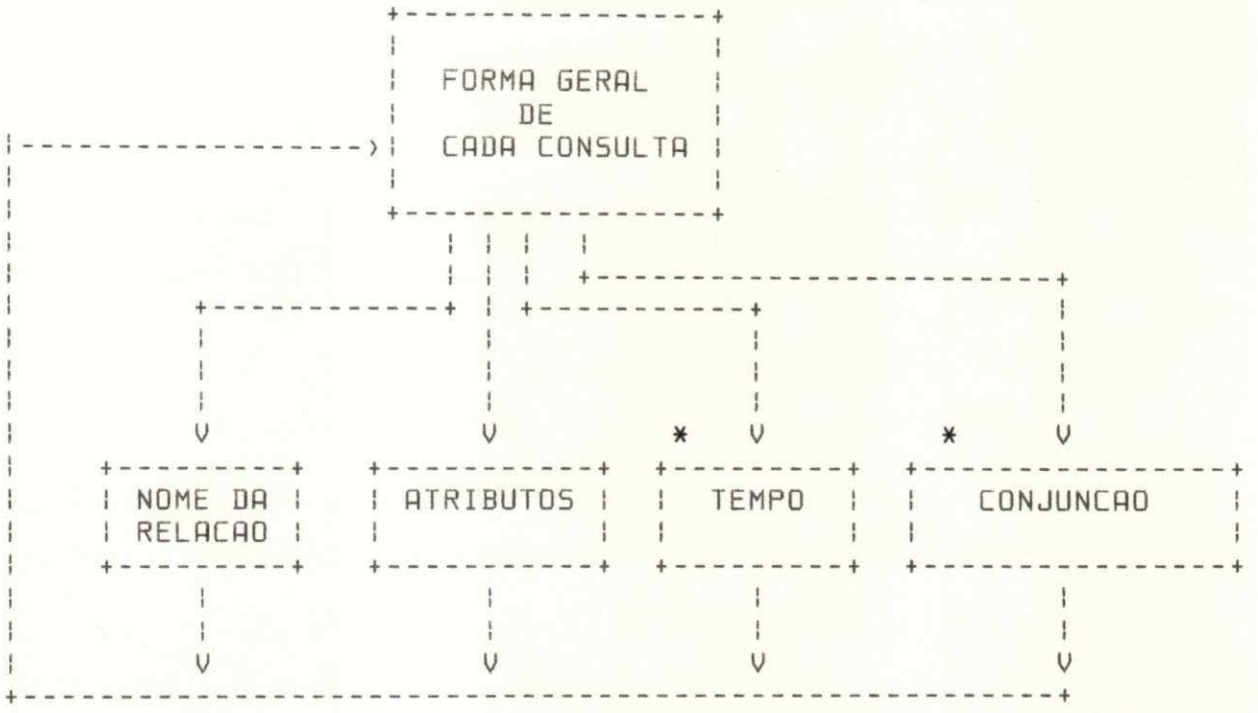
6) O ponto representa o fim da consulta.

7) O valor ‹20.000 significa que o salário é menor que 20.000.

Os demais símbolos como vírgula, ponto e vírgula servem como separadores.

Em geral, como vimos acima, uma consulta é uma expressão formada de símbolos, variáveis, constantes e etc. Algumas consultas são triviais e outras são mais complexas. Uma consulta simples consiste em uma relação e uma consulta complexa envolve várias relações utilizando operadores lógicos como "AND", "OR" e "NOT" para negar um termo.

FIGURA 4.2 : FORMA GERAL DE UMA CONSULTA



OBS.:

1) * significa que é opcional

2) As diferenças de consultas podem ser vistas melhor na gramática da linguagem na seção 5.2.3.

4.2 TIPOS DE CONSULTAS

Este sistema pode responder a dois tipos de consultas sobre os fatos que estão armazenados no banco de dados. Para melhor descrição desta seção incluimos duas relações tais: Relações de "Trabalho" e "gosta-de".

ID	NOME	DEPT	SALARIO	COND
1	CARLOS	DSC	100.000	OR
1	CARLOS	DEE	ALTO	OR
2	JOSE	DEE	‹90.000	_
8	JOAO	*	ALTO	NOT
4	TUNDE	DSC	100.000	NOT
5	MARIO	NPD	140.000	-
9	MARIO	DSC	20.000	-
6	MARCIO	NPD	ALTO	-
7	JOANA	DEE	›50.000	-

TABELA 4.1 RELAÇAO DE TRABALHO

ID	PESSOA1	PESSOA2	INICIO	FIM	COND
1	PAULO	BETA	1987	1988	-
2	PAULO	MARIA	1975	1988	-
3	JOE	ANA	1974	bef(1976)	OR
3	*	NEIDE	1974	bef(1976)	OR

TABELA 4.2: RELAÇAO GOSTA-DE.

i) **Consulta Booleana (lógica).**

Este tipo de consulta pesquisa se existe uma determinada informação no Banco de Dados. Essas consultas podem envolver algumas entidades ou relacionamentos entre essas entidades. Uma coisa importante é que a resposta só pode ser Yes, No, Possible e Unknown ou seja uma resposta lógica. Por exemplo:

Pergunta: Trab(MARIO, NPD, 140.000).

Resposta: Yes.

Pergunta: Trab(MARCIO, NPD, 200.000).

Resposta: Possible

ii) **Consulta de recuperação: (Retrieval)**

Neste tipo, a equivalência da expressão é procurada no BD para poder obter a informação solicitada. Uma interrogação é usada no lugar de dados que o usuário quer saber. Exemplos são:

Pergunta: Trab(Carlos,?,_).

Resposta: Possibly DSC or DEE

Pergunta: Gosta(Paulo,?;1976,1987).

Resposta: MARIA

4.3. LÓGICA QUATRIVALENTE

No passado, modelos de sistemas eram baseados na lógica de dois valores (two-valued logic). Os componentes de valores são restritos ao conjunto de verdadeiro, falso e a lógica assume que todos os fatos que não podem ser provados verdadeiramente são falsos. Uma vez que a estrutura relacional é escolhida, todas as consultas booleanas têm uma resposta definida de sim ou não.

Suponhamos que nós tenhamos os seguintes dados no BD:

Trabalha-para (Carlos,IBM)

Trabalha-para (João, COBRA).

Trabalha-para (Ednaldo, DIGIREDE).

Sabemos que no termo programação lógica, cada consulta é uma prova de uma teorema. Então, se as seguintes perguntas são feitas:

Q: Trabalha-para (João,COBRA)

R: Yes

Q: Trabalha-para (_, BRASCOM)

R: NO

Podemos notar que na última consulta, não há ninguém que trabalha para BRASCOM no BD então, o sistema responde "NO", mas a resposta poderia ser desconhecida (Unknown), desde que o computador não consiga provar a negação da consulta. Consequentemente, podemos ver que proposições lógicas no BD não podem sempre ser sim ou não mas também "desconhecido". Então, há necessidade de trabalhar com incertezas e Codd aplicou a lógica ternária onde os valores de sim, não e desconhecido correspondem 1, Ø e 1/2 respectivamente. Podemos referir ao primeiro conjunto como

Boolean e o segundo como **Kleenean**. Na tentativa de solucionar o problema de dado incompleto, a lógica de três valores ainda não se mostrou suficiente.

Neste trabalho, devido à nossa preocupação de melhorar a resposta, introduzimos o valor "POSSIVEL" adotando a suposição do mundo aberto, criando uma **LÓGICA QUATRIVALENTE**. Adotamos uma lógica quatrovalente na recuperação de informação no BD que pode conter valores nulos [Schiel/85a].

Para consultas compostas, os operadores lógicos como AND, OR e NOT são usadas. Somente a tabela de implicação que não foi implementada. Usamos as letras T, F, P e I para denotar os dados verdadeiros, falsos, possiveis e desconhecidos, respectivamente.

4.3.1. TABELAS DE LÓGICAS.

A Lógica Quatrovalente é baseada na seguintes tabelas de verdades mostradas abaixo:

P & Q	T	F	P	I
T	T	F	P	P
F	F	F	F	F
P	P	F	P	P
I	P	F	P	I

Tabela 4.4.1 : Tabela de "AND"

P V Q	T	F	P	I
T	T	T	T	T
F	T	F	P	I
P	T	P	P	P
I	T	I	P	I

Tabela 4.4.2 : Tabela de "OR"

P	T	F	P	I
not(P)	F	T	P	I

Tabela 4.4.3 : Tabela de "NOT"

P--›Q	T	F	P	I
T	F	F	P	I
F	T	T	T	T
P	T	F	P	I
I	T	T	T	T

Tabela 4.4.4 : Tabela de Implicação

Essas tabelas de lógicas são utilizadas para avaliar as consultas compostas. Para resumir, a tabela de implicação pode ser derivada de NOT (P) v Q. Outras informações sobre estas tabelas encontram-se em [Schiel/85a].

4.4. APOIO AO USUARIO

Em termos de programação lógica, no momento em que um sistema é simples e fácil, melhor será o acesso para o usuário. Este sistema oferece mais uma vantagem para facilitar o trabalho do usuário. Pois ele é totalmente interativo e o usuário pode trabalhar sem muita dificuldade.

Esta parte é composta de quatro facilidades: o primeiro é a senha. É o único mecanismo previsto a segurança do sistema. Isto afasta qualquer usuário que não tem permissão de utilizar o sistema. Também, afasta qualquer dano que poderia ser causado ao sistema. A segunda facilidade é a parte de introdução, que apresenta em detalhe o que o sistema faz e as facilidades oferecidas. A terceira facilidade é o gerenciador de arquivos. O sistema permite aplicação de muitos sistemas, então esta parte se encarrega de criar, carregar e gravar relações no banco de dados. Qualquer aplicação pode ser carregada na memória para ser trabalhada. Finalmente, o sistema mostra como selecionar as operações no sistema. Dependendo do estágio em que o usuário está no sistema, são mostradas as operações existentes e qualquer uma dessas operações poderiam ser selecionadas. Maiores detalhes sobre a implementação podem ser vistos no item 5.1 deste trabalho.

