REPRESENTAÇÃO E MANIPULAÇÃO DE OBJETOS EM BANCOS DE DADOS TEMPORAIS INDETERMINADOS

Ed Porto Bezerra

Tese submetida à Coordenação de Pós-Graduação em Engenharia Elétrica da Universidade Federal da Paraíba, como parte dos requisitos necessários para obtenção do grau de Doutor em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: *Processamento da Informação*

Ulrich Schiel, *Dr. rer. nat.*
Orientador

Campina Grande
Universidade Federal da Paraíba
Agosto de 2000

B574r Bezerra, Ed Porto
Representação e manipulação de objetos em bancos de dados temporais indeterminantes/Ed Porto Bezerra. - Campina Grande - PB: UFPB, 2000. 131p.: il. -

Tese (Doutorado) - UFPB/CCT-Engenharia Elétrica
Inclui Bibliografia

1. Informação - indeterminação
2. Temporal indeterminação
3. Banco de dados temporal I - Título

CDU: 681.3.07: 530.162

REPRESENTAÇÃO E MANIPULAÇÃO DE OBJETOS TEMPORAIS INCOMPLETOS

ED PORTO BEZERRA

Tese Aprovada em 14.08.2000

ULRICH SCHIEL, Dr.rer.nat., UFPB
Orientador

FERNANDO DA FONSECA DE SOUZA, Ph.D., UFPE
Componente da Banca

TARCÍSIO HAROLDO CAVALCANTE PEQUENO, Dr., UFC
Componente da Banca

EDILSON FERNEDA, **Dr., UFPB**
Componente da Banca

MARIA DE FÁTIMA QUEIROZ VIEIRA TURNELL, Ph.D., UFPB
Componente da Banca

BERNARDO LULA JÚNIOR, Dr., UFPB
Componente da Comissão

CAMPINA GRANDE - PB
Agosto - 2000

Para Lue e Ule, filhos.
Para Lu, esposa.
Meus três amores.

Agradecimentos

Pode parecer estranho, mas esta é a parte da tese que mais prazer me deu em escrevê-la. Primeiro porque traz o sentimento gozoso de missão cumprida; depois porque é muito bom agradecer.

Certamente existem milhares de atividades melhores do que o doutoramento, mas a necessidade profissional me convenceu do desafio. Felizmente algumas pessoas o tornaram menos difícil que de costume.

Para Ulrich, orientador, pela parceria decorrente, talvez, por me recomendar um tema que pouco conhecia (segundo Umberto Eco, este critério além de ser mais honesto, é resultante da confiança do orientador no orientando). Para Bernardo e Edilson, amigos-irmãos, adjuvantes e co-orientadores na prática. Para Hamurabi, pela iniciação ao mundo da lógica. Para Fátima, ex-coordenadora da COPELE, e Walfredo por acreditarem. Para Antônio Marcus, atual coordenador da COPELE, pela prontidão. Para Evandro, contemporâneo de doutorado, pelas dicas do seu "manual do doutorando". Para Érica e Sandro pela assessoria para assuntos do LaTex. Para Ângela e Pedro, da secretaria da COPELE, pela presteza. Para as companheiras e os companheiros professores do Departamento de Informática da UFPb, por me proporcionarem a experiência desta capacitação. Para Nevinha, funcionária da Pró-Reitoria de Pós-Graduação, pelo diálogo. Para Diemi e Romildo pela força nas tarefas domésticas. Para Lidiana pela construção da interface do MITO. Para o pessoal da Miniblio (COPIN) e do LabCom (DSC), pelo bom atendimento. Sem vocês, tudo seria muito mais complexo.

Para o PICDT/CAPES, pela bolsa de estudos.

Para Deus, por tudo e pela indeterminação temporal.

"It's present everywhere, but occupies no space.
We can measure it, but we can't see it, touch it,
Get rid of it, or put it in a container.
Everyone knows what it is and uses it every day,
But no one has been able to define it.
We can spend it, save it, waste it, or kill it,
But we can't destroy it or even change it,
And there's never any more or less of it".

Jespersen e Fitz-Randolph, FROM SUNDIALS TO ATOMIC CLOCKS

Sumário

Lista de figuras

Lista de tabelas

Resumo

Para obter um modelo adequado do universo de discurso de alguns sistemas de informação, é importante permitir a descrição de objetos imprecisos com indeterminação temporal. Estes objetos são chamados *objetos temporais indeterminados*. Esta tese apresenta um formalismo, chamado LITO (Logic of Indeterminate Temporal Objects), para representação, consulta e modificação de tais tipos de objetos. São distinguidas operações para consulta lógica, consulta de recuperação, inserção, remoção lógica, refinamento e atualização. Seguindo uma abordagem empírica, um protótipo entitulado MITO (Manipulation of Indeterminate Temporal Objects) foi implementado e vem mostrando estar consistente com a LITO.

Abstract

In order to obtain a suitable model for universe of discourse of some information systems, it is important to allow the description of imprecise objects with temporal indeterminacy. These objects are called *indeterminate temporal objects*. This thesis presents a formalism, called LITO (Logic of Indeterminate Temporal Objects), for the representation, querying and modification of such objects. Operations are created for logic query, retrieval query, insertion, logic deletion, refinement and update. Following an empirical approach, a prototype called MITO (Manipulation of Indeterminate Temporal Objects) was implemented and has been consistent with LITO.

Capítulo 1

Introdução

"Não sei quem me sonho ..."
Fernando Pessoa, CHUVA OBLÍQUA

1.1 Bancos de dados temporais indeterminados

Para se obter um modelo o mais adequado possível da realidade que se pretende modelar em determinadas aplicações, é de fundamental importância a capacidade de se descreverem características incompletas e temporais dos objetos. Particularmente, em alguns casos, estas duas características devem ser tratadas simultaneamente, ou seja, há indeterminação nos dados e no tempo. Para tanto, é desejável que sistemas de informação sejam capazes de permitir a criação, a atualização e a recuperação de objetos desta natureza, de forma que as respostas às consultas sejam as mais realistas possíveis.

Existem muitas aplicações nas quais é importante ter acesso não só ao estado atualizado das informações, mas também às transformações sofridas por estas. Para estas aplicações, a manutenção de dados históricos é um requisito fundamental. Dentre estas, se destacam as aplicações financeiras (gerenciamento de empréstimos bancários e fundos de investimento e de ações da bolsa de valores), as aplicações de programação de atividades (reservas em hotéis, reservas em companhias aéreas e gerenciamento de projetos), as aplicações envolvendo históricos de registro (acompanhamento clínico de pacientes, controle de estoque de mercadorias e sistemas de apoio à gerência) e as aplicações científicas (monitorização do clima atmosférico e de fenômenos sísmicos).

Embora várias aplicações de bancos de dados necessitem do gerenciamento da informação temporal, os Sistemas de Gerenciamento de Bancos de Dados atuais não provêem recursos específicos para estes tipos de informações, exceto quando o tempo é considerado um tipo de dado como outro qualquer.

Devido a ubiqüidade do tempo e sua importância em muitas aplicações de gerenciamento de dados, a pesquisa em gerenciamento de dados temporais continua ativa [GJ99, JS99, Sn00, TW99, WJW98]. Passados pouco mais de 20 anos, esta área tem produzido dois glossários de terminologias [BDE+98, JD98] e uma extensão a estes [Sn00]; um livro com as especificações de uma linguagem de consultas temporais chamada TSQL2 [Sn95] e três livros relevantes à área [EJS98, TCG+93, Sn00]. Uma visão geral sobre modelagem Entidade-Relacionamento Temporal [GJ99] reforça o quão ativa continua a área também no que diz respeito à modelagem de aspectos temporais dos esquemas de bancos de dados. Outro forte indicador desta atividade é o fato de que a linguagem de consultas a bancos de dados SQL3 [Sn00], prestes a ser lançada como padrão, possui um módulo de consultas temporais, chamado SQL/Temporal [Me97, SBJ+96a, SBJ+96b].

Por outro lado, a informação indeterminada está presente em certos sistemas de informação, pois nem sempre se tem conhecimento exato sobre certas informações do mundo real. Segundo Motro [Mo93] "a incerteza permeia nosso entendimento do mundo real. Sendo o propósito dos sistemas de informação modelar o mundo real, eles deveriam ser capazes de tratar a incerteza". As seguintes afirmações servem de exemplo para informações indeterminadas: "a idade de José é de 30 a 34 anos", "o salário de José é maior do que o salário mínimo", "o endereço de José é desconhecido", "o local de trabalho de José não é a UFPb", "o pedreiro é velho", "José está ganhando menos do que Maria", "o projeto vai terminar em menos de 3 anos". Uma explicação sobre os vários termos usados na literatura para referência à informação que não está determinada, tais como estes exemplificados, será vista no capítulo 3. A importância de se poder representar a informação indeterminada é reforçada pela seguinte afirmação: o que se conhece sobre certo evento é que ele ocorrerá em 2001 na cidade de Campina Grande ou na cidade de João Pessoa. Se isto pudesse ser representado num banco de dados, mesmo com a indeterminação a respeito da cidade que sediará o evento, uma consulta sobre quais eventos se realizarão em 2001 retornaria o nome deste evento na sua resposta. Mais especificamente, uma consulta sobre quais eventos acontecerão na Paraíba em 2001 também teria este evento como parte de sua resposta. Por outro lado, uma consulta sobre quais os eventos programados para João Pessoa em 2001 retornaria este evento dentre aqueles possíveis de realizar-se. Esta resposta é mais informativa do que a resposta retornada caso este fato não estivesse representado no banco de dados, ou seja, a resposta não teria este evento. Destarte, se percebe que a representação

e a manipulação da informação de conteúdo indeterminado (no exemplo visto, a cidade é indeterminada) seria mais útil do que a atual limitação dos bancos de dados em permitir apenas que dados exatos sejam representados.

Geralmente a indeterminação pode ocorrer em três aspectos distintos dos objetos de um sistema de informação: no valor de um atributo do objeto, em relação ao objeto como um todo e sobre o tempo de validade [JD98] do objeto.

A indeterminação no valor de um atributo do objeto representa a ignorância do estado exato de um objeto. Por exemplo, o atributo *idade* de José contendo 30 ou 31 anos denota que há dois possíveis estados para representação deste objeto: aquele onde José tem 30 anos e aquele onde ele possui 31 anos.

A indeterminação no objeto representa que a pertinência deste numa classe não é inteiramente determinada. Suponha que a existência de um objeto numa classe esteja condicionada a uma restrição qualquer. Por exemplo, uma classe com objetos representando pessoas físicas pode condicionar aquelas acima de 16 anos como aptas a votar.

A indeterminação temporal ou a indeterminação sobre o tempo de validade do objeto é decorrente do desconhecimento deste tempo, ou seja, não se sabe precisamente quando o fato se tornou verdadeiro no mundo real. A data de recebimento de uma encomenda pode ser um dia entre 5 e 15 de setembro de 2000. Na indeterminação do tempo de validade é sabido que um fato armazenado num banco de dados realmente ocorreu, mas é desconhecido exatamente quando. A indeterminação do tempo de ocorrência de um fato pode ser representada como um conjunto de instantes possíveis, delimitando quando um fato pode ter ocorrido, e uma distribuição de probabilidade sobre tal conjunto [Sn95]. Também pode ser usada uma representação de intervalos assumindo uma distribuição uniforme entre os instantes possíveis.

A representação e o tratamento da informação incompleta vem sendo estudado em várias áreas das ciências da computação, principalmente em inteligência artificial e em bancos de dados, seguindo abordagens distintas. Será visto que os aspectos utilizados pela abordagem aqui proposta são a indeterminação no valor de um atributo e no tempo de validade de um objeto. Bancos de dados que armazenam objetos com estes aspectos são chamados de Bancos de Dados Temporais Indeterminados (BDTI) [Ch94].

Um breve histórico sobre esta pesquisa é delineado a seguir: estudos sobre a dimensão temporal em sistemas de informação foram iniciados por Schiel [Sc85] e estendidos para a representação e a recuperação de informação incompleta [Sc86, Sc87]. Um protótipo para validação destes estudos foi implementado por Oresotu [Or88]. Em decorrência da vitalidade da pesquisa na área de tratamento de informação incompleta, se optou pela atualização de conceitos e sua conseqüente extensão para aspectos ainda pouco explorados como a atualização de objetos temporais indeterminados [BSF97, BSN98] e a formalização destes [BSL99].

Esta pesquisa abrangeu tanto os aspectos teóricos quanto os aspectos práticos envolvendo a indeterminação da informação. Seus aspectos práticos podem ser percebidos pelo estudo de linguagens de consulta temporais. Inicialmente foi estudada a linguagem TSQL2 [Sn95] que é a primeira linguagem de consulta temporal criada pela comunidade de pesquisadores da área de bancos de dados temporais. Outras propostas mais recentes de linguagens de consulta temporais, baseadas em TSQL2, são as seguintes: uma extensão temporal a SQL-92 [MS93, Sn00] e SQL/Temporal [Me97, Sn00]. Todas estas linguagens de consulta foram pioneiras no sentido de propor representações e operações para consulta, inserção, remoção e atualização de instantes e intervalos temporais, porém não suportam a indeterminação da informação. TSQL2 é a única que propõe uma só forma para se representar instantes e intervalos indeterminados.

Quanto aos aspectos teóricos envolvendo a indeteminação informacional, foram estudadas lógicas temporais [va95], uma extensão da teoria relacional [Re94] e uma lógica de base de conhecimento incompleta [Le94]. Estes formalismos abordam apenas formas de representação e manipulação de indeterminação não-temporal. Também foi investigada a sintaxe e a semântica de TSQL2 [DS98] que propõe uma só forma de representação de instante e intervalo indeterminado. Todos os formalismos estudados apresentam soluções parciais ao problema de representação e manipulação de objetos temporais indeterminados. Como relatado anteriormente, a presença destes tipos de objetos é intrínseca ao domínio de certas aplicações, o que reforça a importância de propostas mais abrangentes para representá-los e manipulá-los. Esta tese apresenta uma proposta de um formalismo para representação e manipulação de objetos temporais indeterminados.

Em certas aplicações torna-se fundamental a modelagem das características temporais e indeterminadas dos seus objetos. Todavia, os sistemas de informação baseados em computadores e, principalmente os sistemas gerenciadores de bancos de dados atuais, não dispõem de recursos para que estes objetos temporais indeterminados sejam representados, consultados e atualizados.

Embora existam algumas propostas de linguagens de consulta temporais, estas não suportam a representação da informação indeterminada. Uma destas propostas apresenta uma forma para representação de instantes e intervalos indeterminados. Desta mesma limitação sofrem os formalismos propostos até agora para definições sobre representação e manipulação de objetos temporais indeterminados, principalmente quanto à indeterminação temporal, ou seja, tanto os formalismos quanto as linguagens de consulta temporais apresentam solução parcial à representação e manipulação destes objetos.

Esta tese tem como objetivo principal elaborar um formalismo que forneça definições sobre a representação e a manipulação de objetos temporais indeterminados em BDTI. A manipulação envolve novas operações para consulta de recuperação, consulta lógica, inserção, remoção lógica, refinamento e atualização destes objetos.

Para mostrar a aplicabilidade do formalismo foi implementado em LPA-Prolog [LPA97] um sistema para representação e manipulação de objetos temporais indeterminados, chamado MITO (Manipulation of Indeterminate Temporal Objects). O MITO tem se mostrado em concordância com a parte sintática de uma lógica, chamada LITO (Logic of Indeterminate Temporal Objects), que é resultado do formalismo criado.

1.3 Metodologia utilizada

O procedimento usado para alcançar o objetivo da tese se iniciou com o levantamento da bibliografia considerando a questão da indeterminação informacional em geral e da indeterminação temporal em particular. Após esta fase, seguiu-se o estudo de vários dos formalismos encontrados, abordando a investigação de cálculos e lógicas temporais, e envolvendo a análise de linguagens de consultas temporais. Nesta etapa, foi identificada a ausência de um formalismo que abordasse definições sobre a representação e o tratamento

de objetos temporais indeterminados. Destarte, foi percebido que a proposta para a concepção e o desenvolvimento de um formalismo deste tipo, preencheria este vazio existente na área de bancos de dados e, particularmente, na área de BDTI.

A princípio, planejou-se a elaboração de uma lógica temporal específica para objetos temporais indeterminados, envolvendo tanto sua parte sintática quanto sua parte semântica. A experimentação dos conceitos definidos pela sintaxe da LITO produziu um sistema implementado em LPA-Prolog [LPA97]. Esta implementação foi de suma importância pelo fato de fornecer subsídios para a revalidação dos conceitos expressos na axiomática da LITO. Embora a criação da semântica tivesse que ser postergada, o formalismo elaborado contém novas formas de representação e novas operações para manipulação de objetos temporais indeterminados indicando que seu estado atual se apresenta como solução à questão da representação e manipulação de objetos em BDTI. O MITO tem mostrado a aplicabilidade da LITO.

1.4 Contribuição da tese

A representação e o tratamento da informação indeterminada, da informação temporal e da informação temporal indeterminada (indeterminação temporal) tem sido parcialmente resolvida pelos formalismos propostos até agora. A representação e a manipulação de objetos temporais indeterminados é essencial para certas aplicações que fazem uso de sistemas de informação baseados em computadores, principalmente de sistemas gerenciadores de bancos de dados.

A principal contribuição desta tese é a proposta de um formalismo que fornece definições completas sobre a representação e a manipulação de objetos em BDTI. As novas formas para se representar a indeterminação, o tempo e a indeterminação temporal destes objetos, assim como a concepção e a construção de mecanismos para consultas lógicas, consultas de recuperação, inserções, remoções lógicas, refinamentos e atualizações de objetos em BDTI, enriquecem sobremaneira este formalismo tornando-o parâmetro à solução da modelagem de objetos temporais indeterminados em BDTI.

1.5 Estruturação da tese

Esta tese esta estruturada em sete capítulos logicamente seqüenciados para atender aos seguintes requisitos: contextualizar a importância da indeterminação informacional, temporal ou não, e apresentar o problema da ausência de uma proposta de um formalismo para representação e manipulação de objetos temporais indeterminados; investigar os trabalhos que parcialmente abordam este problema; propor uma solução desenvolvida formalmente para a solução do problema; mostrar uma experimentação para validar a solução proposta e, por último, descrever as conclusões sobre a tese e apontar possíveis continuações da pesquisa.

Especificamente o capítulo 2 aborda o assunto da informação temporal: a ontologia do tempo; o cálculo de intervalos de Allen que tem servido de modelo para várias abordagens temporais; algumas lógicas temporais representativas e uma classificação de três "estilos" destas; bancos de dados temporais; e o cálculo relacional e a lógica temporal como duas linguagens de consultas abstratas. Uma visão geral sobre informação indeterminada em ambiente de banco de dados é tratada no capítulo 3, onde se apresenta a idéia de mundos possíveis, basilar para informação incompleta, e uma classificação para informação indeterminada. No capítulo 4 são abordados trabalhos que refletem o estado da arte da área de representação e tratamento de informação indeterminada em bancos de dados temporais e estão, por conseguinte, relacionados ao tema desta tese.

O primeiro destes trabalhos é uma linguagem de consulta a banco de dados temporais, chamada TSQL2 [Sn95], que foi especificada pela comunidade da área de bancos de dados temporais e tem sido utilizada como referência. TSQL2 trata a indeterminação temporal [DS98] sob o enfoque da distribuição probabilística da incerteza no tempo de validade de uma tupla, transferindo ao usuário a especificação de novos parâmetros para filtrar dados consultados.

Uma extensão temporal a SQL92 [MS93] proposta por Snodgrass [Sn00] é mostrada como contribuição, principalmente às declarações para modificações de um banco de dados temporais. É apresentado também um subconjunto de TSQL2, chamado SQL/Temporal [Me97], proposto como um módulo temporal à linguagem SQL3. SQL/Temporal incorpora apenas os tempos de validade e de transação de TSQL2, não

abordando a indeterminação temporal.

Uma apresentação sucinta de três formalismos envolvendo a indeterminação em banco de dados situa as abordagens formais propostas. O primeiro formalismo descreve a teoria relacional generalizada com valores nulos e informação disjuntiva [Re94]. O segundo descreve a sintaxe e semântica propostas para TSQL2 (descritas na mesma seção onde esta é apresentada) e o último sugere uma lógica de bases de conhecimento incompleto proposta por Levesque [Le94].

No capítulo 5 uma linguagem formal e uma axiomática de uma lógica para objetos temporais indeterminados são discriminadas. Também é apresentada a demonstração de um importante proposição do formalismo, assim como são feitas considerações sobre os procedimentos genéricos Prolog para modificação de um BDTI.

Um sistema para representação e manipulação de objetos temporais indeterminados é apresentado no capítulo 6, com ênfase na exemplificação de suas operações e seus principais algoritmos.

A conclusão da pesquisa e a sugestão de direções futuras são delineadas no capítulo 7. O apêndice contém os *layouts* das telas que compõem a interface do sistema, com exemplos correspondentes a cada uma de suas operações.

Capítulo 2

Informação temporal

"Transcorrei
Ó tempo rei, ó tempo rei
Transformai
As velhas formas do viver."
Gilberto Gil, TEMPO REI

Neste capítulo é introduzido um estudo sobre os principais conceitos do tempo, assim como das lógicas temporais e dos bancos de dados temporais. Duas linguagens de consultas abstratas [Ch94] a bancos de dados temporais também serão abordadas.

Costuma-se considerar o universo físico como composto por duas dimensões: o espaço e o tempo. A cada entidade espacial está associado um elemento temporal que representa o tempo de existência da entidade. Inversamente, a um instante temporal se pode associar todas as entidades espaciais que existiram naquele instante.

A maioria das abordagens sobre o tempo inclui uma classificação das proposições, de acordo com seus diferentes padrões de ocorrência temporal, baseada em critérios cognitivos, lingüísticos ou intuitivos. Estas diferentes classes são chamadas entidades temporais [Al84]. Há diferentes propostas, todas igualmente criticadas pela ausência de uma definição semântica formal, para representação das entidades temporais [BCS+96].

2.1 Ontologia do tempo

Existem diversas formas de se ver a estrutura do tempo, sendo que a determinação da forma depende mais do ambiente que se quer modelar do que de uma definição filosófica da estrutura correta. As propriedades estruturais do tempo são as seguintes:

- *Discreto*, *denso* ou *contínuo*: o tempo pode ser considerado como uma coleção discreta de elementos temporais ou, ao contrário, para quaisquer dois elementos tempo-

rais, sempre há um terceiro entre eles (denso), ou ainda todos os elementos entre estes dois pontos pertencem ao tempo (continuo).

- *Limitado* ou *ilimitado* (*finito* ou *infinito*): o tempo é considerado infinito em uma ou ambas as direções (passado e futuro).
- *Precedência* (*linear*, *ramificada*, *paralelo* ou *circular*): os elementos temporais formam um conjunto totalmente ordenado (linear) ou não.

Vários modelos de tempo têm sido propostos na literatura. Estes vêem o tempo como um conjunto ordenado de instantes temporais T que pode ser discreto, denso ou contínuo. Intuitivamente, os instantes num modelo de tempo *discreto* são isomórficos aos números naturais, ou seja, existe a noção de que todo instante tem um sucessor único. Já os instantes no modelo de tempo *denso* são isomorfos aos números racionais, isto é, entre dois instantes quaisquer há sempre um outro. O modelo de tempo *contínuo* é isomórfico aos números reais, ou seja, são densos e assumem que todo ponto entre dois instantes quaisquer é um instante temporal. Nestes isomorfismos, o número 0 (zero) pode ser interpretado como o presente, os números negativos como o passado e os positivos como o futuro [Sc96].

O domínio temporal T é o conjunto de entidades temporais primitivas (instantes ou intervalos) usadas para definir e interpretar conceitos relacionados ao tempo. O conjunto é ordenado por um relacionamento $\leq$ sobre tais entidades que representa a noção de anterioridade/posterioridade/igualdade. Se este relacionamento define uma relação de ordem total, diz-se que o tempo é linear. Uma variante deste modelo de tempo *linear* é o modelo de tempo *ramificado* que permite modelar vários possíveis futuros. Com isso, se pode descrever a incerteza do futuro. Caso o presente e o passado também sejam incertos, tempos paralelos são usados, um tempo para cada mundo possível. Neste caso, o relacionamento $\leq$ define uma relação de ordem parcial sobre T.

Para a maioria das aplicações de sistemas de informação, tais como aquelas que mantêm registros de empregados, é natural se usar um modelo de tempo discreto e linear.

Certas aplicações podem limitar o domínio temporal. Um domínio temporal é limitado se ele contém limites superior e inferior. Formalmente, o domínio temporal T é limitado se existem $t', t'' \in T$ tal que $t' \leq t \leq t''$ para todo $t \in T$.

As primitivas básicas temporais para qualquer modelo temporal são *instante* e *intervalo*. Um instante é um ponto de tempo sobre uma linha de tempo imaginária e um intervalo é o tempo entre dois instantes [JD98]. A rigor, um destes conceitos pode ser escolhido como primitivo e o outro derivado deste [Vi94].

2.2 O cálculo de intervalos de Allen

Um dos trabalhos pioneiros na formalização do tempo é o de Allen [Al83] que definiu um cálculo de intervalos baseado em 13 relações que correspondem aos relacionamentos mutuamente exclusivos que podem existir entre dois intervalos (FIG. 1). Allen formulou um conjunto de 13 axiomas que definem o comportamento do cálculo:

- dados dois intervalos quaisquer, eles necessariamente estão relacionados entre si segundo um dos 13 relacionamentos;
- os relacionamentos são mutuamente exclusivos;
- alguns relacionamentos têm um comportamento transitivo, como por exemplo, se A é um intervalo antes de um intervalo B e B está antes de um intervalo C, então A está antes de C.

Os predicados são os seguintes:

- *antes de (A,B)*: o intervalo A está antes do intervalo B e eles não se sobrepõem;
- *encontra (A,B)*: o intervalo A está antes do intervalo B, mas não há intervalos entre eles, ou seja, B se inicia quando A acaba;
- *sobrepõe (A,B)*: o intervalo A começa antes de B e eles se sobrepõem;
- *inicia (A,B)*: o intervalo A partilha do mesmo começo de B, mas acaba antes que B acabe;
- *durante (A,B)*: o intervalo A está inteiramente contido em B;
- *termina (A,B)*: o intervalo A partilha do mesmo fim do intervalo B, mas começa depois que B começa;
- *igual a (A,B)*: A e B são o mesmo intervalo.

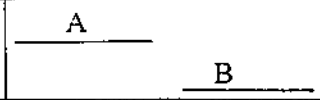
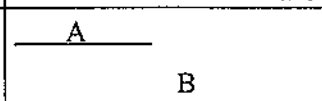
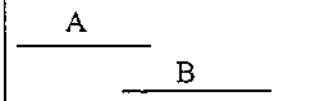
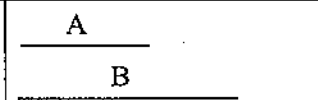
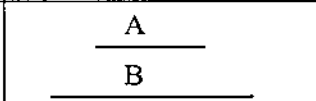
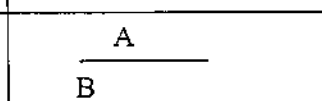
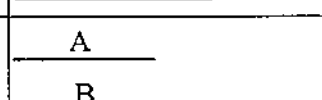
Relacionamento	dual	ilustração
A antes B	B depois A	
A encontra B	B encontrado_por A	
A sobrepõe B	B sobreposto_por A	
A inicia B	B iniciado_por A	
A durante B	B contém A	
A termina B	B terminado_por A	
A igual B	B igual A	

FIGURA 1 - Os 13 relacionamentos entre intervalos temporais

Variantes deste cálculo de intervalos foram desenvolvidas por Bolour e Dekeyser [BD83] e por Schiel [Sc85].

Uma teoria é proposta por Vila [Vi93] onde instantes e períodos são primitivas básicas: os instantes são um conjunto linear, ilimitado e denso; e os períodos se relacionam a estes pelas funções *BEGIN*(period) e *END*(period).

Há críticas a cada uma das três primitivas adotadas (instante, intervalo ou híbrida) [Vi94] e uma discussão sobre elas está fora do objetivo desta tese. A abordagem híbrida é a utilizada nesta pesquisa pelo fato de ser a que mais fielmente representa os objetos temporais do mundo real.

2.3 Lógicas temporais

Lógicas temporais são úteis para o raciocínio sobre um mundo mutante e, a medida que a ordem temporal de ações/eventos pode ser descrita, as lógicas temporais são úteis para a representação de sistemas dinâmicos. Consequentemente, lógicas temporais são

usadas na especificação de sistemas reativos [MP92], e nas aplicações onde o conceito de tempo é central, tais como: sistemas de *planning* [AHT90], representação temporal [Al84], bancos de dados temporais [BCS+96] e raciocínio sobre ação [Gi93]. Além disso, a linguagem da lógica temporal permite a expressão de importantes propriedades dos sistemas. Por exemplo, se uma lógica temporal *discreta* é usada, onde o tempo é representado como uma seqüência de momentos distintos, então fórmulas temporais podem representar os passos individuais de uma execução. Da mesma forma, propriedades como "alguma condição ocorrendo *alguma vez* no futuro" ou "alguma condição ocorrendo *sempre*" podem ser descritas pelo uso de lógicas temporais.

Um sistema lógico que pretende representar ações e planos de agentes deve ser capaz de representar eventos temporais. Destarte, é imprescindível a existência de uma forma para representar sentenças cujo valor verdade variem com o passar do tempo. A *lógica temporal linear*, como a lógica temporal *minimal* [va95] apresentada na próxima seção, e a *lógica temporal ramificada* [Za91] são fundamentais para a representação de tais sentenças.

O raciocínio exigido por um problema como o de mover um objeto de sua posição original para cima de um outro objeto é chamado de raciocínio sobre mudança. Neste caso específico, ou seja, quando o problema considera como provável causa da mudança a realização de ações, está-se diante de um tipo de raciocínio sobre mudança chamado de raciocínio sobre ação [Gi93]. Este raciocínio é um tipo de *raciocínio temporal* já que inevitavelmente para se raciocinar sobre os efeitos de uma ação, se deve considerar a anterioridade e a posterioridade da realização da ação. Silvestre [Si98] analisa as lógicas da ação, como também as lógicas do tempo, e propõe uma formalização do raciocínio sobre ação chamada de *lógica do plano e da ação.*

Uma proposta sugerindo requisitos essenciais para que uma lógica seja classificada como temporal foi feita por Reichgelt [Re89]. De acordo com esta proposta, uma lógica temporal deve possuir cinco requisitos básicos dentre os quais são destacados (por se relacionarem diretamente com o assunto desta tese) os seguintes:

- prover a representação do conhecimento temporal quanto à sua precisão: o conhecimento temporal pode ser indeterminado (segundo a nomenclatura adotada aqui) no sentido de que pode-se representar algo acontecido em algum tempo passado ou algo

que acontecerá, sem se ter conhecimento do instante em que o evento aconteceu ou acontecerá. Ao contrário, o conhecimento temporal é determinado quando o instante exato do acontecimento é conhecido. Sendo assim, o formalismo deve possibilitar a representação tanto do conhecimento determinado quanto do conhecimento determinado.

- permitir a combinação da lógica temporal com outras lógicas: uma lógica temporal deve possibilitar sua combinação com qualquer outro tipo de lógica, como por exemplo, lógica do conhecimento e crença ou lógicas modais com seus operadores de possibilidade e necessidade. Isto torna viável a utilização da lógica temporal em aplicações onde se queira raciocinar sobre crenças pessoais ou sobre a necessidade e a possibilidade de certas proposições.

Nas próximas três seções serão apresentados diferentes "estilos" de lógica temporal, sugeridos por Reichgelt [Re89], baseado no fato de que há três maneiras principais de introduzir o tempo na lógica: a lógica de primeira-ordem com argumentos temporais, as lógicas modais temporais e as lógicas temporais *reificadas*.

2.3.1 Lógica de primeira-ordem com argumentos temporais

Este "estilo" consiste na representação do tempo apenas como mais um parâmetro num cálculo de predicado de primeira-ordem [Da93]. As funções e os predicados são estendidas com o argumento temporal denotando o tempo particular no qual elas devem ser interpretadas. Por exemplo, considere o predicado binário *dança* (x, y) ao qual se acrescentou o argumento temporal, transformando-o em *dança* (x, y, t), denotando que o dançarino x dança com a dançarina y no instante (ou intervalo) t. A sintaxe desta linguagem é semelhante a da lógica de primeira-ordem, exceto que todo predicado n-ário se torna uma relação $n+1$-ária, onde os primeiros elementos são do tipo não-temporal e o último argumento é do tipo temporal. Para possibilitar o raciocínio temporal é introduzida uma relação de ordem *lt* (do inglês *less than*). Por exemplo, a relação *lt* (t_1, t_2) deve ser lida como: t_1 aconteceu antes de t_2. Também foi necessária a adição de uma constante temporal t_0, que se refere ao tempo presente, para que se pudesse expressar que algo foi o caso em algum tempo passado. É possível expressar, por exemplo, que José dançou com Maria em algum tempo do passado como:

$$(\exists t{:}T)\ ((lt\ (t, t_0))\ \&\ (dança(josé, maria, t))$$

onde a notação $(\exists x{:}s)$ significa que existe uma variável x do tipo s.

2.3.2 Lógica modal temporal

A lógica modal temporal é uma extensão da lógica proposicional clássica [Da93]. A idéia básica da lógica modal temporal é dispor de certos operadores modais para designar a veracidade passada ou futura das sentenças. Por exemplo, os operadores modais F e P podem ser usados para este fim. Assim, se α é uma sentença, então $F\alpha$ e $P\alpha$ também são sentenças significando, respectivamente, que ao menos uma vez no futuro será o caso que α e que ao menos uma vez no passado foi o caso que α. A partir destes operadores, podem-se derivar outros tais como G e H, assim definidos:

$G\alpha =_{def} \neg F\neg\alpha$

$H\alpha =_{def} \neg P\neg\alpha$

A sentença $G\alpha$ significa que sempre no futuro será o caso que α e $H\alpha$ significa que sempre no passado foi o caso que α.

Há diversas axiomatizações para a lógica temporal clássica, cada uma delas embutindo características específicas para aspectos temporais. As lógicas temporais clássicas podem ser classificadas em linear e ramificada. Um exemplo de uma axiomatização para uma *lógica temporal linear* é a lógica do tempo *minimal* [va95], apresentada a seguir, que se restringe somente às características essenciais do raciocínio temporal:

Axiomas

1) todas as tautologias do cálculo proposicional

2) $F(\alpha \rightarrow \beta) \rightarrow (F\alpha \rightarrow F\beta)$	$P(\alpha \rightarrow \beta) \rightarrow (P\alpha \rightarrow P\beta)$	*distribuição*
3) $F\alpha \leftrightarrow \neg G\neg\alpha$	$P\alpha \leftrightarrow \neg H\neg\alpha$	*dualidade*
4) $\alpha \rightarrow GP\alpha$	$\alpha \rightarrow HF\alpha$	*conversão*

Regras de inferência

se $\vdash \alpha$ e $\vdash \alpha \rightarrow \beta$ então $\vdash \beta$	*modus ponens*

se |- α então |- Gα se |- α então |- Hα *temporalização*

Em linhas gerais, estes axiomas e regras de inferência especificam o seguinte:

- Os axiomas de distribuição determinam a distribuição dos operadores F e P, considerando a implicação (→);
- Os axiomas de dualidade especificam as formas abreviadas equivalentes aos operadores G e H;
- Os axiomas de conversão indicam que se α é verdade, então também é verdade que sempre será o caso que foi o caso que α, e sempre foi o caso que será o caso que α;

A estrutura de modelo M, sobre a qual é dado o valor semântico das fórmulas, é uma tripla <T, <, V> onde

(i) T é um conjunto não vazio de instantes

(ii) O símbolo < representa uma relação transitiva, não reflexiva, assimétrica e *total entre instantes*

(iii) V : P X T → {Verdade, Falso} é uma função que, para todo símbolo proposicional em P, retorna seu valor semântico num dado instante.

O primeiro componente do modelo M é uma tentativa de capturar o conceito intuitivo que temos de momento de tempo. Conforme essa intuição, dados dois instantes distintos, sempre um deles antecede o outro e vice-versa. O símbolo < especifica essa relação de anterioridade/posterioridade. A assimetria desta relação indica que os elementos de T são entidades únicas e não passíveis de repetição dentro da ordem estabelecida pela relação. Pelo fato da relação < ser total, a estrutura de tempo definida não admite ramificações, ou seja, é linear. Daí o termo lógica temporal linear.

A relação ||- dá o valor semântico das sentenças temporais onde M ||- $_t$ α significa que α é verdade no modelo M no instante t ∈ T:

$M \Vdash_t P$ sse $V_t(P) = Verdade$;

$M \Vdash_t \neg\alpha$ sse $M \neg\Vdash_t \neg\alpha$;

$M \Vdash_t \alpha \wedge \beta$ sse $M \Vdash_t \alpha \wedge M \Vdash_t \beta$;

$M \Vdash_t F\alpha$ sse para algum instante $t' \in T$ tal que $t < t'$, $M \Vdash_{t'} \alpha$;

$M \Vdash_t P\alpha$ sse para algum instante $t' \in T$ tal que $t' < t$, $M \Vdash_{t'} \alpha$;

$M \Vdash_t G\alpha$ sse para todo instante $t' \in T$ tal que $t < t'$, $M \Vdash_{t'} \alpha$;

$M \Vdash_t H\alpha$ sse para algum instante $t' \in T$ tal que $t' < t$, $M \Vdash_{t'} \alpha$;

Note que a lógica temporal linear é uma teoria determinística, ou seja, de acordo com ela o que virá a ser o caso já está determinado. Por exemplo, suponha que no instante *t* um dado não viciado é arremessado. O resultado desta ação já está previamente determinado em *t*. Isto se deve ao postulado da totalidade da relação <, ou seja, como para todo t, t' $\in$ T tem-se t < t', há uma única cadeia de instantes representando um possível curso de história que contém um dos seis possíveis resultados do arremesso. Logo, se tem o determinismo embutido na lógica. Daí, percebe-se a inadequação da lógica temporal linear para problemas como o de *planning* [AHT90] onde se deve considerar todos os possíveis resultados de um evento para se construir o plano. Ou seja, o que se quer é determinar uma seqüência de ações (ou plano), dados um conjunto de ações, um estado inicial e um estado final; de forma que a conseqüência da execução desta seqüência a partir do estado inicial produza o estado final.

Assim, o raciocínio intra-lógico sobre planos é inviável, se a lógica temporal linear for considerada. Ou seja, a não linearidade do futuro só é possível através do relaxamento da totalidade da relação <. Uma *lógica temporal ramificada* [Za91] admite esta condição de relaxamento.

Nesta subseção foi visto que uma lógica modal temporal é uma lógica cuja linguagem é uma extensão do cálculo de predicados ou proposicional com operadores modais temporais. A clássica semântica dos mundos possíveis [Kr63] foi interpretada no contexto temporal para que cada mundo possível represente um tempo diferente. Assim, cada mun-

do possível é associado com um elemento temporal. Os elementos temporais estão relacionados pela relação de precedência temporal (assim como a relação de acessibilidade em lógica modal [Fi93, Ga91, HC96]) e os operadores modais de possibilidade e necessidade são interpretados sobre tempos futuros. Em alguns casos, o passado também é incluído. Por exemplo, F(*dança(josé, maria)*) significa que, alguma vez, José irá dançar com Maria e P(*dança(josé, maria)*) significa que alguma vez no passado José dançou com Maria.

2.3.3 Lógica temporal *reificada*

Esta lógica é uma tentativa de se construir-se uma lógica de alto poder expressivo para a especificação em lógica de primeira-ordem da verdade sobre asserções. Tratar logicamente conceito ou entidade, seja esta temporal ou não, como termo é tratá-lo como uma coisa (coisificá-lo ou *reificá-lo*). A idéia da *reificação* é formalizar conceitos através de termos utilizando as vantagens da lógica de primeira-ordem como, por exemplo, os métodos automáticos de prova. Coisificar uma lógica implica o uso de meta-linguagem onde uma fórmula na linguagem original (linguagem objeto) torna-se um termo na meta-linguagem.

Na lógica temporal *reificada* [Ga95, MK96] pode-se raciocinar sobre os aspectos particulares da verdade de expressões da linguagem de primeira-ordem através do uso de predicados-verdade, como por exemplo, o predicado *TRUE*. Estes predicados têm como argumentos uma fórmula e uma expressão que denota um objeto temporal. Seu formato é o seguinte: TRUE (<*fato não-temporal, qualificação temporal*>) significando que o primeiro argumento é verdade no tempo denotado pelo segundo argumento. Por exemplo, TRUE(*dança(jose, maria), 10/12/98*). Nesta definição, o tempo admite várias interpretações: José e Maria têm estado dançando o dia todo; ou pelo menos um vez durante o dia ou uma só vez durante o dia.

A teoria geral de ações e tempo de Allen [Al84] e o cálculo de eventos de Kowalsky e Sergot [KS86] podem ser interpretados como uma abordagem *reificada*, embora não seja completamente óbvio qual é a lógica básica porque os autores não provêm uma semântica formal às características temporais. Em cada um destes sistemas há um predicado verdade binário cujo significado é que seu primeiro argumento é verdade durante o intervalo denotado pelo seu segundo argumento.

A teoria de Allen [Al84], na qual se baseia o cálculo de eventos de Kowalsky e Sergot [KS96], distingue três tipos de entidades temporais (*fatos*, *eventos* e *processos*) e as define por restrições entre a verdade da proposição sobre um intervalo e sua verdade sobre subintervalos relacionados a um intervalo.

Fatos correspondem a objetos válidos sobre todo um intervalo de tempo particular. Por exemplo, se o fato F (por exemplo, 'fulano está morto') vale sobre um intervalo A, então ele também vale sobre todos os subintervalos de A:

$$vale(F, A) \Leftrightarrow \forall B\ (B \subset A \Rightarrow vale(F, B))$$

Eventos são objetos com duração fixa, sendo que em subintervalos não pode ocorrer o evento todo. Um exemplo de evento é a proposição: 'a chuva de ontem'.

$$ocorre(e, A) \Rightarrow \forall B\ (B \subset A \wedge B \neg= A \Rightarrow \neg ocorre(e, B))$$

Processos ocorrem como algo intermediário entre *fatos* e *eventos* (por exemplo, 'ontem choveu'). Eles ocorrem em um número substancial de intervalos ou sobre subintervalos maiores que um certo tamanho. Devido a problemas de formalização, Allen declarou que um processo deve estar ocorrendo sobre no mínimo um subintervalo:

$$ocorrendo(p, A) \Rightarrow \exists B(B \subset A \wedge ocorrendo(p, B))$$

As lógicas temporais *reificadas* são uma tentativa de obter maior poder de expressividade do que a lógica com argumentos temporais, mas sem ultrapassar os limites da lógica de primeira-ordem.

O principal problema da lógica com argumentos temporais é seu poder expressivo. O fato do tempo ser introduzido apenas como um argumento torna o formalismo inconveniente para certos tipos de expressões. Por exemplo, em lógica modal temporal, (*F* (*dança*(*jose, maria, tango*)) significa que há um tempo onde José dançará tango com Maria no futuro e (*F*(*P dança*(*jose, maria, tango*)) significa que no futuro esta dança terá acontecido. Este exemplo seria representado na lógica de primeira-ordem com argumentos temporais como a seguir:

$$\exists t(now < t \wedge \exists t'.\ t' < t \wedge dança(jose, maria, tango, t'))$$

que é menos natural do que a representação modal temporal.

Com relação à expressividade, a lógica modal temporal é melhor que a lógica de primeira-ordem com argumentos temporais para alguns tipos de declarações, porém isto pode ser resolvido por um sistema com dois níveis: um nível usuário, baseado na lógica modal temporal, no qual este pode expressar suas declarações, e um nível processamento, baseado na lógica de primeira-ordem com argumentos temporais, no qual as declarações podem ser feitas levando-se em consideração as vantagens da teoria de provas da lógica de primeira-ordem. Um problema resultante é que esta idéia requer o desenvolvimento de um programa de tradução de um formalismo ao outro e vice-versa. Outro inconveniente da lógica de primeira-ordem com argumento temporal é o fato da impossibilidade desta lógica se combinar, de forma incremental, com outra lógica modal.

Segundo Vila [Vi94], as desvantagens da lógica modal temporal são referentes à sua eficiência.

A eficiência da notação da lógica modal temporal a torna adequada para aplicações de entendimento de linguagem "natural", onde é amplamente usada [Ga87]. Entretanto, a área onde provavelmente tem mais aceitação é a Teoria de Programação. A idéia principal é especificar um programa e aplicar métodos de dedução para provar propriedades do programa, como por exemplo: corretude, término, possibilidades de bloqueio etc.

Do ponto de vista puramente lógico, os tratamentos modais das lógicas modais temporais é superior ao da lógica de primeira-ordem com argumentos temporais, assim como da lógica temporal *reificada*. Esta superioridade é decorrente da naturalidade da combinação incremental das lógicas modais temporais com outras lógicas modais.

É importante ressaltar que cada um destes "estilos" de lógica temporal é mais adequado a uma determinada aplicação.

2.4 Bancos de dados temporais

Uma grande variedade de aplicações de banco de dados gerencia dados que variam no tempo. Apesar disso, a tecnologia atual de sistemas gerenciadores de bancos de dados provê pouco suporte para o gerenciamento destes dados temporais, embora os pesquisadores da área de bancos de dados temporais tenham definido formas para modelar, armazenar e recuperar este tipo de dados. A atualização destes dados não foi incorporada por nenhum

sistema gerenciador de bancos de dados, pelo menos até o instante em que este documento estava sendo escrito.

Um banco de dados temporal é um banco de dados que suporta o armazenamento e a recuperação de vários estados de seus objetos.

A pesquisa em bancos de dados temporais tem identificado várias propriedades importantes, incluindo suporte para: tempo de validade, tempo de transação, agregados temporais, indeterminação temporal, granularidades temporais, calendários definidos pelo usuário, reestruturação e *versionamento* de esquemas [Sn95]. O assunto desta tese envolve diretamente propriedades sobre o tempo de validade e sobre a indeterminação temporal.

Várias dimensões temporais têm sido associadas aos fatos armazenados num banco de dados. A associação entre tempos e fatos, independentemente do modelo de dados, será examinada. A mais importante destas dimensões é o tempo de validade ou tempo válido de um fato. O tempo de validade de um fato é o tempo quando o fato foi verdade na realidade modelada [JD98]. Um fato pode ter associado qualquer número de instantes ou intervalos supridos pelo usuário. O tempo de validade descreve os estados variantes no tempo da parte da realidade que se está modelando (universo do discurso).

Todos os fatos têm um tempo de existência no mundo, porém este pode não necessariamente ser armazenado no banco de dados, ou pode ser armazenado mesmo que se tenha um conhecimento parcial deste tempo, ou seja, há indeterminação temporal [DS98]. O tempo de validade pode ser desconhecido (por exemplo, a hora do nascimento de alguém), pode ser permanente (por exemplo, a data de nascimento de alguém), pode ser parcialmente conhecido (por exemplo, fulano nasceu antes de 1970) ou pode simplesmente não interessar num certo contexto (por exemplo, a duração de um casamento).

Outro tipo de tempo que pode ser associado aos fatos de um banco de dados temporal é o tempo de transação. O tempo de transação de um fato é o tempo no qual este tornou-se corrente no banco de dados [JD98].

O uso dos dois tempos (de validade e de transação) se torna importante para aplicações em que ocorrem diferenças significativas entre os tempos quando os fatos realmente ocorreram e os tempos de registro destas ocorrências, sendo que estas diferenças devem ser captadas pelo sistema.

Segundo Chomicki [Ch94], um BDTI é aquele que armazena objetos com indeterminação temporal. Esta definição foi estendida aqui para bancos de dados que armazenam também a informação indeterminada do tipo não temporal, ou seja, a informação imprecisa, omissa ou negativa (seção 6.1).

Uma questão importante na representação do tempo é a granularidade desta, ou seja, o grau de precisão em que o tempo é descrito. Por exemplo, é possível armazenar o ano em que um empregado foi promovido. Todavia, se é mais preciso quando o mês e o dia forem guardados juntamente com o ano. A granularidade do tempo depende da aplicação, ou seja, quanto mais fina a granularidade for, mais precisa será a informação representada.

2.5 Linguagens de consultas a bancos de dados temporais

A arquitetura ANSI/SPARC [SKS99] distingue três níveis diferentes em um banco de dados: físico ou interno, conceitual e externo. No contexto de bancos de dados temporais, Chomicki [Ch94] propõe a divisão do nível conceitual para distinguir entre bancos de dados temporais abstrato e concreto. Intuitivamente, os bancos de dados temporais abstratos capturam o significado formal de um banco de dados temporal, independentemente de sua representação. Já os bancos de dados temporais concretos provêem uma representação específica em termos de um certo modelo de dados temporal. Linguagens de consulta temporais abstratas, tais como a lógica de primeira-ordem, a lógica temporal e o cálculo relacional, são definidas para bancos de dados temporais abstratos, enquanto que linguagens de consulta temporais concretas (TSQL2 [Sn95], SQL/Temporal [Me97] e outras descritas em [TCG+93]) são definidas para bancos de dados temporais concretos.

Nesta seção são destacados o cálculo relacional e a lógica temporal como duas linguagens de consulta abstratas. Exemplos de outras linguagens de consulta, ditas concretas, serão vistos no capítulo 4.

- *Cálculo relacional.* Nesta linguagem de consulta temporal abstrata, a resposta a uma consulta de primeira-ordem é o conjunto de valores que tornam a fórmula da consulta verdadeira no banco de dados consultado. Por exemplo, considere a consulta sobre o banco de dados, representado pela relação *INDEP* (TAB. 1), "liste todos os países que perderam e recuperaram sua independência". Esta consulta pode ser formulada

em lógica de primeira-ordem como a seguir:

$$\exists t_1\ t_2\ t\ s_1\ s_2\ \forall s\ INDEP(x,\ s_1,\ t_1) \wedge INDEP(x,\ s_2,\ t_2) \wedge \neg INDEP(x,\ s,\ t) \wedge t_1 < t < t_2$$

onde x é a variável livre que conterá os nomes dos países que satisfazem a consulta.

- *Lógica temporal.* Esta linguagem de consulta temporal abstrata é uma extensão temporal da linguagem de primeira-ordem e contém os operadores temporais *since* e *until* com os seguintes significados:

  - *A since B* é verdade no instante i, se e somente se para algum $j, j < i$, B é verdade no instante j, e para todo $k, j < k \leq i$, A é verdade no instante k.

  - *A until B* é verdade no instante i, se e somente se para algum j, $i < j$, B é verdade no instante j, e para todo k, $i < k \leq j$, A é verdade no instante k.

  A notação "$\blacklozenge A$" (alguma vez no passado A) é usada para denotar "true *since* A" e a notação "$\Diamond A$" (alguma vez no futuro A) para denotar "true *until* A". Por exemplo, a mesma consulta anterior ("liste todos os países que perderam e recuperaram a independência") pode ser formulada em lógica temporal como a seguir:

$$\exists s_1,\ s_2\ \blacklozenge(\blacklozenge INDEP(x,\ s_1) \wedge \Diamond INDEP(x,\ s_2) \wedge \forall s\ \neg INDEP(x,\ s))$$

Um possível método de implementação para consultas lógicas temporais é traduzi-las em consultas na lógica de primeira-ordem e utilizar as implementações já existentes para estas. Outra solução, proposta por Lipeck [LS87], usa máquinas de estado para controlar os operadores temporais.

TABELA 1 - Relação INDEP

País	Capital	Anos de Independência
Reino Tcheco	Praga	{1198,...,1620}
Tchecoslováquia	Praga	{1918,...,1938} ∪ {1945,...1992}
Repúb. Tcheca	Praga	{1993,...}
Eslováquia	Bratislava	{1939,...,1945} ∪ {1993,...}
Polônia	Guiezno	{1025,...,1039}
Polônia	Cracóvia	{1040,...,1595}
Polônia	Varsóvia	{1596,...,1794} ∪ {1918,...,1938} ∪ {1945,...}

Chomicki [Ch94] discute as limitações das linguagens de consultas temporais à

época, com ênfase em cinco propriedades desejáveis a estas, incluindo o tratamento da informação temporal incompleta. Segundo ele, a semântica de um banco de dados temporais incompleto é dada por um conjunto X de bancos de dados temporais abstratos. Uma resposta a uma consulta q, feita sobre X, é uma relação incompleta representando o conjunto de relações, correspondentes às respostas a q, obtidas separadamente para membros individuais de X. Adicionalmente, consultas modais podem requerer, por exemplo, quais fatos são certos (verdadeiros em todos os membros de X) e quais fatos são possíveis (verdadeiros em algum membro de X).

Capítulo 3

Informação indeterminada em bancos de dados

"Tudo é incerto e derradeiro."
Fernando Pessoa, NEVOEIRO

Este capítulo trata da informação indeterminada especificamente em Bancos de Dados. Há outras definições na área de Inteligência Artificial. Também é vista a noção de mundos possíveis e mostradas duas restrições ao relacionamento entre estes e a teoria de bancos de dados. Três classificações de informação não-temporal indeterminada são explicadas, na tentativa de distinguir tipos de indeterminação que podem ocorrer em bancos de dados. A informação temporal indeterminada ou indeterminação temporal também é abordada através da conceituação de instante e intervalo indeterminados.

Nesta tese os termos *indeterminado(a)* e *indeterminação* são usados para referenciar a informação imperfeita [Pa96] em geral, seja esta temporal ou não. Tal escolha foi influenciada pelo glossário de conceitos de bancos de dados temporais elaborado por Jensen e Dyreson [JD98], onde são definidos termos como *indeterminada no tempo* para caracterizar a informação do tipo "não se conhece exatamente quando" e *determinada no tempo* para indicar que o objeto tem um tempo associado que é conhecido precisamente. Eles também definiram duas variantes do primeiro: *instante indeterminado* e *intervalo indeterminado*. Outros termos utilizados são os seguintes: informação incompleta, parcial, imperfeita, vaga, nebulosa, disjuntiva, imprecisa e desconhecida [Pa96].

3.1 Mundos possíveis

Um banco de dados modela alguma parte do mundo real. Se a informação disponível é completa, então há uma clara correspondência entre o banco de dados e o mundo real. Quando o conhecimento sobre o mundo for incompleto, vários cenários ou estados com conhecimento completo podem ser possíveis, porém é desconhecido qual destes representa

o estado real do mundo. Um banco de dados com informação indeterminada implicitamente representa um conjunto de estados possíveis ou de mundos possíveis. Um mundo possível é um estado hipotético do mundo que pode ser representado por um banco de dados convencional com informação completa. Se o conhecimento do mundo for representado por uma teoria lógica, cada mundo possível corresponde a um modelo da teoria.

Uma restrição muito utilizada nos bancos de dados convencionais é a suposição do mundo fechado ou *Closed World Assumption* (CWA) [Re78]. De acordo com a CWA, toda informação não explicitamente representada no banco de dados é assumida ser falsa. Neste caso, diz-se que o banco de dados é visto como um mundo fechado, significando que se supõe que este tem toda informação positiva. Sob a CWA, é possível usar a ausência de informação para inferir que ela é falsa.

Com a CWA é difícil descrever informação incompleta e mundos possíveis. Uma alternativa é a suposição do mundo aberto ou *Open World Assumption* (OWA) [KW85]. Ela declara que a teoria é correta, mas não necessariamente completa, ou seja, uma declaração é conhecida ser falsa em todos os mundos possíveis, somente se sua negação puder ser logicamente derivada. Logo, quando a OWA é empregada, não se toma a ausência de informação para inferir que esta é falsa. Ao contrário, sob a OWA uma informação só é falsa se sua negação é dedutível do banco de dados, senão esta é assumida ser desconhecida. Por exemplo, não se pode inferir que José não está fazendo a disciplina *banco de dados temporais*, pelo simples fato dele não constar na lista dos matriculados na disciplina. Quando o banco de dados é visto como um mundo aberto (OWA), não se faz quaisquer suposições sobre a informação que não está armazenada.

Várias formas intermediárias destas suposições foram propostas na literatura [ZP97].

3.2 Classificação de informação indeterminada

A informação contida num banco de dados é determinada se ela representa com precisão o domínio de aplicação correspondente no mundo real. Entretanto, nem sempre é possível obter todas as informações com a precisão desejada. Uma definição de informação indeterminada é a seguinte: "um item de informação num banco de dados é indeterminado se a informação correspondente no mundo real é conhecida apenas imperfeitamente e este

item contém este conhecimento parcial".

Há na literatura várias classificações dos tipos de informação indeterminada, entretanto nenhuma delas foi definida como padrão. Umas são mais extensivas que outras, porém sempre há tipos comuns em sua essência. As classificações de Bosc e Prade [BP93], de Parson [Pa96] e de Levene e Loizou [LL99] serão destacadas a seguir, pelo fato de envolverem definições que abrangem vários tipos de informação indeterminada.

Segundo Bosc e Prade [BP93] há quatro tipos de imperfeição: incerteza, imprecisão, vagueza e inconsistência. A *incerteza* surge da ausência de informação sobre o estado do mundo. Isto torna impossível determinar se certas declarações sobre o mundo são verdadeiras ou falsas, porém pode-se estimar o grau de certeza da declaração ser verdadeira ou falsa. A *imprecisão* pode surgir da linguagem usada para declarar imprecisões. Por exemplo, a declaração 'Paulo tem 26 anos' é precisa apenas no caso de não se estar interessados na idade exata de Paulo em termos de anos e meses. A *vagueza* é semelhante a definição de imprecisão, porém neste caso usam-se valores nebulosos (ou valores *fuzzy*). Por exemplo, 'Paulo é jovem'. A *inconsistência* descreve a situação onde há dois ou mais valores conflitantes para uma variável; por exemplo 'Paulo tem 28 anos' e 'Paulo tem mais de 30 anos'. Assim, o casamento destas declarações para se obter um valor consensual é impossível, visto que não há um valor consistente para ambas. A inconsistência também surge de mais de uma fonte de informação. É prática comum detectar inconsistências quando da atualização do banco de dados e assim evitá-las, tarefa esta geralmente realizada pelos algoritmos que mantêm as restrições de integridade do banco de dados e que são definidos no esquema conceitual.

Três tipos de *informação imperfeita* são distinguidas por Parsons [Pa96]. A *informação incompleta* significa a ausência de algum valor, e é dada por um "valor nulo". Isto tem dois possíveis significados: o valor existe, mas é desconhecido, ou o valor não existe. Por exemplo, um valor nulo no atributo *número-telefone* pode significar que o objeto tem um número de telefone desconhecido ou ele não tem telefone. A *informação imprecisa* ocorre quando se tem alguma informação, mas a informação não é precisa. A imprecisão pode ser um intervalo valorado (por exemplo: idade entre 20 e 25, salário menor que 1.000, nome é Pedro ou Paulo) ou um valor *fuzzy* (por exemplo: idade é "jovem"). Finalmente, a *informação incerta* associa uma estimação de validade aos valores.

Segundo Levene e Loizou [LL99] são quatro os tipos de informação indeterminada. No primeiro tipo, alguma informação no banco de dados pode estar omissa. A *informação omissa* geralmente engloba duas categorias: a informação omissa aplicável e a inaplicável. A informação omissa aplicável é quando o valor de um atributo existe, porém é desconhecido. Por exemplo, o número de telefone de José é desconhecido. A informação omissa inaplicável é quando o valor de um atributo não se aplica a certo objeto. Por exemplo, o nome da esposa de José não existe porque ele é solteiro. Este caso pode ser caracterizado como excesso de atributos de certos objetos, e não um caso de informação indeterminada. Nos dois casos, a informação omissa pode ser modelada por valores especiais, chamados valores nulos, que agem como marcadores da informação omissa. Há várias interpretações dos valores nulos nestas duas categorias [ZP97].

O segundo caso envolve a modelagem de *informação disjuntiva* que pode ser vista como um caso especial de informação omissa aplicável. Por exemplo, o número de telefone de José pode ser 333-1000 ou 333-1001, mas não se sabe exatamente qual é o correto. A informação disjuntiva pode ser modelada por um conjunto finito de valores, chamado de *OR-set*, sendo que um destes valores é o valor verdadeiro. Logo, o número do telefone de José é um membro do *OR-set* {333-1000, 333-1001}. Outro tipo é aquele que pode ser modelado por um conjunto infinito de valores. Por exemplo, o salário de José pode ser maior ou igual a R$ 2.000,00. O valor real do salário de José é um membro do *OR-set* {2000, 2001, ...}.

A *informação nebulosa* é o terceiro caso. Nele, a pertinência de um valor de atributo pode ser nebulosa, ou seja, a pertinência pode ser um número no intervalo [0, 1] ou um valor lingüístico (baixa, média ou alta). Por exemplo, a estatura de José pode ser armazenada como média. Conjuntos nebulosos são aplicáveis para modelar situações onde há incerteza sobre a pertinência de elemento num conjunto. Por exemplo, é possível saber com um grau de certeza de 0.7 que José é um elemento no conjunto dos de estatura média.

O último tipo de indeterminação trata da abordagem probabilística de informação incompleta, por acoplar a cada valor de atributo, uma probabilidade entre 0 e 1, de acordo com uma distribuição conhecida para aquele domínio do atributo. Esta abordagem permite que se use inferência estatística durante o processamento de consultas, para se obter respostas aproximadas.

3.3 Indeterminação temporal

Um tema recente, catalisador da representação e do tratamento da informação indeterminada por um lado e da informação temporal por outro lado, é a indeterminação temporal, ou seja, quando o objeto tem um tempo associado, mas esse tempo não é conhecido com precisão. Normalmente a indeterminação temporal ocorre com o tempo de validade. A indeterminação no tempo de transação só ocorreria quando houvesse uma perda de dados parcialmente recuperados de arquivos de segurança.

Um usuário pode saber só aproximadamente quando um fato ocorreu. Por exemplo, alguém pode saber que algo aconteceu "entre 14 e 17h", "alguma vez na semana passada" ou "depois do primeiro bimestre do corrente ano". Estes são exemplos de indeterminação no tempo de validade. Tal informação pode ser caracterizada como uma informação do tipo "não sabemos quando" ou, mais precisamente, "não sabemos exatamente quando". Uma de suas fontes é quando geralmente as ocorrências temporais são imprecisas ou omissas. Por exemplo, a data de admissão de José pode ser omissa ou parcialmente conhecida se ele foi admitido entre maio e junho passados.

Declarações que possuem um referência relativa ao tempo são sempre indeterminadas no tempo. Por exemplo, "José trabalha na UFPb desde antes de 1990" contém uma referência relativa ao tempo e é indeterminada no tempo.

Um instante indeterminado é conhecido estar localizado em um intervalo que delimita o conjunto de tempos possíveis [JD98]. A informação indeterminada no tempo poderia ser mais substancial se, por exemplo, um instante indeterminado tivesse uma função de probabilidade associada. Esta função especificaria a probabilidade que o instante tenha ocorrido no intervalo. Por exemplo, "entre 01 de março e 05 de março" é um instante indeterminado onde é possível ter uma função de probabilidade especificando que o dia 03 de março é 50% mais provável de ser o instante determinado. Dependendo da aplicação, é possível usar uma probabilidade uniforme, ou seja, aquela que estabelece que todos os instantes são igualmente prováveis de ser o instante determinado.

Um intervalo indeterminado é aquele delimitado por no mínimo um instante indeterminado [JD98]. Por exemplo, o intervalo [01/01/1999, *after* 31/12/2000] é indeterminado, pois possui o instante indeterminado *after* 31/12/2000 como seu limite superior.

Capítulo 4

Trabalhos relacionados

"Nada existe.
As coisas todas vão se tornando."
Provérbio chinês

Neste capítulo são abordados os trabalhos mais importantes que se relacionam ao tratamento da informação indeterminada, tanto no tempo, quanto na informação não-temporal. Três formalismos envolvendo a representação e a consulta de informação indeterminada são exibidos: uma extensão da teoria relacional para tratar valores nulos e informação disjuntiva [Re94], uma abordagem lógica para representação da informação não-temporal indeterminada [DS98]; e uma lógica de bases de conhecimento incompleto [Le94]. Também são tratadas algumas linguagens de consulta a bancos de dados temporais representativas do estado da arte: TSQL2 [Sn95], uma extensão temporal a SQL-92 proposta por Snodgrass [Sn00] e SQL/Temporal [Me97].

4.1 Uma Teoria Relacional Generalizada para Informação Indeterminada

No capítulo 3 foram discutidos, genericamente, os vários tipos de informação indeterminada, sem qualquer preocupação com seu embasamento formal. Esta seção apresenta, numa abordagem lógica, formas de se representar e de se consultar bancos de dados relacionais [Co79, SKS99] que contenham dois tipos de informação indeterminada: a informação disjuntiva e os valores nulos.

O modelo de dados relacional [Co79], embora amplamente utilizado hoje em dia, necessita de soluções para fenômenos tais como a informação disjuntiva e os valores nulos. Por exemplo, para representar fatos disjuntivos da forma "P é o caso, ou Q é o caso, ou ..., mas não se sabe qual deles realmente o é" e para usar esta forma na derivação de respostas a consultas.

Reiter [Re94] introduz uma reconstrução teórica do modelo relacional.

Seja a seguinte teoria relacional, apresentada aqui de forma incompleta (sem sua linguagem e semântica) apenas com o intuito de mostrar uma formalização para informação indeterminada, para um banco de dados contendo as seguintes relações: Peça = {(p_1), (p_2), (p_3)}, Fornecedor = {(f_1), (f_2)}, Fornecimento ={(f_1, p_1), (f_2, p_2)} e Sub-peça = {(p_1, p_2)}. Uma interpretação para este banco de dados pode ser especificada por um conjunto de fórmulas atômicas: {Peça(p_1), Peça(p_2), Peça(p_3), ..., Fornecimento(f_1, p_1), Fornecimento(f_2, p_2), ..., Sub-peça(p_1, p_2)}.

A teoria de primeira-ordem consiste das seguintes fórmulas:

1) Axioma de fechamento de domínio:

$$(x)\ [=(x, p_1) \vee =(x, p_2) \vee \ldots \vee =(x, f_1) \vee =(x, f_2)]$$

2) Axiomas de Nome Único:

$$\neg=(p_1, p_2),\ \neg=(p_2, p_3),\ \ldots$$

3) Axiomas de Igualdade: reflexividade, comutatividade, transitividade e princípio da substituição de termos iguais

4) Axiomas de Complemento:

4.1) $(x)\ [\text{Peça}(x) \rightarrow =(x, p_1) \vee =(x, p_2) \vee =(x, p_3)]$

4.2) $(x)\ [\text{Fornecedor}(x) \rightarrow =(x, f_1) \vee =(x, f_2)]$

4.3) $(x)(y)[\text{Fornecimento}(x, y) \rightarrow$

$$=(x, f_1) \wedge =(y, p_1) \vee =(x, f_2) \wedge =(y, p_2)]$$

4.4) $(x)(y)[\text{Sub-peça}(x, y) \rightarrow =(x, p_1) \wedge =(y, p_2)]$

Supondo que se deseja representar o seguinte novo fato disjuntivo: "f_2 fornece p_1 ou f_2 fornece p_3, mas não é conhecida qual destas.". Esta disjunção pode ser representada pela fórmula: Fornecimento(f_2, p_1) $\vee$ Fornecimento(f_2, p_3). Para acomodar esta nova fórmula, considera-se antes a contra-positiva do axioma 4.3:

$$(x)(y)[[\neg=(x, f_1) \vee \neg=(y, p_1)] \wedge [\neg=(x, f_2) \vee \neg=(y, p_2)]] \rightarrow \neg\text{Fornecimento}(x, y)].$$

A partir deste axioma e dos axiomas de nome único, se pode provar que ¬Forne-

cimento(f₂, p₁). Similarmente, se pode provar que ¬Fornecimento(f₁, p₃). Porém, estes dois fatos são inconsistentes com o novo fato disjuntivo pela seguinte razão: o axioma 4.3 foi projetado para declarar que as únicas instâncias possíveis de Fornecimento são (f₁, p₁) e (f₂, p₂). Todavia, a nova fórmula declara que há outras instâncias possíveis de Fornecimento, ou seja, as instâncias (f₂, p₁) e (f₂, p₃). Para acomodar estas, deve-se substituir o axioma (4.3) pelo seguinte axioma:

(4.3'): (x)(y)[Fornecimento(x, y) →
=(x, f₁) ∧ =(y, p₁) ∨ =(x, f₂) ∧ =(y, p₂) ∨
=(x, f₂) ∧ =(y, p₁) ∨ =(x, f₂) ∧ =(y, p₃)]

Isto leva a uma nova teoria consistindo das definições das relações, dos axiomas 1, 2, 3, 4.1, 4.2, 4.3' e 4.4, além da fórmula: Fornecimento(f₂, p₁) ∨ Fornecimento(f₂, p₃). Esta teoria provê uma representação intuitivamente correta deste mundo incompletamente especificado, levando a uma teoria relacional generalizada cujas propriedades são descritas em [Re94].

Para acomodar o valor nulo do tipo "existe, mas está desconhecido", suponha que se queira representar o seguinte fato: "Alguém fornece a peça p_4". Esta representação pode ser especificada pela fórmula (∃x) Fornecedor(x) ∧ Fornecimento(x, p₄) que assegura a existência de um indivíduo *x* com as propriedades desejadas. Se este indivíduo for denotado pela constante de *Skolem* α [Ch80], representando o valor nulo, então se pode eliminar o quantificador existencial. Esta constante pode denotar f₁ ou f₂ ou um outro fornecedor desconhecido. Para acomodar esta constante, os axiomas de nome único permanecem os mesmos, enquanto que o axioma de fechamento de domínio deve ser modificado:

1') (x) [=(x, p₁) ∨ =(x, p₂) ∨ ... ∨ =(x, f₁) ∨ =(x, f₂) ∨
=(x, α)]

Da mesma forma, os axiomas de complemento 4.2 e 4.3 devem ser alterados:

4.2') (x) [Fornecedor(x) → =(x, f₁) ∨ =(x, f₂) ∨ =(x, α)]

4.3') $(x)(y)[\text{Fornecimento}(x, y) \rightarrow =(x, f_1) \wedge =(y, p_1) \vee =(x, f_2) \wedge =(y, p_2) \wedge =(x, \alpha) \vee =(y, p_4)]$

Finalmente, os fatos $\text{Fornecedor}(\alpha)$ e $\text{Fornecimento}(\alpha, p_4)$ devem ser acrescidos à teoria recém-modificada.

Note que a ausência do axioma de nome único para a constante α é o que a distingue das outras constantes (f_1, f_2, p_1, ...). Note também que se pode provar $\neg\text{Fornecimento}(f_1, p_2)$ e $\neg\text{Fornecimento}(f_2, p_1)$, porém não se pode provar $\neg\text{Fornecimento}(f_1, p_4)$ ou $\neg\text{Fornecimento}(f_2, p_4)$, o que intuitivamente é desejável.

Como última observação, suponha que se queira representar o fato que "algum fornecedor fornece a peça p_2": $(\exists x)\ \text{Fornecedor}(x) \wedge \text{Fornecimento}(x, p_2)$. Isto leva à escolha de uma nova constante para distinguir este fornecedor (α') do fornecedor anteriormente desconhecido (α). Logo, os axiomas 1', 4.2' e 4.3' devem ser modificados para acomodar α', ou seja, cada novo valor nulo introduzido na teoria modifica os axiomas de complemento e o axioma de fechamento de domínio.

Nesta seção foi visto que, de um ponto de vista lógico, um banco de dados, armazenando valores disjuntivos e valores nulos, pode ser formalizado por meio de uma teoria relacional estendida com valores nulos [Re94, ZP97]. Esta teoria permite também a representação de fatos do tipo: "Alguém fornece a peça p_3 e não é o fornecedor f_3, nem o fornecedor f_4", ou seja, $(\text{Fornecedor}(\alpha) \wedge \text{Fornecimento}(\alpha, p_3) \wedge \neg=(x, f_3) \wedge \neg=(x, f_4)$.

4.2 A Lógica de Bases de Conhecimento Incompleto

Nesta seção serão descritas algumas questões de representação formal para uso de bases de conhecimento incompleta proposta por Levesque [Le94]. Uma base de conhecimento (BC) incompleta é aquela que possui apenas conhecimento parcial do domínio da aplicação representada, ou seja, ela não armazena todos os dados necessários para responder uma consulta. Neste caso, o usuário deve distinguir entre o que é conhecido pela BC e o que é verdadeiro na aplicação modelada. Por exemplo, uma BC pode representar o fato de que alguém é casado, sem precisar saber com quem. Logo, qualquer interação com a BC

deve ser baseada no entendimento de que esta pode ter apenas conhecimento parcial de certos aspectos relevantes da aplicação.

Tipicamente, uma BC pode ser dividida em duas partes. Uma destas partes contém os fatos do domínio da aplicação, como `empregado(pedro)`, `empregado(maria)` e `gerencia(augusto, pedro)`, e a outra parte contém regras, como $\forall x$ `¬gerencia(x, x)` ou $\forall x$ `[gerente(x)` $\equiv \exists y$ `gerencia(x, y)]`. A primeira regra pode ser usada como uma restrição de integridade e a segunda para deduzir quem são os gerentes.

A importância de BCs incompletas se deve ao fato de que, em várias aplicações, a BC experimenta uma evolução contínua. Em cada etapa desta evolução, a informação adquirida pela BC pode ser incompleta. Devido a esta incompletitude, certos sistemas de apoio à decisão não podem esperar que a BC se estabilize numa forma final para só assim poder consultá-la, uma vez que isto pode não acontecer.

Apesar de um sistema baseado em conhecimento, dependente de uma BC incompleta, ter sua capacidade de resolução de problemas comprometida, isto pode ser superado, em alguns casos, pelo uso de raciocínio por *default* [Po94] ou por heurísticas especiais [CWA+75]. Um sistema pode "descobrir" onde o conhecimento está ausente, antes de decidir o que fazer com este conhecimento. Segundo Levesque, isto sugere que uma BC deve capacitar o provimento de informação não apenas da área de aplicação, como também de si própria. Portanto, a linguagem para interação com uma BC deve possibilitar ao usuário a definição e a consulta sobre o que é e o que não é conhecido pela BC.

Uma BC incompleta também pode ser vista como um conjunto de fórmulas da lógica de primeira-ordem. Uma consulta a uma base deste tipo é qualquer fórmula da lógica de primeira-ordem e é respondida pela requisição da relação de *provabilidade* (|-) para determinar o que é e o que não é derivado a partir da BC. Para cada consulta α, há três respostas possíveis:

`Sim, quando BC |-` α

`Não, quando BC |-` $\neg\alpha$

`Desconhecido, quando nem BC |-` α `nem BC |-` $\neg\alpha$

Uma BC completa representa por completo o domínio modelado. Nenhuma consulta a uma BC deste tipo obtém o valor *desconhecido* como resposta. Ao contrário, uma

BC incompleta permite que consultas obtenham respostas desconhecidas. Por exemplo, considere uma BC contendo [empregado(pedro) ∨ empregado(maria)], mas que não contenha empregado(pedro), empregado(maria), ¬empregado(pedro) nem ¬empregado(maria). Esta BC descreve apenas parcialmente um mundo possível, pois especifica que Pedro ou Maria são empregados, mas não particulariza quem de fato o é. Logo, a veracidade da consulta "Pedro está empregado?" não pode ser determinada pelo que está disponível nesta BC, ou seja, esta consulta obtém *desconhecido* como resposta.