4.5. ATUALIZAÇAO

Muitos problemas podem ser criados numa simples atualização de uma determinada tupla em banco de dados. Nesta seção discutimos brevemente alguns problemas de atualizações envolvendo dados incompletos. Além dos problemas mais conhecidos como ambiguidades, duplicidades de tuplas ou dados mal definidos geram inconsistências e anomalias de atualização. Existem vários outros problemas mais complexos quando o banco de dados é relacionado com dados incompletos e valores nulos. Operações simples de atualização, tal como inserir uma tupla, pode criar muitos problemas no BD.

Neste trabalho, nós não entramos em muitos detalhes sobre o assunto de atualização com dados incompletos, porque é muito abrangente e precisa uma infra-estrutura mais inteligente. Somente a atualização ligada ao atributo de condição foi tratada. Deixamos este tratamento para trabalhos futuros. Um trabalho sobre este assunto pode ser encontrado no artigo de Wilkins [Mariane Wilkins/1986a]. Pela natureza do BD, na maioria dos casos, a mudança dos dados sempre existe, devido às mudanças nos eventos do mundo real. Nova informação fica disponivel e informação velha é referida. Neste caso, é necessário ter uma forma de atualizar o BD de tal maneira que a integridade de um BD seja mantida. Isto é, no momento em que um fato é afirmado, continua a existir até que seja mudado por outra atualização.

5. A ESTRUTURA DA IMPLEMENTAÇÃO DO SISTEMA RADINT

Este capitulo, descreve a estrutura da implementação do sistema desenvolvido. A função do processador é tornar consultas enunciadas na linguagem do sistema, analisar e fazer o processamento necessário para avaliar a consulta. O sistema foi dividido em módulos apropriados para facilitar a modificação do mesmo. Este sistema foi desenvolvido num computador IBM 4341 utilizando a linguagem PROLOG, e é executado sob o controle do sistema operacional VM/CMS. O compilador PROLOG é da versão 1.7, desenvolvido em Waterloo, Canadá.

Inicialmente, no desenvolvimento deste trabalho, foi previsto a utilização de um Sistema de Gerenciamento de Banco de Dados (SGBD), de preferência um BD relacional. Neste caso, aproveitariamos a implementação de alguns desses itens.

1) Organização de dados

2) Evitar implementação da parte de atualização.

3) Procurar um SGBD que inclui operadores com tempo.

Com esta idéia, faltará implementar um tradutor que seja capaz de traduzir as consultas dos usuários para comandos de manipulação de BD como linguagem de SEQUEL (SQL). Os resultados obtidos podem ser submetidos novamente e reanalisados para gerar resultados finais como Yes, No, Possible, Unknown. Também, precisariamos implementar um módulo para manipular com a parte temporal da consulta.

Posteriormente, foi tomada uma decisão de eliminá-lo e resolver este problema de outra forma, devido à natureza do trabalho, descobrimos que precisávamos de um software capaz de

incluir os operadores de tempo, manipular com valores nulos, conjunto fuzzy, ter possibilidade de responder consultas booleanas com respostas Yes, No, Possible e Unknown e outros. Também é necessário ter um tradutor eficiente que seja capaz de transformar consultas feitas pelo usuário numa forma canônica para consultas em linguagem de Sequel (SQL). Apesar da necessidade de implementar as rotinas mencionadas, vimos que a decisão tomada era a melhor forma possivel e uso do computador de grande porte, garante espaço suficiente e eficiência de tempo.

Os componentes principais deste sistema podem ser resumidos nos seguintes módulos:

1) Subsistema de apoio.
2) Analisador de Consultas (Scanner/Parser).
3) Subsistema de atualização.
4) Subsistema de Calendário.
5) Subsistema de Dedução.
6) Subsistema de Execução.

Um módulo de execução é responsável pela execução das consultas. A interação do usuário com o sistema é feita no terminal de video com a utilização de uma linguagem de entrada (linguagem de definição e manipulação de dados). A estrutura geral do sistema é mostrada na figura 5.1.

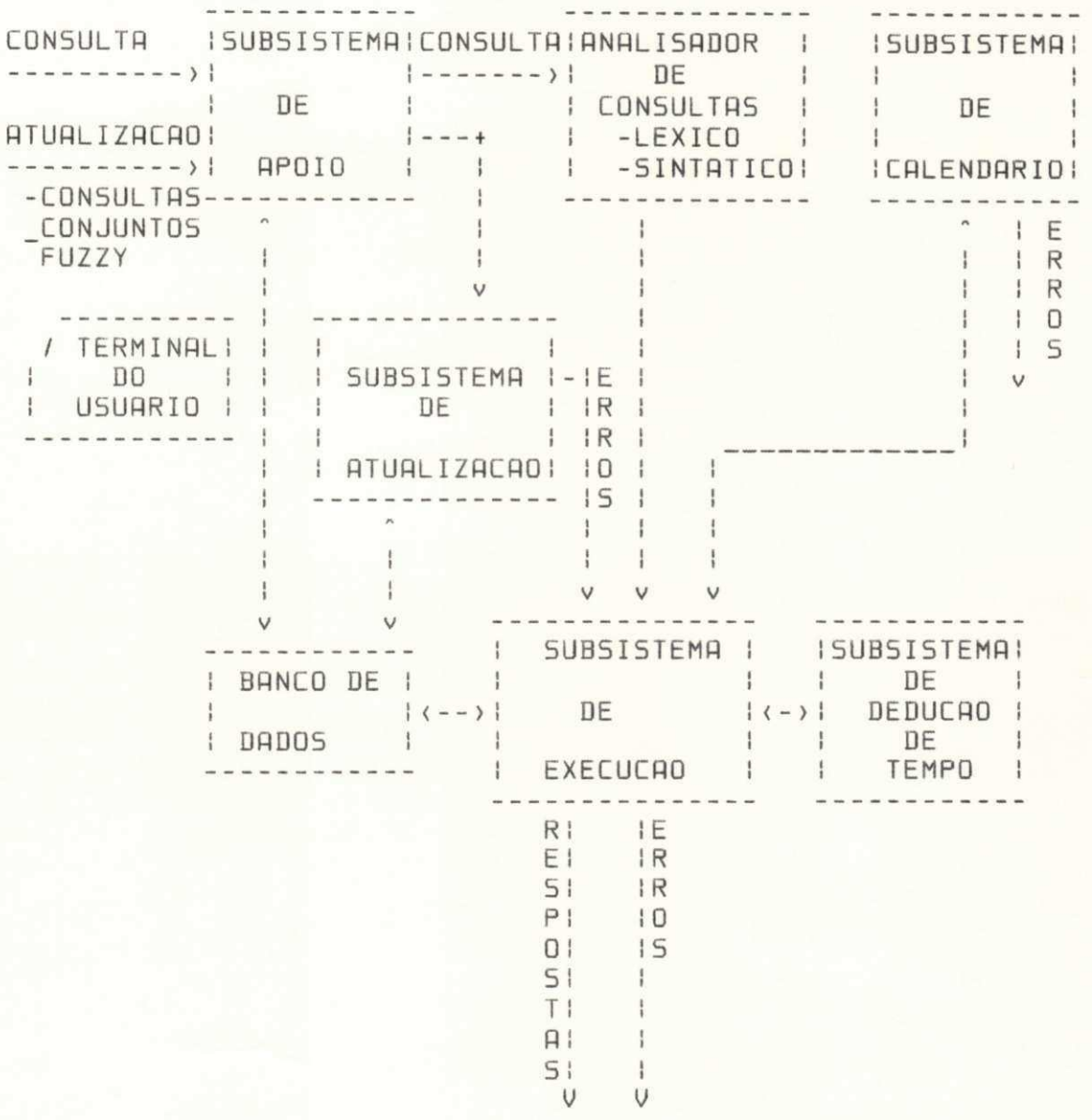


FIGURA 5.1 : ESTRUTURA GERAL DO SISTEMA RADINT

A seguir, apresentamos um resumo da função de cada subsistema:

Subsistema de apoio: Serve para apoiar o usuário na operação do sistema.

Atualização: Faz atualização de informação armazenada no BD.

Analisador de Consultas: É o compilador que analisa as consultas. Estão implementados o analisador léxico e sintático.

Subsistema de Calendário: Para cuidar da parte temporal do sistema.

Subsistema de Dedução de tempo: Para deduzir a parte temporal das consultas.

Subsistema de Execução: Faz a execução das consultas e faz dedução dos dados incompletos armazenados no banco de dados.

5.1. SUBSISTEMA DE APOIO

O subsistema de apoio, consiste basicamente da senha, apresentação do sistema, gerenciamento de arquivos e as operações principais disponíveis no sistema, como foi mencionado na seção 4.4.

A senha serve para verificação ou identificação dos usuários cadastrados ou aqueles que são capazes de utilizar o sistema. Esta proteção abrange tanto o impedimento ao acesso inadequado, como também, por motivo de segurança do sistema. Para garantir isso, o sistema inicia pedindo a senha do usuário, a qual ele é associado. Ao digitar a senha, o sistema verifica pesquisando se o usuário está ou não cadastrado. Em caso negativo, o sistema envia uma mensagem informando que a senha é inválida. Na confirmação da existência da senha, o usuário é considerado apto

a usá-la e o sistema prossegue. Abaixo, mostramos o formato da senha.

Formato da Senha: **‹Nome do usuário› _ ‹Código›**

A parte de apresentação deste subsistema mostra a introdução e um resumo do que o sistema faz e oferece. A parte do gerenciamento dos arquivos, orienta os usuários como criar arquivos para uma aplicação nova e também, mostra as aplicações existentes no sistema para serem trabalhadas e no final do trabalho, as relações e os dados relacionados a aplicação é gravada no banco de dados.