A linguagem KL

Sendo uma BC uma descrição incompleta que comporta um conjunto de mundos possíveis, Levesque propôs o operador modal *K* (do inglês *Known*) abrangendo uma fórmula α (*K*α) que é interpretada como: "a fórmula α é conhecida pela BC". A avaliação de uma consulta para verificar se α é conhecida pela BC, ou seja, "*K*α?", eqüivale a verificar se α é satisfeita por todos os mundos possíveis descritos pela BC. Por exemplo, para descobrir se a BC possui uma lista incompleta de empregados, se deve verificar se a fórmula (∃x) [empregado(x) ∧ ¬*K*[empregado(x)]] é derivada da BC. Da mesma forma, a consulta ∃x empregado(x) pode ser usada para verificar se há empregados, enquanto que a consulta ∃x *K*[empregado(x)] requer os empregados conhecidos pela BC.

Uma linguagem, chamada KL, é proposta por Levesque para se referir ao estado corrente de uma BC. Esta linguagem adiciona aos axiomas da lógica de primeira-ordem, fórmulas do tipo *K*α. Isto permite que se consulte uma BC considerando sua incompletitude. Uma forma de se avaliar uma consulta é a seguinte: BC |= α onde |= é uma relação de *provabilidade* e α é qualquer fórmula de KL, possivelmente contendo operadores *K*. Por exemplo, para descobrir se uma BC possui uma lista incompleta de empregados, se deve pesquisar

BC |= (∃x) [empregado(x) ∧ ¬*K*[empregado(x)]]

Da mesma forma, a consulta (∃x) empregado(x) pode ser usada para pesquisar se há empregados, enquanto que a consulta (∃x) *K*[empregado(x)] procura quem são os empregados conhecidos pela BC.

A linguagem KL possui as mesmas regras de formação da lógica de primeira-ordem, mais a seguinte regra: se $\alpha \in$ KL então $K\alpha \in$ KL. Por conseguinte, há dois tipos de fórmulas em KL: a primeira é do tipo $p(c) \wedge (\exists x)\ q(x)$, onde p e q são predicados e c é uma constante, será verdadeira ou falsa dependendo da interpretação de p, q e c. A segunda é do tipo $K[p(c) \wedge (\exists x)\ q(x)]$ e será verdadeira ou falsa dependendo do que é e do que não é conhecido pela BC.

A interpretação semântica de um fórmula de KL dependerá de uma descrição do mundo e de uma descrição de uma BC. Uma BC pode ser vista como uma descrição parcial de um mundo e pode ser caracterizada por um conjunto de descrições do mundo. Se uma BC for caracterizada por um conjunto *m* de descrições do mundo, então vale o seguinte: a fórmula $K\alpha$ é verdadeira na BC descrita por *m*, se e somente se a fórmula α for verdadeira em todo mundo descrito por um elemento de *m*. Uma descrição do mundo precisa atribuir um valor verdade para sentenças atômicas, a fim de determinar o valor verdade de todas as sentenças da lógica de primeira-ordem. Logo, é possível definir o conjunto *W* de descrições do mundo como $W = \{[\text{átomos} \rightarrow V, F]\}$. Uma descrição de uma BC é um conjunto não-vazio de descrições do mundo *m* onde o conjunto *M* de descrições da BC é definido por: $M = \{m \subseteq W \mid m \neq \varnothing\}$. A interpretação de qualquer fórmula α é feita pela função ϕ, onde $\phi \in \{[KL \times W \times M \rightarrow V, F]\}$, é definida por

$\phi(\alpha, w, m) = w(\alpha)$ onde α é uma fórmula atômica.

$\phi(\neg\alpha, w, m) = V$ sse $\phi(\alpha, w, m) = F$.

$\phi([\alpha \vee \beta], w, m) = V$ sse $\phi(\alpha, w, m) = V$ ou $\phi(\beta, w, m) = V$.

$\phi((\exists x)\alpha, w, m) = V$ sse para alguma constante c, $\phi(\alpha_c^x, w, m) = V$.

$\phi(K\alpha, w, m) = V$ sse para todo w', $\phi(\alpha, w', m) = V$

A axiomatização da linguagem KL é a seguinte:

Axiomas

1) Todos os axiomas da lógica de primeira-ordem

2) $K\alpha$ onde α é um axioma da lógica de primeira-ordem

3) $K(\alpha \rightarrow \beta) \rightarrow (K\alpha \rightarrow K\beta)$

pertinentes à linguagem TSQL2, mas devem ser pesquisados pelos que dela fizerem uso.

Os principais conceitos abordados em TSQL2 são comentados a seguir, com ênfase maior na indeterminação temporal por ser este um conceito mais explorado nesta tese.

4.3.1 Tempo em TSQL2

Um instante é modelado por uma marcação temporal (*timestamp*) associada com uma escala (por exemplo: dia, mês, ano). Uma *timestamp* é um valor de tempo associado com uma tupla ou com um valor de atributo e pode ser especializado para *timestamp tempo de validade*, *timestamp tempo de transação*, *timestamp intervalo* etc. [JD98].

Um período é uma duração temporal, ou seja, uma porção de tempo delimitada por dois instantes, e é modelado pela composição de dois instantes, sendo que o instante inicial é menor ou igual ao instante final.

4.3.2 Tipos de dados

Além dos tipos de dados *data* ('01-12-1998') e *intervalo* ('durante 3 semanas') de SQL-92, tem-se o *período* (de '01-01-1998' até '31-07-1998') com variação e precisão especificáveis.

Um dado chamado substituto (*surrogate*) é introduzido em TSQL2. *Surrogates* são identificadores únicos invisíveis aos usuários que são usados na identificação de objetos associados com atributos com tempo variável, porém não substituem as chaves identificadoras.

Um *elemento temporal* é a união de períodos disjuntos usado para marcar *tabelas estado* (seção 4.3.3).

4.3.3 Tabelas tempo de validade

TSQL2 permite, além das tabelas convencionais de SQL-92, as *tabelas estado tempo de validade* onde cada tupla é marcada com um *elemento temporal*. Por exemplo, a ta-

bela Empregado com atributos *Nome*, *Salário* e *Gerente* pode conter a tupla (Paulo, 1000, Maria). A marcação do *elemento temporal* determina todos os períodos nos quais Paulo foi gerenciado por Maria e recebia $1000. A informação sobre outros valores do salário de Paulo ou outros gerentes seria armazenada em outras tuplas. A marcação é implicitamente associada com cada tupla e não se constitui em uma nova coluna na tabela. A variação, a precisão e a indeterminação das marcações podem ser especificadas.

TSQL2 também permite *tabelas evento tempo de validade* que são aquelas onde cada tupla é marcada com um conjunto de instantes. Por exemplo, a tabela Promoção com atributos *Nome* e *Cargo* pode conter a tupla (Maria, Gerente). A marcação *conjunto de instantes* gravaria os instantes quando Maria foi promovida a gerente. A marcação também é implicitamente associada com cada tupla.

4.3.4 Tabelas tempo de transação

O tempo de transação especifica quando a tupla foi armazenada no banco de dados. Se a tupla (José, 1.000, Maria) foi armazenada em 15/03/95 (com um comando *APPEND*) e removida em 01/06/96 (com um comando *DELETE*), então o tempo de transação seria o período de 15/03/95 a 01/06/96.

Existem ao todo seis tipos de tabelas em TSQL2: as tabelas convencionais (sem suporte temporal), as *tabelas estado tempo de validade* (com tuplas marcadas com elementos tempo de validade), as *tabelas evento tempo de validade* (com tuplas marcadas com conjuntos de instantes tempo de validade), as *tabelas tempo de transação* (com tuplas marcadas com elementos tempo de transação), as *tabelas estado bitemporal* (com tuplas marcadas com tempo de validade e tempo de transação) e as *tabelas de evento bitemporal* (com tuplas marcadas com conjuntos de instantes tempo de validade e tempo de transação).

4.3.5 Reestruturação e *versionamento* de esquemas

Os *elementos temporais* associados com tuplas com valores idênticos podem ser combinados. Por exemplo, para determinar quando Paulo ganhava $1000, independente de quem o gerenciava, a tabela Empregado seria reestruturada sobre as colunas *Nome* e *Salário*. A marcação da tupla reestruturada especificaria os períodos quando Paulo ganhava

S1000, informação esta vinda de várias tuplas básicas especificando gerentes distintos.

O *versionamento* de esquemas especifica a evolução do esquema, ou seja, quando o esquema é alterado, os esquemas anteriores são acessíveis graças ao suporte do tempo de transação.

4.3.6 Indeterminação temporal

É comum que alguém saiba apenas aproximadamente quando um evento ocorreu. Por exemplo, alguém pode saber que um evento aconteceu "entre 4 e 6 da tarde" ou "alguma vez na primeira semana de janeiro". Estes eventos usam indeterminação temporal e podem ser caracterizados como utilizando informação do tipo "não sei exatamente quando".

O modelo de tempo adotado em TSQL2 assume as seguintes propriedades: uma linha é segmentada, cada ponto na linha é chamado um instante, o tempo entre dois instantes é conhecido como um período e o tamanho da linha, ou seja, a quantidade qualquer de segmentos desta, é um intervalo. Cada um destes segmentos é chamado um *chronon*. Um *chronon* é a menor quantidade temporal representável na implementação. A título de exposição do assunto desta seção, é assumido que os *chronons* são consecutivamente rotulados pelos números inteiros da seqüência 0, ..., N, onde N é o número de valores representáveis. Também é assumido que um *chronon* é um dia.

As representações de instantes, intervalos e períodos indeterminados usadas em TSQL2 são resumidas a seguir.

Instantes indeterminados

Um instante é determinado se é conhecido em que ponto no tempo ele está localizado. Se sua localização é omissa, então o instante é indeterminado. A indeterminação refere-se à localização do instante e não se o instante existe [DS98]..

Um instante indeterminado em TSQL2 é descrito por três parâmetros: um limite inferior, um limite superior e uma função denominada *p.m.f.* (probability mass function). Os limites são instantes que delimitam o período em que o instante está localizado; o instante

não está antes do limite inferior nem depois do limite superior. Denotam-se α_* e α^* para o limite inferior e para o limite superior do instante α, respectivamente. Entre os limites está situado um período de indeterminação o qual identifica um conjunto contíguo de *chronons* possíveis. O instante real é algum pertencente a este conjunto. Denota-se um período de indeterminação por $\alpha_* \sim \alpha^*$. Por exemplo, 1/5 ~ 31/5 é um período de indeterminação representando o mês de maio.

Há situações em que os instantes num conjunto de instantes possíveis podem não ser igualmente prováveis, ou seja, alguns instantes são mais prováveis que outros. A função *p.m.f.* determina a probabilidade que cada instante tem de ser o instante real. Esta função, denotada por $P\alpha$, é dada pela seguinte igualdade:

$$P\alpha(i) = Pr[\alpha = i],\ i \in \{0, \ldots, N\}$$

onde α é o instante indeterminado e $Pr[\alpha = i]$ é a probabilidade de α estar localizado no *chronon* i.

Um usuário pode não conhecer a função básica porque a informação está indisponível ou porque a função excede as capacidades de implementação do sistema. Em tais casos, a distribuição é considerada uniforme. Esta representa a perda completa de conhecimento sobre a distribuição. É um tipo de incompletitude de segunda-ordem, ou seja, a distribuição uniforme é uma informação incompleta sobre a informação indeterminada.

Períodos indeterminados

Um período delimitado por instantes indeterminados é chamado período indeterminado [DS98]. Este começaria durante qualquer membro do conjunto de instantes possíveis do instante inicial. Semelhantemente, um período de indeterminado acabaria num dos membros do instante indeterminado final. Como os instantes inicial e final são conhecidos apenas parcialmente, é desconhecido quando um período indeterminado exatamente começa e termina.

Um período indeterminado representa um conjunto de períodos possíveis, sendo um destes o período real. O conjunto de períodos possíveis consiste de toda combinação de instantes possíveis dos instantes inicial e final. Logo, um período possível é obtido pela escolha de um instante dentre o conjunto de instantes possíveis do instante inicial e um

outro dentre o conjunto de instantes possíveis do instante final.

Intervalos indeterminados

Um intervalo determinado é uma duração conhecida de tempo, ou seja, uma contagem de instantes. Por outro lado, um intervalo indeterminado é uma duração imprecisa que descreve um conjunto de durações possíveis. Um instante indeterminado é representado por um número impreciso de instantes, como "entre 2 e 3 instantes". Assim como os instantes indeterminados, um intervalo indeterminado possui uma função de probabilidade associada que dá a probabilidade de cada duração possível.

É mostrado a seguir um banco de dados ilustrativo formado pelas relações *Produção* e *Peças-Recebidas* (TAB. 2 e 3). Este modela uma companhia com dois armazéns (Boeing e Cessna) que suprem peças para a fábrica. Assume-se uma granularidade básica de um dia. A fábrica mantém a relação *Produção* que é a história de aeronaves construídas pela fábrica (TAB. 2).

TABELA 2 - Relação *Produção*

Modelo	Num-serial	Durante
Centurion	AB33	'[01/3 ~ 31/3 – 01/6 ~ 30/6]'
Cutlass	Z19	'[01/6 ~ 30/6 – 01/7 ~ 31/7]'
Centurion	AB34	'[01/6 ~ 30/6 – 01/8 ~ 31/8]'
Caravan	FA2K	'[01/4 ~ 31/4 – 01/5 ~ 31/5]'

A produção de um avião começou em algum dia no conjunto de dias indicado, mas não se sabe em qual destes. Por exemplo, a produção do *Centurion* de número serial AB33 começou em março e terminou em junho. É igualmente provável que a produção tenha começado ou acabado durante qualquer dia dos períodos indicados. Todavia, também é permitida uma probabilidade não uniforme.

A relação *Peças-Recebidas* representa eventos de chegadas de peças de aviões (TAB. 3). Os instantes de ocorrência dos eventos nesta relação também são indeterminados, ou seja, não se sabe exatamente em qual dia as peças chegaram. Assume-se a distribuição uniforme: a peça Reverso chegou num dos seguintes dias {8/6, 9/6, ..., 27/6}, mas

não se favorece nenhum deles. A indeterminação no tempo de validade surge nas duas relações.

TABELA 3 - Relação *Peças-Recebidas*

Armazem	Num_lote	Peça	Quando
Boeing	23	Flap	10/5 ~ 29/5
Cessna	30	Flap	30/5 ~ 18/6
Boeing	24	Reverso	08/6 ~ 27/6
Cessna	31	Flap	13/6 ~ 02/7

Há dois limites bem definidos de uma resposta a uma consulta num banco de dados com informação indeterminada: a resposta definida e a resposta possível [Li79]. A resposta definida é a informação que satisfaz a consulta em *todas* as extensões possíveis do banco de dados, enquanto que a resposta possível é a informação que satisfaz a consulta em *alguma* extensão do banco de dados. Por exemplo, considere uma seleção temporal sobre a relação *Peças-Recebidas* que seleciona aquelas peças recebidas antes de 10/6. Embora a data exata do recebimento do lote 23 enviado pelo armazém *Boeing* seja omissa, sabe-se que esta encomenda chegou antes de 10/6, pois chegou em algum dia entre 10/5 e 29/5. Esta tupla, e nenhuma outra, está na resposta definida. O lote 30 do armazém *Cessna* está na resposta possível, pois se percebe que é possível que esta encomenda tenha chegado antes de 10/6 (e também é possível que não). Da mesma forma, o lote 24 do armazém *Boeing* possivelmente chegou antes de 10/6. A primeira encomenda (lote 23) também está na resposta possível porque uma resposta definida também é uma resposta possível, mas não vale o contrário.

Entre as respostas possíveis e definidas há outras respostas. Por exemplo, assuma que é igualmente provável para cada um dos dias entre 8/6 e 27/6 que o lote 24 tenha sido recebido. Para que a encomenda tenha chegado antes de 10/6, se percebe que ela pode ter chegado em 8/6 ou em 9/6. Como todos os dias são considerados igualmente prováveis, então há uma probabilidade de apenas 0.10 (2 chances entre 20) que a encomenda tenha sido recebida antes de 10/6. Logo, é pouco provável que o lote 24 tenha chegado antes de 10/6. Entretanto, é mais provável que o lote 30 (probabilidade de 0.55, ou seja, 11 chances entre 20) e que o lote 23 (probabilidade total de 1.00) tenham chegado antes de 10/6. Na FIG. 2 são exibidas as respostas definidas, possíveis e prováveis para o exemplo da seleção

temporal. Se a linguagem de consulta suportasse uma distribuição da probabilidade sobre os tempos possíveis associados com um instante indeterminado, poderia se obter uma linguagem de consulta com mais recursos, porém isto não deveria comprometer a eficiência no tempo de resposta de uma consulta nem a usabilidade da linguagem.

Armazém	Num_lote	Peça	Quando
Boeing	23	Flap	10/5 ~ 29/5

a) A Resposta Definida

Armazém	Num_lote	Peça	Quando
Boeing	23	Flap	10/5 ~ 29/5
Cessna	30	Flap	30/5 ~ 18/6

b) A Resposta Provável

Armazém	Num_lote	Peça	Quando
Boeing	23	Flap	10/5 ~ 29/5
Cessna	30	Flap	30/5 ~ 18/6
Boeing	24	Reverso	08/6 ~ 27/6

c) A Resposta Possível

FIGURA 2 – Respostas da Consulta-exemplo

Há dois estágios na determinação de uma resposta a uma consulta: o primeiro estágio recupera os dados relevantes à consulta e o segundo estágio constrói uma resposta que satisfaz as restrições especificadas na cláusula *where*. TSQL2 provê controles, chamados de *credibilidade* e *plausibilidade*, para cada um dos estágios.

A *credibilidade* muda a informação disponível para o processamento da consulta substituindo cada valor de tempo indeterminado por um valor de tempo determinado. Uma substituição típica é pelo *valor esperado*. Por exemplo, o *valor esperado* para o início da produção do *Centurion* com número de série AB33 é 15/3 (assume-se que numa distribuição uniforme de uma seqüência de valores o *valor esperado* é o valor do meio). Em TSQL2, o usuário pode expressar esta preferência selecionando um valor chamado *valor da credibilidade*. Este valor remove a indeterminação.

Dada um distribuição uniforme, é pouco provável que a produção de um *Centurion* de número serial AB33 tenha começado nos primeiros dias de março. Entretanto, nos últimos dias é mais provável que já tenha começado. Um certo usuário pode estar interessado nos tempos mais prováveis (fim de março e início de junho). A *credibilidade* escolhida potencialmente altera todo período numa tabela, restringindo a variação de cada período. De fato, os tempos sem créditos são eliminados ao nível da credibilidade escolhida, no processamento da consulta, permitindo que o usuário controle a qualidade da informação utilizada na consulta.

TSQL2 usa duas funções para suportar o conceito de *seleção de credibilidade*: as funções *Shrink_s* e *Shrink_t* [Sn95]. Estas funções de encurtamento calculam uma versão mais curta dos períodos inicial e final um instante indeterminado; e consequentemente alterar a função *p.m.f.*. A função *Shrink_s* calcula um período de indeterminação resultante da remoção dos primeiros instantes do conjunto de instantes possíveis. A quantidade de instantes removidos é determinada pelo argumento *Y*, correspondente a *seleção de credibilidade*. A função *Shrink_s* é definida como segue:

Shrink_s (Y, ($[t_1, t_2]$, $P\alpha$)), onde o segundo argumento representa o instante indeterminado $\alpha = ([t_1, t_2], P\alpha)$. O valor de Y está entre 0 e 100. Todo instante possível cuja probabilidade cumulativa é menor do que Y é removido. Quanto mais altos forem os valores de Y, mais instantes do conjunto serão removidos. Por exemplo, *Shrink_s*(*100*, α) removerá todo instante exceto o último instante possível em α. Por outro lado, *Shrink_s*(*0*, α) manterá α intocável. A função *Shrink_t* é similar a *Shrink_s*, exceto que aquela remove os instantes finais de um conjunto de instantes possíveis.

Segundo Snodgrass [Sn95], as duas seleções de credibilidade mais comuns são 100 e 0. Para o caso de 100, toda a indeterminação será eliminada do conjunto de instantes possíveis. Já para o caso de 0, toda a indeterminação será mantida.

A *plausibilidade* controla a construção de uma resposta usando as informações com créditos, ou seja, aquelas informações geradas pela função de *credibilidade*. Por exemplo, na FIG. 3 um dono de um *Centurion* pode consultar qual encomenda de *Flap possivelmente* chegou durante a produção de seu avião. Intuitivamente esta consulta releva as restrições sobre o relacionamento entre os dias de produção e o dia em que uma encomenda

foi recebida de *"sobreposição certa?"* para *"sobreposição provável?"*. É o usuário quem escolhe o tipo de sobreposição, atribuindo para a plausibilidade um valor entre 1 e 100. Em geral, um valor de *plausibilidade* igual a 1 usa a informação possível nos dados básicos, enquanto que um valor igual a 100 usa apenas a informação completa.

Consultas podem usar a informação indeterminada no banco de dados. Suponha que os proprietários dos *Centurion* reclamam de um *Flap* defeituoso. Naturalmente, se consulta o banco de dados, com a credibilidade e plausibilidade especificadas, para determinar quais armazéns supriram as peças defeituosas e quando foram recebidas; quais lotes estão envolvidos e os respectivos números de série dos *Centurion*. Esta consulta em TSQL2 é a seguinte:

```
SET DEFAULT PLAUSIBILITY 60
SELECT r.Armazen, r.Num-lote, p.Num-serial, r.Quando
FROM   Pecas_Recebidas AS r WITH CREDIBILITY INDETERMINATE
       Produção AS p WITH CREDIBILITY INDETERMINATE
WHERE  p.Modelo = "Centurion" AND r.Peca = "Flap" AND
       r.Quando OVERLAPS p.Durante
```

FIGURA 3 – Uma consulta TSQL2

A cláusula *FROM* especifica que toda informação das relações deve ser usada, desconsiderando sua credibilidade (*Shrink_s*(*0, α*) e *Shrink_t*(*0, α*)). A cláusula *WHERE* seleciona pares de tuplas *Centurion* e *Flap* que sobrepõem-se com uma plausibilidade de 60 (TAB. 4).

TABELA 4 - Resultado de Consulta TSQL2

Armazém	Num_lote	Num_serial	Quando
Boeing	23	AB33	10/5 ~ 29/5
Cessna	30	AB33	30/5 ~ 18/6
Cessna	31	AB34	13/6 ~ 02/7

Há uma nítida divisão entre a indeterminação nos dados e a indeterminação na consulta. O suporte para a indeterminação no tempo de validade permite que o usuário controle ambas. A *credibilidade* muda a indeterminação nos dados, especificamente no dado temporal, enquanto que a *plausibilidade* gerencia a indeterminação nos relacionamentos

entre os dados, por exemplo mudando o relacionamento de *"sobreposição certa?"* para *"sobreposição provável?"*.

As extensões sintática e a semântica formal de SQL-92 [MS93] e, em conseqüência, de TSQL2 [Sn95] para suportar a indeterminação no tempo de validade estão descritas em [DS98] onde há prova de que a semântica é completa e máxima, além de demonstrações de que tais extensões foram eficientemente implementadas.

4.3.7 Sintaxe e Semântica de TSQL2

Nesta seção são resumidas a sintaxe e semântica de TSQL2 [DS98] descritas com o objetivo de suportar o armazenamento e a recuperação da informação indeterminada no tempo de validade de um banco de dados.

Sintaxe

As extensões sintáticas de SQL-92, incorporadas em TSQL2, são as quatro seguintes: indicar que um atributo temporal é indeterminado, especificar a credibilidade, especificar a plausibilidade e indicar que um literal temporal é indeterminado.

A primeira extensão sintática envolve comandos para especificação do esquema de uma relação. No comando *create table*, um usuário pode adicionar o modificador *indeterminate* antes da especificação de um atributo do tipo instante, período ou intervalo, para declarar que o valor pode estar indeterminado. Também pode adicionar o modificador *with* para especificar as funções de probabilidade *standard* ou *nonstandard*. Atributos com uma distribuição uniforme ou com uma distribuição omissa são do tipo "*with standard distribution*" que é a assumida quando nada for especificado. Para períodos, estes modificadores se aplicam aos seus instantes delimitadores. Os comandos abaixo são exemplos que descrevem as relações mostradas nas TAB. 2 e 3.

```
CREATE TABLE Produção (ModeloCHARACTER (30),
                       Num_serial CHARACTER (10),
                       Durante    INDETERMINATE PERIOD (DATE));

CREATE TABLE Peças-Recebidas (Armazem   CHARACTER (30),
                              Num_lote  INTEGER,
                              Peça      CHARACTER (40),
                              Quando    INDETERMINATE DATE);
```

```
ALTER TABLE Peças-Recebidas ALTER COLUMN Quando
   TO NONSTANDARD DISTRIBUTION;
```

A segunda extensão sintática suporta a *credibilidade* que é denotada via uma frase *with credibility*. Uma *credibilidade* é uma estratégia de substituição para cada instante, período ou intervalo na relação especificada. As estratégias possíveis são as seguintes:

1) INDETERMINATE – retém toda indeterminação; não substitui nenhum dos valores temporais. É assumida se nenhuma for declarada.

2) EXPECTED – Substitui cada valor de tempo indeterminado pelo valor esperado. O valor esperado da distribuição uniforme sobre uma seqüência de valores é o valor do meio da seqüência. Por exemplo, dada a seqüência [1/3 ~ 31/3], com distribuição uniforme, seu valor esperado é 15/3.

3) MAX – Substitui cada valor de tempo indeterminado pelo limite inferior (α_*), exceto para um instante que inicia um período (neste caso se usa o limite superior (α^*)). Para períodos, este valor elimina toda a indeterminação. Para instantes, é usado o primeiro instante possível; enquanto que para intervalos, o menor intervalo possível é o escolhido.

4) MIN - Substitui cada valor de tempo indeterminado pelo limite superior (α^*), exceto para um instante que inicia um período (neste caso se usa o limite inferior (α_*)). Para períodos, este valor converte toda a informação indeterminada para informação determinada. Para instantes, é usado o último instante; enquanto que para intervalos, o maior intervalo possível é o escolhido.

A terceira extensão sintática é a *plausibilidade* que é um número inteiro entre 1 e 100 (inclusive estes) cujo *default* é 100. Uma *plausibilidade* igual a 1 indica que qualquer resposta possível é desejada; e uma *plausibilidade* igual a 100 indica que apenas resposta do tipo definida (FIG. 2) é desejada, ou seja, a cláusula *where* deve ser totalmente satisfeita.

A última extensão sintática é para suportar literais temporais, ou seja, literais como DATE '10/5/1997 ~ 29/5/1997'. Uma função de probabilidade pode ser adicionada ao literal. Por exemplo, DATE '10/5/1997 ~ 29/5/1997 UNIFORM representa um instante com

função de probabilidade uniforme.

Uma consulta TSQL2 ilustrando as várias extensões sintáticas foi mostrada na FIG. 3.

Semântica

Nesta seção é mostrada uma extensão semântica de SQL-92 para suportar a indeterminação temporal. A notação $[S]_{SQL}$ é usada para denotar o significado sintático da construção SQL-92 representada por *S*. Por outro lado, as notações $[\ ]_{ind}$ e $[\ ]_{op}$ servem, respectivamente, para denotar a semântica estendida, ou seja, a semântica indeterminada; e a semântica operacional. Esta última implementa a semântica indeterminada. As três mudanças na semântica de SQL-92 são as seguintes: a redefinição da relação de ordem temporal *Before*; a introdução de um domínio de quatro valores para avaliação da cláusula *where*; e a adição de um operador chamado *Replace* para efetivar a *credibilidade*.

As extensões sintáticas, mostradas na subseção anterior, incluem dois controles adicionais sobre a informação temporal indeterminada: *credibilidade* e *plausibilidade*. Estes valores aparecem como parâmetros adicionais, δ e γ, respectivamente, à semântica indeterminada $[\ ]_{ind}$. Ambos são especificados usando-se comandos SET DEFAULT cujos significados, assumindo *S* como um comando SQL-92 e *d* como um banco de dados, são os seguintes:

$$[\text{SET DEFAULT CREDIBILITY } \delta';\ S]_{ind}\ (\delta,\ \gamma,\ d) = [S]_{ind}\ (\delta,\ \gamma,\ d)$$
$$[\text{SET DEFAULT PLAUSIBILITY } \gamma';\ S]_{ind}\ (\delta,\ \gamma,\ d) = [S]_{ind}\ (\delta,\ \gamma',\ d)$$

Os valores *default* podem ser sobrescritos no próprio comando SELECT de SQL-92. Por exemplo, para sobrescrever a *plausibilidade default*

```
[SELECT <lista-atributos> FROM <lista-relações>
     WHERE <predicado> WITH PLAUSIBILITY γ']ind (δ, γ, d) =
[SELECT <lista-atributos> FROM <lista-relações>
     WHERE <predicado>]ind (δ, γ', d)
```

O valor *default* da *credibilidade* é INDETERMINATE e o valor da *plausibilidade* é 100.

A semântica indeterminada para o comando SELECT de SQL-92, delineada a se-

guir, possui a mesma estrutura que a semântica SQL-92:

```
[SELECT <lista-atributos> FROM <lista-relações>
     WHERE <predicado>]ind (δ, γ, d) =
          [<lista-atributos>]ind (γ, [WHERE <predicado>]ind
          (γ, [<lista-relações>]ind (δ, d)))
```

O comando SELECT de SQL-92 possui um significado diferente na semântica indeterminada. Quando avaliado sobre a semântica SQL-92, um comando SELECT, aplicado a um banco de dados que contém apenas informação do tipo determinada, possui só uma interpretação. Ao contrário, uma recuperação de um BDTI possui no mínimo duas interpretações. A primeira interpretação é que a consulta seleciona informação que *possivelmente* casa com as restrições de recuperação (especificadas por <predicado>). A segunda interpretação é que a consulta seleciona informação que *definitivamente* casa com as restrições de recuperação. Qual destas interpretações será adotada é uma decisão que cabe ao usuário especificar através de construções sintáticas na linguagem de consulta.

Na semântica indeterminada, a interpretação possível do comando SELECT é obtida com a *plausibilidade* igual a 1, enquanto que a interpretação definida é alcançada pela *plausibilidade* igual a 100. Para que se perceba a diferença destas, a cláusula *where* será focada. Antes porém, é necessária a definição do conceito de *complemento*.

Um instante indeterminado pode ser visto como um conjunto de instantes possíveis, sendo que um destes instantes é o instante "real". Cada um destes instantes possíveis representa uma diferente descrição da realidade. Cada possibilidade é chamada um *complemento* do instante indeterminado. A seguinte definição captura esta intuição. Seja $\alpha = (\alpha_* \sim \alpha^*, P_\alpha)$. Um complemento de um instante indeterminado α é α_i, onde α_i é um instante tal que $\alpha_* \leq \alpha_i \leq \alpha^*$. O conjunto de todos os complementos de um instante α é denotado por $C(\alpha)$.

Os complementos de períodos e intervalos também existem. Um *complemento* de um período indeterminado é aquele no qual seus instantes indeterminados são ambos substituídos por seus complementos. Um complemento de um intervalo indeterminado é uma das possíveis durações. Este conceito pode também ser generalizado para tuplas, relações e bancos de dados. Um complemento de x, onde x é uma tupla, uma relação ou mesmo um banco de dados, é x_c, onde x_c é o mesmo que x, mas com cada instante, período ou inter-

valo indeterminado, substituído por seu complemento. Logo, o conjunto de todos os complementos de uma entidade x é denotado por $\mathbf{C}(x)$.

A seguir é dada a interpretação definida da cláusula *where*, onde r denota uma relação e as tuplas são denotadas por t e t':

$$[\texttt{WHERE <predicado>}]_{ind}\ (100, r) = \{t \mid t \in r \wedge \forall t' \in \mathbf{C}(t)([\texttt{<predicado>}]_{SQL}(t'))\}$$

A interpretação definida seleciona apenas aquelas tuplas que são recuperadas pela semântica SQL-92 (note o uso de $[\texttt{<predicado>}]_{SQL}$) em *todo* complemento da tupla.

A interpretação possível seleciona apenas aquelas tuplas que são recuperadas pela semântica SQL-92 em *algum* complemento da tupla.

$$[\texttt{WHERE <predicado>}]_{ind}\ (1, r) = \{t \mid t \in r \wedge \exists t' \in \mathbf{C}(t)([\texttt{<predicado>}]_{SQL}(t'))\}$$

Suportando Plausibilidade

As interpretações definida e possível de uma consulta na semântica indeterminada foram declaradas em termos da semântica SQL-92. Uma implementação destas interpretações é impraticável, pois seria necessário computar o predicado sobre todo complemento possível de uma tupla. Existiriam muitos complementos para cada tupla, dependendo do tamanho de seu período de indeterminação. Para resolver este problema, foi definida uma semântica operacional que implementa a semântica indeterminada.

A semântica do predicado *where* sem indeterminação está baseada numa ordenação dos instantes nas relações básicas [Al83]. Todos os predicados temporais referem-se a ordenação dada pela relação de ordem *Before* para determinar o valor verdade do predicado. Por exemplo, considere o seguinte predicado OVERLAPS:

$$[\texttt{<instante}_1\texttt{>}\ \texttt{OVERLAPS PERIOD(<instante}_2\texttt{>, <instante}_3\texttt{>)}]_{SQL} = \textit{Before}([\texttt{<instante}_2\texttt{>}]_{SQL}, [\texttt{<instante}_1\texttt{>}]_{SQL}) \wedge \textit{Before}([\texttt{<instante}_1\texttt{>}]_{SQL}, [\texttt{<instante}_3\texttt{>}]_{SQL})$$

O valor verdade do predicado depende do resultado da operação *Before*.

Um conjunto de instantes determinados tem apenas uma ordenação temporal: dada

uma expressão temporal formada por operadores temporais, esta ordenação satisfaz ou não a expressão. Já um conjunto de instantes indeterminados possui muitas ordenações temporais possíveis, devido aos muitos complementos daqueles instantes. Algumas destas ordenações são plausíveis. O usuário especifica quais são plausíveis pela atribuição de um valor de plausibilidade apropriado. Uma expressão temporal é satisfeita se existe uma ordenação plausível entre pares de instantes que satisfazem cada predicado na expressão.

Na semântica SQL-92, a relação *Before* é análoga à relação "≤" na representação de instantes.

$$Before(\alpha, \beta) = \alpha \leq \beta.$$

Todavia, na semântica operacional a ordenação temporal é baseada na probabilidade de que um instante esteja antes de outro, ou seja, para quaisquer dois instantes α e β, a probabilidade de que α esteja antes de β é definida por:

$$Pr[\alpha \leq \beta] = \sum_{i, j \in \{0,\ldots,n\} \wedge i \leq j} Pr[\alpha = i] \times Pr[\beta = j]$$

onde *i* e *j* denotam *chronons* [JD98] (dia, por exemplo) e $Pr[\alpha = i]$ é a probabilidade de α estar localizado em *i*. Por exemplo, suponha que $\alpha = 30/5 \sim 18/6$ e que $\beta = 8/6 \sim 27/6$, conforme as tuplas de *Num-lote* = 30 e *Num-lote* = 24 (TAB. 3), respectivamente, e que a *p.m.f.* é uniforme. A probabilidade $Pr[\alpha \leq \beta] = 0.86$, ou seja, a probabilidade do instante indeterminado α estar antes do instante indeterminado β é de 86%, considerando-se a distribuição uniforme.

Para mais flexibilidade no tratamento de instantes indeterminados, a relação de ordem temporal *Before* foi redefinida para incluir mais um parâmetro, γ, denotando a *plausibilidade*. O valor de γ pode ser qualquer inteiro entre 1 e 100, inclusive. Logo, plausibilidades perto de 100 estipulam que só ordenações altamente prováveis são consideradas. Para um par de instantes indeterminados α e β, e uma *plausibilidade* γ, a nova relação de ordem $Before_I$ é definida por:

$$Before_I(\alpha, \beta, \gamma) = \{True \mid Pr[\alpha \leq \beta] \times 100 \geq \gamma\} \cup \{False \mid Pr[\beta < \alpha] \times 100 \geq \gamma\}$$

Há quatro possíveis resultados para a relação $Before_I$ sobre um par de instantes:

1) o relacionamento não é válido nem sua negação: a avaliação retorna o conjunto vazio.

2) o relacionamento vale e sua negação não vale: a relação *Before*$_I$ é avaliada como {True}.

3) o relacionamento não vale, porém sua negação vale: a relação *Before*$_I$ é avaliada como {False}.

4) O relacionamento e sua negação valem: a relação *Before*$_I$ é avaliada como {True, False}.