A última parte mostra as operações principais, tais como consulta (QUERY) e atualização. As operações foram colocadas como opções e qualquer uma pode ser escolhida, dependendo da operação a ser executada. A cada operação é associado um número para ser digitado na escolha da opção desejada. Dentro dessas operações muitas coisas são realizadas como atualização de agora (NOW). Logo em seguida isso vai ser explicado detalhadamente. Também, o comando "HELP" foi implementado para facilitar o trabalho do usuário.

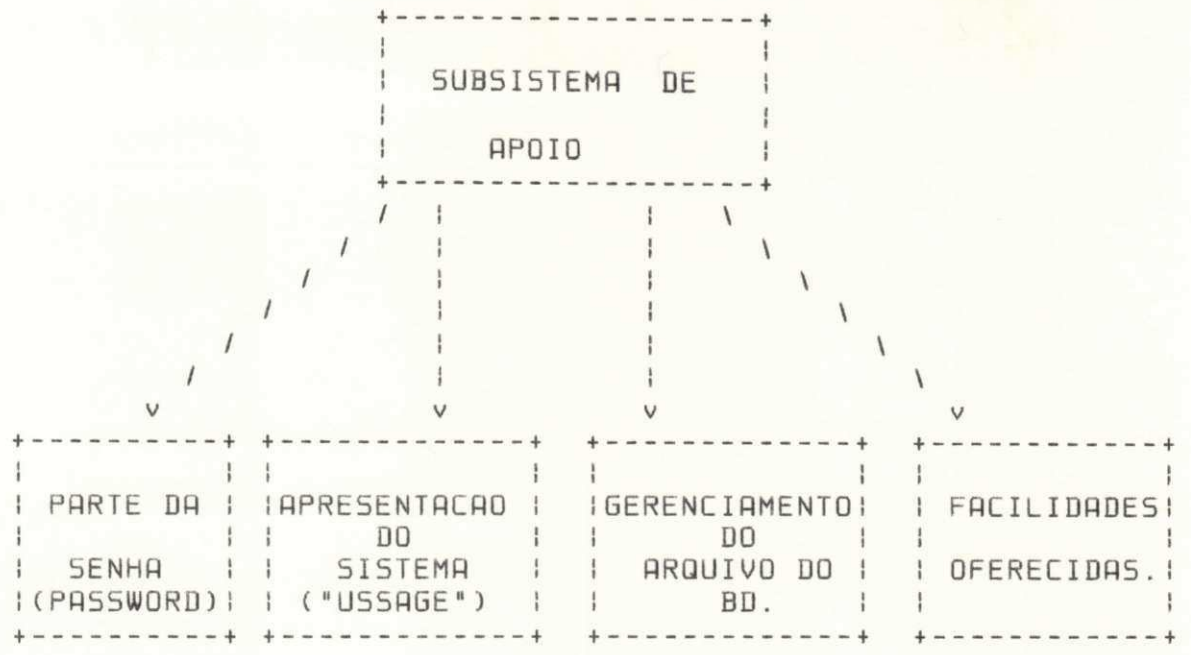


FIGURA 5.2 : ESTRUTURA DO SUBSISTEMA DE APOIO

Existem alguns valores em relação ao tempo que devem ser atualizados ou inicializados devido uma aplicação à outra. Por exemplo, a palavra "NOW" que representa o tempo corrente é sempre atualizada cada vez que queremos usar o sistema. Uma vez que esses valores são atualizados, todas as ocorrências do NOW assumem valor em qualquer parte do banco de dados.

5.2. ANALISADOR DE CONSULTAS

Um dos processos fundamentais em qualquer sistema de processamento é analisar e estabelecer uma estrutura gramática da linguagem de acesso de sistema. Para melhorar ou ter certeza da nossa entrada, as consultas enviadas ao sistema pelo usuário, deve obedecer um processo de checagem utilizando um analisador de consulta ou melhor chamado o COMPILADOR. A função de um analisador é verificar a sintaxe e semântica da consulta recebida. O analisador de consulta consiste em duas partes:

a) Analisador léxico

b) Analisador sintático.

Na verificação de uma consulta de entrada, se ocorre contradições entre os simbolos, os erros léxicos e sintáticos são detectados e o usuário é informado.

O uso da linguagem PROLOG, facilita a implementação deste analisador de consulta. Até que isto é uma grande vantagem por causa do subconjunto da lógica usado no PROLOG conhecido como as CLAUSULAS DE HORN que já foi provada da maior utilidade. Cláusulas de HORN é uma cláusula que têm no minimo uma literal não negada. Inclusive, foi provado que é mais eficiente do que a linguagem LISP na implementação de compilador. Maiores detalhes sobre o analisador léxico e o analisador sintático serão dados nas seções seguintes.

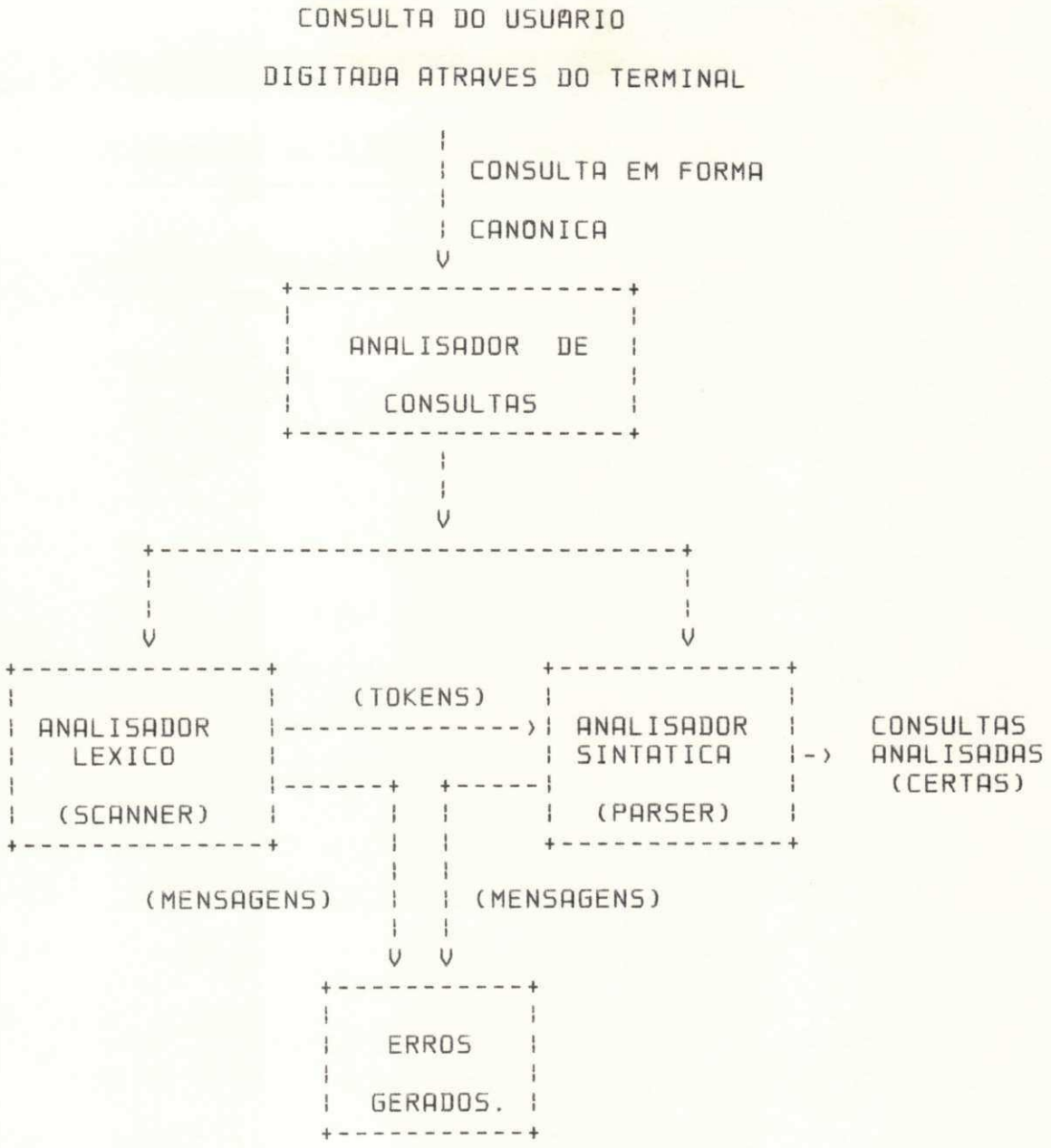


FIGURA 5.3 : ESTRUTURA DO ANALISADOR DE CONSULTAS

5.2.1. ANALISADOR LEXICO (SCANNER)

A função básica do analisador léxico é aceitar os simbolos de entrada, separar os simbolos da linguagem de acordo com grupos logicamente definidos e fornece ao analisador sintático como uma sequência de tokens. Esses tokens podem ser classificados em identificadores, inteiros, delimitadores e outros.

O analisador léxico recebe uma consulta digitada no terminal pelo usuário. Esta consulta é separada numa lista de tokens, baseado na formação dos elementos básicos da linguagem. Este processo de separação é uma aplicação clássica da teoria de autômato de estados finitos. Na realidade, este autômato de estados faz o reconhecimento de simbolos da consulta atráves de uma transição de um estado para outro.

O conjunto de caracteres são formados de combinações que formam um resultado especial simplificado nos elementos básicos da consulta. Cada token reconhecido é associado com uma letra, significando o tipo do simbolo, por exemplo: Identificador representado por I. Os terminais de itens léxicos e as letras correspondem às seguintes:

I) Identificador (I): Um conjunto de letras e digitos que não sejam palavras reservadas, sendo que o primeiro caracter deve ser uma letra. Os identificadores são utilizados basicamente para definir nome da relação, atributos e outros. Não foi considerado um tamanho máximo para os identificadores.

II) Palavras reservadas (P): São palavras que possuem um significado especial dentro do contexto da consulta.

III) Número: Os números são inteiros(T), reais(R), com ou sem

sinal.