A diferença entre a semântica operacional e a semântica SQL-92 é que a primeira usa *Before*$_I$ ao invés de *Before*. Logo, a semântica operacional para o predicado temporal OVERLAPS é

$$[\langle \text{instante}_1 \rangle \text{ OVERLAPS PERIOD}(\langle \text{instante}_2 \rangle, \langle \text{instante}_3 \rangle)]_{OP} =$$
$$\textit{Before}_I([\langle \text{instante}_2 \rangle]_{SQL}, [\langle \text{instante}_1 \rangle]_{SQL}, \gamma)$$
$$\cap\ \textit{Before}_I([\langle \text{instante}_1 \rangle]_{SQL}, [\langle \text{instante}_3 \rangle]_{SQL}, \gamma)$$

O significado da cláusula *where* na semântica indeterminada pode agora ser dado em termos da semântica operacional:

$$[\text{WHERE } \langle \text{predicado} \rangle]_{ind}(\gamma, r) =$$
$$\{t \mid t \in r \wedge \text{True} \in [\langle \text{predicado} \rangle]_{OP}(\gamma, t)\}$$

Suportando Credibilidade

A *credibilidade* é usada para substituir valores temporais indeterminados por valores temporais determinados. Ela estende, opcionalmente, a cláusula *from*. A estratégia de substituição depende do valor da credibilidade e se o valor temporal substituído é um instante, período ou intervalo. O operador *Replace* que efetua esta substituição é definido por

$$\text{Replace}(\delta, t) = (X', R(\alpha_1), \ldots, R(\alpha_n))$$

onde $t = (X', \alpha_1, \ldots, \alpha_n)$ é uma tupla, X' são os valores não-temporais, $\alpha_1, \ldots, \alpha_n$ são os valores temporais e R é a estratégia de substituição. Exemplos do uso do operador *Replace* foram dados na seção anterior.

Por exemplo, dada a tupla t = (*Centurion*, AB33, [1/3 ~ 31/3 ~ 1/6 ~ 30/6]), os quatro valores de *credibilidade* produzidos são os seguintes:

Replace (INDETERMINATE, t) = t
Replace (EXPECTED, t) = (Centurion, AB33, [15/3 - 15/6])
Replace (MAX, t) = (Centurion, AB33, [31/3 - 1/6])
Replace (MIN, t) = (Centurion, AB33, [1/3 - 30/6])

A definição do significado da cláusula *from* na semântica indeterminada, incluindo a especificação de como a *credibilidade* "default" pode ser sobrescrita, é a seguinte:

$[\langle \text{lista-relações} \rangle]_{ind}(\delta, d) =$

$[\langle \text{rel}_1 \rangle, \ldots, \langle \text{rel}_n \rangle]_{ind}(\delta, d) =$

$[\langle \text{rel}_1 \rangle]_{ind}(\delta, d)$ X...X $[\langle \text{rel}_n \rangle]_{ind}(\delta, d)$

onde cada $\langle \text{rel}_i \rangle$ pode ser uma das seguintes construções:

$[\langle \text{relação} \rangle]_{ind}(\delta, d) = \{t' \mid t \in \langle \text{relação} \rangle \wedge t' = \textit{Replace}(\delta, t)\}$

$[\langle \text{relação} \rangle \text{WITH CREDIBILITY } \delta']_{ind}(\delta, d) = \{t' \mid t \in \langle \text{relação} \rangle \wedge t' = \textit{Replace}(\delta', t)\}$

Quando na <lista-relações> é dada uma *credibilidade* δ = INDETERMINATE, é retida toda a indeterminação presente nos valores temporais de *d*, ou seja,

$[\langle \text{lista-relações} \rangle]_{ind}(\text{INDETERMINATE}, d) = [\langle \text{lista-relações} \rangle]_{SQL}(d)$.

Seja o seguinte exemplo, correspondente a consulta Q da FIG. 3, onde a semântica indeterminada do comando SELECT de SQL-92 é especificada:

$$
\begin{aligned}
[Q]_{ind}\ (d) = \{&(r.\text{Armazen}, r.\text{Num-lote}, p.\text{Num-serial}, r.\text{Quando}) \mid \\
&r \in \textit{Replace}(\text{INDETERMINATE}, \text{Pecas\_Recebidas}) \\
&\wedge\ p \in \textit{Replace}(\text{INDETERMINATE}, \text{Produção}) \\
&\wedge\ p.\text{Modelo} = \text{"Centurion"} \wedge r.\text{Peca} = \text{"Flap"} \\
&\wedge\ \textit{True} \in \{\textit{Before}_I(p.\text{Durante}_{\text{início}}, r.\text{Quando}, 60) \cap \\
&\qquad \textit{Before}_I(r.\text{Quando}, p.\text{Durante}_{\text{fim}}, 60))\}
\end{aligned}
$$

A projeção foi expressa em cálculo relacional de tuplas [SKS99].

Se esta consulta fosse aplicada ao banco de dados composto pelas TAB. 2 e 3, resultaria nas três tuplas de TAB. 4.

4.4 Uma extensão temporal a SQL-92

Nesta seção é apresentada uma extensão temporal a SQL-92 [MS93] proposta por Snodgrass [Sn00] onde há basicamente três tipos de declarações orientadas ao tempo:

- *DECLARAÇÕES CORRENTES*: são declarações realizadas sobre o tempo corrente.
- *DECLARAÇÕES SEQÜENCIAIS*: são declarações realizadas em cada instante de tempo.
- *DECLARAÇÕES NÃO-SEQUENCIAIS*: são declarações que ignoram o tempo.

Todas estas declarações se aplicam a consultas e a modificações. Não serão abordadas declarações não-sequenciais devido ao fato de que estas não envolvem o tempo.

O banco de dados no qual foram baseados os exemplos desta seção e da seção 4.5 é formado pelas tabelas descritas a seguir:

- `EMPREGADOS (numero, nome)`
- `CARGOS (cod_cargo, nome_cargo)`
- `EMPREGOS (numero, cod_cargo, data_inicio, data_fim)`
- `SALÁRIOS (numero, valor_sal, data_inicio, data_fim)`

A relação `Empregados` provê dados de empregados. Seus atributos são o número da carteira do trabalho (`numero`) e o nome do empregado (`nome`). A relação `Cargos` armazena dados sobre os cargos que podem ser assumidos pelos empregados. Os atributos desta tabela são o código do cargo (`cod_cargo`) e a descrição do cargo (`nome_cargo`). A relação `Empregos` (TAB. 5) fornece dados dos cargos assumidos por um empregado (a sua história de empregos), relacionando o número da carteira de trabalho (`numero`) com o código do cargo (`cod_cargo`) exercido, no período de `data_inicio` até `data_fim`. A coluna data_inicio indica quando o empregado foi atribuído ao cargo, enquanto que a coluna `data_fim` indica quando o empregado deixou este cargo. Os conteúdos destes atributos estão no formato ano-mês-dia. A relação `Salários` (TAB. 6) armazena dados dos vários salários percebidos pelo empregado. Os atributos desta tabela são o número da carteira de trabalho do empregado (`numero`) e seu salário (`valor_sal`) no período de

validade indicado pela data quando começou a ganhar o salário (data_inicio) e pela data quando deixou de ganhar o salário (data_fim). As colunas data_inicio e data_fim delimitam o período de validade da informação na linha e são conhecidas como *marcações temporais*. As relações Empregos e Salários são tabelas do tipo tempo de validade, ou seja, tabelas que gravam o tempo em que a informação foi válida na realidade modelada. A granularidade temporal delas é de um dia.

TABELA 5 - Relação Empregos

Numero	cod_cargo	data_inicio	data_fim
123	033	1996-10-01	1997-06-01
123	112	1996-01-01	1996-10-01
123	288	1995-03-01	1996-01-01
123	401	1997-10-01	1998-01-01
123	654	1997-06-01	1997-10-01

TABELA 6 – Relação Salários

numero	valor_sal	data_inicio	data_fim
123	15.750	1995-03-01	1996-05-01
123	16.250	1996-05-01	1997-06-01
123	17.000	1997-06-01	1998-01-01

4.4.1 Consultas correntes

Consultas correntes são consultas que extraem o estado atual de uma tabela tempo de validade. Por exemplo, "qual é o nome do cargo atual de Pedro, assumindo que a data de realização da consulta é 01/12/1997?"

```
SELECT nome_cargo
FROM EMPREGADOS, CARGOS, EMPREGOS
WHERE nome = 'Pedro'
      AND EMPREGADOS.numero = EMPREGOS.numero
      AND EMPREGOS.cod_cargo = CARGOS.cod_cargo
      AND data_inicio <= CURRENT_DATE
      AND CURRENT_DATE < data_fim
```

As duas últimas linhas requerem que a data corrente (CURRENT_DATE) esteja no período de validade (data_inicio, data_fim). A resposta é o nome do cargo cujo

cod_cargo é igual a 401.

4.4.2 Consultas seqüenciais

Consultas seqüenciais são consultas que resultam em tabelas do tipo tempo de validade, ao contrário das consultas correntes que retornam o estado atual das tabelas envolvidas. Por exemplo, "quem ganha ou ganhou mais do que $5000?".

```
SELECT *
FROM SALÁRIOS
WHERE valor_sal > 5000
```

A resposta é formada pelas linhas da relação Salários (TAB. 6).

4.4.3 Modificações correntes

Uma modificação corrente se aplica no período entre CURRENT_DATE (*now*) até 9999-12-31 (*forever*).

Inserções correntes

Uma inserção corrente requer que as datas inicial e final sejam especificadas. Por exemplo, " inserir Pedro (numero = 123) como Professor Adjunto (cod_cargo = 071) na relação Empregos (TAB. 5)".

```
INSERT INTO EMPREGOS
VALUES (123, 071, CURRENT_DATE, DATE '9999-12-31')
```

Esta declaração fornece uma marcação temporal de CURRENT_DATE até 9999-12-31.

Na TAB. 7 é mostrado o registro mais recente da relação Empregos, assumindo-se que Pedro foi admitido como Professor Adjunto em 1/1/1998.

TABELA 7 - Registro atual de Pedro em EMPREGOS

Numero	cod_cargo	data_inicio	data_fim
123	071	1998-01-01	9999-12-31

Remoções correntes

Uma remoção corrente é traduzida numa atualização. Por exemplo, "Pedro foi demitido como Professor Adjunto".

```
UPDATE EMPREGOS
SET data_fim = CURRENT_DATE
WHERE numero = 123
      AND cod_cargo = 071
      AND data_fim = DATE '9999-12-31'
```

Na TAB. 8 é mostrado o resultado da remoção lógica em TAB. 7, assumindo-se que sua demissão foi efetivada em 13/3/1998.

TABELA 8 - Registro de Pedro em EMPREGOS após uma remoção corrente

numero	cod_cargo	data_inicio	data_fim
123	071	1998-01-01	1998-03-13

Atualizações correntes

Uma atualização corrente é implementada como uma atualização ao fim da linha corrente em *now* e uma inserção de novos valores, ou seja, ela é logicamente uma remoção acoplada com uma inserção. Por exemplo, "Pedro foi promovido hoje a Professor Titular (cod_cargo = 739)".

```
INSERT INTO EMPREGOS (numero, cod_cargo, data_inicio,
                      data_fim)
SELECT DISTINCT numero, 739, CURRENT_DATE, DATE '9999-12-31'
FROM EMPREGOS
WHERE numero = 123
      AND data_fim = DATE '9999-12-31'

UPDATE EMPREGOS
SET data_fim = CURRENT_DATE,
WHERE numero = 123
      AND data_inicio < CURRENT_DATE
```

A atualização deve ocorrer logo após a inserção. Por exemplo, assumindo a situação atual como sendo a mostrada na TAB. 8 e assumindo também que Pedro foi promovido em 13/3/1998, se tem a seguinte situação após a atualização corrente (TAB. 9).

TABELA 9 - Registros de Pedro em EMPREGOS após uma atualização corrente

numero	cod_cargo	data_inicio	data_fim
123	071	1998-01-01	1998-03-13
123	739	1998-03-13	9999-12-31

Um período de aplicabilidade é um período especificado pelo usuário. Lembre-se que uma atualização corrente tem um período de aplicabilidade de *now* até *forever*. A FIG. 4 mostra três casos com relação ao período de validade de uma atualização corrente. Se um período de validade termina no passado, então a atualização não afetará aquela linha (caso 1). Se a linha é válida atualmente, então a porção antes de *now* deve terminar em *now* e a nova linha, com os valores de atributos já atualizados, deve ser inserida com o período de validade começando em *now* e terminando quando a linha original terminava (caso 2). Se a linha começa no futuro, esta pode ser atualizada como uma atualização convencional (caso 3). Por exemplo, assumindo TAB. 7 como a situação atual, "Hoje Pedro foi promovido a Professor Titular".

```
INSERT INTO EMPREGOS (numero, cod_cargo, data_inicio,
                      data_fim)
SELECT numero, 739, CURRENT_DATE, data_fim
FROM EMPREGOS
WHERE numero = 123
      AND data_inicio <= CURRENT_DATE
      AND data_fim > CURRENT_DATE

UPDATE EMPREGOS
SET data_fim = CURRENT_DATE
WHERE numero = 123
      AND data_inicio < CURRENT_DATE
      AND data_fim > CURRENT_DATE

UPDATE EMPREGOS
SET cod_cargo = 739
WHERE numero = 123
      AND data_inicio > CURRENT_DATE
```

A segunda declaração de atualização modifica linhas com período de validade iniciado no futuro. Esta declaração pode aparecer em qualquer local, porém a primeira deve ocorrer após a inserção.

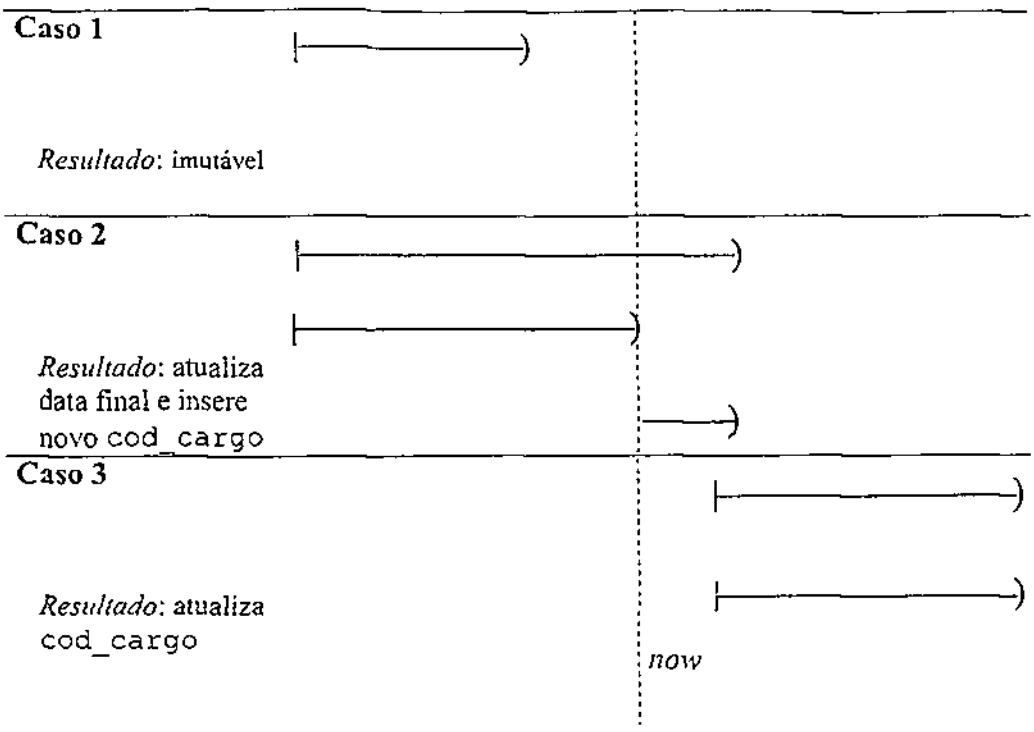


FIGURA 4 - Casos de atualização corrente, com período de validade mostrado

4.4.4 Modificações seqüenciais

Uma modificação seqüencial se aplica sobre um período de aplicabilidade que pode estar no passado, no futuro ou sobrepor *now*.

Inserções seqüenciais

Numa inserção seqüencial o período de aplicabilidade é fornecido. Por exemplo, "Pedro foi contratado como Professor Adjunto para 1997".

```
INSERT INTO EMPREGOS
VALUES (123, 071, DATE '1997-01-01', DATE '1998-01-01')
```

Remoções seqüenciais

Na FIG. 5 uma remoção seqüencial é realizada para cada um dos casos, sendo que o período de validade (PV) da linha original é mostrado acima do período de aplicabilidade (PA) da remoção. No caso 1, a linha original cobre o PA, de modo que os períodos inicial e final precisam ser retidos: o período inicial é retido pela atribuição do início do PA na data final e o período final é retido pela atribuição do fim do PA na data inicial. No caso 2, ape-

nas a porção inicial do PV da linha original é retida. Simetricamente, no caso 3 apenas a porção final do PV é retida. No caso 4, a linha inteira é removida, pois o PA a cobre inteiramente. Por exemplo, "Pedro licenciou-se como Professor Adjunto durante 1997". Para ilustrar o caso 1, assuma que a situação atual é definida pela seguinte tupla na relação Empregos (123, 071, 1996/01/01, 2000/10/30).

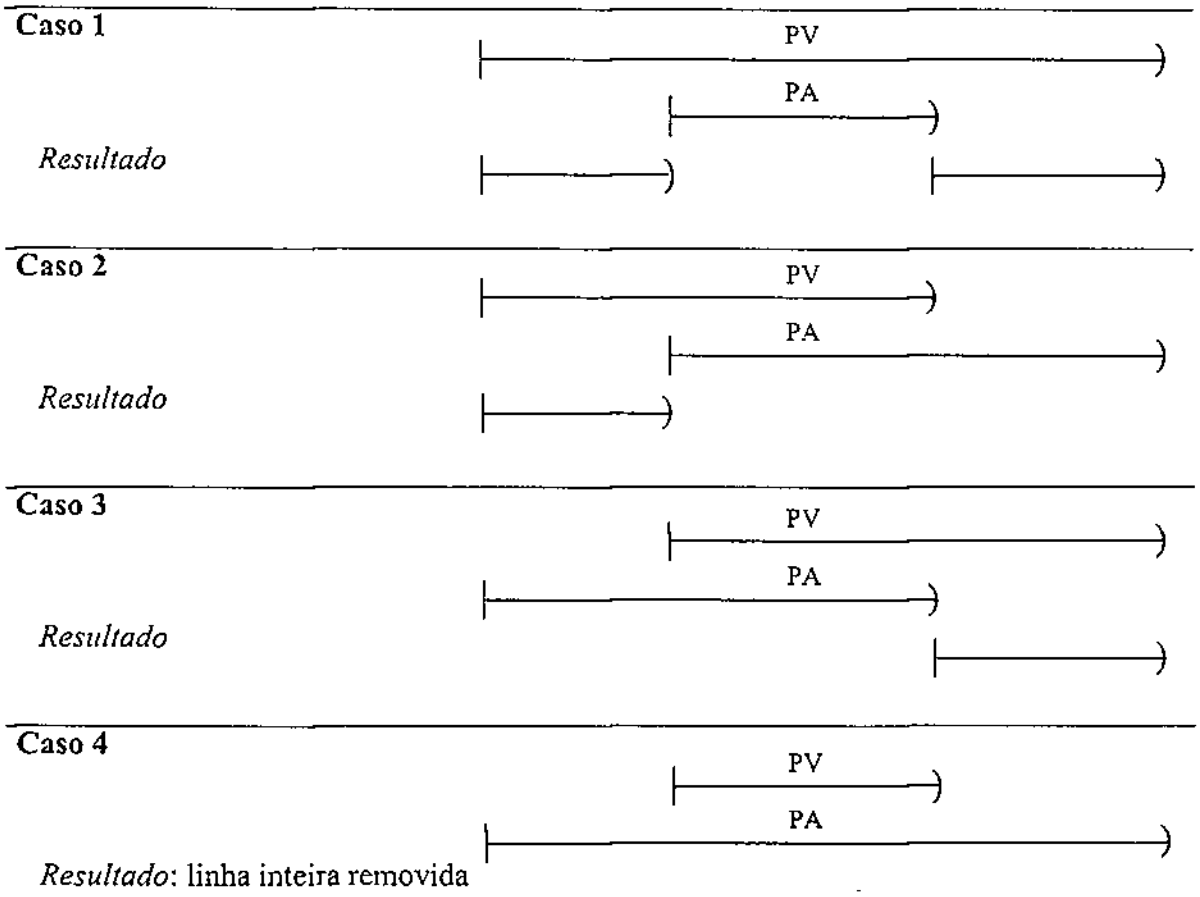


FIGURA 5 – Casos de remoção seqüencial

Uma remoção seqüencial é implementada por 4 declarações: uma inserção, duas atualizações e uma remoção. O caso 1 está refletido nas duas primeiras declarações, enquanto que a segunda declaração cobre o caso 2. Por sua vez, a terceira declaração trata do caso 3; enquanto que a quarta declaração resolve o caso 4.

```
INSERT INTO EMPREGOS
SELECT numero, cod_cargo, DATE '1998-01-01', data_fim
FROM EMPREGOS
WHERE numero = 123
      AND cod_cargo = 071
      AND data_inicio < DATE '1997-01-01'
      AND data_fim > DATE '1997-01-01'

UPDATE EMPREGOS
SET data_fim = DATE '1997-01-01'
WHERE numero = 123
      AND cod_cargo = 071
      AND data_inicio < DATE '1997-01-01'
      AND data_fim >= DATE '1998-01-01'

UPDATE EMPREGOS
SET data_inicio = DATE '1998-01-01'
WHERE numero = 123
      AND cod_cargo = 071
      AND data_inicio < DATE '1998-01-01'
      AND data_fim >= DATE '1998-01-01'

DELETE FROM EMPREGOS
WHERE numero = 123
      AND cod_cargo = 071
      AND data_inicio >= DATE '1997-01-01'
      AND data_fim <= DATE '1998-01-01'
```

Atualizações seqüenciais

Uma atualização seqüencial é o análogo temporal de uma atualização não-temporal, com um período de aplicabilidade especificado.

Os casos mostrados na FIG. 5 são tratados diferentemente numa atualização seqüencial (FIG. 6). No caso 1, as porções inicial e final do PV são retidas (via duas inserções) e a porção afetada é atualizada. No caso 2, apenas a porção inicial é retida, enquanto que no caso 3 apenas a porção final é retida. No caso 4, o PV é retido, pois é coberto pelo PA. Por exemplo, "Pedro foi promovido a Professor Titular (cod_cargo = 739) apenas para o ano de 1997". Para ilustrar o caso 1, assuma que a tupla (123, 071, 1996/01/01, 2000/10/30) expressa a situação em vigor.

```
INSERT INTO EMPREGOS
SELECT numero, cod_cargo, data_inicio, DATE '1997-01-01'
FROM EMPREGOS
WHERE numero = 123
      AND data_inicio < DATE '1997-01-01'
      AND data_fim > DATE '1997-01-01'

INSERT INTO EMPREGOS
SELECT numero, cod_cargo, DATE '1998-01-01', data_fim
FROM EMPREGOS
WHERE numero = 123
      AND data_inicio < DATE '1998-01-01'
      AND data_fim > DATE '1998-01-01'

UPDATE EMPREGOS
SET cod_cargo = 739
WHERE numero = 123
      AND data_inicio < DATE '1998-01-01'
      AND data_fim > DATE '1997-01-01'

UPDATE EMPREGOS
SET data_inicio = DATE '1997-01-01'
WHERE numero = 123
      AND data_inicio < DATE '1997-01-01'
      AND data_fim > DATE '1997-01-01'

UPDATE EMPREGOS
SET data_fim = DATE '1998-01-01'
WHERE numero = 123
      AND data_inicio < DATE '1998-01-01'
      AND data_fim > DATE '1998-01-01'
```

A primeira declaração INSERT trata das porções iniciais dos casos 1 e 2 e a segunda trata das porções finais dos casos 2 e 3. A primeira declaração UPDATE trata da atualização em todos os quatro casos. A segunda e a terceira declarações UPDATE ajustarão as datas inicial (para os casos 1 e 2) e final (para os casos 1 e 3) da porção atualizada. Note que as três declarações UPDATE não causarão impacto na(s) linha(s) recém-inserida(s), pois os PV destas estão fora do PA. Todas estas cinco declarações devem ser avaliadas na ordem mostrada.

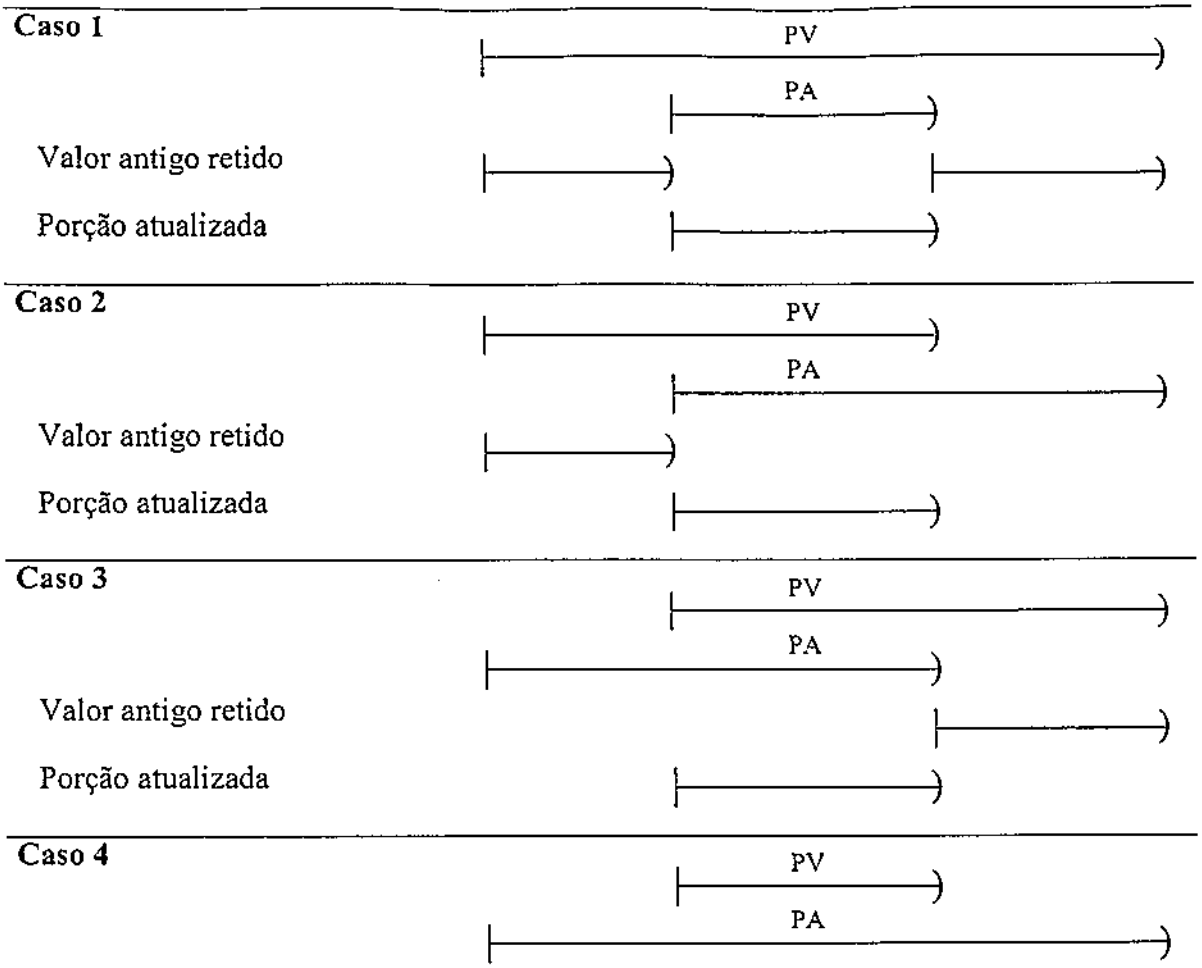


FIGURA 6 - Casos de atualização seqüencial

4.5 SQL/Temporal

SQL/Temporal é a parte temporal proposta para incorporação em SQL3 [Me97, SKK+94]. Novas construções para utilização de tempo de validade [SBJ+96a] e de tempo de transação [SBJ+96b] foram aprovadas pelo comitê ANSI (American National Standards Institute) SQL3 e estão sob consideração no comitê ISO (International Organization for Standardization) SQL3. Estas construções são variantes de construções TSQL2 [Sn95].

As extensões serão abordadas através de exemplos de comandos para consultas, modificações e definições de tabelas.

4.5.1 Definição de tabelas SQL/Temporal

O suporte ao tempo de validade em tabelas SQL/Temporal é especificado na criação (comando CREATE TABLE) ou na alteração de tabelas (comando ALTER TABLE). O comando CREATE TABLE é estendido para definir tabelas com suporte ao tempo de validade. Este comando pode ser seguido pela cláusula "AS VALIDTIME PERIOD (DATE)" que especifica o suporte do tempo de validade na tabela. Uma tabela é vista então como uma coleção de linhas associadas com períodos de tempo de validade. Por exemplo, as seguintes tabelas foram criadas em 1/2/1995.

```
CREATE TABLE empregados
      (numero INTEGER VALIDTIME PRIMARY KEY,
      nome VARCHAR (12),
      datanasc DATE)
AS VALIDTIME PERIOD(DATE)

CREATE TABLE salários
      (numero INTEGER VALIDTIME PRIMARY KEY VALIDTIME
      REFERENCES empregado,
      valor INTEGER)
AS VALIDTIME PERIOD(DATE)
```

As restrições de integridade (chave primária e chave estrangeira) [SKS99] valem para cada ponto no tempo (dia) através das cláusulas "VALIDTIME PRIMARY KEY" e "VALIDTIME REFERENCES". Supondo que estas tabelas foram preenchidas em 1/2/1995 pelos seguintes comandos:

```
INSERT INTO empregados VALUES(5873, 'José', DATE '1961-03-21')
INSERT INTO empregados VALUES(6542, 'Maria', DATE '1963-07-04')
INSERT INTO salários VALUES(6542, 3200)
INSERT INTO salários VALUES(5873, 3300)
```

Assumindo que os seguintes comandos de inserção foram executados em 2/2/1995.

```
INSERT INTO empregados VALUES(3463, 'Rita', DATE '1970-03-09')
INSERT INTO salários VALUES(3463, 3400)
```

As TAB. 10 e 11 mostram o resultado das inserções.

TABELA 10 - Relação Empregados

numero	nome	datanasc	Valid
5873	José	1961-03-21	[1995-02-01 – 9999-12-31)
6542	Maria	1963-07-04	[1995-02-01 – 9999-12-31)
3463	Rita	1970-03-09	[1995-02-02 – 9999-12-31)

TABELA 11- Relação Salários

numero	valor	Valid
5873	3300	[1995-02-01 – 9999-12-31)
6542	3200	[1995-02-01 – 9999-12-31)
3463	3400	[1995-02-02 – 9999-12-31)

Três observações sobre as TAB. 10 e 11: os símbolos "[..)" são usados para denotar intervalo semi-aberto à direita, a coluna *Valid* é implícita ao usuário e o tempo de validade na inserção é assumido se estender até 31-12-9999.

Com o comando SQL3 ALTER TABLE, se pode adicionar o tempo de validade em tabelas SQL-92 já existentes. Por exemplo, o comando `ALTER TABLE Empregados ADD VALIDTIME PERIOD (DATE)` associa um período de *now* a *forever* ([CURRENT_DATE, 9999-12-31]) a cada linha da tabela `Empregados`.

As consultas e modificações, exemplificadas nas seções 4.5.2 e 4.5.3, serão expressas sobre as TAB. 10 e 11, sem as linhas correspondentes a José, e também sobre as tabelas `Empregos` e `Cargos` definidas no início da seção 4.4.

4.5.2 Consultas SQL/Temporal

A idéia básica é que com poucas adições em SQL-92, esta consiga expressar consultas temporais de forma parecida com as consultas não-temporais. As declarações temporais são do tipo corrente e seqüencial, analogamente às declarações da extensão a SQL-92 (seção 4.4).

Consultas correntes

Consultas correntes recuperam a informação corrente. Neste caso, as consultas

SQL-92 se aplicam diretamente, sem especificar o tempo de validade. Por exemplo, seja a consulta: "qual é o nome do cargo atual de Pedro?".

```
SELECT nome_cargo
FROM EMPREGADOS, CARGOS, EMPREGOS
WHERE nome = 'Pedro'
      AND EMPREGADOS.numero = EMPREGOS.numero
      AND EMPREGOS.cod_cargo = CARGOS.cod_cargo
```

Perceba que esta consulta não especifica o tempo de validade, diferentemente daquela consulta da seção 4.4.1.

Consultas seqüenciais

Uma consulta seqüencial caracteriza-se pelo uso da palavra-chave VALIDTIME antes da declaração SELECT e resulta numa tabela com tempo de validade. Por exemplo, seja a consulta: "quem ganha ou ganhou mais do que $5000?".

```
VALIDTIME SELECT *
FROM SALARIOS
WHERE valor_sal > 5000
```

A diferença desta consulta com a da seção 4.4.2 é apenas a palavra-chave VALIDTIME.

Outro exemplo de consulta para recuperar os que foram empregados alguma vez durante o primeiro semestre de 1995:

```
VALIDTIME PERIOD '[1995-01-01 - 1995-07-01)'
SELECT nome FROM EMPREGADOS
```

A TAB. 12 mostra o resultado desta consulta.

TABELA 12 - Relação com tempo de validade

nome	Valid
Maria	[1995-02-01 – 1995-07-01)
Rita	[1995-02-02 – 1995-07-01)

O comando de seleção a seguir determina quem teve aumento salarial, considerando TAB. 15 como a tabela SALÁRIOS.

```
VALIDTIME SELECT nome
FROM EMPREGADOS AS E, SALÁRIOS AS S1, SALÁRIOS AS S2
WHERE E.numero = S1.numero AND E.numero = S2.numero
      AND S1.valor < S2.valor
      AND VALIDTIME(S1) MEETS VALIDTIME(S2)
```

O operador MEETS assegura que o período associado com S1 é imediatamente seguido pelo período associado com S2. O resultado está sem suporte temporal porque VALIDTIME não é seguido por uma expressão periódica (TAB. 13).

TABELA 13 - Relação não-temporal

nome
Rita

Para se obter uma tabela com suporte temporal, ou seja, "quem teve aumento salarial e quando?", deve-se colocar um valor após VALIDTIME para especificar quando cada linha resultante é válida.

```
VALIDTIME VALIDTIME(S2) SELECT nome
FROM EMPREGADOS AS E, SALÁRIOS AS S1, SALÁRIOS AS S2
WHERE E.numero = S1.numero AND E.numero = S2.numero
      AND S1.valor < S2.valor
      AND VALIDTIME(S1) MEETS VALIDTIME(S2)
```

A TAB. 14 mostra o resultado desta consulta com suporte temporal.

TABELA 14 - Relação com suporte temporal

nome	Valid
Rita	[1995-04-01 – 9999-12-31)

4.5.3 Modificações SQL/Temporal

Modificações SQL/Temporal são operações para inserção, remoção ou atualização de tabelas tempo de validade.

Modificações correntes

Uma modificação corrente se aplica de *now* até *forever*. As modificações que não envolvem a palavra-chave VALIDTIME, quando aplicadas a tabelas com suporte ao tempo de validade, são interpretadas como modificações correntes. Por exemplo, suponha a se-

guinte inserção corrente: "Pedro foi contratado como Professor Adjunto (cod_cargo = 071)".

```
INSERT INTO EMPREGOS
VALUES (123, 071)
```

A marcação temporal assumida automaticamente é [CURRENT_DATE, 9999-12-31].

Outros exemplos de remoção e de atualização correntes são respectivamente: "Pedro foi demitido como Professor Adjunto" e "Pedro foi promovido hoje a Professor Titular (cod_cargo = 739)", expressos pelos seguintes comandos SQL/Temporal:

```
DELETE FROM EMPREGOS
WHERE numero = 123
      AND cod_cargo = 071

UPDATE EMPREGOS
SET cod_cargo = 739
WHERE numero = 123
```

O comando de modificação a seguir estabelece um aumento salarial de 5% para Rita, vigorando a partir de 1/4/1995, e expressa uma atualização corrente.

```
UPDATE SALÁRIOS
SET valor = 1.05 * valor
WHERE numero = (SELECT S.numero
                FROM SALÁRIOS AS S, EMPREGADOS AS E
                WHERE nome = 'Rita'
                AND E.numero = S.numero)
```

A TAB. 15 exibe o resultado desta atualização.

TABELA 15 - Relação com suporte temporal resultado de UPDATE

numero	valor	Valid
6542	3300	[1995-02-01 – 9999-12-31)
3463	3400	[1995-02-02 – 1995-04-01)
3463	3570	[1995-04-01 – 9999-12-31)

Modificações seqüenciais

Uma modificação seqüencial se aplica a um período de aplicabilidade especificado imediatamente após a palavra-chave VALIDTIME. Por exemplo, seja a seguinte inserção

seqüencial: "Pedro foi contratado como Professor Adjunto para o período de 1997".

```
VALIDTIME PERIOD '[1997-01-01 - 1997-12-31]'
          INSERT INTO EMPREGOS
          VALUES (123, 071)
```

Outros exemplos, envolvendo remoção e atualização seqüenciais, são: "Pedro foi removido como Professor Adjunto para o período de 1997" e "Pedro foi promovido a Professor Titular para o período de 1997", respectivamente declarados pelos comandos

```
VALIDTIME PERIOD '[1997-01-01 - 1997-12-31]'
           DELETE FROM EMPREGOS
           WHERE numero = 123
                 AND cod_cargo = 071
VALIDTIME PERIOD '[1997-01-01 - 1997-12-31]'
           UPDATE EMPREGOS
           SET cod_cargo = 739
           WHERE numero = 123
```

Assumindo que a data de execução foi 1/7/1995, os comandos de remoção seguintes expressam que Maria tirou licença no segundo semestre de 1995.

```
VALIDTIME PERIOD '[1995-07-01 - 1996-01-01)'
          DELETE FROM SALÁRIOS
          WHERE numero = 6542
VALIDTIME PERIOD '[1995-07-01 - 1996-01-01)'
          DELETE FROM EMPREGADOS
          WHERE numero = 6542
```

Os conteúdos das TAB. 15 e 10 foram modificados respectivamente para os exibidos nas TAB. 16 e 17.