IV) Simbolos especiais(S): São os delimitadores, operadores relacionais e outros como: * ? : () ; .) (= , --

V) Operador lógico: usado para unir expressões relacionais como AND, OR, NOT.

Quando um identificador é reconhecido, a lista de palavras reservadas é checada para ver se corresponde a um elemento da lista. Se a consulta for correta, este listão de tokens é passado para o analisador sintático. Caso contrário, o usuário notifica o tipo de erro cometido na consulta atráves de uma mensagem de erro. Em resumo, os identificadores acima têm as seguintes gramáticas:

⟨IDENT⟩:: = ⟨LETRA⟩|⟨LETRA⟩ ⟨INTEIRO⟩

⟨LETRA⟩:: = A|B|C|....|Z.

⟨INTEIRO⟩:: = 0|1|2|....|9.

Observemos que referimos os terminais como DIGITO, INTEIRO. Os terminais LETRA e DIGITO já são implementados como função embutida na linguagem de PROLOG.

5.2.2. ANALISADOR SINTATICO (PARSER)

A função básica do reconhecedor é verificar a validade sintática da consulta, ou seja, agrupar os tokens numa estrutura sintática. Esta construção sintática é possivel através das regras de gramática. As produções são de uma gramática livre de contexto.

A técnica de reconhecimento utilizada é bem simples. Cada consulta passa pelo analisador léxico. Só depois disso é que a lista de tokens é passada para o analisador sintático. Este método é usado por causa do tamanho razoável de cada consulta. Quando ocorre erros, neste módulo,o tipo de erro ocorrido. A sintaxe da linguagem de consulta é descrita nas regras de gramática mencionadas o usuário é somente notificado do tipo de erro ocorrido. A sintaxe da linguagem de consulta é descrita nas regras de gramática mencionadas anteriormente.

5.2.3. REGRAS DA GRAMATICA

As regras da gramática é do tipo da gramática livre de contexto e são representadas em notação de BNF ("Backus Naur Form"). Uma gramática para uma linguagem é um conjunto de regras de produções que especificam as sequências de palavras aceitáveis naquela linguagem. Da mesma forma, as regras abaixo representam as sequências de palavras e simbolos aceitáveis como uma consulta.

```
<CONSULTA> ::=  <LISTA_CONSULTA>.

<LISTA_CONSULTA> ::=  <CONSLT> <CONT_CONSLT>
                    | (<LISTA_CONSULTA>)<CONT_CONSULT>

<CONT_CONSLT> ::=  <OPR_LOGICO> <OPR_CONJ_OPT> <LISTA_CONSULTA>
                 | E

<CONSLT> ::=  IDENT (<LISTA_ATRIB> <TEMPO_CONSLT>)

<LISTA_ATRIB> ::=  <ATRIB> <CONT_ATRIB>

<CONT_ATRIB> ::=  ,<LISTA_ATRIB>
                | E

<ATRIB> ::=  <OPR_REL> <CONST>
           | _ <INTEIRO_OPC>

<ATRIB> ::=  ? <INTEIRO_OPC> <FUNC_EMB_OPT>

<TEMPO_CONSLT> ::=  ; <TEMPO> <CONT_TEMPO>
                 | E

<CONT_TEMPO> ::=  , <TEMPO>
                | E

<TEMPO> ::= <VALOR_DATA>
          | _ <INTEIRO_OPC>
          | ? <INTEIRO_OPC>
          | <TEMPO_SIMPLES> (<VALOR_DATA>)
          | <TEMPO_DUPLO> (<VALOR_DATA>,<VALOR_DATA>)
          | E

<TEMPO_SIMPLES> ::= AFTER|BEFORE|BEF|AFT|DUR

<TEMPO_DUPLO> ::=  DUR|DURING

<VALOR_DATA> ::=  NOW
                | INTEIRO <CONT=DATA>
```

‹CONT_DATA› ::= / INTEIRO ‹MAIS_CONT_DATA›

| E

MAIS_CONT_DATA ::= / INTEIRO

| E

‹CONST› ::= INTEIRO |REAL|IDENT|‹VALOR_DATA›

‹INTEIRO_OPC› ::= INTEIRO

| E

‹OPR_LOGICO› ::= and | or | not

‹OPR_CONJ_OPT› ::= MINUS|UNION|INTERSECT |E

‹FUN_EMB_OPT› ::= ‹FUNÇAO | E

‹FUNÇAO› ::= SUM|COUNT|MAX|MIN|AVG

‹OPR_REL› ::= ‹ | › | ‹ = | › = | ‹ › | E

OBSERVAÇOES:

E --› representa o conjunto vazio

| --› significa ou.

[ ] --› significa uma ou mais vezes.

Tipos de Simbolos da Gramática

VN (Variáveis não terminais) ---› aparecem mais à esquerda das produções.

VT (Variáveis Terminais) ---› aparecem somente à direita das produções.

5.2.4 DIAGNÓSTICO DE ERROS

O diagnóstico de erros é um processo muito complexo, envolvendo erro semântico na maioria dos casos. Quando um erro é detectado, a ação pode ser por exemplo, desconsiderar os simbolos de entrada anteriomente utilizados, na tentativa de reconhecimento de uma regra.

Neste trabalho, os erros são classificados em três formas; Erros léxicos, Sintáticos e Semânticos. Os erros léxicos são aqueles decorrentes da codificação incorreta de um simbolo terminal ou do uso de caracteres inválidos da linguagem. Os erros sintáticos acontecem quando as regras gramaticais da linguagem não são obedecidas na forma em que elas foram implementadas no programa. Os erros semânticos são erros dos usuários quando não obedecem as definições semânticas da linguagem. Analisamos abaixo em detalhe, as formas com as quais os três erros foram implementados.

a) **Erro Léxico**:

Ao detectar um erro na tentativa de reconhecer um comando, uma mensagem de erro é emitida. Por exemplo, encontrando um caracter que não faz parte do alfabeto da linguagem, então a mensagem: CARACTER INVALIDO é emitida.

b) **Erro Sintático**:

Na implementação das regras de gramáticas, foi colocado um "flag". Quando um terminal falha, se o flag= "S", essa falha caracteriza um erro sintático, pois o usuário é notificado, através de uma mensagem de erro.

c) **Erro Semântico:**

Na execução de consultas, existe a possibilidade de aparecer erros semânticos. A consulta incluindo relação que não foi cadastrada no sistema ou atributo que não confere com atributo do sistema são exemplos de erros semânticos. O sistema detecta esses erros comparando a consulta do usuário com a tabela que contém todos os números das relações e atributos do sistema.

Quando ocorre um erro, a execução é interrompida e a mensagem de erro é mostrada ao usuário. Os erros semânticos são detectados no próprio módulo de analisador de consultas.

5.3. SUBSISTEMA DE ATUALIZAÇAO

Nesta seção, discutiremos brevemente os comandos de manipulação de dados e implementação dos mesmos. Nós não vamos aprofundar muito sobre este assunto pois, tratamos a parte principal ligada a este trabalho.

Os comandos de manipulação de dados servem como intermediário entre o usuário e o sistema. Cada comando opera independente dos demais. A cada comando é associado uma opção ao qual o usuário pode escolher. A opção escolhida corresponde a operação a ser realizada. Os comandos considerados são MOSTRA RELAÇAO, MOSTRA TUPLA, INSERE, DELETA, UPDATE (Atualização). As operações são feitas ao nivel de tupla. Também, atualização de conjunto de fuzzy é feita utilizando os comandos como ALTERAR, RETIRAR que são semelhantes aos comandos mencionados anteriormente. O formato dos comandos é igual ao mostrado abaixo:

‹Comando de Manipulação› (‹Nome da Relação›‹Nome do sistema›‹Lista de atribuição›)

Observemos que para cada operação, maiores cuidados devem ser tomados para o sistema não deletar, inserir ou atualizar tuplas que não existir na relação. Novas relações não podem ser criadas enquanto existir a mesma relação já criada.

Os algoritmos de implementação dos diversos comandos são semelhantes, por isso mostramos somente o algoritmo do comando INSERÇAO, utilizando pseudocódigo.

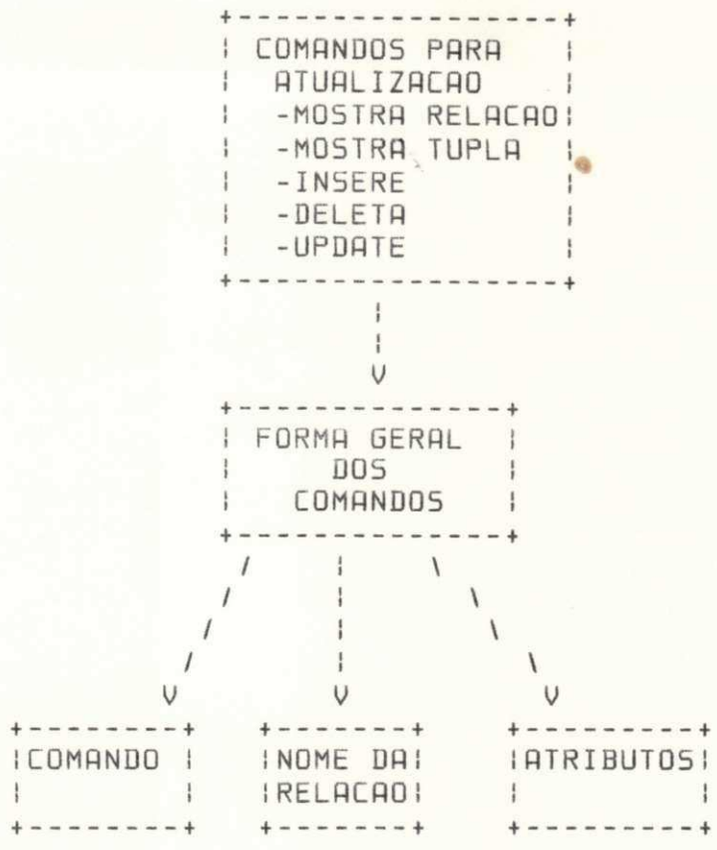


FIGURA 5.4 : COMANDOS DE ATUALIZAÇÃO

Algoritmo de comando INSERE

```
INICIO
  INSERE tupla
  REPETIR-ENQUANTO existiam mais operações de Inserção
         SE a relação existe ENTÃO
            Dê mensagem "TABELA JÁ EXISTE"
            Inclua atributos ID e COND na relação
            Chama rotina le-valor-atributos
         CASO CONTRÁRIO
            Dê mensagem "TABELA NOVA"
            Leia os atributos da relação
            Inclua atributos ID e COND na relação
            Grava os atributos na tabela de relação
            Chama rotina le-valor-atributos
         FIM
   FIM-REPETIR
FIM
```

Para facilitar a manipulação destes comandos, foram criadas duas tabelas chamadas TABELA e RELAÇÃO onde estão guardados os

nomes de atributos de cada relação e os dominios das tuplas respectivamente. Neste caso, quando uma relação é nova, os nomes dos atributos são guardados na TABELA para facilitar a checagem de relações existentes e para posterior manipulação dos mesmos. Também, os dominios de atributos são armazenados e gravados na tabela chamada RELAÇAO.