TABELA 16 – Nova tabela SALÁRIOS após DELETE

numero	valor	Valid
6542	3300	[1995-02-01 – 1995-07-01)
6542	3300	[1996-01-01 – 9999-12-31)
3463	3400	[1995-02-02 – 1995-04-01)
3463	3570	[1995-04-01 – 9999-12-31)

TABELA 17 – Nova tabela EMPREGADOS após DELETE

numero	nome	datanasc	Valid
5873	José	1961-03-21	[1995-02-01 – 9999-12-31)
6542	Maria	1963-07-04	[1995-02-01 – 1995-07-01)
6542	Maria	1963-07-04	[1996-01-01 – 9999-12-31)
3463	Rita	1970-03-09	[1995-02-02 – 9999-12-31)

Capítulo 5

Um formalismo para objetos temporais indeterminados

"E se Deus é canhoto
e criou a mão esquerda?
Isso explica, talvez, as coisas deste mundo."
Carlos Drummond de Andrade, HIPÓTESE

Neste capítulo é apresentada uma lógica denominada LITO (Logic of Indeterminate Temporal Objects), baseada em lógica de primeira-ordem e estendida com operadores modais e operadores temporais, para o tratamento de objetos temporais indeterminados.

5.1 Introdução

Em qualquer sistema convencional de banco de dados que se propõe a tratar da informação determinada (seção 3.2 e seção 4.3.6), o estado do sistema é representado pelo conjunto de instâncias num dado instante. Este é o tipo de representação feita pelo modelo relacional [Co79] onde o estado do mundo real é mapeado através de um conjunto de relações [SKS99]. Todavia, para o tratamento de informação indeterminada (seção 3.2) esta representação tornar-se-ia impraticável devido ao "tamanho potencial" do conjunto de instâncias alternativas que representariam os vários estados possíveis de um objeto num dado instante. Em decorrência disso, um método de representação indireta deveria ser usado, como por exemplo *a lógica de base de conhecimento incompleto* proposta por Levesque [Le94].

O formalismo de uma teoria é importante, pois fornece subsídios para que se definam conceitos através de uma linguagem precisa e com fundamento matemático. Foi visto que a informação indeterminada, seja esta temporal ou não, é intrínseca ao domínio de várias aplicações reais e, portanto, seria importante ter seus conceitos formalizados. A for-

malização de uma teoria para representação e manipulação de objetos temporais indeterminados é a principal motivação desta pesquisa. Entenda-se por manipulação as operações que envolvam consulta, inserção, remoção, atualização e refinamento de objetos em BDTI.

5.2 Análise dos formalismos relacionados

Nesta seção serão analisados os trabalhos abordados no capítulo 4, em especial suas limitações para com o tratamento de objetos temporais indeterminados e suas contribuições para a representação, recuperação, inserção, remoção e atualização de objetos em geral.

A maioria das linguagens de consulta a banco de dados temporais propostas abordam apenas extensões da operação de consulta ou recuperação, principalmente no que se refere ao comando SELECT da SQL-92 [MS93]. SQL/Temporal (seção 4.5) e uma extensão de SQL-92 (seção 4.4), ambas propostas por Snodgrass [Sn00], são primeiras linguagens que abordaram operações de modificação temporal, sendo o tempo uma informação do tipo determinada (seção 3.2).

Segundo Snodgrass [Sn00], "a maioria da pesquisa em bancos de dados temporais tem se concentrado nas consultas; a modificação sempre é postergada ... a implementação da modificação em tabelas temporais não tem recebido atenção.". Em decorrência disso, a formalização de uma teoria para as operações de modificação em banco de dados temporais é inexistente, assim como para as operações de manipulação em BDTIs.

A *teoria relacional generalizada* com informação indeterminada [Re94] utiliza uma abordagem lógica na representação de apenas dois tipos de valores indeterminados: os valores nulos do tipo desconhecido e os valores disjuntivos (seção 3.2). Esta teoria não descreve a informação negativa nem trata de questões relacionadas com a informação temporal, tendo portanto representações limitadas àquelas duas formas de indeterminação.

O formalismo de TSQL2 [DS98] envolve apenas a operação de recuperação de registros em tabelas tempo de validade (seção 4.3.3), ou seja, não abrange as operações de modificação. Por isso, limita sobremaneira o poder de expressão de sua linguagem de consulta. Por outro lado, o significado da indeterminação temporal só é preservado se a *credibilidade* escolhida pelo usuário for a do tipo INDETERMINATE. Os outros três tipos de *credibilidade* anulam completamente o significado original expresso pelo atributo tempo-

ral, alterando seu valor para um tipo determinado. Quanto à *plausibilidade* (seção 4.2.6), trata-se de um mecanismo que controla a construção de uma resposta permitindo ao usuário especificar um valor percentual acima do qual o sistema retornará respostas satisfatórias; perceba que é um parâmetro restrito a aplicações sob a abordagem probabilística.

O efeito negativo destas estratégias de substituição é que elas comprometem a usabilidade desejável em linguagens de interação do usuário com um sistema computacional, ou seja, a abordagem probabilística, embora útil em algumas situações, compromete sobremaneira a facilidade de especificação de uma consulta, pois obriga o usuário a assimilar conceitos nada triviais para se expressar convenientemente.

Com relação as propostas SQL/Temporal e a extensão a SQL-92 foi percebido que estas tratam apenas de períodos do tipo determinado, ao contrário da TSQL2. Por exemplo, as tabelas da extensão a SQL-92 possuem as colunas `data_início` e `data_fim` representando o tempo de validade do registro, enquanto que as tabelas de SQL/Temporal possuem uma coluna, chamada `Valid`, com a mesma finalidade. Suas operações para modificação de um banco de dados são extensões importantes, embora preliminares, pois abrangem apenas períodos do tipo determinados. Também observou-se que tais propostas não incorporam qualquer tipo de indeterminação, seja esta temporal ou não. Já TSQL2 provê apenas uma forma de representação para instantes indeterminados e outra para representar períodos indeterminados. Por exemplo, a notação 10/5 ~ 29/5 é usada para denotar um instante entre 10 e 29 de maio, e a notação [1/3/99 ~ 31/3/99 – 1/6/99 ~ 30/6/99] é empregada para denotar um conjunto de períodos possiveis.

A *Lógica de Base de Conhecimento Incompleta* [Le94] é apropriada para representação de conhecimento incompleto sob a suposição do mundo aberto (seção 3.1), embora sua linguagem KL não suporte modificações numa BC.

5.3 A sintaxe da lógica para objetos temporais indeterminados (LITO)

Esta seção se destina a apresentação da LITO. São descritas inicialmente as decisões tomadas à solução das limitações dos formalismos analisados na seção 5.2; e à adição de alguns de seus conceitos na LITO.

A LITO estende a *teoria relacional generalizada* com informação indeterminada

[Re94] através da incorporação de mais dois tipos de representações para a indeterminação não-temporal (a informação desconhecida inexistente (seção 3.2) e a informação negativa (seção 6.1)), além da representação para a informação temporal indeterminada. Todas estas formas de representação objetivam modelar com mais realismo certas aplicações.

A forma de representação de períodos indeterminados em TSQL2, é adicionada à LITO através do operador temporal *during*. Por exemplo, o período indeterminado [1/3/99 ~ 31/3/99 – 1/6/99 ~ 30/6/99] é representado na LITO pelo intervalo indeterminado [*during*[1/3/99, 31/3/99], *during*[1/6/99, 30/6/99]] o qual denota um intervalo que vale a partir de algum dia de março de 1999, até algum dia de junho de 1999. Esta forma de representação de intervalos indeterminados é estendida para suportar outros operadores (*before*, *after* e *), aumentando ainda mais o poder de expressão de uma linguagem de consulta a banco de dados temporais. Por exemplo, o intervalo indeterminado [*before* 1/3/99, *during*[1/6/99, 30/6/99]] é representado pelo intervalo [*during*[1/1/1, 28/2/99], *during*[1/6/99, 30/6/99]] e denota um intervalo indeterminado que vale desde algum dia entre 1/1/1 e 28/2/99, ou seja, algum dia antes de 1/3/1999, até algum dia entre 1/6/99 e 30/6/99. A combinação paritária destes operadores temporais oferece uma maior variedade de representação do tempo (TAB. 20, seção 6.1), capturando com mais fidelidade a realidade que se pretende modelar.

SQL/Temporal e a extensão a SQL-92 são propostas pioneiras para extensões dos comandos de modificação da SQL-92, todavia se limitam apenas à representação de atributos temporais do tipo determinado, não tratando a indeterminação temporal. A LITO propõe formas de estender linguagens de consulta como estas, através da representação e da manipulação da informação temporal indeterminada.

Quanto à *Lógica de Bases de Conhecimento Incompleta* proposta por Levesque [Le94], foi adotada sua solução, por se tratar de um método de representação adequado à representação de objetos sob a OWA (seção 3.1). Entretanto, em virtude deste método não contemplar a indeterminação temporal nem a modificação de objetos, foi necessária a elaboração de uma nova axiomática e de regras de dedução para objetos temporais indeterminados.

A LITO é composta por uma tripla <*L*, *A*, *R*> onde *L* é a linguagem da lógica, *A* é o conjunto de axiomas que caracteriza os operadores (modais e temporais) e *R* é o conjunto de regras a ser usado para a derivação de novos fatos (*modus ponens*) e no processamento

de consultas de objetos em BDTIs.

5.3.1 Alfabeto

Nesta seção, é introduzido um alfabeto mínimo de símbolos, sendo que outros símbolos úteis serão introduzidos por meio de definições na seção 5.3.4.

Símbolos lógicos:

- *pontuação*: "(", ")", "[", "]", ","
- *conectivos*: $\neg$, $\vee$
- *quantificador*: $\forall$
- *operadores modais*: *M, K*
- *operadores temporais*: *during, begin, end*
- *operadores não-temporais*: *not, or, lt, in*
- *variáveis*:
    - não-temporais: x, x_1, x_2,...
    - instante temporal: ∞, *now*, t, t_1, t_2,...
    - intervalo temporal: I, I_1, I_2,...

Os operadores modais básicos são o *Maybe* (possível), para descrever a informação indeterminada, e o *Known* (conhecido) para caracterizar a informação representada no BDTI. Tais operadores possuem a mesma significação empregada na lógica modal [Ch80, HC96] e na lógica de bases de conhecimento incompleto [Le94], respectivamente. O significado dos operadores temporais será visto na seção 5.3.5. Os operadores não-temporais permitem descrever informação não-temporal indeterminada, ou seja, a informação imprecisa, desconhecida e negativa, segundo a classificação feita na seção 6.1. O conjunto de variáveis é deixado em aberto, com as seguintes restrições: variáveis não-temporais começam sempre com o caracter *x*, variáveis instante temporal começam sempre com o caracter *t*, exceto a variável especial *now*, e variáveis intervalo temporal começam sempre com o caracter *I*. A variável temporal ∞ é chamada de posicional porque seu conteúdo depende de sua posição no intervalo indeterminado (seção 6.1). Os dois conectivos são necessários

para facilitar as definições de outros símbolos (seção 5.3.4).

Símbolos não-lógicos:

- *constantes*:
  - temporais: *.
  - não-temporais: *, *none* e outras cadeias de caracteres.
- *símbolos predicativos*: *Time*, <, =, P, P_1, P_2,...

A constante temporal * serve à representação tanto de valores indeterminados temporais, quanto de valores indeterminados não-temporais. Emprega-se este símbolo pelo fato de ser bastante utilizado na área de banco de dados para denotar um valor nulo [Co79]. A constante não-temporal *none* denota uma informação do tipo desconhecida inexistente (seção 6.1).

O conjunto dos símbolos predicativos é deixado em aberto, com a restrição de que sempre comecem com o símbolo P, exceto em três casos *Time*, < e =. O símbolo predicativo *Time* é utilizado para denotar a parte temporal de um objeto. Os símbolos predicativos < e = denotam as relações de ordem total e de igualdade, respectivamente. Estes símbolos servem para a comparação de termos instante temporal definido e de termos intervalo temporal. O símbolo predicativo = serve também para comparar termos não-temporais. Estes dois últimos símbolos predicativos são utilizados na definição de seis outros símbolos da linguagem *L* (seção 5.3.4).

Todos os símbolos predicativos são binários, exceto P que é unitário e cujo argumento é a variável não-temporal *x* para identificar um objeto (seção 6.1). Os símbolos predicativos P_i possuem esta mesma variável como primeiro argumento e um termo não-temporal (seção 5.3.2) como segundo argumento. O símbolo predicativo *Time* denota o tempo de validade de um objeto e possui a variável não-temporal *x* como primeiro argumento e um termo temporal (seção 5.3.2) como segundo argumento.

5.3.2 Termos

Termos são expressões da linguagem *L* usadas para denotar informações, determinadas e indeterminadas, passíveis de serem armazenadas num BDTI. Os termos de *L* são

classificados em termos não-temporais, termos instante temporal e termos intervalo temporal. O conjunto dos termos de L é o menor conjunto satisfazendo às seguintes condições:

Termos não-temporais:

- toda constante não-temporal é um termo não-temporal;
- toda variável não-temporal é um termo não-temporal;
- se x é uma variável não-temporal e Tx, Tx_1 e Tx_2 são termos não-temporais, então *or*(Tx_1, Tx_2), *not*(Tx) e *lt* x e são termos não-temporais.

Termos instante temporal definido e instante temporal indefinido:

- as constantes temporais * e ∞ são termos instante temporal indefinidos;
- toda variável instante temporal é um termo instante temporal definido;
- se TI é um termo intervalo temporal, então *during* TI é um termo instante temporal indefinido;
- se TI é um termo intervalo temporal, então *begin* TI e *end* TI são termos instante temporal definidos.

A classificação de termos definidos e indefinidos é devido ao fato de que a relação de ordem (<) não se aplica a termos instante temporal indefinidos (* e *during* TI), pois estes não definem exatamente o instante temporal que representam. Esta relação só se aplica a termos instante temporal definidos.

A denominação "termo instante temporal" é utilizada para descrever tanto termos instante temporal definidos, quanto termos instante temporal indefinidos.

Termos intervalo temporal:

- toda variável intervalo temporal é um termo intervalo temporal;
- se Tt_1 e Tt_2 são termos instante temporal, então [Tt_1, Tt_2] é um termo intervalo temporal.

5.3.3 Fórmulas

Fórmulas são expressões da linguagem *L* usadas para denotar fatos no domínio da aplicação representada. Uma fórmula, ou fórmula bem-formada (fbf), é definida recursivamente como:

- se x é uma variável não-temporal, Tx, Tx_1 e Tx_2 são termos não-temporais e TI é um termo intervalo temporal, então $Tx_1 = Tx_2$, P(x), P(x, Tx), *Time*(x, TI) são fbfs;
- se F e G são fbfs, então ¬F, F ∨ G, *M*F e *K*F são fbfs;
- se F é uma fbf e v é uma variável não-temporal, instante temporal, ou intervalo temporal qualquer, então ∀v (F) é uma fbf;
- se Tt_1 e Tt_2 são termos instante temporal definidos, então $Tt_1 = Tt_2$ e $Tt_1 < Tt_2$ são fbfs;
- se TI_1 e TI_2 são termos intervalo temporal, então $TI_1 = TI_2$ é uma fbf.

Note que o símbolo predicativo P pode expressar tanto uma classe de objetos, quanto os atributos desta classe. No caso de expressar uma classe, P será o nome da classe e a variável x conterá o identificador dos objetos da classe. Um atributo de um objeto x será representado por um símbolo predicativo binário P(x, Tx), onde x é a variável que identifica o objeto na classe, Tx representa o valor do atributo e P é o nome do atributo.

5.3.4 Definições

As definições servem para denotar novos símbolos da linguagem *L*. Eles são elaborados a partir dos símbolos já especificados. Portanto, se F e G são fbfs, definem-se os seguintes símbolos:

1. $\exists x\,(F) \Leftrightarrow_{def} \neg\forall x\,(\neg F)$
2. $F \wedge G \Leftrightarrow_{def} \neg\,(\neg F \vee \neg G)$
3. $F \rightarrow G \Leftrightarrow_{def} \neg F \vee G$
4. $F \leftrightarrow G \Leftrightarrow_{def} (F \rightarrow G) \wedge (G \rightarrow F)$

As definições de 1 a 4 são clássicas e definem a quantificação existencial, a

conjunção, a implicação e a equivalência lógica, respectivamente. As duas definições seguintes se aplicam somente a termos instante temporal definidos.

5. $Tt_1 \leq Tt_2 \Leftrightarrow_{def} (Tt_1 < Tt_2) \vee (Tt_1 = Tt_2)$

6. $Tt_1 \in [Tt_2, Tt_3] \Leftrightarrow_{def} (Tt_2 \leq Tt_1) \wedge (Tt_1 \leq Tt_3)$

A relação de pertinência caracteriza o intervalo temporal como um conjunto linear e conexo.

7. $I = [t_1, t_2] \Leftrightarrow_{def} t_1 = \textit{begin}\ I \wedge t_2 = \textit{end}\ I$

Um intervalo I representado por $[t_1, t_2]$ é delimitado pelos instantes temporais definidos t_1 e t_2, onde t_1 é o instante inicial de I e t_2 é o instante final de I.

8. $I_1 \subseteq I_2 \Leftrightarrow_{def} (\textit{begin}\ I_2 \leq \textit{begin}\ I_1) \wedge (\textit{end}\ I_1 \leq \textit{end}\ I_2)$

A relação "está contido em ou é igual a" entre intervalos é definida em função dos limites de cada um dos intervalos.

As definições 9 e 10, a seguir, utilizam a estrutura linear e discreta do tempo e expressam respectivamente os operadores sucessor (*succ*) e predecessor (*pred*) para instantes temporais.

9. $t_2 = \textit{succ}\ t_1 \Leftrightarrow_{def} (t_1 < t_2 \wedge \forall t_3\ (t_1 < t_3 \rightarrow (t_2 \leq t_3)))$

10. $t_2 = \textit{pred}\ t_1 \Leftrightarrow_{def} \textit{succ}\ t_2 = t_1$

A partir dos termos da linguagem usados para expressar indeterminação não-temporal (*lt* ( ), *not*( ) e *or*( )) foram definidos outros termos para novas formas de se representar a indeterminação não-temporal *le* ( ), *gt* ( ) e *ge* ( ).

11. $\textit{le}\ x \Leftrightarrow_{def} \textit{or}(\text{lt}\ x, x)$

12. $\textit{gt}\ x \Leftrightarrow_{def} \textit{not}(\text{le}\ x)$

13. $\textit{ge}\ x \Leftrightarrow_{def} \textit{not}(\text{lt}\ x)$

Para expressar um valor de indeterminação não-temporal pertencente a um conjunto de possíveis valores, foi definido o termo não-temporal *in* (x_1, x_2), sendo x_1 o primeiro elemento e x_2 o último deste conjunto.

14. $\textit{in}(x_1, x_2) \Leftrightarrow_{def} (\textit{ge}\ x_1 \wedge \textit{le}\ x_2)$

Um instante temporal totalmente indeterminado é indicado por *.

15. * $\Leftrightarrow_{def}$ *during* [∞, ∞]

16. *before* t $\Leftrightarrow_{def}$ *during* [∞, *pred* t]

    O operador temporal *before* representa um instante temporal indeterminado e é definido em função do operador temporal *during*, da constante temporal ∞ e do operador *pred*, pois o instante t deve ser excluído do domínio de valores possíveis de *before* t.

17. *after* t $\Leftrightarrow_{def}$ *during* [*succ* t, ∞]

    O operador temporal *after* representa um instante temporal indeterminado após um dado instante t e é definido em função do operador temporal *during*, da constante temporal ∞ e do operador *succ*.

    Os valores assumidos pela variável temporal ∞, nestas três últimas definições, dependerão da localização do operador temporal no intervalo indeterminado (TAB. 20).

5.3.5 Axiomática

Nesta seção é introduzido um sistema de axiomas necessários para o processamento de consultas. Para melhor exposição dos axiomas, estes foram classificados em básicos, temporais e não-temporais em função dos símbolos predicativos envolvidos.

Axiomas básicos:

Os axiomas básicos, assim chamados porque são de uso comum à definição dos axiomas temporais e não-temporais, envolvem os axiomas da lógica de primeira-ordem e o predicado de ordem total (≤).

1. Todos os axiomas da lógica de primeira-ordem
2. O símbolo ≤ denota a relação de ordem total

Axiomas modais:

Os axiomas modais são assim chamados porque caracterizam as modalidades

Maybe (*M*) e *Known* (*K*), além da negação.

3. $\forall x\ (F \vee G) \rightarrow MF$

   Cada componente de uma conjunção pode ser verdadeiro.

4. $\forall x\ KF \rightarrow F$

   Podem existir verdades não conhecidas.

5. $\forall x\ F \rightarrow MF$

   Todo fato verdadeiro é possível.

6. $\forall x\ \neg K\neg F \rightarrow MF$

   Fatos que não são conhecidos como falsos, possivelmente são verdadeiros (suposição do mundo aberto, seção 3.1).

Axiomas não-temporais:

Os axiomas não-temporais envolvem apenas predicados binários do tipo P(x, Tx), onde P denota o nome de um atributo não-temporal de um objeto.

7. $\forall x \forall x_1 \forall x_2\ (KP(x, or(x_1, x_2)) \rightarrow MP(x, x_1))$

   Quando o valor de um predicado P armazenado no BDTI é uma disjunção de valores, então cada um destes valores é possivelmente verdadeiro. Note que também valerá $MP(x, x_2)$ uma vez que a disjunção é comutativa.

8. $\forall x \forall x_1\ (KP(x, lt\ x_1) \rightarrow \exists x_2\ (x_2 < x_1 \wedge P(x, x_2)) \wedge \neg KP(x, x_2)$

   O termo não-temporal $lt\ x_1$ denota que existe um valor válido menor que x_1, mas ele não é conhecido pelo BDTI.

9. $\forall x \forall x_1\ (KP(x, not\ x_1) \rightarrow \neg KP(x, x_1))$

   No BDTI não podem haver contradições entre valores conhecidos.

10. $\forall x \forall x_1\ (KP(x, *) \rightarrow \exists x_1\ P(x, x_1) \wedge \neg KP(x, x_1))$

    Este axioma representa uma generalização do axioma 8 para o caso de não se saber nada sobre algum valor, além de sua existência em si.

11. $\forall x\ (KP(x, none)) \rightarrow \forall x_1\ \neg P(x, x_1)$

O valor nulo não se aplica ao objeto em consideração.

Axiomas temporais:

Os axiomas temporais envolvem predicados *Time*(x, TI), ou seja, predicados que determinam o tempo de validade de um objeto.

12. $\forall x \forall I$ (*Time*(x, I) $\leftrightarrow \forall t$ (t $\in$ I $\rightarrow$ *Time*(x, [t, t])))

    O tempo de um objeto é válido no intervalo I, se e somente se for válido para todos os instantes deste.

13. $\forall x \forall I$ (*MTime*(x, I) $\leftrightarrow \exists t$(t $\in$ I $\wedge$ *Time*(x, [t, t]) $\wedge \neg$*KTime*(x, [t, t])))

    O tempo de um objeto é possivelmente válido no intervalo I, se e somente se for válido para pelo menos um instante de I que não seja conhecido pelo BDTI.

14. $\forall x \forall t_1$ (*Time*(x, [∞, t_1]) $\leftrightarrow \forall t_2(t_2 \leq t_1 \rightarrow$ *Time*(x, [t_2, t_1])))

15. $\forall x \forall t_1$ (*Time*(x, [t_1, ∞]) $\leftrightarrow \forall t_2(t_1 \leq t_2 \rightarrow$ *Time*(x, [t_1, t_2])))

    Os axiomas 14 e 15 determinam o significado da variável temporal ∞, dependendo de sua posição no intervalo.

16. $\forall x \forall t_1 \forall t_2$ (*Time*(x, [t_1, t_2]) $\rightarrow$ ($t_1 \leq t_2$))

    O instante inicial de um intervalo deve ser menor ou igual ao instante final.

17. $\forall x \forall I \forall t_1$ (*Time*(x, [*during* I, t_1]) $\rightarrow \exists t$ (t $\in$ I $\wedge$ *Time*(x, [t, t_1]) $\wedge \neg$*KTime*(x, [t, t_1]))

    O intervalo representado por [*during* I, t_1] é válido desde um instante inicial t pertencente a I e desconhecido pelo BDTI.

18. $\forall x \forall t_1 \forall I$ (*Time*(x, [t_1, *during* I]) $\rightarrow \exists t$ (t $\in$ I $\wedge$ *Time*(x, [t_1, t]) $\wedge \neg$*KTime*(x, [t_1, t]))

    O intervalo representado por [t_1, *during* I] é válido até um instante final t pertencente a I e desconhecido pelo BDTI.

Objetivando clareza na exposição das regras, estas foram classificadas em: regra básica e regras de recuperação (não-temporais e temporais).

Regra básica:

1. se $\vdash F$ e $\vdash F \rightarrow G$ então $\vdash G$ (*modus ponens*)

Regras de recuperação:

As regras para processamento de consultas são dadas como cláusulas de *Horn* ou fórmulas redutíveis a cláusulas de *Horn*. Quando uma consulta é formulada, a variável x do predicado binário da cabeça da regra deve ser instanciada [BCS+96]. A outra variável é livre e conterá o resultado da aplicação da regra, ou seja, conterá a resposta à consulta.

Regras de recuperação não-temporais:

Estas regras não envolvem recuperação de valores temporais. O valor do argumento x_1, especificado pelo predicado $P_1(x, x_1)$ de uma consulta, pode casar completamente ou parcialmente com um valor do mesmo predicado no BDTI. O conjunto de valores possíveis de um atributo indeterminado (por exemplo, salario(x, *lt* 1000)) é chamado de domínio do predicado. O casamento é completo se os domínios dos predicados da consulta e do BDTI são exatamente os mesmos. Caso contrário, ou seja, caso haja qualquer outra interseção entre os domínios destes, o casamento é parcial. Se nenhum destes casos for verificado, então não há casamento algum e o valor verdade resultante desta comparação é *desconhecido*.

O segundo argumento do predicado $P_2(x, x_2)$, da cabeça da regra, especifica os valores recuperados do BDTI qualificados pelos valores verdade possível (M), não ($\neg$) ou sim (ausência de símbolo). Por exemplo, a consulta " qual o salário de Maria?" será satisfeita pela seguinte regra:

$$\texttt{empregado(x, maria)} \wedge \textit{K}\texttt{empregado(x, maria)} \wedge \textit{K}\texttt{salario(x, } x_2\texttt{)} \rightarrow \texttt{salario(x, } x_2\texttt{)}$$

Para clareza de exposição, as regras de recuperação não-temporal foram divididas em dois grupos. No primeiro grupo estão as regras cujos valores do argumento x_1 dos predicados de uma consulta casam completamente, e no segundo grupo estão as regras onde estes valores casam parcialmente.

O primeiro grupo é expresso pelas seguintes regras:

2. $P_1(x, x_1) \wedge KP_1(x, x_1) \wedge KP_2(x, x_2) \rightarrow P_2(x, x_2)$
3. $P_1(x, x_1) \wedge KP_1(x, x_1) \wedge KP_2(x, or(x_2, x_3)) \rightarrow MP_2(x, x_2)$
4. $P_1(x, x_1) \wedge KP_1(x, x_1) \wedge KP_2(x, lt\ x_2) \rightarrow ((x_3 < x_2) \rightarrow MP_2(x, x_3))$
5. $P_1(x, x_1) \wedge KP_1(x, x_1) \wedge KP_2(x, in(x_2, x_3)) \rightarrow ((x_2 \leq x_4) \wedge (x_4 \leq x_3) \rightarrow MP_2(x, x_4))$
6. $P_1(x, x_1) \wedge KP_1(x, x_1) \wedge KP_2(x, *) \rightarrow MP_2(x, x_2)$
7. $P_1(x, x_1) \wedge KP_1(x, x_1) \wedge KP_2(x, not\ x_2) \rightarrow (\forall x_3\ \neg(x_3 = x_2) \rightarrow MP_2(x, x_3))$

O segundo grupo é descrito pelas seguintes regras:

8. $P_1(x, x_1) \wedge KP_1(x, or(x_1, x_3)) \wedge KP_2(x, x_2) \rightarrow MP_2(x, x_2)$
9. $P_1(x, x_1) \wedge KP_1(x, lt\ x_3) \wedge (x_1 < x_3) \wedge KP_2(x, x_2) \rightarrow MP_2(x, x_2)$
10. $P_1(x, x_1) \wedge KP_1(x, in(x_3, x_4)) \wedge (x_3 \leq x_1) \wedge (x_1 \leq x_4) \wedge KP_2(x, x_2) \rightarrow MP_2(x, x_2)$
11. $P_1(x, x_1) \wedge KP_1(x, *) \wedge KP_2(x, x_2) \rightarrow MP_2(x, x_2)$
12. $P_1(x, x_1) \wedge KP_1(x, not\ x_1) \wedge KP_2(x, x_2) \rightarrow \neg P_2(x, x_2)$

Regras de recuperação temporais:

As regras de recuperação temporais envolvem a recuperação de valores temporais. Para exposição mais ordenada destas regras, estas também foram agrupadas. No primeiro grupo estão as regras que recuperam valores de atributos que satisfazem dada restrição temporal, e no segundo grupo estão as que recuperam valores temporais que obedecem uma dada restrição não-temporal.

As respostas de uma consulta a objetos temporais dependem da relação entre os tempos do predicado da consulta e o predicado do BDTI.

O primeiro grupo é expresso pelas seguintes regras:

13. $Time(x, I_1) \wedge KP(x, x_1) \wedge KTime(x, I_2) \wedge I_1 \subseteq I_2 \rightarrow P(x, x_1)$
14. $Time(x, I_1) \wedge KP(x, x_1) \wedge KMTime(x, I_2) \wedge I_1 \subseteq I_2 \rightarrow MP(x, x_1)$

15. $MTime(x, I_1) \wedge K\mathrm{P}(x, x_1) \wedge KMTime(x, I_2) \wedge I_2 \subseteq I_1 \rightarrow \mathrm{P}(x, x_1)$

16. $MTime(x, I_1) \wedge \exists t(K\mathrm{P}(x, x_1) \wedge KTime(x, I_2) \wedge t \in I_1 \wedge t \in I_2) \rightarrow \mathrm{P}(x, x_1)$

17. $MTime(x, I_1) \wedge \exists t(K\mathrm{P}(x, x_1) \wedge KMTime(x, I_2) \wedge t \in I_1 \wedge t \in I_2) \rightarrow M\mathrm{P}(x, x_1)$

Em uma consulta ao tempo de um objeto, o valor verdade da resposta será estabelecido pela precisão no predicado *Time* e/ou no predicado não-temporal.

O segundo grupo é expresso pelas seguintes regras:

18. $\mathrm{P}(x, x_1) \wedge K\mathrm{P}(x, x_1) \wedge KTime(x, I) \rightarrow Time(x, I)$

19. $\mathrm{P}(x, x_1) \wedge K\mathrm{P}(x, x_1) \wedge KMTime(x, I) \rightarrow MTime(x, I)$

20. $\mathrm{P}(x, x_1) \wedge KM\mathrm{P}(x, x_1) \wedge KTime(x, I) \rightarrow MTime(x, I)$

21. $\mathrm{P}(x, x_1) \wedge KM\mathrm{P}(x, x_1) \wedge KMTime(x, I) \rightarrow MTime(x, I)$

5.4 Transformação de intervalos com termos instantes indefinidos

Intervalos temporais indeterminados são aqueles que contêm pelo menos um dos seus limites indeterminados. Um limite indeterminado é representado pelo instante temporal indefinido *during* TI, onde TI é um termo intervalo temporal, ou pelas suas outras formas (*, *before* t e *after* t) descritas respectivamente pelas definições 15, 16 e 17.

Os axiomas 17 e 18 descrevem a validade de predicados temporais com um dos limites indeterminados. No caso em que os dois extremos são indeterminados, vale a proposição demonstrada a seguir. Com esta proposição é possível transformar todos os intervalos temporais indeterminados em uma conjunção de três predicados temporais, sendo que o predicado ao centro desta conjunção é qualificado como do tipo determinado, enquanto que os outros predicados são qualificados como do tipo possível. Esta conjunção representa com mais clareza a proposição original, pois realça a determinação existente entre os limites de um intervalo indeterminado.

PROPOSIÇÃO: $\forall x \forall I_1 \forall I_2 \forall t$ ($Time(x, [during\ I_1, during\ I_2]) \rightarrow$
$MTime(x, I_1) \wedge Time(x, [end\ I_1, begin\ I_2]) \wedge MTime(x, I_2)$).

Para fins de facilitar a demonstração, cada passo da seqüência foi enumerado e, ao

lado direito de cada um destes, foi colocada uma notação entre colchetes indicando o número do passo originário deste, e o número da definição (D) ou do axioma (A) usado na sua derivação.

Primeiramente será demonstrado o seguinte lema:

Lema: $\forall x \forall I_1 \forall I_2 \, (\mathit{Time}(x, I_2) \wedge (I_1 \subseteq I_2) \rightarrow \mathit{Time}(x, I_1))$

1 $\forall x \forall I_1 \forall I_2 \, (\mathit{Time}(x, I_2) \wedge (\mathit{begin}\ I_2 \leq \mathit{begin}\ I_1) \wedge (\mathit{end}\ I_1 \leq \mathit{end}\ I_2))$[hipótese com D8]

Foi aplicada a definição da relação "está contido em ou é igual a".

2 $\forall x \forall I_1 \forall I_2 \, (\mathit{Time}(x, I_2) \wedge ((\mathit{begin}\ I_2 < \mathit{begin}\ I_1) \vee (\mathit{begin}\ I_2 = \mathit{begin}\ I_1)) \wedge$
$((\mathit{end}\ I_2 < \mathit{end}\ I_1) \vee (\mathit{end}\ I_2 = \mathit{end}\ I_1))$ [1 com D5]

A definição da relação de ordem total (≤) foi aplicada.

3. $\forall t \forall x \forall I_1 \forall I_2 \, (t \in I_2 \rightarrow \mathit{Time}(x, [t, t]) \wedge (\mathit{begin}\ I_2 = \mathit{begin}\ I_1) \wedge (\mathit{end}\ I_2 = \mathit{end}\ I_1))$
[2 com A12]

A substituição do predicado $\mathit{Time}(x, I_2)$ pela sua fórmula expressa no axioma 12.

4. $\forall t \forall x \forall I_1 \forall I_2 \, (t \in [\mathit{begin}\ I_2, \mathit{end}\ I_2] \rightarrow \mathit{Time}(x, [t, t]) \wedge (\mathit{begin}\ I_2 = \mathit{begin}\ I_1) \wedge$
$(\mathit{end}\ I_2 = \mathit{end}\ I_1))$ [3 com D7]

A substituição do intervalo I_2 segundo a definição de limites de um intervalo.

5. $\forall t \forall x \forall I_1 \, (t \in [\mathit{begin}\ I_1, \mathit{end}\ I_1] \rightarrow \mathit{Time}(x, [t, t]))$ [4 com A1]

Foi aplicada a relação de igualdade da lógica de primeira-ordem.

6. $\forall t \forall I_1 \forall x \, (t \in I_1 \rightarrow \mathit{Time}(x, [t, t]))$ [5 com D7]

A substituição do intervalo $[\mathit{begin}\ I_1, \mathit{end}\ I_1]$ segundo a definição de limites de um intervalo.

7. $\forall x \forall I_1\ \mathit{Time}(x, I_1)$ [6 com A12]

A substituição da fórmula pelo predicado equivalente.

Demonstração da proposição:

1ª parte: $\forall x \forall I_1 \forall I_2 (Time(x, [during\ I_1, during\ I_2]) \rightarrow MTime(x, I_1)$

1. $\forall x \forall I_1 \forall I_2 \exists t_1 (t_1 \in I_1 \wedge Time(x, [t_1, during\ I_2]))$ [hipótese com A17]

   A aplicação da fórmula expressa no axioma 17 no limite inicial do intervalo.

2. $\forall x \forall I_1 \exists t_1 (t_1 \in I_1 \wedge Time(x, [t_1, t_1]))$ [1 com Lema]

   O intervalo $[t_1, t_1]$ está contido no intervalo $[t_1, during\ I_2]$.

3. $\forall x \forall I_1\ MTime(x, I_1)$ [2 com A13]

   A substituição da fórmula pelo predicado equivalente.

2ª parte: $\forall x \forall I_1 \forall I_2 (Time(x, [during\ I_1, during\ I_2]) \rightarrow Time(x, [end\ I_1, begin\ I_2])$

4. $\forall x \forall I_1 \forall I_2 \exists t_1 \exists t_2 (t_1 \in I_1 \wedge t_2 \in I_2 \wedge Time(x, [t_1, t_2]))$ [1 com A18]

   A aplicação da fórmula expressa no axioma 18 no limite final do intervalo.

5. $\forall x \forall I_1 \forall I_2 \exists t_1 \exists t_2 (t_1 \in [begin\ I_1, end\ I_1] \wedge t_2 \in [begin\ I_2, end\ I_2] \wedge Time(x, [t_1, t_2]))$ [4 com D7]

   As substituições dos intervalos I_1 e I_2 segundo a definição de limites de um intervalo.