Abaixo, mostramos como inserir uma operação de inserção:

```
          ›inserção
            NOME DA RELAÇAO? /* Pede nome da relação */
- Relação-A
            RELAÇAO NOVA     /* Mensagem do Sistema */
- Inserção                   /* Inserção dos Atributos */
- Nome
- Idade
- Sexo
- Fim
            TABELA JA EXISTE
           Inserção:         /* Inserção dos Dominios */
          ID
- 4
          NOME:
- João
          IDADE:
- 22
          SEXO:
- Masculino
          COND:
           OR
```

Observemos que essas informações a serem inseridas são gravadas automaticamente nas tabelas chamadas RELAÇAO e TABELA respectivamente, como já foi mencionado anteriormente.

5.4. SUBSISTEMA DE CALENDARIO

Nesta seção mostraremos como uma consulta que envolve tempo, poderá ser processada. Devido a parte temporal, será um módulo que será responsável pela minipulação do tempo. Este módulo foi dividido em três subsistemas onde cada módulo tem uma função básica:

- Verificação de data.
- Transformação de predicados de tempo numa "Forma Interna" de representação. Uma forma interna de data é um processo de transformação de predicado de tempo numa forma executável.
- Retransformação de data em forma interna num predicado de tempo.

(I) **Verificação de data**:

Além de utilizar o analisador de consulta para fazer análise sintática e semântica de consultas, é necessário verificar se a data digitada pelo usuário é válida. Então, nesta parte do programa faz esta verificação. As criticas que podem ser feitas em cima das datas são:

- Verificar se os dias correspondem ao total de dias de cada mês.
- Verificar se o mês está entre 1 (um) e 12 (doze).
- Verificar se o ano não é negativo.

Para maior aplicação deste projeto, serão permitidas várias formas de representação de data:

DATA (D, M, A)

DATA (M, A)

DATA (A)

Onde D, M, A, significam dia, mês e ano respectivamente.

Exemplos das formas de datas permitidas são:

1987

Ø7/1987

12/12/1987.

(II) **Transformação de predicados de tempo**:

Foram implementados os predicados de tempo, as constantes e os predicados before, after, now, during. É necessário transformar esses predicados de tempo para uma forma interna para poder manipulá-los. Esta parte foi implementada com a rotina chamada TRANSF que consiste em dois parâmetros.

TRANSF(Predicado de tempo, Forma interna correspondente).

Iniciando a transformação, foram adotados alguns simbolos para diferenciar e representar simbolicamente o significado dos predicados de tempo, ou seja, se um intervalo deve ser considerado inteiramente ou se somente é válido alguns tempos dentro de intervalo. As notações usadas são **I**, **U** ou **K**, que servem como prefixos para a parte do tempo.

I --›representa o intervalo inteiro.

U --›representa alguns pontos dentro do intervalo.

K --›representa um intervalo não conhecido.

Cada predicado de tempo é transformado num formato de dado onde é indicado o ponto inicial e final de cada predicado de tempo:

I ‹ponto inicial› ‹ponto final›

Como os predicados devem ser transformados numa forma que

reflete o significado real do mesmo, então foram adotados valores para designar ponto inicial e final para alguns predicados. Por exemplo:

bef(1982) ------> U 01/01/0001 31/12/1981

Para um predicado de before, utilizamos a data 01/01/0001 como ponto inicial.

after(1982) ----> I 01/01/1983 31/12/3000

Para um predicado de after, utilizamos a data 31/12/3000 como o ponto final.

Exemplos de algumas transformações de predicados de tempo são mostradas abaixo:

Predicados de Tempo	Formas Internas de data.
Constantes como 07/1987	I 01/07/1987 - 31/07/1987
NOW	I 22/03/1987 - 22/03/1987
bef(1982)	U 01/01/0001 - 31/12/1981
before(1982)	I 01/01/0001 - 31/12/1981
before(NOW)	I 01/01/0001 - 22/03/1988
aft(10/1982)	U 02/10/1982 - 31/12/3000
after(10/1982)	I 02/10/1982 - 31/12/3000
after(NOW)	I 23/03/1988 - 31/12/3000
dur(4/10/1982)	U 04/10/1982 - 04/10/1982
during(4/10/1982)	I 04/10/1982 - 04/12/1982
during(4/10/1982,10/1982)	I 04/10/1982 - 31/10/1982
*	K 01/01/0001 - 31/12/3000

TABELA 5.5 : TABELA DE TRANSFORMAÇÕES DE PREDICADOS

Observamos que o predicado de NOW é transformado num tempo colocado no inicio do programa. Também, observamos que quando o predicado for before (antes), after (depois) então um dia é diminuído ou um dia é adicionado à data que acompanha o predicado respectivamente.

(III) **Retransformação de Predicados de tempo**

Esta parte transforma as datas em formas internas numa forma de saída ou seja, faz o contrário da tabela acima. Como o tempo é armazenado na forma interna, precisa ser impressa numa forma clara para usuário. A rotina de implementação é chamada:

RE-TRANSF(Forma Interna, Predicado de tempo).

5.5. SUBSISTEMA DE DEDUÇAO DE TEMPO

O subsistema de dedução faz a dedução de tempo. Como sabemos, para obtermos respostas corretas para consultas, é necessário avaliar os fatos armazenados ou conhecidos pelo sistema. Normalmente, a resposta esperada de uma consulta é sim ou não. Além do mais, uma resposta pode fornecer o número de dados encontrados ou informar que tipo de informação que não é disponivel no BD. Neste sistema, como ele se trata de informação incompleta, as respostas incluem respostas de incerteza como Possible, Unknown e outras. As respostas para possivel ou "maybe" só podem ocorrer quando há incertezas. Para o nosso conhecimento, nenhum algoritmo conhecido poderia resolver isso, foi então necessário adotar a nossa forma.

Neste sistema, esta seção cuida da parte de tempo. Será necessário desenvolver uea rotina que é capaz de receber os intervalos e comparar com o tempo armazenado no BD. Os processos de dois tipos de consultas são semelhantes. A única diferença é que a consulta booleana retorna uma resposta booleana enquanto a consulta de recuperação só faz a comparação e vê se a rotina de predicado falhou ou não. As rotinas são mostradas a seguir.

A rotina de tempo para consulta booleana é:

TEMPO_CONS_BOOL(X, Y, A, B, R)

onde (X, Y) e (A, B) são dois intervalos, o primeiro é o da consulta e o segundo é o do BD.

R retorna a resposta booleana.

A rotina de tempo para consulta de recuperação é:

TEMPO_CONS_RECU (X, Y, A, B)

onde (X, Y) e (A, B) são idênticos ao anterior.

A operação com tempo é feita comparando a consulta que envolve tempo com os fatos armazenados no BD. Antes desse processo de comparação, para cada par de intervalos, são gerados os intervalos válidos, como já foi mostrado na seção 3.2.3. Consideramos todas as possibilidades que podem surgir na combinação de predicados de tempo. Nas comparações, foram testados os seguintes casos:

1) Testar se um intervalo é igual a outro.

2) Testar se um intervalo está contido no outro.

3) Testar se um intervalo intercala o outro.

4) Testar se um intervalo difere do outro.

A tabela 5.6 mostra um resumo dos testes.

Dado dois pares de pontos (X,Y) e (A,B) que representam os pontos extremos de dois intervalos. Os intervalos representam intervalo de tempo de uma consulta e de um Banco de Dados respectivamente. Eles são:

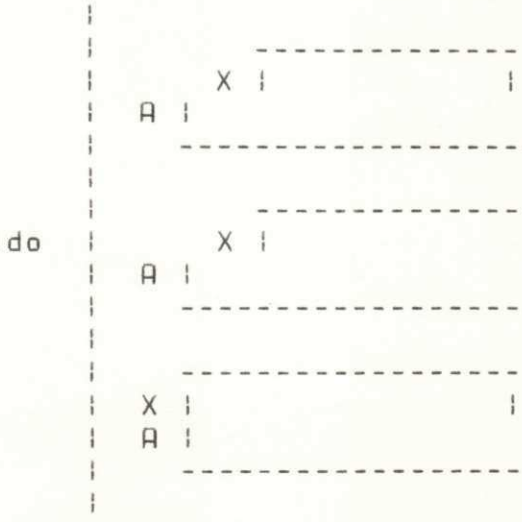
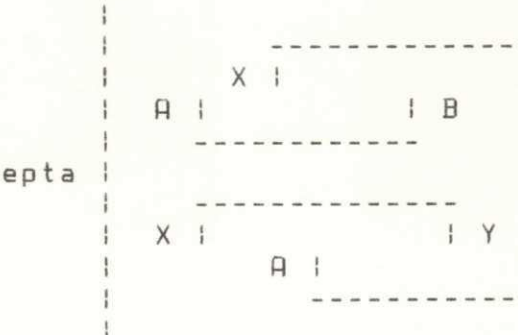
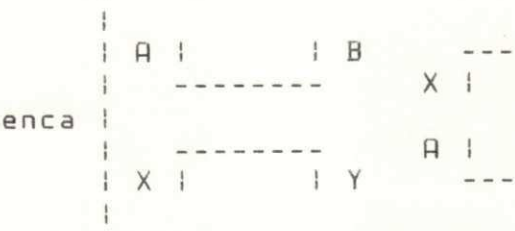
TESTES	COMPARAÇAO DE INTERVALOS	DESCRIÇAO
Igualdade		X = A e Y = B
Contido		X > A e Y < B X > A e Y = B X = A e Y < B
Intercepta		X > A e Y > B e B > X X < A e Y < B e Y > A
Diferenca		B < X Y < A

TABELA 5.6: COMPARAÇAO DE INTERVALOS

Na seção 5.4 as formas Internas de representações de predicado de tempo têm símbolos como I, U e K. Estes símbolos representam como sufixo para distinguir cada predicado de tempo.

Nos testes mostrados na tabela 5.6 se os sufixos dos predicados de tempo fossem iguais, não teria nenhum problema. No caso de sufixos diferentes, tabelas para decidir o que deverá ser feito.

IGUAL	I	U	K
I	yes	Possible	Unknown
U	yes	yes	Unknown
K	yes	Possible	Unknown

CONTIDO	I	U	K
I	yes	Possible	Unknown
U	yes	yes	Unknown

TABELA 5.7: TABELA DE SUFIXOS DE PREDICADOS DE TEMPO

Nos testes de sufixos de operadores de tempo, para casos de interseção (overlap) e diferença, são assumidos respostas Possible e Unknown respectivamente.

5.5.1 OPERAÇAO DO CONJUNTO FUZZY

Nesta seção mostramos como é avaliado os conjunto fuzzy BD para acrescentar as condições de incertezas. Como já trabalhamos com dados incompletos, portanto, achamos que os elementos do conjunto fuzzy podem ser incluídos.

A teoria do conjunto fuzzy do Zadeh é uma área muito abrangente. Normalmente, para os valores fuzzy não podemos associar valores diretamente, termos como alto, baixo etc. porque envolve alguns cálculos apropriados [Zadeh/75]. Os termos de fuzzy na consulta são substituídos por um intervalo de valores associado. Depois, uma operação de mapeamento é aplicada para comparar o valor substituído com o valor correspondente no BD.

Para os conjuntos de valores nebulosos como ‹500 e ›100, as mesmas comparações da tabela 5.6 são feitas. Também, aplicamos as mesmas respostas.

5.6 SUBSISTEMA DE EXECUÇAO

Nesta seção, apresentamos a descrição da execução da consulta do usuário e discutimos como alguns elementos da dados incompletos são executados. Devemos nos convencer de que a execução de consulta com informações completas não é trivial.

Após a verificação de uma consulta pelo Analisador de consulta (léxico e sintático), a mesma é passada para o módulo de execução se for correta. Inicialmente, um tratamento preliminar é feito:

i) Incluir os atributos de identificação (ID) e condição (COND), representando-lhes com símbolo sublinha(-).

ii) Transformar a parte temporal numa forma interna.

iii) Transformar as palavras lógicas como AND, NOT, OR.

Este tratamento é feito para diminuir a complexidade da rotina de execução.

Logo em seguida, a consulta é dividida ("quebrada") em consultas simples. Consulta simples já foi mostrada na seção 4.1. Ao processo de cada consulta simples são acrescentadas como: rotinas para resolver a parte temporal, rotina de teste de condição e etc. Este processo gera uma verdadeira operação de montagem da regra em linguagem de prolog.

Para cada consulta simples, são transformadas em variáveis os simbolos como ? , -, constantes que são utilizados na consulta. Essas variáveis são instanciadas com os valores no Banco de Dados. As rotinas de apoio são feitas separadamente para facilitar a inclusão quando for necessário:

5.6.1 ALGORITMO DE EXECUÇÃO DE CONSULTA

```
PROCEDIMENTO EXECUÇÃO_DE_CONSULTA
  Inicio
    SE a consulta é recuperação ENTÃO /* tem interrogação */
       trata-consulta-recup
    SENÃO
       trata-consulta-bool
    Fim-se
  Fim

PROCEDIMENTO TRATA-CONSULTA-RECUP|TRATA-CONSULTA-BOOL
  Inicio
     SE a consulta é simples ENTÃO
       SE a consulta tem a parte temporal ENTÃO
          Faz_transf
          Preparação-da-parte-temporal.
          Faz_teste_de_atributos   /*Incluindo  teste  de  valor
                                     nebuloso*/
          Transforma-as-variáveis
          Transforma-operadores-de-tempo /* forma interna */
          Transforma Interrogação  |Acrescenta-rotina-transf-bool
          Acrescenta-rotina-transf-recup |Acrescente-teste-cond.
          Acrescenta-teste-atributo
        SENÃO
          Faz_teste_de_atributos
          Transforma-as-variáveis
        Fim-se
        Executa-rotinas-montadas
    SENÃO
        Dividir-em-consulta-simples
        Trata-consulta-recup          /*recursividade */
     Fim-se
  Fim

Procedimento Faz-tansformação-das-variáveis
        Inicio
          Transforma-Sublinha         /* underscore */
          Transforma-Constantes       /* constantes */
          Transforma-palavras-fuzzy
          Transforma-ponto e virgula
        Fim


Procedimento Executa_rotinas_montadas
        Inicio
          Grava-rotina-montada
          Faz_execução
          Faz-Retransf
          Gerar-Impressão
        Fim
```

5.6.2 DEDUÇÃO DE DADOS INCOMPLETOS

Implementamos duas rotinas para dedução de dados incompletos: Rotinas de teste de atributos e condição. A primeira rotina testa cada atributo com domínio constante, e com o valor correspondente no BD. O resultado da lógica é iniciado com UNKNOWN e dependendo do teste intermediário do atributo, o resultado final é armazenado na variável RESULTADO_LÓGICO. A segunda rotina testa as condições em conjunto com o valor do resultado_lógico. Esses testes são resumidos na tabela abaixo.

CONSULTA	BANCO DE DADOS	RESPOSTA LOGICA	TESTE DE CONDICAO	RESULTADO FINAL
			NIL	POSSIBLE
X	*	POSSIBLE	NOT	UNKNOWN
			OR	POSSIBLE
			NIL	YES
X	X	YES	NOT	NO
			OR	POSSIBLE
			NIL	UNKNOWN
X	Y	UNKNOWN	NOT	YES
			OR	UNKNOWN

TABELA 5.8: TABELA DE TESTE DE CONDIÇÃO

Encontramos várias dificuldades na definição de respostas para algumas consultas. Dois destes problemas são analisados abaixo:

1) Combinação de elementos incompletos e a condição NOT numa mesma tupla.

2) Dependências funcionais entre os atributos.

Utilizando a relação trabalho da página 36 como exemplo, para o primeiro caso, podemos fazer a seguinte pergunta: Trab(José,_, 50.000).--› significa que José ganha Cz$ 50.000 qualquer que seja o departamento que ele trabalha. Neste caso, é dificil saber a resposta porque no BD esta o fato que José não ganha Cz$ 50.000 e não trabalha no departamento desconhecido. Podemos imaginar outros casos mais complexo.

Podemos explicar o segundo caso com os seguintes fatos abaixo:

NOME	DEPT.	IDADE	CONDIÇAO
MARIA	DEE	*	OR
MARIA	DSC	20	OR

significa que Maria trabalha no departamento DEE ou DSC com idade 20 ou não conhecido. Neste caso, temos a dependência funciona entre os atributos **Nome** e **Idade**. Isto obriga que os valores da idade tem que ser iguais.

Também, têm casos em que alguns fatos que não são permitidos no sistema tem algum sentido no mundo real. Por exemplo, um fato como:

CARLOS DSC 100.000 OR

CARLOS DEE 120.000 OR

significa que Carlos trabalha no departamento DSC e ganha Cz$ 100.000 ou no DEE e ganha Cz$ 120.000.

Achamos que os casos mencionados acima merecem um estudo mais aprofundado.

5.7. PROCEDIMENTO DE EXECUÇAO DE UMA CONSULTA

Abaixo, descrevemos as principais etapas no processamento de consultas. Isto é, mostramos as etapas de execução desde a própria digitação da consulta até a impressão dos resultados. O processamento de uma consulta pode ser resumido em seis etapas mostradas na figura abaixo:

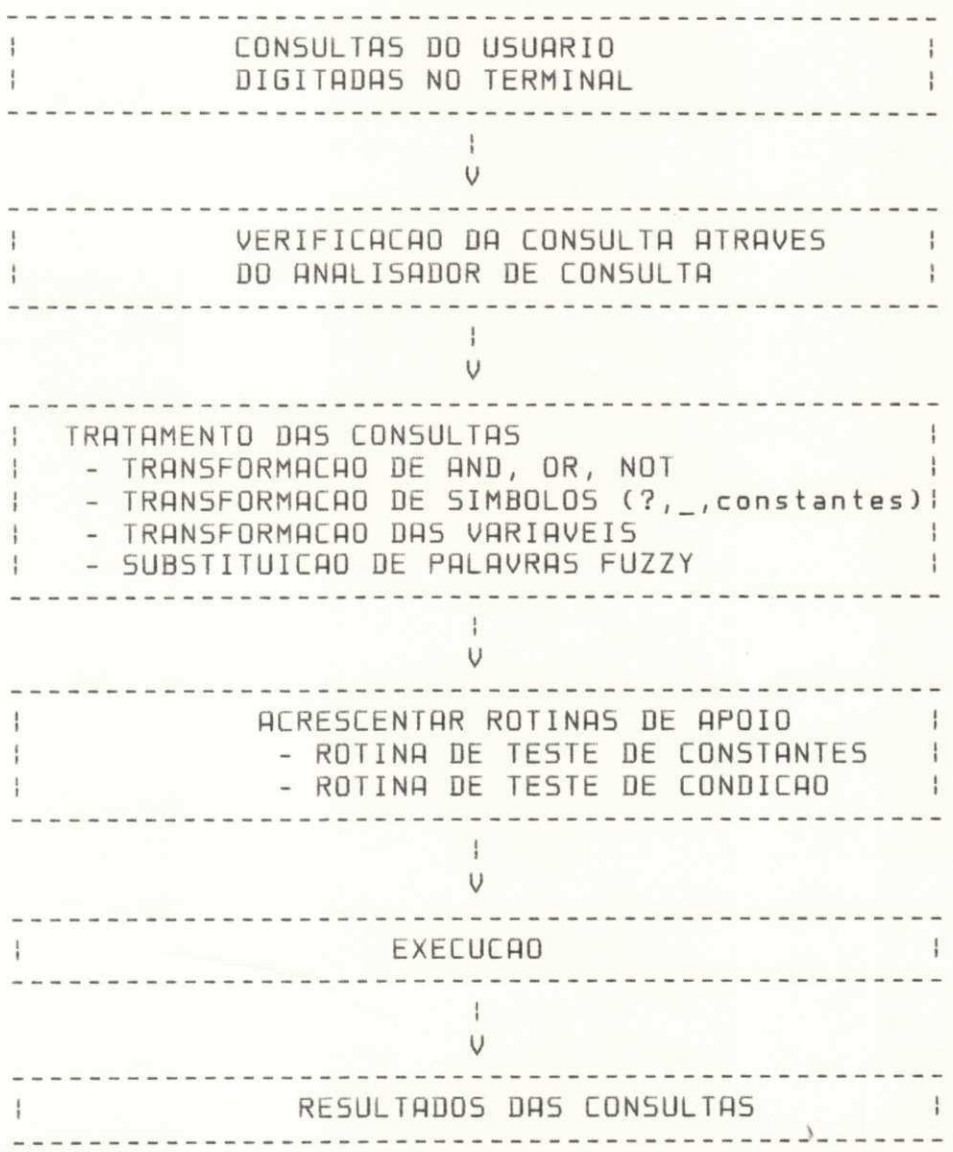


FIGURA 5.9: ETAPAS DE PROCESSAMENTO DE UMA CONSULTA

A seguir estão exemplos de dois tipos de consultas (Booleana e recuperação). Utilizando as relações Trab e gosta da seção 4.7,

vamos ver como essas perguntas abaixo podem ser avaliadas.

CONSULTAS BOOLEANAS

- José trabalha para alguma firma com salário menor que Cz$ 50.000,00?
- Paulo gosta de Beta desde 1987 à 1988?

CONSULTA RECUPERAÇAO

Quem trabalha para algum departamento e recebe menor que Cz$ 120.000?

Quem gosta de outra pessoa desde 1987 à 1988?

I) A primeira etapa mostra como as perguntas são transformadas numa forma canônica.

CONSULTAS BOOLEANAS

Trab(José,_,<50.000).

Gosta(Paulo,Beta;1987,1988).

CONSULTA RECUPERAÇAO

Trab(?,_,<120.000).

Gosta(?,?;1987,1988).

II) A segunda etapa faz a verificação das consultas onde são feitas uma análise sintática e semântica. No caso de erro, a execução da consulta é interrompida e uma mensagem é exibida no terminal.

III) A terceira etapa faz os tratamentos nas consultas. Esses tratamentos incluem transformações dos simbolos de interrogação, sublinha e os separadores. Isto pode ser visto no item IV a seguir.

IV) A etapa quatro acrescenta as rotinas de apoio. Esta etapa é considerada como a mais crítica do sistema. As consultas são colocadas numa forma mais aceitável para o computador (regras de produções de prolog).

CONSULTAS BOOLEANAS

```
<- relação(*.trab.A1.C1.A2.L1.A3.NIL) &
                                    /*Avaliação da relação trab*/
   teste(C1,JOSE) &                 /*Teste de constantes*/
                                    /*Teste de limite de dados*/
   testa_limite_bool(<,50.000,L1)&  /*Teste de condição*/

   teste_cond(A3).

<- relação(*.gosta.A1.C1.L1.A2.A3.A4.NIL) &
                                   /*Avaliação da relação gosta*/
   teste(C1,PAULO) &

   testa_limite_bd(A2,A3,V1,V2) &

   tempo_cons_bool(I.(1.1.1987.NIL).(1.1.1987.NIL).NIL,
                   I.(1.1.1988.NIL).(1.1.1988.NIL).NIL,V1,V2,Z) &

   delax(result_logico(W)) &        /*Deleção*/

   teste_lógico(W,Z,Q) &

   addax(result_logico(Q)) &        /*Inclusão*/

   teste_cond(A4).
```

CONSULTA DE RECUPERAÇAO

```
<- relação(*.trab.A1.X1.A2.L1.A3.NIL) &

   analisa_cond_1(A3) &             /*Teste de condição*/

   testa_limite_recup(<,120.000,L1) &

   re_transf(X1,Y1) &               /*Retransformação de tempo*/

   writech(Y1) &

   writech(' ') &

   analiza_cond_2(A3) &

   newline &

   fail.
```

```
(- relação(*.gosta.A1.X1.X2.A2.A3.A4.NIL) &
   analiza_cond_1(A4) &
   transf_tempo_bd(A2,A3,V1,V2) &
   tempo_cons_recup(I.(1.1.1987.NIL).(1.1.1987.NIL).NIL,
                    I.(1.1.1988.NIL).(1.1.1988.NIL).NIL,V1,V2,Z)&
    writech(X1) &
    writech(' ') &
    writech(X2) &
    writech(' ') &
    analiza_cond_2(A4) &
    newline &
    fail.
```

V) A consulta gerada é guardada no arquivo para ser executada. Isto é, fazer a comparação da consulta feita pelo usuário com o conteúdo do Banco de Dados.

VI) Os resultados das consultas são impressos.

BOOLEANAS

```
Possible
Yes
```

RECUPERAÇAO

```
Possibly CARLOS e MARIO e JOSE
PAULO e BETA
```

6. CONCLUSAO

Cuidadosamente, analisamos e implementamos o sistema RADINT que inclui as rotinas de dedução de tempo e fatos incompletos. O sistema RADINT é interativo, rápido e é geral para várias aplicações. A linguagem de consultas incorporado é simples. A atualização de dados é bem mais prática e fácil de manipular. Alem do mais, o analisador de consultas usa ambas ferramentas de análises léxico e sintática para fazer uma rigorosa checagem na entrada de consultas. O programa consta de aproximadamente 4000 linhas na linguagem PROLOG. O tempo de resposta em média é 2 segundos. Isto é consideravelmente admissivel.

Ao longo da implementação deste trabalho, encontramos várias dificuldades. Por exemplo, não foi possivel cumprir bem a quarta meta, mencionada no capitulo 1 em relação da consistência de dados, por causa da complexidade na atualização de dados incompletos no BD, e também, os problemas de dependências funcionais entre os atributos mencionado na seção 5.6.2. Foram incluidos os operadores de ordenação tais BEFORE(I), AFTER(I), mas as constantes de tempo como "THIS YEAR", "YEAR BEFORE", "LAST YEAR" não foram implementadas. Podemos também acrescentar outros operadores de tempo como "SEVERAL DAYS", "NEARLY A MONTH" e "A FEW WEEK". Achamos que esses problemas devem ser estudados em detalhes, pois deixamos como trabalhos futuros.

Esperamos que este trabalho motive futuras pesquisas no uso e inclusão de valores nulos no sistema de BD. Apesar das dificuldades encontradas ao longo do seu desenvolvimento, este projeto, foi muito importante não só pela grande importância do

software, mas pela possibilidade no uso de PROLOG na implementação do mesmo.

6.1. CONTRIBUIÇOES

Um dos trabalhos relacionados neste projeto foi o desenvolvimento de uma linguagem de consulta temporal TQUEL [Snodgrass/84]. A nossa contribuição foi mais abrangente. Incluimos predicados temporais, valores nulos e termos fuzzy. Mostramos como valores nulos, marcados ou não marcados podem ser incluidos dentro do banco de dados. Implementamos rotinas de dedução, tanto para tratar os dados incompletos como também os predicados temporais. Exploramos o modelo relacional para incluir vários elementos de informação incompleta. Distinguimos dois tipos de consultas onde consultas booleans podem ter respostas como YES, NO, POSSIBLE e UNKNOWN. Isso foi possivel através da suposição do mundo aberto adotada.

Foi implementado um minimo de rotinas necessárias ao sistema, incluindo facilidades interativas que tornam simples a sua utilização até mesmo por usuários leigos.

6.2. DIREÇÃO PARA TRABALHOS FUTUROS

Estas coleções de idéias também poderão melhorar o sistema, se forem incluidas na implementação do sistema. Alguns problemas devem ser resolvidos através de outras técnicas que podem ser incorporadas ao sistema, podem ser divididas nas seguintes áreas:

1) A otimização na avaliação de consultas pode ser tentada para melhorar a eficiência.
2) O sistema pode ser extendido para incluir unidades de tempo tais como horas, minutos e segundos, aumentando assim número de aplicações que podem utilizá-lo.
3) Testes exaustivos podem ser sugeridos como um método para obter melhores respostas de um sistema de BD incompleto.
4) Inclusão de probabilidades associadas aos dados no BD
5) Procedimento de provas pode ser incorporado como por exemplo, o algoritmo de resolução, que é adequado ao sistema de perguntas e respostas.
6) A inclusão de conjunto "fuzzy" no sistema de informação incompleta deve ser mais detalhada.
7) Um sistema de atualização "inteligente".
8) Inclusão de condições mais complexas dos elementos de dados incompletos.
9) Inclusão do valor "NO" e de referências indiretas.
10) Um estudo detalhado das combinações dos dominios do BD com asterisco (*).