6. $\forall x \forall I_1 \forall I_2 \exists t_1 \exists t_2 (t_1 \leq end\ I_1) \wedge (begin\ I_2 \leq t_2) \wedge Time(x, [t_1, t_2]))$ [5 com D6]

   A definição da relação de pertinência foi aplicada duas vezes.

7. $\forall x \forall I_1 \forall I_2 \exists t_1 \exists t_2 ([end\ I_1, begin\ I_2] \subseteq [t_1, t_2] \wedge Time(x, [t_1, t_2]))$ [6 com D8]

   Foi aplicada a definição da relação "está contido em ou é igual a".

8. $\forall x \forall I_1 \forall I_2\ Time(x, [end\ I_1, begin\ I_2])$ [7 com Lema]

3ª parte: $\forall x \forall I_1 \forall I_2 (Time(x, [during\ I_1, during\ I_2]) \rightarrow MTime(x, I_2)$

9. $\forall x \forall I_1 \forall I_2 \exists t_1 (t_1 \in I_2 \wedge Time(x, [during\ I_1, t_1]))$ [hipótese com A18]

A aplicação da fórmula expressa no axioma 18 no limite final do intervalo.

10. $\forall x \forall I_1 \exists t_1 \, (t_1 \in I_1 \wedge \mathit{Time}(x, [t_1, t_1]))$ [9 com Lema]

O intervalo $[t_1, t_1]$ está contido no intervalo [*during* I_1, t_1].

11. $\forall x \forall I_2 \, \mathit{MTime}(x, I_2)$ [10 com A13]

A substituição da fórmula pelo predicado equivalente □.

5.5 Modificação de um BDTI

Os axiomas e as regras descritos na seção 5.3.5 descrevem as regras que regem a funcionalidade de um BDTI e o processamento adequado de consultas a este banco. Entretanto, como todo banco de dados, um BDTI não permanece estático. Ao contrário, ele está sujeito a mudanças que refletem novos estados do mundo que está sendo descrito. No caso particular de um BDTI também deve ser possível incorporar informações mais precisas sobre um mesmo estado do banco de dados.

Uma axiomática, como a da seção 5.3.5, é utilizada para verificar se um certo banco de dados satisfaz às condições de ser válido ou não, de acordo com esta axiomática. Todavia, as mudanças em si não fazem parte deste sistema. Se um certo estado de um banco de dados pode ser considerado um modelo da teoria subjacente, uma mudança deste estado significa construir um novo modelo da mesma teoria.

Uma analogia do processo de modificação de um BDTI, realizado pelas operações de inserção, remoção lógica, refinamento e atualização de objetos temporais indeterminados, pode ser feita com o processo de modificação de uma base de conhecimento Prolog [CNV88].

Um BDTI, no sistema a ser abordado no capítulo 6, é dado por um conjunto de fatos, ou seja, um conjunto de símbolos predicativos ou cláusulas unitárias. Por exemplo, `emprego(`O_1`)` $\wedge$ `empregado(`O_1`, or(Pedro, Maria))` $\wedge$ `companhia(`O_1`, or(UFPE, UFPB))` $\wedge$ `salario(`O_1`, 1400)` $\wedge$ *Time*`(`O_1`, [1/1/1994, *])` etc. Consultar este BDTI significa estabelecer uma hipótese e aplicar as regras de dedução

e os axiomas para avaliar essa hipótese. Modificar um BDTI significa mudar a teoria subjacente, inserindo novos fatos e/ou removendo fatos já armazenados.

Um programa Prolog consiste de um conjunto de fatos, um conjunto de regras, e consultas [CNV88]. Em Prolog puro, os fatos são supostamente dados e não mudam durante a execução do programa, enquanto que todas as respostas às consultas são deduzidas dos fatos e das regras. Entretanto, é desejável, ou até mesmo necessário para certas aplicações, atualizar o banco de dados dinamicamente durante a execução do programa, ou seja, adicionar e/ou remover fatos. Em Prolog, a atualização dos fatos é realizada pelas facilidades extralógicas, implementadas através dos comandos *assert* e *retract*, de processamento de cláusulas. O comando *assert*(X) adicionará o termo X como um fato no banco de dados. Por exemplo, `assert(empregado(05, Pedro))` adicionará o fato `empregado(05, Pedro)` aos dados já existentes no programa. O comando *retract*(X) removerá o termo X do banco de dados. Por exemplo, `retract(empregado(05, Pedro))` removerá o fato `empregado(05, Pedro)` de um banco de dados.

Os procedimentos do sistema apresentado no capítulo 6 para modificação de um BDTI utilizam o mesmo mecanismo de atualização de uma base de conhecimento Prolog, ou seja, cada uma das operações de atualização pode ser executada pelos predicados pré-definidos Prolog *assert*(X) e *retract*(X). Por exemplo, a inserção de um objeto temporal indeterminado num BDTI (seção 6.3.2) significa o armazenamento de um conjunto de cláusulas (fatos) que representam o objeto, o que pode ser realizado por uma série de predicados *assert*(X). Já a remoção lógica de um objeto temporal indeterminado (seção 6.3.4) significa a remoção da cláusula temporal do objeto (`retract(Time(x, I1))`), seguida da inserção de uma nova cláusula temporal indicando a remoção lógica (`assert(Time(x, I2))`. Note que uma remoção lógica num BDTI é realizada por duas modificações numa base de conhecimento Prolog. No capítulo 6 serão definidas as operações de modificação de um BDTI e simuladas suas funcionalidades via procedimentos Prolog contendo predicados *assert* e/ou *retract*.

5.6 Conclusão

Neste capítulo foi abordado um formalismo para representação e manipulação de objetos temporais indeterminados em BDTIs. Inicialmente foram analisados os trabalhos

relacionados a este formalismo, destacando a análise de uma *lógica para bases de conhecimento incompleto*, proposta por Levesque [Le94], e também a sintaxe e a semântica da linguagem de consulta temporal TSQL2 [DS98, Sn95], além de uma *teoria relacional generalizada* com informação disjuntiva e valores nulos [Re94]. Duas linguagens de consulta a bancos de dados relacionais temporais também foram abordadas, com o intuito de exibir as primeiras propostas de utilização de comandos para modificação nestes tipos de bancos de dados. Tais comandos envolvem apenas informação temporal do tipo determinada.

A criação de novas formas para representar objetos temporais indeterminados e a elaboração de mecanismos para consulta, inserção, remoção lógica, refinamento e atualização destes objetos, contribuiu para o desenvolvimento de um formalismo para esta teoria em BDTI. Este formalismo assemelha-se com a parte sintática de uma lógica modal temporal pelo fato de incorporar as modalidades *necessário*, *possível* e *conhecido*, além do operador temporal *during* e seus derivados (*before*, *after* e *).

A forma de representação de fatos na base de conhecimento incompleto proposta por Levesque [Le94], utilizando o operador K para representar fatos conhecidos pela BC, foi adotada na representação de objetos temporais indeterminados no BDTI porque se prestou perfeitamente para a representação destes sob a suposição do mundo aberto (seção 3.1). Entretanto, o formalismo de Levesque é restrito à recuperação de conhecimento incompleto, sob uma abordagem lógica, não envolvendo nenhum tipo de fato temporal, tampouco a modificação de uma BC.

A LITO é uma extensão da lógica de Levesque uma vez que envolve a indeterminação temporal e as operações para modificação de um BDTI.

Com relação a TSQL2, embora a abordagem da LITO seja baseada em lógica e não seja uma abordagem probabilística, é assumida implicitamente nesta tese a *credibilidade* do tipo indeterminada (γ = INDETERMINATE) pelo fato de ser este o único tipo que preserva a indeterminação original. Também é admitida a *plausibilidade* do tipo igual a 1 (δ = 1), ou seja, caso haja um casamento parcial (seção 6.2.1) entre os domínios expressos pelos atributos temporais de uma consulta e do BDTI, então uma resposta do tipo *possível* é gerada para a consulta (evidentemente a resposta depende dos valores verdade resultantes na comparação dos atributos não temporais). Finalmente, a função *p.m.f.* (seção 4.3.6) é uniforme na LITO, pois considera todos os instantes de um intervalo temporal indeterminado

igualmente prováveis na construção de um resposta.

A representação dos instantes e períodos indeterminados de TSQL2 foi estendida, através do operador temporal *during*, para denotar um instante indeterminado, e suas derivações (*before* t, *after* t e *). Esta extensão objetiva aumentar o poder de expressão de uma linguagem de consultas a um BDTI, caso seja adicionada numa linguagem de consultas temporal, devido a uma maior variedade de formas para expressar a incompletitude.

A LITO formaliza as noções intuídas pelas operações de modificação de um BDTI, sejam estas extensões de operações convencionais (inserção, remoção e atualização) ou mesmo uma nova operação criada para refinar a indeterminação de um atributo de um objeto (operação de refinamento de objetos), além das operações de consulta lógica e consulta de recuperação. A LITO independe de uma linguagem de consulta temporal porque descreve a indeterminação, temporal ou não temporal, própria de um modelo de dados indeterminado. Por isso, trata-se de uma base para futuras explorações, tanto para novas extensões das noções formalizadas, mesmo em outros domínios do conhecimento humano, tais como Inteligência Artificial, Representação do Conhecimento, Raciocínio Temporal, etc., quanto para elaboração de uma semântica formal e prova de corretude e completude no domínio da Lógica Temporal.

Um sistema para representação e manipulação de objetos armazenados num BDTI, descrito no próximo capítulo, tem se mostrado consistente com a LITO. Este sistema pode servir como um *front-end* para sistemas gerenciadores de bancos de dados temporais que suportem linguagens de consulta temporais, tais como a SQL/Temporal (seção 4.5) e a extensão proposta para SQL-92 (seção 4.4).

Capítulo 6

Um sistema para processamento de objetos temporais indeterminados

"O tempo é a minha matéria,
o tempo presente, os homens presentes,
a vida presente."

Carlos Drummond de Andrade, MÃOS DADAS

Este capítulo apresenta um sistema para processamento de objetos temporais indeterminados denominado MITO (Management of Indeterminate Temporal Objects), desenvolvido para mostrar a aplicabilidade do formalismo expresso no capítulo 5, destacando aspectos da representação de objetos, exemplos envolvendo todas as operações propostas para BDTIs e os algoritmos mais importantes.

São distinguidas operações para manutenção de BDTIs e operações para consulta e modificação de BDTIs. As operações para manutenção são apropriadas para um administrador do banco de dados temporais (criação e remoção de classe, correção e remoção física de objetos). As operações para consulta (consulta de recuperação e consulta lógica) e para modificação de objetos (remoção lógica, inserção, refinamento e atualização de objetos) são adequadas para usuários finais.

Na próxima seção será mostrado como os objetos de um BDTI podem ser representados internamente. Na seção 6.2 as consultas lógicas e de recuperação serão conceituadas, enquanto que a seção 6.3 abordará as operações de modificação de um BDTI.

6.1 Representação de objetos

Objetos são compostos por uma identificação e um conjunto de atributos. Uma identificação de objeto (O_{id}) é um par (id, t), onde id é um código único no sistema e t é o instante quando o objeto foi inserido no banco de dados. Cada remoção lógica, refinamento

ou atualização de um objeto (*id*, *t*) gera uma cópia do objeto original, incorporando as modificações. Esta nova cópia é identificada por (*id*, *t'*), onde *t'* é o instante de execução da operação. Adicionalmente a este tempo de transação (seção 4.3.4), cada objeto tem um intervalo tempo de validade. Logo, cada objeto possui um atributo especial chamado *tempo* contendo o tempo de validade do objeto; e a história do objeto pode ser recuperada através de seu *id*.

O banco de dados exemplo lista quatro instâncias de uma classe *emprego*, relacionando empregados a companhias com seus respectivos salários (TAB. 18). Neste exemplo não leva em consideração a decomposição da O_{id}. Também o dia é assumido, para propósitos de exposição, como granularidade básica.

TABELA 18 - Banco de Dados Exemplo

O_{id}	Empregado	Companhia	Salário	Tempo
O_1	Pedro *or* Maria	UFPE *or* UFPB	1400	[1/1/1994, *]
O_2	*not* Maria	UFCE	*in*(2000, 3000)	[*before* 1/1/1983, *now*]
O_3	Maria	UFPB	*lt* 1500	[1/1/1994, *now*]
O_4	*	UFPB	*none*	[1/1/1970, 31/12/1990]

Dentre as várias classificações de informação indeterminada apresentadas no capítulo 3, foram selecionados alguns tipos apropriados à representação de objetos temporais indeterminados. Ou seja, objetos cujos atributos podem conter tanto a informação não-temporal indeterminada, quanto a informação temporal indeterminada. Estes tipos, descritos a seguir, são: a informação omissa, a informação negativa e a informação imprecisa.

- *Informação omissa*: aplica-se quando a informação é existente, mas é desconhecida ou quando a informação é inexistente. Quando o valor de um atributo for desconhecido (valor nulo de Codd [Co79]), ele é representado pelo símbolo *. Quando o valor de um atributo for inexistente, ele é denotado pela palavra *none*. Por exemplo, *emprego*(O_4) declara que houve um empregado que trabalhou na UFPB sem receber salário (*salario*(O_4, *none*)), entre 1/1/1970 e 31/12/1990, mas não se sabe quem era (*empregado*(O_4, *)). O tempo de validade do *emprego*(O_1) também denota que seu instante final é desconhecido (*tempo*(O_1, [1/1/1994, *])).

- *Informação negativa*: é o conhecimento explícito de fatos falsos. Por exemplo, *em-*

prego(O_2) declara que há um empregado na UFCE com salário entre $2000 e $3000 trabalhando desde antes de 1983 até hoje, porém tal funcionário certamente não é Maria (*empregado*(O_2, *not* Maria)).

- *Informação imprecisa*: todos os outros tipos de informação indeterminada que não seja omissa ou negativa, são classificados como informação imprecisa. Por exemplo, valores disjuntivos (*empregado*(O_1, *or*(UFPE, UFPB))), intervalos não-temporais (*salario*(O_2, *in*(2000, 3000)) e (*salario*(O_3, *lt* 1500))) e intervalo temporal (*tempo*(O_2, [*before* 1/1/1993, *now*])).

Foram vistas representações simplificadas para instantes indeterminados (definições 15 até 16 da seção 5.3.4) onde o instante é omisso (denotado pelo símbolo *), ou acontece antes de um dado instante t (*before* t), ou acontece depois de um dado instante t (*after* t). A TAB. 19 mostra cada um destes instantes indeterminados transformado pelo sistema num formato que usa o operador *during*. Note que um usuário pode escolher entre uma representação com ou sem *during* para um instante indeterminado, embora a representação sem *during* pareça mais expressiva.

TABELA 19 - Representações para instantes indeterminados

Instante	Representação
*	*during*[∞, ∞]
before t	*during*[∞, *pred* t]
after t	*during*[*succ* t, ∞]

É assumida uma interpretação posicional para a variável temporal ∞. Os valores *default* do sistema para ∞ são 1/1/1, quando ∞ está posicionado no instante inicial ([∞, _]), e 31/12/9999 quando ∞ está posicionado no instante final de um intervalo temporal ([_,∞]). Todavia, a variável temporal ∞ pode assumir outros valores (TAB. 20). Por exemplo, a valoração estabelecida no último caso desta tabela, onde o fato é temporalmente válido num intervalo que se iniciou durante um intervalo *I* e se encerrou antes de um instante *t* ([*during* I, *before* t]), é interpretada levando-se em conta o todo da informação temporal, ou seja, sabe-se que ela vale a partir de um instante que acontece durante *I* e que continua a valer até um outro instante que acontece entre o instante final de *I* e o instante predecessor de *t* ([*during* I, *during* [*end* I, *pred* t]]). Logo, a variável temporal ∞ que representa o tem-

po final *before* t (*during*[∞, *pred* t]) na TAB. 19) não assume o valor 1/1/1, mas é reinterpretada para assumir um valor pertencente a um intervalo delimitado inicialmente pelo instante final de *I* (*end* I) e finalmente pelo predecessor de *t*.

Para casos onde o limite de um intervalo temporal for uma constante temporal numa granularidade diferente de dia, também é assumida a valoração posicional. Por exemplo, a constante temporal 3/2000 no intervalo [3/2000, _] é convertida para [1/3/2000, _], enquanto que esta mesma constante temporal no intervalo [_, 3/2000] é valorada para [_, 31/12/2000]. Entretanto, se um usuário pretende representar um instante indeterminado, ele deve especificá-lo como *during* [1/3/2000, 30/3/2000]. O sistema faz a verificação automática do total de dias de cada mês, inclusive nos anos bissextos.

TABELA 20 – Valoração posicional da variável temporal ∞

Intervalo	Valoração
[*, *after* t]	[*during* [1/1/1, *succ* t], *during* [*succ* t, 31/12/9999]]
[*, *before* t]	[*during* [1/1/1, *pred* t], *during* [1/1/1, *pred* t]]
[*, *during* I]	[*during* [1/1/1, *begin* I], *during* I]
[*, t]	[*during* [1/1/1, t], t]
[*after* t, *]	[*during* [*succ* t, 31/12/999], *during* [*succ* t, 31/12/999]]
[*, *]	[*during* [1/1/1, 31/12/999], *during* [1/1/1, 31/12/999]]
[t_1, *after* t_2]	[t_1, *during* [*succ* t_2, 31/12/9999]]
[*during* I, *after* t]	[*during* I, *during* [*succ* t, 31/12/9999]]
[*before* t_1, *after* t_2]	[*during* [1/1/1, *pred* t_1], *during* [*succ* t_2, 31/12/9999]]
[*before* t_1, t_2]	[*during* [1/1/1, *pred* t_1], t_2]
[*before* t_1, *during* I]	[*during* [1/1/1, *pred* t], *during* I]
[*before* t, *]	[*during* [1/1/1, *pred* t], *during* [*pred* t, 31/12/9999]]
[*during* I, *]	[*during* I, *during* [*end* I, 31/12/9999]]
[t, *]	[t, *during* [t, 31/12/9999]]
[*after* t_1, *after* t_2]	[*during* [*succ* t_1, *succ* t_2], *during* [*succ* t_2, 31/12/9999]]
[*after* t_1, *before* t_2]	[*during* [*succ* t_1, *pred* t_2], *during* [*succ* t_1, *pred* t_2]]
[*after* t_1, t_2]	[*during* [*succ* t_1, t_2], t_2]]
[*after* t, *during* I]	[*during* [*succ* t, *begin* I], *during* I]]
[*before* t_1, *before* t_2]	[*during* [1/1/1, *pred* t_1], *during* [*pred* t_1, *pred* t_2]]
[t_1, *before* t_2]	[t_1, *during* [t_1, *pred* t_2]]
[*during* I, *before* t]	[*during* I, *during* [*end* I, *pred* t]]

Também é assumida que qualquer informação indeterminada, seja esta temporal ou não, pode ser representada por um conjunto de valores disjuntivos ou *OR-sets* (seção 3.2). TAB. 21 mostra a representação em conjunto de valores disjuntivos para os valores dos atributos de alguns objetos representados na TAB. 18.

TABELA 21 - Valores indeterminados representados como conjuntos disjuntivos

Valor	Representação
or(pedro, maria)	{pedro, maria}
in(2000, 3000)	{2000, 2001, ...,3000}
lt 1500	{1,..., 1498, 1499}
before 1/1/1983	{1/1/1, 2/1/1, ..., 31/12/1984}
*	D (domínio do atributo)
not(UFPB)	D – {UFPB}
now	assume a data corrente

O banco de dados mostrado na TAB. 18 é representado no sistema como uma conjunção de cláusulas unitárias (FIG. 7), em consonância com o formalismo estabelecido no capítulo 5. A classe é convertida em um predicado unário contendo os O_{id} e cada atributo gera um predicado binário relacionando o valor do atributo ao objeto via seu O_{id}. Neste sentido o tempo é considerado como mais um atributo, denominado *tempo*.

```
emprego(O1) ∧ empregado(O1, or(Pedro, Maria)) ∧ companhia(O1,
or(UFPE, UFPB)) ∧ salario(O1, 1400) ∧ tempo(O1, [1/1/1994, *]) ∧

emprego (O2) ∧ empregado(O2, not Maria)) ∧ companhia(O2, UFCE) ∧
salario(O2, in(2000, 3000)) ∧ tempo(O2, [before 1/1/1983, now]) ∧

emprego (O3) ∧ empregado(O3, Maria) ∧ companhia(O3, UFPB) ∧ sala-
rio(O3, lt 1500) ∧ tempo(O3, [1/1/1994, now]) ∧

emprego(O4) ∧ empregado(O4, *) ∧ companhia(O4, UFPB) ∧ sala-
rio(O4, none) ∧ tempo(O4, [1/1/1970, 31/12/1990])
```

FIGURA 7 – Banco de dados como um conjunto de cláusulas unitárias

Note que o identificador faz a conexão entre vários predicados relacionando os atributos do mesmo objeto e que o nome do predicado unário é chamado o *nome da classe.*

6.2 Consultas de objetos

Para formular consultas, um usuário precisa conhecer a estrutura do banco de dados (TAB. 18). Esta forma é mais amigável do que a mostrada na FIG. 7.

O sistema permite dois tipos de consultas: consultas lógicas e consultas de recuperação. Nas consultas lógicas um usuário pergunta pela validade de algum fato e obtém uma das seguintes respostas: *sim*, *não*, *possível* e *desconhecido*. A resposta *desconhecido* é retornada se não existe nenhum objeto no BDTI que satisfaça o fato consultado nem mesmo parcialmente (seção 5.3.5).

As consultas de recuperação retornam valores do banco de dados que satisfaçam parcial ou completamente o fato requerido.

Uma consulta é expressa na seguinte forma geral:

<nome-objeto> "(" *<atributo>* ":" *<valor>* {["," *<atributo>* ":" *<valor>*]} ")"
[";" *<intervalo>*]"."

onde:

- *nome-objeto* é o nome de um objeto (predicado unário);
- *atributo* é o nome de um atributo do objeto (predicado binário);
- *valor* pode ser uma constante, um valor omisso, impreciso ou negativo, ou o símbolo especial '?' significando que este valor deve ser recuperado;
- *intervalo* é a parte temporal da consulta. Um intervalo é um par de instantes indeterminados ou não. Também aqui pode ocorrer a constante especial '?'.

As regras do formalismo (seção 5.3.5) são a base para processar consultas sobre um sistema com indeterminação temporal.

O sistema foi projetado para tratar de intervalos indeterminados dos tipos [*during* I, *t*], [*t*, *during* I] e [*during* I_1, *during* I_2], além do intervalo determinado [t_1, t_2]. Por isso, os delimitadores indeterminados (*after*, *before* e *) dos atributos temporais são convertidos para um destes tipos (TAB. 19). Por exemplo, o intervalo com tempo inicial indeterminado [*after* 31/12/1990, 1/5/1995] é convertido para [*during* [1/1/1991, 1/5/1995], 1/5/1995] (TAB. 20). Na TAB. 22 é exibida a interpretação para intervalos contendo o operador *du-*

ring feita pela conjunção de predicados Time(x, TI) e baseada na proposição da seção 5.4.

TABELA 22 - Interpretação para intervalos contendo *during*

Intervalo	Interpretação
[*during* I, t]	MTime(x, I) ∧ Time(x, [*end* I, t])
[t, *during* I]	Time(x, [t, *begin* I]) ∧ MTime(x, I)
[*during* I_1, *during* I_2]	MTime(x, I_1) ∧ Time(x, [*end* I_1, *begin* I_2]) ∧ MTime(x, I_2)

Um intervalo do tipo [*during* [1/1/1992, 31/12/1992], *during* [1/1/1994, 31/12/1994]] declara que o objeto vale desde algum dia de 1992 e deixa de valer em algum dia de 1994. Logo, ele é interpretado como possível de ter-se iniciado em 1992; certamente válido no período de 31/12/1992 (*end* 1992) até 1/1/1994 (*begin* 1994); e como possível de ter-se acabado em 1994. Lembre que a possibilidade, expressa pelo axioma 13 (seção 5.3.5), declara que o intervalo vale em pelo menos um instante.

6.2.1 Consultas lógicas

O processamento de consultas lógicas é uma dedução para provar as cláusulas declaradas numa consulta. Para cada objeto armazenado no BDTI com mesmo nome de objeto na consulta, seus atributos são comparados com os correspondentes na consulta. Os valores de atributo podem casar completamente (retornando o valor verdade *sim*); podem casar parcialmente (retornando o valor verdade *possível*); podem se contradizer (retornando o valor verdade *não*) ou podem não casar (retornando o valor verdade *desconhecido*). Os objetos do BDTI cujos predicados não casam com os correspondentes na consulta geram o resultado *desconhecido*. Para uma consulta F, o processo de dedução pode gerar um dos tipos de fórmulas KF, $K\neg$F e MF para cada objeto armazenado no BDTI. Estas são respectivamente interpretadas como *sim*, *não* e *possível*. Entretanto, se nenhum objeto do BDTI satisfizer a consulta, isto significa que vale a fórmula $\neg K$F, ou seja, a resposta à consulta é *desconhecido*.

A FIG. 8 mostra o algoritmo que expressa o processamento das regras básicas não-temporais (seção 5.3.5). Ele compara o domínio de valores, sejam estes numéricos ou não, do atributo da consulta (*DomCons*) com o domínio de possíveis valores do atributo correspondente no banco de dados (*DomBd*). Uma resposta é gerada para cada atributo consultado, de forma que após a comparação de cada um destes atributos, inclusive do atributo

temporal, se consultado, ter-se-á uma lista de respostas que é a entrada para o algoritmo que encontra a resposta final, conforme será visto mais adiante (FIG. 12). Os comentários denotados por chaves serão referenciados nos exemplos mostrados nas TAB. 23 e 24.

```
Se DomCons ∩ DomBd ≠ ∅
Então     Se DomCons ⊂ DomBd
     Então Retorne Possível {resultado 1}
     Senão     Se DomCons ⊇ DomBd
       Então Retorne Sim
       Senão Retorne Possível {resultado 2}
Senão     Se not (DomCons) ou not (DomBd)
     Então Retorne Não
     Senão Retorne Desconhecido
```

FIGURA 8 – Algoritmo de comparação de atributos não-temporais

A TAB. 23 mostra o resultado da comparação de uma consulta que recupera objetos com salário entre $1600 e $2000 (`salário = in(1600, 2000)`) com vários valores numéricos armazenados em um BDTI.

A TAB. 24 mostra os resultados da comparação dos atributos não-numéricos de uma consulta, com `companhia = UFPB or UFPE`, com vários domínios de objetos armazenados em um BDTI.

Os atributos temporais da consulta e do BDTI podem assumir valores verdade *possível* (*M*Time(x, I)) e *sim* (Time(x, I)), quando interpretados (TAB. 22). Tais valores verdade são denotados por *SimCons* e *PossívelBd* no algoritmo da FIG. 9.

TABELA 23 - Resultados da comparação de valores numéricos

Resultado	**Valores de *salário* num banco de dados**
1 – possível	*>1500, ≥1600, in(1300, 2100), ...*
sim	*1700, in(1600,2000), in(1700, 1900), 1600 or 1700,...*
2 - possível	*>1600, >1700, in(1700, 2100), not(2000),...*
não	*not(in(1600, 2000))*
desconhecido	*>2000, >2200, <1600, ...*

TABELA 24 - Resultados da comparação de valores não-numéricos

Resultado	Valores de *companhia* num banco de dados
1 - possível	*UFPB or UFPE or UFRN, *,...*
sim	*UFPB or UFPE, UFPB, UFPE.*
2- possível	*UFRN or UFPB,...*
não	*not(UFPB or UFPE)*
desconhecido	*UFCE, UFRN,...*

O algoritmo exibido na FIG. 9 descreve o processo de comparação dos atributos temporais. Note que o resultado só é determinado quando os valores verdade da consulta e do banco de dados são *Sim*.

```
Se DomCons ∩ DomBd ≠ ∅
Então para cada par de sub-intervalos adjacentes
      Se SimCons ∩ SimBd ≠ ∅
      Então     Retorne Sim
      Senão     Retorne Possível
Senão     Retorne Desconhecido
```

FIGURA 9 – Algoritmo para comparação de atributos temporais

Os intervalos temporais da consulta e do BDTI são comparados e, havendo interseção, delimitados em seus sub-intervalos com os valores verdade originais. Para cada par de sub-intervalos adjacentes, da consulta e do BDTI, é encontrada uma resposta de acordo com o algoritmo da FIG. 9 e que é acrescentada à lista de respostas parciais. Esta lista é a entrada do algoritmo, mostrado na FIG. 11, para encontrar a resposta final para o atributo temporal do objeto. Esta resposta final é incorporada à lista de respostas parciais do objeto.

Por exemplo, se for consultado algo valendo no intervalo Time(x, [1/1/1980, 31/12/1990]) e for encontrado um objeto no BDTI valendo no intervalo Time(O_8, [1/1/1980, *during* 12/1990]). No algoritmo mostrado na FIG. 8 é possível verificar que a primeira condição é satisfeita, ou seja, há interseção entre os intervalos da consulta e do BDTI. Após delimitados seus sub-intervalos, se comparam os valores verdade para cada par de sub-intervalos.

Na FIG. 10 é mostrado que o resultado da comparação dos sub-intervalos 1 e 2 é *sim*, pois não satisfazem a condição (*SimCons* ∩ *PossívelBd* ≠ ∅). Já com os sub-

intervalos 3 e 4 esta condição é satisfeita e seu resultado é *possível*. A lista de respostas do atributo temporal conterá uma resposta com valor verdade *sim* e uma outra resposta com valor verdade *possível* e retornará a resposta final *possível* quando processada pelo algoritmo da FIG. 11.

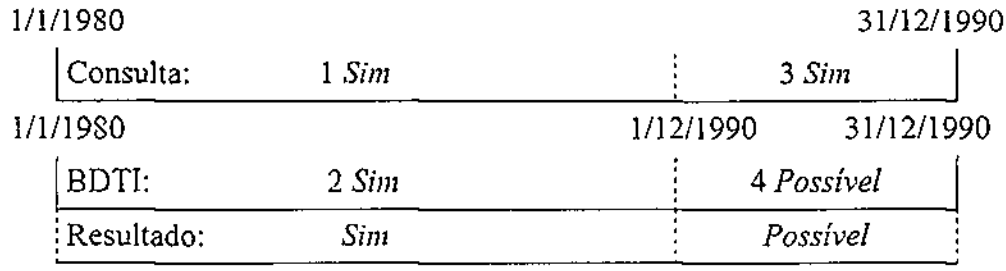


FIGURA 10 – Comparação de sub-intervalos

Uma avaliação intra-objeto [BSF97] determina o valor verdade para cada objeto do BDTI a partir dos da lista de respostas parciais de cada objeto. O algoritmo mostrado na FIG. 11 possui duas funções distintas, por isso ele é processado duas vezes consecutivas. Na primeira invocação, ele recebe a lista de respostas parciais gerada para o atributo temporal, como visto na FIG. 10. Na segunda invocação, ele recebe a lista de respostas parciais correspondendo a todos os casamentos dos atributos do objeto a fim de encontrar a resposta de um objeto. Este processo é repetido para todos os objetos no BDTI com casamento parcial ou total. Finalmente, o algoritmo da FIG. 12 é aplicado para achar a resposta final da consulta lógica.

```
Receba lista-de-respostas-parciais do(s) atri-
buto(s)
  Se ∃ resposta-parcial = Desconhecido
  Então Retorne Desconhecido
  Senão Se ∀ resposta-parcial = Sim
        Então Retorne Sim
        Senão Se ∃ resposta-parcial = Não
              Então Retorne Não
              Senão Retorne Possível
```

FIGURA 11 – Algoritmo para o valor verdade do atributo temporal e para o valor verdade de um objeto

```
Receba lista-de-respostas-parciais do(s) atri-
buto(s)
  Se ∀ resposta-parcial = Desconhecido
  Então Retorne Desconhecido
  Senão Se ∃ resposta-parcial = Sim
        Então Retorne Sim
        Senão Se ∃ resposta-parcial = Não
              Então Retorne Não
              Senão Retorne Possível
```

FIGURA 12 – Algoritmo para a resposta final da consulta lógica

Note que os algoritmos das FIG. 11 e 12 diferem apenas nos dois quantificadores das duas primeiras condições.

Como ilustração, o processamento da consulta lógica "Maria trabalhou na UFPB com salário menor do que $1500, desde 1/1/1994 até agora?" é exibido na FIG. 13. Esta consulta é expressa por:

emprego(empregado: Maria, companhia: UFPB, salario: <1500); [1/1/1994, now]).

Consulta	Empregado: *Maria*	Companhia: *UFPB*	Salario: *<1500*	Tempo: *[1/1/1994, now]*	Respostas Parciais	Resposta Final
Objeto O_1	Pedro *or* Maria	UFPE *or* UFPB	1400	[1/1/1994, *]		
Lista parcial	*Possível*	*Possível*	*Sim*	*Possível*	***Possível***	
Objeto O_2	*not* Maria	UFCE	*in*(2000,3000)	[*before* 1/1/1983, *now*]		
Lista parcial	*Não*	*Desconhecido*	*Desconhecido*	*Sim*	***Desconhecido***	
Objeto O_3	Maria	UFPB	*lt* 1500	[1/1/1994, *now*]		
Lista parcial	*Sim*	*Sim*	*Sim*	*Sim*	***Sim***	***Sim***
Objeto O_4	*	UFPB	*none*	[1/1/1970, 31/12/1990]		
Lista parcial	*Possível*	*Sim*	*Desconhecido*	*Desconhecido*	***Desconhecido***	

FIGURA 13 – Respostas no processamento de uma consulta

6.2.2 Consultas de recuperação

Uma consulta de recuperação preenche atributos da consulta, indicados pelo símbolo '?', com valores recuperados do BDTI. Os valores dos atributos da consulta denotam as condições para recuperação. Estes são comparados com os valores de atributos correspondentes em cada objeto do BDTI. Um valor verdade para cada objeto é obtido pelo

mesmo algoritmo aplicado às consultas lógicas (FIG. 11). Este valor verdade qualifica o(s) dado(s) recuperado(s).

Suponha a consulta: "em quais companhias e em que períodos Maria tem trabalhado recebendo um salário menor do que $5.000", ou seja,

emprego(*empregado*: Maria, *salário*: < 5000, *companhia*: ?; ?).

As respostas a esta consulta são as seguintes (FIG. 14): de acordo com o objeto O_1 é possível que Maria tenha trabalhado na UFPE ou na UFPB entre 1/1/1994 e um tempo final desconhecido. Conforme o objeto O_2, alguém que não é Maria trabalha ganhando um salário entre $2.000 e $3.000, ou seja, menor do que $5.000, na UFCE, no período que se iniciou antes de 1/1/1983 e que se estende até o dia de hoje. Logo, a resposta final é que Maria não trabalhou na UFCE neste período. O objeto O_3 retorna que certamente Maria trabalha na UFPB desde 1/1/1994 até hoje. O desconhecimento do empregado que está representado pelo objeto O_4 resulta que é possível que Maria tenha trabalhado na UFPB de 1/1/1970 a 31/12/1990.

Consulta	empregado: Maria	salário: < 5000	companhia: ?	tempo: ?	Resposta Final
Objeto O_1	*Possível*	*Sim*	UFPE *or* UFPB	[1/1/1994, *]	possível(UFPE *or UFPB*, [1/1/1994, *])
Objeto O_2	*Não*	*Sim*	UFCE	[*before* 1/1/1993, *now*]	Não
Objeto O_3	*Sim*	*Sim*	UFPB	[1/1/1994, *now*]	sim(UFPB, [1/1/1994, *now*])
Objeto O_4	*Possível*	*Sim*	UFPB	[1/1/1970, 31/12/1990]	possível(UFPB, [1/1/1970, 31/12/1990])

FIGURA 14 – Processando uma consulta de recuperação

Em termos formais, uma consulta de recuperação é uma fórmula F(x), onde x é uma variável a ser preenchida pelo casamento dos valores do BDTI. No caso geral, um processamento de consulta recupera três conjuntos de valores X_1, X_2 e X_3 onde para cada elemento x de X_1 tem-se F(x), para elementos de X_2 tem-se MF(x) e para elementos de X_3 tem-se ¬F(x). Isto gera três tipos de respostas: *sim*(X_1), *possível*(X_2) e *não*(X_3).

Suponha a seguinte consulta: "quem trabalhou no período de 1/1/1995 a 30/6/1995?. Esta consulta é expressa por *emprego*(*empregado*:?; [1/1/1995, 30/6/1995]) e

transformada na seguinte hipótese: empregado(x, x_1) ∧ tempo(x, [1/1/1995, 30/6/1995]).

O processamento se inicia comparando o atributo representando a condição da consulta (tempo(x, [1/1/1995, 30/6/1995])) com os valores de cada atributo temporal dos objetos do BDTI. Para o objeto *emprego*(O_1) obtém-se *K*Time(O_1, [1/1/1994, *]) que é representado como *K*Time(O_1, [1/1/1994, *during* [1/1/1994, 31/12/9999]]) (TAB. 19 e 20). A interpretação deste intervalo gera a conjunção dos seguintes predicados (TAB. 22): Time(O_1, [1/1/1994, 1/1/1994] ∧ *M*Time(O_1, [1/1/1994, 31/12/9999]]). De acordo com o algoritmo para comparação dos atributos temporais (FIG. 9) do par de sub-intervalos adjacentes da consulta (tempo(x, [1/1/1995, 30/6/1995])) e do BDTI (*M*Time(O_1, [1/1/1995, 30/6/1995]])), o valor verdade obtido na comparação dos atributos temporais é *possível*. Logo, a resposta desta consulta de recuperação para o objeto O_1 é *Possível*(Pedro or Maria) a qual é interpretada como é possível que Pedro ou Maria tenha trabalhado no período de 1/1/1995 a 30/6/1995. Segundo a lógica definida no capítulo anterior este processamento é expresso pela regra temporal de número 14 (seção 5.3.5) cujas variáveis quantificadas seriam instanciadas, para o objeto O_1, com os seguintes valores: Time(x, [1/1/1995, 30/6/1995]) ∧ Kempregado(O_1, or(Pedro, Maria)) ∧ KMTime(O_1, [1/1/1994, *]) ∧ [1/1/1995, 30/6/1995]) ⊆ [1/1/1994, *] → Mempregado(O_1, or(pedro, Maria)).