VM (Virtual Memory) digitando:

PROLOG 1500

APENDICE A

MANUAL DE USO DO SISTEMA "RADINT"

Nesta seção mostramos como usar o sistema, ou seja os passos que devem ser tomados no manuseio deste sistema. Atualmente, o sistema pode ser usado no computador IBM no ambiente de VM/CMS.

Iniciando, o compilador PROLOG é ativado pelo sistema operacional VM (Virtual Memory) digitando:

```
PROLOG  1500
```

Para carregar o nosso sistema, basta digitar:

```
LOAD (RADINT)
```

onde RADINT é o nome do sistema.

O resumo das facilidades oferecidas pelo sistema é apresentado. Logo em seguida, o sistema pede à senha para permitir o uso do mesmo com o seguinte formato:

```
‹NOME-DO-USUARIO›-‹CODIGO DO USUARIO›
```

Caso o usuário forneça senha não autorizada, o sistema repete solicitação da senha até que seja digitada uma senha autorizada. Também, a data é atualizada. Como o sistema é interativo, direciona o usuário e mostra o que deve ser feito através de MENU. O sistema não têm sinal PRONTO, mas mostra sempre na tela mensagens sobre as operações a seguir. Para sair do sistema, basta digitar o comando:

```
"SAI".
```

No caso de qualquer problema relacionado com o espaço da memória, os comandos abaixo devem ser digitados antes de carregar o sistema.

```
DEF STOR 3M
I CMS
```

Isto é, para alocar mais espaço na memória para o sistema.

APENDICE B

CONCEITOS BASICOS SOBRE CONJUNTO "FUZZY"

Abaixo mostramos algumas definições e exemplos sobre os elementos básicos de conjuntos fuzzy que precisamos neste sistema. O assunto "CONJUNTO FUZZY" é muito abragente, portanto nós concentramos nos itens que achamos de maior necessidade para nosso trabalho. As definições abaixo, são baseadas nos conceitos de Zadeh [Zadeh/75].

DEFINIÇOES:

Conjunto fuzzy: é uma classe de objetos com grau continuo de membros do conjunto, ou seja;

Dado X = {x}, uma coleção de objetos, um subconjunto **A** de X, é um conjunto de pares ordenados {(x, UA(x))}, x E X, onde UA(x) é grau de membro de x em A, e UA é a função do membro (UA (x) E [0,1]).

VARIAVEL LINGUISTICA: é uma variável cujos valores são palavras da linguagem natural ou artificial. As palavras são menos precisas do que valores numéricos. Por exemplo, SITUAÇAO é uma variável fuzzy com palavras linguisticas às quais EXCELENTE, BOM, MEDIO e MAL e outros aos quais os seguintes intervalos podem ser associados: 10 a 9, 8 a 7, 6 a 5, 3 a 0.

REFERENCIAS BIBLIOGRAFICAS

[ALLEN, J.F. /83]. Maintaining Knowledge About Temporal Intervals. Communications of the ACM Vol. 26 No. 11 Nov. 1983.

[ALLEN, J.F. /84]. Towards a General theory of Action and Time. Artificial Intelligence 23. pp. 123-154. 1984.

[ANSI/X3/SPARC/75] Study Group on Data Base Management Systems 1975. "Interim Report", FDT (ACM SIGMOD Records) 7,2 1975.

[BISKUP, J. /81]. A Formal Approach to Null Values in Database Relations; Advances in Database theory. (Gallaire, H./Minker, J./Nicolas, J.M) Vol.I, Plenum, N.Y. 1981

[BISKUP, J. /83]. A Foundation of Codd's Relational Maybe - Operations. ACM Transactions on Database Systems, Vol. 8, no. 4, pp. 608-636, Dec. 1983.

[BOSSU, G., SIEGEL, P. /87]. Nonmonotonic Reasoning and Databases. Advances in Database Theory (Gallaire, H./Minker, J./Nicolas, J.M.) Vol. II, Plenum Press, NY, 1984

[BRUCE, B.C. /72]. A Model for Temporal References and its Application in a Question Answering Program. Artificial Intelligence 3, pp. 1-25, 1972.

[BUCKLES, B.P., PETRY, F.E. /82]. A Fuzzy Representation of Data for Relational Databases. Fuzzy sets and systems 7, pp.213-226, 1982. North-Holland Publishing Company.

[CODD, E.F./70]. A Relational Model of Data for Large Shared Data Banks. Communications of the ACM. Vol. 13, No. 6, pp. 377-387, 1970.

[CODD, E.F./79]. Extending The Database Relational Model to Capture More Meaning. ACM Transactions Database Systems 4,4, pp. 397- 434, DEC. 1979.

[COELHO, H., CTTA, J. C., PEREIRA. L. M. /80]. How to solve with Prolog. Ministério da Habitação e Obras Publicas, Lab. Nacional de Engenharia Civil, 2a. Edição, Lisboa, 1980.

[CLIFFORD, J. WARREN, D. S. /83]. Formal Semantics for Time in Databases. ACM. Transactions in Database Systems, Vol.8, No. 2, pp. 214-254 - June 1983.

[CLOCKSIN, IN. F., MELLISH. C. S./81]. Programming in Prolog. Springer-Verlag, Berlin, Heidelberg, New York 1981.

[DAHL, V./77]. On Database Systems Development Through logic. ACM. Transactions an Database Systems, Vol. 7. No.1. March 1982.

[DATE, C.J.]. Introdução aos Sistemas de Bancos de Dados. Editora Campus Ltda. Rio de Janeiro, 1980.

[GALLAIRE, H. MINKER, J. and NICOLAS, J.M/84]. Logic and Databases: A Deductive Approach. Computing Surveys Vol. 16, No. 2, pp. 153-183, June 1984.

[GOLSHANI, F./85]. Growing Certainty With Null Values. Information Systems Vol. 10, No. 3, pp. 289-297, 1985

[GRANT, J./77]. Null Values in a Relational Database. Information Processing Letters Vol.6, No.5, pp. 156-157. oct. 1977

[GRANT, J./79]. Partial Values in a Tabular Database Model.

[IMIELINSKI, T. /84]. On Algebraic Query Processing in Logical Databases: Advances in Database theory (Gallaire, H./Minker, J./Nicolas, J.M.) Vol.II, Plenum Press, M.Y, 1984.

[IMIELINSKI, T. /85]. Query Processing in Deductive Databases With Incomplete Information. Department of Computer Science Rutgers University, New Brunswick, New Jersey 08903.

[IMIELINSKI, T., LIPSKI. W. JR./84]. Incomplete information in Relational Databases. JACM. Vol. 31 No. 4. 1984.

[KAHN,K., GORRY, G.A./77]. Mechanizing Temporal Knowledge. Artificial Intelligence 9 pp. 87-108, 1977.

[KELLER, A.M, WILKINS, M.W./85]. On the use of an Extended Relational Model to Handle Changing Incomplete Information. IEEE. Transactions on software Engineering, vol. SE-11, No.7, July 1985.

[KLOPPROGGE, M.R., LOCKEMANN, P.C./]. Modelling Information Preserving Databases: Consequences of the Concept of Time. Proc. 9th VLBD, Florence, 1983

[KOWALSKI, R./81]. Logic as a Database Language, Tech. Rep., Dept. of Computing, Imperial College, London, June 1981.

[LEE, R.M., COELHO, H., COTTA, C./85]. Temporal Inferencing on Adminstrative Databases. Information Systems Vol. 10 No.2, pp. 197-206, 1985.

[LEVESQUE, H. J.]. The Logic of Incomplete Knowledge Bases. On Conceptual Modelling, Springer Verlag, 198 .

[LIPSKI, W. JR./79]. On semantic Issues Connected with Incomplete Information Databases. ACM. Transaction on Database Systems. Vol.4. No. 3 pp. 262-296, sept, 1979.

[LI,DEYI.]. A Prolog Database System. Dept. Of Computer Science, Heriot-Watt University, Edinburgh, UK. Research Studies Press Ltd.

[OVERMYER,R., STONEBRAKER,M./82]. Implementation of a Time Expert in a Data Base System. Dept. of Electrical Engineering and Computer Science, University of Califonia, Berkeley, CA.

[PRADE, H./84]. Lipski's Approach to Incomplete Information Data Bases Restated and Generalized in the Setting of Zadeh's Possibility Theory. Information Systems. Vol. 9. No. 1 pp. 27- 42, 1984.

[RESCHER,N.; URQUHART,A./71]. Temporal Logic, Springer Verlag, New York, 1971.

[SCHIEL,U./85a]. Representation and Retrieval of Incomplete and Temporal Information. Technical Report, Dept. of Computer Science, UFPB, Paraiba, Brasil.

[SCHIEL,U./85b]. The Time Dimension in Information Systems. In Theoretical and Formal Aspects of Information Systems, North Holland, 1985.

[SERNADAS,A./80]. Temporal Aspects of Lógical Procedure Definiton. Information Systems Vol.5. pp. 167-187. 1980.

[SNODGRASS,R./84]: The Temporal Query Language TQUEL, Dept. of Computer Science, University of North Carolina, Chapel Hill, No. 27514.

[TAHANI,V./76]. A Conceptual Framework for Fuzzy Query Processing A Step Toward Very Intelligent Database Systems. Information Processing and Management. Vol. 13, pp. 289-303.

[ULLMAN, J. C./80]. Principles of Database Systems. 1980. Computer Science Press, Inc.

[WILKINS, M.W./86a] A Model-theoretic Approach to Updating Logical Databases, Proc. of the 5th ACM PODS, Cambridge,

March 1986.

[ZADEH,L.A./65]. Fuzzy Sets. Information and Control 8, pp. 338-353. 1965.

[ZADEH, L. A./75]. The Concept of a Linguistic Variable and its Application to Approximate Reasoning I. Information Sciences 8, pp. 199 - 249, 1975.

[ZADEH, L. A./75]. The Concept of a Linguistic Variable and its Application to Approximate Reasoning II. Information Sciences 8, pp. 301 - 357, 1975.