O processo continua para os outros objetos do BDTI. Para o objeto *emprego*(O_2) obtém-se *K*Time(O_2, [*before* 1/1/1983, *now*]), que é representado como *K*Time(O_2, [*during* [1/1/1, 31/12/1982], 4/4/2000]) (TAB. 19), assumindo 4/4/2000 como a data corrente. A interpretação para este intervalo é feita pela conjunção dos seguintes predicados (TAB. 22): *M*Time(O_2, [1/1/1, 31/12/1982]) ∧ Time(O_2, [31/12/1982, 4/4/2000]). De acordo com o algoritmo para comparação dos atributos temporais (FIG. 9) do par de sub-intervalos adjacentes da consulta (tempo(x, [1/1/1995, 30/6/1995])) e do BDTI (Time(O_2, [1/1/1995, 30/6/1995]])), o valor verdade obtido na comparação dos atributos temporais é *Sim*. Logo, a resposta desta consulta de recuperação para o objeto O_2 é *sim*(*not* Maria) a qual é interpretada como certamente alguém que não é (se chama) Maria trabalhou no período de 1/1/1995 a 30/6/1995. Todo este processamento é expresso pela regra temporal de número 13 (seção 5.3.5) cujas variáveis quantificadas seriam instanciadas, para o objeto O_2, com os

seguintes valores: Time(x, [1/1/1995, 30/6/1995]) ∧ *K*empregado(O_2, *not* Maria) ∧ *K*Time(O_2, [*before* 1/1/1983, *now*]) ∧ [1/1/1995, 30/6/1995] ⊆ [*before* 1/1/1983, *now*] → empregado(O_2, *not* Maria).

Para o objeto *emprego*(O_3) obtém-se *K*Time(O_3, [1/1/1994, *now*]), que é representado como *K*Time(O_3, [1/1/1994, 4/4/2000]), assumindo 4/4/2000 como a data corrente. De acordo com o algoritmo para comparação dos atributos temporais (FIG. 9) do par de sub-intervalos adjacentes da consulta (tempo(x, [1/1/1995, 30/6/1995])) e do BDTI (Time(O_3, [1/1/1994, 4/4/2000])), o valor verdade obtido na comparação dos atributos temporais é *Sim*. Logo, a resposta desta consulta de recuperação para o objeto O_3 é *sim*(Maria) a qual é interpretada como certamente Maria trabalhou no período de 1/1/1995 a 30/6/1995. Todo este processamento é expresso pela regra temporal de número 13 (seção 5.3.5) cujas variáveis quantificadas seriam instanciadas, para o objeto O_3, com os seguintes valores: Time(x, [1/1/1995, 30/6/1995]) ∧ *K*empregado(O_3, Maria) ∧ *K*Time(O_3, [1/1/1994, 4/4/2000]) ∧ [1/1/1995, 30/6/1995]) ⊆ [1/1/1994, 4/4/2000] → *K*empregado(O_3, Maria).

Finalmente, para o objeto *emprego*(O_4) obtém-se *K*Time(O_4, [1/1/1970, 31/12/1990]). De acordo com o algoritmo para comparação dos atributos temporais (FIG. 9) do par de sub-intervalos adjacentes da consulta (tempo(x, [1/1/1995, 30/6/1995])) e do BDTI (Time(O_4, [1/1/1970, 31/12/1990])), o valor verdade obtido na comparação dos atributos temporais é *desconhecido* (*DomCons* ∩ *DomBd* ≠ ∅). Logo, a resposta desta consulta de recuperação para o objeto O_4 é *desconhecido*.

O resultado final para a consulta "quem trabalhou no segundo semestre se 1995?" deve ser interpretado pelo usuário como: é possível que Pedro ou Maria tenha trabalhado no período de 1/1/1995 a 30/6/1995; um outro empregado que não se chama Maria também trabalhou neste período e certamente Maria trabalhou.

6.3 Modificação de objetos

A modificação de objetos armazenados num BDTI envolve todas as operações que de alguma forma mudam o estado de um BDTI. Estas operações são criação e remoção de

classes; remoção física, correção, inserção, remoção lógica, refinamento e atualização de objetos. Quatro destas operações foram elaboradas para manutenção do BDTI e por isso são de uso exclusivo do administrador do BDTI, pois podem causar violação da integridade do banco de dados se utilizadas indevidamente. Estas operações são as seguintes: operações para criação e remoção de classes (seção 6.3.1); e operações para remoção física e correção de objetos (seção 6.3.3). As demais operações foram projetadas para um usuário final.

6.3.1 Criação e remoção de classes

A operação de criação de classes define a estrutura das classes, ou seja, especifica todos os atributos destas. O procedimento genérico Prolog para isto é o seguinte:

```
nova_classe(Classe, Atributos) :-
                        assert(classe(Classe, Atributos)).
```

Por exemplo, a execução da cláusula `nova_classe(emprego, [Empregado, Companhia, Salario, Tempo])` terá como conseqüência a inclusão no sistema de uma nova classe *emprego* e seu conjunto de atributos.

A operação de remoção de uma classe só poderá ser efetivada no caso em que a classe não tenha instâncias. O procedimento genérico Prolog para remoção de uma classe é o seguinte:

```
remove_classe(Classe) :-
                        retract(classe(Classe, _)).
```

Quando diz-se que estes procedimentos são genéricos, a intenção é mostrar que eles apenas dão uma idéia do mecanismo de execução da operação. Deve-se, a partir de agora, considerar que os exemplos de procedimentos Prolog não refletem sua real forma de implementação.

6.3.2 Inserção de objetos

A operação de inserção de objetos é similar a operação de inserção convencional, com exceção de que aquela permite que os valores de atributos sejam indeterminados. A inserção de objetos cria uma nova O_{id} com seus valores de atributos relacionados e arma-

zena este objeto no BDTI. Se a inserção especifica uma nova versão de um objeto preexistente (id, t_0), a nova O_{id} será (id, t_{now}), onde t_{now} representa a data corrente. Caso contrário, uma O_{id} completamente nova é criada. Se nenhum parâmetro temporal é dado, o instante temporal inicial recebe a data corrente e o instante temporal final recebe a variável *now*.

A TAB. 25 ilustra a inserção do objeto, cujo O_{id} é O_5, no banco de dados exemplo (TAB. 18), representando o fato de que "Paulo foi admitido em 30/03/2000, na UFPB, recebendo um salário fora da faixa de $1000 a $2500".

TABELA 25 – Ilustrando uma inserção

O_{id}	Empregado	Companhia	Salario	Tempo
O_5	Paulo	UFPB	*not in*(1000, 2500)	[30/03/2000, *now*]

Um procedimento genérico Prolog para inserção deste objeto é descrito a seguir. Os termos O_{id}, `Empregado`, `Companhia`, `Salario` e `Tempo` são instanciados, respectivamente, pelos valores de TAB. 25.

```
insere(Oid, [Empregado,Companhia, Salario, Tempo]) :-
                assert(emprego(Oid)),
                assert(empregado(Oid, Empregado)),
                assert(companhia(Oid, Companhia)),
                assert(salario(Oid, Salario)),
                assert(tempo(Oid, Tempo)).
```

6.3.3 Remoção física e correção de objetos

A remoção física exclui um objeto do BDTI. A rigor, esta operação não é empregada num banco de dados temporal, pois pode excluir a história de um objeto. Todavia, pode ser necessária em caso de qualquer erro no povoamento de dados.

Um procedimento genérico Prolog para remoção física de um objeto é o seguinte:

```
remove_fisico(Oid) :-
                    retract(emprego(Oid)),
                    retract(empregado(Oid, _),
                    retract(companhia(Oid, _),
                    retract(salario(Oid, _),
                    retract(tempo(Oid,_).
```

A correção de objetos muda o valor de um atributo que porventura esteja errado. Esta operação é semelhante a operação de atualização de um banco de dados convencional (não-temporal), pois sobrepõe o novo valor de um atributo no lugar do antigo valor. Evidentemente num banco de dados temporal seu uso deve ser cauteloso, devido a possibilidade de perda de valor histórico.

Um procedimento genérico Prolog para correção de um objeto é o seguinte:

```
corrige(Oid, [Empregado2, Companhia2, Salario2, Tempo2]) :-
    remove_fisico(Oid),
    insere(Oid, [Empregado2, Companhia2, Salario2, Tempo2]).
```

Note que este procedimento chama inicialmente o procedimento *remove_fisico* e em seguida o procedimento *insere* (seção 6.3.2), uma vez que uma correção convencional é expressa por uma remoção de um objeto e por uma inserção deste mesmo objeto com o (s) novo(s) valor(es) de atributo(s) corrigido(s).

6.3.4 Remoção lógica de objetos

Uma remoção lógica significa, em bancos de dados temporais, que um certo objeto deixa de ser válido no banco de dados a partir de uma data especificada, inclusive a data corrente. A remoção lógica é aplicável apenas aos objetos correntes, ou seja, àqueles tendo como instante final a variável *now* ou um intervalo indeterminado abrangendo a data corrente. Por exemplo, os instantes temporais finais *after* 1/1/1998, *before* 31/12/2000, * e *during* [01/01/2000, 31/12/2000] aceitam uma remoção lógica vigorando a partir de hoje. Ao contrário, os instantes temporais finais *after* 31/12/2001 e *during* [1/1/2001, 31/12/2001] não são aplicáveis a partir de hoje porque eles denotam objetos válidos até uma data futura, ou seja, atualmente eles são irremovíveis.

O processamento de uma remoção lógica segue as seguintes etapas. Na primeira etapa, o usuário escolhe um objeto a ser removido e fornece uma data, a partir da qual a remoção será realizada. Caso contrário, a data corrente é assumida. A pertinência desta data no intervalo denotado pelo termo instante temporal final de cada objeto é verificada e a remoção lógica é considerada aplicável ou não. Lembre que nos casos onde o instante final é representado por *after* t, *before* t ou *, a pertinência da data fornecida é verificada respectivamente nos intervalos com estes instantes finais (TAB. 20).

A TAB. 26 mostra todos os resultados possíveis para o atributo temporal, se observadas as condições para remoção lógica de um objeto. As condições para verificação da aplicabilidade da operação de refinamento são expressas nas colunas *Não Aplicável* e *Aplicável*, respectivamente. A coluna *Resultado* contém o novo valor do atributo temporal. O valor de *now* é a data corrente e o valor de *Data* é a data fornecida pelo usuário. Os valores de t_1 e t_2 correspondem respectivamente ao tempo inicial e final de um intervalo expresso pelo operador temporal *during*. O valor lógico *True* indica que a condição sempre é satisfeita.

A TAB. 27 mostra o resultado da seguinte remoção lógica: "remover o objeto O_3 a partir de 19/01/2000".

TABELA 26 – Condições de aplicabilidade de uma remoção lógica

Tempo	Não Aplicável	Aplicável	Resultado
[_, *after* t]	*succ* t > *Data*	*succ* t ≤ *Data*	[_, *Data*]
[_, *during* [t_1, t_2]]	*Data* < t_1 ou *Data* > t_2	t_1 ≤ *Data* ≤ t_2	[_, *Data*]
[_, *before* t]	*pred* t < *Data*	*pred* t ≥ *Data*	[_, *Data*]
[_, *]	-	*True*	[_, *Data*]
[_, *now*]	-	*True*	[_, *Data*]

TABELA 27 – Ilustrando uma remoção lógica

O_{id}	Empregado	Companhia	Salario	Tempo
O_3	Maria	UFPB	*lt* 1500	[1/1/1994, 19/01/2000]

Um procedimento genérico Prolog para remoção lógica deste objeto é descrito a seguir. Os termos O_{id} e Tempo instanciam, respectivamente, com os valores O_3 e [1/1/1994, 19/01/2000].

```
remove_logico(Oid, Tempo) :-
                retract(tempo(Oid, _)),
                assert(tempo(Oid, Tempo)).
```

6.3.5 Refinamento de objetos

Uma operação de refinamento ocorre quando se tem uma informação mais precisa sobre um atributo indeterminado. Ela armazena um novo valor resultante da interseção entre os domínios dos valores indeterminados no BDTI e na operação. Esta operação foi concebida para reduzir a informação indeterminada tanto de um atributo não temporal, quanto dos limites de um atributo temporal. Esta operação não terá efeito em duas situações: se não houver interseção entre os domínios dos valores indeterminados no BDTI e na operação, ou o domínio do valor da operação for menos preciso do que o domínio do atributo no BDTI.

A operação de refinamento de objetos corresponde a uma operação de remoção lógica seguida por uma operação de inserção de objeto. O valor da operação de refinamento é trocado pela interseção deste valor com o atributo correspondente no BDTI. Caso o objeto seja passível de refinamento, ele é inserido no BDTI contendo o novo valor refinado e os outros valores de atributos preservados. Este objeto tem um O_{id} com o mesmo *id* do objeto anterior e com um novo *t*.

A TAB. 28 mostra o caso onde o objeto $O_3 = (3, t_o)$ tem seu salário alterado para > 1300 no tempo t_1, a partir de 1/1/2000. Um novo objeto (*emprego*(O_3')) é inserido, sendo $O_3' = (3, t_1)$, o atributo *Salario* é refinado e os outros atributos preservados. O novo valor do salário corresponde a interseção entre o valor antigo (< 1500) e o valor informado (> 1300), ou seja, o novo valor é representado por *in*(1301, 1499).

TABELA 28 - Objeto refinado

O_{id}	Empregado	Companhia	Salario	Tempo
O_3	Maria	UFPB	< 1500	[1/1/1994, 31/12/1999]
O_3'	Maria	UFPB	*in* (1301, 1499)	[1/1/1994, *now*]

Como esta operação representa um refinamento de um atributo já conhecido, embora de forma vaga, o tempo inicial de O_3' é o mesmo de O_3, deixando claro que o valor

in(1301, 1499) de *Salário* continua tendo validade neste intervalo todo.

Um procedimento genérico Prolog para refinamento deste objeto é descrito a seguir. Os termos O_{id} e Tempo possuem os valores correspondentes ao objeto *emprego*(O_3) da TAB. 28, enquanto que os termos Novo_O_{id}, Empregado, Companhia, Salario e $Tempo_2$ possuem, respectivamente, os valores correspondentes ao objeto *emprego*(O_3') da TAB. 28.

```
refina(Oid, Tempo, Novo_Oid,
          [Empregado2, Companhia2, Salario2, Tempo2]) :-
                    retract(tempo(Oid, _)),
                    assert(tempo(Oid, Tempo)),
                    assert(emprego(Novo_Oid)),
                    assert(empregado(Novo_Oid, Empregado2),
                    assert(companhia(Novo_Oid, Companhia2)
                    assert(salario(Novo_Oid, Salario2),
                    assert(tempo(Novo_Oid, Tempo2).
```

É possível também refinar cada limite indeterminado do atributo *Tempo*. Neste caso, o instante temporal escolhido para o refinamento é que determina se este é aplicável, ou seja, se há interseção entre os domínios do limite indeterminado e do intervalo representado pelo instante temporal escolhido. O domínio da interseção é o novo valor refinado. Note que a constante temporal * não deve ser escolhida como instante temporal, pois sempre aumentaria a indeterminação do limite. Note também que a data corrente (*now*) só deve ser usada para refinar o limite final.

A TAB. 30 mostra o caso onde o objeto $O_2 = (2, t_o)$ tem seu tempo refinado em *after* 26/06/1982, a partir de uma data qualquer (*Data*). Um novo objeto (*emprego*(O_2')) é inserido, contendo $O_2' = (2, t_1)$, o atributo *Tempo* refinado e os outros atributos preservados. O novo valor de *Tempo* (*during*[27/06/1982, 31/12/1982]) é resultante da interseção entre o intervalo representado pelo instante temporal *after* 26/06/1982 (*during* [*succ* 26/06/1982, ∝]) e o intervalo temporal representado pelo limite indeterminado *before* 1/1/1983 (*during*[∝, 31/12/1982]).

TABELA 29 - Objeto refinado em Tempo

O_{id}	Empregado	Companhia	Salario	Tempo
O_2	*not* Maria	UFCE	*in*(2000, 3000)	[*before* 1/1/1983, *Data*]
O_2'	*not* Maria	UFCE	*in*(2000, 3000)	[*during* [27/06/1982, 31/12/1982], *now*]

Note que houve uma remoção lógica do objeto O_2 na data de execução da operação de refinamento do atributo Tempo. (*Data*) e uma inserção do novo objeto O_2', preservando o instante indeterminado do instante final (*now*).

A estrutura do procedimento genérico Prolog para refinamento de *Tempo* é similar a estrutura do procedimento *refina* visto nesta seção.

6.3.6 Atualização de objetos

A operação de atualização de objetos armazena um novo valor de atributo, sem levar em conta o valor antigo. Esta operação também corresponde a uma operação de remoção lógica seguida de uma operação de inserção de objeto, como em bancos de dados temporais, contendo o seguinte: o valor atualizado, o atributo *Tempo* vigorando no intervalo [*Data*, *now*] e os outros valores de atributos preservados.

A TAB. 20 mostra o caso onde o objeto $O_3 = (3, t_o)$ teve seu salário atualizado para > 1300, a partir de 1/1/2000 até *now*.

TABELA 30 - Objeto atualizado

O_{id}	Empregado	Companhia	Salario	Tempo
O_3	Maria	UFPB	< 1500	[1/1/1994, 31/12/1999]
O_3'	Maria	UFPB	> 1300	[1/1/2000, *now*]

Um procedimento genérico Prolog para atualização deste objeto é similar à estrutura do procedimento `refina` (seção 6.3.5). Os termos O_{id} e `Tempo` possuem os valores correspondentes aos do objeto *emprego*(O_3) da TAB. 30, enquanto que os termos O_{id}', `Empregado, Companhia, Salario` e `Tempo'` possuem, respectivamente, os valores correspondentes aos do objeto *emprego*(O_3') da TAB. 30.

6.4 Conclusão

Neste capítulo foram apresentadas as formas adotadas para representação de objetos num BDTI. Estas formas servem aos tipos de informação indeterminada escolhidos para tratamento pelo MITO. Também foram discutidos as operações de consultas e de modificação de objetos em um BDTI.

Uma consulta lógica retorna um dos seguintes valores-lógicos como resposta: *sim*, *não*, *possível* e *desconhecido*. Os sistemas gerenciadores de bancos de dados atuais possuem linguagens de consulta que não dispõem de consultas do tipo lógicas. Suas linguagens de consulta recuperam valores do banco de dados. Caso tais valores não estejam armazenados no banco de dados a resposta será vazia, segundo a suposição do mundo fechado (seção 3.1). Valores possíveis de satisfazer a consulta também não são considerados. As consultas lógicas foram elaboradas devido a suposição do mundo aberto (seção 3.1), para representação e manipulação de objetos indeterminados. Sob esta suposição, respostas possíveis e desconhecidas são obtidas por uma consulta lógica. Logo, este tipo de consulta aumenta o poder de expressão de linguagens de consulta que a suportem.

Uma consulta de recuperação no MITO retorna valores do BDTI prefixados por um dos seguintes valores lógico: *sim*, *não* ou *possível*. Por isso, se pode perceber uma consulta de recuperação como uma extensão de uma consulta lógica. O valor desconhecido possui a mesma significação tanto como resposta de uma consulta lógica, quanto na resposta de uma consulta de recuperação, ou seja, o fato consultado não está representado como conhecido pelo BDTI.

As operações de modificação de um BDTI podem ser agrupadas em dois tipos: aquelas voltadas para o administrador do BDTI e aquelas elaboradas para um usuário final. As do primeiro grupo permitem a criação e remoção de classes de objetos; e a correção e a remoção física de objetos. As do segundo grupo permitem a inserção de novos objetos, a remoção lógica de objetos a partir de uma data onde seu tempo de validade se expira, o refino de valores indeterminados e a atualização de valores sejam estes determinados ou não.

Capítulo 7

Conclusão e trabalhos futuros

"Por que acaba?
Porque a cada
segundo outro
segundo
novo
valor
idade".

Ed Porto Bezerra, ITINERANTE

A importância fundamental de se poder descrever todas as características reais de objetos, dentre as quais a indeterminação e a temporalidade, é atualmente desconsiderada pelos modelos de dados e pelos sistemas de informação, especialmente os bancos de dados. Em alguns casos estas duas características se condensam no que se chama indeterminação temporal [DS98], ou seja, o tempo de validade de um objeto não é precisamente definido. Afirmações como "o endereço de José é desconhecido" e "o endereço de José é rua tal ou avenida qual" são exemplos de informação indeterminada, enquanto que "a data de nascimento do bebê é, segundo a ultrasonografia, entre 15 e 20 de outubro" e "o tempo de serviço de José começou em 3/2/1989 e acabou em setembro de 1999" são exemplos de indeterminação temporal.

Os Sistemas Gerenciadores de Bancos de Dados deveriam ser capazes de permitir a representação e a manipulação de objetos temporais indeterminados, isto é, objetos com atributos não-temporais e temporais contendo valores que não são exatos ou determinados. Bancos de dados temporais indeterminados (BDTIs) são aqueles que suportam a representação e a manipulação destes tipos de objetos. Entenda-se manipulação de objetos como toda e qualquer operação para recuperação, inserção, remoção lógica, correção, refinamento e atualização de objetos.

A falta de um formalismo para tratar a representação e a manipulação de objetos em

BDTIs levou à construção de uma lógica, chamada LITO (Logic of Indeterminate Temporal Objects), para o tratamento destes tipos de objetos, sendo esta a principal motivação desta tese. Sua parte prática objetivou a conferência da aplicabilidade da LITO e resultou na implementação em LPA-Prolog [LPA97] de um sistema, chamado MITO (Manipulation of Indeterminate Temporal Objects), para representação, recuperação, inserção, correção, remoção lógica, refinamento e atualização de objetos em BDTIs. Uma afirmativa favorável à implementação de operações para modificação (inserção, remoção e atualização) na área de bancos de dados temporais foi feita por Snodgrass [Sn00] ao sugerir operações para SQL3: "a maioria da pesquisa em bancos de dados temporais tem se concentrado nas consultas; a modificação é sempre postergada. A implementação da modificação em tabelas temporais não tem recebido atenção". Foi visto que as operações para modificação no MITO (inserção, remoção lógica e correção) são extensões das operações de modificação propostas para SQL3. Já as operações de refinamento e atualização são inovações desta tese.

Os formalismos abordados no capítulo 4 se mostraram importantes na fundamentação de alguns conceitos, tais como: as informações indeterminadas do tipo disjuntiva e desconhecida (valor nulo), a indeterminação temporal do tipo empregado em TSQL2 e a representação de fatos numa base de conhecimentos (operador *K*). A concepção de um formalismo abrangendo uma linguagem, um conjunto de axiomas e um conjunto de regras para dedução de novos fatos e o processamento de consulta de objetos que estendesse os formalismos já existentes, foi o começo de um processo de investigação que teve como continuação a implementação de um protótipo (MITO) para a verificação da aplicabilidade deste formalismo (LITO). Sendo a LITO uma abordagem lógico-temporal, de fato ela é a parte sintática de uma lógica temporal, beneficiará diversas áreas de aplicação (seção 2.3).

A sugestão de novas formas de representação para objetos temporais indeterminados e de mecanismos para consultá-los e modificá-los é outra contribuição deste trabalho à área de bancos de dados temporais, envolvendo as áreas de lógica aplicada a bancos de dados e de representação de informação indeterminada. Ou seja, as formas de representação de valores de atributos através de operadores não-temporais (*not*, *in*, *lt* e *or*) e temporais (*during*, *begin* e *end*), além de outras derivadas destes; e a criação de mecanismos que resultam em novos valores para os atributos de objetos removidos logicamente, atualizados e refinados constituem um novo ponto de partida para outras discussões e investigações

nas áreas que tratam a indeterminação de dados.

A linguagem TSQL2 [Sn95] é a precursora das linguagens de consulta a bancos de dados temporais. Ela estende a linguagem SQL-92 para que esta suporte tabelas temporais, embora não trate da modificação de um banco de dados temporais com indeterminação temporal.

Algumas correspondências entre TSQL2 e a abordagem seguida por esta tese são as seguintes:

- a constante *none* foi aqui usada para representar um valor desconhecido e inexistente, enquanto que TSQL2 não trata este tipo de valor;
- TSQL2 considera a indeterminação temporal para o atributo temporal associada uma função de probabilidade chamada *p.m.f.* (seção 4.3.6). Esta função permite a especificação de uma distribuição de probabilidade não-homogênea da ocorrência do instante verdadeiro no intervalo dado. Aqui foram implementadas especificamente suas funções *Shrink_s*(*0*, [t_1, t_2], *Uniform*) e *Shrink_t*(*0*, [t_1, t_2], *Uniform*) no algoritmo que compara os intervalos indeterminados da consulta e do BDTI (FIG. 9). O parâmetro *Uniform* indica que a distribuição é homogênea ou uniforme;
- TSQL2 define um período indeterminado cujos limites coincidem com o do operador temporal *during* aqui abordado. Foi proposta uma representação mais abrangente para a indeterminação temporal: os limites de um intervalo indeterminado podem ser representados por operadores temporais especificando tempos imprecisos e desconhecidos. Isto aumenta o poder de expressão para representar fatos indeterminados no tempo;
- a necessidade de comparar períodos a fim de processar junções [SKS99] sobre intervalos temporais (em TSQL2) e executar consultas e modificações (na abordagem aqui usada) precisa de uma redefinição da relação de ordem *Before* (seção 4.3.7). Em TSQL2 isto é obtido usando-se um parâmetro chamado *plausibilidade* como a probabilidade de um período α estar antes de um período β. Aqui foram elaborados procedimentos sobre operadores modais (*K* e *M*) para aplicações onde a função *p.m.f.* da TSQL2 é ausente.

As propostas posteriores a TSQL2, para extensões a SQL-92, são variantes desta e

também não consideram a indeterminação temporal. Foi mostrado que a extensão temporal a SQL-92 [Sn00] e a SQL/Temporal [Me97] suportam apenas consultas e modificações sobre um banco de dados temporais com tempo de validade determinado.

As modificações envolvendo apenas registros com data final determinada são desenvolvidas na extensão a SQL-92. A abordagem proposta nesta pesquisa, suporta operações para inserção, remoção lógica, atualização e refinamento de objetos temporais indeterminados, portanto estendendo sobremaneira a abrangência daquelas operações e, por conseguinte, aumentando seu poder de expressão.

A linguagem SQL/Temporal [SBJ+96a, SBJ+96b, Me97] é a parte temporal proposta para SQL3, aprovada pelo comitê ANSI SQL3 como a parte 7 da SQL3 e atualmente sob consideração no comitê ISO SQL3. Trata-se do primeiro passo em direção à incorporação de mecanismos para representação e manipulação do tempo numa linguagem-padrão de acesso a bancos de dados. Esta proposta não aborda a indeterminação temporal nem a indeterminação não-temporal, tendo portanto o poder de expressão de sua linguagem limitado à tempos determinados.

Uma série de extensões pode ser realizada a partir do estado atual desta pesquisa. A seguir serão relacionadas algumas destas extensões, sem atenção à prioridade de execução:

- O modelo da informação indeterminada pode ser explorado em várias direções. Por exemplo, um atributo condicional pode ser acrescentado a um objeto, condicionando sua validade à validade de outros objetos. Esta condição é dada por uma consulta lógica. Não se tem avaliado o caso geral (possivelmente cíclico) dos objetos condicionados, apenas foram avaliados casos envolvendo poucos objetos sem qualquer referência cíclica, ou seja, um objeto referenciado poder referenciar outro pertencente a uma cadeia de objetos. Suspeita-se que o envolvimento de uma quantidade maior de objetos pode levar a uma cadeia de condições co-dependentes e, direta ou indiretamente, afetar o desempenho do sistema. O atributo condicional também permitiria a negação de um objeto como um todo, caso o valor verdade do atributo condicional fosse falso, ou a possibilidade de um objeto, caso a consulta lógica do atributo condicional retornasse o valor verdade possível;
- Estender uma interface de consultas visuais, como TVQE [FSC97], para tratar com a informação indeterminada;

- A negação, com possíveis implicações, do atributo temporal como um todo (Time(x, *never* I)) ou em parte (Time(x, [*never* I_1, t]) ou Time(x, [t, *never* I_2]), onde t é um instante determinado ou não);
- A continuação do formalismo com a criação de uma semântica e a prova de completude e corretude, gerando assim uma lógica modal temporal (seção 2.3.2) para objetos temporais indeterminados;
- A análise de desempenho do protótipo, em termos de custo para execução das suas operações em comparação com o custo de operações similares em algum outro protótipo para consultas a bancos de dados temporais [SB96];
- A introdução de informação *default* [Po94] em bancos de dados em geral, e em BDTIs em particular, é importante porque várias aplicações precisam tratar com grande número de parâmetros desconhecidos. A informação *default* provê uma capacidade extra para manusear incerteza porque ela declara valores médios e comuns que são prováveis de ser verdade na ausência de evidência provando o contrário.

Referências bibliográficas

[Al83] Allen J. "Maintaining Knowledge about Temporal Intervals", *Communications of ACM*, volume 26, número 11, páginas 832-843, 1983

[Al84] Allen J. "Towards a General Theory of Action and Time", *Artificial Intelligence*, número 23, páginas 123-154, 1984

[AHT90] Allen J., Hendler J. e Tate A. *Readings in Planning*, Morgan Kaufmann Publishers Inc., 1990

[BDE+98] Bettini C., Dyreson C.E., Evans W.S., Snodgrass R.T. e Wang X.S. "A Glossary of Time Granularity Concepts", em [EJS98], páginas 406-413, 1998

[BCS+96] Böhlen M., Chomicki J., Snodgrass R. e Toman D. "Querying TSQL2 Databases with Temporal Logic", Apers P., Bouzeghoub M. e Gardarin G. (Ed.), *Advances in Database Technology: EDBT'96*, Springer-Verlag, LCNS número 1057, página 325-341, 1996

[BD83] Bolour A. e Dekeyser L. "Abstractions in Temporal Information", *Information Systems*, volume 8, número. 1, páginas 41-49, 1983

[BSF97] Bezerra E, Schiel U. e Ferneda E. "MITO - Manipulation of Incomplete Temporal Objects", *Anais do XII Simpósio Brasileiro de Banco de Dados*, Fortaleza – Ce, páginas 364-374, 1997

[BSN98] Bezerra E, Schiel U. e Nóbrega H. "Uma abordagem formal à representação e manipulação de objetos temporais incompletos", *Anais da VII Semana de Informática da Bahia*, Salvador – Ba, páginas 1-15, 1998

[BSL99] Bezerra E, Schiel U. e Lula Jr. B. "Towards a Logic For Representation and Retrieval of Incomplete Temporal Objects", *Proc. of 1st. Workshop International on Computer Science and Information Technologies, CSIT'99*, Moscow, Russia, 1999

[BP93] Bosc P e Prade H. "An Introdution to Fuzzy Set and Possibility Theory Based Approaches to the Treatment of Uncertainty and Imprecision in Database Management Systems", *Proceeding of Second Workshop Uncertainty Management in Information Systems: From Needs to Solutions*, Catalina, CA, 1993

[Co79] Codd E. "Extending the database relational model to capture more meaning", *ACM Transactions on Database Systems*, volume 4, número 4, páginas 397-434, 1979

[Ch80] Chellas B. *Modal Logic: An Introduction*, Cambridge University Press, 1980

[CNV88] Covington M. A., Nute D. e Vellino A. *Prolog Programming in Depth*, Scott, Foresman and Company, Computer Books, 1988

[Ch94] Chomicki J. "Temporal Query Languages: A Survey", Gabay D, Ohlbach H. (eds.) *Temporal Logic: ICTL'94*. Springer-Verlag, (*Lecture Notes in Computer Science* número. 827), páginas 506-534, 1994

[CWA+75] Collins A., Warnock E., Aiello N. e Miller M., "Reasoning from Incomplete Knowledge", D.Bobrow e A.Collins (editores), *Representation and Understanding*, Academic Press, 1975

[DS98] Dyreson C. e Snodgrass R. "Suporting Valid-time Indeterminacy", *ACM Transactions on Database Systems*, volume 23, número 1, páginas 1-57, 1998

[Da93] Davis M. "First Order Logic", Gabbay D., Hogger J. e Robinson J. (Ed.), *Handbook of Logic in Artificial Intelligence and Logic Programming, volume 1, Logical Foundations*, Oxford University Press, 1993

[EJS98] Etzion O., Jajodia S. e Sripada S. "*Temporal Databases: Research and Practice*", Springer-Verlag, 1998

[FSC97] Fernandes Silva S. L., Schiel U. and Catarci T. "Visual Query Operators for Temporal Databases", Proc. *4th Intl. Workshop on Temporal Representation and Reasoning* (*TIME'97*), Florida, USA, pages 46-53, 1997

[Fi93] Fitting M. "Basic Modal Logic", Gabbay D., Hogger J. e Robinson J. (Ed.), *Handbook of Logic in Artificial Intelligence and Logic Programming, volume 1, Logical Foundations*, Oxford University Press, 1993

[GJ99] Gregersen H. e Jensen C.S. "Temporal Entity-Relationship Models – A Survey", IEEE Transactions on Knowledge and Data Engineering, volume 11, número 3, maio/junho, 1999

[Ga95] Galton A. "Time and Change", Gabbay D., Hogger J. e Robinson J. (Ed.), *Handbook of Logic in Artificial Intelligence and Logic Programming, volume 4, Epistemic and Temporal Reasoning*, Oxford University Press, 1995

[Gi93] Ginsberg M. *Essentials of Artificial Intelligence*, Morgan Kaufmann Publishers, San Francisco, 1993

[Ga91] Gamut L. *Logic Language and Meaning, volume 2, Intensional Logic and Logical Grammar*, The University of Chicago Press, Chicago, 1991

[Ga87] Galton A. *Temporal Logics and their Applications*, Academic Press, páginas 197-236, 1987

[HC96] Hughes G. e Cresswell M. A New *Introduction to Modal Logic*, Routledge, 1996

[JD98] Jensen C.S. e Dyreson C.E. (eds.). *"A Consensus Glossary of Temporal Database Concepts – February 1998 Version"*, em [EJS98], páginas 367-405, 1998

[JS99] Jensen C.S. e Snodgrass R.T. "Temporal Data Management", IEEE Transactions on Knowledge and Data Engineering, volume 11, número 1, janeiro/fevereiro, 1999

[KW85] Keller A. e Wilkins M. "On the Use of an Extended Relational Model to Handle Changing Incomplete Information", *IEEE Transactions on Software Engineering*, número 7, 1985

[KS86] Kowalsky F e Sergot M. "A Logic-Based Calculus of Events", *New Generation Computing*, número 3, 1986

[Kr63] Kripke S. "Semantical Considerations on Modal Logic", *Acta Philosophica Fennica*, número 16, páginas 83-94, 1963

[Li79] Lipski W. "On Semantic Issues Connected with Incomplete Information Databases", *ACM Transactions on Database Systems*, volume 4, páginas 262-296, 1979

[Le94] Levesque H. "The Logic of Incomplete Knowledge Bases", Mylopoulus K, Brodie M. (eds) *Readings of Artificial Inteligence & Database.* Morgan Kaufmann Publishers, Inc., páginas 328-341, 1994

[LL99] Levene M. e Loizou G. "Incomplete Information in the Relational Data Model", capítulo 5 de *A Guided Tour of Relational Databases and Beyond*, Springer-Verlag London, 1999

[LS87] Lipeck U. e Saake G. "Monitoring Dynamic Integrity Constraints Based on Temporal Logic", *Information Systems*, volume 12, 1987

[LPA97] LPA WIN-PROLOG 3.5. Win32 Programming Guide, 1997

[MK96] Jixin M. e Brian K. "A Reified Temporal Logic", *The Computer Journal*, volume 39, número 9, 1996

[Me97] Melton J. *SQL/Temporal*, (ISO/IEC JTC 1/SC 21/WG 3 DBL-LGW-013, 1997

[MP92] Manna Z. e Pnueli A., *The Temporal Logic of Reactive and Concurrent Systems: Specification*, Springer-Verlag, New York, 1992

[Mo93] Motro A. *"Sources of Uncertainty in Information Systems"*, Proceedings of Second Workshop Uncertainty Management and Information Systems: From Needs to Solutions, Catalina, Califórnia, 1993

[MS93] Melton, J. e Simon, A. *Understanding the New SQL: A Complete Guide,* Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1993

[Or88] Oresotu B. O. *Um sistema de representação e recuperação de dados incompletos e informação temporal.* Dissertação de mestrado, COPIN/UFPb, Campina Grande, 1988

[Pa96] Parsons S. "Current Approaches to Handling Imperfect Information in Data and Knowledge Bases", *IEEE Transactions on Knowledge and Data Engineering*, número 3, 1996

[Po94] Poole D. "Default Logic". Em Gabbay D., Hogger D. e Robinson J. (ed.), *Handbook of Logic in Artificial Intelligence and Logic Programming, volume 3, Nonmonotonic Reasoning and Uncertain Reasoning*, Oxford Univer-

sity Press, 1994

[Si98] Silvestre R. S. *Um Tratamento Formal Para o Raciocínio Sobre Ação em Inteligência Artificial*, Dissertação, Departamento de Computação, Universidade Federal do Ceará, Fortaleza,1998

[Re78] Reiter R. "On Closed World Databases". Em Gallaire H. e Minker J. (Ed.), *Logic and Databases*, páginas 55-76, Plenum, New York, 1978

[Re78a] Reiter R. "*On Reasoning by Default*". Proceedings of Second TINLAP Conference, Urbana, IL, páginas 210-218, 1978

[Re94] Reiter R. "Towards a Logical Reconstruction of Relational Database Theory". Em M.L.Brodie, J.Mylopoulos e J.W.Schmidt (Ed.), *On Conceptual Modelling*, páginas 191-233, Springer-Verlag, 1994

[Re89] Reichgelt H. "A Comparison of First-order and Modal Logics of Time", Jackson P, Harmelen F. van (Ed.) *Logic-based Knowledge Representation*, MIT Press, páginas 143-176, 1989

[SBJ+96a] Snodgrass R.T., Böhlen M.H., Jensen C.S. e Steiner A. *Adding Valid Time to SQL/Temporal*, change proposal, ANSI X3H2-96-501r2, ISSO/IEC JTC 1/SC 21/WG 3 DBL-MAD-146r2, Novembro 1996, 77 páginas. Na URL: <fto://ftp.cs.arizona.edu/tsql/tsql2/sql3/mad146.ps> (versão atual: 21/11/1996).

[SBJ+96b] Snodgrass R.T., Böhlen M.H., Jensen C.S. e Steiner A. *Adding Transaction Time to SQL/Temporal*, change proposal, ANSI X3H2-96-502r2, ISSO/IEC JTC 1/SC 21/WG 3 DBL-MAD-147r2, Novembro 1996, 47 páginas. Na URL: <fto://ftp.cs.arizona.edu/tsql/tsql2/sql3/mad147. ps> (versão atual: 21/11/1996)

[SKK+94] Snodgrass, R.T., Kulkarni K., Kucera H. e Mattos N. *Proposal for a New SQL Part – Temporal*, 1994. (ISO/IEC JTC1/SC21/WG3 DBL RIO-75, X3H2-94-481.)

[SB96] Steiner, A. e Böhlen, M.H. The TimeDB Temporal Database Prototype, Versão 1.07, Novembro 1996. Na URL: http://www.iesd.auc.dk.general/DBS/tdb/TimeCenter (versão corrente de 26/03/1997)

[Sc96] Schiel U. "Aspectos Temporais em Sistemas de Informação", Relatório Técnico, UFPb/CCT/DSC, DSC/001/96, 2ª edição, 1996

[Sc87] Schiel U. "Representation and Retrieval of Incomplete and Temporal Information", Relatório Técnico, UFPB/CCT/DSC, número DSC/02/87, 1987

[Sc86] Schiel U. "Representação e Recuperação de Informação Temporal e Incompleta", *Anais do 3º Simpósio Brasileiro de Inteligência Artificial*, Rio de Janeiro, páginas 271-281, Novembro 1986

[Sc85] Schiel U. "The Time Dimension in Information System", *Information Systems: Theoretical and Formal Aspects*. Semadas A., Bubenko Jr. e Olivé A. (Ed.), North-Holland, páginas 67-76, 1985

[SKS99] Silberschatz A., Korth H. F. e Sudarshan, S. *Database System Concepts*, 3rd. Edition, WCB McGraw-Hill, 1999

[Sn95] Snodgrass R. (Ed.), *The TSQL2 Temporal Query Language*, Kluwer Academic Publishers, 1995

[Sn00] Snodgrass R. *Developing Time-Oriented Applications in SQL*, Morgan Kaufmann Publishers, San Francisco, California, 2000

[TCG+93] Tansel A, Clifford J, Gadia S, Jajodia S. e Segev A, Snodgrass R. "*Temporal Databases*", The Benjamin/Cummings Publishing Company, Inc., 1993

[TW99] Tsotras V.J. e Wang X.S. "Temporal Databases", em *Encyclopedia of Electrical and Eletronics Engineering*, John Wiley and Sons, New York, 1999

[va95] van Benthem J. "Temporal Logic", Gabbay D., Hogger J. e Robinson J. (Ed.), *Handbook of Logic in Artificial Intelligence and Logic Programming, volume 4, Epistemic and Temporal Reasoning*, Oxford University Press, 1995

[Vi93] Vila L. "Instants, Periods and the Divided Instant Problem", *Proc. of QUARDET'93*, IMACS, 1993.

[Vi94] Vila L. "A Survey on Temporal Reasoning in Artificial Inteligence", *AI-COM*, volume 7, número 1, páginas 4-28, 1994

[WJW98] Wu Y., Jajodia S. e Wang X.S. "Temporal Database Bibliography Update" em [EJS98], páginas 338-366, 1998

[ZP97] Zimányi E. e Pirotte A. "Imperfect Knownledge in Databases", capítulo 3, páginas 35-87, *Uncertainty Management in Information Systems: From Needs to Solutions*, Kluwer Academic Publisher, 1997

[Za91] Zanardo A. "A Complete Deductive System for Since-Until Branching-Time Logic", *Journal of Philosophical Logic*, 1991

Apêndice

Este apêndice mostra as janelas da primeira versão da interface do sistema MITO. Ela foi implementada na linguagem Prolog [LPA97] e possui janelas de entrada e/ou saída de dados contendo botões, janelas para edição, janelas de rolagem (dos tipos *listbox* e *combobox*), além de janelas para mensagens de advertência e para mensagens de erro. Atualmente a interface comporta 10 janelas, sendo uma delas a janela principal (FIG. A1) e as outras correspondem a cada uma das operações do MITO (FIG. A2 a A11).

A seguir é elaborada uma lista com exemplos de operações, sobre o banco de dados da TAB. 18, com o objetivo de apresentar as diversas janelas da interface do sistema MITO:

1) "Corrigir o salário do objeto 1, da classe *emprego*, para 1000" (FIG. A2).
2) "Refinar a companhia do objeto 1, da classe *emprego*, para *UFPe*" (FIG. A3).
3) "Alguém recebe um salário acima de 1000 na classe *emprego*?" (FIG. A4).
4) "Quais os empregados armazenados no banco de dados?" (FIG. A5).
5) "Inserir na classe *emprego* o empregado Carlos, na companhia UFRN, com salário de 1000 e no tempo desde após 31/12/1999 até agora" (FIG. A6).
6) "Inserir a classe *dependente* com os atributos *empregado* e *nome-dependente*" (FIG. A7).
7) "Remover fisicamente o objeto 1 da classe *dependente*" (FIG. A8).
8) "Remover logicamente o objeto 1, da classe *emprego*, a partir de hoje". (FIG. A9).
9) "Remover a classe *dependente*" (FIG. A10).
10) " Atualizar o salário do objeto 1, da classe *emprego*, para um valor maior que 1500" (FIG. A11).

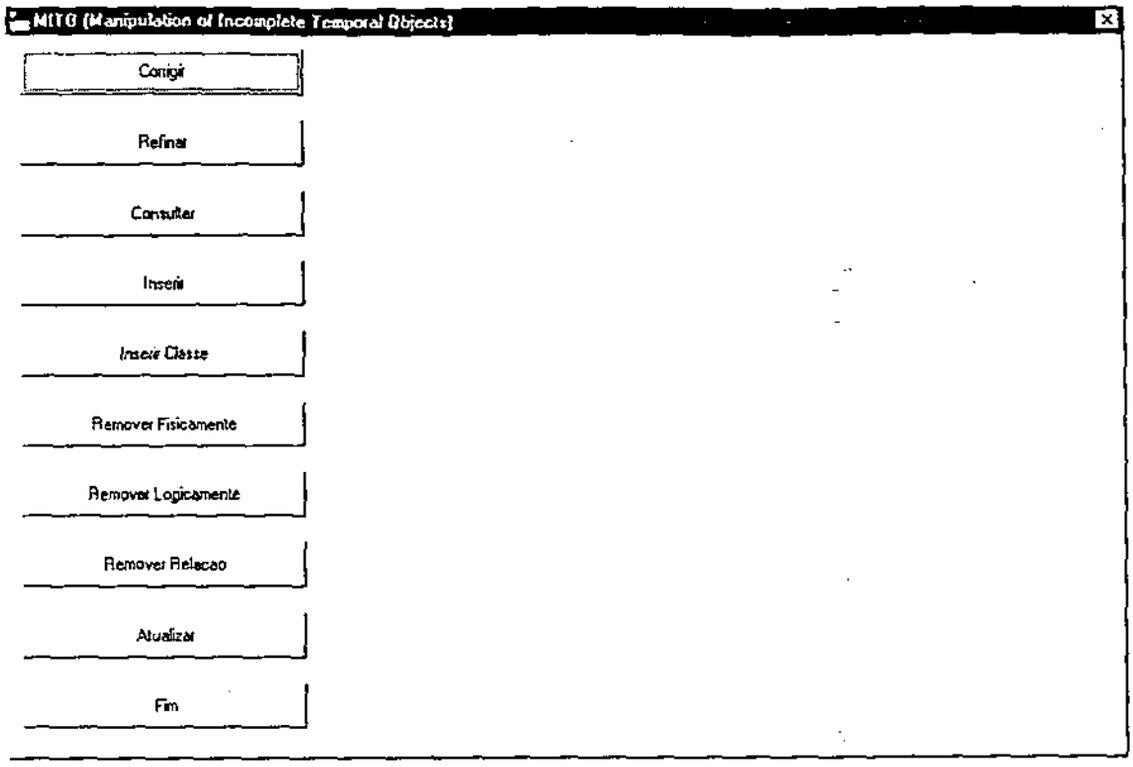


FIGURA A1: Janela principal do sistema MITO.

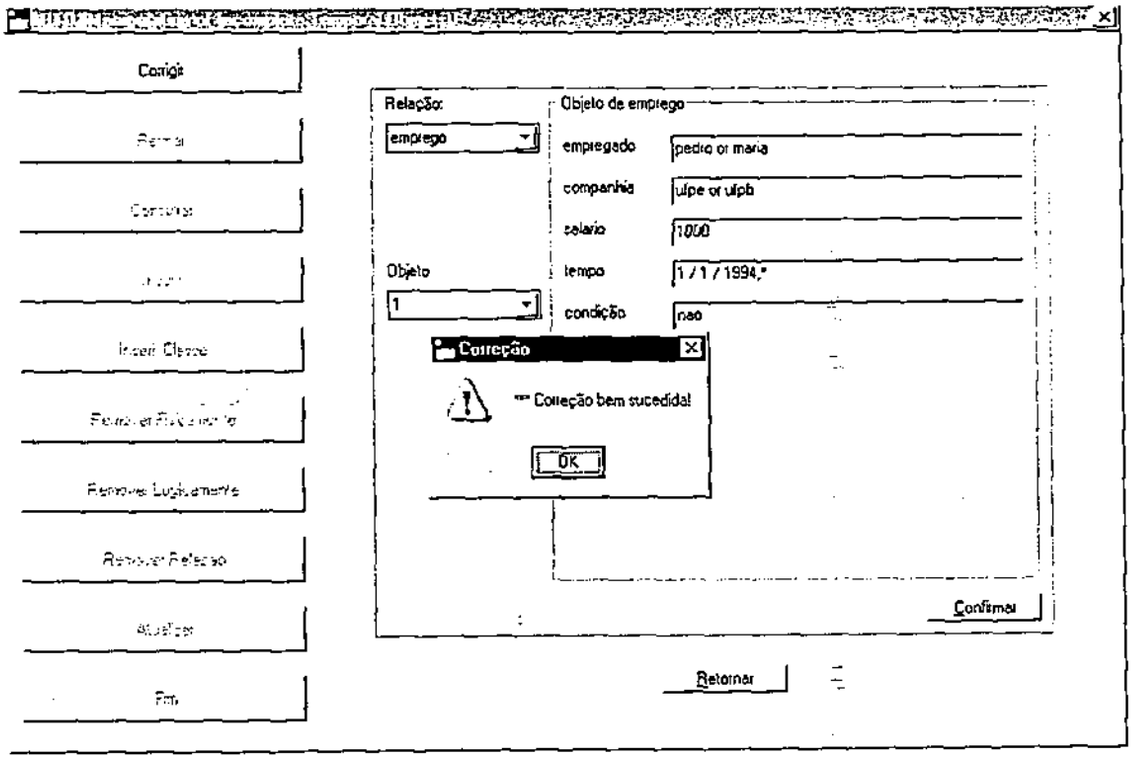


FIGURA A2: Correção de objetos

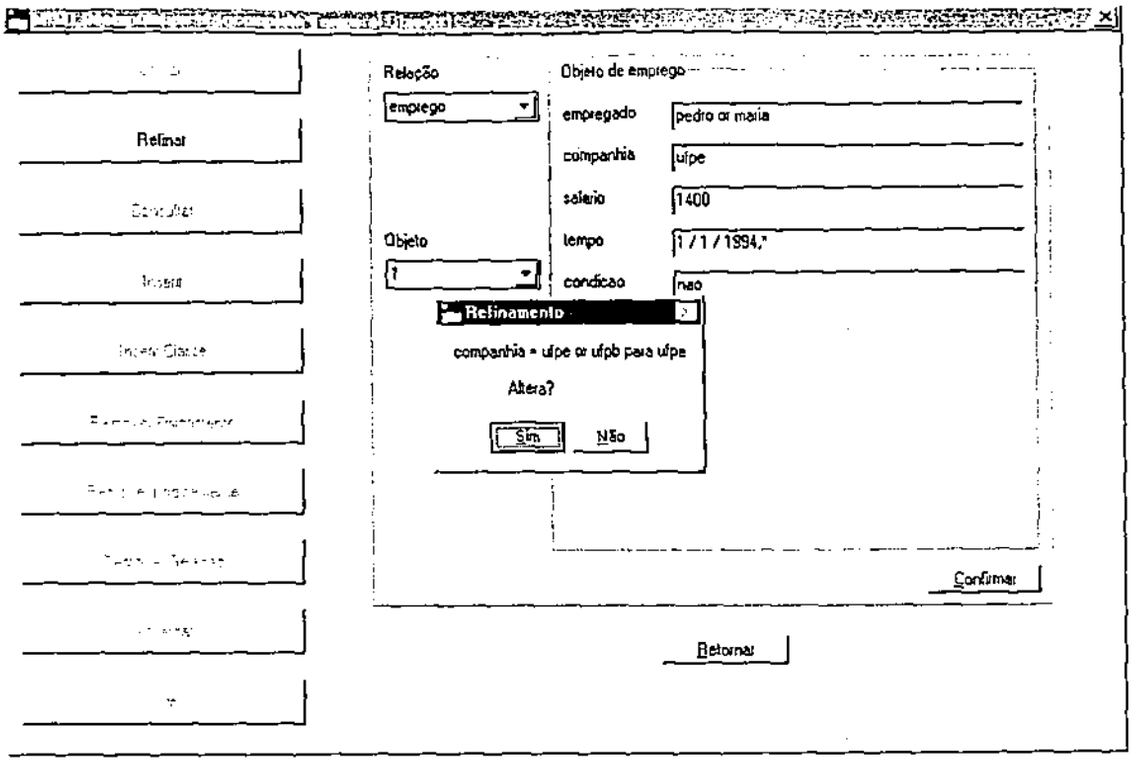


FIGURA A3: Refinamento de objetos

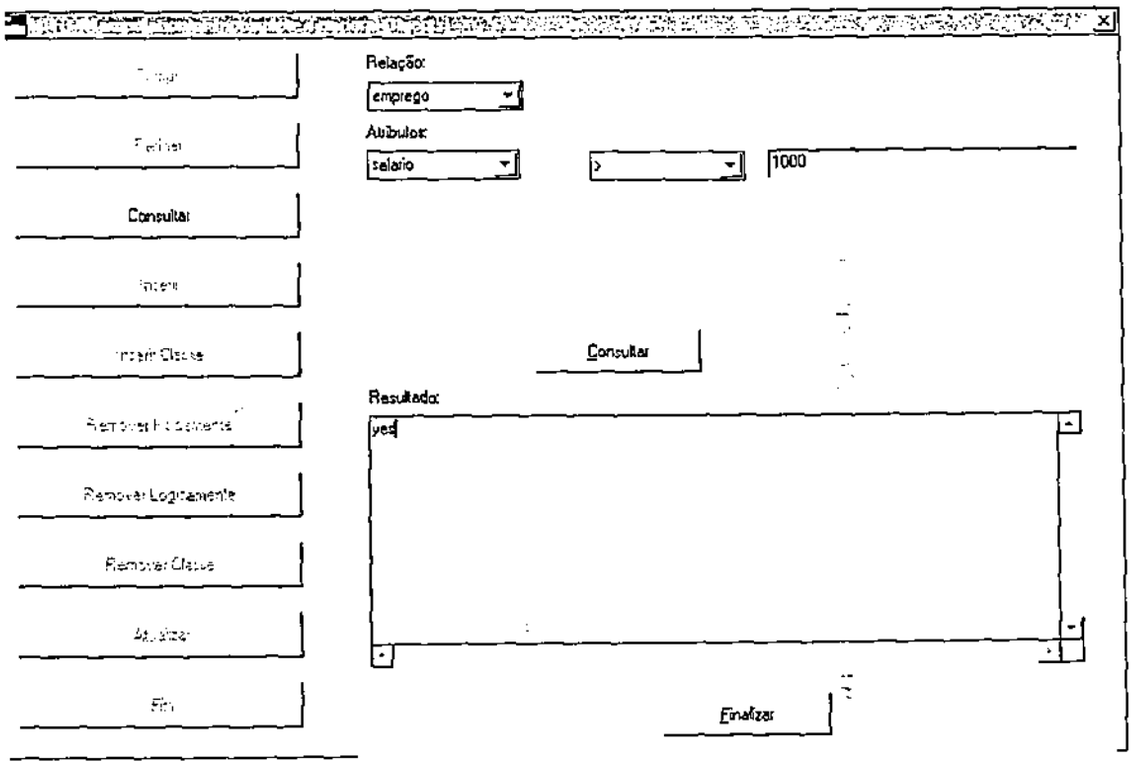


FIGURA A4: Consulta lógica de objetos

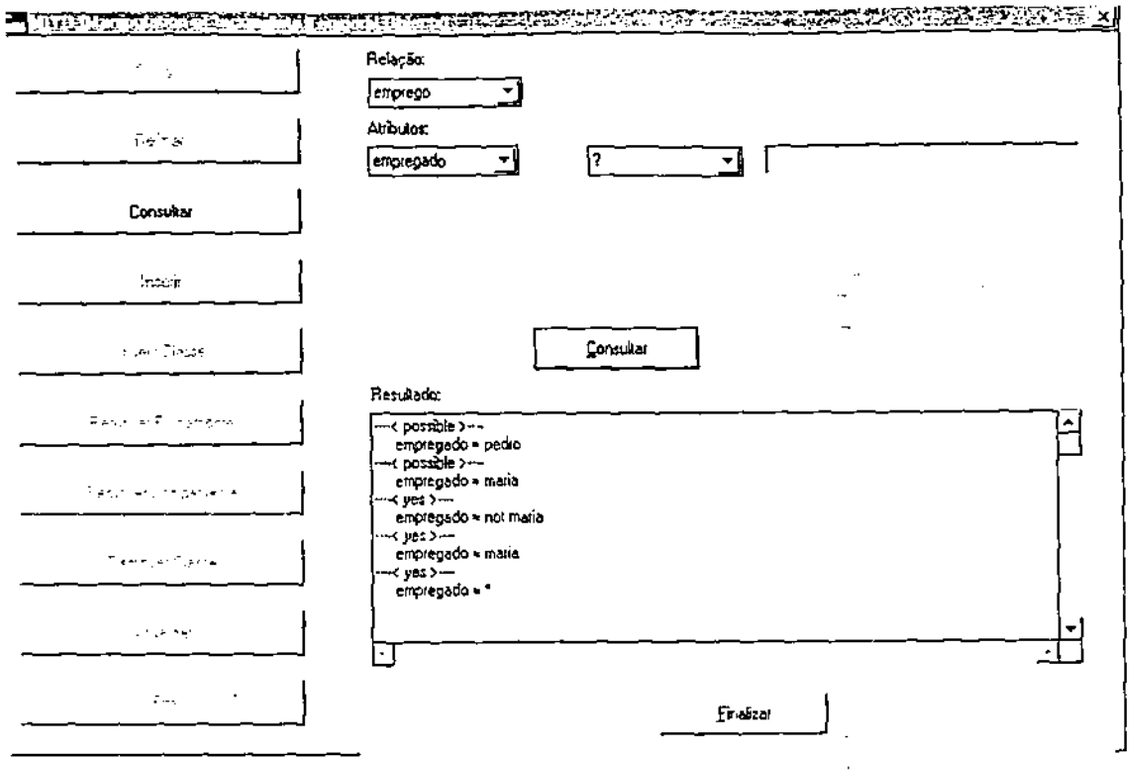


FIGURA A5: Consulta de recuperação de objetos

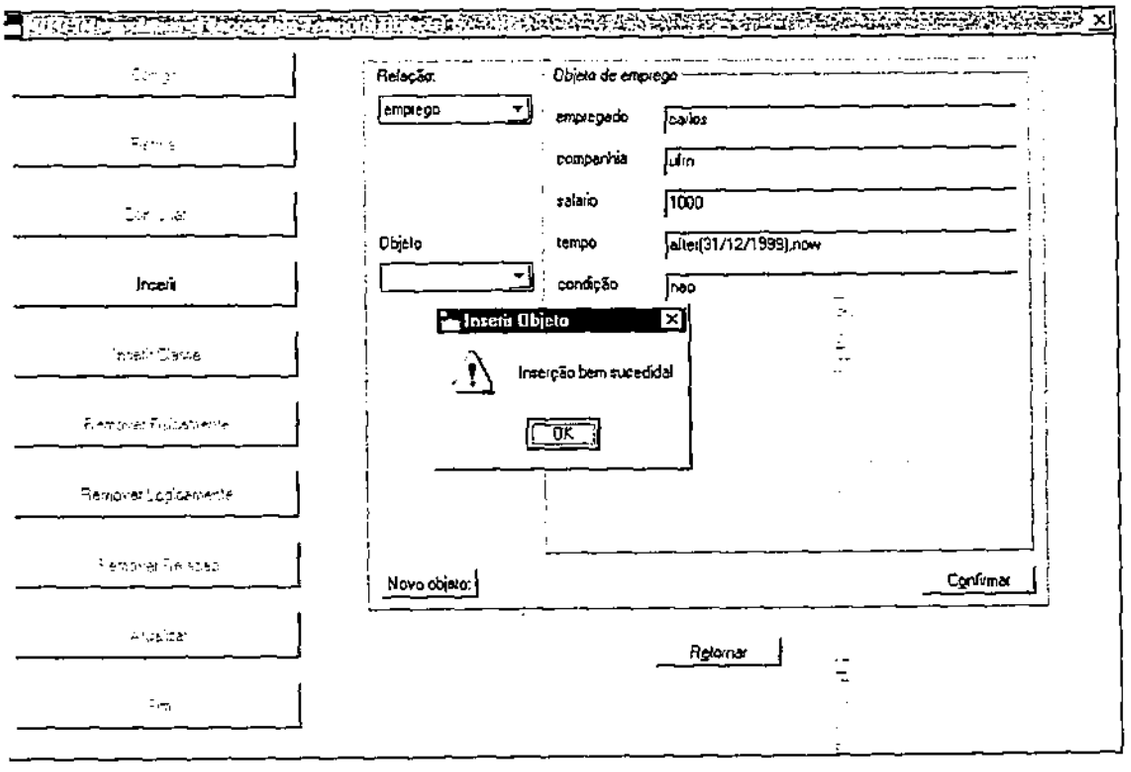


FIGURA A6: Inserção de novos objetos

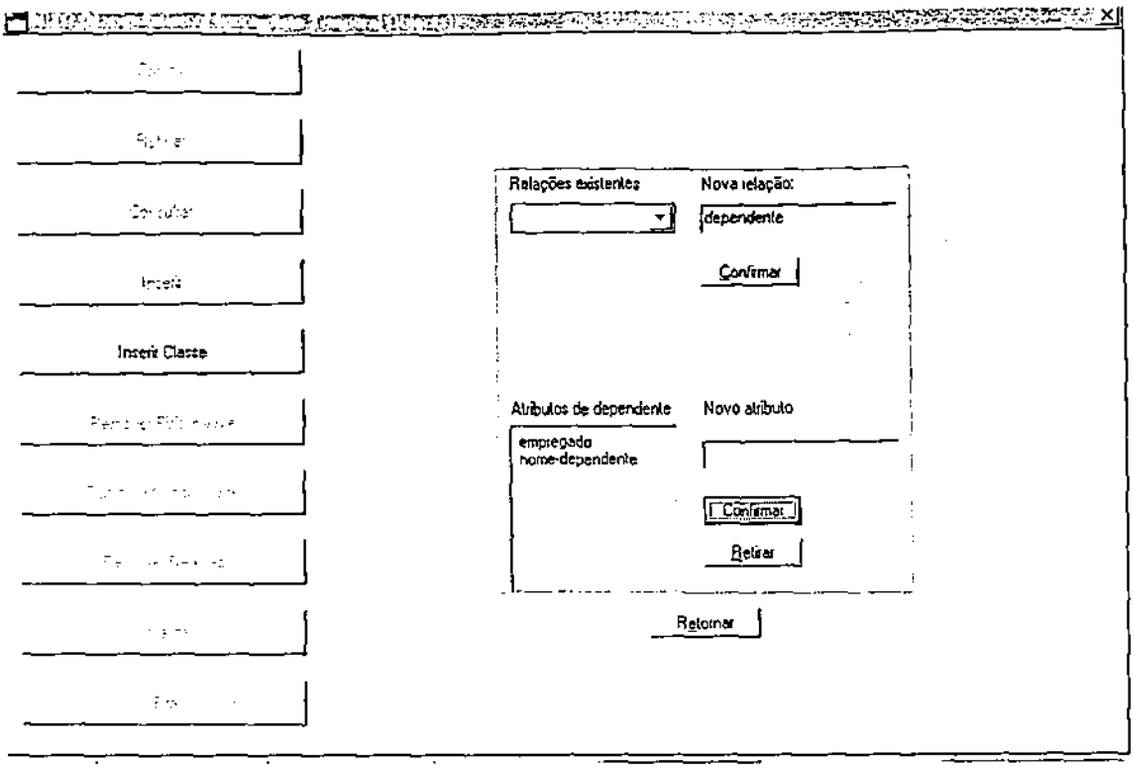


FIGURA A7: Inserção de novas classes

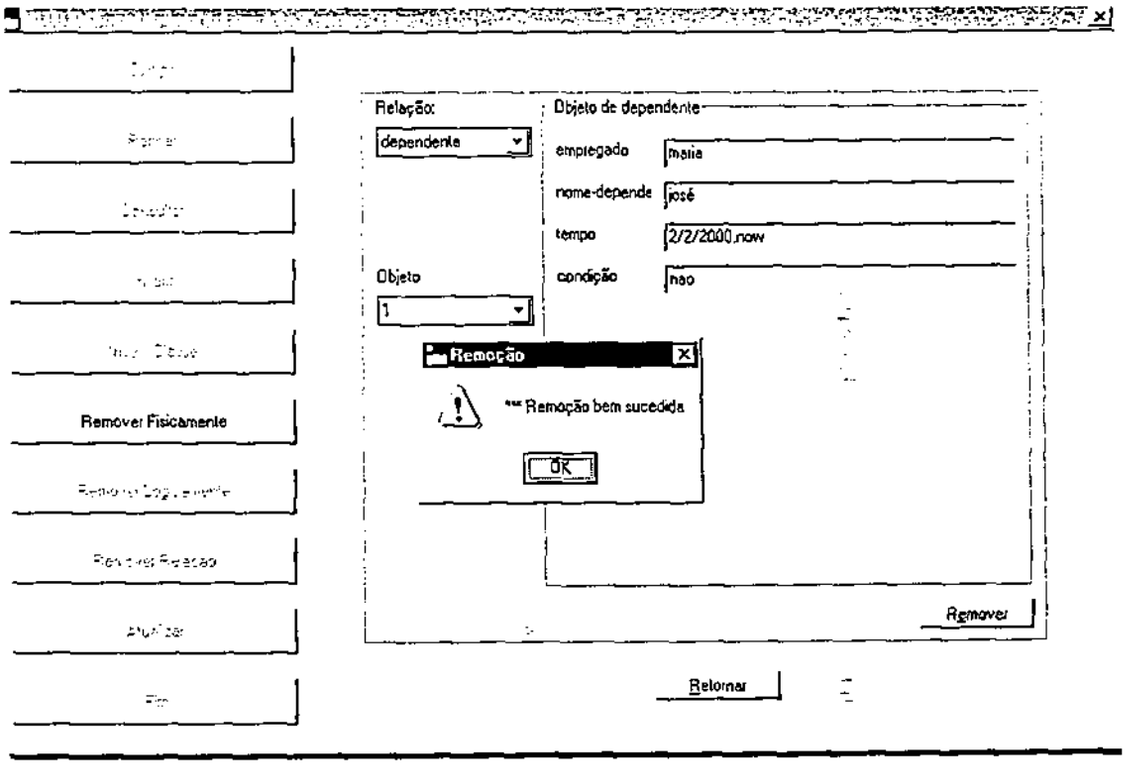


FIGURA A8: Remoção física de objetos

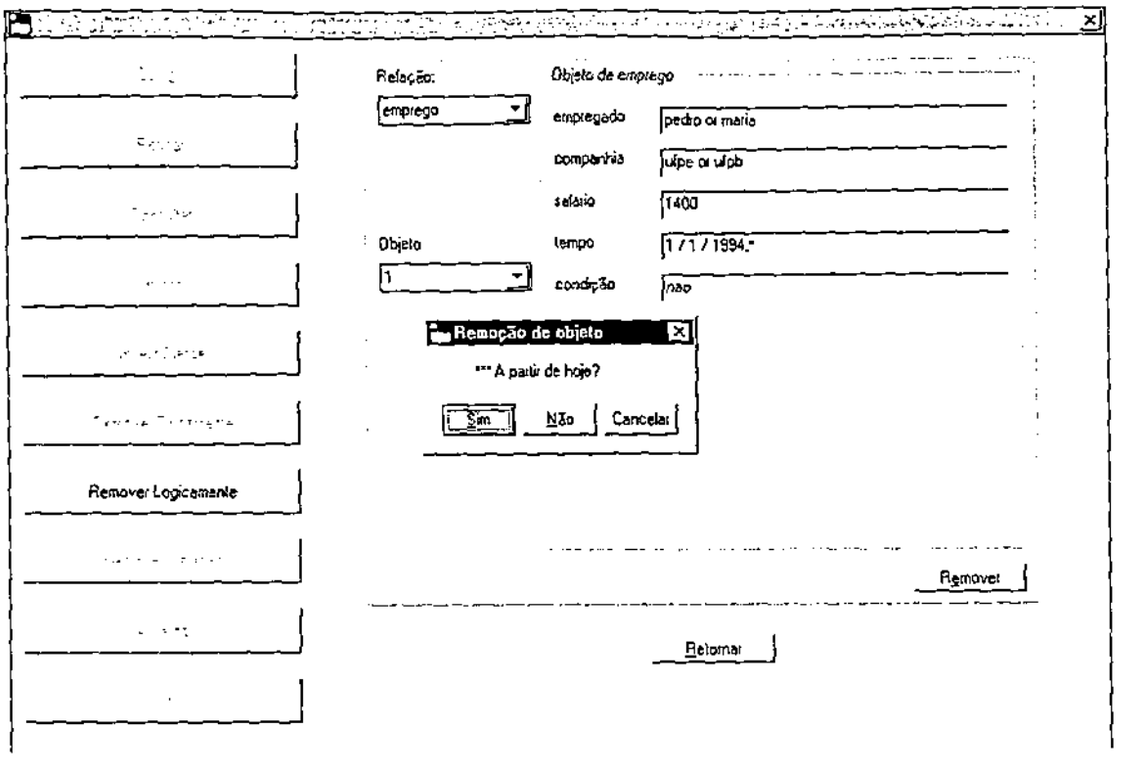


FIGURA A9: Remoção lógica de objetos

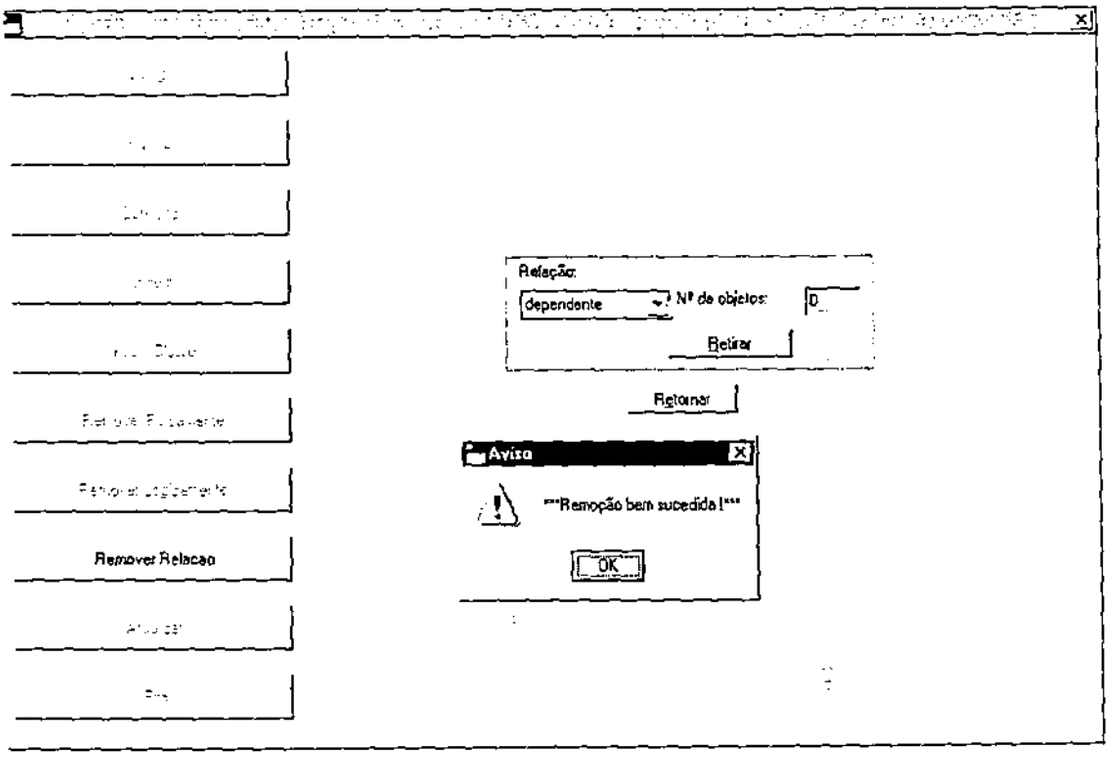


FIGURA A10: Remoção de classes

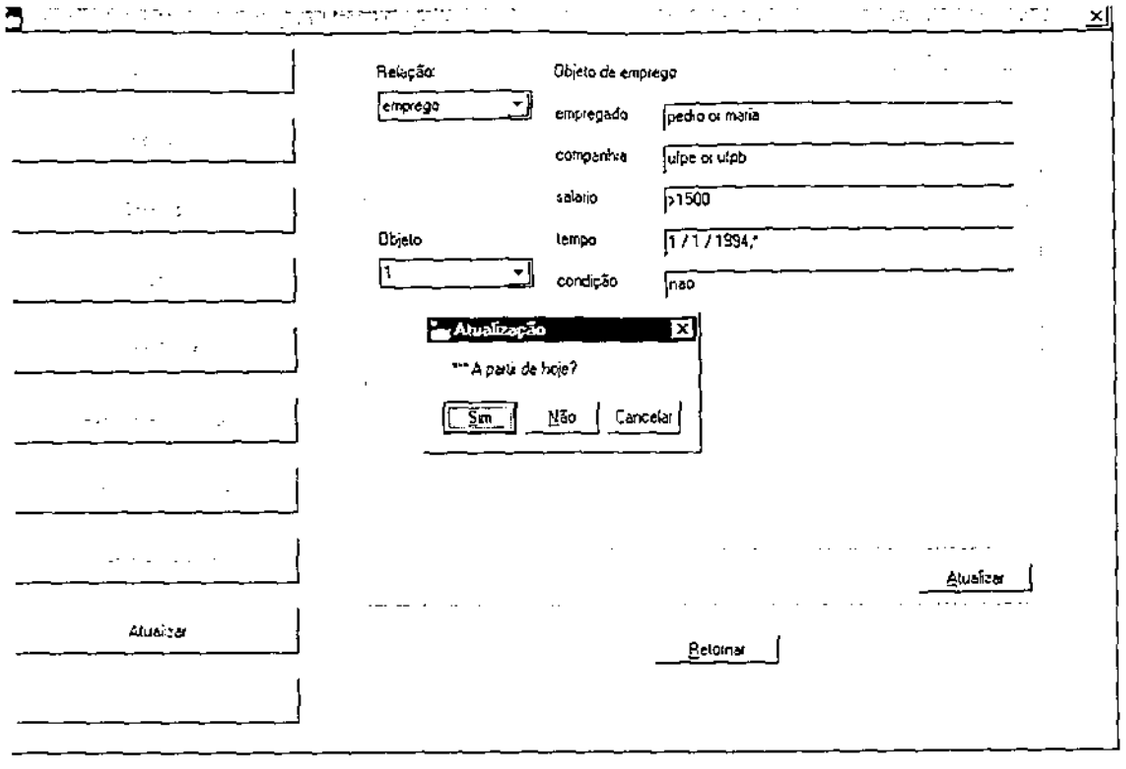


FIGURA A11: Atualização de objetos